



IT210 - SISTEMI IT

OPERATIVNI SISTEMI

Lekcija 01

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 01

OPERATIVNI SISTEMI

- ▼ OPERATIVNI SISTEMI
- ▼ Poglavlje 1: SISTEMI DATOTEKA
- ▼ Poglavlje 2: FUNKCIJE OPERATIVNOG SISTEMA
- ▼ Poglavlje 3: KARAKTERISTIKE FAT32 I FAT16 SISTEMA DATOTEKA
- ▼ Poglavlje 4: UNIX FAMILIJA OPERATIVNIH SISTEMA
- ▼ Poglavlje 5: APPLE SISTEMSKE DATOTEKE
- ▼ Poglavlje 6: Pokazne vežbe: SUSE Linux i VBox
- ▼ Poglavlje 7: Domaći zadatak
- ▼ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Funkcije operativnih sistema i razlike između njih

Kada je reč o računarskim sistemima, njegov hardver bi bio potpuno beskoristan bez softvera. Softver „naređuje“ računaru šta i kako da radi. Softver se sastoji od sistemskih programa i aplikativnih programa. Zadatak sistemskih programa je da učine računarski sistem upotrebljivim, a aplikativni programi služe da zadovolje neku potrebu korisnika. Najvažniji deo sistemskih programa je operativni sistem.

U ovoj lekciji obradićemo:

- Pregled funkcija operativnog sistema
- Sistemske datoteke
- Razlike između operativnih sistema

▼ Poglavlje 1

SISTEMI DATOTEKA

DATOTEKE

Datoteka je imenovani skup uređenih podataka zapisanih u definisanom formatu

Datoteka (engl. **file**) je imenovani skup uređenih podataka zapisanih u definisanom formatu. Datoteke se čuvaju na spoljnim memorijama kao što su diskovi, CD, DVD ili fleš memorija. Sa aspekta operativnog sistema datoteke su objekti. Datoteke pored imena imaju i svoje attribute kojim se definiše:

- Tip datoteke
- Pokazivač na početak datoteke
- Trenutna veličina datoteke
- Pravo pristupa
- Vreme i datum kreiranja
- Vreme i datum zadnje izmene i zadnjeg pristupa

Pored ovih, neki operativni sistemi imaju i druge attribute. Atributi datoteke se čuvaju odvojeno od datoteke, uobičajeno u kontrolnom bloku datoteke koji je deo sistema za upravljanje datotekama.

Datoteke mogu biti izvršne i datoteke sa podacima. **Izvršne datoteke su programi koji su prevedeni na mašinski kod, spremni za izvršavanje i zapisani u binarnom obliku.** Uobičajeni sufiks imena ovih datoteka je **exe** ili **com**. Datoteke sa podacima mogu biti tekstualne (**plain text**) i binarne.

Kod većine operativnih sistema se iza imena datoteke dodaje sufiks kojim je definisan tip datoteke. Sufiks pomaže operativnom sistemu da poveže datoteku sa odgovarajućim programima. Sufiksi nekih tipova datoteka su dati u Tabeli 1.

Tip datoteke datoteke	Sufiks
Tekstualna datoteka (plain text)	txt
Izvršna datoteka	exe, com
Web strana	html, htm
Dokument kreiran u MS Word-u	doc
Digitalna slika zapisana u Joint Photographic Experts Group formatu	jpg

Slika 1.1.1 Tabela-1 Sufiksi nekih tipova datoteka [Izvor: Autor]

SISTEM DATOTEKA

Sistem datoteka organizuje datoteke u baze podataka za skladištenje, organizaciju, manipulaciju i pronalaženje od strane operativnog sistema

Sistem datoteka omogućuje aplikacijama da zapisuju i čitaju fajlove sa spoljnih memorija. **Sistem datoteka organizuje datoteke u baze podataka za skladištenje, organizaciju, manipulaciju i pronalaženje od strane operativnog sistema.** Datoteke se organizuju u hijerarhijske strukture oblika stabla. Sistem datoteka se sastoji od jednog ili više drafvera i dinamički povezanih datoteka koje definišu formate podataka i osobine fajl sistema. Sistem datoteka može da postoji na memorijskim uređajima (npr. hard diskovima, optičkim diskovima i sl.).

Ovi uređaji su fizička osnova, ali ne i deo sistema datoteka operativnog sistema.

Ukupan memorijski prostor koji ima disk može da sadrži jednu ili više **particija**. Particije se kreiraju prilikom formatiranja visokog nivoa diska. Ukoliko jedan fizički disk ima dve ili više particija, operativni sistem će ih videti kao dva ili više odvojena logička diska. Deljenje diska na particije se vrši vrlo često da bi se logički odvojili programi od podataka, ili da bi se pojedinim korisnicima dodelili različiti delovi diska.

▼ 1.1 SISTEMSKE DATOTEKE WINDOWS OS

SISTEM DATOTEKA WINDOWS OPERATIVNOG SISTEMA

NTFS, ReFS, FAT32, FAT16, FAT12, CDFS, UDF

Operativni sistem Windows podržava sledeće sisteme datoteka:

- NTFS – NT (New Technology) File system
- Resilient File System (ReFS)
- FAT32 – File Allocation Table – 32-bitna verzija
- FAT16 i FAT12 - File Allocation Table – 16-bitna verzija
- CDFS
- UDF

Prilikom formatiranja diska korisnik se odlučuje koji će sistem datoteka koristiti. Windows trenutno preporučuje NTFS sistem datoteka, pa će se ovde dati njegove karakteristike. FAT32, FAT16 i FAT12 su stari sistemi datoteka koji se još sreću, pa će se samo ukratko pokazati njihove specifičnosti.

LOGIČKE KOMPONENTE WINDOWS OPERATIVNOG SISTEMA

Volumen, direktorijum i datoteke

Sistem datoteka operativnog sistema Windows ima sledeće logičke komponente:

- Volumene (engl. *volumes*) – kolekcija direktorijuma i datoteka
- Direktorijume (engl. *directories*) – hijerarhijska kolekcija direktorijuma i datoteka
- Datoteke (engl. *files*) – grupa logički povezanih podataka

Pored ovih komponenata, Longhorn NTFS sistem datoteka podržava i simbolične

VOLUMEN

Volumen je prostor na memorijskom uređaju koji se upravlja od strane sistema datoteka kao posebna logička memorijска jedinica

Najviši oblik organizacije u NTFS je volumen. Volumen je prostor na memorijskom uređaju koji se upravlja od strane sistema datoteka kao posebna logička memorijска jedinica. Particija mora da sadrži bar jedan volumen, a jedan volumen može da se prostire na jednoj ili više particija. Volumen koji se nalazi samo na jednoj particiji se naziva prosti volumen, a volumen koji ima podatke na više od jedne particije se naziva multiparticijski volumen.

DIREKTORIJUM

Direktorijum predstavlja hijerarhijsku kolekciju direktorijuma i datoteka

Direktorijum (engl. *folder*) je komponenta sistema datoteka koja sadrži jedan ili više fajlova ili direktorijuma. Jedino ograničenje broja datoteka koje mogu da budu u jednom direktorijumu je fizička veličina particije na kojoj je lociran direktorijum. Direktorijum koji sadrži druge direktorijume je nadređen (*parent*) njima. Direktorijumi koji se nalaze u nekom direktorijumu su pod direktorijumi (*child*) nadređenog direktorijuma. Hijerarhijska struktura direktorijuma se naziva stablo direktorijuma.

NTFS sistem datoteka implementira logičku vezu između direktorijuma i fajlova koje sadrži preko tabele zapisa direktorijuma (engl. *directory entry table*). Kada se nova datoteka smesti u direktorijum, kreira se novi zapis u tabeli sa imenom datoteke.

DATOTEKA

Datoteka je grupa logički povezanih podataka

Za jednu datoteku može postojati više zapisa u tabeli zapisa direktorijuma. Ako se za postojeću datoteku u direktorijumu kreira dodatni zapis on se naziva **čvrsta veza** (engl. *hard link*). Nema ograničenja za broj čvrstih veza koje se mogu kreirati za jednu datoteku.

Podaci koji su potrebni sistemu datoteka da upravlja datotekama se u NTFS sistemu datoteka čuvaju u datoteci. Drugi sistemi datoteka, FAT 32 i FAT16, čuvaju ove informacije u regionu diska koji ne pripada sistemu datoteka. Prednost čuvanja podataka sistema datoteka u fajlu je u tome što je lociranje, pristup i održavanje olakšano, a u slučaju delimičnog oštećenja diska, podaci se brzo mogu prebaciti na drugo sigurnije mesto.

KLASTER

Klaster je osnovna memorijska čelija koja se sastoji od jednog ili više fizičkih sektora na disku

Klaster je osnovna memorijska čelija koja se sastoji od jednog ili više fizičkih sektora na disku. Ovo važi za sve sisteme datoteka koje podržava Windows. Klasteri omogućuju sistemu datoteka da administrira podatke na disku nezavisno od veličine sektora koji je postavljen od strane hardverskog disk kontrolera. Promenom veličine klastera administrator može da optimizuje brzinu čitanja i pisanja datoteka, tako što će veličinu klastera prilagoditi prosečnoj veličini datoteka.

U narednoj tabeli su uporedo date neke osobine NTFS, FAT32 i FAT16 sistema datoteka.

Osobine	NTFS5	NTFS	FAT32	FAT16	ReFS
Enkripcija	Da	Ne	Ne	Ne	Ne
Disk kvote	Da	Ne	Ne	Ne	Ne
Kompresija datoteka	Da	Da*	Ne	Ne	Ne

Slika 1.2.1 Tabela-1 Uporedba osobina NTFS, FAT32 i FAT16 [Izvor: Autor]

. *Ne ako je klaster veći od 4KB

Sledeće karakteristike su ograničene samo veličinom raspoložive memorije u svim sistemima fajlova:

- Maksimalna veličina memorijskog prostora
- Maksimalni broj disk uređaja po serveru
- Maksimalni broj otvorenih lokalnih datoteka
- Maksimalni broj istovremeno zaključanih datoteka.

OGRANIČENJA NTFS, FAT32, FAT16 I REFS

Predstavljanje osnovnih ograničenja sistemskih datoteka NTFS, FAT32, FAT16 i ReFS

Ograničenje	NTFS5	NTFS	FAT32	FAT16	ReFS
Operativni sistem	Windows 2000 Windows XP Windows 2003 Server Windows 2008 Windows Vista Windows 7 Windows 8	Windows NT Windows 2000 Windows XP Windows 2003 Server Windows 2008 Windows Vista Windows 7	DOS v7 i više Windows 98 Windows ME Windows 2000 Windows XP Windows 2003 Server Windows Vista Windows 7	DOS Sve verzije Microsoft Windows	Windows 8 Server
Veličina datoteke	$2^{32} - 1$	$2^{32} - 1$	4194304	65536	$2^{64} - 1$
Max. veličina klastera	64 KB	64 KB	64 KB	64 KB	128K
Max. veličina volumena	2^{64} - 1 klastera	2^{32} - 1 klastera	32GB za sve OS 2TB za neke OS	2 GB za sve OS 4GB za neke OS	2^{78} B sa veličinom klastera 16KB
Max. veličina datoteke	2^{64} B (16 EB-ExaBytes) minus 1KB	2^{44} B (16 TB) minus 64KB	4GB - 2 B	2GB (Ograničeno samo sa veličinom volumena)	2^{64} B (16 EB-ExaBytes) minus 1KB
Max. dužina imena datoteke	255	255	255	255	255

Slika 1.2.2 Tabela-2 Ograničenja NTFS, FAT32, FAT16 i ReFS [Izvor: Autor]

UPOREĐENJE WINDOWS SISTEMSKIH DATOTEKA

This video looks at the four file systems supported by Windows. These are ReFS, NTFS, FAT and exFAT. The video looks at what each file system is capable of and its limitations

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ 1.2 KARAKTERISTIKE NTFS SISTEMA DATOTEKA

OSOBINE NTFS

NTFS je projektovan za fajl servere, mrežne servere i desktop računare

NTFS sistem datoteka je preporučeni sistem za operativne sisteme Windows Server 2003, Windows XP, Windows 2000 i Windows NT. On je projektovan za zadovoljenje potreba fajl servera, mrežnih servera i desktop računara, a i kako bi se otklonili neki nedostaci FAT32 i FAT16 sistema datoteka. NTFS ima sledeće osobine:

- **Oporavljivost podataka**
- **Tolerantnost na kvar**
- **Bezbednost podataka**
- **Unicode imena**
- **Kompresija i podrška „retkim“ datotekama**
- **Šifrovanje**

- **Klasteri**
- **Imena datoteka**

OSOBINE NTFS – OPORAVLJIVOST PODATAKA I TOLERANTNOST NA KVAR

NTFS sistem datoteka će učiniti sve što je moguće da poništi promene nekompletnih operacija

Oporavljivost podataka

NTFS ograničava mogućnost oštećenja podataka organizacijom ulazno- izlaznih operacija na principu transakcije. Transakcije su atomske, što znači da će se izvršiti samo cele ulazno-izlazne operacije, dok se nekompletne operacije ne mogu izvršiti. Ovo znači da će, ako bilo šta prekine transakciju koja je u toku, kao na primer nestanak struje ili poništenje ulazno- izlazne operacije, NTFS sistem datoteka učiniti sve što je moguće da poništi promene nekompletnih operacija i vrati sistem u stanje pre nego je ta ulazno-izlazna operacija započela.

Tolerantnost na kvar

NTFS dozvoljava korišćenje metode redundantnog memorisanja podataka na dva diska istovremeno. Drugi disk sa redundantnim podacima se naziva ogledalo (engl. **mirror**) i na njemu se istovremeno upisuju svi podaci kao i na originalnom disku. Tako se osigurava da se podaci sa oštećenog diska mogu obnoviti sa njegovog ogledala u slučaju kvara ili greške. NTFS sistem datoteka uvek koristi metodu redundantnih podataka da zaštitи interne strukture podataka koje sadrže metapodatke koji su neophodni za očuvanje integriteta sistema datoteka.

OSOBINE NTFS – BEZBEDNOST PODATAKA, UNICODE IMENA I IMENA DATOTEKA

Datoteke i direktorijumi su zaštićeni objekti, a pristup njima se može ograničiti

Bezbednost podataka

NTFS sistem tretira datoteke i direktorijume kao zaštićene objekte saglasno arhitekturi bezbednosti objekata operativnog sistema Windows. Pristup objektima tipa datoteke i direktorijuma može biti ograničen tako što će se samo zadatom korisniku ili grupi dozvoliti pristup.

Unicode imena

NTFS sistem tretira datoteke i direktorijume kao zaštićene objekte saglasno arhitekturi bezbednosti objekata operativnog sistema Windows. Pristup objektima tipa datoteke i direktorijuma može biti ograničen tako što će se samo zadatom korisniku ili grupi dozvoliti pristup.

Imena datoteka

Imena datoteka u NTFS sistemu datoteka mogu da imaju do 255 karaktera. Sufiks koji opisuje tip datoteke nije obavezan, ali se iz praktičnih razloga može koristiti

OSOBINE NTFS – KOMPRESIJA I PODRŠKA „RETKIM“ DATOTEKAMA, ŠIFROVANJE I KLASTERI

Kompresija NTFS datoteka je podržana na individualnoj bazi

Kompresija i podrška „retkim“ datotekama

NTFS sistem datoteka podržava kompresiju datoteka na individualnoj bazi. Za kompresiju se koristi Lempel – Ziv algoritam bez gubitaka podataka. Specijalni algoritam kompresije se koristi za takozvane „retke“ datoteke koje sadrže veliki broj nula. Po ovom algoritmu kompresija se vrši tako što se upisuju samo podaci koji su različiti od nule i broj ponavljanja nula između njih.

Šifrovanje

NTFS sistem datoteka omogućuje šifrovanje i dešifrovanje datoteka i direktorijuma pomoću **Encrypted File System-a**.

Klasteri

NTFS sistem datoteka koristi 64-bitne indekse klastera. Ovo daje mogućnost adresiranja volumena do 16 EB (oko 16 milijarde GB). Međutim Windows XP i Windows Server 2003 operativni sistemi ograničavaju indekse klastera na 32 bita, što znači da je maksimalna veličina volumena 128 TB (ako se koriste 64-bitni klasteri). U Tabeli 1 su date podrazumevajuće veličine klastera volumena NTFS sistema datoteka.

Veličina volumena	Veličina klastera
0 do 512 MB	512 bajta
513 MB do 1 GB	1 KB
1025 MB do 2 GB	2 KB
2 GB i više	4 KB

Slika 1.3.1 Tabela-1 Veličina klastera volumena NTFS sistema datoteka [Izvor: Autor]

PREDNOSTI I MANE NTFS SISTEMA

Pristup datotekama je veoma brz i pouzdan

Neke od **prednosti** korišćenja NTFS sistema su:

- Pristup datotekama je veoma brz i pouzdan
- MFT omogućava oporavak sistema od problema bez gubitka značajne količine podataka
- Sigurnost je znatno povećan u odnosu na FAT

- Enkripcija datoteka se vrši sa EFS sistem za šifrovanje datoteka (EFS - [Encrypting File System](#))
- Koristi se kompresija za smanjenje veličine datoteka
- Štedi prostor na disku

Neke od **mana** korišćenja NTFS sistema su:

- Veliko opterećenje
- Ne preporučuje se za volumene manje od 4 GB
- NTFS volumenima se ne može pristupiti iz: MS-DOS, Windows 95, Windows 98, Linux

▼ Poglavlje 2

FUNKCIJE OPERATIVNOG SISTEMA

ULOGA OPERATIVNOG SISTEMA

Operativni sistem se sastoji od niza specifičnih programa čija je uloga da upravljaju radom samog računarskog sistema i omogući izvršavanje aplikativnih i uslužnih programa

Operativni sistem (engl. **operating system**) se sastoji od niza specifičnih programa čija je uloga da upravljaju radom samog računarskog sistema i omogući izvršavanje aplikativnih i uslužnih programa (engl. **utility programs**). On, takođe, nadgleda i upravlja svim ulazno-izlaznim operacijama sistema i operacijama koje se odnose na obradu podataka. Zbog toga je rad računarskog sistema bez operativnog sistema nemoguć. **Operativni sistem predstavlja posrednika između korisnika, odnosno programa koje on koristi i računarskog sistema, jer obezbeđuje izvršenje korisnikovih komandi i aplikativnih programa.**

Korisnik komunicira sa računarskim sistemom korišćenjem korisničkog interfejsa, koji je deo operativnog sistema. Na taj način korisnik može da upravlja radom jezgra operativnog sistema, a preko njega i radom čitavog računarskog sistema. Korisnik posredstvom korisničkog interfejsa može da pokrene aplikativne ili uslužne programe. U tom slučaju operativni sistem postaje posrednik između programa i hardvera, a ulogu korisničkog interfejsa preuzima program koji se izvršava.

Primer aplikativnih programa su: Microsoft Office (Word, Excel, i sl.), Autocad, programi za računovodstvo i dr. U načelu, za aplikativne programe kažemo da su izgrađeni da reše konkretne problem u mnogim oblastima. Sa druge strane, uslužni programi olakšavaju korisnicima pojedine poslove poput povezivanja podataka na CD, antivirusne zaštite, deinstalacija softvera, omogućavanja korisniku računara upravljanje i kontrolu lokacija podataka i hardverskih komponenata.

ZADACI OPERATIVNOG SISTEMA

Tri zadatka: upravljanje resursima, apstrakcija hardvera, korisnički interfejs

Operativni sistemi se stalno obogaćuju novim funkcijama ili se uz njih dobijaju uslužni programi. Ono što je u nekom operativnom sistemu standardna funkcija, u nekom drugom operativnom sistemu se izvodi preko uslužnih programa. Pored toga različite klase operativnih sistema imaju različite funkcije. Na primer, operativni sistemi PC računari i real-time operativni sistemi imaju veliki broj različitih funkcija. Zbog toga je vrlo teško odrediti

granice operativnog sistema i definisati spisak funkcija za koji se može reći da važi za sve operativne sisteme. Zato će ovde biti reči samo o najvažnijim funkcijama operativnih sistema.

Funkcije koje ima operativni sistem posledice su zadataka koji se postavljaju operativnom sistemu, pa će se prvo razmotriti najvažniji zadaci operativnog sistema. Na osnovu prethodnog razmatranja se može zaključiti da operativni sistem ima tri zadatka:

- Upravljanje resursima
- Apstrakciju hardvera
- Korisnički interfejs

▼ 2.1 UPRAVLJANJE RESURSIMA

UPRAVLJANJE RESURSIMA

Osnovne metode dodeljivanja resursa su: multipleksiranje, zaključavanje (locking) i upravljanje pristupa

Računarski sistem se može posmatrati i kao skup resursa. Resurs je bilo koja hardverska ili softverska komponenta računarskog sistema koja je sposobna da obavi neki zadatak. Pod resursom se tako, na primer, može smatrati tastatura, procesor, memorija ili neka datoteka. Upravljanje resursima je proces dodeljivanja resursa aplikacijama. Sve dok je neki resurs dodeljen aplikacija može da ga koristi na unapred definisani način. Za dodeljivanje resursa se koriste različite metode. Izbor metode zavisi od prirode i načina funkcionisanja resursa. Osnovne metode dodeljivanja resursa su: multipleksiranje, zaključavanje (engl. locking) i upravljanje pristupa.

Multipleksiranje (engl. **multiplexing**) je metod dodeljivanja istog resursa različitim aplikacijama koje u istom vremenskom periodu zahtevaju dodeljivanje resursa. U ovakvim slučajevima operativni sistem po nekom pravilu dodeljuje resurs jednoj aplikaciji tokom kratkog vremenskog perioda koji je dovoljan da se izvrši neki elementaran zadatak. Nakon toga se resurs dodeljuje nekoj drugoj aplikaciji. Multipleksiranje se može opisati i kao kvazi paralelan način pristupa. Ovaj metod se koristi za dodeljivanje procesora, diskova, magistrala i sličnih resursa.

Blokiranje (engl. **locking**) je metod ekskluzivnog dodeljivanja resursa jednoj aplikaciji sve dok ona ima potrebe za tim resursom. Za to vreme druge aplikacije ne mogu da pristupe tom resursu, već se njihov zahtev upisuje u red čekanja. Kada se resurs oslobodi, prva aplikacija iz reda dobija pravo na korišćenje resursa. Tipičan resurs koji se koristi za ovu metodu je štampač. Nije moguće deliti vreme štampača na nivou jedne strane papira. Stoga je neophodno da jedna aplikacija završi započeto štampanje, da bi se štampač dodelio drugoj. Blokiranje se kao metod koristi i za upravljanje dodeljivanja jedinicama magnetne trake, a u nekim slučajevima kada aplikacija traži ekskluzivno pravo, i za pristup datotekama.

Upravljanje pristupa (engl. **access control**) je metod upravljanja resursima kojim operativni sistem realizuje bezbednosna ograničenja. U ovom slučaju se ne radi o upravljanu pristupa

aplikacije hardveru, nego o tome da se jedan proces zaštitи od drugih. Na primer, operativni sistem može da dozvoli jednoj aplikaciji da kreira datoteku kojoj drugi korisnici ne mogu da pristupe.

✓ 2.2 APSTRAKCIJA HARDVERA

ZNAČAJ APSTRAKCIJE HARDVERA

Pod apstrakcijom hardvera se podrazumeva pristup kojim se sva kompleksnost hardvera pojednostavljuje modelom koji nameće operativni sistem

Pod apstrakcijom hardvera se podrazumeva pristup kojim se sva kompleksnost hardvera pojednostavljuje modelom koji nameće operativni sistem. Time su korisnici, programeri i programi rasterećeni od nepotrebnih detalja i informacija. Svu složenost komuniciranja sa hardverom preuzima operativni sistem tako što korisnicima i programima nudi jednostavan i dobro definisan interfejs.

Na primer, operativni sistemi tipično nude model SCSI (**Small Computer System Interface**) diska kao skup blokova u kojima se podaci mogu zapisati. SCSI je skup standard koji se koriste za fizičku konekciju i transfer podataka između računara i perifernih uređaja. SCSI standardi definišu komande protokole i električne i optičke interfejse. U stvarnosti, proces zapisivanja podataka zahteva poznavanje SCSI protokola, SCSI adaptera magistrale, PCI magistrale, pa je i pisanje programa za pisanje i čitanje podataka sa diska vrlo složen posao. Da se korisnici ne bi opterećivali ovakvim detaljima, ovi programi su ugrađeni u sam operativni sistem.

Apstrakcija hardvera koju nudi operativni sistem pruža dve prednosti:

- **Jednostavan interfejs prema aplikaciji.** Napor za pisanje rutina kojim se pristupa hardveru su smanjeni jer se ove rutine pišu samo jednom i mogu se koristiti iz bilo koje aplikacije. Programeri koji pišu aplikativne programe koriste jednostavne interfejse i time su rasterećeni od ovih kompleksnih zadataka.
- **Kompatibilnost.** Tipičan operativni sistem može da se instalira na računarima koji se međusobno razlikuju po ugrađenim komponentama. Tako je, na primer, operativni sistem Windows moguće instalirati i na računaru koji koristi CRT i na računaru koji koristi TFT monitor. Isrtavanje slike na ovim monitorima zahteva različite rutine. S druge strane, programer u trenutku pisanja aplikacije ne zna na kakvim će se sve platformama izvršavati program. Uvođenjem operativnog sistema između aplikacije i hardvera programeri ne moraju da vode računa o komponentama koje će imati računar na kome će aplikacija raditi. Operativni sistem će preuzeti zadatke od aplikacije i aktivirati one svoje rutine koje su zadužene za upravljanje instaliranim hardverskim komponentama.

✓ 2.3 KORISNIČKI INTERFEJS

ZADATAK KORISNIČKOG INTERFEJSA

Zadatak korisničkog interfejsa je da ostvari vezu između korisnika i jezgra operativnog sistema

Treći zadatak operativnog sistema je da obezbedi mehanizam kojim će korisnik upravljati računarskim sistemom i pokretati aplikacije. Ovaj mehanizam se naziva **korisnički interfejs**, a njegov zadatak je da ostvari vezu između korisnika i jezgra operativnog sistema. Kompletan komunikacija računara i korisnika se obavlja posredstvom interfejsa, pa je obično mišljenje korisnika o nekom operativnom sistemu presudno zavisno od efikasnosti i kvaliteta korisničkog interfejsa.

Korisnički interfejs operativnog sistema omogućava korisniku unos i primanje informacija. Korisnički interfejs zasnovan na tekstu prikazuje tekst, a njegove komande se obično unose u komandnu liniju pomoću tastature. Pomoću grafičkog korisničkog interfejsa, funkcije se izvršavaju klikom na dugmiće, ikone i menije pomoću pokazivačkog uređaja (engl. **pointer device**). Moderni grafički interfejsi su evoluirali od korisničkih interfejsa baziranih na tekstu, mada i dalje postoje neki operativni sistemi koji koriste ovakve interfejse.

U većini operativnih sistema, primarni korisnički interfejs je grafički, odnosno umesto da unosite tekstualne komande, komande se izdaju manipulisanjem različitih grafičkih objekata (kao što su ikone). Osnovni princip različitih grafičkih korisničkih interfejsa (GUI) je u velikoj meri isti, pa znajući kako da koristite na primer grafički interfejs Windows operativnog sistema, najverovatnije ćete znati kako koristite i Linux ili neki drugi GUI.

Većina GUI-a imaju sledeće osnovne komponente:

- početni meni sa programskim grupama
- listu sa zadacima koja prikazuje programe koji su aktivni
- desktop
- razne ikone i prečice.

✓ 2.4 PREGLED FUNKCIJA OPERATIVNOG SISTEMA

NAJAVAŽNIJE FUNKCIJE OPERATIVNOG SISTEMA

Imajući u vidu sve zadatke koje operativni sistem treba da izvrši mogu se definisati najvažnije funkcije operativnog sistema

Imajući u vidu sve zadatke koje operativni sistem treba da izvrši mogu se definisati najvažnije funkcije operativnog sistema. U najvažnije funkcije operativnog sistema spadaju:

- Obezbeđivanje korisničkog interfejsa za administratore i korisnike
- Interpretacija komandnog jezika
- Planiranje i raspoređivanje poslova
- Upravljanje resursima
- Upravljanje ulazno-izlaznim operacijama
- Upravljanje prekidima i greškama
- Očuvanje integriteta procesa i sistema
- Praćenje korišćenja računarskih resursa.

OBEZBEĐIVANJE KORISNIČKOG INTERFEJSA ZA ADMINISTRATORE I KORISNIKE

Korisnički interfejs može biti alfanumerički i grafički

Posmatrano na nivou celog računarskog sistema, zadatak **korisničkog interfejsa** je da obezbedi dvosmernu interakciju između korisnika i računarskog sistema. Deo ovog zadatka preuzima hardver, odnosno ulazno-izlazni uređaji računarskog sistema. Sa aspekta operativnog sistema, korisnički interfejs je deo operativnog sistema koji programski realizuje interakciju korisnika sa računarskim sistemom posredstvom ulazno-izlaznih uređaja, a deo softver. Pomoću korisničkog interfejsa korisnik izdaje komande operativnom sistemu, pokreće programe i komunicira sa programima i operativnim sistemom.

Korisnički interfejs može biti alfanumerički i grafički. Neki operativni sistemi imaju više različitih korisničkih interfejsa. Tako se sreću korisnički interfejsi koji su namenjeni administratorima i oni koji su namenjeni običnim korisnicima računarskog sistema. Grafički korisnički interfejs (engl. *Graphic user interface*) prikazuje vizuelne elemente uz pomoć kojih se intereaguje sa računarom.

KOMANDNI INTERPRETER

Komandni interpreter interpretira linje teksta u kontekstu određenog operativnog sistema

Komandni interpreter (engl. *command interpreter*) ima zadatak da proveri validnost izdate komande operativnom sistemu i da inicira njeno izvršenje. On interpretira linije teksta u kontekstu određenog operativnog sistema. U opštem slučaju komanda se sastoji od mnemonika komande, njenih opcija ili modifikatora i parametara. U nekim slučajevima komanda se unosi sa greškom ili nepotpuno. Komandni interpreter zato mora da proveri validnost komande i u slučaju da je ona nevalidna mora da preko korisničkog interfejsa stavi do znanja korisniku razloge neizvršenja komande. Ovakva dijagnostika služi da pomogne korisniku da ispravno unese komandu ili shvati zašto komanda ne može da se izvrši.

PLANIRANJE I RASPOREĐIVANJE POSLOVA

Planiranje i raspoređivanje poslova je proces kojim se kod multiprogramske sistema iniciraju poslovi, održava lista poslova koji čekaju na izvršenje i dodeljuju resursi koji se zahtevaju

Planiranje i raspoređivanje poslova (engl. **scheduling**) je proces kojim se kod multiprogramske sistema iniciraju poslovi, održava lista poslova koji čekaju na izvršenje i dodeljuju resursi koji se zahtevaju. Deo operativnog sistema koji izvršava ovu funkciju se naziva **raspoređivač** (engl. **scheduler**) ili **dispečer** (engl. **dispatcher**). Da bi se odredila pozicija nekog posla u listi poslova koji čekaju na obradu koriste se informacije o prioritetu, vremenu provedenom u redu poslova i potrebnim resursima.

UPRAVLJANJE RESURSIMA

Upravljanje procesorom, memorijom, datotekama, magistralama, ulazno-izlaznim uređajima

Upravljanje resursima je jedan od najkompleksnijih zadataka operativnog sistema jer računarski sistem raspolaže velikim brojem resursa. Funkcija upravljanja resursima obuhvata:

- Upravljanje procesorom u smislu dodele procesora nekom programu
- Upravljanje operativnom i virtuelnom memorijom: dodeljivanje memorije programima i oslobađanje memorije
- Upravljanje datotekama: kreiranje, pisanje, čitanje, brisanje i evidentiranje datoteka na spoljnim memorijama
- Upravljanje magistralama
- Upravljanje ulazno-izlaznim uređajima: dodeljivanje i oslobađanje uređaja

Upravljanje ulazno-izlaznim operacijama je funkcija operativnog sistema kojom se postiže zahtevana funkcionalnost ulaznih i izlaznih uređaja. Delovi operativnog sistema koji obavljaju ove funkcije su posebno pisani za pojedine vrste ulazno izlaznih uređaja. Na primer, upravljanje radom matričnog štampača se razlikuje od upravljanja radom laserskog štampača, pa je za svaki od ovih uređaja potreban poseban program.

UPRAVLJANJE PREKIDIMA I GREŠKAMA

Proces koji je u toku treba prekinuti tako da se očuva integritet podataka i da se kasnije može nastaviti.

Izuzeci u normalnom radu operativnog sistema nastaju kada je neophodno prekinuti proces koji se izvršava.

Prekidi (engl. **interrupts**) omogućavaju hardveru da pošalje signal procesoru. Na primer, dok kucate na tastaturi, kontroler tastature (hardverski uređaj koji upravlja tastaturom) izdaje električni signal procesoru da bi upozorio

operativni sistem na nove pritiske na tastaturi.

Ovi električni signali su takozvani prekidi. Procesor prima prekid i signalizira operativni sistem kako bi operativni sistem mogao da odgovori na nove podatke.

Hardverski uređaji generišu prekide asinhrono (u odnosu na procesorski sat). Shodno tome, jezgro može biti prekinuto u bilo kom trenutku da obrađuje prekide.

Prekidi predstavljaju elektronske signale hardverskih uređaja koji su usmereni ka kontroloru prekida:

- prilikom prijema prekida, kontrolor prekida šalje signal procesoru
- procesor detektuje signal i prekida svoj trenutni rad kako bi obradio prekid
- procesor može da obavesti operativni sistem da je došlo do prekida, kako bi operativni sistem mogao da odgovori na njega.

Potreba za prekidom može da se javi zbog toga što se pojavio neki drugi proces koji ima viši prioritet ili ne može da čeka ili se dogodila neka programska ili hardverska greška. Funkcija **upravljanja prekidima i greškama** ima zadatak da razrešava ovakve situacije. **Proces koji je u toku treba prekinuti tako da se očuva integritet podataka i da se kasnije može nastaviti.**

OČUVANJE INTEGRITETA PROCESA I SISTEMA

Zaštita računarskog sistema i zaštita podataka od neovlašćenog pristupa

Očuvanje integriteta procesa i sistema je vrlo važno za bezbedan rad računarskog sistema. Ova funkcija operativnog sistema brine o zaštiti računarskog sistema programa i podataka od neovlašćenog korišćenja kao i štete koja može da nastane zbog nedostataka u programima i operativnom sistemu ili slučajnih grešaka korisnika.

PRAĆENJE KORIŠĆENJA RAČUNARSKIH RESURSA

Vodenje evidencije korišćenja resursa iz bezbednosnih razloga

Jedna od važnih funkcija operativnog sistema je **praćenje korišćenja računarskih resursa**. Evidencija o korišćenju računarskih resursa se vodi iz bezbednosnih i finansijskih razloga. Sa aspekta istraživanja narušavanja integriteta sistema vrlo je važno imati tačnu evidenciju koji su korisnici koristili sistem i koje su operacije vršili. **Operativni sistem zapisuje podatke o korišćenju resursa u log datoteke koje se u slučaju potrebe mogu analizirati.**

▼ Poglavlje 3

KARAKTERISTIKE FAT32 I FAT16 SISTEMA DATOTEKA

FAT32

FAT32 koristi 32-bitne identifikatore klastera, ali su 4 bita rezervisana, tako da se za indeksiranje koriste samo 28 bitova

FAT32 sistem datoteka organizuje podatke na diskovima i disketama. Najvažnija prednost ovog sistema je da je pristup podacima moguć iz operativnih sistema DOS, Windows i OS/2. FAT sistem je jedini koji trenutno podržava diskete. FAT32 je podrazumevajući sistem datoteka za operativne sisteme Windows 95 OSR2, Windows 98 i Windows Me.

FAT32 koristi 32-bitne identifikatore klastera, ali su 4 bita rezervisana, tako da se za indeksiranje koriste samo 28 bitova.

Validna imena datoteka imaju sledeću sintaksu:

[[drive:]][[directory\]]filename[[.extension]]

Parametar *drive* je ime postojećeg uređaja ili particije i može biti bilo koje slovo od A do Z. Ime uređaja je praćeno simbolom „：“.

Parametar *direcory* specificira putokaz do datoteke i mora da se završi simbolom „\“ da bi se putokaz odvojio od imena datoteke. Ako specificirani direktorijum nije tekući direktorijum, putokaz se satoji od imena svih naddirektorijuma datoteke do korena hijerarhijskog stabla direktorijuma. Koren se označava simbolom „\“.

Na primer, ako je datoteka *primer.txt* u direktorijumu *tekstovi*, koji se nalazi u direktorijumu *rad*, koji se nalazi u korenom direktorijumu particije D onda je puno ime datoteke ili apsolutni putokaz:

D: \rad\tekstovi\primer.txt

Ime direktorijuma može da se sastoji od do 8 slova, brojeva i sledećih specijalnih karaktera:

\$ % ' - _ @ { } ~ ` ! # ()

Parametar *filename* specificira ime datoteke koje može biti praćeno sufiksom (extension) od maksimalno 3 slova. Između imena i sufiksa stoji karakter „.“ (tačka). Maksimalna veličina klastera može biti 32 KB, pa FAT32 sistem datoteka može teoretski da radi sa volumenima kapaciteta do 8 TB.

Veličina volumena	Veličina klastera
32MB do 8 GB	512 baita
8 do 16 GB	1 KB
16 do 32 GB	2 KB
32 GB	4 KB

Slika 3.1 Veličine klastera u FAT32 sistemu datoteka [Izvor: Autor]

PREDNOSTI I MANE FAT SISTEMA

FAT sistem ima efikasno korišćenje prostora na disku

Neke od **prednosti** korišćenja FAT sistema su:

- Efikasno korišćenje prostora na disku (ne moraju da koristi kontinualno skladište (neprekidni prostor u jednoj celini) za velike fajlove)
- Imena fajlova mogu biti do 255 karaktera (FAT32)
- Lako se oporavljaju izbrisane datoteke nakon brisanja ((i) sistem stavlja E5h na prvo mesto u imenu datoteke; (ii) datoteka ostaje na disku

Neke od **mana** korišćenja FAT sistema su:

- Čuvanje više datoteka na particiji usporava performanse
- Hard disk lako fragmentira
- Nedostatak bezbednosti (NTFS dozvoljavanja pravo pristupa datotekama i direktorijumima)
- Problem sa integritetom datoteka (izgubljeni klasteri, nevalidna imena datoteka i direktorijuma)

FAT16

FAT16 koristi 16 bita, tako da može da radi sa do 65 536 klastera

Ovaj sistem datoteka, koji se više gotovo ne koristi, za indeksiranje klastera upotrebljava 16 bita, tako da može da radi sa do 65.536 klastera. Pri maksimalnoj veličini klastera koja je 64 KB, ukupan memoriski kapacitet volumena može biti do 4GB.

UDF

UDF - **Universal Disk Format** je definisan od strane organizacije Optical Storage Technology Association (OSTA) kao sistem datoteka za optičke medije koji bi trebao da se primenjuje u svim operativnim sistemima. UDF je projektovan da zameni stari CDFS format i da bi se dodala podrška za DVD medije. Implementacija sistema datoteka je izvedena prema standardu ISO 13346. Osnovne osobine UDF-a su:

- Imena datoteka mogu biti dužine do 255 karaktera
- Maksimalna dužina putokaza datoteke može biti do 1023 karaktera

CDFS

Sistem datoteka CDFS (**CD-ROM file system**) je projektovan 1988 za rad CD-ROM medije. Ovaj sistem je usaglašen sa standardom ISO 9660. CDFS sistem datoteka ima sledeća ograničenja:

- Imena direktorijuma i datoteka mogu biti duga maksimalno 32 karaktera
- Stablo direktorijuma može imati do 8 hijerarhijskih nivoa po dubini

ŠTA JE BRŽE FAT16, FAT32 ILI NTFS?

Za male volumene, FAT16 ili FAT32 obično daju brži pristup datotekama nego NTFS

Za male volumene, FAT16 ili FAT32 obično daju brži pristup datotekama nego NTFS zbog toga što:

- je struktura FAT sistema jednostavnija
- veličina FAT direktorijuma je manja za jednak broj datoteka.

FAT nema kontrolu koja reguliše da li korisnik može da pristupi datoteci ili direktorijumu. Zbog toga, sistem ne mora da proveri dozvole za pristup pojedinačnim datotekama ili da određeni korisnik ima pristup datoteci ili direktorijumu. Ova prednost je minimalna, jer OS tek treba da utvrdi da li je fajl samo za čitanje, odnosno da li je fajl na FAT ili NTFS volumenu.

Kada se uporede brzine operacije koje se izvršavaju nad velikim direktorijumima, koji sadrže i duga i kratka imena datoteka, brzina operacije FAT zavisi od same operacije i veličine direktorijuma. Ako FAT traži datoteku koja ne postoji, on mora da pretraži ceo direktorijum. Ova operacija traje duže na FAT strukturi nego na B-stablu koji koristi NTFS sistem. Koristeći matematičku terminologiju, prosečno vreme da se pronađe datoteka u FAT direktorijumu je opisana funkcijom **$N / 2$** , gde **N** predstavlja broj datoteka. U NTFS datoteci, prosečno vreme da se pronađe datoteka je opisano funkcijom **$\log n$** .

▼ Poglavlje 4

UNIX FAMILIJA OPERATIVNIH SISTEMA

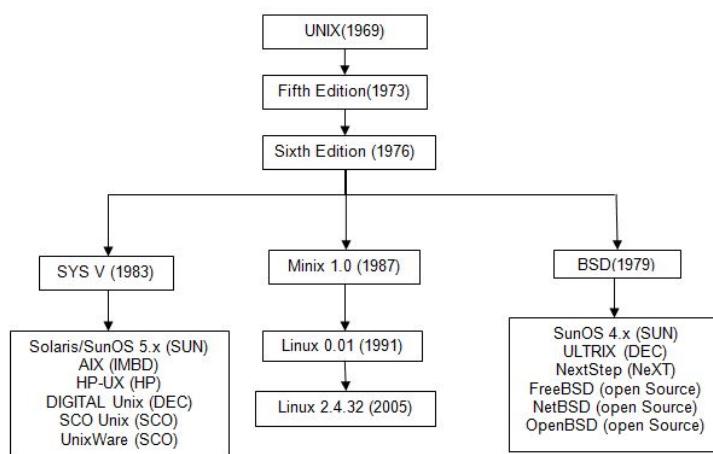
RAZVOJ UNIX OPERATIVNIH SISTEMA

Početak razvoja UNIX-a je bio 1969.godine u AT&T Bell Lab u jeziku C

Od početka svog razvoja operativni sistem **UNIX** je bio open-source kako bi se uspelo u ideji da bude operativni sistem koji će moći da se implementira na različitim hardverskim platformama. Mnoge organizacije i pojedinci su razvili svoje verzije UNIX-a. To je dovelo do toga da danas postoji veliki broj UNIX baziranih operativnih sistema koji su međusobno kompatibilni, pa možemo reći da postoji UNIX familija operativnih sistema.

Tim koga su predvodili Ken Tompson i Denis Ritchie počeo je da razvija UNIX 1969. godine u AT&T Bell Telephone laboratorijama. Godine 1973. objavljena je verzija UNIX-a napisana u jeziku C, što je omogućilo da se kompajlira za različite hardverske platforme. Danas postoji preko 100 različitih verzija UNIX-a. Da bi UNIX opstao i ostao sinonim za open-source sisteme, formirane su različite organizacije koje se bave pitanjima standardizacije i razvoja UNIX-a. Najvažnija organizacija je X/Open koja se bavi razvojem i integracijom standarda.

Na slici je prikazan istorijski razvoj UNIX familije operativnih sistema.



Slika 4.1.1 Razvoj UNIX familije operativnih sistema [Izvor: Autor]

PODGRUPE UNIX OPERATIVNIH SISTEMA

Komercijalne i open source verzije UNIX-a čine dve osnovne grupe ovog operativnog sistema

Može se reći da danas postoje dve podgrupe u familiji operativnih sistema UNIX. Prvu grupu čine **komercijalne verzije UNIX**-a namenjene određenoj vrsti računara. U ovu podgrupu spadaju:

- HP/UX za Hewlett-Packard računarske sisteme
- Solaris za Sun i SPARC-kompatibilne računarske sisteme
- IRIX za Silicon Graphics računarske sisteme
- Digital UNIX za Digital Alpha računarske sisteme
- AIX za IBM računarske sisteme
- SCO UNIX namenjen računarima baziranim x86 procesorima (2001. godine Caldera je kupila SCO)

Razvoj ovakvih operativnih sistema je skup, a broj instalacija nedovoljno veliki da bi se pokrili troškovi razvoja. Zbog toga ove kuće prelaze na neku od verzija UNIX-a iz podgrupe

Drugu podgrupu čine **open-source verzije UNIX**-a, koje su pisane sa idejom da se slobodno distribuiraju i da mogu da rade na različitim hardverskim platformama. U ovu grupu spadaju:

- FreeBSD
- NetBSD
- OpenBSD
- Minix
- Linux

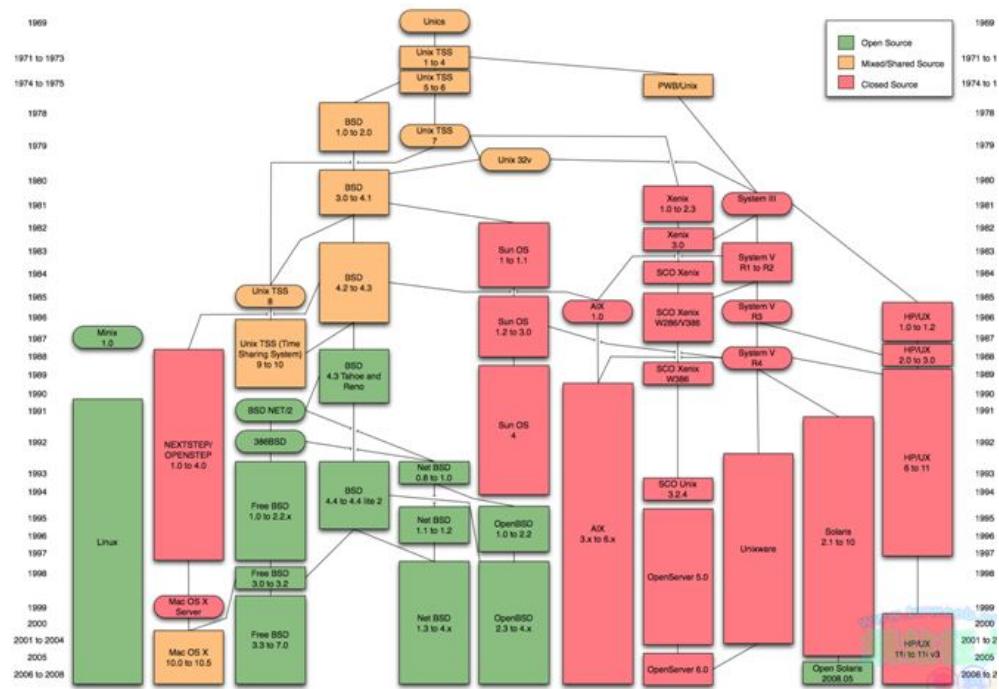
Najpoznatija varijanta UNIX-a iz ove podgrupe je Linux.

Za sve operativne sisteme bazirane na UNIX-u važe iste zajedničke osobine:

- Izvanredna konektivnost – UNIX je bio i ostao osnova Interneta
- Stabilnost – UNIX se koristi preko 30 godina
- Skalabilnost – UNIX se koristi i na PC računarima i na super računarima
- Multi-programski i multi-korisnički operativni sistem.

RAZVOJ OPEN SOURCE UNIX SISTEMA

Pregled istorijskog razvoja open source UNIX sistema



Slika 4.1.2 Razvoj open-source UNIX sistema

[Izvor: http://postimages.twimg.org/Clanguages/Unix_history.png]

4.1 SISTEM DATOTEKA UNIX I LINUX OPERATIVNIH SISTEMA

SISTEM DATOTEKA

U UNIX-u je sve datoteka; ono što nije datoteka je proces

Među korisnicima UNIX operativnog sistema, a isto važi i za Linux, kruži sledeća krilatica: „U UNIX-u je sve datoteka; ono što nije datoteka je proces“. U UNIX-u je pojam datoteke mnogo generalniji nego u Windows-u. Takođe gotovo da nema razlike između datoteke i direktorijuma, jer je direktorijum datoteka koja sadrži imena onih datoteka i direktorijuma koji su sadržani u posmatranom direktorijumu. U UNIX-u se, pored podataka i programa, pod datotekom smatraju i ulazni i izlazni uređaji i uopšte svi uređaji.

Sistem datoteka operativnih sistema UNIX i Linux je gotovo identičan. U oba sistema se koristi hijerarhijska struktura u obliku stabla na čijem je vrhu koren direktorijum (**root**) i koji se u putokazima simbolički obeležava znakom „/“.

PARTICIJE

U UNIX-u postoje dve vrste particija: particije sa podacima i swap particije

U cilju lakše manipulacije podacima i datotekama i zbog zaštite podataka UNIX operativni sistem koristi **particije**. Postoje dve vrste particija:

- Particije sa podacima u kojoj su smešteni objekti sistema datoteka,
- Swap particija koja služi kao proširenje operativne memorije računara, dakle neka vrsta virtualne memorije. Ova particija se koristi kao memorijski prostor onda kada procesima koje računar izvršava nije dovoljna instalirana operativna memorija. Onda se deo podataka iz radne memorije prebacuje na ovu particiju, a kasnije se po potrebi opet vraća u operativnu memoriju i zamenjuje (swap) nepotrebne podatke u radnoj memoriji.

Svi sistemi imaju **jednu korenу (root) particiju, jednu ili više particija sa podacima i jednu ili više swap particiju**. Svaka od particija može imati svoj sistem datoteka. To znači da se na istom računaru može instalirati jedna particija sa NTFS sistemom datoteka, druga sa UNIX-ovim sistemom datoteka ext2 ili ext3, a treća sa FAT32.

Standardna korena particija sadrži sve datoteke potrebne za podizanje sistema, kao što su konfiguracione datoteke, najosnovnije komande, serverske programe, sistemske biblioteke, privremene (temporary) datoteke i direktorijum administratora.

VRSTE DATOTEKA

Vrste datoteka su uvedene kako bi se datoteke međusobno razlikovale

Da bi se datoteke međusobno razlikovale uveden je pojam **vrste datoteke**. UNIX-ov sistem datoteka razlikuju nekoliko vrsta datoteka:

- Regularne datoteke
- Direktorijumi
- Specijalne datoteke
- Veze
- Domenski soketi
- Imenovani kanali

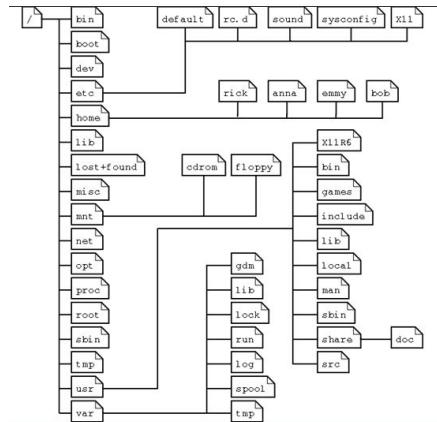
Regularne datoteke. Ovo su datoteke koje sadrže podatke, rezultate programa i izvršne programe.

DIREKTORIJUMI

Tipično stablo u UNIX i Linux sistemima su formirani od korenog direktorijuma

Direktorijumi. Direktorijumi su datoteke koje sadrže informacije o datotekama i direktorijumima koji su sadržani u posmatranom direktorijumu.

UNIX i Linux imaju neke uobičajene direktorijume u kojima se smeštaju datoteke određenog tipa. Na slici je prikazano tipično stablo direktorijuma korene particije koje je formirano RedHat distribucijom Linux-a.



Slika 4.2.1 Stablo direktorijuma korene particije koje je formirano RedHat distribucijom Linux-a

[Izvor: <https://tldp.org/LDP/intro-linux/html/images/FS-layout.png>]

Ispod korenog direktorijuma koji je na vrhu hijerarhijskog stabla su direktorijumi čija je namena opisana u tabeli niže.

Direktorijum	Sadržaj
/bin	Sistemski programi, sistemski administratori i korisnici
/boot	Datoteke za startovanje sistema, kernel, vmlinuz. Vrlo često i grub (Grand Unified Boot loader) podaci.
/dev	Sadrži reference na sve hardverske komponente sistema osim CPU. Komponente su prikazane kao specijalne datoteke.
/etc	Najvažnije konfiguracione datoteke sistema.
/home	Osnovni direktorijum korisnika.
/initrd	U nekim distribucijama informacije potrebne za podizanje sistema.
/lib	Bibliotečke datoteke potrebne sistemu i programima.
/lost+found	Svaka particija ima lost+found direktorijum u kome se čuvaju datoteke koje su spašene prilikom otkaza sistema.
/misc	Pomoćni direktorijum.
/mnt	Standardna tačka montiranja spoljnih sistema datoteka, kao što su na primer CD-ROM ili digitalni foto aparat.
/net	Standardna tačka montiranja za udaljene sisteme datoteka koji se nalaze negde na mreži.
/opt	Direktorijum u kome se čuvaju softver drugih proizvođača (third party software).
/proc	Virtualni sistem datoteka u kome se čuvaju informacije o svim procesima koji su aktivni u sistemu.
/root	Osnovni direktorijum administratorskih korisnika.
/sbin	Programi koje koristi sistem ili sistemski administrator.
/tmp	Direktorijum za smeštaj privremenih datoteka sistema.
/usr	Korisnički programi, biblioteke, dokumenti i slično.
/var	Direktorijum u kome se privremeno smeštaju datoteke sa promenljivama, privremene datoteke koje su kreirali korisnici, kao što su log datoteke, datoteke koje čekaju u redu za štampanje, datoteke prevućene sa Interneta, datoteke za narezivanje na CD i slično.

Slika 4.2.2 Tabela-1 Opis direktorijuma u korennoj partičiji

[Izvor: Autor]

SPECIJALNE DATOTEKE

Specijalne datoteke sadrže podatke o drajverima za sve uređaje osim za centralni procesor

Ovo su datoteke sa podacima o drajverima za sve uređaje osim centralnog procesora. Veći deo ovih datoteka je smešten u direktorijumu /dev. U tabeli niže se daje lista imena najčešćih uređaja koji se tretiraju kao specijalne datoteke.

Ime	Uredaj
cdrom	CD uređaj
console	Trenutno korišćena konzola
cua*	Senjski portovi
dsp*	Uređaji za uzorkovanje (samplering) i zapisivanje
fd*	Disketne jedinice
hd[a-t][1-16]	Standardna podrška za IDE diskove sa maksimalnim brojem particija na svakom
ir*	Infracrveni uređaji
isdn*	Upravljači ISDN konekcije
js*	Joystickovi
lp*	Štampači
mem	Memorija
midi*	midi plejer
mixer* i music	Idealizovani model miksera
modem	Modem
mouse	Sve vrste miševa
null	Kanta za otpatke bez dna (Bottomless garbage can)
par*	Paralelni portovi
pty*	Pseudo terminali
radio*	Radio
ram*	Uređaj sa koga se učitava OS (boot device)
sd*	SCSI diskovi sa particijama
sequencer	Kontroler MIDI uređaja
tty*	Virtualna konzola koja simulira vt100 terminal
usb*	USB kartice i skeneri
video*	Grafička kartica

Slika 4.2.3 Tabela-2 Lista imena najčešćih uređaja koji se tretiraju kao specijalne datoteke [Izvor: Autor]

VEZE, DOMENSKI SOKETI I IMENOVANI KANALI

Specijalni tipovi datoteke

Veze

Pomoću ovih datoteka se datoteke mogu učiniti vidljivim u različitim delovima stabla sistema datoteka. Pomoću veza ([links](#) ili [shortcuts](#)) se mogu za istu datoteku kreirati dva imena. Postoje dve vrste veza:

- **Čvrste reference** (engl. [hard links](#)) povezuju dva ili više imena sa istom datotekom unutar jedne particije. Ako se obrišu sve čvrste reference, praktično se briše i datoteka.
- **Simboličke reference** ili prečice (engl. [soft links](#)) koje su u stvari male datoteke u kojima se nalazi pokazivač na datoteku na koju se odnosi simbolička referenca. Simboličke reference se mogu odnositi i na datoteke koje se nalaze u drugim particijama. Uklanjanje simboličke reference nema uticaja na datoteku na koju pokazuje.

Domenski soketi

Ovo je specijalni tip datoteke, sličan TCP/IP soketima, koji obezbeđuje umrežavanje procesa uz zaštitu koju obezbeđuje kontrola pristupa sistema datoteka.

Imenovani kanali (engl. **Named pipes**)

Ovo su specijalne datoteke koje služe da obezbede dvosmernu komunikaciju između dva procesa (IPC - **Inter Process Communication**). Tako na primer, izlaz iz jednog procesa može biti zapisivan u ovakvu datoteku, kako bi bio ulaz za drugi proces.

LINUX SISTEM DATOTEKA - VIDEO

Linux File System/Structure Explained

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 4.2 OPIS SISTEMA DATOTEKA UNIX-a

TRADICIONALNI SISTEMI DATOTEKA UNIX FAMILIJE

Različite particije mogu imati različite sisteme datoteka

Prilikom formatiranja particija uređaja spoljne memorije na sistemima koji koriste operativni sistem UNIX ili njegove derivate, administrator odlučuje koji će sistem datoteka instalirati. Različite particije mogu imati različite sisteme datoteka. UNIX podržava veliki broj sistema datoteka iz UNIX, Windows i MacOS familije operativnih sistema. UNIX sistemi datoteka se mogu podeliti na dve grupe: tradicionalni sistemi datoteka i sistemi datoteka sa dnevnikom (**journaling filesystem**). Sistemi datoteka sa dnevnikom upisuju u posebnu datoteku sve izmene koje se vrše na disku. Ukoliko se operacija upisivanja obavi regularno ovi podaci se brišu. U slučaju da dođe da prekida operacije upisa iz bilo kog razloga, sadržaj diska može da se vrati u prvobitno stanje na osnovu podataka iz dnevnika. Najvažniji **tradicionalni sistemi datoteka** UNIX familije operativnih sistema su:

- Minix – sistem datoteka operativnog sistema Minix, koji je bio prethodnica Linux-a
- xia – ekstenzija Minix-a
- ext2fs (**Linux second extended filesystem**) – prioritetni sistem datoteka Linux operativnog sistema

Najvažniji **sistemi datoteka sa dnevnikom** UNIX familije operativnih sistema su:

- ext3fs – (**Linux third extended filesystem**) – ext2fs proširen dnevnikom transakcija
- ReiserFS
- XFS – Silicon Graphic-ov sistem datoteka koji radi na IRIX i Linux operativnom sistemu

U tabeli niže se daju uporedne karakteristike nekih sistema datoteka koje se koriste u Linux operativnom sistemu

Osobine	Minix FS	Ext FS	Ext2 FS	Xia FS
Maksimalna veličina sistema datoteka	64 MB	2 GB	4 TB	2 GB
Maksimalna veličina datoteke	64 MB	2 GB	2 GB	64 MB
Maksimalna veličina imena datoteke	16/30 c	255 c	255 c	248 c
Datum kreiranja, poslednje modifikacije	Ne	Ne	Da	Da
Proširivost	Ne	Ne	Da	Ne
Promenljiva veličina bloka	Ne	Ne	Da	Ne
Održavan	Da	Ne	Da	?

Slika 4.3.1 Uporedne karakteristike nekih sistema datoteka koje se koriste u Linux operativnom sistemu [Izvor: Autor]

OTVORENOST UNIX BAZIRANIH OPERATINIH SISTEMA

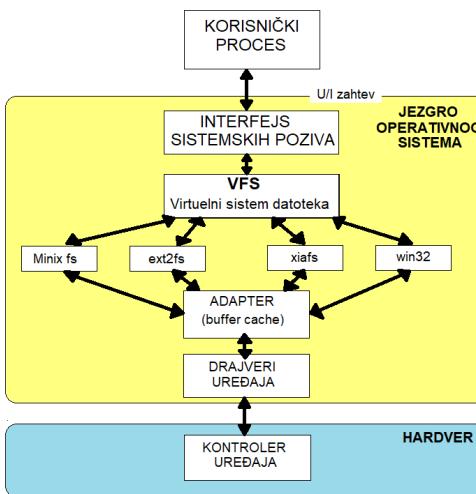
Da bi operativni sistemi iz familije UNIX-a mogli da istovremeno koriste više različitih sistema datoteka uveden je virtualni sistem datoteka

Kao primer otvorenosti UNIX baziranih operativnih sistema navode se sistemi datoteka sa kojima može da radi operativni sistem Linux:

- prioritetni sistem datoteka: ext2fs
- sistemi datoteka koje se mogu instalirati na particijama: ext2fs, ext3fs, FAT
- sistemi datoteka koji se mogu čitati/pisati: FAT32, VFAT, FFS, NTFS, Minix, Xenix, coda, DOS, xia,
- sistemi datoteka koji se mogu čitati: NTFS, HFS (Macintosh), UFS, HPFS, sysvfs, adfs,

Da bi operativni sistemi iz familije UNIX-a mogli da istovremeno koriste više različitih sistema datoteka uveden je virtualni sistem datoteka (**VFS - Virtual File System**). VFS je deo jezgra operativnog sistema čiji je zadatak da bude posrednik između procesa i različitih sistema datoteka, tako što integriše sve sisteme datoteka u jedan virtualni sistem datoteka. Na taj način se programeri rasterećuju brige oko toga koji će sistem datoteka biti instaliran.

Kada neki proces generiše sistemski poziv koji se odnosi na ulazno-izlazne operacije sa datotekama, jezgro operativnog sistema poziva funkciju koja se nalazi u VFS-u. Ova funkcija obavlja radnje koje su struktorno nezavisne, a zatim preusmerava poziv na jednu od funkcija koja zavisi od fizičkog sistema datoteka (na primer ext2fs), koja je odgovorna za struktorno zavisne operacije. Rutine sistema datoteka koriste funkciju **buffer cache** da proslede ulazno-izlazni zahtev uređaju.



Slika 4.3.2 Virtuelni sistem datoteka (VFS) operativnih sistema UNIX familije [Izvor: Autor]

EXT4 SISTEM DATOTEKA - VIDEO

Introduction to the Ext4 File System for Linux

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

❖ 4.3 ARHITEKTURA UNIX-a

NIVOI ARHITEKTURE UNIX-A

Jezgro - Ljuska - Alati

Arhitektura operativnog sistema UNIX se može podeliti u tri nivoa funkcionalnosti. Najniži nivo je **jezgro** koje raspoređuje zadatke, upravlja resursima i upravlja bezbednošću. Sledeći nivo je **ljuska** (**shell**) koja deluje kao korisnički interfejs. Njen zadatak je da interpretira korisničke komande i startuje aplikacije. Najviši nivo predstavljaju različiti korisni **alati** koji proširuju funkcionalnost UNIX-a.

UNIX operativni sistem ima sledeće osobine i mogućnosti:

- Izvršenje više programa istovremeno (**multitasking**)
- Višekorisnički rad (**multiuser**)
- Jezgro napisano u jeziku visokog nivoa (uobičajeno C)
- Programski interfejs
- Koristi datoteke da bi referencirao uređaje i druge objekte (ovde se misli na to da se uređaji posmatraju kao datoteke)
- Veliki broj jednostavnih alata

- Koristi kanale (**pipes**) i filtere da izvrši kompleksne zadatke upotrebom jednostavnih alata
- Podrazumevajući korisnički interfejs je na bazi komandne linije
- Ugrađene protokole za umrežavanje (**Transmission Control Protocol/Internet Protocol - TCP/IP**)
- Sistemski servisi su obezbeđeni pomoću samostartujućih demon (**daemon**) procesa

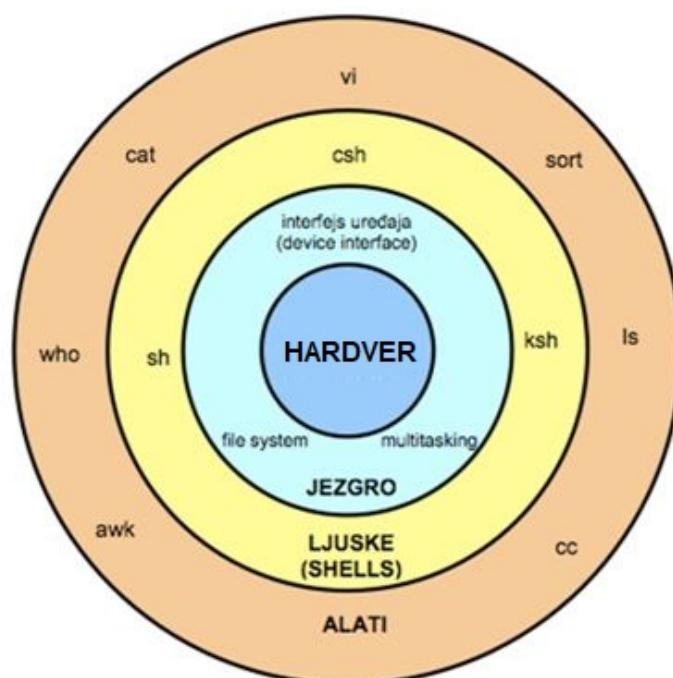
▼ 4.4 LJUSKE UNIX OPERATIVNOG SISTEMA

ČEMU SLUŽE LJUSKE UNIX OPERATIVNOG SISTEMA

Ljuske su interfejs između korisnika i sistema

Operativni sistem UNIX koristi ljuske kao interpretatore komandnog jezika. Ljuske su interfejs između korisnika i sistema. Razvoj UNIX-a je bio praćen i razvojem ljuski, tako da se sada korisnicima UNIX-a nudi veliki broj ljuski. Najpoznatije ljuske su:

- **sh** ili Bourne Shell – originalna UNIX ljuska.
- **bash** ili Bourne Again shell – standardna GNU ljuska koja se najčešće koristi uz Linux operativni sistem. Kompatibilna je sa sh je dobijena proširenjem ove ljuske, što znači da komande koje rade u sh rade i u bash.
- **csh** ili C shell – sintaksa ove ljuske podseća na programski jezik C, pa je posebno dobro prihvaćena od strane programera.
- **tcsh** ili Turbo C shell – proširena verzija csh, koja se odlikuje brzinom i lakoćom korišćenja.
- **ksh** ili Korn shell – proširena verzija sh.



Slika 4.4.1 Arhitektura UNIX-a [Izvor: Autor]

▼ Poglavlje 5

APPLE SISTEMSKE DATOTEKE

APFS DATOTEKE

APFS datoteke pružaju preciznost na nivou jedne nanosekunde, a ova funkcija, u kombinaciji sa novom funkcijom kloniranja olakšava skladištenje više verzija datoteke u minimalnom prostoru

Sistemska datoteka koje koriste Apple uređaji, uključujući operativne sisteme macOS, iOS, tvOS i watchOS, se zove **Apple New File System** (**APFS**). APFS je zamenio stare HFS+ sistemske datoteke, koje su se koristile na svim Mac uređajima od 1998. Dok je HFS+ bio baziran na svom prethodniku HFS, koji se koristio od 1985. APFS je optimizovan za SSD uređaje (fleš memorija). APFS podržava većinu funkcija u HFS+. Ono što APFS želi da uradi je da unificira sve Apple brendirane proizvode sa sistemom datoteka koji se može skalirati od uređaja koje nosimo na ruci do iMac Pro uređaja.

APFS je kompletno novi sistem koji je napravljen za moderne hardver i klaud, tako da dodaje enkripciju, bezbednost i pouzdanost, koji nisu postojali u prethodnim starijim Apple sistemima. APFS dodaje povećanu sigurnost i pouzdanost i značajno povećava brzinu prenosa datoteka.

APFS nudi mnogo prednosti u pogledu SSD performanse i trajnosti, kao i šifriranje za bilo koji tip diska. Međutim, APFS nije kompatibilan sa starijim verzijama OS X ili MacOS.

APFS koristi integriranu enkripciju umesto prethodne tehnike za enkripciju koja je koristila karakteristiku FileVault u OS X koja je polako šifrirala i dešifrovala čitav disk. APFS može šifrirati čitave diskove i pojedinačne datoteke sa odvojenim ključevima za datoteku i za njegove metapodatke. Na taj način se dobija granularna kontrola koja omogućuje pojedinačnim korisnicima da modifikuje podatke u datoteci.

Prethodni Apple-ov sistem datoteka je radio sa preciznošću od jedne sekunde, što nije dovoljno da se prate izmene datoteka sa današnjim hardverom. APFS datoteke pružaju preciznost na nivou jedne nanosekunde, a ova funkcija, u kombinaciji sa novom funkcijom kloniranja olakšava skladištenje više verzija datoteke u minimalnom prostoru.

Druge prednosti APFS diskova uključuju fleksibilnu raspodelu prostora, tako da dva APFS diska mogu pozajmiti prostor jedan od drugog kada im je potreban, a ne samo kada su kreirani. Tako APFS gradi podršku za datoteke "sparse files" koje ne popunjavaju čitav prostor na disku koji im je dodeljen.

▼ Poglavlje 6

Pokazne vežbe: SUSE Linux i VBox

INSTALACIJA OS-A

Za realizovanje ove vežbe će biti korišćen softverski paket za virtualizaciju Oracle VirtualBox.

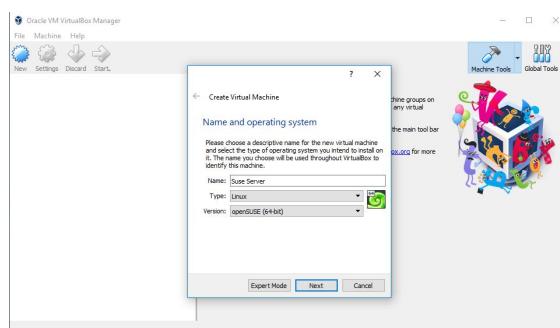
Predviđeno vreme pokaznih vežbi je 60 minuta.

Suse Linux distribucija dolazi u nekoliko verzija, već pripremljenih za odredjene uloge, kao što su Suse Enterprise Edition, Suse ED for SAP, Suse ED for High Performance Computing itd. Ovo su sve komercijalne verzije Suse distribucije, a OpenSuse je jedina besplatna varijanta.

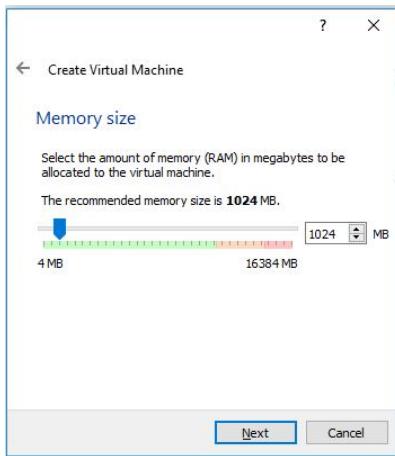
U vreme pisanja ove vežbe, najnovija verzija OpenSuse distribucije je 12.3. OpenSuse se može besplatno preuzeti sa <https://software.opensuse.org/distributions/leap?locale=en>. Dolazi u nekoliko verzija sa različitim desktop okruženjima, i za 32-bitnu i 64-bitnu arhitekturu procesora. Za potrebe ovog rada će biti korišćena 64-bitna verzija sa GNOME okruženjem.

Za početak je potrebno napraviti VM za server. Nova VM se kreira u Oracle VB manager-u klikom na opciju New. Početna podešavanje za VM se postavljaju kroz wizard kreacije.

Na slici 1 i 2 se vide inicijalna podešavanja kao što su: Ime mašine, tip OS-a i veličina RAM memorije.



Slika 6.1.1 Podešavanje imena servera i tipa OS-a [Izvor: Autor]

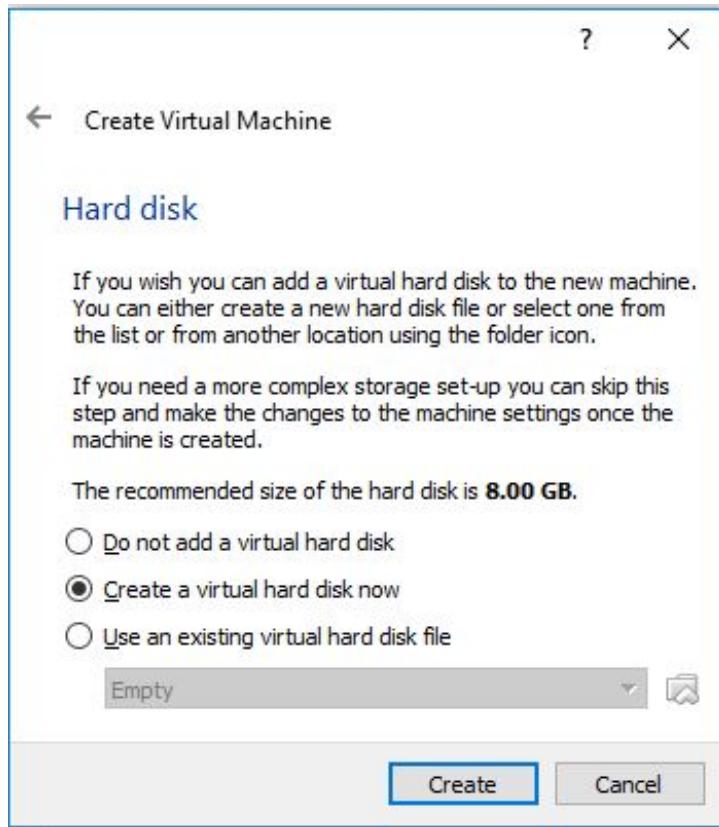


Slika 6.1.2 Podešavanje veličine memorije [Izvor: Autor]

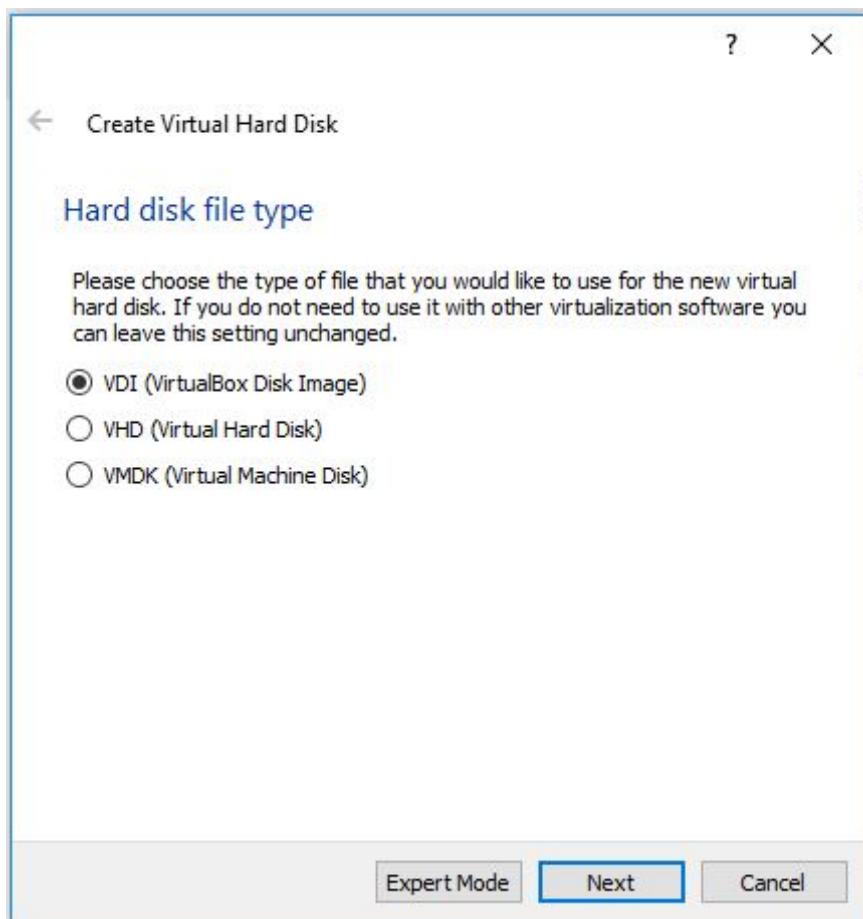
KREIRANJE DISKA I IZBOR FORMATA

Za novu VM je potrebno napraviti novi hard disk.

Na slikama 3 i 4 se vidi izbor opcije za kreiranje novog diska i formata disk slike. U ovom slučaju koristimo VDI format.



Slika 6.1.3 Kreiranje diska [Izvor: Autor]

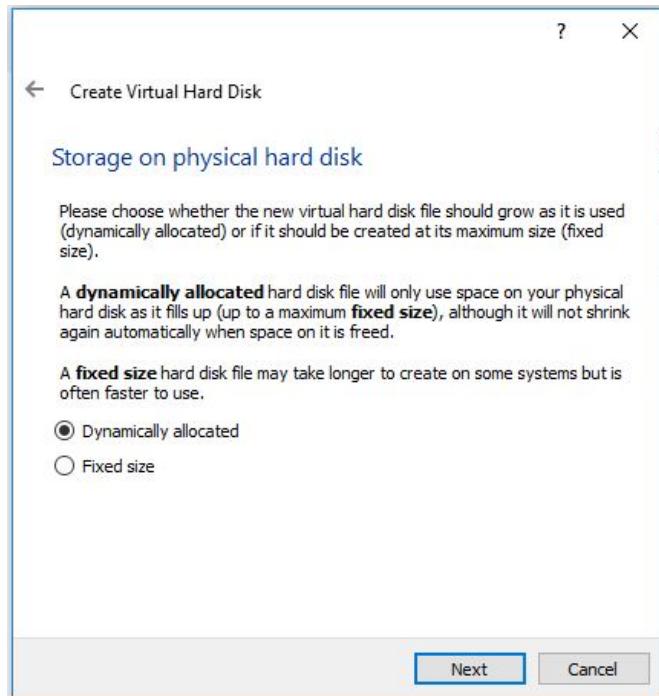


Slika 6.1.4 Izbor formata [Izvor: Autor]

IZBOR ALOKACIJE NA DISKU

Kreiranja mašine završava se izborom načina alokacije prostora na disku.

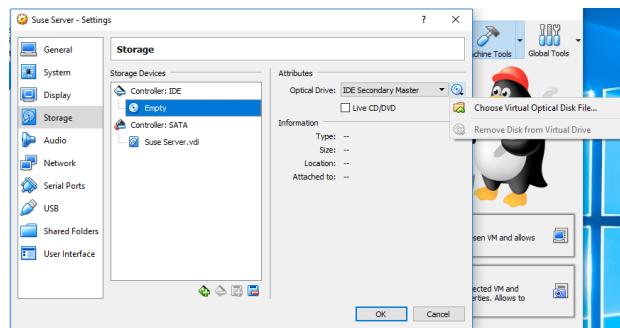
U ovom slučaju koristi se dinamička alokacija prostora. Ova opcija dozvoljava da se disk guest mašine širi po potrebi i zauzima samo onoliko prostora koliko mu je potrebno.



Slika 6.1.5 Izbor alokacije [Izvor: Autor]

Kada je mašina kreirana, potrebno je otvoriti Settings i zatim otvoriti karticu Storage.

Zatim treba mountovati CD image operativnog sistema na čitač VM. Ovaj CD image će se koristiti za bootovanje mašine.

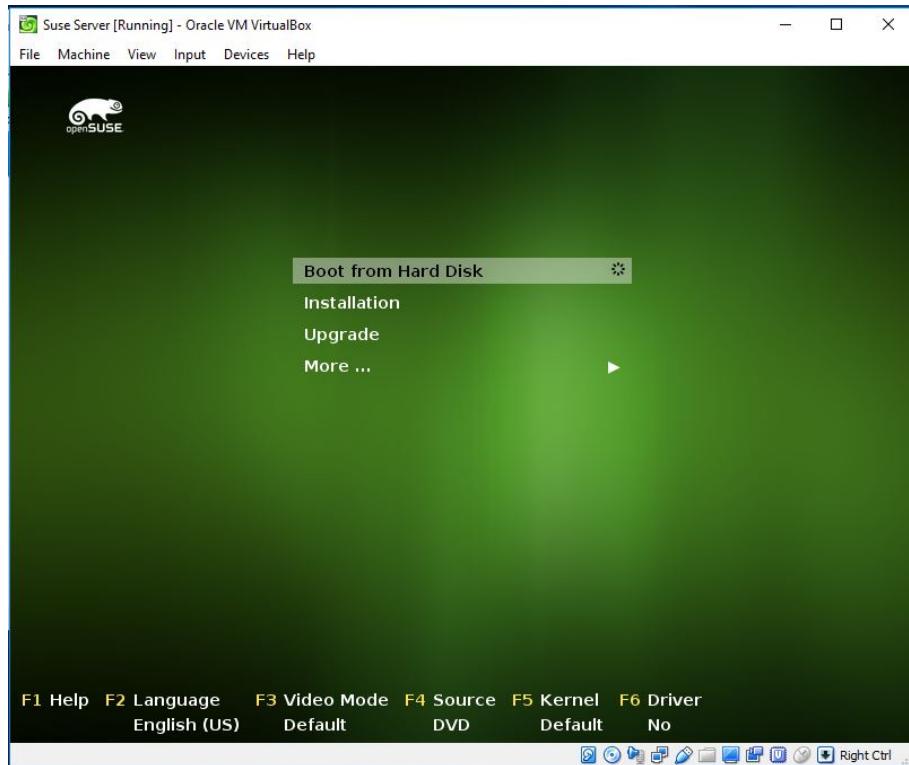


Slika 6.1.6 Izbor alokacije [Izvor: Autor]

PODEŠAVANJA

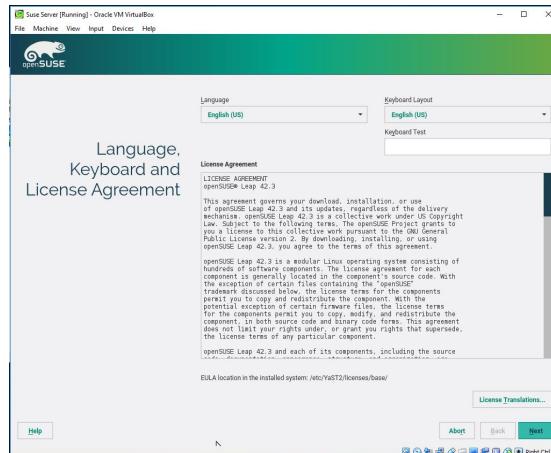
Prilikom boota sa CD-a, prikazuje se glavni meni Suse distribucije.

Za instalaciju bira se Install opcija.



Slika 6.1.7 Početak instalacije [Izvor: Autor]

Potrebno je prihvati licence agreement, a potom podesiti vremensku zonu koja se koristi. Ova podešavanja se vide na slikama 8 i 9 .



Slika 6.1.8 Licence agreement [Izvor: Autor]



Slika 6.1.9 Podešavanje vremenske zone [Izvor: Autor]

PODEŠAVANJE PARTICIJE I KREIRANJE KORISNIKA

Posle kreiranja particija, kreira se jedan non-root korisnik.

Potrebno je podesiti particije diska za instalaciju sistema.

Na slici 10 se vidi podešavanje particija:

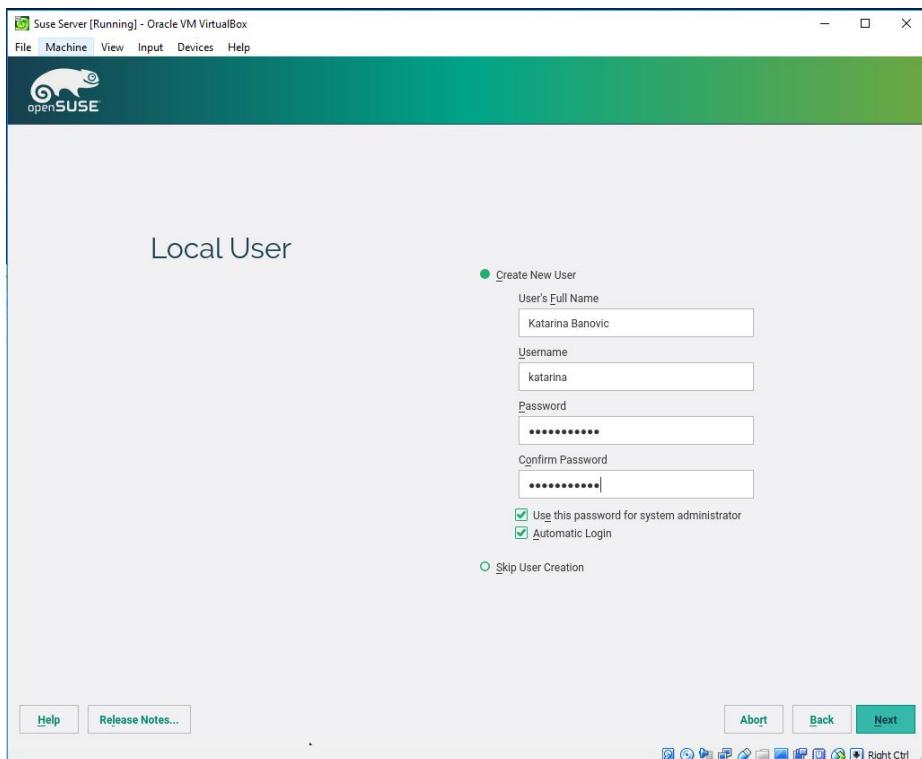
/dev/sda1 je mountovana kao swap particija

/dev/sda2 je mountovana kao root particija, i formatirana u Ext

/dev/sda3 je mountovana kao /home, i formatirana u ext4

Available Storage on linux							
Device	Size	F	Enc	Type	FS Type	Label	Mount Point
/dev/sda	8.00 GB			VBOX-HARDDISK			
/dev/sda1	1.47 GB	F		Linux swap	Swap		swap
/dev/sda2	5.00 GB	F		Linux native	Ext4		/
/dev/sda3	1.52 GB	F		Linux native	Ext4		/home

Slika 6.1.10 Podešavanje particija [Izvor: Autor]



Slika 6.1.11 Kreiranje korisnika [Izvor: Autor]

PARTICIJE I MOUNT POINTI

Swap mount point koristi operativni sistem kao virtualnu memoriju.

Particija je region na disku kojim operativni sistem može upravljati. Svaka particija sadrži posebne metapodatke i fajl sistem.

U linuxu, svaka particija može imati svoj mount point, tj. direktorijum na koji se particija mapira.

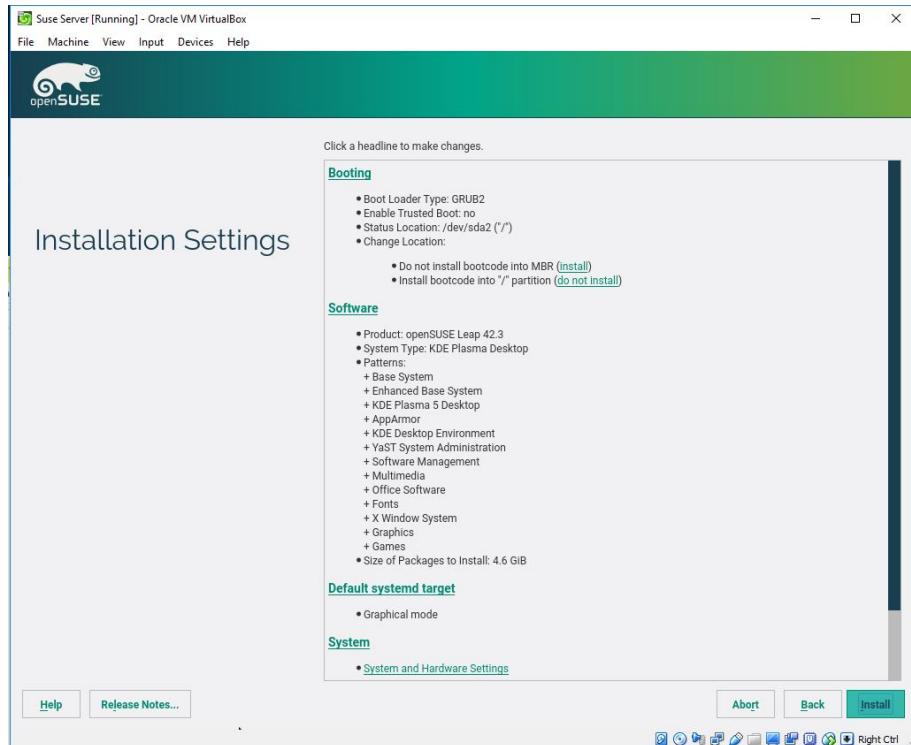
Root mount point odgovara root putanji (/). Direktorijumi /home i /boot su često mount pointi za posebne particije.

Swap mount point koristi operativni sistem kao virtualnu memoriju.

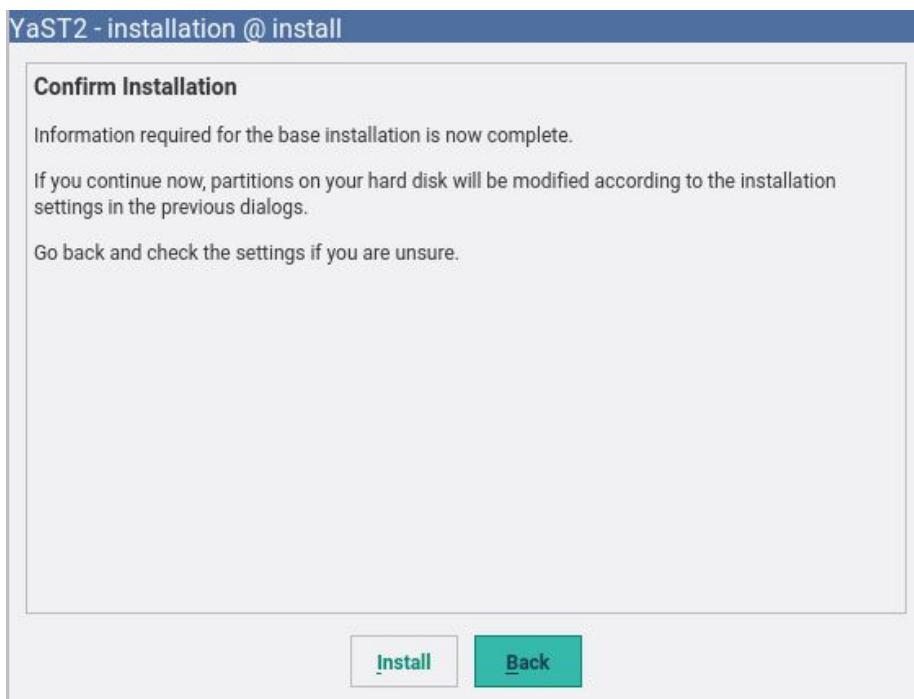
PREGLED PODEŠAVANJA

Posle završenog procesa instalacije, moguće je bootovati linux sa hard diska

Posle pregleda zadatih podešavanja, proces instalacije može početi.



Slika 6.1.12 Pregled podešavanja [Izvor: Autor]



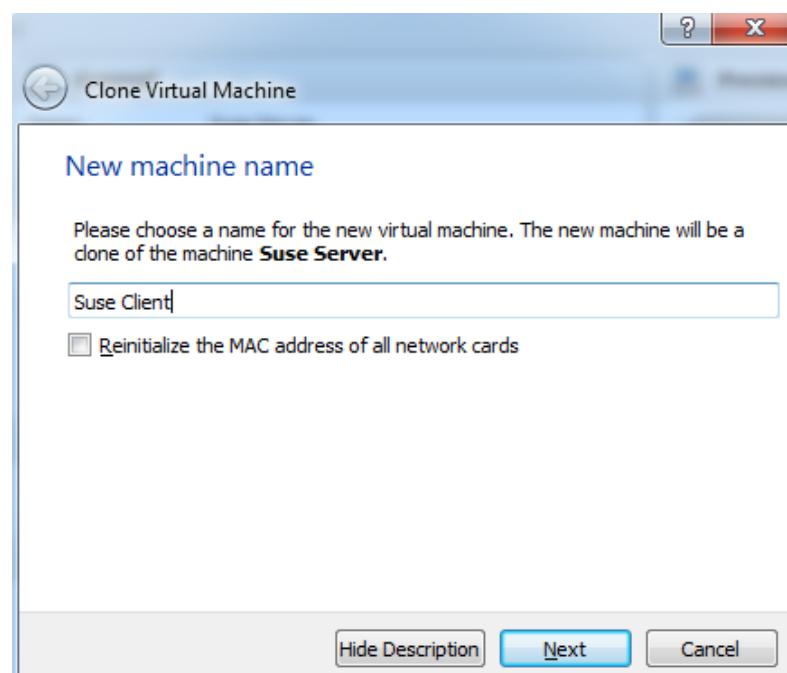
Slika 6.1.13 Potvrda instalacije [Izvor: Autor]

KLONIRANJE MAŠINE

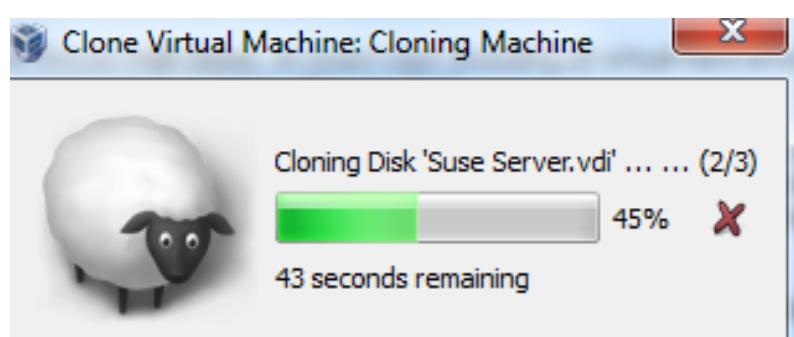
Pošto već imamo jednu instancu Suse Linux instalacije na Vbox mašini, možemo koristiti clone opciju da napravimo mašinu koja će služiti kao DHCP klijent.

Ovim se izbjegava ponavljanje procesa instalacije.

Koristi se full clone opcija, koja kopira i hard diskove mašina.



Slika 6.1.14 Podešavanje klonirane mašine [Izvor: Autor]



Slika 6.1.15 Kloniranje mašine [Izvor: Autor]

✓ 6.1 Pokazne vežbe: DHCP server instalacija

DHCP

DHCP je protokol za dinamičko konfigurisanje uredjaja na mreži

Predviđeno vreme pokaznih vežbi je 15 minuta.

DHCP je protokol za dinamičko konfigurisanje uredjaja na mreži. Pomoću ovog protokola uredjaji mogu dobiti podešavanja za IP adresu, default gateway, DNS server itd. DHCP server je host na mreži na kome je podešen DHCP serverski softver.

Kada se uredjaj sa dhcp klijentom konektuje na mrežu, na broadcast adresu mreže šalje **DHCP discovery** zahtjev. Ovim zahtjevom klijent pita sve uredjaje na mreži da li nude usluge DHCP servera.

Serveri koji dobiju discovery poruku, za klijenta rezervisu IP adresu, i odgovaraju sa **offer** porukom. Ovom porukom klijentu nude rezervisani IP adresu. Ako postoji više DHCP servera na mreži, klijent će dobiti više offer poruka.

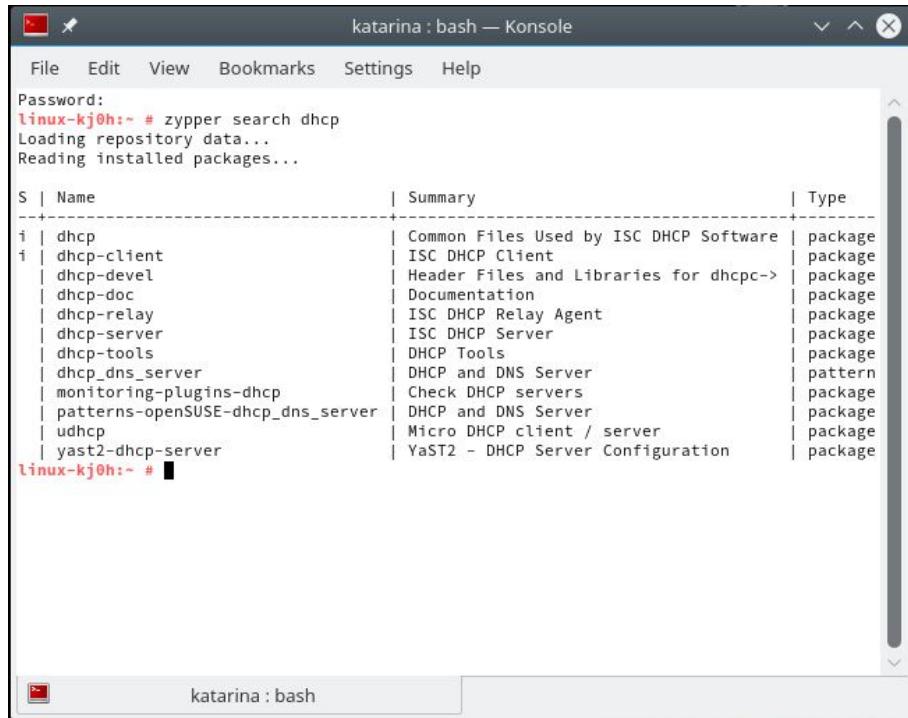
Na offer poruku klijent odgovara sa **request** porukom. Ovom porukom klijent saopštava serveru da je prihvatio ponudjena podešavanja. Klijent će poslati samo jednu request poruku, bez obzira koliko je ponuda dobio.

Na request poruku server će odgovoriti sa DHCP **acknowlege** porukom, kojom klijentu potvrđuje da je zahtjev prihvaćen.

POTREBNI PAKETI ZA INSTALACIJU

Paketi potrebni za ISC DHCP server se nalaze u Suse repozitoriju, i mogu se instalirati uz pomoć alata zypper.

Na slici 1 se vidi pretraga repozitorija za pakete koji imaju dhcp u imenu.



```

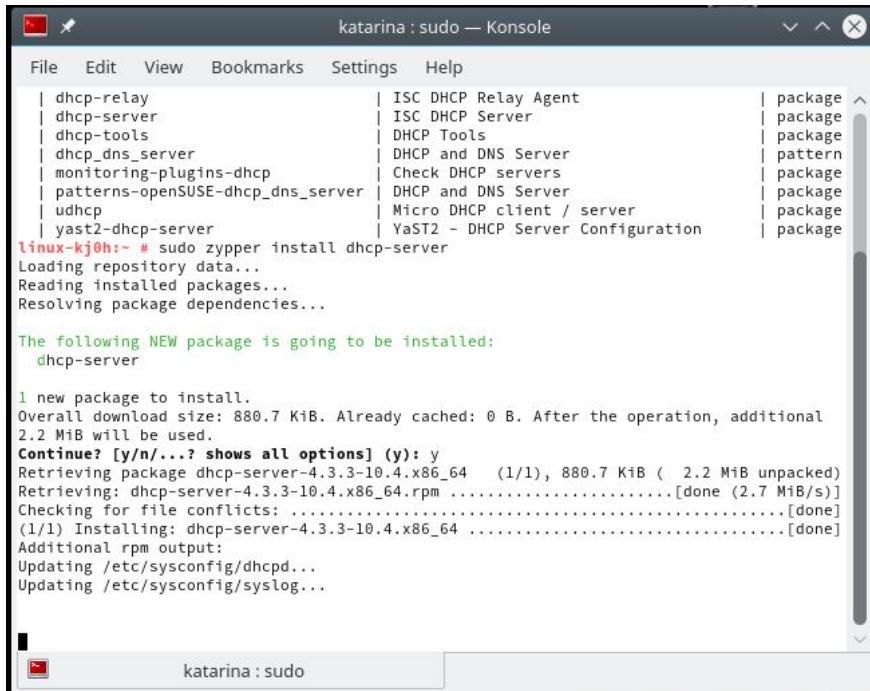
katarina : bash — Konsole
File Edit View Bookmarks Settings Help
Password:
linux-kj0h:~ # zypper search dhcp
Loading repository data...
Reading installed packages...

S | Name | Summary | Type
+---+
i | dhcp | Common Files Used by ISC DHCP Software | package
i | dhcp-client | ISC DHCP Client | package
i | dhcp-devel | Header Files and Libraries for dhcpc-> | package
i | dhcp-doc | Documentation | package
i | dhcp-relay | ISC DHCP Relay Agent | package
i | dhcp-server | ISC DHCP Server | package
i | dhcp-tools | DHCP Tools | package
i | dhcp_dns_server | DHCP and DNS Server | pattern
i | monitoring-plugins-dhcp | Check DHCP servers | package
i | patterns-openSUSE-dhcp_dns_server | DHCP and DNS Server | package
i | udhcp | Micro DHCP client / server | package
i | yast2-dhcp-server | YaST2 - DHCP Server Configuration | package
linux-kj0h:~ #

```

Slika 6.2.1 Pretraživanje repozitorijuma [Izvor: Autor]

Koristi se komanda `zypper install` da se instalira odabrani paket. Package manager će automatski da nađe sve pakete od kojih odabrani paket zavisi i da njih prvo instalira. Naravno, za ovu operaciju potrebno je imati root privilegije.



```

katarina : sudo — Konsole
File Edit View Bookmarks Settings Help
| dhcp-relay | ISC DHCP Relay Agent | package
| dhcp-server | ISC DHCP Server | package
| dhcp-tools | DHCP Tools | package
| dhcp_dns_server | DHCP and DNS Server | pattern
| monitoring-plugins-dhcp | Check DHCP servers | package
| patterns-openSUSE-dhcp_dns_server | DHCP and DNS Server | package
| udhcp | Micro DHCP client / server | package
| yast2-dhcp-server | YaST2 - DHCP Server Configuration | package
linux-kj0h:~ # sudo zypper install dhcp-server
Loading repository data...
Reading installed packages...
Resolving package dependencies...

The following NEW package is going to be installed:
  dhcp-server

1 new package to install.
Overall download size: 880.7 KiB. Already cached: 0 B. After the operation, additional
2.2 MiB will be used.
Continue? [y/n/...? shows all options] (y):
Retrieving package dhcp-server-4.3.3-10.4.x86_64 (1/1), 880.7 KiB ( 2.2 MiB unpacked)
Retrieving: dhcp-server-4.3.3-10.4.x86_64.rpm .....[done (2.7 MiB/s)]
Checking for file conflicts: .....[done]
(1/1) Installing: dhcp-server-4.3.3-10.4.x86_64 .....[done]
Additional rpm output:
Updating /etc/sysconfig/dhcpd...
Updating /etc/sysconfig/syslog...

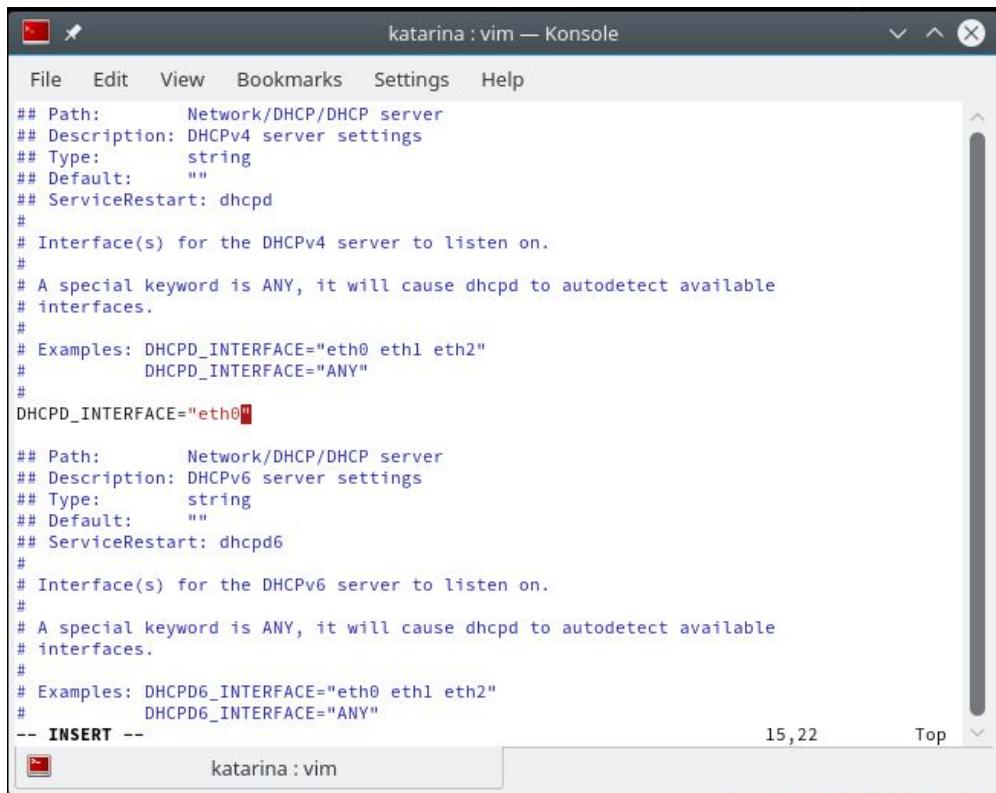
```

Slika 6.2.2 Instalacija DHCP servera [Izvor: Autor]

PODEŠAVANJE INTERFEJSA

Posle instalacije potrebno je konfigurisati DHCP server.

Prvo se postavlja interfejs na kome radi DHCP server u fajlu /etc/sysconfig/dhcpd.



```
File Edit View Bookmarks Settings Help
## Path: Network/DHCP/DHCP server
## Description: DHCPv4 server settings
## Type: string
## Default: ""
## ServiceRestart: dhcpcd
#
# Interface(s) for the DHCPv4 server to listen on.
#
# A special keyword is ANY, it will cause dhcpcd to autodetect available
# interfaces.
#
# Examples: DHCPD_INTERFACE="eth0 eth1 eth2"
#             DHCPD_INTERFACE="ANY"
#
DHCPD_INTERFACE="eth0"

## Path: Network/DHCP/DHCP server
## Description: DHCPv6 server settings
## Type: string
## Default: ""
## ServiceRestart: dhcpcd6
#
# Interface(s) for the DHCPv6 server to listen on.
#
# A special keyword is ANY, it will cause dhcpcd to autodetect available
# interfaces.
#
# Examples: DHCPD6_INTERFACE="eth0 eth1 eth2"
#             DHCPD6_INTERFACE="ANY"
-- INSERT --
katarina : vim
```

Slika 6.2.3 Podešavanje interfejsa [Izvor: Autor]

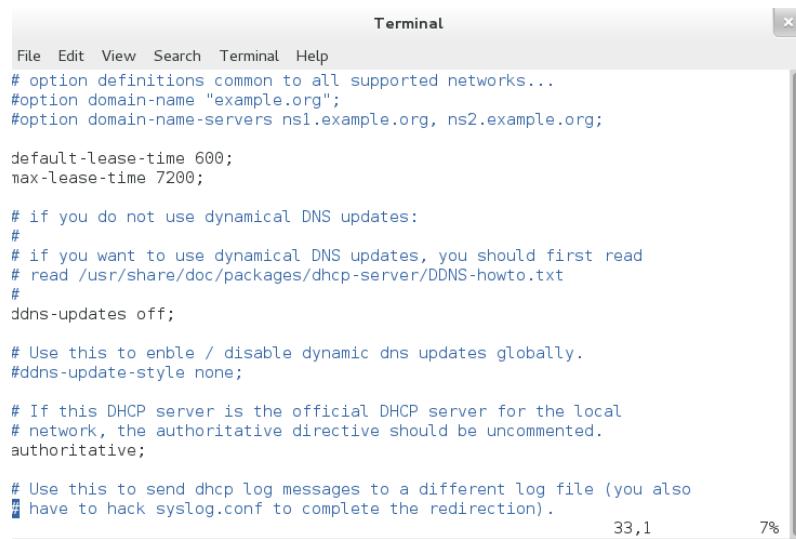
KONFIGURACIONE OPCIJE

Glavni konfiguracioni fajl za dhcp server nalazi se na /etc/dhcpd.conf.

Opcije *default-lease-time* i *max-lease-time* određuju trajanje dhcp lease-a. Defaultno vrijeme se koristi ako klijent posebno ne zatraži neku vrijednost, a ako klijent sam zatraži vrijednost ta vrijednost ne može biti veća od *max-lease-time*.

Opcija *ddns-updates* kontroliše Dynamic DNS funkcionalnost mreže. Opcija *ddns-updates* je isključena, tako da hostovi ne mogu automatski ažurirati dns podešavanja.

Opcija *authoritative* označava da je ovaj server jedini DHCP server na mreži. Ovo znači da on poznaje sve validne IP adrese, i da ako se pojavi klijent sa adresom koja nije validna, ili koju je dobio od nekog drugog servera, ovaj DHCP server će mu tražiti da resetuje *lease*.



```
Terminal
File Edit View Search Terminal Help
# option definitions common to all supported networks...
#option domain-name "example.org";
#option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;
max-lease-time 7200;

# if you do not use dynamical DNS updates:
#
# if you want to use dynamical DNS updates, you should first read
# read /usr/share/doc/packages/dhcp-server/DDNS-howto.txt
#
ddns-updates off;

# Use this to enable / disable dynamic dns updates globally.
#ddns-update-style none;

# If this DHCP server is the official DHCP server for the local
# network, the authoritative directive should be uncommented.
authoritative;

# Use this to send dhcp log messages to a different log file (you also
# have to hack syslog.conf to complete the redirection).
#log-facility local7;
```

Slika 6.2.4 Sama podešavanje mreže se nalaze u istom fajlu [Izvor: Autor]

MREŽNA PODEŠAVANJA

Deklarisan je subnet 10.0.2.0 sa netmaskom 255.255.255.0.

Za ovaj subnet definisan je pool IP adresa od 10.0.2.16 do 10.0.2.255. DHCP klijenti koji se budu prijavili na server će dobiti neku od adresa iz ovog pool-a.

Opcija routers postavlja vrijednost defaultnog gateway-a za mrežu.

Opcija *domain-name-server* postavlja vrijednosti DNS servera.

```
lease {
    interface "eth0";
    fixed-address 10.0.2.17;
    option subnet-mask 255.255.255.0;
    option routers 10.0.2.15;
    option dhcp-lease-time 600;
    option dhcp-message-type 5;
    option domain-name-servers 8.8.8.8;
    option dhcp-server-identifier 10.0.2.15;
    renew 1 2017/12/08 00:04:43;
    rebind 1 2017/12/08 00:08:45;
    expire 1 2017/12/08 00:09:38;
}
```

Slika 6.2.5 Podešavanja [Izvor: Autor]

STARTOVANJE DHCP SERVERA

DHCP server je instaliran kao servis, i startuje se komandom systemctl.

Na slici 6 se vidi startovanje DCHP servera.

Sa klijentske mašine, DHCP klijent se pokreće komandom:

dhcpcd eth0

```
subnet 10.0.2.0 netmask 255.255.255.0
{
    range 10.0.2.16 10.0.2.255;
    option routers 10.0.2.15;
    option domain-name-servers 8.8.8.8;
}
```

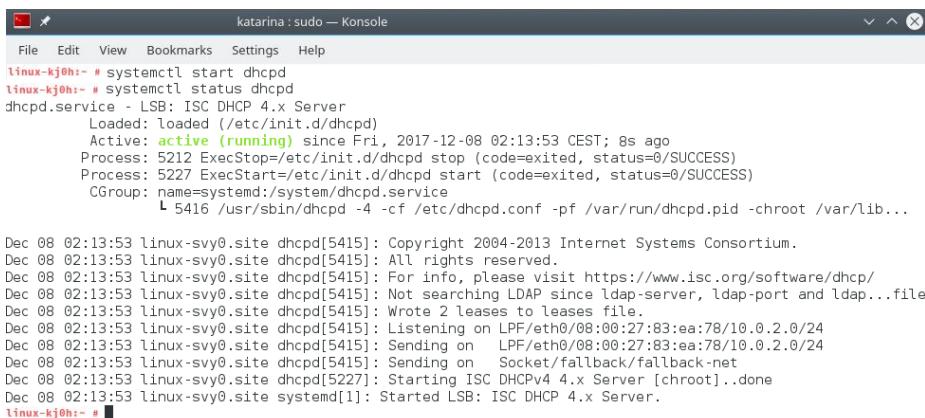
Slika 6.2.6 Startovanje DHCP servera [Izvor: Autor]

Da bi provjerili da server propisno radi, potrebno je pregledati trenutni lease koji klijent ima, i da li ga je uopšte i dobio.

Ovi podaci se mogu naći u fajlu `/var/lib/dhcp/dhclient.leases`.

Na slici 7 se vidi sadržaj ovog fajla. Vidi se da je klijent dobio IP adresu 10.0.2.17, prvu iz poola IP adresa servera. DNS server je podešen na zadatu vrijednost, kao i default gateway. Lease time je postavljen na 600, što je default vrijednost. Znači da klijent nije tražio posebno vrijednost vremena.

Ovim je osnovno podešavanje DHCP servera završeno.



```
File Edit View Bookmarks Settings Help
linux-kj0h:~ # systemctl start dhcpcd
linux-kj0h:~ # systemctl status dhcpcd
dhcpcd.service - LSB: ISC DHCP 4.x Server
   Loaded: loaded (/etc/init.d/dhcpcd)
   Active: active (running) since Fri, 2017-12-08 02:13:53 CEST; 8s ago
     Process: 5212 ExecStop=/etc/init.d/dhcpcd stop (code=exited, status=0/SUCCESS)
     Process: 5227 ExecStart=/etc/init.d/dhcpcd start (code=exited, status=0/SUCCESS)
   CGroup: name=systemd:/system/dhcpcd.service
          └─ 5416 /usr/sbin/dhcpcd -4 -cf /etc/dhcpcd.conf -pf /var/run/dhcpcd.pid -chroot /var/lib...
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Copyright 2004-2013 Internet Systems Consortium.
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: All rights reserved.
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: For info, please visit https://www.isc.org/software/dhcp/
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Not searching LDAP since ldap-server, ldap-port and ldap...file
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Wrote 2 leases to leases file.
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Listening on LPF/eth0/08:00:27:83:ea:78/10.0.2.0/24
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Sending on  LPF/eth0/08:00:27:83:ea:78/10.0.2.0/24
Dec 08 02:13:53 linux-svy0.site dhcpcd[5415]: Sending on  Socket/fallback/fallback-net
Dec 08 02:13:53 linux-svy0.site dhcpcd[5227]: Starting ISC DHCPv4 4.x Server [chroot]..done
Dec 08 02:13:53 linux-svy0.site systemd[1]: Started LSB: ISC DHCP 4.x Server.
linux-kj0h:~ #
```

Slika 6.2.7 Pokrenuti DHCP server [Izvor: Autor]

6.2 Zadaci za samostalni rad: Linux i VBox

OPIS ZADATAKA ZA SAMOSTALNU VEŽBU

Instalirati Ubuntu Linux koristeći virtuelnu mašinu Vbox

Predviđeno vreme izrade zadatka je 60 minuta.

Koristeći uputstvo sa sledećeg linka:

<https://www.ubuntu.com/download/desktop/install-ubuntu-desktop>

instalirati Ubuntu Linux koristeći virtuelnu mašinu Vbox.

▼ Poglavlje 7

Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ01

Naći listu Linux distribucija, dati njihov kratak opis, uslove korišćenja i navesti odgovarajuće web izvore

Predviđeno vreme izrade domaćeg zadatka je 60 minuta.

1. Koristeći Internet pronaći listu Linux distribucija koja se nalazi na adresi <http://lwn.net/Distributions/>. Odabratи četiri distribucije tako da njihovi redni brojevi odgovaraju brojevima vašeg indeksa.

Na primer, ako Vam je broj indeksa **2531** odabraćete redne brojeve **2, 5, 3 i 1**. Za studente koji imaju trocifren broj indeksa za treći broj uzeti sumu brojeva indeksa.Ukoliko su dva broja ista, za drugi uzmite prvi naredni, a ako je jedan od brojeva 0, uzmite 1.

Zadatak pripremiti u obliku standardnog Word (ili OpenOffice) dokumenta koji treba da sadrži naziv Linux distribucije, kratak opis, uslove korišćenja i odgovarajuću Web adresu.

2. Sa sajta <https://linuxmint.com/> preuzeti instalaciju operativnog sistema Linux Mint i instalirati u VirtualBoxu. U Word dokumentu opisati instalaciju sa odgovarajućim slikama.

Dokument snimiti po imenom **IT210-DZ01-Ime_Prezime_brojIndexa**, gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovan dokument poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210 - DZ01)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovoj lekciji smo obradili funkcije i zadatke operativnog sistema. Fokus je dat na sistemskim datotekama u Windows i Unix operativnim sistemima. Date je uporedna analiza Windos sistemskih datoteka NTFS, FAT16, FAT32 i ReFS, u pogledu maksimalne veličine memoriskog prostora, veličine datoteka, volumena i klastera. Takođe, data je uporedna analiza Unix-ovih sistema datoteka Minix FS, Ext FS, Ext2 FS i Xia FS, kao i sama arhitektura njegovog operativnog sistema.

LITERATURA

1. Branislav Đorđević, Dragan Pleskonjić, Nemanja Maček, Operativni sistemi, Mikro knjiga, 2005.
2. Larry Twork, Larry Mead, Bill Howison, JD Hicks, Lew Brodnax, Jim McMicking, Raju Sakthivel, David Holder, Jon Collins, Bill Loeffler, UNIX Application Migration Guide, Chapter 2: UNIX and Windows Compared, msdn.microsoft.com, 2002.
3. William Boswell, Inside Windows Server 2003, Addison Wesley, 2003.
4. Machtelt Garrels, Introduction to Linux, A Hands on Guide, CoreSequence.com, Version 1.4.
5. Steve D. Pate, UNIX Filesystem, Evolution, Design and Implementation, Wiley, 2003.
6. Ririshmy Card, Theodore Ts'o, Stephen Tweedie, Design and Implementation of the Second Extended Filesystem, Proceedings of the First Dutch International Symposium on Linux, ISBN 90-367-0385-9.



IT210 - SISTEMI IT

KONCEPTI I OSNOVE
UPRAVLJANJA INFORMACIJAMA

Lekcija 02

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 02

KONCEPTI I OSNOVE UPRAVLJANJA INFORMACIJAMA

- ✓ KONCEPTI I OSNOVE UPRAVLJANJA INFORMACIJAMA
- ✓ Poglavlje 1: Definicija informacionog sistema
- ✓ Poglavlje 2: Podaci
- ✓ Poglavlje 3: Korišćenje baze podataka za skladištenje podataka
- ✓ Poglavlje 4: Sistemi baza podataka
- ✓ Poglavlje 5: Upravljanje znanjem
- ✓ Poglavlje 6: Pokazne vežbe: Osnove SQL jezika upita
- ✓ Poglavlje 7: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da se razume šta je informacioni sistem i koju vrednost za njega ima upravljanje informacijama

U ovoj lekciji biće reči o svrsi, korišćenju i vrednosti informacionog sistema. Kako bi bolje mogli da razumemo informacione sisteme, potrebno je da savladamo terminologiju koja se koristi u okviru informacionog sistema i njegove baze podataka, a koji se odnosi na razliku između podataka, informacija i znanja. Takođe, biće obrađene teme koje se odnose na upravljanje podacima, životni ciklus podataka, sisteme baza podataka, upravljanje znanjem i sistemima za upravljanje znanjem.

▼ Poglavlje 1

Definicija informacionog sistema

ŠTA JE INFORMACIONI SISTEM?

Informacioni sistem je deo poslovnog sistema čiji je zadatak da prikuplja, obrađuje, prosleđuje, prikazuje i čuva informacije

Vrlo je teško definisati šta je **informacioni sistem** (IS), pre svega što je teško utvrditi granice jednog informacionog sistema neke organizacije. Ipak, može se reći da je:

Informacioni sistem je deo poslovnog sistema čiji je zadatak da prikuplja, obrađuje, prosleđuje, prikazuje i čuva informacije.

Ovakva definicija je dovoljno opšta da obuhvati sve aspekte informacionog sistema. Ono što se iz nje može videti je da su mogući informacioni sistemi i bez primene računara. Računarski podržani informacioni sistemi imaju iste funkcije kao i klasični poslovni informacioni sistemi koji su postojali i pre primene računara. Korišćenjem računara, IS su postali efikasniji i pouzdaniji.

Osnovne definicije

Informacioni sistemi su zasnovani na nekim osnovnim konceptima kao što su podaci, informacije, znanje, metapodaci i baze podataka, pa će se ovde dati njihove definicije.

Podaci su sirove činjenice, simboli i brojevi. Van konteksta ili bez dodatnog objašnjenja podaci ne govore ništa. Sa aspekta računarstva se može reći da je podatak vrednost nekog atributa posmatranog objekta. Podaci su neophodna osnova za procese proračuna, analize i rezonovanja.

Informacije su potrebni podaci čije je značenje poznato. Podaci postaju informacije kada se prezentuju u nekom kontekstu na način koji je prikladan specifičnoj situaciji.

Znanje je sposobnost da se informacije iskoriste da bi se sa njima nešto postiglo.

Metapodaci su podaci o karakteristikama podataka.

✓ 1.1 Svrha i korišćenje informacionog sistema

OSNOVNE FUNKCIJE INFORMACIONOG SISTEMA

Osnovne funkcije informacionog sistema su: unos podataka, obrada podataka, generisanje informacija, dokumenata i izveštaja, čuvanje podataka, pretraživanje i prenos podataka

Informacioni sistemi služi za pravilno i plansko prikupljanje, obradu, čuvanje i prenos podataka kako bi se poslovnom sistemu prezentovale tražene informacije. Svaki informacioni sistem se zasniva na podacima i funkcijama koje obezbeđuje informacioni sistem. Informacioni sistemi podatke čuvaju uglavnom u bazama podataka.

Neki informacioni sistemi imaju specifične funkcije, ali se generalno može reći da informacioni sistemi imaju 6 osnovnih funkcija:

1. Unos podataka (pribavljanje podataka)
2. Obrada podataka
3. Generisanje informacija, dokumenata i izveštaja
4. Čuvanje podataka
5. Pretraživanje podataka
6. Prenos podataka

Informacioni sistemi se koriste u svim oblastima poslovanja. Ovde će se navesti samo neki informacioni sistemi i njihova svrha:

- Državni, odnosno vladini informacioni sistemi imaju zadatku da pruži pomoć državnim institucijama u upravljanju državom i njenim entitetima.
- Zdravstveni informacioni sistemi se koriste u zdravstvenim organizacijama kao podrška u procesu dijagnostikovanja, određivanja terapije, lečenja i praćenja pacijenata, kao i za upravljanje resursima zdravstvenih organizacija.
- Geografski informacioni sistemi se koriste za planiranje i upravljanje prostornim i infrastrukturnim resursima.
- Poslovni informacioni sistemi čuvaju informacije o resursima organizacije i njenim aktivnostima (transakcijama). Na osnovu ovih informacija se automatizuju pojedini poslovni procesi i donose se odluke potrebne za upravljanje organizacijom.
- Lični informacioni sistemi čuvaju informacije o kontaktima, događajima, obavezama i pomažu korisnicima da organizuju svoj privatni i poslovni život.

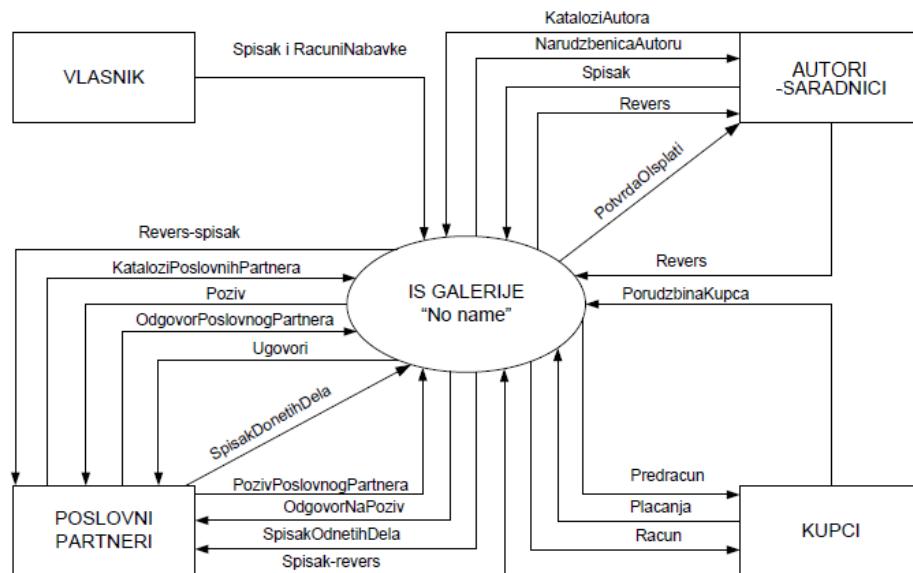
PRIMER: GLOBALNI PRIKAZ IS-A GALERIJE SLIKA

Globalni prikaz IS-a galerije prikazuje interakciju sistema galerije slika sa spoljašnjim okruženjem

Sa dijagrama se vidi da je sistem galerije u interakciji sa četiri spoljna sistema: VLASNIK, AUTORI_SARADNICI, POSLOVNI PARTNERI i KUPCI. Strelice koje ulaze u sistem predstavljaju njegove ulazne podatka i imenovane su tako da odslikavaju značenje i strukturu svakog pojedinačnog toka podataka. Ulazni podaci se u okviru informacionog sistema galerije obrađuju i na osnovu njih se generišu odgovarajući tokovi iz sistema koji se usmeravaju prema pojedinim spoljašnjim sistemima.

Pitanje: Koje podatke treba sačuvati u sistemu da bi se kao izlaz dobio Predračun za Kupca?

Rešenje: Porudžbina Kupca



Slika 1.1.1 Globalni prikaz IS-a galerije slika [Izvor: Autor]

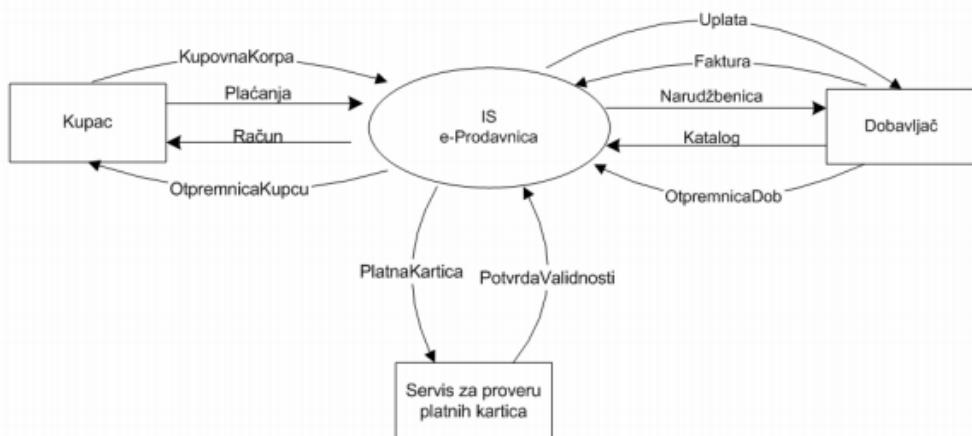
PRIMER: GLOBALNI PRIKAZ IS-A ELEKTRONSKE PRODAVNICE

Globalni prikaz IS-a elektronske prodavnice prikazuje interakciju sistema elektronske prodavnice sa spoljašnjim okruženjem

Na slici 2. je prikazan dijagram IS e-prodavnice. Ovaj IS je u interakciji sa Kupcem, Dobavljačem i Servisom za proveru platnih kartica. Kupac kupljene proizvode sa elektronske prodavnice (koja ima ulogu prodavca) stavlja u Kupovnu Kopru na osnovu čega e-Prodavnica šalje Račun za plaćanje. Kada Kupac plati kupljene proizvode (Plaćanje) roba se šalje Kupcu sa Otprimnicom.

E-prodavnica naručuje robu od Dobavljača na sličan način s tim što u ovom slučaju e-prodavnica ima ulogu kupca a Dobavljač ulogu prodavca. Interakcija sa Servisom za proveru platnih kartica se obavlja u dva smera: e-prodavnica šalje Platnu karticu na proveru a Servis šalje potvrdu da je Platna kartica validna.

Pitanje: Koje podatke treba sačuvati u IS-u da bi e-Prodavnica mogla da izvrši Uplatu prema Dobavljaču?



Slika 1.1.2 Globalni prikaz IS-a elektronske prodavnice [Izvor: Autor]

✓ 1.2 Vrednost informacionih sistema

KADA JE INFORMACIONI SISTEM VREDAN?

Informacioni sistem koji može da pruži odgovore na više pitanja i zadovolji više potreba svojih korisnika smatra se vrednjim

Vrednost jednog informacionog sistema zavisi od količine i kvaliteta podataka koje sadrži i od kvaliteta i vrste funkcija koje on obezbeđuje. Informacioni sistem koji može da pruži odgovore na više pitanja i zadovolji više potreba svojih korisnika smatra se vrednjim. Praksa je pokazala da organizacije koje imaju efikasan informacioni sistem imaju veće šanse za opstanak i dalji razvoj.

Efikasno korišćenje informacionih sistema može organizacijama da pomogne na različite načine, kao što su:

- Pružanje boljih i kvalitetnijih usluga klijentima organizacije
- Projektovanje kvalitetnijih proizvoda
- Organizovanje efikasnije proizvodnje
- Donošenje odluka za upravljanje organizacijom na osnovu pravovremenih i pouzdanih informacija
- Smanjenje grešaka u sprovođenju poslovnih aktivnosti
- Povećanje produktivnosti
- Poboljšanje informisanosti zaposlenih, klijenata i javnosti
- Smanjenje potrebe za ljudskim radom
- Smanjenje troškova poslovanja
- Efikasno praćenje i upravljanje finansijskim resursima
- Efikasno praćenje i upravljanje organizacijskim resursima.

✓ Poglavlje 2

Podaci

KARAKTERISTIKE PODATAKA

Osnovne karakteristike podataka su tačnost, kompletност, pouzdanost, relevantnost, blagovremenost i ekonomičnost

Da bi informacije bile vredne za korisnike, **podaci** na kojima se one zasnivaju treba da zadovoljavaju niz karakteristika:

- Tačnost
- Kompletnost
- Pouzdanost
- Relevantnost
- Blagovremenost
- Ekonomičnost

KARAKTERISTIKE PODATAKA – TAČNOST, KOMPLETNOST, POUZDANOST, RELEVANTNOST

Tačne informacije su bez grešaka, kompletne sadrže sve potrebne podatke, pouzdane zavise od metode prikupljanja podataka i pouzdanosti izvora, a relevantne su vredne za korisnika

- **Tačnost** - Tačne informacije su bez grešaka. Pogrešne informacije se mogu dobiti ako su podaci koji su obrađivani pogrešni ili ako se podacima dodeli pogrešan smisao. Na primer, ako se podatak *Broj godina* odnosio na broj godina radnog staža, a ne na godine starosti, onda su dobijene informacije netačne.
- **Kompletност** - Kompletna informacija sadrži sve potrebne podatke. Vrednost informacije vrlo brzo opada ako neki podaci nedostaju, a u većini slučajeva informacija postaje bezvredna. Na primer, informacija o poletanju aviona je bezvredna ako se zna samo vreme poletanja, a ne zna se sa kog izlaza. Kompletne informacije se mogu dobiti samo na osnovu svih potrebnih relevantnih podataka.
- **Pouzdanost** - Pouzdanost informacija se zasniva na pouzdanosti podataka. Pouzdanost podataka zavisi od metode prikupljanja podataka i pouzdanosti izvora. Ukoliko postoji sumnja u pouzdanost izvora podataka, potrebno je obezbediti alternativne izvore radi verifikacije podataka.

- **Relevantnost** - Informacija treba da bude relevantna za korisnika, jer u suprotnom ona postaje bezvredna. Na primer, putniku za London je irelevantno gde je ulaz za putnike za Pariz. Irelevantne informacije opterećuju korisnika i mogu da izazovu zabunu.

KARAKTERISTIKE PODATAKA – BLAGOVREMENOST, EKONOMIČNOST, PERCEPTIVNOST, AŽURNOST

Blagovremene informacije su relevantne informacije koje su dostupne na vreme, ekonomične koje ne opterećuju resurse prilikom obrade, perceptivne koje su prilagođene potrebama korisnik

- **Blagovremenost** - Informacije treba da budu dostupne korisniku u trenutku kad su mu potrebne. Informacije koje kasne postaju bezvredne, a mogu i da nanesu štetu. Na primer, ako kasno saznamo kada avion treba da poleti, odnosno ako je on već odleteo, ova informacija nije blagovremena zato što smo propustili let. Da bi informacije bile dostupne na vreme, potreбно je unapred obezbediti relevantne podatke. Osim toga, vreme potrebno za obradu podataka ne treba da ugrozi ažurnost (koga interesuje prognoza vremena za juče).
- **Ekonomičnost** - U nekim slučajevima obrada podataka u cilju dobijanja informacija može da zahteva skupe računarske resurse ili da zahteva jako veliko vreme za obradu. U takvim slučajevima potreбno je odmeriti da li vrednost informacija opravdava potrebna finansijska ulaganja.
- **Perceptivnost** - informacije treba da budu prilagođene percepcionim osobinama korisnika. Korisnici mogu da koriste različite organe za prijem informacija koje takođe mogu da budu u različitim obliku. Tako su, na primer, odštampane informacije je bezvredne za slepe korisnike. Korisniku koji vidi fotografija neke osobe je mnogo vrednija informacija nego njen opis (crna kosa, plave oči, isturene jagodice). Takođe će fotografija u boji predstavljati vredniju informaciju nego crno-bela fotografija.
- **Ažurnost** - Postoje dve vrste podataka: statički i dinamički. Statički podaci se ne menjaju ili se menjaju vrlo retko, pa ne postoji potreba za njihovim stalnim ažuriranjem. Na primer naziv jedne organizacije je podatak koji se verovatno neće promeniti u toku životnog veka informacionog sistema. Dinamički podaci se menjaju tokom vremena. Primeri dinamičkih podataka mogu da budu: stanje na bankovnom računu organizacije, broj komada nekog proizvoda u magacinu, nivo goriva u rezervoaru. Dinamički podaci se moraju ažurirati često jer od toga zavisi tačnost informacija. Idealno ažuriranje podrazumeva da se izvrši promena vrednosti podataka onda kada je nastala promena.

✓ 2.1 Izazovi u upravljanju podacima

KOJI IZAZOVI POSTOJE PRILIKOM UPRAVLJANJA PODACIMA?

Podaci su obično rasuti širom cele organizacije i njih prikupljaju mnogi pojedinci koji koriste različite metode i uređaje za skupljanje tih podataka

Podaci se obrađuju u nekoliko faza, a često i na nekoliko različitih mesta. To može predstavljati problem i veliki izazov. Upravljanje podacima u organizacijama je teško iz više razloga. Količina podataka koju treba čuvati se vremenom povećava eksponencijalno. Mnogo starih podataka o poslovanju treba da se čuva duže vreme, pri čemu se novi podaci brzo dodaju. Na primer, kako bi podržali oko 40 miliona korisnika koji igraju "fantasy football", veb lokacije kao što su ESPN.com, NFL.com i CBSSportsLine.com moraju da upravljaju terabajtom sportskih podataka.

Podaci su obično rasuti širom cele organizacije i njih prikupljaju mnogi pojedinci koji koriste različite metode i uređaje za skupljanje tih podataka. Podaci se često čuvaju na raznim serverima, različitim lokacijama i na različitim računarskim sistemima. Podaci dolaze iz internih izvora (npr. korporativne baze podataka), ličnih izvora (npr. lične misli, mišljenja i iskustva) i eksternih izvora (npr. komercijalne baze podataka, vladini izveštaji i korporativni veb sajтови).

Podatke takođe možemo prikupiti sa veba, u obliku "clickstream" podataka. "Clickstream" podaci su oni podaci koje posetioci i kupci proizvode kada posete neku veb lokaciju i kliknu na hipervezu (engl. [hyperlink](#)). "Clickstream" podaci pružaju mogućnosti praćenja aktivnosti korisnika na vebu, uključujući i njihovo ponašanje i česta pretraživanja.

Novi izvori podataka kao što su blogovi, podcast-ovi, videocast-ovi i RFID tagovi i drugi bežični senzori se konstantno razvijaju. Dobar deo ovih podataka nisu struktuirani. Primeri nestruktuiranih podataka su digitalne slike, digitalni video, glasovni paketi i muzičke note u MP3 fajlu.

Sigurnost, kvalitet i integritet podataka su veoma važni, ali su oni lako ugroženi. Pored toga, pravni zahtevi koji se odnose na podatke, razlikuju se među zemljama i različitim oblastima privrede, a pri tom se oni često menjaju. Zbog ovih problema, podacima je teško upravljati. Kao rezultat toga, organizacije koriste baze podataka (engl. [database](#)) i skladišta podataka (engl. [data warehouses](#)) za efikasnije i efektivnije upravljaju svojim podacima.

TOP 6 IZAZOVA U UPRAVLJANJU PODACIMA

Pet glavnih izazova u upravljanju informacijama su: automatizacija, količina podataka, promena formata i revizija, analiza i integracija

Kako bi firma i njeni nadležni mogli da donesu pravovremene poslovne odluke, potrebno je da imaju adekvatnu podršku u donošenju odluka. U današnje vreme, firme troše puno

novca kako bi imali ovu adekvatnu podršku. Na isti način na koji kvaliteta goriva utiče na performanse motora automobila, ponuda i kvalitet podataka direktno utiče na odlučivanja neke kompanije i prevashodno utiče na ishod njenog poslovanja. Dakle, u cilju optimizacije donošenja odluka, odnosno takozvane inteligencije koja se izvodi iz sakupljenih podataka, ovde navodimo pet izazova na koje kompanije trebaju da obrate pažnju:

1. **Automatizacija** - Podaci mogu da poteknu od mnogo različitih izvora: interno u okviru organizacije ili spolja od proizvođača podataka. Zbog različitih izvora se stvaraju različiti skupovi podataka, koji imaju svoja različita pravila i standarde koji su važili tokom njihovog prikupljanja, uključujući različite formate, prihvativije opsege, i granularnosti (vreme publikovanja podataka i njihova relevantnost). Rukovanje sa više različitih skupova podataka sa različitim standardima postao je glavni izazov. Ovaj izazov postaje još složeniji kada su uzme u obzir i vremenski aspekt, pa analiza podataka ne bude dostavljena na vreme, naročito u situacijama kada je vremenski odaziv u donošenju odluka kritičan. Iz tih razloga, automatizacija podataka je od suštinskog značaja, jer omogućava da se uhvaćeno podaci pravovremeno analiziraju koristeći minimalne resurse.
2. **Količina podataka** - Što više podataka se prikup, potrebno je više nadgledanje i validacije. Bez sofisticiranih i inteligentnih sistema za upravljanje podacima, prikupljeni podaci neće moći da se iskoriste i dobijaće se samo fragmentirane i nepouzdane informacije.
3. **Promena formata i revizija** - Jedan od glavnih izazova u prikupljanju podataka je nedoslednost. Kako firme razviju i transformišu aspekte svog poslovanja, one imaju tendenciju da promene i formate svojih podataka ili prestanu sa povremenim prikupljanjem podataka za potrebe održavanja ili iz nekih drugih razloga.
4. **Analiza** - Znanje koje se može izvesti iz analize koja koristi netačne ili nekonistentne podatke može biti pogubno za analitičare, menadžerei druga nadležna lica u okviru organizacije. Sistemi koji mogu stvoriti smislene analize iz kompleksnih podataka mogu uštedeti organizacijama dosta vremena, novca, stresa i neizvesnosti, omogućavajući im da iskoriste prednosti pouzdanih podataka.
5. **Integracija** - Organizacije se suočavaju sa izazovom kada svi prikupljeni podaci iz različitih izvora moraju da ispunjavaju sve interne sistemske zahteve u toj organizaciji. Krajnji cilj je da se prikupljeni podaci integrišu u sve relevantne poslovne sisteme organizacije.
6. **Infomatizacija** - Prenos podataka u digitalni format kako bi mogla da se obavi digitalna transformacija preduzeća

INFORMATIZACIJA, DIGITALIZACIJA, DIGITALNA TRANSFORMACIJA

Digitalna transformacija je proces integracije digitalnih tehnologija u sve oblasti poslovanja, uz radikalne promene u načinu korišćenja tehnologije, ljudi i poslovnih procesa, radi unapređenja

Informatizacija je proces stvaranja digitalnih verzija analognih ili fizičkih stvari poput dokumenata, fotografija, snimaka, zvukova i dr .

Digitalizacija predstavlja sledeći korak, proces u kome se informacije (koje su, posle digitizacije, već u digitalnom obliku) objedinjuju tako da se povežu i lakše koriste, radi pojednostavljenja ili ubrzanja nekih operacija koje su se ranije radile ručno. Praktično, isto ono što se radilo na stari, analogni, „papirni“ način, digitalizacijom je omogućeno da se radi na digitalni način. Nije menjana suština, samo je promenjen oblik, uz brojne prednosti koje računari i umreženost sami po sebi donose, poput automatizacije koja u mnogo čemu zamenjuje ljudski rad. Zahvaljujući digitizaciji i digitalizaciji, informacije su postale lako dostupne za upotrebu na različitim platformama, uređajima i interfejsima. I to je digitalni svet u kome danas živimo.

Digitalna transformacija je ono što se dešava tek nakon digitalizacije. Digitalna transformacija je proces integracije digitalnih tehnologija u sve oblasti poslovanja, uz radikalne promene u načinu korišćenja tehnologije, ljudi i poslovnih procesa, radi unapređivanja korisničkih iskustava u skladu sa stalnim promenama na tržištu. Digitalna transformacija u poslovanju ustvari čini integraciju digitalnih tehnologija u sve sfere poslovanja koja donose promene i poboljšanje u istom.

Poslodavci traže nove načine za poboljšanje produktivnosti, pri čemu digitalna tehnologija igra ključnu ulogu u pomaganju zaposlenima da postanu efikasniji u svojim primarnim radnim zadacima. Digitalna transformacija pruža dragocenu priliku osnovnim poslovnim funkcijama, poput finansija i ljudskih resursa, da se odvoje od manuelnih procesa i automatizuju ključna područja, poput platnih spiskova, omogućavajući liderima da se fokusiraju na širu poslovnu sliku.

Digitalnu transformaciju ili **DX** definišemo kao integraciju digitalnih tehnologija u sva područja poslovanja koja rezultira fundamentalnim promenama u poslovanju i isporučivanju vrednosti korisnicima. Dakle, korišćenje digitalnih tehnologija koje radikalno transformišu model poslovanja, generišu nove tokove prihoda i kompletno menjaju poslovne procese.

✓ 2.2 Životni ciklus podataka

KOJI JE ŽIVOTNI CIKLUS PODATAKA U PREDUZEĆU?

Ciklus: prikupljanje podataka, smeštanje podataka u bazu podataka, obrada i formatiranje podataka iz baze, analiza podataka, generisanje znanja, predstavljanje znanje i njegovo čuvanje

Preduzeće koristi obrađene podatke, koji su obrađeni u informacije i znanje. Menadžeri ove informacije i znanja primenjuju u rešavanju poslovnih problema. Preduzeće može da transformiše podatke u znanje i rešenja na više načina. Kao prvo, potrebno je **prikupiti podatke iz različitih izvora i smestiti te podatke u baze podataka. Odabrani podaci iz baze podataka organizacije se zatim obrađuju i formatiraju za skladište podataka. Zatim korisnici mogu da pristupe podacima u skladištu podataka kako bi ih analizirali.**

Analiza se vrši sa alatima za analizu podataka. Ove aktivnosti na kraju generišu znanje koje se može koristiti za podršku u poslovnom odlučivanju. I podaci (u raznim vremenskim intervalima tokom procesa) i znanja (izvedena na kraju procesa) moraju biti predstavljena korisnicima. Ova prezentacija se može postići korišćenjem različitih alata za vizualizaciju.

Oformljeno znanje se može čuvati u bazi znanja organizacije i može se koristiti zajedno sa alatima za podršku u odlučivanju i donošenju rešenja za organizacione probleme.

UPRAVLJANJE ŽIVOTNIM CIKLUSOM PODATKA

Upravljanje životnim ciklusom podataka je usluga koja treba da obezbedi poverljivost, integritet i dostupnost prikupljenih podataka

Upravljanje životnim ciklusom podataka (Data Lifecycle Management - DLM) je usluga koja treba da obezbedi **poverljivost, integritet i dostupnost** prikupljenih podataka, koje organizacija koristi ili skladišti, tokom "korisnog života" tih podataka. Ova usluga uključuje i sigurno uklanjanje tih podataka na kraju životnog ciklusa tih podataka, kako bi se sprečilo njihovo neovlašćeno korišćenje, što može dovesti do prekida u poslovnim aktivnostima.



Slika 2.1.1 Faze u upravljanju životnim ciklusom podataka

[Izvor: <https://www.dsdlabs.com/wp-content/uploads/2017/03/Data-Lifecycle-Management.jpg>]

Upravljanje životnim ciklusom podataka sadrži sledeće faze:

- **Planiranje** (engl. **Plan**) - kreiranje plana kako će se rukovati podacima u organizaciji
- **Sakupljanje** (engl. **Collect**) - prikupljanje podataka iz različitih izvora i njihova organizacija u neku bazu podataka
- **Osiguranje** (engl. **Assure**) - određivanje standarda i formata u kojima će se podaci čuvati, kako bi se obezbedio njihov integritet za kasnije korišćenje
- **Čuvanje** (engl. **Preserve**) - čuvanje podataka u repozitorijumu zajedno sa njegovim metapodacima
- **Otkrivanje** (engl. **Discover**) - omogućiti pretragu metapodataka kroz korisnički interfejs i korišćenje tog interfejsa za pronalaženje relevantnih podataka
- **Integriranje** (engl. **Integrate**) - transformacija različitih skupova podataka (različiti formati, ontologije, šeme kodiranja) u jedinstveni skup sa istom reprezentacijom podataka
- **Analiza** (engl. **Analyze**) - primena statističkih i analitičkih modela, kakobi se došlo do znanja
- **Arhiviranje** (engl. **Archive**) - obezbeđenje i čuvanje podataka, kako bi se podaci čuvali na bezbednom mestu i omogućili njihovo korišćenje u nekom kasnijem periodu, a i dalje pružali njihov maksimalni integritet.

PRIMER BRIDGEHEAD SOFTWARE KOMPANIJE

BridgeHead Software - Protected Data Lifecycle Management

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 3

Korišćenje baze podataka za skladištenje podataka

UPOTREBA BAZA PODATAKA I MOGUĆI PROBLEMI

Mogući problemi kod kreiranja i ažuriranja baze podataka su redundantnost podataka, izolacija podataka i nekonzistentnost podataka

Korišćenje baze podataka može da eliminiše mnoge probleme koji se mogu pojaviti prilikom upravljanja podacima. Baze podataka su raspoređene tako da je jedan skup softverskih programa - sistem za upravljanje bazama podataka (engl. **the database management system**) - omogućava svim korisnicima da pristupe svim podacima . Ovaj sistem smanjuje sledeće probleme:

- Redundantnost podataka: redundantnost podataka predstavlja slučaj kada se isti podatak čuva na više mesta u jednoj bazi.
- Izolacija podataka: nastaje kada aplikacija ne može da pristupi podacima koje koristi druga aplikacija.
- Nekonzistentnost podataka: Nastaje kada se različiti primeri podataka ne slažu.

Takođe, prilikom upotrebe baze podataka mora se voditi računa o sledećim pitanjima:

- Bezbednost podataka
- Integritet podataka: Podaci nekada moraju da zadovolje određena ograničenja kao što su nepostojanje slova u JMBG broju i određena dužina JBMG.
- Nezavisnost podataka: Aplikacija i podaci treba da budu nezavisni jedan od drugog

✓ 3.1 Hijerarhija podataka

HIJERARHIJSKA POVEZANOST ZAPISA I TABELA SA BAZOM PODATAKA

Zapis predstavlja logičko grupisanje srodnih podataka, tabela predstavlja logičko grupisane srodne zapise, a skup tabel čine bazu podataka

Podaci su organizovani hijerarhijski. Najniži nivo u hijerarhiji predstavlja bit, a najviši nivo cela baza podataka. Bit (binary digit) predstavlja najmanju komponentu u celom računarskom sistemu. Termin "binarno" predstavlja činjenicu da bit može da ima dve vrednosti 0 ili 1. Grupa od 8 bitova predstavlja jedan bajt, što ustvari predstavlja jedan karakter. Bajt može biti slovo, broj ili simbol. Logički grupisani karakteri u reč, mala grupa reči ili identifikacioni broj se nazivaju poljem. Na primer, ime studenta u univerzitetkoj arhivi bi se pojavilo u polju "ime", a JMBG bi se pojavio u polju "JMBG." Polja mogu da sadrže podatke koji nisu isključivo slova i brojevi. Polje može da sadrši sliku ili bilo koji drugi tip multimedija.

Na primer, baza podataka ustanove koja registruje motorna vozila može da sadrži i fotografije vlasnika vozila. Logičko grupisanje srodnih podataka, kao što su na primer imena vlasnika vozila, datum, klasa vozila i broj šasije čine jedan zapis (engl. record). Logičko grupisanje srodnih zapisa naziva se datoteka (engl. file) ili tabela (engl. table). Na primer, zapis određenog predmeta koji se sastoji od šifre predmeta, imena profesora i studentove ocene čine datoteku za taj predmet. Logičko grupisanje srodnih tabela čine bazu podataka (engl. database).

3.2 Kreiranje baze podataka

MODEL PODATAKA

Model podataka je dijagram koji predstavlja entitete u bazi podataka i njihove odnose

Podaci moraju biti organizovani tako da korisnici te podatke mogu preuzimati, analizirati i da ih mogu razumeti. Ključ za efikasno projektovanje baze podataka je model podataka (engl. data model). Model podataka je dijagram koji predstavlja entitete u bazi podataka i njihove odnose. Entitet (engl. entity) je osoba, mesto, stvar ili događaj (kao što su kupac, zaposleni ili proizvod) o kojima se čuvaju informacije. Entiteti se obično mogu identifikovati u radnom okruženju korisnika. Zapis obično opisuje entitet. Svaka karakteristika ili kvalitet određenog entiteta se naziva atribut (engl. attribute).

Svaki zapis u datoteci mora da sadrži najmanje jedno polje koje jedinstveno identificuje taj zapis, tako da mu se može biti pristupiti, da može da se ažurira i sortira. Ovo polje koje se koristi za identifikaciju se ustvari zove primarni ključ (engl. primary key). Na primer, zaposleni u nekoj firmi će koristiti JMBG kao svoj primarni ključ. U nekim slučajevima, lociranje određenog zapisa zahteva upotrebu sekundarnih ključeva. Sekundarni ključ (engl. secondary key) predstavlja drugo polje koje sadrži identifikacione informacije ali obično ne identificuje zapis sa potpunom tačnošću. Na primer, datum rođenja zaposlenog može biti njegov sekundarni ključ, tako da za određeni datum možemo pogledati koliko zaposlenih u firmi je rođenog tog dana. Zbog toga što više zaposlenih može imati isti dan rođenja, to ne bi trebalo da bude primarni ključ.

▼ Poglavlje 4

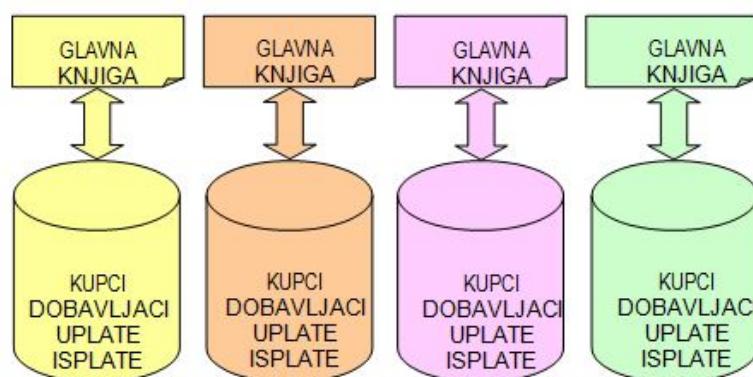
Sistemi baza podataka

PRIMER NEINTEGRISANOG INFORMACIONOG SISTEMA

Loše organizovane i neinteigrisane baze podataka u jednom preduzeću mogu voditi ka nefikasnošću i lošem poslovanju

Informacioni sistem jedne organizacije se, između ostalog, sastoji od skupa podataka i aplikacija. Svaka aplikacija ima potrebe da radi sa skupom podataka. Vrlo je verovatno da će se mnogi podaci ponavljati u većini aplikacija. Sa druge strane, svaka od ovih aplikacija će svoje podatke zapisivati u datotekama koristeći format koji je poznat samo toj aplikaciji.

Na sledećoj slici dat je primer informacionog sistema organizacije koja koristi četiri aplikacije: Prodaja, Proizvodnja, Plate i Glavna knjiga. Svaka od ovih aplikacija radi sa skupom podataka. Prodaja, na primer, koristi grupe podataka o: proizvodima, kupcima, radnicima, radnim mestima. Aplikacija Proizvodnja, takođe koristi podatke o proizvodima, radnicima i radnim mestima, ali ima i dodatnu grupu podataka o mašinama u proizvodnji. Ovoj aplikaciji nisu potrebni podaci o kupcima, ali su zato potrebni aplikacija Glavna knjiga. Svi ovi podaci su upisani u zasebne datoteke koje mogu biti na istom računarskom sistemu ili češće na različitim računarima koji koriste različite operativne sisteme i sisteme datoteka.



Slika 4.1.1 Primer informacionog sistema koji ne koristi bazu podataka [Izvor: Autor]

Iz ovog primera je očigledno da će se isti podaci unositi i čuvati na više mesta. Ovo je samo jedan od nedostataka ovakvog neintegrisanog rešenja. Gledano u celini ovakvi neintegrisani informacioni sistemi imaju niz nedostatka o kojima će nadalje biti reči.

PROBLEMI KOJI SE JAVLJAJU KOD NEINTEGRISANIH INFORMACIONIH SISTEMA

Redundantnost podataka, nepotreban razvoj istih ili sličnih algoritama, istovremeno korišćenje aplikacija od strane više korisnika

Kao što je rečeno najveći problem je u tome što se isti ili slični podaci unose i čuvaju na na više mesta korišćenjem različitih aplikacija. Pojava istih podataka u jednom informacionom sistemu se naziva **redundantnost podataka**. Redundantnost podataka je krajnje nepoželjna iz sledećih razloga:

- Prikupljanje i unos istih podataka vodi ka neefikasnosti organizacije.
- Potrebni su višestruki memorijski kapaciteti za čuvanje podataka.
- Podaci će verovatno biti nekonzistentni. Ukoliko se dogodi promena nekog podatka, moguće je da će on biti izmenjen u jednoj, ali ne i u svim aplikacijama, tako da će u istom informacionom sistemu jedan podatak imati različite vrednosti.

Sledeći problem koji se javlja kod neintegrisanih informacionih sistema je **nepotreban razvoj istih ili sličnih algoritama**. Za isti skup podataka razvijaju se posebne rutine za unos, čitanje, zapisivanje ili pretraživanje podataka, iako su podaci i algoritmi isti. Isti problem se javlja i prilikom promena u aplikaciji. Ovo kao posledicu ima skup informacioni sistem koga je teško održavati.

Poseban problem predstavlja **istovremeno korišćenje aplikacije od strane više korisnika**. Ako je jedan korisnik učitao datoteku sa podacima koje aplikacija koristi, ostali korisnici ne mogu da vide izmene koje je on uneo sve dok on ne zapiše izmenjene podatke. Ovo dovodi do netačnosti i nekonzistentnosti podataka.

Sve ovo je uticalo na to da se potraže nova rešenja koja će otkloniti uočene nedostatke ovakvih informacionih sistema. Rešenje je nađeno u sistemima baza podataka.

ŠTA ČINI SISTEM BAZE PODATAKA?

Baza podataka i sistem za upravljanje bazama podataka zajedno čine sistem baze podataka

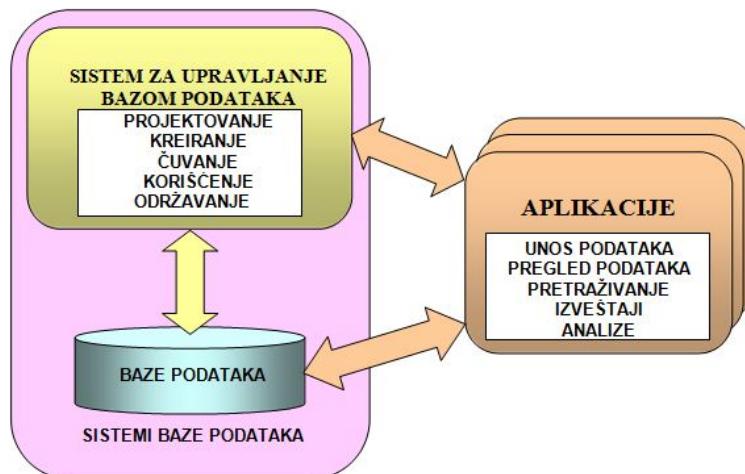
Kao što je ranije rečeno, **baza podataka** predstavlja kolekciju logički povezanih podataka, organizovanih tako da mogu da je koriste različiti korisnici ili programi i da iz nje dobiju potrebne informacije. Baza podataka se sastoji od podataka i metapodataka. **Metapodaci** su podaci koji opisuju strukturu baze podataka, njene tabele, ključeve i atributе, veze između tabela i pravila za očuvanje integriteta, prava pristupa i drugo.

Da bi jedna baza podataka mogla da se projektuje, kreira, koristi i održava potreban je čitav niz specijalizovanih programskih alata. Sistem programa kojim se vrši projektovanje, kreiranje, čuvanje, korišćenje i održavanje baze podataka naziva se **sistem za upravljanje bazama podataka** – **SubP** (engl. **data base management system - DBMS**).

Podaci predstavljaju jedinstveni resurs u nekom informacionom sistemu i njima sa upravlja na jedinstven način. SUBP omogućava da prosečan korisnik baze podataka ne mora da poznaje računarsku tehniku i ona je za njega jedan običan alat.

Najgrublje gledano, SUBP se sastoji od:

- Procesora upita
- Menadžera memorije.



Slika 4.1.2 Sistemi za upravljanje bazama podataka [Izvor: Autor]

Baza podataka i sistem za upravljanje bazama podataka zajedno čine sistem baze podataka. Dobro uređeni i prikupljeni podaci u samoj bazi podataka nemaju nikakvu vrednost ako se ne koriste. Zbog toga se uz neku bazu razvija jedna ili više aplikacija koje služe za eksploraciju podataka.

Aplikacije za korišćenje baze podataka mogu biti razvijene pomoću alata koje nudi sistem za upravljanje bazom podataka ili u nekom drugom jeziku, kao što je na primer Java ili C++.

SUBP ARHITEKTURA - VIDEO

DBMS Architechture

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ 4.1 Sistemi za upravljanje bazama podataka

ULOGA SISTEMA ZA UPRAVLJANJE BAZAMA PODATAKA (SUBP)

SUBP treba da omogući: kreiranje baze, postavljanje upita, čuvanje, integritet i konzistentnost podataka, logičku i fizičku nezavisnost, efikasan rad, minimalnu redundansu, jednostavnu komunikaciju

Korišćenjem sistema baza podataka svi podaci, koji su bili rasparčani u više zasebnih datoteka, su objedinjeni unutar jedne baze podataka, čime je izbegнутa redundantnost. Pristup podacima se obavlja na jedinstveni način, što doprinosi efikasnijem razvoju aplikacija. Konzistentnost podataka se održava zahvaljujući mehanizmima koji su ugrađeni u sistem za upravljanje bazama podataka (SUBP). Ovo omogućuje da više korisnika istovremeno može da pristupi podacima, a da se tačnost i konzistentnost podataka održi.

Od svakog SUBP zahteva se da omogući:

- **Kreiranje** nove baze podataka kao i specificiranje njene šeme (logičke strukture podataka) korišćenjem specijalizovanog jezika koji se zove **data definition language** - **DDL**
- **Postavljanje upita**, odnosno dobijanje informacija iz baze podataka koje zadovoljavaju određene kriterijume, kao i modifikovanje podataka jezikom koji se zove **data manipulation language** - **DML**
- **Čuvanje** veoma velike količine podataka
- Automatsko **čuvanje integriteta i konzistentnosti** podataka, kako u normalnom radu, tako i u slučaju hardverskih ili softverskih otkaza ili pri nestanku napajanja.
- Nesmetano **paralelno korišćenje od strane više korisnika**
- **Sigurnost** - zaštitu podataka od pogrešnog unosa, neovlašćenog pristupa ili neautorizovanog korišćenja
- **Logička nezavisnost podataka** - razdavaja se logička definicija cele baze podataka od lokalnih logičkih definicija za jednu aplikaciju. Uvođenje novih veza i vrsta podataka ne utiče na postojeće aplikacije.
- **Fizička nezavisnost programa od podataka** - razdvaja se logička definicija baze podataka od njene stvarne fizičke građe.
- **Efikasan rad** - obezbeđivanje efikasnog i brzog pristupa podacima tokom upita ili tokom modifikacije
- **Minimalna redundandnsa podataka**, to jest, da je višestruko pamćenje istih podataka svedeno na minimum
- **Jednostavno komuniciranje sa bazom podataka**, kako za krajnjeg korisnika, tako i za programere.
- **Podešavanje baze i kontrola** - rutinsko održavanje, praćenje performansi i njihovo poboljšanje radi efikasnosti, razvoj aplikacija, nadogradnja. Ove poslove obavlja **administrator baze podataka**. Administratoru za to služe razni pomoći alati, a jedno od njih je **rečnik podataka** (meta podaci, baza o bazi).

▼ 4.2 SUBP – Procesor upita

ZADACI PROCESORA UPITA

Zadatak procesora upita je da obradi podatke koje definišu entitete i njihove veze, da definiše upite i da ih izvrši

Procesor upita (query processor, query evaluation engine) ima nekoliko zadataka:

- *Obradu podataka koji definišu entitete baze i veza između njih pomoću DDL interpretatora.* Ovi podaci se definišu jezikom za definisanje podataka (**DDL - Data Definition Language**). *Na osnovu ovih definicija se kreira konceptualna ili logička šema baze podataka i rečnik podataka.*
- *Definisanje upita, njegovo prevođenje i optimizacija.* Za definisanje upita se kod relacionih baza podataka koristi **DML - Data Manipulation Language**. DML je deo strukturnog jezika upita SQL (**Structured Query Language**) koji je standardni jezik relacionih baza podataka. SQL spada u deklarativne jezike. Drugim rečima on kazuje sistemu za upravljanje bazama podataka šta treba da se uradi, ali ne i kako. Upit može da se definiše na različite načine:
 - Korišćenjem administratorskih alata
 - Korišćenjem alata SUBP za postavljanje upita od strane analitičara
 - Korišćenjem nekog programskog jezika koji je deo SUBP
 - Korišćenjem jezika višeg nivoa, odnosno aplikacijom koja je razvijena u ovakvim jezicima.
- *Izvršavanje upita*. Kada je jedan upit definisan mašina za izvršavanje upita ga interpretira i izvršava upit. *Mašina za izvršavanje upita transformiše neproceduralni iskaz upita u niz akcija koji treba da obavi sistem za upravljanje bazama podataka da bi korisnik dobio traženi odgovor.*

▼ 4.3 SUBP – Menadžer skladištenja

KOMPONENTE MENADŽERA SKLADIŠTENJA (STORAGE MANAGER)

Menadžer skladištenja ima četiri komponente i to menadžer: datoteka (file manager), bafera (buffer manager), autorizacije i integriteta, transakcije

Menadžer skladištenja (engl. **storage manager**) je sistem koji upravlja skladištenjem i pristupom podacima. On ima četri komponente:

- **Menadžer datoteka** (engl. **File Manager**) upravlja zapisivanjem i čitanjem datoteka baze podataka. Ove datoteke su tipično fizički smeštene na disku.

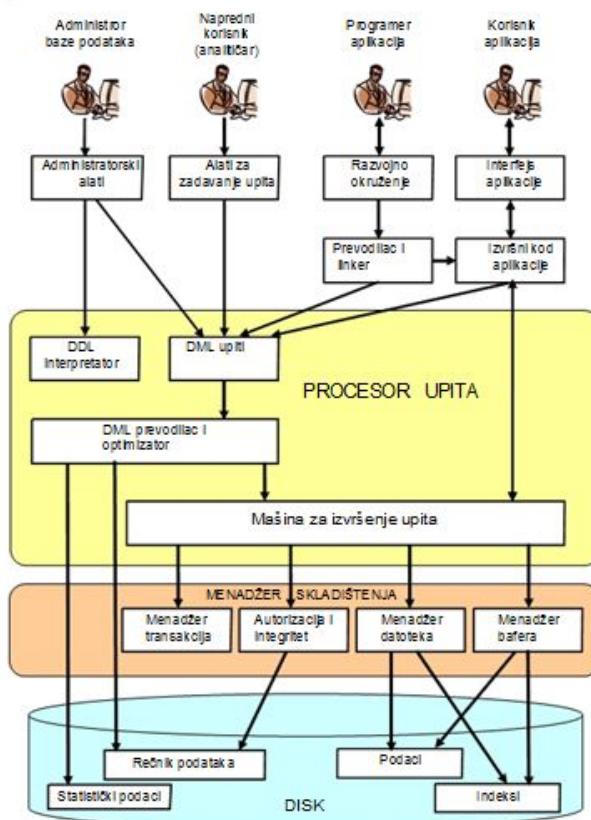
- **Menadžer bafera** (engl. *Buffer manager*) ima zadatak da upravlja smeštanjem podataka iz baze u radnu memoriju računara.
- **Menadžer autorizacije i integriteta** proverava prava pristupa podacima i brine o očuvanju integriteta baze podataka.
- **Menadžer transakcija** ima zadatak da obezbedi konzistentan način promena zapisa u bazi. On se brine da se konzistentnost podataka očuva pri simultanom korišćenju podataka od strane više korisnika i u incidentnim situacijama kao što su hardverski softverski otkazi ili nestanak napajanja. Pored toga ovaj modul je zadužen i za oporavak baze podataka u slučajevima kada je konzistentnost baze narušena.

ŠEMA SISTEMA ZA UPRAVLJANJE BAZAMA PODATAKA

Tipični korisnici SUBP-a su: administrator, analitičar, programer i korisnik

Na narednoj slici je data šema sistema za upravljanje bazama podataka. Na sliци su prikazani tipični korisnici:

- Administrator baze podataka, koji koristi alate za administriranje baze
- Analitičar, koji kreira specifične upite
- Programer, koji projektuje bazu podataka i aplikacije koje služe za unos podataka i njihovu obradu.
- Korisnik aplikacije, koji unosi podatke, obrađuje podatke ili generiše tražene standardne izveštaje koristeći razvijene aplikacije.



Slika 4.2.1 Šema sistema za upravljanje bazama podataka [Izvor: Autor]

ORGANIZACIJA BAZE PODATAKA

Radi lakšeg upravljanja i korišćenja baze podataka, postoji interna, konceptualna i eksterna organizacija baze podataka

Sistem za upravljanje bazama podataka uobičajeno dolazi u paketu sa mnogim alatima kao što su:

- Alati za projektovanje baze podataka
- Alati za razvoj baze podataka
- Alati za održavanje baze podataka
- Alati za korišćenje baze podataka

Baze podataka se obično zapisuju na disku korišćenjem menadžera skladištenja na način koji se naziva **interna ili fizička organizacija**, koja je prilično složena i za običnog korisnika teško shvatljiva. Zbog toga se baze podataka prikazuju uprošćeno, tako da im organizacija bude bliska analitičarima i programerima, a samim tim i manipulacije sa njima. Ovakav prikaz naziva se **konceptualna ili logička organizacija** i ona predstavlja korisnikovo viđenje organizacije podataka. Različite aplikacije pristupaju različitim podacima u bazi, tako da postoji i treća organizacija koja se naziva **eksterna organizacija**. Polazeći od ovakvog sagledavanja arhitektura sistema za upravljanje bazama podataka se može predstaviti sa tri nivoa: interni, konceptualni i eksterni.

Za stvaranje baze podataka potrebno je zadati šemu i poglede, a SUBP će se tada automatski pobrinuti za raspored i zapisivanje podataka na fizičkom nivou. Programi i korisnici ne pristupaju podacima direktno, već posredstvom SUBP. Korisnik komunicira sa bazom podataka preko programa razvijenih u nekom programskom jeziku koji nazivamo jezik domaćin (**Host Language**) ili direktno preko upitnih jezika (preko interfejsa za postavljanje upita ili preko specifičnog korisničkog interfejsa). Komunikacija programa i korisnika sa SUBP se obavlja na lokalnom logičkom nivou.

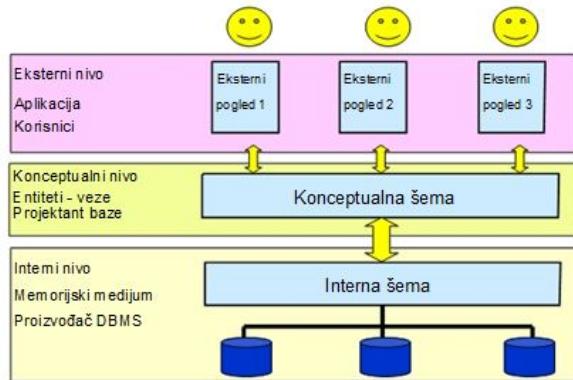
NIVOI U ARHITEKTURI SISTEMA ZA UPRAVLJANJE BAZAMA PODATAKA

Arhitektura sistema za upravljanje bazama podataka se može predstaviti sa tri nivoa: interni, konceptualni i eksterni.

Arhitektura sistema za upravljanje bazama podataka se može predstaviti sa tri nivoa:

1. Interni (fizički) nivo, koji se odnosi na fizičko organizovanje podataka na spoljašnjim memorijama. Pristup ovom nivou imaju programi koji čine sistem za upravljanje podataka i neke napredne aplikacije. Prema tome, korisnici internog pogleda na podatke su projektanti sistema za upravljanje bazama podataka i programeri. Ovaj nivo ima svoje podstrukture: prostor na disku (cilindri i blokovi) kao i datoteke sa zapisima. Raspored skladištenja opisuje kako se logičke definicije preslikavaju na fizičke uređaje, kao i metode pristupa.
2. Konceptualni nivo, (šema baze podataka) koji definiše entitete baze, atribute i ograničenja, kao i relacije između entiteta. Konceptualni nivo se odnosi se na logičku predstavu o celoj bazi. Ovim pogledom na bazu podataka se koriste projektanti, analitičari i administratori baze. Zapis logičke definicije zove se šema. To je tekst ili dijagram kojim se imenuju i definišu svi podaci, veze među podacima, kao i pravila kojima se čuva integritet baze podataka.
3. Eksterni nivo – korisnički logički nivo: Čine je aplikacioni programi. Aplikacioni programi se služe samo delom podataka koje postoje u BP. Deo podataka koje vide pojedine aplikacije ili kojim se služe, zove se pregled (**view**). Korisnici koji koriste različite aplikacije imaju različite predstave o bazi podataka.

Ovakva arhitektura treba da omogući da promena fizičke strukture podataka ne podrazumeva i promenu opšte logičke strukture, odnosno da učine nezavisnom logičku od fizičke strukture baze podataka, kao i aplikacione programe od fizičke i logičke strukture baze.



Slika 4.2.2 Arhitektura sistema za upravljanje bazama podataka [Izvor: Autor]

▼ Poglavlje 5

Upravljanje znanjem

ZNANJE U KONTEKSTU INFORMACIONIH TEHNOLOGIJA

Znanje je informacija koja se može smestiti u određeni kontekst, koja je relevantna i na osnovu koje se mogu preuzeti neke mere.

Podaci i informacije su veoma bitni zato što predstavljaju imovinu jedne firme. Znanje je od velikog značaja za kompaniju. Uspešni menadžeri uvek korisne intelektualnu svojinu i prepoznaju njenu vrednost. Međutim, oni nisu bili sistematični i nisu osigurali da se znanje deli i širi na način na koji bi organizacija imala najviše koristi. Štaviše, industrijski analitičari procenjuju da je većina kompanijskog znanja nije smeštena u relacione baze podataka. Umesto toga, oni su razbacani u e-mail-ovima, Word dokumentima, tabelama i prezentacijama na raznim računarima u kompaniji. Ovaj način rada predstavlja problem kada kompanije žele da integrišu to znanje. Rezultat toga je obično neefikasno donošenje odluka.

Upravljanje znanjem (engl. **Knowledge management(KM)**) je proces koji pomaže organizaciji da manipuliše važnim znanjem koji je deo organizacijske memorije, obično u nestruktuiranom formatu. Kako bi organizacija bila uspešna, znanje kao kapital, mora postojati u formatu koji se može razmenjivati između ljudi. Takođe, mora da ima mogućnost rasta.

U kontekstu informacionih tehnologija, znanje se razlikuje od podataka i informacije. Podatak predstavlja kolekciju činjenica, merenja, statistike i slično, a informacija predstavlja organizovane ili obrađene podatke koji su organizovani na pravovremenim načinima i koji su tačni. Znanje (engl. **knowledge**) je informacija koja se može smestiti u određeni kontekst, koja je relevantna i na osnovu koje se mogu preuzeti neke mere. Jednostavno rečeno, znanje je informacija u akciji. Intelektualni kapital je drugi naziv za znanje.

EKSPLICITNO I IMPLICITNO ZNANJE

Eksplicitno znanje je dokumentovano u obliku koji može da se distribuira drugima ili je pretvoren u proces ili strategiju. Implicitno znanje se sastoji od iskustava te organizacije...

Eksplicitno znanje (engl. **Explicit knowledge**) je objektivno, tehničko i racionalno znanje. U organizaciji, eksplicitno znanje se sastoji od procedura, izveštaja, proizvoda, strategija, ciljeva, osnovnih kompetencija preduzeća i IT infrastrukture. Drugim rečima, eksplicitno znanje je znanje je dokumentovano u obliku koji može da se distribuira drugima ili je

prevoren u proces ili strategiju. Opis kako da se obrade prijave na konkurs za posao, koji dokumentovan kao procedura u kadrovskoj službi, predstavlja primer eksplizitnog znanja.

Nasuprot tome, **implicitno znanje** (engl. **tacit knowledge**) predstavlja kumulativni skup subjektivnog učenja ili učenja iz iskustva. U organizaciji, implicitno znanje se sastoji od iskustava te organizacije, stručnosti, ekspertiza, umeća, poslovnih tajni, veština, razumevanja i učenja. Ono takođe obuhvata organizacionu kulturu, koja odražava prošla i sadašnja iskustva ljudi i procesa u organizaciji, kao i dominantnih vrednosti. **Implicitno znanje se uglavnom sporo, neprecizno i teško prenosi.** Takođe je veoma lično. Konačno, zato što je nestruktuirano, teško se formalizuje ili dokumentuje. Na primer, prodavci koji su radili sa pojedinim klijentima tokom vremena sasvim dobro saznaju potrebe tih kupaca. Ovo znanje se obično ne beleži. U stvari, to bi u principu bilo teško da se pretoči u pisanu formu.

❖ 5.1 Sistemi za upravljanje znanjem

ŠTA JE SISTEM ZA UPRAVLJANJE ZNANJEM?

Sistem za upravljanje znanjem se odnosi na korišćenje savremenih informacionih tehnologija kako bi sistematizovali i ubrzali intrakompanijskim i interkompanijskim upravljanjem znanja

Cilj upravljanja znanjem je da pomogne organizaciji da koristi znanje koje ima na najefikasniji način. Istoriski gledano, sistemi za upravljanje informacija su bili fokusirani na prikupljanje, čuvanje, upravljanje i prenošenje eksplizitnog znanja. Sada je poznato da je potrebno da se eksplizitno i implicitno znanje integrišu u informacione sisteme.

Sistem za upravljanje znanjem (engl. **Knowledge management systems (KMSs)**) se odnosi na korišćenje savremenih informacionih tehnologija kao što su Internet, intranet, ekstranet, skladišta podataka, kako bi sistematizovali, unapredili i ubrzali intrakompanijskim i interkompanijskim upravljanjem znanja. KMS ima cilj da pomogne organizaciji da se što efikasnije nose sa brzim promenama, prometom i smanjenju kompanije na taj način što će učiniti dostupan ljudski kapital organizacije.

CIKLUS SISTEMA ZA UPRAVLJANJE ZNANJEM

Ciklus sistema za upravljanje znanjem se sastoji od kreiranja, beleženja, pročišćavanja, čuvanja, upravljanja i distribucije znanja

Rad sistema za upravljanje znanjem se može opisati kroz ciklus koji se sastoji iz šest koraka. Razlog zašto je sistem cikličan je zbog toga što se znanje dinamički pročišćava tokom vremena. Znanje u efikasnom sistemu za upravljanjem znanja nikada nije konačno zato što se okruženje menja tokom vremena, pa se samim tim i znanje ažurira kako bi odgovaralo tim izmenama. Ciklus funkcioniše na sledeći način:

- 1. Kreiranje znanja.** Znanje se kreira tako što ljudi pronađu nove načine da nešto urade ili razviju specifična znanja i umeća. Ponekad se koristi i eksterno znanje.
- 2. Beleženje znanja.** Novo znanje mora da se prepozna kao vredno i mora da se predstavi na razumljiv način.
- 3. Pročišćavanje znanja.** Novo znanje se mora staviti u kontekst kako bi moglo da se primenjuje. U ovom slučaju se implicitno znanje mora beležiti kao i eksplisitno znanje.
- 4. Čuvanje znanja.** Korisno znanje mora da bude sačuvano u razumnom formatu u repozitorijumu za znanja, tako da mu i drugi iz organizacije mogu pristupiti.
- 5. Upravljanje znanjem.** Kao i biblioteka, znanje mora da bude ažurno. Znanje sem mora regularno da se proverava kako bi se proverilo da li je relevantno i tačno.
- 6. Distribucija znanja.** Znanje mora da bude dostupno svima u organizaciji kojima je potrebno, u bilo kom trenutku i bilo kada.

▼ Poglavlje 6

Pokazne vežbe: Osnove SQL jezika upita

KOMPLEKSНОСТ БАЗА ПОДАТКА

Za velike računarske sisteme se mogu koristiti Oracle, SQL Server i slično, dok je za male personalne računare popularan MS Access

Predviđeno vreme pokazne vežbe je 20 minuta.

Baza podataka je računarski sistem za čuvanje podataka. Generalno, ovakav sistem čine podaci, hardver koji fizički smešta podatke, softver koji koristi odgovarajući sistem datoteka i konačno korisnik koji podatke pretvara u informacije. Baza podataka omogućava:

- skladištenje podataka
- standardizovani metod pristupanja i promene podataka.

Baze podataka mogu biti veoma kompleksne, npr. sistem računa u banci koji se brine o hiljadama korisnika i njihovom novcu, ili vrlo jednostavan sistem kao što su podaci sa vizit karti na prenosivom računaru nekog poslovног čoveka. Veoma je važno da baza dozvoljava skladištenje podataka, njihovo preuzimanje ili promenu u trenutku kada je to potrebno, na lak i efikasan način bez obzira na količinu podataka kojom se manipuliše.

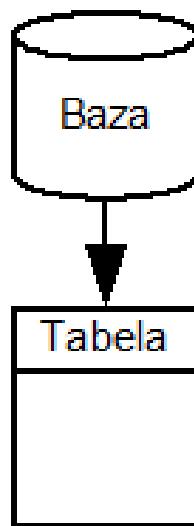
Baze se koriste na velikim, moćnim *mainframe* računarima za poslovne aplikacije. Tada se koriste *Oracle*, *SQL Server* i sl. Za male, personalne računare, *MS Access* je popularan program koji je baze podataka približio prosečnom korisniku. U mnogim *web* aplikacijama postoji potreba da se pojedini podaci čuvaju u bazi podataka.

U velikom broju slučajeva radi se o relacionim bazama, sa osnovnim zahtevima brzine i lakoće administriranja. Jedna relaciona baza sadrži tabele koje se sastoje od kolona i vrsta, a međusobno su povezane.

TABELE U BAZAMA PODATAKA

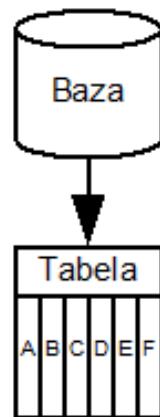
Tabela sadrži podatke o određenoj temi, a tabela može da sadrži mnogo različitih kolona

Tabele su najveći element baza podataka i predstavljaju drugi korak u redosledu kreiranja, odmah nakon kreiranja same baze. Na slici 1 prikazana je tabela kao deo baze podataka.



Slika 6.1.1 Tabela [Izvor: Autor]

Svaka tabela baze može se podeliti na odgovarajuće kolone, kao što je prikazano na slici 2.



Slika 6.1.2 Tabela i kolone [Izvor: Autor]

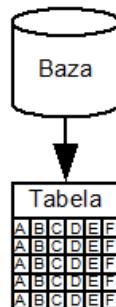
KOLONE I ZAPISI U TABELAMA

Svaki red (vrsta) u jednoj tabeli predstavlja jedan zapis. Svaka kolona u tabeli predstavlja podatke smeštene u tačno određeno polje te tabele

Kolone se kreiraju u isto vreme kada i tabele. Definisanje kolone je najvažniji deo komande kreiranja tabele. Za definisanje kolone neophodno je opredeliti se za njihov tip i dužinu. Tip kolone direktno je povezan sa tipom podatka koji se smešta u tu kolonu. Na primer, ukoliko je u kolonu potrebno smestiti četvorocifreni celi broj, onda se polja definišu kao *small integer* - SMALLINT, što znači da omogućavaju smeštanje brojeva od 0 do 65 535.

U slučaju da se ova polja definišu na neki drugi način, npr. kao tip TEXT, koji je namenjen podacima dužine do 65 535 karaktera, neefikasno bi se koristio prostor jer bi se od tog broja karaktera koristila samo četiri za zapis četvorocifrenog broja. O važnosti definisanja moguće

dužine podatka više reči biće u nastavku. Podaci se u tabelu unose kao vrste ili zapisi. Na slici 3 dat je prikaz vrsta u tabeli.



Slika 6.1.3 Kolone i redovi [Izvor: Autor]

Svaka tabela ima i svoje ime. U nastavku je dat primer tabele **Student**:

indeks	prezime	ime	prosek
1	Petrović	Petar	7,56
2	Mirković	Mirko	8,00
3	Marković	Marko	9,50

Slika 6.1.4 Tabela Student [Izvor: Autor]

Ova tabela sastoji se od tri vrste, po jedna za svakog studenta i četiri kolone različitog tipa i dužine.

Svaki zapis u tabeli sadrži informacije o jednoj stavci, kao što je određeni zaposleni. Zapis se sastoji od polja, poput polja za ime, prezime i prosek. Zapis se obično zove i red, a polje se obično zove i kolona.

PRISTUP I MANIPULACIJA PODACIMA U BAZI PODATAKA

SQL omogućava pristup bazama podataka

Za pristup i manipulaciju podacima u bazama koristi se SQL (Structured Query Language), standardni računarski jezik za upravljanje sistemima baza podataka. SQL omogućava pristup bazama podataka i ima sledeće osobine:

- zadovoljava ANSI (American National Standard Institute) standard računarskih jezika,
- omogućava izvršavanje upita nad nekom bazom podataka, i
- jednostavan je za učenje.

SQL se može podeliti na: DDL (Data Definition Language) i DML (Data Manipulation Language) jezike. DDL omogućava kreiranje ili brisanje tabela u bazama podataka. DML se sastoji od naredbi za preuzimanje, ažuriranje, brisanje ili dodavanje podataka u tabeli.

❖ 6.1 Pokazne vežbe: Komande DDL jezika i MS Access

PRIMER 1

Za kreiranje tabele koristi se komanda **CREATE TABLE**

Predviđeno vreme pokazne vežbe je 30 minuta.

Zadatak (Vreme izrade zadatka: 5 minuta)

Uz pomoć Access i naredbi SQL jezika kreirati tabelu **Student** baze podataka **FIT** sa četiri kolone: **indeks**, **prezime**, **ime** i **način**, gde je prvi podatak celobrojna vrednost. Ostali podaci su tekstualni, od čega su prva dva maksimalne dužine 30, a poslednji 8 karaktera.

Rešenje:

Generalno, za kreiranje tabele koristi se sledeća SQL sintaksa:

```
CREATE TABLE ime_tabele(  
    ime_kolone1 tip_podatka,  
    ime_kolone2 tip_podatka,  
    .....  
);
```

U ovom slučaju ova naredba se koristi na sledeći način:

```
CREATE TABLE Student(  
    indeks integer,  
    prezime varchar(30),  
    ime varchar (30),  
    način varchar (8)  
);
```

Da bi kreiranje tabele bilo efikasno u smislu memorijskog prostora koje može da zauzme, često se ograničava dužina podataka koji se mogu uneti u tabelu. Za ovaj primer, dužina imena i prezimena ograničena je na 30 karaktera, što bi trebalo da bude dovoljno za unos podataka koji su karakteristični za naše podneblje, ali ne treba zanemariti i mogućnost pojave izuzetaka. Način studiranja je ograničen na dve vrednosti: internet i klasično, gde su obe dužine 8 karaktera.

NAJČEŠĆE KORIŠĆENI SQL TIPOVI PODATAKA

Najčešće korišćeni SQL tipovi podataka su integer, int, smallint, tinyint, decimal, numeric, char, varchar, date

Na slici 1 navedeni su najčešće korišćeni tipovi kod primene SQL jezika:

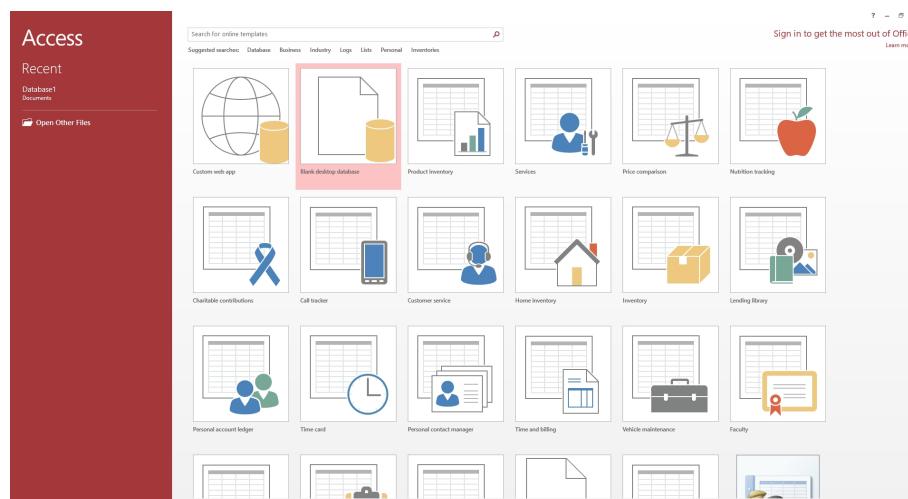
Tip podatka	Opisuje
integer(broj)	Opisuje celobrojne vrednosti. Ukoliko je potrebno definisati, maksimalni broj cifara navodi se u zagradama.
int(broj)	
smallint(broj)	
tinyint(broj)	
decimal(broj,d)	Opisuje razlomljene vrednosti. Maksimalni broj cifara definisan je uz pomoć promenljive broj, dok je sa d definisan broj cifara sa desne strane decimalnog zareza.
numeric(broj,d)	
char(broj)	Opisuje string (može sadržati slova, brojeve i posebne znake) konstantne dužine koja se definiše u zagradama.
varchar(broj)	Opisuje string (može sadržati slova, brojeve i posebne znake) promenljive dužine, čija se maksimalna vrednost definiše u zagradama.
date(yyyyymmdd)	Opisuje datum. Sa yyyy definiše se godina, mm mesec i sa dd dan.

Slika 6.2.1 SQL tipovi podataka [Izvor: Autor]

PRIMENA MS ACCESS-A

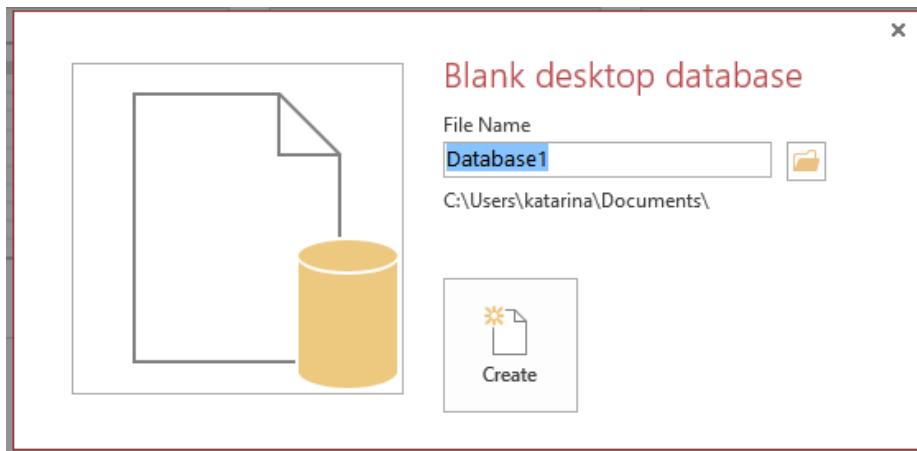
Prvi korak podrazumeva kreiranje nove baze podataka opcijom New

U nastavku biće opisan postupak korišćenja MS Access okruženja. Access se pokreće na jedan od dva standardna načina: duplim klikom na odgovarajuću ikonu, ukoliko se ona nalazi na radnoj površini ili preko **Start** menija: **Start→All Programs→Microsoft Access**. Potom, se u meniju File bira opcija **New....** Nakon toga, u prozoru koji se pojavio, kliknuti na opciju **Blank database**, da bi se kreirala nova, prazna baza podataka



Slika 6.2.2 Izgled MS Access alata [Izvor: Autor]

U novom prozoru koji se pojavio , na samom početku, potrebno je snimiti bazu koja se kreira. Baza se imenuje i bira se direktorijum u kom će biti sačuvana.



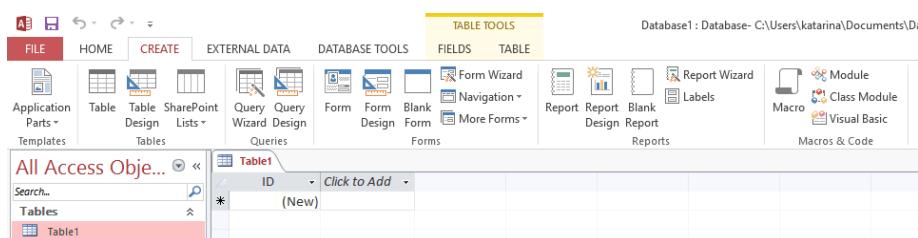
Slika 6.2.3 Imenovanje i snimanje baze [Izvor: Autor]

Klikom na opciju Create završava se kreiranje baze.

KORIŠĆENJE SQL UPITA U MS ACCESS-U

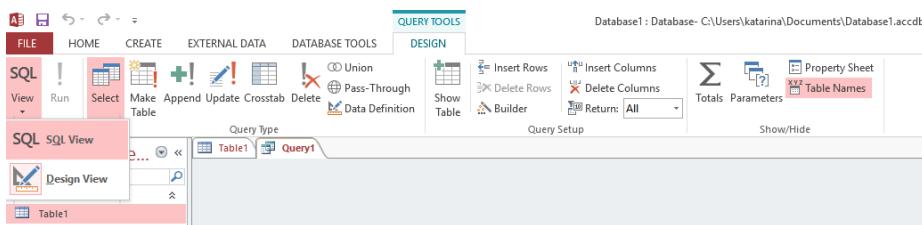
Queries se kreira pomoću Queries Wizard-a opcijom Create query in Design view

Da bi se koristio SQL jezik upita, potrebno je kliknuti na tab Create u liniji menija koja je prikazana na slici 4. Među prikazanim opcijama u Create tabu, sada treba izabrati opciju Query Design nakon čega će biti prikazana nova radna površina.



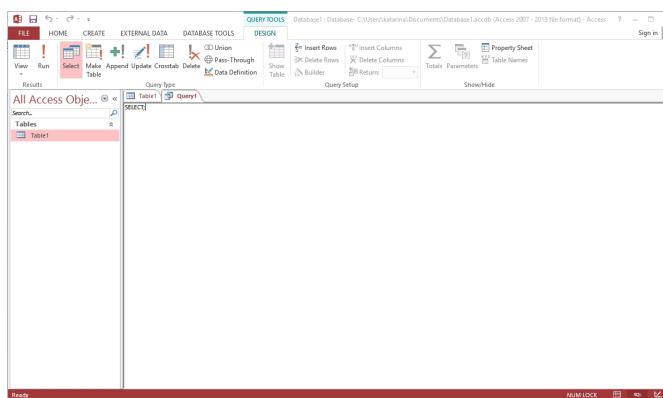
Slika 6.2.4 Queries Wizard [Izvor: Autor]

Prozor koji će se potom pojaviti, zatvoriti opcijom **Close**. Na liniji sa menijima pojaviće se novi meni - **View**. U ovom meniju nalazi se pod-meni **SQL View** na koji treba kliknuti kao što je to prikazano na slici 5.



Slika 6.2.5 Opcija SQL view [Izvor: Autor]

Klikom na ovu opciju biće prikazana radna površina za pisanje SQL upita koja je prikazana na slici 6.

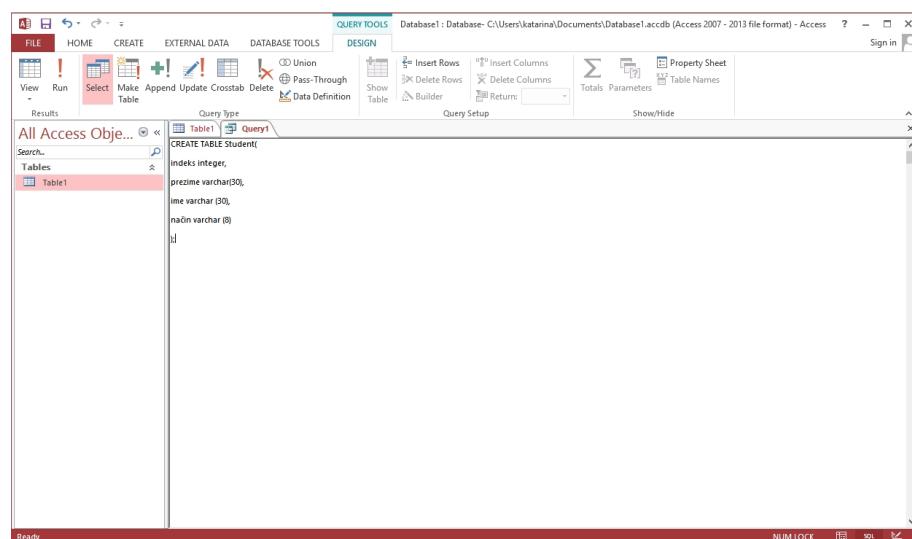


Slika 6.2.6 Radna površina za pisanje SQL upita [Izvor: Autor]

UNOS I IZVRŠAVANJE UPITA U MS ACCESS-U

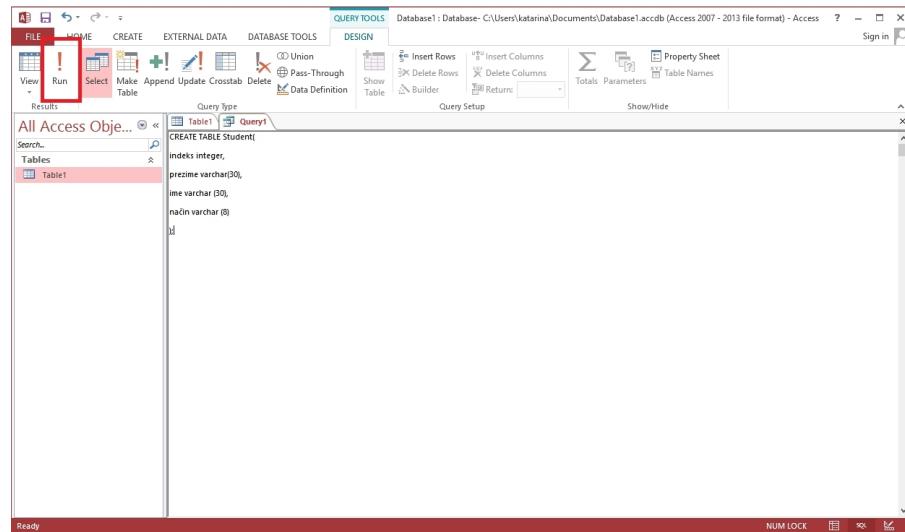
Data Definition Query se koristi za unos upita, nakon čega se upit izvršava opcijom Run

U otvorenom prozoru unose se upiti SQL jezika. Za primer kreiranja tabele Studenti, upit je prikazan na slici 7.



Slika 6.2.7 Upit za kreiranje tabele Studenti [Izvor: Autor]

Upit se izvršava tako što se u liniji menija klikne na opciju **Run** (slika 8).

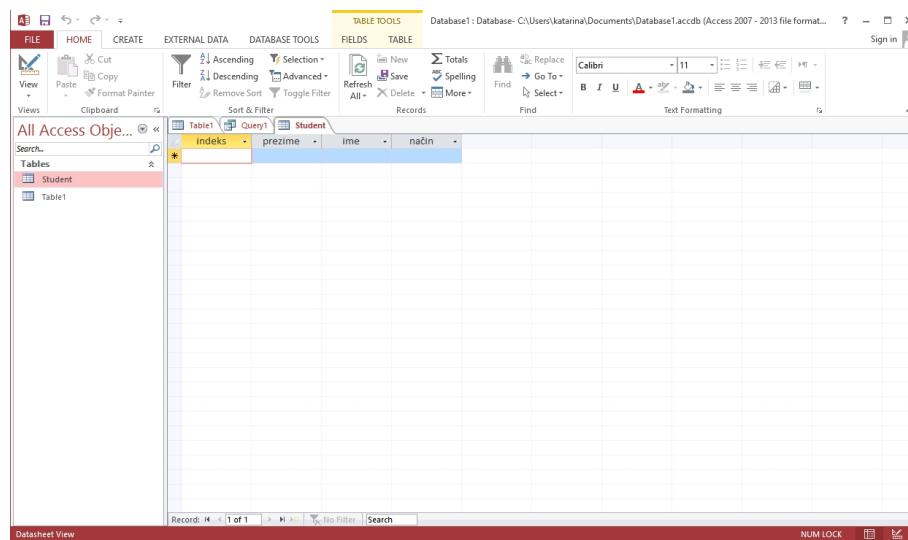


Slika 6.2.8 Izvršavanje upita [Izvor: Autor]

OFORMLJENA TABELA NAKON IZVRŠENJA UPITA

Tabelu je moguće videti nakon izvršenja upita

Nakon izvršenja upita, u delu **Tables** pojaviće se nova tabela **Student**.



Slika 6.2.9 Prikaz nove tabele [Izvor: Autor]

KREIRANJE JEDINSTVENOG INDEKSA

Kreiranje jedinstvenog indeksa (unique index) znači da dve vrste ne mogu imati indekse sa istim vrednostima.

Da bi rad sa bazama podataka bio efikasniji koriste se odgovarajući indeksi u tabelama. Njihovim kreiranjem omogućeno je brže i efikasnije lociranje podataka. Moguće je kreirati

indeks nad jednom ili više kolona i svakom od njih dodeliti ime. Korisnik nije u prilici da vidi indekse, oni se samo koriste za ubrzavanje rada.

Sa druge strane, ažuriranje podataka u tabeli koja sadrži indekse zahteva više vremena od tabele gde nema indeksa. Razlog je činjenica i da je indekse potrebno ažurirati. Jedna od preporuka je kreiranje indeksa samo nad kolonama koja se često koriste prilikom traženja podataka. **Kreiranje jedinstvenog indeksa (unique index)** znači da dve vrste ne mogu imati indekse sa istim vrednostima. Naredba izgleda ovako:

CREATE UNIQUE INDEX ime_indeksa

ON ime_tabele (ime_kolone);

Narednom naredbom kreira se jednostavni indeks, što znači da je dozvoljeno korišćenje iste vrednosti:

CREATE INDEX ime_indeksa

ON ime_tabele (ime_kolone);

Na sličan način formira se primarni ključ (*primary key*):

CREATE TABLE ime_tabele(

ime_kolone1 tip_podatka PRIMARY KEY,

ime_kolone2 tip_podatka,.....);

U ovom primeru je najprikladnije kao primarni ključ koristiti broj indeksa:

CREATE TABLE Student

(indeks integer PRIMARY KEY,

prezime varchar(30),

ime varchar (30),

način varchar (8));

Na kraju ovog primera, važno je spomenuti da se korišćenjem SQL jezika, na sličan način kao i tabela, kreira se i sama baza:

CREATE DATABASE ime_baze;

PRIMER 2

Za dodavanje kolone u tabelu koriste se komande ALTER TABLE i ADD

Zadatak (Vreme izrade zadatka: 5 minuta)

U tabeli Student, kreiranoj u prethodnom primeru, dodati novu kolonu **prosek** u kojoj će se nalaziti prosek ocena za vreme studiranja studenta.

Potom iz tabele izbrisati kolonu koja se odnosi na način studiranja

Rešenje:

Za dodavanje kolone u tabelu koristi se sledeća naredba:

ALTER TABLE ime_tabele

ADD ime_kolone tip_podatka;

Za ovaj primer, s obzirom da se radi o prosečnoj oceni, tip podatka koji bi odgovarao je **numeric**. Naredba izgleda ovako:

ALTER TABLE Student

ADD prosek numeric;

Ova naredba unosi se u prozor upita i izvršava naredbom **Run**, na način opisan u prvom primeru. Na taj način dobija se tabela prikazan na slici 10.

Za brisanje kolone koristi se sledeća naredba:

ALTER TABLE ime_tabele

DROP COLUMN ime_kolone;

Što za ovaj primer znači:

ALTER TABLE Student

DROP COLUMN način;

Za brisanje čitave tabele, odnosno baze, koriste se sledeće naredbe SQL jezika:

DROP TABLE ime_tabele;

DROP DATABASE ime_baze;

indeks	prezime	ime	način	prosek
*				

Slika 6.2.10 Ažurirana tabela [Izvor: Autor]

✓ 6.2 Pokazne vežbe: Osnovne komande DML jezika

PRIMER 1

*Za prethodno kreiranu tabelu podaci se mogu uneti sa **INSERT INTO** i **VALUES** komandama*

Predviđeno vreme pokaznih vežbi: 45 minuta.

Za prethodno kreiranu tabelu **Student**, napisati SQL upite kojom će se u tabelu uneti sledeći podaci:

indeks	prezime	ime	prosek
1	Petrović	Petar	7.56
2	Nenadić	Marko	8.00
3	Marković	Marko	9.50

Slika 6.3.1 Primer tabele [Izvor: Autor]

Vreme izrade primera je 5 minuta.

Rešenje:

Za unošenje nove vrste u tabelu, zapravo novog podatka koristi se SQL naredba sledeća sintakse:

INSERT INTO ime_tabele VALUES (vrednost1, vrednost2 ...);

gde broj promenljivih vrednost odgovara broju kolona u tabeli. Za ovaj primer to bi izgledalo ovako.

INSERT INTO Student VALUES (1, 'Petrović', 'Petar', 7.56);

Postupak je potrebno ponoviti za sve date studente.

Slika 6.3.2 Uneti podaci u tabelu [Izvor: Autor]

Postoji mogućnost unošenja podataka u samo određene kolone po sledećem obrascu:

INSERT INTO ime_tabele (kolona1, kolona2,...) VALUES (vrednost1, vrednost2 ...);

gde se iza imena tabele unose nazivi kolona u koje se unose vrednosti respektivno.

Na primer za postojeću tabelu **Student** može se koristiti sledeći upit za unošenje podataka o izvesnom studentu sa imenom Kosta, za koga se trenutno zna samo da će imati broj indeksa 10:

INSERT INTO Student (indeks, ime) VALUES (10, 'Kosta');

pa će tabela sada izgledati kao na slici 3.

Slika 6.3.3 Uneti podaci u tabelu [Izvor: Autor]

PRIMER 2

Za odabir podataka iz tabele mogu se koristiti komande SELECT i FROM

Zadatak (Vreme izrade zadatka: 5 minuta)

Za tabelu dobijenu u prethodnom primeru, uz pomoć SQL jezika prikazati imena i prezimena svih studenata, a potom pronaći sve studente koji se zovu Marko. Na kraju, pronaći sve studente čiji je broj indeksa manji od 3.

Rešenje:

Za izbor podataka u tabeli koristi se sledeći izraz:

SELECT kolona1, kolona2,...

FROM ime_tabele;

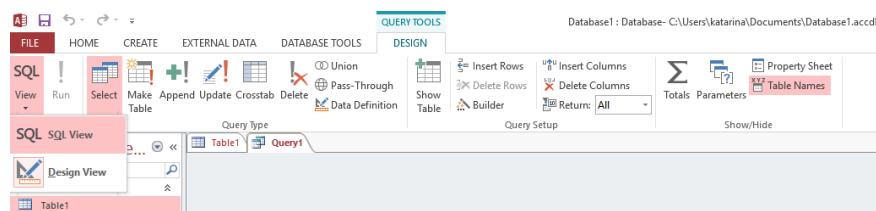
Za prikazivanje imena i prezimena svih studenata iz tabele u prethodnom primeru, ovaj izraz izgleda ovako:

SELECT Ime, Prezime FROM Student;

Ime	Prezime
Petar	Petrović
Marko	Nenadić
Marko	Marković
Kosta	

Slika 6.3.4 Rezultat upita [Izvor: Autor]

Da bi ponovo bilo moguće postavljati upite, na liniji sa menijima potrebno je kliknuti na **View?SQL View**.



Slika 6.3.5 SQL View opcija [Izvor: Autor]

PRIMER 2 - IZBOR PODATAKA U TABELI

Za izbor podataka u tabeli koristi se WHERE koja se dodaje naredbi SELECT

Za izbor podataka iz tabele uz ispunjavanje određenih uslova, koristi se odrednica **WHERE** koja se dodaje **SELECT** naredbi na sledeći način:

SELECT ime_kolone FROM ime_tabele

WHERE kolona operator vrednost;

Napomena: kod nekih verzija SQL operator **<>** piše se kao **!=**.

<i>operator</i>	<i>Opis</i>
=	Jednako
<>	Nejednako
>	Veće od
<	Manje od
>=	Veće od ili jednako
<=	Manje od ili jednako
BETWEEN	Između vrednosti u određenom rasponu
LIKE	Traženje obrazaca

Slika 6.3.6 SQL operatori [Izvor: Autor]

Da bi se iz date tabele dobili podaci o studentma koji se zovu Marko koristi se sledeći upit:

SELECT * FROM Student

WHERE ime='Marko';

Znak "*" koristi se umesto navođenja svih kolona tabele. Rezultat izvršenja upita prikazan je na slici 7.

The screenshot shows a Microsoft Access query results grid titled 'Student'. The columns are labeled 'indeks', 'prezime', 'ime', and 'prosek'. There are two rows of data. The first row has 'indeks' value 2, 'prezime' value 'Nenadić', 'ime' value 'Marko', and 'prosek' value 8. The second row has 'indeks' value 3, 'prezime' value 'Marković', 'ime' value 'Marko', and 'prosek' value 9.5. A yellow asterisk (*) is in the first row's 'indeks' column. The 'prosek' column for the second row is highlighted with a red border.

Slika 6.3.7 Rezultat upita [Izvor: Autor]

Treći zahtev u primeru je da se pronađu studenti čiji je broj indeksa manji od 3. Svako pojavljivanje uslova u SQL jeziku podrazumeva korišćenje naredbe **WHERE**:

SELECT * FROM Student

WHERE indeks<3;

LOGIČKI OPERATORI

U WHERE delu upita moguće je koristiti logičke operatore za kombinovanje više uslova.

U **WHERE** delu upita moguće je koristiti logičke operatore za kombinovanje više uslova.

SQL podržava sledeće operatore:

- AND
- OR
- NOT
- XOR

Opšti oblik upita sa logičkim operatorima je:

SELECT ime_kolone FROM ime_tabele

WHERE kolona operator_poredjenja vrednost logički_operator kolona operator_poredjenja vrednost...;

Na primer:

SELECT * FROM Student

WHERE ime='Marko'

AND prezime='Mehić';

PRIMER 4

Za brisanje nekog podatka, odnosne vrste, u tabeli koristi se naredba **DELETE**

Zadatak (Vreme izrade zadatka: 5 minuta)

Iz tabele Student izbrisati studenta sa brojem indeksa 2, a potom ostale studente prikazati poređane po abecednom redu u odnosu na prvo slovo prezimena.

Rešenje:

Za brisanje nekog podatka, odnosne vrste, u tabeli koristi se naredba **DELETE** u sledećem obliku:

DELETE FROM ime_tabele

WHERE ime_kolone = vrednost;

Za ovaj primer to znači sledeće:

DELETE FROM Student

WHERE indeks = 2;

Rezultujuća tabela je data na slici 8.

indeks	prezime	ime	prospekt
1	Petrović	Petar	7.56
3	Marković	Marko	9.5
13	Kosta		
*			

Slika 6.3.8 Tabela sa obrisanim redom [Izvor: Autor]

Da bi se preostali podaci poređali po abecednom redu u odnosu na prvo slovo prezimena koristi se naredba:

SELECT prezime, ime FROM Student

ORDER BY prezime;

Modifikatori **ASC** i **DESC** se mogu koristiti da sortiraju rezultat upita po rastućem i opadajućem redosledu. Na primer,

SELECT prezime, ime FROM Student

ORDER BY prezime DESC;

Rezultat je tabela data na slici 9.

prezime	ime
Petrović	Petar
Marković	Marko
Kostić	

Slika 6.3.9 Sortiran izlaz upita [Izvor: Autor]

PRIMER 5

Ovaj primer demonstrira korišćenje operatora u SQL-u

Zadatak (Vreme izrade zadatka: 5 minuta)

Formirati tabelu **Osoba** i popuniti je podacima prikazanim na slici 10. Pronaći podatke o svim osobama sa imenom Katarina u tabeli, koje žive u Novom Sadu. Potom, pronaći podatke o svim osobama koji se zovu Dejan ili žive u Beogradu.

prezime	ime	grad	starost
Milanović	Katarina	Novi Sad	30
Stamenković	Andelija	Beograd	24
Milošević	Joksim	Novi Sad	45
Jovanović	Dejan	Niš	32
Arambašić	Dejan	Beograd	47

Slika 6.3.10 Primer podataka [Izvor: Autor]

Rešenje:

Bilo kakvo pronalaženje nekog podatka u tabeli podrazumeva korišćenje izraza:

SELECT ... WHERE ...

Ukoliko je prilikom traženja nekog podatka potrebno zadovoljiti dva ili više uslova koriste se logički operatori. U ovom slučaju potrebno je zadovoljiti istovremeno dva uslova, pa se koristi operator **AND** koji podatak bira samo ako su svi uslovi koje povezuje zadovoljeni.

Za prvi deo ovog primera to podrazumeva sledeću naredbu:

SELECT * FROM Osoba

WHERE ime='Katarina' AND grad='Novi Sad';

Table2 Query					Osoba
prezime	ime	grad	starost		
Milanović	Katarina	Novi Sad	30		
*					

Slika 6.3.11 Rezultat AND upita [Izvor: Autor]

Ukoliko se javlja uslov *ili* koristi se operator **OR**. Tada je za pronalaženje podatka dovoljno da je barem jedan od uslova koje operator povezuje zadovoljen. Za ovaj primer naredba izgleda ovako:

SELECT * FROM Osoba

WHERE ime='Joksim' OR grad='Beograd';

a rezultat je dat na slici 12.

Table2 Query					Osoba
prezime	ime	grad	starost		
Stamenković	Anđelija	Beograd	24		
Milošević	Joksim	Novi Sad	45		
Arambašić	Dejan	Beograd	47		
*					

Slika 6.3.12 Rezultata OR upita [Izvor: Autor]

PRIMER 6

Primena naredbe IN

Zadatak (Vreme izrade zadatka: 5 minuta)

Za tabelu iz prethodnog primera pronaći sve osobe koje žive u Novom Sadu ili Nišu.

Rešenje:

Osim korišćenja naredbe OR postoji i mogućnost korišćenja naredbe IN sledeće sintakse:

SELECT ime_kolone FROM ime_tabele

WHERE ime_kolone IN (vrednost1,vrednost2,..);

Za zadati primer SQL upit izgleda ovako:

SELECT * FROM Osoba

WHERE grad IN ('Novi Sad','Niš');

Rezultat primene upita je dat na slici 14.

prezime	ime	grad	starost
Milanović	Katarina	Novi Sad	30
Milošević	Joksim	Novi Sad	45
Jovanović	Dejan	Niš	32
*			

Slika 6.3.13 Rezultat IN upita [Izvor: Autor]

PRIMER 7

Za pronalaženje podataka koji se nalaze između dve određene vrednosti koristi se BETWEEN...AND operator

Zadatak (Vreme izrade zadatka: 5 minuta)

Za primer tabele **Osoba** pronaći sve podatke za osobe čija je starost između 30 i 45 godina.
Rešenje:

Za pronalaženje podataka koji se nalaze između dve određene vrednosti koristi se **BETWEEN...AND** operator i to na sledeći način:

SELECT ime_kolone FROM ime_tabele

WHERE ime_kolone BETWEEN vrednost1 AND vrednost2;

Za ovaj primer može se pripremiti sledeći upit:

SELECT * FROM Osoba

WHERE starost BETWEEN 30 AND 45;

Rezultat ovako primjenjenog upita je prikazan na slici 15.

prezime	ime	grad	starost
Milanović	Katarina	Novi Sad	30
Milošević	Joksim	Novi Sad	45
Jovanović	Dejan	Niš	32
*			

Slika 6.3.14 Rezultat BETWEEN upita [Izvor: Autor]

Važno je napomenuti da **BETWEEN...AND** operator daje različite rezultate u različitim sistemima. Kod nekih, podaci o osobama čija su prezimena navedena u upitu ne bi se pojavili. Kod drugih sistema postoji mogućnost da se pored podataka između zadatih parametara, pojave i podaci vezani za prvo i/ili drugo prezime navedeno u operatuoru. U svakom slučaju, poželjno je pre korišćenja proveriti kako sistem reaguje na ovaj operator.

Takođe, operator se može koristiti i uz dodatnu negaciju što omogućava pronalaženje podataka koji se nalaze izvan gore dobijene oblasti . Operator se primenjuje na sledeći način:

SELECT * FROM Osoba

WHERE starost

NOT BETWEEN 30 AND 45;

prezime	ime	grad	starost
Stamenković	Anđelija	Beograd	24
Arambašić	Dejan	Beograd	47
*			

Slika 6.3.15 Rezultat NOT BETWEEN upita [Izvor: Autor]

PRIMER 8

*Za pronalaženje podataka može se koristiti LIKE uz operator **

Zadatak (Vreme izrade zadatka: 5 minuta)

U tabeli **Osoba** pretražiti podatke i pronaći sve osoba čije se prezime završava na "ović", zatim počinje sa "Mi" i konačno, one čije prezime sadrži "am".

Rešenje:

Za pronalaženje podataka po ovom kriterijumu koristi se operator LIKE i to u sledećem obliku:

SELECT ime_kolone FROM ime_tabele

WHERE ime_kolone LIKE '*vrednost';

Za ovaj primer to znači:

SELECT * FROM Osoba

WHERE prezime LIKE '*ović';

prezime	ime	grad	starost
Milanović	Katarina	Novi Sad	30
Stamenković	Anđelija	Beograd	24
Jovanović	Dejan	Niš	32
*			

Slika 6.3.16 Rezultat LIKE upita [Izvor: Autor]

Na sličan način, sledećim upitom može se doći do podataka o svim osobama čije prezime počinje sa „Mi“:

SELECT * FROM Osoba

WHERE prezime LIKE 'Mi*'

Za sve one osoba koje bilo gde u prezimenu sadrže slova „am“ koristi se sledeći upit:

SELECT * FROM Osoba

WHERE prezime LIKE '*am*'

The screenshot shows a Microsoft Access query window titled "Table2 Query". The table has four columns: "prezime", "ime", "grad", and "starost". There are two rows of data: one for Stamenković Andelija from Beograd at age 24, and another for Arambašić Dejan from Beograd at age 47. A yellow box highlights the last row of the table.

prezime	ime	grad	starost
Stamenković	Andelija	Beograd	24
Arambašić	Dejan	Beograd	47
*			

Slika 6.3.17 Rezultat LIKE upita [Izvor: Autor]

PRIMER 9

Nad tabelom Radnik (id, ime, prezime, odeljenje), izvršiti upite:

Zadatak (Vreme izrade zadatka: 5 minuta)

Nad tabelom Radnik (id, ime, prezime, odeljenje, plata), izvršiti sledeće upit:

Naći ime i prezime svih radnika u IT odeljenju čija je plata veća od 1200 EUR.

SELECT ime, prezime FROM Radnik

WHERE plata > 1200 AND odeljenje = "IT";

Naći ime, prezime i platu svih radnika čije se ime završava na slovo „J“ i počinje na slovo „A“.

SELECT ime, prezime, plata FROM Radnik

WHERE ime LIKE "A*J";

OSNOVNE KOMANDE DML JEZIKA - VIDEO

Osnovne komande DML jezika

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 6.3 Zadaci za samostalni rad

OPIS ZADATKA ZA SAMOSTALNU VEŽBU

Kreirati bazu podataka Bioskop i tabelu Film

Predviđeno vreme za izradu zadatka je 40 minuta.

Korak 1: Kreirati bazu podataka Bioskop i tabelu Film.

Korak 2: Definisati koje atribute tabela Film treba da poseduje, pritom vodeći računa da možete uraditi upite.

Korak 3: Uneti minimalno 10 redova.

Korak 4: Napisati upit koji će:

- naći sve filmove čije ime počinje na slovo P
- pronaći sve filmove koje je režirao Gaj Hamilton
- ispisati sve filmove koje su osvojili više od 2 oskara
- pronaći filmove koje su izdati posle 1990 godine.
- pronaći sve filmove koji su izdati u svim zemljama osim Amerike
- pronaći sve glumce koji su glumili u filmu "Vertigo"
- pronaći sve režisere čije ime počinje na slovo A
- pronaći najnoviji film
- pronaći najstariji film
- pronaći sve glumce koji su glumili u dva ili više filmova

✓ Poglavlje 7

Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ02

Kreirati bazu i napisati upite prema datim specifikacijama

Predviđeno vreme izrade domaćeg zadatka je: 45 minuta.

1. Kreirati bazu podataka bolnica_db. U njoj kreirati potrebnu tabelu (ili tabele) i kolone kako bi se mogli uraditi sledeći upiti:
2. Pronaći sve zaposlene koji nisu iz Srbije
3. Pronaći sve zaposlene koji su glavni na određenom odeljenju
4. Pronaći sve pacijente koji imaju zakazan pregled u narednih 7 dana
5. Pronaći prezimena doktora čiji pacijenti imaju imena koja počinju na slovo L
6. Pronaći sve zaposlene koji imaju između 27 i 35 godina
7. Pronaći ukupni broj pregleda u proteklih 7 dana

Zadatak poslati kao Access file sa urađenim zadacima.

Dokument snimiti po imenom **IT210-DZ02-Ime_Prezime_brojIndexa**, gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovan dokument poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-majla napisati IT210 – DZ02)

▼ Zaključak

ZAKLJUČAK

U ovoj lekciji bilo je reči o osnovnim konceptima koji se odnose na upravljanje informacijama. U okviru ove lekcije važno je spoznati važnost podataka, informacija i znanja, ali i pitanja u vezi njihovog korišćenja, kao i njihovog životnog ciklusa. Kao sastavni deo informacionog sistema, baza podataka igra važnu ulogu, pa treba znati objasniti prednosti korišćenja baze podataka, kao i proceduru skladištenja podataka i ulogu skladišta podataka u podršci u odlučivanju.

Literatura

1. R. Kelly Rainer Jr., Erfaim Turban, Uvod u informacione sisteme, drugo izdanje, Wiley, 2009.
2. Goujun Lu, Multimedia Database management systems, Artech House, 1999.
3. Branislav Lazarević, Zoran Marjanović, Nenad Aničić, Slađan Babarogić, Baze podataka, FON, Beograd, 2004.
4. Michael Abbey, Michael Corey, Ian Abramson, Osnove Oracle 8i, Kompjuter biblioteka, Čačak, 2001.
5. Paul Dorsey, Peter Koletzke, Oracle JDeveloper 3, Kompjuter biblioteka, Čačak, 2002.
6. Craig Mullins, Administracija baza podataka, Kompjuter biblioteka, Čačak, 2003.
7. Ralph Stair, Principles of Information Systems, Boyd & Fraser, 1992.



IT210 - SISTEMI IT

ARHITEKTURA ORGANIZACIJE
PODATAKA I MODELIRANJE
PODATAKA

Lekcija 03

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 03

ARHITEKTURA ORGANIZACIJE PODATAKA I MODELIRANJE PODATAKA

- ✓ ARHITEKTURA ORGANIZACIJE PODATAKA I MODELIRANJE PODATAKA
- ✓ Poglavlje 1: ARHITEKTURA ORGANIZACIJE PODATAKA
- ✓ Poglavlje 2: MODELIRANJE PODATAKA
- ✓ Poglavlje 3: NORMALNE FORME
- ✓ Poglavlje 4: IDEF METODI ZA MODELIRANJE
- ✓ Poglavlje 5: MODELIRANJE BAZE PODATAKA UZ POMOĆ UML-a
- ✓ Poglavlje 6: Pokazna vežba: MODELIRANJE PODATAKA
- ✓ Poglavlje 7: Pokazna vežba: SQL AGREGATNE FUNKCIJE
- ✓ Poglavlje 8: Zadaci za samostalni rad
- ✓ Poglavlje 9: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da vas uvede u osnove modeliranje i arhitekturu podataka koji su kasnije osnova za dalji rad na bazama podataka

U ovoj lekciji biće obrađene dve glavne teme, a to su arhitektura organizacija podataka i modeliranje podataka. U okviru ove lekcije ćemo:

- Dati definiciju modela podataka i listu mogućih modela
- Opisati osobine relacionog modela uključujući relacije, zapise, atribute, domene i operatore
- Opisati operacije select, project, union, intersection, difference, i natural join
- Opisati razliku između funkcionalnih zavisnosti i ključeva
- Objasniti 1, 2 i 3 normalnu formu
- Opisati razliku između različitih modela baze podataka
- Projektovanje i interpretacija ER dijagrama
- Korišćenje IDEF1 i UML za modeliranje baza podataka

U ovoj lekciji biće obrađene sledeće teme:

- Arhitektura organizacije podataka

- Modeli podataka
- Relacioni model
- Normalne forme
- Funkcionalne zavisnosti
- 1NF
- 2NF
- 3NF

- Modeliranje podataka

- Konceptualni model
- Entity Relationship dijagrami
- Logički modeli
- Fizički modeli
- IDEF1X, UML

▼ Poglavlje 1

ARHITEKTURA ORGANIZACIJE PODATAKA

MODELI PODATAKA

Model podataka definiše skup objekata, njihovih atributa i međusobnih veza koji su potrebni i dovoljni za funkcionisanje informacionog sistema

Svaki informacioni sistem, odnosno baza podataka na kome se on zasniva, ima svoju namenu, odnosno domen poslovnog sistema koga podržava. Realni poslovni sistemi se karakterišu beskonačnim brojem atributa, od kojih mnogi nisu relevantni za posmatrani informacioni sistem. Izrada informacionog sistema koji je zadužen da pruži podršku poslovanju u nekom domenu poslovnog sistema podrazumeva da se kreira model informacionog sistema. **Model informacionog sistema** se dobija apstrakcijom svih podataka koji se koriste u nekom domenu poslovnog sistema. Pri tom se definiše model podataka nad kojim će informacioni sistem funkcionisati.

Model podataka definiše skup objekata, njihovih atributa i međusobnih veza koji su potrebni i dovoljni za funkcionisanje informacionog sistema. Model podataka je konceptualni opis domena problema. Model uključuje: **entitete, njihove atribute, ograničenja atributa, veze i ograničenja veza**.

Tokom istorije razvoja SUBP (sistema za upravljanje bazama podataka) su se koristili različiti modeli podataka nad kojima su se gradili sistemi za upravljanje bazama podataka.

Prvi SUBP opšte namene **Integrated Data Store** je projektovao Charles Bachman iz General Electric ranih šezdesetih godina prošlog veka. Ovaj sistem je bio baziran na **mrežnom modelu** podataka koji je bio standardizovan od strane **Conference on Data Systems Languages** (CODASYL) i snažno je uticao na sve baze koje su razvijane tokom šezdesetih. Krajem šezdesetih godina prošlog veka IBM je razvio sistem za upravljanje podacima **Information Management System** (IMS) koji je bio baziran na **hijerarhijskom modelu podataka**. Godine 1970. Edgar Codd iz IBM-a je predložio novi metod za reprezentaciju podataka koji je nazvan **relacioni model**. Ovaj model je bio mnogo podesniji za modeliranje podataka, pa je od 1980. pa nadalje postao opšte prihvaćen i danas najmasovnije korišćen. Razvoj objektno-orientisane paradigme u programiranju doprineo je i pojavi **objektno-orientisanih modela baza podataka**.

Dakle, baze podataka se prema modelu podataka koje koriste mogu podeliti na sledeće kategorije: **hijerarhijske, mrežne, relacione, objektno orijentisane, objektno - relacioni**.

Obzirom da se relacioni model podataka danas koristi kod većine SUBP, ovde će biti reči samo o njemu.

✓ 1.1 RELACIONI MODEL BAZE PODATAKA

OSNOVNI ELEMENTI RELACIONE BAZE PODATAKA

Entiteti, atributi, ograničenja, relacije, zapis, ključ

Osnovni elementi relacione baze podataka su:

- entiteti
- atributi
- veze ili relacije
- zapis
- jedinstveni identifikator ili ključ.

ŠTA JE ENTITET?

Entitet je bilo koja realna ili imaginarna klasa objekata ili pojmove o kojima se podaci skupljaju, memorišu i održavaju

Relacioni model baze podataka se zasniva na entitetima. **Entitet** je bilo koja realna ili imaginarna klasa objekata ili pojmove o kojima se podaci skupljaju, memorišu i održavaju. Entitet je:

- "Nešto" od značaja za biznis za koji je potrebno da podaci budu poznati
- Naziv za skup sličnih stvari koje se mogu nabrojati
- Obično imenica

Primeri entiteta su: predmeti, događaji, ljudi.

Za realne objekte je karakteristično da oni fizički postoje. Na primer, realni objekti su: osobe, preduzeća, mašine, artikli, prodavnice. Suprotno tome, entiteti su često i imaginarni objekti ili pojmovi koji postoje samo konceptualno. Primeri imaginarnih mogu da budu: troškovi, prodaja, ispit, itd.

ENTITET I INSTANCE

Entitet ima instance, a instanca predstavlja pojedinačni slučaj entiteta

Entitet ima instance, a instanca predstavlja pojedinačni slučaj entiteta.

Entitet — — - - - Instanca

- **OSOBA** — — - Ljubica Dragić, Ivan Vukić
- **PROIZVOD** — — - Nike Air Max, Opel Astra
- **TIP PROIZVODA** — — - cipela, video igre
- **POSAO** — — - električar, mehaničar, IT inženjer
- **NIVO VEŠTINA** — — - početni, napredni
- **KONCERT** — — - U2 u Kombak Areni, Beyonce u NY
- **ŽIVOTINJA** — — - pas, mačka
- **AUTO** — — - Toyota Corolla, Opel Astra

Dalmatinac, sijamska mačka, krava i koza su instance entiteta **ŽIVOTINJA**. Neki entiteti imaju mnogo instanci, a neki samo nekoliko. Instances mogu biti:

- nešto opipljivo, kao što su OSOBA ili PROIZVOD
- nešto neopipljivo, kao što su NIVO VEŠTINA
- događaj, kao što je koncert.

Da li je PAS instance ili entitet?

Odgovor je da zavisi:

- ako uzmemo u obzir mnogo različitih vrsta životinja, onda ima smisla da razmišljamo o entitetu **ŽIVOTINJA** i da uključimo njegove instance kao što su konj, pas, mačka
- ako uzgajamo pse, onda je potrebno da čuvamo podatke o različitim rasama pasa i tada je PAS entitet, a njegove instance bi na primer bile terijer, pudlica, labrador i sl.

ŠTA JE ATRIBUT?

Atribut je specifičan deo informacija koji pomaže da se bliže opiše entitet

Entiteti se u modelu opisuju preko svojih svojstava koji se nazivaju **atributi**. Atributi predstavljaju izabrana svojstva objekta koji je predstavljen entitetom. Kao i entitet, **atribut predstavlja nešto od značaja za poslovanje**.

Atribut je specifičan deo informacija koji pomaže da se:

- opiše entitet
- kvantifikuje entitet
- kvalifikuje entitet
- bliže odredi entitet.

Atributi mogu da imaju **ograničenja** koja limitiraju njihove vrednosti. Ovakva ograničenja se nazivaju staticka pravila integriteta baze podataka. Između entiteta postoje **veze** ili **relacije** koje takođe mogu da imaju svoja ograničenja.

Na primer, podaci o osobama u nekoj bazi podataka predstavljaju jedan entitet. Osobe imaju svoje attribute koji mogu biti: ime prezime, matični broj, pol. Svaki atribut može imati jedno ili više ograničenja. Na primer, pol može uzimati samo vrednosti muški ili ženski. Entitet osobe može, na primer, biti povezan sa entitetom radna mesta.

Atributi mogu biti prosti ili složeni. **Prost atribut** ima samo jednu komponentu koja se dalje ne može deliti. Ovakvi atributi se nazivaju atomični. Na primer, atribut godina_rođenja je prost atribut. **Složeni atribut** se sastoji od više komponenti koje je moguće podeliti. Neka na primer, atribut kupac u nekoj bazi podataka može da uzme vrednost: Zoran Petrović. Ovaj atribut ima dve komponente: ime i prezime kupca.

VREDNOST ATRIBUTA

Vrednost atributa može biti broj, karakter, datum, slika, zvuk i slično

Atribut ima jednu vrednost. Vrednost atributa može biti broj, karakter, datum, slika, zvuk i sl. Ovo se naziva tipom podataka ili formati. Svaki atribut čuva jedan podatak tačno jednog tipa podataka.

Entiteti — — - - - Atributi

- **MUŠTERIJA** — — - - - prezime, godine, veličina cipele, mesto prebivališta, email
- **AUTO** — — - - - model, težina, cena
- **PORUDŽBINA** — — - - - datum porudžbine, datum isporuke
- **POSAO** — — - - - naziv, opis
- **TRANSAKCIJA** — — - - - datum

Neki atributi moraju da imaju vrednosti - to su obavezni atributi. Na primer, u većini biznisa koje prate lične informacije, potrebno je da postoji podatak *ime*. Atributi koji nisu obavezni se nazivaju opcioni. Na primer: broj telefon je često opcion, osim recimo kod mobilnih aplikacija.

ZAPISI I PRIMARNI KLJUČ

Atribut koji jedinstveno razlikuje jedan zapis od drugih se naziva primarni ključ

U slučaju relacionih baza podataka svakom entitetu odgovara jedna tabela. Tabela se sastoji od niza zapisa. Jedan **zapis** ili **slog** predstavlja jednu vrstu u tabeli i sadrži podatke koji opisuju jednu instancu klase koju opisuje entitet. Svaki zapis se sastoji od polja u kojima su upisane vrednosti atributa. Jedno polje odgovara jednom atributu objekta. Tabela ima onoliko kolona koliko klasa ima atributa, i onoliko vrsta koliko se instanci te klase čuva u bazi.

Zapisi u tabeli se međusobno moraju razlikovati bar po vrednosti jednog atributa. **Atribut koji jedinstveno razlikuje jedan zapis od drugih se naziva primarni ključ**. Primarni ključ se takođe naziva i **jedinstveni identifikator** (engl.**unique identifier (UID)**)

ZAPOSLENI ima jedinstveni identifikator. Jedinstveni identifikator može biti jedan atribut ili više atributa koji jedinstveno izdvajaju svakog zaposlenog. Kako možemo pronaći specifičnog zaposlenog koji radi u kompaniji? Koji podatak jedinstveno identificuje jednog **ZAPOSLENOG**?

Razmotrimo studente na predavanju. Svaki student je opisan sa nekoliko karakteristika ili atributa. Koji atributi mogu da omoguće da se identificuje jedan student od ostatka grupe? To je najverovatnije indeks studenta i koristili bi ga kao njegov identifikator.

RELACIONA BAZA PODATAKA - PRIMER

Primer tabele Studenti koja koristi broj indeksa kao primarni ključ

U okviru neke baze podataka postoji entitet Studenti. Ovaj entitet opisuje klasu osoba koje se nazivaju studenti. Svaki student predstavlja jednu instancu klase studenti. Ova klasa ima u odnosu na druge klase posebna svojstva koja su opisana nekim atributima. Entitet studenti je opisan sledećim atributima:

Studenti (BrojIndeksa, Ime, Prezime, JMBG, DatumRodjenja, GodinaUpisa)

Tabela u kojoj se nalaze podaci o studentima nazvana je Studenti. Obzirom da se svakom studentu dodeljuje indeks sa jedinstvenim brojem, za primarni ključ je izabrano polje BrojIndeksa. Na sledećoj slici je dat primer ove tabele sa 3 zapisa, dakle sa podacima za 3 studenta. Svakom studentu odgovara jedna vrsta u tabeli, odnosno jedan zapis. Kolone tabele nose nazine atributa. Svaki zapis ima po 6 polja.

Studenti : Tab					
BrojIndeksa	Ime	Prezime	JMBG	DatumRodjenja	GodinaUpisa
124	Zoran	Kostić	1504971730052	15.4.1971	2003
199	Ana	Jetić	0307972730055	3.7.1972	2004
215	Slavko	Stojanović	2412973730014	24.12.1973	2005
0					

Slika 1.1.1 Primer tabele "Studenti" [Izvor: Autor]

PRIMER - ENTITETI I ATRIBUTI - POSTAVKA ZADATKA

Cilj ovog primera je da napravite razliku između entiteta i atributa

Postoje tri entiteta koja definišu posao jednog DJ-a: PESMA, DOGAĐAJ, KLIJENT. Ova tri entiteta su data u prve tri kolone u tabeli niže. Četvrta kolona sadrži kolekciju atributa. Odredite koji od atributa može da pripada kom entitetu. Na primer, da li Naslov može biti atribut PESME, DOGAĐAJA i KLIJENTA?

PESMA	DOGAĐAJ	KLIJENT	
			Naslov
			Opis
			Mesto održavanja
			Ime
			Broj telefona
			Datum izlaska
			Prezime
			Tip
			email adresa

Slika 1.1.2 Tabela za određivanje atributa [Izvor: Autor]

PRIMER - ENTITETI I ATRIBUTI - REŠENJE

Moguće rešenje zadatka - X označava da atribut može pripadati tom entitetu

Podsetimo se, prve tri kolone predstavljaju entitete PESMA, DOGAĐAJ i KLIJENT. Četvrta kolona predstavlja moguću listu atributa. Znakom X označeno je moguće rešenje da određeni atribut pripada datom entitetu

PESMA	DOGAĐAJ	KLIJENT	
X			Naslov
X	X		Opis
	X		Mesto održavanja
		X	Ime
		X	Broj telefona
X			Datum izlaska
		X	Prezime
X	X		Tip
		X	email adresa

Slika 1.1.3 Rešenje za atribut entiteta [Izvor: Autor]

PRIMER - PRIMARNI KLJUČ

Za svaki od entiteta odredite atribut koji može da bude primarni ključ

Za svaki od entiteta odredite atribut koji može da bude jedinstveni identifikator, odnosno primarni ključ

Entitet: STUDENT

Atributi: **student ID, ime, prezime, adresa**

UID: student ID

Entitet: **FILM**

Atributi: ***naziv, datum izlaska, producent, reditelj***

UID: kombinacija naziva i datuma izlaska, ili bi se pravio novi veštački UID kao što bi bio film ID

Entitet: ORMARIC

Atributi: ***veličina, lokacija, broj***

UID: broj

RELACIJE

Odnosi između entiteta se nazivaju relacije

Mogućnost identifikacije odnosa između entiteta olakšava razumevanje veza između različitih podataka. Odnosi između entiteta se nazivaju relacije. Na primer, entiteti STUDENT i PREDMET su međusobno povezani. Relacije pomažu da se vidi kako različiti delovi sistema utiču jedni na druge. Prilikom modelovanja nekog poslovanja, odnosi između entiteta su toliko važni kao i sami entiteti.

Relacije:

- Predstavljaju nešto od značaja ili važnosti za posao
- Pokazuju kako su entiteti povezani jedni sa drugima
- Postoje samo između entiteta (ili jednog entiteta sa samim sobom)
- su dvosmerne
- imaju naziv na oba kraja
- Imaju optionalnost
- Imaju kardinalnost

OPCIJONOST I KARDINALNOST RELACIJE

Relacije mogu biti obavezne ili opcione. Kardinalnost određuje stepen povezanosti dva entiteta dajući odgovor na pitanje: "Koliko?"

Relacije mogu biti obavezne ili opcione. Razmotrimo dva entiteta **ZAPOSLENI** i **POSAO**. U zavisnosti šta nam je poznato za ova dva entiteta, možemo odrediti optionalnost tako što ćemo odgovoriti na sledeća pitanja:

Da li svaki zaposleni mora da ima posao?

- drugim rečima, da li je ovo obavezna ili optionalna relacija za zaposlenog?

Da li svaki posao mora da obavlja neki zaposleni?

- drugim rečima, da li je ovaj posao obavezna ili optionalna relacija za posao?

Šta je kardinalnost relacija?

Kardinalnost meri količinu nečega. Kod relacija, kardinalnost određuje stepen povezanosti dva entiteta dajući odgovor na pitanje: "Koliko?". Na primer:

- Koliko radnih mesta može imati jedan radnik? Samo jedno radno mesto? Ili više od jednog radnog mesta?
- Koliko zaposlenih može imati jedan konkretni posao? Samo jedan zaposleni? Ili više od jednog zaposlenog?

Primeri:

- Svaki **ZAPOSLENI** mora da ima **jedan i samo jedan POSAO**
- Svaki **POSAO** može da radi **više ZASPOLENIH**
- Svaki **PROIZVOD** mora da bude klasifikovan kao **jedan i samo jedan TIP PROIZVODA**
- Svaki **TIP PROIZVODA** može da klasificiše **jedan ili više PROIZVODA**

▼ 1.2 FUNKCIONALNE ZAVISNOSTI

KADA SU ATRIBUTI FUNKCIONALNO ZAVISNI?

Za atribut Y se kaže da je funkcionalno zavisan od atributa X, ako i samo ako svakoj vrednosti atributa X odgovara samo jedna vrednost atributa Y

Funkcionalna zavisnost atributa entiteta je vrsta ograničenja integriteta koja generalizuju koncept ključa. Definicija druge, treće i Boyce-Codd-ove normalne forme su zasnovane na konceptu funkcionalne zavisnosti, pa će se ovde dati njena definicija.

Posmatrajmo entitet R koji ima više atributa. Neki atributi mogu biti složeni što znači da se sastoje od više prostih atributa. Pretpostavimo da u entitetu R postoje atributi X i Y, koji mogu biti i složeni. Za atribut Y se kaže da je funkcionalno zavisan od atributa X, ako i samo ako svakoj vrednosti atributa X odgovara samo jedna vrednost atributa Y. Ova funkcionalna zavisnost se simbolički predstavlja kao:

$$R.X \rightarrow R.Y$$

što se čita kao: Y je funkcionalno zaviso od X, ili X funkcionalno određuje Y. Na primer, u prethodnoj tabeli Studenti, svakoj vrednosti atributa BrojIndeksa odgovara samo jedna vrednost atributa JMBG, pa se može reći da je JMBG funkcionalno zavisan od BrojIndeksa.

Pored funkcionalne zavisnosti za definisanje normalizacije potrebno je upoznati se i sa konceptom tranzitivne funkcionalne zavisnosti. U entitetu R koji ima attribute X, Y, Z, atribut Z je tranzitivno zavisan od atributa X ako je:

$$R.X \rightarrow R.Y \quad R.Y \rightarrow R.Z \quad R.X \rightarrow R.Z \quad R.Y \dashrightarrow R.X \quad R.Z \dashrightarrow R.X$$

Drugim rečima atribut Z je tranzitivno funkcionalno zavisан od atributa X ako je funkcionalно zavisан od X i nekog atributa Y koji je takođe funkcionalно zavisан od X.

✓ Poglavlje 2

MODELIRANJE PODATAKA

KONCEPTUALNI MODEL

Konceptualni model podataka sadrži opis objekata, njihovih atributa i veze između njih

Konceptualni model opisuje karakteristike informacionog sistema za koji se projektuje baza podataka. Konceptualni model podataka sadrži opis entiteta, njihovih atributa i veze između njih. On nije šema baze podataka koja opisuje fizički izgled tabela. Konceptualni model podataka je poslovno-orientisan pogled na informacije. Ovaj model je dosta visokog nivoa i često ne obuhvata sve detalje, ali se zato odlikuje jednostavnosću i jasnoćom.

Konceptualni model je nezavisan od modela podataka koji će se implementirati i sistema za upravljanje bazama podataka. Proces definisanja konceptualnog modela se sastoji od sledećih faza:

1. Identifikovanje objekata i njihovih atributa u prostoru problema

2. Definisanje veza

- Kardinalnost veza
- Opcionalnost veza
- Atributi veza
- Ograničenja veza

3. Detaljno definisanje objekata

- Veze između objekata i prostora problema
- Radni proces koji kreira, modifikuje, koristi i briše instance objekata
- Poslovna pravila i ograničenja koja se odnose na objekte
- Lista atributa

4. Definisanje domena atributa

- Tip podataka atributa
- Dozvoljena oblast vrednosti atributa
- Format podataka za unos atributa

Svi podaci koji su definisani tokom ovog procesa predstavljaju konceptualni model.

Jedna od tehnika za prikaz konceptualnih modela baza podataka su E/R dijagrami.

NAČINI MODELOVANJE KONCEPTUALNOG MODELA

Metoda strukturne i sistemske analize, opis scenarija korišćenja, korišćenje paterna, normalizacija relacija, direktno i reverzno inženjerstvo

Za definisanje konceptualnog modela se koriste različiti intelektualni alati za modeliranje, kao što su:

- **Integracija podmodela dobijenih dekompozicijom sistema metodom strukturne sistemske analize**
- **Direktno modelovanje na bazi verbalnog opisa sistema dobijenog iz opisa scenarija slučajeva korišćenja**
- **Konkretizacija opštih modela i korišćenje uzorka (paterna)**
- **Normalizacija relacija**
- **Transformacija iz jednog modela u drugi (direktno i reverzno inženjerstvo).**

Prva tri alata se koriste za izgradnju modela objekti – veze u konvencionalnim metodama ili dijagrama klase u objektnim metodama. Metod normalizacije relacija se koristi za relacione modele.

▼ 2.1 E/R model

OSNOVNI KONCEPTI E/R MODELA

U ovom modelu objekti imaju svoja svojstva koja se iskazuju preko atributa i veza prema drugim objektima

Model objekti - veze ([E/R model](#) – Entity - Relationship model) je najpopularniji i najčešće korišćeni model za projektovanje baza podataka, pre svega relacionih. U ovom modelu entiteti imaju svoja svojstva koja se iskazuju preko atributa i veza prema drugim objektima. Postoji više različitih verzija ovih modela, koji poseduju specifične koncepte za opis strukture sistema. Model entiteti veze se obično koristi u fazi modelovanja realnog sistema, odnosno projektovanja baze podataka, a potom se transformiše u neki drugi najčešće relacioni model, za koga postoji odgovarajući sistem za upravljanje bazom podataka.

Model objekti - veze koristi tri osnovna koncepta:

- **Entitet koji je od interesa za bazu**
- **Atribut, koji predstavljaju neku karakteristiku određenog entiteta**
- **Veza, odnosno relacija između dva entiteta**

ODNOS IZMEĐU OBJEKTA, ATRIBUTA I DOMENA

Atributi označavaju svojstva objekata koji su uzeti iz domena, koji predstavlja skup mogućih vrednosti

Stanje nekog entiteta se u realnosti može opisati sa beskonačnim brojem atributa. Međutim, prilikom stvaranja neke šeme projektant se ograničava na one atribute koji su relevantni za posmatrani informacioni sistem.

Atributi opisuju svojstva objekta u nekom trenutku vremena.

Atributi uzimaju vrednost iz skupa mogućih vrednosti, koje nazivamo domenima. Domeni mogu biti:

- predefinisani tipom promenljive koja je dodeljena atributu (na primer Text, Date, Time, Integer)
- semantički, odnosno korisnički definisani (na primer atribut Pol može uzeti vrednost M ili Ž)

Na sledećoj slici je korišćenjem simbola modela objekti-veze prikazan objekat **PolazniciKursa** koji ima atribute: *MatičniBroj*, *Ime*, *Prezime*, *StručnaSprema*, *Jezici*. Svi atributi, osim atributa *Jezici*, su jednoznačni. Na primer, jedan polaznik kursa može da ima samo jedno ime ili samo jedan matični broj. Međutim, polaznik kursa može da zna više jezika, pa je atribut *Jezici* višežnačan.

Jedan entitet može da ima različite instance. Na primer, entitet PolazniciKursa može da ima instance: Petar Petrović, Marko Marković, itd.

Zahteva se da atributi, osim što čine entitet i što ih identifikator funkcionalno određuje, ne mogu imati nikakvu drugu međusobnu vezu. Ključ je atribut, ili skup atributa, čija vrednost jednoznačno određuje pojedinačne instance entiteta. Na primer, za entitet PolazniciKursa kandidat za ključ je atribut matični broj. Dve različite instance objekta ne mogu imati isti ključ, jer ključ mora biti jedinstven za svaku instancu objekta.



Slika 2.1.1 Objekat PolazniciKursa prikazan E/R modelom [Izvor: Autor]

VEZE (RELACIJE)

Relacije predstavljaju međusobni odnos između objekata i mogu biti 1:1, 1:N, N:M

Veza ili relacija predstavlja međusobni odnos entiteta. Veze imaju opcionalnost i kardinalnost. Opcionalnost veza podrazumeva da veza može biti obavezna ili opciona. Posmatrajmo dva

entiteta ZAPOSLENI i POSAO, opcionalnost veze se može odrediti odgovorom na sledeća pitanja:

- Da li svaki zaposleni ima posao? - Drugim rečima, da li je ovo obavezna ili opciona relacija za zaposlenog?
- Da li svaki posao mora da bude dodeljen jednom zaposlenom? - Drugim rečima, da li je ovo obavezna ili opciona veza za posao?

Kardinalnost meri kuantitet nečega. U relaciji, kardinalnost određuje stepen koliko je entitet povezan sa drugim entitetom. Ovo možemo odrediti odgovorom na pitanje "Koliko?"

Na primer:

- Koliko poslova može da obavlja zaposleni? Samo jedan posao? Ili više poslova?
 - Koliko zaposlenih može da radi jedan specifičan posao? Jedan zaposleni? Ili više zaposlenih?
- avesti operativni sistem da je došlo do prekida, kako bi operativni sistem mogao da odgovori na njega.

Relacije koje se uspostavljaju između različitih entiteta mogu biti trojake:

1. **Relacije jedan prema jedan** (1:1), gde jednoj instanci jednog objekta odgovara tačno jedna instanca drugog objekta
2. **Relacije jedan prema više** (1:N), gde jednoj instanci jednog objekta odgovara više instanci drugog objekta
3. **Relacije više prema više** (N:M), gde je više (N) instanci jednog objekta u vezi sa više (M) instanci drugog entiteta

Ne moraju sve instance da učestvuju u nekoj vrsti veze. Postoje veze koje su obavezne. Tada se kaže da instanca obavezno učestvuje u nekoj vrsti veze sa drugom instancom. U suprotnom se kaže da je učestvovanje u vezi neobavezno. Ova pojava se naziva opcionalnost veza. Na primer, svaki polaznik mora da pohađa neki kurs. Suprotno tome, nije obavezno da svaki kurs ima polaznike. Moguće je da je kreiran neki kurs za koga polaznici nikad nisu bili zainteresovani.

Veze mogu imati atribute. Atributi se dodeljuju vezama kada ih nije moguće pripisati niti jednom od entiteta. Veze mogu imati i ograničenja. Na primer, jedan kurs može imati maksimalno 20 polaznika.

PRIMER OPCIONALNOSTI I KARDINALNOSTI

Primer opcionalnosti i kardinalnosti na entitetima ZAPOSLENI, POSAO, PROIZVOD, TIP PROIZVOD

Primeri:

- Svaki **ZAPOSLENI** mora da ima **jedan** i samo **jedan** **POSAO**
- Svaki **POSAO** može da radi **više** **ZASPOLENIH**

- Svaki **PROIZVOD** mora da bude klasifikovan kao jedan i samo jedan **TIP PROIZVODA**
- Svaki **TIP PROIZVODA** može da klasificuje jedan ili više **PROIZVODA**

Određujemo optionalnost postavljanjem pitanja:

Da li svaki zaposleni mora da ima posao?

- drugim rečima, da li je ovo obavezna ili opcionalna relacija za zaposlenog?

Da li svaki posao mora da obavlja neki zaposleni?

- drugim rečima, da li je ovaj posao obavezna ili opcionalna relacija za posao?

Utvrđujemo kardinalnost postavljanjem pitanja:

- Koliko radnih mesta može imati jedan radnik? Samo jedno radno mesto? Ili više od jednog radnog mesta?
- Koliko zaposlenih može imati jedan konkretni posao? Samo jedan zaposleni? Ili više od jednog zaposlenog?

PRIMER RELACIJA NA POSLOVNOM SCENARIJU RESTORANA

Primer kardinalnosti i optionalnosti između entiteta MUŠTERIJA i PORUDŽBINA

Koje relacije postoje u sledećem poslovnom scenariju?

- "Mušterija ulazi u naš restoran i poručuje hrani. Mušterija može da poruči samo za sebe ili za sebe i druge. Na primer, majka poručuje za sebe i svoju decu.
- Mi smatramo da je majka vlasnik ove porudžbine i da je odgovorna za plaćanje. U nekom vremenu, mušterija može da napravi više različitih porudžbina".

Optionalnost

MUŠTERIJA pravi PORUDŽBINU

Koja je optionalnost ove relacije: mora ili može?

Svaka **PORUDŽBINA** mora biti napravljena od strane jedne (i samo jedne) **MUŠTERIJE**.

Svaka **MUŠTERIJA** mora napraviti jednu ili više **PORUDŽBINA**.

Kardinalnost

Kardinalnost = Koliko?

Svaka **PORUDŽBINA** mora biti napravljena od strane jedne i samo jedne **MUŠTERIJE**.

Svaka **MUŠTERIJA** mora napraviti jednu ili više **PORUDŽBINA**.

E/R MODEL - PRIMER

Primer E/R modela koji manipuliše sa podacima o kursevima u edukativnoj instituciji

Kao primer modela objekti – veze daje se problem modeliranja podataka jednog informacionog sistema koji treba da manipuliše sa podacima o kursevima koje jedna edukativna institucija drži. Baza podataka će sadržati podatke o sledećim objektima (entitetima): ***Kurs, Predavači, Polaznici kursa.***

Objekat ***Kurs*** ima svoje atribute: ***Naziv, Broj časova***

Objekat ***Predavač*** ima svoje atributе: ***Ime***, ***Prezime***, ***Telefon***

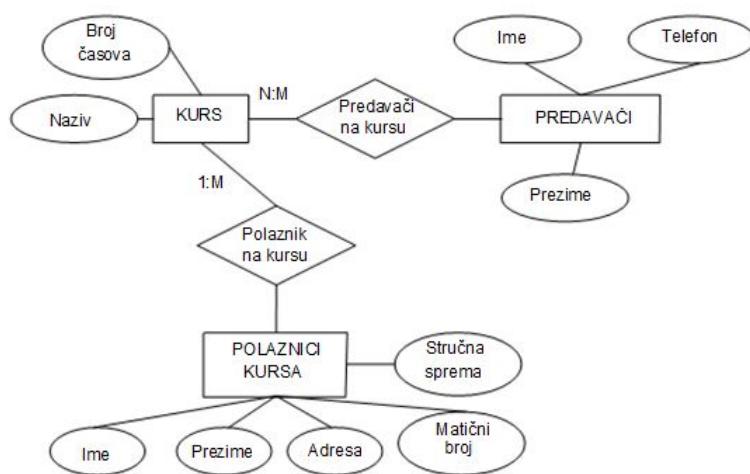
Objekat **PolazniciKursa** ima atribute: **MatičniBroj, Ime, Prezime, StručnaSprema, Jezici**.
Svaki kurs ima svoje polaznike i svoje predavače.

Atributi mogu imati različita ograničenja kao što su, na primer:

- Za polaznike kursa obavezno uneti ime i prezime
 - Za matični broj obavezno uneti tačno 13 cifara
 - Jedan polaznik može da učestvuje samo na jednom kursu

Veze (relacije) u primeru:

- Jedan kurs ima više različitih polaznika
 - Na jednom kursu učestvuje više predavača , ali i jedan predavač može da učestvuje na više kurseva, pa je veza N:M



Slika 2.1.2 E/R dijagram za bazu podataka Kurs [Izvor: Autor]

ER-model je dovoljno jednostavan da ga ljudi iz različitih struka mogu razumeti, pa zato služi za komunikaciju projektanta baze podataka i budućih korisnika i to u najranijoj fazi razvoja baze.

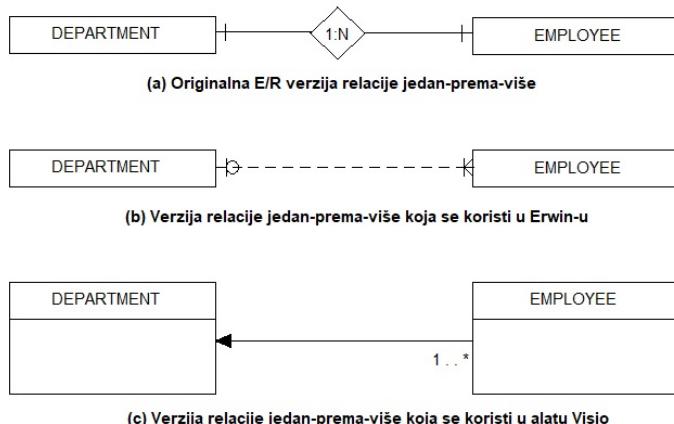
Sadašnji SUBP ne mogu direktno implementirati ER-šemu, pa je ovu šemu potrebno detaljnije razraditi, i potom modifikovati u skladu sa pravilima odgovarajućih relacionih, mrežnih ili hijerarhijskih modela. Prelazak na relacioni model i njegovu implementaciju nije komplikovan ukoliko je ER šema dobro projektovana.

VERZIJE E-R DIJAGRAMA

Originalni Chen-ov model, modeli koji se koriste u CASE alatu ERWIN ili PowerDesigner i VISIO

Originalnim, Chen-ovim E/R modelom (koji je korišćen u ranijim primerima) je definisano da se za predstavljanje relacija koristi romb, entiteta pravougaonik a atributa elipse koje su povezane sa odgovarajućim pravougaonikom. Međutim, ovaj originalni model se danas retko koristi iz dva razloga. Prvo, postoji mnogo verzija E/R modela i te verzije koriste različite simbole. Drugo, softverski proizvodi za modeliranje podataka (CASE alati) koriste različite tehnike.

Kroz sledeće tri slike su prikazane tri verzije jedan-prema-više relacije, opcionog-prema-obaveznog ograničenja. Slika 3.(a) prikazuje originalnu E/R verziju; Slika 3.(b) verziju koja se koristi u CASE alatu Erwin ili PowerDesigner; Slika 3.(c) verziju iste relacije koju koristi CASE alat Visio.



Slika 2.1.3 Različite verzije E/R dijagrama [Izvor: Autor]

PRIMER - VIDEO

Entity-Relationship Diagrams

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

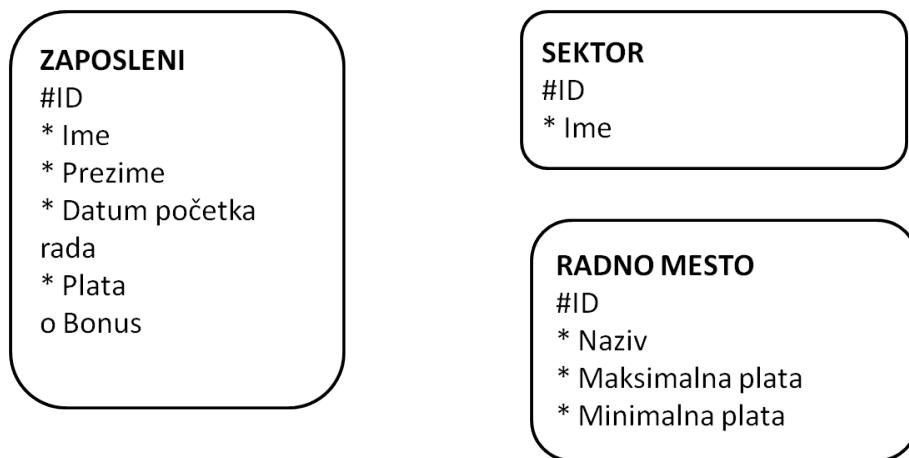
✓ 2.2 ANALIZA E/R MODELA

PRAVILA CRTANJA E/R MODELA

Obavezni atributi su obeleženi zvezdicom "", a opcioni kružićem "o"*

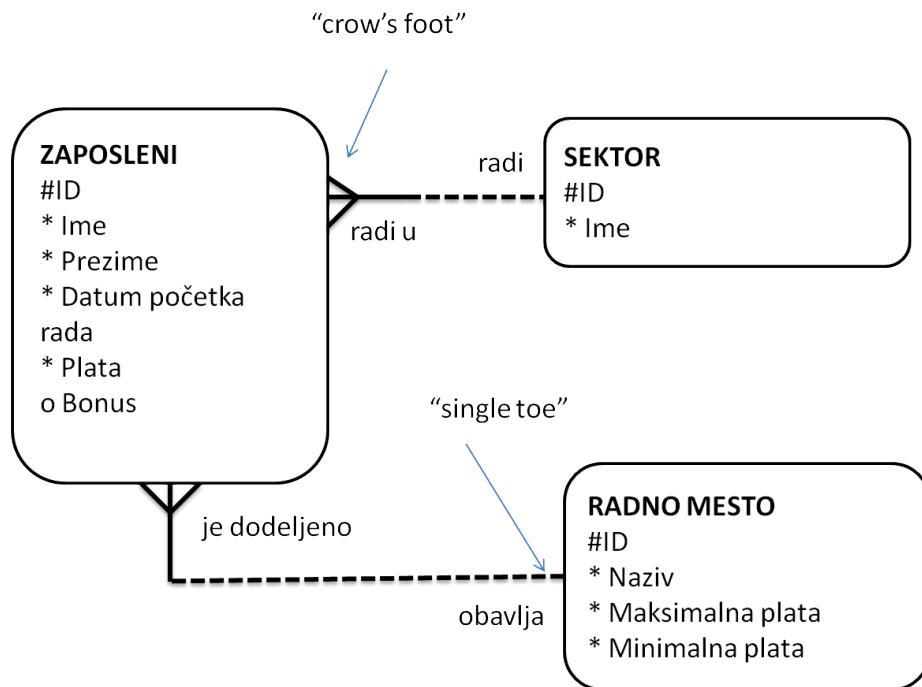
Pravila crtanja E/R modela su sledeća:

- Naziv entiteta se nalazi na vrhu
- Atributi su navedenih odmah ispod naziva atributa
- Obavezni atributi su obeleženi zvezdicom "*"
- Opcioni atributi su obeleži kružićem "o"
- Jedinstveni identifikator je obeležen tarabom "#"



Slika 2.2.1 Primer entiteta E/R modela [Izvor: Autor]

Relacije su linije koje povezuju entitete. Ove linije mogu biti pune ili isprekidane. Pune linije označavaju da je relacija obavezna, a isprekidana da je opcionalna. Ove linije se završavaju kao jedna linija (engl. **single toe**) ili tipom "svračije noge" (engl. **crows foot**).



Slika 2.2.2 Primer relacija u E/R modelu [Izvor: Autor]

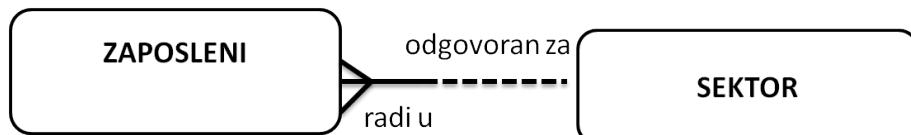
TUMAČENJE RELACIJA

Relacije se čitaju s leva u desno i s desna u levo

Uopšteno gledano, relacije sa E/R dijagrama možemo pročitati na sledeći način

- SVAKI
- **ENTITET A**
- **OPCIONO** (mora biti/može biti)
- **IME VEZE** ili **RELACIJE**
- **KARDINALNOST** (jedan i samo jedan/jedan ili više)
- **ENTITET B**

Pogledajmo sledeći primer i pročitajmo relaciju s leva u desno od entiteta **ZAPOSLENI** ka entitetu **SEKTOR**.



Slika 2.2.3 Primer relacije između entiteta ZAPOSLENI i SEKTOR [Izvor: Autor]

- SVAKI
- **ZAPOSLENI** (Entitet A)
- **MORA** (opcionalnost, puna linija)
- **RADITI U** (ime veze ili relacije)
- **JEDNOM (I SAMO JEDNOM)** (kardinalnost, single toe)

- **SEKTORU** (Entitet B)

Čitajući ovaj primer s desna u levo, od entiteta **SEKTOR** i isprekidane linije, ka entitetu **ZAPOSLENI** i završetku "crow's foot".

- SVAKI
- **SEKTOR** (Entitet A)
- **MOŽE BITI** (opcionalnost, puna linija)
- **ODGOVORAN ZA** (ime veze ili relacije)
- **JEDNOG ILI VIŠE (I SAMO JEDNOM)** (kardinalnost, single toe)
- **ZAPOSLENOG** (Entitet B)

✓ 2.3 LOGIČKI MODEL

ŠTA PREDSTAVLJA LOGIČKO PROJEKTOVANJE BAZE PODATAKA?

Logičko projektovanje relacionih baza podataka je transformisanje konceptualnog modela u šemu koji se korišćenjem DDL unosi u SUBP

Logički model baze podataka definiše strukturu podataka pogodnu za projektovanje i implementaciju baze podataka ili za razvoj aplikacija koje koriste bazu. Logički model se sastoji od potpuno normalizovanih objekata čiji su svi atributi definisani. Pored toga, definisan je tip podataka i domen svih atributa, kao i atributi koji su kandidati za ključeve. Logički model ne sadrži više-prema-više relacije između objekata. Ako su one postojale u konceptualnom modelu one se transformišu u asocijativne objekte. Drugim rečima, logički model treba da bude kompletan opis baze podataka na osnovu koga će se razviti željena baza podataka.

Logičko projektovanje relacionih baza podataka je postupak kojim se ranije definisani konceptualni model transformiše u šemu, odnosno opis baze podataka u izabranom sistemu za upravljanje bazama podataka. Cilj je da se dobije potpuno definisan relacioni model podataka koji se onda korišćenjem DDL jezika unosi u sistem za upravljanje bazama podataka (SUBP).

Ako je konceptualni model već preveden u relacioni model, na primer normalizacijom tabela, nikakva transformacija nije potrebna. Međutim, ako je konceptualni model izrađen kao model objekti – veze, potrebno je izvršiti njegovu transformaciju u relacioni model.

Transformacija modela objekti – veze u relacioni model se vrši tako što se najpre model objekti – veze transformiše u fizički model objekti – veze. Ovaj model preciznije definiše strukturna pravila integriteta i lako se transformiše u relacioni model. Po dobijanju fizičkog modela objekti – veze vrši se njegova transformacija u relacioni model.

Većina CASE alata koja prati SUBP vrše automatsku transformaciju modela objekti-veze i generišu bazu podataka u izabranom SUBP.

2.4 FIZIČKI MODEL

TRANSFORMACIJA LOGIČKOG U FIZIČKI MODEL BAZE PODATAKA

Prvi korak u transformisanju logičkog modela u fizički model je transformisanje koncepata logičkog modela u fizičke objekte

Fizički model podataka se kreira da bi se izvršila transformacija logičkog modela u fizičku implementaciju baze podataka korišćenjem nekog SUBP-a. Ovaj model služi administratoru baze podataka da implementira specificiranu bazu.

Da bi se uspešno kreirao fizički model potrebno je dobro poznavati osobine izabranog SUBP, kao što su:

- koje objekte podržava SUBP i koje su fizičke strukture i datoteke potrebne za podršku tih objekata,
- kako je podržano indeksiranje i održavanje referencijalnog integriteta
- koji su tipovi podataka na raspolaganju
- koji su tipovi ograničenja na raspolaganju

Prvi korak u transformisanju logičkog modela u fizički model je transformisanje koncepata logičkog modela u fizičke objekte. Ovo podrazumeva sledeće aktivnosti:

- Transformacija objekata u tabele
- Transformacija atributa u kolone
- Transformacija domena u tipove podataka i ograničenja
- Specifikacija primarnih ključeva
- Specifikacija redosleda kolona
- Definisanje referencijalnih ograničenja za sve veze
- Definisanje prostora tabela (**tablespace**)

KORACI U IMPLEMENTACIJI FIZIČKOG MODELA

Transformacija objekata u tabele, atributa u kolone, domena u tipove podataka i ograničenja, specifikacija primarnih ključeva, specifikacija redosleda kolona

Transformacija objekata u tabele. Fizički pandam objektima, odnosno entitetima je tabela. U većini slučajeva svakom objektu iz logičkog modela će odgovarati jedna tabela u fizičkom modelu. Međutim, postoje slučajevi kada je neke tabele potrebno spojiti ili razdvojiti da bi se postigle bolje performanse baze podataka.

Transformacija atributa u kolone. Fizički pandam atributima su kolone u tabelama. Atributi svakog objekta treba da budu preslikani u kolone odgovarajuće tabele.

Transformacija domena u tipove podataka i ograničenja. Za svaki logički domen atributa je potrebno odabratи adekvatan tip podataka iz seta tipova koji nudi SUBP. Na primer, za godinu rođenja se može izabrati tip podataka *integer*, a za matični broj, koji ima čak 13 cifara, je možda bolje izabrati tip podataka *text*. Neki tipovi podataka zahtevaju bliže definisanje podataka. Na primer tip podataka *text* zahteva da se specificira maksimalni broj karaktera, mada neki SUBP dozvoljavaju da se specificira da tekst može biti promenljive dužine.

Pored tipa podataka, ovde je potrebno definisati i ograničenja vrednosti podataka. Na primer, atribut BrojRadnihSatiDnevno se može ograničiti na vrednosti između 0 i 12, jer su takvi poslovni zahtevi. Ovakva ograničenja znatno smanjuju mogućnost unosa pogrešnih vrednosti atributa. Takođe treba definisati da li atribut može imati Null (nema vrednosti) vrednost, ili neku podrazumevajuću vrednost koja se automatski upisuje kao vrednost atributa prilikom kreiranja novog zapisa.

Specifikacija primarnih ključeva. Prilikom izrade logičkog modela primarni ključevi su definisani i treba ih preslikati u fizički model. U slučajevima kada to iz nekih razloga nije moguće treba izabrati surogat.

Specifikacija redosleda kolona. Sa aspekta tačnosti izvršenja operacija redosled kolona u tabelama je irelevantan. Rezultati upita će biti isti bez obzira da li je neka kolona treća ili osma po redu. Međutim, redosled kolona može bitno da utiče na brzinu izvršavanja operacija, pa je u nekim slučajevima potrebno promeniti redosled koji nudi logički model podataka.

DEFINISANJE REFERENCIJALNIH OGRANIČENJA I PROSTORA TABELE

Za svako referencijalno ograničenje je potrebno definisati set pravila koja određuju status kolone sa stranim ključem prilikom insertovanja i ažuriranja

Definisanje referencijalnih ograničenja za sve veze . Fizički pandam logičkim vezama su referencijalna ograničenja. Da bi se definisala referencijalna ograničenja potrebno je definisati primarni ključ u jednoj tabeli i strani ključ u zavisnoj tabeli. Referencijalna ograničenja povezuju primarni ključ za strani. Za svako referencijalno ograničenje je potrebno definisati set pravila koja određuju status kolone sa stranim ključem prilikom insertovanja i ažuriranja, kao i status zavisne vrste kada se primarni ključ briše. Na primer, kada se briše primarni ključ koji referencira na postojeći strani ključ, ovo pravilo definiše da li će SUBP izbeći brisanje primarnog ključa ili će obrisati i primarni i strani ključ, ili će se kao vrednost stranog ključa upisati **Null** vrednost. Ovakvim pravilima se garantuje referencijalni integritet baze podataka.

Definisanje prostora tabela (**tablespace**). Iako su podaci relacionog modela fizički organizovani u tabele, odgovarajuće datoteke u kojima su smešteni podaci nisu jednostavno preslikane kolone i vrste. Tabele se preslikavaju u fizičke strukture koje se nazivaju prostor tabela (**tablespace**). Baza podataka se sastoji iz jednog ili više prostora tabela. Svaki prostor tabela može da sadrži jednu ili više tabele. Pored planiranja prostora tabela, potrebno je proceniti potreban fizički prostor na spoljnim memorijama potrebnim za čuvanje podataka i

indeksa. Takođe je potrebno definisati potrebu za kompresijom podataka metod koji će se koristiti.

▼ Poglavlje 3

NORMALNE FORME

ANOMALIJE KOJE SE MOGU JAVITI U BAZAMA PODATAKA

Anomalije prilikom dodavanja novog zapisa, brisanja zapisa i izmena vrednosti (ažuriranje zapisa)

Prilikom projektovanja baze podataka je potrebno odrediti:

- Koje tabele (entitete) će baza imati
- Koji su atributi svake tabele pojedinačno
- Kakve su veze između tabela.

Vrlo često se pri tom dogodi da se isti podaci pojavljuju više puta u istoj bazi. Ponavljanje istog podatka u jednoj bazi podataka se naziva **redundantnost** podataka.

Treba imati na umu da će se tokom korišćenja baze podataka nad njom vršiti osnovne operacije ažuriranja:

- dodavanje novog zapisa
- brisanje postojećeg zapisa
- izmena vrednosti jednog ili više atributa u postojećem zapisu.

U slučaju da u bazi podataka ima previše redundantnosti doći će do anomalije u postupcima ažuriranja koje mogu ugroziti konzistentnost podataka. Da bi se objasnili razlozi pojavljivanja anomalija iskoristiće se kao primer sledeća tabela neke baze podataka koja sadrži podatke o studentima i predmetima koje oni slušaju.

Studenti : Table							
BrojZapisa	BrojIndeksa	Ime	Prezime	Predmet	Profesor	Smer	
1	124	Zoran	Koštić	Matematika	Ristić	IT	
2	199	Ana	Jeftić	Osnove IT	Gogić	IT	
3	215	Slavko	Stojanović	Programiranje	Bojić	SE	
4	444	Zoran	Koštić	Engleski	Mišić	IT	
5	454	Andrija	Jovanović	Osnove IT	Gogić	IT	
6	215	Slavko	Stojanović	Engleski I	Mišić	SE	
7	124	Zoran	Koštić	Osnove IT	Gogić	IT	
8	199	Ana	Jeftić	Programiranje	Bojić	IT	
*	0	0					

Slika 3.1.1 Tabela-1 Primer tabele za primenu normalnih formi [Izvor: Autor]

Anomalije prilikom dodavanja novog zapisa se ogledaju u tome što je prilikom dodavanja grupe novih podataka neophodno ubaciti i grupu postojećih podataka čime se uvećava redundantnost. Ako je u prethodnoj tabeli potrebno ubaciti novog studenta potrebno je ubaciti i naziv predmeta i ime profesora koji ga drži, iako ti podaci već postoje u bazi. Drugi

primer ove anomalije bi se pojavio ako bi trebalo ubaciti podatke o novom predmetu. To ne bi bilo moguće dok ga neki student ne odabere.

Anomalije prilikom brisanja zapisa se javljaju kada se zbog brisanja neke grupe podataka, automatski brišu i neki drugi podaci. Ako se student Zoran Kostić ispiše sa fakulteta, pa se iz baze izbrišu podaci o njemu, automatski će se izbrisati i podaci o matematici i njenom profesoru.

Anomalije prilikom izmena vrednosti se javljaju kada se u stvarnosti promeni vrednost nekog atributa. Ako je ovaj podatak bio redundantan u bazi potrebno ga je promeniti svuda gde se pojavljivao. Ako se u prethodnom primeru profesor za predmet Osnove IT promenio, potrebno je izvršiti izmene u tri zapisa.

NORMALIZACIJA

Normalizacija se koristi kako bi se smanjile ili otklonile anomalije u bazama podataka

Normalizacija je postupak poboljšanja šeme eliminisanjem uzroka koji dovode do anomalija u postupcima ažuriranja baze podataka. Procesom normalizacije se tabele, dovode u normalne forme koje obezbeđuju minimum redundantnosti.

Jedan od načina rešavanja anomalija je dekompozicija tabela, pri čemu se jedna tabela zamenjuje sa više drugih tabela tako da se uklone anomalije. Mogu se prepoznati nekoliko pravila za normalizaciju:

- Polja treba da budu atomična, što znači da svaki podatak treba da bude razbijen na što jednostavnije delove
- Svaki zapis – red u tabeli bi trebalo da ima jedinstveni identifikator – primarni ključ
- Primarni ključ treba da bude jednostavan, stabilan i kratak
- Svako polje bi trebalo da obezbedi neku dodatnu informaciju o zapisu u kome primarni ključ služi za identifikaciju (drugim rečima, nema smisla imati zapis u kome je jedini atribut primarni ključ)
- Informacije u tabelama ne bi trebalo da se pojavljuju na više mesta, već samo ne jednom.

Normalne forme su rezultat primene gornjih pravila. Razlikuju se pet normalnih formi (NF). To su 1NF, 2NF, 3NF, 4NF i 5NF. Svaka sledeća normalna forma u redosledu omogućava sve bolji řemu baze podataka od prethodne i manju redundantnost.

E.F. Codd je u svojim radovima od 1970-1974 definisao drugu i treću normalnu formu (2NF & 3NF), a zatim i poboljšanu varijantu 3NFa koja se zove Boyce-Coddova normalna forma (BCNF). Relacija koja zadovoljava Boyce-Coodovu normalnu formu sigurno zadovoljava i 2NF i 3NF. Norme 4NF i 5NF su prvenstveno od teorijskog značaja, jer je teško u praksi naći relacije koje jesu u BCNF a nisu u 4NF i 5NF, a istovremeno se u brzini i samom unapređenju baze ne dobija mnogo u odnosu na uloženi rad za normalizaciju.

3.1 NORMALNA FORMA 1NF

KADA TABELA ZADOVOLJAVA PRVU NORMALNU FORMU (1NF)?

Da bi tabela zadovoljavala prvu normalnu formu potrebno je da nema atributi koji su i sami entiteti

Da bi tabela zadovoljavala **prvu normalnu formu** potrebno je da nema atributi koji su i sami entiteti. Drugim rečima sve vrednosti podataka u tabeli treba da su atomske. U tabeli 1 je dat primer tabele Studenti koja ne zadovoljava 1NF.

Studenti										
Broj Indeksa	SifraPredmet	Ime	Prezime	Semestar	Ocena	Predmet	Profesor	SifraSmera	Smer	
199	MAT	Ana	Jeftić	2	7	Matematika	Ristić	01	IT	
199	IT101				8	Osnove IT	Gogić	01	IT	
454	IT101	Andrija	Jovanović	1	8	Osnove IT	Gogić	02	SE	
124	IT101	Zoran	Kostić	1	10	Osnove IT	Gogić	01	IT	
124	ENG100				7	Engleski I	Mišić	01	IT	
124	MAT				6	Matematika	Ristić	01	IT	
215	ENG100	Slavko	Stojanović	2	9	Engleski I	Mišić	02	SE	
215	PR200				10	Programiranje	Bojić	02	SE	

Slika 3.2.1 Tabela-1 Tabela Studenti koja ne zadovoljava 1NF [Izvor: Autor]

Očigledno je da su atributi ocena i predmet i sami tabele. Prethodna tabela će zadovoljiti 1NF ako se sve vrednosti njenih atributa postanu atomske. To dovodi do normalizovane tabele (tabela 2).

Studenti										
Broj Indeksa	SifraPredmet	Ime	Prezime	Semestar	Ocena	Predmet	Profesor	SifraSmera	Smer	
199	MAT	Ana	Jeftić	2	7	Matematika	Ristić	01	IT	
199	IT101	Ana	Jeftić	2	8	Osnove IT	Gogić	01	IT	
454	IT101	Andrija	Jovanović	1	8	Osnove IT	Gogić	02	SE	
124	IT101	Zoran	Kostić	1	10	Osnove IT	Gogić	01	IT	
124	ENG100	Zoran	Kostić	1	7	Engleski I	Mišić	01	IT	
124	MAT	Zoran	Kostić	1	6	Matematika	Ristić	01	IT	
215	ENG100	Slavko	Stojanović	2	9	Engleski I	Mišić	02	SE	
215	PR200	Slavko	Stojanović	2	10	Programiranje	Bojić	02	SE	

Slika 3.2.2 Tabela-2 Tabela Studenti u prvoj normalnoj formi [Izvor: Autor]

Dakle, tabela R je u prvoj normalnoj formi ako su sve vrednosti njenih atributa atomske. Sve tabele u relacionom modelu podataka moraju da budu u 1NF.

Tabela Studenti se formalno opisuje na sledeći način:

Studenti (BrojIndeksa, SifraPredmeta, Ime, Prezime, Semestar, Ocena, Predmet, Profesor, SifraSmera, Smer)

Primarni ključ tabele Studenti je složeni ključ koji se sastoji od atributa BrojIndeksa, SifraPredmeta, pa su oni podvučeni u opisu tabele.

U tabeli Studenti se mogu uočiti sledeće funkcionalne zavisnosti:

BrojIndeksa, SifraPredmeta → Ime, Prezime, Semestar, Ocena, Predmet, Profesor, SifraSmera, Smer

BrojIndeksa → Ime, Prezime, Semestar, SifraSmera, NazivSmera

SifraPredmeta → Predmet, Profesor

SifraSmera → Smer

▼ 3.2 NORMALNA FORMA 2NF

KADA TABELA ZADOVOLJAVA DRUGU NORMALNU FORMU (2NF)?

Tabela je u drugoj normalnoj formi ako je tabela u prvoj normalnoj formi i svi atributi potpuno funkcionalno zavise od primarnog ključa, a ne od nekog njegovog dela.

Tabela je u **drugoј normalnoј formi** ako je tabela u prvoj normalnoj formi i svi atributi potpuno funkcionalno zavise od primarnog ključa, a ne od nekog njegovog dela. 2NF je uvek ispunjena kada je primarni ključ prost atribut, odnosno kada se primarni ključ sastoji samo od jednog atributa.

Druga normalna forma će se objasniti na primeru entiteta Studenti koja je data u prethodnoj tabeli i koja zadovoljava 1NF. Iz funkcionalne zavisnosti entiteta Studenti

BrojIndeksa, SifraPredmeta → Ime, Prezime, Semestar, Ocena, Predmet, Profesor, SifraSmera, Smer se vidi:

- da je primarni ključ ove tabele složeni ključ **BrojIndeksa, SifraPredmeta**
- da su svi ne-ključni atributi osim atributa **Ocena** nepotpuno funkcionalno zavisni od primarnog ključa.

Dakle, entitet Studenti nije u 2NF. Entitet Studenti može da se prevede u 2NF dekompozicijom tabele Studenti na tri sledeće tabele:

Studenti1					
Broj Indeksa	Ime	Prezime	Semestar	SifraSmera	Smer
199	Ana	Jeftić	2	01	IT
199	Ana	Jeftić	2	01	IT
454	Andrija	Jovanović	1	02	SE
124	Zoran	Koštić	1	01	IT
124	Zoran	Koštić	1	01	IT
124	Zoran	Koštić	1	01	IT
215	Stavko	Stojanović	2	02	SE
215	Slavko	Stojanović	2	02	SE

Slika 3.3.1 Tabela-1 Studenti1 (BrojIndeksa, Ime, Prezime, Semestar, SifraSmera, Smer) [Izvor: Autor]

Prijava		
Broj Indeksa	SifraPredmeta	Ocena
199	MAT	7
199	IT101	8
454	IT101	8
124	IT101	10
124	ENG100	7
124	MAT	6
215	ENG100	9
215	PR200	10

Slika 3.3.2 Tabela-2 Prijava (BrojIndeksa, SifraPredmeta, Ocena) [Izvor: Autor]

Predmeti		
SifraPredmeta	Predmet	Profesor
MAT	Matematika	Ristić
IT101	Osnove IT	Gogić
IT101	Osnove IT	Gogić
IT401	Osnove IT	Gogić
ENG100	Engleski I	Mišić
MAT	Matematika	Ristić
ENG100	Engleski I	Mišić
PR200	Programiranje	Bojić

Slika 3.3.3 Tabela-3 Predmeti (SifraPredmeta, Predmet, Profesor) [Izvor: Autor]

Očigledno je da se u tabeli Predmeti pojavljuje redundantnost jer se isti zapisi ponavljaju. Posle normalizovanja, ova tabela dobija oblik:

Predmeti		
SifraPredmeta	Predmet	Profesor
MAT	Matematika	Ristić
IT101	Osnove IT	Gogić
ENG100	Engleski I	Mišić
PR200	Programiranje	Bojić

Slika 3.3.4 Tabela-4 Tabela koja zadovoljava 2NF [Izvor: Autor]

Sada se može reći da sve tabele zadovoljavaju 2NF.

▼ 3.3 NORMALNA FORMA 3NF

KADA TABELA ZADOVOLJAVA TREĆU NORMALNU FORMU (3NF)?

Kaže se da je relacija u 3NF ako zadovoljava 1NF i 2NF i ako svaki atribut koji nije deo primarnog ključa zavisi samo od primarnog ključa, odnosno ne postoji zavisnost između ne-ključnih atributa.

Kaže se da je relacija **u trećoj normalnoj formi** ako je u prvoj i drugoj normalnoj formi i ako svaki atribut koji nije deo primarnog ključa zavisi samo od primarnog ključa, odnosno ne postoji zavisnost između ne-ključnih atributa. Može se reći da je relacija u 3NF ako je u 2NF i ako svi njeni ne-ključni atributi zavise od primarnog ključa. Drugim rečima, za 3NF se zahteva da svaka tabela sadrži podatke o samo jednom tipu entiteta.

Tabela **Studenti1** ne zadovoljava uslove 3NF jer postoji zavisnost između ne-ključnih atributa **SifraSmera** i **Smer**. Ova dva atributa su tranzitivno zavisna od primarnog ključa **BrojIndeksa**. Svođenje ovakve tabele na 3NF se vrši dekompozicijom tabele na dve nove tabele, tako da prva tabela sadrži primarni ključ i sve netranzitivno zavisne attribute, a druga tabela sadrži ostatak atributa između kojih postoji funkcionalna zavisnost. U slučaju tabele **Studenti1** svođenje na 3NF će se izvršiti dekompozicijom na sledeće dve tabele:

Studenti			
Broj Indeksa	Ime	Prezime	Semestar
199	Ana	Jeftić	2
454	Andrija	Jovanović	1
124	Zoran	Kostić	1
215	Slavko	Stojanović	2

Slika 3.4.1 Tabela-1 Studenti2 (BrojIndeksa, Ime, Prezime, Semestar) [Izvor: Autor]

Smerovi	
SifraSmera	Smer
01	IT
02	SE

Slika 3.4.2 Tabela-2 Smerovi (SifraSmera, Smer) [Izvor: Autor]

Na kraju treba primetiti da je 3NF najstrožija. Relacija koja zadovoljava 3NF, mora da zadovoljava i 1NF i 2NF.

U nekim slučajevima se iz praktičnih razloga odstupa od normalnih formi. Taj proces se naziva denormalizacija. Tom prilikom se između tabela postavlja relacija 1:1. Na primer, ako više atributa relacije čine jednu celinu i uvek se pojavljuju zajedno (ulica, kućni broj i grad, ...). Pretraživanje velikog broja malih relacija nastalih normalizacijom može bitno da produži vreme potrebno za dobijanje rezultata pretraživanja. Projektant baze podataka mora da proceni da li da sprovede potpunu normalizaciju ili ne, a što zavisi od više činilaca (npr. koliko često će se i koji podaci pretraživati).

NORMALIZACIJA - VIDEO

*Database Normalization In DBMS | Boyce Codd Normal Form (BCNF)
And 1NF 2NF 3NF Explained*

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 4

IDEF METODI ZA MODELIRANJE

IDEF

IDEF predstavlja seriju standarda koji definišu jezik modeliranja, odgovarajuća pravila i tehnike za razvoj grafičke reprezentacije sistema ili organizacije

Metodi integracije definicija IDEF (Integration Definition) su serija standarda američke organizacije Federal Information Processing Standard (FIPS) koji definišu jezik modeliranja, odgovarajuća pravila i tehnike za razvoj grafičke reprezentacije sistema ili organizacije. Ovi metodi obezbeđuju naučni pristup rešavanju problema. Metodi vode projektante kroz disciplinovani, pouzdani pristup dobijen na osnovu ekspertskog iskustva. Metodi, takođe, naznačavaju važne objekte, relacije i ograničenja, i prikrivaju irrelevantne informacije i nepotrebne detalje. Metodi su projektovani da poboljšaju efikasnost rada kako pojedinaca tako i timova uključenih u aktivnosti razvoja sistema.

IDEF metodi obezbeđuju pojedincima i timovima da disciplinovanim, pouzdanim pristupom rešavaju probleme kao podrška aktivnostima kao što su:

- inženjering preduzeća,
- reinženjering,
- razvoj informacionih sistema velikih razmara,
- integracija preduzeća.

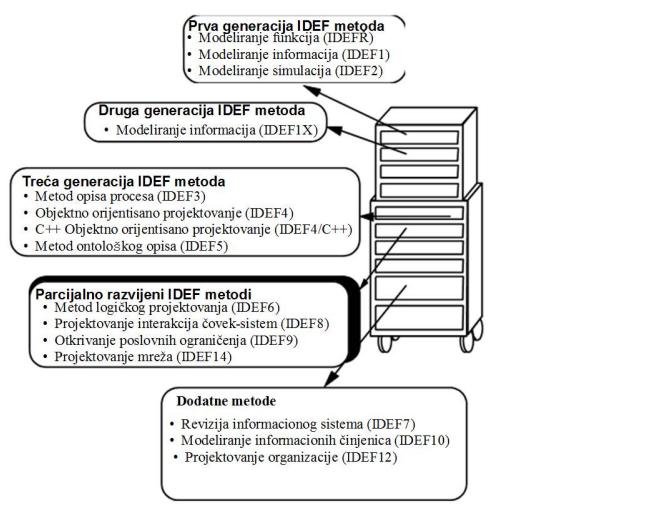
IDEF metodi takođe poboljšavaju odziv u uslovima brzih i kontinuiranih promena pomažući korisnicima da:

- ispravno razumeju trenutno okruženje
- predlažu promene
- ispitaju alternativna rešenja
- predvide uticaje promena
- uspešno implementiraju promene.

IDEF STANDARDI U UPOTREBI

Prva, druga i treća generacija IDEF metoda, parcijalno razvijene IDEF metode

Prva generacija IDEF metoda proistiće iz **Air Force Integrated Computer Aided Manufacturing (ICAM)** programa kasnih sedamdesetih. ICAM program je razvio IDEF0 metod za modeliranje funkcija, IDEF1 metod za modeliranje informacija i IDEF2 metod za modeliranje simulacija. Drugi ICAM projekat je kasnije razvio IDEF1X – proširen metod modeliranja informacija. IDEF1X je semantička metoda modeliranja podataka. Ona je proizvod je analize i poboljšanja IDEF1 metoda, sa težnjom da pomogne u proceduri logičkog projektovanja baze podataka. Treća generacija IDEF metoda se pojavila iz potrebe za tehnologijom metoda koja će podržavati razvoj proširenih informacija podržanih sistema koji podržavaju simultani inženjering. Ovaj razvoj je finansirala Air Force Armstrong laboratorija kroz IICE program i proizvela je metode za opis procesa (IDEF3), objektno orijentisano projektovanje (IDEF4 i IDEF4++) i ontološki opis. Kroz IICE program su preliminarno ustanovljeni i metodi za otkrivanje poslovnih ograničenja (IDEF9), logičko projektovanje (IDEF6), projektovanje interakcije čovek-sistem (IDEF8) i projektovanje mreža (IDEF14).



Slika 4.1.1 IDEF standardi koji su u upotrebi [Izvor: https://upload.wikimedia.org/wikipedia/commons/thumb/e/e4/IDEF_Methods.svg/1200px-IDEF_Methods.svg.png]

4.1 IDEF1X

MODEL IDEF1X STANDARDA

IDEF1X standard opisuje semantiku i sintaksu jezika modeliranja, kao i pravila i tehnike za razvoj logičkih modela podataka

IDEF1X je IDEF (**Integration Definition**) standard druge generacije namenjen pre svega za logičko modeliranje baza podataka [7]. Originalan naziv ovog standarda izdatog 1993. godine od strane američkog National Institute of Standards and Technology je **Integration Definition for Information Modeling** (IDEF1X).

IDEF1X standard opisuje semantiku i sintaksu jezika modeliranja, kao i pravila i tehnike za razvoj logičkih modela podataka. On se koristi za kreiranje grafičkih modela koji predstavljaju strukturu i semantiku podataka unutar nekog okruženja ili sistema. Ovakvi modeli se mogu

koristiti za podršku upravljanju podacima kao resursa, integraciju informacionih sistema i projektovanje baza podataka. Osnovne konstrukcije jednog IDEF1X modela su:

1. Entiteti o kojima se čuvaju podaci koji su predstavljeni pravougaonikom (na primer ljudi, ideje, događaji, klijenti), koji mogu biti:

- Identifikaciono nezavisni entiteti
- Identifikaciono zavisni entiteti

2. Veze između entiteta koje se predstavljaju linijama koje spajaju pravougaonike i koje mogu biti:

- Identifikaciona spojna veza
- Neidentifikaciona spojna veza
- Kategorizaciona veza
- Nespecifična veza

3. Karakteristike entiteta koje su predstavljene imenima atributa unutar pravougaonika. Atributi, odnosno ključevi mogu biti:

- Atributi
- Primarni ključevi
- Alternativni ključevi
- Strani ključevi

4. Beleške.

IDEF1X model se sastoji od jednog ili više pogleda i definicija entiteta i domena (atributa) koji se koriste u pogledima. Svaki model mora da bude praćen izjavom o svrsi, koja opisuje zašto je model projektovan, izjavom o delokrugu, koja opisuje područje koje model obuhvata i opisom konvencija koje je autor modela koristio.

4.2 UML

ULOGA UML-A

UML služi za specifikaciju, vizuelno predstavljanje, projektovanje i dokumentovanje poslovnih zahteva i programske sistema

UML – Unified Modeling Language je objedinjeni jezik modeliranja koji služi za specifikaciju, vizuelno predstavljanje, projektovanje i dokumentovanje poslovnih zahteva i programskih sistema. Iako je UML objektno orijentisan, njegova primena nije ograničena samo na modeliranje objektno orijentisanih aplikacija. Prva verzija UML-a stvarana je između 1995. i 1997. godine objedinjavanjem tri postojeće objektno orijentisane metode za modeliranje čiji su autori Booch, Rumbaugh i Jacobson [8]. Object Management Group (OMG) je 1997. godine usvojio verziju 1.1 kao standard za UML. OMG je juna 1998. izdao verziju 1.2, a u jesen

1998. godine i poboljšanu verziju 1.3. Jedan deo specifikacije verzije UML 2.0 je usvojen 2004. godine.

UML za modeliranje koristi dijagrame. Dijagrami mogu da se crtaju i ručno, ali se uobičajeno koriste specijalni programi za UML modeliranje, koji imaju grafičke editore sa unapred pripremljenim grafičkim modelima komponenata koje koristi UML jezik. Ovde će se samo kratko prikazati osnovni tipovi modela koje koristi UML 2.0, a zatim će se, prikazati korišćenje UML-a za modeliranje baza podataka.

UML 2.0 definiše 13 osnovnih tipova dijagrama, koji su svrstani u dve opšte grupe:

- Dijagrami za strukturno modeliranje (**Structural Modeling Diagrams**)
- Dijagrami za modeliranje ponašanja (**Behavioral Modeling Diagrams**)

STRUKTURNI MODELI

U grupu struktturnih modela spadaju dijagrami paketa, klase, objekata, kompozitnih struktura, komponenata i razmeštaja

Strukturalni modeli definišu statičku arhitekturu modela. Oni se koriste da definišu elemente koje čine model, kao što su: **klase, objekti, interfejsi i fizičke komponente**. Pored toga **strukturalni modeli se koriste za modeliranje veza i zavisnosti između elemenata**. U grupu struktturnih modela spadaju:

- **Dijagrami paketa**, koji se koriste da podele model u logičke kontejnere ili pakete i opišu interakciju između njih na visokom nivou
- **Dijagrami klase ili strukturalni dijagrami**, definišu osnovne gradivne blokove modela: tipove, klase i ostale elemente koje se koriste za gradnju modela.
- **Dijagrami objekata**, koji pokazuju kako su povezane i korišćene instance strukturalnih elemenata tokom izvršenja programa.
- **Dijagrami kompozitnih struktura**, koji obezbeđuju prikaz elemenata strukture u slojevima i fokusiraju se na unutrašnje detalje, konstrukcije i veze.
- **Dijagrami komponenata**, koji se koriste za modeliranje kompleksnih struktura, uobičajeno sagrađenih od jedne ili više klasa.
- **Dijagrami razmeštaja** se koristi za prikaz fizičkog rasporeda hardvera i softvera.

MODELI PONAŠANJA

Modeli ponašanja se koriste za prikaz interakcija i trenutnih stanja u modelu tokom vremena

Modeli ponašanja se koriste za prikaz interakcija i trenutnih stanja u modelu tokom vremena. U ovu grupu modela spadaju:

- **Dijagrami slučajeva korišćenja**, koji se koriste za modeliranje interakcije korisnik/sistem.
- **Dijagrami aktivnosti**, koji se koriste za opis procesa.
- **Dijagrami stanja**, koji se koriste za prikaz stanja modela tokom vremena.

- **Dijagrami komunikacije**, koji se koriste za prikaz mreža i komunikaciju između objekata u mreži.
- **Sekvencijalni dijagrami**, prikazuju redosled razmena poruka između objekata korišćenjem vertikalne vremenske linije.
- **Vremenski dijagrami**, prikazuju stanje objekata tokom vremena, kao i poruke koje dovode do promene stanja.
- **Dijagram pregleda interakcija**, koji omogućuje da se kombinuju interakcioni fragmenti sa tačkama odlučivanja i tokovima.

▼ Poglavlje 5

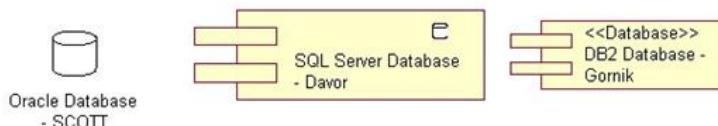
MODELIRANJE BAZE PODATAKA UZ POMOĆ UML-a

DEFINISANJE BAZE PODATAKA U UML-U

Za definisanje baze podataka u UML-u se koristi stereotip <<Database>>

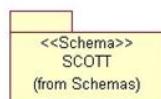
UML omogućuje modeliranje relacionih i objektno-relacionih baza podataka. U UML-u je moguće definisati novi specijalan element modela korišćenjem postojećih. Stereotip je mehanizam koji omogućuje definisanje novih elemenata modela. Stereotip ima ime koje se navodi između znakova „<<“ i „>>“. Zbirka specifičnih stereotipa koji su specijalno projektovani za modeliranje baza podataka se naziva UML profil za projektovanje baza podataka. On omogućuje da se modeliraju različiti koncepti koji se koriste u bazama podataka, kao što su: tabele, ključevi, relacije, pogledi, uskladištene procedure i domeni.

Za definisanje baze podataka u UML-u se koristi stereotip <<Database>> u dijagramu komponenata. Kao komponenta, baza podataka mora da ima ime. Za prikaz stereotipa koji predstavlja bazu podataka u dijagramu komponenata se mogu koristiti tri reprezentacije koje su predstavljene na narednoj slici.



Slika 5.1 Tri reprezentacije za prikaz stereotipa koji predstavlja bazu podataka [Izvor: Autor]

Opis modela podataka u nekoj bazi se čuva u šemi. Šeme se prikazuju u dijagramu paketa kao stereotip <<Schema>>. Na narednoj slici je prikazana reprezentacija paketa kojim je modelirana šema baze podataka **SCOTT**.

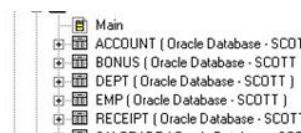


Slika 5.2 Reprezentacija paketa kojim je modelirana šema baze podataka SCOTT [Izvor: Autor]

TABELE I ATRIBUTI U UML-U

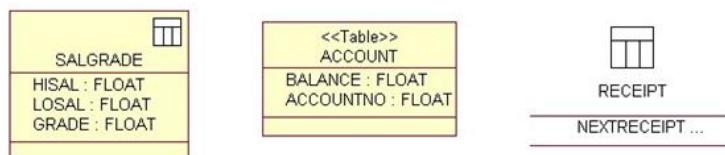
U UML-u se tabele prikazuju kao klase, pri čemu se koristi stereotip <<Table>>.

Tabela je osnovni element modeliranja baza podataka. Ona predstavlja set zapisa koji imaju iste atributе. U UML-u se tabele prikazuju kao klase, pri čemu se koristi stereotip <<Table>>. Za svaku klasu se definišu i atributi. Atributi se mogu označiti kao ključevi, Null atributi i jedinstveni (kada nije moguće ponavljanje iste vrednosti atributa u nekom drugom zapisu). Na sledećoj slici su prikazane tabele šeme **SCOTT**. Ova šema sadrži tabele Account, Bonus, Dept, Emp, Receipt itd



Slika 5.3 Tabele šeme SCOTT [Izvor: Autor]

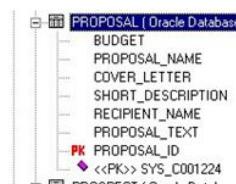
Sama tabela se u dijagramu predstavlja nekom od reprezentacija prikazanih na sledećoj slici.



Slika 5.4 Predstavljanje tabele [Izvor: Autor]

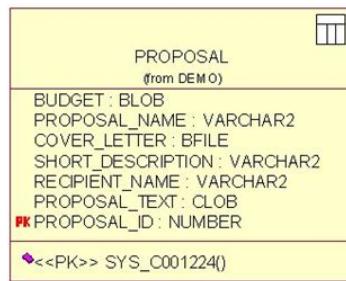
Smeštanjem svih tabela jedne šeme u paket, kreira se asocijacija između tabele i šeme.

Atributi koji predstavljaju primarne i strane ključeve se posebno označavaju (**tagged values**). Za primarni ključ se koristi tag PK, a za strane ključeve FK. Na sledećoj slici su prikazani atributi tabele **Proposal**, koja ima primarni ključ *Proposal_ID*



Slika 5.5 Atributi tabele Proposal [Izvor: Autor]

Na sledećoj slici je prikazana reprezentacija tabele **Proposal** u dijagramu.

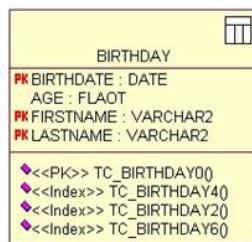


Slika 5.6 Reprezentacija tabele Proposal [Izvor: Autor]

MODELIRANJE TABELE SA INDEKSIMA I VEZAMA

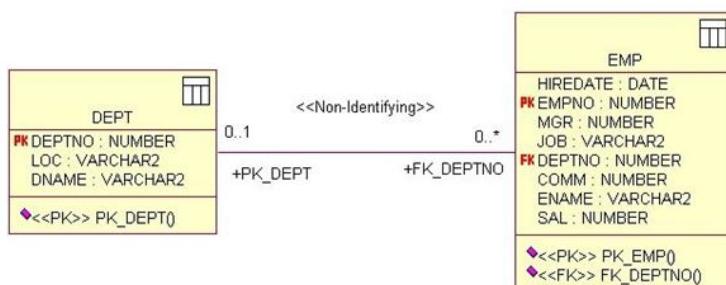
Za modeliranje indeksa se koristi stereotip <<Index>>.

Za modeliranje indeksa se koristi stereotip <<Index>>. Na sledećoj slici je prikazana reprezentacija tabele sa indeksima.



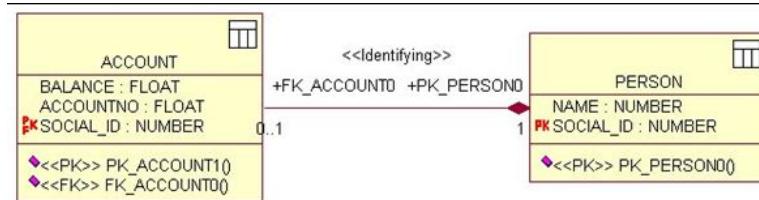
Slika 5.7 Reprezentacija tabele sa indeksima [Izvor: Autor]

Veze se u UML dijagramima mogu prikazati identifikacionim i neidentifikacionim stereotipovima. Na sledećoj slici je prikazana identifikaciona veza između tabela **Dept** i **Emp**.



Slika 5.8 Identifikaciona veza između tabela Dept i Emp [Izvor: Autor]

Identifikaciona veza je veza između dve zavisne tabele, gde podređena tabela ne može da egzistira bez nadređene tabele. U sledećem primeru je tabela **Person** nadređena, a **Account** podređena.



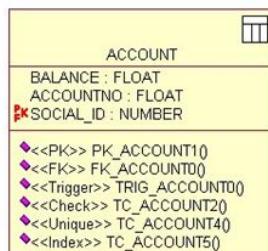
Slika 5.9 Primeru nadređene tabele Person i podređene Account [Izvor: Autor]

TIPOVI PODATAKA I OGRANIČENJA

Svakom atributu tabele moguće je dodeliti tip podataka, podrazumevajući vrednost i ograničenja.

Svakom atributu tabele moguće je dodeliti tip podataka, podrazumevajući vrednost i ograničenja. Primeri tipova podataka su: **char, date, float, long, number, nvarchar2, raw, varchar2**.

Ograničenja se mogu odnositi samo na neki atribut ili na celu tabelu. Sva ograničenja se modeliraju kao stereotipne operacije. Na sledećem primeru je prikazana klasa koja modelira tabelu **Account**, koja ima primarni i strani ključ, okidače (**trigger**), indeks i atribut koji mora da ima jedinstvene vrednosti.



Slika 5.10 Klasa koja modelira tabelu Account [Izvor: Autor]

Okidači se modeliraju stereotipom **<<Trigger>>**. Oni pokreću neku operaciju kada se ispunи neki uslov ili dogodi neki događaj.

Provera dozvoljenih vrednosti atributa se modelira stereotipom **<<Check>>**, a jedinstvenost vrednosti atributa stereotipom **<<Unique>>**.

▼ Poglavlje 6

Pokazna vežba: MODELIRANJE PODATAKA

PRIMER: MODELIRANJE PODATAKA

U ovom primeru će biti predstavljen proces modeliranja podataka za sledeći opis dela sistema za bolnicu

Predviđeno vreme pokazne vežbe je 30 minuta.

Prvi korak u modeliraju podataka za neki informacioni sistem je analiza zahteva korisnika. U ovom primeru će biti predstavljen proces modeliranja podataka za sledeći opis dela sistema za bolnicu:

Članovi Vašeg tima rade na razvoju sistema za lokalnu bolnicu u kome će se čuvati informacije o pacijentima, brojevima soba pacijenata, doktorima pacijenata, receptima, i informacije o lekovima.

Vaš zadatak je da napravite listu biznis pravila koja će se koristiti u izradi sistema. Napišite 10 strukturnih pravila, 5 proceduralnih pravila i 2 programerska pravila. Svako pravilo bi trebalo da bude po jedna rečenica.

Na osnovi ovih pravila, nacrtajte ER dijagram.

KONCEPTUALNI MODEL: IDENTIFIKOVANJE ENTITETA I ATRIBUTA

Entiteti postoje nezavisno i mogu se jedinstveno identifikovati

Prvi korak u izradi konceptualnog modela je identifikovanje entiteta i atributa u modelu. Kada imamo tekst sa opisom sistema, dobar početni korak je napraviti listu imenica u tekstu, ovo su dobri kandidati za entitete i atrbute.

Entiteti mogu biti konkretnе stvari, ali i apstraktni pojmovi. Entiteti postoje nezavisno i mogu se jedinstveno identifikovati.

Atributi opisuju entitete, i ne mogu postojati nezavisno od entiteta. Npr. pacijent je entitet: to je konkretna stvar koju možemo jedinstveno identifikovati i razlikovati od drugih. Opisan je svojim atributima, kao što su ime, prezime, starost itd. Starost ne može biti entitet, jer njegovo postojanje nema smisla bez entiteta koji opisuje.

Iz datog opisa sistema možemo identifikovati sledeće entitete i njihove attribute. Svakom entitetu ćemo dodeliti identifikator:

Pacijent: ID, pacijentlme

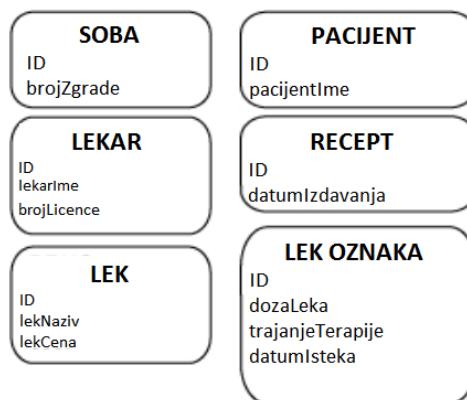
Soba: ID, brojZgrade

Lekar: ID, ime, brojLicence

Recept: ID, datumIzdavanja

Lek: ID, lekNaziv, lekCena

Lek oznaka: ID, dozaLeka, trajanjeTerapije, datumIsteka



Slika 6.1 Prikaz entiteta i njihovih atributa [Izvor: Autor]

REŠENJE

Rešenje zadatka

Definisani zahtevi:

Strukturna pravila:

Pacijent je neko ko je primljen u bolnicu

Svaka dodeljena soba mora imati svoj broj i broj zgrade

Svaka soba može imati jednog ili više pacijenata

Lekar mora imati važeći broj licence

Svaki lek mora biti prepisan od strane lekara

Svaki lek mora imati oznaku koja pokazuje broj, dozu, trajanje terapije i datum isteka leka

Svaki lek mora imati informacije o šifri, nazivu i ceni

Svaki recept mora imati broj i datum

Svaki lekar može biti dodeljen jednom ili više pacijenata

Svaki pacijent mora da ima dodeljenog lekara

Proceduralna pravila:

Izmene recepata mogu vršiti samo licencirani lekari

Pacijenti ne mogu ponovo popunjavati recepte bez potpisa lekara

Lekari ne mogu uklanjati oznake sa lekova

Pacijenti ne mogu menjati sobe bez preporuke lekara
 Medicinski tehničari ne mogu pacijentima dodeljivati druge sobe bez potvrde lekara

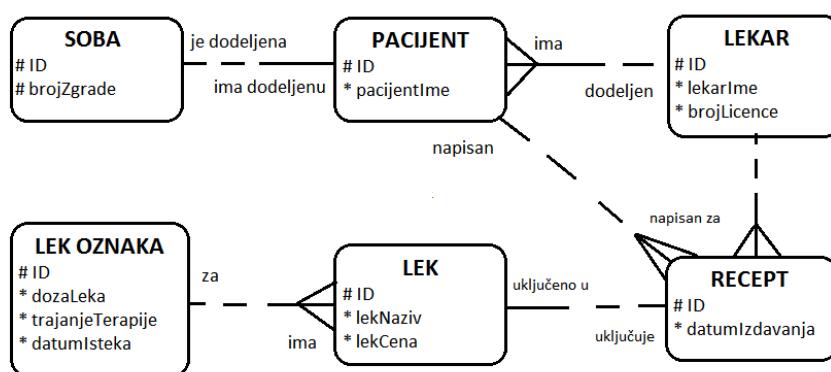
Programerska pravila:

Lekovi se naplaćuju po važećim cenama u vreme otpusta pacijenta
 Honorar lekara može uključivati dodatne cene u zavisnosti od komplikacija, dodatnih zahteva pacijenta, itd.

E/R DIJAGRAM

entiteti su predstavljeni pravougaoncima, atributi elipsama.

Izradjeni model možemo predstaviti dijagramom na slici. Ovaj dijagram koristi Chen notaciju: entiteti su predstavljeni pravougaoncima, atributi elipsama.



Slika 6.2 Dijagram konceptualnog modela [Izvor: Autor]

IDENTIFIKOVANJE RELACIJA

Relacije možemo identifikovati glagolima koji povezuju entitete

Relacije određuju odnos izmedju entiteta. Relacije možemo identifikovati glagolima koji povezuju entitete. Npr. pacijent ima sobu, lek ima oznaku it. Kardinalnost relacije određuje brojnu relaciju izmedju entiteta i mogu biti **jedan na jedan** (1-1), **jedan na više** (1-n) i **više na više** (n-m).

Iz teksta možemo identifikovati prvu relaciju u modelu:

Pacijent-Lekar N-M, jedan pacijent može imati jednog lekara, a jedan lekar može imati više pacijenata.

Na prvi pogled, relacija izmedju korisnika i knjige je takođe N-M, jer jedan korisnik može posuditi više knjiga, i jednu knjigu može posuditi više korisnika. Međutim, traži se da sistem pamti datum posudjivanja i vraćanja knjige.

U ovom slučaju, jedno rešenje je da uvedemo novi entitet **Zaduženje** sa atributima datum posudjivanja, i datum vraćanja, koji predstavlja specifičnu instancu posudjivanja knjige. Ovo je primer entiteta koji je apstraktan pojam, i ne pominje se eksplisitno u zahtevima.

Sada imamo sledeće relacije:

Pacijent-Lekar N-M, jedan pacijent može imati jednog lekara, a jedan lekar može imati više pacijenata.

Lek-Oznaka 1-N, jedan lek može imati samo jednu oznaku dok jedna oznaka se može nalaziti na više lekova

Lekar-Recept 1-N, jedan recept može biti prepisan od samo jednog lekara a jedan lekar može pisati više recepata

Recept-Pacijent 1-N, jedan pacijent može imati više prepisanih recepata a jedan recept može biti prepisan za samo jednog pacijenta

TRANSFORMACIJA KONCEPTUALNOG U RELACIONI MODEL

Entiteti odgovaraju tabelama

Izradjeni konceptualni model transformišemo u relacioni koristeći sledeća pravila:

- Entiteti odgovaraju tabelama, i identifikatori entiteta su primarni ključevi
- Kod relacije 1-N, u tabelu koja odgovara N strani relacije dodaje se kolona koja je foreign key u tabelu na strani 1
- Kod relacije N-M, dodaje se tabela asocijacija sa dve kolone koje su foreign key u odgovarajuće tabele.

U ovoj fazi modelovanja možemo da identifikujemo i tipove podataka atributa.

Primenom ovih pravila na prethodni model dobijamo sledeće table:

Soba: ID (int), brZgrade(int)

Pacijent: ID(int), pacijentIme(varchar), ID_soba(fk)

Lekar: ID (int), IME (varchar), brLicence(int)

Oznaka: ID (int), dozaLeka(int), trajanjeTerapije(int), datumIsteka(date)

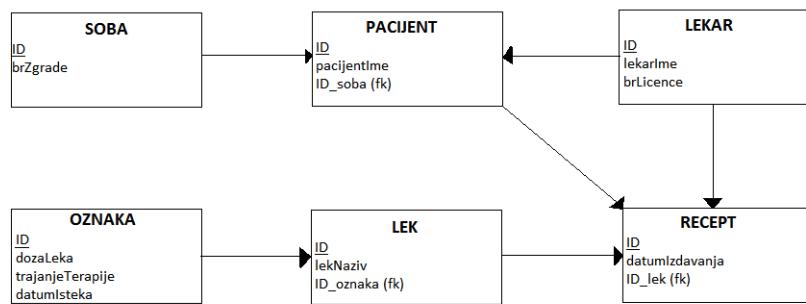
Lek: ID(int), lekNaziv(varchar), ID_oznaka(fk)

Recept: ID(int), datumIzdavanja(date), ID_lek(fk)

RELACIONI DIJAGRAM

Na slici je dat relacioni dijagram.

Na slici je dat relacioni dijagram.



Slika 6.3 Relacioni dijagram [Izvor: Autor]

✓ Poglavlje 7

Pokazna vežba: SQL AGREGATNE FUNKCIJE

AGREGATNE FUNKCIJE

Agregatne funkcije se koriste kada je potrebno dobiti izveštaj sa rezultatima u kojima ima više redova iz tabele

Predviđeno vreme pokazne vežbe je 20 minuta.

Često je potrebno dobiti izveštaj sa rezultatima u kojima učestvuju više redova iz tabele, na primer, zbir svih vrednosti itd. Za ovo se koriste agregatne funkcije.

Primer 1 (Predviđeno vreme za izradu primera je: 5 minuta)

Za datu tabelu **Osoba**, pronaći koliko se različitih gradova u njoj spominje.

Prezime	Ime	Grad	Starost
Milanović	Katarina	Novi Sad	30
Stamenković	Andelija	Beograd	24
Milošević	Joksim	Novi Sad	45
Jovanović	Dejan	Niš	32
Arambašić	Dejan	Beograd	47

Slika 7.1.1 Početna Osoba [Izvor: Autor]

Rešenje:

Komanda uz pomoć koje se utvrđuju sve različite vrednosti u nekoj koloni je sledeća:

SELECT DISTINCT ime_kolone FROM ime_tabele;

Za ovaj primer to bi značilo:

SELECT DISTINCT Grad FROM Osoba;

Kao rezultat dobija se tabela na slici 2.

Grad
Beograd
Niš
Novi Sad

Slika 7.1.2 Rezultat DISTINCT upita [Izvor: Autor]

SQL UPIT ZA IZRAČUNAVANJE VREDNOSTI NEKE FUNKCIJE

SELECT funkcija(ime_kolone) FROM ime_tabele;

Primer 2 (Predviđeno vreme za izradu primera je: 5 minuta)

Za tabelu **Osoba** iz Primera 1 izračunati prosečnu starost osoba čiji se podaci nalaze u tabeli.

Rešenje:

Generalno, SQL upit za izračunavanje vrednosti neke funkcije je:

SELECT funkcija(ime_kolone) FROM ime_tabele;

Za ovaj primer to bi značilo sledeće:

SELECT AVG(Starost) FROM Osoba;

Rezultat je prikazan na slici 3.

Query4	
Expr1000	35.6

Slika 7.1.3 Rezultat AVG funkcije [Izvor: Autor]

U tabeli 1 dat je spisak i opis nekih najvažnijih Access funkcija.

Funkcija	Opis
AVG(kolona)	Određuje srednju vrednost kolone
COUNT (kolona)	Određuje broj vrsta u jednoj koloni (izuzev onih koji imaju vrednost NULL)
FIRST (kolona)	Određuje vrednost prvog zapisa u zadatoj koloni
LAST (kolona)	Određuje vrednost poslednjeg zapisa u zadatoj koloni
MAX(kolona)	Određuje najveću vrednost u zadatoj koloni
MIN(kolona)	Određuje najmanju vrednost u zadatoj koloni
SUM(kolona)	Određuje sumu svih vrednosti u zadatoj koloni

Slika 7.1.4 Tabela-1 Access funkcije [Izvor: Autor]

GROUP BY

GROUP BY iskaz se koristi zajedno sa agregatnom funkcijom da grupiše rezultate jedne ili više kolona.

Primer 3 (Predviđeno vreme za izradu primera je: 5 minuta)

Za tabelu **Osoba** izračunati prosečnu starost osoba po gradovima.

Rešenje:

Za rešavanje ovog problema koristi se **GROUP BY** izraz sledeće sintakse:

SELECT ime_kolone1, SUM(ime_kolone2) FROM ime_tabele GROUP BY ime_kolone1;

Za ovaj primer to znači:

SELECT Grad, AVG(Starost) FROM Osoba GROUP BY Grad;

Kao rezultat dobija se tabela na slici 5.

Grad	Expr1001
Beograd	35.5
Niš	32
Novi Sad	37.5

Slika 7.1.5 Rezultat upita [Izvor: Autor]

GROUP BY izraz grupiše podatke po vrijednosti jedne ili više kolona. Ovo znači da će group by izraz grupisati sve redove sa istom vrijednosti datih kolona u jedan red.

U prethodnom slučaju smo u SELECT izrazu koristili jednu agregiranu i jednu neagregiranu kolonu (Starost i Grad).

Ako bismo izostavili GROUP BY izraz, prosečna vrednost bi se računala za čitavu tabelu, a ne samo za dati grad.

Takodje, ako bismo zadali upit za ime i prezime, i grupisali ga po imenu, dobili bi samo jedan red za dato ime, mada u bazi postoji više redova sa istim imenom.

Iz ovih slučajeva, možemo izvući sledeće pravilo:

Kada u SELECT izrazu imamo i neagregirane i agregirane kolone, rezultati se moraju grupisati po vrednosti svih neagregiranih kolona.

Neke baze će dozvoliti i upite koje krše ovo pravilo, ali u tom slučaju rezultati neće biti smisleni.

HAVING

HAVING je uveden jer se WHERE izraz ne može koristiti sa agregatnim funkcijama

Izraz **HAVING** omogućava i definisanje nekih dodatnih uslova. Opšti upit glasi:

**SELECT ime_kolone1 FROM ime_tabeleGROUP BY ime_kolone1HAVING
SUM(ime_kolone2) uslov vrednost;**

Na primer, ukoliko se želi samo spisak gradova gde je prosečna starost stanovnika veća od ili jednaka 35, koristio bi se sledeći izraz:

SELECT Grad FROM OsobaGROUP BY GradHAVING AVG(starost)>= 35;

HAVING je uveden jer se WHERE izraz ne može koristiti sa agregatnim funkcijama.

HAVING je najlakše shvatiti kao WHERE koji radi sa funkcijama, ali to poredjenje nije u potpunosti tačno.

WHERE izraz filtrira podatke pre agregacije, tj. grupisanja, dok se HAVING izraz izvršava posle grupisanja podataka.

UNION I UNION ALL

Za objedinjavanje podataka koji se nalaze u dva različite tabele koriste se naredbe UNION i UNION ALL

Primer 4 (Predviđeno vreme za izradu primera je: 5 minuta)

Neka su date dve tabele **Student_klasično** i **Student_internet** koje sadrže spisak studenata koji studiraju na klasičan način, odnosno preko Interneta. Kreirati tabelu u kojoj će se nalaziti imena svih studenata bez obzira na način studiranja.

Student_klasično	
indeks	student
1	Petar Petrović
2	Marko Marković

Student_internet	
indeks	student
1	Kosta Kostić
2	Marko Marković

Slika 7.1.6 Tabele Student_klasično i Student_internet [Izvor: Autor]

Rešenje:

Tabele se kreiraju i snimaju na način prikazan na vežbama druge nedelje.

Za objedinjavanje podataka koji se nalaze u dva različite tabele koriste se naredbe UNION i UNION ALL. Ove naredbe, dva različita upita pretvaraju u jedan formirajući jednu tabelu. Ova opcija najbolje funkcioniše kod dve tabele sa sličnim kolonama, jer je neophodno ispuniti uslov da imaju iste tipove.

Primenom sledećeg obrasca:

**SELECT Student FROM Student_klasično
UNION SELECT Student FROM Student_internet;**

dobija se sledeći rezultat prikazan na slici 7.

student
Kosta Kostić
Marko Marković
Petar Petrović
*

Slika 7.1.7 Rezultat UNION [Izvor: Autor]

Primenom opcije UNION ALL rezultat je nešto drugačiji (slika 8):

```
SELECT Student FROM Student_klasično
UNION ALL SELECT Student FROM Student_internet;
```

student
Kosta Kostić
Marko Marković
Petar Petrović
Marko Marković
*

Slika 7.1.8 Rezultat UNION ALL operatora [Izvor: Autor]

✓ 7.1 Pokazna vežba: MS ACCESS FORME I IZVEŠTAJI

PRIMER 1 - CREATE TABLE

Kreiranje baze Imenik sa tabelom Telefon i kolonama ime, prezime i telefon koristeći CREATE TABLE i kreiranje formi kroz Wizard opcije MS Access-a

Predviđeno vreme pokazne vežbe je 30 minuta.

Uz pomoć Access kreirati bazu **Imenik**, i u njoj tabelu **Telefon** sa sledećim kolonama: **ime**, **prezime** i **telefon**. Prva dva podatka su tekstualna, maksimalne dužine 30 karaktera, a poslednji celobrojna vrednost. Kreirati formu koja će omogućiti unos i brisanje podataka, kao i prikazivanje izveštaja o svim podacima u tabeli. Koristeći forme, uneti podatke iz sledeće tabele i prikazati izveštaj.

ime	prezime	Telefon
Katarina	Milanović	55 555
Andelija	Stamenković	77 777
Joksim	Milošević	88 888

Slika 7.2.1 Podaci u tabeli [Izvor: Autor]

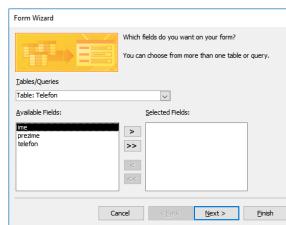
Rešenje:

Baza i tabela formiraju se na način prikazan u okviru druge vežbe, a uz pomoć sledećeg SQL upita:

```
CREATE TABLE Telefon(
```

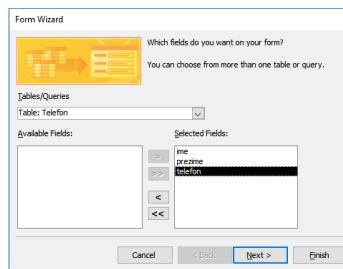
ime VARCHAR(30), prezime VARCHAR(30), telefon INTEGER;

Za kreiranje odgovarajuće forme za unos koristi se opcija **Forms** levog dela prozora .



Slika 7.2.2 Wizard opcija [Izvor: Autor]

Željeni obrazac može se kreirati uz pomoć opcije **Create form by using wizard**. Nakon dvostrukog klika na ovu opciju dobija se prozor kao na slici 3, gde se prolaskom niza koraka omogućava efikasno kreiranje formulara. Na početku je potrebno izabrati koje kolone tabele će se prikazivati. Kako će se obrazac u ovom primeru koristiti i za unos podataka, poželjno je da to budu sve definisane kolone. To se postiže opcijom **>>** u prikazanom prozoru, a potom se bira opcija **Next**.

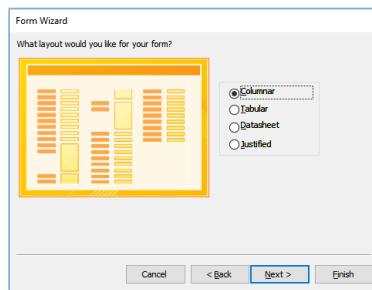


Slika 7.2.3 Odabrana polja iz tabele [Izvor: Autor]

PRIMER 1- DEFINISANJE IZGLEDA FORME

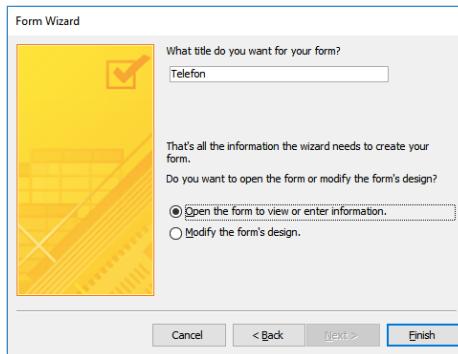
Izgled forme se definiše opcijama Columnar i Standard

U naredna dva prozora prozora (slika 4 i slika 5) definiše se izgled forme. Za ovaj primer dovoljno je koristiti prepostavljene opcije **Columnar** i **Standard**.



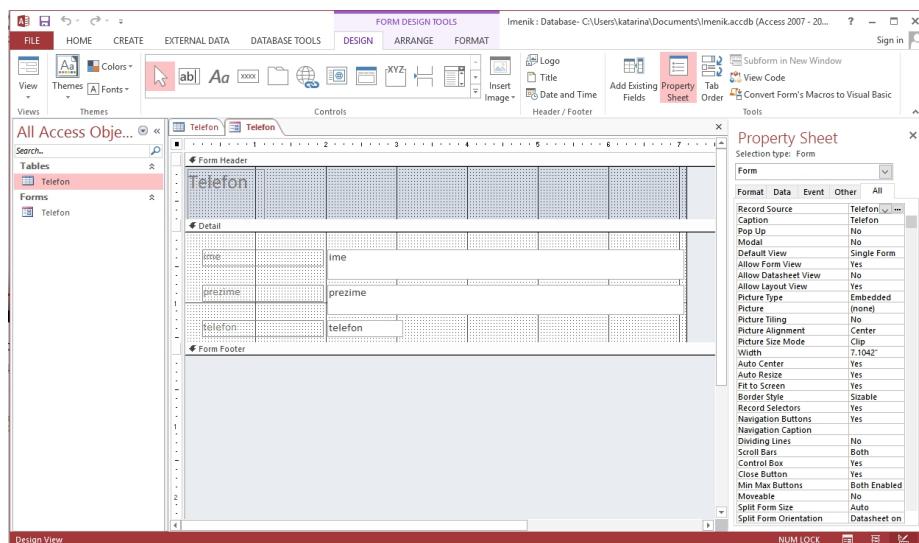
Slika 7.2.4 Columnar [Izvor: Autor]

Konačno, u poslednjem **Form Wizard** prozoru formi se može dati ime navođenjem u polju na vrhu prozora.



Slika 7.2.5 Imenovanje forme [Izvor: Autor]

Aktiviranjem opcije **Modify the form's design** u prethodnom prozoru korisniku se pruža mogućnost da definisanu formu dodatno menja. Sa **Finish** se završava inicijalni proces. U prozoru koji će se pojaviti forma se može dodatno menjati, u ovom slučaju dodavanjem opcija za dodavanje, brisanje i prikazivanje izveštaja. U prozoru sa alatima bira se opcija **Command Button** Slika 6.

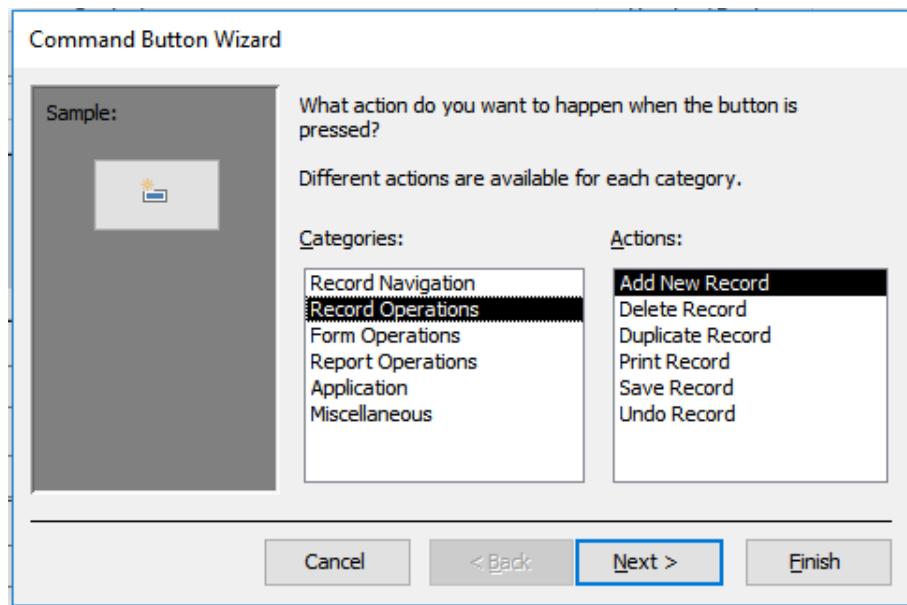


Slika 7.2.6 Prikaz dizajna forme [Izvor: Autor]

PRIMER 1 - KREIRANJE DUGMETA ZA UNOS PODATAKA U TABELU

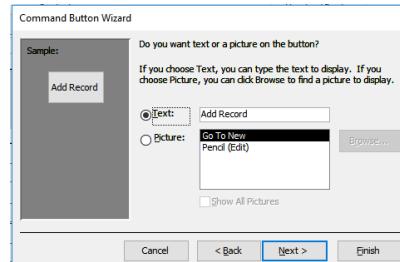
Da bi nacrtano dugme predstavljalo opciju za unos podatka u tabelu potrebno je u levom delu prozora izabrati Record Operations, a potom u desnom delu opciju Add New Record

Kada se u meniju sa alatima izabere dugme, potrebno je postaviti ga u formu. Na slici 7 je prikazan prozor koji se otvori kada se dugme postavi na formu



Slika 7.2.7 Definsanje funkcije dugmeta [Izvor: Autor]

Da bi nacrtano dugme predstavljalo opciju za unos podatka u tabelu potrebno je u levom delu prozora izabrati **Record Operations**, a potom u desnom delu opciju **Add New Record**. Nakon klika na **Next** definiše se izgled dugmeta. Za ovaj primer može se izabrati opcija sa prikazivanjem teksta **Dodaj**.



Slika 7.2.8 Dodavanje novog unosa [Izvor: Autor]

Poslednja opcija je definisanje imena dugmeta. SaFinish se završava kreiranje opcije za dodavanje podatka u tabelu.



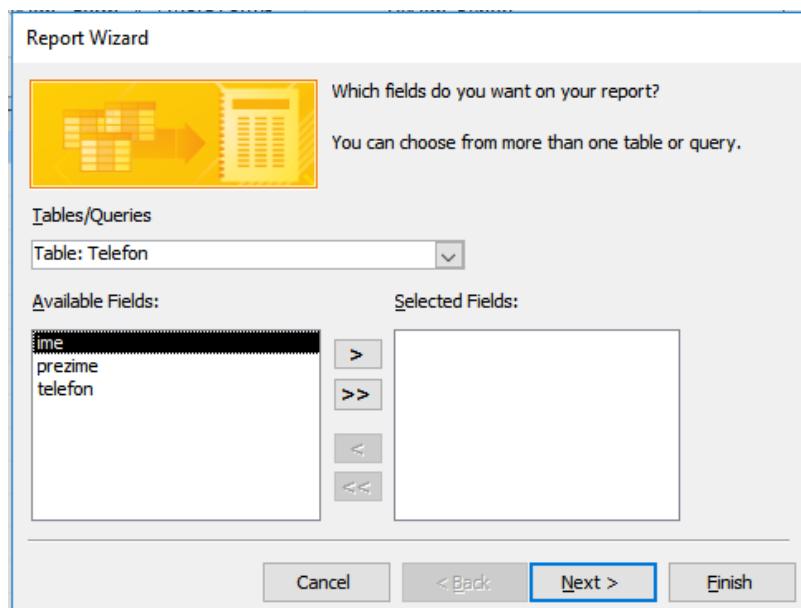
Slika 7.2.9 Imenovanje dugmeta [Izvor: Autor]

PRIMER 1 - PRIKAZIVANJE IZVEŠTAJA

Za prikazivanje odgovarajućeg izveštaja potrebno je u prozoru **Imenik:Database** kliknuti na opciju **Reports** i dva puta na **Create report by using wizard**.

Za prikazivanje odgovarajućeg izveštaja potrebno je u prozoru **Imenik:Database** kliknuti na opciju **Reports** i dva puta na **Create report by using wizard**.

U nastavku se opcijom >> biraju sve kolone za tabele za prikazivanje u izveštaju (slika 11).



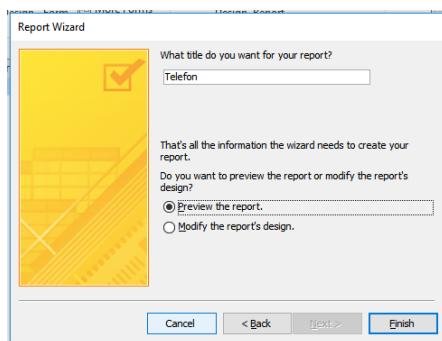
Slika 7.2.10 Report Wizard [Izvor: Autor]

U naredna dva koraka vrši se definisanje izgleda izveštaja.

Prozor koji će se prikazati na ekranu predstavlja izgled obrasca izveštaja. Isti se zatvara i klikne se na prozor **Telefon:Form**. Na već prikazan način iscrtava se dugme koje će služiti za prikazivanje izveštaja i u prozoru se biraju opcije **Report Operations** i **Preview Report** (slika 13).



Slika 7.2.11 Izgled forme [Izvor: Autor]

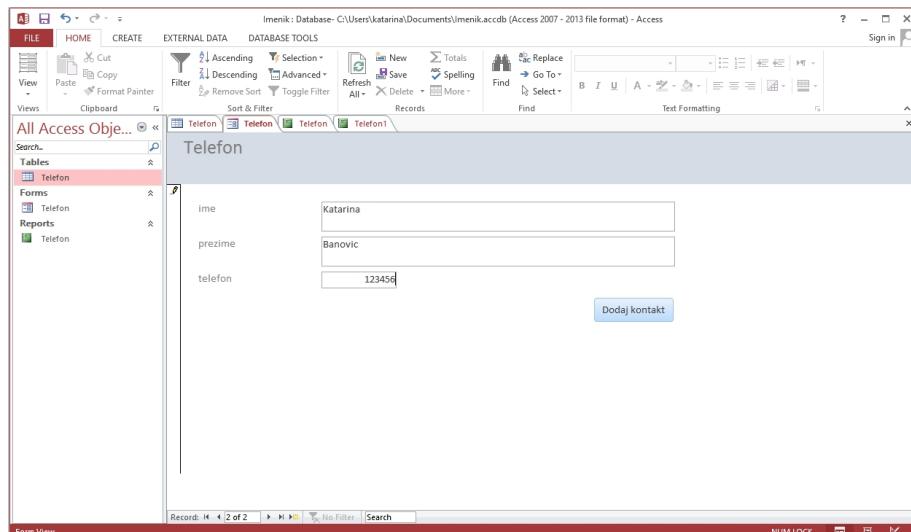


Slika 7.2.12 Imenovanje izveštaja [Izvor: Autor]

PRIMER 1 - IZBOR IZVEŠTAJA

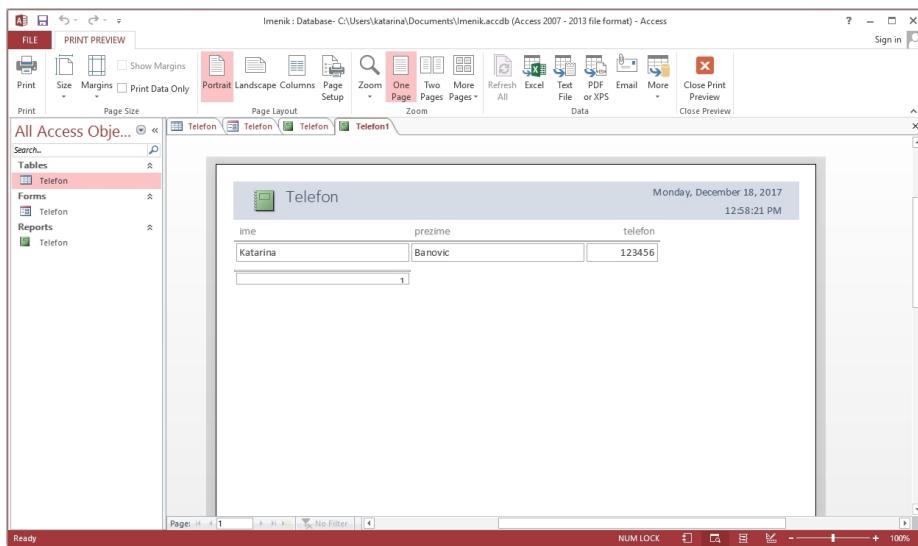
U slučaju da je definisano više izveštaja preko Command Button Wizard-a je moguće odrediti izbor

Na slići 13 prikazan je popunjena forma za kreiranje izveštaja.



Slika 7.2.13 Popunjena forma [Izvor: Autor]

Opcijom **Finish** završava se proces kreiranja formulara. Opcijom **Save** na liniji sa standardnim alatima snima se izveštaj. Kada se u delu **Forms** dva puta kratko klikne na formu **Telefon** dobija se prozor (slika 14) koji omogućava unos i brisanje podataka, kao i njihovo prikazivanje u obliku izveštaja.



Slika 7.2.14 Prikaz izveštaja [Izvor: Autor]

PRIMER 2 – KREIRANJE BAZE

Kreiranje baze Katalog sa tabelama Izdanja (kolone: id, naziv, autor, izdavač) i Izdavači (kolone: id, naziv)

Uz pomoć SQL jezika u Access-u, kreirati bazu **Katalog** sa tabelom **Izdanja** koja ima sledeće kolone: **id** tipa integer koja je ujedno primarni ključ, **naziv** i **autor** tipa varchar, kao i kolonu **izdavač** tipa integer. Deo baze je i tabela **Izdavači** sa kolonama: **id** tipa integer koja je i primarni ključ i **naziv** tipa varchar. Odnosom jedan prema više povezati dve tabele baze, tako da se kolona izdavač prve tabele odnosi na naziv izdavača u drugoj tabeli. Kreirati obrasce za unos podataka u obe tabele i prikazivanje izveštaja o nazivu izdanja, autoru i izdavaču. U tabele uneti sledeće podatke i prikazati izveštaj:

Izdanje			
id	naziv	autor	izdavač
1	Na stranputici	Lili Brand	1
2	Svetlost dana	Grejem Swift	2
3	Mamac	David Albahari	1
4	Opsada	Helen Danmor	2

Izdavač	
id	naziv
1	Narodna knjiga
2	Laguna

Slika 7.2.15 Tabele [Izvor: Autor]

Rešenje:

Tabele se kreiraju na već poznat način sledećim SQL naredbama:

CREATE TABLE Izdavač (

id integer primary key,

naziv varchar);

CREATE TABLE Izdanja(

id integer primary key,

naziv varchar,

autor varchar,

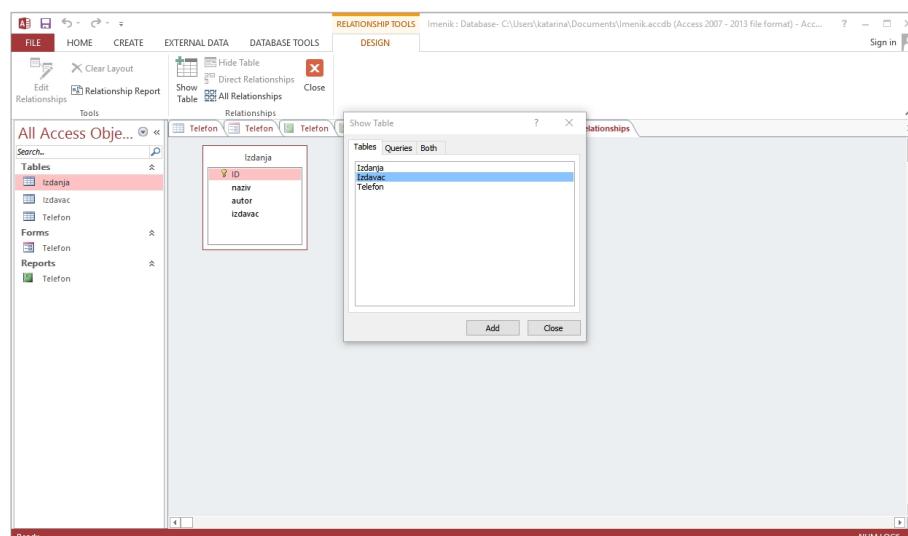
izdavač integer references Izdavač);

Uz pomoć ključne reči **references**, obeležene žutom bojom u datom primeru, uspostavlja se relacija jedan prema više između dve tabele. Veoma je važan i redosled kreiranja tabela. U ovom primeru potrebno je prvo kreirati tabelu **Izdavač**, jer će se ona kasnije koristiti prilikom kreiranja tabele **Izdanja**.

PRIMER 2 - PROVERA RELACIJA

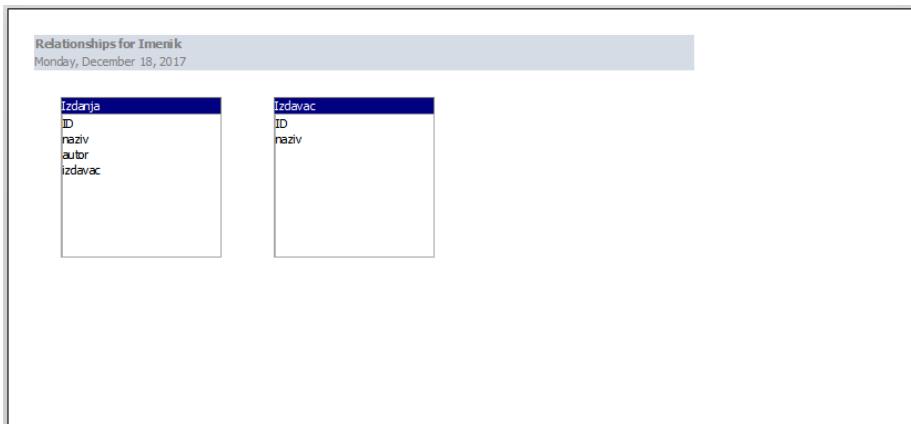
Postojanje relacija se proverava opcijom Relationships, a izveštaj se kreira u delu Report sa opcijom Create report by wizard

Da bi se proverilo postojanje relacija, koristi se opcija **Relationships** na liniji sa standardnim alatima. Opcijom Add dodaju se sve željene tabele.



Slika 7.2.16 Relationships opcija [Izvor: Autor]

Klikom na dugme Relathionship report kreira se izveštaj o vezama.



Slika 7.2.17 Relationship report [Izvor: Autor]

MS ACCESS FORME – VIDEO 1

MS Access forme – Kreiranje forme za unos podataka

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

MS ACCESS FORME – VIDEO 2

MS Access forme – Kreiranje izveštaja unetih podataka

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

MS ACCESS FORME – VIDEO 3

MS Access forme – Relacije i kreiranje izveštaja iz više tabela

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 8

Zadaci za samostalni rad

ZADACI ZA VEŽBU

Izraditi model podataka za sledeće sisteme:

Predviđeno vreme za izradu zadatka je: 20 minuta

Izraditi model podataka za sledeće sisteme:

- Informacioni sistem apoteke
- Web prodavnica
- Sistem za upravljanje ljudskim resursima u firmi
- Sajt za aukcije

OPIS ZADATKA ZA SAMOSTALNU VEŽBU

Kreirati bazu podataka Knjižara i tabelu Knjiga

Predviđeno vreme za izradu zadatka je: 25 minuta

Korak 1: Kreirati bazu podataka Knjižara i tabelu Knjiga. Definisati koje attribute tabela Knjiga treba da ima kako bi bilo moguće odraditi upite.

Korak 2: Koristeći MS Access pripremiti formu za unos i izveštaj za prikazivanje podataka iz baze.

Korak 3: U tabelu uneti minimalno 10 redova.

Korak 4: Napisati upit koji će:

- naći knjige koje su izdate posle 1990 godine
- naći prosečnu cenu knjiga
- izlistati sva imena knjiga čije ime počinje i završava se na slovo A
- naći sve knjige koje su skuplje od 3500RSD a jeftinije od 5000RSD.
- sortirati knjige po rastućoj ceni
- naći knjige koje su izdate pre 2000 godine i nisu izdate u Beogradu
- pronaći koliko knjiga ima svaki autor
- pronaći najskuplju knjigu svakog autora
- pronaći autora koji ima najmanje izdatih knjiga
- pronaći sve knjige koje počinju na slovo M i izdate su između 1995 i 2005
- sortirati knjige po datumu izdavanja, počevši od najstarije
- izlistati sve knjige koje su izdate u martu 2015 a koje su jefinije od 1500RSD
- pronaći ime i prezime autora koji ima najviše knjiga žanra fantastika

✓ Poglavlje 9

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ03

Kreirati bazu podataka koristeći MS Access – pripremiti formu za unos, izveštaj za prikazivanje podataka i tražene upite

Predviđeno vreme za izradu domaćeg zadatka 60 minuta.

1. Kreirati bazu podataka **IT210-DZ3-BI**, gde umesto **BI** unosite vaš broj indeksa, i tabelu **Filmovi_BI** koju čine kolone: **naziv, zanr, reziser, godina, rejting, glavni_glumac**. Koristeći MS Access pripremiti formu za unos i izveštaj za prikazivanje podataka iz baze. U tabelu uneti bar 10 redova.
2. Napisati upit koji će naći prosečnu vrednost u koloni rejting.
3. Napisati upit koji će naći sve glumce koji igraju glavnu ulogu u 2 ili više filmova
4. Napisati upit koji će naći filmove koji imaju rejting između 9 i 9.5 kao i njihove režisere
5. Napisati upit koji će pronaći najnoviji film žanra "Komedija"

Kao rešenje domaćeg zadatka poslati Access bazu sa odgovarajućom tabelom, formom i upitim.

Bazu poslati na adresu predmetnog asistenta.

(**napomena: u Subject-u e-mejla napisati IT210 - DZ03**)

▼ ZAKLJUČAK

ZAKLJUČAK

Cilj ovog predavanja je bio da se definicija modela podataka i lista mogućih modela za njihovo modeliranje. Opisane su osobine relacionog modela uključujući relacije, zapise, attribute, domene i operatore. Objasnijene su prve tri normalne forme koje su date u kontekstu anomalija koje mogu da se dese kod kreiranja baza podataka. Takođe, opisana je razliku između različitih modela baze podataka

Literatura

- [1] Goujun Lu, Multimedia Database management systems, Artech House, 1999.
- [2] Branislav Lazarević, Zoran Marjanović, Nenad Aničić, Slađan Babarogić, Baze podataka, FON, Beograd, 2004
- [3] Michael Abbey, Michael Corey, Ian Abramson, Osnove Oracle 8i, Kompjuter biblioteka, Čačak, 2001.
- [4] Paul Dorsey, Peter Koletzke, Oracle JDeveloper 3, Kompjuter biblioteka, Čačak, 2002.
- [5] Craig Mullins, Administracija baza podataka, Kompjuter biblioteka, Čačak, 2003.
- [6] Ralph Stair, Principles of Information Systems, Boyd & Fraser, 1992.
- [7] Federal Information Processing Standards Publication 184, INTEGRATION DEFINITION FOR INFORMATION MODELING (IDEF1X), National Institute of Standards and Technology
- [8] Grady Booch, James Rumbaugh, Ivar Jacobson, UML vodič za korisnike, CET, Beograd, 2000
- [9] Erich Naiburg, Robert Maksimchuk, UML za projektovanje baza podataka, CET, Beograd, 2001
- [10] Ivana Stanojević, Dušan Surla, Uvod u objedinjeni jezik modelovanja, Grupa za informacione tehnologije, Novi Sad, 1999.



IT210 - SISTEMI IT

WEB SISTEMI I TEHNOLOGIJE

Lekcija 04

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 04

WEB SISTEMI I TEHNOLOGIJE

- ✓ VEB SISTEMI I TEHNOLOGIJE
- ✓ Poglavlje 1: HTTP protokol
- ✓ Poglavlje 2: HTML
- ✓ Poglavlje 3: XHTML
- ✓ Poglavlje 4: XML
- ✓ Poglavlje 5: VEB SERVISI
- ✓ Poglavlje 6: Pokazna vežba: HTML5
- ✓ Poglavlje 7: Pokazna vežba: DHTML
- ✓ Poglavlje 8: Zadaci za samostalni rad: HTML5 i CSS3
- ✓ Poglavlje 9: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da objasni metod rada HTTP protokola i da da pregled veb tehnologija i veb standarda

U ovoj lekciji biće dat pregled sledećih veb tehnologija:

- HTTP protokol
- HTML/XHTML
- XML
- Veb servisi

Fokus vežbi i primera u vežbama je dat na korišćenju event-driven tehnika za manipulaciju podacima u bazi.

▼ Poglavlje 1

HTTP protokol

ŠTA JE HTTP PROTOKOL?

HTTP je protokol za prenos podataka koji definiše poruke između klijenta i servera na bazi zahteva i odgovora

Protokol za prenos podataka koji se koristi širom Weba zove se [HTTP \(HyperText Transfer Protocol\)](#). Njime se definišu poruke koje se šalju između klijenta i servera, odnosno zahtevi i odgovori. Svaka interakcija sadrži jedan tekstualni (ASCII) zahtev i odgovor tipa MIME prema standardu RFC 822. Svi klijenti i svi serveri moraju da poštuju ovaj protokol.

Kada čitač želi da prikaže neku stranicu koja se nalazi na serveru, on šalje tom serveru zahtev za tom datotekom. Kao što smo prethodno opisali čitač uspostavlja TCP konekciju sa serverom na priključak 80, mada s formalnog stanovišta taj postupak nije obavezan. Razlog zašto se TCP koristi je taj kako bi se briga o poslatim porukama, njihovim potvrdama i eventualnom ponovom stanju prebacila na sam TCP i rasteretila čitač i server.

U verziji 1.0 protokola HTTP, pošto se veza uspostavi, šalje se samo jedan zahtev i dobija samo jedan odgovor. Tada se TCP veza raskida. Kada su se Web strane sastojale isključivo od HTML teksta, takav postupak je sasvim odgovarao. Međutim, Web strane danas sadrže mnogo vše od običnog teksta, zato verzija 1.1 protokola HTTP podržava trajne veze (engl. [persistent connections](#)).

Sada HTTP protokol omogućava da nakon uspostavljanja TCP veze i jednokratne razmene zahteva i odgovora, mogu da se šalju dodatni zahtevi i da se primaju odgovori na njih. Ukidanjem uspostavljanja i raskidanja TCP konekcije za svaki zahtev, smanjen je broj nepotrebnih operacija po zahtevu. Zahtevi se mogu slati i serijski, tj. može se poslati zahtev 2 pre nego što stigne odgovor na zahtev 1.

HTTP protokol je napravljen prvenstveno kako bi se koristio za rad na Webu, međutim definisane su i funkcionalnosti koje mogu da podržavaju i za buduće objektno orijentisane aplikacije. [Zbog toga HTTP ima operacije koje se odnose na slanje zahteva i odgovora, ali i druge operacije koje se nazivaju metode.](#)

OSNOVNE DEFINICIJE

Osnovni pojmovi koji su povezani sa razumevanjem HTTP protokola i rada web-a su resurs, veza, poruka, klijent, server, mrežni prolaz, korisnički agent i tunel

Da bi se pravilno razumeo HTTP protokol, potrebno je definisati neke osnovne pojmove [1]:

Resurs (engl. *resource*) je mrežni objekt koji se sastoji od podataka ili servis koji može biti identifikovan URI-jem. Resurs je najčešće datoteka, ali resurs može takođe biti dinamički generisan rezultat upita, izlaz iz nekog CGI skripta ili nekog drugog skripta.

Veza (engl. *connection*) je virtualno kolo transportnog sloja koje se uspostavlja između dva programa u cilju komunikacije.

Poruka (engl. *message*) je osnovna jedinica u HTTP komunikaciji koja se sastoji od okteta i koja se prenosi preko ostvarene veze.

Klijent (engl. *client*) je program koji uspostavlja vezu da bi poslao neki zahtev.

Server je aplikacioni program koji prihvata konekciju da bi poslao odgovor na traženi zahtev. Jeden program može biti i klijent i server zavisno od uloge koju trenutno igra. Severi mogu da deluju kao izvorni server, proksi, mrežni prolaz ili tunel, menjajući ponašanje saglasno zahtevu.

Izvorni server (engl. *origin server*) je server na kome se dati resurs nalazi ili će biti kreiran.

Proksi (engl. *proxy*) je posredni program koji deluje kao server i klijent u svrhu pravljenja zahteva u ime drugih klijenata. Zahtevi se opslužuju internu ili se prosleđuju drugim serverima.

Mrežni prolaz (engl. *gateway*) je server koji deluje kao posrednik nekim drugim serverima. Za razliku od proksija, mrežni prolaz prima zahtev kao da je on izvorni server; klijent koji šalje zahtev ne mora da bude svestan da komunicira sa mrežnim prolazom.

Korisnički agent (engl. *user agent*) je klijent koji inicira zahtev. Ovo je najčešće čitač veb strana, ili neki robot program za prevlačenje veb strana.

Tunel (engl. *tunnel*) je posrednički program koji deluje kao slepi prenosnik između dva čvora na mreži. Kada se jednom aktivira, on se ne posmatra kao učesnik u HTTP komunikaciji, mada je možda bio inicijalizovan HTTP zahtevom. Tunel prestaje da postoji kada obe povezane strane zatvore vezu.

▼ 1.1 NAČIN FUNKCIJONISANJA HTTP PROTOKOLA

STRUKTURA ZAHTEVA I ODGOVORA HTTP PROTOKOLA

Klijent šalje zahtev sa određenom strukturom poruke definisane standardom, dok klijent obrađuje taj zahtev i šalje odgovor koji je takođe u određenom formatu

HTTP protokol je protokol tipa zahtev/odgovor. Klijent šalje serveru **zahtev** koji se sastoji od:

- metode zahteva,
- URI-ja,
- verzije protokola,
- poruke koja sadrži modifikatore zahteva,
- informacija o klijentu
- opcionog tela sa sadržajem

Odgovor servera se sastoji od:

- statusne linije koja uključuje verziju protokola poruke i poruku o uspešnosti odgovora ili kod greške u slučaju neuspešnog odgovora
- informacije o serveru
- meta informacije o entitetu koji se šalje
- tela entiteta

JEDNOSTAVNA HTTP VEZA

Jednostavna HTTP veza se uspostavlja između klijenta, odnosno korisničkog agenta (na primer, veb čitač) koji upućuje serveru zahtev koji treba da se primeni na nekom resursu

Većina HTTP komunikacija se inicijalizuje od strane korisničkog agenta (KA) i sastoji se od zahteva koji treba da se primeni na resurse nekog izvorišnog servera (IS). U najjednostavnijem slučaju, ovo se postiže jednom vezom (V) između korisničkog agenta i izvorišnog servera.



Slika 1.1.1 Jednostavna veza između korisničkog agenta (KA) i izvorišne strane (IS) [Izvor: Autor]

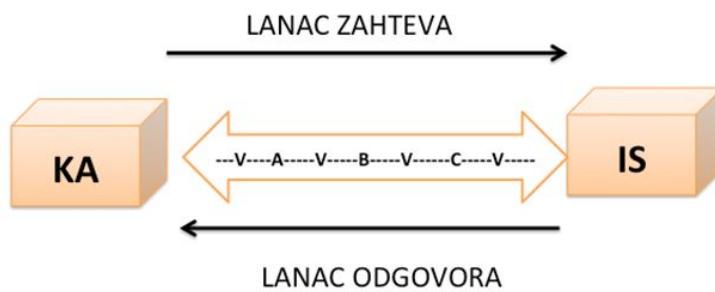
HTTP VEZA SA TRI POSREDNIKA

Kada između klijenta i servera postoje posrednici, zahtev će proći kroz odvojene veze (eng. link)

Složeniji slučaj se javlja kada je jedan ili više posrednika uključeno u lanac zahtev/odgovor. Postoje tri česta oblika posrednika: proksi, mrežni prolaz i tunel. Proksi je prosleđujući agent koji prima zahtev za URI-jem u njegovoj apsolutnom obliku, prepisuje celu ili samo deo poruke i prosleđuje ga ka serveru koji je identifikovan URI-jem. Mrežni prolaz je prijemni agent, koji se ponaša kao sloj iznad nekog drugog servera, i ako je potrebno, prevodi zahtev na

protokol servera koji je ispod njega. Tunel deluje kao veza između dva čvora koja ne menja poruku. Koristi se kada je potrebno komunikaciju izvršiti preko posrednika (kakav je na primer zaštitni zid – firewall) iako posrednik ne razume sadržaj poruke. Na narednoj slici je prikazan slučaj kada između korisničkog agenta i izvođenog servera postoje tri posrednika. Zahtev i odziv će proći kroz četiri odvojene veze. Ovde se namerno pravi razlika zbog toga što neke HTTP komunikacione opcije mogu da se primene samo za povezivanje sa najbližim ne-tunel susedom, samo sa krajnjom tačkom lanca, ili sa svima u lancu

Učesnici u ovoj komunikaciji mogu istovremeno da budu angažovani u višestrukim komunikacijama. Na primer B može da prima zahteve i od drugih klijenata osim od A i/ili šalje zahteve i drugim serverima, a ne samo serveru C.

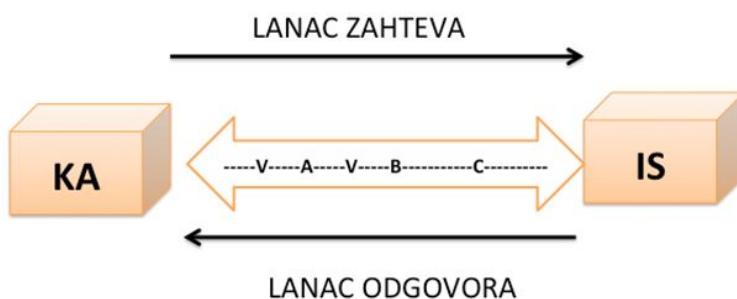


Slika 1.1.2 Veza sa tri posrednika [Izvor: Autor]

VEZA SA NALAŽENJEM ODGOVORA U POSREDNIKU

Kada neki član u lancu komunikacije ima keširani odgovor na zahtev korisničkog agenta, onda će se lanac zahtev/odgovor skratiti.

Svaki član komunikacije koji ne deluje kao tunel može da ima interni keš za manipulaciju sa zahtevima. Ukoliko neki član u lancu komunikacije ima keširani odgovor na zahtev korisničkog agenta, onda će se lanac zahtev/odgovor skratiti. Na sledećoj šemi je prikazan primer u kome B ima keširanu kopiju ranijeg odziva izvođenog servera dobijenu preko C. U tom slučaju nije potrebno uspostavljati vezu između B i C i dalje prema IS, čime se lanac zahtev/odgovor skratio a proces ubrzao.



Slika 1.1.3 Veza sa nalaženjem odgovora u posredniku B [Izvor: Autor]

▼ 1.2 PARAMETRI PROTOKOLA

POLJA ZAGLAVLJA U PORUCI HTTP PROTOKOLA

Definicija parametara HTTP protokola data je u dokumentu RFC2616

HTTP protokol koristi tokom komunikacije sledeće parametre unutar pojedinih polja zaglavljva [2]:

- HTTP verziju
- Jedinstveni identifikator resursa - Uniform Resource Identifiers (URI)
- Datum i vreme - Date/Time formats
- Metod kodiranja - Character set
- Kodiranje sadržaja - Content coding
- Transfer coding
- Tip medija - Media type
- Token proizvoda - Product Tokens
- Quality Values
- Language Tags
- Entity Tags
- Range Units

U daljem tekstu će biti opisani neki od ovih parametara.

VERZIJE PROTOKOLA

Obzirom da verzija protokola definiše mogućnosti protokola pošiljaoca, proksi i mrežni prolaz ne smeju da šalju poruku sa verzijom koja je veća od njihove trenutne verzije

Verzija protokola služi da omogući pošiljaocu da pokaže format poruke i njegovu sposobnost da razume buduću HTTP komunikaciju. Opšti oblik ovog parametra je:

HTTP-Version = "HTTP" "/" 1*DIGIT "." 1*DIGIT

Ako se koristi verzija 1.1 HTTP protokola, onda ovaj parametar ima oblik: HTTP/1.1

Aplikacije proksija i mrežnih prolaza treba da vode računa kada prosleđuju poruku sa verzijom protokola koja se razlikuje od aplikacije. Obzirom da verzija protokola definiše mogućnosti protokola pošiljaoca, proksi i mrežni prolaz ne smeju da šalju poruku sa verzijom koja je veća od njihove trenutne verzije. Ako je zahtev stigao sa višom verzijom od one koje koristi proksi ili mrežni prolaz, oni mogu da degradiraju verziju, odgovore sa greškom, ili pređu u tunelski režim rada.

DATUM I VREME

Klijenti i serveri koji komuniciraju HTTP protokolom mogu da prihvate različite formate za prikaz vremena

HTTP aplikacije su istorijski gledano dozvoljavale tri različita formata za zapis datuma i vremena. To su:

Sun, 06 Nov 1994 08:49:37 GMT; [RFC 1123](#)

Sunday, 06-Nov-94 08:49:37 GMT; [RFC 850](#), zastareo standard koji bez godine sa četiri cifre

Sun Nov 6 08:49:37 1994; [ANSI C](#) format

HTTP/1.1 klijenti i serveri moraju da prihvate sva tri formata zbog kompatibilnosti sa HTTP/1.0. Međutim, u svojim zaglavljima moraju da koriste samo format definisan RFC1123 standardom.

METOD KODIRANJA I KODIRANJE SADRŽAJA

„character set“ definiše metod kojim se pomoću jedne ili više tabele niz okteta (bajtova) prevodi u niz karaktera, „content coding“ definiše metod kojim će se transformisati sadržaj entiteta

Metod kodiranja

HTTP parametar [„character set“](#) definiše metod kojim se pomoću jedne ili više tabele niz okteta (bajtova) prevodi u niz karaktera. [Set karaktera](#) je definisan tokenom. Raspoloživi set tokena je definisan registrom koji održava organizacija IANA ([Internet Assigned Numbers Authority](#)).

Kodiranje sadržaja

Parametar [kodiranje sadržaja](#) (engl. [content coding](#)) definiše metod kojim će se transformisati sadržaj entiteta. Ovaj parametar se koristi da omogući kompresiju dokumenata ili neku drugu transformaciju bez gubljenja informacija i tipa medija. Parametar može da uzme vrednosti koje su definisane u registru IANA-e. Na primer ako je vrednost parametra kodiranje sadržaja „gzip“ za dekompresiju sadržaja treba koristiti GNU zip program.

TIP MEDIJA

Tip medija se koristi u zaglavljima da bi ukazao na tip podataka koji se prenosi

Parametar [tip medija](#) (engl. [media type](#)) se koristi u zaglavljima **Content-type** i **Accept** da bi ukazao na tip podataka koji se prenosi. Parametar se sastoji od tipa medija, podtipa i opcionih

parametara. Dozvoljene vrednosti parametara su definisane u registru koji održava IANA. Na primer, vrednost parametra tip medija pri prenosu HTML datoteke je

text/html

gde je *tekst* tip, a *html* podtip medija.

Za detaljan opis svih karakteristika preporučuje se čitanje RFC616 dokumenta [3].

▼ 1.3 HTTP PORUKE

STRUKTURA HTTP PORUKE

HTTP poruka se sastoji iz startne linije, zaglavija, prazne linije koja sadrži karakter CRLF i tela poruke

HTTP poruke mogu biti tipa zahtev (engl. **request**) i odgovor (engl. **response**). Poruke se sastoje od [4]:

- startne linije
- jednog ili više zaglavija
- prazne linije koja sadrži samo karakter CRLF i koja označava da nema više linija sa zaglavljima
- opcionog tela poruke



Slika 1.2.1 HTTP poruka [Izvor: Autor]

generic-message = start-line *(message-header CRLF) CRLF [message-body]
start-line = Request-Line | Status-Line

Startna linija (engl. **start line**) se razlikuje za zahteve i odgovore.

Zaglavija mogu biti: opšta, zaglavija zahteva, zaglavija odgovora i zaglavija entiteta.

Telo poruke je optionalno jer u nekim slučajevima nema potrebe za njim. Telo poruke se razlikuje od tela entiteta samo kada se vrši šifriranje sadržaja entiteta.

message-body = entity-body

|<entity-body encoded as per Transfer-Encoding>

Opšta zaglavlja se koriste za poruke zahteva i poruke odgovora. Ova zaglavlja se ne odnose na entitet koji se prenosi. Postoje samo nekoliko tipova opštih zaglavlja, a to su:

Cache-Control, Connection, Date,

Pragma (specificira implementaciju za sve u zahtev/odgovor lancu),

Transfer-Encoding,

Upgrade (jednostavan mehanizam za prelazak sa HTTP 1.1 u neki drugi protokol),

Via (ako odgovor ide preko proksija, proksi ne menja odgovor servera nego se upisuju interni protokol i konekcije)

✓ 1.4 HTTP ZAHTEV

METODE HTTP PROTOKOLA

Token metode pokazuje koja će se metoda primeniti na resursu identifikovanom Request-URI-jem

Startna linija poruke sa zahtevom od klijenta ka serveru se naziva linija zahteva. Ona se sastoji od tri elementa:

- Tokena metode koja će se primeniti na resurs (**methods**)
- Identifikator resursa (**Request-URI**)
- Verzije protokola koji se koristi (**HTTP-Version**)

koji su međusobno odvojeni karakterom **SP** (space). Na kraju linije zahteva je karakter **CRLF**. Opšti oblik linije zahteva je:

Request-Line = Method SP Request-URI SP HTTP-Version CRLF

Token metode pokazuje koja će se metoda primeniti na resursu identifikovanom Request-URI-jem. Moguće metode zahteva su:

- OPTIONS
- GET
- HEAD
- POST
- PUT
- DELETE
- TRACE

OPISI HTTP METODA

HTTP metode su GET, HEAD, PUT, POST, DELETE, TRACE, OPTIONS

Metod **OPTIONS** predstavlja zahtev za informacijama o raspoloživim komunikacionim opcijama u lancu zahtev-odgovor identifikovanom pomoću URI-ja. Na taj način klijent opcije ili zahteve vezane za resurs, kao i mogućnosti servera bez angažovanja samog servera.

Metod **GET** omogućuje preuzimanje informacija identifikovanih URI-jem. To može da bude neka datoteka ili podaci koje je generisao neki proces.

Metod **HEAD** je isti kao GET s tom razlikom što server ne sme u odgovoru da vrati telo poruke. Ovaj metod se koristi kada je potrebno dobiti meta informacije o entitetu koji je zahtevan, a da se pri tom ne prenosi i samo telo entiteta. Na taj način je moguće lako testirati validnost, dostupnost i poslednju izmenu hiperlinkova.

Metod **DELETE** zahteva da odredišni server obriše resurs identifikovan URI-jem.

Metod **TRACE** se koristi za vraćanje eha poslatog zahteva odredišnom serveru. Drugim rečima, klijent može da vidi kakvu je poruku primio odredišni server. Na taj način klijent može da sazna koji su međuserveri menjali originalni zahtev, što je vrlo korisno za dijagnostiku.

Metod **POST** se koristi da se odredišnom serveru prosledi entitet koji se nalazi u zahtevu. POST metod omogućuje sledeće funkcije:

- Slanje napomena o postojećim resursima,
- Slanje poruka njuzgrupama, mejling listama, elektronskim oglasnim tablama
- Slanje bloka podataka, kao što su podaci iz popunjene forme, procesu koji obrađuje podatke,
- Upisivanje podataka u neku bazu podataka korišćenjem operacije append.

Metod **PUT** zahteva da se entitet primi i sačuva pod identifikacijom koja je definisana poljem Request-URI. Ako se URI odnosi na već postojeći resurs, entitet treba smatrati modifikovanom verzijom onog koji postoji na odredišnom serveru. Ako URI ne ukazuje na postojeći resurs, odredišni server može da kreira resurs sa tim URI-jem.

NAPOMENA

Metoda POST identificuje resurs kome će se pridružiti neki entitet, a PUT identificuje gde će se pridruženi entitet čuvati, odnosno zamenjuje staru verziju resursa ukoliko već postoji

Osnovna razlika između **POST** i **PUT** zahteva je u različitom značenju koji ima URI. URI u POST zahtevu identificuje resurs koji će procesirati pridruženi entitet. Taj resurs može biti proces koji prihvata podatke, mrežni prolaz ka nekom drugom protokolu ili zaseban entitet koji prihvata napomene o postojećim resursima. URI u PUT zahtevu identificuje gde će se pridruženi entitet čuvati.

Svi serveri moraju da podržavaju metode GET i HEAD. Ostale metode su opcione. Lista metoda koju neki server podržava je data u polju Public response-header.

Zaglavla zahteva omogućuju klijentu da serveru prosledi dodatne informacije o zahtevu i o samom klijentu. Ova polja deluju kao modifikatori.

Moguća zaglavila poruke zahteva su:

- **Accept**
- **Accept-Charset**
- **Accept-Encoding**
- **Accept-Language**
- **AuthorizationFrom**
- **Host**
- **If-Modified-Since**
- **If-Match**
- **If-None-Match**
- **If-Range**
- **If-Unmodified-Since**
- **Max-Forwards**
- **Proxy-Authorization**
- **Range**
- **Referer**
- **User-Agent**

STRUKTURA HTTP ZAHTEVA - VIDEO

Parts of an HTTP Request

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 1.5 HTTP ODGOVOR

STRUKTURA HTTP ODGOVORA

Startna linija HTTP odgovora se sastoji od HTTP verzije, statusnog koda i opisa statusnog koda

Nakon prijema i interpretiranja poruke zahteva, server odgovara porukom odgovora (engl. **response message**). Startna linija poruke odgovora se sastoji od tri elementa:

- **HTTP verzije,**
- **Statusnog koda (Status-Code)**
- **Opis statusnog koda.**

koji su međusobno odvojeni karakterom SP (**space**). Na kraju linije zahteva je karakter CRLF. Opšti oblik statusne linije je:

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Statusni kod je trocifreni broj koji opisuje status pokušaja da se razume i odgovori na zahtev. Svakom statusnom kodu odgovara jedan kratak tekstualni opis statusa. Prva cifra statusnog koda definiše tip odziva. Postoji pet vrednosti prve cifre:

1XX - Informativno - Zahtev primljen, proces se nastavlja,

2XX - Uspešno - Akcija je uspešno primljena, shvaćena i prihvaćena,

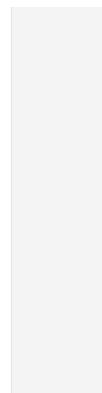
3XX - Redirekcija - Dodatne akcije se moraju izvršiti da bi se kompletirao zahtev,

4XX - Greška klijenta - Zahtev ima lošu sintaksu ili se ne može ispuniti,

5XX - Greška servera - Server ne može da ispuni validan zahtev klijenta.

U narednoj tabeli su dati statusni kodovi i njihov opis.

Statusni kod	Opis
100	Continue
101	Switching Protocols
200	OK
201	Created
202	Accepted
203	Non-Authoritative Information
204	No Content
205	Reset Content
206	Partial Content
300	Multiple Choices
301	Moved Permanently
302	Moved Temporarily
303	See Other
304	Not Modified
305	Use Proxy
306	Redirect
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable
407	Proxy Authentication Required
408	Request Time-out
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Request Entity Too Large
414	Request-URI Too Large
415	Unsupported Media Type
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Time-out
505	HTTP Version not supported



Slika 1.3.1 Tabela-1 Statusni kodovi HTTP odgovora [Izvor: Autor]

Zaglavla odgovora dozvoljavaju serveru da prosledi klijentu dodatne informacije o serveru i daljem pristupu resursu identifikovanog URI-jem. Moguća zaglavla odgovora su:

- **Age**
- **Location**
- **Proxy-Authenticate**
- **Public**
- **Retry-After**
- **Server**
- **Vary**
- **Warning**
- **WWW-Authenticate**

STRUKTURA HTTP ODGOVORA - VIDEO

HTTP Response

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 1.6 HTTP ENTITET

ZAGLAVLJA ENTITETA

Zaglavljia entiteta su meta informacije o telu entiteta ili, ako telo entiteta nije prisutno, o resursima identifikovanih zahtevom

Poruke zahteva i odgovora mogu da prenose neki entitet. Entitet se sastoji od zaglavlja entiteta i tela entiteta, mada neki odgovori mogu da sadrže samo zaglavljia.

Zaglavljia entiteta su meta informacije o telu entiteta ili, ako telo entiteta nije prisutno, o resursima identifikovanih zahtevom. Moguća zaglavljia entiteta su:

- **Allow**
- **Content-Base**
- **Content-Encoding**
- **Content-Language**
- **Content-Length**
- **Content-Location**
- **Content-MD5**
- **Content-Range**
- **Content-Type**
- **ETag**
- **Expires**
- **Last-Modified**

Telo entiteta je u formatu i kodu definisanom zaglavljima entiteta. Ukoliko postoji telo entiteta unutar neke poruke potrebno je da entitet sadrži zaglavljje **Content-Type**.

▼ Poglavlje 2

HTML

MOGUĆNOSTI HTML I NJEGOVA INTERPRETACIJA

Za interpretaciju HTML dokumenta se koriste različiti korisnički agenti. Korisnički agent može biti vizualni veb čitač

Hipertekstualni markup jezik (HTML - Hyper Text Markup Language) je jezik koji služi za publikovanje dokumenata na World Wide Web-u. Dokumenti, takozvane veb strane, mogu da se sastoje od teksta, tabele, lista, slika i elemenata za unos podataka. Svaki dokument može da sadrži i meta informacije koje ukazuju na to što sadrži dokument. Unutar dokumenta se uobičajeno nalaze hiperlinkovi koji pomoću URI-ja daju vezu ka drugim veb stranicama. Dokument može da sadrži i ugnježdene skriptove, aplete i/ili objekte.

Za interpretaciju HTML dokumenta se koriste različiti **korisnički agenti**. Korisnički agent može biti vizualni veb čitač koji može da radi u tekstualnom ili grafičkom režimu, nevizuelni veb čitač, koji može biti audio ili Brajov, roboti za pretraživanje, proksi ili neki drugi uređaji.

Koristeći ove mogućnosti HTML-a autori veb strana su u mogućnosti da:

- Publikuju veb dokumente koji imaju naslove, tekst, tabele, liste, slike i druge ugnježdene objekte kao što su matematički, zvučni ili video objekti,
- Omoguće čitaocima veb strana da pristupe drugim informacijama koje su referencirane hiperlinkovima,
- Uključe u veb strane forme, komandnu dugmad, izborne liste, opcione grupe i druge elemente koje služe za unos podataka i komandi.

VERZIJE HTML-A

U cilju pojednostavljenja opisa veb strana, HTML 4.0 uvodi veću razliku između strukture dokumenta i prezentacije

Autori veb strana koriste HTML jezik da, paralelno sa sadržajem, predstave strukturne, prezentacione i semantičke informacije potrebne veb čitaču da pravilno i efikasno interpretira dokument.

HTML verzija 4 je objavljena 1998. godine, a revidirana verzija HTML 4.01 je iz 1999. godine. Ova verzija HTML-a omogućuje kreiranje dokumenta na bilo kom jeziku i korišćenje različitih pisama. Time je omogućeno efektivno indeksiranje dokumenata za mašine za pretraživanje, kvalitetnija tipografija i bolja konverzija iz teksta u govor. Posebna pažnja

pri razvoju HTML-a je posvećena poboljšanju dostupnosti sadržaja osobama sa fizičkim ograničenjima.

U cilju pojednostavljenja opisa veb strana, HTML 4.0 uvodi veću razliku između strukture dokumenta i prezentacije, ohrabrujući tako upotrebu stilova (**style sheets**) umesto HTML prezentacionih elemenata i atributa. Informacije o stilu mogu biti specificirane unutar HTML dokumenta ili u eksternim datotekama, i mogu se odnositi na individualne elemente ili na grupe elemenata. Mechanizam za definisanje stilova ne zavisi od jezika kojim se opisuje stil. U nameri da se nametne korišćenje stilova WWW konzorcijum planira da u narednim verzijama ukine neke prezentacione elemente i attribute. Takvi elementi (tagovi) su u verziji 4.01 označeni za izbegavanje (**deprecated**).

Najnovija verzija HTML-a je HTML 5.0. HTML 5.0 dopunjuje HTML sa sintaktičkim svojstvima koje uključuju elemente `<video>`, `<audio>`, `<header>` i `<canvas>`. Ove osobine su dodate kako bi omogućile lakše upravljanje multimedijalnim i grafičkim sadržajima bez dodatne potrebe za plugin-ovima i API. HTML5 takođe omogućava obogaćenje i semantičkog zadržaja dokumenta sa atributima kao što su `<section>`, `<article>`, `<header>` i `<nav>`.

Iz istog razloga su dodati i neki novi atributi, dok su određeni elementi i atributi odbačeni u odnosu na prethodnu verziju.

▼ 2.1 KONSTRUKCIJE HTML JEZIKA

STRUKTURA I KONSTRUKCIJA HTML JEZIKA

HTML je markap jezik koristi konstrukcije: elemente, atribute, karakter reference i komentare

Hipertekstualni markup jezik HTML je zasnovan internacionalnom standardu ISO 8879 – **Standard Generalized Markup Language** ili skraćeno SGML. Ovaj standard definiše format u tekstualnim dokumentima. Međutim SGML se koristi i za definisanje drugih specijalizovanih jezika za predstavljanje dokumenata. SGML dokument koristi posebnu datoteku koja se zove **Document Type Definition** (DTD) koja definiše tagove koji se koriste za opis formatiranja teksta, atribute i entitete nekog markup jezika, kao i način na koji se oni zajedno koriste.

Pošto SGML opisuje svoje sopstveno formatiranje naziva se meta-jezik. SGML je vrlo složen i obiman jezik koji između ostalog uključuje i hipertekst linkove. HTML je jedan od jezika koji je razvijen na bazi SGML. Pomoću SGML-a je unapred definisan set tagova i njihovih atributa koji se koriste u HTML-u.

HTML datoteka, koja opisuje jednu stranicu je sastavljena od teksta i tagova. Tag je osnova svake HTML strane, i kazuje veb čitaču kako da prikaže veb stranu. HTML datoteka mora da ima ekstenziju tipa **.html** , **.htm**, ali se može kreirati u bilo kom tekstu editoru ili pomoću nekog alata za kreiranje HTML strana.

HTML jezik koristi sledeće konstrukcije:

- Elemente

- Atribute
- Karakter reference
- Komentare.

✓ 2.2 POZICIONIRANJE ELEMENATA

POZICIONIRANJE

Postoji nekoliko različitih tehnika za pozicioniranje elemenata

Kada je u pitanju raspoređivanje i pozicioniranje sadržaja na stranici, postoji nekoliko različitih tehnika koje se mogu koristiti. Odabir tehnike u velikoj meri zavisi od sadržaja i ciljeva stranice, jer neke tehnike mogu biti bolje od drugih.

Ove tehnike omogućavaju ručno pozicioniranje elemenata korišćenjem specifičnih koordinata. Na primer, moguće je odrediti da element bude 20 piksela desno od roditelj elementa ili 55 piksela na dole od ivice ekrana.

Na primer, sposobnost elemenata da "plutaju" (engl. **float**) jedan pored drugog pruža lep i čist izgled koji odgovara različitim elementima na stranici. Međutim, kada je potrebna stroža kontrola, elementi se mogu postaviti koristeći druge tehnike, uključujući relativno ili apsolutno pozicioniranje.

SVOJSTVO FLOAT

Kod ovog načina pozicioniranja, položaj elementa zavisi od ostalih elemenata koji se nalaze oko njega.

U suštini, svojstvo **float** omogućava nam da uzmemo element, uklonimo ga iz normalnog toka stranice i pozicionira ga levo ili desno od svog roditeljskog elementa. Svi ostali elementi na stranici će se onda kretati oko plivajućeg elementa.

Kada se **float** svojstvo koristi istovremeno na više elemenata, on omogućava mogućnost kreiranja rasporeda plutajućih elemenata direktno pored ili suprotno jedni drugima.

Kod ovog načina pozicioniranja, položaj elementa zavisi od ostalih elemenata koji se nalaze oko njega. U sledećem primeru napravićemo četiri elementa bez određenog pozicioniranja. Rezultat je prikazan na slici 1



Slika 2.1.1 Elementi bez pozicioniranja [Izvor: Autor]

Može se uočiti da su svi elementi jedan ispod drugog. Ako želimo da elemente dva i tri postavimo jedan pored drugog, treba im postaviti float svojstvo. Takođe su im promenjene veličine, tako da će element sa idjem dva biti veći. Rezultat je prikazan na slici 2.

```
#dva {  
    float: left;  
    width: 63%;  
}  
  
#tri {  
    float: right;  
    width: 30%;  
}
```

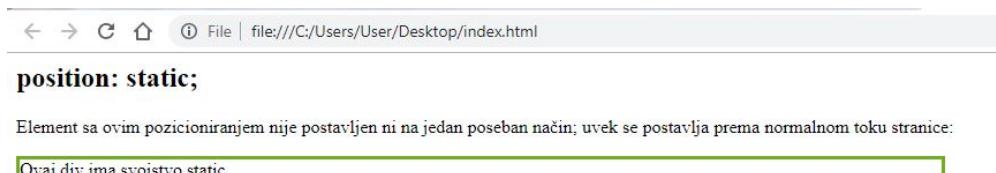


Slika 2.1.2 Primena float svojstva [Izvor: Autor]

STATIČKO POZICIONIRANJE

Raspoređuje elemente redosledom kojim se pojavljuju u HTML-u

Static svojstvo je default vrednost i raspoređuje elemente redosledom kojim se pojavljuju u HTML-u. Na ovo svojstvo ne utiču atributi kao što su **top**, **bottom**, **left** ili **right**.

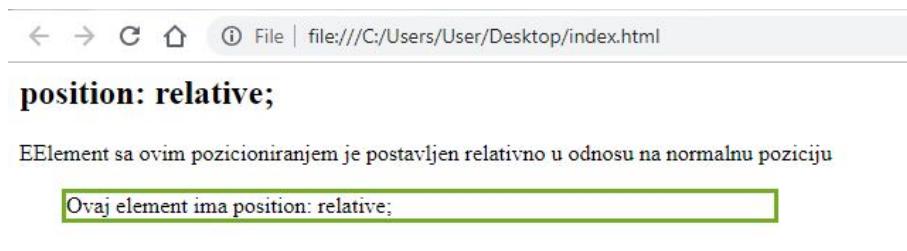


Slika 2.1.3 Svojstvo static [Izvor: Autor]

RELATIVNO POZICIONIRANJE

Element se pomera u odnosu na svoju normalnu poziciju

Kada postavite položaj u odnosu na neki element, bez dodavanja drugih atributa za pozicioniranje (gornji, donji, desni, levo), ništa se neće dogoditi. Kada dodate dodatnu poziciju, na primer left:20px element će se pomeriti 20px desno od svoje normalne pozicije. Ovde možete videti da je ovaj element relativan samom sebi. Kada se element kreće, nijedan drugi element na stranici neće biti pomeren.

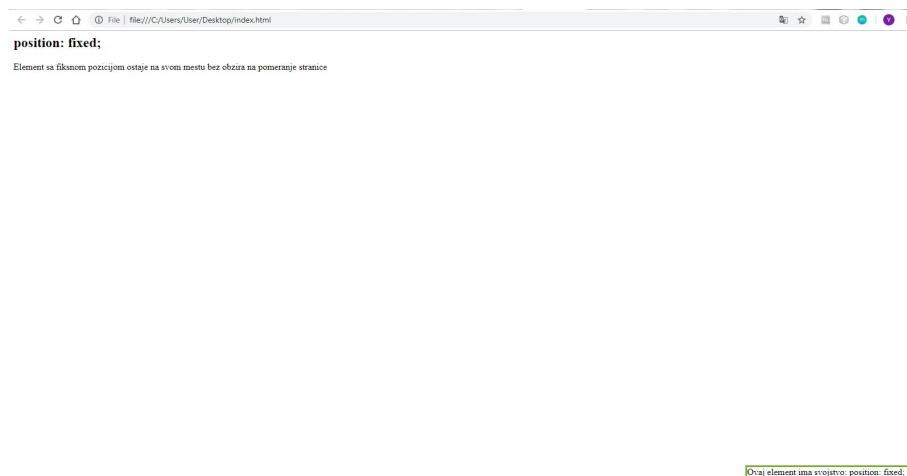


Slika 2.1.4 Relativno pozicioniranje [Izvor: Autor]

FIXED POZICIONIRANJE

Ovim svojstvom elementi se fiksiraju na stranici

Ovim svojstvom elementi se fiksiraju na stranici. Ovo znači da čak iako se stranica pomera, fiksirani element neće menjati svoju poziciju. Atributi top, bottom, left, right se ne odnose na element sa fixed pozicijom. Na slici 5, element je fiksiran u donjem desnom uglu.



Slika 2.1.5 Fixed pozicioniranje [Izvor: Autor]

ABSOLUTE POZICIONIRANJE

Najčešći način za dodavanje CSS je pomoću odvojenog CSS fajla

Element sa pozicijom absolute je pozicioniran u odnosu na najbliži pozicionirani roditeljski element. Ukoliko nema roditeljskih elemenata, onda se pozicionira u odnosu na telo dokumenta



Slika 2.1.6 Absolute pozicioniranje [Izvor: Autor]

Z-INDEX

Z-index omogućava kontrolu dubine elemenata na stranici

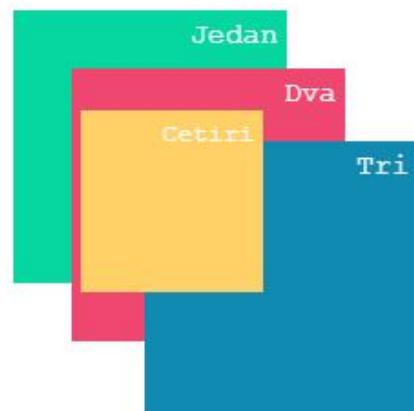
Veb stranice se uglavnom smatraju dvodimenzionalnim, što se moglo primeriti radom sa do sada prikazanim HTML elementima. Elementi se na stranice postavljaju jedan ispod drugog ili jedan pored drugog po x i y osi. Kako bi se promenio raspored elemenata može se uvesti i z osa kao i svojstvo z-index.

Z-index omogućava kontrolu dubine elemenata na stranici. Ovo daje 3D efekat stranici tako što je elemente moguće postaviti ispred ili iza drugih elemenata što stranici daje dubinu.

Ono što određuje gde će biti postavljen element je vrednost z-indexa. Vrednosti mogu biti negativne i pozitivne. Negativne vrednosti postavljaju element dalje na stranicu, odnosno u pozadinu dok pozitivne vrednosti približavaju element odnosno postavljaju ga ispred ostalih elemenata.

Na slici 7 prikazana su četiri kvadrata različitih boja. Kvadrat plave boje pa prvoj slici ima sledeću vrednost z-index svojstva: z-index: 1; dok narandžasti ima z-index: 2; i može se videti da je narandžasti postavljen preko plavog kvadrata.

Na slici 8 prikazano je šta se desi kada se ove dve vrednosti zamene odnosno narandžasti kvadrat ima vrednost z-indexa 1 a plavi ima vrednost 2. Sada se narandžasti nalazi iza plavog kvadrata.



Slika 2.1.7 Prvi primer [Izvor: Autor]



Slika 2.1.8 Drugi primer [Izvor: Autor]

▼ Poglavlje 3

XHTML

ŠTA JE XHTML I KOJE SU NJEGOVE PREDNOSTI?

XHTML je familija dokumenata i modula koji reprodukuju i proširuju HTML, a baziran je na XML-u i projektovan je da radi u spremi sa XML baziranim korisničkim agentima

XHTML (Extensible HyperText Markup Language) je proširivi hipertekstualni markap jezik. Specifikacija XHTML 1.0 je usvojena januara 2000. godine. XHTML je familija trenutnih i budućih tipova dokumenata i modula koji reprodukuju i proširuju HTML 4. XHTML je baziran na jeziku XML i pre svega je projektovan da radi u spremi sa XML baziranim korisničkim agentima. Praktično, XHTML je dobijen reformulacijom jezika HTML 4 primenom XML 1.0. Njegova namena je da postane jezik za opis sadržaja koji će biti podudaran sa XML-om i koji će moći, ako se prate jednostavne smernice, da radi sa korisničkim agentima namenjenim jeziku HTML 4.

Korišćenje XHTML-a ima sledeće prednosti:

- XHTML dokumenti su usklađeni (konformni) sa XML-om, kao takvi mogu da budu čitani, editovani i validirani standardnim XML alatima.
- XHTML dokumenti mogu biti pisani da rade isto ili bolje nego što su to radili pre sa postojećim
- HTML 4 korisničkim agentima, kao i sa novim XHTML 1.0 korisničkim agentima.
- XHTML dokumenti mogu da koriste aplikacije (na primer skriptove ili aplete) koji se zasnivaju na HTML Document Object Model ili na XML Document Object Model.
- Sa daljim razvojem XHTML familije, dokumenti koji odgovaraju XHTML 1.0 specifikaciji će moći da sarađuju sa drugim XHTML okruženjima.

HTML 4.01 / XHTML 1.0

Pregled osnovnih tagova, formatiranja teksta, blokova, linkova, ulaza, izlaza, lista i slika

U narednim tabelama su opisani elementi HTML 4.01 i XHTML 1.0 jezika. Elementi su grupisani po funkcijama. U koloni DTD je upisana oznaka vrste DTD koja dozvoljava dati element. Upotrebljene oznake su: S - za striktnu formulaciju, T - za prelaznu ili tranzicionu formulaciju i F za tabelarnu.

Osnovni tagovi		
Tag	Namena	DTD
<!DOCTYPE>	Definiše tip dokumenta	STF
<html>	Definiše HTML dokument	STF
<body>	Definiše element tela	STF
<h1> to <h6>	Definiše zaglavje u tekstu veličine 1 do 6	STF
<p>	Definiše paragraf	STF
 	Prekid linije	STF
<hr>	Definiše horizontalnu liniju	STF
<!--...-->	Definiše komentar	STF

Slika 3.1 Osnovni tagovi [Izvor: Autor]

Formatiranje teksta		
Tag	Namena	DTD
	Definiše bold tekst	STF
	Izbegavati. Definiše text font, size, and color	TF
<i>	Definiše italic tekst	STF
	Definiše naglašeni tekst	STF
<big>	Definiše veliki tekst	STF
	Definiše strong tekst	STF
<small>	Definiše mali tekst	STF
<sup>	Definiše uzdignuti (superscripted) tekst	STF
<sub>	Definiše spušteni (subscripted) tekst	STF
<bdo>	Definiše smer teksta	STF
<u>	Izbegavati. Definiše podvučeni tekst	TF

Slika 3.2 Formatiranje teksta [Izvor: Autor]

Output		
Tag	Namena	DTD
<pre>	Definiše ranije formatirani tekst	STF
<code>	Definiše kod	STF
<t>	Definiše teletype tekst	STF
<kbd>	Definiše tekst koji će biti unet sa tastature	STF
<var>	Definiše a varijable	STF
<dfn>	Definiše definiciju za neki termin	STF
<samp>	Definiše primer kompjuterskog koda	STF
<xmp>	Izbegavati. Definiše ranije formatirani tekst	
Blokovi		

Slika 3.3 Output [Izvor: Autor]

Blokovi		
Tag	Namena	DTD
<acronym>	Definiše akronim	STF
<abbr>	Definiše skraćenicu	STF
<address>	Definiše element adresе	STF
<blockquote>	Definiše dugacke navode	STF
<center>	Izbegavati. Definiše centrirani tekst	TF
<q>	Definiše kratke navode	STF
<cite>	Definiše citat	STF
<ins>	Definiše umetnut tekst	STF
	Definiše obrisani tekst	STF
<s>	Izbegavati. Definiše precrtni tekst	TF
<strike>	Izbegavati. Definiše precrtni tekst	TF

Slika 3.4 Blokovi [Izvor: Autor]

Linkovi		
Tag	Namena	DTD
<a>	Definiše link	STF
<link>	Definiše link u zagлавju	STF
Okviri		
Tag	Namena	DTD
<frame>	Definiše okvir (podprozor)	F
<frameset>	set okvira	F
<noframes>	Definiše sekciju bez okvira	TF
<iframe>	Definiše okvir unutar bloka teksta	TF

Slika 3.5 Linkovi [Izvor: Autor]

Ulaz	Namena	DTD
<form>	Definiše formu	STF
<input>	Definiše ulazno polje	STF
<textarea>	Definiše tekst polje	STF
<button>	Definiše push dugme	STF
<select>	Definiše izbornu listu	STF
<optgroup>	Definiše opcionalnu grupu	STF
<option>	Definiše jedan element u izbornoj listi	STF
<label>	Definiše labelu kontrole	STF
<fieldset>	Definiše grupu kontrola i labela	STF
<legend>	Definiše naslov grupe kontrole	STF
<input type="checkbox">	Izbegavati. Definiše jednolinjsko ulazno polje	TF

Slika 3.6 Ulaz [Izvor: Autor]

Lista	Namena	DTD
	Definiše neuređenu listu	STF
	Definiše uređenu listu	STF
	Definiše element listi	STF
<dir>	Izbegavati. Definiše listu direktorijuma	TF
<dl>	Definiše listu definicija	STF
<dt>	Definiše definicioni izraz	STF
<dd>	Definiše opis definicije	STF
<menu>	Izbegavati. Definiše listu menija	TF
Stlike	Namena	DTD
	Definiše sliku	STF
<map>	Definiše image map-u	STF
<area>	Definiše područje unutar image map-e	STF

Slika 3.7 Liste i slike [Izvor: Autor]

HTML 4.01 / XHTML 1.0 - TABELE I STILOVI

Pregled tabela, stilova, meta informacija i ugnježdavanja skripti i objekata

Tabele	Namena	DTD
<table>	Definiše tabelu	STF
<caption>	Definiše naslov tabele	STF
<th>	Definiše zaglavljive kolone	STF
<tr>	Definiše vrstu	STF
<td>	Definiše celiju	STF
<thead>	Definiše zaglavljivo tело tabele	STF
<tbody>	Definiše telo tabele	STF
<tfoot>	Definiše tekst ispod tabele (tfoot)	STF
<col>	Definiše atribute za kolone	STF
<colgroup>	Definiše grupe kolona u tabeli	STF
Stilovi	Namena	DTD
<style>	Definiše definiciju stila	STF
<div>	Definiše sekciju u dokumentu	STF
	Definiše sekciju u dokumentu	STF

Slika 3.8 Tabele i stilovi [Izvor: Autor]

Meta Informacije		
Tag	Namena	DTD
<head>	Definiše informacije o dokumentu	STF
<title>	Definiše naslov dokumenta	STF
<meta>	Definiše meta informacije	STF
<base>	Definiše osnovni URL za sve linkove na strani	STF
<basefont>	Izbegavati. Definiše osnovni font	TF
Programiranje		
Tag	Namena	DTD
<script>	Definiše skript	STF
<noscript>	Definiše ne-skript sekciju	STF
<applet>	Izbegavati. Definiše aplet	TF
<object>	Definiše ugnježđeni objekt	STF
<param>	Definiše parametar objekta	STF

Slika 3.9 Meta informacije i ugnježdavanje skripti i objekata [Izvor: Autor]

▼ Poglavlje 4

XML

ULOGA XML

XML koristi sličnu strukturu kao HTML, ali dok HTML definiše kako će se elementi prikazati, XML definiše šta elementi sadrže

XML (EXtensible Markup Language) je meta jezika koji se koristi za definisanje drugih specijalnih jezika koji služe za definisanje podataka koji se publikuju na veb stranama i u business-to-business dokumentima. XML koristi sličnu strukturu kao HTML, ali dok HTML definiše kako će se elementi prikazati, XML definiše šta elementi sadrže. Ovo je od posebnog značaja za business-to-business veb aplikacije i elektronsku razmenu podataka.

XML opisuje klasu objekata sa podacima koji se nazivaju XML dokumenti i delimično opisuje ponašanje računarskih programa koji procesiraju ove podatke. XML dokumenti se sastoje od entiteta koji sadrže raščlanjene i neraščlanjene podatke. Raščlanjeni podaci se sastoje od karaktera, od kojih su neki podaci, a neki oznake (markap). Markap kodira opis memorisanih entiteta i logičku strukturu dokumenta.

Pošto XML dokument nema unapred definisanu prezentaciju u veb čitaču potrebno je njeno definisanje. Za ovo se koristi nekoliko tehnika kao što su:

- Smeštanje veb sadržaja u XML dokument, a zatim njegovo prevođenje u HTML ili XHTML format korišćenjem XSL (eXtensible Stylesheet Language) transformacije
- CSS (Cascading Style Sheets) upotreboom eXtensible Style Sheet Transformations (XSLT)
- neki oblik server-side programiranja ili direktnim renderovanjem XML-a u veb čitaču vezivanjem CSS-a direktno za korisnički definisane elemente.
- XML je sličan SGML jeziku, pa i XML tagovi mogu da budu opisani u DTD formatu ili pomoću mnogo delotvornijeg gramatičkog mehanizma koji se zove šema.

RAZLIKE IZMEĐU HTML 4 I XML 1.0

Svaki HTML dokument može lako postati XML dokument ako se poštuju data pravila

Većina elemenata koje koriste HTML 4 i XML 1.0 su isti. Međutim, kako je XHTML praktično XML aplikacija, XHTML zahteva mnogo striktnije poštovanje nekih pravila pisanja elemenata. Praktično, svaki HTML dokument može lako postati XML dokument ako se poštjuju pravila od kojih se neka navode u daljem tekstu.

1. Dokument mora da bude ispravno oblikovan. Ovo najčešće znači da elementi moraju da imaju startne i završne tagove i da nema preklapanja elemenata. U XHTML-u samo prazni

(EMPTY) elementi mogu da budu bez završnih tagova. Ukoliko se za neki prazan element izostavlja završni tag, onda startni tag mora da se završi znakom „/“. Na primer, je ispravno napisan startni tag element br bez završnog taga.

2. Imena elemenata i atributa moraju biti napisana malim slovima.
3. Vrednosti atributa moraju da budu uvek između znakova navoda. Na primer **<td rowspan="3">**.
4. XML, pa samim tim ni XHTML, ne dozvoljava skraćeno pisanje para atribut-vrednost. U HTML-u je, na primer, dozvoljeno napisati skraćeno **<dl compact>**, a u XHTML-u se to mora napisati kao **<dl compact="compact">**.
5. Više blanko (**SP**) karaktera unutar vrednosti atributa se zamenjuju jednim blanko karakterom.

▼ Poglavlje 5

WEB SERVISI

ŠTA JE WEB SERVIS?

Veb servis je softverski sistem projektovan da podrži interoperativnu interakciju između računara preko mreže. Mehanizam za razmenu poruka je opisan u veb servis opisu WSD

Veb servis je softverski sistem projektovan da podrži interoperativnu interakciju između računara preko mreže. Svaki veb servis ima interfejs koji je opisan u mašinski čitljivom formatu, tačnije WSDL-om. Drugi sistemi saraduju sa veb servisom korišćenjem SOAP poruka.

Veb servis je apstraktни pojam koji mora da bude implementiran pomoću konkretnog agenta. Agent je softver ili hardver koji šalje i prima poruke. Servis je resurs koji se odlikuje apstraktnim setom funkcionalnosti koju obezbeđuje. Jedan isti servis se može obezbediti različitim agentima.

Svrha veb servisa je da obezbedi neku funkcionalnost, odnosno uslugu, u ime njegovog vlasnika, koji može biti neka osoba ili organizacija. Osoba ili organizacija koja obezbeđuje odgovarajućeg agenta za pružanje servisa se naziva **provajder entitet**.

Osoba ili organizacija koja želi da koristi veb servis se naziva entitet tražilac. Ovaj entitet koristi agenta tražioca za razmenu poruka sa agentom provajdera. Uobičajeno agent tražioca inicira razmenu poruka, mada to nije pravilo. Da bi mogli da uspešno razmenjuju poruke, entitet tražilac i entitet provajder najpre moraju da se saglase oko semantike i mehanizma za razmenu poruka.

Mehanizam za razmenu poruka je opisan u veb servis opisu WSD (**Web Service Description**). WSD je mašinski čitljiva specifikacija interfejsa veb servisa napisana jezikom za opis veb servisa WSDL (**Web Service Description Language**). WSD definiše formate poruka, tipove podataka, transportne protokole i format transportne serijalizacije koji treba da bude korišćen između agenta tražioca i agenta provajdera.

Semantika veb servisa je zajedničko očekivanje tražioca i provajdera o ponašanju servisa, a posebno se odnosi na odziv na poruke koje se šalju servisu. Praktično, semantika definiše značenje i svrhu interakcije.

Veb servisi mogu imati različite uloge u zavisnosti od scenarija interakcija. Potrebno je razmotriti kontekst u kome se veb servisi koriste i stanje trenutnog zadatka koji se izvršava. Isti veb servis može menjati uloge ili igrati više uloga u isto vreme. Veb servisi mogu biti:

- pružilac servisa
- primalac servisa

- posrednik.

POPULARNOST VEB SERVISA

Veb servisi dele poslovnu logiku, podatke i procese kroz programski interfejs preko mreže

Tokom poslednjih nekoliko godina, primena veb servisa se proširila i veb servisi su postali popularni kod programa aplikacija - i to sa dobim razlogom. Tehnologija veb servisa predstavlja važan način pomoću kojih kompanije komuniciraju jedne sa drugima, ali i sa svojim klijentima. Za razliku od tradicionalnih klijent / server modela, kao što su veb server ili veb stranica, veb servisi ne pružaju korisniku funkcionalnosti kroz grafički korisnički interfejs. Umesto toga, veb servisi dele poslovnu logiku, podatke i procese kroz programski interfejs preko mreže. U ovom slučaju interfejsi aplikaciju komuniciraju jedni sa drugima, a ne sa korisnicima. Programer može dodati grafički korisnički interfejs veb servisu, kako bi ponudio specifičnu funkciju korisnicima (kao što je veb stranica ili u izvršni program).

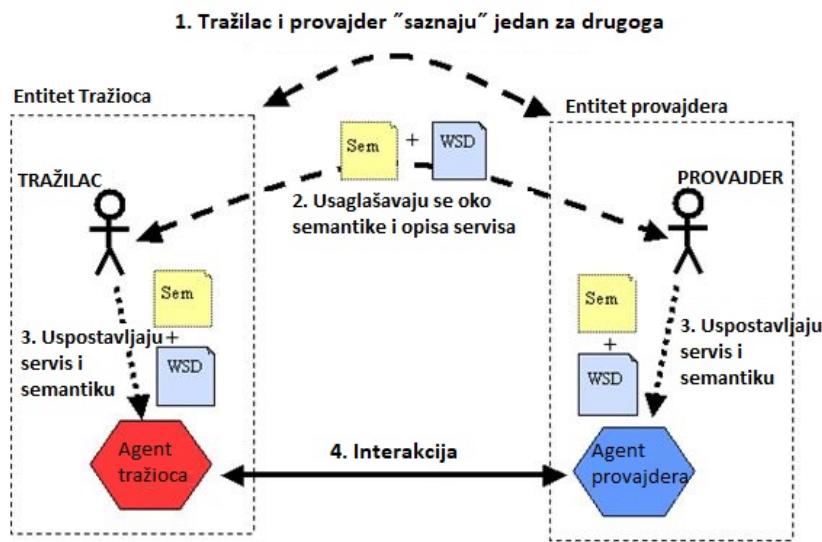
Veb servisi distribuiranog računarskog modela omogućavaju komunikaciju između aplikacija. Na primer, jedna aplikacija koja se koristi za kupovinu i naručivanje može da komunicira sa aplikacijom koja kontroliše zalihe za konkretnе stvari koje treba da se preraspoređuju. Zbog ovog nivoa integracije aplikacija, veb servisima je porasla popularnost, a prevashodno zbog toga što omogućavaju poboljšanje poslovnim procesa.

PROCES KORIŠĆENJA VEB SERVISA

Pregled procesa korišćenja veb servisa

Postoji mnogo načina na koji tražilac i provajder mogu biti angažovani prilikom upotrebe veb servisa. Generalno, proces korišćenja veb servisa se može razložiti na 4 koraka.

1. Entiteti tražioca i provajdera saznavaju jedan za drugog, ili bar jedan zna za drugog,
2. Tražilac i provajder se usaglašavaju oko opisa servisa i semantike koja će upravljati interakcijom između agenata,
3. Agenti tražioca i provajdera uspostavljaju servis i semantiku,
4. Agenti tražioca i provajdera razmenjuju poruke i tako izvršavaju neki zadatak u ime entiteta tražioca i entiteta provajdera.



Slika 5.1 Generalni proces angažovanja veb servisa [Izvor: https://www.w3.org/TR/2003/WD-ws-arch-20030808/images/roles_figure_1.gif]

ŠTA SU VEB SERVISI (VIDEO)?

Web Services Tutorial 1 - What Are Web Services?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 6

Pokazna vežba: HTML5

NOVI TAGOVI

Novi standard uvodi i bolju podršku formularima pa novi tagovi olakšavaju unos podataka sa html strana

Predviđeno vreme pokazne vežbe je 50 minuta.

HTML5 je novi Web standard. U okviru ovog novog standarda dolazi ceo jedan skup novih tagova koji bi trebali da obezbede nove mogućnosti, uklone ili bar umanje potrebu za spoljnim dodacima poput Flash-a, da bolje upravljaju greškama, da odgovarajućim novim tagovima umanje potrebu za skriptingom, olakšaju uređenje strana pomoću CSS-a (pogotovo prilagođeno CSS3), takođe treba da obezbedi nezavisnost od strane platforme i uređaja.

Posebno interesantne opcije novog HTML5 standarda su: **canvas** element koji prestavlja platno za crtanje i crteže, ugrađen multimedijalni plejer koji olakšava puštanje video i audio materijala, mogućnost pamćenja podataka u lokalnim resursima, novi elementi koji su usmereni na sadržaj poput **<article>**, **<footer>**, **<header>**, **<nav>**, **<section>**. Novi standard uvodi i bolju podršku formularima pa novi tagovi olakšavaju unos podataka sa html strana. Novi tagovi koji su od velike koristi u formularima su: **<calendar>**, **<date>**, **<time>**, **<email>**, **<url>** i **<search>**.

CANVAS

Canvas je područje po kome može da se crta, u slučaju HTML5 najčešće se crta pomoću javascripta

Još jedna novina u HTML5 je **canvas**. Veoma interesantan koncept pozajmljen od klasičnih programskih jezika. Canvas je područje po kome može da se crta, u slučaju HTML5 najčešće se crta pomoću javaskripta. Sam tag canvas obezbeđuje samo prostor za crtanie i dostavlja taj prostor na upotrebu onome ko crta po njemu. Tako da je sam HTML5 tag vrlo jednostavan:

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Da bi se po njemu crtalo potreban je odgovarajući JavaScript. Evo par primera:

Crtanje kvadrata:

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctxt=c.getContext("2d");
```

```
    ctxt.fillStyle="#FF0000";
    ctxt.fillRect(0,0,150,75);
</script>
```

Crtanje linije:

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctxt=c.getContext("2d");
    context.beginPath();
    ctxt.moveTo(10,10);
    ctxt.lineTo(150,50);
    ctxt.lineTo(10,50);
    ctxt.stroke();
</script>
```

Pravljenje gradijanta:

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctxt=c.getContext("2d");
    var grd=ctxt.createLinearGradient(0,0,175,50);
    grd.addColorStop(0,"#FF0000");
    grd.addColorStop(1,"#00FF00");
    ctxt.fillStyle=grd;
    ctxt.fillRect(0,0,175,50);
</script>
```

Učitavanje slike:

```
<script type="text/javascript">
    var c=document.getElementById("myCanvas");
    var ctxt=c.getContext("2d");
    var img=new Image()
    img.src="img_flwr.png"
    ctxt.drawImage(img,0,0);
</script>
```

STANDARDNI ATRIBUTI

HTML5 postoji čitav niz novih atributa koji obezbađuju nove funkcionalnosti

U predhodnoj verziji HTML-a bilo je samo par standardnih atributa poput *id*, *class*, *accesskey*, *lang* itd., ali u novoj verziji HTML5 postoji čitav niz novih atributa koji obezbađuju nove funkcionalnosti. Standardni atributi su opcije koje mogu da se dodaju na bilo koji tag. Standardni atribut u HTML5 koji je vezan za *drag'n'drop* mogućnosti je atribut ***draggable*** koji može da ima vrednosti ***true***, ***false*** i ***auto***. Atribut *hidden* postavlja taj element kao nevidljiv. Subject na primer povezuje element sa nekim elementom preko njegovog id-a

dakle vrednost ovog atributa je id nekog drugo elementa. Nova mogućnost je i uključivanja spellcheck mogućnosti na sadržajem nekog elementa, za to se koristi standardni atribut **spellcheck** koji može da ima vrednost true ili false. Mogućnost davanja dodatnog kontekstnog menija se postiže standardnim atributom **contextmenu** u koji ima vrednost menu_id. Postoji čak i mogućnost dodavanja sopstvenih novih atributa samo treba da počnu sa **data-** i onda ide naziv koji autor odabere.

DOGAĐAJI

HTML5 uvodi veliki broj novih akcije što obezbeđuje delako veće mogućnosti u kombinaciji sa JS

Događaji su akcije koje mogu da se izvedu nad nekim od elementama. HTML5 uvodi veliki broj novih akcije što obezbeđuje delako veće mogućnosti u kombinaciji sa JS. Na primer nad prozorom je bilo samo tri akcije, *onload*, *onblur* i *onfocus*. Sada ih ima daleko vise i njihov spisak se nalazi na slici 1.

Akcije	Vrednost	Opis
onafterprint	script	Pokreće skript nakon što se štampa završi
onbeforeprint	script	Pokreće skript pre puštanja na štampu
onbeforeunload	script	Pokreće skript pre nego što učita stranu
onblur	script	Pokreće skript kada prozor nema više fokus (postoji od ranije)
onerror	script	Pokreće skript kada se pojavi greška
onfocus	script	Pokreće skript kada prozor dobije fokus (postoji od ranije)
onhaschange	script	Pokreće skript kada se dokument promeni
onload	script	Pokreće skript kada se strana učita (postoji od ranije).
onmessage	script	Pokreće skript kada se aktivira pruka
onoffline	script	Pokreće skript kada dokument nije više online
ononline	script	Pokreće skript kada se nađe online
onpagehide	script	Pokreće skript kada se strana sakrije
onpageshow	script	Pokreće skript kada strana postane vidljiva
onpopstate	script	Pokreće skript kada se promeni istorija strane
onredo	script	Pokreće skript kada se pokrene naredba redo
onresize	script	Pokreće skript kada prozor promeni veličinu
onstorage	script	Pokreće skript kada se dokument učita
onundo	script	Pokreće skript kada se pokrene naredba undo
onunload	script	Pokreće skript kada korisnik napusti dokument

Slika 6.1 Događaji [Izvor: Autor]

AKCIJE

Akcije vezane za rad sam mišem su obogaćene sa dve akcije vezane za točkić miša i scroll onmousewheel i onscroll

Akcije vezane za rad sam mišem su obogaćene sa dve akcije vezane za točkić miša i ***scroll onmousewheel*** i ***onscroll***. Takođe su dodate akcije za rad sa ***drag'n'drop*** mehanizmom ***ondrag***, ***ondragend***, ***ondragenter***, ***ondragleave***, ***ondragover***, ***ondragstart*** i ***ondrop***.

Akcije vezane za nove mogućnosti multimedijalnog sadržaja prikazane su na slici 2.

Akcija	Vrednost	Opis
onabort	script	Pokreće skript kada se odustane od događaja.
oncanplay	script	Pokreće skript kada multimedijalni sadržaj može da se pusti, ali u nekim slučajevima zaustavlja baferovanje sadržaja.
oncanplaythrough	script	Pokreće skript kada sadržaj može da se pusti do kraja bez baferovanja.
ondurationchange	script	Pokreće skript ako se promeni dužina sadržaja.
onemptied	script	Pokreće skript kada sadržaj odjednom nestane usled greške u mreži, ili učitavanju ili iz bilo kog drugog razloga.
onended	script	Pokreće skript kada puštanje sadržaja dođe do kraja.
onerror	script	Pokreće skript kada dođe do greške u učitavanju.
onloadeddata	script	Pokreće skript kada se sadržaj učita.
onloadedmetadata	script	Pokreće skript kada se podaci o sadržaju učitaju.
onloadstart	script	Pokreće skript kada browser počne sa učitavanjem sadržaja.
onpause	script	Pokreće skript kada je puštanje sadržaja pauzirano.
onplay	script	Pokreće skript kada se pritisne dugme play.
onplaying	script	Pokreće skript kada se počne sa puštanjem sadržaja
onprogress	script	Pokreće skript kada se učitava sadržaj.
onratechange	script	Pokreće skript kada se promeni brzina puštanja.
onreadystatechange	script	Pokreće skript kada je promenjeno stanje „Ready-state”
onseeked	script	Pokreće skript kada prestane da se traži sadržaj, odnosno kada je opcija seeking false.
onseeking	script	Pokreće skript kada počne da traži sadržaj odnosno kada je opcija seeking true.

Slika 6.2 Akcije u vezi sa multimedijom [Izvor: Autor]

▼ Poglavlje 7

Pokazna vežba: DHTML

DHTML - OSNOVE

DHTML - Dynamic HyperText Markup Language

Predviđeno vreme pokazne vežbe je 40 minuta.

DHTML je termin koji označava kombinaciju web tehnologija koje se koriste za kreiranje dinamičnih web stranica. Dinamičkim HTML-om se mogu kreirati animacije, dinamični meniji ili različiti efekti za tekst, slike ili bilo koji drugi element web stranice.

Tehnologije koje se najčešće koriste za DHTML su:

- HTML
- JavaScript
- CSS
- DOM ([Document Object Model](#))

Neke od prednosti korišćenja DHTML-a su brzina, lako menjanje sadržaja i dodavanje novih funkcionalnosti na web stranice, efikasnije upravljanje sadržajem.

Važno je napomenuti da DHTML nije ni skripting jezik ni web standard već samo način kombinovanja tehnologija koje statične HTML stranice čine dinamičnim i interaktivnim.

PRIMER 1

Prvi zadatak

Zadatak: (Predviđeno vreme za izradu zadatka: 10 minuta)

Kreirati event kojim se duplim klikom na neki deo teksta on menja u drugi tekst.

Rešenje:

```
<html>
  <head>
    <script type="text/javascript">
      function promeniTekst(txt)
      {
        document.getElementById('tekst').innerHTML='DHTML je termin koji
označava kombinaciju web tehnologija koje se koriste za kreiranje dinamičnih web
```

```
stranica.';  
        }  
    </script>  
</head>  
<body>  
    <p>Dva puta kliknite na "Šta je DHTML?"</p>  
    <table>  
        <tr>  
            <th ondblclick="promeniTekst()">Šta je DHTML?</th>  
            <th id="tekst"></th>  
        </tr>  
    </table>  
</body>  
</html>
```

Na slikama 1 i 2 je prikazan rezultat zadatka.



Slika 7.1 Stranica pre događaja [Izvor: Autor]

Kada se dva puta klikne na tekst "Šta je DHTML?" dobija se sledeći prikaz:



Slika 7.2 Prikaz stranice nakon duplog klika [Izvor: Autor]

PRIMER 2

Drugi zadatak

Zadatak: (Predviđeno vreme za izradu zadatka: 10 minuta)

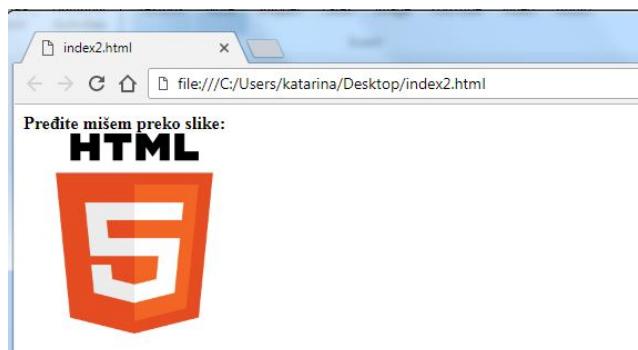
Na stranicu postaviti sliku koja će se povećavati kada se preko nje pređe mišem.

Rešenje:

```
<html>  
    <head>  
        <script type="text/javascript">  
            function moveover()
```

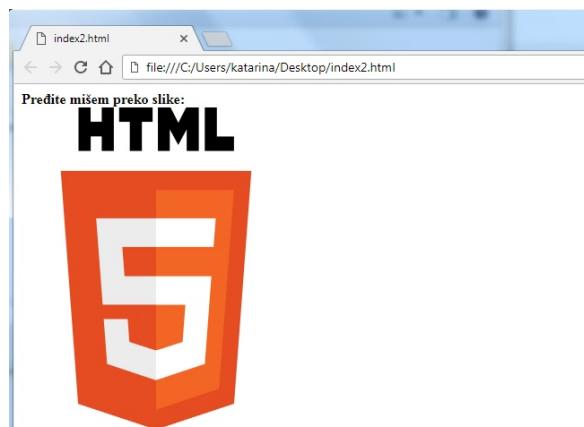
```
{  
    document.getElementById('image').width="300";  
    document.getElementById('image').height="360";  
}  
function moveback()  
{  
    document.getElementById('image').width="200";  
    document.getElementById('image').height="180";  
}  
</script>  
</head>  
<body>  
    <b>Predite mišem preko slike:</b><br/>  
      
</body>  
</html>
```

Na slici 3 je prikazana slika pre povećanja:



Slika 7.3 Slika pre povećanja [Izvor: Autor]

Kada se mišem pređe preko slike, ona izgleda ovako:



Slika 7.4 Slika nakon povećanja [Izvor: Autor]

PRIMER 3

Treći zadatak

Zadatak: (Predviđeno vreme za izradu zadatka: 10 minuta)

Napraviti trešćući link, odnosno link koji će na 100 milisekundi da menja boju.

Rešenje:

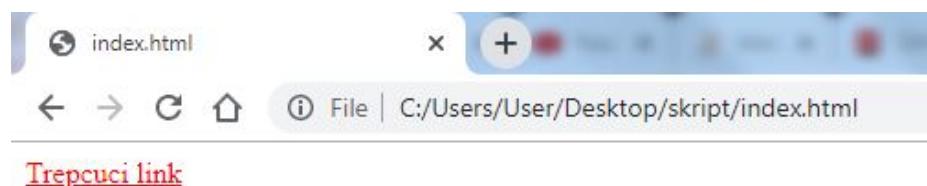
```
<html>
<head>
<script type="text/javascript">
function blinklink()
{
if (!document.getElementById('blink').style.color)
{
    document.getElementById('blink').style.color="red";
}
if (document.getElementById('blink').style.color=="red")
{
    document.getElementById('blink').style.color="black";
}
else
{
    document.getElementById('blink').style.color="red";
}
timer=setTimeout("blinklink()",100);
}

function stoptimer()
{
clearTimeout(timer);
}
</script>
</head>

<body onload="blinklink()" onunload="stoptimer()">
<a id="blink" href="default.asp.htm">Blinking link</a>
</body>

</html>
```

Na slici 5 prikazana je prva boja linka:



Slika 7.5 Prva boja linka [Izvor: Autor]

Na slici 6 prikazana je druga boja linka:



Slika 7.6 Druga boja linka [Izvor: Autor]

▼ Poglavlje 8

Zadaci za samostalni rad: HTML5 i CSS3

OPIS ZADATKA ZA SAMOSTALNU VEŽBU

Kreirati HTML prezentaciju

Predviđeno vreme izrade zadatka je 30 minuta.

Za potrebe Univerziteta Metropolitan kreirati HTML prezentaciju, koja sadrži sledeće elemente i funkcionalnosti:

- Početna strana - sa logoom fakulteta i banerom
 - Gornji, horizontalni, levo poravnat meni koji se nalazi na svim stranicama i vodi ka stranama: Početna (Home strana), Domaći i projektni zadaci, Predispitne obaveze, Lista, Kontakt;
 - Vertikalni meni, poravnat sa desne strane, koji se nalazi na svim stranicama a koji sadrži potkategorije izabrane kategorije iz vertikalnog menija:
 - Domaći i projektni zadaci – sadrži spisak predmeta i poseduje mogućnost preuzimanja domaćih i projektnih zadataka za svaki predmet pojedinačno,
 - Predispitne obaveze – tabelarni prikaz broja poena za svaki domaći i projektni zadatak za svakog studenta po predmetu sa zbirom svih poena za svakog studenta po predmetu.
 - Lista – spisak studenata, grupisanih po predmetima, koji imaju pravo da prijave ispit u važećem roku.
- Kontakt - kontakti studentske službe, asistenata i profesora.

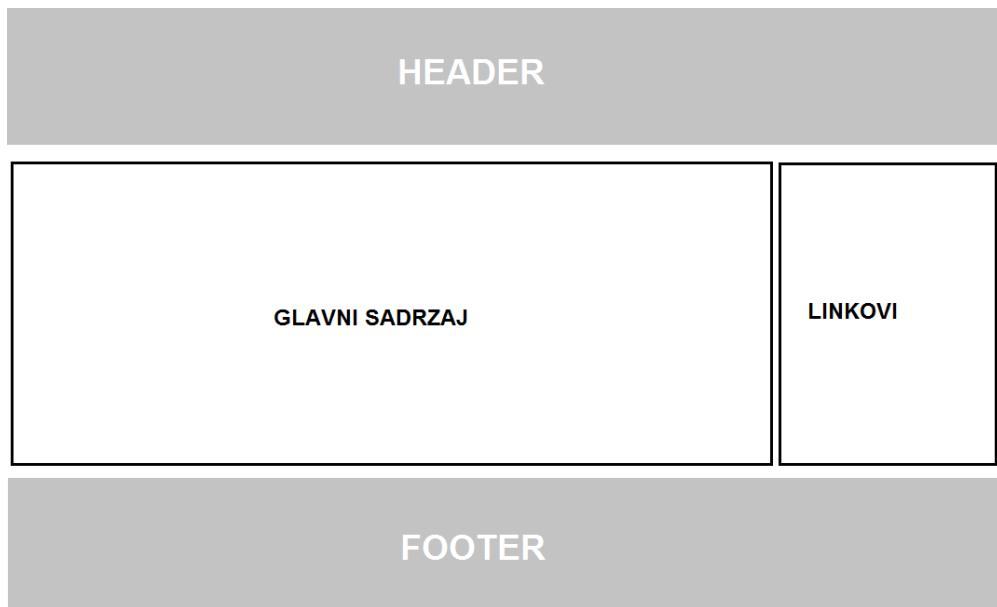
Potrebno je da se koriste HTML5 media elementi i CSS3 elementi.

POZICIONIRANJE - VEŽBANJE

Pozicionirati elemente kao na slici

Vreme predviđeno za izradu zadatka je 15 minuta.

Koristeći naučene tehnike pozicioniranja, napraviti stranicu sa sledećim elementima:



Slika 8.1 Izgled strane za vežbanje [Izvor: Autor]

✓ Poglavlje 9

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ04

Kreirati prezentaciju koristeći HTML5 i CSS4

Predviđeno vreme za izradu domaćeg zadatka je: 45 minuta.

Kreirati HTML prezentaciju na temu po Vašem izboru. HTML prezentacija treba da:

- Sadrži bar tri strane koje su povezane i koje omogućavaju vraćanje na početnu stranu
- Koristi HTML5 media elemente
- Koristi pomenute CSS4 elemente.
- Pozicionirati elemente pomoću naučenih tehnika
- U footer staviti ime, prezime i broj indeksa

Odgovarajuće datoteke snimite pod imenom **IT210-DZ04-Ime_Prezime_brojIndexa** sa odgovarajućom ekstenzijom gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovan dokumente poslati na adresu predmetnog asistenta.

✓ ZAKLJUČAK

ZAKLJUČAK

U ovom predavanju je dat pregled veb tehnologija i standarda koji su ključ za rad svih veb sistema. Prevashodno dat je pregled rada HTTP protokola kao i njegov način funkcionisanja i njegove arhitekture. Kao nezaobilazan deo veb tehnologija upoređeni su HTML, XHTML i XML.

Literatura

- [1] Network Working Group, RFC2616, The Internet Society, 1999.
- [2] Network Working Group, RFC2616, Section 1, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec1.html> (20.12.2020).
- [3] Network Working Group, RFC2616, Section 3, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html> (20.12.2020).
- [4] Network Working Group, RFC2616, Section4, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html>, (20.12.2020).
- [5] Network Working Group, RFC2616, Section 5, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html>, (20.12.2020).
- [6] Network Working Group, RFC2616, Section 6, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html>, (20.12.2020).
- [7] Network Working Group, RFC2616, Section 7, <http://www.w3.org/Protocols/rfc2616/rfc2616-sec7.html>, (20.12.2020).
- [8] Web Services Architecture, W3C Working Group Note 11 February 2004, file:///C:/Users/miroslava.raspopovic/AppData/Local/Temp/Rar\$Dla0.358/%20http://www.w3.org/TR/2004/NOTE-ws-arch-20040211https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ (20.12.2020).
- [9] Tanenbaum, Andrew S. Computer Networks. Upper Saddle River, NJ: Prentice Hall PTR.



IT210 - SISTEMI IT

DIGITALNI MEDIJI I RAZVOJ VEB
SAJTOVA

Lekcija 05

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 05

DIGITALNI MEDIJI I RAZVOJ VEB SAJTOVA

- ✓ DIGITALNI MEDIJI I RAZVOJ VEB SAJTOVA
- ✓ Poglavlje 1: DIGITALNI MEDIJI NA VEBU
- ✓ Poglavlje 2: KOMPRESIJE ZAPISA
- ✓ Poglavlje 3: STRIMOVANJE
- ✓ Poglavlje 4: RAZVOJ KORISNIČKOG VEB INTERFEJSA
- ✓ Poglavlje 5: PITANJE DOSTUPNOSTI
- ✓ Poglavlje 6: IMPLEMENTACIJA I INTEGRACIJA
- ✓ Poglavlje 7: Pokazna vežba: HTML5 forme
- ✓ Poglavlje 8: Pokazna vežba: HTML5 i CSS4
- ✓ Poglavlje 9: Pokazna vežba: HTML multimedije
- ✓ Poglavlje 10: Zadaci za samostalan rad: HTML5 i CSS4
- ✓ Poglavlje 11: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da da pregled digitalnih medija i da da osnovne smernice za razvoj veb sajtova

U ovoj lekciji biće obrađene sledeće teme:

1. Digitalni multimediji

- Digitalne biblioteke
- Formati medija
- Streaming medija

2. Razvoj veb sajtova

- Arhitektura informacija
- Veb interfejs
- Pitanje dostupnosti

▼ Poglavlje 1

DIGITALNI MEDIJI NA VEBU

DIGITALNE BIBLIOTEKE

Digitalne biblioteke kombinuju računarsku i komunikacionu opremu, sadržaj i softver potreban za reprodukovanje, emuliranje, sakupljanje, katalogiziranje, traženje i diseminaciju informacija

Razvoj Interneta i World Wide Web-a je omogućio da se ogromna količina digitalnih informacija publikuje na veb serverima. S druge strane, ovako ogromna količina informacija otežava da se pronađe tražena informacija, čime je ugrožena jedna od osnovnih funkcija veba. Tako je veb zajednica došla do stadijuma u kome više nije dovoljno imati pristup informacijama, nego je potrebno imati i podršku za efikasno traženje informacija.

Da bi se pretraživanje informacija na vebu učinilo efikasnim, uvode se nove funkcionalnosti za indeksiranje sadržaja i traženje informacija koje se nazivaju digitalne biblioteke. Kako je veb globalni informacioni prostor, i ove biblioteke se nazivaju globalne digitalne biblioteke.

Osnovna namena digitalnih biblioteka je efikasno i efektivno nalaženje informacija. Međutim, ovo nije dovoljno.

Osnovne aktivnosti korisnika digitalnih biblioteka se mogu klasifikovati u sledeće kategorije:

- Lociranje i izbor relevantnih izvora,
- Pribavljanje informacija iz izvora,
- Interpretiranje pribavljenih informacija, Upravljanje procesom filtriranja informacija u lokalnu,
- Deljenje informacija sa drugima.

Ove aktivnosti ne mora da budu sekvenčialne, već mogu da se preklapaju i ponavljaju. Nema jedinstvene definicije globalnih digitalnih biblioteka. Sa aspekta upravljanja informacija, digitalne biblioteke su sistemi koji kombinuju računarsku opremu, komunikacionu opremu, sadržaj i softver potreban za reprodukovanje, emuliranje, sakupljanje, katalogiziranje, traženje i diseminaciju informacija. Sa aspekta korisnika, digitalne biblioteke su sistemi koji obezbeđuju efikasan pristup velikim, organizovanim skladištima informacija i znanja.

SVRHA DIGITALNIH BIBLIOTEKA

Digitalne biblioteke pružaju korisnicima široki set različitih usluga, kao što su: traženje, organizacija, pristup i distribucija

Ako se krene od prethodnih definicija, može se reći da je svrha digitalnih biblioteka:

- Da na sistematski način prikupljaju, čuvaju i organizuju digitalne informacije i znanje u obliku digitalnih bibliotečkih kolekcija,
- Da omoguće ekonomično i efikasno dostavljanje informacija svim delovima društva,
- Da ojačaju komunikaciju i kolaboraciju između i unutar istraživačkih, poslovnih, vladinih i edukativnih zajednica,
- Da obezbede svim ljudima uslove za učenje tokom celog života.

Digitalne biblioteke pružaju korisnicima široki set različitih usluga, kao što su: traženje, organizacija, pristup i distribucija. Ovi servisi su omogućeni setom alata koji rade nad informacionim objektima. Pod informacionim objektima se podrazumevaju digitalni sadržaji, odgovarajući metapodaci, metodi usluga i metodi upravljanja.

Ključne komponente digitalnog bibliotečkog sistema su: korisnički interfejs, skladište digitalnih objekata, sistem za upravljanje i sistem za pretraživanje.

Korisnički interfejs digitalnog bibliotečkog sistema se sastoji od dva dela. Jedan deo služi za stvarnu interakciju bibliotečkog sistema sa korisnikom. Drugi deo se sastoji od programa koji korisniku pružaju usluge kao što su: interpretiranje digitalnih informacija, pregovaranje o uslovima korišćenja informacija, konverzija formata zapisa, itd.

SKLADIŠTE DIGITALNIH OBJEKATA

Skladište digitalne biblioteke služi za čuvanje informacionih objekata koje čine digitalni sadržaji i metapodaci

Skladište digitalne biblioteke služi za čuvanje informacionih objekata. Informacione objekte čine digitalni sadržaji i metapodaci. Digitalni sadržaj može biti:

- tekst
- bit mapirana slika
- vektorski crtež
- audio zapis
- video zapis

ili kombinacija različitih tipova sadržaja. **Metapodaci uključuju i globalne jedinstvene identifikatore za digitalne objekte.** Skladišta mogu biti različita, kao što su moderne digitalne skladišta, zastarele baze podataka, veb serveri itd. Da bi digitalni bibliotečki sistem mogao da pristupi različitim vrstama skladišta uvodi se **protokol za pristup skladištima** (RAP - Repository Access Protocol) kao interfejs između bibliotečkog sistema i skladišta.

Zadatak RAP-a je da:

- Prepozna prava pristupa pre nego što korisnik pristupi digitalnom objektu,
- Podržava široki spektar metoda diseminacije digitalnih objekata
- Obezbedi otvorenu arhitekturu sa dobro definisanim interfejsima.

SISTEMI ZA UPRAVLJANJE I PRETRAŽIVANJE

Zadatak sistema za upravljanje je da upravlja identifikacijom i čuvanjem digitalnih objekata, a sistema za pretraživanje da ga na efikasan način pruži informacije o njemu

Sistem za upravljanje

Sistem za upravljanje ima dva osnovna zadatka:

- da upravlja identifikacijom digitalnih objekata
- da upravlja sadržajem koji se čuva u skladištima.

Pored toga sistem za upravljanje ima zadatak da vrši indeksiranje digitalnih objekata i priprema kolekcije digitalnih objekata na osnovu definisanih kriterijuma.

Sistem za pretraživanje

Sistem za pretraživanje ima zadatak da na efikasan način korisniku pruži informacije o traženom digitalnom objektu. Ovaj sistem koristi metapodatke, indekse i kolekcije da na najbrži način nađe tražene informacije.

▼ 1.1 MULTIMEDIJA

ŠTA SU TO MULTIMEDIJE?

Pod multimedijom se uobičajeno podrazumeva korišćenje računara za kreiranje, čuvanje i interpretaciju multimedijalnog sadržaja

Multimedija se može definisati kao korišćenje različitih medija, kao što su tekst, grafika, animacija, audio i video, za prezentovanje informacija. Danas se multimedija uglavnom zasniva na medijima zapisanim u digitalnom formatu, pa se pod multimedijom uobičajeno podrazumeva korišćenje računara za kreiranje, čuvanje i interpretaciju multimedijalnog sadržaja.

Gotovo da nema oblasti ljudskog rada i života u kojoj se multimedija ne koristi. Posebnu primenu multimedija je našla na vebu, jer omogućuje da se tekstualne informacije dodatno obogate slikama, zvukom i filmom. Na taj način se dobija takozvani bogati sadržaj.

1.2 FORMATI SLIKA

NAJČEŠĆE KORIŠĆENI DIGITALNI FORMATI SLIKA

Najčešće korišćeni digitalni formati za slike su GIF, JPEG, PNG, TIFF

Najčešće korišćeni digitalni formati za slike su:

- **GIF - Graphic Interchange Format**
- **JPEG - Joint Photographic Experts Group**
- **PNG - Portable Network Graphic**
- **TIFF - Tagged Image File Format**
- **Animirani GIF**

GIF je bio prvi format koji je bio korišćen za prikazivanje slika na vebu, i do danas je ostao među najpopularnijim. Podržan je od svih veb čitača. GIF je 8-bitni indeksni format, pa je svaki piksel slike opisan jednom od prethodno indeksiranih 256 boja iz unapred definisane palete. GIF89a verzija ovog formata koristi LZW (Lempel-Zev-Welch) kompresiju bez gubitaka i ima opciju za interlakciju. Slici zapisanoj u ovom formatu se može zadati određeni stepen transparentnosti. Pored toga ovaj format podržava animirane slike.

JPEG format omogućuje zapisivanje bit mapiranih kolor slika korišćenjem 24-bitnih RGB informacija o boji piksela. Ovaj format omogućuje i zapisivanje slika sa različitim nivoima sivog. JPEG je u stvari kompresioni algoritam koji se koristi najčešće za bit mapiranih slika. Ovaj algoritam spada u klasu metoda kompresije sa gubitkom informacija. Međutim ovi gubici su neprimetni za ljudsko oko..

Čak i pri stepenu kompresije od 10:1 do 20:1 gubici informacija nisu vidljivi. Slike zapisane u JPEG formatu je pre prikazivanja potrebno dekompresovati, što veb čitači rade automatski.

PNG (izgovara se ping) format je specijalno razvijen za slike koje se prikazuju na vebu. Međutim, zbog nedostatka dobrih alata za kompresiju, ovaj format nije tako široko prihvaćen kao GIF. PNG format podržava zapisivanje bit mapiranih slika u 8-bitnom indeksnom format (kao i GIF), 16-bitnom formatu za slike sa različitim nivoima sivog i 48-bitnom RGB kolor formatu. Za kompresiju podataka se koristi metod bez gubitaka informacija. Od 1996. godine je postao preporučeni format W3C konzorcijuma.

TIFF (**T**agged **I**mage **F**ile **F**ormat) format omogućuje zapis monohromatskih slika, slika sa različitim nivoima sivog, 8-bitnih i 24-bitnih kolor bitmapiranih slika. TIFF koristi nekoliko kompresionih metoda. LZW metod kompresije omogućuje kompresioni odnos od 1.5:1 do 2:1.

Animirani GIF se zasniva na statičkom GIF formatu i koristi se za zapis kratkih bitmapiranih animacija. Jedna animacija se sastoji od više slika, odnosno frejmova, koje se nalaze u slojevima jedan iznad drugog. Uzastopnim prikazivanjem pojedinih frejmova gledalac doživljava animaciju. GIF animacije ne sadrže zvuk

✓ 1.3 FORMATO AUDIO ZAPISA

POTREBA ZA KOMPRESIJAMA AUDIO ZAPISA

Formate audio zapisa možemo podeliti u nekomprimovani i komprimovani format sa i bez gubitaka

Audio zapisi se koriste za čuvanje zvuka. Posebnu primenu audio zapisi imaju za čuvanje govora i muzike. Za zapis zvuka prilikom snimanja se najčešće koriste WAV i AIFF format. I jedan i drugi format zapisuju zvuk bez kompresije. Potreban memorijski prostor M u bajtovima za audio zapis dužine jedne sekunde se može izračunati prema izrazu:

$$M = F * A * K / 8$$

gde je:

F - frekvencija uzorkovanja u Hz

A - broj bitova za zapis amplitude zvuka (uobičajeno 8 ili 16) K - Broj kanala (K=1 za mono i K=2 za stereo)

Ako se, na primer, snima stereo zvuk sa frekvencijom uzorkovanja od 22.050 Hz u 16-bitnom kvalitetu, onda je za svaku sekundu audio zapisa potrebno 88,2 KB prostora. Očigledno je da je za prenos ovako kvalitetnog zvuka preko Interneta potrebno imati jako brzu vezu. Pošto Internet ne garantuje uvek dovoljno brzu vezu vrši se kompresija audio podataka.

Zbog toga formate audio zapisa možemo podeliti na:

- Nekomprimovani audio formati, kao što su WAV, AIFF, AU
- Komprimovani audio formati bez gubitaka, kao što su FLAC, WavPack, Shorten, Apple Lossless
- Komprimovani audio formati sa gubicima, kao što su Vorbis, MP3

Microsoft-ov Windows Media Audio (WMA) može da zapisuje audio i bez i sa gubicima.

WAV, AIFF I MIDI FORMATI

Zbog preobimne veličine zapisa WAV i AIFF formati se retko koriste za prenos audio zapisa preko Interneta

WAV (Waveform Audio File) format je originalno razvijen kao standardni audio format za Windows operativni sistem, ali je podržan i od drugih operativnih sistema. WAV format podržava proizvoljnu frekvenciju uzorkovanja zvuka (sampling rate) i proizvoljan broj bitova za zapis amplitude. Frekvencija uzorkovanja zvuka se uobičajeno kreće u intervalu od 8 kHz do 48 kHz. Za potrebe reprodukcije zvuka na vebu se koriste uglavnom frekvence od 8 i 11,025 kHz, jer su tako veličine zapisa najmanje. Za zapis amplitude zvučnog talasa uglavnom se koristi 8 ili 16 bitova. Zvuk se može snimiti kao mono ili stereo.

AIFF (Audio Interchange File Format) je razvijen kao standardni audio format za Macintosh operativne sisteme. Ovaj format može da podrži do šest kanala zvuka, a frekvencija uzorkovanja i broj bitova za zapis amplitude se mogu proizvoljno birati.

Zbog preobimne veličine zapisa WAV i AIFF formati se retko koriste za prenos audio zapisa preko Interneta. U tu svrhu su razvijeni drugi formati, kao što su MP3 i RealAudio. Ovi formati komprimuju zapis uz odgovarajući gubitak kvaliteta.

MIDI (Musical Instrument Digital Interface) je u odnosu na prethodne, sasvim drugi tip audio formata. Originalno je razvijen za komunikaciju između digitalnih muzičkih instrumenata. Ovaj zapis ne sadrži digitalnu reprezentaciju analognog zvuka. Umesto toga MIDI zapis se sastoji od numeričkih komandi koje iniciraju seriju nota sa instrukcijama o njihovoj dužini i jačini. Ove note se mogu reprodukovati MIDI plejerima ugrađenim u audio kartice. MIDI fajlovi su, u odnosu na druge audio formate, ekstremno mali. Tako je, na primer, minut muzike u MIDI formatu 10 KB, a u WAV formatu 10 MB. Veliko ograničenje MIDI zapisa je što je pogodan samo za zvukove muzičkih instrumenata, a ne i za ostale vrste zvuka.

▼ 1.4 FORMATI VIDEO ZAPISA

VEZA IZMEĐU SLIKA I VIDEO ZAPISA

Čisti video zapis, bez zvuka, se sastoji od niza digitalnih slika ili frejmova, koji se prikazuju određenom brzinom

Digitalni video zapis se dobija korišćenjem digitalnih video kamera ili digitalizacijom analognog video snimka. Čisti video zapis, bez zvuka, se sastoji od niza digitalnih slika ili frejmova, koji se prikazuju određenom brzinom. Brzina smenjivanja frejmova je definisana sa tri faktora. Pre svega, brzina mora da bude dovoljno velika da kretanja na filmu teku glatko. **Iskustvo je pokazalo da se pri prikazivanju 25 slika u sekundi i više sva kretanja na filmu čine kontinualna.**

Drugi faktor koji utiče na broj frejmova u sekundi je kapacitet prenosnog medija za transport video zapisa. **Što je više frejmova u sekundi, više podataka treba preneti u sekundi, pa su i veći zahtevi za prenosnim kapacitetom.** Zbog toga se teži ka minimalno mogućem broju frejmova u sekundi koji ne ugrožava kvalitet videa.

Treći faktor je vezan za uređaj za prikazivanje video zapisa. CRT monitori stvaraju sliku tako što se elektronskim zrakom pobude zrna fosfora koja onda zasvetle samo nekoliko milisekundi. Da se slika ne bi izgubila zrna fosfora je neophodno pobuđivati najmanje 50 puta u sekundi. Ovo se naziva osvežavanje slike. Kako bi 50 frejmova u sekundi previše opteretilo prenosne medije, video se ipak zapisuje sa 25 frejmova u sekundi, ali se za emitovanje koristi tehnika preplitanja (engl. **interlace**).

Slika se na ekranu osvežava 50 puta u sekundi, ali tako što se u jednom prolazu elektronskog zraka osvežavaju sve parne linije, a u drugom prolazu sve neparne linije. Kako oko ne može da primeti treperenja koja potiču od malih objekata, slika izgleda stabilno. Drugim rečima, jedan

frejm se prikazuje sa dva prolaza elektronskog zraka. Ova tehnologija se naziva preplitanje 2:1 i ona omogućuje da se sa 25 frejmova u sekundi dobije stabilna slika na ekranu.

POTREBA ZA KOMPRESIJOM VIDEA

Zbog velikog potrebnog broja slika u sekundi, veličina digitalnih video zapisa je ogromna, pa se zapisi komprimuju

Bazirano na prethodnim razmatranjima, u svetu se koriste dva standarda za emitovanje video snimaka. U Evropi, Kini i Australiji se koristi PAL sistem koji radi sa 25 frejmova u sekundi i 50 iscrtavanja slike po ekranu. U Severnoj Americi i Japanu se koristi NTSC sistem koji radi sa 30 frejmova u sekundi i 60 iscrtavanja slike po ekranu. Broj iscrtavanja u slike sekundi odgovara frekvenciji električne struje u ovim zemljama, čime se izbegavaju smetnje pri prikazu slike na ekranu.

Neka je, na primer, potrebno snimiti digitalni zapis sa 25 slika u sekundi u trajanju od 10 minuta sa slikom veličine 512 x 512 piksela i dubinom piksela od 24 bita. Onda će veličina zapisa biti

$$10 * 60 * 25 * 512 * 512 * 3 = 11.796.480.000 \text{ Byte} = 10,98 \text{ GB}$$

Zbog ovako velikog potrebnog broja slika u sekundi, veličina digitalnih video zapisa je ogromna, pa se zapisi komprimuju. O tehnikama kompresije će biti više reči kasnije.

FLASH

Fleš video zapisi se mogu publikovati na vebu ali oni sami za sebe takođe mogu biti veb stranice jer imaju ugrađenu interaktivnost

Flash je multimedijalni format kompanije Adobe (originalno Macromedia) koji daje mogućnost zapisa animacija velikog formata, interaktivne grafike i integrisanog zvuka. Veličina zapisa je relativno mala jer se za animaciju i crteže koristi vektorska grafika. Mala veličina zapisa čini Flash format zbog toga vrlo pogodnim za veb aplikacije. Fleš video zapisi se mogu publikovati na vebu ali oni sami za sebe takođe mogu biti veb stranice jer imaju ugrađenu interaktivnost. Zahvaljujući naprednoj skripting tehnici Fleš omogućuje sledeće multimedijalne forme:

- animaciju
- interaktivne navigacione alate
- animirane banere
- interaktivne mape sa mogućnošću zumiranja
- interaktivne igre
- kreiranje kompleksnih veb aplikacija sa interaktivnim formama i naprednim interfejsom
- crtane filmove itd.

Za kreiranje Flash filmova se koristi Flesh alat za produkciju koji informacije o filmu zapisuje u .fla datoteci. Ova datoteka sadrži odvojene delove o svim elementima koji čine film i njihovu vremensku liniju. Format zapisa je takav da se lako može editirati. Ako se film spremi za publikovanje na vebu informacije se zapisuju u Small Web Format-u .swf.

Prednosti korišćenja Flesh-a su mala veličina datoteka, skalabilnost prozora u kome se vrši prikaz bez gubljenja detalja, visok kvalitet slike, integrisani zvuk, ugrađena striming tehnologija, i dobra podrška od svih veb čitača. Posebna prednost Fleš formata je što se radi o otvorenom formatu tako da ga softverske kuće koriste za različite primene.

Nedostatak Flesh formata je u tome što se za prikaz sadržaja zahteva plug-in Flesh plejer. Ovaj plejer se može prevući i instalirati besplatno, ali mnogi korisnici to odbijaju da urade.

▼ 1.5 FORMATI MULTIMEDIJALNIH KONTEJNERA

ŠTA SU MULTIMEDIJALNI KONTEJNERI?

Multimedijalni kontejneri su datoteke koje sadrže različite vrste podataka o pojedinim tipovima medija koji čine jednu celinu

Kontejneri ili multimedijalni kontejneri su datoteke koje sadrže različite vrste podataka o pojedinim tipovima medija koji čine jednu celinu. Jednostavniji kontejneri sadrže samo audio informacije, dok napredni kontejneri mogu da sadrže audio i video podatke, tekst prevoda, poglavila i metapodatke zajedno sa informacijama potrebnim za vremensku sinhronizaciju prilikom emitovanja.

U poznatije multimedijalne kontejnere spadaju:

- AVI - standardni Microsoft-ov kontejner
- MOV - standardni QuickTime kontejner
- MPEG-2 Transport Stream - standardni kontejner za digitalno emitovanje
- MP-4 - standardni kontejner za MPEG-4
- RealMedia - standardni kontejner za RealVideo i RealAudio

AVI (Audio Video Interleave) je format Microsoft-ovog multimedijalnog kontejnera koji se koristi još od 1992. godine. U jednom AVI kontejneru se može naći više vremenski sinhronizovanih audio i video strimova, mada se ta opcija retko koristi. AVI format je specijalni slučaj RIFF (Resource Interchange File Format) formata koji podatke u fajlu deli na segmente.

Svaki segment je identifikovan takozvanim FourCC tagom. AVI fajl uzima oblik jednog segmenta u datoteci u RIFF formatu. Ovaj segment je podeljen na dva obavezna i jedan opcioni podsegment.

Prvi pod segment je identifikovan hdrl tagom. Ovaj podsegment je zaglavljivo datoteke i sadrži metapodatke o videu, kao što su širina, visina i broj frejmova. Drugi podsegment je identifikovan movi tagom. Ovaj podsegment sadrži stvarne audio-vizuelne podatke koji čine jedan AVI film. Treći, opcioni, podsegment je identifikovan idx1 tagom. U njemu su sadržani indeksi lokacija segmenata unutar datoteka.

MOV KONTEJNER

Unutar MOV kontejnera se može zapisati više staza koje su vremenski sinhronizovane

Audio-video podaci u movi podsegmentu mogu biti kodirani i dekodirani softverskim modulima koji se nazivaju codec (**coder - decoder**). AVI datoteka može da ima audio-vizuelne podatke unutar segmenata i gotovo svim kompresionim šemama kao što su Full Frames (**Uncompressed**), Intel Real Time Video, Indeo, Cinepak, Motion JPEG, Editable MPEG, VDOWave, ClearVideo / RealVideo, QPEG, MPEG-4, XviD, DivX i druge.

Kontejner **MOV** je kontejner multimedijalne tehnologije kompanije Apple Computer. Korišćenjem alata za produkciju QuickTime ove firme unutar MOV kontejnera se može zapisati više staza koje su vremenski sinhronizovane. Svaka staza može da sadrži poseban tip multimedijalnih podataka kao što su audio, video, prelazni efekti i tekst. Podaci na stazama su digitalno kodirani strimovi ili reference na druge multimedijalne datoteke koje se nalaze na mreži.

Kada se multimedijalni materijal potpuno izmontira zapisuje se video prezentacija. Ona sadrži sve multimedijalne sadržaje koji su dodati na vremenu liniju.

Ovako dobijena prezentacija se može:

- Prikazivati na računaru sa diska ili CD-a
- Staviti na neki deljeni memoriski uređaj na mreži kako bi drugi mogli da je vide
- Publikovati na nekoj veb strani kao video strim korišćenjem Windows Media Services.

▼ Poglavlje 2

KOMPRESIJE ZAPISA

VRSTE KOMPRESIJA

Nakon dekompresije zapisa komprimovanog sa kompresijom bez gubitaka, dobija se originalna datoteka, a sa gubicima, dobija se datoteka koja nije ista kao i original

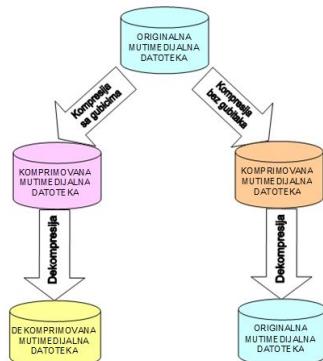
Veličine datoteka sa multimedijalnim sadržajem su uobičajeno velike. Ovo ih čini nepodesnim za čuvanje, a pogotovo za prenos preko interneta. Još veći problem se javlja kada je potrebno emitovati multimedijalni sadržaj, jer bi onda vreme prenosa moralo da bude kraće od vremena emitovanja.

Zbog toga se vrši kompresija originalnih multimedijalnih datoteka. U principu postoje dve vrste kompresija: bez gubitaka podataka i sa gubitkom podataka. U slučaju kompresije bez gubitka podataka, nakon dekompresije se dobija originalna datoteka. U slučaju kompresije sa gubicima, dekompresovana datoteka nije ista sa originalom, a kvalitet sadržaja je po pravilu degradiran.

Algoritam kojim se vrši kompresija i dekompresija naziva se kodek (codec - Compres - Decompres). Za kodiranje sadržaja se koristi koder, a za dekompresiju se koristi plejer. Prema tome, ukoliko korisnik ne poseduje adekvatni kodek, on nije u mogućnosti da čita i interpretira komprimovane multimedijalne datoteke.

Audio i video zapisi su poznati po tome što postoje delovi zapisa koji su vrlo dinamični i delovi zapisa koji su statički. Na primer, film dobijen naglim zakretanjem kamere je dinamičke prirode, jer se svaki kadar razlikuje od prethodnog.

Međutim, ako kamera snima, na primer, neku zgradu bez pomeranja, dobija se statički zapis, što znači da će više frejmova imati isti sadržaj. Zbog toga se prilikom kompresije može primeniti metoda promenljivog broja bitova (VBR - variable bit rate) pomoću koje je moguće dinamičke scene kodirati sa više podataka u jedinici vremena, a statičke sa manje podataka. Na taj način se postiže veći stepen kompresije uz isti kvalitet. Za razliku od VBR metode, postoji metoda sa konstantnim brojem bitova (CBR - constant bit rate) u jedinici vremena kod koje se kodiranje vrši konstantno i nezavisno od dinamike sadržaja. Ova metoda se koristi kada je ograničena brzina prenosa podataka. Kvalitet dekodiranog sadržaja CBR metode je po pravilu gori od VBR metode, pogotovo kod delova zapisa koji imaju izrazitu dinamičnost.



Slika 2.1.1 Proces kompresije i dekompresije [Izvor: Autor]

✓ 2.1 KOMPRESIJA DIGITALNIH SLIKA

NA KOJI NAČIN SE VRŠI KOMPRESIJA SLIKA?

Za kompresiju digitalnih slika se koriste metode prostornog poduzorkovanja, prediktivnog kodiranja, vektorske kvantizacije, fraktalnih slika i transformacionog kodiranja

Kompresija digitalnih slika je bazirana na redundantnosti podataka koji opisuju jednu sliku i na činjenici da je ljudska percepcija ograničena. Kod digitalnih slika, susedni pikseli često imaju iste karakteristike boje, pa su opisani istim podacima. Ovo se naziva prostorna redundantnost. Prostorna redundantnost se može otkloniti korišćenjem metoda prediktivnog kodiranja i transformacionog kodiranja.

Kada se radi o slikama, ljudi mogu tolerisati male greške u podacima vezanim za sliku. Ove greške ne utiču na prepoznavanje objekata niti na komunikaciju koja treba da bude postignuta slikom. Ovo znači da dekomprimovana slika ne mora potpuno da odgovara originalu.

Za kompresiju digitalnih slika se koriste metode prostornog poduzorkovanja, prediktivnog kodiranja, vektorske kvantizacije, fraktalnih slika i transformacionog kodiranja. Ovde će biti objašnjena samo metoda prostornog poduzorkovanja.

Metod prostornog poduzorkovanja je jednostavna metoda kompresije slika koja polazi od činjenice da pri kompresiji nije neophodno kodirati svaki piksel iz originalne slike. Umesto toga se, na primer, zapisuju podaci o svakom četvrtom ili svakom 8 pikselu.

Prilikom dekompresije, podaci o nedostajućim pikselima se određuju interpolacijom. Ova metoda daje relativno male stepene kompresije. Na primer, ako se zapišu podaci o svakom četvrtom pikselu, stepen kompresije će biti 4:1.

JPEG (Joint Photographic Experts Group) je prvi ISO standard za kompresiju slika različitog nivoa sivog i kolor slika. Međutim isti metod se koristi i za kompresiju videa. Ovaj standard je zasnovan na kombinaciji više matematičkih metoda za kompresiju. JPEG specificira četiri načina rada:

- Kompresiju sa gubicima zasnovanu na diskretnoj kosinusnoj transformaciji,
- Proširenu kompresiju sa gubicima, koja je takođe zasnovana na diskretnoj kosinusnoj transformaciji,
- Kompresiju bez gubitaka i
- Hiperarhijsko kodiranje, kod koga se svaka slika kodira sa više rezolucija.
- Popularnost JPEG standarda je u tome što se i pri kompresijama 10:1 do 20:1 vrlo teško mogu uočiti razlike, mada je moguće ostvariti i kompresije od 100:1 kada su razlike uočljive.

KOMPRESIJA SLIKA - VIDEO

How Image Compression Works

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 2.2 KOMPRESIJA ZVUKA

KOMPRESIJE SA GUBICIMA I BEZ GUBITAKA

Rengenske snimke, telemedicinu i slične aplikacije je potrebno koristiti ili kompresiju bez gubitaka ili preneti ovakve podatke bez ikakve kompresije. Veličina i kvalitet multimedijalnog sa

Komunikaciona mreža mora da ima sposobnost da obradi veliku količinu multimedijalnog sadržaja, a da pri tom bude pravedna i prema drugim tipovima saobraćaja. Tabela 1 prikazuje koliko opsega zahtevaju neke od češće korišćenih multimedijalnih aplikacija. Iako smo spomenuli da je potrebno da se za prenos multimedija koriste razne kompresije, nije moguće komprimovati sve prenose multimedija. Postoji dva tipa kompresija:

1. **kompresija sa gubicima** – tehnika koja podrazumeva da se informacije sa redundantnim podacima eliminišu
2. **kompresija bez gubitaka** – tehnika u kojoj se ne gube bilo kakve informacije, a podaci stižu na odredište isto kao što su i poslati.

Kompresija sa gubicima postiže veću kompresiju podataka, međutim ova metoda nije prikladna za sve tipove multimedija ili aplikacija.

KOMPRESIJA ZVUKA BEZ GUBITAKA

Ova metoda se uglavnom koristi unutar aplikacija za obradu zvuka, za čuvanje audio zapisa, a u mnogo manjem obimu za prenos zvuka preko Interneta

Kompresija audio podataka se vrši u cilju smanjivanja veličine audio zapisa. Algoritmi kompresionih algoritama se nazivaju audio codec-i. Za kompresiju audio podataka se koriste algoritmi bez gubitaka i sa gubitkom podataka.

Kompresija zvuka bez gubitaka omogućuje odnos kompresije do 2:1. Veći odnos kompresije se ne može dobiti jer zvučni talasi imaju vrlo brze slučajne promene, pa je vrlo teško pronaći ponavljanje istog niza bitova, što se tipično koristi kod generičkih metoda kompresije. Nakon dekompresije podataka, audio zapis ima potpuno istu formu koju je imao pre kompresije. Ova metoda se uglavnom koristi unutar aplikacija za obradu zvuka, za čuvanje audio zapisa, a u mnogo manjem obimu za prenos zvuka preko Interneta. Međutim, kako je cena i veličina memorijskog prostora sve manji problem, a brzina prenosa podataka preko Interneta postaje sve veća i veća, algoritmi za kompresiju zvuka bez gubitaka postaju popularniji. Jedan od popularnijih formata audio zapisa sa kompresijom bez gubitaka je FLAC, a u upotrebi su i Shorten i TTA. Microsoft je razvio svoj metod za komprimovanje audio zapisa bez gubitaka Windows Media Audio Lossless (WMAL). Ovim metodom moguće je postići stepen kompresije 2:1 do 3:1 zavisno od sadržaja.

KOMPRESIJA ZVUKA SA GUBICIMA

Kompresija zvuka sa gubicima se zasniva na psihoaustičnom modelu

Kompresija zvuka sa gubicima se zasniva na psihoaustičnom modelu. Poznato je naime da čovek ne može da čuje zvuke slabog intenziteta paralelno sa zvukom jakog intenziteta. Ova pojava se naziva maskiranje. Ljudi takođe ne mogu da čuju ni zvuke izuzetno visoke frekvencije (preko 20 kHz). Zbog toga se takvi podaci ne zapisuju čime se vrši kompresija. Audio zapis koji se dobije posle dekompresije nije isti kao pre kompresije, ali su gubici u kvalitetu zanemarljivi i za čoveka neprimetni. Kompresija zvuka sa gubicima se koristi i kod DVD, digitalne televizije, antenskog i kablovskog radija. U odnosu na kompresiju bez gubitaka, kompresija sa gubicima ima veću kompresiju podataka 20%. Da bi se odredilo koje su informacije u audio zapisu perceptualno irrelevantne, većina kompresionih algoritama sa gubicima koristi transformacije, kao što je modifikovana diskretna kosinusna transformacija (**MDCT - modified discrete cosine transform**), da prevede uzorke zvučnih talasa iz vremenskog domena u frekventni domen. Svakoj komponenti frekvencije se dodeljuje odgovarajuća jačina zvuka. Čujnost spektralnih komponenata se određuje proračunavanjem praga maskiranja (engl. **masking threshold**). Zvuk koji je ispod ovog praga se ne zapisuje prilikom kompresije.

MPEG

MPEG audio je u stvari familija od tri audio kompresionih šema, koje se nazivaju MPEG Audio layer 1, Layer 2 i Layer 3

Standard za kompresiju zvuka **MPEG Audio** je generički standard koji kompresuje zvuk bez obzira na njegov karakter, tj. namenjen je i za muziku i za govor i za druge zvukove. MPEG audio koristi prag maskiranja da bi odbacio sve zvuke koje čovek inače ne bi čuo. Zbog toga se kaže da je MPEG Audio kompresija perceptualno bez gubitaka, mada se posle dekompresije ipak ne dobija originalni zapis.

MPEG Audio dozvoljava uzorkovanje zvuka frekvencijom od 32 kHz, 44,1 kHz i 48 kHz. Kompresovani strim može da ima jedan (mono) ili dva (stereo) kanala. Zavisno od frekvencije uzorkovanja, moguće je postići kompresiju od 2,7:1 do 24:1. Testiranja su pokazala da ni najbolji audio eksperti ne mogu da primete razliku pri MPEG Audio kompresiji od do 6:1.

MPEG audio je u stvari familija od tri audio kompresionih šema, koje se nazivaju MPEG Audio layer 1, Layer 2 i Layer 3. Kompleksnost algoritma i stepen kompresije raste sa brojem lejera.

Najpoznatiji audio format sa kompresijom zvuka sa gubicima je MP3, koji omogućuje odnos kompresije od 12:1. MP3 (**MPEG Audio Layer 3**) je tehnički gledano MPEG-1 Layer 3 datoteka. MPEG-1 je kompresiona tehnologija koja je deo MPEG-1 multimedijalnog standarda kreiranog od strane Moving Picture Experts Group. MPEG-1 standard podržava tri tipa informacija video, audio i film kao sinhronizovanu kombinaciju videa i audija.

Popularnost MP3 formata leži u činjenici da nakon kompresije daje izvanredan kvalitet zvuka i vrlo male datoteke sa zapisom. Na primer, jedan minut visoko kvalitetnog WAV zapis zahteva 40 MB prostora, a MP3 format 3,5 MB. Zbog kompresionog odnosa 12:1 MP3 zapis se vrlo često koriste na vebu za publikovanje audio informacija, a posebno muzike. MP3 zapis se može reprodukovati softverski i hardverski, preko takozvanih MP3 plejera. MP3 zapis se dobija tako što se originalni audio zapis u WAV ili AIFF formatu konvertuje u MP3 format korišćenjem nekog od brojnih konvertora (na primer Xing AudioCatalyst ili iTunes). Postoje i drugi audio formati koji koriste kompresiju sa gubicima, kao što su Vorbis i Windows Media Audio.

MICROSOFT KODECI SA GUBICIMA

Microsoft nudi dva audio kodeka sa gubicima: Windows Media Audio i Windows Media Audio Profesional

Microsoft nudi dva audio kodeka sa gubicima: **Windows Media Audio** i **Windows Media Audio Professional**. Windows Media Audio uzorkuje zvuk pri 44,1 ili 48 kHz koristeći 16 bita za zapis podataka, i na taj način obezbeđuje kvalitet zvuka u klasi CD-a. Potrebna brzina prenosa za emitovanje ovako kodiranog zvuka je 64 - 192 Kb/s. Za uzorkovanje se koristi VBR metoda.

Windows Media Audio Profesional radi sa frekvencijom od 96 kHz i koristi 24 bita za zapis podataka o zvuku. Namjenjen je najzahtevnijim aplikacijama i aplikacijama koje podržavaju

stereo zvuk, ili zvuk sa sistemom 5.1 kanala i 7.1 kanala (surround sound). Očekivana brzina prenosa podataka je od 128 -768 kb/s.

Za kompresiju govora se koriste posebne metode kao što su: nelinearna kvantizacija i linearno prediktivno kodiranje. U ovom slučaju se koristi činjenica da je frekventni opseg ljudskog glasa mnogo uži nego kod muzike, a sam zvuk mnogo manje kompleksan.

✓ 2.3 KOMPRESIJA VIDEA

KOMPRESIJE VIDEA I STANDARDI

Za kompresiju videa se koristi prostorna redundansa, kao i percepcione mogućnosti čoveka

Kao i kod slika, za kompresovanje videa se koristi činjenica da postoji prostorna redundansa u podacima koji opisuju pojedine frejmove videa, kao i činjenica da su percepcione mogućnosti čoveka ograničene. Za razliku od pojedinačnih slika, ovde se još prilikom kompresije eksploratiše i činjenica da su uzastopni frejmovi vrlo slični, a nekada i potpuno isti. Ovo se naziva vremenska redundansa. Metode koje se koriste za kompresiju videa, prema tome, su iste kao i one koje se koriste za kompresiju slika, ali su još obogaćene metodom određivanja kretanja.

Metod određivanja kretanja se zasniva na vremenskoj redundantnosti. To znači da je grupa piksela koja odgovara nekom detalju slike na jednom frejmu na različitoj poziciji na narednom frejmu. Prilikom stvarne kompresije, slika frejma se deli na blokove konstantne veličine. Zatim se analiziraju slike dva uzastopna frejma i određuje se pomeranje bloka. Poziciono pomeranje bloka se naziva vektor kretanja. Zatim se određuje razlika u pikselima originalnog i pomerenog bloka. Razlika u pikselima je vrlo mala od frejma do frejma, pa je mnogo efikasnije zapisivati razliku u pikselima nego podatke o celom bloku. Prilikom kodiranja se za svaki frejm zapisuje vektor kretanja i razlike u pikselima.

Za kodiranje videa je razvijena serija MPEG standarda. **Moving Pictures Expert Group** (MPEG) je formirana 1988 kao telo ISO/IEC tehničkog komiteta za informacione tehnologije sa zadatkom da razvije standarde za kompresovanu reprezentaciju digitalnog videa i pridruženog zvuka.

Cilj je bio da se razviju tri standarda:

- MPEG-1 za kodiranje videa VHS kvaliteta u rezoluciji 360 x 280 piksela i 30 slika u sekundi, pri brzini od 1,5 Mb/s, što je u to doba odgovaralo brzini čitanja CD-ROM-ova.
- MPEG-2 za kodiranje videa u kvalitetu digitalne televizije u rezoluciji 720 x 480 piksela i 30 slika u sekundi, pri brzini od 2 do 10 Mb/s.
- MPEG-3 za kodiranje videa u kvalitetu HDTV (high definition TV) pri brzini oko 40 Mb/s. Rad na ovom standardu je prekinut 1992. jer je ustanovljeno da se ciljevi mogu postići MPEG-2 tehnologijom.

Tokom rada na standardima MPEG-1 i 2 su se uočile dodatne potrebe, pa je radna grupa predložila izradu novog standarda MPEG-4 koji će omogućiti efikasno memorisanje, prenos i manipulaciju sa multimedijalnim podacima. Glavna osobina MPEG-4 da on omogućuje rad sa audio-video objektima (AVO) tretirajući ih nezavisno i dozvoljavajući njihovu kombinaciju na nekoj sceni.

KOMPRESIJA VIDEA - VIDEO

How Video Compression Works

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

STRIMOVANJE

PROTOKOLI KOJI UČESTVUJU U PROCESU STRIMOVANJA

Strimovanje je proces kontinualnog prenosa podataka, uobičajeno multimedijalnih, prilikom čega klijent interpretira pristigle pakete brzinom kojom oni dolaze

Strimovanje je proces kontinualnog prenosa podataka, uobičajeno multimedijalnih. Audio i video podaci koji se prenose su kompresovani na neki način. Server deli multimedijalnu datoteku na paket i šalje ih klijentu. Dekodiranje i interpretacija ovih podataka se na odredišnom serveru vrši u realnom vremenu, a podaci se ne memorišu. Serija povezanih paketa se naziva strim (stream - tok). Strimovanje se razlikuje od običnog prenosa datoteka u tome što klijent ne čeka da se kompletna datoteka prenese da bi je interpretirao. Umesto toga klijent interpretira pakete brzinom kojom oni dolaze, a nakon toga ih odbacuje.

Strimovanje se obavlja preko dve vrste protokola: UDP i TCP.

User Datagram Protocol (UDP) prenosi multimedijalni strim kao seriju malih paketa na najbrži mogući način. Ovaj protokol je jednostavan i efikasan, ali ne obezbeđuje ponovno slanje izgubljenih ili oštećenih paketa prilikom transporta.

Protokol TCP je pouzdan, jer obezbeđuje sve pakete u ispravnom stanju, ali na uštrb vremena. U nekim slučajevima njegova brzina ne može da obezbedi minimalnu brzinu potrebnu za kontinuiranu interpretaciju strima na klijentskoj strani. Zbog toga se primenjuje tehnika bafera, koja obezbeđuje određenu rezervu vremena koja može biti potrošena onda kada neki paketi kasne.

Specijalno za potrebe strimovanja su razvijeni Real-time Transport Protocol (RTP), Real Time Streaming Protocol (RTSP) i Real Time Control Protocol (RTCP).

Strimovanje u realnom vremenu koristi RTP protokol. Multimedijalni paketi se šalju u relanom vremenu, što znači da se jedan minut filma, recimo, šalje jedan minut. Svaki paket ima vremensku oznaku, tako da se prikazuje sinhronizovano. Strimovanje u realnom vremenu može biti jedan - prema - jedan (unicast) i jedan - prema - više (multicast).

UNIKAST

Prilikom unikasta server svakom klijentu šalje poseban strim

Kod unikast strimovanja klijent šalje serveru zahtev koristeći **RTSP** (Real Time Streaming Protocol). Server preko istog protokola odgovara klijentu šaljući mu informacije koje opisuju strim sesiju. Jedna strim sesija može da se sastoji od jednog ili više strimova podataka, kao što su na primer audio i video podaci. Server obaveštava klijenta koliko strimova treba da očekuje, koji je tip medija u pitanju i koji je kodek korišćen. Nakon toga se šalje strim preko **RTP** protokola.



Slika 3.1 Osnovna arhitektura prilikom strimovanja [1]

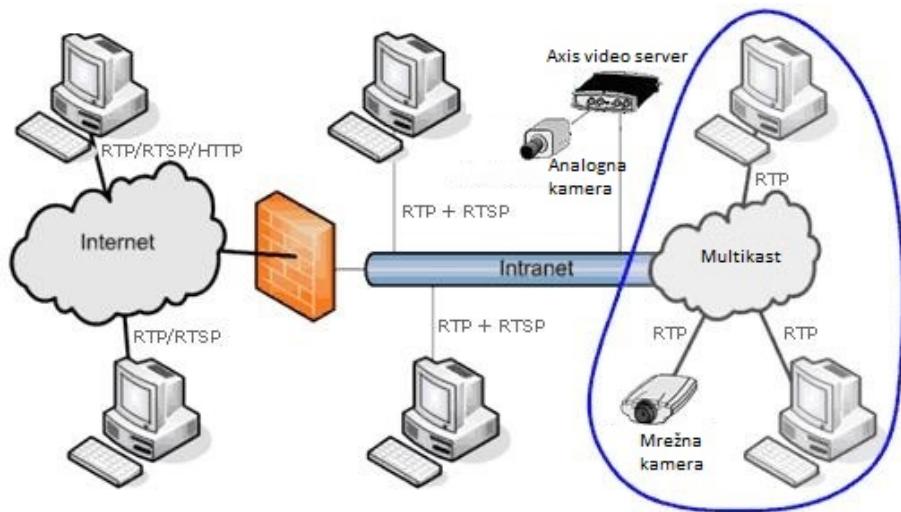
[Izvor: <http://mirror.informatimago.com/next/developer.apple.com/documentation/QuickTime/REF/Streaming-1.gif>]

Prilikom unikasta server svakom klijentu šalje poseban strim. Dok klijent prima multimedijalni strim, u mogućnosti je da po sopstvenom izboru skoči na željeni deo multimedijalnog zapisa, ne čekajući da se pročitaju svi paketi. Time klijent praktično zahteva od servera da nastavi strimovanje od neke željene tačke u vremenu.

MULTIKAST

Pri multikast strimovanju server šalje jedan strim do više klijenata različitim putevima mreže

Pri multikast strimovanju server šalje jedan strim do više klijenata različitim putevima mreže. Ovim se redukuje mrežni saobraćaj potreban da se pošalju strimovi velikom broju klijenata. Svi klijenti primaju strim tako što se pridružuju multikastu. Klijent saznaće kako da se priključi multikastu čitanjem **SDP** (Session Description Protocol) datoteke.



Slika 3.2 Multikast strimovanje [2]

[Izvor: <https://netcam.cz/encyklopedie-ip-zabezpeceni/img/mpeg4-01.gif>]

Poznato je da od 2005. godine neki ruteri ne podržavaju multikast i da ga mnogi fajervolovi blokiraju. Da bi klijenti koji su iza rutera koji ne podržavaju multikast mogli da se pridruže multikast uvode se reflektori. Reflektor je RTSP server koji se priključuje multikastu i onda konvertuje multikast u seriju unikasta, šaljući strimove pojedinim klijentima.

Kada se klijent pridruži multikastu, korisnik nije u mogućnosti da se kreće napred i nazad kroz multimedijalni zapis, jer je strim isti za sve pridružene klijente. U ovom slučaju klijent može samo da napravi pauzu ili prekine strimovanje.

Najpoznatiji serveri za strimovanje zvuka su *RealAudio*, *Windows Media*, *Liquid Audio* itd. Za strimovanje videa se koriste *QuickTime*, *Windows Media*, *Real Media* i drugi serveri.

UNIKAST I MULTIKAST UPOREĐENJE - VIDEO

UNICAST MULTICAST BROADCAST

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 4

RAZVOJ KORISNIČKOG VEB INTERFEJSA

PROJEKTOVANJE I PROBLEMI

Dobar veb interfejs je pitanje dobrog balansiranja strukture i veza menija, sadržaja i drugih povezanih dokumenata i multimedijalnih sadržaja

Projektovanje korisničkog veb interfejsa se odnosi na projektovanje navigacije i prezentacije informacija jednog veb sajta. Dobar veb interfejs je pitanje dobrog balansiranja strukture i veza menija, sadržaja i drugih povezanih dokumenata i multimedijalnih sadržaja.

Prilikom razvoja korisničkog veb interfejsa se javljaju mnogi problemi.

- Jezik za opis prikaza veb stranica, HTML, nije bio namenjen za kreiranje stranica koje će koristiti opšta populacija. Očekivalo se da će takve stranice uglavnom koristiti visoko obrazovani ljudi sa dobrom tehničkom osnovom. HTML je ograničen brojem objekata i stilovima interakcije, pa tako ne obezbeđuje metode za prezentovanje informacija na najefektivniji način.
- Mogućnosti navigacije su vrlo skromne. Većina navigacionih mehanizama je ograničena na koncept „Napred“ i „Nazad“, pri čemu korisnik često ne zna šta je napred, a šta nazad.
- Projektovanje korisničkog veb interfejsa zavisi od arhitekture informacija i od toka zadataka. Međutim, nijedan od ova dva koncepata nije standardizovan.
- Nema precizne definicije korisnika veb sajta. Oni koriste različite hardverske i softverske alate da pristupaju veb stranama. Korisnici imaju različite intelektualne, senzorne, motorne i socijalne karakteristike.
- Korisnici pristupaju veb sajtovima iz različitih okruženja. U nekim slučajevima osvetljenje i buka smetaju korisnicima da ispravno prime informacije. Nekada su korisnici u pokretu ili obavljaju paralelno druge poslove, pa ne mogu da koriste sve motorne funkcije za navigaciju.

RAZLIKE IZMEĐU GUI I VEB KORISNIČKOG INTERFEJSA

Razlike leže u korisničkom hardveru, izvoru podataka i informacija, zadacima i fokusu korisnika, vizuelnog stila, navigacije, prezenatcionih elemenata i vremena odziva

Projektanti veb korisničkog interfejsa su preuzeli metode i iskustva od projektanata grafičkog korisničkog interfejsa (GUI). Iako ima nekih sličnosti, ogroman je broj razlika između GUI i veb korisničkog interfejsa. U sledećoj tabeli date su samo neke razlike

	GUI	WEB
Uređaji	Korisnički hardver je poznat i sa definisanim karakteristikama	Ogromne varijacije u hardveru. Prikaz na ekranu zavisan od hardvera
Fokus korisnika	Podaci i aplikacija	Informacije i navigacija
Podaci/ Informacije	Tipično kreirani i korišćeni od strane poznatih i pouzdanih korisnika. Poznate karakteristike. Organizovani na razumljiv način. Podeljeni na privatne i	Nepoznat sadržaj. Izvor podataka nije uvek pouzdan. Nije uvek poznat autor podataka. Promenljiva organizacija podataka.
Zadatak korisnika	Instalacija, konfiguracija, personalizacija korišćenje i ažuriranje programa Otvaranje, korišćenje i zatvaranje datoteka. Dugo vreme provođenja u aplikaciji.	Pristupanje sajtu, čitanje strana, popunjavanje formi, registracija, učešće u transakcijama, prevlačenje datoteka i zapisivanje. Promena strana vrlo česta. Nepoznavanje sajta kome se
Prezentacioni elementi	Prozori, meniji, kontrole, podaci, poruke, itd.	Dve komponente: čitač i veb strana. Unutar strane: tekst, slike, video, audio i animacije.
Navigacija	Meni, liste, stabla, dijalog,	Linkovi, favorit i URL brojevi.
Vreme odziva	Trenutno	Nedefinisano
Vizualni stil	Definisano razvojnim alatom	Individualan, nedefinisani, često umetnički

Slika 4.1.1 Tabela-1 Razlike između GUI i veb korisničkog interfejsa [Izvor: Autor]

✓ 4.1 ARHITEKTURA INFORMACIJA NA VEBU

INFORMACIONA STRUKTURA VEB SAJTA

Postoje najmanje dve informacione strukture potrebne za razvoj veb sajta, a to su informaciona arhitektura, poznatija kao mapa sajta, i mapa veb stranice

Veb stranice su bazirane na hipertekstu. **Hipertekst** je dokument u kome se pojedini pojmovi iz dokumenta referenciraju na druge dokumente. Referenciranje se vrši korišćenjem URI-ja. **Hipermedija** je pojam kojim se opisuje korišćenje teksta, slike, audio i video elemenata u hipertekst dokumentu. Sve ove različite forme informacija su međusobno povezane tako da korisnik lako može da prelazi sa jednih na druge ili da ih istovremeno kombinuje. Drugim rečima, razlog za primenu hiperteksta i hipermedija je da se postigne efektivna komunikacija sa korisnikom.

Postoje najmanje dve informacione strukture potrebne za razvoj veb sajta, a to su informaciona arhitektura, poznatija kao mapa sajta, i mapa veb stranice.

EFEKTIVNA KOMUNIKACIJA

Efektivna komunikacija na vebu se može postići dobrim formatiranjem sadržaja, adekvatnim korisničkim interfejsom i navigacionom šemom, kao i adekvatnim multimedijalnim sadržajem

Jedan od osnovnih zahteva koji se postavlja pri projektovanju svake veb strane je da ostvari efektivnu komunikaciju sa korisnikom. Veb stranice koje ne obezbeđuju efektivnu komunikaciju nemaju dovoljan broj posetilaca, a time se gubi i njihova osnovna funkcija.

Veb sajtovi se projektuju za različite namene, kao što su prezentacija organizacije, publikovanje vesti, zabava, edukacija itd. Svaki od ovih tipova sajtova nameće potrebu za specifičnim oblicima komunikacije. Pored toga, pojedine grupe korisnika veb sajtova imaju imaju različite afinitete i potrebe kada je u pitanju komunikacija sa sajtom.

Generalno gledano, može se reći da se efektivna komunikacija postiže:

- Dobrim formatiranjem sadržaja veb strane,
- Izborom i implementacijom adekvatnog korisničkog interfejsa
- Projektovanjem i implementacijom efikasne navigacione šeme
- Korišćenjem različitih tipova sadržaja, kao što su slike, audio i video objekti.

FORMATIRANJE SADRŽAJA VEB STRANE

Za formatiranje sadržava veb strane se mogu koristiti: tipografski efekti, raspored sadržaja, efekti pozadine

Nalaženje i čitanje informacija na nekoj veb strani u dobroj meri zavisi od dizajna stranice i od načina formatiranja teksta. Formatiranje veb strane se postiže metodama koje se mogu svrstati u tri grupe:

- Tipografski efekti utiču na kvalitet i izgled teksta. Tipografski efekti se postižu izborom tipa fonta, veličine fonta, naglašavanjem fonta, prostorom između karaktera, prostorom između redova i kontrastom.
- Raspored ćelija sa sadržajem doprinosi opštoj preglednosti strane. Raspored se postiže korišćenjem margina, kolona, tabele, banera, zaglavlja i sličnih ćelija.
- Efekti pozadine se koriste u pozadini i oko teksta i drugih elemenata.

Autori veb strana mogu postići emocionalne i psihološke efekte ako posebno obrate pažnju na sledeće smernice:

- Naglašavanje značaja
- Interes
- Čitljivost.

SMERNICE ZA FORMATIRANJE SADRŽAJA VEB STRANE

Za efikasnu veb stranicu treba обратити пажњу на наглашавање знаčaja, интересу и читљивости

Naglašavanje značaja. Veličina, pozicija i boja teksta upravlja pogledom korisnika. Ovi atributi takođe emituju mogu informacije o relativnom značaju teksta na koga su primenjeni. Zato je potrebno korišćenjem ovih atributa posebno naglasiti one delove teksta kojim autor želi da naglasi značaj. Na primer, nazivi proizvoda na nekoj veb strani koja je deo veb kataloga treba da budu posebno istaknuti. Kontra efekt se može postići ako na strani postoji previše naglašenih elemenata, jer čitalac ne može zbog obilja elemenata da napravi relativnu razliku u značaju.

Čitljivost. Veliki blokovi teksta su teži za čitanje od malih blokova. Zbog toga tekst treba razbiti na male celine koje su odvojene belinom, okvirom ili nekim grafičkim elementom. Segmentiranje teksta se može izvesti i tipografskim elementima kao što su numeracija ili buliti.

Drugi faktor koji utiče na čitljivost je **veličina fonta** kojom je ispisan tekst. Ako je veličina fonta fiksirana, korisnik neće moći da poveća veličinu fonta za prikaz u svom čitaču. Treba imati na umu da neki čitaoci nemaju dobar vid i da onda neće moći da pročitaju tekst.

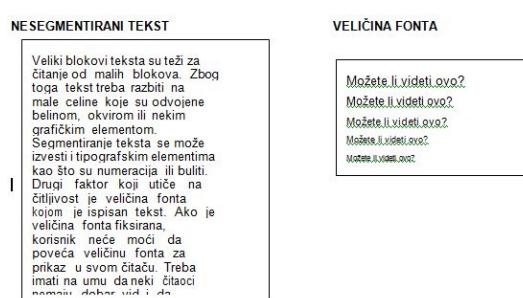
Tip fonta je sledeći faktor koji utiče na čitljivost. Poznato je da se jednostavniji fontovi najlakše čitaju. Fontovi sa serifima i italic fontovi imaju otežano čitanje, a najteže se čitaju dekorativni fontovi i tekst sa redukovanim prostorom između karaktera.

Na čitljivost takođe utiče **kontrast** između boje pozadine i boje teksta. U nekim sličajevima, slab kontrast može da dovede do loše čitljivosti.

Interes. Veliki blokovi teksta, pored teškoća u čitanju, smanjuju interes čitaoca. Segmentirani tekst povećava interes za primanjem novih informacija.

PRIMERI

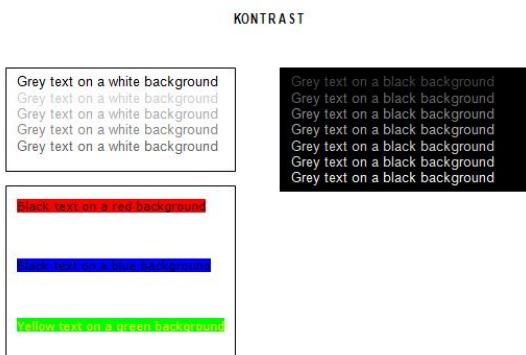
Primeri za primenu različitih načina struktuiranja i naglašavanja teksta na veb strani



Slika 4.2.1 Primer nesegmentiranog teksta i primene različitih veličina fontova [Izvor: Autor]



Slika 4.2.2 Primer segmentiranog teksta i primene različitih tipova fontova [Izvor: Autor]



Slika 4.2.3 Primeri kontrasta [Izvor: Autor]

▼ 4.2 INTERFEJS VEB SAJTA

KARAKTERISTIKE KOJE TREBA DA IMA INTERFEJS VEB SAJTA

Interfejs veb sajta uključuje četiri različite konstrukcije: upotrebljivost, vizuelizaciju, funkcionalnost i dostupnost

Interfejs veb sajta uključuje četiri različite konstrukcije: upotrebljivost, vizuelizaciju, funkcionalnost i dostupnost.

Upotrebljivost se definiše kao intuitivnost i lakoća u navigaciji kroz sajt. Pored toga upotrebljivost zavisi i od brzine učitavanja stranice, mogućnosti štampanja, i drugih faktora. Vizuelizacija se odnosi na kreiranje interesantnih i estetskih sadržaja.

Funkcionalnost veb sajta se postiže obezbeđivanjem različitih funkcija koje će korisniku olakšati korišćenje sajta. Neki primeri funkcionalnosti su pretraživanje, personalizacija, glasanje i često postavljana pitanja. Tipičan primer funkcionalnosti je mogućnost da korisnik prilikom prikazivanja vremenske prognoze izabere prikaz temperature u C ili F. Izbor prikaza promene vrednosti akcija na berzi neke kompanije na dnevnom, nedeljnem ili mesečnom nivou je drugi primer funkcionalnosti. Funkcionalnost može da se ilustruje i na primeru veb formi za unos i slanje podataka. Veb strana koja omogućuje proveru validnosti unetih

podataka na strani klijenta svakako je funkcionalnija od veb strane koja to radi na serverskoj strani.

Osobe sa posebnim potrebama očekuju od interfejsa veb sajtova alternativne načine za pristup sadržaju. Povećanje **dostupnosti** se postiže uvođenjem promenljive veličine fontova, dodavanjem labela i naslova svim elementima sadržaja, umetanjem audio i video objekata itd.

▼ 4.3 NAVIGACIONE ŠEME

METODE ZA NAVIGACIJU

Veb sajt treba da ima različite navigacione šeme kako bi odgovorio različitim vrstama korisnika.

Različiti korisnici imaju potrebu da koriste različite metode navigacije kroz veb sajtove kako bi došli do informacija koje traže. U nekim slučajevima korisnik ne ulazi na sajt preko naslovne strane, nego preko neke druge koju je našao na pretraživaču. Motivacija korisnika za kretanjem kroz sajt može biti potreba da se dođe do neke tačno određene informacije ili servisa ili samo želja da se sazna nešto više. Veb sajt treba da ima različite navigacione šeme kako bi odgovorio različitim vrstama korisnika.

Za navigaciju se mogu koristiti različiti alati kao što su: horizontalni i bočni navigacioni meniji, prikaz linkova sa sličnim sadržajem, search, next page, back itd. Korisnik očekuje da ih nađe na svakoj strani veb sajta i da oni rade na konzistentan način.

Većina veb sajtova nudi više metoda za navigaciju. Ipak mogu se prepoznati tipska rešenja koja se bliže opisuju u daljem tekstu.

Metode za navigaciju mogu biti:

- Hijerarhijska organizacija
- Organizacija bazirana na zadatku
- Alfabetska organizacija
- Hronološka organizacija
- Organizacija bazirana na popularnosti.

OPISI METODA ZA NAVIGACIJU

Hijerarhijska organizacija, organizacija bazirana na zadatku i popularnosti, alfabetska i hronološka organizacija

Hijerarhijska organizacija. Ovaj metod polazi od dobre i sveobuhvatne organizacije informacija po hijerarhijskom principu. Na vrhu hijerarhije su generalne kategorije. Svaka kategorija se dalje deli na određeni broj podkategorija u nekoliko nivoa. Na dnu hijerarhijskog

stabla su veb strane sa informacijama. Tipičan sajt koji primenjuje ovu metodu je Yahoo, koji na naslovnoj strani ima kategorije kao što su: Umetnost, Biznis, Računari, Edukacija itd. Svaka od ovih kategorija ima svoje podkategorije. Prilikom definisanja kategorija treba pratiti način na koji korisnici razmišljaju. Za imena kategorija je potrebno odabrati termine koji su dovoljno opisni i ostvaruju jasnu razliku u odnosu na druge kategorije. Broj podkategorija neke kategorije treba da bude između 20 i 50. Preveliki broj kategorija otežava i usporava nalaženje željene kategorije.

Organizacija bazirana na zadatku. Mnogi sajтови pružaju određene informacione usluge korisnicima. Posetioci pristupaju sajtu da bi obavili neke zadatke kao što su, rezervacija i nabavka avionskih karata, planiranje automobilskih ruta od mesta A do mesta B, nalaženje i kupovina digitalnih sadržaja itd. U ovakvim slučajevima treba proučiti logički redosled kojim korisnici izvršavaju neki zadatak. U takvim slučajevima, kompletiranje jednog zadatka treba da dovede do otvaranja nove stranice koja omogućuje rešavanje narednog zadatka.

Alfabetska organizacija. Ova organizacija se često koristi kada su korisnicima poznate tačne reči ili fraze o kojoj traže informacije. U odnosu na hijerarhijsku organizaciju koja može da ima više nivoa, ova metoda omogućuje brži pristup do informacija.

Hronološka organizacija. Ova organizacija omogućuje korisnicima da pronađu sadržaj čiji je nastanak ili publikovanje povezano sa vremenom. Tipičan primer ovakvih sajtova su sajтовi novinskih agencija, koje iz minuta u minut, iz dana u dan publikuju vesti.

Organizacija bazirana na popularnosti. Neki korisnici prilikom pristupanja na sajt žele da vide koji su sadržaji ili proizvodi najpopularniji. Ovo je uobičajena organizacija za sajtove koji prodaju digitalne sadržaje kao što su muzika i filmovi, kao i kod sajtova koji prodaju literaturu. Da bi se postigla ispravna organizacija posetiocima je potrebno pružiti i informaciju o tome na osnovu kojih kriterijuma je rangirana popularnost i za koji period. Određivanje liste popularnosti treba obavljati u kratkim vremenskim intervalima kako bi lista stalno reflektovala trenutno stanje.

▼ 4.4 KORISNIČKI ORIJENTISAN DIZAJN

NA ČEMU SE ZASNIVA KORISNIČKI ORIJENTISAN DIZAJN

Korisnički orijentisan veb dizajn se zasniva na poznavanju korisnika i uključivanju korisnika u proces projektovanja veb sajta

Pošto je smisao postojanja veb sajtova zadovoljenje korisničkih potreba, najčešće primenjivani metod projektovanja veb sajtova je **korisnički orijentisani dizajn** (user-driven design ili customer-centric design). Korisnički orijentisan veb dizajn se zasniva na dve kategorije informacija:

- Poznavanje korisnika i
- Uključivanje korisnika u proces projektovanja veb sajta.

Poznavanje korisnika veb sajta je neophodno da bi se došlo do rešenja koje će korisniku odgovarati. Pri tom se korisnik može posmatrati kao **individualna osoba** ili kao **generalni pripadnik ljudskog roda**. Poznavanje korisnikove individue znači imati detaljan profil tipičnog korisnika, što obuhvata: demografske karakteristike, obrazovanje, ponašanje, navike, znanje i potrebe.

Da bi se došlo do profila korisnika treba odgovoriti na mnoga pitanja. Ko su ljudi koji će posećivati sajt? Koliko su oni stari? Kakvo je njihovo obrazovanje? Koje druge sajtove često posećuju? Kakva im je Internet veza? Kojim jezikom govori? Koriste li stručnu terminologiju itd.

Projektanti veb sajta takođe treba da imaju i sledeće informacije o korisniku:

- Koje će zadatke korisnik obavljati na veb sajtu
- Na koji način veb sajt može da smanji uobičajeni rad korisnika ili mu omogući da postane bolji ekspert,
- Koje tehnologije su korisniku na raspolaganju
- Koja su socijalna i organizaciona ograničenja korisnika.

Druga kategorija potrebnih informacija se odnosi na generalne osobine ljudi. Za projektante veb sajtova važno je poznavanje osnovnih ljudskih fizičkih i kognitivnih sposobnosti. Važno je znati kako ljudski vizuelni, motorni i memorijski sistem funkcioniše.

I pored dobrog poznavanja korisnika, projektovani veb sajt ne mora da zadovolji potrebe tipičnog korisnika. Zato je potrebno uključiti korisnika u sam proces projektovanja i implementacije veb sajta kako bi se dobio trenutni korisnički odziv na rezultate projektovanja.

Sam proces projektovanja veb sajta je cikličan iterativan proces koji se sastoji od tri faze: projektovanje, izrada prototipa, evaluacija. Evaluaciju prototipa treba da vrši korisnik i eksperti. Cilj je da se u više iteracija dođe do rešenja kojim će korisnik biti zadovoljan. Na taj način se potencijalni problemi rešavaju u ranoj fazi projektovanja sajta. Pre početka projektovanja projektant treba da postavi poslovne ciljeve, razmotri potrebe korisnika i da definiše merljive ciljeve.

▼ Poglavlje 5

PITANJE DOSTUPNOSTI

WEB DOSTUPNOST

Veb dostupnost se definiše kao praksa projektovanja veb sajtova dostupnih svima, a posebno korisnicima sa posebnim potrebama

Ono što ne treba zaboraviti prilikom dizajniranja veb sajta je, da je krug mogućih posjetioca u najširem mogućem smislu reči različit. To znači da treba računati da će sajtu pristupati korisnici sa različitim fizičkim, mentalnim, socijalnim i intelektualnim karakteristikama. Neki od njih će u nekim domenima biti prosečni ili superiorni, ali se možda po nekim karakteristikama mogu svrstati u inferiorne. Tako se može dogoditi da delovi sadržaja koji su objavljeni, pa čak i ceo sajt ne bude dostupan nekim korisnicima. Zbog toga se pri projektovanju i implementaciji veb sajtova mora voditi računa o veb dostupnosti.

Veb dostupnost (engl. **web accessibility**) se definiše kao praksa projektovanja veb sajtova dostupnih svima, a posebno korisnicima sa posebnim potrebama.

U korisnike sa posebnim potrebama se, pre svega, svrstavaju:

- osobe sa oslabljenim vidom
- daltonisti
- slepi
- osobe sa oslabljenim sluhom
- gluvi
- osobe sa oslabljenim motornim karakteristikama, kao što su osobe koje imaju mišićnu distrofiju, celebralnu paralizu, Parkinsonovu bolest itd.
- osobe čiji maternji jezik nije onaj na kome su objavljene veb strane
- osobe koje koriste znakovni jezik
- osobe sa kognitivnim slabostima, kao što su osobe sa demencijom i disleksijom
- osobe koje imaju problema sa čitanjem i razumevanjem teksta
- osobe koje koriste tekstualne veb čitače, male ekrane i sporu Internet vezu
- osobe koje koriste stare verzije veb čitača, glasovne veb čitače ili koriste različite operativne sisteme.

INICIJATIVA ZA WEB DOSTUPNOST

Poštovanje smernica za dostupnost veb sadržaja čini veb sajt dostupnijim svim korisnicima, bez obzira koji veb čitač koriste

Imajući u vidu probleme osoba sa posebnim potrebama u vezi korišćenja informacija objavljenih na veb sajtovima Worl Wide Web konzorcijum (W3C) je 1999. godine pokrenuo inicijativu za veb dostupnost ([Web Accessibility Initiative - WAI](#)). U sklopu ove inicijative W3C je maja 1999. godine objavio smernice za dostupnost veb sadržaju ([Web Content Accessibility Guidelines 1.0](#)). Novembra 2005. godine je objavljena verzija 2.0 ovog dokumenta kao predlog.

Navedene smernice objašnjavaju kako da se načini veb sadržaj koji će biti dostupan korisnicima sa posebnim potrebama. One su namenjene, pre svega, autorima veb strana i proizvođačima alata za razvoj veb strana. Iako su načinjene prevashodno zbog korisnika sa posebnim potrebama, njihovo poštovanje čini veb sajt dostupnijim svim korisnicima, bez obzira koji korisnički agent koriste (na primer grafički veb čitač, audio veb čitač, mobilni telefon, računar ugrađen u automobilu i slično). Pored toga poštovanje smernica doprinosi boljoj pristupačnosti veb sadržaju u posebnim uslovima kao što su: bučno okruženje, preosvetljene prostorije, okruženje kada ruke korisnika moraju da budu slobodne za druge operacije i slično. Smernice nemaju nameru da obeshrabre autore veb strana da koriste ultimedijalne sadržaje, već daju objašnjenja kako da se multimedijalni sadržaj učini dostupniji.

Razmotrimo, kao primer jedne smernice, kako se može napraviti tekstualni ekvivalent za slike. Pošto ljudi sa oslabljenim vidom i neki drugi korisnici ne mogu da vide slike, potrebno je u veb stranu ugraditi tekstualni ekvivalent za sliku. Tekstualni sadržaj može biti prikazan, pored uobičajenog grafičkog načina, i sintetizovanim govorom i Brajovom azbukom. Svaki od ovih metoda koristi različita čula: vid, sluh i dodir, čineći informacije dostupnim različitim grupama korisnika sa posebnim potrebama. Da bi se načinio tekstualni ekvivalent za sliku, tekst koji objašnjava sliku mora da postigne istu funkciju i svrhu koju je autor strane htio da postigne sa tom slikom. Na primer, ako je slika zemljine kugle stavljena na veb stranu u cilju dekoracije, onda bi tekst „pogled na Zemlju iz vaseone“ bio sasvim adekvatan. Ako je svrha slike da ilustruje neku geografsku informaciju, onda je potrebno publikovati adekvatnu tekstualnu informaciju.

TEHNOLOGIJE KOJE POBOLJŠAVAJU DOSTUPNOST

U cilju poboljšanja veb dostupnosti za osobe sa posebnim potrebama razvijen je čitav niz pomoćnih tehnologija koje omogućavaju navigaciju i prijem informacija publikovanih na veb sajt

Zadatak autora veb strana je da obezbede tekstualne ekvivalente slika i drugih multimedijalnih sadržaja. S druge strane, zadatak korisničkih agenata kao što su veb čitači i druge pomoćne tehnologije je da prezentuju informacije. U cilju poboljšanja veb dostupnosti za osobe sa posebnim potrebama razvijen je čitav niz pomoćnih tehnologija koje omogućavaju navigaciju i prijem informacija publikovanih na veb sajtu. U daljem tekstu se navode neke.

Programi za prepoznavanje govora mogu da pomognu osobama sa oslabljenim motornim karakteristikama da glasom unesu navigacione komande ili popunjavaju veb forme.

Programi za uvećanje prikaza sadržaja, omogućavaju osobama sa oslabljenim vidom da čitaju sadržaj. Neki veb čitači imaju ugrađene ovakve alate.

Specijalne tastature olakšavaju unos podataka osobama sa oslabljenim motornim karakteristikama.

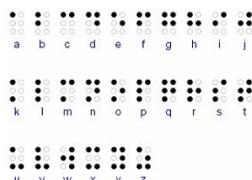
Programi za čitanje sadržaja prikazanog na ekranu (engl. **screen reader**) i svega što se događa na njemu upotrebom sintetizatora govora se koriste od strane slepih korisnika ili korisnika sa oslabljenim vidom. Ovakvi programi mogu da budu korisni i za osobe koje imaju teškoće sa čitanjem i učenjem, ili u radnim uslovima koji ne dozvoljavaju korisniku da koristi displej.

Pored sintetizovanja glasa, programi za čitanje sadržaja mogu da prikazuju informacije i na Brajovim displejima. JAWS (**Job Access With Speech**) koji korisnicima može glasom ili pomoći Brajovog displeja da prenese informacije koje su prikazane na displeju.

BRAJOVI DISPLEJI

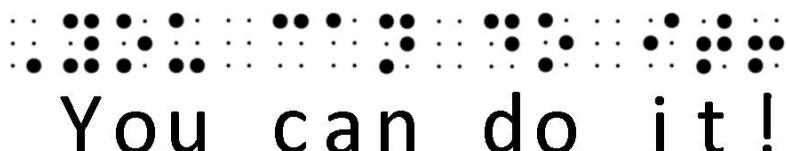
Brajovi displeji sa osvežavanjem su specijalne table iz kojih mogu da se uzdignu tačke koje odgovaraju Brajovim karakterima

Brajovi displeji sa osvežavanjem su specijalne table iz kojih mogu da se uzdignu tačke koje odgovaraju Brajovim karakterima. Za prezentaciju jednog karaktera se koristi 6 tačaka. Na narednoj slici je prikazana reprezentacija tekstualnih karaktera Brajovim pismom. Sleppe osobe pipajući tačke prepoznaju slova ili brojeve. Podizanje tačkama se upravlja iz računara. Na taj način je moguće slepim osobama prikazati tekst koji je publikovan na veb sajtu, a koga one ne mogu da vide.



Slika 5.1.1 Brajova azbuka

[Izvor: <https://www.louisbrailleonlineresource.org/uploads/1/9/4/4/19443713/editor/brl-alphabet.jpg?1517511252>]



Slika 5.1.2 Primer ispisan Brajovom azbukom

[Izvor: https://brailleworks.com/wp-content/uploads/2016/10/Grade-1-Braille-Example_2-700x141.jpg]

Sadržaj koji se prikazuje na Brajovim displejima se može i stampati na specijalnim Brajovim štampačima, koji vrše reljefnu štampu. Neke od ovih tehnologija mogu da se primene samo ako im veb sajt to omogućuje.

Poštovanje pojedinih smernica uobičajeno donosi prednosti različitim grupama korisnika sa posebnim potrebama. Na primer, korišćenje stilova za upravljanje fontovima i eliminacija HTML-ovog FONT elementa će olakšati autoru upravljanje stranicama, učiniti te strane mnogo dostupnije osobama sa oslabljenim vidom i smanjiti vreme čitanja stranica svim korisnicima.



Slika 5.1.3 Primer Brajovih displeja [Izvor: Autor]



Slika 5.1.4 Brajov štampač

[Izvor: https://exceed.lv/images/braila_produkti/BRA22.png]

▼ 5.1 OSNOVNA PRAVILA DOSTUPNOSTI

SMERNICE KOJE SADRŽE OSNOVNA PRAVILA DOSTUPNOSTI

Smernice sadrže dva osnovna pravila dostupnosti: osiguranje elegantne transformacije i osiguranje razumljivosti i navigacionosti sadržaja.

Smernice sadrže dva osnovna pravila dostupnosti:

- Osiguranje elegantne transformacije i
- Osiguranje razumljivosti i navigacionosti sadržaja.

Smatra se da se strana transformiše elegantno ako ostaje dostupna uprkos nametnutim fizičkim, senzornim i kognitivnim ograničenjima korisnika, radnim ograničenjima i tehnološkim barijerama. Da bi se osigurala elegantna transformacija treba poštovati sledeće smernice:

- Odvojiti strukturu od prezentacije,
- Obezbediti tekst i tekstualne ekvivalente za multimedijalne sadržaje. Tekst treba da može da bude interpretiran u gotovo svim uređajima za čitanje i dostupan svim korisnicima,
- Kreirati dokumente koji su funkcionalni čak i kada korisnik ne može da vidi i/ili čuje.

- Obezbediti informacije koje istu svrhu i funkciju kao audio ili video na način koji je pogodan za alternativne senzorske kanale. Ovo ne znači da treba kreirati unapred snimljenu audio verziju celog sajta da bi bila dostupna slepim osobama. Korisnici koji su slepi mogu koristiti čitače ekrana da dođu do publikovanih tekstualnih informacija.
- Kreirati dokumente čija se dostupnost ne zasniva na jednom tipu hardvera. Strane treba da budu dostupne i za korisnike koji ne koriste miša, imaju male ekrane, nisku rezoluciju, crno - belu sliku ili možda uopšte nemaju ekran

Veb stranice treba projektovati tako da sadržaj bude razumljiv i da je moguća navigacija. Da bi se ovo osiguralo treba poštovati sledeće smernice:

- Jezik treba da bude jasan i jednostavan,
- Navigacioni mehanizam unutar i između stranica treba da bude razumljiv. Treba imati na umu da neki korisnici ne mogu da koriste navigacione mape, skrol barove, okvire i ikone. Korisnici takođe nekada ne mogu da dođu do informacija koje slede iz konteksta cele strane, jer vide samo deo strane (mali displeji ili displeji sa uvećanjem) ili zato što pristupaju sadržaju reč po reč (sintetizatori govora i Brajova azbuka). Bez orijentacionih informacija korisnici neće moći da razumeju vrlo velike tabele, liste i menije.

▼ Poglavlje 6

IMPLEMENTACIJA I INTEGRACIJA

PROCES RAZVOJA I IMPLEMENTACIJE VEB SAJTA

Faze u procesu razvoja i implementacije veb sajta su pripremna faza, faza projektovanja i faza implementacije i evaluacije

Razvoj veb sajta je složen problem, pa autor treba da prati uobičajen redosled koraka koji su tipični za razvoj novih sistema. Može se reći da se proces razvoja i implementacije veb sajta sastoji od sledećih faza:

1. Pripremna faza

- Definisanje ciljeva i problema
- Definisanje korisnika
- Definisanje zahteva

2. Faza projektovanja

- Projektovanje strukture
- Projektovanje navigacije
- Projektovanje sistema za pretraživanje
- Izbor tehnologije

3. Faza implementacije i evaluacije

- Izrada prototipa
- Proizvodnja HTML koda i drugog skripta
- Validacija
- Testiranje

PRIPREMNA FAZA I FAZA PROJEKTOVANJA

U pripremnoj fazi autor treba da precizno definiše zahteve za novi sajt, a u projektnoj fazi da isprojektuje sistem

Tokom pripremne faze autor treba da se upozna sa razlozima za postavljanje novog veb sajta. Na osnovu toga je potrebno odrediti koji su tipični korisnici veb sajta. Ovo je vrlo važno za dobro definisanje zahteva. Praksa je pokazala da samo klijent - centričan pristup u projektovanju veb sajta daje dobre rezultate. Zato je važno znati sve osobine prosečnog korisnika, koje će zadatke obavljati na sajtu, koje će tehnologije koristiti i u kojim uslovima.

Na osnovu svega ovog je potrebno precizno definisati zahteve koji se postavljaju veb sajtu. Najvažniji deo zahteva se odnosi na sadržaj koji će biti publikovan na veb sajtu i funkcionalnost koju će veb sajt imati.

U drugoj fazi, **fazi projektovanja**, se vrši projektovanje sistema. Pošto je poznat sadržaj koji će se publikovati na sajtu potrebno je definisati strukturu veb sajta. To podrazumeva da se najpre definije mapa veb sajta. Vrlo je važno logički podeliti informacije koje se publikuju. Za svaku stranicu koja se publikuje treba odrediti naslov i sadržaj. Naslov stranice treba da bude dovoljno indikativan kako bi pomogao korisniku da pronađe informacije.

Pri tom treba zadovoljiti potrebe organizacije koja publikuje veb sajt ali i očekivanja korisnika. Iste informacije se mogu publikovati korišćenjem različitih medija (tekst, slike, zvuk, video) pa je potrebno definisati i koji će se mediji koristiti za koje informacije. Osim toga, na umu treba stalno imati zahteve korisnika sa specijalnim potrebama.

Posebnu pažnju treba posvetiti projektovanju navigacije. Navigaciju treba uskladiti sa funkcijama veb sajta. Pored korišćenja osnovnih principa navigacije, i ovde je potrebno voditi računa o korisnicima sa posebnim zahtevima.

U slučajevima veb sajtova sa velikom količinom informacija, publikovanje mape sajta nije dovoljno za brzo nalaženje željenih informacija. Naslovi veb stranica govore o sadržaju ali korisnik ne mora da prepozna pravu stranicu. Zbog toga se ovakvim sajтовima, kao poseban oblik navigacije, pridodaje sistem za pretraživanje.

Kada su projektovani svi elementi veb sajta bira se tehnologija kojom će se sajt implementirati. Ovo podrazumeva izbor skript jezika, razvojnih i produkcionih alata, alata za kreiranje multimedijalnog sadržaja, veb servera i opciono baze podataka. Pri izboru tehnologija treba imati na umu činjenicu da korisnici mogu da koriste različite uređaje za pristup sajtu.

FAZA IMPLEMENTACIJE I EVALUACIJE

Tokom faze implementacije se najpre razvija tehnički i vizuelni prototip sajta. Evaluacija obuhvata validaciju sadržaja, dizajna, navigacije i linkova

Tokom **faze implementacije** se najpre razvija tehnički i vizuelni prototip sajta. Ovo podrazumeva implementaciju navigacionog mehanizma i dela sadržaja koji se publikuje. U ovoj ranoj fazi implementacije treba ispitati zadovoljstvo korisnika i proveriti stepen ispunjenosti njegovih očekivanja. U sledećem koraku treba korišćenjem izabranih alata generisati HTML strane, napraviti slike, zvučne i video sadržaje ukoliko se publikuju na sajtu. Ako se koriste stranice sa stilovima, prethodno je potrebno definisati stilove. Ukoliko se radi o dinamičkim stranama treba razviti i odgovarajući skript za klijentsku i ili serversku stranu.

Kada je sajt potpuno implementiran i funkcionalan vrši se njegova **validacija**. Validacija obuhvata sadržaj, dizajn, navigaciju i linkove. Za validaciju sajtova mogu se koristiti i specijalni programski alati za validaciju. Oni su posebno korisni ako se vrši validacija veb dostupnosti prema specificiranim zahtevima.

Poslednji korak implementacije je testiranje veb sajta u realnim uslovima korišćenja. Testiranje sajta treba da dokaže da je sajt ispunio sve specificirane zahteve.

U nekim slučajevima veb sajt je potrebno integrisati sa drugim veb sajтовима ili sa informacionim sistemom organizacije.

✓ Poglavlje 7

Pokazna vežba: HTML5 forme

ŠTA SU HTML FORME?

HTML forme su jedan od načina interakcije sa korisnikom

Predviđeno vreme za pokaznu vežbu je 30 minuta.

HTML forme su veoma bitan deo svake web stranice. Njihova uloga je da interaguju sa korisnikom. Različite forme koriste različite načine interakcije u zavisnosti od toga koja im je uloga, koji podaci se u formu unose, itd.

HTML5 uvodi neke nove tipove polja za unos podataka koja do sada nisu postojala, I samim tim pružaju formama više funkcionalnosti, opcija kao I bolji dizajn.

Prilikom kreiranja formi treba imati u vidu ciljnu grupu aplikacije I prilagoditi dizajn njima. Takođe, cela aplikacija bi trebalo da bude responsive, odnosno da se može pravilno prikazati na svakom uređaju (PC, tablet, mobilni telefon..).

Dve glavne uloge formi:

- **Slanje podataka** – forme mogu služiti za slanje podataka koji mogu ići na različite lokacije. Neke forme, kao što su kontakt forme, se koriste za slanje mail-ova ili poruka na određenu adresu, dok neke mogu da komuniciraju sa bazom I u nju upisuju potrebne podatke
- **Validacija podataka** – forme za login I registraciju su veoma šesto korišćene I zahtevaju unos podataka u određenoj formi. Na primer, šifra mora imati više od 8 karaktera, e-mail adresa mora biti validna, itd. Tada se koristi provera unetih podataka odnosno da li poštuju sva neophodna pravila. Ukoliko nisu ispoštovana, podaci se moraju ponovo uneti.

PRIMER FORME

Primer login forme

Sledeći listing prikazuje jednostavnu login formu koja ima dva polja za unos (korisničko ime i lozinka) i dugme za logovanje. HTML dokument bi trebalo da izgleda ovako:

```
<html>
  <head>

  </head>
  <body>
```

```
<h2>Login Forma</h2>

<form action="login.php">
    <div class="container">
        <label><b>Unesite korisničko ime</b></label>
        <input type="text" placeholder="Korisničko ime" name="uname" required>

        <label><b>Unesite lozinku</b></label>
        <input type="password" placeholder="Lozinka" name="psw" required>

        <button type="submit">Login</button>
    </div>
</form>
</body>
</html>
```

```
form {
    border: 3px solid #f1f1f1;
}

input[type=text], input[type=password] {
    width: 200px;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    box-sizing: border-box;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 200px;
}

button:hover {
    opacity: 0.8;
}

.container {
    padding: 16px;
}
```

LOGIN FORMA IZGLED

Izgled forme iz prvog zadatka

Kreirana forma bi trebalo da izgleda ovako:

Login Forma

The form consists of two input fields: 'Korisničko ime' and 'Lozinka', both with placeholder text 'Unesite korisničko ime' and 'Unesite lozinku'. To the right of these fields is a large green rectangular button labeled 'Login'.

Slika 7.1 Izgled login forme [Izvor: Autor]

KONTAKT FORMA

Primer forme za slanje poruke

Sledeći primer predstavlja izgled kontakt forme za slanje mail-a. Korisnik od podatka unosi naslov mail-a, zatim adresu na koju se šalje mail i tekst poruke koja se šalje. Sledeći listing predstavlja HTML dokument:

```
<html>
  <head>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>

    <h2>Kontakt Forma</h2>

    <form action="login.php">
      <div class="container">
        <label><b>Naslov mail-a</b></label>
        <input type="text" placeholder="Naslov" name="title" required>

        <label><b>Unesite e-mail adresu</b></label>
        <input type="text" placeholder="E-mail adresa" name="address" required>

        <label><b>Unesite tekst poruke</b></label>
        <textarea placeholder="Poruka" id="txtarea" name="msg" rows="5" cols="26"></textarea>

        <button type="submit">Login</button>
      </div>
    </form>
  </body>
</html>
```

Zatim treba definisati CSS koji će stranici dati lepši izgled.

```
form {
  border: 3px solid #f1f1f1;
}

input[type=text], input[type=password] {
  width: 200px;
```

```
padding: 12px 20px;  
margin: 8px 0;  
display: block;  
border: 1px solid #ccc;  
box-sizing: border-box;  
}  
  
button {  
background-color: #4CAF50;  
color: white;  
padding: 14px 20px;  
margin: 8px 0;  
border: none;  
cursor: pointer;  
width: 200px;  
}  
  
button:hover {  
opacity: 0.8;  
display: block;  
}  
  
.container {  
padding: 16px;  
}  
  
#txtarea{  
display: block;  
}
```

KONTAKT FORMA IZGLED

Izgled forme iz drugog zadatka

Kreirana forma bi trebalo da izgleda ovako:

Kontakt Forma

Naslov mail-a

Unesite e-mail adresu

Unesite tekst poruke

Login

Slika 7.2 Izgled kontakt forme [Izvor: Autor]

▼ Poglavlje 8

Pokazna vežba: HTML5 i CSS4

PRAVLJENJE ANIMACIJA

Pravljenje animacije je vrlo atraktivna opcija jer sad mnogi efekti zbog kojih je ranije bio neophodan flash mogu da se urade samo s HTML-om i CSS-om

Predviđeno vreme pokazne vežbe je 30 minuta.

Verzija CSS-a 3 ima čitav niz novina koje olakšavaju uređenje HTML strana ali i veliki broj novih opcija koje nisu postojale u predhodnim verzijama od kojih je svakako najinteresantnija mogućnost kreiranja animacija.

Pravljenje animacije je vrlo atraktivna opcija jer sad mnogi efekti zbog kojih je ranije bio neophodan flash mogu da se urade samo s HTML-om i CSS-om.

ANIMACIJA - SAT

Primer analognog sata koristeći HTML i CSS

Jedan od najinteresantnijih primera kako radi CSS3 je animirani analogni časovnik pomoću HTML-a i CSS-a.

HTML deo koda:

```
<div id="experiment">
    <p class="start"><a href="#clock" id="start">Start Clock</a></p>
    <div id="clock">
        <div id="hour">
            
        </div>
        <div id="minute">
            
        </div>
        <div id="second">
            
        </div>
    </div>
</div>
```

CSS deo koda:

```
/*
Clock Experiment
Date: 24th March 2009
Author: Paul Hayes
*/
.start {
text-align: center;
font-size: 2em;
font-weight: bold;
margin: 5em;
}

#clock {
position: relative;
width: 378px;
height: 378px;
background-image: url('../images/clockFace.png');
left: 50%;
margin: 5em 0 0 -189px;
}

#clock div {
position: absolute;
}

/* The magic */
#clock img[src*='second'] {
-webkit-transition: -webkit-transform 600000s linear;
}

#clock:target img[src*='second'] {
-webkit-transform: rotate(3600000deg);
}

#clock img[src*='minute'] {
-webkit-transition: -webkit-transform 360000s linear;
}

#clock:target img[src*='minute'] {
-webkit-transform: rotate(36000deg);
}

#clock img[src*='hour'] {
-webkit-transition: -webkit-transform 216000s linear;
}

#clock:target img[src*='hour'] {
-webkit-transform: rotate(360deg);
}
```

MENJANJE BOJA I VELIČINE KORISTEĆI DIV

Primer koda koji menja boju i veličinu nekom divu

Jedan efekat koji je uvek iziskivao flash sajtove je obična tranzicija kao na power point slajdovima. Sa CSS3 i to je moguće i lako napraviti. Primer koda koji menja boju i veličinu nekom divu:

```
div {  
background-color: red;  
width: 3em;  
height: 3em;  
-o-transition-property: background-color, width, height;  
-webkit-transition-property: background-color, width, height;  
transition-property: background-color, width, height;  
-o-transition-duration: 4s, 8s, 5s;  
-webkit-transition-duration: 4s, 8s, 5s;  
transition-duration: 4s, 8s, 5s;  
-o-transition-delay: 0s, 0s, 2s;  
-webkit-transition-delay: 0s, 0s, 2s;  
transition-delay: 0s, 0s, 2s;  
}  
div:hover {  
background-color: blue;  
width: 15em;  
height: 10em;  
}
```

✓ Poglavlje 9

Pokazna vežba: HTML multimedije

HTML VIDEO ZAPISI

HTML podržava niz audio i video formata

Predviđeno vreme pokazne vežbe je 30 minuta.

HTML5 uvodi nove elemente; audio i video zapise. Pre ove verzije HTML-a, video i audio zapisi su se mogli puštati samo pomoću nekog plug-in-a, najčešće flasha. HTML5 podržava niz formata za audio i video zapise. Za dodavanje ovih elemenata koriste se tagovi **<audio>** i **<video>**.

Ovako bi izgledao element u HTML dokumentu:

```
<video src="rabbit320.mp4" controls> </video>
```

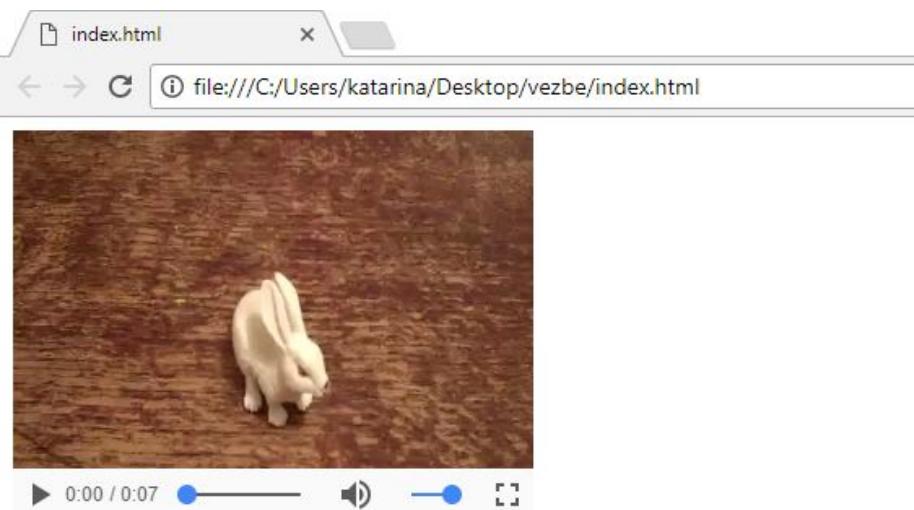
src predstavlja putanju do video zapisa, a **controls** znači da će se na videu prikazivati kontrole za pauzu, premotavanje, itd.

U slučaju da korisnik ne može da pusti video iz nekog razloga (ukoliko na primer ima stariju verziju browsera), trebalo bi pružiti alternativni link ka videu.

```
<video src="rabbit320.mp4" controls>
  <p>Your browser doesn't support HTML5 video. Here is a <a
  href="rabbit320.webm">link to the video</a> instead.</p>
</video>
```

Između početnog i završnog video taga može se ubaciti tag p kao i link ka videu.

Izgled ubačenog videa:



Slika 9.1 Video zapis [Izvor: Autor]

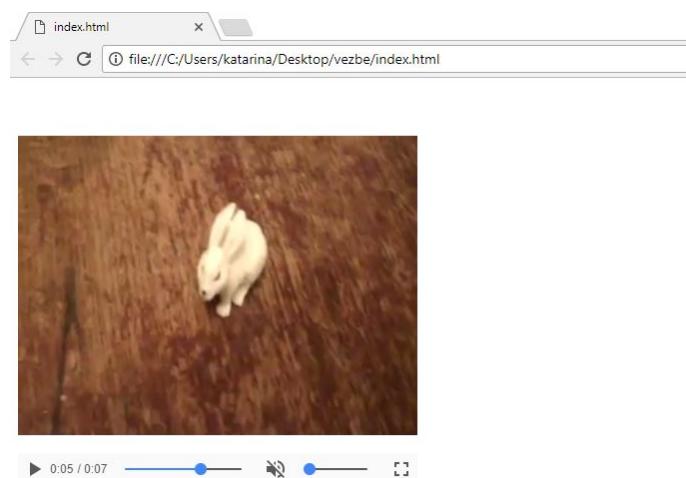
HTML VIDEO ZAPISI - DODATNE FUNKCIONALNOSTI

HTML5 podržava razne opcije za upravljanje videom

Pored **controls** atributa, postoje još i **autoplay**, **loop**, **muted**, itd. Može se podešiti i veličina videa (visina i širina) kao i svaki drugi HTML element.

```
<video controls width="400" height="400"  
       autoplay loop muted>  
  <source src="rabbit320.mp4">  
  <p>Your browser doesn't support HTML5 video. Here is a <a  
      href="rabbit320.mp4">link to the video</a> instead.</p>  
</video>
```

Na slici 2 je prikazan izgled video zapisa nakon primenjenih opcija.



Slika 9.2 Dodatne opcije [Izvor: Autor]

Kada se stranica učitala, video se automatski pustio, zbog atributa **autoplay**. Takođe može se videti da je zvuk ugašen zbog atributa muted.

HTML AUDIO

Zvuk se može postaviti u pozadini, da se kreira link do izvora zvuka i da se ugradi zvuk

Postoje tri pristupa upotrebe zvuka koji se razlikuju po načinu reprodukcije kada web čitač nađe na njih. Ovi načini su:

- postavljanje zvuka u pozadini, što znači da će zvuk biti reprodukovani čim se otvoriti stranica,
- kreiranje hiperveze do izvora zvuka, što znači da se reprodukcija vrši uz pomoć druge aplikacije, i
- ugrađivanje zvuka, kada reprodukciju izvršava pretraživač ili dopunski modul.

Kao i video zapis, i audio se lako unosi. Koriste se tagovi `<audio>` i može imati različite attribute za upravljanje audiom.

```
<audio controls>
    <source src="r.mp3" type="audio/mp3">
</audio>
```

Izgled ubačenog audio zapisa:



Slika 9.3 Izgled audio zapisa sa kontrolama [Izvor: Autor]

▼ Poglavlje 10

Zadaci za samostalan rad: HTML5 i CSS4

ZADACI ZA VEŽBANJE

Kreirati sledeće forme za unos podataka

Predviđeno vreme za izradu zadataka je 45 minuta.

1. Kreirati formu za unos knjiga u bazu biblioteke. Od podataka treba uneti sledeće: naziv knjige, autora, godinu izdavanja, žanr, ISBN, datum unosa, izdavač. (Predviđeno vreme izrade: 5 minuta)
 2. Kreirati formu za upis studenata na kurs stranog jezika. Osmisliti minimum 6 atributa za unos, i uneti podatke o bar 5 studenata. (Predviđeno vreme izrade: 10 minuta)
 3. Kreirati formu za unos podataka o robi za veliki trgovinski lanac. Osmisliti 10 atributa i popuniti tabelu sa primerima. (Predviđeno vreme izrade: 10 minuta)
 4. Kreirati HTML stranu sa formom za unos podataka o igračkama koje se prodaju u prodavnici. (Predviđeno vreme izrade: 20 minuta)
- Svaka igračka ima sledeće podatke: ID, ime, ime proizvođača, datum proizvodnje, cenu.
 - Svaka uneta igračka mora se upisati u bazi koja je unapred kreirana.
 - Na novoj strani kreirati tabelu u kojoj će se ispisati igračke koje su upisane u bazi.
 - Igračke čija je cena veća od 2000 dinara biće ispisane crvenom bojom, ostale zelenom bojom.

✓ Poglavlje 11

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ05

Opis domaćeg zadatka

Predviđeno vreme za izradu domaćeg zadatka je 30 minuta.

- Kreirati bazu podataka sa podacima o zaposlenim u nekoj firmi
- Kreirati formu za unos novog radnika.
- Podaci koje baza mora sadržati su: ime, prezime, odeljenje, plata, godine radnog staža.
- Kreirati stranu sa tabelom u kojoj su ispisani podaci o svim radnicima.
- Kreirati stranu na kojoj su date statistike o radnicima kao što su: najveća, najmanja i prosečna plata, sortirati radnike po godinama radnog staža.
- Koristiti bazu podataka po izboru. Dostaviti fajl baze podataka ili SQL za kreiranje baze kao rešenje.

Potrebne datoteke snimiti pod imenom: **IT210-DZ05-Ime_Prezime_brojIndexa** gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovane datoteke poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210 – DZ05)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovoj lekciji smo opisali globalne digitalne biblioteke i naveli njihov značaj i prednosti korišćenja. Kako se na vebu koriste razni multimediji, opisali smo neke od osnovnih formata multimedija, kao i način na koji se oni komprimuju kako bi lakše bili prenosivi preko veba. Opisali smo proces razvoja veb interfejsa, arhitekturu informacija, kao i osnovne problem dostupnosti sadržaja za sve na Internetu.

Literatura

- [1] Tim Monroe, Broadcasting Movies Over a Network, <http://www.mactech.com/articles/mactech/Vol.18/18.03/Mar02QTToolkit/index.html> (27.01.2014)
- [2] Axis, http://www.axis.com/products/video/about_networkvideo/mpeg4.htm (26.01.2014)
- [3] Branislav Đorđević, Dragan Pleskonjić, Nemanja Maček, Operativni sistemi, Mikro knjiga, 2005.
- [4] Larry Twork, Larry Mead, Bill Howison, JD Hicks, Lew Brodnax, Jim McMicking, Raju
- [5] Sakthivel, David Holder, Jon Collins, Bill Loeffler, UNIX Application Migration Guide, Chapter 2: UNIX and Windows Compared, msdn.microsoft.com, 2002.
- [6] William Boswell, Inside Windows Server 2003, Addison Wesley, 2003.
- [7] Machtelt Garrels, Introduction to Linux, A Hands on Guide, CoreSequence.com, Version 1.4 Last updated 20030427 Edition
- [8] Steve D. Pate, UNIX Filesystem, Evolution, Design and Implementation, Wiley, 2003
- [9] Rlrishmy Card, Theodore Ts'o, Stephen Tweedie, Design and Implementation of the Second Extended Filesystem, Proceedings of the First Dutch International Symposium on Linux, ISBN 90-367-0385-9
- [10] Wendy Chisholm, Gregg Vanderheiden, Ian Jacobs, Web Content Accessibility Guidelines 1.0, W3C, 1999.
- [11] Ben Caldwell, Wendy Chisholm, John Slatin, Gregg Vanderheiden, Web Content Accessibility Guidelines 2.0, Working Draft, W3C, 2005.



IT210 - SISTEMI IT

**MAPIRANJE I RAZMENA
PODATAKA**

Lekcija 06

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 06

MAPIRANJE I RAZMENA PODATAKA

- ✓ MAPIRANJE I RAZMENA PODATAKA
- ✓ Poglavlje 1: METAPODACI
- ✓ Poglavlje 2: XML
- ✓ Poglavlje 3: PROVERA ISPRAVNOSTI U XML DOKUMENTU
- ✓ Poglavlje 4: OSNOVNI KONCEPTI RAŠČLANJIVANJA PARSOVANJA XML DOKUMENTA
- ✓ Poglavlje 5: XSL transformacije
- ✓ Poglavlje 6: Pokazna vežba - Kreiranje osnovnog XML dokumenta
- ✓ Poglavlje 7: Pokazna vežba: Audio zapisi na veb stranama
- ✓ Poglavlje 8: Zadaci za samostalnu vežbu
- ✓ Poglavlje 9: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije da se razume mapiranje i razmena podataka, meta podaci, XML; DTD; XML šeme, raščlanjivanje XML dokumenta, XSL; XSLT i Xpath.

U ovoj lekciji biće obrađene sledeće teme:

- Mapiranje i razmena podataka
- Meta podaci
- XML, DTD, XML šeme
- Raščlanjavanje
- (parsing)XML dokumenta
- XSL, XSLT i XPath

▼ Poglavlje 1

METAPODACI

METAPODACI

Metapodaci su podaci o podacima. Metapodaci postoje za sve grupe objekata bez obzira da li su u elektronskom obliku ili ne.

Koncept meta podataka (eng. metadata) se pojavio pre veba, a prvi put je počeo da ga koristi Džek Mejers, 1960-ih godina. On ga je koristio za opis skupova podataka. Meta podaci su podaci o podacima. Na primer, osnovne informacije, kao što su ime autora nekog dokumenta, datum kreiranja dokumenta, linkovi na dokumente koji su povezani sa tim dokumentom su metapodaci o nekom dokumentu. Jedan od oblika meta podataka na koji ste sigurno nailazili je kartica knjiga u biblioteci, koja sadrži meta informacije o knjizi.

Meta podaci postoje za skoro sve grupe objekata, bez obzira da li se oni čuvaju u elektronskom obliku ili ne. Mapa sa rasporedom fabrika na primer, ima pridružene meta podatke, kao što su razmera, datum kreiranja i sl. Kod proizvoda kao što su mape, meta podaci se vide na samoj mapi, tako da ih iskusni korisnici mogu lako da interpretiraju.

Na internetu stvari nisu uvek tako jednostavne. Opšte prihvaćeni standardi još uvek ne postoje, tako da može biti velikih problema da se pronađe informacija koju tražite.

Trenutne generacije mašina za pretraživanje imaju bez sumnje velike mogućnosti, tako da su sposobne da vrati veliki broj pogodaka za svako zadato pretraživanje.

Skoro je nemoguće odbaciti pogotke koji nisu relevantni, tako da ostanu samo oni za koje je neko zainteresovan.

Zadatak

U nekoliko različitih pretraživača (Google, Bing, Yahoo i sl.) pokušajte da nađete tekst Ariadne. Koliko referenci i linkova ste pronašli? Da li ste našli relevantne podatke? Da li su vam ponuđeni i linkovi koji nisu relevantni vašoj pretrazi?

Ovaj jednostavan primer ilustruje neke od problema kod pronalaženja informacija na webu. On je analogan adresaru koji sadrži vaše adrese, koje nisu složene po prezimenu, već po svim poljima (prezime, ulica, mesto zaposlenja itd.). Kada u takvom adresaru pokušate da pronađete adresu, nema načina da znate na osnovu kog polja su dobijeni rezultati. Pod pretpostavkom da želite da kontaktirate urednika web časopisa i tražite ga po njegovom prezimenu, nema načina da znate da li ste rezultat dobili na osnovu prezimena, ili nekog drugog podatka u kome se to prezime pominje.

Da bi vaš adresar bio zaista koristan treba da dodate i neke meta podatke koji će opisati sa čime je određeni tekst povezan (na primer Ovo je prezime, ovo je grad i sl).

✓ 1.1 POSTOJEĆI PRISTUPI U RADU SA METAPODACIMA

STANDARDI ZA RAD SA METAPODACIMA

S obzirom da podaci na internetu rastu, teško je naći informaciju, pa se koriste metapodaci za opis tih informacija, kako bi se lakše pronašle. MARC je složena šema za biblioteke.

Danas postoji veliki broj standarda čija je namena opis elektronskih resursa, ali se ovi uglavnom bave opisom specifičnih resursa, pri čemu se često koriste komplikovane šeme, koje zavise od konkretnе oblasti, tako da se oni koji nisu eksperti u toj oblasti ne mogu lako snaći.

U okruženjima kao što su tradicionalne biblioteke, gde katalogiziranje i prikupljanje podataka rade profesionalci, se možda i mogu prihvati složene šeme meta podataka, kao što su MARC ([Machine Readable Catalogue](#)). U haotičnom svetu interneta se, međutim, svakodnevno javljaju novi resursi, koje često održavaju zainteresovani pojedinci, a ne centralizovane organizacije. U takvim slučajevima je teško lako pronaći informaciju. Mašine za pretraživanje, sa svim njihovim nedostacima, su često jedino rešenje.

U takvim okruženjima je očigledno da su potrebni meta podaci, ali ti meta podaci moraju biti u obliku koji mogu interpretirati kako ljudi, tako i mašine za pretraživanje. Ti meta podaci se moraju dodavati jednostavno, tako da bilo koji autor web strana može da opiše sadržaj svoje strane. Zbog toga se moraju praviti kompromisi između činjenice da je potrebno obezbediti što više informacija, a da pri tome tehnika ostane dovoljno jednostavna da je može koristiti maksimalan broj ljudi sa minimalnim tehničkim znanjem.

EKSPERTSKI PRISTUP

Postoje veliki broj tehnika za opis resursa kao što su MARC i DIF.

Postoje projekti kao što su TEI ili NSDI.

Danas postoji veliki broj tehnika za opis resursa kod elektronskih medijuma, počev od [MARC](#) standarda, koji se koristi u bibliotekama, pa do [DIF](#) ([Directory Interchange Format](#)) formata koji se koristi za definisanje meta podataka kod satelitskih snimaka.

Postoje i različiti projekti poput [Text Encoding Initiative](#)([TEI](#)) koji se odnose na standardizovan opis teksta u elektronskom formatu, ili [NSDI](#) ([National Spatial DataInfrastructure](#)) koji se koristi za opis prostornih podataka.

Svaki od pomenutih formata je napravljen da funkcioniše u okviru određenog delokruga, tako da se teško može primeniti na različite resurse. Većina ovih šema meta podataka su izuzetno složene i usklađene sa činjecom da su ih pravili eksperti, a da ih interpretiraju računari

PRISTUP KOJI KORISTE MAŠINE ZA PRETRAŽIVANJE

Indeksiranje strane se može vršiti na osnovu META atributa DESCRIPTION i KEYWORDS

Prepoznavši potrebu da se pronađe način da se pretraživanje prilagodi onome što korisnik traži, maštine za pretraživanje su počele da koriste oznake **META** u HTML dokumentima. Indeksiranje strane se može vršiti na osnovu META atributa DESCRIPTION i KEYWORDS. Sadržaj atributa DESCRIPTION se vraća kao rezultat pretraživanja, a ne kao što je uobičajeno, prvih nekoliko redova teksta.

Na primer, ako u delu <**HEAD**> nekog web dokumenta stavite:

```
<META NAME="description" CONTENT="Meta podaci">  
<META NAME="keywords" CONTENT="Dublin Core, metadata">
```

Današnji pretraživači se ne oslanju toliko na ključne reči koje su definisane u samom HTML-u, ali obraćaju pažnju više na DESCRIPTION.

Interesantan video, kao i spisak podataka koji se koriste za pretragu, bilo da su meta ili ne, možete pogledati ovde:

Meta tags that Google understands

<https://support.google.com/webmasters/answer/79812?hl=en>

▼ 1.2 WARWICK OKRUŽENJE

WARWICK OKRUŽENJE

Warwick je arhitektura koja je napravljena za rukovanje različitim skupovima meta podataka.

Uvek su postojali različiti standardi za meta podatke. Warwick je arhitektura koja je napravljena za rukovanje različitim skupovima meta podataka. U ovom okruženju se koristi model sa "paketom kontejnerom". U pitanju je samo konceptualno okruženje. To znači da se način rukovanja kontejnerima i paketima mora da obezbedi u konkretnoj aplikaciji modela. Kontejner je mehanizam za držanje skupova paketa na okupu. Postoje tri tipa paketa:

- Primitivni paket, koji sadrži jedan ili više meta podataka. Svaki primitivni paket ima tipa, na primer MARCK paket, Dublin Core paketa, FGDC paket.

- Indirektni paket, koji ukazuje na neki drugi resurs sa informacijama, na primer preko linka.
- Kontejner paket. Paket može biti kontejner, i ne postoji ograničenje u nivoima gneždenja.

Ovakvo konceptualno okruženje se naziva Warwick okruženje. Čini se da je izuzetno jednostavno, ali je u suštini to vrlo dobar model za rukovanje meta podacima. Prednost je da je modularan (meta podaci se nalaze u paketima), da može da se proširuje (nema ograničenja u broju paketa koji se ubacuju u kontejner), da je distribuisan (preko indirektnih paketa) i rekursivan (paket može biti i kontejner).

✓ 1.3 STANDARD DUBLIN CORE

STANDARD DUBLIN CORE

Dublin Core je napravljen kao opšti standard, namenjen upotrebi u bibliotekama, arhivama, vladinim agencijama, kao i kod svih drugih koji publikuju informacije preko interneta.

Skup elemenata Dublin Core je razvijen tokom 1995 i 1996-e godine, kao odgovor na potrebu za poboljšanjem dobijanja informacija, posebno na WWW-u. Dublin Core je napravljen kao opšti standard, namenjen upotrebi u bibliotekama, arhivama, vladinim agencijama, kao i kod svih drugih koji publikuju informacije preko interneta.

Kada se pojavio Dublin Core je bio ograničen svojim ciljevima. Interno je bio namenjen za opis dokumenata poput objekata, kao što su HTML strane, PDF datoteke i grafičke slike. Dokazano je da je praktično vrlo teško definisati šta ovaj sadržaj znači, a posebno šta ne znači, odnosno šta ne uključuje.

Standard je namerno tako napravljen da sadrži mali skup elemenata, koji se mogu primeniti na različite informacione resurse. U standardu trenutno postoji petnaest elemenata.

OSNOVNI SKUP ELEMENATA ZA DUBLIN CORE

Dublin Core sadrži petnaest osnovnih elemenata. Svaki od tih elemenata se može dalje proširivati, preko atributa SCHEME i TYPE.

Dublin Core sadrži petnaest osnovnih elemenata. Svaki od tih elemenata se može dalje proširivati, preko atributa SCHEME i TYPE. Neki od elemenata su:

Ime elementa	Opis elementa
Subject	Tema kojom se bavi objekat koji se opisuje
Title	Ime objekta
Author	Osobe (ili više njih) koje su odgovorne za sadržaj objekta
Publisher	Agent ili agenciju koji su odgovorni za postavljanje objekta
OtherAgent	Osobe, kao što su urednici ili prevodnici, koji su dali doprinos
Date	Datum publikovanja
ObjectType	Žanr objekta, kao što su na primer priča, pesma, rečnik i sl.
Form	Format podataka, kao što su Postscript, HTML i sl.
Identifier	Niz karaktera ili broj koji jednoznačno identificuju objekat.
Relation	Veze između tog i drugih objekata
Source	Objekti, štampani ili u elektronskom formatu iz kojih je objekat izveden
Language	Jezik kojim je sadržaj pisan
Coverage	Prostorna lokacija i vremensko trajanje karakteristika objekta

Slika 1.1.1 Prikaz elemenata [Izvor: Autor]

Ako se meta podaci ubacuju na web strane, onda se koristi oznaka <META> iz HTML-a. Ova oznaka se nalazi u delu <HEAD> na strani. Evo kako to izgleda:

```
<META NAME="DC.imeElementa" CONTENT="Vrednost elementa">
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
<HEAD>
<TITLE>Metadata for the masses</TITLE>
<META NAME="package" CONTENT="(TYPE=begin) Dublin Core">
<META NAME="DC.title" CONTENT="(TYPE=long) Metadata for the masses: what is it, how
can it help me, and how can I use it?">
</HEAD>
<BODY>
...{telo dokumenta}...
```

Kada se meta podaci pišu na ovakav način, korisnik može da ubaci koje god hoće elemente iz prethodne tabele. Svako polje se može ponavljati više puta, da bi se opisali svi detalji. Neka polja se možda uopšte ne koriste, a druga se mogu ponavljati više puta.

Opšti oblik ubacivanja elementa iz Dublin Corea u HTML je:

```
<META NAME="DC.imeElementa" CONTENT="Vrednost elementa">
```

odnosno

```
<META NAME="DC.author" CONTENT="Miroslav Trajanovic">
```

▼ Poglavlje 2

XML

EXTENSIBLE MARKUP LANGUAGE (XML)

XML je skraćenica za eXtensible Markup Language (proširivi markup jezik). U pitanju je jezik koji se koristi za opis dokumenata i podataka na standardizovani način.

XML je skraćenica za eXtensible Markup Language(proširivi markup jezik). U pitanju je jezik koji se koristi za opis dokumenata i podataka na standardizovani način. Sve to se opisuje u tekstualnom obliku, tako da se lako može prenosi preko interneta. XML je, kao i HTML baziran na jeziku SGML(Standard Generalized Markup Language). SGML je značajan ne samo zato što su iz njega nastali svi savremeni markup jezici, već i zato što je to zvanični ISO standard.

Iako je HTML dobar za opis izgleda dokumenta na vebu, on ne omogućuje da se shvati značenje podataka sadržanih u dokumentu. Na primer, na HTML strani se može nalaziti lista sa cenama nekih proizvoda. Ti podaci se na HTML strani mogu prikazati na različite načine.

Kad se podaci jednom ubace u HTML i prikažu u pretraživaču, vrednosti podataka postaju deo markup jezika na strani. Više ne postoje pojedinačni podaci, već su to sada delovi sadržaja, ubaćeni između različitih HTML oznaka. Nema nekog standardnog načina da se sa strane izvade vrednosti gore pomenutih cena.

Sa eksplozijom weba, raste i potreba za univerzalnim formatom koji bi mogao da funkcioniše kao najmanji zajednički sadržalac za razmenu podataka, a da sve to radi preko popularnog i standardizovanog protokola, kao što je HTTP.

1996-e godine, World Wide Web Consortium (W3C) je formirao XML radnu grupu predvođenu Jon Bosakom, koja je pokušala da zadovolji ove potrebe. Tako je nastao XML, čija je prva verzija standardizovana 1998. godine. Trenutno je u upotrebi verzija XML 1.1 iz 2004. godine.

NA KOJIM OSNOVNIM ELEMENTIMA POČIVA XML?

Tri osnovna elementa XML-a na kojima počiva XML su:proširivost, struktura i ispravnost podataka

Tri osnovna elementa XML-a na kojima počiva XML su:

- **Proširivost.** XML opisuje struktuisane podatke kao tekst. Format je otvoren za proširenja. To znači da se bilo koji podaci koji se mogu tekstualno opisati i koji se mogu ubaciti u XML oznake, mogu prihvati kao XML. Proširenja jezika treba jedino da slede osnovnu sintaksu XML-a, a sve ostalo je stvar izbora. Jedina ograničenja su sami podaci, odnosno pravila za njihovu proveru, koja su takođe deo XML-a.
- **Struktura.** Struktura XML-a je obično toliko složena da bi čovek mogao da je lako prati. Treba imati na umu da XML dokumenti i nisu namenjeni za čitanje od strane ljudi. Postoje XML parseri i drugi alati koji lako izlaze na kraj sa XML-om. Pored toga XML je napravljen da bude otvoreni format za razmenu podataka, tako da je predstavljanje podataka u XML-u obično mnogo duže nego u originalnom formatu. XML nije napravljen da bi štedeo prostor na disku ili saobraćaj na mreži.
- **Ispravnost podataka.** Pored zahteva vezanih za sintaksu, podaci predstavljeni u XML dokumentu se mogu opcionalno proveravati u pogledu strukture i sadržaja. Tu postoje dva različita standarda za proveru. Jedan je DTD (**Document Type Definition**), a drugi XML Schema.

XML - VIDEO

XML Tutorial for Beginners | What is XML | Learn XML

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 2.1 ELEMENTI U XML-u

ELEMENTI U XML-U

XML dokumenti mogu da sadrže elemente, atribute i text.

Pošto je XML napravljen za opis podataka i dokumenata, preporuka W3C XML, je vrlo rigorozna u pogledu malog skupa zahteva koji razlikuju dokument sa gomilom oznaka i XML dokument. XML dokumenti koji odgovaraju standardu i preporukama vezanim za formatiranje su tzv. dobro-obrazovani dokumenti. Ovakvi dokumenti mogu da sadrže **elemente, atribute i tekst.**

Elementi

Elementi se prikazuju na sledeći način:

<element> </element>

Postoje nekoliko osnovnih pravila za elemente koji postoje u XML dokumentima. Imena elemenata mogu da sadrže slova, brojeve, crtice, podvlake, tačke i zareze. U imenu elementa ne sme da nalazi blanko karakter.

Imena elemenata mogu početi slovom, podvlakom ili dvotačkom, ali ne mogu početi brojem ili neki drugim karakterom koji nije iz alfabeta.

Pored ovih pravila treba voditi računa i o upotrebi crtica i tačaka u imenima. Iako je to u XML-u dozvoljeno, to može dovesti do problema u programskim jezicima ili bazama podataka u koje će ovi podaci otići.

XML može da ima samo jedan koren element.

Sledeći primer je neispravan

<x> ... </x> <y> ... </y>

Pravilno formatiranje bi bilo

<root> <x> ... </x> <y> ... </y> </root>

XML elementi mogu sadržati i druge elemente:



Slika 2.1.1 Primer XML dokumenta [Izvor: Autor]

Prazan element (eng. **empty element**) se može predstaviti na dva načina

1. <hr> </hr>
2. <hr />

ORGANIZACIJA XML ELEMENATA

XML dokument je struktuiran po principu stabla

XML dokument je struktuiran po principu stabla. XML dokument počinje sa korenim elementom (engl. **root element**), koji se dalje grana u svoje elemente decu (engl. **child elements**). Svaki element može dalje da se grana u nove elemente, odnosno da ima svoje "child" elemente.

Primer ove strukture je prikazan u sledećem XML kodu:

```
<root>
    <child>
        <subchild> . . . . </subchild>
    </child>
</root>
```

Pojmovi roditelj, dete i braća/sestre se koriste za opis odnosa između XML elemenata. Tako, roditelj ima decu, deca imaju roditelje, a braća i sestre su deca na istom nivou.

Element može da sadrži:

- tekst

- atribut
- drugi element
- kombinaciju svega navedenog.

U sledećem primeru **<naslov>**, **<autor>**, **<godina>** i **<cena>** sadrže tekst pošto sadrže tekst cene, kao što je 3000 i 4000.

<knjizara> i **<knjiga>** sadrže druge elemente.

<knjiga> sadrži atribut ha (kategorija="decija").

```
<knjizara>
  <knjiga kategorija="decija">
    <naslov> Hari Poter </naslov>
    <autor> J. K. Rowling </autor>
    <godina> 2005 </godina>
    <cena> 3000 </cena>
  </knjiga>
  <knjiga kategorija="web">
    <naslov> Ucenje XML </naslov>
    <autor> Erik T. Ray </autor>
    <godina> 2003 </godina>
    <cena> 4000 </cena>
  </knjiga>
</knjizara>
```

PRAVILA XML TAGOVA

XML tagovi su osetljivi na mala i velika slova; tagovi se moraju zatvarati pravilnim redom

1. XML tagovi su "case-sensitive"

<address> Ovo je pogrešna sintaksa **</Address>**
<address> Ovo je ispravna sintaksa **</address>**

2. Tagovi se moraju zatvarati pravilnim redom. Element koji se nalazi unutar drugog elementa se mora zatvoriti pre spoljašnjeg elementa

<outer_element> <internal_element>

Ovaj tag je zatvoren pre outer_element

</internal_element> </outer_element>

3. Vrednost XML atributa je potrebno da se nalazi među znake navoda

NEISPRAVNO (datum se ne nalazi između znaka navoda):

```
<note date=12/11/2007>
  <to>Tove</to>
  <from>Jani</from>
</note>
```

ISPRAVNO:

```
<note date="12/11/2007">
  <to>Tove</to>
  <from>Jani</from>
</note>
```

ATRIBUTI

Atributi sadrže vrednosti koje su vezane za neki element.

Atributi sadrže vrednosti koje su vezane za neki element i uvek su deo oznake otvaranje elementa:

<element atribut=" vrednost"> </element>

Element može imati više jedinstvenih atributa.

Atribut daje više informacija o XML-u. Tačnije oni definišu svojstva elemenata.

XML atribut ima sledeću sintaksu:

<element-name attribute1 attribute2 >

....sadržaj..

< /element-name>

Gde **attribute1** i **attribute2** imaju sledeću sintaksu

name = " value"

Vrednost **value** može biti između "" ili ''

Osnovna pravila koja važe za elemente se primenjuju i na attribute, sa još nekim dodacima. Ime atributa mora biti iza imena elementa, nakon čega sledi znak jednakosti (=), posle čega ide vrednost atributa pod jednostrukim ili dvostrukim navodnicima. Ako vrednost atributa sadrži neke navodnike, onda se za definisanje atributa koristi različiti tip navodnika, u odnosu na one koji su u sadržaju atributa.

TEKST

Tekst je obično vrednost podataka pridružen elementima i atributima

Tekst se postavlja između oznake otvaranja i zatvaranja elementa i obično predstavlja vrednost podatka, koji je **pridružen elementima i atributima, koji ga okružuju**.

<element atribut ="vrednost"> text </element>

Za tekst ne važe pravila koja smo pomenuli za elemente i attribute, tako da se tu može naći bilo koji tekst.

XML KOMENTARI

XML komentar počinje sa <!-- i završava sa -->

XML komentar počinje sa <!-- i završava sa -->, a između se može staviti tekst <!--Your comment-->

```
<?xml version="1.0" encoding="UTF-8" ?>
<!--Students grades are uploaded by months-->
<class_list>
<student>
<name> Tanmay </name>
<grade> A </grade>
</student>
</class_list>
```

Pravila za XML komentare:

- Komentari se ne mogu stavljati pre XML deklaracije
- Komentari se mogu naći bilo gde u dokumentu
- Komentari se ne mogu koristiti u vrednostima atributa
- Komentari se ne mogu ugnježdavati u druge komentare

▼ 2.2 STRUKTURA XML-a

STRUKTURA XML-A

Elementi, atributi i tekst vrlo važni za XML dokumente, ti objekti sami ne daju dobro-oblikovan XML dokument i potrebno je podvrgnuti ih određenim strukturnim i sintaktičkim pravilima.

Iako su elementi, atributi i tekst vrlo važni za XML dokumente, ti objekti sami ne daju dobro-oblikovan XML dokument. Dokument postaje takav tek ako se ovi elementi podvrgnu određenim strukturnim i sintaktičkim pravilima.

U sledećem primeru je dat jedan XML dokument:

```
<?xml version="1.0" encoding="UTF-8"?>
<korenielement>
    <prvielelement pozicija="1">
        <nivo1 dete="0">Ovo je prvi nivo ugneždenih elemenata </nivo1>
    </prvielelement >
    <drugielement pozicija ="2">
        <nivo1 dete="1">
            <nivo2> Ovo je drugi nivo ugneždenih elemenata </nivo2>
        </nivo1></drugielement>
    </korenielement>
```

Većina XML dokumenata počinje elementom <?xml?>. Ovo je tzv. deklaracija dokumenta. Tu se zadaje verzija, kao i način šifriranja karaktera (obično UTF-8). Verziju dokumenta koriste parseri kod analize dokumenata.

Posle ove deklaracije koja je opcionalna, sledi koren element. U našem primeru to je element:
<korenelement>

Unutar korenog elementa se mogu gnezdati drugi elementi, ali ovaj element mora biti prvi i mora biti jedinstven u celom dokumentu. Ovo se može uporediti sa diskom računara, na kojem postoji jedan vrhovni direktorijum, na kome se nalaze svi ostali direktorijumi i datoteke.

Nakon ovog elementa slede ugnezdeni elementi, atributi i tekst.

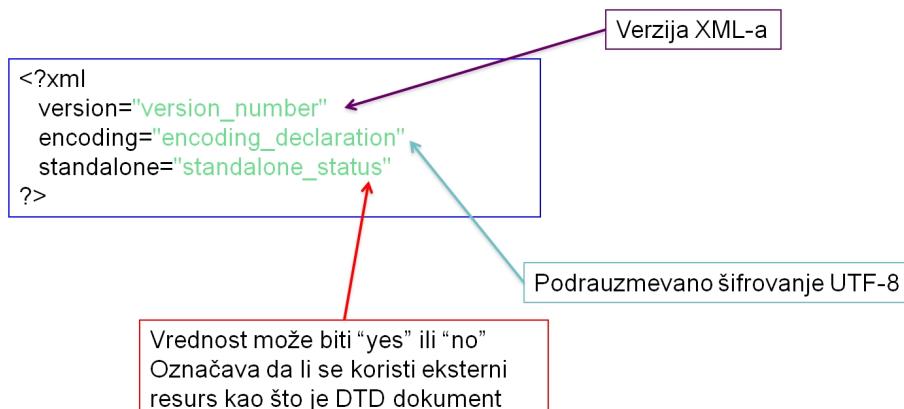
<prvielelement pozicija="1"> <nivo1 dete="0"> Ovo je prvi nivo ugnezdenih elemenata
</nivo1>
</prvielelement >

XML DEKLARACIJA

XML deklaracija sadrži: verziju, šifrovanje i eksterni resurs

XML deklaracija sadrži:

- Verziju XML-a
- Podrazumevano šifrovanje
- Naznaku da li se koristi eksterni resurs



Slika 2.2.1 Struktura XML deklaracije [Izvor: Autor]

Primer XML deklaracije:

Deklaracija bez parametara

<?xml >

Deklaracija sa definisanom verzijom

<?xml version="1.0">

Deklaracija sa definisanim svim parametrima

<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

PRIKAZ DEFINISANJA ELEMENATA UNUTAR XML

Prvi element ima atribut po imenu pozicija sa vrednošću 1. Unutar prvog elementa se ugnježdava nivo 1. Drugi element ispod korenog elementa je drugi element.

Prvi element ima atribut po imenu pozicija, sa vrednošću 1. Ovaj atribut daje dodatne podatke vezane za prvilement. U ovom slučaju podatak ukazuje na to da je pozicija ovog elementa kod sortiranja 1. Generalno, atributi se koriste za dodavanje više informacija i opisa vrednosti elemenata, kao i teksta koji je pridružen elementima.

Unutar prvog elementa je ugnezden element nivo1, koji ima atribut dete. Ime elementa smo upotrebili da opišemo nivo ugnezdanja, a vrednost atributa dete smo upotrebili da opišemo koliko drugih nivoa postoji unutar ovog (0). Fraza Ovo je prvi nivo ugnezdenih elemenata predstavlja tekstualni podatak, koji je deo elementa nivo1.

Drugi element ispod korenog elementa je drugielement. On liči na prvi element, ali je vrednost njegovog atributa pozicija 2. Unutar ovog elementa imamo drugi element nivo1. Odavde možemo da zaključimo da dobro oblikovan XML dokument može da ima više primeraka istog elementa. Jedini izuzetak je koreni element, koji mora biti jedinstven.

```
<drugielement pozicija = "2">  
<nivo1 dete="1">  
<nivo2> Ovo je drugi nivo ugnezdenih elemenata </nivo2>  
</nivo1>  
</drugielement >
```

Na kraju se XML dokument završava zatvaranjem oznake **<korenielement>**.

PRIMERI

Primeri XML za jelovnik i katalog biljaka

Primer XML jelovnika:

```
<?xml version="1.0" encoding="UTF-8"?>  
<breakfast_menu>  
<food>  
<name>Belgian Waffles</name>  
<price>$5.95</price>  
<description>Two of our famous Belgian Waffles with plenty of real maple  
syrup</description>  
<calories>650</calories>  
</food>  
<food>  
<name>Strawberry Belgian Waffles</name>
```

```
<price>$7.95</price>
<description>Light Belgian waffles covered with strawberries and whipped cream</description>
<calories>900</calories>
</food>
<food>
  <name>Berry-Berry Belgian Waffles</name>
  <price>$8.95</price>
  <description>Light Belgian waffles covered with an assortment of fresh berries and whipped cream</description>
  <calories>900</calories>
</food>
<food>
  <name>French Toast</name>
  <price>$4.50</price>
  <description>Thick slices made from our homemade sourdough bread</description>
  <calories>600</calories>
</food>
<food>
  <name>Homestyle Breakfast</name>
  <price>$6.95</price>
  <description>Two eggs, bacon or sausage, toast, and our ever-popular hash browns</description>
  <calories>950</calories>
</food>
</breakfast_menu>
```

Primer XML za katalog biljaka:

```
<?xml version="1.0" encoding="UTF-8"?>
<CATALOG>
  <PLANT>
    <COMMON>Bloodroot</COMMON>
    <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$2.44</PRICE>
    <AVAILABILITY>031599</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Columbine</COMMON>
    <BOTANICAL>Aquilegia canadensis</BOTANICAL>
    <ZONE>3</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.37</PRICE>
    <AVAILABILITY>030699</AVAILABILITY>
  </PLANT>
  <PLANT>
    <COMMON>Marsh Marigold</COMMON>
    <BOTANICAL>Caltha palustris</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Mostly Sunny</LIGHT>
    <PRICE>$6.81</PRICE>
```

```
<AVAILABILITY>051799</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Cowslip</COMMON>
<BOTANICAL>Caltha palustris</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$9.90</PRICE>
<AVAILABILITY>030699</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Dutchman's-Breeches</COMMON>
<BOTANICAL>Dicentra cucullaria</BOTANICAL>
<ZONE>3</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$6.44</PRICE>
<AVAILABILITY>012099</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Ginger, Wild</COMMON>
<BOTANICAL>Asarum canadense</BOTANICAL>
<ZONE>3</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$9.03</PRICE>
<AVAILABILITY>041899</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Hepatica</COMMON>
<BOTANICAL>Hepatica americana</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$4.45</PRICE>
<AVAILABILITY>012699</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Liverleaf</COMMON>
<BOTANICAL>Hepatica americana</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$3.99</PRICE>
<AVAILABILITY>010299</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Jack-In-The-Pulpit</COMMON>
<BOTANICAL>Arisaema triphyllum</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$3.23</PRICE>
<AVAILABILITY>020199</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Mayapple</COMMON>
<BOTANICAL>Podophyllum peltatum</BOTANICAL>
```

```
<ZONE>3</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$2.98</PRICE>
<AVAILABILITY>060599</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Phlox, Woodland</COMMON>
<BOTANICAL>Phlox divaricata</BOTANICAL>
<ZONE>3</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$2.80</PRICE>
<AVAILABILITY>012299</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Phlox, Blue</COMMON>
<BOTANICAL>Phlox divaricata</BOTANICAL>
<ZONE>3</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$5.59</PRICE>
<AVAILABILITY>021699</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Spring-Beauty</COMMON>
<BOTANICAL>Claytonia Virginica</BOTANICAL>
<ZONE>7</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$6.59</PRICE>
<AVAILABILITY>020199</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Trillium</COMMON>
<BOTANICAL>Trillium grandiflorum</BOTANICAL>
<ZONE>5</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$3.90</PRICE>
<AVAILABILITY>042999</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Wake Robin</COMMON>
<BOTANICAL>Trillium grandiflorum</BOTANICAL>
<ZONE>5</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$3.20</PRICE>
<AVAILABILITY>022199</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Violet, Dog-Tooth</COMMON>
<BOTANICAL>Erythronium americanum</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$9.04</PRICE>
<AVAILABILITY>020199</AVAILABILITY>
</PLANT>
```

```
<PLANT>
    <COMMON>Trout Lily</COMMON>
    <BOTANICAL>Erythronium americanum</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Shade</LIGHT>
    <PRICE>$6.94</PRICE>
    <AVAILABILITY>032499</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Adder's-Tongue</COMMON>
    <BOTANICAL>Erythronium americanum</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Shade</LIGHT>
    <PRICE>$9.58</PRICE>
    <AVAILABILITY>041399</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Anemone</COMMON>
    <BOTANICAL>Anemone blanda</BOTANICAL>
    <ZONE>6</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$8.86</PRICE>
    <AVAILABILITY>122698</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Grecian Windflower</COMMON>
    <BOTANICAL>Anemone blanda</BOTANICAL>
    <ZONE>6</ZONE>
    <LIGHT>Mostly Shady</LIGHT>
    <PRICE>$9.16</PRICE>
    <AVAILABILITY>071099</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Bee Balm</COMMON>
    <BOTANICAL>Monarda didyma</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Shade</LIGHT>
    <PRICE>$4.59</PRICE>
    <AVAILABILITY>050399</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Bergamot</COMMON>
    <BOTANICAL>Monarda didyma</BOTANICAL>
    <ZONE>4</ZONE>
    <LIGHT>Shade</LIGHT>
    <PRICE>$7.16</PRICE>
    <AVAILABILITY>042799</AVAILABILITY>
</PLANT>
<PLANT>
    <COMMON>Black-Eyed Susan</COMMON>
    <BOTANICAL>Rudbeckia hirta</BOTANICAL>
    <ZONE>Annual</ZONE>
    <LIGHT>Sunny</LIGHT>
```

```
<PRICE>$9.80</PRICE>
<AVAILABILITY>061899</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Buttercup</COMMON>
<BOTANICAL>Ranunculus</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$2.57</PRICE>
<AVAILABILITY>061099</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Crowfoot</COMMON>
<BOTANICAL>Ranunculus</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$9.34</PRICE>
<AVAILABILITY>040399</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Butterfly Weed</COMMON>
<BOTANICAL>Asclepias tuberosa</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Sunny</LIGHT>
<PRICE>$2.78</PRICE>
<AVAILABILITY>063099</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Cinquefoil</COMMON>
<BOTANICAL>Potentilla</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$7.06</PRICE>
<AVAILABILITY>052599</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Primrose</COMMON>
<BOTANICAL>Oenothera</BOTANICAL>
<ZONE>3 - 5</ZONE>
<LIGHT>Sunny</LIGHT>
<PRICE>$6.56</PRICE>
<AVAILABILITY>013099</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Gentian</COMMON>
<BOTANICAL>Gentiana</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$7.81</PRICE>
<AVAILABILITY>051899</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Blue Gentian</COMMON>
```

```
<BOTANICAL>Gentiana</BOTANICAL>
<ZONE>4</ZONE>
<LIGHT>Sun or Shade</LIGHT>
<PRICE>$8.56</PRICE>
<AVAILABILITY>050299</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Jacob's Ladder</COMMON>
<BOTANICAL>Polemonium caeruleum</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$9.26</PRICE>
<AVAILABILITY>022199</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Greek Valerian</COMMON>
<BOTANICAL>Polemonium caeruleum</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$4.36</PRICE>
<AVAILABILITY>071499</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>California Poppy</COMMON>
<BOTANICAL>Eschscholzia californica</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Sun</LIGHT>
<PRICE>$7.89</PRICE>
<AVAILABILITY>032799</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Shooting Star</COMMON>
<BOTANICAL>Dodecatheon</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Mostly Shady</LIGHT>
<PRICE>$8.60</PRICE>
<AVAILABILITY>051399</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Snakeroot</COMMON>
<BOTANICAL>Cimicifuga</BOTANICAL>
<ZONE>Annual</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$5.63</PRICE>
<AVAILABILITY>071199</AVAILABILITY>
</PLANT>
<PLANT>
<COMMON>Cardinal Flower</COMMON>
<BOTANICAL>Lobelia cardinalis</BOTANICAL>
<ZONE>2</ZONE>
<LIGHT>Shade</LIGHT>
<PRICE>$3.02</PRICE>
<AVAILABILITY>022299</AVAILABILITY>
```

```
</PLANT>  
</CATALOG>
```

XML - VIDEO

An Introduction to XML: The Basics Part 1

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

PROVERA ISPRAVNOSTI U XML DOKUMENTU

PRAVILA KOJA SE ODNOSE NA STRUKTURU I SINTAKSU XML

Ispavnost XML dokumenata se određuje na osnovu DTD (Document Type Definition) ili XML šeme

Kao što smo već pomenuli, postoje striktna pravila koja se odnose na osnovnu strukturu i sintaksu dobro-oblikovanih XML dokumenata.

Ispavnost XML dokumenata se određuje na osnovu DTD (Document Type Definition) ili XML šeme. Postoje još neki formati koji se mogu koristiti za proveru ispravnosti podataka. Lista tih formata se može pronaći na adresi <http://www.oasis-open.org/cover/schemas.html>. Ipak dva pomenuta formata se najviše koriste.

Provera XML dokumenata može da se vrši na osnovu pravila definisanih u DTD-u ili šemi. Dobrooblikovan XML dokument koji zadovoljava sve zahteve jedne ili druge specifikacije se naziva ispravnim dokumentom.

XML dokumenti nisu ispravni sami po sebi. Provera ispravnosti se dešava tokom analize (parsovanja) dokumenata. Većina parsera koji danas postoje ima ugrađene mogućnosti za proveru ispravnosti i obično podržavaju proveru na osnovu XML šeme ili DTD-a. Poneki od njih podržava i neki drugi tip provere ispravnosti. Pored toga, definisanjem vrednosti neke promenljive, ili pozivom druge klase u parseru, može se isključiti provera ispravnosti ili zanemariti direktive iz DTD-a ili XML šeme.

Na primer, NewML daje standardni format za pakovanje informacija sa vestima, u XML dokumentima. NewML definiše šta naslov, datum publikovanja, zaglavje, tekst članka i druge infomracije vezane za vest. NewML takođe definiše i kako elementi treba da budu postavljeni. NewML su dobro-oblikovani XML dokumenti, koji takođe odgovaraju NewML specifikaciji. Ovu specifikaciju publikuje i održava International Press Telecommunications Council (ITPC). ITPC je napravio DTD kojeg mogu da koriste oni koji prave vesti i da na osnovu njega proveravaju ispravnost svojih XML dokumenata. (Rojters i druge novinske agencije imaju svoje vesti koje su kompatibilne sa specifikacijom NewML.) Ako neke novine žele da prave vesti koje su kompatibilne sa NewML specifikacijom, oni mogu da preuzmu DTD sa adrese <http://www.iptc.org>. Nakon što se preuzme DTD programeri mogu da pomoću parsera proveravaju dokumente koje prave.

Naredni primer prikazuje jedan jednostavan XML dokument, sa dodatim referencama za DTD i XML šemu.

PRIMER PROVERE ISPRAVNOSTI U XML DOKUMENTU

Primer provere ispravnosti

Ispravna XML datoteka je dobro struktuirana XML datoteka, koja je napisana u skladu sa pravilima DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

DOCTYPE deklaracija, u gore navedenom primeru, se odnosi na eksternu DTD datoteku. Sadržaj datoteke je prikazan niže. Svrha DTD-a je da definiše strukturu XML dokumenta. Ona definiše strukturu sa spiskom dozvoljenih elemenata:

```
<!DOCTYPE note
[
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]
```

Prikazani primer DTD-a se tumači na sledeći način:

- **!DOCTYPE note** označava da je koren element **note**
- **!ELEMENT note** definiše da element mora da sadrži elemente: "to,from,heading,body"
- **!ELEMENT to** definišete da je element tipa "#PCDATA"
- **!ELEMENT from** definišete da je element tipa "#PCDATA"
- **!ELEMENT heading** definišete da je element tipa "#PCDATA"
- **!ELEMENT body** definišete da je element tipa "#PCDATA"

3.1 PROVERA ISPRAVNOSTI XML DOKUMENATA POMOĆU DTD-a

DTD

Document Type Definition (DTD) je prvobitni način za proveru ispravnosti XML dokumenta. Prikaz DTD dokumenta je dat.

Document Type Definition (DTD) je prvobitni način za proveru ispravnosti XML dokumenta. Iako položaj DTD-a iza XML deklaracije na početku dokumenta može da vas navede da je to XML dokument, DTD u suštini predstavlja XML dokument koji nije dobro oblikovan. Razlog je što se u ovim dokumentima poštuje DTD sintaksa, a ne XML sintaksa.

U prethodnom primeru je referenca na DTD postavljena u prvom elementu ispod XML deklaracije.

<!DOCTYPE rootelement SYSTEM “verysimplexml.dtd”>

Evo kako izgleda datoteka `verysimplex.dtd`, koja se koristi u prethodnom primeru:

Primer koji analiziramo na dalje:

```
<?xml version="1.0" encoding="UTF-8"?> <!ELEMENT korenielement (prvielelement,  
drugielement)>  
<!ELEMENT prvielelement (nivo1)>  
<!ATTLIST prvielelement  
pozicija CDATA #REQUIRED >  
<!ELEMENT nivo1 (#PCDATA | nivo2)*>  
<!ATTLIST nivo1  
dete (0 | 1) #REQUIRED >  
<!ELEMENT drugielement  
pozicija CDATA #REQUIRED >  
<!ELEMENT nivo2 (#PCDATA)>  
<!ELEMENT drugielement (nivo1)>
```

ANALIZA PRIMERA

Prikaz nivoa unutar DTD-a

Prvi red je deklaracija XML dokumenta, koja parseru govori koja je verzija XML-a i tip kodiranja podataka.

Naredni red definiše da ispravan XML dokument mora da sadrži elemente `prvielelement` i `drugielement`, koji moraju da budu ispod elementa `korenielement` i moraju da budu u zadatom redosledu.

<!ELEMENT korenielement (prvielelement, drugielement)>

Nakon toga ovaj DTD definiše atribute za element `prvielelement`. Ovaj element mora imati element `nivo1`, koji se nalazi direktno ispod njega.

`<!ELEMENT prvielelement (nivo1)>`

Posle toga se definišu atributi elementa `prvielelement`. Deklaracija ATTLIST govori da ispravan XML dokument treba da ima atribut pozicija za svaki primerak elementa `prvielelement` (#REQUIRED), kao i da su u pitanju obični podaci tipa karakter (CDATA).

`<!ATTLIST prvielelement pozicija CDATA #REQUIRED >`

Naredna deklaracija govori da element `nivo1` može da sadrži jednu ili dve stvari. Znak | je u DTD-u ekvivalentan sa logičkim ili. Element `nivo1` može da sadrži drugi ugnezđeni element, po imenu `nivo2`, ili vrednost nekog podataka tipa karakter (PCDATA).

`<!ELEMENT nivo1 (#PCDATA | nivo2)*>`

Naredna deklaracija ATTLIST govori da element `nivo1` ima atribut `dete`, koji može da ima jednu od dve vrednost, 0 ili 1.

`<!ATTLIST nivo1`

`dete (0 | 1) #REQUIRED>`

Ostale deklaracije liče na ove koje smo već opisali.

Kao što vidite na osnovu ovog DTD-a, deklaracije elemenata i atributa ne moraju biti u istom redosledu kao elementi i atributi koje predstavljaju. Na parseru je da tako presloži DTD da se vide veze između elemenata, kao i da osigura da su primenjena sva pravila, definisana u DTD-u.

✓ 3.2 PROVERA ISPRAVNOSTI XML DOKUMENTA POMOĆU XML ŠEMA

PROVERA ISPRAVNOSTI XML DOKUMENTA UZ POMOĆU XML ŠEMA

W3C Shema je zvanično odobrena definicija šeme. Za razliku od DTD-a, format XML šeme odgovara pravilima koja važe za dobro-oblikovane XML dokumente.

W3C Shema je zvanično odobrena definicija šeme. Za razliku od DTD-a, format XML šeme odgovara pravilima koja važe za dobro-oblikovane XML dokumente. Šema (**Schema**) dozvoljava mnogo finiju kontrolu nad podacima koji se opisuju.

Usled formata koji se koristi za XML dokumente, kao i kontrole nad elementima u njemu, često se dešava da su šeme mnogo složenije i duže nego XML dokumenti koje opisuju. Ipak, šeme su za programere mnogo jednostavnije za čitanje, u odnosu na DTD.

Reference na šeme se prave pomoću kreiranja prostora imena XMLSchemaInstance. U prethodnom primeru smo koristili deklaraciju šeme

```
<korenielement xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xsi:noNamespaceSchemaLocation="verysimplexml.xsd">
```

U ovom slučaju deklaracija prostora imena ukazuje na <http://www.w3.org/2001/XMLSchemainstance>. Na ovoj adresi se nalazi dokument, sa kratkim opisom načina na koji treba da se korsiti šema. Vrednost noNamespaceSchemaLocation nam govori da za šemu ne postoji unapred definisani prostor imena. To znači da sve elemente u XML dokumentu treba proveravati samo na osnovu šeme. Šema se nalazi u dokumentu `verysimplex.xsd`. Pošto nije zadata nikakva putanja, to ova datoteka treba da se nalazi u istom direktorijumu kao i XML datoteka koja se proverava. Pomoću atributa schemaLocation možete definisati lokacije šeme. Isti atribut se koristi i za definisanje prostora imena, ali u tom slučaju morate da zadate prostor imena koji odgovara tom atributu. Deklaracija se zadaje pre upotrebe atributa schemaLocation. Evo kako to izgleda:

```
<rootelement
xmlns:fe="http://www.benztech.com/schemas/verybasic"
xsi:schemaLocation="http://www.benztech.com/schemas/verybasic ">
```

XML ŠEMA

XML šema opisuje strukturu XML dokumenta isto kao i DTD

XML šema opisuje strukturu XML dokumenta isto kao i DTD.

XML šeme imaju više prednosti u odnosu na DTD:

- XML šeme su napisane u XML-u
- XML sheme su proširive
- SML šeme podržavaju tipove podataka.

Primer XML šeme:

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

Ovu šemu možemo interpretirati na sledeći način:

- <xs:element name="note"> definiše element pod nazivom "note"
- <xs:complexType> element "note" je tipa complex
- <xs:sequence> tip complex predstavlja niz elemenata
- <xs:element name="to" type="xs:string"> element "to" je tipa string (tekst)
- <xs:element name="from" type="xs:string"> element "from" je tipa string
- <xs:element name="heading" type="xs:string"> element "heading" je tipa string
- <xs:element name="body" type="xs:string"> element "body" je tipa string

Zašto koristiti XML šemu?

- Sa XML šemom, XML datoteka može da sadrži opis sopstvenog formata.
- Sa XML šemom, nezavisne grupe ljudi mogu da se dogovore o standardima za razmenu podataka.
- Sa XML šemom se mogu proveriti podaci.

PRIMER XSD DOKUMENTA

Izgled verysimplex.xsd

Datoteka verysimplex.xsd izgleda ovako:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
<xs:element name="prvielelement">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="nivo1"/>
    </xs:sequence>
    <xs:attribute name="pozicija" type="xs:boolean"    use="required"/>
  </xs:complexType>
  </xs:element>
  <xs:element name="nivo1">
    <xs:complexType mixed="true">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="nivo2"/>
      </xs:choice>
      <xs:attribute name="dete" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:NMTOKEN">
            <xs:enumeration value="0"/>
            <xs:enumeration value="1"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="nivo2" type="xs:string"/>
<xs:element name="korenielelement">
  <xs:complexType>
```

```
<xs:sequence>
    <xs:element ref="prvielelement"/>
    <xs:element ref="drugielement"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="drugielement">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="nivo1"/>
    </xs:sequence>
    <xs:attribute name="pozicija" type="xs:byte" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

KREIRANJE DEFINICIJE ZA PRVI ELEMENT

Definicija za prvi element ga opisuje kao složeni tip podatka. Taj element sadrži jedan ugnezdeni element, nivo1, kao i atribut pozicija. Taj atribut je obavezan.

Odmah iza deklaracije se zadaje prostor imena xs. Za dati URL , <http://www.w3.org/2001/XMLSchema>, ukazuje na web sajt W3C, gde postoji dokumentacija za XML šemu.

Definicija za prvi element ga opisuje kao složeni tip podatka. Taj element sadrži jedan ugnezdeni element, nivo1, kao i atribut pozicija. Taj atribut je obavezan.

```
<xs:element name="prvielelement">
<xs:complexType>
    <xs:sequence>
        <xs:element ref="nivo1"/>
    </xs:sequence>
    <xs:attribute name="pozicija" type="xs:boolean" use="required"/>
</xs:complexType>
</xs:element>
```

OBJAŠNJAVANJE NIVOA "NIVO1"

Prikaz objašnjavanja nivoa

Naredni element je nivo1. U pitanju je opcionalni element (minOccurs="0") i taj element se u dokumentu može pojaviti neograničeni broj puta (maxOccurs="unbounded"). U ovom elementu je ugnezden element nivo2. Zatim je definisan obavezan atribut dete. Tip ovog atributa je NMOKEN, što je u ovom slučaju string. Ovaj atribut može imati samo jednu od dve vrednosti 0 i 1. Ovo je definisano nabranjem (**enumeration**).

```
<xss:element name="nivo1">
<xss:complexType mixed="true">
<xss:choice minOccurs="0" maxOccurs="unbounded">
<xss:element ref="nivo2"/>
</xss:choice>
<xss:attribute name="dete" use="required">
<xss:simpleType>
<xss:restriction base="xs:NMTOKEN">
<xss:enumeration value="0"/>
<xss:enumeration value="1"/>
</xss:restriction>
</xss:simpleType>
</xss:attribute>
</xss:complexType>
</xss:element>
```

Kako element nivo2 nema atribut i ugnezdeni elemente, to se on može opisati u jednom redu i onda se upotrebiti kao referenca, pomoću iskaza ref=.

```
<xss:element name="nivo2" type="xs:string"/>
```

```
<xss:element ref="nivo2"/>
```

Kao i kod DTD-a deklaracije elemenata i atributa u šemi ne moraju da budu u istom redosledu kao elementi i atributi u XML dokumentu. Zadatak parsera je da presloži šemu u nešto što definiše veze između elemenata, bez obzira na redosled.

Naredni element je koren element. Ovaj element u sebi mora imati prvelement i drugielement. Samo onda je to ispravan element.

```
<xss:element name="korenielement">
<xss:complexType>
<xss:sequence>
<xss:element ref="prvelement"/>
<xss:element ref="drugielement"/>
</xss:sequence>
</xss:complexType>
</xss:element>
```

OBJASNJAVANJE NIVOA "DRUGIELEMENT"

Prikaz koda narednog elementa koji je drugi element, koji mora da sadrži ugnezdeni elementi nivo1.

Naredni element je drugi element, koji mora da sadrži ugnezdeni elementi nivo1. Tu je i atribut pozicija, čiji je tip ovog puta byte.

```
<xss:element name="drugielement">
<xss:complexType>
<xss:sequence>
```

```
<xs:element ref="nivo1"/>
</xs:sequence>
<xs:attribute name="pozicija" type="xs:byte" use="required"/>
</xs:complexType>
</xs:element>
```

Na kraju se zatvara šema:

```
</xs:schema>
```

Na kraju se zatvara šema:

```
</xs:schema>
```

▼ Poglavlje 4

OSNOVNI KONCEPTI RAŠČLANJIVANJA PARSOVANJA XML DOKUMENTA

TIPOVI PARSOVANJA

Dva tipova parsovajna su DOM i SAX

Jedna prednosti podataka definisanih u XML-u je njihova prenosivost. XML sam po sebi ne znači integraciju podataka. Aplikacijama koje šalju i primaju podatke preko XML-a je potreban interfejs da bi napravile XML i da bi te podatke integrisale u postojeće aplikacije.

U lingvistici parsovanje predstavlja razlaganje rečenica, tako da se mogu uspostaviti veze i struktura jezika. Ove strukture se najčešće predstavljaju u obliku stabla. Parsovanje putem računara je slično, ali se najčešće koristi za razlaganje i interpretaciju karaktera u nizovima karaktera. Pošto je XML po definiciji skup karaktera u nizu karaktera, to se i razlaganje i odvajanje pojedinih delova XML dokumenata, takođe naziva parsovanjem.

Parsovanjem XML dokumenata se identifikuju i konvertuju elementi koji se nalaze u tom dokumentu. Tom prilikom ili nastaje struktura stabla, ili događaji u zavisnosti od toga koji se tip parsera koristi.

Postoje dva tipa parsovanja:

- **Document Object Model** (DOM). Kod ovakvog parsovanja se dokument razlaže na ugnezđene elemente, koji se u DOM terminologiji nazivaju čvorovima. Čvorovi iz DOM ukazuju na dokumente ili delove dokumenta, elemente, atribute, tekstualne podatke, instrukcije za obradu, komentare, kao i druge tipove podataka koji se mogu pojaviti u XML dokumentu.
- **Simple API for XML** (SAX). Kod ovakvog parsovanja se XML dokument razlaže na događaje. Kada se jednom pronađu čvorovi i događaji, oni se koriste za konverziju elemenata iz XML-a u druge tipove podataka.

DOM I SAX - VIDEO

DOM & SAX Parsers

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

❖ 4.1 DOCUMENT OBJECT MODEL (DOM)

UVOD U DOM

DOM se može koristiti za kreiranje XML dokumenata, za kretanje kroz njegove strukture, kao i za ubacivanje, promenu i brisanje čvorova.

Jedini način parsovanja XML dokumenata koji je zvanična preporuka W3C konzorcijuma je DOM (objektni model dokumenta). DOM se može koristiti za kreiranje XML dokumenata, za kretanje kroz njegove strukture, kao i za ubacivanje, promenu i brisanje čvorova. Ovakav način parsovanja može biti sporiji nego ako se koristi SAX parser, pošto DOM u memoriji pravi prezentaciju celog dokumenta, bez obzira na veličinu tog dokumenta. Sa druge strane DOM je koristan ako treba iz dokumenta izvaditi neki podataka više puta, ili ako treba izvaditi sve elemente. DOM ostaje u memoriji sve dok kod koji je napravio model radi.

STRUKTURA DOM-A

Svaki čvor iz DOM-a predstavlja odgovarajuću stavku iz originalnog XML dokumenta. Čvor ima svoj tip (nodeType), svoje ime (nodeName) i svoju vrednost (nodeValue).

DOM je u suštini predstavljanje XML podataka u obliku stabla. Koreni i svi ostali elementi i atributi iz XML dokumenta su predstavljeni kao čvorovi u okviru jednog čvora celog dokumenta. Svaki čvor iz DOM-a predstavlja odgovarajuću stavku iz originalnog XML dokumenta. Elementi, atributi i tekstualni elementi se gnezde na isti način kao u originalnom XML dokumentu. Koreni čvor DOM modela uvek odgovara korenom elementu XML dokumenta. Ostali čvorovi se smeštaju u odnosu na koren element.

Čvorovi u DOM-u predstavljaju sve tipove podataka iz XML dokumenta. Čvor ima svoj tip (nodeType), svoje ime (nodeName) i svoju vrednost (nodeValue). Na primer, ako bismo dokument koji smo dali u prethodnom tekstu parsovali pomoću DOM parsera, onda bi postojao koren element, sa nodeName korenElement, a taj element je tipa nodeType. Prvi element je takođe tipa nodeType. Oba elementa imaju nodeValue null, što važi za sve elemente. Atribut pozicija zatim postaje čvor tipa attribute, pri čemu je nodeName pozicija. Vrednost je 1.

Tekstualna vrednost elementa nivo1 ima nodeName #text,.nodeType takođe text, a nodeValue je Ovo je prvi nivo ugnezdenih elemenata.

DOM i XML strukture se vrlo lako mogu vizuelno prikazati pomoću alata koji postoje. Neki od alata su na primer, Microsoft XML Notepad, ili IBM XML Viewer.

DOM 1, DOM 2 I DOM 3

DOM 1 podržava osnovnu navigaciju i uređivanje DOM čvorova. DOM 2 proširuje DOM 1 podrškom prostora imena.

DOM Level1 (nivo 1) i Level 2 su specifikacije koje su zvanično odobrene od strane W3C konzorcijuma. Obe specifikacije su završene i programeri koji prave aplikacije na osnovu ovih standarda mogu biti sigurni da su standardi kompletni i da se neće više menjati. Sa druge strane DOM Level 1 nije kompatibilan sa DOM Level 2. Nema ni garancije da će DOM Level 3 biti kompatibilan sa prethodnim verzijama.

DOM 1 podržava osnovnu navigaciju i uređivanje DOM čvorova u XML i HTML dokumentima. DOM 2 proširuje DOM 1 podrškom prostora imena, kao i novim mogućnostima kao što su filtrirani pogledi ili događaji.

HTML DOM I XML DOM

HTML DOM definiše standardni način pristupa i manipulaciji HTML dokumenata

HTML DOM definiše standardni način pristupa i manipulaciji HTML dokumenata. On predstavlja HTML dokument kao strukturu drveta. Svim HTML elementima se može pristupiti putem HTML DOM-a.

Ovaj primer menja vrednost HTML elementa čiji je id = "demo":

```
<!DOCTYPE html>
<html>
<body>

<h1 id="demo">This is a Heading</h1>

<button type="button"
onclick="document.getElementById('demo').innerHTML = 'Hello World!'">Click Me!
</button>

</body>
</html>
```

XML DOM definiše standardni način pristupa i manipulaciji XML dokumenata. On predstavlja XML dokument kao strukturu drveta. Svim XML elementima se može pristupiti putem XML DOM-a.

```
<knjizara>
  <knjiga kategorija="decija">
    <naslov> Hari Poter </naslov>
    <autor> J K. Rowling </autor>
    <godina> 2005 </godina>
```

```
<cena> 3000 </cena>
</knjiga>
<knjiga kategorija="web">
<naslov> Ucenje XML </naslov>
<autor> Erik T. Ray </autor>
<godina> 2003 </godina>
<cena>4000</cena>
</knjiga>
</knjizara>
```

Ovaj kod preuzima tekstualnu vrednost elementa `<naslov>` u XML dokumentu:

```
txt = xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```

▼ 4.2 SIMPLE API FOR XML (SAX)

SIMPLE API FOR XML (SAX)

SAX je skraćenica za jednostavni API za XML. SAX se koristi za opis, parsovanje i manipulaciju XML dokumentima.

SAX je skraćenica za jednostavni API za XML. Ovakvo parsovanje je brže u odnosu na DOM, ali je malo komplikovanije za upotrebu. Predstavljanje XML dokumenata u SAX-u ne prati strukturu čvorova, kao što je slučaj kod DOM-a. SAX parsovanje više liči na postupak kada želite da iz poglavlja neke knjige izvadite neke informacije, tako što ćete otici na početak tog poglavlja, čitati poglavlje i stati tamo gde poglavlje završava. DOM parser bi celu knjigu prebacio u DOM format. Drugim rečima, SAX obezbeđuje konkretnе delove informacija koje su Vam potrebne, dok DOM vraća i reformatira ceo dokument, a onda iz takvog dokumenta vadi istu informaciju.

Kao i DOM i SAX se koristi za opis, parsovanje i manipulaciju XML dokumentima. Za razliku od DOMa SAX dokument razlaže na niz događaja, koji predstavljaju delove dokumenta. To su događaji tipa `StartDocument`, `StartElement`, `EndElement`, `ProcessingInstruction`, `SAXWarning`, `SAXError` i `EndDocument`. SAX nije preporučen od strane W3C, iako DOM pozajmljuje neke ideje iz naprednijih SAX modela. Za SAX ne postoji zvanična dokumentacija, a jedina trenutna implementacija je pisana u Javi.

▼ Poglavlje 5

XSL transformacije

EXTENSIBLE STYLESHEET LANGUAGE TRANSFORMACIJE

XSL je skraćenica za Extensible Stylesheet Language. XSLT je napravljen na osnovu jezika DSSSL (Document Style Semantics and Specification Language).

XSL je skraćenica za **Extensible Stylesheet Language**. XSL specifikacija (u vidu preporuke) opisuje proces primene XSL dokumenta na XML dokument, uz primenu neke mašine za transformaciju. Takođe je opisan i XSL jezik. XSLT je napravljen na osnovu jezika DSSSL (**Document Style Semantics and Specification Language**), čija je namena bila definisanje formatiranje izlaza iz SGML dokumenta.

U osnovi XSLT se odnosi na definisanje pravila pomoću kojih se jedan XML dokument može transformisati u drugi XML dokument. XSLT dokument (**stylesheet**) sadrži pravila šablonu. Svako pravilo ima obrazac i šablon. XSLT procesor poređi elemente i ostale čvorove u ulaznom XML dokumentu sa obrascima u XSL dokumentu. Kada se pronađe odgovarajući on šablon iz tog pravila upisuje u izlazno stablo. Kada se ceo proces završi, procesor može dalje da izlazno stablo pretvori u neki XML dokument ili HTML ili neki drugi tip datoteke.

Struktura datoteke sa stilom (**stylesheet**) i njena sintaksa su definisani od strane W3C konzorcijuma. Mašine za transformaciju se prave na osnovu te definicije. Mašine za transformaciju podržavaju različite jezike, obično onaj jezik u kome su i napisane. I pored široke lepeze XSLT mašina, postoje dva osnovna pravca. To su one koje prate i podržavaju Java programski jezik i one koje prate Microsoft.

Jedna od prvih mašina za transformaciju za Javu je bila LotusXSL mašina, koju je IBM poklonio Apache grupi, gde je ona postala Xalan Transformation Engine. Od tada je Apache grupa razvila i verziju Xalan 2, koja ima integrisane SAX i DOM parsere.

Microsoftova podrška za XML 1.0 i XSLT je počela sa Internet Explorerom 5, koji je podržavao i DOM prostore imena iz XML-a kao i XML šeme. Ova podrška je tada bila samo delimično implementirana. Podrška je kasnije proširena i odvojena od čitača i postavljena u tzv. MSXML parser.

KAKO RADE XSL TRANSFORMACIJE?

Transformacije se izvode pomoću XSL procesora. Načini obavljanja transformacija su: eksplisitna referenca na XSL, programerska referenca na XSL datoteku, ugrađivanje XML-a u X

Transformacije se, kao što je već pomenuto, izvode pomoću XSL procesora. To je program čiji je zadatak da na osnovu XML datoteke, XSL dokumenta, definisanog načina transformacije napravi izlaz. XSL procesor čita XML dokument i obavlja transformacije nad atributima, elementima i tekstrom, na bazi instrukcija iz XSL datoteke.

XSL datoteke su dobro-oblikovani XML dokumenti, koji su u skladu sa W3C standardom. Izlazni format je takođe definisan u okviru XSL datoteke i može biti HTML, tekst ili XML.

Instrukcije za transformaciju izvornog XML dokumenta u izlazni XMI dokument se nalaze u XSL datotekama.

Generalno postoje tri načina na koji se može obaviti transformacija.

- **Eksplisitna referenca na XSL:** U XML dokumentu se može zadati eksplisitna referenca na XSL datoteku. Ovakav način rada se najviše koristi kada se želi razdvajanje podataka u XML dokumentima od načina prikazivanja, koji je definisan u XSL datoteci. XML se obično na serveru transformiše u HTML, pre nego što se taj HTML pošalje do korisnika.
- **Programska referenca na XSL datoteku:** Programi mogu da deklarišu ulaznu XML datoteku, XSL datoteku, kao i izlaznu datoteku, a onda se poziva XSLT procesor koji obavlja transformaciju. Ova tehnika se koristi na serverima za odvajanje podataka iz XML dokumenata od karakteristika za prikazivanje (HTML), i to kod servera koji koriste XML. Obično jedna XSL datoteka kontroliše više XML dokumenata. Ovo je takođe način na koji se odvijaju transformacije iz XML formata u drugi XML format.
- **Ugrađivanje XML-a u XSL datoteku:** XML podaci se mogu ugraditi u XSL dokument. Ovo se ne preporučuje iz istih razloga iz kojih se ne preporučuje ni ugrađivanje DTD-a u dokument. Ako se koristi ovakav način, vrlo je teško održavati takav sistem.

OSNOVNI ELEMENTI XSL-A

Osnovni element XSL-a su direktive i funkcije

Osnovni element XSL-a su direktive i funkcije. Način funkcionisanja je najbolje objasniti na primeru. Izvorni XML dokument koji će se transformisati izgleda ovako:

Programska referenca na XSL datoteku: Programi mogu da deklarišu ulaznu XML datoteku, XSL datoteku, kao i izlaznu datoteku, a onda se poziva XSLT procesor koji obavlja transformaciju. Ova tehnika se koristi na serverima za odvajanje podataka iz XML dokumenata od karakteristika za prikazivanje (HTML), i to kod servera koji koriste XML. Obično jedna XSL datoteka kontroliše više XML dokumenata. Ovo je takođe način na koji se odvijaju transformacije iz XML formata u drugi XML format.

Ugrađivanje XML-a u XSL datoteku: XML podaci se mogu ugraditi u XSL dokument. Ovo se ne preporučuje iz istih razloga iz kojih se ne preporučuje ni ugrađivanje DTD-a u dokument. Ako se koristi ovakav način, vrlo je teško održavati takav sistem.

XSLT

XSLT je skraćenica za XSL transformaciju.

XSL (eXtensible Stylesheet Language) je jezik koji određuje stil za XML.

XSLT je skraćenica za XSL transformaciju. XSLT transformiše jedan XML dokument u drugi XML dokument. XSLT primer koji daje sledeći prikaz

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tyler
Greatest Hits	Dolly Parton
Still got the blues	Gary Moore
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T'Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen

Slika 5.1.1 Prikaz XSLT primera [Izvor: https://www.w3schools.com/xml/tryxslt.asp?xmlfile=cdcatalog&xsltfile=cdcatalog_choose]

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
```

```
<price>9.90</price>
<year>1988</year>
</cd>
<cd>
<title>Greatest Hits</title>
<artist>Dolly Parton</artist>
<country>USA</country>
<company>RCA</company>
<price>9.90</price>
<year>1982</year>
</cd>
<cd>
<title>Still got the blues</title>
<artist>Gary Moore</artist>
<country>UK</country>
<company>Virgin records</company>
<price>10.20</price>
<year>1990</year>
</cd>
<cd>
<title>Eros</title>
<artist>Eros Ramazzotti</artist>
<country>EU</country>
<company>BMG</company>
<price>9.90</price>
<year>1997</year>
</cd>
<cd>
<title>One night only</title>
<artist>Bee Gees</artist>
<country>UK</country>
<company>Polydor</company>
<price>10.90</price>
<year>1998</year>
</cd>
<cd>
<title>Sylvias Mother</title>
<artist>Dr. Hook</artist>
<country>UK</country>
<company>CBS</company>
<price>8.10</price>
<year>1973</year>
</cd>
<cd>
<title>Maggie May</title>
<artist>Rod Stewart</artist>
<country>UK</country>
<company>Pickwick</company>
<price>8.50</price>
<year>1990</year>
</cd>
<cd>
<title>Romanza</title>
```

```
<artist>Andrea Bocelli</artist>
<country>EU</country>
<company>Polydor</company>
<price>10.80</price>
<year>1996</year>
</cd>
<cd>
    <title>When a man loves a woman</title>
    <artist>Percy Sledge</artist>
    <country>USA</country>
    <company>Atlantic</company>
    <price>8.70</price>
    <year>1987</year>
</cd>
<cd>
    <title>Black angel</title>
    <artist>Savage Rose</artist>
    <country>EU</country>
    <company>Mega</company>
    <price>10.90</price>
    <year>1995</year>
</cd>
<cd>
    <title>1999 Grammy Nominees</title>
    <artist>Many</artist>
    <country>USA</country>
    <company>Grammy</company>
    <price>10.20</price>
    <year>1999</year>
</cd>
<cd>
    <title>For the good times</title>
    <artist>Kenny Rogers</artist>
    <country>UK</country>
    <company>Mucik Master</company>
    <price>8.70</price>
    <year>1995</year>
</cd>
<cd>
    <title>Big Willie style</title>
    <artist>Will Smith</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>9.90</price>
    <year>1997</year>
</cd>
<cd>
    <title>Tupelo Honey</title>
    <artist>Van Morrison</artist>
    <country>UK</country>
    <company>Polydor</company>
    <price>8.20</price>
    <year>1971</year>
```

```
</cd>
<cd>
    <title>Soulsville</title>
    <artist>Jorn Hoel</artist>
    <country>Norway</country>
    <company>WEA</company>
    <price>7.90</price>
    <year>1996</year>
</cd>
<cd>
    <title>The very best of</title>
    <artist>Cat Stevens</artist>
    <country>UK</country>
    <company>Island</company>
    <price>8.90</price>
    <year>1990</year>
</cd>
<cd>
    <title>Stop</title>
    <artist>Sam Brown</artist>
    <country>UK</country>
    <company>A and M</company>
    <price>8.90</price>
    <year>1988</year>
</cd>
<cd>
    <title>Bridge of Spies</title>
    <artist>T` Pau</artist>
    <country>UK</country>
    <company>Siren</company>
    <price>7.90</price>
    <year>1987</year>
</cd>
<cd>
    <title>Private Dancer</title>
    <artist>Tina Turner</artist>
    <country>UK</country>
    <company>Capitol</company>
    <price>8.90</price>
    <year>1983</year>
</cd>
<cd>
    <title>Midt om natten</title>
    <artist>Kim Larsen</artist>
    <country>EU</country>
    <company>Medley</company>
    <price>7.80</price>
    <year>1983</year>
</cd>
<cd>
    <title>Pavarotti Gala Concert</title>
    <artist>Luciano Pavarotti</artist>
    <country>UK</country>
```

```
<company>DECCA</company>
<price>9.90</price>
<year>1991</year>
</cd>
<cd>
    <title>The dock of the bay</title>
    <artist>Otis Redding</artist>
    <country>USA</country>
    <company>Stax Records</company>
    <price>7.90</price>
    <year>1968</year>
</cd>
<cd>
    <title>Picture book</title>
    <artist>Simply Red</artist>
    <country>EU</country>
    <company>Elektra</company>
    <price>7.20</price>
    <year>1985</year>
</cd>
<cd>
    <title>Red</title>
    <artist>The Communards</artist>
    <country>UK</country>
    <company>London</company>
    <price>7.80</price>
    <year>1987</year>
</cd>
<cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <country>USA</country>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
</cd>
</catalog>
```

XSL(T) JEZICI

XSLT je jezik za transformaciju XML dokumenata.

XSLT je jezik za transformaciju XML dokumenata.

XPath je jezik za navigaciju kroz XML dokumente.

XQuery je jezik za upite u XML dokumentima.

XSLT koristi XPath da bi pronašao informacije u XML dokumentu. XPath se koristi za navigaciju kroz elemente i atributе u XML dokumentima. U procesu transformacije, XSLT koristi XPath da definiše delove dokumenta koji treba da odgovaraju jednoj ili više unapred definisanih

šablonu. Kada se pronađe podudarnost, XSLT će transformisati odgovarajući deo dokumenta u novi dokument.

PRIMER TRANSFORMACIJE

XSL stzle sheet referencu treba dodati u XML dokument

Želimo da konvertujemo sledeći XML dokument ("cdcatalog.xml") u XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
    <country>UK</country>
    <company>CBS Records</company>
    <price>9.90</price>
    <year>1988</year>
  </cd>
  <cd>
    <title>Greatest Hits</title>
    <artist>Dolly Parton</artist>
    <country>USA</country>
    <company>RCA</company>
    <price>9.90</price>
    <year>1982</year>
  </cd>
  <cd>
    <title>Still got the blues</title>
    <artist>Gary Moore</artist>
    <country>UK</country>
    <company>Virgin records</company>
    <price>10.20</price>
    <year>1990</year>
  </cd>
  <cd>
    <title>Eros</title>
    <artist>Eros Ramazzotti</artist>
    <country>EU</country>
    <company>BMG</company>
    <price>9.90</price>
    <year>1997</year>
  </cd>
```

```
<cd>
    <title>One night only</title>
    <artist>Bee Gees</artist>
    <country>UK</country>
    <company>Polydor</company>
    <price>10.90</price>
    <year>1998</year>
</cd>
<cd>
    <title>Sylvias Mother</title>
    <artist>Dr. Hook</artist>
    <country>UK</country>
    <company>CBS</company>
    <price>8.10</price>
    <year>1973</year>
</cd>
<cd>
    <title>Maggie May</title>
    <artist>Rod Stewart</artist>
    <country>UK</country>
    <company>Pickwick</company>
    <price>8.50</price>
    <year>1990</year>
</cd>
<cd>
    <title>Romanza</title>
    <artist>Andrea Bocelli</artist>
    <country>EU</country>
    <company>Polydor</company>
    <price>10.80</price>
    <year>1996</year>
</cd>
<cd>
    <title>When a man loves a woman</title>
    <artist>Percy Sledge</artist>
    <country>USA</country>
    <company>Atlantic</company>
    <price>8.70</price>
    <year>1987</year>
</cd>
<cd>
    <title>Black angel</title>
    <artist>Savage Rose</artist>
    <country>EU</country>
    <company>Mega</company>
    <price>10.90</price>
    <year>1995</year>
</cd>
<cd>
    <title>1999 Grammy Nominees</title>
    <artist>Many</artist>
    <country>USA</country>
    <company>Grammy</company>
```

```
<price>10.20</price>
<year>1999</year>
</cd>
<cd>
<title>For the good times</title>
<artist>Kenny Rogers</artist>
<country>UK</country>
<company>Mucik Master</company>
<price>8.70</price>
<year>1995</year>
</cd>
<cd>
<title>Big Willie style</title>
<artist>Will Smith</artist>
<country>USA</country>
<company>Columbia</company>
<price>9.90</price>
<year>1997</year>
</cd>
<cd>
<title>Tupelo Honey</title>
<artist>Van Morrison</artist>
<country>UK</country>
<company>Polydor</company>
<price>8.20</price>
<year>1971</year>
</cd>
<cd>
<title>Soulsville</title>
<artist>Jorn Hoel</artist>
<country>Norway</country>
<company>WEA</company>
<price>7.90</price>
<year>1996</year>
</cd>
<cd>
<title>The very best of</title>
<artist>Cat Stevens</artist>
<country>UK</country>
<company>Island</company>
<price>8.90</price>
<year>1990</year>
</cd>
<cd>
<title>Stop</title>
<artist>Sam Brown</artist>
<country>UK</country>
<company>A and M</company>
<price>8.90</price>
<year>1988</year>
</cd>
<cd>
<title>Bridge of Spies</title>
```

```
<artist>T`Pau</artist>
<country>UK</country>
<company>Siren</company>
<price>7.90</price>
<year>1987</year>
</cd>
<cd>
  <title>Private Dancer</title>
  <artist>Tina Turner</artist>
  <country>UK</country>
  <company>Capitol</company>
  <price>8.90</price>
  <year>1983</year>
</cd>
<cd>
  <title>Midt om natten</title>
  <artist>Kim Larsen</artist>
  <country>EU</country>
  <company>Medley</company>
  <price>7.80</price>
  <year>1983</year>
</cd>
<cd>
  <title>Pavarotti Gala Concert</title>
  <artist>Luciano Pavarotti</artist>
  <country>UK</country>
  <company>DECCA</company>
  <price>9.90</price>
  <year>1991</year>
</cd>
<cd>
  <title>The dock of the bay</title>
  <artist>Otis Redding</artist>
  <country>USA</country>
  <company>Stax Records</company>
  <price>7.90</price>
  <year>1968</year>
</cd>
<cd>
  <title>Picture book</title>
  <artist>Simply Red</artist>
  <country>EU</country>
  <company>Elektra</company>
  <price>7.20</price>
  <year>1985</year>
</cd>
<cd>
  <title>Red</title>
  <artist>The Communards</artist>
  <country>UK</country>
  <company>London</company>
  <price>7.80</price>
  <year>1987</year>
```

```
</cd>
<cd>
    <title>Unchain my heart</title>
    <artist>Joe Cocker</artist>
    <country>USA</country>
    <company>EMI</company>
    <price>8.20</price>
    <year>1987</year>
</cd>
</catalog>
```

Zatim kreirajte XSL Style Sheet ("cdcatalog.xsl") sa templejtom za transformacije:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
    <h2>My CD Collection</h2>
    <table border="1">
        <tr bgcolor="#9acd32">
            <th style="text-align:left">Title</th>
            <th style="text-align:left">Artist</th>
        </tr>
        <xsl:for-each select="catalog/cd">
        <tr>
            <td><xsl:value-of select="title"/></td>
            <td><xsl:value-of select="artist"/></td>
        </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Dodati XSL style sheet referencu u XML dokument ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
    <cd>
        <title>Empire Burlesque</title>
        <artist>Bob Dylan</artist>
        <country>USA</country>
        <company>Columbia</company>
        <price>10.90</price>
        <year>1985</year>
    </cd>
    .
    .
</catalog>
```

❖ 5.1 XPath

OSNOVNI OPERATORI XPATH-A

XPath koristi sličnu sintaksu sa onom koja se koristi kod putanja u sistemu datoteka na računaru, ali u kontekstu XML dokumenta

Atribut **match** je jedan od nekoliko atributa iz XSLT-a koji se koriste za pronalaženje čvorova u XML dokumentu. Uparivanje obrazaca je potpomognuto XPath izrazima, koji parsovane čvorove XML dokumenta izražavaju preko stabla. XPath koristi sličnu sintaksu sa onom koja se koristi kod putanja u sistemu datoteka na računaru, ali u kontekstu XML dokumenta. XPath stablo deli XML dokument u niz povezanih čvorova, elemenata, teksta, atributa, i ostalih čvorova. Zamislite da XSLT procesor parsuje dokument i svaki od elemenata u njemu postavi u odgovarajući direktorijum. Svaki od atributa i ostalih čvorova ide u svoj poseban direktorijum, sa posebnim identifikatorima. Sistem datoteka počinje korenom (/), a svaki element naslednik se nalazi u nekom poddirektorijumu ispod glavnog direktorijuma. XPath ne radi baš potpuno isto, ali na površini izgleda isto.

Osnovni operatori iz XPath-a za pronalaženje lokacije su dati na slici 1. Pored ovih postoje i drugi izrazi koji omogućavaju da se pronađe tekući čvor, kao i da se počev od tog čvora pronađu njegovi preci, naslednici i svi ostali.

Operator	Opis
.	Tekući čvor
..	Čvor roditelj
/	Koren element
//	Svi naslednici
@	Identifikator atributa
*	Svi izvedeni čvorovi

Slika 5.2.1 Osnovni operatori iz Xpath-a za pronalaženje lokacije [Izvor: Autor]

▼ Poglavlje 6

Pokazna vežba - Kreiranje osnovnog XML dokumenta

UVOD

Poput HTML jezika, XML koristi odgovarajuće elemente sa tagovima za kreiranje Web stranica, ali i drugih dokumenata.

(Ukupno vreme trajanja iznosi 65 minuta.)

World Wide Web (WWW) karakteriše veliki broj informacija, pri čemu je veoma mali procenat organizovan na metodičan način. To je razlog sve češćeg korišćenja XML jezika ([Extensible Markup Language](#)). Poput HTML jezika, XML koristi odgovarajuće elemente sa tagovima za kreiranje Web stranica, ali i drugih dokumenata. XML je napravljen da bude potpuno otvoren, tj. pruža mogućnost korisniku da sam kreira svoje tagove. To nije slučaj sa HTML jezikom. XML uvodi potpuno novi način razmišljanja o Internetu i elektronskim informacijama uopšte. XML je generički jezik koji se koristi za opisivanje drugih markup jezika. XML je veoma uopšten i otvoren za dopunjavanje. XML dokument čini nekoliko osnovnih komponenti koje opisuju izgled različitih delova. To su:

- tagovi elemenata,
- reference entiteta,
- komentari,
- instrukcije za obradu, i
- deklaracije tipa dokumenta.

Ove komponente čine osnovnu strukturu XML jezika i određuju izgled dokumenata.

U HTML dokumentu za označavanje se koriste unapred definisani tagovi. Sa druge strane, XML označavanje je više opisno i ne mora obavezno da ima veze sa sadržinom.

Za strukturu dokumenta mnogo su važniji elementi. Element čine odgovarajući tagovi i sadržina. Element može imati početni i završni tag ili samo jedan prazan tag.

Sadržina se često naziva i character data što znači da se sastoji od znakova. Može se reći da su tagovi specifični znakovni nizovi koji se koriste za predstavljanje elemenata u XML dokumentima.

PRIMER 1

Kreirati XML dokument sa elementima.

(Predviđeno vreme izrade zadatka iznosi 10 minuta.)

Zadatak

Kreirati XML dokument koji predstavlja poruku sa elementima koji se odnose na pošiljaoca, primaoca, naslov i tekst poruke.

Rešenje:

Svaki XML dokument počinje deklaracijom koja definiše XML verziju:

<?xml version="1.0"?>

Kako tagove u XML korisnik definiše sam, to bi u slučaju ovog primera moglo izgledati ovako:

```
<?xml version="1.0" encoding="UTF-8"?>
<poruka>
    <od>Janka</od>
    <za>Tijanu</za>
    <naslov>Čestitka</naslov>
    <tekst>Sve najbolje u Novoj godini!</tekst>
</poruka>
```

Dokument se snima kao npr. primer1.xml koristeći opcije Notepad editora na poznati način, u npr. folder C:\IT210\Vezba12.

U XML dokumentu pravi se razlika između označavanja i sadržine. U ovom primeru oznake, odnosno tagovi: **<od>**, **<za>**, **<naslov>** ili **<tekst>**, predstavljaju označavanje, a između njih se nalazi odgovarajuća sadržina.

PRIMER 2

Kreirati XML dokument koji će predstaviti studente.

(Predviđeno vreme izrade zadatka iznosi 10 minuta.)

Zadatak

Kreirati XML dokument koji će predstaviti studente FIT-a: Petra Petrovića i Marka Markovića, sa atributom koji će opisivati način studiranja. Prvi student studira preko interneta, a drugi tradicionalno.

Rešenje:

Dokument se može kreirati na sledeći način:

```
<?xml version="1.0" encoding="UTF-8"?>
<fit>
    <student nacin_studiranja="Internet">
        <ime>Petar</ime>
        <prezime>Petrović</prezime>
    </student>
    <student nacin_studiranja="Tradicionalno">
```

```
<ime>Marko</ime>
Marković</prezime>
</student>
</fit>
```

Prilikom pisanja XML dokumenta potrebno je, između ostalog, imati na umu sledeća pravila:

1. Svi XML elementi moraju obavezno imati završni tag,
2. XML pravi razliku između malih i velikih slova,
3. XML elementi moraju biti pravilno ugnježdeni između početnog i završnog taga,
4. XML dokument mora imati koren - root element, vrednosti atributa se moraju uvek navoditi između znaka navoda...

U našem primeru pravilo 1. je poštovano jer svaki element ima i svoj završni tag.

Dokument ima root element <fit>, čime je zadovoljeno pravilo 4.

HIJERARHIJA ELEMENATA

Svaki element može da ima svoj pod element.

Za root element ostali elementi predstavljaju elemente nižeg ranga i po pravilu su „uvučeni” uz pomoć npr. tastera Tab. Ovi elementi mogu imati svoje pod-elemente. U tom slučaju elementi višeg nivoa nazivaju se roditeljima, a nižeg deca:

```
<root>
  <roditelj>
    <dete>.....</dete>
  </roditelj>
</root>
```

Kada neki element sadrži elemente-decu, to znači da su oni ugnježdeni u okviru elementa višeg nivoa. U ovom primeru, element sa tagovima <ime> i </ime> ugnježden je u element sa tagovima <student> i </student>. Vrednost atributa nacin_studiranja navedena je između navodnika (pravilo 5.).

PRIMER 3

Napraviti XML dokument za CD katalog.

(Predviđeno vreme izrade zadatka iznosi 15 minuta.)

Zadatak

Napraviti XML dokument koji će predstavljati katalog CD izdanja sa naslovom, izvođačem, godinom izdanja i nazivom izdavača.

Koristiti podatke iz sledeće tabele:

Rešenje:

Poštujući pravila navedena u prethodnom primeru, ovaj dokument bi mogao da izgleda ovako:

```
<?xml version="1.0" encoding="UTF-8"?>
<katalog>
    <CD>
        <naslov>Belgrade coffe shop 5</naslov>
        <izvodjac>Razni izvodjači</izvodjac>
        <godina>2005</godina>
        <izdavac>B92</izdavac>
    </CD>
    <CD>
        <naslov>Beogradska Arena 2005.</naslov>
        <izvodjac>Zdravko Čolić</izvodjac>
        <godina>2006</godina>
        <izdavac>CITY RECORDS</izdavac>
    </CD>
</katalog>
```

Svaki element ima svoj zatvarajući tag i jasno su odvojeni elementi-roditelji od elemenata dece. Jedan XML dokument ne čine samo označavanje i sadržina.

Pored toga veoma su važne i instrukcije za obradu i deklaracija tipa dokumenta. Instrukcije za obradu su posebne komande koje se prosleđuju programima koji se koriste za pregled XML dokumenta. To najčešće odgovarajući Web čitač. Instrukcije za obradu počinju sa **<?** i završavaju sa **?>**.

Funkcija za obradu koja se nalazi u svakom XML dokumentu je već korišćena u primerima ove vežbe i nalazi se uvek na početku dokumenta. U najjednostavnijem obliku ona izgleda ovako:

```
<?xml version="1.0"?>
```

Osim navedenih, veoma važna komponenta svakog XML dokumenta je deklaracija tipa dokumenta koja je izuzetno važna zato što opisuje strukturu dokumenta. Deklaracija tipa izvršava tri osnovna zadatka:

1. definiše koreni – root element dokumenta,
2. definiše elemente, attribute i entitete koji su specifični za dati dokument, i
3. identificuje spoljašnju definiciju tipa dokumenta (DTD).

IZBEGAVANJE KORIŠĆENJA ATRIBUTA

Atribute treba izbegavati kad god je to moguće.

Još jedna od preporuka je da se korišćenje atributa izbegava uvek kada je to moguće u korist korišćenja elemenata.

Traženi XML dokument u tom slučaju može se kreirati i na sledeći način:

```
<?xml version="1.0" encoding="UTF-8"?>
<fit>
```

```
<student>
    <nacin_studiranja>Internet</nacin_studiranja>
    <ime>Petar</ime>
    Petrović</prezime>
</student>
<student>
    <nacin_studiranja>Tradicionalno</nacin_studiranja>
    <ime>Marko</ime>
    Marković</prezime>
</student>
</fit>
```

PRIMER 4

Kreirati HTML dokument koji će podatke prikazati u obliku odgovarajuće tabele.

(Predviđeno vreme izrade zadatka iznosi 20 minuta.)

Zadatak

Napraviti HTML dokument koji će podatke iz prethodnog primera prikazati u čitaču u obliku odgovarajuće tabele.

Rešenje:

Važnost XML jezika prilikom kreiranja Web stranica je dvostruka. Prvo, Internet teži da se razvija u pravcu bolje strukturiranog skladištenja i prikazivanja informacija. Drugo, omogućena je upotreba specijalizovanih XML rečnika za označavanje posebnih tipova informacija.

U ovom primeru, pored osnovnih elemenata, HTML dokument treba da u telu sadrži i red koji će ukazivati na XML dokument koji će se koristiti.

```
<html>
    <head>
    </head>
    <body>
        <xml
            src="primer3.xml"
            id="xmldso"
            async="false">
        </xml>
        <table
            datasrc="#xmldso"
            width="100%"
            border="1">
            <thead>
                <th>Naslov</th>
                <th>Izvodjac</th>
                <th>Godina</th>
            </thead>
```

```
<tr align="left">
    <td><span datafld="naslov"></span></td>
    <td><span datafld="izvodjac"></span></td>
    <td><span datafld="godina"></span></td>
</tr>
</table>
</body>
```

▼ Poglavlje 7

Pokazna vežba: Audio zapisi na veb stranama

HTML AUDIO

Zvuk se može postaviti u pozadini, da se kreira link do izvora zvuka i da se ugradi zvuk.

(Ukupno vreme trajanja iznosi 70 minuta.)

Postoje tri pristupa upotrebe zvuka koji se razlikuju po načinu reprodukcije kada web čitač nađe na njih. Ovi načini su:

- postavljanje zvuka u pozadini, što znači da će zvuk biti reprodukovani čim se otvoriti stranica,
- kreiranje hiperveze do izvora zvuka, što znači da se reprodukcija vrši uz pomoć druge aplikacije, i
- ugrađivanje zvuka, kada reprodukciju izvršava pretraživač ili dopunski modul.

PRIMER 1 - POSTAVLJANJE ZVUKA U POZADINI

<BGSOUND SRC=„imefajla.aif“ LOOP=“infinite”>

(Predviđeno vreme izrade zadatka iznosi 10 minuta).

Zadatak

Kreirati HTML dokument koji će prilikom otvaranja u web čitaču prikazati animiranu sliku **ocean.gif** i pozadinski zvuk zapisan u datoteci **ocean.wav**.

Rešenje:

Pozadinski zvuk je zvuk koji se reprodukuje kada korisnik otvoriti web stranicu.

Za korišćenje pozadinskog zvuka na web stranici koristi se tag **<bgsound>** sa atributom **loop** koji određuje koliko puta će se zvuk reprodukovati.

Ovom atributu može se dodeliti vrednost „infinite“ i u tom slučaju će se zvuk ponavljati sve dok je stranica otvorena.

U ovom primeru ovaj tag se koristi na sledeći način:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<IMG SRC="ocean.gif">
<BGSOUND SRC="ocean.aif" LOOP="infinite">
</BODY>
</HTML>
```

PRIMER 2 - HIPERVEZA PREMA ZVUČNOJ DATOTECI

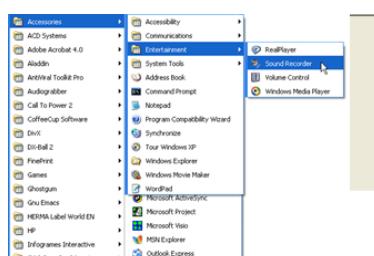
Kreiranje zvučnog zapisa koristeći Sound Recorder (Start→All Programs→Accessories→Entertainment→Sound Recorder).

Zadatak

Kreirati HTML dokument sa sledećim tekstom: **Dobro došli na FIT**, kao naslov drugog reda. Ispod toga nalazi se tekst: Pozdravna reč, koji predstavlja hipervezu prema zvučnoj datoteci **fit.wav**. Zvučni zapis snimiti koristeći Sound Recorder.

Rešenje:

Većina zvukova koji su dostupni putem WWW zaštićena je autorskim pravima. Međutim, čak i jednostavniji kućni sistemi omogućavaju snimanje sopstvenih zvučnih zapisa uz pomoć mikrofona, kasetofona ili video rekordera. Računari sa Windows operativnim sistemom imaju jednostavan editor zvuka koji se zove Sound Recorder (**Start→All Programs→Accessories→Entertainment→Sound Recorder**).



Slika 7.1 Sound Recorder u meniju [Izvor: Autor]

Nakon pokretanja, korisnik može da koristi standardne opcije za snimanje i reprodukciju zvučnih zapisa (slika 2).



Slika 7.2 Sound recorder prozor [Izvor: Autor]

Snimanje se aktivira uz pomoć krajnjeg desnog dugmeta. Neka pozdravna poruka u ovom slučaju bude: „Dobro došli na FIT“. Nakon izgovorene poruke, snimanje se prekida uz pomoć drugog dugmeta sa desne strane. U ovom slučaju neka zvučna datoteka bude snimljena kao fit.wav u folderu Vezba8, uz pomoć opcije Save u meniju File (slika 3).

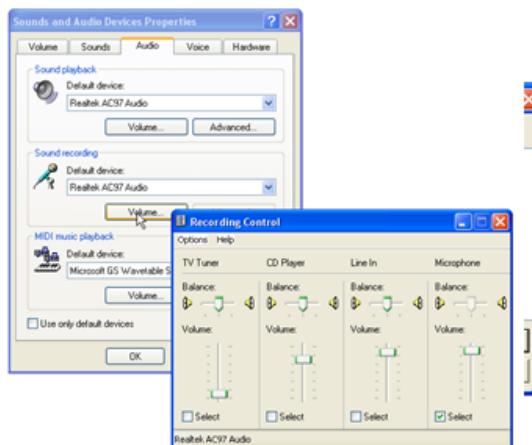


Slika 7.3 Čuvanje fajla [Izvor: Autor]

PRIMER 2 – POVEZIVANJE ZVUKA SA STRANICOM

Za povezivanje zvuka sa stranicom koristi se tag <a> i atribut href za identifikovanje izvora.

Za podešavanje jačine snimljenog glasa koriste se opcije koje se nalaze u **Control Panel→Sounds and Audio Devices→Audio** i u tom prozoru opcija **Volume** u delu **Sound recording**. Klikom na ovu opciju otvara se prozor **Recording Control**. Da bi glas uopšte bio snimljen, potrebno je da bude „čekirano“ polje **Select** u delu **Microphone**. Osim toga, date su mogućnosti kontrolisanja jačine snimljenog zvuka i sa drugih ulaznih uređaja .



Slika 7.4 Podešavanje zvuka [Izvor: Autor]

Za dodatna podešavanja zvuka mogu se koristiti opcije u prozoru Volume Control, do kojeg se dolazi na sledeći način: **Start→All Programs→Accessories→Entertainment→Volume Control**

Sadržaj ovog prozora može izgledati drugačije u zavisnosti od hardvera i softvera. Hiperveza do zvuka na web stranici predstavlja najjednostavniji način upotrebe zvuka. U ovom slučaju slušanje zvuka je neobavezno, a pomoćna aplikacija reprodukuje zvuk. Za povezivanje zvuka

sa stranicom koristi se već objašnjeni tag `<a>` i atribut `href` za identifikovanje izvora, u ovom slučaju URL adrese zvučne datoteke.

Za ovaj primer može da se koristi sledeći dokument:

```
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <H2>Dobro došli na FIT</H2><BR>
    <A HREF="fit.wav">Pozdravna reč</A>
  </BODY>
</HTML>
```

„Klikom“ na link Pozdravna rec, pokreće se Windows Media Player ili druga aplikacija koja reprodukuje dati zvučni zapis. Jedan od načina da se dođe do zvučnih zapisa koji se mogu koristiti je uz pomoć arhiva na WWW.

PRIMER 3 - KORIŠĆENJE AUDIO ZAPISA U OKVIRU HTML DOKUMENTA

Za korišćenje audio zapisa u okviru HTML dokumenata koristi se tag `<embed>`.

(Predviđeno vreme izrade zadatka iznosi 20 minuta).

Zadatak

Kreirati HTML dokument sa sledećim spiskom:

- Žaba
- Krava
- Mačka
- Pas

Pored svakog od naziva neka bude prikazana linija sa kontrolnim opcijama za reprodukciju odgovarajućeg zvučnog zapisa. Odgovarajuće datoteke možete preuzeti sa e-learning sistema.

Rešenje:

Web čitač može da reprodukuje zvuk ugrađen na stranicu i bez posebne aplikacije. To je popularan način jer se na računaru na otvara dodatni prozor za reprodukciju.

Međutim, čitači imaju veća ograničenja vezano za formate zvučnih datoteka koje mogu da se reprodukuju. Za korišćenje audio zapisa u okviru HTML dokumenata koristi se tag `<embed>` u sledećem obliku:

```
<HTML>
  <HEAD>
```

```
</HEAD>
<BODY>
<UL>
<LI>
    Žaba
    <EMBED SRC="zaba.wav" WIDTH="45" HEIGHT="25" AUTOSTART="false">
</LI>
<LI>
    Krava
    <EMBED SRC="krava.wav" WIDTH="45" HEIGHT="25" AUTOSTART="false">
</LI>
<LI>
    Macka
    <EMBED SRC="macka.wav" WIDTH="45" HEIGHT="25" AUTOSTART="false">
</LI>
<LI>
    Pas
    <EMBED SRC="pas.wav" WIDTH="45" HEIGHT="25" AUTOSTART="false">
</LI>
</UL>
</BODY>
</HTML>
```

Kao atributi ovog taga pojavljuju se:

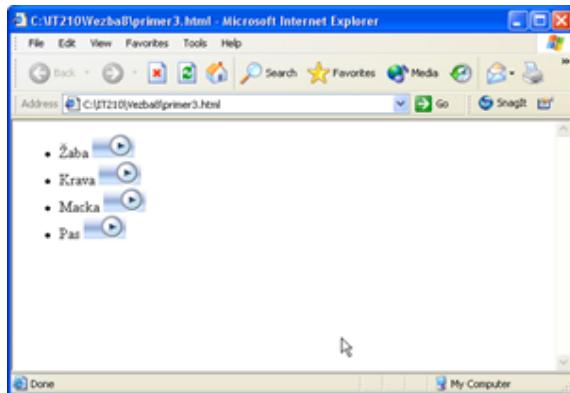
- **SRC**, koji ukazuje na putanju do audio datoteke koja će se koristiti,
- **AUTOSTART**, koji može imati vrednosti „true“ ili „false“, u zavisnosti od toga da li je potrebno da se datoteka pokrene učitavanjem web stranice,
- **WIDTH**, koji definiše širinu linije sa kontrolnim opcijama (izborom dovoljno velike širine biće prikazana cela linija), i
- **HEIGHT**, koji definiše visinu linije sa kontrolnim opcijama.

PRIMER 3 - POKRETANJE ZVUKA PRILIKOM UČITAVANJA STRANE

Koriste se atributi SRC i AUTOSTART sa vrednošću „true“, što će omogućiti pokretanje zvučne datoteke prilikom učitavanja web strane.

(Predviđeno vreme izrade zadatka iznosi 15 minuta).

Nakon snimanja pripremljenog dokumenta i njegovog otvaranja u web čitaču, dobija se dokument gde se klikom na opciju za pokretanje pored naziva životinje, u web čitaču reproducuje odgovarajući zvučni zapis (slika 5).



Slika 7.5 Prozor browsera [Izvor: Autor]

Da bi se prikazala cela linija sa kontrolnim opcijama dovoljno je da se ne koristi atribut width u početnom tagu.

Kao rezultat pored svake stavke pojaviće se linija sa svim opcijama.

Ovaj primer može se modifikovati i na sledeći način:

```
<HTML>
  <HEAD>
  </HEAD>
  <BODY>
    <UL>
      <LI>
        Žaba
        <EMBED SRC="zaba.wav" HIDDEN="true" AUTOSTART="true" LOOP="two">
      </LI>
      <LI>
        Krava
        <EMBED SRC="krava.wav" HIDDEN="true" AUTOSTART="true" LOOP="one">
      </LI>
      <LI>
        Macka
        <EMBED SRC="macka.wav" HIDDEN="true" AUTOSTART="true" LOOP="three">
      </li>
      <LI>
        Pas
        <EMBED SRC="pas.wav" HIDDEN="true" AUTOSTART="true" LOOP="four">
      </LI>
    </UL>
  </BODY>
</HTML>
```

U ovom slučaju koriste se već poznati atributi SRC i AUTOSTART koji u ovde ima vrednost „true“, što znači da će se sve zvučne datoteke pokrenuti učitavanjem web strane. Kao novi, pojavljuju se atributi:

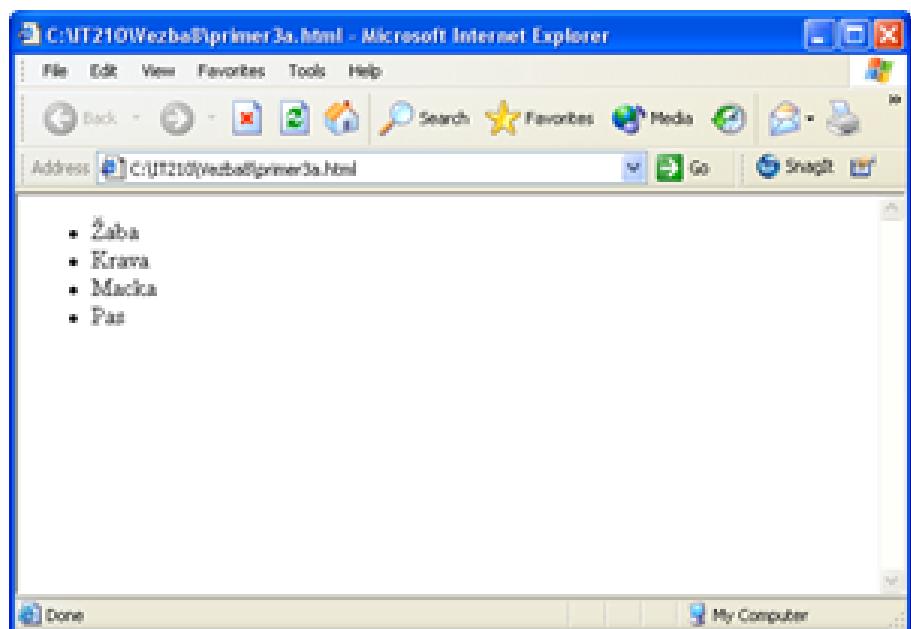
- **HIDDEN**, sa vrednošću „true“, što znači da se linija sa kontrolnim opcijama neće prikazivati u čitaču (podrazumevana vrednost je „false“)

- **LOOP**, koji definiše broj ponavljanja zvučnog zapisa.

PRIMER 3-REZULTAT

Zvučni zapis će se reprodukovati onoliko puta koliko je definisano atributom LOOP

Nakon snimanja i otvaranja u web čitaču ovog dokumenta, reprodukuju se svi zvučni zapisi i to **zaba.wav** dva puta, **krava.wav** jednom, **macka.wav** tri puta i **pas.wav** četiri puta, u skladu sa vrednošću atributa loop (slika 6).



Slika 7.6 Lista [Izvor: Autor]

✓ Poglavlje 8

Zadaci za samostalnu vežbu

OPIS ZADATKA ZA SAMOSTALNU VEŽBU

Kreirati XML dokument

(Predviđeno vreme izrade zadatka je 45 minuta.)

- 1.) Kreirati XML dokument sa podacima o igračkama koje se prodaju u online prodavnici. Dokument treba da sadrži sledeće podatke: Ime igračke, Vrsta, Cena, Proizvođač. Upisati podatke o najmanje pet igračaka. Tagove i strukturu dokumenta definisete sami.
- 2.) Kreirati XML dokument sa podacima o knjigama koje se prodaju u knjižari. Potrebno je osmisliti i definisati minimum 6 atributa koje će opisivati knjigu. Upisati podatke o najmanje 5 knjiga.
- 3.) Popuniti tabelu student koristeći sledeći xml:

```
<FIT>
<Student>
    <BrIndexa> 1556 </BrIndexa>
    <Ime> Milica </Ime>
    Petrovic </Prezime>
    <GodinaStudija> I </GodinaStudija>
    <Smer> SE </Smer>
</Student>
<Student>
    <BrIndexa> 1589 </BrIndexa>
    <Ime> Nikola </Ime>
    Micić </Prezime>
    <GodinaStudija> II </GodinaStudija>
    <Smer> IS </Smer>
</Student>
<Student>
    <BrIndexa> 1001 </BrIndexa>
    <Ime> Petar </Ime>
    Bojović </Prezime>
    <GodinaStudija> III </GodinaStudija>
    <Smer> IS</Smer>
</Student>
<Student>
    <BrIndexa> 576 </BrIndexa>
    <Ime> Nikola </Ime>
    Marković </Prezime>
    <GodinaStudija> Iv </GodinaStudija>
    <Smer> RI </Smer>
</Student>
```

```
<Student>
    <BrIndexa> 899</BrIndexa>
    <Ime>Velibor </Ime>
    Nikolic </Prezime>
    <GodinaStudija> II</GodinaStudija>
    <Smer> RI</Smer>
</Student>
<FIT>
```

4.) Kreirati i popuniti tabelu Pas koristeći sledeći xml:

```
<Zivotinja>
<Pas>
    <Rasa> Dogo argentino </Rasa>
        <Ime> Djuki </Ime>
        <DatumOkota>22.3.2012 </DatumOkota>
        <DatumPrimanjaVakcine> 22.8.2012 </DatumPrimanjaVakcine>
</Pas>
<Pas>
    <Rasa>Dalmatinac </Rasa>
        <Ime> Milo </Ime>
        <DatumOkota>12.2.2011 </DatumOkota>
        <DatumPrimanjaVakcine> 05.9.2011 </DatumPrimanjaVakcine>
</Pas>
<Pas>
    <Rasa>Mops</Rasa>
        <Ime> Radovan </Ime>
        <DatumOkota>12.3.2010 </DatumOkota>
        <DatumPrimanjaVakcine> 08.11.2011 </DatumPrimanjaVakcine>
</Pas>
</Zivotinja>
```

✓ Poglavlje 9

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ06

Kreirati XML dokument sa podacima o restoranu.

(Predviđeno vreme izrade domaćeg zadatka je 60 minuta.)

Kreirati XML dokument sa podacima o meniju jela u restoranu. Dokument treba da sadrži sledeće podatke: naziv jela, tip jela (predjelo, glavno, dezert...), tri glavna sastojka, cenu i veličinu porcije.

Upisati podatke o najmanje šest jela.

Nazive tagova i strukturu dokumenta definišete po vašem izboru.

Dokument snimiti pod imenom **IT210-DZ06-Ime_Prezime_brojIndeksa**, gde su ime, prezime i broj indeksa vaši podaci. Arhiviran fajl poslati predmetnom asistentu na e-mail.

(napomena: u Subject-u e-majla napisati **IT210 - DZ06**).

▼ ZAKLJUČAK

ZAKLJUČAK

U ovoj lekciji definisan je pojam meta podataka, kao i postojeći pristupi rada meta podacima. Dat je osnovni pregled Warwick i Dublin Core okruženja. Takođe, objašnjeno je kako se XML i model objekta dokumenta mogu koristiti za razmenu podataka između sistema. Pored toga opisano je kako se XSL, XSLT i XPath koriste za prenos toka podataka.

Literatura

1. W3School, <https://www.w3schools.com/xml/> (04.02.2020)
2. Tutorials point, <https://www.tutorialspoint.com/xml/index.htm> (04.02.2020)



IT210 - SISTEMI IT

KOMUNIKACIJA IZMEĐU SISTEMA

Lekcija 07

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 07

KOMUNIKACIJA IZMEĐU SISTEMA

- ✓ KOMUNIKACIJA IZMEĐU SISTEMA
- ✓ Poglavlje 1: Distribuirane aplikacije
- ✓ Poglavlje 2: CORBA
- ✓ Poglavlje 3: DCOM
- ✓ Poglavlje 4: RMI
- ✓ Poglavlje 5: Poređenje CORBA, DCOM, RMI tehnologija
- ✓ Poglavlje 6: Veb servisi
- ✓ Poglavlje 7: Pokazna vežba – XPATH, XSLT
- ✓ Poglavlje 8: Zadatak za samostalnu vežbu
- ✓ Poglavlje 9: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da se razume integracija sistema, arhitektura za integraciju sistema, DCOM, CORBA, RMI, Veb Servisi i middleware.

U ovoj lekciji biće obrađene sledeće teme

- Arhitekture za integraciju sistema
- DCOM
- CORBA
- RMI
- Veb Servisi i middleware

✓ Poglavlje 1

Distribuirane aplikacije

PRIKAZ DISTRIBUIRANIH APLIKACIJA

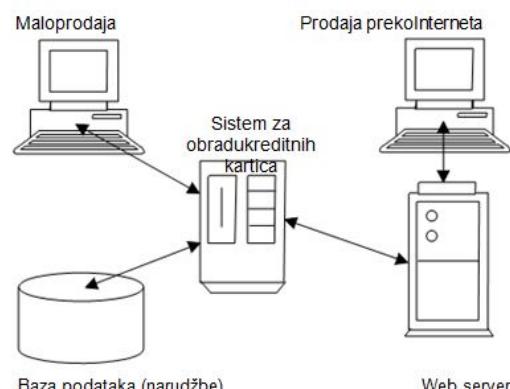
Distribuisane aplikacije su računarske aplikacije, kod kojih podaci sa kojima rade, kao i izračunavanja funkcionišu na različitim računarima

Velike aplikacije koje prate poslovanje celog preduzeća, često zahtevaju integraciju različitih procesa i aplikacija, koje tim procesima rukuju. Te aplikacije mogu raditi na različitim računarima. Ako želimo da te aplikacije komuniciraju, mora postojati šema za slanje poruka iz jednog procesa u drugi.

Distribuirane aplikacije su računarske aplikacije, kod kojih podaci sa kojima rade, kao i izračunavanja funkcionišu na različitim računarima. Cilj je da se zadaci koje korisnici postavljaju pred računarske aplikacije, podele na različite računare i da se tako iskoristi snaga više procesora na tim mašinama.

Jedna od prednosti distribuisanih aplikacija je i u tome da istu komponentu mogu koristiti više drugih. Kao što se vidi iz primera sa slike 1, može se napraviti podsistem za proveru kreditne kartice, koji se zatim istovremeno koristi kod maloprodaje, ali i kod prodaje putem Interneta.

Umesto da se komponenta za proveru kreditne kartice pravi za svaku aplikaciju koja je koristi, može se napraviti sistem koji će preko nekog protokola za distribuisane aplikacije, omogućiti da se poruke šalju od jedne do druge aplikacije. Na taj način se aplikacija za proveru kreditne kartice može posmatrati kao još jedan servis mreže, koji se može koristiti iz različitih aplikacija.



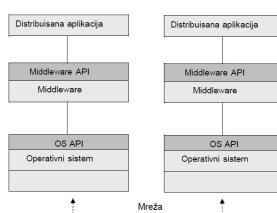
Slika 1.1.1 Primer distribuiranih aplikacija [Izvor: Autor]

✓ 1.1 Middleware tehnologije

SLOJEVI MIDDLEWARE TEHNOLOGIJE

Middleware se može definisati kao nivo koji se nalazi između distribuisanih aplikacija i operativnih sistema

Koncept **middleware** se pojavio 90-ih godina prošlog veka u cilju rešavanja problema koji su se javljali kod distribuisanih sistema. Osnovni motiv je bio rešavanje problema vezanih za mrežu, tako da je osnovna struktura na početku bila klijent/server arhitektura. Danas postoje različiti middleware slojevi. **Generalno middleware se može definisati kao nivo koji se nalazi između distribuisanih aplikacija i operativnih sistema.** Ovo je ilustrovano na sledećoj slici.



Slika 1.2.1 Postoje različiti middleware slojevi [Izvor: Autor]

Istorijski gledano, middleware je na početku odgovarao modelu klijent/server. Na primer, DCE (**Distributed Computing Environment**) definiše svoj RPC (**Remote Procedure Call**) protokol. Implementacija RPC-a sadrži kompjajler koji prevodi definiciju interfejsa u tzv. stub na strani klijenta. Ovaj program pakuje poziv procedure i potrebne parametre u pakete, a stub na strani servera, prima taj paket i pretvara ga u poziv na lokalnom serveru. Sličan koncept se koristi i u današnjim middleware tehnologijama.

Nešto kasnije je razvijen drugi tip middleware tehnologija, tzv. **Message Oriented Middleware** (MOM). U pitanju je i dalje klijent/server arhitektura, ali je prenosivost i fleksibilnost poboljšana jer aplikacije mogu da budu distribuisane na različitim platformama. Ova tehnologija smanjuje složenost razvoja aplikacija koje obuhvataju različite operativne sisteme, time što se programer izoluje od detalja vezanih za operativni sistem i mrežne interfejse. Problem je što jedna MOM implementacija obično nije kompatibilna sa drugom. Programeri su koristili različite šeme, ali su ih ove starije strategije terale da se više bave komunikacijom preko mreže, nego što su mogli da se posvete rešavanju konkretnog problema. Takođe se morala uzeti u obzir i činjenica da promena provajdera ili klijenta neminovno dovodi do ponovnog pisanja programa za komunikaciju, tako da su visoko plaćeni programeri radili stvari koje se nisu direktno odnosile na poslovni proces. Kako je objektno orijentisano programiranje dobijalo na popularnosti, tako je postalo očigledno da ga treba primeniti i kod ove komunikacije. Programeri su tražili način da preko mreže šalju objekte, koji su predstavljali entitete iz poslovnog procesa. Tako su nastali distribuisani objekti.

✓ 1.2 Distribuirani objekti

ŠTA OMOGUĆAVA MEHANIZAM DISTRIBUIRANIH OBJEKATA?

Mehanizam sa distribuisanim objektima omogućava da objekti implementirani na jednom računaru šalju poruke do objekata koji rade u drugom adresnom prostoru, obično preko mreže.

Mehanizam sa distribuiranim objektima omogućava da objekti implementirani na jednom računaru šalju poruke do objekata koji rade u drugom adresnom prostoru, obično preko mreže. Drugim rečima, ako bi postojao standardni mehanizam, tako da Java može da pošalje poruku do objekta na drugom računaru, isto tako lako kao što je šalje do objekta iz iste aplikacije, onda bi objekti mogli da premoste jaz između poslovnog procesa i softvera u distribuiranom okruženju.

Naravno da rešavanje ovog problema donosi niz izazova. Na primer, projektanti takvih distribuiranih objektnih sistema moraju da vode računa o:

- Heterogenosti platformi
- Lokaciji
- Rukovanju memorijom
- Različitim jezicima
- Komunikaciji preko mreže
- Perzistenciji objekata
- Standardizaciji

HETEROGENOST, LOKACIJA, RUKOVANJE MEMORIJOM, RAZLIČITI JEZICI

*Prenosni mehanizmi moraju da vode računa o formatu podataka.
Okruženje treba da ima mogućnost pretrage i rukovanje memorijom*

- Heterogenost platformi. U okruženju distribuiranih objekata je za očekivati da različite softverske komponente i mrežni servisi kojima želite da pristupate rade na različitim hardverskim platformama i pod različitim operativnim sistemima. Prenosni mehanizam za distribuirane objekte mora da vodi računa i o formatu podataka.
- Lokacija. U distribuiranom objektnom okruženju mora postojati način za pronalaženje objekta koji treba da pruži uslugu. Kako će, na primer, aplikacija za maloprodaju pronaći objekat kreditne kartice i mašinu na kojoj se on nalazi
- Rukovanje memorijom. Slanje poruke objektu koji se nalazi u drugom adresnom prostoru donosi nove probleme. Na primer, ako od udaljenog objekta zatražite ime kupca, kako će vam string sa imenom biti vraćen. Da li po vrednosti ili po referenci? Ne može biti potpuno po referenci, jer referenca na drugi adresni prostor nema značenje u vašoj aplikaciji. Ako se vrednost pošalje po vrednosti, kako se onda rukuje promenama vrednosti. Ako se promeni kopija, original neće biti promenjen. Pored toga, ako imate

referencu na udaljeni objekat, kako će ta udaljena aplikacija znati kako može da obriše taj objekat iz memorije?

- **Različiti jezici.** Nijedan programski jezik ne može da zadovolji potrebe svih tipova aplikacija. Zbog toga treba očekivati da aplikacije koje komuniciraju budu pisane u različitim programskim jezicima, koji mogu imati potpuno različite koncepte. Na primer, kako će Java aplikacija poslati poruku objektu iz C++-a kada ovaj jezik podržava višestruko nasleđivanje, koje Java ne podržava. Distribuisano objektno okruženje mora biti u stanju da obavlja prevodenje iz jednog jezika u drugi.

MREŽNA KOMUNIKACIJA, PERZISTENCIJA OBJEKATA I STANDARDIZACIJA

U komunikaciji preko mreže mora postojati mehanizam za rukovanje mrežnim protokolom, a preporučuje se i standardizacija za rad sa distribuiranim objektima

- **Komunikacija preko mreže.** Poruke između objekata se moraju slati na takav način da klijentska aplikacija ne mora da se bavi ničim vezanim za mrežu, da bi komunicirala sa servisom sa kojim se povezuje. To znači da mora postojati mehanizam koji će rukovati mrežnim protokolom. Na primer, kada iz aplikacije pošaljete poruku, obično očekujete da će ona stići tamo gde je potrebno. U distribuiranom okruženju je sasvim moguće da poruka nikad ne stigne na odredište. Nije moguće svešto je vezano za mrežu potpuno sakriti od programera, ali što se većim delom može rukovati bez programera, to bolje.
- **Perzistencija objekata.** Komunikacija sa objektima preko mreže postavlja i pitanje kako sačuvati referencu na udaljeni objekat, za kasniju upotrebu.
- **Standardizacija.** Čak i ako se pronađu rešenja za sve ove probleme, postavlja se pitanje standardizacije. Ako šema za rad distribuiranih objekata nije široko prihvaćena, onda od nje nema mnogo koristi. Koja je korist od toga što imamo način za povezivanje aplikacija i servisa, ako ne postoje aplikacije koje taj mehanizam podržavaju.

KAKO TREBA DA IZGLEDA REŠENJE?

Prikaz uspešne šeme za rad distribuiranih objekata.

Uspešna šema za rad distribuisanih objekata, minimalno mora da:

- Podržava različite platforme
- Omogućava pronalaženje objekta
- Rešava probleme za rukovanje memorijom
- Podržava komunikaciju između različitih programskih jezika
- Transparentno rukuje komunikacijom preko mreže
- Obezbedi mehanizam za čuvanje referenci na udaljene objekte
- Dobije podršku u softverskoj industriji

U krajnjoj liniji kod za distribuiranu aplikaciju treba da za programera izgleda isto kao i kod za aplikaciju koja nije distribuisana. Pogledajmo na primer, sledeći Java kod (ovo nije kompletan primer):

```
HelloObject h = new HelloObject();  
  
h.reciZdravo("Miroslav");
```

U distribuiranoj verziji će objekat **HelloObject** biti napravljen na udaljenoj mašini. Ovaj objekat se zatim registruje pomoću nekog tipa servisa za imena, tako da ga klijenti kasnije mogu da pronađu.

Distribuisana verzija može da izgleda ovako:

```
HelloInterface h = new HelloFactory("mojserver", "imeobjekta");  
h.reciZdravo("Miroslav");
```

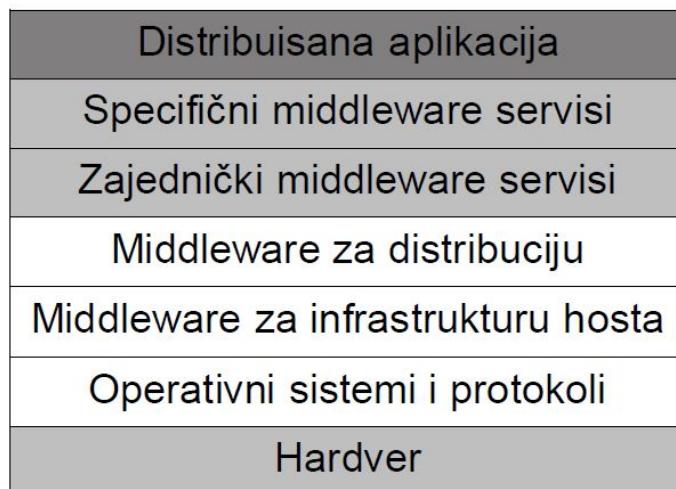
Razlike su relativno male. Pre svega objekat se vraća kao interfejs, a ne kao pravi objekat. Na taj način se implementacija udaljenog objekta može promeniti bez uticaja na klijentski kod. Drugo što nam je potrebno je mehanizam za pronalaženje mašine na kojoj se nalazi distribuirani objekat. Ovo se može uraditi kreiranjem potklase koja kao parametre uzima ime udaljene mašine i registrovano ime objekta. Mehanizam za distribuciju objekta je klasa koja obavlja svu potrebnu komunikaciju za povezivanje sa udaljenim serverom, pronalazi određeni objekat i pravi proksi za taj objekat u adresnom prostoru vaše mašine. Kada se pozove metod reciZdravo(), parametar Miroslav se šalje do udaljenog objekta, koji ga obrađuje. Mehanizam za slanje podataka je sakriven od programera.

Danas postoje različite šeme za rad sa distribuiranim objektima i sve rešavaju probleme koji opisani u prethodnom tekstu.

DISTRIBUTED OBJECT COMPUTING (DOC)

Distribuirani objekti se ralizuju preko DOC tehnologija koji se sastoje od više nivoa

Distribuirani objekti se u praksi realizuju na osnovu **Distributed Object Computing (DOC)** tehnologija. DOC se može opisati preko nivoa koji ga čine. Ti nivoi su prikazani na sledećoj slici.



Slika 1.3.1 Prikaz nivoa DOC-a [Izvor: Autor]

U praksi su neki nivoi iz ove arhitekture razvijeni pre nego što je bilo definisana DOC tehnologija. Kao što se sa slike vidi DOC arhitektura se sastoji iz četiri nivoa i ona pokušava da bude most koji povezuje operativni sistem i distribuirane aplikacije. Dekompozicija na nivoe je pokušaj da se pojednostavi projektovanje i klasifikuju funkcije.

MIDDLEWARE ZA INFRASTRUKTURU HOSTA

Osnovni razlog postojanja ovog nivoa je kreiranje mrežnog okruženja za komunikaciju viših nivoa i aplikacija

Ovaj nivo je blisko povezan sa operativnim sistemom i protokolima za komunikaciju. Osnovni razlog postojanja ovog nivoa je kreiranje mrežnog okruženja za komunikaciju viših nivoa i aplikacija. Pojedini autori i ne prepoznaju ovaj nivo. Razlog je u tome što implementacija ovakvih nivoa ima znatno više funkcija nego što je samo podrška za distribuirane aplikacije. Primer za ovaj nivo može biti JVM ([Java Virtual Machine](#)). Ona obezbeđuje način za izvršavanje koda nezavisno od platforme, time što se apstrahuju razlike između operativnih sistema i arhitektura procesora. Očigledno je da JVM obezbeđuje okruženje u kome se mogu praviti drugi nivoi middlewarea. Prema tome JVM je middleware koji pomaže rešavanje problema različitih platformi. Sa druge strane, JVM ima mnogo više funkcija nego što je ova.

MIDDLEWARE ZA DISTRIBUCIJU

Neki autori ovaj nivo definišu kao najviši nivo distribuisanog modela, čiji API i komponente automatizuju i proširuju mrežne mogućnosti operativnog sistema

Middleware za distribuciju je ono što se danas smatra za middleware. Neki autori ovaj nivo definišu kao najviši nivo distribuiranog modela, čiji API i komponente automatizuju i proširuju mrežne mogućnosti operativnog sistema.

Na tržištu postoje različiti proizvodi koji potiču sa ovog nivoa. I pored toga, postoje tri glavna proizvoda, **CORBA**, **RMI** i **DCOM**. CORBA je u suštini specifikacija, a ne konkretan proizvod. Ovu specifikaciju implementiraju različiti proizvođači. RMI omogućava kreiranje distribuiranih aplikacija pisanih u Javi. DCOM se odnosi na Microsoftovu viziju distribuiranog softvera. Teoretski se DCOM može implementirati na bilo kojoj platformi, u bilo kom jeziku. Ipak pošto je Microsoft orijentisan na Windows platformu, to i DCOM postoji uglavnom za tu platformu.

MIDDLEWARE - VIDEO

What is Middleware? Application Servers

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 2

CORBA

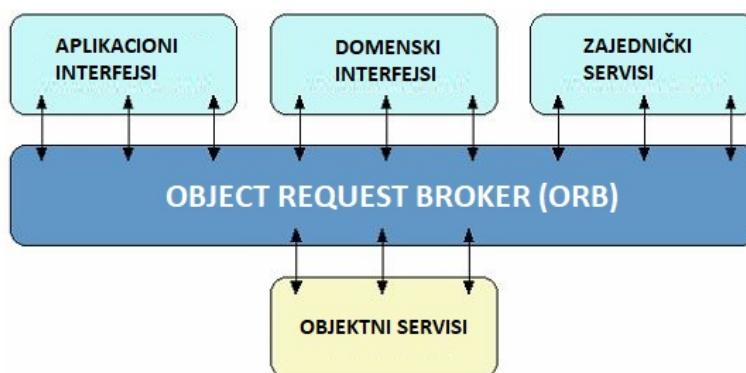
KOMUNIKACIJA PREKO ORB-A

CORBA omogućuje integraciju različitih objektnih sistema. ORB omogućava da klijenti i objekti komuniciraju u distribuiranom okruženju

CORBA je deo Object Management Architecture (OMA), koju je razvila OMG (Object Management Group). U pitanju je najraspostranjeniji middleware. Ona omogućava integraciju različitih objektnih sistema. Na sledećoj slici je prikazan osnovni OMA referentni model.

ORB (Object request Broker) omogućava da klijenti i objekti komuniciraju u distribuiranom okruženju. Preko ORB-a komuniciraju četiri vrste objekata. To su:

- Objektni servisi. U pitanju su interfejsi za opšte servise, koji će verovatno postojati u bilo kom programu koji koristi distribuirane objekte.
- Zajednički servisi (Common Facilities) su interfejsi za horizontalne servise orijentisane prema krajnjim korisnicima, koji se primenjuju u većini aplikacija.
- Domenski interfejsi su interfejsi specifični za konkretnе grupe aplikacija (finansije, telekom, transport).
- Aplikacioni interfejsi su nestandardni interfejsi koji postoje za specifične aplikacije.



Slika 2.1 Osnovni OMA referentni model [Izvor: Autor]

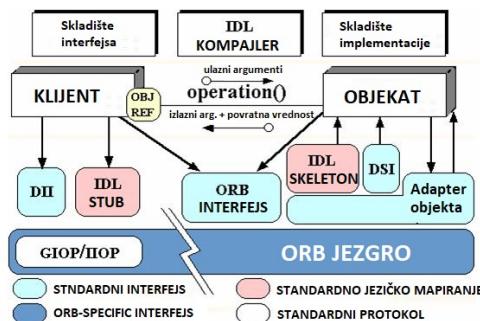
OSNOVNE KOMPONENTE CORBA ORB ARHITEKTURE

CORBA je u suštini arhitektura za ORB, koja rukuje poslovima kao što su dodeljivanje objekata, implementacija objekata, mehanizmima za komunikaciju objekata itd.

Ključna komponenta u OMA arhitekturi je ORB, odnosno CORBA. Nije teško zaključiti da ORB treba da obezbedi isporuku zahteva do objekata i vraćanje odgovora do klijenta koji je taj zahtev poslao. U distribuisanom okruženju ORB treba da se obezbedi i transparentnost. CORBA je u suštini arhitektura za ORB, koja rukuje poslovima kao što su dodeljivanje objekata, implementacija objekata, mehanizmima za komunikaciju objekata itd. Ako se koristi CORBA većina od onog što treba obaviti za komunikaciju objekata je transparentno. U tom smislu četiri kategorije objekata iz OMA arhitekture se mogu preko ORB-a, povezati u distribuisano okruženje. Pri tome ne treba brinuti o komunikaciji između objekata. Na sledećoj slici je prikazana CORBA ORB arhitektura sa osnovnim komponentama.

Glavne komponente su:

- Klijenti
- Implementacija objekta
- Reference objekta
- OMG Interface Definition Language (jezik za definisanje interfejsa)
- Stub i Skeleton
- Dinamički poziv i skeleton interfejsi
- Adapter objekti
- ORB interfejs
- Skladište interfejsa i implementacija (Interface repository i Implementation repository)
- GIOP (General Inter-ORB Protocol)



Slika 2.2 CORBA ORB arhitektura sa osnovnim komponentama [Izvor: Autor]

OSNOVNE KOMPONENTE CORBA ORB ARHITEKTURE – KLIJENTI, IMPLEMENTACIJA OBJEKATA, OMG IDL

Klijent poznaje jedino logičku strukturu objekta, na osnovu interfejsa. Implementacija objekta definiše semantiku objekta. Referenca na objekat definiše objekat u okviru ORB-a

Klijenti

Na bazi objektnog modela klijent nekog objekta ima pristup do reference na taj objekat i preko nje poziva operacije tog objekta. Klijent poznaje jedino logičku strukturu objekta, na osnovu interfejsa. On koristi ponašanje tog objekta pozivom tih operacija. Klijenti su na jednom kraju programa i treba da budu u stanju da iniciraju proces i pošalju zahtev objektu. Treba voditi računa i o prenosivosti klijenta.

Implementacija objekata

Implementacija objekta definiše semantiku objekta, obično putem definisanja podataka za instancu objekata, kao i koda za njegove metode. Implementacija objekta prema CORBA standardu ne zavisi od ORB-a ili klijenta, jer klijent i ne zna kako je objekat implementiran.

Reference objekata

U distribuiranom okruženju klijent ne zna gde se nalazi objekat kome pristupa. Referenca na objekat definiše objekat u okviru ORB-a. I klijent i implementacija objekta imaju samo predstavu o referenci na objekat, tako da su izolovani od stvarne reprezentacije tog objekta.

OMG Interface Definition Language (jezik za definisanje interfejsa)

OMG IDL definiše tipove objekata definisanjem njihovih interfejsa. Interfejs objekta definiše operacije i tipove koje objekat podržava i prema tome definiše zahteve koji se tom objektu mogu poslati. OMG IDL je deklarativni jezik (nije programski jezik). Interfejsi se definišu nezavisno od implementacije objekta. Najlepše je što objekti mogu da se prave u različitim programskim jezicima, a da i dalje mogu da komuniciraju preko interfejsa.

OMG IDL se može prebaciti u klase iz C++-a i interfejse u Javi. CORBA specifikacija definiše sve osnovne tipove, izvedene tipove, šablonske tipove, reference na objekte i nasleđivanje interfejsa.

IDL u suštini predstavlja način na koji konkretna implementacija objekta obaveštava svoje potencijalne klijente koje su operacije na raspolaganju i kako ih treba pozvati.

OSNOVNE KOMPONENTE CORBA ORB ARHITEKTURE – STUB I SKELETON, DINAMIČKI POZIV I SKELETON INTERFEJSI

Stub poziva ostatak ORB-a pomoću interfejsa koji su privatni i prema tome i optimizovani za određeno ORB jezgro. Skeleton obezbeđuju interfejse za pristup objektima.

Stub i skeleton

Stub na strani klijenta i skeleton na strani implementacije objekta imaju svrhu da omoguće statičko pozivanje objekata, slično kao kod RPC mehanizma. Stub poziva ostatak ORB-a pomoću interfejsa koji su privatni i prema tome i optimizovani za određeno ORB jezgro. Skeleton obezbeđuju interfejse za pristup objektima. Tokom procesa poziva, stub radi direktno sa klijentskim ORB-om i pakuje zahtev. Kad zahtev stigne do objekta, ORB na serveru i skeleton sarađuju i otpakuju zahtev i prosleđuju ga objektu. Ako se radi na ovaj način, onda su u pitanju statički pozivi, pošto i stub i skeleton imaju kompletno apriori znanje o OMG IDL interfejsima CORBA objekta koji se poziva.

Dinamički poziv i skeleton interfejsi

U CORBA arhitekturi postoji i mogućnost dinamičkog poziva. Ovo se radi preko interfejsa za dinamički poziv (**DII - Dynamic Invocation Interface**), kao i dinamičkog interfejsa skeleta (**DSI - Dynamic Skeleton Interface**). DII podžava dinamički poziv od strane klijenta. Ovo znači da klijent umesto poziva konkretnog objekta može definisati objekat koji se poziva, operaciju

koju treba obaviti, kao i parametre. DSI obezbeđuje dinamičko prosleđivanje do objekata. Implementaciji objekta se pristupa preko interfejsa koji radi slično kao DII na strani klijenta

OSNOVNE KOMPONENTE CORBA ORB ARHITEKTURE – ADAPTER OBJEKTI, ORB INTERFEJS, SKLADIŠTE I GIOP

Skladište interfejsa obezbeđuje prezistentne objekte koji predstavljaju informacije iz IDL-a i na raspolaganju su u trenutku izvršavanja programa

Adapter objekti

Adapter objekta je kritična komponenta u CORBA arhitekturi. On služi za spajanje implementacije objekta i ORB-a, čime se obezbeđuje pristup do ORB servisa. Bez adaptera ne bi bilo moguće da CORBA podrži različite stilove implementacije objekata. CORBA podržava više tipova adaptera. Za svaki programski jezik može biti potreban različit adapter.

ORB interfejs

ORB interfejs je apstraktни interfejs koji sadrži različite pomoćne funkcije za konverziju referenci na objekte u stringove i obrnuto, kao i za kreiranje liste argumenata za zahteve koji se šalju preko interfejsa za dinamički poziv. Različiti proizvođači ovaj entitet realizuju na različite načine.

Skladište interfejsa i implementacija (Interface repository i Implementation repository)

Skladište interfejsa obezbeđuje prezistentne objekte koji predstavljaju informacije iz IDL-a i na raspolaganju su u trenutku izvršavanja programa. Skladište implementacija sadrži informacije koje omogućavaju da ORB pronađe i aktivira implementacije objekata.

GIOP (General Inter-ORB Protocol)

Da bi različiti ORB-ovi mogli da komuniciraju, CORBA specifikacija definiše metodologiju poznatu pod imenom GIOP. GIOP predstavlja skup zahteva sa porukama koje ORB može da napravi preko mreže. GIOP povezuje ORB zahteve sa različitim načinima transporta preko mreže. Internet Inter -ORB protokol (IIOP) koji povezuje GIOP poruke i TCP/IP je standardizovana verzija GIOP-a, koju sve ORB implementacija moraju biti u stanju da koriste.

▼ Poglavlje 3

DCOM

ŠTA JE DCOM?

DCOM je distribuisano proširenje COM-a koje omogućava komponentama koje se distribuišu preko mreže da komuniciraju direktno na siguran i pouzdan način

DCOM je distribuisano proširenje **COM-a (Component Object Model)**, koje omogućava komponentama koje se distribuišu preko mreže da komuniciraju direktno na siguran i pouzdan način. On se nalazi na DCERPC nivou i obavlja pozive udaljenih procedura. U pitanju je binarni standard, što omogućava da se koristi bilo koji jezik koji može da da binarni kod, kompatibilan sa standardom. Osnovna karakteristika DCOM-a je da se operativni sistem ponaša kao centralni register referenci na objekte.

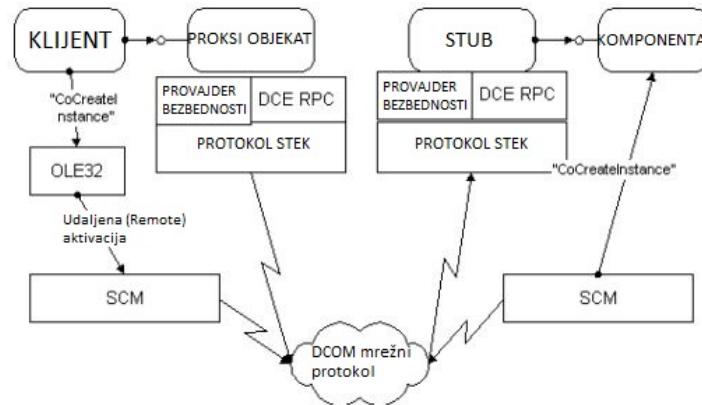
NAČIN AKTIVIRANJA OBJEKATA

Kod aktivacije unutar procesa instanca objekta servera se pravi i radi u adresnom prostoru klijenta; izvan procesa se objekat server nalazi u adresnom prostoru drugog procesa

DCOM podržava dva načina aktiviranja objekata, u procesu i izvan procesa. Kod aktivacije unutar procesa instanca objekta servera se pravi i radi u adresnom prostoru klijenta. Kod aktiviranja izvan procesa se objekat server nalazi u adresnom prostoru drugog procesa. Taj drugi proces može biti lokalni, kada se nalazi na istoj mašini kao klijent, ili udaljeni, kada se nalazi na drugoj mašini. U zavisnosti od same aplikacije, vidljivosti objekata za klijente i načina postavljanja komponenti, programer može izabrati jedan od ova dva načina aktiviranja. Ako svaki klijent traži posebnu instancu servera, onda se može izabrati aktiviranje unutar procesa. Ako se traži da posebna instanca servera radi nezavisno, prikupljujući zahteve od klijenata u letu, onda treba izabrati aktiviranje izvan procesa. DCOM klijenti objekte mogu pronaći na više načina. ID interfejsa objekta servera se može registrovati u sistemskom registru maštine klijenta. Ovo je fiksna konfiguracija. Druga opcija je da se omogući da klijent eksplisitno zada lokaciju objekta servera. Treća opcija je da se zada ime koje jednoznačno identificuje COM objekat.

Svaka COM komponenta izlaže svoje mogućnosti preko COM interfejsa. Klase objekata implementiraju jedan ili više ovih interfejsa. Svaki COM interfejs i klasa objekta ima globalni identifikator, preko kojeg se mogu jednoznačno identifikovati. Svaki COM server pravi

instance objekata. Na sledećoj slici su prikazane sve komponente koje postoje u COM arhitekturi.



Slika 3.1 Komponente COM arhitekture [Izvor: Autor]

SERVICE CONTROL MANAGER (SCM)

SCM (service control manager) sadrži metode pomoću kojih klijent obavlja udaljeno aktiviranje objekta servera

SCM (service control manager) sadrži metode pomoću kojih klijent obavlja udaljeno aktiviranje objekta servera. Ovaj nivo prima poziv od OLE32 i šalje ga na serversku mašinu. Ako je to prvi zahtev za tim objektom serverom, onda se na serveru kreira nova instance tog objekta. Ovaj poziv klijentu vraća pokazivač na interfejs, tako da klijent može da poziva metode udaljenog objekta. U narednom pozivu iste instance objekta, vraća se isti pokazivač na interfejs. Klijent sa objektom serverom može da komunicira samo preko metoda koji su opisani ovim interfejsom. COM interfejsi se definišu pomoću Microsoft Interface definition Language (MIDL) jezika.

Svaki COM interfejs bi trebalo da se izvede iz standardnog baznog interfejsa IUnknown, bilo direktno bilo indirektno. Ovaj interfejs ima tri metoda. To su **AddRef**, **QueryInterface** i **Release**. Na taj način klasa objekta mora da implementira sva ova tri metoda. Svaki put kada se klijent poveže, brojač referenci na instancu objekta se povećava za jedan, a kada se klijent odjavlji, taj broj se smanjuje za jedan. Ovo se radi pozivom metoda AddRef i Release. Preko ovog mehanizma, DCOM upravlja vezom sa klijentom. Ako je u nekom trenutku broj referenci veći od jedan, to znači da je najmanje jedan klijent povezan sa tim objektom. Kada se broj referenci smanji na nulu, objekat server se izbacuje iz memorije. Metod QueryInterface uzima ID interfejsa i vraća traženi Com interfejs.

Kad klijent jednom dobije pokazivač na objekat, on poziva metode kao da su lokalni metodi. Kod svakog poziva metoda se čita stek i piše u bafer koji sadrži podatke preko RPC mreže. Na strani klijenta se ovo radi preko stuba, koji se zove Proksi. Na strani servera postoji skeleton, koji pravi stek na toj strani i poziva metod.

DCOM - VIDEO

Distributed Component Object Model (DCOM)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 4

RMI

ŠTA JE JAVA RMI?

Java RMI omogućava da objekat koji postoji u jednoj virtuelnoj mašini (JVM) da pristupa objektu koji se nalazi u drugoj JVM

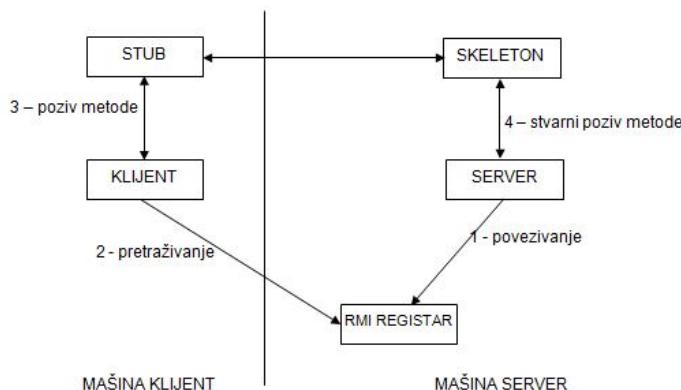
Java RMI omogućava implementaciju distribuisanih objekata u Javi. Ona omogućava da objekat koji postoji u jednoj virtuelnoj mašini (JVM) da pristupa objektu koji se nalazi u drugoj JVM. RMO prati tzv. jezik-centrični model. To znači da i klijent i server moraju biti implementirani u Javi. Na taj način oni mogu da koriste sve mogućnosti Java, kao što su JNI ili JDBC. To takođe znači da se objektu serveru može pristupiti sa bilo koje mašine, čime se postiže nezavisnost platforme. Sa druge strane obaveza upotrebe Java sprečava komunikaciju sa objektima koji su napisani u drugim programskim jezicima.

OSNOVNI MODEL

Osnovni model kod RMI tehnologije se sastoji od klijentskog programa, koji pristupa udaljenom objektu, i serverskog programa, koji sadrži udaljeni objekat.

Osnovni model kod RMI tehnologije se sastoji od klijentskog programa, koji pristupa udaljenom objektu, i serverskog programa, koji sadrži udaljeni objekat. Da bi klijent mogao da se poveže sa udaljenim objektom, mora postojati referenca na objekat koji se nalazi u serverskom programu. Klijent udaljeni objekat može da pronađe na dva načina. Ova dva načina se razlikuju u načinu na koji klijent dobija referencu na udaljeni objekat.

1. Eksplicitno dobijanje reference na udaljeni objekat. RMI sadrži registar po imenu RMIREGISTRY, koji bi trebalo da bude postavljen na serverskoj mašini. Kada se pravi instanca objekta servera, ona se sama registruje u lokalnom RMIREGISTRY. Ova akcija se obavlja pozivom metoda *Naming.bind()*. Kada pokušava da se poveže sa udaljenim objektom, klijent traži u RMIREGISTRY referencu tog objekta. Na sledećoj slici je prikazana ova arhitektura i redosled događaja u tipičnoj RMI aplikaciji.
2. Implicitno dobijanje reference na udaljeni objekat. Referenca na udaljeni objekat se može dobiti i kao parametar, ili povratna vrednost iz poziva metoda. Ovo se takođe može smatrati kao način pristupa udaljenom objektu. Prepostavlja se da RMI klijent zna na kojoj se mašini trenutno nalazi udaljeni objekat, tako da referenca može da se potraži baš na toj mašini.



Slika 4.1 Prikaz arhitekture i redosleda događaja u tipičnoj RMI aplikaciji [Izvor: Autor]

STUB I SKELETON

Nezavisno od načina na koji se referenca dobije, kad je klijent jednom dobije, dalje se poziv udaljenih metoda odvija preko stuba i skeleta

Nezavisno od načina na koji se referenca dobije, kad je klijent jednom dobije, dalje se poziv udaljenih metoda odvija preko stuba i skeleta. Stub na strani klijenta služi kao proksi za server. Skeleton na strani servera rukuje pozivima metoda i prenosi ih lokalnom objektu serveru. Kada klijent pozove metod udaljenog objekta, taj poziv prvo ide do stuba. Tu se radi pakovanje tih podataka, koji se šalju na server. Skeleton na strani servera raspakuje te podatke i prosleđuje ih do objekta servera. Po završetku rada metoda, skeleton prima povratne vrednosti, pakuje sadržaj i šalje to do stuba na klijentu koji ih raspakuje. Za klijenta je ceo proces transparentan i za klijenta ovaj poziv metoda izgleda kao lokalni poziv. Za pakovanje argumenata i povratnih vrednosti metoda se koristi serijalizacija objekata, koja postoji u Javi.

▼ Poglavlje 5

Poređenje CORBA, DCOM, RMI tehnologija

POREĐENJE NA OSNOVU FAKTORA VEZANIH ZA PROGRAMSKE JEZIKE I PLATFORMU

U zavisnosti od programskih jezika i platformi CORBA, DCOM i RMI imaju svoje prednosti

Poređenje ove tri tehnologije se može vršiti na osnovu različitih faktora.

RMI	CORBA	DCOM
Može se implementirati samo u Javi	Pošto je CORBA specifikacija implementacija je moguća u svim jezicima, koji podržavaju ORB biblioteke i mapiranje jezika	DCOM je binarni standard. Može se implementirati u bilo kom jeziku koji bi mogao da napravi zahtevani binarni kod.
Može da radi na svim platformama za koje postoji implementacija Java Virtuelne mašine.	Može da radi na svim platformama (UNIX, mainframe, Windows) za koje postoji ORB implementacija.	Može da radi na svim platformama za koje postoji implementacija COM servisa. Ipak, postoji samo za Windows platformu.

Slika 5.1 Prikaz poređenja CORBA, DCOM, RMI tehnologija na osnovu različitih faktora [Izvor: Autor]

POREĐENJE NA OSNOVU FAKTORA VEZANIH ZA IMPLEMENTACIJU

Poređenje se može obaviti na osnovu IDL specifikacija, transparentnost lokacije, registracija objekta, kao i načinu dobijanja reference na objekat server od strane klijenta

Ovo poređenje se odnosi na način na koji tri tehnologije omogućavaju implementaciju distribuisanih aplikacija. Razmotreni su IDL specifikacija, transparentnost lokacije (kako klijent zna gde se nalazi klijent), registracija objekta (način na koji se objekat server registruje), kao i način dobijanja reference na objekat server od strane klijenta.

RMI	CORBA	DCOM
IDL podržava višestruko nasleđivanje	IDL podržava višestruko nasleđivanje	IDL ne podržava višestruko nasleđivanje. Umesto toga se višestruki interfejsi mogu ubaciti u jednu klasu, a kretanje kroz interfejs se obezbeđuje posebnim
Dozvoljava da se izuzeci definišu na nivou IDL-a	Dozvoljava da se izuzeci definišu na nivou IDL-a	Ne dozvoljava da se izuzeci definišu na nivou MIDL-a
Na strani klijenta postoji Stub, a na strani servera skeleton.	Na strani klijenta postoji Stub, a na strani servera skeleton.	Stub na strani klijenta se zove Proksi, a na strani servera Stub.
Ime interfejsa jednoznačno identificuje interfejs. Implementacija objekta servera se povezuje sa jedinstvenim imenom u RMIREGISTRY.	Ime interfejsa jednoznačno identificuje interfejs. Implementacija objekta servera se povezuje sa jedinstvenim imenom u skladistištu implementacija.	Svaki interfejs ima jedinstveni identifikator interfejsa (IID). Svaka klasa implementacije objekta ima jedinstven ID klase (CSLID). I jedan i drugi identifikator se čuvaju u registrima sistema.
Klijent može da dobije referencu na udaljeni objekat pozivom metoda lookup() na udaljenoj mašini.	Klijent može da dobije referencu na udaljeni objekat servera pozivom metoda bind(), ili vezivanjem sa Naming and Trading servisom.	Klijent može da dobije pokazivač na udaljenu referencu pozivom metoda CoCreateInstance().
Da bi dobio referencu klijentski kod mora da zna koja je udaljena mašina na kojoj se nalazi objekat server.	Klijentski kod ne mora da zna na kojoj se mašini nalazi objekat server.	Klijentski kod ne mora da zna na kojoj se mašini nalazi objekat server.
Preko Java refleksije (Reflection) je moguć dinamički poziv objekata.	Informacije o udaljenom interfejsu u vreme izvršenja se nalaze u skladistištu interfejsa. Ovom skladistištu klijent može da šalje upite preko interfejsa za dinamičke pozive.	Dinamički poziv je moguć ako interfejs nasledjuje interfejs

Slika 5.2 Poređenje u odnosu na faktore vezane za implementaciju [Izvor: Autor]

POREĐENJE NA BAZI PROTOKOLA ZA KOMUNIKACIJU I KORIŠĆENIH RESURSA

Prikaz poređenja CORBA, DCOM, RMI tehnologija na bazi protokola za komunikaciju

U sledećoj tabeli su prikazani protokoli za komunikaciju, kao i resursi sistema koji se koriste.

RMI	CORBA	DCOM
Kao protokol za komunikaciju se koristi Java Remote Method Protocol (JRMP)	Kao protokol za komunikaciju se koristi Internet-Inter ORB (IIOP) protokol.	Kao protokol za komunikaciju se koristi Object Remote Procedure Call (ORPC)
Za pronalaženje i aktiviranje implementacije objekta odgovorna je JVM.	Za aktiviranje objekta je odgovoran Basic Object Adapter (BOA) ili Portable Object Adapter (POA), dok je ORB odgovoran za pronalaženje objekta.	Za aktiviranje i pronalaženje objekta je odgovoran DCOM Service Control Manager.

Slika 5.3 Protokoli za komunikaciju RMI, CORBA, DCOM [Izvor: Autor]

POREĐENJE NA OSNOVU DODATNIH KARAKTERISTIKA

CORBA, DCOM, RMI se mogu uporediti i na osnovu mogućnost kreiranja objekata, mogućnosti distribuiranog skupljanja i sl.

RMI	CORBA	DCOM
Omogućava kreiranje objekta RMI SecurityManager. Ovim se osigurava da objekat koji ide do klijenta ne pristupa resursima sistema.	CORBA Security servis podržava identifikaciju, autorizaciju, autentikaciju i kontrolu pristupa. Omogućava se i praćenje sigurnosti.	Sigurnost je podržana time što se korisniku omogućava da definiše autentikaciju na nivou klijenta, kao i nivo pristupa za objekte.
Distribuisanim skupljanjem otpada rukuje JVM.	Ne postoji distribuisano skupljanje otpada.	Distribuisano skupljanje otpada se aktivira preko mehanizma na osnovu kojeg objekat server detektuje da li je klijent povezan.

Slika 5.4 Prikaz poređenja na osnovu dodatih karakteristika [Izvor: Autor]

POREĐENJE PERFORMANSI

Na osnovu ping eksperimenta, može se zaključiti da je RMI najbrži, i to kada se parametri prosleđuju po vrednosti. Kada je reč o množenju matrica vidi se da je najbrža CORBA.

Poređenje performansi je izvršeno na bazi rezultata pingovanja, kao i množenja matrice vektorom. U ping primeru, je niz sa 100 elemenata tipa float poslat po vrednosti na udaljeni server. Klijentu je vraćen obrnuti niz. U tabeli je prikazano prosečno vreme za 10 takvih poziva.

Kod množenja matrice vektorom, je množena matrica 2×100 sa vektorom 100×1 . Serveru su poslate reference na dva ulazna objekta, a klijentu je vraćena referenca na matricu koja se dobija kao proizvod. U tabeli je prikazano prosečno vreme za 10 ovakvih množenja.

Eksperiment	Prosleđivanje parametara	RMI (ms)	CORBA (ms)	DCOM (ms)
Ping	Po vrednosti	25.792	163.283	135.545
Množenje matrice i vektora	Po referenci	6781.155	1546.716	123 305.330

Slika 5.5 Prikaz poređenja na osnovu performansi [Izvor: Autor]

Ako se pogledaju rezultati ping eksperimenta, može se zaključiti da je RMI najbrži, i to kada se parametri prosleđuju po vrednosti. Moguć razlog ovakvog rezultata može biti efikasnost serijalizacije objekata u Javi.

Kada je reč o množenju matrica vidi se da je najbrža CORBA. Moguć razlog je da je prosleđivanje objekata po referenci u CORBA specifikaciji podržano preko procesa stringifikacije i destringifikacije. Veliko vreme kod DCOM tehnologije je rezultat toga što je DCOM koji je korišćen u eksperimentu bio proizvod nezavisnog proizvođača, kao i načina prenosa referenci na objekte u DCOM tehnologiji.

▼ Poglavlje 6

Veb servisi

OPŠTI MIDDLEWARE SERVISI

Primer ovih servisa su CORBA servisi, koje promoviše OMG.

Opšti middleware servisi se prave uz upotrebu middlewarea za distribuciju, ali su usmereni na određenu poslovnu logiku. Cilj je pojednostavljenje razvoja u tom smeru. U većini slučajeva ovi servisi su proširenje prethodnog nivoa za distribuciju. Primer ovih servisa su CORBA servisi, koje promoviše OMG.

Jedan prilično uspešan primer servisa iz ove grupe su EJB ([Enterprise JavaBeans](#)) firme Sun, koji omogućavaju programerima da distribuisani sistem prave povezivanjem unapred napravljenih softverskih servisa (binova).

WEB SERVISI I MIDDLEWARE

U najširem smislu Veb servis je aplikacija koja se može publikovati, pronaći i pozvati preko interneta

Kao što smo videli iz prethodnih primera sva tri distribuirana tipa middleware tehnologija funkcionišu na sličan način. Razlike su više u tome koje su karakteristike podržane, kao i nivou složenosti. U svim slučajevima, rezultat je blisko povezivanje klijenta i servera. Zbog različitih protokola DCOM server se ne može pozvati od strane RMI klijenta (bez dodatnih problema). Pored toga pomenute middleware tehnologije se koriste uglavnom kod intranet aplikacija, tako da je pitanje kako to funkcioniše kroz [firewall](#). Naravno da je to moguće (preko HTTP tunela), ali to nije prirodno stanje ovakvih aplikacija.

U najširem smislu Veb servis je aplikacija koja se može publikovati, pronaći i pozvati preko interneta. Tipični primeri mogu biti dobijanje informacija o ceni, dobijanje vremenskih izveštaja, rezervacija avionskih karata i sl.

Ovi veb servisi mogu za obavljanje svog posla koristiti druge veb servise. Očigledno je da nema velike razlike između veb servisa i servera kod distribuisanih aplikacija. Razlika je u osnovnim nivoima koji obavljaju logiku aplikacije i manipulaciju podacima.

Ono što je kod distribuisanih tehnologija koje smo opisali pomenuto kao nedostatak je i glavni razlog što ove tehnologije ne mogu da budu veb servisi. U WWW svetu, postoji heterogeno okruženje, kako na strani servera tako i na strani klijenta, tako da nije unapred poznato koja vrsta middleware tehnologija se nalazi na drugoj strani. Zbog toga je za veb servise bio potreban novi pristup.

DEFINICIJA VEB SERVISA

Veb servisi su učaurene, slabo povezane, ugovorene funkcije, koje se nude preko standardnog protokola

Definicija veb servisa koja postoji na adresi WebServices.org je:

Veb servisi su učaurene, slabo povezane, ugovorene funkcije, koje se nude preko standardnog protokola.

pri čemu:

- Učaurene znači da se implementacija funkcija nikada ne vidi spolja
- Slabo povezane znači da implementacija funkcije ne traži promenu funkcije koja je poziva.
- Ugovorene znači da postoji publikovan opis ponašanja funkcije, načina povezivanja sa funkcijom, kao i ulaznih i izlaznih parametara.

Veb servisi:

- su komponente aplikacija
- komuniciraju koristeći otvorene protokole
- su samostalni i sami sebe opisuju
- mogu ih koristiti druge aplikacije
- HTTP i XML su osnova za veb usluge

Korišćenjem veb servisa, aplikacija može objaviti svoju funkciju ili poruku ostatku sveta. Veb usluge koriste XML za kodiranje i dekodiranje podataka, a SOAP za transport (koristeći otvorene protokole).

ARHITEKTURA DISTRIBUIRANIH APLIKACIJA

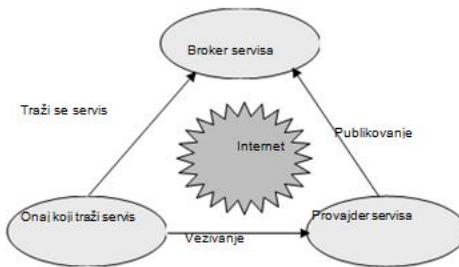
Pošto veb servisi takođe predstavljaju distribuisane aplikacije, oni sadrže posrednika, provajdera usluga i onoga ko traži usluge

U tradicionalnom klijent/server modelu postoji server koji nudi neke funkcije, koje klijent može da koristi. Između klijenta i servera postoji neka vrsta servisa za pretraživanje.

Pošto veb servisi takođe predstavljaju distribuirane aplikacije, oni sadrže iste tri komponente.

- **Posrednika (broker)** koji se ponaša kao servis za pretraživanje između onog ko daje usluge i onog ko ih traži.
- **Provajder usluga** je onaj koji publikuje svoje usluge do ovog posrednika.
- **Onaj koji traži usluge** i koji od posrednika traži da mu pronađe odgovarajućeg provajdera.

Veza između ovih komponenti je ilustrovana na sledećoj slici.



Slika 6.1.1 Prikaz klijent-server modela [Izvor: Autor]

Middleware tehnologije koje smo spominjali za komunikaciju koriste neku vrstu binarnog protokola. Veb servisi koriste XML i to preko HTTP protokola. To znači da nema problema koje može da doneše **firewall**. Obično **firewall** ne blokira HTTP port. Ako pogledate definiciju videćete da veb servisi ne moraju da koriste HTTP. Umesto toga može da se koristi i FTP ili SMTP. XML je format za razmenu podataka koji je široko rasprostranjen. On je osnovni gradivni element skoro svih nivoa koji se koriste kod Veb servisa.

Ovi nivoi zajedno čine tzv. stek veb servisa. On se sastoji od sledećih delova:

- XML (**Extensible Markup Language**)
- SOAP (**Simple Object Access Protocol**)
- WSDL (**Web Services Definition Language**)
- UDDI (**Universal Discovery Description Integration**)



Slika 6.1.2 Prikaz stek veb servisa [Izvor: Autor]

✓ 6.1 SOAP

ŠTA JE SOAP?

Simple Object Access Protocol (SOAP)

Slično kao što različiti ljudi pod veb servisima podrazumevaju različite stvari, tako i SOAP različitim ljudima predstavlja različite stvari. Da li je to RPC mehanizam? Da li je to standard za komunikaciju softvera različitih proizvođača na različitim platformama? Da li je to protokol za razmenu dokumenata? Da li je to jezik za **business-to-business** komunikaciju?

U suštini SOAP je sve od gore pomenutog. Možda je najbolji način da se opiše SOAP da se objasni skraćenica njegovog imena.

S- SIMPLE (JEDNOSTAVAN)

Osnovni pristup sa predstavljanjem podataka preko XML-a i njihov transport preko Interneta uz upotrebu HTTP protokola je jednostavan

S u skraćenici SOAP znači **simple** (jednostavan). Osnovni pristup sa predstavljanjem podataka preko XML-a i njihov transport preko Interneta uz upotrebu HTTP protokola je jednostavan. Kod SOAP protokola se sve što ide preko mreže izražava HTTP i SMTP zaglavljima, MIME kodiranjem i specijalnom XML gramatikom za kodiranje podataka i objekata.

Da li je SOAP poruka baš tako jednostavna? Na primer, da li je ideja predstavljanja SOAP dokumenta sa attachment-ima, preko mail i MIME metafora jednostavna. Ona je jednostavna zato što se koristi konvencija koju većina poznaje. Da li je XML jednostavan? On može biti jednostavan ili složen, onoliko koliko vi želite da bude. XML obezbeđuje način da se podacima koji se prenose preko mrežedoda neko semantičko značenje. Ipak, XML šema je daleko od toga da bude jednostavna.

SOAP ima konvencije za kreiranje omotača za vaše podatke. On ima eksplisitna pravila za kodiranje podataka aplikacije, čak i za takve stvari kao što su nizovi binarnih podataka, tako da se može predstaviti u obliku čitljivom za ljudi. Ništa tu nije jednostavno, ali se sve može shvatiti i objasniti.

Kod SOAP-a ništa nije namerno skriveno. Svaki aspekt SOAP zahteva sam sebe opisuje i velikim delom je baziran na dokazanim konvencijama. Tu je i prava snaga SOAP-a. Platforme i programski jezici na obe strane SOAP konverzacije su nezavisni jedni od drugih, ali mogu da komuniciraju sve dok je svaka strana te konverzacije u stanju da:

Šalje i prima podatke preko mreže, uz upotrebu HTTP ili SMTP protokola.

- Razume MIME pravila kodiranja, kao i osnove konstruisanja binarnih attachmenta na osnovu tih pravila.
- Konstruiše i rekonstruiše XML dokumente koji odgovaraju pravilima koja postoje u SOAP-u
- Obavlja određenu akciju, ako je akcija naznačena u SOAP dokumentu.

O - OBJECT (OBJEKAT)

SOAP je u potpunosti u stanju da opiše poziv udaljene procedure ili objekta

O u skraćenici SOAP znači "objekat" i vezano je za način na koji se COM objekti pozivaju preko Interneta. SOAP je u potpunosti u stanju da opiše poziv udaljene procedure ili metoda. Evo jednog tipičnog SOAP dokumenta koji opisuje poziv metoda udaljenog objekta:

```
POST /StockQuote HTTP/1.1
Host: www.example.org
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
```

```
SOAPAction: "http://example.org/2001/06/quotes"
<env:Envelope xmlns:env="http://www.w3.org/2001/09/soap-envelope" >
  <env:Body>
    <m:GetLastTradePrice
      env:encodingStyle="http://www.w3.org/2001/09/soap-encoding"
      xmlns:m="http://example.org/2001/06/quotes">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

Ne ulazeći u strukturu ove poruke, može se napomenuti da su ovde bitni ime metoda (GetLastTradePrice), kao i njegov parametar DIS.

A-ACCESS (PRISTUP)

Ključna osobina SOAP-a i veb servisa je njihova pristupačnost.

Ključna osobina SOAP-a i veb servisa je njihova pristupačnost. Inicijalno je namera onih koji su pravili SOAP bila da se SOAP konverzacija odvija vezivanjem za drugi protokol nižeg nivoa, najverovatnije HTTP-a ili SMTP-a. Ovi protokoli su izabrani zato što su skoro univerzalno dostupni. Većina firewall-a su u stanju da dozvoljavaju HTTP sesije i SMTP razmene, tako da SOAP konverzacija može da lako prelazi granice.

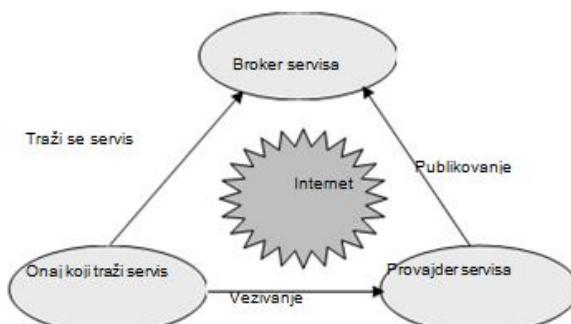
Sa druge strane SOAP povezivanje se može napraviti za bilo koji protokol. U ovom trenutku najviše se koristi HTTP, ali je moguć i SOAP preko RMI-a, ili JMS-a.

P- PROTOKOL

Kako se SOAP koristi u okruženju veb servisa?

Kada se svi ovi faktori uzmu zajedno dobija se protokol. SOAP je protokol baziran na XML-u koji se koristi za razmenu informacija u distribuisanom okruženju.

Sledeća slika pokazuje kako se SOAP koristi u okruženju veb servisa.



Slika 6.2.1 Korišćenje SOAP u okruženju veb servisa [Izvor: Autor]

SINTAKSNA PRAVILA

SOAP poruka MORA biti kodirana pomoću XML-a

Evo nekoliko važnih sintaksnih pravila:

- SOAP poruka **MORA** biti kodirana pomoću XML-a
- SOAP poruka **MORA** koristiti SOAP Envelope namespace
- SOAP poruka **MORA** koristiti SOAP Encoding namespace
- SOAP poruka **NE SME** sadržati DTD referencu
- SOAP poruka **NE SME** sadržati uputstva za obradu XML-a

Primer strukture SOAP poruke:

```
<?xml version="1.0"?>

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

  <soap:Header>
  ...
  </soap:Header>

  <soap:Body>
  ...
    <soap:Fault>
    ...
    </soap:Fault>
  </soap:Body>

</soap:Envelope>
```

SOAP Envelope element je koreni element SOAP poruke. Ovaj element definiše XML dokument kao SOAP poruku.

```
<?xml version="1.0"?>

<soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  ...
  Informacija poruke ide ovde
  ...
</soap:Envelope>
```

SOAP - VIDEO

Understand the Difference Between SOAP and REST APIs

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

SOAP I REST - VIDEO

REST Vs SOAP - What is the difference? | Tech Primers

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 6.2 WSDL

ŠTA JE WSDL?

WSDL (jezik za opis web servisa) opisuje mrežne servise. Za opis se koristi XML

WSDL (jezik za opis web servisa) opisuje mrežne servise. Za opis se koristi XML. Distribuiranim sistemima on obezbeđuje dokumentaciju i ima za cilj da omogući aplikacijama da komuniciraju na automatizovani način.

Dok SOAP definiše komunikaciju između onog ko traži i pruža usluge, WSDL opisuje servis koju pružalač usluga daje. On se može koristiti kao recept za generisanje SOAP poruka, preko kojih se servisu pristupa.

Na jednom nivou WSDL se ne razlikuje od IDL jezika koji postoje kod CORBA specifikacije ili Microsofta. Svi ovi jezici se koriste za definisanje interfejsa (signature metoda) i tipova podataka za diskretne delove programske logike.

Na drugom nivou WSDL je sasvim drugačiji, jer nudi jedan stepen mogućnosti proširenja koji ne postoji kod IDL specifikacije. Ova mogućnost proširenja omogućava da se WSDL koristi za:

- Opisivanje krajnjih tačaka i njihovih poruka, bez obzira na format poruka ili mrežni protokol koji se koristi.
- Tretiranje poruka kao apstraktnih opisa podataka koji se razmenjuju.
- Tretiranje tipova portova kao apstraktnih skupova operacija web servisa. Tip porta se kasnije može mapirati u konkretni protokol i format podataka.

Kako broj različitih formata za komunikaciju i protokola koji se koriste na Internetu nastavlja da raste, postaje vrlo bitno pronaći standardni način za opis načina na koji dve mašine treba da komuniciraju. WSDL opisuje šta servis radi, kako se pozivaju njegove operacije i gde se mogu pronaći. U WSDL-u postoje posebne definicije i terminologija za definisanje web servisa, krajnjih tačaka u komunikaciji gde se servisi nalaze, formata za poruke koje se šalju i primaju od servisa, kao i apstraktan način povezivanja sa konkretnim protokolom i formatom podataka.

Sve što se u WSDL datoteci definiše je apstraktno. Tu su samo definicije parametara i ograničenja, koji govore kako se komunikacija treba da odvija. Implementacija veb servisa mora da se pridržava onog što je zadato u WSDL datoteci, mada postoje i neke mogućnosti za definisanje specifičnosti. WSDL daje i mogućnost definisanja povezivanja, kojim se konkretnom protokolu i formatu podataka pridružuje jedan apstraktan skup definicija poruka. Tzv. bindingextension je tip povezivanja koji je definisan za glavni protokol. Ovakva proširenja postoje za SOAP 1.1, HTTP GET, HTTP POST i MIME.

KAKO KOD OPISA SERVISA NASTAJE KOD?

Sa stanovišta programiranja, mogućnost generisanja koda na osnovu WSDL-a je jedna od najvećih prednosti. Postoje i metodi za generisanje WSDL-a na osnovu postojećih komponenti.

Pošto je WSDL samo apstraktни opis interfejsa veb servisa, onda je razumljivo da se na osnovu WSDL definicije može napraviti kod za implementaciju, kao i da se WSDL definicije mogu automatski napraviti na osnovu postojećeg koda. Sa stanovišta programiranja, mogućnost generisanja koda na osnovu WSDL-a je jedna od najvećih prednosti. Takođe postoje i metodi za generisanje WSDL-a na osnovu postojećih komponenti. Obe tehnike su prava blagodet za programere.

Pitanje koje programeri treba da sebi postave, je šta treba prvo napraviti? Da li se prvo pravi implementacija servisa, nakon čega se automatski generišu interfejsi? Ili se prvo pravi WSDL za veb servis, a onda se pomoću alata prave odgovarajuće komponente, koje taj servis implementiraju?

Odgovor je da je verovatno bolje da se programeri usresrede na kreiranje veb servisa i njegovih metoda. Tek kada se napravi implementacija i kada je programer siguran da sve radi kako treba, mogu se upotrebiti alati koji automatski napraviti WSDL datoteke. Većina alata koji danas postoje na tržištu su u stanju da to urade. Na primer, **CapeConnect** automatski generiše WSDL, WebLogic Server omogućava da napravite EJB ili JMS odredište, a da onda preko Ant skripta napravite WSDL.

WSDL DOKUMENT

WSDL dokument opisuje veb servis

WSDL dokument opisuje veb servis. On određuje lokaciju usluge i metode usluge, koristeći svoje glavne elemente:

- <types> - definiše tipove podataka koje koriste veb servisi (XML schema)
- <message> - definiše elemente podataka za svaku operaciju
- <portType> - opisuje operacije koje mogu da se obavljaju, kao i poruke
- <binding> - definiše protokol i format podataka za svaki tip porta.

Osnovna struktura WSDL dokumenta izgleda ovako:

```
<definitions>

<types>
    definicija tipa podataka .....
</types>

<message>
    definicija podataka koji se prenose .....
</message>

<portType>
    skup operacija .....
</portType>

<binding>
    specifikacije protokola i formata podataka .....
</binding>

</definitions>
```

Uprošćeni primer jednog XML dokumenta:

```
<message name="getTermRequest">
    <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
    <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
    <operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
    </operation>
</portType>
```

U ovom primeru element `<portType>` definiše "glossaryTerms" kao ime porta, a "getTerm" kao naziv operacije.

Operacija "getTerm" ima ulaznu poruku pod nazivom "getTermRequest" i izlaznu poruku pod nazivom "getTermResponse".

Elementi `<message>` definišu delove svake poruke i povezanih tipova podataka.

ELEMENT

Tip request-response je najčešći tip operacija

`<portType>` element definiše web servise, operacije koje se mogu obaviti i poruke. Tip request-response je najčešći tip operacija, ali WSDL definiše 4 tipova:

- Jednosmerna (engl. **One-way**) - ova operacija može da primi poruku, ali ne može da vrati odgovor
- Zahtev-odgovor (engl. **Request-response**) - ova operacija može da primi zahtev i da pošalje odgovor
- Traženje odgovora (engl. **Solicit-response**) - ova operacija može da pošalje zahtev i čeka odgovor
- Notifikacija (engl. **Notification**) - ova operacija može da pošalje poruku, ali neće čekati odgovor

Primer one-way operacije je:

```
<message name="newTermValues">
  <part name="term" type="xs:string"/>
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="setTerm">
    <input name="newTerm" message="newTermValues"/>
  </operation>
</portType >
```

U prethodnom primeru, portType "glossaryTerms" definiše jednosmernu operaciju pod nazivom "setTerm". Operacija "setTerm" omogućava unos novih termina koristeći "newTermValues" poruku sa ulaznim parametrima "term" i "value". Međutim, za operaciju nije definisan nikakav izlaz.

Primer operacije zahtev-odgovor:

```
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

U prethodnom primeru, portType "glossaryTerms" definiše operaciju zahteva i reakcije pod nazivom "getTerm". Operacija "getTerm" zahteva ulaznu poruku pod nazivom "getTermRequest" sa parametrom pod nazivom "term", i želi da vrati izlaznu poruku pod nazivom "getTermResponse" sa parametrom pod nazivom "value".

❖ 6.3 UDDI

ŠTA JE UDDI?

UDDI (univerzalni opis, otkrivanje i integracija) projekat daje standardizovan način za publikovanje i otkrivanje informacija o veb servisima.

UDDI (univerzalni opis, otkrivanje i integracija) projekat daje standardizovan način za publikovanje i otkrivanje informacija o veb servisima. Ovaj projekat je inicijativa koja dolazi iz industrije, koja pokušava da napravi framework za opis servisa, otkrivanje onog što se nudi i integraciju servisa, pri čemu sve to ne sme da zavisi od platforme koja se koristi. Osnovna namena UDDI-a u arhitekturi servisa je pronalaženje.

Veb servisi su postali osnova za sve oblike elektronskog poslovanja. Preduzeća pozivaju servise drugih preduzeća da bi obavila svoje poslovne transakcije. U okruženju u kome učestvuјe samo nekoliko preduzeća, manuelno pronalaženje poslovnih partnera može biti jednostavno. Koliko može biti složeno da se pronađe da li jedan od vaših nekoliko poslovnih partnera može da zadovolji vaše potrebe.

Ovaj model, međutim, pada u vodu, kada se broj preduzeća sa kojima komunicirate povećava, zajedno sa brojem i tipom interfejsa koje ona nude. Kako ćete pronaći sve poslovne partnere sa kojim biste mogli da sarađujete. Ako pokušate da ih manuelno tražite, nikad nećete biti sigurni da ste pronašli svakog mogućeg partnera. UDDI je registar koji je distribuisan na mnogo mesta. UDDI registar servisa pokušava da reši nabrojane probleme.

PRISTUP UDDI SERVISIMA

Da bi se omogućio pristup do UDDI servisa, UDDI adresar izlaže skup API-ja, u formi SOAP veb servisa

Pre UDDI projekta, nije postojao široko prihvaćen način da jedno preduzeće pronađe svoje kupce i partnere, sa informacijama o njihovim proizvodima i uslugama. Takođe, nije postojao univerzalan metod za integraciju sistema i procesa koji već postoje.

U UDDI registru se konceptualno mogu registrovati tri tipa informacija. Specifikacija ne navodi eksplicitno ove tipove, ali navodi šta UDDI može da čuva o nekom poslu.

- **Bele strane:** Ovo su osnovne informacije za kontakt, kao i identifikaciju preduzeća, uključujući ime, adresu, informacije za kontakt, kao i jedinstvene identifikatore (za porez i sl.). Ove informacije omogućavaju drugima da pronađu vaš veb servis na osnovu poslovnih informacija.
- **Žute strane:** Ovo su informacije koje opisuju veb servis. Koriste se različite taksonomije. Te informacije omogućavaju drugima da pronađu veb servis na bazi njegove kategorizacije (da li je to proizvodnja ili prodaja automobila).

- **Zelene strane:** Ovo su tehničke informacije koje opisuju ponašanje i funkcije koje veb servis podržava. Tu su informacije koje opisuju kako se veb servis koristi i gde se nalazi.

Da bi se omogućio pristup do UDDI servisa, UDDI adresar izlaže skup API-ja, u formi SOAP veb servisa. API je podeljen na dva logička dela. To su API za raspitivanje i API za publikovanje. API zaraspitivanje se dalje deli na dva dela. Jedan se koristi za konstruisanje programa koji omogućavaju pretraživanje informacija u UDDI registru, a drugi se koristi u slučaju da poziv ne uspe.

Osnovna komponenta je UDDI registracija posla. Ovo je XML datoteka, koja opisuje poslovni entitet i njegove veb servise. Specifikacija definiše protokol zasnovan na SOAP-u i HTTP-u, koji se koristi za pronalaženje i registrovanje servisa.

▼ 6.4 RESTFUL VEB SERVISI

OSNOVE RESTFUL VEB SERVISA

Danas se RESTfull izdvojio kao dominantan mrežni servis, potisnuo je SOAP i WSDL jer je značajno jednostavniji za korišćenje.

Representational State Transfer (**REST**) opisuje bilo koji jednostavan interfejs koji prenosi podatke preko standardizovanog interfejsa (kao što je HTTP) bez dodatnog sloja razmene poruka, kao što je SOAP.

Ovi servisi su jednostavnije integrисани sa HTTP-om od SOAP servisa, ne zahtevaju XML poruke ili WSDL opise servisa. Danas se RESTfull izdvojio kao dominantan mrežni servis, potisnuo je SOAP i WSDL jer je značajno jednostavniji za korišćenje.

Podaci se najčešće prebacuju u JSON formatu mada je dostupan i XML i YAML format. Zasniva se na REST arhitekturi, veoma je fleksibilan i jednostavan za razumevanje. Može biti izvršen na bilo kom klijentu ili serveru koji ima HTTP/HTTPS podršku. RESTful servisi treba da imaju sledeće osobine i karakteristike:

- Nepostojanje stanja (engl. **stateless**)
- Mogućnost keširanja (engl. **cacheable**)
- Uniformni interfejs (engl. **uniform interface URI**)
- Izričito korišćenje HTTP metoda
- Transfer XML i/ili JSON

REST pruža set pravila za kreiranje usluga koje se smatraju resursima, ili izvorima konkretnih informacija i mogu se identifikovati po jedinstvenom URI-u. Klijent pristupa resursu pomoću URI-a i standardizovanom fiksiranom skupu metoda. Kada se koristi HTTP protokol za pristup RESTful resursima, identifikator resursa je URL izvora i standardna operacija koja se obavlja na tom resursu je jedna od HTTP metoda: GET, PUT, DELETE, POST ili HEAD.

Web programiranje, <https://www.webprogramiranje.org/web-servisi-osnove/>

▼ Poglavlje 7

Pokazna vežba – XPATH, XSLT

XPATH - VEŽBA

Xpath je jezik koji predstavlja osnovni elementi XSLT jezika, ima biblioteku standardnih funkcija, koristi path izraze za navigaciju i prestavlja sintaksu za definisanje XML dokumenta.

XPath je jezik koji:

- je W3C standard,
- predstavlja osnovni element XSLT jezika,
- ima biblioteku standardnih funkcija,
- koristi tzv. path izraze za navigaciju u XML dokumentima,
- predstavlja sintaksu za definisanje delova XML dokumenata...

XPath predstavlja preduslov za korišćenje XSLT standarda. Ima preko 100 ugrađenih funkcija za rad sa stringovima, numeričkim vrednostima, za upoređivanje datuma i vremena, boolean vrednosti itd. Uz pomoć path izraza XPath može da koristi tzv. čvorove ili skupove čvorova u XML dokumentu.

Kao što je već rečeno, XML dokumenti smatraju se zapravo stablima koja se sastoje od čvorova. Čvorovi se sa aspekta XPath dele na: elemente, attribute, prostore imena, instrukcije procesiranja, komentare i dokument. U datom već korišćenom XML dokumentu:

```
<fakultet>
  <student>
    <indeks>1</indeks>
    Petar</ prezime>
    <ime>Marković</ime>
  </student>
</fakultet>
```

Čvor `<fakultet>` predstavlja čvor-dokument, a `<prezime>` čvor-element. Svaki element ima roditelja. U ovom primeru `<student>` predstavlja roditelja za `<indeks>`, `<prezime>` i `<ime>`. Generalno, čvor-element nema nijedno ili ima jedno ili više deteta. Svi navedeni čvorovi su deca čvora `<student>`. Braća su čvorovi koji imaju istog roditelja.

XPATH – ČESTO KORIŠĆENI IZRAZI

Prikaz često korišćenih izraza u Xpath-u

Svi navedeni čvorovi ujedno su i braća. Čvor se naziva predak ukoliko je roditelj nekog čvora, odnosno roditelj roditelja itd. U ovom primeru preci su čvorovi fakultet i student. Na sličan način potomke čvora fakultet predstavljaju svi ostali čvorovi. XPath koristi, kao što je rešeno, tzv. path izraze – putanje, za izbor čvorova XML dokumenta.

Na slici 1 dati su primeri često korišćenih izraza.

Putanja	Opis
<code>nodename</code>	Bira sve čvorove-decu datog čvora
<code>/</code>	Bira iz korenog čvora
<code>//</code>	Bira čvorove u dokumentu počev od zadatog čvora bez obzira gde se nalaze
<code>.</code>	Bira zadati čvor
<code>..</code>	Bira čvor-roditelj datog čvora

Slika 7.1 Prikaz često korišćenih izraza [Izvor: Autor]

Na slici 2 dati su neki izrazi za dati primer XML dokumenta.

Putanja	Rezultat
<code>fakultet</code>	Bira sve čvorove-decu elementa fakultet
<code>/fakultet</code>	Bira koreni element fakultet
<code>fakultet/student</code>	Bira sve elemente student koji su deca čvora fakultet
<code>//student</code>	Bira sve elemente student bez obzira na to gde se nalaze u dokumentu
<code>fakultet//student</code>	Bira sve elemente student koji su potomci elementa fakultet, bez obzira na to gde se nalaze ispod elementa fakultet u stablu

Slika 7.2 Izrazi za dati primer XML dokumenta [Izvor: Autor]

XPATH - PRIMERI IZRAZA-PUTANJA SA PREDIKATIMA

Prikaz izraza - putanja sa prediktima

Predikati se koriste za nalaženje određenog čvora ili čvora koji ima određenu vrednost. Predikati su uvek navode u srednjim zagradama.

Na slici 3 dati su primeri izraza-putanja sa predikatima, kao i rezultati njihovog izvršenja.

Putanja	Opis
/fakultet/student[1]	Bira se prvi element student koji je dete elementa fakultet
/fakultet/student[last()]	Bira poslednji element student koji je dete elementa fakultet
/fakultet/student[last()-1]	Bira se pretposlednji element student koji je dete elementa fakultet
/fakultet/student[position()<3]	Biraju se prva dva elementa student koji su deca elementa fakultet
/fakultet/student[prosek>8.00]	Biraju se svi elementi student koji imaju prosek studiranja veći od 8.00
/fakultet/student[prosek>8.00]/prezime	Biraju se svi elementi prezime, elementa student, elementa fakultet čiji je prosek veći od 8.00

Slika 7.3 Prikaz primera izraza – putanja sa prediktima i rezultata njihovog izvršavanja [Izvor: Autor]

PRIKAZ WILDCARD SIMBOLA

Prikazane su tabele sa wildcard simbolima.

Za pristup nepoznatim elementima XML dokumenta koriste se wildcard simboli. Na slici 4 je navedeno par ovih simbola. Na slici 5 je takođe dat prikaz korišćenja wildcard simbola.

Wildcard	Opis
*	Odgovara bilo kom element-čvoru
node()	Odgovara bilo kom čvoru, bilo kog tipa

Slika 7.4 Wildcard simboli [Izvor: Autor]

Tabela 14.5	
Putanja	Rezultat
/fakultet/*	Bira sve čvorove-decu elementa fakultet
//*	bira sve elementa u dokumentu

Slika 7.5 Primeri korišćenja wildcard simbola [Izvor: Autor]

U XPath koriste se ose za izbor skupa čvorova u odnosu na dati čvor. Na slici 6 dati su nazivi osa i odgovarajući rezultati.

Naziv ose	Rezultat
ancestor	Bira sve pretke datog čvora
ancestor-or-self	Bira sve pretke datog čvora i sam čvor
child	Bira svu decu-čvorove datog čvora
descendant	Bira sve potomke datog čvora
descendant-or-self	Bira sve potomke datog čvora i sam čvor
following	Bira sve u dokumentu nakon završnog taga datog čvora
following-sibling	Bira svu braću nakon datog čvora
namespace	Bira sve prostore imena čvorove datog čvora
parent	Bira roditelja datog čvora
preceding	Bira sve u dokumentu što se nalazi pre početnog taga datog čvora
preceding-sibling	Bira svu braću pre datog čvora
self	Bira dati čvor

Slika 7.6 Nazivi osa i odgovarajući rezultati [Izvor: Autor]

KREIRANJE LOKACIJE ČVORA

Lokacija nekog čvora može biti absolutna ili relativna.

Lokacija nekog čvora može biti absolutna ili relativna. U oba slučaja lokaciju čini niz koraka razdvojenih znakom „/“ na način prikazan na slici 7.

/korak/korak/...	Absolutna lokacija
korak/korak/...	Relativna lokacija

Slika 7.7 Prikaz dodavanja lokacije – absolutna i relativna [Izvor: Autor]

Svaki korak se upoređuje sa čvorovima u datom skupu čvorova. Korak se sastoji od:

- ose, koja definiše odnos između izabranog i datog čvora,
- čvor testa, koji identificuje čvor unutar ose,
- predikata, kojih i ne mora biti, a koji detaljnije definišu izabrani skup čvorova. Generalno, sintaksa lokacije je:

ime_ose::čvor_test[predikat]

Neki primeri su dati na slici 8.

Primer	Rezultat
child::student	Bira sve čvorove student koji su deca datog čvora
child::*	Bira svu decu datog čvora
child::text()	Bira tekstove dece datog čvora
child::node()	Bira svu decu datog čvora
descendant::student	Bira sve čvorove student potomke datog čvora
ancestor::student	Bira sve student čvorove pretke datog čvora
ancestor-or-self::student	Bira sve čvorove student pretke datog čvora, kao i dati čvor pod uslovom da je u pitanju čvor student

Slika 7.8 Primer nekih funkcija [Izvor: Autor]

PRIKAZ OPERATORA XPATH-A

Data je lista operatora koje se mogu koristiti u XPath-u.

Na slikama 9 i 10 nalazi se lista operatora koji se mogu koristiti u XPath izrazima:

Operator	Opis	Primer	Rezultat
	Određuje dva skupa čvorova	//student //profesor	Vraća skup čvorova koji uključuje sve čvorove profesor i student
+	Zbir	6+4	10
-	Razlika	6-4	2
*	Množenje	6*4	24
div	Deljenje	8 div 4	2
=	Jednakost	prosek=8.80	true ako je prosek 8.80, odnosno false u ostalim slučajevima

Slika 7.9 Lista operatora koji se mogu koristiti u XPathu [Izvor: Autor]

Operator	Opis	Primer	Rezultat
<	Manje od	prosek<8.80	true ako je prosek manji od 8.80, false u ostalim slučajevima
<=	Manje od ili jednako	prosek<=8.80	true ako je prosek manji od ili jednak 8.80, false u ostalim slučajevima
>	Veće od	prosek>8.80	true ako je prosek veći od 8.80, false u ostalim slučajevima
>=	Veće od ili jednako	prosek>=8.80	true ako je prosek veći od ili jednak 8.80, false u ostalim slučajevima
or	ili	prosek=8.80 or prosek=8.70	true ako je prosek jednak 8.80 ili 8.70, false u ostalim slučajevima
and	i	prosek>8.00 and prosek<8.80	true ako je prosek između 8.00 i 8.80, false u ostalim slučajevima
mod	Modul (ostatak pri deljenju)	5 mod 2	1

Slika 7.10 Lista operatora koji se mogu koristiti u XPathu [Izvor: Autor]

PRIMER 1 - XPATH

Za ovaj primer može se koristiti XML dokument iz prethodnog primera proširen traženim elementima

(Predviđeno vreme izrade zadatka iznosi 20 minuta.)

Zadatak

Za zadatu XML datoteku, koristeći XPath jezik pronaći sve studente koji studiraju na klasičan način ili imaju prosečnu ocenu veću od 8.00. Struktura XML dokumenta je:

```

<fakultet>
    <student>
        <indeks/>
        <ime/>
        <nacin/>
        <prosek/>
    </student>
</fakultet>

```

Rešenje: Za ovaj primer može se koristiti XML dokument iz prethodnog primera proširen traženim elementima:

```

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="primer2.xsl"?>
<fakultet>
    <student>
        <indeks>1</indeks>

```

```
Marković</ prezime>
<ime>Petar</ime>
<nacin>klasicno</nacin>
<prosek>7.77</prosek>
</student>
<student>
<indeks>2</indeks>
Živanović</ prezime>
<ime>Živan</ime>
<nacin>internet</nacin>
<prosek>8.20</prosek>
</student>
<student>
<indeks>3</indeks>
Trnavčević</ prezime>
<ime>Radojka</ime>
<nacin>klasicno</nacin>
<prosek>9.00</prosek>
</student>
<student>
<indeks>4</indeks>
Stolić</ prezime>
<ime>Jasna</ime>
<nacin>internet</nacin>
<prosek>7.50</prosek>
</student>
</fakultet>
```

PRIMER 1 – SNIMANJE I PRONALAŽENJE

Dokument se snima kao primer2.xml i pronalaze se studenti koji studiraju klasično ili imaju odgovarajući prosek.

Dokument se snima kao primer2.xml.

Za pronalaženje svih studenata koji studiraju klasično ili imaju odgovarajući prosek, koristi se sledeće:

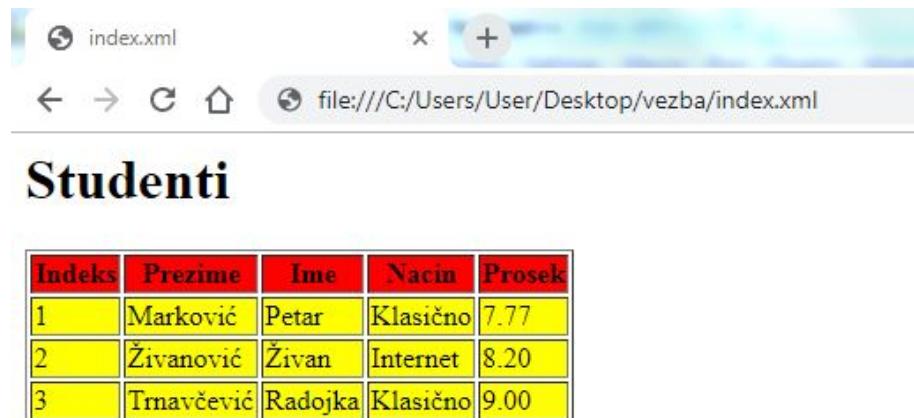
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
<html>
<head />
<body>
<h2>Studenti</h2>
<table border="1">
<tr bgcolor="red">
<th align="left">Indeks</th>
<th align="left">Prezime</th>
<th align="left">Ime</th>
```

```
<th align="left">Nacin</th>
<th align="left">Prosek</th>
</tr>
<xsl:for-each select="/fakultet/student[nacin='klasicno' or
prosek>8.00]">
    <tr bgcolor="yellow">
        <td>
            <xsl:value-of select="indeks" />
        </td>
        <td>
            <xsl:value-of select=" prezime" />
        </td>
        <td>
            <xsl:value-of select="ime" />
        </td>
        <td>
            <xsl:value-of select="nacin" />
        </td>
        <td>
            <xsl:value-of select="prosek" />
        </td>
    </tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

PRIMER 1 - PRIKAZ PRIMERA NAKON POKRETANJA U ČITAČU

Primer otvoren u web čitaču

Ukoliko se kod snimi kao primer2.xsl na poznati način i XML dokument otvori u čitaču dobija se prikaz kao na slici 11.



The screenshot shows a web browser window with the title 'index.xml'. The address bar displays 'file:///C:/Users/User/Desktop/vezba/index.xml'. The main content area contains the heading 'Studenti' and a table with the following data:

Indeks	Prezime	Ime	Nacin	Prosek
1	Marković	Petar	Klasično	7.77
2	Živanović	Živan	Internet	8.20
3	Trnavčević	Radojka	Klasično	9.00

Slika 7.11 prikaz primer2.xml u web čitaču [Izvor: Autor]

XSLT

XSLT ima osobine da prestavlja W3C preporuku, koristi Xpath za navigaciju u XML dokumentima, i najvažniji je deo XSL jezika.

Jedan od zadataka XML jezika je strogo struktuiranje informacija, sa željom da se omogući njihova dosledna obrada i razumevanje. Za te svrhe nije potreban nikakav stil, kao npr. kod HTML dokumenata. Međutim, u pojedinim prilikama, korišćenje stilova je neophodno. Postoji nekoliko tehnika koje omogućavaju primenu stilova kod XML dokumenata. Korišćenje stilova u XML dokumentima podrazumeva opis načina vizuelnog prikazivanja informacija u dokumentu. Za te svrhe koriste se odgovarajući stilski listovi, koji sadrže listu stilova koji će se primeniti na informacije sadržane u XML dokumentu. Kako mnogi XML dokumenti i nisu kreirani za prikazivanje, stilski listovi omogućavaju prikazivanje u npr. Web čitaču.

Stilski list transformiše XML dokument u HTML koji može da se pregleda u čitaču. Na ovaj način sadržaj dokumenta je odvojen od formata, što i predstavlja jedan od ciljeva XML jezika. Tehnologija CSS stilova može se primeniti i na XML dokumente, ali za njih postoji bolja alternativa – XSL. XSL je tehnologija koja se koristi za konverziju XML dokumenata u druge formate, najčešće HTML. Međutim, XSL je jezik opšte namene i koristi se za transformisanje u bilo koji markup jezik. Može se reći da su osnovne funkcije XSL: transformacija XML dokumenata u druge formate i dodavanje stilova uz posebna pravila za formatiranje.

Ova dva zadatka obavljaju različite tehnologije: XSLT (XSL Transformacija) i XSL objekti za formatiranje. XSLT ima sledeće osobine:

- predstavlja W3C preporuku,
- koristi XPath za navigaciju u XML dokumentima,
- XSLT je najvažniji deo XSL jezika.

Najveći problem primene XSL objekata za formatiranje je podrška. Mali broj čitača u potpunosti podržava ovu tehniku. Korišćenje XSLT podrazumeva postojanje jasne hijerarhije u dokumentu sa korenim root elementom na vrhu strukture – stabla. Svi ostali elementi

koji se nalaze ispod korenog smatraju se granama stabla. Koreni element je važan zato što predstavlja početnu tačku za XSL procesor, aplikaciju koja obrađuje XSL stil i koristi ga za transformisanje u npr. HTML dokument. U većini slučajeva ova aplikacija ugrađena je u čitač. Kada XSL procesor obrađuje stilski list, koristi šablone koji opisuju specijalne uzorke u XML dokumentu, odnosno traži informacije koje se podudaraju sa određenim uzorkom. Često je uzorak ime elementa, pa kada XSL procesor nađe na element određenog imena, tada primenjuje određeni šablon za transformisanje podataka.

XSLT - PRIMER

Primer kako se koristi XSLT za transformaciju XML u HTML

Prolazeći kroz čitav dokument, nalazeći odgovarajuće uzorke, primenom šablona XML dokument se transformiše. Ukoliko je rezultat ovog procesa HTML dokument on se može prikazati u web čitaču.

(Predviđeno vreme izrade zadatka iznosi 15 minuta.)

Zadatak

Koristeći XSLT izvršiti transformaciju datog XML dokumenta u HTML dokument.

Struktura XML dokumenta je sledeća:

```
<fakultet>
  <student>
    <indeks/>
    <ime/>
  </student>
</fakultet>
```

Rešenje:

Neka je XML dokument dat kao:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="primer1.xsl"?>
<fakultet>
  <student>
    <indeks>1</indeks>
    <prezime>Marković</prezime>
    <ime>Petar</ime>
  </student>
  <student>
    <indeks>2</indeks>
    <prezime>Živanović</prezime>
    <ime>Živan</ime>
  </student>
  <student>
    <indeks>3</indeks>
    <prezime>Trnavčević</prezime>
```

```
<ime>Radojka</ime>
</student>
<student>
    <indeks>4</indeks>
    Stolić</ prezime>
    <ime>Jasna</ime>
</student>
</fakultet>
```

Ovaj dokument snima se kao primer1.xml. Red obeležen žutom bojom definiše datoteku sa opisanim stilom, koja će se koristiti za transformaciju ovog dokumenta.

STYLESHEET ELEMENT

*Element **stylesheet** je koren element svih XSL stilskih listova i on deklariše prostor imena, koji služi za identifikovanje XSL rečnika*

Deklaracija XSLT u dokumentima vrši se uz pomoć sledećih elemenata:

```
<xsl:transform version="1.0">
< xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

ili potpuno ravnopravno:

```
<xsl:stylesheet version="1.0">
< xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Element **stylesheet** je koren element svih XSL stilskih listova i on deklariše prostor imena, koji služi za identifikovanje XSL rečnika. Ovaj omogućava korišćenje svih elemenata i atributa definisanih u prostoru imena i dodeljuje im prefiks xsl.

Za dati primer koristi stil koji je dat u nastavku i snima se u odgovarajući folder pod nazivom primer1.xsl.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/">
        <html>
            <head />
            <body>
                <h2>Studenti</h2>
                <table border="1">
                    <tr bgcolor="red">
                        <th align="left">Indeks</th>
                        <th align="left">Prezime</th>
                        <th align="left">Ime</th>
                    </tr>
                    <xsl:for-each select="fakultet/student">
                        <tr bgcolor="blue">
```

```
<td>
    <xsl:value-of select="indeks" />
</td>
<td>
    <xsl:value-of select="prezime" />
</td>
<td>
    <xsl:value-of select="ime" />
</td>
</tr>
</xsl:for-each>
</table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

DEKLARISANJE XSL DATOTEKE

Svaka XSL datoteka počinje XML deklaracijom i elementom koji definiše da se radi XSLT dokumentu.

Svaka XSL datoteka počinje XML deklaracijom i elementom koji definiše da se radi XSLT dokumentu. Za element **<xsl:template>** koristi se definisanje šablona, gde atribut match određeni šablon pridružuje pojedinim XML elementima. Atribut match može se koristiti za definisanje obrasca za čitav XML dokument. Vrednost atributa je zapravo XPath izraz, o čemu će više reći biti u nastavku, i u ovom slučaju znak „/“ znači da se radi o celom dokumentu. Može se reći da XSL predstavlja rečnik XML dokumenta. Šabloni transformišu XML dokument tako što XSL procesor pronađe podatak koji je identičan nekom uzorku u stilskom listu i šalje ga šablonu na dalju konverziju. Šablon transformiše dokument tako što počinje korenim elementom i nastavlja se na čitav dokument.

Već je rečeno da se pri obradi koriste uzorci za upoređivanje sa podacima u XML dokumentu. Preciznije, uzorak u XML dokumentu identificiše element ili atribut koji odgovara nekom delu strukture podataka, odnosno nekoj grani stabla. Uzorci se mogu uporediti sa putanjama u sistemima datoteka gde se specifičuju direktorijumi i datoteke. U ovom slučaju uzorci specifičuju elemente i atribute.

Na primer, u standardnom HTML dokumentu, element **<head>** koji se pojavljuje u okviru taga **<html>**, može se identifikovati uz pomoć uzorka: `html/head`. Kada XSL procesor identificiše deo XML dokumenta identičan uzorku, on taj podatak „šalje“ šablonu na transformaciju. Zato se u ovom primeru, budući da je cilj obrada celog XML dokumenta, koristi samo uzorak „/“ koji identificiše koren element dokumenta i naziva se koren uzorak. Kada se definišu drugi uzorci, koren uzorak se podrazumeva i ne navodi se na početku (npr. za `html/head`).

Element **<xsl:for-each>** (obeležen crvenom bojom) je verovatno najvažniji od svih elemenata XSL šablona. Koristi se za petlju koja prolazi kroz sve elemente datog XML dokumenta da bi pronašla uzorce i primenila šablove. Atribut select određuje koji elementi dokumenta su uključeni u petlju. U ovom slučaju to su svi elementi student, definisani

putanjom fakultet/student. Vrednosti indeks, prezime i ime za svaki element student formatiraju se zajedno kao niz podataka u tabeli. Redosled prolaza kroz petlju može se definisati atributom order-by.

PRIKAZ DOKUMENTA U WEB ČITAČU

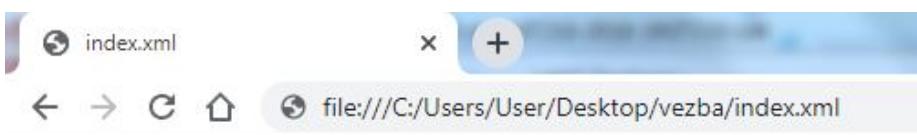
Kako se primer vidi u web čitaču

Element **<xsl:value-of>** (obeležen narandžastom bojom) koristi se za formiranje šablonu umetanjem vrednosti elementa ili atributa u transformisani dokument.

Vrednost atributa select označava element dokumenta koji se transformiše. Ovaj element omogućava korišćenje standardnih HTML tagova koji utiču na format prikazanog teksta na stranici. U ovom slučaju korišćeni su standardni početni i završni tagovi **<td>** i **<tr>** za kreiranje tabele u HTML dokumentu.

Poslednja dva elementa **</xsl:template>** i **</xsl:stylesheet>** (obeležena plavom bojom) označavaju da se radi o završetku obrasca i odgovarajuće transformacije;

Nakon snimanja i otvaranja dokumenta primer1.xml u web čitaču dobija se prikaz dat na slici 12.



Slika 7.12 Prikaz dokumenta u web citaju [Izvor: Autor]

✓ Poglavlje 8

Zadatak za samostalnu vežbu

PRVI ZADATAK

Opis prvog zadatka za samostalni rad.

(Predviđeno vreme izrade prvog zadatka iznosi 20 minuta).

1) Koristeći XSLT izvršiti transformaciju datog XML dokumenta u HTML dokument.

Struktura XML dokumenta je sledeća:

```
<?xml version="1.0" encoding="UTF-8"?>
<Životinja>
    <Pas>
        <Rasa />
        <Ime />
        <DatumOkota />
        <DatumPrimanjaVakcine />
    </Pas>
</Životinja>
```

2) Za zadatu XML datoteku, koristeći XPath jezik pronaći sve pse koji nisu primili vakcinu ili su rase mops. Struktura XML dokumenta je:

```
<?xml version="1.0" encoding="UTF-8"?>
<Životinja>
    <Pas>
        <Rasa />
        <Ime />
        <DatumOkota />
        <DatumPrimanjaVakcine />
    </Pas>
</Životinja>
```

3.) Za zadatu XML datoteku, koristeći XPath jezik pronaći sve pse koje su rase dalmatinac ili su došli na svet posle 2000 godine.

Struktura XML dokumenta je:

```
<?xml version="1.0" encoding="UTF-8"?>
<Životinja>
    <Pas>
        <Rasa />
        <Ime />
        <DatumOkota />
        <DatumPrimanjaVakcine />
```

```
</Pas>  
</Životinja>
```

4.) Koristeći XSLT izvršiti transformaciju datog XML dokumenta u HTML dokument:

```
<?xml version="1.0" encoding="UTF-8"?>  
<Parfem>  
    <ImeParfema />  
    <Proizvodjac />  
    <Cena />  
    <DatumProizvodnje />  
    <RokTrajanja />  
</Parfem>
```

DRUGI ZADATAK

Opis drugog zadatka za samostalni rad.

(Predviđeno vreme izrade drugog zadatka iznosi 20 minuta).

1) Koristeći XSLT izvršiti transformaciju datog XML dokumenta u HTML dokument.

Struktura XML dokumenta je sledeća:

```
<?xml version="1.0" encoding="UTF-8"?>  
<prodavnicaNamestaja>  
    <artikal>  
        <naziv />  
        <proizvodjac />  
        <godina />  
        <boja />  
        <kolicina />  
        <cena />  
    </artikal>  
</prodavnicaNamestaja>
```

Popuniti proizvoljnim podacima. Uneti minimum 7 redova.

2) Za zadatku XML datoteku, koristeći XPath jezik pronaći:

- sve artikle kojih nema u prodavnici, odnosno količina je 0
- artikle koji su proizvedeni između 2015 i 2021 godine
- proizvođača koji ima najviše proizvoda u prodavnici
- sve artikle koji su crvene boje ili počinju na slovo S
- najskupli artikal koji je plave boje ili je proizведен 2009. godine

✓ Poglavlje 9

DOMAĆI ZADATAK

DOMAĆI ZADATAK - DZ07

Opis domaćeg zadatka

(Predviđeno vreme izrade domaćeg zadatka je 80 minuta.)

Za XML date strukture, koristeći XPATH jezik, pronaći:

- imena i prezimena svih studenata koji su položili IT210, ili koji su stariji od 22 godine.
Podatke prikazati u HTML tabeli.
- pronaći najstarijeg studenta
- pronaći najmlađeg studenta

```
<?xml version="1.0" encoding="UTF-8"?>
<studenti>
    <student>
        <ime>Milos</ime>
        <prezime>Ljubinkovic</prezime>
        <starost>21</starost>
        <ispiti>

            <ocena>9</ocena>
            <sifra>IT210</sifra>
            </predmet>

            <ocena>10</ocena>
            <sifra>IT101</sifra>
            </predmet>
        </ispiti>
    </student>
</studenti>
```

Sve datoteke arhivirati u .ZIP fajlu. Naziv arhiviranog fajla treba da bude **IT210-DZ07-ime_prezime_brojIndeksa.zip**, gde su ime, prezime i broj indeksa vaši podaci. Arhiviran fajl poslati predmetnom asistentu na e-mail.

(napomena: u Subject-u e-mejla napisati **IT210 - DZ07**).

▼ ZAKLJUČAK

ZAKLJUČAK

Nakon što smo opisali sve tehnologije, možemo prikazati kompletну sliku distribuisane aplikacije, koje primenjuje SOAP, WSDL i UDDI.

1. Provajder veb servisa opisuje svoj servis u WSDL dokumentu i publikuje ga na UDDI. Ovo se radi preko UDDI API za publikovanje (zasnovano na SOAP-u).
2. Onaj ko traži uslugu, koristi UDDI API za pretraživanje, da bi pronašao odgovarajućeg pružaoca usluga. Ako se pronađe neki, može se potražiti WSDL dokument.
3. Na osnovu WSDL dokumenta se pravi SOAP zahtev.
4. SOAP zahtev se šalje do pružaoca usluga, posle čega se obrađuje odgovor.

Iako kreiranje svih potrebnih dokumenata možda izgleda komplikovano, treba imati na umu da se ovi dokumenti obično generišu automatski, slično stubu i skeletonu kod drugih middleware tehnologija. Programeri ne moraju da brinu o komunikaciji. Različiti alati i API-ji će automatski generisati odgovarajuće XML dokumente.

Kao što se vidi veb servisi se mogu smatrati "još jednom middleware tehnologijom", slično drugim tehnologijama, kao što su CORBA ili RMI, ali su definitivno mnogo pogodniji za pristup preko interneta.

To znači da se problemi koji postoje kod distribuisanih aplikacija, kao što su performanse, proširivost i sigurnost, javljaju i kod veb servisa. Zamislite na primer, da veb servis koristi drugi veb servis za proveru kreditne kartice. Šta će se desiti sa servisom, pa čak i sa celim poslom, ako ne postoji odgovarajuća sigurnost? Ili šta će se desiti kada se servisu uputi u istom trenutku na hiljade zahteva? Sve su to problemi koje tek treba rešavati.

Literatura

[1] Web programiranje, <https://www.webprogramiranje.org/web-servisi-osnove/> (20.12.2018).



IT210 - SISTEMI IT

TEHNIKE KODIRANJA

Lekcija 08

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 08

TEHNIKE KODIRANJA

- ✓ TEHNIKE KODIRANJA
- ✓ Poglavlje 1: Projektni šabloni
- ✓ Poglavlje 2: Vrste šabloni
- ✓ Poglavlje 3: Skript tehnike
- ✓ Poglavlje 4: Skript jezici za programiranje klijentske strane
- ✓ Poglavlje 5: Tehnike pisanja koda
- ✓ Poglavlje 6: Pokazna vežba - XAMPP instalacija
- ✓ Poglavlje 7: Pokazna vežba – Uvod u PHP
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je da se obrađe teme: integrativnog kodiranja, DesignPatterna, script tehnike, pisanje skript jezika, tehnike pisanja koda, verzije i upravljanje verzijama

U ovoj lekciji biće obrađene sledeće teme:

- Integrativno kodiranje
- Design Patterns
- Skript tehnike
- Pisanje skripta i uloga skript jezika
- Tehnike pisanja koda
- Uporedni prikaz tehnika
- Verzije i upravljanje verzijama

▼ Poglavlje 1

Projektni šabloni

MOGUĆNOST PONOVNOG KORIŠĆENJA SOFTVERSKIH MODULA

Jedna od osnovnih prednosti korišćenja objektno-orientisanog pristupa u programiranju je mogućnost ponovnog korišćenja softverskih modula

Jedna od osnovnih prednosti korišćenja objektno-orientisanog pristupa u programiranju je mogućnost ponovnog korišćenja (engl. **reusability**) softverskih modula. Klase koje su jednom napisane dobro mogu se koristiti i pri pisanju novih programske rešenja. Na taj način se znatno ubrzava proces pisanja koda. Međutim, pisanje klase koja će moći ponovo da se koriste nije tako lako kao što izgleda iz sledećih razloga:

- Treba uočiti važne objekte koji će se verovatno javljati i u drugim problemima i od njih stvoriti klase,
- Treba načiniti klase koje će imati balansiranu granularnost,
- Treba definisati interfejse i hijerarhiju nasleđivanja i definisati ključne relacije između klasa.

Da bi klasa mogla da se nanovo upotrebni potrebno joj je dati opštost. Ali, treba imati na umu da se klase pišu za program koji se trenutno razvija i da programeri, uobičajeno, u tom trenutku nemaju mnogo vremena da razmišljaju o ponovnom korišćenju istih klasa. Zbog toga, jednom napisana klasa ne postaje istog trenutka podesna za ponovnu upotrebu. Tek se nakon nekoliko njenih primena, pri kojima se ona menja dodavanjem opštosti i fleksibilnosti, može reći da je pogodna za ponovnu upotrebu.

Iskusni programeri pri rešavanju novog problema retko polaze od nule, već novo rešenje zashivaju na dobrom rešenjima iz svoje prakse. Usavršavanje ovih rešenja ih čini ekspertima. Zbog toga se u mnogim OO sistemima može uočiti ponavljanje šabloni klasa i objekata koji komuniciraju međusobno. Korišćenje ovakvih šabloni rešava specifične projektne probleme i na taj način čini OO projektovanje mnogo brže i efikasnije, pri čemu se primjenjeni šablon dodatno usavršava. Programeri koji poznaju ovakve šabline mogu odmah da ih primene bez potrebe da ih ponovo otkrivaju.

ŠTA SU PROJEKTNI ŠABLONI?

Projektni šabloni su opisi objekata i klasa koje komuniciraju i koji su prilagođeni za rešavanje opšteg projektnog problema u specifičnom kontekstu

Iskusni programeri pri rešavanju novog problema retko polaze od nule, već novo rešenje zasnivaju na dobrom rešenjima iz svoje prakse. Usavršavanje ovih rešenja ih čini ekspertima. Zbog toga se u mnogim OO sistemima može uočiti ponavljanje šablonu klasa i objekata koji komuniciraju međusobno. Korišćenje ovakvih šablonu rešava specifične projektne probleme i na taj način čini OO projektovanje mnogo brže i efikasnije, pri čemu se primjenjeni šablon dodatno usavršava. Programeri koji poznaju ovakve šabalone mogu odmah da ih primene bez potrebe da ih ponovo otkrivaju.

Projektni šabloni (engl. **design patterns**) su opisi objekata i klasa koje komuniciraju i koji su prilagođeni za rešavanje opšteg projektnog problema u specifičnom kontekstu[3].

Svaki projektni šablon sistematski imenuje, objašnjava i određuje neko važno rešenje koje se često pojavljuje u objektno-orientisanim sistemima. Cilj šablonu je da opiše projektantsko iskustvo u formi koju drugi programeri mogu efikasno iskoristiti.

Interesantno je da je ideja o projektantskim šablonima došla iz arhitekture. Arhitekta Christopher Alexander,, koji se bavio arhitekturom gradova, je u knjizi *A Pattern Language* [4] napisao: „Svaki šablon opisuje problem koji se stalno ponavlja u našem okruženju i onda opisuje suštinu rešenja tog problema na takav način da ga možete primeniti milion puta, a da ništa ne učinite na isti način dva puta“.

Napomena:

Ovde je za engleski izraz „ pattern“ upotrebljen izraz **šablon** koji po autorovom mišljenju najbolje odsljika značaj reči pattern. Alternativni prevodi mogu biti obrazac, model ili uzorak (uzor).

ELEMENTI ŠABLONA

U opštem slučaju šabloni imaju četiri elemenata: ime šablonu, opis problema, rešenje i posledice.

U opštem slučaju šabloni imaju četiri elementa:

1. **Ime šablonu** se sastoji od jedne do dve reči koje daju opis projektnog problema, njegovog rešenja i konsekvenci. Davanje imena šablonu pomaže da se stvari rečnik prepoznatljiv od strane programerske zajednice. Sa ovakvim rečnikom lakše je razumevanje programera u međusobnim komunikacijama i razumljivija je dokumentacija. Međutim, nalaženje pravog imena nekog šablonu je najteži deo razvoja kataloga šablonu.
2. **Opis problema** na koji može da se primeni neki šablon je drugi element. On opisuje problem i kontekst. Opis može da opiše specifični projektni problem i listu uslova koji treba da budu zadovoljeni da bi šablon mogao da se primeni.
3. **Rešenje** opisuje elemente koji čine projekat, njihove odnose, odgovornost i saradnju između njih. Rešenje ne treba da opisuje neki specijalni slučaj problema ili implementaciju zbog toga što šablon treba da se koristi u mnogim različitim

situacijama. Šablon treba da obezbedi apstraktni opis projektnog problema i kako se generalnom primenom klasa i objekata problem može rešiti.

4. **Posledice** pokazuju rezultate i šta se gubi primenom šablona. Opis posledica je dragocen kada se odlučuje o projektnim alternativama i procenjuju dobici i gubici od primene nekog šablona. Konsekvene primene nekog šablona uobičajeno su gubici u memoriji i vremenu izvršenja. Međutim, konsekvene mogu biti i ograničenja vezana za primenu nekog jezika ili implementaciju. Konsekvene takođe mogu imati uticaj na fleksibilnost, proširivost i portabilnost sistema.

Projektni šabloni imenuju, apstrahuju i identifikuju ključne aspekte opštih projektnih struktura što ih čini korisnim za ponovnu upotrebu u objektno orijentisanim sistemima. Šabloni identifikuju klase i instance, njihovu ulogu i kolaboraciju i distribuciju odgovornosti.

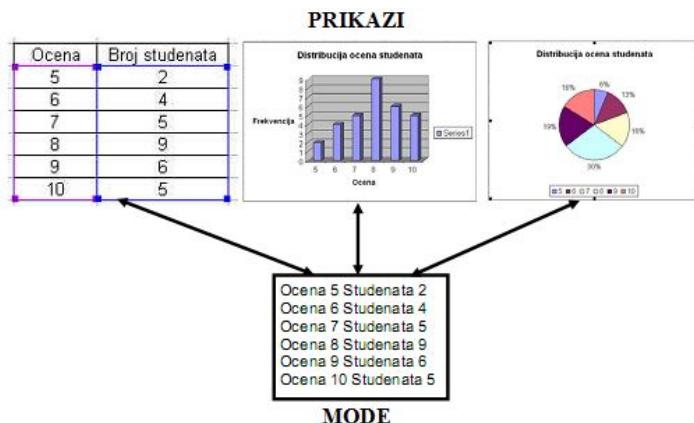
▼ 1.1 MVC

PRIKAZ MVC MODELA

MVC koristi tri klase koje su razdvojene zadacima, čime aplikacija postaje fleksibilnija – Model, View, Controller

U jeziku Smalltalk-80 se za gradnju interfejsa koristi trojka klasa **Model-View-Controller** (MVC). Razmatranjem projektnih šablona unutar MVC prikazaće se ideja šablona.

MVC se sastoji od tri vrste objekata. **Model** je aplikacioni objekt, **View** je prikaz na ekranu, a **Controller** definiše način na koji korisnički interfejs reaguje na ulaz od strane korisnika. Korišćenjem ove tri klase razdvojeni su zadaci čime je aplikacija postala fleksibilnija, a pojedine klase se mogu iskoristiti i u drugim aplikacijama. U suprotnom, korisnički interfejs bi bio jedan veliki program koji bi bio težak za održavanje, a njegove komponente ne bi mogle da se primenjuju u drugim aplikacijama. **MVC razdvaja prikaz i model uspostavljajući između njih protokol tipa preplata-obaveštenje** (engl. *subscribe-notify*). Prikaz mora da osigura da će ono što se pojavljuje na ekranu odgovarati stanju modela. **Kad god se promene podaci modela, model obaveštava prikaze koji zavise od njega**, zahvaljujući tome, svaki pogled ima mogućnost da ažurira sebe. Ovakav pristup omogućava da se jednom modelu pridruži više prikaza kako bi se obezbedile različite prezentacije. **Pored toga, moguće je kreirati novi prikaz, bez potrebe da se model menja.**



Slika 1.1.1 Prikaz modela i tri prikaza [Izvor: Autor]

Slika 1 prikazuje model i tri prikaza. Model sadrži neke vrednosti, a prikazi predstavljaju podatke na tri različita načina: tabela, histogram i prikaz u obliku pite. Model komunicira sa prikazima uvek kada se neka vrednost modela promeni, a prikazi komuniciraju sa modelom da bi pristupili vrednostima koje prikazuju.

Primer sa slike 1 koji pokazuje odvajanje prikaza od modela se može generalizovati. Generalizovani problem bi mogao da glasi: razdvojiti objekte tako da promena jednog može da utiče na proizvoljan broj drugih bez potrebe da izmenjeni objekt zna detalje drugih. Još generalniji problem je opisan projektnim šablonom **Observer**.

✓ 1.2 Opis projektnih šablona

KONZISTENTAN FORMAT PROJEKTNIH ŠABLONA

Konzistentni format čini projektni šablon i sastoji se iz: imena i klasifikacija, namene, motivacije, učesnika, kolaboracije, posledica, primenjivosti, strukture, primera i sl.

Projektni šabloni se opisuju korišćenjem konzistentnog formata koji se sastoji od sledećih delova:

- **Ime šablona i klasifikacija.** Ime šablona treba da dobro opisuje njegovu suštinu. Dobro ime zna da zaživi i postane deo svakodnevne komunikacije. Takođe se navodi kojoj klasi pripada šablon. O mogućim klasama će biti reči kasnije.
- **Namena.** U ovom delu opisa treba odgovoriti na sledeća pitanja: Šta radi šablon? Zašto je napravljen i šta je njegova namena? Koje posebne projektne probleme rešava?
- **Poznat i kao.** Ovde se navode i druga imena koje ima šablon (ako ih ima).
- **Motivacija.** Scenario koji ilustruje projektni problem i kako klase i objekti šablona rešavaju problem. Scenario treba da pomogne u razumevanju narednog dela Primenljivost, koje je mnogo apstraktnije.

- **Primenljivost.** Koje su situacije u kojima se šablon može primeniti? Koji su primeri projekata na koje se šablon odnosi? Kako se mogu prepoznati takve situacije?
- **Struktura.** Struktura se opisuje grafičkom prezentacijom upotreboom tehnike objektnog modeliranja (**OMT - Object Modeling Technique**). Takođe se koriste i interakcioni dijagrami da bi se ilustrovao redosled zahteva i kolaboracije između objekata.
- **Učesnici.** U ovom delu se opisuju klase i/ili objekti koji učestvuju u šablonu i njihova odgovornost.
- **Kolaboracija.** Opisuje kako učesnici sarađuju u cilju izvršenja zadataka za koje su odgovorni.
- **Posledice.** U ovom poglavlju se navode moguće posledice upotrebe datog šablonu.
- **Implementacija.** Ovde se navode moguće zamke prilikom implementacije šablonu, saveti i tehnike za uspešnu implementaciju. Ukoliko ima razlika u implementaciji za različite jezike, to se navodi u ovom delu opisa šablonu.
- **Primer koda.** Delovi koda koji pokazuju kako se šablon može implementirati.
- **Poznati slučajevi upotrebe.** U ovom delu se navode primjeri upotrebe šablonu u realnim sistemima. Daju se najmanje dva primera iz različitih domena.
- **Srodnii šabloni.** Ovde se navode šabloni koji su slični datom. Navode se razlike između ovih šabloni. Ukoliko se dati šablon koristi u kombinaciji sa nekim drugim, navodi se njegovo ime.

▼ Poglavlje 2

Vrste šablonu

NAJČEŠĆE KORIŠĆENI ŠABLONI

Prikazani su najčešće korišćeni šabloni u C++ jeziku

Niže su dati neki najčešće korišćeni šabloni u jeziku C++ [3]. I drugi jezici, kao što je Java, koriste iste ili vrlo slične šablove. U prvoj koloni je dato njihovo originalno ime i mogući prevod. U drugoj koloni je dat samo kratak opis namene šablonu.

Abstract Factory ([Apstraktna fabrika](#)) - Obezbeđuje interfejs za kreiranje familije povezanih i zavisnih objekata bez specificiranja njihovih konkretnih klasa

Adapter - Pretvara interfejs klase u interfejs koji očekuju klijenti. Adapter dozvoljava da klase koje ne bi mogle da rade zbog nekompatibilnih interfejsa mogu da rade zajedno.

Bridge ([Most](#)) - Odvaja apstrakciju od njene implementacije tako da se obe mogu menjati nezavisno

Builder (Graditelj) - Odvaja konstrukciju kompleksnih objekata od njihove reprezentacije -tako da ista konstrukcija procesa može da ima različite reprezentacije.

Chain of Responsibility ([Lanac odgovornosti](#)) - Izbegava spajanja pošiljaoca zahteva od njegovog prijemnika tako što daje šansu da više od jednog objekta mogu da rukuju zahtevom. Povezuje prijemne objekte u lanac i prosleđuje zahtev kroz lanac sve dok neki objekat rukuje sa njim.

Command - Učauruje zahtev kao objekt i na taj način dozvoljava korisniku da parametrizuje klijente sa različitim zahtevima, redovima ili log zahtevima i podrži operacije koje se mogu opozvati (operacije na koje je moguće primeniti operaciju *undo*).

Composite - Komponuje objekte u strukture u obliku stabla da bi predstavio hijerarhiju deo-celina. Kompoziti omogućuju klijentima da individualne objekte i kompozicije objekata tretiraju jedinstveno.

Decorator ([Dekorater](#)) - Dodaje dodatne odgovornosti nekom objektu dinamički. Dekorator obezbeđuje fleksibilnu alternativu metodi stvaranja potklasa u cilju proširenja funkcionalnosti.

Facade ([Fasada](#)) - Obezbeđuje unificirani interfejs koji treba da zameni set interfejsa u podsistemu. Fasada definiše interfejs visokog nivoa koji olakšava upotrebu podsistema.

Factory Method ([Metod fabrika](#)) - Definiše interfejs za kreiranje nekog objekta, ali dozvoljava podklasama da odluče koju će klasu da instanciraju. Factory metod dozvoljava klasama da odgodeinstanciranje potklasa.

Flyweight - Koristi deljenje da efikasno podrži veliki broj fino granuliranih objekata.

Interpreter - Za dati jezik definiše reprezentaciju njegove gramatike zajedno sa prevodiocem koji koristi reprezentaciju da prevede rečenice u jeziku.

Iterator - Obezbeđuje način da se pristupi elementima agregiranog objekta sekvensijalno bez potrebe za izlaganjem njihove reprezentacije

Mediator (Posrednik) - Definiše objekt koji učauruje metode na koji set objekata vrši

interakciju. Mediator potpomaže labave veze omogućujući da objekti ne referenciraju jedan na drugog eksplisitno, što omogućuje da se njihove interakcije menjaju nezavisno.

✓ 2.1 Katalog šablon

KLASIFIKACIJA ŠABLONA

Projektni šabloni se razlikuju po granularnosti i stepenu apstrakcije.

Klasifikacija je izvedena po dva kriterijuma: namena i domen

Projektni šabloni se razlikuju po granularnosti i stepenu apstrakcije. U cilju lakošeg nalaženja pravog šablonu daje se klasifikacija koja je izvedena po dva kriterijuma: **namena** i **domen** [3]. Namena šablonu opisuje šta šablon radi. Namena šablonu može biti: kreativna, strukturalna ili ponašajna. **Kreativni šabloni** kreiraju objekte. **Strukturalni šabloni** se bave sastavljanjem klasa i objekata. **Ponašajni šabloni** definišu način na koji klase i objekti vrše interakciju i distribuiraju odgovornost.

Drugi kriterijum, nazvan **domen**, specificira da li se šablon primjenjuje prevashodno na klase ili objekte. **Šabloni klase** se bave vezama između klasa i njihovih potklasa. Ove veze se uspostavljaju putem nasleđivanja i one su statičke (fiksne) u trenutku prevođenja koda. **Šabloni objekta** se bave vezama između objekata, koje mogu biti menjane tokom izvršenja programa, pa su više dinamičke.

		Namena		
		Kreativna	Strukturalna	Ponašajna
Domen	Klase	Factory Method	Adapter	Interpreter Template Method
	Objekti	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer State Strategy Visitor

Slika 2.1.1 Klasifikacija izvedena po nameni i domenu šablonu [Izvor: Autor]

Kreativni šabloni klasa prenose neke delove procesa kreiranja objekata na potklase, dok kreativni šabloni objekata prenose delove procesa kreiranja objekata na druge objekte. **Strukturalni šabloni klase koriste nasleđivanje da komponuju klase, dok strukturalni šabloni objekta opisuju načine za kreiranje objekata.** **Ponašajni šabloni** klasa koriste nasleđivanje da opišu algoritme i tok upravljanja, dok ponašajni šabloni objekata opisuju kako grupa objekata sarađuje da izvrši zadatak koji jedan samostalni objekat ne može da izvrši.

2.2 Singleton

OPIS SINGLETON-A

SINGLETON osigurava da klasa ima samo jednu instancu i obezbeđuje globalnu tačku za pristup njoj. Koristi se kada treba da postoji tačno jedna instanca klase koja može biti dostupna

Namena

Osigurava da klasa ima samo jednu instancu i obezbeđuje globalnu tačku.

Motivacija

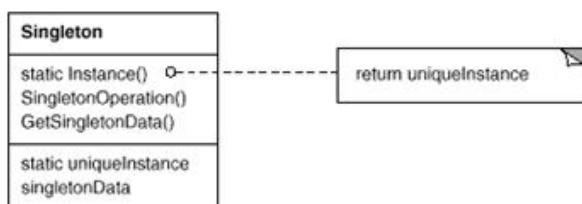
Za neke klase je važno da imaju samo jednu instancu. Na primer, iako ima više štampača u sistemu, treba da postoji samo jedan red za štampu, treba da postoji samo jedan sistem datoteka i jedan menadžer prozora itd. Postavlja se pitanje, kako osigurati da klasa ima samo jednu instancu i da pristup njoj bude jednostavan? Globalna promenljiva čini jedan objekat pristupačnim, ali ga ne štiti od instanciranja više objekata. Bolje rešenje je učiniti klasu odgovornom da vodi računa o svojoj jedinoj instanci. Klasa može da obezbedi da se druga instanca ne može kreirati presretanjem zahteva za kreiranjem novih objekata i da obezbedi način za pristup instanci. Takva klasa se naziva Singleton.

Primena

Singleton šablon se koristi kada:

- Treba da postoji tačno samo jedna instanca klase i ona mora da bude dostupna klijentima sa dobro poznate pristupne tačke,
- Kada jedina instanca treba da bude proširiva dodavanjem potklasa, a klijentima treba omogućiti da koriste i prošire instancu bez promene njenog koda.

Struktura



Slika 2.2.1 Prikaz singletona [Izvor: https://miro.medium.com/max/1028/1*WXXQZp1glrQxLqrQ_TDN7Q.png]

Učesnici

Singleton definiše operaciju instance koja omogućuje klijentima pristup njegovoj jedinstvenoj instanci. Instanca je class operacija (funkcija **static member** u C++) može da bude odgovoran za kreiranje svoje sopstvene jedinstvene instance.

Kolaboracija

Klijent pristupa Singleton instanci isključivo preko Singeltonove operacije instance.

OPIS SINGLETON - KONSEKVENCE

Singleton šablon ima nekoliko prednosti: Upravljeni pristup jedinoj instanci, redukuje prostor imena, dozvoljava pofinjavanje operacija i reprezentacije, dozvoljava promenljiv broj instanci

Konsekvence

Singleton šablon ima nekoliko prednosti:

1. **Upravljeni pristup jedinoj instanci.** Zbog toga što Singleton klasa učauruje svoju jedinu instancu, ona ima striktnu kontrolu nad tim kako i kada klijenti mogu da joj pristupe.
2. **Redukuje prostor imena.** Singleton šablon predstavlja poboljšanje u odnosu na globalne promenljive. Pomoću njega se izbegava zagađivanje prostora imena globalnim promenljivima koje čuvaju svoje jedine instance.
3. **Dozvoljava pofinjavanje operacija i reprezentacije.** Iz Singleton klase se može izvesti potklasa i lako je konfigurisati aplikaciju sa instancom ove proširene (odnosno nasleđene) klase. Moguće je konfigurisati aplikaciju sa instancom potrebne klase tokom izvršenja programa.
4. **Dozvoljava promenljiv broj instanci.** Šablon omogućuje da se programer predomisli i dozvoli postojanje više od jedne instance Singleton klase. Pored toga, isti pristup se može iskoristiti za upravljanjem broja instanci koje aplikacija koristi. Jedino operacije koje dozvoljavaju pristup Singleton instanci treba da budu promenjene.

▼ 2.3 Factory Method

OPIS FACTORY METHOD – NAMENA, POZNAT KAO, MOTIVACIJA

Definiše interfejs za kreiranje objekta ali omogućuje da potklase odluče koju će klasu instancirati. Factory Method šablon dozvoljava da klase prepuste instanciranje potklasama.

Namena

Definiše interfejs za kreiranje objekta ali omogućuje da potklase odluče koju će klasu instancirati. **Factory Method** šablon dozvoljava da klase prepuste instanciranje potklasama.

Takođe poznat kao

virtuelni konstruktor

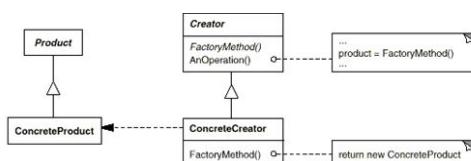
Motivacija

Framework sistemi koriste apstraktne klase da definišu i održavaju veze između objekata. Framework je obično odgovoran i za kreiranje ovih objekata.

Posmatrajmo, na primer, framework aplikacije koja može da istovremeno predstavi više dokumenta korisniku. Dve ključne apstraktne klase ovog framework-a su klase **Application** i **Document**. Obe klase su apstraktne pa klijent mora da iz njih izvede potklase da bi dobio njihove implementacije koje su zavisne od aplikacije. Da bi se, na primer, kreirala aplikacija za crtanje, definisće se klase **DrawingApplication** i **DrawingDocument**. Klasa **Application** je odgovorna za upravljanje dokumentima i treba da kreira nove dokumente uvek kada korisnik iz menija izabere **Open** ili **New**.

Zbog toga što je za instanciranje dokumenata potrebno imati potklase koje zavise od aplikacije, klasa **Application** ne može da predviđa koju potklasu **Document**-a da instancira. Klasa **Application** jedino zna kada treba da bude kreiran novi dokument, a ne zna koju vrstu dokumenta treba da kreira. Dakle, problem je u sledećem: framework mora da instancira klasu, ali zna samo za apstraktne klase koje ne može da instancira.

Šablon Factory Method nudi rešenje ovog problema. On učaruje znanje o tome koju **Document** potklasu da kreira i pomera ovo znanje van framework-a.



Slika 2.3.1 Prikaz šablonu Factory method [Izvor: <https://www.startertutorials.com/patterns/wp-content/uploads/2013/11/9-factory-method-structure.png>]

Application potklase redefinišu apstraktnu operaciju **CreateDocument** iz klase **Application** tako da vraća odgovarajuću **Document** potklasu. Kada se **Application** potklasa instancira, ona onda može da instancira dokumente za određenu aplikaciju iako ne zna njihovu klasu. Operacija **CreateDocument** se naziva factory method jer je odgovorna za „proizvodnju“ objekta.

FACTORY METHOD – PRIMENA, UČESNICI, STRUKTURA, KOLABORACIJA

Šablon Factory Method treba koristiti kada: klasa ne može da predviđa objekte klase koje treba da kreira, želi da njene potklase specificiraju objekte koje kreiraju i kad se delegira odgovornost

Primena

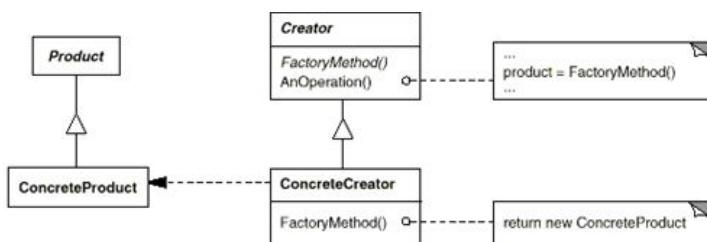
Šablon **Factory Method** treba koristiti kada:

- klasa ne može da predviđa objekte klase koje treba da kreira,
- klasa želi da njene potklase specificiraju objekte koje kreiraju,
- klasa delegira odgovornost jednoj od nekoliko helper potklasa, a programer želi da sazna koja helper klasa je delegat.

Učesnici

- **Product (*Document*)** - definiše interfejsa objekta koga kreira factory method-a
- **ConcreteProduct (*MyDocument*)** - implementira Product interfejs.
- **Creator (*Application*)** - deklariše factory method, koji vraća objekat tipa Product. Creator može takođe da definiše podrazumevajuću implementaciju factory method-e koja vraća podrazumevajući ConcreteProduct objekt. može da pozove factory method da kreira Product objekt.
- **ConcreteCreator (*MyApplication*)** - menja (engl. *overrides*) factory method da bi vratio instancu ConcreteProduct.

Struktura



Slika 2.3.2 Prikaz factory method strukture [Izvor: <https://www.startertutorials.com/patterns/wp-content/uploads/2013/11/9-factory-method-structure.png>]

Kolaboracija

Creator se oslanja na njegove potklase da bi definisao factory method, tako da on vraća instancu odgovarajućeg **ConcretProduct**.

Konsekvence

Šablon **Factory Method** eliminiše potrebu da se unutar koda ugrađuju klase koje zavise od aplikacije. Umesto toga, programski kod radi samo sa **Product** interfejsom, pa zbog toga može da radi sa bilo kojim korisnički definisanim **ConcreteProduct** klasama.

✓ 2.4 Facade

PRIKAZ FACADE ŠABLONA

Facade obezbeđuje jedinstveni interfejs ka grupi interfejsa u podsistemu. Podela sistema na podsisteme pomaže u smanjenju kompleksnosti problema.

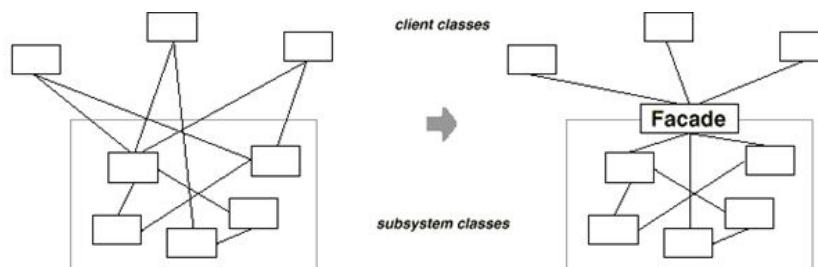
Namena

Obezbeđuje jedinstveni interfejs ka grupi interfejsa u podsistemu. Šablon facade definiše interfejs visokog nivoa koji olakšava korišćenje podistema.

Motivacija

Podela sistema na podsisteme pomaže u smanjenju kompleksnosti problema. Čest projektni zadatak je minimizirati komunikaciju i zavisnost između podistema. Jedan od načina da se postigne taj cilj je da se iskoristi objekt facade koji obezbeđuje jedan jedinstven interfejs ka opštijim elementima podistema.

Razmotrimo primer programskog okruženja koje aplikacijama daje pristup do njegovog kompjuterskog podistema. Ovaj podistem sadrži klase kao što su **Scanner**, **Parser**, **ProgramNode**, **BytecodeStream** i **ProgramNodeBuilder** koje čine kompjajler. Neke specijalizovane aplikacije mogu da imaju potrebu da pristupaju ovim klasama direktno. Ali, većina klijenata kompjajlera nije zainteresovana za detalje kao što su parsovanje ili generacija koda. Oni jedino žele da kompjajliraju neki kod. Njima tako snažan interfejs niskog nivoa samo komplikuje zadatke.

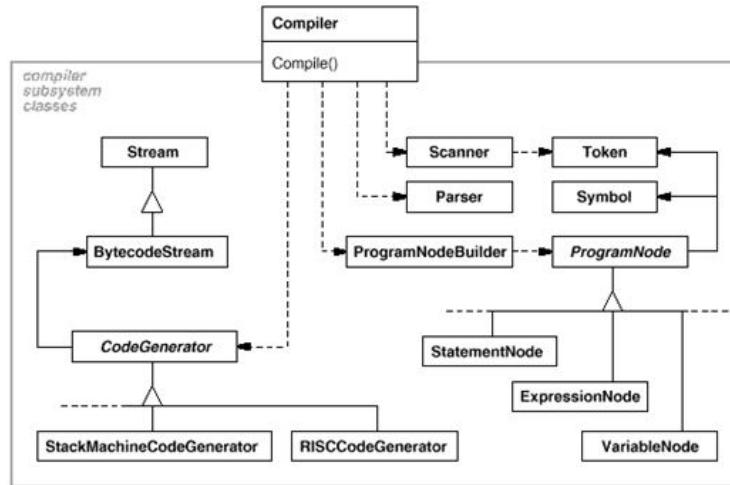


Slika 2.4.1 Prikaz Facade šablonu [Izvor: <https://ziabaig.files.wordpress.com/2017/10/facade.png?w=681&h=215>]

Da bi se obezbedio interfejs visokog nivoa koji štiti klijente od svih ovih klasa, kompjajler podistem ima klasu **Compiler**. Ova klasa definiše jedinstveni interfejs ka funkcionalnosti kompjajlera. Klasa **Compiler** deluje kao fasada. Ona nudi klijentima jedan, jednostavan interfejs ka podistem kompjajlera. Ona spaja zajedno sve klase koje čine funkcionalnost kompjajlera ne sakrivajući ih kompletno. Za one kojima je to potrebno funkcionalnost nižeg nivoa nije skrivena.

PRIMENA FACADE

Facade šablon treba koristiti kada: Treba obezbediti jednostavan interfejs ka kompleksnom podsistemu

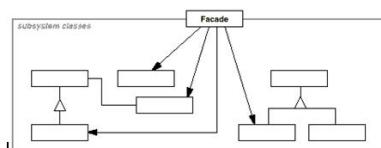


Slika 2.4.2 Prikaz rada facade [Izvor: http://4.bp.blogspot.com/_k4180KU275M/TOUkG_YchfI/AAAAAAAAs/M_T3GSEFLSA/s1600/Facade4.jpg]

Primena

Facade šablon treba koristiti kada: Treba obezbediti jednostavan interfejs ka kompleksnom podsistemu. Postoje mnoge zavisnosti između klijenata i implementacija klasa neke abstrakcije. Tada treba upotrebiti fasadu da bi se razdvojio podistem od klijenata i drugih podistema, što podistem čini nezavisnim i portabilnim. Treba podisteme izgraditi iz više slojeva.

Struktura



Slika 2.4.3 Primena facade šablon - podistemi višesloja [Izvor: Autor]

Učesnici

- **Facade (*Compiler*)** - zna koje klase podistema su odgovorne za primljeni zahtev i delegira zahteve klijenta odgovarajućim objektima podistema.
- **subsystem classes (*Scanner*, *Parser*, *ProgramNode*, etc.)** - implementira funkcionalnost podistema, odradjuje zadatke dobijene od Facade objekta i nema znanja o fasadi, tj. nema referencu na fasadu.

Kolaboracija

- 1.** Klijent komunicira sa podsistomom slanjem zahteva fasadi, koja ga prosleđuje odgovarajućim objektima podsistema.
- 2.** Klijenti koji koriste fasadu ne moraju da pristupaju objektima podsistema direktno.

▼ Poglavlje 3

Skript tehnike

SKRIPTING JEZICI

Skript jezici su projektovani za povezivanje postojećih programskih komponenata. Skript jezici su specifični jezici visokog nivoa koji se interpretiraju i koji se odlikuju jednostavnosću i efikasnošću

Skript jezici su specifični jezici visokog nivoa koji se interpretiraju i koji se odlikuju jednostavnosću i efikasnošću. Mogu biti opšte namene ili ograničeni za neku specifičnu oblast kao što je:

- pisanje skripta za klijentsku stranu veb aplikacije,
- pisanje skripta za serversku stranu veb aplikacije,
- pisanje skripta za neki operativni sistem,
- pisanje skripta za neku klasu aplikacija, kao što je na primer Microsoft Visual Basic for Applications koji se koristi za automatizovanje Office aplikacija.

U tipične skript jezike spadaju **Perl**, **Tcl/Tk**, **Rexx**, **Python**, **VBA** i drugi. Razlika između skript jezika i klasičnih programskih jezike kao što su C++ ili Java je u njihovoј nameni. Klasični programski jezici su projektovani za građenje strukture podataka i algoritama koji rade sa njima od nule. Za razliku od njih, skript jezici su projektovani za povezivanje postojećih programskih komponenata. Oni prepostavljaju postojanje seta programskih komponenata i njihova svrha je da postojeće komponente spoje zajedno. Programske jezice imaju strogo definisane tipove podataka kako bi moglo da se upravlja kompleksnošću podataka. Skript jezici uobičajeno nemaju tipove podataka kako bi se pojednostavilo spajanje komponenata i omogućio brzi razvoj aplikacija.

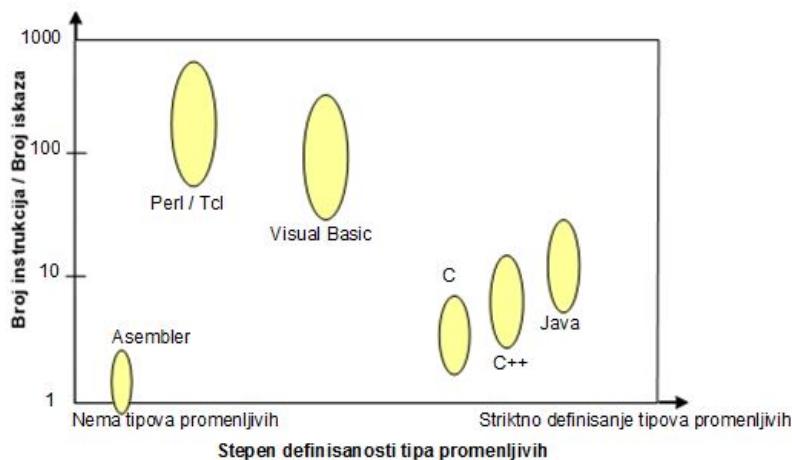
Skript jezici i klasični programski jezici su komplementarni i većina današnjih programskih platformi obezbeđuje i jednu i drugu vrstu jezika. Trend razvoja računarskih tehnologija poslednjih decenija koji se odlikuje grafičkim korisničkim interfejsima, komponentnom arhitekturom, razvoju Interneta i veb aplikacija sa jedne i sve bolji skript jezici sa druge strane učinili su da primena skript jezika bude sve šira. Predviđa se da će se trend povećanog korišćenja skript jezika nastaviti i u budućnosti. Klasični programski jezici će se i dalje koristiti ali prevashodno za pisanje komponenata, dok će se skript jezici koristiti za povezivanje komponenata u aplikacije.

Tokom svog razvoja klasični programski jezici su prešli dug put od početno korišćenog mašinskog koda. Kod asemblerских jezika gotovo svakoj programskoj instrukciji je odgovarala jedna programska instrukcija. Programske konstrukcije asemblera barataju hardverom na najnižem nivou definišući pristup i alociranje registara. Jezici visokog nivoa koji su se pojavili pedesetih godina dvadesetog veka su bili napredak jer je jedna programska instrukcija

prevodena u nekoliko mašinskih instrukcija (na primer za jezik C statistički prosek je oko 5 mašinskih za jedan C iskaz). Klasični programski jezici nisu tako efikasni kao asembler, ali su sa aspekta programiranja mnogo produktivniji.

NAMENA SKRIPT JEZIKA

Skript jezici podrazumevaju postojanje programskih komponenti pisanih u drugim programskim jezicima. Zbog toga što im je namena da povežu ove postojeće komponente



Slika 3.1.1 Prostorni odnos skripting jezika [Izvor: Autor]

Na slici 1 je prikazana pozicija pojedinih programskih i skript jezika u prostoru koji je definisan osama: Stepen definisanosti tipa promenljivih i Broja mašinskih instrukcija za jedan programski iskaz. Kao što se vidi Asembler je najneproduktivniji jezik. S druge strane skript jezici spadaju u najproduktivnije.

Skript jezici nisu namenjeni za pisanje aplikacije od početka. Oni podrazumevaju postojanje programskih komponenti pisanih u drugim programskim jezicima. Zbog toga što im je namena da povežu ove postojeće komponente, skript jezici se često nazivaju jezici za lepljenje ili jezici za integraciju sistema.

Na primer, u Unix shell-u svi filter programi mogu da čitaju niz bajtova sa nekog ulaza, da ga obrađuju i da zapišu izlazni niz na nekom od izlaza. Sledeći Shell script se sastoji od 3 Unix komande koje su međusobno povezane tako što je izlaz iz jedne komande ulaz u drugu. Zadatak skripta je da u izabranom domenu prebroji linije koje sadrže string fakultet.

select | grep fakultet | wc

Naredba **select** čita tekst koji je trenutno izabran na displeju i šalje ga na izlaz komande. Naredba **grep** čita ovaj izlaz i šalje na svoj izlaz samo linije koje sadrže string fakultet.. Naredba **wc** broji linije na svom ulazu i prikazuje rezultat na svom izlazu. Svaki od ovih programa se u drugim skriptovima može upotrebiti za različite zadatke. Osnovna osobina

klasičnih programskega jezika je rigorozno poštovane pravila da svaka promenljiva mora da ima deklarisan tip.

PREDNOSTI SKRIPT JEZIKA

Skript jezici nemaju striktnu definiciju tipova promenljivih

Da bi objekti mogli međusobno da komuniciraju kreiraju se interfejsi. Svaki interfejs može da komunicira samo sa objektima čije su promenljive definisane tačno određenim tipovima. **Ako se želi povezivanje postojećih objekata sa interfejsom potrebno je promeniti tipove njihovih promenljivih i ponovo kompajlirati deo ili celu aplikaciju.** U nekim slučajevima, kada korisnik ima samo binarni kod to je nemoguće. Skript jezici nemaju striktnu definiciju tipova promenljivih čime se ovaj problem prevaziđa. Da bi smo ilustrovali ovu prednost, razmotrimo naredni primer napisan u skript jeziku Tcl:

```
button .b -text Hello! -font {Times 16} -command {puts zdravo}
```

Ova komanda kreira novu kontrolu dugme u kojoj se prikazuje tekstualni string u fontu **Times** i sa veličinom od **16** tačaka. Klikom na dugme **Hello** ispisaće se poruka: zdravo. U ovoj liniji skripta upotrebljeno je čak 6 različitih elemenata: kontrola za dugme (**.b**), imena osobina kontrole (**-text, -font, -command**), jednostavani stringovi (**Hello!** i **zdravo**), podatak za ime i veličinu fonta (**Times 16**) i Tcl skript (**puts zdravo**). Tcl reprezentuje sve ove elemente jedinstveno kao stringove.

Osobine kontrole mogu biti navedene bilo kojim redom, a mogu biti i nespecificirane. U ovom primeru čak 20 osobina kontrole button nije zadato.

Isti primer zahteva 7 linija koda u dve metode kada se implementira u Javi. Sa C++ i Microsoft Foundation Classes potrebno je 25 linija koda u tri procedure. Samo setovanje fonta zahteva nekoliko sledećih linija.

```
CFont *fontPtr = new CFont();
fontPtr->CreateFont(16, 0, 0, 0, 700, 0, 0, 0, ANSI_CHARSET,
OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS, DEFAULT_QUALITY,
DEFAULT_PITCH|FF_DONTCARE, "Times New Roman");
buttonPtr->SetFont(fontPtr);
```

Veći deo koda u prethodnom primeru je posledica rigoroznog poštovanja tipova promenljivih. Da bi se setovao font kontrole dugme potrebno je pozvati njegovu metodu **SetFont**, ali ovoj metodi se mora proslediti pointer ka **CFont** objektu. Da bi se to uradilo potrebno je deklarisati i inicijalizovati novi objekat. Da bi se inicijalizovao **CFont** objekt treba pozvati njegov **CreateFont** metod. Metod **CreateFont** zahteva čak 14 različitih argumenata koji moraju biti specificirani.

PREDNOSTI SKRIPT JEZIKA I PREPORUKE KADA IH KORISTITI

Skript aplikacije mogu razviti 5 do 10 puta brže. Skript jezici se interpretiraju, pa je njihovo izvršenje mnogo sporije nego izvršavanje kompajliranih programa.

Očigledna je neefikasnost klasičnih programskih jezika u odnosu na skript jezike. Skript jezici vrše proveru tipa promenljivih u poslednjem mogućem momentu, odnosno u trenutku izvršenja skripta. Kod klasičnih programskih jezika provera promenljivih se vrši prilikom kompajliranja. Programer će biti siguran da njegov program neće krahirati prilikom izvršenja. Cena koja se za to plaća je ograničenje kako se podaci mogu koristiti. Ovo dovodi do više programskog koda i manje fleksibilnosti programa.

Praksa je pokazala da se korišćenjem skripta aplikacije mogu razviti 5 do 10 puta brže. S druge strane, skript jezici se interpretiraju, pa je njihovo izvršenje mnogo sporije nego izvršavanje kompajliranih programa. Programi pisani klasičnim jezicima se izvršavaju 10 do 20 puta brže.

Iz prethodnih razmatranja mogu se izvesti preporuke koje pokazuju kada treba koristiti skript jezike, a kada klasične programske jezike. Skript jezike treba koristiti kada:

- je zadatak aplikacije da poveže postojeće programske komponente,
- aplikacija treba da manipuliše mnoštvom različitih stvari,
- aplikacija ima obilje manipulacija sa stringovima,
- se aplikacija često menja,
- se aplikacija često proširuje.

S druge strane, klasične programske jezike treba koristiti uvek kada:

- aplikacija izvršava kompleksne algoritme koji zahtevaju veliko procesorsko vreme,
- aplikacija ima složenu strukturu podataka,
- aplikacija manipuliše sa velikim setom podataka, a vreme izvršenja je kritično,
- se aplikacija ne menja često

✓ 3.1 Vrste skript jezika

PODELA SKRIPT JEZIKA PREMA NAMENI

Većina skript jezika je razvijena za specifične namene, mada su neki evoluirali u opšte jezike

U praksi se koristi veliki broj skript jezika. Većina njih je razvijena za specifične namene, mada su neki od njih evoluirali u opšte jezike. Radi sagledavanja svih oblasti u kojima se koriste

skript jezici, ovde će se dati podela skript jezika prema njihovoј osnovnoј nameni. Neki od jezika se mogu koristiti u više oblasti. Dakle, skript jezici se prema svojoj nameni dele na:

- Skript jezici operativnih sistema
- Makro jezici
- Skript jezici aplikacija
- Dinamički skript jezici opšte namene
- Skript jezici za programiranje serverske strane
- Skript jezici za programiranje klijentske strane.

SKRIPT JEZICI OPERATIVNIH SISTEMA

*Prvi skript jezici koji su se pojavili su bili skript jezici namenjeni automatizaciji čestih zadataka administratora operativnih sistema.
Skriptovi se mogu pisati direktno u komandnoj liniji i izvršav*

Prvi skript jezici koji su se pojavili su bili skript jezici namenjeni automatizaciji čestih zadataka administratora operativnih sistema. Ovi jezici su nazivani **batchlanguages** ili **job control languages**. Sa pojavom Unix-a stvorena je posebna vrsta skriva specijalno namenjena Unix familiji, koja je nazvana shell.

Jezici za upravljanje poslovima (**Job Control Languages**) i ljudske (**shells**) su namenjeni za upravljanjem procesima koji se izvršavaju u nekom računarskom sistemu. Skripte se mogu pisati direktno u komandnoj liniji i izvršavati odmah. Ako je potrebno napraviti veći skript kreira se datoteka koja se kasnije može pozvati na izvršenje.

Sledeći primer prikazuje jedan shell skript za kopiranje svih datoteka tipa txt i mp3 tekućeg direktorijuma u direktorijume koju su definisani parametrima **\$1** i **\$2** respektivno. Skript napisan u nekom tekstu editoru se zapisuje kao datoteka.

```
cp *.txt      $1
cp *.mp3 $2
```

Pozivanje ovog skripta bi moglo da ima ovakav oblik:

kopiranje tekstovi filmovi

Iza naziva skripta navedena su dva parametra koji se redom dodeljuju parametrima specificiranim u skriptu. Tako će u ovom slučaju biti **\$1**=tekstovi, a **\$2**=filmovi. Korisnik može prilikom ponovnog korišćenja da promeni vrednost parametara.

Sledeći primer je Unix shell skript koji proverava password korisnika.

```
# ! /bin /sh
# Ovo je komentar
# Ovo je program za proveru pasvorda
VALID_PASSWORD ="secret" #ovo je korisnikov pasvord
echo "Molimo vas unesite pasvord :"
read PASSWORD
```

```
if [ "$PASSWORD" == "$VALID_PASSWORD" ] ; then
echo "Pristup dozvoljen !"
else
echo "Pristup odbijen !"
fi
```

PRIMER SKRIPTING JEZIKA OPERATIVNIH SISTEMA

Dat je spisak nekih skript jezika operativnog sistema

U ovu klasu skript jezika spadaju:

- AppleScript
- bash - Bourne Again Shell - Unix shell
- csh - C shell - Unix shell
- DCL - Digital Command Language za operativni sistem VMS
- JCL - Job Control Language
- ksh - Korn shell - Unix shell
- MS-DOS batch - skript jezik za Microsoft DOS
- REXX - REstructured EXtended eXecutor - IBM-ov skript jezik za operativne sisteme VM/CMS, MVS/TSO i OS/2.
- sh - Unix shell
- Winbatch - skript jezik za operativni sistem Windows.

MAKRO JEZICI

Makro jezici omogućavaju automatizaciju zadataka koji se često ponavljaju

Sa pojavom grafičkih korisničkih interfejsa počeli su da se koriste specijalni skript jezici nazvani makro jezici. Ovi jezici omogućavaju automatizaciju zadataka koji se često ponavljaju. Makro jezici omogućavaju interakciju sa prozorima, menijima, dugmadima i vrše akcije koje korisnik radi prilikom korišćenja operativnog sistema.

- AutoHotkey
- Autolt
- Expect

SKRIPT JEZICI APLIKACIJA

Skript jezici aplikacija služe za podešavanje radnog okruženja ili za automatizaciju čestih zadataka unutar programa

Mnoge složene aplikacije imaju svoje skript jezike koji služe za podešavanje radnog okruženja ili za automatizaciju čestih zadataka unutar programa. Na taj način se korisnik oslobođa od izvršavanja čestih istih procedura. Najpoznatiji skript jezik za aplikacije je svakako **VisualBasic**

for Application (VBA) koji se koristi za automatizaciju procedura unutar programa koji pripadaju Microsoft-ovom Office paketu. Program za 2D i 3D projektovanje AutoCad ima svoj skript jezik koji se zove AutoLisp. Pored automatizacije procedura, ovaj jezik se može iskoristiti i za proračun različitih geometrijskih i mehaničkih veličina ili za parametarsko projektovanje.

Dinamički skript jezici opšte namene

Neki jezici su kreirani kao skript jezici ali su tokom svog razvoja evoluirali u programske jezike opšte namene. Po osobinama, kao što su interpretiranje, dinamičnost i fleksibilnost u definisanju tipova promenljivih, ovi jezici spadaju u skript jezike.

U ovu klasu jezika spadaju:

- Perl
- PHP
- Python
- Ruby
- Tcl i drugi.

Ovde će se kao ilustracija ove klase skript jezika prikazati tipični predstavnici PHP i Python.

▼ 3.2 PHP

KADA SE KORISTI PHP?

PHP je skript jezik namenjen prevashodno za razvoj veb aplikacija koje se izvršavaju na serverskoj strani

Slika 1 - PHP

PHP je skript jezik namenjen prevashodno za razvoj veb aplikacija koje se izvršavaju na serverskoj strani. Inicijalno, skraćenica **PHP** je bila akronim od **Personal Home Page**. Zvanično ime ovog skript jezika je danas PHP: Hypertext Preprocesor. PHP se može koristi za:

- **Kreiranje dinamičkog veb sadržaja** pisanjem skripta koji se izvršava na serverskoj strani. Da bi se generisao HTML potreban je PHP parser i veb server koji će slati generisane HTML dokument klijentima. Pored toga PHP može da generiše i XML dokumente, grafiku, Flash animacije, PDF datoteke i druge specifične formate dokumenata.
- **Izvršavanje skripta pisanog u komandnoj liniji.** Ova funkcionalnost je slična onoj koju pruža recimo Unix shell.
- **Izradu GUI aplikacija koje se izvršavaju na klijentskoj strani.** Za ove potrebe se koristi biblioteka PHP-GTK.



Slika 3.2.1 PHP logo

[Izvor: <https://mpng.subpng.com/20180921/kvs/kisspng-php-image-magic-quotes-emblem-logo-php-vector-1-free-php-graphics-download-5ba4b4319ad475.2145990715375206896342.jpg>]

PHP je jedan od najšire podržanih i korišćenih **open-source** skript jezika. Može da se izvršava na svim značajnim operativnim sistemima kao što su: Windows, Unix, Linux i Mac OS X. Obzirom da je PHP omiljen u **open-source** zajednici najčešće se koristi u kombinaciji sa MySQL sistemom za upravljanje bazama podataka, mada se ravnopravno mogu koristiti i Oracle, Microsoft SQL Server, IBM DB2, PostgreSQL i drugi sistemi.

PHP može da radi u kombinaciji sa svim važnim veb serverima, mada se najčešće primenjuje uz Apache. Zbog vrlo česte upotrebe pojedinih platformi uz koje se koristi, stvorene su posebne skraćenice da opišu arhitekturu implementacije:

- **LAMP - Linux, Apache, MySQL, PHP**
- **WIMP - Windows, IIS, MySQL, PHP**
- **WAMP - Windows, Apache, MySQL, PHP**

Prvu verziju PHP je razvio Rasmus Lerdorf 1995. godine. U projekat se uključuju i dva izraelca, Zev Suraski i Andi Gutmans, koji su 1999 napisali novo jezgro PHP i nazvali ga Zend engine. Od 2004. godine je u opticaju verzija PHP 5 koja se stalno poboljšava.

PRIMENA PHP

PHP za kreiranje serverskih aplikacija koristi ogroman broj otvorenih biblioteka

Kao i drugi skript jezici, PHP definiše tip promenljivih dinamički, u trenutku izvršavanja skripta. Promenljive ne moraju biti deklarisane i mogu da čuvaju bilo koji tip objekta. Polja (vektori i matrice) mogu da budu heterogena, što znači da se u istom polju mogu nalaziti objekti različitog tipa.

Iako je skript jezik, PHP ima sve osobine objektno - orijentisanih jezika. Jedan od razloga za široku primenu PHP za kreiranje serverskih aplikacija je ogroman broj otvorenih biblioteka koje stoje na raspolaganju programerima.

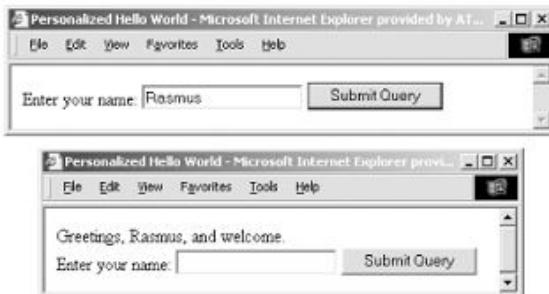
Kao primer koji treba da ilustruje primenu PHP-a za kreiranje dinamičke strane daje se PHP kod koji kreira i procesira jednostavnu formu koja dozvoljava unos imena. Kada korisnik unese ime ono se ispisuje na ekranu. Primer je preuzet iz [1].

PHP program pristupa vrednostima u formi preko **`$_POST`** i **`$_GET`** varijabila polja.

Forma i poruka koja se ispisuje unutar prozora su prikazane na slici 2.

```
<html>
<head>
<title>Personalized Hello World</title> </head>
<body>
<?php if(!empty($_POST['name'])) {>
echo "Greetings, ${_POST['name']}, and welcome.";
} ?>

<form action=<?php $PHP_SELF; ?>" method="post">
Enter your name: <input type="text" name="name" />
<input type="submit" />
</form>
</body>
</html>
```



Slika 3.2.2 Forma i poruka koja se ispisuje unutar prozora [Izvor: Autor]

PHP - VIDEO

What Is PHP?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 3.3 Python

KADA SE KORISTI PYTHON?

Python se koristi za pisanje skripta koji će automatizovati izvršenje zadataka operativnog sistema ili za povezivanje ranije napisanih softverskih komponenata

Python spada u skript jezike opšte namene. Uobičajeno se definiše kao objektno orijentisan skript jezik. Može da se koristi za pisanje skripta koji će automatizovati izvršenje zadataka

operativnog sistema ili za povezivanje ranije napisanih softverskih komponenata. Pored toga Python omogućava pisanje različitih mrežnih i veb aplikacija.

Po koncepciji je sličan jezicima Perl, Ruby, Smalltalk i Tcl. Razvijen je kao open-source projekat koga je inicirao Guido van Rossum 1990. godine. Piton ima standardni Internet modul koji omogućuje da Python programi obavljaju različite mrežne zadatke u klijentskom i serverskom režimu kao što su:

- komunikacija preko soketa,
- ekstrakcija informacija iz formi koje su poslate CGI skriptu na serverskoj strani,
- transfer datoteka preko FTP protokola,
- komunikacija preko XML-RPC, SOAP-a i telnet-a,
- čitanje i parsovanje HTML strana,
- obrada XML datoteka,
- slanje, prijem i slanje elektronske pošte.

Kao i Perl i Tcl, Python koristi CGI - **Common Gateway Interface** protokol za pernos informacija od veb servera do veb klijenta i obratno.

▼ Poglavlje 4

Skript jezici za programiranje klijentske strane

PISANJE SKRIPTA KOJI SE IZVRŠAVAJU NA KLIJENTSKOJ STRANI

Pisanje skripta koji se izvršavaju na klijentskoj strani (client-side scripting) se odnosi na klasu računarskih programa na vebu koji se izvršavaju u veb čitaču na klijentskom računaru.

Pisanje skripta koji se izvršavaju na klijentskoj strani (**client-side scripting**) se odnosi na klasu računarskih programa na vebu koji se izvršavaju u veb čitaču na klijentskom računaru. Pomoću ovakvog skripta se omogućuje da se prikaz na veb strani dinamički definiše zavisno od korisničkih akcija, uslova koji vladaju u okruženju i drugih promenljivih. Najpoznatiji skript jezici za programiranje klijentske strane su JavaScript i VBScript.

Skript se uobičajeno umeće unutar HTML dokumenta, ali se može nalaziti i u odvojenoj datoteci na koju postoji referenca unutar HTML dokumenta. Skript može da se pojavi više puta unutar jedne HTML strane i to unutar HEAD i BODY elemenata. Skript kod može da bude vezan za događaje i elemente pisanjem koda unutar HTML atributa. Ime atributa treba da bude neko od onih koje je podržano unutrašnjim događajima za taj element (na primer *onLoad* za sliku). Ovakav skript će se izvršiti uvek kada se dogodi ovaj događaj za definisani element.

HTML-ov model unutrašnjih događaja se sastoji od sledećih događaja:

- Dokument - **onLoad, onUnload**
 - Forma - **onSubmit, onReset**
 - Dokument i ulazno-izlazni elementi
- Focus- **onFocus, onBlur**
- Ulazno-izlazni elementi
- Mouse
- Movement - **onMouseOver, onMouseOut, onMouseMove**
- Button - **onClick, onDbClick, onMouseDown, onMouseUp**
- Keyboard - **onKeyPress, onKeyDown, onKeyUp**
- Drugi (TEXT/TEXTAREA)
- Gubljenje fokusa sa promenom sadržaja -**onChange** (takođe OPTION)

-- Izbor teksta - **OnSelect**

▼ 4.1 VBScript

NAČIN KORIŠĆENJA VBSCRIPT-A

VBScript je Microsoft-ovo rešenje za dodavanje dinamike veb stranama programiranjem klijentske strane

Visual Basic Script Edition([VBScript](#)) je Microsoft-ovo rešenje za dodavanje dinamike veb stranama programiranjem klijentske strane. Kao i drugi skript jezici i VBScript se interpretira, što je u ovom slučaju prepušteno Windows Scripting Host-u. VBScript je proistekao iz Visual Basic-a. Njegova primena je vrlo široka. VBScript se može izvršavati:

- u Windows GUI okruženju,
- sa komandne linije
- u Internet Explorer-u na klijentskoj strani i
- na veb serveru serverske strane, mada je VBScript pisan prevashodno za programiranje klijentske strane.

Kada se koristi u Internet Exploreru, VBScript može biti ugnježden u neku HTML stranu ili se pomoću njega može napraviti samostalna aplikacija. Kada je ugnježden u HTML stranicu Internet Explorer ga može interpretirati odmah nakon čitanja ili kasnije prilikom nekog događaja. Umetanje VBScript-a se vrši na sličan način kao i umetanje drugih skriptova. U narednom primeru, VBScript je umetnut unutar **BODY** elementa.

```
<html>
<head>
</head>
<body>
<script type="text/vbscript">
document.write("Pozdrav studentima FIT-a!")
</script>
</body>
</html>
```

Programer veb stranice je sloboden da umetne neograničeni broj skriptova unutar HTML stranice. Skript se može umetati i unutar zaglavlja i unutar tela stranice.

```
<html>
<head>
<script type="text/vbscript">
ovde su VBScript iskazi
</script>
</head>
<body>
<script type="text/vbscript">
```

```
ovde su VBScript iskazi
</script>
</body>
```

PROCEDURE, FUNKCIONALNE PROCEDURE I USLOVNI ISKAZI

VBScript ima mogućnost kreiranja podprogramske pod-procedura i funkcijskih procedura

Kao i drugi skript jezici VBScript može da radi sa promenljivima čiji se tip dinamički određuje. VBScript ima mogućnost kreiranja podprogramske pod-procedura i funkcijskih procedura. Pod-procedure se prilikom pozivanja izvršavaju, ali ne vraćaju nikakvu vrednost. Njima se mogu proslediti argumenti potrebni za izvršavanje.

Funkcionalne procedure mogu takođe da izvršavaju neke akcije, ali mogu i da vrate neku vrednost. Za upravljanje izvršenjem skripta se koriste različiti uslovni (If) iskazi i petlje. U sledećem primeru je prikazana primena if iskaza.

```
if payment="Gotovina" then
msgbox "Plaćanje se vrši gotovinom!"
elseif payment="Visa" then
msgbox "Plaćanje se vrši Visa karticom."
elseif payment="AmEx" then
msgbox "Plaćanje se vrši American Express karticom."
else
msgbox "Nepoznat metod plaćanja."
end If
```

VBSCRIPT - VIDEO

What is VBScript?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 4.2 Skript jezici za programiranje serverske strane

SKRIPT ZA SERVERSKU STRANU

Skripta se izvršava na veb serveru, te je moguće da se podaci za generisanje strane čitaju iz neke baze podataka

Postizanje dinamičnosti veb strana, pored programiranja klijentske strane, može da se postigne i pisanjem skripta za serversku stranu (**Server-side scripting**). U ovom slučaju skript se koristi da na osnovu zahteva klijenta dinamički generiše traženu HTML stranu. Obzirom da se u ovom slučaju skript izvršava na veb serveru, moguće je da se podaci za generisanje strane čitaju iz neke baze podataka. U odnosu na skript koji se izvršava na klijentskoj strani, programiranje klijentske strane ima prednost u tome što:

- ne zahteva korišćenje resursa klijentske strane, čime je sigurnost klijenta podignuta na znatno viši nivo,
- moguće je postići mnogo veći stepen dinamičnosti veb strana na osnovu zahteva klijenta,
- moguće je generisati stranu na osnovu upita serverskoj bazi podataka onako kako ga je definisao klijent.

Prvi skriptovi za serversku stranu su pisani u programskom jeziku C, skript jeziku Perl i Shell skriptu korišćenjem Common Gateway Interface protokola. Skript se pozivao i izvršavao u operativnom sistemu servera, a rezultat se prosleđivao klijentu preko veb servera. Na sličan način se za programiranje serverske strane mogu koristiti i skript jezici Python i Tcl.

Noviji skript jezici za serversku stranu, kao što su PHP i ASP se izvršavaju u samom veb serveru. U skript jezike za programiranje serverske strane spadaju:

- ASP - Microsoftovo rešenje koje se uglavnom koristi na Windows platformama
- ASP.NET - Novije Microsoft rešenje koje umesto skript jezika koristi kontrole i kod pisan kao .NET bytecode i obezbeđuje event-driven model
- ColdFusion - je rešenje kompanije Macromedia za generisanje dinamičkog veb sadržaja korišćenjem jezika **ColdFusion Markup Language (CFML)**. CFML nudi veliki broj elemenata za različite zadatke kao što su: pristup bazama podataka, datotekama, mejl i drugim serverima, upravljanje izvršenjem koda (if, petlje) i drugim elementima.
- JSP - **Java Server Pages**
- PHP
- Server-side JavaScript - skript jezik koji je projektovan prevashodno za programiranje klijentske strane, ali se ponekad koristi i na serverskoj strani.

4.3 ASP

ŠTA JE ASP?

ASP omogućuju autoru da unutar veb strane ugradi logiku pomoću VBScript-a i JScript-a kako bi se postigla potrebna dinamičnost

Microsoft-ove **Active Server Pages** (ASP) su popularna tehnologija za razvoj dinamičkih veb strana uglavnom u Windows okruženju. ASP omogućuju autoru da unutar veb strane ugradi logiku pomoću VBScript-a i JScript-a kako bi se postigla potrebna dinamičnost. Za kompleksne komponente moguće je ugraditi i ActiveX kontrole pisane u klasičnim programskim jezicima. Standardna distribucija uključuje i komponente za pristup bazama podataka. Kada klijent

pošalje zahtev serveru za nekom ASP stranom, kod u stranici se izvrši na serveru, a kombinacija statičkog i dinamičkog sadržaja se pošalje klijentu. ASP stranice se interpretiraju. ASP .NET, zadnja verzija ASP, ima brojne dodatne funkcije. Kao alternativa pisanju skripta, dinamički sadržaj se može generisati elementima koji su slični onima u HTML i XML-u. Na taj način se dobija funkcionalnost koja odgovara action elementima u JSP. ASP .NET strane se kompajliraju. ASP .NET se može koristiti ne samo za gradnju dinamičkih veb strana, nego i za pisanje veb aplikacija i XML veb servisa.

▼ 4.4 JSP

ŠTA JE JSP?

JSP se grade nad Java servletima. Ovaj skript jezik je projektovan sa idejom da se poveća produktivnost prilikom programiranja dinamičkog veb sadržaja.

JavaServer Pages(JSP) su deo Sun-ove kolekcije Enterprise Java uveden 1999. godine. JSP se grade nad Java servletima. Ovaj skript jezik je projektovan sa idejom da se poveća produktivnost prilikom programiranja dinamičkog veb sadržaja. Tehnologijom JSP se mogu generisati ne samo HTML dokumenti, nego i XML i drugi tipovi markap dokumenata. Servleti su delovi koda koji dodaju novu funkcionalnost serverima i to uglavnom veb serverima. Većina veb servera podržava servlete, a kako su pisani u programskom jeziku Java, mogu se koristiti na bilo kom OS-u. JSP može da se sastoji od sledećih delova:

- statičkih podataka kao što je na primer HTML kod
- JSP direktiva (kao što su include, page, taglib) koje ukazuju JSP kompjleru kako da generiše servlete.
- JSP skript elemenata i promenljivih.

Postoje tri osnovna skript elementa:

- **declaration tag**, kojim se definicije promenljivih smeštaju unutar tela java servlet klase
- **scriptlet tag**, kojim se iskazi smeštaju unutar _jspService() metode java servlet klase
- **expression tag**, kojim se izrazi koji treba da budu odreženi smeštaju unutar java servlet klase.

JSP akcija koje su u stvari XML tagovi koji pozivaju ugrađenu funkcionalnost veb servera. Na raspolaganju su sledeće akcije:

- **jsp:include**. Omogućuje da se unutar različitih JSP-ova umetne neki JSP kod, koji se ponaša kao potprogram.
- **jsp:param**. Služi da se zadaju parametri koji će biti dodati tekućim parametrima zahteva
- **jsp:forward**. Koristi se za prosleđivanje zahteva i odgovora ka drugim JSP i servletima. Kontrola se nikad ne vraća trenutnoj JSP.
- **jsp:plugin**. Generiše tag za uključivanje apleta, koji odgovara datom veb čitaču

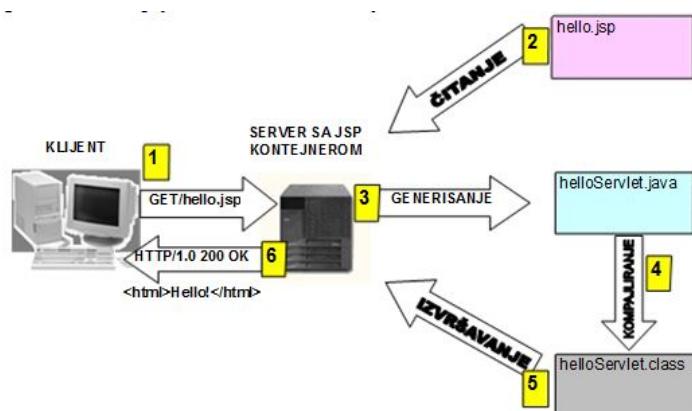
- **jsp:fallback.** Prikazuje odgovarajući sadržaj ukoliko veb čitač ne podržava aplete.
- **jsp:getProperty.** Uzima karakteristike iz specificiranog Java bean-a
- **jsp:setProperty.** Setuje karakteristike u specificiranom Java bean-u
- **jsp:useBean.** Kreira ili koristi Java bean raspoloživ u tekućoj JSP strani.

Korisnik može da kreira JSP tag biblioteke koje se koriste kao proširenje standardnih HTML i XML tagova. Pomoću biblioteka tagova se mogu proširiti mogućnosti veb servera nezavisno od platforme na kojoj je instaliran.

JSP KOMPAJLIRANJE I MVC ŠABLON

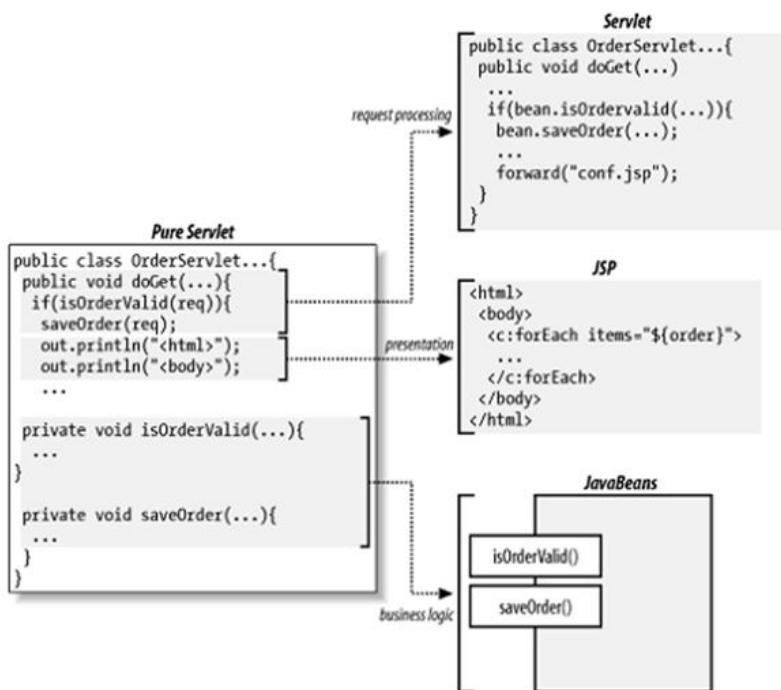
Za kompajliranje JSP se koristi JSP kompajler. Za pisanje JSP preporučuje se korišćenje MVC šablon kako bi se odvojila prezentacija od obrade zahteva i memorisanja podataka.

Za kompajliranje JSP se koristi JSP kompajler. Kao rezultat kompajliranja dobijaju se servleti. Servleti mogu da budu kompajlirani kao Java kod ili byte kod.



Slika 4.1.1 Prikaz kompajliranja JSP-a [Izvor: Autor]

Za pisanje JSP preporučuje se korišćenje MVC šablon (Model-View-Controller) kako bi se odvojila prezentacija od obrade zahteva i memorisanja podataka. Na sledećoj slici je prikazan primer koji ilustruje primenu MVC šablonata.



Slika 4.1.2 Primena MVC šablona [Izvor: https://lh4.googleusercontent.com/PDla4IAANoYEvihiWFN698BCfoi_zegCr4Yi-HCAtoFehNYsFKcwRKJy4EQWStkersrfjRIQzdQQGpuKoKcmKoBpBro7nnizpwiXJO0ZmEw1H1Xa2nE]

PREDNOSTI JSP-A

JSP omogućuje postizanje dinamičnosti pomoću elemenata i pisanjem skripta

Odvajanje obrade zahteva i biznis logike od prezentacije omogućuje da se razvojni zadatak podeli programerima sa različitim veštinama. Java programeri razvijaju obradu zahteva i biznis logiku a autori veb strana projektuju korisnički interfejs i prezentaciju. Svaki deo aplikacije je moguće menjati nezavisno.

U odnosu na druge tehnologije za programiranje klijentske strane JSP ima sledeće prednosti:

- JSP omogućuje postizanje dinamičnosti pomoću elemenata i pisanjem skripta
- JSP omogućuje programerima da kreiraju korisničku biblioteku tagova koja odgovara potrebama specifičnih aplikacija
- JSP se kompajliraju čime se postiže brže izvršavanje na serveru
- JSP se mogu koristiti u kombinaciji sa servletima koji su zaduženi za biznis logiku
- JSP je specifikacija, a ne proizvod, što znači da se različiti dobavljači trude da ponude kvalitetniji proizvod
- JSP je integralni deo J2EE, što znači da se JSP mogu koristiti kako za najjednostavnije aplikacije, tako i za najsloženije.

JSP - VIDEO

JSP Tutorial #2 - JSP and Servlets Overview

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 5

Tehnike pisanja koda

TRI TEHNIKE PISANJA KODA

Tehnike pisanja koda su: adopt, adapt i make

U trenutku kada je doneta odluka da se napravi novo softversko rešenje, projektanti su u prilici da izaberu jedan od tri načina da se dođe do rešenja. Programerima su na raspolaganju sledeće tehnike:

- Adopt - prihvatanje postojećeg softverskog rešenja
- Adapt - prilagođavanje postojećeg softverskog rešenja
- Make - gradnja sopstvenog potpuno novog rešenja od početka.

Pre nego što se doneše odluka koja će se od navedenih tehnika koristiti, potrebno je jasno definisati ciljeve i izvršiti analizu troškovi/dobit. Pored toga potrebno je poznavati prednosti i nedostatke pojedinih tehnika.

✓ 5.1 Tehnike pisanja koda - Adopt

TEHNIKA PISANJA KODA - ADOPT

Adopt se koristi kada postoji softversko rešenje koje odgovara potrebama organizacije, kada je cena razvoja sopstvenog rešenja veća na postojeće, i kad se ne može čekati na razvoj sopstveno

Postoje mnoga opšta softverska rešenja koja su specijalizovana za automatizaciju pojedinih standardnih procesa čija je metodologija opšte prihvaćena. U ovu kategoriju spadaju programi koji su u svakodnevnoj širokoj upotrebi, ko što su, na primer, programi za obradu teksta ili slanje elektronske pošte, ali i programi koji se koriste za neke vrlo specifične zadatke kao što su, na primer, programi za analizu ponašanja konstrukcija metodom konačnih elemenata. Za razvoj ovakvih programa potrebno je utrošiti na stotine programer/godina. Cena ovih programa uglavnom je daleko ispod cene razvoja sopstvenog rešenja.

Ukoliko imaju potrebu za ovakvim aplikacijama, korisnici se uglavnom opredeljuju za kupovinu gotovih aplikacija i njihovim prihvatanjem (**adopt**). Korisnici ovakvih programa uobičajeno nemaju potrebu za njihovim menjanjem jer su rešenja dovoljno opšta da odgovaraju korisničkim potrebama. Većina ovakvih aplikacija nije predviđena za veća

prilagođavanja korisnicima. Umesto toga, očekuje se da se korisnici prilagode programskom rešenju.

Adopt tehnika se koristi kada:

- Na tržištu postoji softversko rešenje koje odgovara potrebama organizacije,
- Kada je cena razvoja sopstvenog rešenja veća u odnosu na cenu postojećeg rešenja
- Kada se zbog vremenskih ograničenja ne može čekati na razvoj sopstvenog rešenja, a na tržištu već postoji isto ili slično rešenje.

Prednosti korišćenja adopt tehnike su:

- Do rešenja se dolazi vrlo brzo
- Cena gotove aplikacije je daleko ispod cene razvoja novog rešenja
- Ne troše se ljudski i materijalni resursi za razvoj aplikacije
- Dobija se pouzdana aplikacija, tj. aplikacija čije su osobine poznate

Nedostaci korišćenja adopt tehnike su:

- Korisnik ne može da menja aplikaciju niti da je u većoj meri prilagođava svojim potrebama. Umesto toga, korisnik se prilagođava aplikaciji.
- Korisnik je zavisan od dobavljača.

✓ 5.2 Tehnike pisanja koda - Adapt

PREDNOSTI I NEDOSTACI ADAPT TEHNIKE PISANJA KODA

To su rešenja koja imaju određenu standardnu funkcionalnost, ali se mogu prilagođavati i adaptirati u cilju zadovoljenja specifičnih zahteva klijenta.

Slične poslovne organizacije uobičajeno imaju približno iste zahteve za softverskim rešenjima. Ovo je zbog toga što su mnogi poslovni procesi standardizovani ili kao posledica primene dobre prakse ili zbog zakonske regulative. Ipak, svaka organizacija ima svoje specifičnosti kao što su: struktura organizacije, poslovni procesi, poslovna pravila. Zbog toga mnoge softverske kuće nude programska rešenja koja imaju određenu standardnu funkcionalnost, ali se mogu prilagođavati i adaptirati u cilju zadovoljenja specifičnih zahteva klijenta. Uobičajeno se ovakav softver pravi modularno, tako da korisnik može da izabere one module koji su mu u datom trenutku potrebni, ne opterećujući nepotrebno svoj budžet. Kada programsko rešenje nije moguće prilagoditi potrebama klijenata i njegovim poslovnim procesima, klijent usklađuje svoje poslovne procese sa mogućnostima programa.

Prednosti primene adapt tehnike su:

- Polazi se od gotove aplikacije koja je prethodno iztestirana i proverena u praksi u mnogim organizacijama

- Vreme adaptiranja softvera je uobičajeno kraće od vremena razvoja novog rešenja
- Isporučilac softvera stalno radi na usavršavanju rešenja i isporučuje nove verzije
- Cena razvoja programa se deli na više organizacija, tako da su investicije klijenata mnogo manje nego pri samostalnom razvoju
- Inicijalna ulaganja su manja nego za razvoj sopstvenog programskog rešenja
- Nisu potrebni kadrovski i materijalni resursi kakvi su potrebni za razvoj sopstvenog programskog rešenja.

Adapt tehnika ima i neke nedostatke, kao što su:

- Prilagođavanje složenih softverskih rešenja nekad može da traje i godinama
- Prilagođavanje zahteve angažovanje eksperata čija je cena rada uobičajeno visoka, tako da cena uvođenja i prilagođavanja može da bude i nekoliko puta veća od nabavne cene softvera.
- Korisnik ne može ili može u vrlo maloj meri da promeni kod
- Korisnik je zavisan od dobavljača. Ako dobavljač propadne, korisnik ostaje bez podrške
- U nekim slučajevima softversko rešenje nije moguće potpuno prilagoditi zahtevima, pa je neophodno redefinisanje poslovnih procesa.

PRIMER ADAPT TEHNIKE I RASPOLOŽIVE METODE PRILAGOĐAVANJA

Tipičan primer primene adapt tehnike je implementacija ERP (Enterprise Resource Planning) sistema.

Tipičan primer primene adapt tehnike je implementacija ERP ([Enterprise Resource Planning](#)) sistema. Ovo su izrazito složeni sistemi koji se sastoje od velikog broja takođe složenih modula, kao što su: Upravljanje proizvodnjom, upravljanje ljudskim resursima (HRM), upravljanje klijentima (CRM), upravljanje lancem snabdevanja (SCM), prodaja, računovodstvo, poslovna inteligencija itd. Malo je organizacija, pa čak i među najvećim, koje ulaze u rizik sopstvenog razvoja ovakvih sistema. S druge strane na tržištu postoje rešenja koja su opšta ili blago specijalizovana za pojedine oblasti poslovanja. Korisnici kupuju ovakva rešenja i plaćaju istoj ili nekoj drugoj specijalizovanoj organizaciji za prilagođavanje sistema.

Za prilagođavanje programskog rešenja korisniku, koriste se više ili manje fleksibilne metode. U raspoložive metode prilagođavanja, između ostalog, spadaju:

- izbor potrebnih opštih modula
- izbor specijalizovanih modula za neku privrednu granu
- prilagođavanje izgleda izlaznih dokumenata uz uključivanje elemenata vizuelnog identiteta korisnika (logo, boja i slično)
- promena parametara programa kojim se menja logika izvršenja programa, ulazne forme i izveštaji
- promena upita u bazi podataka
- kreiranje specijalizovanog front-end rešenja prema zahtevima korisnika uz korišćenje standardizovane baze podataka i procedura u njoj

- pisanje specijalizovanih makroa za izvršavanje specifičnih zadataka jezikom za programiranje same aplikacije (na primer VBA)
- pisanjem posebnih programskih modula nekim skript jezikom radi dodavanja funkcionalnosti koja je potrebna određenom korisniku

▼ 5.3 Tehnike pisanja koda - Make

MAKE

Tehnika make se odnosi na izgradnju potpuno novog sopstvenog programskog rešenja

Tehnika make se odnosi na izgradnju potpuno novog sopstvenog programskog rešenja. Ova tehnika se primenjuje kada na tržištu nema rešenja koje odgovara potrebama korisnika ili kada se zbog drugih razloga proceni da je efikasnije ući u sopstveni razvoj. Organizacija koja se opredeli za ovu tehniku mora da bude kadrovski sposobna da izvede jedan ovakav projekat.

Prednosti primene make tehnike su:

- Korisnik je vlasnik koda pa su sve izmene i dopune moguće
- Programsko rešenje se može potpuno prilagoditi zahtevima korisnika
- Nije potrebno menjati poslovne procese da bi se prilagodili programskom rešenju
- Uvođenje rešenja je olakšano jer nema velikih prilagođavanja, a i eksperti su iz iste organizacije
- Dobro programsko rešenje može biti proizvod koji se može ponuditi partnerima ili izbaciti na tržište

Nedostaci make tehnike su:

- Potrebni su znatni ljudski i materijalni resursi za projektovanje, razvoj, uvođenje i održavanje programa
- Cena razvoja je uobičajeno veća nego kupovina gotovog rešenja, ukoliko takvo ili slično postoji na tržištu
- Postoji rizik da projekat razvoja rešenja ne bude dobar, ili da nikad ne bude završen
- Do rešenja se ne može doći tako brzo kao kupovinom gotovog rešenja

✓ Poglavlje 6

Pokazna vežba - XAMPP instalacija

ŠTA JE XAMPP?

XAMPP je softverski paket sadrži Apache distribucije za Apache server

(Ukupno vreme trajanja iznosi 45 minuta.)

XAMPP softverski paket sadrži Apache distribucije za Apache server, MariaDB, PHP i Perl. To je u osnovi lokalni host ili lokalni server. Ovaj lokalni server radi na vašem ličnom laptop računaru. Možete jednostavno da instalirate ovaj softver na svoj računar i testirate klijente ili vaše veb sajtove pre nego što ih otpremite na udaljeni veb server.

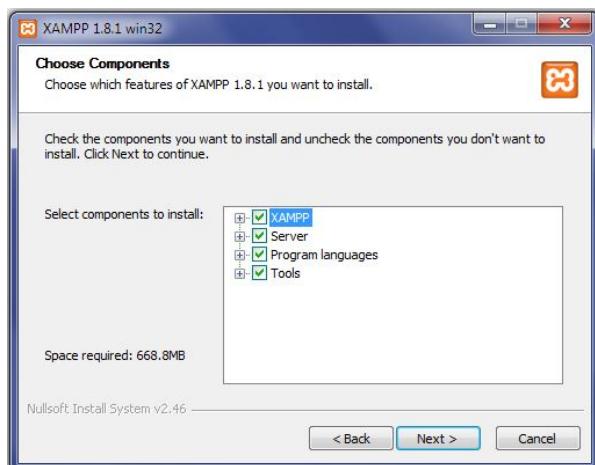
Instalaciju možete preuzeti sa sledećeg sajta: <https://www.apachefriends.org/download.html>

POČETAK INSTALACIJE

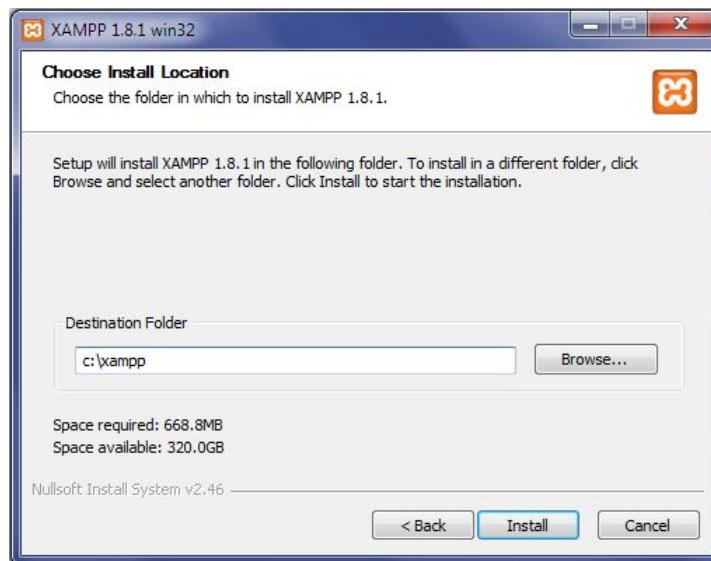
U ovoj sekciji biće prikazani koraci u instalaciji XAMPP-a



Slika 6.1 Početak instalacije [Izvor: Autor]



Slika 6.2 Odabir komponenti za instalaciju [Izvor: Autor]



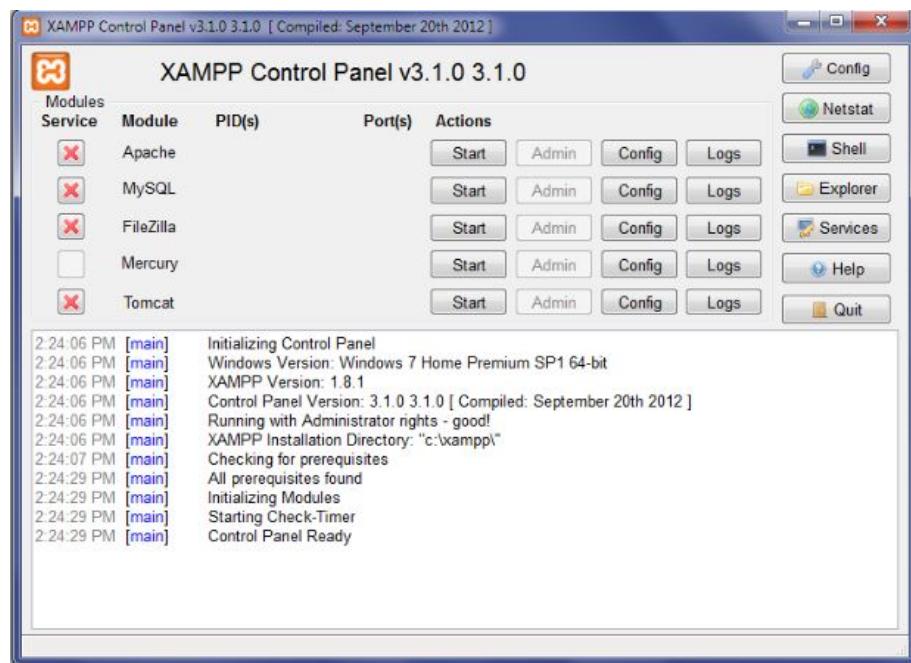
Slika 6.3 Odabir lokacije instalacije [Izvor: Autor]

Uglavnom se ostavlja default lokacija, odnosno C disk. Nakon ovog koraka, klikom na dugme Install započeće instalacija.

POKRETANJE XAMPP-A

U ovoj sekciji biće prikazano kako se pokreće XAMPP

Nakon uspešne instalacije, otvara se XAMPP Control Panel (Slika 4)



Slika 6.4 XAMPP Control Panel [Izvor: Autor]

Ovde se mogu pokrenuti svi potrebni servisi. Za ove vežbe, dovoljno je pokrenuti Apache i MySQL.

▼ Poglavlje 7

Pokazna vežba – Uvod u PHP

OSNOVNI POJMOVI

PHP je skript jezik opšte namene koji je pogodan za Web razvoj

(Ukupno vreme trajanja iznosi 45 minuta.)

PHP je skript jezik opšte namene koji je pogodan za Web razvoj. Po nekim istraživanjima, PHP se nalazi na četvrtom mestu najpopularnijih programskih jezika, odmah iza Java, C i Visual Basic jezika. PHP se izvršava na Web serveru gde se kao ulaz pojavljuje PHP kod, a izlaz je Web strana.

Postoje programski paketi koji omogućavaju kreiranje dinamičkih Web strana na računaru koji ne mora da bude povezan na WWW. WAMP rešenja su paketi programa kreiranih za Windows OS. WAMP je akronim, kreiran od naziva operativnog sistema (Windows) i osnovnih komponenti paketa: Apache, MySQL i PHP.

Apache je Web server koji omogućava pregled Web strana uz pomoć standardnih čitača poput Internet Explorer-a ili Firefox-a. MySQL je program za upravljanje bazama podataka.

PHP je jezik koji omogućava manipulaciju podacima u bazi i generisanje Web strana. Slično, postoji i LAMP rešenje za Linux OS. Za razliku od samog Windows OS, komponente WAMP/LAMP paketa su open source.

PRVI PHP PROGRAM

PHP skript počinje sa <?php a završava se sa ?> znakom

Blok u dokumentu koji se odnosi na PHP skript počinje sa **<?php**, a završava se sa **?>**. Ovaj blok može se nalaziti bilo gde u HTML dokumentu. PHP datoteka obično sadrži HTML tagove i dodatni skript kod.

Primer:

Kreirati PHP skript koji će u čitaču prikazati poruku: „**Pozdrav!**“.

Postoje dve komande za slanje i prikazivanje nekog teksta u čitaču: print i echo.

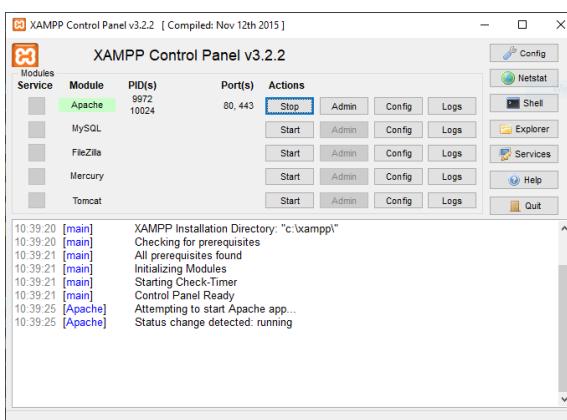
Dokument može da izgleda ovako:

```
<html>
<body>
```

```
<?php
    echo "Pozdrav!";
?
</body>
</html>
```

Svaka linija koda završava se sa **tačkom-zarez**. To je separator i koristi se da razdvoji jedan skup instrukcija od drugih. Sadržaj se na već poznati način unosi u Notepad i potrebno je da se snimi na adresu: **c:\xampp\htdocs**. Dokument snimamo kao npr. "primer1.php".

Da bi dokument bio pravilno prikazan potrebno je pokrenuti Apache servis u XAMPP control panelu.



Slika 7.1.1 Pokretanje Apache servisa [Izvor: Autor]

Nakon toga u čitač se unosi sledeća adresa: <http://localhost/private/primer1.php>.

Pojavljeće se snimljeni dokument.

KOMENTARI

U PHP jeziku znaci // i / */ se koriste za komentare*

U PHP jeziku znaci **//** se koriste za komentare koji se nalaze u jednom redu, a znaci **/* */** za duže komentare.

```
<html>
    <body>
        <?php

            //Ovo je komentar

            /*Ovo je
            duži
            komentar
            */

        ?>
```

```
</body>  
</html>
```

PROMENLJIVE

Promenljive se koriste za čuvanje vrednosti, kao što su alfa-numerički znaci, brojevi ili nizovi

Promenljive se koriste za čuvanje vrednosti, kao što su alfa-numerički znaci, brojevi ili nizovi. Kada se uvede, promenljiva, odnosno njena vrednost, se može koristiti u datom skriptu uvek kada je to potrebno. Sve promenljive počinju znakom \$. Pravilan način korišćenja promenljive:

\$ime_promenljive = vrednost;

Čest je slučaj da se tokom pisanja skripta izostavi znak \$. U tom slučaju skript neće funkcionisati. Na primer, ako želimo da u skriptu koristimo jednu string i jednu celobrojnu promenljivu, to može da izgleda ovako:

```
<?php  
    $txt = "FIT";  
    $broj = 54;  
?>
```

U PHP jeziku nije potrebno deklarisati promenljive pre nego što se koriste. PHP automatski konvertuje promenljivu u odgovarajući tip, u zavisnosti od načina njihovog korišćenja. Naziv promenljive mora na početku da sadrži slovo ili znak underscore (_).

Na ostalim mestima u nazivu mogu se naći mala i velika slova, underscore ili brojevi. Naziv ne bi trebalo da sadrži razmake. Ako se, ipak, sastoji od više od jedne reči one se razdvajaju znakom underscore ili velikim slovima (npr. \$ovo_je_promenljiva ili ovoJePromenljiva).

ARITMETIČKI OPERATORI I OPERATORI DODELJIVANJA

U PHP-u se mogu koristiti aritmetički operatori. Operatori dodeljivanja dodeljuju vrednost promenjivoj

U PHP-u se mogu koristiti različiti operatori, dati na sledećim slikama.

Operator	Opis	Primer	Rezultat
+	Sabiranje	$x = 3$ $x + 2$	5
-	Oduzimanje	$x = 3$ $6 - x$	3
*	Množenje	$x = 3$ $x * 6$	18
/	Deljenje	$6 / 3$	2
%	Modul	$6 \% 3$ $8 \% 3$	0 2
++	Inkrement	$x = 3$ $x++$	$x = 4$
--	Dekrement	$x = 3$ $x--$	$x = 2$

Slika 7.1.2 Aritmerički operatori [Izvor: Autor]

Operator	Primer	Rezultat
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
.=	$x .= y$	$x = x.y$
%=	$x \%= y$	$x = x \% y$

Slika 7.1.3 Operatori dodeljivanja [Izvor: Autor]

OPERATORI POREĐENJA

Operatori poređenja porede vrednosti

Operatori poređenja:

Operator	Opis	Primer
<code>==</code>	jednako	<code>3 == 6</code> daje rezultat false
<code>!=</code>	nije jednako	<code>3 != 6</code> daje rezultat true
<code>></code>	veće od	<code>3 > 6</code> daje rezultat false
<code><</code>	manje od	<code>3 < 6</code> daje rezultat true
<code>>=</code>	veće od ili jednako	<code>3 >= 6</code> daje rezultat false
<code><=</code>	manje od ili jednako	<code>3 <= 6</code> daje rezultat true

Slika 7.1.4 Operatori poređenja [Izvor: Autor]

LOGIČKI OPERATORI

Logički operatori izvršavaju logičke operacije nad operandima

Logički operatori:

Operator	Opis	Primer
<code>&&</code>	logičko I	<code>x = 3</code> <code>y = 6</code> <code>(x < 9 && y > 0)</code> daje rezultat true
<code> </code>	logičko ILI	<code>x = 3</code> <code>y = 6</code> <code>(x == 3 y == 3)</code> daje rezultat true
<code>!</code>	negacija	<code>x = 3</code> <code>y = 6</code> <code>!(x == y)</code> daje rezultat true

Slika 7.1.5 Logički operatori [Izvor: Autor]

KONTROLA TOKA

If-else konstrukcija se koristi za kontrolu toka programa

Često tok izvršenja nekog programa zavisi od ispunjenja određenih uslova, odnosno izvršavaće se različite instrukcije programa, u zavisnosti od definisanih odluka. Ove odluke se zovu i uslovne naredbe. Naredba **if...else** se koristi kada je potrebno izvršiti skup nekih instrukcija kada je ispunjen zadati uslov, odnosno neki drugi skup instrukcija ako taj uslov nije ispunjen. Naredba **elseif** se koristi u kombinaciji sa **if...else** naredbom kada je potrebno izvršiti deo programa ako je ispunjen jedan od navedenih uslova.

Opšti slučaj **if...else** naredbe izgleda ovako:

```
if (uslov)
    kod koji će se izvršiti ako je uslov ispunjen;
else
    kod koji će se izvršiti ako uslov nije ispunjen;
```

ELSEIF KONSTRUKCIJA

Elseif konstrukcija je kombinacija else i if konstrukcija

U slučajevima kada je potrebno izvršiti deo koda ako je ispunjen jedan od nekoliko uslova koristi se elseif naredba na sledeći način:

```
if (uslov1)
    kod koji će se izvršiti ako je uslov1 ispunjen;
else if (uslov2)
    kod koji će se izvršiti ako je uslov2 ispunjen;
else
    kod koji će se izvršiti ako uslov nije ispunjen;
```

Sledi primer **elseif** konstrukcije. U ovom slučaju ako je ispunjen uslov da je zadatak dobar i da se radi o 15. domaćem zadatku, pojaviće se prva poruka.

Ako je samo ispunjen uslov da je zadatak tačan, u čitaču će se pojaviti drugi tekst.

Konačno, ako nije ispunjen nijedan od prethodna dva uslova pojaviće se informacija da zadatak nije dobar.

```
<html>
<body>
<?php
    $redniBroj=15;
    $zadatak="dobar";

    if ($redniBroj==15 && $zadatak=="dobar") {
        echo "Čestitam! Uradili ste sve domaće zadatke.";
    }
    else if ($zadatak=="dobar") {
        echo "Odlično!";
    }
    else {
        echo "Žao mi je. <br/>";
        echo "Zadatak nije dobar.";
    }
?>
</body>
</html>
```

WHILE PETLJA

While petlja je osnovna konstrukcija za ponavljanje jednog dela koda

U programiranju je čest slučaj da je potrebno jedan isti kod izvršavati više puta. PHP za te namene koriste petlje: **while**, **do...while**, **for** i **foreach**.

Petlja while će se izvršavati sve dok je ispunjen navedeni uslov:

```
while (uslov)
    kod koji će se izvršiti;
```

Primer:

Koristeći while petlju ispisati nazine 15. nedelja u semestru u formatu: „x. nedelja“, gde je x redni broj nedelje.

Na početku skripta je definisana promenljiva *i*, kojoj je dodeljena vrednost 1. Kod koji se nalazi u while petlji, izvršavaće se sve dok je ispunjen uslov da je *i* ≤ 15 , pri čemu se svakim izvršavanjem petlje vrednost promenljive uvećava za jedan (**\$i++**).

Pre poslednjeg izvršavanja petlje promenljiva *i* se uvećava za jedan i postaje jednakata petnaest. Nakon prikazivanja te vrednosti, vrednost promenljive se ponovo povećava za jedan (*i* = 16), pa uslov za izvršavanje petlje više nije ispunjen. što znači da skript prestaje sa izvršavanjem.

U delu sa echo komandom, za spajanje prikaza vrednosti promenljive i teksta: „.. nedelja“, koristili smo tačku. To je tzv. concatenation operator koji npr. spaja dva teksta u jedan prikaz.

DO-WHILE PETLJA

Do-while petlja se koristi kada je telo petlje potrebno izvršiti bar jednom

Kod while petlje, izvršavanje koda unutar petlje nije obavezno, odnosno ako uslov nije na početku ispunjen kod se neće izvršiti.

To bi se desilo da smo definisali na početku da je npr. *\$i=16*. Petlja do...while, za razliku od while petlje, izvršiće kod bar jednom. U opštem slučaju, **do...while** petlja izgleda ovako:

```
do
{
    kod koji će se izvršiti;
}
while (uslov);
```

Na primer, prethodni primer bi u tom slučaju mogao da izgleda ovako:

```
<html>
<body>
<?php
    $i=0;

    do
    {
        $i++;
        echo "$i . nedelja<br/>";
    }

    while ($i<15);
?>
</body>
</html>
```

FOR PETLJA

For petlja je kraća verzija while petlje

Na sličan način se koristi i **for** petlja. Ova komanda se koristi kada se unapred zna broj izvršavanja. Opšti oblik izgleda ovako:

```
for (inicijalizacija; uslov; inkrement)
{
    kod koji će se izvršiti;
}
```

Petlja for ima tri parametra. Prvi parametar inicijalizuje promenljivu, drugi predstavlja uslov, a treći sadrži inkrement potreban za izvršavanje petlje.

Inicijalno se promenljivoj dodaje vrednost jedan, i ona se u toku izvršavanje petlje inkrementalno povećava za jedan (**\$i++**) sve dok je ispunjen uslov da je: **\$i<=15**.

Korišćenje ove petlje za poslednji primer ispisivanja naziva nedelja može da izgleda ovako:

```
<html>
<body>
<?php
    for ($i=1; $i<=15; $i++)
    {
        echo "$i. nedelja<br/>";
    }
?>
</body>
</html>
```

FUNKCIJE

Funkcija se definiše koristeći ključnu reč function

Sintaksa funkcija u PHP jeziku je slična kao u JS.

```
function foo($arg_1, $arg_2, /* ... , */ $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
```

Funkcija se definiše koristeći ključnu reč function. U zagradi se nabrajaju parametri funkcije.

Ključna reč return vraća vrednost iz funkcije.

Prilikom poziva funkcije, prosledjuju se stvarni parametri. Na primer:

```
foo(1,"asdf",5)
```

▼ 7.1 Pokazna vežba: PHP parametri

UVOD U PHP

HTML forme su način da se podaci prikupe od korisnika i proslede prema nekom udaljenom serveru, a zatim dalje obrade

(Ukupno vreme trajanja iznosi 45 minuta.)

HTML forme su način da se podaci prikupe od korisnika i proslede prema nekom udaljenom serveru, a zatim dalje obrade. Jedan od primera sa kojim se korisnici često sreću jeste korišćenje nekog pretraživača. Na primer, osnovu korisničkog interfejsa Google pretraživača čini Web forma sa jednim poljem za unošenje pojmoveva na osnovu kojih se vrši pretraživanje i dva dugmeta za prosleđivanje.

Jedna od stvari na koju posebno treba obratiti pažnju prilikom rada sa PHP jezikom je mogućnost automatskog korišćenja sadržaja neke HTML forme u PHP skriptu.

PRIMER 1

Kreiranje HTML forme za automatski prikaz imena i starosti korisnika

(Predviđeno vreme izrade zadatka iznosi 20 minuta.)

Zadatak:

Kreirati HTML formu i odgovarajući PHP skript koji će omogućiti da se automatski prikažu ime i starost korisnika. Formu čine dva polja, jedno za unos imena i drugo za unos godina.

Rešenje:

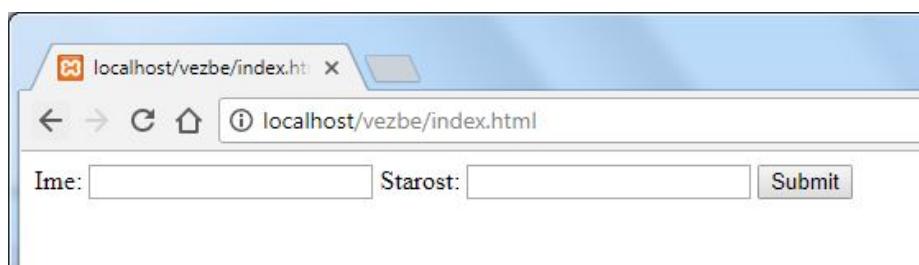
HTML dokument sa dva polja za unos podataka može da izgleda ovako:

```
<html>
  <body>
    <form action="prikaz.php" method="post">
      Ime: <input type="text" name="ime" />
      Starost: <input type="text" name="starost" />
      <input type="submit" />
    </form>
  </body>
</html>
```

Dокумент se snima kao npr. primer1.html. Ovaj dokument sadrži dva polja za unos teksta i jedno Submit Query dugme. Kada se u polja unesu odgovarajući podaci, oni se prosleđuju PHP skriptu koji se nalazi u datoteci prikaz.php. Skript može da izgleda ovako:

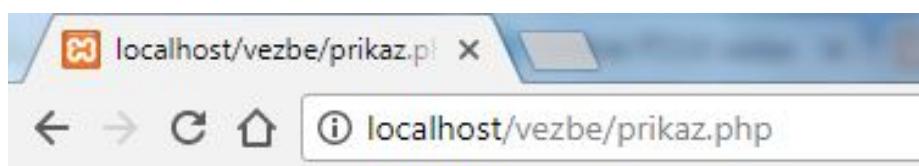
```
<?php echo $_POST["ime"]; ?> je star
<?php echo $_POST["starost"]; ?> godina
```

Nakon snimanja skripta i otvaranja HTML dokumenta uz pomoć dobija se forma data na slici 1.



Slika 7.2.1 Dobijena forma [Izvor: Autor]

U zadata polja unose se parametri i kada se klikne na Submit Query, isti se prosleđuju PHP skriptu. Dobija se rezultat prikazan na slici 2.



Petar je star 19 godina

Slika 7.2.2 Dobijen rezultat nakon što se klikne na Submit Query [Izvor: Autor]

PRIMER 1 – PREUZIMANJE I VALIDACIJA PODATAKA

*Datoteka koristi promenljivu **<code>\$</code>_POST***

Prilikom posleđivanja, URL ne sadrži nikakve dodatne informacije i izgleda ovako:

`http://localhost/prikaz.php`

Nakon posleđivanja, datoteka **prikaz.php** koristi promenljivu **`$POST`** za preuzimanje podataka. Ova promenljiva predstavlja niz imena promenljivih i odgovarajućih vrednosti, posleđenih uz pomoć HTTP POST metode. Ova promenljiva se koristi za prikupljanje vrednosti iz forme uz pomoć metode koja se u dokumentu definiše sa **`method="post"`**. Informacije koje se posleđuju ovom metodom nisu vidljive za druge i ne postoji nikakva ograničenja vezana za npr. dužinu podataka koji se mogu proslediti.

Prikazani postupak često se koristi za **validaciju podataka** unetih u formu. Validaciju treba raditi uvek kada je to moguće. **Validacija na klijentskoj strani je brža i smanjuje potrebu da se koristi server**. Međutim, kod sajtova koji imaju veliki broj korisnika, gde je potrebno voditi računa o raspoloživim resursima servera, važno je razmišljati i o bezbednosti. **Validaciju podataka na serverskoj strani treba koristiti uvek kada postoji potreba da se pristupi nekoj udaljenoj bazi podataka**.

Dobar način za validaciju na serverskoj strani je da forma posledi parametre samo sebi, umesto prelaska na drugu stranu. U tom slučaju, korisnik će dobiti poruku o grešci na istoj strani na kojoj se nalazi i forma. To olakšava pronalaženje grešaka.

PRIMER 1 – GET METODA

Osim promenljive

POST, zaprosleđivanje parametara sekorištipromenljiva _GET

Osim promenljive **`$POST`**, za posleđivanje parametara koristi se i promenljiva **`$GET`**, koja koristi odgovarajući GET metod. Promenljiva **`$GET`** predstavlja skup imena promenljivih i vrednosti koje se šalju uz pomoć HTTP GET metode. Informacije koje se šalju iz forme uz pomoć GET metode su vidljive za svakog, odnosno prikazaće se u adresnom prostoru čitača. Postoji i ograničenje vezano za podatke koji se mogu proslediti, tako da je dužina ograničena na 100 karaktera.

Ako bismo formu iz prvog primera promenili tako da sada izgleda ovako:

```
<form action="welcome.php" method="get">
    Ime: <input type="text" name="ime" />
    Starost: <input type="text" name="starost" />
    <input type="submit" />
</form>
```

kada se klikne na Submit Query dugme, URL koji se posleđuje izgledaće ovako:

`http://localhost/private/prikaz.php?ime=Petar&starost=19`

U tom slučaju PHP skript treba promeniti tako da koristi `$_GET` promenljivu:

```
<?php echo $_GET["ime"]; ?> je star  
<?php echo $_GET["starost"]; ?> godina.
```

Imajući u vidu da se imena promenljivih i njihove vrednosti vide u okviru URL, GET metod ne bi trebalo koristiti za prosleđivanje šifara ili drugih važnih poverljivih informacija. Sa druge strane, na ovaj način omogućen je tzv. bookmark datestrane, što je ponekad veoma korisno.

Osim navedenih, koristi se i `$_REQUEST` koja sadrži vrednosti promenljivih `$_POST`, `$_GET` i `$_COOKIE`, tako da se ova promenljiva koristi za dobijanje podataka prosleđenih i uz pomoć `GET` i `POST` metoda.

Često se prilikom korišćenja PHP jezika javlja potreba da se koristi više sličnih promenljivih. Umesto uvođenja različitih naziva za promenljive, što može biti problem naročito ako je njihov broj veliki, koriste se nizovi. Svaki element u nizu ima svoj ID (indeks), tako da mu se može lako pristupiti.

PODELA NIZOVA

Nizovi se dele u tri kategorije: numerički, asocijativni, višedimenzionalni

Nizovi se dele u tri kategorije:

- numerički nizovi sa numeričkim ID oznakama,
- asocijativni nizovi gde je svakom ID ključu pridružena neka vrednost,
- višedimenzionalni nizovi koji sadrže jedan ili više znakova

Promenljivama u numeričkom nizu vrednosti se mogu dodeliti na različite načine..

PRIMER 2

Napisi PHP skript određivanja prisustva studenata na predavanju

(Predviđeno vreme izrade zadatka iznosi 20 minuta.)

Zadatak:

Neka je `$student` niz sastavljen od imena studenata: Petar, Kosta, Jovan i Dragan. Napisati PHP skript koji će prikazati tekst da prvi i treći student nisu na času dok su ostali prisutni.

Rešenje:

Indeksi kod nizova obično počinju od nule, pa prvi student u nizu ima indeks 0, drugi ima indeks 1 itd. PHP skript može da izgleda ovako:

```
<?php  
$student[0] = "Petar";
```

```
$student[1] = "Kosta";
$student[2] = "Jovan";
$student[3] = "Dragan";
echo $student[0] . " i " . $student[2] . " nisu prisutni.<br/>
Ostali studenti: ". $student[1] . " i " . $student[3] . ", su prisutni.";
?>
```

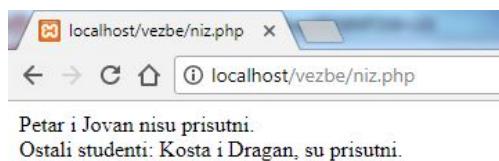
Nakon snimanja kao npr. "primer2.php" u čitaču će se dobiti rezultat dat na slici 3.

U PHP jeziku postoji i mogućnost automatskog dodeljivanja indeksa. Za poslednji primer, kreiranje niza moglo je da se izvede i na sledeći način:

```
$student = array("Petar", "Kosta", "Jovan", "Dragan");
```

Navedeni studenti imali bi isti indeks u nizu **\$student**.

Kada je potrebno uz pomoć niza zapamtiti neke specifične vrednosti za određene promenljive, koristi se asocijativni niz. Kod asocijativnih nizova, vrednosti indeksa definiše korisnik.



Slika 7.2.3 Rezultat skripte [Izvor: Autor]

PRIMER 3

Kreirati niz \$smer koji sadrži imena smerova

(Predviđeno vreme izrade zadatka iznosi 20 minuta.)

Zadatak:

Kreirati asocijativni niz \$smer koji sadrži nazive smerova koje su upisali studenti, gde se kao ID pojavljuje ime studenta.

Rešenje:

U ovom slučaju, niz se koristi da se različitim studentima dodele odgovarajući smerovi na sledeći način:

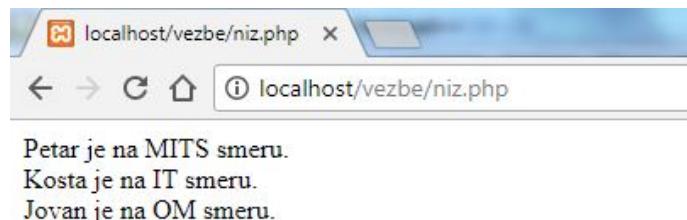
```
<?php
$smer['Petar'] = "MITS";
$smer['Kosta'] = "IT";
$smer['Jovan'] = "OM";
echo "Petar je na " . $smer['Petar'] . " smeru.<br/>";
echo "Kosta je na " . $smer['Kosta'] . " smeru.<br/>";
echo "Jovan je na " . $smer['Jovan'] . " smeru.";
```

Ako skript snimimo kao "primer3.php", prikaz u čitaču izgleda kao na slici 4.

Slično kao i u drugom primeru, ovaj asocijativni niz se može kreirati na sledeći način:

```
$smer = array("Petar"=>MITS, "Kosta"=>IT, "Jovan"=>OM);
```

U nastavku će biti prikazano još nekoliko primera korišćenja PHP jezika.



Slika 7.2.4 Rezultat prikazan u veb čitaču [Izvor: Autor]

PRIMER 4

Kreirati PHP skript koji ispisuje datum u crvenoj boji

(Predviđeno vreme izrade zadatka iznosi 15 minuta.)

Zadatak:

Kreirati PHP skript koji će u čitaču prikazati trenutni datum isписан crvenom bojom, veličine 24.

Rešenje:

Funkcija date() omogućava preuzimanje trenutnog vremena na serveru i njegovo formatiranje u skladu sa potrebama korisnika. U zavisnosti od parametara funkcije, način prikazivanja datuma može biti različit.

Parametri mogu biti sledeći: d - dan (ima vrednost od 01 do 31), m - mesec (ima vrednost od 01 do 12), Y - tekuća godina.

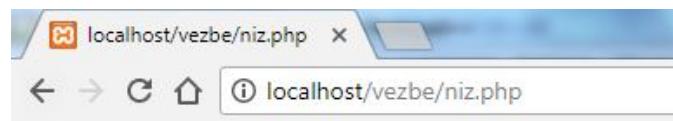
Osim toga, kao parametri mogu se pojaviti i sledeći karakteri: "/", "." ili "-", koji služe za razdvajanje vrednosti za dan, mesec i godinu, tako da se funkcija može koristiti na jedan od sledećih načina:

- date("Y/m/d"),
- date("Y.m.d"),
- date("Y-m-d")

Osim toga, korisnik može da definiše način prikazivanja datuma uz pomoć datih parametara, pa ako je npr. potrebno da datum bude isписан u formatu **dan/mesec/godina**, funkciju **date()** treba da koristi na sledeći način: **date("d/m/y")**.

U ovom slučaju HTML dokument i odgovarajući skript mogu da izgledaju ovako:

```
<html>
  <head>
  </head>
  <body>
    <font face="Arial" color="#FF0000" size="24"><strong>
      <?php print(date("d/m/y")); ?>
    </strong></font>
  </body>
</html>
```



05/12/17

Slika 7.2.5 Prikaz datuma u veb čitaču [Izvor: Autor]

PRIMER 5

Kreirati PHP skript za generisanje slučajnog broja

(Predviđeno vreme izrade zadatka iznosi 25 minuta.)

Zadatak:

Kreirati PHP skript za generisanje slučajnog broja u domenu od 1 do 5, promenu boje pozadine u zavisnosti od dobijenog broja. Ako je dobijeni broj jednak 1, boja pozadine treba da bude plava, za broj 2 zelena, broj 3 crvena, za broj 4 žuta i za dobijeni broj 5 bela.

Rešenje:

Za generisanje slučajnog broja koristi se **rand()** funkcija u sledećem obliku: **rand(min,max)**;

gde su **min** i **max**, odgovarajuća minimalna i maksimalna vrednost koju funkcija **rand()** može da generiše. U ovom slučaju koristićemo: **rand(1,5)**. Ovaj broj će se generisati pri svakom učitavanju ili ažuriranju strane i taj broj dodeljujemo promenljivoj **\$broj**.

Za boju pozadine uvedena je promenljiva **\$pozadina**, čija je vrednost odgovarajući heksadecimalni broj koji zavisi od vrednosti promenljive **\$broj**. Tražene heksa-decimalne vrednosti su: #0000FF za plavu boju, #008000 za zelenu, #FF0000 za crvenu, #FFFF00 za žutu, i #FFFFFF za belu.

Za dodeljivanje odgovarajuće vrednosti promenljivoj **\$pozadina** koristićemo **if...else** i **elseif** naredbe, o kojima je već bilo reči.

Da bi dobijena vrednost mogla da se koristi kao vrednost atributa bgcolor u HTML dokumentu, koristi se **print** naredba koja HTML deo izdvaja od PHP skripta.

Skript može da izgleda ovako:

```
<html>
    <head>
    </head>
    <body>
        <?php
            $broj = rand(1,5);
            print("Slučajni broj je: $broj");
            if($broj == 1)
            {
                $pozadina = "#0000FF";
            }
            else if($broj == 2)
            {
                $pozadina = "#008000";
            }
            else if($broj == "3")
            {
                $pozadina = "#FF0000";
            }
            else if($broj == "4")
            {
                $pozadina = "#FFFF00";
            }
            else
            {
                $pozadina = "#FFFFFF";
            }
            print("<body bgcolor=\"$pozadina\">\n");
        ?>
    </body>
</html>
```

▼ 7.2 Zadaci za samostalnu vežbu

OPIS ZADATAKA ZA SAMOSTALNU VEŽBU

Primena PHP i HTML

(Predviđeno vreme izrade zadatka iznosi 60 minuta.)

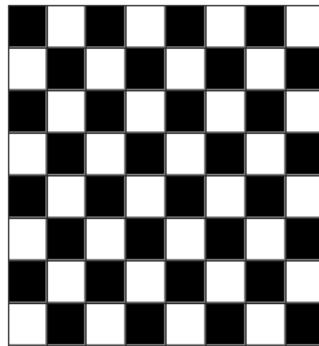
- 1.) Koristeći PHP sabrati prvih 100 parnih brojeva i ištampati.
- 2.) Koristeći PHP ispitati dvostruku vrednost prirodnih brojeva od 8 do 16.
- 3.) Koristeći HTML kreirati drop down listu sa spiskom automobila. Nakon odabira PHP skripta obrađuje ulaz iz forme i ispisuje na HTML stranu.
- 4.) Ispisati današnji datum u zelenoj boji.

5.) Generisati random broj od 1 do 1000. Ako je broj manji od 500 i paran, ispisati tekst "Broj je paran i manji od 500". Ukoliko je veći od 500 i neparan, ispisati tekst "Broj je neparan i veći od 500".

6.) Koristeći PHP napisati funkciju koja će sortirati niz.

7.) Koristeći PHP petlje ispisati zvezdicama prvo slovo vašeg imena.

8.) Koristeći PHP i ugnježdene petlje kreirati šahovsku tablu pomoću HTML table elementa kao na slici ispod. Širinu table postaviti na 270px, a širinu i visinu celija table na 30px.



Slika 7.3.1 Izgled šahovske table [Izvor: Autor]

▼ 7.3 DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ08

Kreirati PHP skriptu.

(Predviđeno vreme izrade domaćeg zadatka je 60 minuta.)

1. Kreirati HTML stranu sa:

- Formom za unos podataka po izboru. Forma treba da sadrži bar jedan textbox, radio box, check box i select box.
- Napisati PHP skriptu koja će obraditi ulaz sa forme i ispisati podatke na HTML stranu.
- Kao rešenje zadatka dostaviti PHP/HTML fajlove.

2. Kreirati HTML stranu sa formom i poljem za unos broja. Napisati PHP skriptu koja će ispisati zbir svih neparnih brojeva od 1 do unetog broja.

Sve datoteke arhivirati u .ZIP fajlu. Naziv arhiviranog fajla treba da bude **IT210-DZ08-ime_prezime_brojIndeksa.zip**, gde su ime, prezime i broj indeksa vaši podaci. Arhiviran fajl poslati predmetnom asistentu na e-mail.

(napomena: u Subject-u e-mejla napisati IT210 – DZ08).

▼ Zaključak

ZAKLJUČAK

Cilj ovog predavanja je bio da opiše značaj korišćemja design patterns-a. Opisani su ukratko sledeće design pattern-i: MVC, singleton, factory method, facade. Identifikovani su najvažniji skript jezici za pisanje skripti sa serverske i klijentske strane. Takođe, navedeni su problemi koji se javljaju prilikom izrade novog ili adaptiranja postojećeg softvera.

Literatura

1. Rasmus Lerdorf, Kevin Tatroe, Programming PHP, O'reilly, 2002.
2. Brandon Goldfedder, The Joy of Patterns: Using Patterns for Enterprise Development, Addison Wesley, 2001.
3. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns CD, Addison Wesley, 1998.
4. Christopher Alexander, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, and Shlomo Angel. A Pattern Language. Oxford University Press, New York, 1977.
5. Hans Bergsten, JavaServer Pages, 3rd Edition, O'Reilly, 2003.



IT210 - SISTEMI IT

INTERAKCIJA IZMEĐU ČOVEKA I RAČUNARA

Lekcija 09

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 09

INTERAKCIJA IZMEĐU ČOVEKA I RAČUNARA

- ✓ INTERAKCIJA IZMEĐU ČOVEKA I RAČUNARA
- ✓ Poglavlje 1: Ljudski faktor – Kognitivni principi
- ✓ Poglavlje 2: Razumevanje korisnika
- ✓ Poglavlje 3: Ergonomija
- ✓ Poglavlje 4: Aspekti interakcije čovek-računar aplikacionog domena
- ✓ Poglavlje 5: Pristupi
- ✓ Poglavlje 6: Pokazna vežba: PhpMyAdmin
- ✓ Poglavlje 7: Zadaci za samostalnu rad: PHP, HTML i SQL
- ✓ Poglavlje 8: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Cilj ove lekcije je da se obrađe teme: kognitivni pristupi, razumevanje korisnika, projektovanje za čoveka, ergonomija, tipovi okruženja, kognitivni modeli

U ovoj lekciji biće obrađene sledeće teme:

- Kognitivni principi
- Razumevanje korisnika
- Projektovanje za čoveka
- Ergonomija
- Tipovi okruženja
- Kognitivni modeli

▼ Poglavlje 1

Ljudski faktor – Kognitivni principi

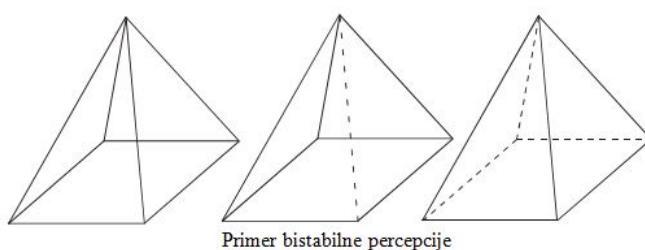
KOGNITIVNI PRINCIPI - PERCEPCIJA

Sa aspekta kognitivnih nauka percepcija je proces prikupljanja, interpretiranja, selektovanja i organizacije senzorskih informacija.

Sa aspekta kognitivnih nauka **percepcija** je proces prikupljanja, interpretiranja, selektovanja i organizacije senzorskih informacija. Za prikupljanje informacija se koriste **čula**. Prilikom percepcije se okolina skenira istovremeno različitim čulima i na osnovu informacija koje se dobijaju od pojedinih čula se u mozgu stvara virtualna slika okoline.

Kognitivni psiholozi tvrde da tokom svog razvoja čovek stvara model funkcionisanja okoline i sveta. Čovek oseća objektivni svet koga mapira u percepcije koje su privremene i mogu da se menjaju tokom vremena. Drugim rečima, **ista osoba može da isti objektivni svet doživi drugačije u različitim vremenskim trenucima, zato što drugačije interpretira, selektuje i organizuje informacije**. Kako čovek prikuplja nove informacije, njegova percepcija se pomera. S druge strane, različite osobe mogu isti realni objekat da dožive različito jer im je percepcija različita.

Na slici 1 je prikazan žičani model piramide. Ako se postavi pitanje koja je od stranica najbliža gledaocu, različite osobe će dati različite odgovore, zavisno od njihove percepcije.



Slika 1.1 Primer bistabilne percepcije [Izvor: Autor]

Kao što jedan objekat može da stvori više različitih percepcija, tako neki objekti ne dovode ni do kakve percepcije kod nekih osoba, jer u iskustvu te osobe nema nikakve osnove za tu percepciju. Ova zbumujuća dvosmislenost percepcije se naziva kamuflaža ili mimikrija.

ČULA

Čula su sistemi preko kojih se pobude iz spoljnog sveta primaju, pa tako predstavljaju osnovni instrument za percepciju

Čula su sistemi preko kojih se pobude iz spoljnog sveta primaju, pa tako predstavljaju osnovni instrument za percepciju. Svako čulo ima neki tip senzora koji može da prihvati pobude i odgovarajući region, ili grupu regiona, u mozgu gde se primljeni signali primaju i interpretiraju.

Čovek poseduje najmanje devet čula, a kod drugih organizama se susreću još dva (magnetocepacija i elektrocepacija). Neki naučnici tvrde da, pored devet nespornih čula, čovek raspolaze i sa drugim čulima. Tako po nekim naučnicima čovek ima čak 21 čulo.

Vid je sposobnost detektovanja vidljivih elektromagnetnih talasa, poznatih kao svetlost, pomoću očiju i mozga koji ih interpretira u sliku.

Sluh je čulo percepcije zvuka koje kao senzore koristi membranu u uhu. Ova membrana vibrira pri promeni vazdušnog pritiska na nju koju izazivaju zvučni talasi. Oblast frekvencija talasa koju ljudski sluh može da detektuje je individualna i kreće se od 9 do 20.000 Hz.

Ukus je čulo koje stvara različite osećaje prilikom dodira materije i jezika. Postoji saglasnost da postoji najmanje 4 tipa receptora ukusa na jeziku, pa neki naučnici tvrde da su to u stvari 4 različita čula, što potvrđuju i činjenicom da se i informacije beleže u sasvim različitim delovima mozga. Receptori na jeziku mogu da detektuju sledeće ukuse: slatko, slano, kiselo i gorko. Postoji i peti, skoro otkriveni, receptor za osećaj koji se naziva umami, koji reaguje na glutaminsku amino kiselinsku koju se uobičajeno sreće u mesu.

Miris, kao i ukus spadaju u klasu hemijskih čula. U nosu postoje na stotine olfaktivnih (mirisnih) receptora koji reaguju na posebne molekularne osobine.

SOMATSKA ILI TELESNA ČULA

Grupu somatskih ili telesnih čula čine dodir, termocepcija i nocicepcija.

Grupu somatskih ili telesnih čula čine dodir, termocepcija i nocicepcija.

Dodir ili taktilni osećaj je čulo percepcije pritiska na površinu kože. Čulo dodira koristi taktilne senzore koji mogu da razlikuju promenu pritiska na koži i tako daju osećaj čvrstog, mekog, glatkog, hrapavog itd.

Termocepcija je čulo pomoću koga se oseća toplota odnosno hladnoća. Termocepcijiski receptori se nalaze u koži. Smatra se da postoji i druga vrsta termo receptora, homeostatički termoceptori, koji se nalaze u telu i koji reaguju na unutrašnju temperaturu tela.

Nocicepcija je čulo pomoću koga se oseća bol. Po nekim autorima radi se o tri čula koja reaguju na bol na koži, koskama i organima.

Ekvilibriocepacija je percepcija ravnoteže. Receptori ovog čula se nalaze u otvorima sa fluidom u srednjem uhu.

Propriocepacija je čulo koje ljudima pomaže da shvate gde im se nalaze neki delovi tela u nekom trenutku vremena. Ljudi neprestano koriste ovo čulo iako su nesvesni njega. Na primer, čovek u mraku zna koliko da podigne nogu kada ide stepenicama, ili može da dodirne vrh nosa i kada zažmuri.

Na žalost, čovek ne može da iskoristi sva ova čula u interakciji sa računarskim sistemima. Izlazni uređaji računara za sada su na niskom nivou. Tipičan korisnik je u komunikaciji sa računarom ograničen na čula vida i sluha. Slepote koriste čulo dodira za rad sa dinamičkim Brajovim displejima. Za doživljavanje virtualnog sveta svakako da bi bilo interesantno ostvariti izlazne uređaje koji bi mogli da komuniciraju sa korisnicima i preko čula mirisa, ukusa i termocepcije.

KOGNITIVNI PRINCIPI - MEMORIJA

Memorisanje ili pamćenje je sposobnost mozga da smesti, zapamti i kasnije pronađe informacije

Pamćenje je psihička funkcija koja nam omogućava da informacije koje smo ranije primili, osećanja koja smo ranije imali ili nešto što smo ranije zamišljali, zapamtimo i kasnije ponovo oživimo i dovedemo u svest. Memorisanje ili pamćenje je sposobnost mozga da smesti, zapamti i kasnije pronađe informacije. Pitanjima pamćenja se bavi grana nauke koja se naziva kognitivna neurologija. Sa aspekta procesiranja informacija postoje **tri faze u korišćenju memorije**:

- **Kodiranje** - obrada i kombinovanje primljenih informacija
- **Pamćenje** - kreiranje permanentnih zapisa kodiranih informacija
- **Nalaženje (sećanje)** - pronalaženje memorisanih informacija kao odziv na neko pitanje u nekom procesu

Rad memorije zavisi od mnogo faktora. Kodiranje informacija zavisi od percepcije i koncentracije osobe u trenutku kodiranja. Pamćenje zavisi u velikoj meri od karakteristika mozga. Proces nalaženja informacija ne mora uvek da bude uspešan iako su informacije upamćene. U nekim slučajevima, osoba se seti odgovora na neko pitanje na koje prethodno nije mogla da da odgovor.

Zavisno od dužine trajanja, mogu se razlikovati **tri tipa pamćenja**:

- **Senzorsko pamćenje**. Ovo pamćenje najkraće traje, od nekoliko milisekunda do sekunde, i odgovara inicijalnom momentu kada se informacije primaju preko čula. Neke od ovih informacija se predaju području za kratkotrajno pamćenje. Čula primaju stalno ogromne količine informacija. Koje će informacije biti prosleđene iz senzorske memorije u kratkoročnu zavisi od naše pažnje i interesovanja. Na primer, ljudske oči primaju oko 12 slika u sekundi, što znači da će tokom budnog dela dana koji traje približno 16 sati čovek primiti 691.200 slika. Samo neke od ovih slika će biti upamćene.
- **Kratkotrajno pamćenje** čuva informacije od jedne sekunde do jednog minuta. Ova memorija je poznata i kao radna memorija. U njoj se nalaze podaci koji trenutno okupiraju našu pažnju.
- **Dugotrajno pamćenje** se odnosi na period od više godina. Podaci koji se nalaze u ovoj memoriji mogu, ali ne obavezno i ne uvek, da budu dostupni korisniku do kraja života.

DUGOTRAJNO PAMĆENJE – DEKLARATIVNO I PROCEDULARNO

Zavisno od vrste informacija koje se čuvaju dugotrajno pamćenje može biti podeljeno na deklarativno (ili eksplisitno) i proceduralno (ili implicitno) pamćenje.

Da bi trajno upamatio informacije, čovek mora da prosledi informacije iz senzorskog pamćenja kratkotrajnom, a onda iz kratkotrajnog dugotrajnom pamćenju. Informacije koje dopru do memorije za dugotrajno pamćenje ostaju trajno sačuvane. U suprotnom, ako neke informacije ne dopru do dugotrajnog pamćenja, čovek neće moći da ih se seti posle kratkog vremenskog perioda.

Zavisno od vrste informacija koje se čuvaju dugotrajno pamćenje može biti podeljeno na **deklarativno** (ili **eksplisitno**) i **proceduralno** (ili **implicitno**) pamćenje. **Deklarativno pamćenje** zahteva svestan zahtev za nalaženjem neke memorisane informacije. Deklarativno pamćenje se dalje može podeliti na:

- **semanticko pamćenje** - pamte se činjenice nezavisno od konteksta,
- **pamćenje epizoda** - pamte se informacije vezane za neki kontekst, kao što su vreme i mesto,
- **autobiografsko pamćenje** - pamte se događaji iz života. Ovo je slično pamćenju epizoda.
- **Vizuelno pamćenje** - pamte se slike ljudi, događaja, mesta, životinja,

Proceduralno pamćenje ne zahteva eksplisitno traženje da bi se došlo do neke informacije. Na primer, čovek ne treba da se podseti kako se hoda. Proceduralno pamćenje uglavnom služi za čuvanje informacija vezanih za motorne veštine.

KOGNITIVNI PRINCIPI – REŠAVANJE PROBLEMA

Rešavanje problema se može definisati kao sistematsko traženje u oblasti mogućih akcija u svrhu postizanja nekog cilja ili rešenja

Rešavanje problema je kognitivni proces vrlo visokog nivoa pri kome se kombinuju osnovne kognitivne veštine. Smatra se da je rešavanje problema najsloženija intelektualna funkcija.

Rešavanje problema se može definisati kao sistematsko traženje u oblasti mogućih akcija u svrhu postizanja nekog cilja ili rešenja.

Problemi imaju sledeće tipične karakteristike:

- **Problemi su netransparentni**, tj. nije jasno kako postaviti problem, šta su ulazi a šta izlazi
- **Problemi imaju višestruke ciljeve** koje je teško izraziti i koji su nekada u suprotnosti
- **Problemi znaju da budu previše kompleksni** zbog neprebrojivosti, povezanosti i heterogenosti

- **Problemi nekada imaju dinamiku**, tj. tokom vremena se menjaju ograničenja, karakteristike i dolazi do nepredvidljivih pojava.

Za rešavanje problema se koriste različite tehnike. Izbor adekvatne tehnike je takođe poseban problem. U nekim slučajevima se koristi više metoda da bi se našlo rešenje jednog problema.

Generalno metode za rešavanje problema se mogu podeliti na opšte i specijalizovane. **Opšte metode rešavanja problema** se mogu koristiti za rešavanje klase sličnih problema. **Specijalizovane metode** služe za rešavanje jednog određenog problema i pri tom koriste specifičnosti problema da bi se došlo do rešenja.

Neke od poznatijih opštih metoda za rešavanje problema su:

- Nalaženje **analogije** sa sličnim problemom za koji postoji rešenje.
- **Pokušaj i greška** (engl. **trial and error**) je metod koji se često primenjuje kada nema bolje metode. Po ovoj metodi se predloži neko rešenje i onda se ispituje da li je problem rešen. Ako nije, nastavlja se sa drugim pokušajima. Primer primene ove metode je slaganje puzla.
- **Istraživanje** literaturnih izvora na temu datog problema jer je možda već nađeno rešenje za dati ili sličan problem.
- Izrada modela koji odgovara problemu i traženje rešenja unutar **modela**.
- Organizovanje sesija **brainstorming-a**
- Primena **logike i kombinatorike**.

▼ Poglavlje 2

Razumevanje korisnika

ZBOG ČEGA PROJEKTANT INTERFEJSA TREBA DA RAZUME KORISNIKA?

Projektant interfejsa mora da razume da tipičan korisnik ima ograničene mogućnosti percepcije, pa izlaze iz računarskog sistema treba da prilagodi ovim ograničenjima.

Projektovanje dobrog interfejsa čovek-mašina polazi od potreba korisnika, njegovih želja, mogućnosti i ograničenja. Sa druge strane projektanti interfejsa su ograničeni tehničkim mogućnostima ulaznoizlaznih uređaja i programskih alata koji im stoje na raspolaganju za implementaciju interfejsa.

Kognitivne teorije percepcije prepostavljaju da postoji siromaštvo pobuda (stimulusa) koje čovek može da dobije od računarskog sistema i radne okoline. Čula, sama po sebi, nisu u mogućnosti da stvore jedinstvenu sliku sveta, odnosno u ovom slučaju sliku o načinu funkcionisanja računarskog sistema, već zahtevaju obogaćivanje pomoći kognitivnog modela koga stvara korisnik.

Polazeći od ovih premissa, jasno je da projektant interfejsa treba da potpuno razume korisnika kako bi izgradio dobar interfejs. **Projektant mora da razume da tipičan korisnik ima ograničene mogućnosti percepcije, pa izlaze iz računarskog sistema treba da prilagodi ovim ograničenjima.**

Mogućnost pamćenja čoveka je ograničena. Njegova pažnja se ne može držati tokom dugog vremenskog perioda. Interfejs treba prilagoditi ovoj činjenici. Na kraju, čovek ima ograničene sposobnosti za rešavanje problema. Isuviše složen interfejs, koji nije prilagođen mogućnostima korisnika, biće neupotrebljiv.

Kada projektant ima jasnu sliku o korisniku u prilici je da stvori takav sistem koji će pomoći korisniku da stvori dobar kognitivni model računarskog sistema, što će mu omogućiti njegovo efikasno korišćenje.

✓ 2.1 Projektovanje za čoveka

POTREBA ZA PROJEKTOVANJEM ZA ČOVEKA

Potrebno je: razumeti korisničke aktuelne potrebe , razumeti korisničke „bolne tačke“, identifikovati IT trendove

Još 1984. godine Niels Bjorn-Andersen je u svom čuvenom radu ‘Are “Human Factors” Human?’ skrenuo pažnju IT stručnjacima na potrebu da se računarske tehnologije prilagode korisnicima. On je postavio jedno vrlo važno pitanje: „ Da li činimo sve što je moguće da prilagodimo tehnologiju poznatim „ljudskim nedostacima“, da smanjimo otpor ka primeni tehnologija i obezbedimo tehnologije koje će biti instrument u oslobođanju intelektualnih sposobnosti ljudskog bića“.

U istom radu Niels Bjorn-Andersen predlaže da kreiranje novih tehnologija postane demokratski proces u kome će sami korisnici projektovati radne sisteme uz podršku eksperata kao konsultanata. Ovaj rad se smatra kamenom temeljcem discipline informacionih tehnologija koja se bavi projektovanjem računarskih sistema za čoveka. Od tada su projektanti računarskih sistema počeli da posvećuju pažnju mogućnostima i potrebama korisnika. Više nije primarni cilj novog rešenja bilo samo bolje tehničko rešenje, nego veće zadovoljstvo korisnika.

Kompanija Intel u svojim razvojnim projektima potencira na premisi „projektovanje za čoveka“. Da bi se ovo realizovalo potrebno je razumeti zahteve i želje krajnjih korisnika, što se može postići samo stavljanjem korisnika u kontekst njegovog okruženja. Poslovna politika kompanije nalaže da se tokom projektovanja saznaju sve potrebe, želje, ograničenja i problemi korisnika, a između ostalog je potrebno i:

1. **Razumeti korisničke aktuelne potrebe** kao što su pristup informacijama, korišćenje najvažnijih informacija i aplikacija u bilo koje vreme i na bilo kom mestu, potreba da se stalno bude na mreži,

2. **Razumeti korisničke „bolne tačke“** kao što su blokiranje sistema, virusi, spam, inoviranje • softvera, problemi sa korisničkim interfejsom itd.,

3. **Identifikovati IT trendove koje su okrenute krajnjim korisnicima**, a koje korisnici očekuju od današnjih proizvoda, kao na primer:

- Zabava pri kretanju
- Povećana produktivnost na mobilnim uređajima -
- Tolerantni korisnički interfejsi

✓ Poglavlje 3

Ergonomija

ČIME SE BAVI ERGONOMIJA?

Ergonomija se bavi razumevanjem međudejstva između čoveka i drugih elemenata sistema i profesija koja primenjuje teoriju, principe, podatke i metode za projektovanje

Prema definiciji Međunarodnog ergonomskog udruženja (International Ergonomic Association), ergonomija (ili ljudski faktor) je naučna disciplina koja se bavi razumevanjem međudejstva između čoveka i drugih elemenata sistema i profesija koja primenjuje teoriju, principe, podatke i metode za projektovanje koje će optimizovati ljudsko blagostanje i ukupne sistemske performanse

Ergonomisti utiču na projektovanje i evaluaciju zadataka, poslova, proizvoda, okruženja i sistema da bi ih učinili kompatibilnim sa potrebama, mogućnostima i ograničenjima ljudi.

Može se reći da, što se računarstva tiče, postoje tri važna aspekta ergonomije:

- Fizička ergonomija
- Kognitivna ergonomija i
- Organizaciona ergonomija

ERGONOMIJA I DIZAJN - VIDEO

"A great video that explains the need to consider ergonomics when designing for comfortable human use."

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ 3.1 Fizička ergonomija

ŠTA JE FIZIČKA ERGONOMIJA?

Fizička ergonomija se bavi odzivom ljudskog tela na fizičko i fiziološko opterećenje koje radnici doživaljavaju na radnom mestu.

Fizička ergonomija se bavi odzivom ljudskog tela na fizičko i fiziološko opterećenje koje radnici doživaljavaju na radnom mestu. Postoje različiti razlozi za pojavu fizičkog i fiziološkog opterećenja, ali se najčešće ubrajaju:

- Rukovanje materijalom i sredstvima rada
- Ponavljanje istih operacija tokom dužeg vremenskog perioda, a ponekad i čitavog radnog dana
- Vibracija okoline ili sredstava za rad
- Položaj tela koji ne odgovara prirodnom stavu
- Prevelika buka ili stalni šum
- Neadekvatno osvetljenje.

Iako se na prvi pogled čini da korisnici računarskih sistema nisu fizički opterećeni prilikom rada, uočeni su mnogi zdravstveni problemi osoba koje većinu radnog vremena provode za računarom. Najčešći problem je pojava osteoartrita, odnosno umanjenja funkcija zglobova.

Procenjuje se da oko 25% stanovništva ima osteoartritist, dok je u krugovima korisnika računara ovaj procenat uvećan. Osteoartritist se javlja kao posledica lošeg položaja tela ili delova tela. Pored toga, većina korisnika računara, posle dugotrajnog rada ima sindrom karpalnog tunela koji se javlja zbog pritiska na medialni nerv u zglobu šake. Ovo je posledica rada sa tastaturom i mišem. Nisu zanemarljive ni posledice koje mogu da se jave zbog elektromagnetne radijacije monitora koji rade sa tehnologijom katodne cevi.

Zbog ovoga je prilikom projektovanja i uređenja radnog mesta i izbora računarske opreme potrebno voditi računa o nizu faktora koji sa aspekta fizičke ergonomije utiču na zdravlje. Ovde se navode neki važniji faktori.

Radno mesto treba opremiti nameštajem koji se može prilagoditi korisniku. Radni sto treba da ima pravilnu visinu tako da ne zahteva savijanje ramena korisnika. Dubina stola treba da je dovoljna da se monitor postavi na pravilnu udaljenost. Stolica treba obavezno da ima podešavanje visine, kako bi se usaglasila sa stolom i visinom korisnika.

IZBOR RAČUNARSKE OPREME SA ASPEKTA FIZIČKE ERGONOMIJE

Tastatura je najkorišćeniji deo ulaznog dela korisničkog interfejsa računara, nakon tastature najupotrebljavaniji je miš.

Vidljivost na radnom mestu treba da bude optimalna. **Dobro osvetljenje** radnog mesta se najbolje postiže prirodnom svetlošću. Prozorska svetlost treba da pada na monitor korisnika sa bočne strane. Na taj način se izbegava pojavljivanje odbljeska na displeju. **Pravilna visina monitora** smanjuje opterećenje vrata pa je potrebno prilagoditi vertikalnu visinu monitora i podešiti njegov ugao tako da bude upravan na pogled korisnika pri normalnom držanju glave.

Tastatura je najkorišćeniji deo ulaznog dela korisničkog interfejsa računara, pa joj treba posvetiti posebnu pažnju. **Visina tastature** treba da bude pravilna, ali je isto tako važan nagib tastature. Nažalost, većina tastatura ima mogućnost podizanja zadnjeg dela, iako je pravilan nagib tastature takav da deo gde je razmaknica bude na višem nivou od dela sa funkcionalnim tasterima. Levorukim osobama treba obezbediti tastaturu za levoruke.



Slika 3.1.1 Prikaz tastature [Izvor: Autor]

Nakon tastature, najupotrebljavани deo interfejsa je **miš**. Prilikom izbora miša treba voditi računa da **veličina miša** treba da odgovara veličini dlana, a da njegov oblik bude anatomski. Postoje miševi i za levoruke. Prilikom **korišćenja miša**, ne treba ga previše stezati i pritiskati tastere snažno. **Prevelika rezolucija ekrana** zahteva precizna pomeranja miša što izaziva zamor.



Slika 3.1.2 Prikaz miša Prikaz tastature [Izvor: Autor]

Prilikom **izbora monitora** treba voditi računa o njegovom tipu, veličini i mogućnostima podešavanja visine i nagiba. TFT monitori daju mnogo stabilniju sliku od CRT monitora. Pored toga CRT monitori imaju određenu dozu zračenja. Ona nije velika i brzo opada sa udaljenošću od monitora. Zbog toga se preporučuje da udaljenost CRT monitora od tela bude minimalno 75 cm, a od okolnih monitora u prostoriji 1 m. Veći monitori omogućuju uvećani prikaz dokumenata ili više vidljivih prozora što smanjuje potrebu za korišćenjem miša, a time i zamor. Previše **bučni ventilatori** za hlađenje komponenata u kućištu izazivaju zamor, pa treba birati "tihe" računare.

▼ 3.2 Kognitivna Ergonomija

ŠTA JE KOGNITIVNA ERGONOMIJA?

Kognitivna ergonomija se bavi analizom kognitivnih procesa tokom radnog vremena i poboljšanjem uslova za njihovo izvršavanje

Kognitivna ergonomija se odnosi se na mentalne procese prilikom obavljanja poslova. Pored fizičkog, korišćenje računarskih sistema zahteva i intenzivnu mentalnu aktivnost pa dolazi do zamora i poremećaja. U takvim uslovima ljudi će češće praviti greške što može da ugrozi poslovne procese.

Mentalno opterećenje može biti izazvano:

- Zbog stalne odgovornosti za kvalitet i brzinu izvršavanja radnih zadataka
- Potrebom da se donesu odluke za vrlo kratko vreme ili u realnom vremenu
- Potrebom da se stalno bude oprezan i da se prati pojava nepredvidljivih događaja
- Potrebom da se za duže vreme održe performanse neke veštine
- Strahom da ne dođe do ljudske greške u nekom procesu
- Lošim Interfejsom čovek-računar.

Kognitivna ergonomija se bavi analizom kognitivnih procesa tokom radnog vremena i poboljšanjem uslova za njihovo izvršavanje korišćenjem metoda kao što su:

- Uspostavljanje korisnički-centričnog interfejsa za interakciju između čoveka i računara,
- Projektovanje informacionog sistema koji podržava kognitivne zadatke, kao što su na primer sistemi poslovne inteligencije,
- Razvoj programa za edukaciju i trening,
- Reinženjering radnih procesa kako bi moglo da se upravlja kognitivno opterećenje i poveća ljudska pouzdanost.

Uspešne ergonomiske intervencije u oblasti kognitivnih zadataka zahtevaju duboko razumevanje ne samo razumevanja radnih zadataka, nego i strategija koje koriste korisnici u vršenju kognitivnih zadataka, kao i ograničenja ljudske kognicije. Sredstva i alati koji se koriste za vršenje zadataka mogu da unesu svoja ograničenja. Zbog toga analiza kognitivnih zadataka treba da obuhvati kako interakciju korisnika sa svojim zadacima, tako i interakciju sa softverskim alatima i opremom. Rezultati analize treba da pomognu u projektovanju takvog interfejsa čovek-mašina da ljudske performanse budu očuvane u uslovima kada su informacije nepouzdane, događaje je teško predvideti postoje konfliktni ciljevi ili kada je ograničeno vreme za donošenje odluka.

✓ 3.3 Organizaciona ergonomija

ŠTA JE ORGANIZACIONA ERGONOMIJA?

Organizaciona ergonomija se odnosi na optimizaciju socio-tehničkog sistema

Organizaciona ergonomija se odnosi na optimizaciju socio-tehničkog sistema koji obuhvata:

- Organizacionu strukturu organizacije
- Pravila rada u organizaciji
- Procese u organizaciji
- Problemi koji se rešavaju organizacionom ergonomijom utiču kako na fizičko, tako i na mentalno opterećenje osoba.

Tipični problemi su:

- Rad u smenama ne odgovara svim osobama, a radna sposobnost osoba u različitim periodima dana je individualna

- Dnevni raspored rada može da bude neuravnotežen u smislu da je zbog prirode procesa organizacije većinu poslova potrebno uraditi u nekom delu radnog vremena
- Radnici nisu zadovoljni svojim poslom
- Radnici nisu motivisani za rad
- Radnici su opterećeni stalnim nadzorom njihovih nadređenih
- Radnici nisu spremni za timski rad
- U organizaciji su narušeni osnovni etički principi.

Stručnjaci za organizacionu ergonomiju analiziraju probleme, organizaciju i procese i menjaju ih u cilju optimiziranja socio-tehničkog sistema.

ERGONOMIJA - VIDEO

Ergonomics At Work Place | Types of Ergonomics | Benefits | Ergonomics Evaluation Techniques

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 4

Aspekti interakcije čovek-računar aplikacionog domena

TIPOVI KORISNIČKIH INTERFEJSA

Sa aspekta interakcije čovek računar, današnji računarski sistemi nude tri tipa okruženja: karakter-bazirani interfejs, GUI i veb interfejs

Prema korišćenoj tehnološkoj bazi i paradigmi interakcije koju su koristili, korisničke interfejse možemo podeliti u sledeće grupe:

- **Hardverski korisnički interfejs**
- **Karakter-bazirani interfejs**
- **Grafički korisnički interfejs**
- **Percepcijski korisnički interfejsi**
- **Korisnički interfejsi zasnovani na pažnji**
- **Elektro-fiziološki korisnički interfejsi**
- **Osetni korisnički interfejsi**
- **Organski korisnički interfejs**
- **Veb interfejsi.**

OPIS TIPOVA KORISNIČKIH INTERFEJSA

Grafički korisnički interfejsi su zasnovani na WIMP (windows, icons, menus, pointer) paradigmii interakcije

Hardverski korisnički interfejsi su vezani za početke komunikacije čoveka i računara koje karakteriše ručno povezivanje hardverskih komponenti, upotreba dugmadi, prekidača, signalnih lampica i bušenih kartica.

Karakter-bazirani interfejs je bio preteča grafičkom korisničkom interfejsu i danas se sreće samo u nekim starim aplikacijama.

Grafički korisnički interfejsi su zasnovani na WIMP (engl. windows, icons, menus, pointer) paradigmii interakcije

Percepcijski korisnički interfejsi integrišu perceptivne korisničke interfejse (engl. *perceptive user interfaces*), višenačinske korisničke interfejse (engl. *multimodal user interfaces*) i multimedijalne korisničke interfejse (engl. *multimedia user interfaces*). Perceptivni korisnički interfejsi imaju sposobnost opažanja korisnika i njegovih aktivnosti.

Multimedijalni korisnički interfejsi prvenstveno prenose informacije od računara ka korisniku. Višenačinska komunikacija je obostrana.

Korisnički interfejsi zasnovani na pažnji (engl. *attentive user interfaces* - AUI) kao osnovni cilj imaju prepoznavanje prostora korisnikovih namera u cilju optimizacije resursa za obradu informacija korisnika i računara. Ovo postiže merenjem i modelovanjem usmerenosti korisnikove pažnje ka zadatku, uređajima i drugim ljudima.

Elektro-fiziološki korisnički interfejsi kombinuju tehnologije za opažanje fizioloških signala iz medicinskih aplikacija sa raznim tehnikama izrade interaktivnih računarskih sistema. Predstavljaju platformu za mnoge aplikacije kao što su interfejsi između mozga i računara (engl. *brain-computer interfaces* - BCI), proteze, i različite oblike interakcije bez korišćenja ruku.

Osetni korisnički interfejsi (engl. *tangible user interfaces* - TUI) su vezani za disciplinu interakcije čoveka i računara zasnovanu na korišćenju metafora koje oblikuju digitalne informacije u fizičke forme iz realnog sveta (engl. *tangible interaction*). Cilj je postizanje integracije fizičkog i digitalnog sveta kako bi se unapredila sama komunikacija čoveka i računara. Ovo se postiže angažovanjem znanja i veština čoveka u korišćenju svakodnevnih objekata iz realnog sveta poput prostornih, motoričkih i socijalnih veština. Osnovni problemi vezani za razvoj ove vrste korisničkih interfejsa su konceptualne, metodološke i tehničke prirode.

Organski korisnički interfejsi (engl. *organic user interfaces* - OUI) predstavljaju vrstu korisničkih interfejsa koji koriste neplanarne objekte kao primarna sredstva ulaza i izlaza. Ova vrsta interfejsa ne predviđa čisto planarnu strukturu prikaza, kao kod LCD displeja, već displeje koji primaju oblik objekata koji se sreću u realnom svetu.

GRAFIČKI KORISNIČKI INTERFEJS

Grafički korisnički interfejs je skup metoda i mehanizama za interakciju sa sistemom koji se u osnovi bazira na grafički predstavljenim objektima i grafičkim ulaznim uređajima tipa indikatora

Grafički korisnički interfejs je skup metoda i mehanizama za interakciju sa sistemom koji se u osnovi bazira na grafički predstavljenim objektima i grafičkim ulaznim uređajima tipa indikatora. Tipičan ulazni uređaj je miš. Tipični objekti su prozori, ikone, dugmad, meniji. Korisnik vrši interakciju sa objektima tako što bira neki grafički objekat i vrši neku akciju nad njim. Svi objekti imaju poznato ponašanje. Informacije se mogu prezentovati korisniku kao tekst, slika, zvuk ili film.

Svi savremeni operativni sistemi i aplikacije danas koriste grafički korisnički interfejs koji je izgrađen oko nekoliko koncepata koji se opisuju u daljem tekstu.

VIZUELNA PREZENTACIJA

Vizuelna prezentacija je vizuelni aspekt grafičkog interfejsa.

Vizuelna prezentacija je vizuelni aspekt grafičkog interfejsa. Vizuelna prezentacija se postiže prikazom linija, crteža, ikona, fontova različitih stilova i veličina, bojama, fotografijama, animacijama i filmovima. Vizuelna prezentacija se koristi za prikaz informacionih objekata i za prikaz interakcionih elemenata. U interakcione elemente spadaju:

- prozori,
- dijalog boksovi,
- meniji u obliku traka, padajući meniji i pop-up meniji,
- ikone koje predstavljaju programe ili datoteke,
- kontrole kao što su tekst boksovi, liste, trake za skrolovanje, dugmad, itd.
- pokazivači i kurzori

Zadatak vizuelne prezentacije je da vizuelno prikaže realni svet korisnika realistično, jednostavno i jasno dajući pri tom grafičkim elementima razumljivi značaj.

INTERAKCIJA TIPOVIMA IZABERI I KLIKNI

Elementi grafičkog interfejsa nad kojim treba izvršiti neku akciju moraju biti najpre izabrani pokazivačem ili nekom drugom tehnikom

Elementi grafičkog interfejsa nad kojim treba izvršiti neku akciju moraju biti najpre izabrani pokazivačem ili nekom drugom tehnikom. Da bi se izvršila akcija potrebno je pritisnuti taster miša, tj. kliknuti mišem preko izabranog objekta. Primarni mehanizam za izvršavanje ovih akcija je miš, ali se kod većine sistema ove operacije mogu izvršiti i tastaturom.

OGRANIČENI SET OPCIJA INTERFEJSA

Set alternativnih akcija koje korisnik u nekom trenutku može da izvede je definisan onim što vidi na ekranu.

Set alternativnih akcija koje korisnik u nekom trenutku može da izvede je definisan onim što vidi na ekranu. Korisnik ne može da izvede neku akciju ako grafički element koji predstavlja tu akciju nije vidljiv.

VIZUELIZACIJA INFORMACIJA

Vizuelizacija je kognitivni proces koji omogućuje ljudima da razumeju informacije

Vizuelizacija je kognitivni proces koji omogućuje ljudima da razumeju informacije. U nekim slučajevima ograničenost ljudske percepcije ne dozvoljava korisniku da primi informaciju. Ovo se može dogoditi ako je informacija preobimna, ima na primer previše brojeva, ili ako je informacija apstraktna. U ovakvim slučajevima grafički interfejs pomaže korisniku da dobije vizuelnu predstavu informacija.

❖ 4.1 Direktna manipulacija

KONCEPT DIREKTNE MANIPULACIJE

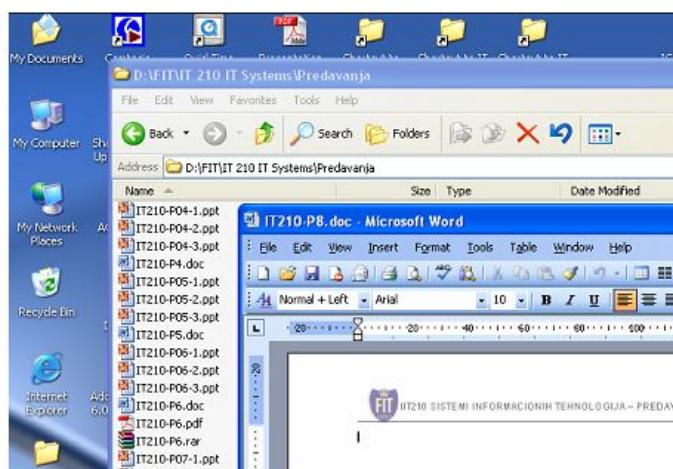
Namena direktne manipulacije je da omogući korisniku da manipuliše objektima koji su mu prezentovani, koje odgovaraju manipulacijama nad fizičkim objektima

Koncept direktne manipulacije je predložio još 1982. godine Shneiderman. Ovaj koncept ima sledeće karakteristike:

Sistem je dizajniran kao proširenje realnog sveta. Prepostavlja se da korisnik dobro poznaje objekte i procese u svom radnom okruženju. Sistem jednostavno pravi repliku takvog radnog okruženja iscrtavajući ga na displeju. Korisnik ima mogućnost da pristupa i modifikuje te objekte među kojima su i prozori. Tako, na primer u operativnom sistemu Windows, desktop predstavlja repliku radnog stola na kome se nalaze dokumenti, alati i pribor. Korisniku je omogućeno da radi u poznatom okruženju i da se fokusira na podatke, a ne na aplikacije i alate. Fizička organizacija sistema, koja je vrlo složena i koju korisnik ne poznaje, je sakrivena od njega.

Kontinualna vidljivost objekata i akcija. Kao i na radnom stolu, objekti mogu da budu stalno vidljivi ili sakriveni. Akcije se izvode pomeranjem miša, a rezultat pomeranja je odmah vidljiv kao pomeranje kursora po ekranu. Komandni objekti su vidljivi i postavljeni su uobičajeno na ivicama prozora.

Akcije su brze i inkrementalne sa vidljivim prikazom rezultata. Svaka izvedena operacija menja prikaz na displeju tako da korisnik može odmah da vidi rezultate akcije.



Slika 4.1.1 Prikaz rezultata akcije [Izvor: Autor]

Inkrementalne akcije su reverzibilne. Ukoliko se uoči da izvedena akcija nije bila željena ili daje neočekivane rezultate, moguće je vratiti se na prethodno stanje (akcija **undo**).

Koncept direktnе manipulacije nije svojstven samo za grafičko radno okruženje. Sličan koncept je postojao i kod starih karakter-baziranih tekst editora. Međutim, sa pojmom grafičkog okruženja koncept direktnе manipulacije je usavršen i postao je mnogo efikasniji.

✓ 4.2 Indirektna manipulacija

KONCEPT INDIREKTNE MANIPULACIJE

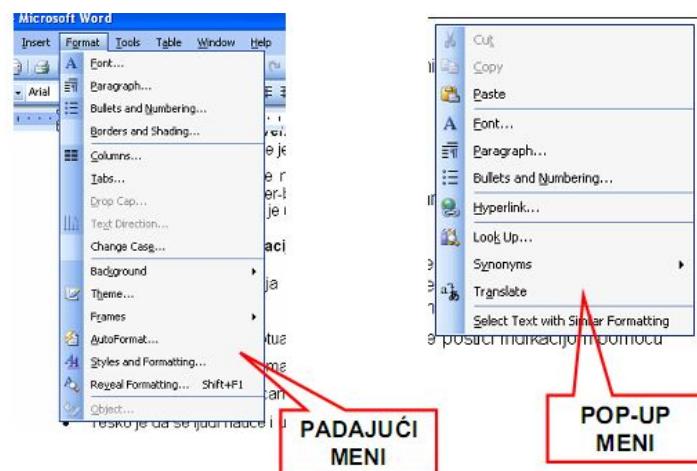
Kada je zbog bilo kog razloga nemoguće primeniti direktnu, primenjuje se indirektna manipulacija.

U praksi direktna manipulacija svim objektima na ekranu i akcijama nije moguća zbog nekog od sledećih razloga:

- Operaciju je teško konceptualizovati u grafičkom sistemu
- Grafičke mogućnosti sistema mogu biti ograničene
- Raspolozivi prostor na ivicama prozora za postavljanje kontrola za manipulaciju je ograničen
- Teško je da se ljudi nauče i upamte sve potrebne operacije i akcije.

Kada je zbog bilo kog razloga nemoguće primeniti direktnu, primenjuje se indirektna manipulacija. Pri indirektnoj manipulaciji grafički simboli se zamjenjuju tekstrom, kao što se obično postiže padajućim i **pop-up** menijima. Meniji su tekstualne liste mogućih akcija, pa predstavljaju instrument indirektnе manipulacije. Umesto identifikovanja objekta mišem unosi se potreban tekst.

Izvršavanje akcija koje se nalaze u padajućim i **pop-up** menijima se može postići indikacijom pomoću miša ili kombinacijom tastera (na primer **Ctrl-A** za izbor svih objekata u tekućem prozoru). U oba slučaja se radi o indirektnoj manipulaciji.



Slika 4.2.1 prikaz menija [Izvor: Autor]

KOMBINACIJA DIREKTNE I INDIREKTNE MANIPULACIJE

Korišćenje metode direktne manipulacije za repetativne zadatke može da bude zamorno i dosadno, pa iskusni korisnici preferiraju indirektnu metodu

Većina operacija u grafičkom korisničkom okruženju se može postići i metodama direktne i indirektne manipulacije. Korisnik može da bira koju će metodu koristiti i to je stvar individualne sklonosti. Korišćenje metode direktne manipulacije za repetativne zadatke može da bude zamorno i dosadno, pa iskusni korisnici preferiraju indirektnu metodu

Većina grafičkih sistema bazirana na prozorima je kombinacija metoda direktne i indirektne manipulacije. Većina komandnih akcija može da se izvrši direktnom manipulacijom, ali ima mnogo slučajeva kada je neophodno koristiti metod indirektne manipulacije. Ako je na primer potrebno zapisati datoteku pod novim imenom (akcija **Save As**) neophodno je primeniti koncept indirektne manipulacije da bi se ukucalo novo ime datoteke.

▼ 4.3 Prednosti i nedostaci grafičkog sistema

PREDNOSTI GRAFIČKOG SISTEMA

Simboli se prepoznaju brže nego tekst. Istraživanja su pokazala da slike ili ikone, odnosno simboli, mnogo brže prosleđuju informaciju korisniku nego tekst

Jednostavnost grafičkog korisničkog interfejsa i intuitivnost koja je njime postignuta doprinela je smanjenju potrebe da korisnik pamti složene komande i procedure, a period obuke je drastično smanjen. Takođe je povećana efektivnost obrade informacija. Uspeh grafičkog okruženja zasnovan je na sledećim činjenicama:

Simboli se prepoznaju brže nego tekst. Istraživanja su pokazala da slike ili ikone, odnosno simboli, mnogo brže prosleđuju informaciju korisniku nego tekst. Zbog toga se u grafičkom okruženju uvode standardizovane biblioteke grafičkih simbola koje imaju isto značenje i na operativnom sistemu i u aplikacijama. Na taj način se i proces učenja ubrzava.



Slika 4.3.1 prikaz simbola u Wordu [Izvor: Autor]

Bolji prikaz informacija i brže rešavanje problema. Iskustvo u korišćenju grafičkih sistema je pokazalo da se vizualno ili prostorno predstavljene informacije lakše perceptuju i pamte i doprinose uspešnom rešavanju problema.

Lakše pamćenje. Zbog veće jednostavnosti početnicima je lakše da upamte koncepte

operacija.

Okruženje je mnogo prirodnije. Grafičko predstavljanje objekata je mnogo prirodnije i bliže urođenim ljudskim sposbnostima.

Ludska bića počnu da komuniciraju sa okolinom najpre vizualno, a tek kasnije jezikom. Osim toga vizuelna memorija čoveka je mnogo jača od drugih vrsta memorije.

Podržan je mnogo konkretniji način razmišljanja. Objekti prikazani grafički su direktno upotrebljivi, pa nema potrebe za mentalnom dekompozicijom zadatka sa više kompleksnih sintatičkih formi. Potreba za apstraktnim razmišljanjem je zbog toga smanjena. Pošto su objekti vidljivi obezbeđena je i slika njihovog trenutnog konteksta.

Manje grešaka. Konkretnije razmišljanje pruža manje prostora za greške. Reverzibilnost akcija dalje olakšava rad jer je uvek moguće vratiti objekte na stanje pre akcije (komanda **undo**).

Povećanje osećaja kontrole. Korisnik odmah nakon iniciranja akcije vidi rezultate što mu daje osećaj da ima kontrolu nad sistemom.. Ukoliko je pogrešio, korisnik zna da može da se vратi na prethodno stanje, pa se gubi strah od korišćenja sistema.

Prevazilaženje jezičkih barijera. Većina teksta baziranih radnih okruženja koriste engleski jezik za interakciju sa sistemom. Jezik grafičkih simbola je mnogo univerzalniji i lakši za učenje.

Manje kucanja. Korišćenje miša kao ulaznog uređaja u mnogome smanjuje potrebu za kucanjem.

NEDOSTACI GRAFIČKOG SISTEMA

Nedostaci su: uvećani problemi projektovanja, učenje i dalje je potrebno, nema napretka u upotrebljivosti, nesavršenost grafičkih simbola, povećane šanse za zbunjivanje korisnika

Grafičko radno okruženje nije uvek bolje od tekstualnog. Pokušaj da se grafički pristup nametne u svim sistemskim komponentama i povećana složenost mogu da izazovu kontraefekte. Neki od nedostataka grafičkih sistema su:

Uvećani problemi projektovanja. Projektantima grafičkih sistema na raspolaganju su različite metode, alati i biblioteke gotovih klasa. Bogatstvo mogućnosti, na žalost, ne garantuje da će se realizovati dobro rešenje. Pri projektovanju grafičkog sistema povećava se broj faktora koji utiče na kvalitet rešenja. Nije dovoljno samo napraviti sistem koji funkcioniše, već je potrebno postići i intuitivnost, efikasnost, estetiku i jednostavnost sistema.

Učenje je i dalje potrebno. Iako je učenje olakšano, korišćenje grafičkog okruženja za početnike nije jednostavno jer je potrebno da se upozna sa konceptima, značenjem grafičkih elemenata, njihovim mogućnostima i ograničenjima.

Nema napretka u upotrebljivosti. Poznato je da iskusnim korisnicima smeta preveliki broj akcija (na primer otvaranje prozora, promena fokusa, klik mišem) koje treba preduzeti da bi se završio neki zadatak. Velike softverske kuće rade eksperimentalna istraživanja čiji je cilj da se upotrebljivost grafičkog okruženja poboljša, ali se rezultati ne objavljaju zbog konkurenциje.

Nesavršenost grafičkih simbola. Iako grafički elementi za većinu korisnika predstavljaju direktnu informaciju, neki korisnici ne moraju da razumeju značenje. Ikone i simboli nisu standardizovani i različiti su u različitim operativnim sistemima i programskim bibliotekama.

Razlog za ovo može da bude zbog niskog stepena obrazovanja ili različite kulture iz koje korisnik potiče. Simbol koji je apsolutno jasan jednom inženjeru, ne mora da govori ništa jednom lekaru. U takvim slučajevima dvosmislenost može da se izbegne samo tekstualnim natpisom (engl. **caption**) koji prati ili zamjenjuje ikonu. Iskustvo je takođe pokazalo da korisnik može da napravi razliku u značenju ikona samo za njihov ograničen broj. Preveliki broj ikona može da zbuni korisnika. I u ovom slučaju je tekstualni opis jedino rešenje.

Previše manipulativnih radnji. Grafički korisnički interfejs zahteva često pomeranje miša i pritisak na tastere. Pri tom treba postići određenu preciznost i brzinu. Ovo je za neke korisnike, a posebno stare i one sa posebnim potrebama vrlo teško. **Povećane šanse za zbunjivanje korisnika.** Preveliki broj otvorenih prozora na ekranu može da zbuni neiskusne korisnike. Uočeno je da korisnici maksimiziraju prozore da bi izbegli konfuziju.

✓ 4.4 Veb korisnički interfejs

PROJEKTOVANJE VEB INTERFEJSA

Projektovanje veb interfejsa je u osnovi projektovanje navigacije i prezentacije informacija

Dok su se prvi tekst bazirani i grafički korisnički interfejsi razvijali u velikim softverskim kućama, razvoj prvih veb korisničkih interfejsa razvijan je uglavnom od strane pojedinaca entuzijasta. Zbog toga prvi veb interfejsi nisu imali upotrebljivost koju danas imaju.

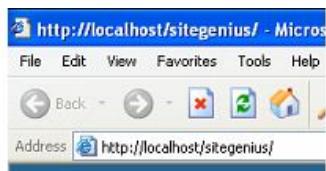
Projektovanje veb interfejsa je u osnovi projektovanje navigacije i prezentacije informacija. Kvalitet veb interfejsa u velikoj meri zavisi od organizacije strukture podataka, menija, sadržaja i linkova na druge dokumente.

Veb interfejs ima mnogo zajedničkih osobina sa grafičkim korisničkim interfejsom ali i mnogo specifičnosti. Ovde će se ukazati na neke specifičnosti veb interfejsa.

Uredaji. Projektant veb interfejs nikada ne zna sa kojih će se sve uređaja pristupati veb sajtu. To može da bude mobilni telefon, personalni digitalni asistent, laptop ili personalni računar sa grafičkim monitorom. Funkcionisanje i upotrebljivost istog veb interfejsa na ovim uređajima može biti različito. Zbog toga se prave različiti veb interfejsi za različite klase uređaja.

Komponente. Veb sistem se sastoji od dve komponente: veb čitača i veb stranice. Veb čitači su tipično aplikacije sa grafičkim korisničkim interfejsom. Veb stranice su kompoziti koji mogu da se sastoje od različitih informacionih objekata kao što su tekst, slike, zvuk, video i linkovi ka drugim stranicama.

Navigacija. Najveća specifičnost veb interfejsa je navigacija. Ona se postiže akcijama koje omogućuje veb čitač ili akcijama koje omogućuje veb strana. Veb čitači uobičajeno omogućavaju eksplicitno zadavanje URL-a i prelazak na prethodnu ili narednu stranu, kao i odlazak na home veb stranu.



Slika 4.4.1 prikaz pretraživanja i navigacije [Izvor: Autor]

Navigacija ugrađena u stranu zavisi od toga kako je strana projektovana i kojim je alatima realizovana. Uobičajeni mehanizmi navigacije su meniji i linkovi.

PROJEKTOVANJE VEB APLIKACIJE - VIDEO

How to design a web application from start to finish

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 4.5 Kognitivni modeli

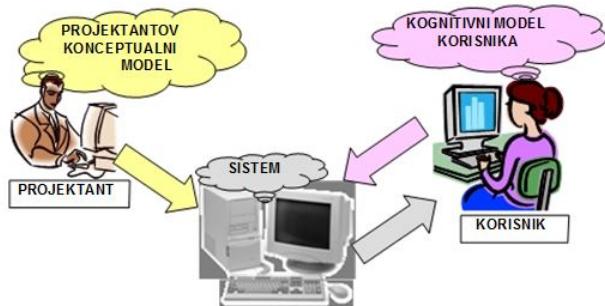
KONGITIVNI MODELI

Kognitivni modeli su interne reprezentacije trenutne percepcije realnog sveta koje ljudi kreiraju u svom mozgu

Kognitivni modeli su interne reprezentacije trenutne percepcije realnog sveta koje ljudi kreiraju u svom mozgu. Ljudi stalno kreiraju i pristupaju internim reprezentacijama trenutnih situacija. Kognitivni modeli pomažu ljudima da rezonuju kako bi na osnovu percepcije okoline izveli neku akciju. U nekim slučajevima, čovek za istu situaciju može da stvori više modela koji međusobno konkurišu. Tada čovek ima dilemu kako da razume svet. Kada čovek ima jedan ispravan kognitivni model onda lakše uči i rezonuje.

Tokom interakcije sa računarskim sistemom ljudi formiraju kognitivni model interakcije koji sadrži njihovo viđenje funkcija, mogućnosti i ograničenja sistema. Korisnik koristi ovaj model da rezonuje o sistemu, da shvati ponašanje sistema i da objasni zašto sistem reaguje na taj način. Ukoliko korisnik nije formirao kognitivni model funkcionisanja sistema ili je on loše formiran korisnik će biti nezadovoljan, a njegov rad će biti neefikasan. Korisnički interfejs je veza između projektantovog konceptualnog modela funkcija sistema, mogućnosti i ograničenja i korisnikovog kognitivnog modela. Interfejs treba da prikaže korisniku sistem na takav način da vodi razvoj korisnikovog kognitivnog modela ka podudarnosti sa projektantom konceptualnim modelom.

Projektanti sistema treba da razviju takav korisnički interfejs koji će kod korisnika razviti kognitivni model koji odgovara konceptualnom modelu sistema. Ova metoda se u literaturi naziva „*sistem uzročnog prenosa*“.



Slika 4.5.1 Prikaz kognitivnog modela [Izvor: Autor]

RAZVOJ KOGNITIVNOG MODELA

Mentalni model se razvija postupno, paralelno sa iskustvom osobe.

Mentalni model se razvija postupno, paralelno sa iskustvom osobe. Kada se sretne sa novim računarskim sistemom korisnik počne da gradi novi kognitivni model na osnovu svojih očekivanja i iskustva u radu sa sistemom. Ako konceptualni model sistema odgovara uspostavljenom kognitivnom modelu, kognitivni model se ojačava i korišćenje sistema se čini korisniku intuitivnije. U suprotnom se javljaju problemi u učenju rada sa sistemom.

Osobe koje imaju iskustva u radu sa nekim sistemom će pri prelasku na rad sa novim sistemom poneti svoj izgrađeni kognitivni model. Ako rad novog sistema odgovara ovom modelu prihvatanje novog sistema će biti lako.

Osobe se razlikuju po tome kako stvaraju kognitivni model, pa se kaže da imaju različite kognitivne stilove. Neki ljudi su bolji u verbalnom razmišljanju jer bolje rade sa rečima i jednačinama. Drugi su bolji u prostornom rezonovanju, jer lakše manipulišu sa simbolima, crtežima i slikama. Neke osobe su analitički mislioci jer imaju smisla za sistematsku analizu problema.

Neki ljudi donose odluke na osnovu intuicije, dok su drugi više konkretni u razmišljanima. Praksa je pokazala da verbalisti, analitičari i konkretni mislioci preferiraju tekstualni stil interfejsa. Suprotno, osobama čije je stil razmišljanja prostorni, intuitivni ili apstraktни više odgovara grafički korisnički interfejs.

▼ Poglavlje 5

Pristupi

APLIKACIONI PRISTUP

Prvi grafički sistemi su bili aplikaciono orijentisani, jer su proistekli iz teksta baziranih sistema

Prvi grafički sistemi su bili aplikaciono orijentisani, jer su proistekli iz teksta baziranih sistema. Tekstualno bazirani sistemi su se nazivali aplikacije. Korisnik je pozivao aplikaciju da bi rešavao neki zadatak. Tek po otvaranju aplikacije korisnik je birao datoteku ili objekat sa kojim će raditi.

OBJEKTNO-ORJENTISANI PRISTUP

Objektno - orijentisani pristup dozvoljava korisniku da se fokusira na zadatak koji treba da izvrši. Pri tom je minimizirana vidljivost operativnog sistema ili aplikacije

Kod objektno orijentisanog pristupa korisnik najpre bira objekat, a zatim bira akciju koju će nad njim izvršiti. Objektno orijentisani pristup dozvoljava korisniku da se fokusira na zadatak koji treba da izvrši, pri tom je minimizirana vidljivost operativnog sistema ili aplikacije.

Objektno-orijentisano grafičko radno okruženje se sastoji od objekata i akcija. Objekti su ono što korisnik vidi na ekranu. Nad svakim objektom se može izvesti niz unapred definisanih akcija. Objekti se mogu sastojati od pod objekta. Na primer, jedna prezentacija se sastoji od slajdova. Slajd se sastoji od naslova i slike.

IBM-OV PRISTUP

IBM-ov pristup deli objekte u tri klase: objekti za predstavljanje podataka, kontejneri i uređaji

IBM-ov pristup System Application Architecture Common User Access Advanced Interface Design Reference ([SAA_CUA](#)) deli objekte u tri klase: objekti za predstavljanje podataka, kontejneri i uređaji.

Objekti za predstavljanje informacija služe za prikaz informacija na ekranu.

Kontejneri objekata su objekti koji sadrže druge objekte. Oni služe da grupišu objekte kako bi se olakšao pristup njima i kako bi se sa njima lakše manipulisalo. Postoje tri vrste kontejnera: radno mesto, direktorijum i radni prostor. Radno mesto je desktop na kome su smešteni svi objekti. Direktorijumi su kontejneri opšte namene za dugotrajno čuvanje objekata. Radni prostor je privremeni direktorijum u kome se čuvaju objekti sa **kojima se trenutno radi**.

Uredaji su objekti koji predstavljaju fizičke objekte iz realnog sveta, kao što je na primer štampač. Ovakvi objekti mogu da prime druge objekte da bi izvršili neku akciju. Na primer, datoteka se može dati štampaču da bi se odštampao njen sadržaj.

Drugi važan aspekt IBM-ovog pristupa je pogled.

Pogledi su metode da se vidu informacije nekog objekta. SSA CUA razlikuje četiri vrste pogleda: kompozitni, sadržaj, atributi i pomoć.

Kompozitni pogled predstavlja informacije i objekte koji su sadržani unutar nekog objekta.

Sadržaj predstavlja pogled na listu komponenata nekog objekta.

Atributi (engl. **settings**) su pogled na karakteristike nekog objekta.

Pomoć (engl. **help**) omogućuje pogled na informacije o načinu rada sa objektom.

MICROSOFT-OV PRISTUP

Moguće veze između objekata su: kolekcije, ograničenja, kompoziti i kontejneri

Microsoft Windows specificira karakteristike objekata zavisno od veza koje postoje između njih. Objekti mogu da egzistiraju unutar konteksta drugih objekata, a jedan objekat može da utiče na to kako će se drugi objekat prikazati i ponašati. Moguće veze između objekata su: kolekcije, ograničenja, kompoziti i kontejneri.

Kolekcija je najjednostavnija veza. Kolekcija je skup objekata koji ima zajednički aspekt. Kolekcija se može dobiti kao rezultat upita ili višestrukim izborom objekata. Nad kolekcijom objekata se mogu izvršiti operacije. Na primer, korisnik može da izabere neke datoteke u nekom direktorijumu, i da tu kolekciju objekata prekopira u drugi direktorijum.

Ograničenje je jača veza između objekata. Promena jednog objekta iz seta utiče na neke druge objekte iz istog seta. Na primer, dokument organizovan u stranice je primer ograničenja. Dodavanje teksta na jednoj stranici će promeniti sadržaj stranica iz date stranice.

Kompoziti su grupe objekata između kojih postoji tako jaka veza da se i sama agregacija ovih objekata može smatrati objektom. Na primer, skup grafičkih objekata može biti grupisan. Sve dok se ne izvrši operacija **ungroup** ova agregacija će biti jedan objekat.

Kontejner je objekat koji sadrži druge objekte. Na primer, dokument je kontejner za neki tekst, a direktorijum je kontejner za dokumente. Kontejner može da utiče na ponašanje svog sadržaja. On može da doda ili prikrije osobine ili operacije objekata smeštene unutar njega.

Kontejner može da upravlja pristupom sadržaju objekata ili da upravlja koje će vrste objekata primati.

Druga važna karakteristika objekata u Windows okruženju je perzistencija. **Perzistencija** je očuvanje stanja objekta kada je jednom uspostavljeno. Stanje objekta, kao što je na primer veličina prozora, pozicija kurzora ili pozicija.

AKCIJE I ATRIBUTI OBJEKATA

Akcije mogu biti ili komande ili promena atributa objekata.

Akcije

Nad svakim objektom se mogu preduzeti neke akcije. Akcije mogu biti ili komande ili promena atributa objekata. Komande su akcije kojim se manipuliše objektima. Komande se izvršavaju odmah. Tipični primjeri komandi su otvaranje prozora, otvaranje dokumenta, štampanje dokumenta.

Atributi objekata

Svaki objekat ima svoje attribute. Atributi su jedinstvene karakteristike objekta koje mogu biti promenjene od strane korisnika. Na primer, korisnik može promeniti stil fonta nekog paragrafa, veličinu fonta i njegovu boju.

TRENUTNE TENDENCIJE U RAZVOJU KORISNIČKOG INTERFEJSA

Neke od trenutnih tendencija su: minimalizam, skeumorphism, content chunking, dugačke stranice

Neke od trenutnih tendencija u razvoju korisničkog interfejsa su sledeće:

1. **Minimalizam** - kompleksne ikonice se menjaju jednostavnijim, interfejsi sa velikim brojem boja se zamenjuju sa jednostavnim interfejsima sa manje boja. Za većinu korisnika, ovaj nedostatak vizuelnih detalja funkcioniše jako dobro - interfejs je lak za učenje i snalaženje, dok elementi dizajna ne odvraćaju pažnju korisnika na njegovom putu ka ispunjenju zadataka koji pokušava da ostvari

- Kada koristiti ovaj pristup: minimalizam je odličan način da se dizajnira veb aplikacija koja se fokusira na sadržaju koji generišu. Međutim, imajte na umu da će mnogi klijenti naći ovaj pristup previše pojednostavljenim po svom ukusu, pa je preporučljivo da proverite sa njima pre primene ovog pristupa u svom projektu.

2. **Skeuomorphism** - Skeuomorphic dizajn korisničkog interfejsa je prvi popularizovao Apple, a zatim je on usvojen od strane mnogih drugih kompanija. U osnovi, ovaj pristup se oslanja na imitirajući izgled i funkcionalnost tradicionalnih i poznatih objekata kako bi interfejs bio što intuitivniji za korišćenje. Na primer, koristeći drvene police sa knjigama gde se nalaze digitalni sadržaji asocira korisnika na biblioteku i lakše se snalazi u potrazi sa dokumentom koji mu je

potreban. Dok se jedni zalažu za ovaj pristup, drugi smatraju da je ovaj pristup loš, zbog toga što imitacijom realnog sveta, uvodimo njegove nedostatke i u digitalni svet (na primer, veliki broj knjiga koje je teško vizuelno pretraživati kao i u biblioteci)

3. "**Content chunking**" - ova tehnika predstavlja veliku količinu sadržaja u manjim vizuelnim delovima, tako da je lakše za ljude da čitaju i mentalno procesiraju. Na primer, deljenje ovog teksta u manje delove i sekciјe sa naslovima, sa pratećim slikama će činiti ovaj tekst lakšim za čitanje u poređenju sa dugim monotonim tokom teksta. Ovaj pristup se pojavio zajedno sa internetom, kada su ljudi krenuli da čitaju veliku informaciju, uz potrebu da to što efikasnije pročitaju.

- Kada koristiti ovaj pristup: Upotreba ove tehnike je nešto što na kraju zavisi od klijenta. Međutim, možete da ga koristite u predlogu dizajna koji ukazuju kako informacije treba da izgledaju na sajtu, objašnjavajući zašto je korisno da se one predstave na taj način.

4. **Dugačke stranice** - ovaj pristup je nekada bio široko odbijen i od strane klijenata i od strane projektanata. Jedan od argumenata za korišćenje ovakvog pristupa je da su korisnici navikli na vertikalno kretanje po sajtu (uz pomoć miša), pa deljenje sadržaja po posebnim stranama otežava korisniku da nađe sadržaj, zbog toga što zahteva više napora od korisnika da ga pronađu i do ga.

- Kada koristiti ovaj pristup: kada se predstavljaju karakteristike i prednosti proizvoda ili drugog sličnog sadržaja. Ako sadržaj nije direktno povezan, trebalo bi da ga podelite na posebnim stranama.

▼ Poglavlje 6

Pokazna vežba: PhpMyAdmin

KORIŠĆENJE BAZA PODATAKA IZ PHP JEZIKA

Integracija baza podataka i sajtova predstavlja veoma važan segment u razvoju Web interfejsa

Predviđeno vreme pokazne vežbe je 70 minuta.

Razvoj WWW omogućio je pristup velikom broju informacija. Međutim, često se dešava da suština informacije ostane u senci grafičkog okruženja. U želji da privuku što veći broj posetilaca, programeri koristeći se snažnim alatima, razvijaju vizuelno atraktivne Web stranice često uskraćujući posetiocima važne informacije. Snaga WWW je upravo u mogućnosti prikazivanja sadržaja koji na taj način postaju dostupni velikom broju ljudi.

Integracija baza podataka i sajtova predstavlja veoma važan segment u razvoju Web interfejsa, koji korisnicima omogućava da na lak i efikasan način dođu do željenih informacija. Primeri korišćenja baza podataka na Web stranicama su brojni. Na primer, sajt sa velikim brojem informacija teško je ažurirati. Poseban problem predstavlja promena ukupnog dizajna sajta. Ukoliko bi se čitav sadržaj smestio u bazu, za promenu čitavog dizajna potrebno je promeniti nekoliko stranica.

U marketinškom pogledu korišćenje baza je značajno, jer omogućava da se različitim posetiocima prikažu različiti baneri. Otvaranjem strane, iz baze se bira baner koji se prikazuje. Moguće je jednostavno praćenje broja pojavljivanja nekog sadržaja i njegova jednostavna promena. Dobar primer korišćenja baza na WWW predstavljaju i forumi. Ovaj pristup mnogo je efikasniji i pruža veliki broj mogućnosti.

Veliki značaj za korišćenje baza imaju skript jezici poput PHP, Perl, Python i sl. Koristeći ove jezike u kombinaciji sa nekim od sistema za upravljanje bazama kakav je npr. MySQL, moguće je na jednostavan način integrisati bazu podataka sa Web sajtom.

ODBC

ODBC je široko rasprostranjen API interfejs za pristup bazama podataka

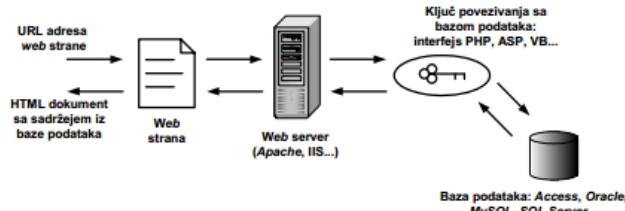
ODBC (engl. [Open Database Connectivity](#)) je standard koji je razvio Microsoft. Svaki komercijalni sistem za upravljanje bazama podataka i većina sistema koji nisu komercijalni imaju ODBC drajvere, koji omogućavaju pristup tim sistemima. To znači da se ne moraju

koristiti drajveri tih sistema. ODBC je važan jer omogućava pristup bazi podataka koja se nalazi u Windows okruženju sa UNIX ili Linux sistema.

ODBC je široko rasprostranjen API interfejs (**Application Programming Interface**) za pristup bazama podataka. Ovaj standard omogućio je korišćenje u distribuiranim okruženju sistema kao što su MS Access, Oracle, SQL Server itd. ODBC omogućava apstraktni sloj koji krije specifičnosti pristupa određenoj bazi podataka i dozvoljava programerima razvoj bez potrebe da ulaze u detalje funkcionisanja baze. Na slici 1 prikazan je redosled pristupa bazi podataka korišćenjem web strana.

Ukratko, proces se odvija na sledeći način:

1. u adresnu liniju čitača unosi se URL ili se klikne na odgovarajuću hipervezu,
2. serveru se šalje zahtev za otvaranje odgovarajućeg dokumenta,
3. server sve datoteke sa ekstenzijom .php šalje posebnom interfejsu na obradu,
4. PHP čita stranu i obrađuje sve naredbe, pa i one koje se odnose na pristup ili promenu podataka u bazi, unoseći rezultate rada u HTML dokument, i
5. PHP pretvara sve naredbe i rezultate u HTML kod i vraća zahtev serveru, koji zatim šalje željenu stranu.



Slika 6.1.1 Proces pristupa bazi podataka [Izvor: Autor]

PHPMYADMIN

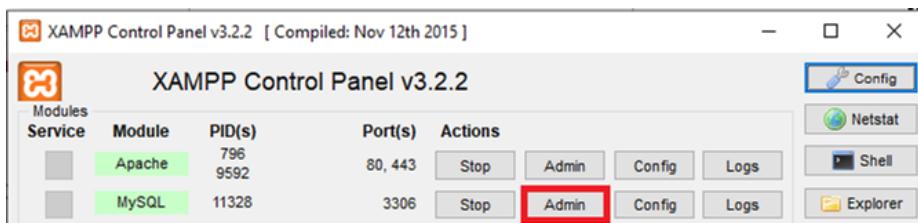
PhpMyAdmin je alat koji služi za upravljanje MySQL-a preko Web-a

PhpMyAdmin je alat napisan u PHP-u koji služi za upravljanje MySQL-a preko Web-a.

Prednosti phpMyAdmin-a su što daje mogućnost jednostavnog upravljanja podataka (upravljanje tabelama, redovima, kolonama, relacijama, korisnicima...) preko grafičkog interfejsa. Naravno, pored toga moguće je i izvršavati SQL upite preko konzole.

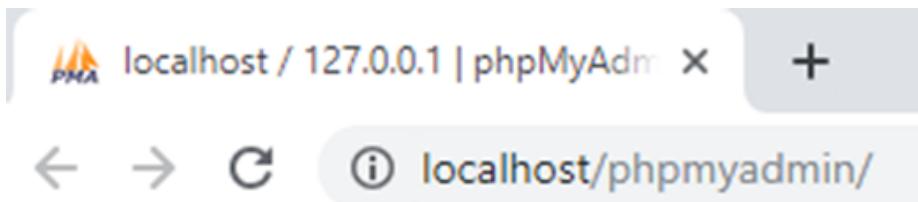
Kako bi pristupili phpMyAdmin-u, koristimo XAMPP Control Panel u kojem pored Apache servisa pokrećemo i MySQL servis.

Odabirom opcije Admin za MySQL servis, otvara se index strana phpMyAdmin-a u pretraživaču.



Slika 6.1.2 Pokretanje MySQL servisa i pristupanje phpMyAdmin-u [Izvor: Autor]

Takođe, početnoj strani phpMyAdmin-a se može pristupiti i unosom `localhost/phpmyadmin` u pretraživač.



Slika 6.1.3 Link za pristup phpMyAdmin-u [Izvor: Autor]

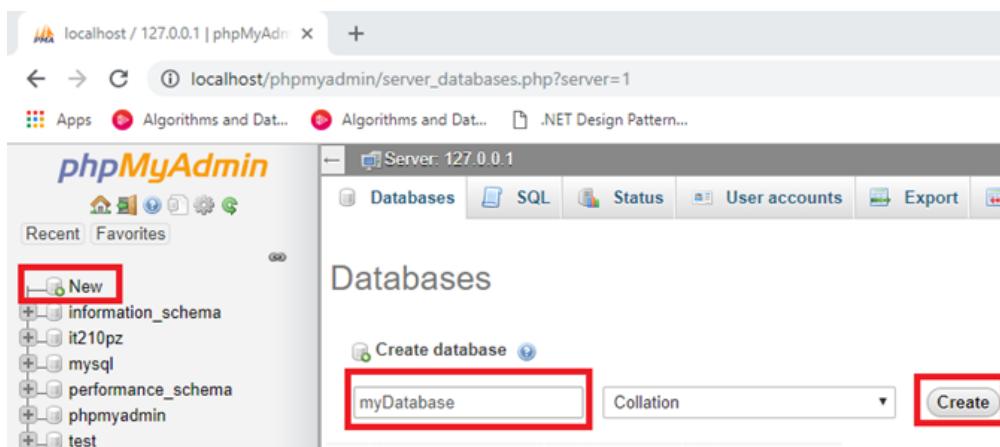
KREIRANJE BAZE PODATAKA U PHPMYADMIN

Kreiranje nove baze podataka

Za potrebe narednog primera potrebno je da kreiramo novu bazu podataka pomoću phpMyAdmin-a.

Baza treba da ima naziv `myDatabase`, a moguće je kreirati odabirom opcije `new` u levom delu phpMyAdmin-a na vrhu liste baze podataka koje već postoje.

Nakon klika na opciju `new` pojavljuje se mogućnost za imenovanje baze podatka.



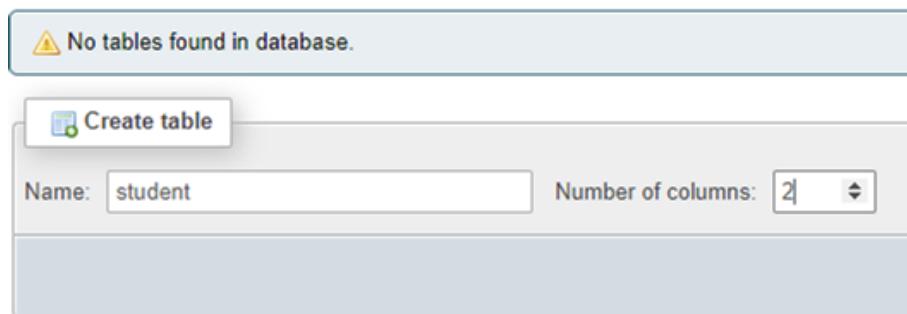
Slika 6.1.4 Kreiranje i imenovanje baze podataka [Izvor: Autor]

KREIRANJE TABELE

Potrebno je kreirati tabelu

Pošto se kreira baza automatski se dobija prikaz za kreiranje tabele gde treba da se navede naziv tabele i broj kolona koje će da sadrži.

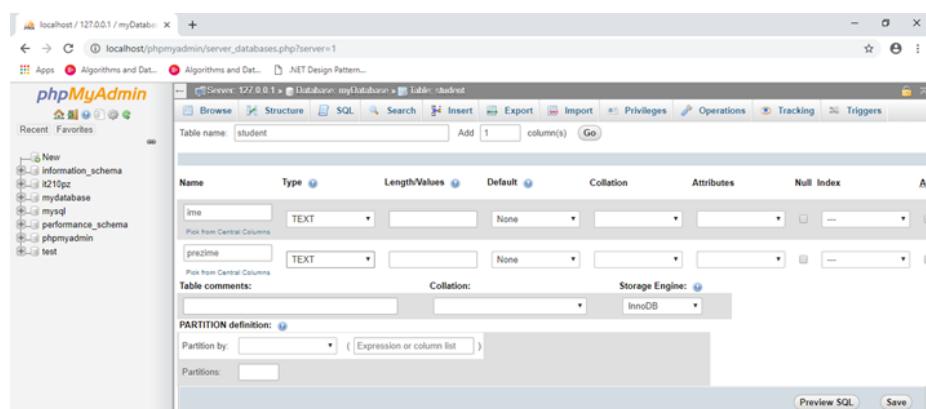
Potrebno je da kreiramo tabelu student koja će imati dve kolone ime i prezime.



Slika 6.1.5 Forma za kreiranje tabele [Izvor: Autor]

Nakon toga se dobija prikaz sa opcijama za definisanje tih kolona (njihov naziv, tip podataka, postavljanje kolone na autoincrement, postavljanje kolone kao primary key, index itd.)

U našem slučaju potrebno je definisati jednu kolonu za ime koja će biti tipa text i drugu kolonu prezime koje je, takođe, tipa text.



Slika 6.1.6 Kreiranje tabele i njenih kolona [Izvor: Autor]

PRIKAZ STRUKTURE KREIRANE TABELE

Strukturu tabele je moguće i naknadno menjati

Tabelu je moguće modifikovati i naknadno. Ukoliko je potrebno, postoje opcije za brisanje kolona ili za njihovu izmenu. Takođe, postoji i opcija za dodavanje jedne ili više kolona. Na slici je prikaz strukture tabele student.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	ime	text	latin1_swedish_ci		No	None			Change Drop More
2	prezime	text	latin1_swedish_ci		No	None			Change Drop More

Slika 6.1.7 Struktura tabele student [Izvor: Autor]

DODAVANJE REDOVA U TABELU

Neophodno je uneti podatke u tabelu

Za dodavanje zapisa (reda) u tabelu mogu se koristiti dve opcije:

- SQL – preko konzole i SQL upita
- Insert – preko grafičkog interfejsa



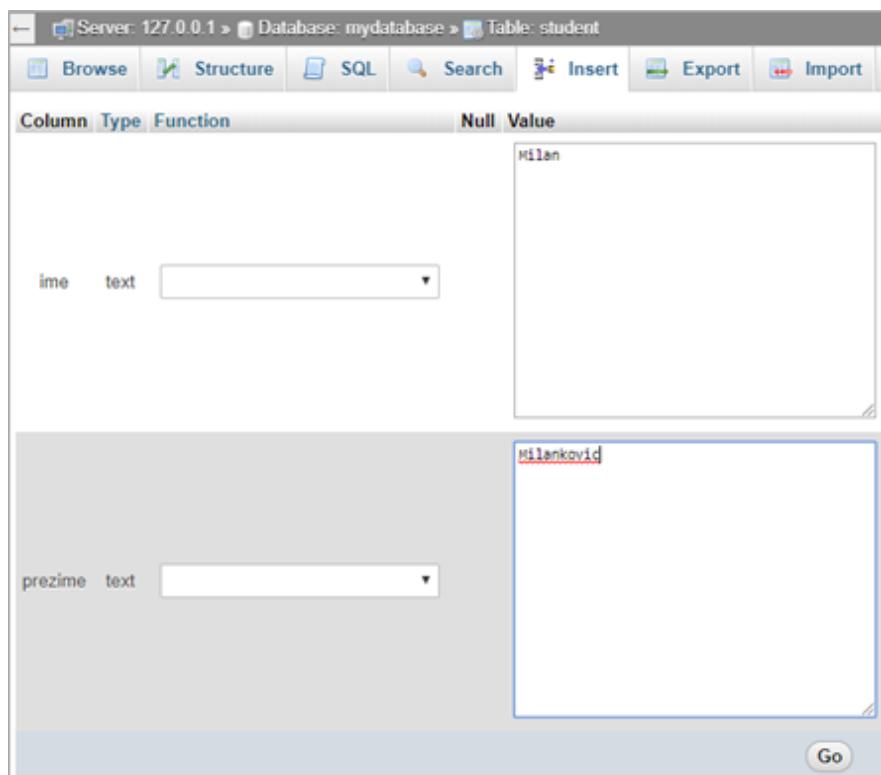
Slika 6.1.8 Opcije za dodavanje novih zapisa [Izvor: Autor]

Prilikom dodavanja preko SQL upita, treba odabratи opciju SQL u navigacionom baru, a potom uneti željeni upit i izvršiti ga klikom na Run.

```
Run SQL query/queries on table myDatabase.student: Run
1 INSERT INTO `student` (`ime`, `prezime`) VALUES ('Marko', 'Markovic');
```

Slika 6.1.9 Dodavanje reda preko konzole [Izvor: Autor]

Osim preko SQL upita, moguće je i dodati novi red u tabelu putem grafičkog interfejsa i to preko opcije Insert.



Slika 6.1.10 Unos putem grafičkog interfejsa [Izvor: Autor]

FORMA ZA UNOS PODATAKA

Podaci uneti u tekstualno polje, nakon klika na dugme Unesi, šalju se PHP skripti koja podatke dalje prosleđuje u bazu podataka

Predviđeno vreme za izradu primera je 25 minuta.

Primer

Tabele kreirane baze su prazne. Sa druge strane, HTML je staticke prirode i ne moze da utice na podatke u bazi. To je razlog korišćenja PHP skripta. U ovom slučaju HTML se koristi za kreiranje formulara kojim se podaci prosleđuju dalje. Kod HTML dokumenta može da izgleda ovako:

```
<html>
  <head>
  </head>
  <body>
    <form action="primer1.php" method="post">
      Unesite ime: <input type="text" name="ime"><br/>
      Unesite prezime: <input type="text" name="prezime"><br/><br/>
      <input type="Submit" value="Unos">
    </form>
  </body>
</html>
```

U ovom slučaju koriste se dva polje za unos imena i prezimena i dugme kojim se podaci iz polja šalju. Tekstualna polja imaju atribut name koji omogućava PHP skriptu da koristi vrednosti koja se unose u polja. Dokument se snima na poznati način npr. kao **primer1.html**.

Podaci uneti u tekstualna polja, nakon klika na dugme Unesi, šalju se PHP skriptu koji podatke dalje prosleđuje u bazu. Za unos podataka u datu bazu može se koristiti sledeći PHP skript:

```
<?php
    $ime=$_POST['ime'];
    $prezime=$_POST['prezime'];

    try{
        $con = new PDO("mysql:host=localhost;dbname=mydatabase", "root","");
        $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    }
    catch(PDOException $e){
        echo "Error: " . $e->getMessage();
    }

    $sql="INSERT INTO student (ime, prezime) VALUES ('$ime', '$prezime')";
    $exc = $con->query($sql);

    if($exc) {
        print "Informacija je uspešno uneta u bazu.";
    }
    else {
        print "Greska prilikom unosa.";
    };
    echo "<p><a href=primer1.html> Povratak </a>";
?>
```

PRIKAZIVANJE PODATAKA IZ BAZE

U ovom primeru se kreira HTML dokument i PHP skript koji će omogućiti prikazivanje svih osoba koje se nalaze u bazi lista iz prethodnog primera

U ovom primeru se kreira HTML dokument i PHP skript koji će omogućiti prikazivanje svih osoba koje se nalaze u bazi lista iz prethodnog primera. Kao poslednji red prebrojati koliko ima ukupno redova u tabeli.

U ovom primeru za aktiviranje skripta može se koristiti samo jedno dugme:

```
<html>
    <head>
    </head>
    <body>
        <form action="primer2.php" method="post">
            <input type="Submit" value="Prikaži sve">
        </form>
```

```
</body>  
</html>
```

Klikom na dugme *Prikaži* sve poziva se PHP skripta koja izvršava upit i potom podatke uzete iz baze prikazuje u vidu tabele pomoću while petlje.

Kod PHP skripte **primer2.php** se nalazi u nastavku:

```
<?php  
try{  
    $con = new PDO("mysql:host=localhost;dbname=mydatabase", "root", "");  
    $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);  
}  
catch(PDOException $e){  
    echo "Error: " . $e->getMessage();  
}  
  
$sql="SELECT * FROM student";  
$exc = $con->query($sql);  
  
$nbrow = 0;  
echo "<table border=1><th>Ime</th><th>Prezime</th>";  
while($row = $exc->fetch()){  
    echo "<tr><td>" . $row["ime"] . "</td><td>" . $row["prezime"] .  
"</td></tr>";  
    $nbrow++;  
}  
echo "<tr><td colspan=2>Ukupno: " . $nbrow . " studenata</td></tr></table>";  
echo "<br/><a href=primer1.html> Unos nove adrese </a>";  
?>
```

FUNKCIJE ZA ČITANJE PODATAKA IZ BAZE

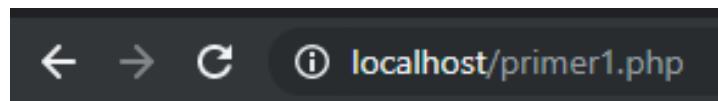
Uz pomoć funkcije fetch() proverava se da li postoje vrednosti u posmatranoj vrsti tabele

U prvom primeru bilo je potrebno kreirati formu za unos podataka u bazu. Da bi se konektovali na bazu kreirali smo novi PDO objekat. Prvi parametar konstruktora PDO klase je connection string, koji zavisi od vrste baze podataka, i može se lako naći na internetu. Druga dva parametra su username i password za bazu. U slučaju greške, koristi se standardna try/catch konstrukcija za obradu izuzetka.

Nakon toga je navedena promenljiva koja sadrži SQL upit koji treba da se izvrši, u ovom slučaju unos imena i prezimena koji su uzeti iz prethodno kreirane forme.

Metoda query kreiranog objekta se koristi za izvršavanje datog SQL koda.

Na kraju se nalazi provera pomoću if-else petlje koja ukoliko je upis bio uspešan ispisuje određenu poruku.



Slika 6.1.11 Uspešno dodavanje u bazu [Izvor: Autor]

U drugom primeru je bilo potrebno prikazati podatke iz tabele.

Ponavljaju se linije za konekciju na bazu u try/catch konstrukciji. Nakon toga se navodi SQL upit za čitanje podataka iz baze, odnosno, tabele student. Sa metodom query izvršavamo upit i podatke čuvamo u varijabli `\$ exc`. Definisana je promenljiva `\$ nbrow` koja treba da predstavlja broj vrsta tabele. Na početku je ova vrednost nula.

Pomoću fetch() proverava se da li postoje vrednosti u posmatranoj vrsti tabele. Ukoliko postoji, tada je rezultat ove funkcije TRUE, u suprotnom je FALSE.

Ukoliko vrsta nije prazna, vrednostima u njoj može se pristupati preko naziva kolone ili brojevima (0 predstavlja prvu kolonu, 1 drugu itd.). U ovo slučaju 0 bi se odnosila na ime, 1 na prezime.

U okviru while petlje, `\$ nbrow` brojač uvećava se za jedan i ispisuju se ćelije tabele sa podacima trenutnog reda koju su u promenljivoj `\$ row`. Nakon prolaska trenutne vrste, fetch() prelazi na narednu vrstu. Petlja će se izvršavati sve dok fetch() ne dođe do kraja tabele.

Na kraju se prikazuje ukupan broj vrsta, odnosno osoba iz tabele, a ispod tabele se nalazi link koji vodi nazad na HTML stranu.

REZULTAT DRUGOG PRIMERA

Skript generiše HTML stranu sa podacima iz baze

Za realizaciju je potrebno u čitaču otvoriti prvo primer2.html. Klikom na dugme Prikaži sve, aktivira se skript koji prikazuje sve osobe iz baze i na kraju daje ukupan broj redova tabele.

The screenshot shows a table with two columns: 'Ime' (Name) and 'Prezime' (Last Name). The table contains five rows of data, followed by a summary row at the bottom.

Ime	Prezime
Marko	Markovic
Milan	Milankovic
Pera	Peric
Milan	Markovic
Ukupno: 4 studenata	

Unos nove adrese

Slika 6.1.12 Rezultat izvršenja skripte [Izvor: Autor]

▼ 6.1 Pokazna vežba: PHP i PDO

PDO

Preferirani način za pristup bazi u modernom PHP programiranju je PDO.

Predviđeno vreme pokazne vežbe je 20 minuta.

Opisani način za pristup bazi podataka je prihvatljiv u slučaju manjih aplikacija, ali ima više nedostataka. Npr. ako kompletan API za pristup bazi je ODBC specifičan. Ako bismo hteli da koristimo bazu podataka koja nema odgovarajući ODBC driver, moralni bismo da koristimo drugi API.

Preferirani način za pristup bazi u modernom PHP programiranju je PDO. PDO je biblioteka koja apstrahuje pristup različitim relacionim bazama pod jedinstvenim interfejsom.

Potrebno je naglasiti da PDO ne apstrahuje drivere za bazu, niti prevodi SQL za različite vendore. Za propisno funkcionisanje PDO biblioteke, potrebno je da odgovarajući driver bude instaliran na sistemu.



Slika 6.2.1 Arhitektura PDO biblioteke

[Izvor: <https://i.stack.imgur.com/BRmBz.png>]

PDO PRIMER

Da bi se konektovali na bazu kreiramo novi PDO objekat

Sledi primer korišćenja PDO za pristup mysql bazi:

```
try {
    $conn = new PDO('mysql:host=localhost;dbname=myDatabase', $username, $password);

    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $data = $conn->query('SELECT * FROM Student WHERE ime = "Milan"');

    foreach($data as $row) {
        echo $row['ime'];
        echo $row['prezime'];
    }
} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}
```

Da bi se konektovali na bazu kreiramo novi PDO objekat, kao u prvoj liniji. Prvi parametar konstruktora PDO klase je connection string, koji zavisi od vrste baze podataka, i može se lako naći na internetu. Druga dva parametra su username i password za bazu.

Metoda query kreiranog objekta se koristi za izvršavanje datog SQL koda. Potom se vrši iteracija kroz niz redova. Svaki red je asocijativni niz napunjen podacima iz baze.

U slučaju greške, koristi se standardna try/catch konstrukcija za obradu izuzetka.

PREPARED STATEMENT

Prepared Statement je dio SQL koda koji je prethodno pripremljen i parametrizovan.

Prepared Statement je deo SQL koda koji je prethodno pripremljen i parametrizovan.

```
try {
    $conn = new PDO('mysql:host=localhost;dbname=myDatabase', $username, $password);
```

```
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$stmt = $conn->prepare('SELECT * FROM Student WHERE ime = :name');

$stmt->bindParam(":name", "Milan");
$data = $stmt->execute();

foreach($data as $row) {
    echo $row['ime'];
    echo $row[' prezime'];
}

} catch(PDOException $e) {
    echo 'ERROR: ' . $e->getMessage();
}
```

Koristi se metoda PDO klase prepare da se kreira novi SQL izraz i dodeli variabli stmt. U SQL stringu, vrednost parametra je označena placeholderom :name. Ovo znači da je ovaj dio upita promenjiv.

Metoda objekta statement bindParam postavlja konkretnu vrednost parametra. Prvi argument metode je ime parametra, a drugi vrednost.

Moguće je koristit i anonimne parametre, kod kojih je placeholder znak ?, ali je onda potrebno povezivati parametre po rednom broju. Prvi način je znatno čitljiviji.

TRANSAKCIJE

Glavna odlika transakcija je da će se transakcija izvršiti samo ako su se uspešno izvršile sve naredbe u transakciji

Transakcije su način da se više upita izvrši kao grupa. Glavna odlika transakcija je da će se transakcija izvršiti samo ako su se uspešno izvršile sve naredbe u transakciji. Ovo je izuzetno korisno.

Uzmimo primer transfera novca sa računa u banci. U tom slučaju imamo dva SQL upita: prvi će da umanji vrednost na prvom računu, a drugi će da poveća vrednost na nekom drugom računu.

U slučaju da ovu operaciju izvršavamo bez transakcija, u slučaju bilo kakve greške, može se desiti da novac jednostavno nestane.

```
try {
    $dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    $dbh->beginTransaction();
    $dbh->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
    $dbh->exec("insert into salarychange (id, amount, changedate)
        values (23, 50000, NOW())");
    $dbh->commit();
```

```
} catch (Exception $e) {  
    $dbh->rollBack();  
    echo "Failed: " . $e->getMessage();  
}
```

Transakcije počinju metodom beginTransaction(), a završavaju se metodom commit(). Sve SQL naredbe izmedju poziva ove dve metode su dio jedne transakcije.

Metoda rollback će da otkaže kompletну transakciju.

PDO DODATNA PODEŠAVANJA

SetAttribute metoda sa parametrom ATTR_ERRMODE podešava ponašanje biblioteke prilikom grešaka

`$ dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);`

SetAttribute metoda sa parametrom **ATTR_ERRMODE** podešava ponašanje biblioteke prilikom grešaka. Generalno se preporučuje mod sa izuzecima.

▼ Poglavlje 7

Zadaci za samostalnu rad: PHP, HTML i SQL

OPIS ZADATAKA ZA VEŽBU

Kreirati HTML stranu sa formom za unos podataka

Predviđeno vreme izrade zadatka je 45 minuta.

- Kreirati HTML stranu sa formom za unos podataka o igračkama koje se prodaju u prodavnici.
- Svaka igračka ima sledeće podatke: ID, ime, ime proizvođača, datum proizvodnje, cenu.
- Svaka uneta igračka mora se upisati u bazi koja je unapred kreirana.
- Na novoj strani kreirati tabelu u kojoj će se ispisati igračke koje su upisane u bazi.
- Igračke čija je cena veća od 2500 dinara biće ispisane crvenom bojom, ostale zelenom bojom.

✓ Poglavlje 8

Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ09

Kreirati PHP HTML stranu sa PHP skriptom

Predviđeno vreme izrade domaćeg zadatka je 90 minuta.

- Kreirati bazu podataka sa podacima o stanju robe neke prodavnice (po želji)
- Kreirati PHP stranu sa formom za unos nove robe. Forma mora biti otporna na **SQL Injection**.
- Kreirati PHP stranu sa tabelom u kojoj su ispisani podaci o robi.
- Kreirati PHP stranu na kojoj su date statistike o ceni robe: najveća, najmanja i sortirati po rastućem redosledu.
- Koristiti bazu podataka po izboru sa najmanje šest polja. Dostaviti PHP fajlove i fajl baze podataka ili SQL za kreiranje baze kao rešenje.
- Kreirane PHP strane je potrebno odvojiti posebno (svaka stranica ima svoju ulogu)

Sve datoteke arhivirati u .ZIP fajlu. Naziv arhiviranog fajla treba da bude **IT210-DZ09-ime_prezime_brojIndeksa.zip**, gde su ime, prezime i broj indeksa vaši podaci. Arhiviran fajl poslati predmetnom asistentu na e-mail.

(napomena: u Subject-u e-mejla napisati **IT210 - DZ09**).

▼ Zaključak

ZAKLJUČAK

U ovoj lekciji je opisana veza između kognitivnih principa i njihove primene na interfejse i proizvode, konceptualni uslovi za analizu ljudske interakcije sa proizvodom. Ukazano je na razlike u ograničenjima korisničkog interfejsa za veb iklašičnu aplikaciju. Opisana je veza između razvoja korisničkog interfejsa i korisnikovog poznavanja aplikacionog domena, kao i različita interaktivna okruženja sa GUI na primerima koji se ne odnose na računare.

Literatura

1. Hans Bergsten, JavaServer Pages, 3rd Edition, O'Reilly, 2003.
2. Wilbert O. Galitz, The Essential Guide to User Interface Design - An Introduction to GUI Design Principles and Techniques, Second Edition, John Wiley & Sons, Inc, 2002.



IT210 - SISTEMI IT

**INTERAKCIJA IZMEĐU ČOVEKA I
RAČUNARA - II DEO**

Lekcija 10

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 10

INTERAKCIJA IZMEĐU ČOVEKA I RAČUNARA - II DEO

- ✓ INTERAKCIJA IZMEĐU ČOVEKA I RAČUNARA - II DEO
- ✓ Poglavlje 1: TESTIRANJE UPOTREBLJIVOSTI
- ✓ Poglavlje 2: PROCES PROJEKTOVANJA I VRŠENJA TESTA
- ✓ Poglavlje 3: RAZVOJ EFEKTIVNOG INTERFEJSA
- ✓ Poglavlje 4: DOSTUPNOST
- ✓ Poglavlje 5: BIOMETRIKA
- ✓ Poglavlje 6: Pokazna vežba: Test upotrebljivosti
- ✓ Poglavlje 7: Zadatak za samostalni rad: Test upotrebljivosti
- ✓ Poglavlje 8: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilje ove lekcije da se objasni cilj i način sprovodenja testa upotrebljivosti, kao i da se razumeju osnovni stilovi interakcije i zahtevi korisničkog interfejsa

U ovoj lekciji biće obrađene sledeće teme:

- Testiranje upotrebljivosti
- Standardi upotrebljivosti
- Iskustvo korisnika
- Stilovi interakcije
- Podudarnost elemenata interfejsa sa korisničkim zahtevima
- Sindrom stresa izazvan ponavljanjem istih operacija

▼ Poglavlje 1

TESTIRANJE UPOTREBLJIVOSTI

SVRHA TESTA UPOTREBLJIVOSTI

Test upotrebljivosti služi za evaluaciju funkcionisanja korisničkog interfejsa u uslovima realnog korišćenja sistema

Test upotrebljivosti služi za evaluaciju funkcionisanja korisničkog interfejsa u uslovima realnog korišćenja sistema. Pri tom korisnik vrši specificirane zadatke, mere se performanse i rezultati se upoređuju sa prethodno zadatim ciljevima. Tokom testiranja se takođe prikupljaju informacije o problemima koji se javljaju pri korišćenju kao što su: **greške korisnika u korišćenju sistema, zbumjenost, frustracija i žalbe korisnika**. Dobra je praksa da korisnik dok radi sa sistemom glasno govori šta radi kako bi projektanti snimili njegov kognitivni model. Ako test upotrebljivosti u bilo kojoj fazi razvoja pokaže loše rezultate, potrebno je redizajnirati sistem.

Test upotrebljivosti ima i neke nedostatke. Pre svega, test upotrebljivosti je jako skup jer je dugotrajan i zahteva angažovanje više korisnika. Test se vrši u specijalnim laboratorijama i zahteva dobru opremu. Drugi nedostatak testa upotrebljivosti leži u tome da korisnici koji testiraju novi sistem ne rade dugo sa njim i ne mogu da pruže sliku o tome kako će sistem funkcionišati sa iskusnim korisnicima, odnosno koje će probleme imati korisnici koji dovoljno dugo rade sa sistemom i postanu iskusni.

Test upotrebljivosti ima dva cilja. Prvi cilj je da se uspostavi most između projektanta i korisnika. Tokom testiranja projektant uči o korisničkim ciljevima, njegovoј percepciji, načinu rezonovanja i na taj način stiče sliku o kognitivnom modelu korisnika. Za to vreme korisnik stiče prva iskustva u korišćenju nove aplikacije.

Drugi cilj testiranja je da se izvrši evaluacija novog sistema. Detektuju se potencijalni problemi i greške. Proverava se koliko korisnički interfejs odgovara korisnicima. Testiranje takođe može da posluži za upoređenje alternativnih rešenja kako bi se odlučili koje od rešenja više odgovara postavljenim ciljevima.

VAŽNOST TESTA UPOTREBLJIVOSTI

Upotrebljivost sistema se može proceniti samo kada različiti korisnici koriste sistem za sve predviđene (i nepredviđene) zadatke

Detaljni test upotrebljivosti je važan iz sledećih razloga:

- Nije moguće predvideti upotrebljivost samo na osnovu izgleda aplikacije. Upotrebljivost sistema se može proceniti samo kada različiti korisnici koriste sistem za sve predviđene (i nepredviđene) zadatke.
- Projektant i korisnik imaju različite kognitivne modele koje treba uskladiti.
- Intuicija projektanta nije uvek tačna. Tokom projektovanja programeri mnoge odluke donose na osnovu intuicije, a ne na osnovu činjenica. Tek se tokom testiranja mogu proveriti neki koncepti.
- Ne postoji prosečan korisnik. Svaki korisnik je specifičan po željama, osobinama percepције, motornim sposobnostima, intelektualnim mogućnostima, sklonostima ka upotrebi pojedinih ulaznih uređaja itd.
- Standardi za projektovanje i smernice ne garantuju upotrebljivost. Standarde i smernice je dobro poštovati, ali svaki sistem ima specifičnosti koje nisu njima predviđene.
- Sistemi razvijani od strane više projektanata su po pravilu nekonzistentni jer nema „prosečnog projektanta“.
- Može se doći do kvalitetnijeg rešenja u odnosu na konkurentska.
- Test upotrebljivosti ne treba raditi samo nakon završetka razvoja novog sistema. Umesto toga, sa prvim testovima treba početi u ranoj fazi projektovanja i nastaviti sa stalnim testovima tokom celog procesa razvoja.

▼ Poglavlje 2

PROCES PROJEKTOVANJA I VRŠENJA TESTA

PLAN TESTA

Prvu stvar koju treba definisati prilikom planiranja testa je obim testa koji će se vršiti.

Test upotrebljivosti zahteva da se projektuje plan testiranja, da se izaberu učesnici, izvrši test i analiziraju dobijeni rezultati.

Plan testa

Za testiranje grafičkog korisničkog interfejsa je potrebno napraviti plan testa koji se dobija kao odgovor na pet pitanja. Prvu stvar koju treba definisati prilikom planiranja testa je obim testa koji će se vršiti. Da bi se definisao obim testa treba specificirati:

- **U kojim** će se sve **fazama** razvoja vršiti testiranje
- **Koliko dugo** će se vršiti testiranje - ovo može da bude od nekoliko dana pa do cele godine
- Potrebna **finansijska sredstva** za testiranje - ovo može da bude od 1 do 10% cene projekta
- **Očekivani broj korisnika** budućeg sistema i kritičnost interfejsa za rad aplikacije

SVRHA TESTA I METODOLOGIJA

Svrha testa određuje šta sve treba testirati, a metodologija na koji način i uz koja ograničenja

Svrha testa

Drugo pitanje na koje treba dati odgovor prilikom planiranja testa je svrha testa. Ovde treba definisati šta će se sve testirati, što uobičajeno obuhvata:

- ciljne performanse sistema
- šta se od testa očekuje da postigne u učenju sistema
- upotrebu ekrana, estetika, prepoznavanje kontrola
- navigaciju u sistemu, metod navigacije, meniji, dugmad, ikone
- efikasnost operacija, koliko koraka je potrebno da bi se izvršio neki zadatak, koliko je brz odziv sistema, oblik odziva sistema.

Metodologija

Treće pitanje se odnosi na metodologiju koja će se koristiti za vršenje testa. Ovde treba specificirati:

- tip testa koji se vrši
- ograničenja testa obzirom na stepen dovršenosti sistema
- učesnike iz razvojnog tima

OPREMA I LOKACIJA ZA TESTIRANJE, SCENARIO TESTA I TESTIRANJE VEB SAJTA

Potrebno je odrediti specifikaciju lokacije, opreme i uslova pod kojim će se vršiti test, takođe, koji će se zadaci ispunjavati

Oprema i lokacija za testiranje

Četvrto pitanje se odnosi na opremu i lokaciju za testiranje. Odgovor na ovo pitanje podrazumeva specifikaciju:

- lokaciju na kojoj će se test izvršiti (kod korisnika, u laboratoriji za testiranje upotrebljivosti, u prostorijama razvojnog tima i slično)
- opremu koja će biti korišćena za testiranje (računarski sistemi, video kamere, jednosmerna ogledala)
- uslova pod kojim će se test vršiti (osvetljenje, buka, temperatura, broj ljudi u istoj prostoriji).

Scenario testa

Poslednje pitanje plana testa se odnosi na scenario testa. On se sastoji iz zadataka koji korisnici treba da obave u cilju testiranja.

Testiranje veb sajta

Testiranje veb sajta je isto kao i testiranje grafičkog korisničkog interfejsa. Međutim, kako se veb interfejs razlikuje u nekim elementima od grafičkog, potrebno je testirati i:

- sve veb čitače, servere i monitore koji će se koristiti rad pri različitim brzinama prenosa podataka kroz mrežu navigaciju i ispravnost linkova
- vizualni stil strana
- korektnost sadržaja čitljivost sadržaja
- razumljivost grafika i ikona
- štampanje veb strana
- kompatibilnost sa smernicama za dostupnost.

✓ 2.1 UČESNICI U TESTU

IZBOR UČESNIKA TESTIRANJA

Treba izabrati učesnike sa odgovarajućom kvalifikacijom za posao kome je sistem namenjen

Izbor pravih osoba za testiranje je kritično za uspeh testa. Treba izabrati učesnike sa odgovarajućom kvalifikacijom za posao kome je sistem namenjen. Treba izabrati korisnike koji odgovaraju celom spektru korisnika obzirom na godine, pol i iskustvo. Istraživanja su pokazala [2] da je optimalan broj učesnika u testu aplikacije 5, jer se sa tim brojem detektuje oko 85% problema u sistemu. Pri testiranju veb sajtova stvari mogu biti nešto drugačije jer su korisnici nedefinisani. Ako se očekuje da će nekom sajtu pristupati različite grupe ljudi, na primer deca, roditelji i stari ljudi, čije je ponašanje različito, preporučuje se uključivanje po 3 osobe iz svake grupe u test. Ne treba zaboraviti i posebnu grupu za ljude sa posebnim potrebama.

✓ 2.2 IZVOĐENJE TESTA

IZVOĐENJE TESTA I PRIKUPLJANJE PODATAKA

Centralna faza testiranja upotrebljivosti je izvođenje samog testa i prikupljanje podataka. Ona se sastoji od niza akcija koje treba izvesti pre, za vreme i posle testiranja

Centralna faza testiranja upotrebljivosti je izvođenje samog testa i prikupljanje podataka. Ona se sastoji od niza akcija koje treba izvesti pre, za vreme i posle testiranja. Ove akcije su:

1. Pre početka testa

- Objasniti da je cilj testiranje sistema, a ne učesnika
- Objasniti kako će se rezultati testa koristiti za dalje poboljšanje sistema
- Ako učesnici potpisuju saglasnost, objasniti detalje dokumenta
- Ukoliko će se rad korisnika snimati, zamoliti korisnike da razmišljaju glasno
- Odgovoriti na sva pitanja učesnika u testu
- Pripremiti tabelu za unos podataka koji se snimaju tokom testiranja
- Pripremiti upitnik koga učesnici popunjavaju nakon testa

2. Tokom testa

- Minimizirati broj ljudi koji će komunicirati sa ljudima tokom testa
- Ako je potrebno da posmatrači budu u prostoriji, ograničiti njihov broj na dva do tri
- U tabeli za unos podataka upisati: Vreme za izvršenje zadatka

- greške učinjene pri izvršenju zadataka, neočekivane akcije korisnika, korišćene i nekorišćene funkcije sistema, teškoće koje je korisnik imao pri korišćenju funkcija sistema, sistemske greške
- Notirati tehnike koje su korisnici koristili da bi rešili probleme kad su naišli na teškoće u korišćenju
- Ako tokom testa korisnici razmišljaju glasno, snimiti njihove pretpostavke i zaključke
- Snimiti proces testiranja video kamerom
- Ne prekidati učesnike testa, osim ako to nije absolutno neophodno
- Ako učesnici traže pomoć - dati im ohrabrenje ili savet, dati im najpre opšte, pa ako je potrebno konkretnе savete, zapisati koji su saveti dati
- Pažljivo pratiti tipične znake stresa kod korisnika, kao što su: korisnik duže sedi ne radeći ništa, okrivljuje sebe ili druge zbog problema, prevrće dokumente, ne čitajući ih, obezbediti kratke prekide kada je to potrebno

3. Nakon testa

- Održati završni intervju sa učesnicima, i reći korisnicima šta se saznalo tokom testa
- Organizovati popunjavanje upitnika u kome učesnici ocenjuju sistem i izvršene zadatke
- Čuvati privatnost korisnika i snimljeni materijal ne prikazivati neovlašćenim licima.

PRIMER - PRE POČETKA TESTA (SA MODERATOROM)

*Primer kako pozdraviti i pripremiti učesnike za njihovu sesiju testiranja.
Objasniti svrhu i objasniti uloge tokom sesije.*

Dobrodošlica i svrha

Hvala što ste došli danas. Hteo sam da vam dam par smernica šta ćete danas raditi i da vam pre početka pružim mogućnost da postavite pitanja koja možda imate.

Danas od vas tražimo da ocenite veb sajt i da na tom sajtu odradite nekoliko zadataka. Naš cilj je da vidimo koliko je jednostavno ili teško koristiti sajt.

Uloga moderatora

Ja sam ovde da zabeležim vaše reakcije i komentare koje ćete imati tokom pregledanja današnjeg sajta. U sali do nas je moj kolega koji mi pomaže da sve to zabeležimo i koji će takođe posmatrati vaše današnje reakcije.

Tokom ove sesije, zamolio bih vas da razmišljate na glas dok obavljate sve zadatke. Ja neću moći da vam dajem sugestije i da vam pružam pomoć, ali s vremenom na vreme ću vam prići i zamoliti vas da pojasnите šta ste rekli ili da mi kažete koju ste informaciju pokušali da nađete ili šta očekujete da se dogodi.

Uloga testera

- Danas ću vas zamoliti da potražite neke informacije na sajtu i da mi kažete koliko vam je bilo lako ili teško da pronađete te informacije. Cilj ovih aktivnosti je da vidimo koliko je teško ili lako ljudima da koriste sajt.
- Nema tačnog ili netačnog odgovora. Ako imate nekih pitanja, komentara ili ako ste zbumeni dok radite, molim vas da mi kažete.
- Ako mislite da ne možete da izvršite zadatak do kraja ili ste se blokirali i ne znate kako da nastavite, molim vas da mi kažete. Ja ću vas pitati šta bi ste uradili u stvarnom životu i zatim ću vas ili usmeriti kako da nastavite ili vam reći da pređete na sledeći zadatak.

- Dok budete koristili sajt, molim vas da to radite kao što biste radili kod kuće ili na poslu. Zamolio bih vas da prolazite kroz zadatke na osnovu onoga što vidite na ekranu, ali ako dođete u situaciju da niste sigurni gde i kako da nešto pronađete, slobodno koristite opciju "Pretraga".
 - Mi ćemo zabeležiti ovu sesiju za buduće analize. Snimamo vaše lice, vaš glas i ono što vidite na ekranu. Vaše ime neće biti povezano sa podacima ili rezultatima iz ove evaluacije.
 - Možda ću vam tokom testiranja postavljati i druga pitanja, a na kraju ćemo imati vremena za vaša pitanja.
- Imate li nekih pitanja pre nego što počnemo?

PRIMER - PRE POČETKA TESTA (BEZ MODERATORA)

Primer kako pozdraviti i pripremiti učesnike za njihovu sesiju testiranja

Dobrodošlica i svrha

Hvala što ste prihvatali da učestvujete u evaluaciji veb sajta. Danas tražimo od vas da budete evaluator ovog veb sajta i da izvršite na njemu nekoliko zadataka. Naš cilj je da vidimo koliko je jednostavno ili teško koristiti ovaj sajt. Zabeležićemo vaše reakcije i mišljenja, a možda ćemo vas i zamoliti da objasnite ono što izgovorite.

Uloga moderatora

Ja sam ovde da zabeležim vaše reakcije i komentare tokom korišćenja veb sajta.

U sali do nas je moj kolega koji mi pomaže da sve to zabeležimo i koji će takođe posmatrati vaše današnje reakcije.

Tokom sesije neću moći da vam pomognem i da vam dajem sugestije. Međutim, s vremenom na vreme ću vas možda zamoliti da objasnite nešto što ste kazali ili uradili.

Uloga testera

Zamoliću vas da tražite određene informacije na ovom sajtu, kako bi otkrili da li nailazite na neke poteškoće. Ovo ćemo uraditi tako što ćemo vam dati nekoliko scenarija ili zadataka koje ćete obaviti na sajtu. Takođe ćemo vam postaviti pitanja na kraju sesije, kako bi čuli koliko ste bili zadovoljni ovim sajtom.

Stvari koje treba imati na umu

Evo nekih stvari koje bi trebalo da znate o vašem učešću na ovom testu:

- Današnji test je da ocenimo koliko smo napravili ovaj sajt lakim za korišćenje
- Nema tačnog ili netačnog odgovora. Ako imate nekih pitanja, komentara ili ako ste zbumjeni dok radite, molim vas da mi kažete.
- Ako mislite da ne možete da izvršite zadatak do kraja ili ste se blokirali i ne znate kako da nastavite, molim vas da mi kažete. Ja ću vas pitati šta bi ste uradili u stvarnom životu i zatim ću vas ili usmeriti kako da nastavite ili vam reći da pređete na sledeći zadatak.
- Mi ćemo zabeležiti ovu sesiju za buduće analize. Vaše ime se neće pojavit na izveštajima sa ovog testa. Molim vas da popunite formular za saglasnost. (ako to već nije urađeno)
- Na kraju, dok budete koristili sajt, molim vas da to radite kao što biste radili kod kuće ili na poslu. Zamolio bih vas da kada tražite neke informacije, da to uradite što brže i preciznije možete.

Da li imate nekih pitanja pre nego što počnemo?

PRIMER IZVOĐENJA TESTA UPOTREBLJIVOSTI - VIDEO

Intervju vođen tokom testa upotrebljivosti - primer

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ 2.3 ANALIZA REZULTATA TESTA

KATEGORISANJE PROBLEMA

Nakon izvršenog testa probleme je potrebno sortirati po stepenu prioriteta njihovog rešavanja

Nakon izvršenog testa potrebno je prikupiti sve podatke i proučiti sve probleme koje su korisnici imali tokom korišćenja sistema i notirati njihovu frekvenciju. Probleme je potrebno sortirati po stepenu prioriteta njihovog rešavanja. Za kategorisanje problema se može koristiti stepen prioriteta prikazan u tabeli 1.

Stepen Prioriteta	Vrsta problema
0	Prijavljeni problem nije problem upotrebljivosti
1	Kozmetički problem. ne mora da se rešava sve dok projektantski tim nema viška slobodnog vremena
2	Minorni problem upotrebljivosti. Otklanjanju treba dati nizak stepen prioriteta
3	Veliki problem upotrebljivosti. Vrlo je važno da se otkloni i treba mu dati visok
4	Neophodno da se reši pre nego što se program objavi ili se web sajt publikuje.

Slika 2.1.1 Tabela-1 Kategorizacija problema koji se primete tokom testa upotrebljivosti [Izvor: Autor]

Za uočene probleme treba predložiti strategiju rešavanja i konkretna rešenja. Na osnovu toga je potrebno izvršiti potrebne izmene sistema. Na kraju ovog postupka se ponovo vrši test i proces treba ponavljati sve dok upotrebljivost sistema ne bude dovedena u stanje definisano zahtevima.

✓ 2.4 STANDARDI UPOTREBLJIVOSTI

INTERNACIONALNI STANDARDI

Postoji više internacionalnih standarda koji se odnose na upotrebljivost i interakciju čovek-računar

Postoji više **internacionalnih standarda** koji se odnose na upotrebljivost i interakciju čovek-računar. Standardi su razvijeni od strane **International Organisation for Standardisation (ISO)** i **International Electrotechnical Commission (IEC)**. Ovi standardi se mogu podeliti na grupe standarda koji se odnose na:

- **korišćenje proizvoda** (efektivnost, efikasnost i zadovoljstvo u posebnim slučajevima upotrebe) korisnički interfejs i interakciju
- **procese koji se koriste prilikom razvoja proizvoda**
- **sposobnost organizacije da primeni korisnički orijentisan proces projektovanja**

Standardi se dalje mogu podeliti na principe i preporuke i specifikacije. U narednoj tabeli su prikazani važeći internacionalni standardi vezani za upotrebljivost.

Korišćenje u kontekstu	Principi preporuka	Specifikacija
	ISO/IEC 9126-1: Software Engineering - Product quality - Part 1: Quality model	ISO 20282: Usability of everyday products
	ISO/IEC TR 9126-4: Software Engineering - Product quality - Part 4: Quality in use metrics	
	ISO 9241-11: Guidance on Usability	
Interfejs i interakcija	ISO/IEC TR 9126-2: Software Engineering - Product quality - Part 2 External metrics	ISO 9241: Ergonomic requirements for offices work with visual display terminals. Parts 3-9
	ISO/IEC TR 9126-3: Software Engineering - Product quality - Part 3 Internal metrics	ISO/IEC 10741-1: Dialogue interaction - Cursor control for text editing
	ISO 9241: Ergonomic requirements for offices work with visual display terminals. Parts 10-17	ISO/IEC 11581: Icon symbols and functions
	ISO 11084: Ergonomic design of control centres	ISO 13408: Ergonomic requirements for work with visual displays based on flat panel displays
	ISO 14916: Software ergonomics for multimedia user interfaces	ISO/IEC 14754: Pen-based interfaces - Common Gestures for text editing with pen-based systems
	IEC TR 61997: Guidelines for the user interface in multimedia equipment for general purpose use	ISO/IEC 18021: Information Technology - User interface for mobile tools
Dokumentacija	ISO/IEC 18019: Guidelines for the design and preparation of software user documentation	ISO/IEC 18788: Ergonomic requirements and measurement techniques for electronicvisual displays
Proces razvoja	ISO 13407: Human-centred design processes for interactive systems	ISO/IEC 15931: Software user documentation process
	ISO TR 16982: Usability methods supporting human centred design	ISO/IEC 14598: Information Technology - Evaluation of Software Products
Sposobnost (Capability)	ISO TR 19529: Ergonomics of human-system interaction - Human-centred lifecycle process descriptions	
Drugo	ISO 9241-1: Part 1: General Introduction	
	ISO 9241-2: Part 2: Guidance on task requirements	
	ISO 10076-1: Ergonomic principles related to mental workload - General terms and definitions	
	ISO/DTS 16071: Guidance on accessibility for human-computer interfaces	

Slika 2.2.1 Tabela-1 Spisak internacionalnih standarda [Izvor: Autor]

DEFINICIJA UPOTREBLJIVOSTI PREMA STANDARDIMA

ISO 9241-11 objašnjava kako da se identifikuju informacije koje je potrebno uzeti u obzir kada se specifičuje i evaluira upotrebljivost u funkciji mera korisnikovih performansi i zadovoljstva

Jedan od najvažnijih standarda koji definiše upotrebljivost je ISO 9241-11: Smernice za upotrebljivost iz 1998. godine. Ovaj standard, koji je samo jedan iz serije 9241 standarda, definiše upotrebljivost kao: mera efektivnosti, efikasnosti i zadovoljstva prilikom korišćenja proizvoda od strane specifikovanog korisnika u cilju postizanja specifikovanih zadataka u specifikovanom kontekstu upotrebe.

ISO 9241-11 objašnjava kako da se identifikuju informacije koje je potrebno uzeti u obzir kada se specifičuje i evaluira upotrebljivost u funkciji mera korisnikovih performansi i zadovoljstva. Pored toga, date su i smernice kako treba opisati kontekst upotreba proizvoda i mere upotrebljivosti na eksplicitan način.

ISO/IEC 9126

Prema ISO/IEC 9126 upotrebljivost je set atributa koji se odnose na napor pri upotrebi softverskog proizvoda i na individualnu ocenu takve upotrebe od strane određenog seta korisnik

Drugi standard koji se odnosi na upotrebljivost je ISO/IEC 9126: Evaluacija softverskih proizvoda Karakteristike kvaliteta i smernice za njihovu upotrebu iz 1991. godine. U zajednici softverskog inženjerstva termin upotrebljivost je mnogo bliže vezan za korisnički interfejs. ISO/IEC 9126, koji je razvijen kao standard softverstvog inženjerstva, definiše upotrebljivost kao relativno nezavisno doprinos kvalitetu softverskog proizvoda koji potiče od korisničkog interfejsa i evaluacije.

Prema ISO/IEC 9126 upotrebljivost je set atributa koji se odnose na napor pri upotrebi softverskog proizvoda i na individualnu ocenu takve upotrebe od strane određenog seta korisnika.

Ovaj standard je 2000. godine zamenjen sa četiri nova delova standarda. Jedan od delova koji je još u fazi finalnog nacrtta (engl. Final Draft International Standard - FDIS) je ISO/IEC FDIS 9126-1: Softversko inženjerstvo - Kvalitet proizvoda - Deo 1: Model kvaliteta. On opisuje šest komponenata kvaliteta softvera koje su važne tokom razvoja proizvoda: funkcionalnost, pouzdanost, upotrebljivost, efikasnost, održavanje i prenosivost. Jedna od ovih komponenata, upotrebljivost se u ovom predlogu standarda definiše kao sposobnost softverskog proizvoda da bude razumljiv, atraktivan, lak za učenje i upotrebu za korisnike kada se koristi pod specifikovanim uslovima.

Pored toga ISO/IEC 9126-1 definiše i mnogo širi termin „**kvalitet prilikom korišćenja**“ kao sposobnost softverskog proizvoda da omogući specifikovanim korisnicima da postignu specifikovane ciljeve sa efektivnošću, produktivnošću, sigurnošću i zadovoljstvom u specifikovanom kontekstu upotrebe. Kvalitet prilikom korišćenja je kombinovani efekt svih

šest komponenata kvaliteta softvera kada je proizvod u upotrebi. Ukupni cilj je da se postigne kvalitet prilikom korišćenja kako za krajnje korisnike kao i za korisnike koji pružaju podršku za softverski proizvod. Funkcionalnost, pouzdanost, efikasnost i upotrebljivost određuju kvalitet prilikom korišćenja za krajnjeg korisnika. Održavanje i portabilnost određuju kvalitet prilikom korišćenja za korisnike koji pružaju podršku.

Drugi delovi standarda ISO/IEC 9126 definišu metriku za upotrebljivost i kvalitet prilikom korišćenja.

▼ Poglavlje 3

RAZVOJ EFEKTIVNOG INTERFEJSA

ISKUSTVO KORISNIKA

Korisnici računarskih sistema imaju čitav niz karakteristika koje mogu da utiču na način kako oni koriste sistem

Korisnici računarskih sistema imaju čitav niz karakteristika koje mogu da utiču na način kako oni koriste sistem. Jedna grupa ovih karakteristika je naročito važna sa aspekta razvoja efektivnog interfejsa čovek-računar. Ova grupa karakteristika se odnosi na znanje i iskustvo korisnika vezano za sistem sa kojim radi. U ovu grupu karakteristika spadaju:

- **Poznavanje sistema**
- **Poznavanje aplikacije**
- **Poznavanje zadatka**
- **Poznavanje i korišćenje drugih sistema**
- **Obrazovanje**
- **Poznavanje jezika i kultura**

Projektanti interfejsa treba da poznaju ove karakteristike korisnika kako bi projektovali efektivni interfejs.

POZNAVANJE SISTEMA

Poznavanje sistema obuhvata osnove operativnog sistema, razumevanje principa na kojima funkcioniše računarski sistem i iskustvu u korišćenju ulazno izlaznih uređaja

Pod poznavanjem sistema se podrazumeva osnovno znanje i iskustvo vezano za korišćenje računarskih sistema. Ovo znanje obuhvata osnove operativnog sistema, razumevanje principa na kojima funkcioniše računarski sistem i iskustvo u korišćenju ulazno izlaznih uređaja. Ovakvo znanje se stiče tokom školovanja, ali i posle dužeg korišćenja računarskih sistema.

Korisnici se prema poznavanju sistema dele na potpuno neiskusne, početnike, iskusne i eksperte. Nema tačne definicije ovih kategorija. Pored toga iskusan korisnik jednog sistema može biti početnik za drugu vrstu sistema.

Iskustvo je pokazalo da postoje razlike u upotrebljivosti istog sistema koju osećaju korisnici sa različitim stepenom poznavanja sistema. Ono što je lako za neiskusne korisnike i početnike ne mora da bude lako za eksperte i obratno.

Za neiskusne korisnike i početnike je karakteristično da:

- zavise od sistemskih funkcija koje pomažu memoriji prepoznavanja kao što su: meniji, promptne informacije, prozori sa instrukcijama i uputstvima,
- zahtevaju ograničeni vokabular, mali broj mogućnosti, mogu da rešavaju jednostavne zadatke i potreban im je informativni odziv sistema, imaju veću želju za učenjem i žele da vežbom dostignu status eksperta.

Rasprostranjeno je mišljenje da su osobe koje se teško privikavaju za rad sa novim sistemom smatraju ograničenim i sva krivica pada na njih. Međutim, treba imati na umu da su ljudi različiti i da se ne mogu redizajnirati. Različiti ljudi na različit način doživljavaju sistem. S druge strane, redizajniranje korisničkog interfejsa je moguće i samo se njegovim poboljšanjem računarski sistemi mogu učiniti upotrebljivijim za sve korisnike.

Za iskusne korisnike sistema je karakteristično da:

- **rad zasnivaju na prethodnom iskustvu i intuiciji**
- **očekuju brzi odziv sistema**
- **imaju potrebu za manje informativnim odzivom**
- **traže efikasnost koju očekuju kroz skraćivanje obuke, kratke informacije, novi vokabular i efikasniji korisnički interfejs.**

Što se veb sistema tiče, početnici očekuju pregled mogućnosti, komandnu dugmad za izbor akcija, vođenje kroz zadatak. Iskusnim korisnicima više odgovaraju uredene strukture, očigledni linkovi, reverzibilnost i sigurnost u pronalaženju informacija. Eksperti pri korišćenju veb sistema očekuju jednostavnu navigaciju, kompaktne i stručne informacije, brzo učitavanje stranica, veliki broj servisa i mogućnost da promene i prilagode veb strane svojim potrebama.

RAZLIKA IZMEĐU KORIŠĆENJA SISTEMSKOG I VEB INTERFEJSA

Kako korisnik upoznaju jednu aplikaciju i postaje ekspert za nju, tako se menja i njegov odnos prema korisničkom interfejsu

Postoji razlika između korišćenja sistemskog i veb interfejsa. Sistemski interfejs se koristi svakodnevno i posle dužeg vremena svi korisnici postanu eksperti. S druge strane, veb sistemi su različiti od sajta do sajta, a i na jednom sajtu se često menjaju. Zbog toga ima mnogo manje eksperata za interfejs veb sistema nego za interfejse računarskih sistema.

Microsoft je identifikovao tipične probleme koje imaju neiskusni korisnici i početnici u radu sa sistemom. Prema njihovom istraživanju početnici imaju probleme sa:

- dvostrukim klikom na mišu i prevlačenjem
- prozorima koji se preklapaju, a ponekad i potpuno skrivaju prozor koji se nalazi ispod
- organizacijom direktorijuma i datoteka

Eksperti koji dobro poznaju sistem, teže ka efikasnijem radu očekuju od korisničkog interfejsa da im omogući:

- startovanje akcija dvostrukim klikom
- **pop-up** menije
- razdvojive menije prečice (engl. **shortcuts**)
- komandnu liniju.
- Poznavanje aplikacije

Kako korisnik upoznaju jednu aplikaciju i postaje ekspert za nju, tako se menja i njegov odnos prema korisničkom interfejsu. Na početku korisnik očekuje više informacija od aplikacija i vođenje kroz zadatke. Mnogi novi termini su mu nepoznati i očekuje objašnjenje. Kasnije, kada postane familijaran sa aplikacijom ove informacije počinju da ga opterećuju jer su mu nepotrebne. Zato korisnički interfejs treba učiniti prilagodljivim kako bi korisnik tokom vremena mogao da menja stil interfejsa u skladu sa svojim potrebama.

KORISNIČKO ISKUSTVO (USER EXPERIENCE)

“User experience’ encompasses all aspects of the end-user’s interaction with the company, its services, and its products.”

Termin "korisničko iskustvo" (engl. **user experience** (UX)) je uveo kognitivni naučnik Don Norman u ranim 1990-im dok je bio potpredsednik Advanced Technology Group u Apple-u.

Evo kako je formalno definiše:

“User experience’ encompasses all aspects of the end-user’s interaction with the company, its services, and its products.”

U email-u u kome Norman piše o poreklu ovog termina, napisao je:

“I invented the term because I thought Human Interface and usability were too narrow: I wanted to cover all aspects of the person’s experience with a system, including industrial design, graphics, the interface, the physical interaction, and the manual.

Since then, the term has spread widely, so much so that it is starting to lose its meaning.”

UX nije ograničen na vizuelni interfejs proizvoda. To je koncept koji ima više dimenzija uključujući:

- **Proces preko koga je korisnik saznao o proizvodu**
- **Redosled akcija koje korisnik obavlja da bi došao do željenog cilja**
- **Misli i osećanja koja se javljaju dok korisnik pokuša da ostvari svoj zadatak**
- **Utisci o sprovedenoj interakciji sa proizvodom/uslugom u celini.**

RAZLIKA IZMEĐU UI I UX

Korisnički interfejs predstavlja niz vizuelnih elemenata koje koristimo za interakciju sa uređajem. Korisničko iskustvo je unutrašnji doživljaj tokom interakcije sa proizvodom/uslugom

Na najosnovnijem nivou, korisnički interfejs (engl. **user interface (UI)**) predstavlja niz ekrana, stranica i vizuelnih elemenata, kao što su dugmad i ikone, koje koristimo za interakciju sa uređajem.



Slika 3.1.1 Korisnički interfejs

[Izvor: https://media-assets-01.thedrum.com/cache/images/thedrum-prod/s3-news-tmp-90538-1_0i-ozqrnd3f-s1f6tyubea--2x1--940.png]

Korisničko iskustvo (engl. **user experience (UX)**), sa druge strane, je unutrašnji doživljaj koji osoba dok ima interakciju sa nekim proizvodom i uslugom kompanije.



Slika 3.1.2 Korisničko iskustvo

[Izvor: <https://www.usertesting.com/sites/default/files/inline-images/UXrocks.jpg>]

Don Norman i Jakob Nielsen su sumirali razliku između UI i UX kada su rekli:

"It's important to distinguish the total user experience from the user interface (UI), even though the UI is obviously an extremely important part of the design.

As an example, consider a website with movie reviews. Even if the UI for finding a film is perfect, the UX will be poor for a user who wants information about a small independent release if the underlying database only contains movies from the major studios."

Google je još jedan dobar primer sa jednostavnim interfejsom. Kada otvorite stranicu Google-a, jedva da postoji UI, samo logo, forma za unos reči za pretragu i nekoliko dugmića, kao i rezultati pretrage strana. Ali kada otkucate nešto u tom polju za pretragu, dobijate pristup velikom skupu informacija i znanja za manje od jedne sekunde.

Sada zamislite da svaki put kada radite pretragu na Google-u da je potrebno 15 sekundi da se dobiju rezultati. Čak i ako interfejs ostao isti, vaše iskustvo sa Google-om će biti drastično drugačije.

ŠTA SU TALENTOVANI LJUDI U IT INDUSTRiji REKLI ZA RAZLIKU IZMEĐU UI I UX

Različiti ljudi imaju različite stavove na temu objašnjenja razlike između UI i UX

"Start with a problem we'd like to solve. UX design is focused on anything that affects the user's journey to solve that problem, positive or negative, both on-screen and off. UI design is focused on how the product's surfaces look and function. The user interface is only piece of that journey. I like the restaurant analogy I've heard others use: UI is the table, chair, plate, glass, and utensils. UX is everything from the food, to the service, parking, lighting and music."

- Ken Norton – Partner at Google Ventures, Ex-Product Manager at Google

A UX designer is concerned with the conceptual aspects of the design process, leaving the UI designer to focus on the more tangible elements

- Andy Budd – Co-founder of Clearleft, Founder of UX London

There is no difference between UX and UI design because they are two things that aren't comparable to each other

- Craig Morrison – Head of Product at RecordSetter, Founder of Usability Hour

UI is generally about visual design and information design around screens. UX is about the complete experience, and it may not be even about the screen

- Patrick Neeman – Director of Product Design at Apptio, Founder of Usability Counts

UI is focused on the product, a series of snapshots in time. UX focuses on the user and their journey through the product

- Scott Jenson – Product Strategist at Google

Dodatno čitanje:

<https://www.usertesting.com/blog/2016/04/27/ui-vs-ux/>

POZNAVANJE ZADATKA

Tok izvršenja programa treba da odgovara radnom toku na koji su korisnici navikli i koji je propisan poslovnim pravilima organizacije

Od korisnika aplikacije se sa pravom očekuje da su iz struke i da poznaju svoj posao. Aplikacije uobičajeno automatizuju zadatke koji su se pre toga obavljali ručno. Korisnički interfejs treba da ima grafičke simbole koji odgovaraju alatima koje korisnici koriste u neautomatizovanom radu. Forme za unos podataka treba da odgovaraju formama koje su korisnici ranije popunjavali ručno. Tok izvršenja programa treba da odgovara radnom toku na koji su korisnici

navikli i koji je propisan poslovnim pravilima organizacije. Ako su ovi uslovi ispoštovani, svaki korisnik koji dobro poznaje svoje zadatke i ima iskustva u radu će se brzo privići na novu aplikaciju.

Problemi nastaju sa korisnicima koji su novi u poslu koji automatizuje aplikacija. Pred njima je dvostruka barijera: nemaju dovoljno znanja da obavljaju zadatke i ne poznaju rad sa aplikacijom. Korisnički interfejs za ovakve korisnike treba da bude vrlo informativan, a korisnik treba da bude vođen kroz aplikaciju u unapred definisanim koracima koji odgovaraju propisanom radnom toku. Razume se da je ovakav interfejs potpuno neodgovarajući za iskusne korisnike, pa je potrebno imati mogućnost da korisnici biraju odgovarajući interfejs.

POZNAVANJE I KORIŠĆENJE DRUGIH SISTEMA

Aplikacije moraju biti u skladu sa postojećim kognitivnim modelom korisnika

U praksi se dešava da jedan korisnik na svom radnom mestu koristi više različitih aplikacija jer obavlja više različitih zadataka. Pored toga, organizacija je ponekad prinuđena da zameni postojeću aplikaciju novom. U oba slučaja se zahteva da aplikacije budu kompatibilne sa postojećim kognitivnim modelom korisnika. Što je veća kompatibilnost, vreme prihvatanja nove aplikacije će biti kraće, a upotrebljivost aplikacije veća.

OBRAZOVANJE, JEZIK I KULTURA

Zbog različitog nivoa obrazovanja korisnički interfejs treba projektovati tako da bude razumljiv svim korisnicima

Obrazovanje

Korisnici aplikacija mogu da imaju različito obrazovanje. Iste zadatke u nekoj organizaciji mogu koristiti i osobe sa akademskim i srednjoškolskim obrazovanjem. Zbog toga tekstualni deo korisničkog interfejsa, vokabular i gramatičke strukture treba projektovati tako da odgovaraju i budu razumljive svim korisnicima.

Poznavanje jezika i kultura

Korisnički interfejs se uglavnom bazira na tekstu i grafičkim elementima u obliku ikona. Ukoliko tekstualni deo interfejsa nije na maternjem jeziku korisnika ili na jeziku kojim on dobro vlada upotrebljivost aplikacije će biti drastično umanjena. Slično će se dogoditi i ukoliko simboli koji su predstavljeni ikonama ne odgovaraju kulturi korisnika.

DODATNI MATERIJAL

Dodatni materijal sa primerima koji je interesantan za dodatno čitanje

- Facebook's Product Design Director Explains One Of Its Biggest UX Changes In Years - Not everything in life is Likable.

<http://www.fastcodesign.com/3057113/facebook-product-design-director-explains-one-of-its-biggest-ux-changes-in-years>

- The Definition of User Experience, by Don Norman and Jakob Nielsen

<https://www.nngroup.com/articles/definition-user-experience/>

UX DIZAJN - VIDEO

What is UX Design? The Definition of User Experience Design

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ 3.1 STILOVI INTERAKCIJE

ŠTA SE PODRAZUMEVA POD STILOM INTERAKCIJE?

Stilovi interakcije mogu biti: komandni jezik, popunjavanje forme, izbor menija, direktna i indirektna manipulacija

Pod stilom interakcije se podrazumeva način na koji korisnik komunicira ili vrši na drugi način interakciju sa računarskim sistemom. Inicijalno je ovaj koncept bio vezan samo za interakciju čovek-računar, ali se sada koristi i za interakciju čoveka sa drugim uređajima, kao što je na primer mobilni telefon.

Različiti korisnici preferiraju različite stlove interakcije, što zavisi od njihovog kognitivnog modela, stepena poznавања računarskog sistema, iskustva i afiniteta. Međutim, računarski sistemi omogućuju samo nekoliko mehanizma za interakciju pa se stili interakcije svode na:

- komandni jezik
- popunjavanje formi
- izbor iz menija
- direktnu manipulaciju i
- indirektnu manipulaciju.

Razume se da postoji i mogućnost kombinacije navedenih stilova. Neke zadatke je lakše obavljati jednim, a neke drugim stilom interakcije, pa korisnici uglavnom kombinuju one stlove koji su im na raspolaganju. U svakom slučaju upotrebljivost jednog sistema se bitno poboljšava ako nudi veći broj stlova interakcije.

KOMANDNI JEZIK

Komandni jezik znatno opterećuje kognitivni sistem korisnika jer je interakcija zasnovana na sećanju, a ne na memoriji prepoznavanja

Komandni jezik je najraniji oblik stila interakcije, i u doba njegovog pojavljivanja i jedini. Pre toga nije bilo prave interakcije između čoveka i računara. Ovaj stil interakcije se i danas koristi uglavnom kod Unix familije operativnih sistema. Interakcija sa računarom preko komandnog prompta (engl. **command prompt**) je najomiljeniji način interakcije za eksperte koji odlično poznaju sistem jer se sa minimumom korisničkog dejstva može izvršiti čitav niz akcija.

Komandni jezik znatno opterećuje kognitivni sistem korisnika jer je interakcija zasnovana na sećanju, a ne na memoriji prepoznavanja. Komande, kao i mnogi opcioni parametri moraju da budu naučeni. Činjenica da su mnoge komande skraćenice punog naziva komande ne čine zadatak laksim. Generalno, komande se ne uče lako i brzo.

Prednosti interakcionog stila zasnovanog na komandnom jeziku su:

- fleksibilnost
- pogodniji je za eksperte
- podržava kreiranje korisnički definisanih skriptova i makroa
- podesan je za rad u mrežnim uslovima čak i pri vrlo maloj brzini prenosa informacija.
- Nedostaci interakcionog stila zasnovanog na komandnom jeziku su:
 - pamćenje komandi je kratkotrajno, pa korisnici često koriste uputstva
 - komande se teško uče
 - često se greši
 - teško je obezbediti konkretne poruke o grešci i pomoći zbog velikog broja različitih kombinacija i konteksta u kojima se komande mogu primeniti
 - nije podesan za neiskusne korisnike.

POPUNJAVANJE FORMI

Za popunjavanje formi potrebna je samo tastatura, pa je unos podataka brz i prilagođen rutinskom radu

Stil interakcije zasnovan na popunjavanju formi, takozvani **fill in the blanks**, je namenjen korisnicima koji nisu eksperti. Podaci se mogu unositi samo u polja koja su predviđena za to. Prelazak iz polja u polje se vrši pritiskom na **taster TAB**, a slanje podataka i završetak rada sa formom pritiskom na **ENTER**. Za rad sa formama nije potreban nikakav drugi ulazni uređaj osim tastature. Zbog toga što korisnik nema potrebe da sa tastature prelazi na miša, unos podataka je jako brz i potpuno prilagođen rutinskom radu, kao što je na primer unos velikog broja podataka sa upitnika i obrazaca.

Stil interakcije zasnovan na popunjavanju formi ne zahteva obavezno grafičko radno okruženje. Na slici 1 je prikazana jedna jednostavna forma u tekst-baziranom okruženju. Akcije nad formom se izvršavaju kombinacijom tastera Ctrl i nekog slova. Na primer izlazak iz forme uz pamćenje njenog sadržaja se postiže kombinacijom Ctrl-S.



Slika 3.2.1 Jednostavna forma u test baziranom okruženju [Izvor: Autor]

Ista forma se može realizovati i u grafičkom okruženju. Na primer, ako bi prethodnu formu trebalo publikovati na vebu, onda bi ona imala oblik kao na slici 2.

Slika 3.2.2 Veb forma [Izvor: Autor]

Čak i danas se mnoge aplikacije, kao što su računovodstveni programi, baziraju na stilu popunjavanja formi pre svega zbog jednostavnosti i brzine rada. **Prednosti interakcionog stila baziranog na formama** su:

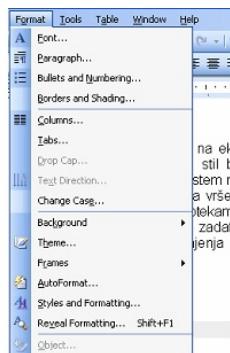
- **jednostavan unos podataka**
- **kratko vreme učenja** u smislu da su polja unapred predviđena i da ih je potrebno samo prepoznati
- **korisnik se vodi kroz formu po unapred definisanim pravilima**, kao što je na primer akcija TAB.

Nedostatak ovog stila je što je ceo ekran popunjjen formom i što zahteva krutu formalizaciju poslovnog procesa. Korisnik ne može da izvrši ni jednu drugu akciju osim seta predviđenih, pa se i sam oseća kao mašina.

IZBOR IZ MENIJA

Stil interakcije „izbor iz meniaj“ se koristi kako u operativnim sistemima i aplikacijama, tako i za navigaciju na veb stranama

Meni je set opcija prikazanih na ekranu pri čemu izbor i izvršenje jedne opcije dovodi do promene stanja interfejsa. Interakcioni stil baziran isključivo na izboru iz menija podrazumeva da su sve moguće opcije ugrađene u sistem menija. Komande se uglavnom grupišu tako da se u istom meniju nalaze sve opcije potrebne za vršenje neke klase zadataka. Tako se na primer, sve akcije koje se mogu koristiti u radu sa datotekama grupišu u meniju **File**. Labele u meniju su razumljive tako da korisnik može da izvršava zadatke uz minimum prethodnog učenja i korišćenjem memorije prepoznavanja. U cilju smanjenja potrebnog prostora za prikaz opcija, meniji se organizuju kao padajući ili **pop-up** meniji.



Slika 3.2.3 Padajući meni [Izvor: Autor]

Opcije koje nije moguće koristiti se na nekin način označavaju kao neaktivne.

Ovakav stil interakcije se koristi kako u operativnim sistemima i aplikacijama, tako i za navigaciju na veb stranama.



Slika 3.2.4 Primer menija [Izvor: Autor]

PREDNOSTI I MANE STILA ZASNOVANOG NA MENIJIMA

Korišćenje menija je bolji izbor za početnike, ali nije podesan za male grafičke displeje

Prednosti stila interakcije zasnovanog na menijima su:

- idealan za početnike
- dozvoljava istraživanje i učenje, jer korisnik može da se šeta po menijima da vidi koje su opcije moguće
- ne zahteva učenje sintakse komandi
- podržava donošenje odluka
- jednostavan je za realizaciju rutina koje manipulišu greškama jer ulaz korisnika ne treba da bude parsovan.

Nedostaci stila interakcije baziranog na menijima:

- previše menija može da unese kompleksnost koja nije prihvatljiva većini korisnika
- spora interakcija za često korišćenje i operacije koje se ponavljaju
- nije podesan za male grafičke displeje

Koncept direktne manipulacije koji se primenjuje u grafičkim okruženjima zasniva se na stilu interakcije baziranom na menijima.

PODUDARNOST ELEMENATA INTERFEJSA SA KORISNIČKIM ZAHTEVIMA

Isti korisnici u različitim kontekstima preferiraju različite stilove interakcije, zbog toga je potrebno da interfejs bude fleksibilan i da omogući primenu različitih stilova interakcije

Korisnici imaju različite potrebe, želje i mogućnosti. Nije moguće napraviti jedinstveni korisnički interfejs koji će odgovarati svim korisnicima. Čak i isti korisnici u različitim kontekstima preferiraju različite stilove interakcije.

Jedini način da se ovaj problem prevaziđe je izrada fleksibilnih interfejsa koji istovremeno omogućuju primenu različitih stilova interakcije. Vrednost ovakvog interfejsa će biti još veća ako interfejs omogućuje dodatna prilagođavanja (kustomizaciju) od strane korisnika.

Razume se da je razvoj ovakvih rešenja složen i mukotrpan. Zbog toga su ovakva rešenja, ako se postignu, po pravilu skupa.

▼ Poglavlje 4

DOSTUPNOST

KONTROLISAN PRISTUP

Bez obzira na to koji se metod koristi korisnik mora da se autentikuje pre korišćenja sistema

Pristup računarskim resursima informacionog sistema iz razloga bezbednosti mora da bude kontrolisan. Da bi se proverilo da li je neka osoba autorizovana da pristupi resursima vrši se autentikacija. Za autentikaciju se trenutno koriste tri glavne metode:

- Nešto što znate, uobičajeno lozinka ili personalni identifikacioni broj
- Nešto što imate, uobičajeno neka identifikaciona kartica (engl. **token**)
- Nešto što ste vi, uobičajeno biometrijska osobina.

Mada sve tri metode služe da se postigne isti cilj, način na koji se postiže cilj nije isti. Prva dva metoda su zasnovana na poverljivim elementima: **poznavanju lozinke ili posedovanju tokena**. Biometrijska metoda se razlikuje od prethodne dve po tome što karakteristike koje se koriste za autentikaciju nisu tajna.

Nijedna od ovih metoda nije savršena. Korisnici vrlo često zaboravljaju lozinku. Identifikaciona kartica može da ne bude uz korisnika ili da bude ukradena. Biometrijski metod je nepouzdan kada se vrši identifikacija preko Interneta. Zbog toga se za pouzdanu autentikaciju najčešće koristi kombinacija ovih metoda.

Bez obzira na to koji se metod koristi korisnik mora da se autentikuje pre korišćenja sistema. U doba kada je korisnik mogao da koristi samo jedan računar i jednu aplikaciju, bilo je dovoljno izvršiti samo jednu autentikaciju. Sa porastom broja računara i broja raspoloživih aplikacija sve češće se događa da jedna osoba koristi više računara i mnogo aplikacija koje zahtevaju autentikaciju. Računari mogu biti u istoj ili različitim prostorijama organizacije. Pored toga, uvođenjem mobilnih i ugnježdenih sistema korisnik ima potrebu da koristi i računarske sisteme van organizacije. Za korišćenje svakog od ovih sistema korisnik mora da izvrši autentikaciju. U slučaju kad često menja prostorije korisnik mora da se na istom računaru autentikuje nekoliko puta dnevno, što postaje opterećujuće. Da bi to izbegao, korisnici se pri prestanku rada sa jednim sistemom ne odjavljuju, i napuštajući prostoriju ostavljaju mogućnost da neko drugi, pod njihovim identitetom, ugrozi integritet sistema.

SINDROM STRESA

Ponavljanje istih operacija dovodi do sindroma stresa. Korisnik postaje ogorčen i gubi volju da uradi predviđene radnje.

Problem postaje još veći sa uvođenjem veba. Da bi pristupio nekim veb sajtovima i koristio neke veb aplikacije, od korisnika se očekuje da se autentikuje. Svaki novi veb portal zahteva novu autentifikaciju. Ovo ponavljanje istih operacija dovodi do sindroma stresa. Korisnik postaje ogorčen i gubi volju da uradi predviđene radnje.

Za rešenje ovog problema se predlažu različita rešenja. U slučaju da se radi o pristupu računarskim resursima u istoj organizaciji sa istog računara uobičajeno rešenje je single-sign-on metod. On podrazumeva da se korisnik samo jednom autentificira i sistem proverava za koje je resurse korisnik autorizovan. Ako korisnik mora da menja računare, tokom svog rada, ili pristupa drugim sistemima preko veba problem postaje složeniji. Kako bi se autentifikacija učinila što manje napornom razvijaju se mnoge metode. Većina ovih metoda koristi biometriku kao način za pouzdanu i brzu autentifikaciju.

SINGLE SIGN-ON METODA - VIDEO

What is Single Sign-on (SSO) System? How it Works?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 5

BIOMETRIKA

BIOMETRIKA KAO DISCIPLINA

Biometrika je disciplina koja se bavi automatskim metodama za jedinstveno prepoznavanje osoba na osnovu njihovih fizičkih i ponašajnih osobina

Biometrika je disciplina koja se bavi automatskim metodama za jedinstveno prepoznavanje osoba na osnovu njihovih fizičkih i ponašajnih osobina. Fizičke osobine koje se mogu iskoristiti za biometriku su:

- otisak prstiju
- očna retina
- dužica oka
- oblik lica
- oblik šake
- DNA

Ponašajne osobine koje se mogu iskoristiti za biometriku su:

- potpis
- hod
- način kucanja na tastaturi.

Glas je mešavina fizičkih i ponašajnih osobina.

Nisu sve navedene osobine podjednako primenljive za biometrijsku autentikaciju. Neke nisu upotrebljive jer ih je teško meriti, a druge jer nisu jedinstvene za svaku osobu.

Da bi biometrijski sistem mogao da se koristi potrebno je izvršiti merenja biometrijskih osobina osoba i stvoriti biometrijske uzorke osoba. Za svaku osobu se u bazi podataka čuvaju podaci o njenom identitetu, kao što su ime, prezime, matični broj, personalni lični broj, lozinke i drugi slični podaci i podaci o jednoj ili više biometrijskih osobina. Biometrika se koristi za dve namene: identifikaciju i verifikaciju.

IDENTIFIKACIJA I VERIFIKACIJA

Identifikacija je proces pronalaženja identiteta nepoznate osobe.

Verifikacija je proces u kome se određuje da li je osoba koja se predstavila stvarno ta osoba

Identifikacija je proces pronalaženja identiteta nepoznate osobe za koju postoji neka biometrijska osobina. Data biometrijska osobina nepoznate osobe se upoređuje sa podacima u bazi podataka. Ako se dati biometrijski podaci slažu sa nekim od biometrijskih uzoraka u bazi podataka, onda je poznat identitet osobe. Pošto je za nalaženje biometrijskog uzorka koji se slaže sa biometrijskim osobinama nepoznate osobe potrebno proveriti sve podatke iz baze podataka, identifikacija zahteva računarske sisteme velike procesorske snage. Ovaj metod se, na primer, koristi u forenzici za identifikovanje nepoznate osobe.

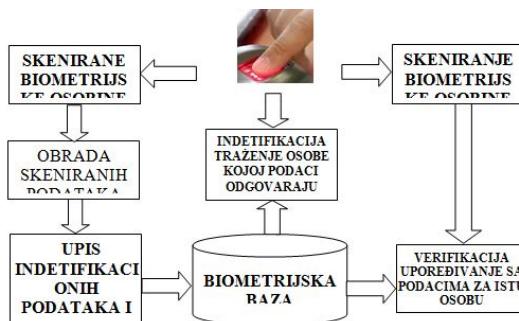
Verifikacija je proces u kome se određuje da li je osoba koja se predstavila stvarno ta osoba. U ovom slučaju se data biometrijska osobina osobe upoređuje samo sa njenim biometrijskim uzorkom u bazi podataka. Dakle, sada nije potrebno proveriti sve uzorke u bazi podataka, nego samo uzorak osobe čiji se identitet verificuje. Zbog toga se za verifikaciju ne zahtevaju računari velike procesorske snage, a samo vreme verifikacije je vrlo kratko. Ovaj metod se, na primer, koristi za pristup prostorijama ili računarskim resursima. Pored toga, sistemi za praćenje prisustva osoba na radnom mestu ili na skupštinskim zasedanjima, mogu takođe da budu implementirani na bazi biometrijske autentikacije i to sa mnogo većom pouzdanošću.

BIOMETRIJSKA AUTENTIKACIJA

Biometrijski sistemi autentikacije svoj rad zasnivaju na bazi podataka koja sadrži biometrijske podatke o registrovanim korisnicima

Biometrija se koristi u informacionim tehnologijama za **biometrijsku autentikaciju** korisnika kako bi se kontrolisao pristup osobama računarskim resursima.

Biometrijski sistemi autentikacije svoj rad zasnivaju na bazi podataka koja sadrži biometrijske podatke o registrovanim korisnicima. Prilikom registracije korisnika, meri se ili snima jedna ili više njegovih osobina. Izmerene vrednosti se obrađuju i čuvaju zajedno sa ostalim podacima o korisniku. Na taj način sistem za autentikaciju ima digitalnu reprezentaciju korisnika. Pre bilo kog pokušaja da koristi sistem, korisnik je u obavezi da uradi biometriku datih osobina. **Sistem za autentikaciju onda upoređuje izmerene veličine sa digitalnom reprezentacijom korisnika i ukoliko se podaci poklapaju, korisniku se dozvoljava pristup.**



Slika 5.1 Način rada sistema za autentikaciju [Izvor: Autor]

Dva biometrijska merenja iste osobine nikad ne daju iste rezultate. Razlozi mogu da budu različiti. Na primer, prst može da bude zaprljan, osvetljenje prilikom merenja drugačije,

udaljenje od kamere promenjeno itd. Zbog toga se prilikom biometrijske autentikacije dozvoljava izvesna greška, koja se zavisno od primenjenih tehnologija kreće od 60 do 99,9%.

U procesu upoređivanja se koristi Hemingova distanca. **Hemingova distanca** je mera sličnosti dva niza bitova koja opisuju neku biometrijsku karakteristiku osobe. Ako su dva niza identična, onda je Hemingova distanca jednaka nuli, a ako su potpuno različita, jednaka jedinici. Na taj način Hemingova distanca predstavlja procenat različitih bitova u ukupnom broju upoređenih bitova. Kada se registrovani korisnik prijavi, njegova Hemingova distanca je mala, pa će mu sistem za autentikaciju dozvoliti pristup. Ali ako neki drugi korisnik pokuša da se prijavi pod njegovim imenom, Hemingova distanca će biti velika i on će biti odbijen.

BIOMETRIJSKE METODE

Biometrijske metode se međusobno razlikuju po jedinstvenosti, permanentnosti, kolektibilnosti, performansama, prihvatljivosti i pouzdanosti

Biometrijske metode se međusobno razlikuju po sledećim karakteristikama:

- **Jedinstvenost** - opisuje koliko dobro biometrijska osobina razlikuje jednu osobu od druge
- **Permanentnost** - opisuje koliko dobro biometrijska osobina ostaje konstantna sa starenjem
- **Kolektibilnost** - opisuje koliko lako se meri biometrijska osobina.
- **Performanse** - opisuje tačnost, brzinu i robusnost sistema za snimanje biometrijske osobine
- **Prihvatljivost** - opisuje stepen prihvatanja tehnologije od strane ljudi
- **Pouzdanost** - opisuje koliko je teško prevariti biometrijski autentikacioni sistem

Prednost biometrijske autentikacije je u tome što je zasnovana na jedinstvenim karakteristikama osobe. Za razliku od personalnog identifikacionog broja, lozinke ili identifikacione kartice, biometrijska osobina se ne može zaboraviti ili ukrasti.

OTISAK PRSTA

Kompletan otisak se sastoji od stotinjak minijatura koje su raspoređene na različitim lokacijama prsta

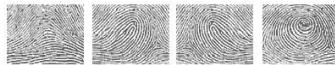
Od svih biometrijskih metoda najčešće se koristi otisak prsta, pre svega zbog pouzdanosti i sazrele i jeftine tehnologije potrebne za ovo merenje. Jedinstvenost otiska prsta se zasniva na dve vrste podataka: globalnom obliku linija i lokalnom obliku linija.

Globalni oblik linija omogućuje da se otisak klasificuje u jednu od četiri kategorije:

- **luk**
- **leva petlja**

- **desna petlja**
- **spirala**

Na narednoj slici su dati primeri ovih kategorija.



Slika 5.2 Oblici linija u otisku [Izvor: Autor]

Karakteristike lokalnog oblika linija se nazivaju minijature ili tačke identifikacije. One su jedinstvene za svaku osobu. One se koriste identifikaciju osobe. U minijature spadaju:

- **bifruktacija**
- **završetak linije**
- **tačka ili ostrvo**

Na narednoj slici su dati primeri minijatura.

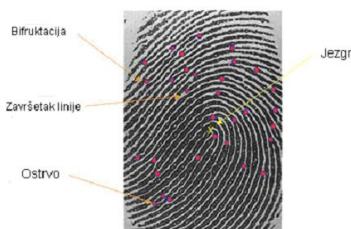


Slika 5.3 Minijature [Izvor: Autor]

Trenutno sistemi za autentikaciju koriste minijature za prepoznavanje osobe. Kompletan otisak se sastoji od stotinjak minijatura koje su raspoređene na različitim lokacijama prsta. Prilikom skeniranja otiska snima se 30 do 60 minijatura zavisno od oblika prsta i površine skenera.

Tačke minijatura se predstavljaju oblakom tačaka u koordinatnom sistemu. One se memorisu zajedno sa uglovima tangenata na linije na kojima se nalaze. Svi ovi podaci čine referentni uzorak neke osobe, odnosno biometrijski model otiska prsta. Ovaj referentni model se kasnije upoređuje sa podacima dobijenim skeniranjem prilikom autentikacije.

Za autentikaciju se koristi skener prsta. To je ulazni uređaj koji sistemu za autentikaciju šalje digitalnu sliku otiska prsta. Dobar i pouzdan skener je osnova za kvalitetnu autentikaciju.



Slika 5.4 Minijature u otisku prsta [Izvor: Autor]

▼ Poglavlje 6

Pokazna vežba: Test upotrebljivosti

PRIMER TESTA UPOTREBLJIVOSTI

U ovoj sekciji će se uraditi test upotrebljivosti sajta prodavnice za prodaju nameštaja

Predviđeno vreme pokazne vežbe je 90 minuta.

Na sledećem primeru biće prikazano kako se vrši jedan test upotrebljivosti. Potrebno je napraviti sajt za prodavnicu nameštaja, namestaj.rs i uraditi test upotrebljivosti za taj sajt.

Prvi korak je da se napravi plan testa upotrebljivosti, a potom zajedno sa studentima u grupi formirati "mini test okruženje". Odredite jednog u grupi koji će služiti kao moderator testiranja. Zajedno analizirajte dobijene rezultate i prezentujte ih tokom vežbi.

Potrebno je da plan i prezentacija rezultata oba budu pravilno formatirani dokumenti.

Podsetimo se prethodnih koraka

1. **Plan testa**
2. **Svrha i metodologija testa**
3. **Scenario testa**
4. **Učesnici**
5. **Izvođenje testa i prikupljanje podataka**
6. **Analiza dobijenih podataka**

PLAN TESTA

U ovoj sekciji biće prikazan plan testa upotrebljivosti

Prvi korak u kreiranju testa upotrebljivosti jeste plan testa. Svrha plana jeste dokumentovanje toga šta treba uraditi, kako će se sprovoditi test, koji parametri će se meriti, broj ispitanika koji će raditi test, itd.

U nastavku je prikazan plan testa za dati primer sajta prodavnice nameštaja.

Plan testa:

Plan testiranja upotrebljivosti će se vršiti tokom realizacije kreiranja sajta i nakon njegovog kreiranja, radi provere njegove upotrebljivosti kao i održavanja i redovnog update-ovanja sajta. Testiranje će se vršiti u zavisnosti od obima i kompleksnosti zahtevanog sajta, najmanje jedan dan nakon razvoja svakog ključnog elementa (dela) sajta, a potom na utvrđenom

vremenskom razmaku, određenom na osnovu iskustva sa prethodnih testova i glavnog završnog testa pre objavljanja sajta. Finansijska sredstva za testiranje će iznosi 5% od cene projekta, jer je vlasnicima sajta veoma važno da on bude funkcionalan, upotrebljiv i u skladu sa potrebama korisnika. Broj očekivanih korisnika se ne može sa preciznošću odrediti, ali se prognozira broj od oko 1000 ljudi na mesečnom nivou.

SVRHA I METODOLOGIJA TESTA

U ovoj sekciji biće prikazane svrha i metodologija testa

U sledećem koraku treba objasniti parametre koji se testiraju, koji rezultati se očekuju od testa, dok metodologija testa predstavlja tip testiranja koji je izabran.

Primer:

Svrha testa je pre svega u ispitivanju funkcionalnosti sajta kao i njegovoј jednostavnosti. Stoga će test obuhvatati:

- proveru efikasnosti operacija (koliko koraka je potrebno da bi se izvršio neki zadatak, koliko je brz odziv sistema, oblik odziva sistema),
- proveru lakoće korišćenja (navigaciju u sistemu, metod navigacije, meniji, dugmad, ikone),
- upotrebu ekrana i estetiku, kao i prilagodljivost različitim vrstama uređaja,
- ciljne performanse sistema.

Takođe, potrebno je proveriti i da li je sajt čitljiv u svim veb čitačima, serverima i na monitorima različitih uređaja.

Za izvršenje ovog testa izabran je metod nadgledanog testa sa moderatorom. Moderator će dočekati ispitanike, objasniti im šta se testira, kako će se test izvoditi, koja je njihova uloga. Moderator zatim nadgleda test i nakon toga prikuplja dobijene podatke.

SCENARIO TESTA

U ovoj sekciji biće prikazan scenario testa upotrebljivosti

Scenario testa predstavlja akcije koje ispitnik treba da testira odnosno konkretnе zadatke koje treba da uradi.

Primer:

Svaki od korisnika će dobiti nekoliko vrsta zadataka koji je potrebno obaviti na različitim uređajima, a odnosiće se na realan postupak korišćenja sajta. Primeri zadataka su dati u nastavku:

- Pristupiti sajtu namestaj.rs. Na sajtu pronaći proizvod „stolica“ i informisati se o vrstama i modelima mogućeg izgleda stolica. Nakon toga potražiti proizvod „fotelja“ i odlučiti se

za jedan model. Poslati komentar firmi o pregledanim proizvodima koristeći formular dat na sajtu.

- Pristupiti sajtu namestaj.rs. Pronaći kontakt kompanije i poslati mejl kompaniji koristeći dugme za direktno otvaranje mejla. Upload-ovati sliku sa svog računara kao prilog mejlu. Takođe, ubaciti link sa sajta ka proizvodu koji Vam se dopada.
- Pristupiti sajtu namestaj.rs. Pregledati sve ponuđene proizvode na sajtu. Pronaći lokaciju firme na mapi koja je prikazana na sajtu. Pronaći istorijat kompanije i ostaviti komentar koristeći formu datu na sajtu.
- Pristupiti sajtu namestaj.rs. Pregledati reference na sajtu i partnere kompanije. Vratiti se na početnu stranu. Odatle ući u svaku od glavnih kartica menija i vratiti se na početnu stranu.
- Pristupiti sajtu namestaj.rs. U polju za pretragu pronaci proizvod „krevet“, zatim sortirati proizvode po cenama od najmanje do najveće. Izabrati jedan proizvod po želji i zatim ga naručiti koristeći formu za naručivanje.

UČESNICI

U ovoj sekciji biće prikazani učesnici testa upotrebljivosti

S obzirom da je veb sajt u pitanju, teško je konkretnizovati i detaljno opisati ciljnu grupu, odnosno potencijalne posetioce sajta. Shodno delatnosti posla, može se prepostaviti da će sajtu najviše pristupati zaposleni ljudi stariji od 25 godina bez obzira na pol, kao i kompanije i javna preduzeća u opštini u kojoj firma namestaj.rs posluje i okolini. Korisnici bi takođe mogli biti i ljudi u penziji pa je i njih potrebno uključiti u test.

U testiranju ovog sajta učestvovaće 10 ispitanika. Ispitanici će biti sledećih starosnih dobi:

- 25-30 godina - 4 osobe
- 30-50 godina - 4 osobe
- 50+ godina - 2 osobe

Učesnici u testu će imati različiti stepen obrazovanja i različita iskustva u korišćenju interneta.

IZVOĐENJE TESTA I PRIKUPLJANJE PODATAKA

U ovoj sekciji biće prikazan način izvođenja testa upotrebljivosti i način prikupljanja podataka

Pre početka testa objasniti testerima da je cilj testiranje sajta, a ne učesnika i kako će se rezultati testa koristiti za dalje poboljšanje sajta. Objasniti sva prava testera i pripremiti tabele za unos podataka koji se dobijaju za vreme testiranja.

Tokom testa - Beleženje podataka o iskustvima korišćenja sajta u tabele, snimanje procesa testiranja video kamerom, kao i snimanje ekrana svakog testera. Ukoliko tester nađe na problem, dati mu savet, najpre najopštiji, pa konkretniji ako je potrebno i beleženje datog saveta.

Nakon testa održati završni intervju sa testerima i reći testerima sta se saznalo tokom testa. Ocenjivanje samog testa, sajta i faza u testiranju sajta.

ANALIZA DOBIJENIH PODATAKA

U ovoj sekciji biće prikazana analiza dobijenih podataka testa upotrebljivosti

Nakon završenog testiranja, uslediće analiza prikupljenih rezultata. Za analizu rezultata koristiće se metoda prikupljanja svih problema na koje su korisnici naišli, zatim njihovo svrstavanje u kategorije po sličnosti, a potom njihova konkretizacija, interpretacija, razvrstavanje po stepenu važnosti i preusmeravanje na njihovo rešavanje. Nakon rešavanja problema, potrebno je ponoviti test kako bi se proverilo da li je rešenje odgovarajuće i da li je problem zaista rešen.

▼ Poglavlje 7

Zadatak za samostalni rad: Test upotrebljivosti

OPIS ZADATKA ZA SAMOSTALNI RAD

Samostalna priprema testa upotrebljivosti

Predviđeno vreme izrade zadatka je 45 minuta.

Prateći prethodno definisane korake, definišite plan testa upotrebljivosti i testirajte informacioni sistem Univerziteta.

<https://isum.metropolitan.ac.rs>

Kao rezultat ovog zadatka potrebno je da pripremite test upotrebljivosti i da demonstrirate način na koji bi realizovali njegovo testiranje.

1. Plan testa
2. Svrha i metodologija testa
3. Scenario testa
4. Učesnici
5. Izvođenje testa i prikupljanje podataka
6. Analiza dobijenih podataka

✓ Poglavlje 8

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ10

Izvršiti plan testiranja

Predviđeno vreme izrade domaćeg zadatka je 80 minuta.

Pronaći neki sajt po vašoj želji na kome će se izvršiti plan testiranja upotrebljivosti. Opisati sledeće korake kreiranja testa upotrebljivosti:

1. Plan testa
2. Svrha i metodologija testa
3. Scenario testa
4. Učesnici
5. Izvođenje testa i prikupljanje podataka
6. Analiza dobijenih podataka

Dokument snimiti pod imenom **IT210-DZ10-ime_prezime_brojIndeksa**, gde su ime, prezime i broj indeksa vaši podaci. Tako imenovan dokument arhivirati i poslati na adresu predmetnog asistenta na e-mail.

(napomena: u Subject-u e-majla napisati IT210 - DZ10)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovoj lekciji su:

- Prikazani koraci za izvođenje jednostavnog testa upotrebljivosti aplikacije
- Prikazani najvažniji standardi za upotrebljivost
- Definisani stilovi interakcija
- Navedene prednosti i nedostaci ograničavanja pristupa IT resursima korišćenjem biometričkih tehnologija
- Objasnjeni sindromi stresa izazvanim ponavljanjem istih operacija

Literatura

[1] Wilbert O. Galitz, *The Essential Guide to User Interface Design - An Introduction to GUI Design Principles and Techniques*, Second Edition, John Wiley & Sons, Inc, 2002.

[2] Jakob E. Bardram, Rasmus E. Kjaer, and Michael O. Pedersen, *Context-Aware User Authentication—Supporting Proximity-Based Login in Pervasive Computing*, Proceedings of Ubicomp 2003, Springer Verlag, 2003.



IT210 - SISTEMI IT

OSNOVE RAČUNARSKIH MREŽA

Lekcija 11

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 11

OSNOVE RAČUNARSKIH MREŽA

- ✓ OSNOVE RAČUNARSKIH MREŽA
- ✓ Poglavlje 1: STANDARDI
- ✓ Poglavlje 2: RAČUNARSKA MREŽA
- ✓ Poglavlje 3: REPETITORI
- ✓ Poglavlje 4: RAZVODNICI
- ✓ Poglavlje 5: MREŽNI MOSTOVI
- ✓ Poglavlje 6: RUTERI
- ✓ Poglavlje 7: MREŽNA KARTICA
- ✓ Poglavlje 8: ŽIČNI PRENOSNI MEDIJUM
- ✓ Poglavlje 9: OSI model
- ✓ Poglavlje 10: Referentni mrežni model TCP/IP
- ✓ Poglavlje 11: RUTIRANJE
- ✓ Poglavlje 12: ALGORITMI RUTIRANJA
- ✓ Poglavlje 13: PROTOKOLI RUTIRANJA
- ✓ Poglavlje 14: Pokazne vežbe: PHP šabloni
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je upoznavanje sa osnovnim elementima i načinom rada računarskih mreža

U ovoj lekciji biće obrađene sledeće teme:

- Standardizaciona tela
- Elementi računarske mreže
- ISO/OSI mrežni model
- Rutiranje u računarskim mrežama

▼ Poglavlje 1

STANDARDI

STANDARDIZACIONA TELA

Standardizaciona tela, ili kako se često nazivaju komiteti, se obično formiraju od predstavnika istaknutih kompanija u oblasti kako bi se dobilo kvalitetno rešenje

Za mrežnu komunikaciju se koriste dve vrste standarda: de facto i de jure. De facto standardi su široko rasprostranjeni i prihvaćeni standardi koji formalno nikada nisu usvojeni od nekog standardizacionog tela. Oni nastaju kao posledica brzog tehnološkog razvoja. Jedna kompanija ili konzorcijum, kao lider u nekoj oblasti, ponudi ili nametne rešenje koje biva prihvaćeno od drugih kompanija i tako postane de facto standard. U nekim slučajevima ovi standardi su vlasništvo jedne kompanije koja ih ne publikuje i ne nudi drugim kompanijama. Primer ovakvog standarda je bio IBM-ov standard System Network Architecture. Ovakvi standardi se nazivaju standardi zatvorenih sistema. Publikovani i dostupni standardi se nazivaju standardi otvorenih sistema. Primer ovakvog standarda je bio Novell-ov NetWare. Sve do pojave Open System Interconnection (OSI) modela, većina de facto standarda je bila zatvorena.

De jure standardi nisu vlasništvo nijedne kompanije već predstavljaju zajedničko dobro. Oni se uglavnom razvijaju od strane nezavisnih standardizacionih tela koja javno objavljaju standard sa namerom da se poboljša konektivnost i interoperabilnost. Na taj način svi proizvođači mogu da primene standard. Primer nezavisnog de jure standarda je TCP/IP.

Standardizaciona tela, ili kako se često nazivaju komiteti, se obično formiraju od predstavnika istaknutih kompanija u oblasti kako bi se dobilo kvalitetno rešenje koje ni jednu kompaniju neće dovesti u povlašćeni položaj. Druge kompanije prihvataju standard jer time smanjuju rizik i cenu razvoja sopstvenog rešenja. S druge strane, prihvatanje standarda znači da će njihovi proizvodi biti kompatibilni sa drugim proizvodima koji su bazirani na istom standardu.

Najvažnije standardizacione organizacije koje se bave pitanjima vezanim za rad i korišćenje računarskih mreža su:

- International Standards Organization (ISO)
- International Telecommunication Union (ITU)
- Institute of Electrical and Electronic Engineers (IEEE)
- Internet Architecture Board (IAB)
- Internet Engineering Task Force (IETF)

INTERNACIONALNA STANDARDIZACIONA ORGANIZACIJA,

Do sada je ISO objavio preko 5000 standarda iz različitih oblasti

Internacionalna standardizaciona organizacija (ISO) je smeštena u Ženevi u Švajcarskoj. Njen zadatak je da razvija i publikuje standarde i koordiniše aktivnosti nacionalnih standardizacionih tela. Do sada je ISO objavio preko 5000 standarda iz različitih oblasti.

Godine 1977. ISO je pokrenuo proces projektovanja komunikacionih standarda baziranih na arhitekturi otvorenih sistema. Ovaj model je postao poznat kao Open Systems Interconnection (OSI) model. OSI je razvijen u saradnji sa International Electrotechnical Commission (IEC), takođe međunarodnim standardizacionim telom koje se pretežno bavi standardima iz oblasti električne i elektronike. U oblasti informacionih tehnologija aktivnosti ISO i IEC se delimično preklapaju, s tim da se IEC više bavi hardverom, a ISO softverom. Ove dve organizacije su 1987. godine formirale zajedničko telo Joint Technical Committee 1 (JTC 1) čija je odgovornost razvoj ISO, a time i IEC, standarda iz oblasti informacionih tehnologija. Unutar tehničkog komiteta osnovane su radne grupe specijalizovane za neke usko stručne oblasti.

PROCES USVAJANJA STANDARDA

Kako bi se osiguralo da će predloženi standard biti globalno prihvачen, proces usvajanja jednog standarda od njegovog predloga pa do publikovanja je podeljen u 7 koraka

Kako bi se osiguralo da će predloženi standard biti globalno prihvачen, proces usvajanja jednog standarda od njegovog predloga pa do publikovanja je podeljen u 7 koraka:

- Radni predlog se dodeljuje odgovarajućem tehničkom komitetu, odnosno odgovarajućoj radnoj grupi. Radna grupa priprema tehničku specifikaciju za predloženi standard i publikuje je kao draft proposal (DP). Tokom najmanje tri meseca zainteresovani članovi ISO mogu da daju komentare na predloženi DP i da glasaju za predlog. Kada se stavovi oko specifikacije usaglase predlog se šalje Centralnom sekretarijatu ISO.
- U periodu od dva meseca Centralni sekretarijat je u obavezi da registruje DP.
- Centralni sekretarijat uređuje tekst dokumenta da bi ga usaglasio sa ISO praksom. Ovakav dokument se publikuje kao draft international standard (DIS).
- Tokom narednih šest meseci traje period glasanja. Da bi bio prihvачen DIS treba da dobije odobrenje većine članova tehničkog komiteta i 75% članova glasača. U ovom periodu su moguće revizije standarda kako bi se postigla potrebna većina glasova. Ako je protiv standarda i dalje više od dva glasa iz tehničkog komiteta, male su šanse da će DIS biti publikovan kao standard.
- Prihvaćeni DIS će u periodu od tri meseca biti vraćen Centralnom sekretarijatu koji ga podnosi Savetu ISO, koji deluje kao bord ISO direktora.
- DIS se usvaja od strane Saveta kao internacionalni standard (IS).
- ISO publikuje novi IS.

INTERNATIONAL TELECOMMUNICATION UNION (ITU)

ITU je odgovorna za veliki broj mrežnih i komunikacionih standarda uključujući i sledeće serije X i V standarda

International Telecommunication Union (ITU) je internacionalna organizacija Ujedinjenih nacija sa sedištem u Ženevi koja koordinira globalne telekomunikacione mreže i usluge u vladinom i privatnom sektoru. Do 1993. godine ova organizacija je bila poznata pod imenom International Telegraph and Telephone Consultative Committee. ITU je odgovorna za veliki broj mrežnih i komunikacionih standarda uključujući i sledeće serije X i V standarda:

- V.35 serial interface standard
- V.90 56-Kbps modem standard
- X.25 packet-switching network standard
- X.400 message-handling system (MHS) standard
- X.435 standard for electronic data interchange (EDI) over X.400
- X.500 directory service recommendations
- X.509 digital certificate and authentication standard
- Standards for Standardized Generalized Markup Language (SGML).

Deo organizacije ITU koji je odgovoran za međunarodne telekomunikacione standarde se naziva ITU Telecommunications Standardization Sector, ili skraćeno ITU-T.

INTERNET SOCIETY

Većina organizacija koje se bave standardima vezanim za Internet su deo udruženja Internet Society (ISOC).

Većina organizacija koje se bave standardima vezanim za Internet su deo udruženja Internet Society (ISOC). Ovo je profesionalno udruženje koje je formirano 1992. godine sa zadatkom da rukovodi razvojem Interneta. ISOC se sastoji od preko 150 pojedinačnih organizacija iz različitih zemalja. Najvažnije organizacije su Internet Engineering Task Force (IETF) i Internet Architecture Board (IAB) koje se bave standardima. Jedan od najvećih poduhvata ISOC-a je bio formiranje neprofitne organizacije Internet Corporation for Assigned Names and Numbers (ICANN), koja nadgleda Internet Domain Name System (DNS).

IEEE

IEEE radi od 1963. godine i svojom aktivnošću se posebno nametnuo u oblasti standarda u oblasti računarstva i komunikacija

Institut inženjera elektrotehnike i elektronike (engl. Institute of Electrical and Electronic Engineers (IEEE)) je organizacija koja okuplja preko 300.000 inženjera, naučnika i studenata elektronike. IEEE radi od 1963. godine i svojom aktivnošću se posebno nametnuo u oblasti

standarda u oblasti računarstva i komunikacija. Standardizacione aktivnosti se izvode u okviru specijalizovanih radnih grupa. Sa aspekta mreža najznačajniji je rad sledećih grupa:

- 802.1 Higher Layer LAN Protocols Working Group
- 802.3 Ethernet Working Group
- 802.11 Wireless LAN Working Group
- 802.15 Wireless Personal Area Network (WPAN) Working Group
- 802.16 Broadband Wireless Access Working Group
- 802.17 Resilient Packet Ring Working Group
- 802.20 Mobile Broadband Wireless Access (MBWA) Working Group
- 802.21 Media Independent Handoff Working Group
- 802.22 Wireless Regional Area Networks

INTERNET ARCHITECTURE BOARD (IAB)

Internet Architecture Board (IAB) je grupa za tehničke savete ISOC-a.

Internet Architecture Board (IAB) je grupa za tehničke savete ISOC-a. Ona je formirana još 1983. godine sa zadatkom da nadgleda razvoj Internet protokola i standarda. Sastoji se od 13 eksperata volontera koje nominuje IETF, a potvrđuje ISOC. Zadaci IAB-a su:

- Nadgledanje evolucije arhitekture Interneta i protokola sa dugoročnog strateškog nivoa
- nadgledanje procesa razvoja Internet standarda i publikovanje serije standardizacionih dokumenata koji se nazivaju Request for Comments (RFC).
- Obezbeđuje smernice ISOC-u koje se tiču tehničkih i proceduralnih pitanja vezanih za internet.

INTERNET ENGINEERING TASK FORCE (IETF)

Rad u IETF se izvodi preko mnogobrojnih specijalizovanih radnih grupa

Internet Engineering Task Force (IETF) je međunarodna zajednica mrežnih inženjera, mrežnih administratora, istraživača i proizvođača čiji je cilj da osigura rad i evoluciju Interneta. IETF je formiran od strane ISOC-a, a njegov svakodnevni rad nadgleda IAB. Rad u IETF se izvodi preko mnogobrojnih specijalizovanih radnih grupa koje se bave pitanjima kao što su: rutiranje, transport, sigurnost, aplikacije, korisnički servisi i drugo. Neke radne grupe rade na proširenjima i novim verzijama protokola kao što su:

- Hypertext Transfer Protocol (HTTP),
- Lightweight Directory Access Protocol (LDAP),
- Network News Transfer Protocol (NNTP),
- Point-to-Point Protocol (PPP),
- Simple Network Management Protocol (SNMP),
- Common Indexing Protocol,
- Internet Open Trading Protocol, Internet Printing Protocol

RFC

Praktično, RFC opisuje neki Internet standard, protokol ili tehnologiju koju predlaže IET

Ove radne grupe pripremaju dokumente koji se nazivaju Internet Draft, čiji je životni vek šest meseci i posle koga moraju biti obrisani, ažurirani ili prevedeni u dokument koji se naziva Request for Comments (RFC). Praktično, RFC opisuje neki Internet standard, protokol ili tehnologiju koju predlaže IETF. Proces ratifikovanja takvog standarda se bazira na konsenzusu. Jednom predloženi RFC se razmatra od strane različitih tehničkih grupa i klasificiše u jednu od pet mogućih grupa:

- Required
- Recommended Elective
- Limited Use
- Not Recommended

Nakon klasifikacije RFC se testira od strane pojedinaca, istraživačkih laboratorija i tehničkih grupa. Da bi jedan RFC postao standard, treba da pođe kroz tri stadijuma:

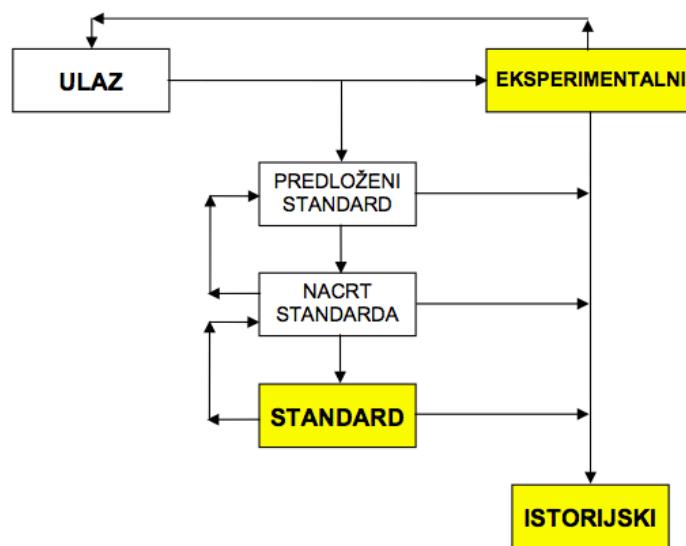
- Proposed standard: stabilan, razumljiv i opšte prihvaćen kao koristan. U ovom stanju ostaje najmanje 6 meseci.
- Draft standard: dovoljno stabilan za implementaciju standarda u aplikacijama i mrežnim tehnologijama. U ovom stanju ostaje najmanje četiri meseca.
- Internet standard: tehnički sazreo standard, implementiran u mnogim aplikacijama ili tehnologijama i koristan za Internet zajednicu.

RFC ŽIVOTNI CIKLUS

Finalnu odluku o tome da li će jedan RFC postati standard donosi IAB na osnovu preporuke IETF-a.

Finalnu odluku o tome da li će jedan RFC postati standard donosi IAB na osnovu preporuke IETF-a.

Na narednoj slici prikazan je životni ciklus jednog RFC. On može da uzme neko od prolaznih ili stalnih stanja. Na slici su prolazna stanja prikazana belim, a stalna stanja žutim pravougaoncima. Ako je neki predlog postao standard, on ostaje standard sve dok ne bude zamenjen novim standardom. Onda prelazi u istorijsko stanje.



Slika 1.1 RFC životni ciklus [Izvor: Autor]

▼ Poglavlje 2

RAČUNARSKA MREŽA

RAČUNARSKE MREŽE I DISTRIBUIRANI SISTEMI

Za dva računara se smatra da su povezana onda kada mogu međusobno da komuniciraju odnosno da razmenjuju podatke

Računarska mreža predstavlja skup povezanih računara koji su međusobno povezani u logičku celinu tako da koristeći jedinstvenu tehnologiju mogu da komuniciraju. Za dva računara se smatra da su povezana onda kada mogu međusobno da komuniciraju odnosno da razmenjuju podatke. Podaci se razmenjuju putem medijuma kao što su bakarni provodnici, optička vlakna, bežično i dr. Računarsku mrežu čine mrežni čvorovi, prenosni medijum i protokoli. Termini računarska mreža i distribuirani sistem označavaju različite koncepte.

Distribuirani sistem je softverski sistem, koji se izvršava na mreži, tako da skup nezavisnih računara korisnici vide kao jedinstveni sistem. Ipak, računarske mreže i distribuirani sistemi se dosta poklapaju (prenose se datoteke sa jednog mesta na drugo, razlika je u tome ko pokreće prenos: korisnik ili sistem). Najpoznatiji distribuirani sistem je WWW u kome sve liči na dokument – web stranu. Internet nije jedinstvena mreža, već je to mreža koja povezuje mnoge mreže, a Web je distribuirani sistem koji se izvršava preko Interneta.

✓ Poglavlje 3

REPETITORI

UPOZNAVANJE SA REPETITORIMA

Repetitori samo prosleđuju podatke, nemaju sposobnost filtriranja i na mestima gde se očekuje intenzivan mrežni saobraćaj treba ih izbegavati.

Signali podležu pogoršanju i izobličavanju, dok putuju kroz kabl. Ovaj efekat se naziva slabljenje (engl. attenuation). Kada je kabl predugačak, signal se toliko izobliči da dolazi do grešaka u prenosu podataka kroz mrežu. Prenošenje signala na veće daljine omogućava se uvođenjem repetitora (engl. repeater). Uloga repetitora je da kada se postavi nasred dugačkih kablova obnavlja signal i prosleđuje ga dalje. Repetitori su obični pojačivači signala koji ne prevode, niti filtriraju mrežne signale iz jednog kabla u drugi, već samo pojačavaju signal koji dobiju na svom ulazu. Oba kabla koja su spojena repetitorom moraju imati isti okvir, logički protokol i metode pristupa.

Metode koje se najčešće koriste su višestruki pristup s detekcijom nosioca i prepoznavanjem sudara (engl. Carrier Sense Multiple Access with Collision Detection, CSMA / CD) i prosleđivanje tokena. Ako se koristi segment koji provodi samo tokene za povezivanje segmenata koji koristi samo CSMA/CD, repetitor tada neće biti upotrebljiv. Repetitori spadaju u tzv. glupe uređaje, jer nemaju složenu logiku. Oni samo prosleđuju podatke, nemaju sposobnost filtriranja i na mestima gde se očekuje intenzivan mrežni saobraćaj treba ih izbegavati.



Slika 3.1 Repetitori [Izvor: Autor]

▼ Poglavlje 4

RAZVODNICI

ULOGA HABA

Hab prima pakete iz mreže i šalje ih svim uređajima koji su povezani na njega

Razvodnik ili hab (engl. hub) je centralni uređaj za povezivanje računara u zvezdanu topologiju. Razvodnik je i uređaj koji se koristi za pristup većem broju radnih stanica (engl. Multistation Access Unit, MAU).

Svi uređaji koji su povezani na hab mogu da komuniciraju između sebe. **Hab prima pakete iz mreže i šalje ih svim uređajima koji su povezani na njega.** Uloga haba nije da čita i obrađuje podatke koji prolaze kroz njega, tako da on nije "svestan" kome su ti paketi zaista namenjeni kada ih prosledi. Hab ima više portova na koji su računari povezani. Manji hab ima 4-5 porta, dok veći imaju 8, 12, 16 i 24 portova.

Oni se dele na:

1. pasivne razvodnike koji ne obrađuje podatke (obična razvodna kutija)
2. aktivne razvodnike obnavljaju signal podataka i održavaju potrebnu snagu signala
3. Inteligentni.

Pasivni hab ne pojačava signal pre nego što ga prenese, on samo difuzno emituje pakete svima, dok aktivni hab služi i kao repetitor, tako da pojačava signal. Inteligentni hab ima jednu dodatnu funkcionalnost u odnosu na aktivni hab. Oni omogućavaju udaljeno upravljanje koristeći upravljački protokol SNMP i virtualni LAN (VLAN).

Sistemi sa razvodnicima su prilagodljivi i imaju određene prednosti nad sistemom bez razvodnika.

Razvodnici imaju ulogu kao i repetitori tj. obnavljaju i prosleđuju signale na isti način. Oni obično imaju od 8 do 12 priključaka i tada ih nazivaju repetitori s više priključaka (engl. multiport repeaters). Da bi aktivni razvodnici radili potrebno je i električno napajanje. *Hibridni razvodnici* podržavaju više vrsta kablova

▼ 4.1 SVIČ

ULOGA SVIČA

Hab prosleđuje pakete svim računarima povezanim na njega, a svič samo onome kome je taj paket namenjen

Komutator ili svič (engl. switch) omogućuje da se na jedan segment mreže poveže više računara, s tim što svič za razliku od haba preusmerava pakete ka računara kome su ti paketi namenjeni. Ovo preusmeravanje obavlja na osnovu MAC adrese. Uobičajeno preklopniči isporučuju sa 8, 16 i 24 ulaza. Preklopniči su zamenili razvodne kutije zato što omogućuju bolju komunikaciju priključenim računarima. Prednost preklopnika je u tome što on, nakon prijema poruke pročita kom čvoru je poruka upućena, i tu poruku pošalje na odgovarajuću izlaznu liniju. Na taj način se izbegava kolizija signala.

▼ Poglavlje 5

MREŽNI MOSTOVI

UPOZNAVANJE SA MREŽnim MOSTOVIMA

Ponašanje sa podacima kod mrežnih mostova je takvo da dok paketi podataka prolaze kroz mrežni most, u njegovu memoriju se upisuju informacije o hardverskim adresama uređaja

Kod opterećenih mreža postavljaju se mrežni mostovi. Oni imaju ulogu repetitora, koji produžava efektivan domet mrežnog kabla i može da izdeli mrežu da bi izolovao zagušeni deo mreže. Nedostatak je što nemaju sposobnost razlikovanja protokola, odnosno prosleđuju sve protokole kroz mrežu.

Karakteristike mrežnih mostova su:

- Usmeravanje podataka. Mrežni mostovi proveravaju izvornu, odredišnu adresu i formiraju tabele usmeravanja, da bi mogli da prosleđuju podatke odgovarajućim delovima mreže. Oni imaju sposobnost učenja kako treba prosleđivati podatke.
- Smanjenje opterećenja mreže. Upotrebom mrežnih mostova za "parcelisanje" velikih mreža u više manjih segmenata, moguće je smanjiti gužvu u mrežnom saobraćaju, čime se poboljšaju ukupne performanse mreže.
- Daljinske veze. Mrežni mostovi se koriste za spajanje manjih, međusobno vrlo udaljenih mreža. Dve zasebne lokalne mreže koje su udaljene jedna od druge, mogu se povezati u jednu mrežu korišćenjem dva udaljena mrežna mosta. Oni se povezuju pomoću sinhronih modema preko iznajmljene telefonske linije.

Ponašanje sa podacima kod mrežnih mostova je takvo da dok paketi podataka prolaze kroz mrežni most, u njegovu memoriju se upisuju informacije o hardverskim adresama uređaja. Te informacije koristi za izradu tabela usmeravanja zasnovanih na izvornim adresama. Na početku njegova memorija i tabela su prazni. Prosleđivanjem paketa kopira se izvorišna adresa u tabelu usmeravanja. Na ovaj način mrežni most saznaje koji se računari nalaze u kojim segmentima mreže.

Kada mrežni most primi paket, adresa se proverava i ako je nema upisuje se u tabelu. Ako je adresa u tabeli, na istom segmentu mreže taj okvir se odbacuje, jer se predpostavlja da je računar već primio taj okvir. U suprotnom mrežni most prosleđuje okvir na odgovarajući priključak. Na ovaj način se vrši filtriranje, čime se smanjuje opterećenje mreže.

U velikim mrežama, računari se grupišu po odeljenjima, jer se mnogo više podataka šalje u okviru odeljenja. Upotrebom mreže moguće je smanjivanje gužve u mrežnom saobraćaju. Ako je potrebno može da se postavi i više mrežnih mostova.

▼ Poglavlje 6

RUTERI

UPOZNAVANJE SA RUTERIMA

Ruteri se primenjuju kod složenih mreža gde mrežni mostovi ne mogu da daju efikasne rezultate u vidu sledećih vrsta usmeravanja podataka

Usmerivač (engl. **router**) je uređaj koji zna adrese svakog segmenta, vrši izbor najbolje putanje za slanje, filtriranje i usmeravanje podataka ka odgovarajućim segmentima. Primjenjuje se kod složenih mreža gde mrežni mostovi ne mogu da daju efikasne rezultate u vidu sledećih vrsta usmeravanja podataka:

1. Statičko usmeravanje se još zove i ručno usmeravanje, jer administrator mreže ručno podešava putanju. Tabele usmeravanja su fiksne, tako da se koristi uvek ista putanja.
2. Dinamičko usmeravanje podrazumeva da se inicijalno podešavaju, ali se posle prilagođavaju promenama stanja mreže. Za putanju koriste one koje su jeftinije, raspoložive i manje opterećene.

Mostovi usmerivači (engl. **bridge-router**) mogu da se ponašaju za jedan protokol kao usmerivač, a za drugi kao mrežni most. U određenim situacijama, upotreba ovog uređaja može biti efikasnija i jeftinija, u odnosu na dva posebna uređaja. S napretkom tehnologije gubi se funkcionalna razlika između mrežnih mostova i usmerivača.



Slika 6.1 Ruteri [Izvor: Autor]

TABELE RUTIRANJA

Da bi se usmerili podaci mora tačno da se navede mrežna adresa, logičke instrukcije za povezivanje na druge mreže, moguće putanje, trošak slanja, itd

Usmerivači imaju svoje tabele usmeravanja, koje imaju mrežne adrese. Da bi se usmerili podaci mora tačno da se navede mrežna adresa, logičke instrukcije za povezivanje na druge mreže, moguće putanje, trošak slanja, itd. Na osnovu ovih podataka usmerivač odabere najbolju putanju za prenos podataka. Ovaj postupak duže traje, od onog koji obavlja mrežni most.

Usmerivač posmatra adresu paketa i ako je ona nepoznata, on je upućuje na glavni mrežni prolaz. Kontroliše podatke koji prolaze, smanjuje opterećivanje mrežnog saobraćaja, odnosno efikasnije koriste komunikacione kanale od mrežnih mostova. Prednost usmerivača je što može da koristi više aktivnih putanja između segmenata lokalne mreže.

Ako je neki usmerivač neispravan, podaci se mogu prenositi drugim putanjama. Uvek traži pravi izbor za prenos podataka. Da bi ovo radili potrebni su im moći algoritmi: OSPF (engl. **Open Shortest Path First**, otvorи најпре краћи пут) ili NLSP (engl. **NetWare Link Services Protocol**, Netwareov Protokol mrežnih usluga).

▼ Poglavlje 7

MREŽNA KARTICA

UPOZNAVANJE SA MREŽNIM KARTICAMA

Mrežna kartica zastupa računar u mreži i prilagođava podatke koje razmenjuju računar i kabl

Mrežne kartice (engl. Network Interface Card, *NIC*) služe za pretragu pojedinačnog računara sa mrežnim kablovima. Ona zastupa računar u mreži i prilagođava podatke koje razmenjuju računar i kabl. Zadatak mrežne kartice je:

- da pretvori paralelne bajtove podataka u serijske bitove, kada se podaci šalju i obratno;
- da upravlja pristupom mreži i ostvari fizičku vezu sa kablom;
- da poveća efektivni protok korišćenjem timskog rada adaptera - adapter otporan na greške (engl.**adapter fault tolerance**, AFT);
- da primenjuje tehniku prilagodljive za raspodele opterećenja (engl. **adaptive load balancing**, ALB).

▼ Poglavlje 8

ŽIČNI PRENOSNI MEDIJUM

UPOZNAVANJE SA ŽIČNIM PRENOSnim MEDIJUMIMA

Na fizičkim kablovima koji povezuju računare i uređaje zasnivaju se mreže svih konfiguracija i veličina.

Na fizičkim kablovima koji povezuju računare i uređaje zasnivaju se mreže svih konfiguracija i veličina.

Zahtevi koje moraju ispuniti mrežni kablovi su:

- otpornost na preslušavanje (engl. **crosstalk**), smetnje u vidu elektromagnetne indukcije signala iz žice;
- otpornost na smetnje zbog uticaja spoljnih elektromagnetnih polja;
- lakoća postavljanja.

Kabovi koji su otporni na preslušavanje i interferenciju omogućavaju veći domet i brži prenos.
Vrste kablova:

1. *Attachment unit interface AUI* -Postoje dve varijante : standard i office.
2. Neoklopljena upredena parica
3. Koaksijalni kabl - Postoje dve varijante: debeli i tanki koaksijalni kabl
4. Optički kablovi

AUI kabl je višežilni oklopljeni kabl koji se koristi za povezivanje mrežnih uređaja. Napravljen je od četiri odvojena i izolovana para žica, obmotane košuljicom kabla. Za njega je karakteristično veliko slabljenje signala na većim daljinama. Standardna dužina kabla je do 50 m. Koristi se u zonama gde je udaljenost manja od njegove dužine.

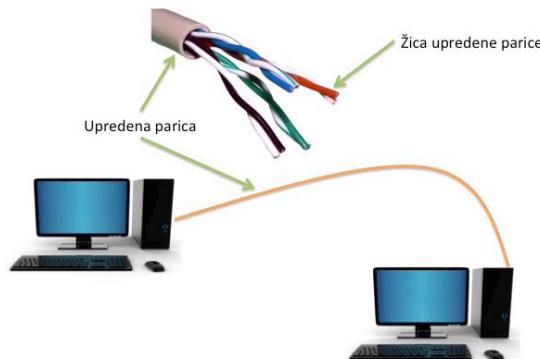
Office kabl se koristi kada je veći broj uređaja smešten na malom prostoru i kada ne može da se postavi standardni kabl.

UPREDENE PARICE

Razlog za upredanjem žica je da bi se poništio signal generisan u različitim navojima, tako da ceo sklop zrači mnogo manje.

Upredene parice su bakarne, jednožilne ili višežilne, mogu biti sa ili bez oklopa. Bez oklopa se nazivaju neoklopljene upredene parice (engl. **unshielded twisted pair** - UTP) i oklopljene

(engl. **shielded twister pair** – STP). Razlog za upredanjem žica je da bi se poništio signal generisan u različitim navojima, tako da ceo sklop zrači mnogo manje. Upredena parica se najčešće koristi u telekomunikacionim sistemima. Da nisu upredene, pojedinačne parice bi ometale jedna drugu. Brzina prenosa zavisi od debljine žice i rastojanja. Za daljine od nekoliko kilometara, može se postići brzina od više megabita u sekundi. Zahvaljujući niskoj ceni i prihvatljivim performansama, upredena parica se masovno koristi. Standardni konektor koji se koristi za UTP kablove je RJ-45,



Slika 8.1 Upredene parice [Izvor: Autor]

Upredena parica se realizuje u više oblika

Kategorija	Brzina prenosa	Primena
1	1Mbps	Prenos glasa (telefonska linija)
2	4Mbps	LocalTalk
3	16Mbps	10BaseT Ethernet
4	20Mbps	TokenRing
5	100Mbps (2 para)	100BaseT Ethernet
	1000Mbps (4 para)	Gigabit Ethernet
5e	1000Mbps	Gigabit Ethernet
6	10000Mbps	Gigabit Ethernet

Slika 8.2 Kategorije kablova [Izvor: Autor]

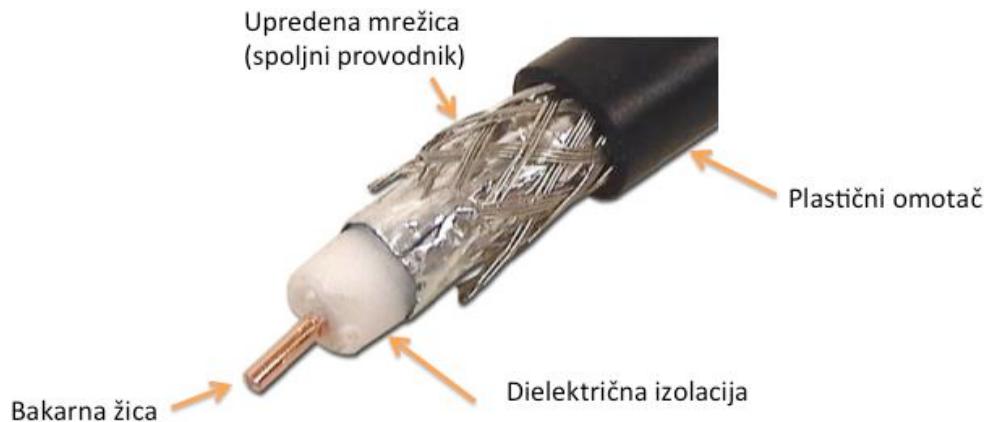
KOAKSIJALNI KABL

Koaksijalni kabl omogućava prenos od 10 do 100 MBps

Koaksijalni kabl se pravi od dva ili više različitih materijala sa zajedničkom centralnom osom. Debeli koaksijalni kabl ima oklop koji je napravljen više slojeva. Prvi sloj je od metalne žice upletene u mrežu. Drugi sloj je metalna folija, zatim sledi dielektrična izolacija i bakarno jezgro. Izolacioni materijal štiti od elektromagnetskih uticaja i smanjuje njegovu fleksibilnost.

Koaksijalni kabl omogućava prenos od 10 do 100 MBps, relativno je jeftin iako je skuplji od UTP kablova, međutim ponekad koaksijalni kabl može biti i jeftiniji, na primer za mrežu koja je projektovana u topologiji magistrale, pošto je potrebno da se koristi manja dužina kablova ukoliko se koristi koaksijalni kabl. Kada se koriste upredene parice, za Ethernet prenos može biti do 100m, dok sa koaksijalnim kablom ta se dužina povećava i do 500m, samim tim potrebno mu je manje pojačivača nego kada se koriste upredene parice. Koaksijalni kabl je jeftiniji od optičkih kablova. Kada se projektuje mreža važno je voditi računa koji tip

koaksijalnog kabla je potrebno koristiti, s obzirom da ima više debljina ovog kabla. Deblji koaksijalni kablovi se koriste samo kod mreža sa specijalnom namenom.



Slika 8.3 Koaksijalni kabl [Izvor: Autor]

OPTIČKI KABLOVI

Optički kablovi su napravljeni od stakla ili plastike. Koriste se za svetlosne bljeskove kao metod transmisije

Optički kablovi su napravljeni od stakla ili plastike. Koriste se za svetlosne bljeskove kao metod transmisije. Pošto ne koriste električnu struju, otporni su na spolja elektromagnetska polja. Prenose signale na velike daljine do 2 km. Kabl se sastoji od čvrstog provodnog staklenog jezgra, oko koga je obmotan sloj polupropusnog materijala koji reflektuje signale.

U jednorežimskim vlaknima, debljina jezgra je 8 do 10 mikrometara. Jezgro je okruženo oblogom od stakla čiji je indeks prelamanja manji od indeksa prelamanja jezgra, kako bi se sva svetlost zadržala u jezgru. Oko svega se nalazi plastični omotač koji štiti oblogu. Vlakna se najčešće grupišu u snopove i zaštićuju dodatnim spoljnim omotačem.

Za emitovanje signala obično se koriste dve vrste svetlosnih izvora: svetlosne diode LED (engl. *Ligth Emitting Diodes*) i poluprovodnički laseri. U tabeli 1 prikazano je poređenje poluprovodničkih lasera i svetlosnih dioda (LED) kao izvora svetlosti. Na kraju optičkog vlakna suprotnom od svetlosnog izvora, nalazi se fotodioda koja generiše električni signal kada je pogodi zrak svetlosti. Tipično vreme reakcije fotodiode iznosi 1 ns, što ograničava brzinu prenosa podataka na oko 1 Gb/s. Termički šum takođe utiče, pa svetlosni impuls mora da ima dovoljnu energiju da bi bio detektovan. Kada su impulsi dovoljno snažni, učestalost grešaka se može proizvoljno smanjiti.

Svojstvo	Svetlosna dioda	Poluprovodnički laser
Brzina prenosa	mala	velika
Vrsta optickog vlakna	jednorežimsko	jednorežimsko ili višerežimsko
Rastojanje	malо	veliko
Zivotni vek	dug	kratak
Osetljivost na temperaturu	zanemarljiva	značna
Cena	niska	visoka

Slika 8.4 poređenje dioda i lasera [Izvor: Autor]

POREDJENJE PRENOSNIH MEDIJUMA

Pošto je prenos svetlosnih signala po prirodi jednosmeran, za dvosmernu komunikaciju su potrebna ili dva vlakna ili dve frekvencije u istom vlaknu

Optička vlakna imaju brojne prednosti u poređenju sa bakarnim provodnicima. Pre svega, vlakna obezbeđuju mnogo veći propusni opseg od bakra. Zbog malog slabljenja repetitore treba postaviti na razdaljinama od oko 50 km, što u odnosu na svakih 5 km u slučaju bakarnih provodnika, predstavlja znatnu uštedu. Optička vlakna su neosetljiva u odnosu na naponske udare, elektromagnetsko polje i prestanak napajanja. Ne korodiraju u vazdušnoj sredini, pa su idealna za teške fabričke uslove. Vlakna su i mnogo lakša od bakra. Vlakno je bolje i zbog mnogo nižih troškova instaliranja i održavanja mehaničkog potpornog sistema. Vlakno je skoro potpuno obezbeđeno od krađe informacija.

Optičko vlakno poseduje i neke loše strane. Naime, lako se oštećuje kada se previše savije. Pošto je prenos svetlosnih signala po prirodi jednosmeran, za dvosmernu komunikaciju su potrebna ili dva vlakna ili dve frekvencije u istom vlaknu. Najzad, interfejsi za optička vlakna su skuplja. Bez obzira na sve, čim se radi o rastojanjima od nekoliko metara, prednost je na strani optičkih vlakana.

RAZLIKE NAJKORIŠĆENIJIH INTERNET KONEKCIJA

Cable vs DSL vs Fiber Internet Explained

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 9

OSI model

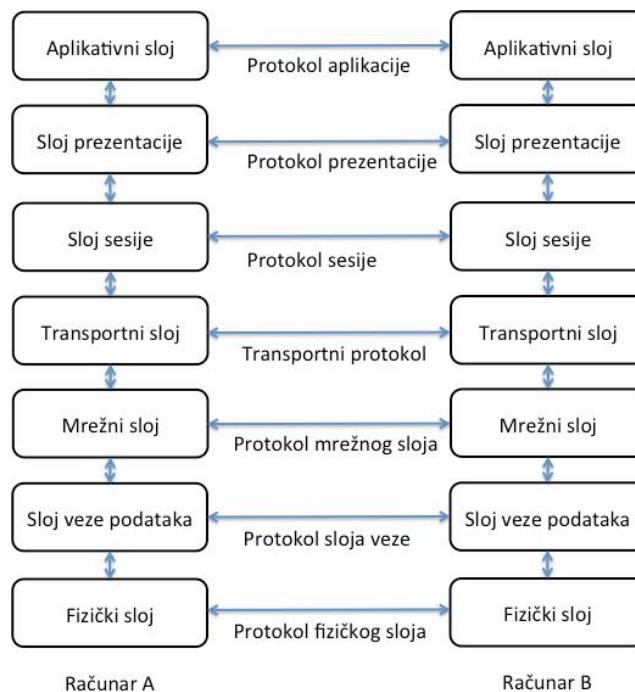
SLOJEVI OSI MODELA

OSI model ima sedam slojeva, od kojih su tri niža sloja fizički sloj, sloj veze podataka i mrežni sloj

Ovaj model se zasniva na predlogu Međunarodne organizacije za standardizaciju (ISO). Model se zove referentni sistem ISO OSI (engl. International Standards Organization - Open Systems Interconnection), zato što treba da poveže otvorene sisteme – one koji su otvoreni za komuniciranje s drugim sistemima. U daljem tekstu ćemo ovaj model zvati skraćeno OSI model.

OSI model ima sedam slojeva, od kojih su tri niža sloja fizički sloj, sloj veze podataka i mrežni sloj. Transportni sloj povezuje niže i više slojeve, odnosno sloj sesije, prezentacije i aplikacije. Kod OSI modela se vodilo računa da je potrebno definisati funkcije svakog sloja tako da se imaju u vidu jasno definisani standardi. Takođe, broj slojeva je definisan tako da se funkcije svakog sloja jasno razlikuju, kako se ne bi smeštale u isti sloj više funkcija na silu.

Svaki od slojeva OSI modela će biti opisan ponaosob. Treba naznačiti to da OSI model ne predstavlja arhitekturu i da se njime ne zadaju specifične usluge i protokoli. Iako je ISO organizacija predvidela standarde za svaki od slojeva, ovi standardi nisu deo modela.



Slika 9.1 OSI model [Izvor: Autor]

FIZIČKI SLOJ

Prvi i najniži sloj OSI referentnog modela je fizički sloj

Prvi i najniži sloj OSI referentnog modela je **fizički sloj** (engl. **physical layer**). Česta skraćenica koja se koristi za fizički sloj je PHY. U odnosu na druge slojeve, fizički sloj se razlikuje u tome što predstavlja jedini sloj u kome se podaci fizički prenose preko mreže. Uloga fizičkog sloja jeste da preneses niz bitova duž komunikacionog kanala. Drugim rečima, cilj je da kada se pošalje bit 1, da primalac primi taj isti bit, a ne bit 0. Iako se ovakvim objašnjenjem stiče utisak da se fizički sloj sastoji samo od komponenti za slanje i primanje podataka poput mrežnih kartica i kablova, on ne predstavlja samo hardverske komponente mreže, već takođe i specificira niz karakteristika jedne mreže.

SLOJ VEZE PODATAKA

Osnovna uloga sloja veze je da prenese niz podataka susednom sloju bez greške

Drugi sloj referentnog OSI modela je sloj veze podataka (engl. **Data Link Layer**). Česta skraćenica koja se koristi za sloj veze je DLL. Sloj veze podataka se često naziva samo sloj veze. Osnovna uloga sloja veze je da prenese niz podataka susednom sloju bez greške, što se postiže maskiranjem stvarnih grešaka tako da ih mrežni sloj ne vidi. Pošiljalac podatke deli u okvire sa podacima (eng. **data frames**), a okvire šalje jedne za drugim. Sloj veze možemo podeliti na dva podsloja, koja se razlikuju po arhitekturi specificiranoj u IEEE 802:

1. Logička kontrolna veza (engl. **Logical Link Control (LLC)**). Osnovna funkcija LLC podsloja je da podeli podatke na okvire (engl. **data frames**) i da vrši kontrolu ispravnosti prenosa. Okviri podataka se najčešće sastoje od nekoliko stotina do nekoliko hiljada bajtova, kako bi se smanjila šteta koja nastaje pri oštećenju ili gubljenju podataka pri prenosu. Primalac potvrđuje ispravan prijem svakog okvira šaljući pošiljaocu okvir za potvrdu (engl. **acknowledgement frame**). Jedan od problema na ovom sloju je (a i kod većine viših slojeva) neusaglašenost brzine slanja i brzine primanja podataka. Osnovne funkcije podsloja LLC su sledeće:

- Pakovanje podataka u okvire (frejmove)
- Adresiranje čvorova
- Kontrola ispravnosti prenosa
- Eliminacija dupliranih okvira i kontrola toka

2. MAC (engl. **Media Access Control**). Omogućuje ravnopravan pristup zajedničkom medijumu u cilju održavanja ponuđenog protoka mreže. MAC adresa je jedinstvena 6-to bajtna adresa koja identificuje čvorove na mreži.

MREŽNI SLOJ

Osnovna uloga mrežnog sloja je da usmerava pakete preko fizičkih linija mreže u zavisnosti od trenutnog stanja i opterećenja mreže

Treći sloj OSI referentnog modela je **mrežni sloj** (engl. **network layer**) koji upravlja radom sloja veza i fizičkim slojem. Osnovna uloga mrežnog sloja je da usmerava pakete preko fizičkih linija mreže u zavisnosti od trenutnog stanja i opterećenja mreže, a u cilju izbegavanja zagušenja mreže. Zadatak protokola mrežnog nivoa je da na osnovu logičkih adresa izvorišnog i odredišnog čvora odredi putanju kojom će se slati poruka. Pri projektovanju mrežnog sloja ključno je specificirati način na koji se paketi upućuju od izvora ka odredištu, pri čemu se prenos obavlja sukcesivno od čvora do čvora. Putanju ne određuju ni pošiljalac ni primalac paketa, već to čine ruteri na osnovu logičke adrese. Ruteri određuju kom susednom ruteru će proslediti primljenu poruku.

Druga važna uloga mrežnog sloja je da vrši segmentaciju podataka. Protokol mrežnog sloja deli podatke na segmente. Ovi segmenti zajedno sa logičkim adresama primaoca i pošiljaoca čine datagram. Veličina dagrama je unapred definisana dužinom okvira koju predviđa protokol sloja podataka. Datagrami se šalju kroz mrežu nezavisno jedan od drugog, ali pri njihovom priјemu na odredište se svi datagrami rekonstruišu u polaznu poruku, što je omogućeno zbog jedinstvenog rednog broja koji je dodat svakom datagramu u procesu segmentacije.

Mrežni sloj treba da kontroliše zagušenja saobraćaja, i generalno vodi računa o kvalitetu ponuđene usluge (zadrškama, vremenu prolaska, neravnomernosti pristizanja paketa). Zadatak mrežnog sloja je da, između ostalog, omogući povezivanje heterogenih mreža (različiti načini adresiranja, baratanjem paketima različitih veličina, korišćenje različitih protokola i sl.).

TRANSPORTNI SLOJ

Transportni sloj povezuje izvor i odredište

Četvrti sloj OSI referentnog modela je **transportni sloj** (engl. **transport layer**) ili još poznat kao "srednji" sloj. Transportni sloj prihvata podatke od sloja iznad, po potrebi ih razvrstava u manje grupe, tzv. pakete, i prosleđuje mrežnom sloju obezbeđujući da svi delovi ispravno stignu na odredište. Transportni sloj na taj način sakriva logičku i fizičku strukturu mreže od gornjeg sloja, sloja sesije.

Najpopularnija vrsta transportne veze je kanal od tačke do tačke sa ispravljanjem grešaka. Druge vrste transportnih usluga su npr. prenošenje izolovanih poruka bez garancije redosleda pristizanja, kao i difuzno slanje poruka na više odredišta.

Transportni sloj povezuje izvor i odredište. U nižim slojevima, protokoli povezuju svaki računar sa njegovim najbližim susedima. Između izvornog i odredišnog računara postoji više usmerivača.

SLOJ SESIJE

Osnovni zadatak sloja sesije je da upravlja i sinhronizuje komunikaciju između aplikacija

Sloj sesije (engl. **session layer**) omogućava korisnicima da uspostave sesiju (engl. **session**). Osnovni zadatak ovog sloja je da upravlja i sinhronizuje komunikaciju između aplikacija. Sloj sesije je obavezan da održi uspostavljenu komunikaciju dok se prenos ne završi. Tipične funkcije ovog sloja su prijavljivanje (engl. **login**), autorizacija, sinhronizacija i odjavljivanje (engl. **logout**). Sesije nude različite usluge:

- upravljanje dijalogom (engl. **dialog control**) tj. vodi računa o tome na kome je red da šalje poruke
- rad sa tokenima (engl. **token management**) tj. sprečavanje korisnika da istovremeno pokrenu istu operaciju
- sinhronizovanje (engl. **synchronization**) tj. proveravanje niza podataka pri prenosu radi sprečavanja pada sistema

SLOJ PREZENTACIJE

Sloj prezentacije bavi se sintaksom i semantikom prenetih informacija

Za razliku od nižih slojeva, koji uglavnom premeštaju bitove sa jednog mesta na drugo, sloj prezentacije (engl. **presentation layer**) bavi se sintaksom i semantikom prenetih informacija. Ovaj sloj obrađuje apstraktne strukture podataka i omogućava da se definisu i razmenjuju strukture podataka višeg nivoa. On obezbeđuje da se informacije pošalju u formi koje su razumljive na odredištu, ali pri tom vrši kompresiju i dekompresiju podataka, kao i šifriranje i dešifriranje.

SLOJ APLIKACIJE

Pravila komunikacije između dva programa su deo protokola aplikacionog sloja

Sloj aplikacija (engl. **application layer**) je najviši sloj OSI referentnog modela. On sadrži više protokola potrebnih korisnicima. Na primer, kada korisnik želi da otvorи Web stranu u čitaču, on serveru šalje zahtev za otvaranje te stranice koristeći HTTP protokol. Ukoliko ta stranica postoji, server kao odgovor šalje tu stranu. Da bi dva programa mogla da uspostave komunikaciju, neophodno je da poštuju ista pravila. Ova pravila komunikacije između dva programa su deo protokola aplikacionog sloja. Za prenos datoteka, elektronske pošte i poruka diskusionih grupa, postoje drugi protokoli aplikacija.

▼ Poglavlje 10

Referentni mrežni model TCP/IP

OSNOVNI OPIS REFERENTNOG MREŽNOG MODELA TCP/IP

TCP/IP referentni model ima 4 sloja

Referentni model TCP/IP je predak svih regionalnih računarskih mreža, ARPANET, a danas ga koristi njegov naslednik globalni Internet. Postavljeni zahtevi uticali su da se izabere mreža sa komutacijom paketa, bazirana na međumrežnom sloju bez direktnog uspostavljanja veze. Slika 1 predstavlja upoređenje dva referentna modela- OSI i TCP/IP model. Može se videti da za razliku od OSI modela, TCP/IP model ima 4 sloja. U modelu TCP/IP nema sloja sesije, niti sloja prezentacije. Za njima nije bilo potrebe, pa nisu ni uključeni u model.

OSI model	TCP/IP model
Aplikacioni sloj	
Prezentacioni sloj	Aplikacioni sloj
Sloj sesije	
Transportni sloj	Transportni sloj
Mrežni sloj	Međumrežni sloj
Sloj veze podataka	
Fizički sloj	Mrežni interfejs

Slika 10.1 Upoređivanje arhitekture OSI i TCP/IP modela [Izvor: Autor]

MREŽNI INTERFEJS

Mrežni interfejs ne predstavlja sloj u klasičnom smislu reči, već predstavlja interfejs između računara i prenosnih veza

Mrežni interfejs ne predstavlja sloj u klasičnom smislu reči, već predstavlja interfejs između računara i prenosnih veza. Referentni model TCP/IP ne objašnjava detaljno šta se tu događa, osim što ističe da računar mora da se poveže s mrežom pomoću nekog protokola kako bi mogao da šalje IP pakete. Sam protokol za povezivanje s mrežom nije definisan i menja se od računara do računara i od jedne mreže do druge.

MEĐUMREŽNI SLOJ

Zadatak međumrežnog sloja je da pakete koji su prosleđeni od strane računara prosleđuju na njihovo odredište

Međumrežni sloj (internet sloj) predstavlja spoj delova mreže koji predstavlja čitavu mrežnu arhitekturu. Zadatak međumrežnog sloja je da pakete koji su prosleđeni od strane računara prosleđuju na njihovo odredište. Prosleđeni paketi mogu da stignu na odredište drugaćijim redosledom nego što su poslati, te je uloga viših slojeva da ih spoje u prvobitan redosled kada je to potrebno.

Međumrežni sloj definiše zvanični format paketa Internet protokola (engl. **Internet Protocol**, IP), kao i njegovog pratećeg protokola ICMP (engl. **Internet Control Message Protocol**) koji se koristi za upravljanje porukama. Zadatak međumrežnog sloja je da isporuči IP pakete tamo gde treba da stignu. Jasno je da je najveći problem usmeravanje i izbegavanje zagruženja.

TRANSPORTNI SLOJ

Dva transportna protokola u okviru transportnog sloja TCP/IP modela su TCP i UDP

Sloj iznad međumrežnog sloja u modelu TCP/IP se naziva **transportni sloj** (engl. **transport layer**). On je namenjen za ravnopravnu konverzaciju između ravnopravnih procesa na izvornom i odredišnom računaru, baš kao i transportni OSI sloj.

Dva protokola su definisana u okviru transportnog sloja TCP (engl. **Transmission Control Protocol**), a koji predstavlja protokol za pouzdan prenos podataka sa uspostavljanjem direktnе veze, i UDP (engl. **User Datagram Protocol**), protokol bez uspostavljanja direktnе veze koji ne garantuje pouzdan prenos podataka. Detalji ova protokola će biti obrađeni u kasnijim poglavljima.

APLIKACIONI SLOJ

Aplikacioni sloj sadrži sve protokole višeg nivoa (TELNET, FTP, SMTP, DNS, HTTP)

Najviši sloj TCP/IP modela je **sloj aplikacija** (engl. **application layer**). Aplikacioni sloj sadrži sve protokole višeg nivoa (TELNET, FTP, SMTP, DNS, HTTP).

TELNET protokol za virtuelni terminal omogućava korisniku da sa svog računara daljinski prijavi na drugi računar i na njemu radi. FTP protokol za prenos podataka omogućava efikasno prenošenje podataka s jednog računara na drugi. Elektronska pošta je na početku ličila na prenos datoteka, ali je kasnije za nju razvijen poseban protokol (SMTP).

Tokom godina sloju aplikacija su dodati i drugi protokoli:

- DNS sistem imenovanja adresa za preslikavanje imena računara u njihove mrežne adrese
- HTTP protokol za preuzimanje strana s WWW
- RTP protokol za isporuku paketa u realnom vremenu, obično se koristi za multimedijalni sadržaj

SLIČNOSTI IZMEĐU MODELAA

Sličnosti između TCP/IP i ISO OSI modela se zasnivaju na principu rada nezavisnih protokola, višeslojnom radu mreže

Sličnosti između modela:

- oba se zasnivaju na principu skupa nezavisnih protokola
- funkcionalnost slojeva je slična (svi slojevi zaključno sa transportnim slojem treba da obezbede transportnu uslugu)
- dalji slojevi iznad transportnog predstavljaju aplikacije koje su korisnici transportnih usluga.

Za OSI model su ključna tri koncepta: usluge, interfejsi, protokoli. U OSI modelu, protokoli su bolje sakriveni i mogu se lakše zameniti sa napretkom tehnologije. Model OSI ima 7 slojeva a model TCP/IP 4 sloja. Oba imaju (među)mrežni sloj, transportni sloj i sloj aplikacija.

Model OSI u mrežnom sloju podržava komunikaciju sa uspostavljanjem direktne veze ili bez nje. Ali, u transportnom sloju podržava samo komunikaciju sa uspostavljanjem direktne veze. Model TCP/IP u mrežnom sloju podržava samo režim komunikacije bez uspostavljanja direktne veze, ali u transportnom sloju podržava oba režima i to: sa uspostavljanjem direktne veze ili bez nje.

NEDOSTACI

Oba referentna modela imaju svoje prednosti i mane. Iako je TCP/IP model široko prihvaćen i on ima svoje nedostatke u definisanju nižih slojeva

Neki od nedostataka OSI modela su sledeći:

- **Loša sinhronizacija** - ako se objave prerano, pre završenih istraživanja, novost neće u potpunosti biti shvaćena, a rezultat su loši standardi. Ako se objave prekasno, kompanije su već investirale velik novac u pravcu koji se razlikuje od standarda, pa se standardi ne poštuju. OSI model se pojavio kada je TCP-IP model već bio široko prihvaćen.
- **Loša tehnologija** - OSI model je bio prepun nedostataka, baš kao i njegovi protokoli. Izbor 7 slojeva je bio više političke, nego tehničke prirode, a njegova dva sloja (sesija i prezentacija) su gotovo prazni, dok su druga dva (veze podataka i mrežni) pretrpani. Takođe, OSI model, zajedno s definicijama svojih usluga i protokolima, izuzetno je složen.

- **Loša realizacija** - model OSI je izjednačen sa niskim kvalitetom servisa. Kod realizacije TCP/IP modela postignuta je uzlazna linija.
- **Loša politika** - preovladalo je mišljenje da se tehnički inferioran standard nudi istraživačima i programerima.

Neki od nedostataka TCP/IP modela su sledeći:

- Model ne razgraničava jasno koncepte usluga, interfejsa i protokola, pa nije od velike pomoći kada treba projektovati nove mreže sa novim tehnologijama.
- Model nije dovoljno uopšten i šturo opisuje bilo koji skup protokola osim skupa TCP/IP.
- Model TCP/IP ne razdvaja fizički sloj i sloj veze podataka. Ove slojeve treba imati kao zasebne.
- Fizički sloj se bavi prenosnim osobinama bakarnog voda, optičkog vlakna i bežičnih komunikacija. Zadatak sloja veze podataka je da označi početak i kraj okvira podataka i da ga prenese s jedne na drugu stranu uz određenu pouzdanost.

UPOREĐENJE ISO OSI I TCP/IP MODELA - VIDEO

Network + Certification N10-006: OSI Model vs. TCP/IP Model

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 11

RUTIRANJE

ŠTA PREDSTAVLJA RUTIRANJE?

Rutiranje je postupak kojim se podaci od izvorišnog čvora na jednoj mreži usmeravaju ka odredišnom čvoru neke druge mreže

Usmeravanje ili rutiranje je postupak kojim se podaci od izvorišnog čvora na jednoj mreži usmeravaju ka odredišnom čvoru neke druge mreže. Za rutiranje se koriste uređaji koji se zovu usmerivači (ruteri). Usmeravanje je moguće jedino u mrežama koje koriste protokole koji dozvoljavaju rutiranje. TCP/IP protokol omogućuje usmeravanje paketa na osnovu odredišne IP adrese.

ARP I MAC ADRESA

Lokalni host koji želi da pošalje paket udaljenoj mreži proverava prvo svoju internu tabelu usmeravanja da bi odredio kom obližnjem usmerivaču da prosledi paket.

Proces usmeravanja počinje na izvorišnom čvoru. Lokalni host koji želi da pošalje paket udaljenoj mreži proverava prvo svoju internu tabelu usmeravanja da bi odredio kom obližnjem usmerivaču da prosledi paket. Zatim host koristi ARP (engl. Address Resolution Protocol) protokol da odredi MAC adresu tog usmerivača. MAC (engl. Media Access Control) adresa je jedinstvena 6-to bajtna adresa koja identificuje čvorove na mreži. MAC je podsloj sloja veze u OSI referentnom modelu, a MAC adresa se koristi u sloju veze za identifikaciju čvorova.

MAC adresa se sreće i pod drugim imenima kao što su: Ethernet adresa, hardverska adresa ili fizička adresa. MAC adresu upisuje proizvođač mrežnog uređaja uobičajeno u ROM-u mrežne kartice. Uz neke kartice se isporučuje i softver koji omogućuje promenu ovog broja, ali ovo nije dobro jer može izazvati konflikt ukoliko se u istoj mreži nađu dva čvora sa istom MAC adresom. Jedinstvenost MAC adresa mrežnih uređaja se postiže time što Institute of Electrical and Electronics Engineers (IEEE) svakom proizvođaču dodeljuje određeni blok adresa.

Prva 3 bajta predstavljaju proizvođača mrežne kartice, a poslednja 3 bajta jedinstveno identificuju karticu. Kada se u TCP/IP mreži šalju paketi oni imaju svoju IP adresu odredišnog čvora. Pomoću ARP protokola se logička IP adresa prevodi u fizičku MAC adresu. Sloju veze se predaje paket sa MAC adresom.

Kada neki usmerivač primi neki paket od izvorišnog usmerivača, on pročita odredišnu adresu i traži takvu adresu u svojoj tabeli usmeravanja. Ako ne može da nađe takvu adresu on

jednostavno odbaci paket. U suprotnom, on prosleđuje paket odredišnom hostu, ukoliko je host u okviru mreže povezane na taj usmerivač, ili ka nekom drugom usmerivaču koji dalje prosleđuje paket sve dok on ne dođe do odredišnog hosta. Tokom prenosa paketa od usmerivača do usmerivača njegova IP adresa odredišnog hosta ostaje ista, ali se MAC adresa menja jer uvek odgovara sledećem usmerivaču duž puta.

PRIMER RUTIRANJA

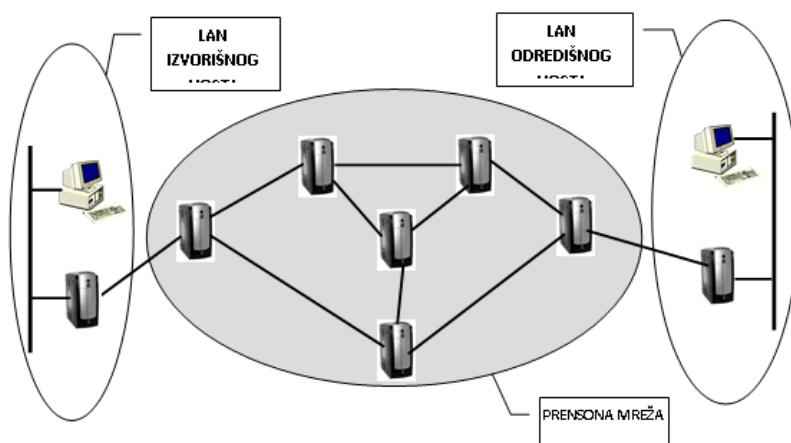
Tabela usmeravanja se popunjava pomoću algoritma za usmeravanje

Postupak usmeravanja će se opisati na primeru prikazanom na slici 1. U ovom slučaju host H1 šalje podatke hostu H2. H1 se nalazi u svojoj lokalnoj mreži koja komunicira sa drugim mrežama preko usmerivača U1. Odredišni host H2 se nalazi u svojoj lokalnoj mreži koja komunicira sa drugim mrežama preko usmerivača U8. Prepostavimo da su obe lokalne mreže udaljene. U tom slučaju neophodno je da za međusobnu komunikaciju koriste usluge i opremu kompanije koja se bavi prenosom podataka. Usmerivači ove kompanije, U2 do U7 i prenosne linije su unutar osenčene elipse.

Kada izvorišni host H1 počne da šalje podatke odredišnom hostu H2, oni se najpre dele na pakete, a zatim se paket po paket šalju do lokalnog usmerivača U1. Usmerivač čuva primljene podatke sve dok ne dobije kompletan prvi paket i nakon izvršene provere kontrolnog zbirka, počne da šalje isti paket usmerivaču kompanije za prenos podataka, u ovom slučaju U2. U ovom slučaju usmerivač U1 nije imao drugu mogućnost, jer je usmerivač U2 jedini sa kojim on ima vezu. Usmerivač U2, prima i čuva podatke sve dok ne stigne ceo paket, a onda proverava kontrolni zbir. Ovaj princip slanja se naziva „čuvaj i prosledi“, a prenos od jednog do drugog usmerivača „skok“ (engl. hop)

Usmerivač U2 sada ima mogućnost da pošalje isti paket usmerivaču U3 ili usmerivaču U5. Da bi odlučio kojom će linijom poslati podatke usmerivač koristi tabelu usmeravanja iz koje bira optimalan put paketa. Tabela usmeravanja se popunjava pomoću algoritma za usmeravanje, o kojima će kasnije biti reči.

Opisani postupak se ponavlja sve dok paket ne stigne do odredišnog usmerivača U8. Usmerivač U8 zatim šalje paket do odredišnog hosta H2, koji se nalazi u njegovoj lokalnoj mreži.



Slika 11.1 Primer mreže [Izvor: Autor]

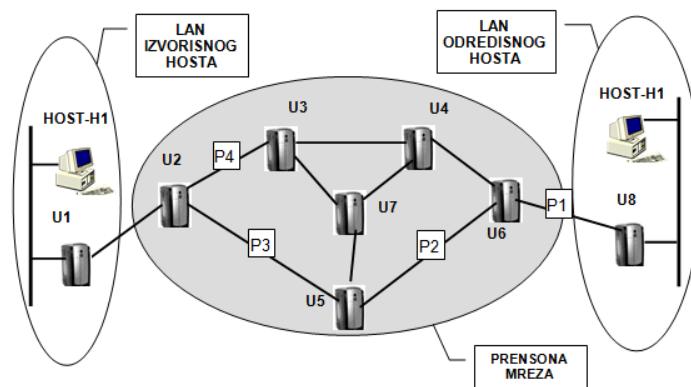
VRSTE USLOGA PRENOŠA U MREŽNOM SLOJU

Moguće su dve vrste usluga prenosa podataka u mrežnom sloju: sa uspostavljanjem i bez uspostavljanja direktne veze

Kojom će se putanjom kretati paketi zavisi od toga kakve usluge nudi mrežni sloj. U principu, moguće su dve vrste usluga prenosa podataka u mrežnom sloju:

- bez uspostavljanja direktne veze
- sa uspostavljanjem direktne veze.

Ukoliko se prenos vrši bez uspostavljanja direktne veze za upravljanje prenosom se koristi **UDP protokol**. Pojedini paketi mogu da se kreću različitim putanjama od izvorišnog do odredišnog hosta. Za ilustraciju procesa prenosa će se koristiti slika 2.



Slika 11.2 Rutiranje bez uspostavljanja direktne veze [Izvor: Autor]

U ovom slučaju izvorišni host H1 šalje podatke koji su podeljeni u 4 paketa. U trenutku kada je prvi paket stigao do usmerivača U2, algoritam za usmeravanje je ažurirao tabele usmeravanja na osnovu trenutnog stanja na mreži. Za ovaj primer su interesantne tabele usmeravanja usmerivača U2, U5 i U6 i one izgledaju ovako:

Usmerivač U2		Usmerivač U5		Usmerivač 6	
Odredište	Linija	Odredište	Linija	Odredište	Linija
U1	U1	U1	U2	U1	U5
U2	-	U2	U2	U2	U5
U3	U3	U3	U7	U3	U4
U4	U3	U4	U7	U4	U4
U5	U5	U5	-	U5	U5
U6	U5	U6	U6	U6	-
U7	U5	U7	U7	U7	U5
U8	U5	U8	U6	U8	U8

Slika 11.3 Tabela rutiranja [Izvor: Autor]

AŽURIRANJE TABELE USMERIVAČA

U slučaju preopterećenja mreže, algoritam ažurira tabelu usmerivača

Pošto je odredišni host vezan za usmerivač U8, usmerivač U2 će na osnovu svoje tabele usmeriti i proslediti prvi paket do usmerivača U5. Usmerivač U5 će u svojoj tabeli potražiti izlaznu liniju koja vodi ka odredišnom usmerivaču U8. U ovom slučaju to je linija koja vodi ka usmerivaču U6, pa će prvi paket biti prosleđen do njega. Na kraju, će usmerivač U6 proslediti paket do usmerivača U8 u čijoj mreži se nalazi odredišni host H2. Ukoliko stanje na mreži ostane nepromenjeno duže vremena, isti postupak i putanja bi se ponovila i za pakete 2, 3 i 4.

Prepostavimo da je nakon što je usmerivač U2 prosledio paket 3 došlo do promene opterećenja mreže i da je linija između U2 i U5 postala preopterećena. U tom slučaju će algoritam za usmeravanje ažurirati tabelu usmerivača U2 koja će sada imati sledeći oblik:

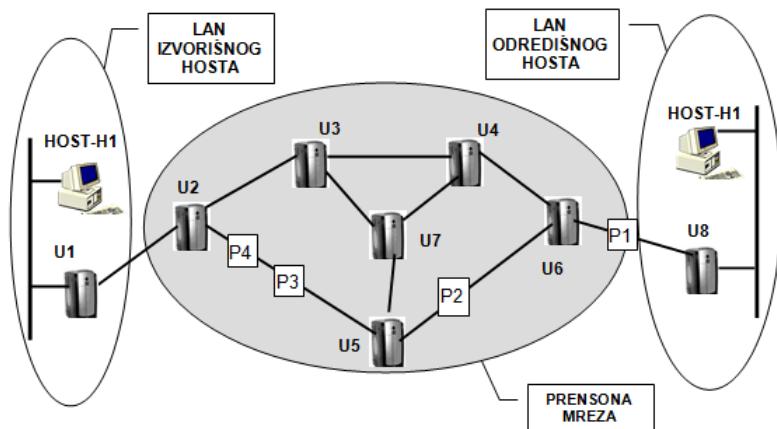
Usmerivač U2	
Odredište	Linija
U1	U1
U2	-
U3	U3
U4	U3
U5	U5
U6	U3
U7	U3
U8	U3

Slika 11.4 Tabela rutiranja [Izvor: Autor]

RUTIRANJE SA USPOSTAVLJANJEM DIREKTNE VEZE

Ukoliko se prenos vrši sa uspostavljanjem direktne veze za upravljanje prenosom se koristi TCP protokol

Ukoliko se prenos vrši sa uspostavljanjem direktne veze za upravljanje prenosom se koristi TCP protokol. U ovom slučaju se najpre uspostavlja virtuelno kolo, a onda se počinje sa slanjem paketa. Svi paketi se kreću istom putanjom od izvorišnog do odredišnog hosta. Za ilustraciju procesa prenosa će se koristiti slika 5.



Slika 11.5 Rutiranje sa uspostavljanjem direktnе veze [Izvor: Autor]

U ovom slučaju algoritam za usmeravanje je pre početka prenosa ažurirao tabele usmeravanja tako da je optimalna putanja paketa od usmerivača U1 sledeća: U1 - U2 - U5 - U6 - U8. Tokom prenosa paketa se može dogoditi da se uslovi na mreži promene. Međutim virtuelno kolo se formira jednom za celu sesiju prenosa, pa će svi paketi proći ovom putanjom.

Proces usmeravanja ima dva koraka:

- **Određivanje najbolje putanje** (rute) preko koje paket treba da prođe da bi došao do odredišnog hosta
- **Prosleđivanje paketa** do odgovarajuće udaljene mreže saglasno njegovoj odredišnoj IP adresi.

▼ Poglavlje 12

ALGORITMI RUTIRANJA

METRIKA PUTANJA

Na osnovu metrike (broja skokova, kašnjenja paketa, opterećenost čvorova, brzina prenosa i sl.) usmerivač određuje optimalnu putanju

Jedan od glavnih zadataka mrežnog sloja OSI modela je usmeravanje paketa od izvorišnog ka odredišnom čvoru. U većini slučajeva paketi pri tom pređu preko više mreža, dakle u više skokova.

Prosleđivanje paketa se vrši nezavisno od strane svakog usmerivača duž putanje koju paket mora da prođe. Pri tom paket skače od mreže do mreže, odnosno od usmerivača do usmerivača. Za usmeravanje, usmerivači koriste tabele usmeravanja (engl. **routingtable**) koje sadrže metriku putanja, odnosno informacije o potencijalnim putanjama kao što su cena prenosa i opterećenje putanje. Na osnovu metrike usmerivač određuje optimalnu putanju.

Za metriku putanja se mogu koristiti podaci kao što su:

- broj usmerivača kroz koji paket treba da prođe do odredišnog čvora (broj skokova)
- kašnjenje paketa zbog obrade na svakom od usmerivača
- opterećenost pojedinih usmerivača trenutnim saobraćajem
- brzina prenosa podataka između dva usmerivača
- relativna pouzdanost usmerivača.

Tabele usmeravanja mogu biti statičke ili dinamičke. Statičke tabele popunjavaju mrežni administratori ručno. Algoritmi na osnovu kojih se definišu podaci u statičkoj tabeli se nazivaju neprilagodljivi algoritmi jer svoju odluku ne donose na osnovu podataka o trenutnom saobraćaju na mreži. Na osnovu određenih parametara i stanja na mreži, popunjavaju se tabele rutiranja na osnovu raznih algoritama dinamički, te te tabele nazivamo dinamičkim.

PRINCIP OPTIMALNOSTI

Princip optimalnosti tvrdi da ako se neki usmerivač B nalazi na optimalnoj putanji između usmerivača A i C, onda se optimalna putanja između usmerivača B i C nalazi na istoj putanji

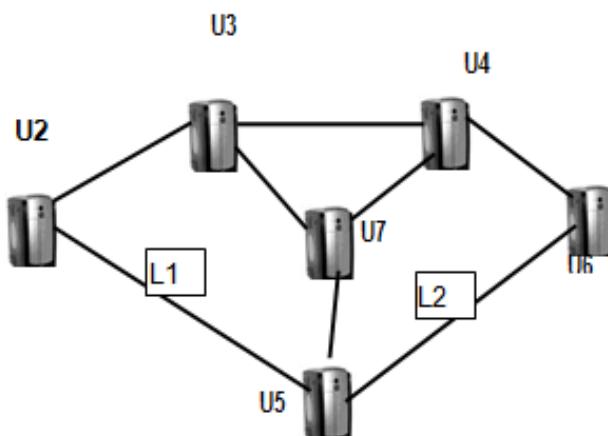
Cilj prilagodljivih algoritma usmeravanja je da postignu optimalni prenos podataka. Postoje različiti prilagodljivi algoritmi zavisno od toga:

- odakle uzimaju informacije - od okolnih ili svih usmerivača u mreži

- koliko često se izvršavaju - nakon nekog vremenskog intervala ili kada se promeni topologija mreže ili kada se promeni opterećenje mreže
- koja metrika se koristi za optimizaciju - broj skokova do odredišta ili razdaljina ili vreme potrebno za prenos

Algoritmi za usmeravanje koriste **princip optimalnosti**. Princip optimalnosti tvrdi da ako se neki usmerivač B nalazi na optimalnoj putanji između usmerivača A i C, onda se optimalna putanja između usmerivača B i C nalazi na istoj putanji.

Princip optimalnosti će biti ilustrovan na primeru mreže sa naredne slike. Neka se usmerivač U5 nalazi na optimalnoj putanji između usmerivača U2 i U6. Dakle, ona se sastoji od linija L1 i L2. Na osnovu principa optimalnosti, optimalna putanja između usmerivača U5 i U6 je linija L2. Ovo je sasvim logično, jer kada bi postojala bolja putanja između U5 i U6 ona bi se nadovezivala na liniju L1, što protivreći pretpostavci da je putanja L1,L2 optimalna.

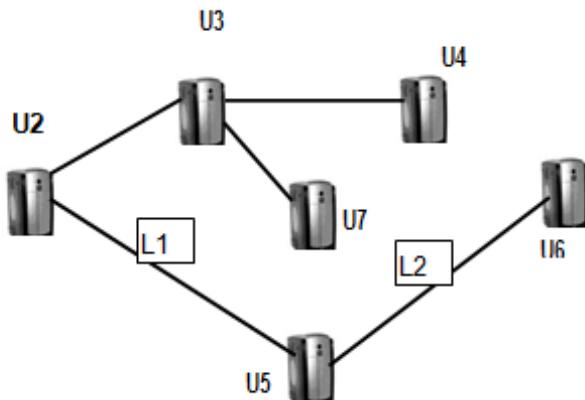


Slika 12.1 Princip optimalnosti [Izvor: Autor]

STABLO OPTIMALNOSTI

Na osnovu principa optimalnosti nameće se zaključak da će skup optimalnih putanja do pojedinih usmerivača činiti stablo čiji će koren biti u odredištu.

Na osnovu principa optimalnosti nameće se zaključak da će skup optimalnih putanja do pojedinih usmerivača činiti stablo čiji će koren biti u odredištu. Na slici je prikazano optimalno stablo kada je odredišni usmerivač U2. Kao merilo optimalnosti je uzet minimalni broj skokova. Treba primetiti, da ako je ovo jedini uslov, može doći do toga da se za jedan usmerivač pojavi više stabla optimalnih putanja. Takođe, stablo optimalne putanje sprečava da se pojavi petlja, tj. da jedan paket više puta prolazi preko jednog usmerivača.



Slika 12.2 Stablo optimalnosti [Izvor: Autor]

U praksi se koriste različiti algoritmi za usmeravanje, kao što su:

- usmeravanje *najkraćom putanjom*, koji je baziran na najmanjem broju skokova ili najmanjem kašnjenju u prenosu signala
 - usmeravanje zasnovano na *vektoru razdaljine*
 - Usmeravanje zasnovano na *stanju veze*
 - *hijerarhijsko usmeravanje* itd.

▼ Poglavlje 13

PROTOKOLI RUTIRANJA

PODELA PROTOKOLA RUTIRANJA

Dinamički usmerivači koriste algoritme za usmeravanje kako bi na osnovu informacija dobijenih od drugih usmerivača ažurirali tabele usmeravanja

Protokoli rutiranja se koriste za komunikaciju između usmerivača. Ovi protokoli su softverska implementacija algoritama usmeravanja. Dinamički usmerivači koriste algoritme za usmeravanje kako bi na osnovu informacija dobijenih od drugih usmerivača ažurirali tabele usmeravanja. Dobar protokol usmeravanja treba da ima sledeće karakteristike:

- brzu rekalkulaciju informacija potrebnih za tabele usmeravanja nakon promena uslova saobraćaja na mreži
- sprečava usmeravanje koje dovodi do stvaranja petlji u putanji paketa
- obezbeđuje optimalnu putanju kojom će paket biti prosleđen do odredišta na osnovu metrike usmeravanja.

Obzirom na administrativne granice mreže protokoli usmeravanja se mogu podeliti na:

- Interne protokole (engl. **Interior Gateway Protocols -IGP**) koji se koriste za razmenu informacija između usmerivača unutar nekog administrativnog područja ili autonomnog sistema.
- Eksterne protokole (engl. **Exterior Gateway Protocols- EGP**) koji se koriste za razmenu informacija između ruteru u različitim administrativnim sistemima ili između različitih autonomnih sistema.

Obzirom na vrstu algoritma koju koriste protokoli usmeravanja se mogu podeliti na:

- **protokole bazirane na vektoru rastojanja**(engl. **distance-vector routing protocols**) koji se zbog potrebe za obilnom komunikacijom koriste samo u manjim mrežama sa manjim brojem usmerivača
- **protokole bazirane na stanju veza**(engl. **Link-state routing protocols**) koji drugim usmerivačima šalju samo stanje njihovih veza ka drugim usmerivačima, minimizirajući na taj način komunikaciju što ih čini pogodnim za velike mreže sa velikim brojem usmerivača.

ROUTING INFORMATION PROTOCOL

RIP je intradomenski protokol usmeravanja koji može da funkcioniše za dati domen usmeravanja

Routing Information Protocol (RIP) je protokol koji se koristi u domenu koji ima manji i srednji broj mreža. Algoritam na kome je baziran definiše tabele usmeravanja samo na osnovu broja skokova do odredišne mreže. RIP je intradomenski protokol usmeravanja koji može da funkcioniše za dati domen usmeravanja. Koristi se za manje domene sa do 50 usmerivača. RIP je bio popularan osamdesetih godina prošlog veka, da bi ga zamenio OSPF protokol. Microsoft Windows NT Server, Windows 2000 Server i Windows .NET Server podržavaju RIP.

INTERIOR GATEWAY ROUTING PROTOCOL

IGRP koristi različite metrike za određivanje cene usmeravanja, kao što su opterećenje veza, propusnu moć veza, kašnjenje, pouzdanost i Maximum Transmission Unit (MTU)

Interior Gateway Routing Protocol (IGRP) je protokol baziran na algoritmu koji koristi vektor rastojanja. IGRP se koristi za domene koji imaju srednji ili veliki broj mreža. I on spada u klasu intradomenskih protokola usmeravanja. IGRP koristi različite metrike za određivanje cene usmeravanja, kao što su opterećenje veza, propusnu moć veza, kašnjenje, pouzdanost i Maximum Transmission Unit (MTU). Neki od ovih parametara se mere dinamički, a neki se unose od strane administratora mreže. IGRP podržava višeputansko usmeravanje radi balansiranja opterećenja veza i prevazilaženja problema nastalih kvarovima na vezama ili usmerivačima.

OPEN SHORTEST PATH FIRST

OSPF je hijerarhijski intradomenski protokol usmeravanja koji se uglavnom koristi unutar velikih autonomnih sistema

Open Shortest Path First ([OSPF](#)) je dinamički protokol baziran na algoritmu koji koristi informacije o stanju veza. Koristi se za manje, srednje i veće IP mreže, kao što su velike privatne kompanijske mreže koje se prostiru i u više zemalja ili Internet. OSPF je hijerarhijski intradomenski protokol usmeravanja koji se uglavnom koristi unutar velikih autonomnih sistema, a ne za usmeravanje između autonomnih sistema. Nastao je evolucijom ranijeg OSI protokola zvanog intermediate-system- to intermediate-system (IS-IS). Predložen je od strane Internet Engineering Task Force (IETF) kao zamena za RIP. OSPF podržava usmeravanje preko višestrukih putanja i koristi metriku. Podaci o aktivnim usmerivačima se dobijaju tako što usmerivač svakih 10 do 15 s šalje HELLO paket okolnim usmerivačima kako bi video da li su aktivni.

Popularnost OSPF protokola leži i u činjenici da je velike domene moguće hijerarhijski podeliti na manje poddomene. Usmerivači u jednom poddomenu onda vode računa samo o svojim usmerivačima, što čini tabele usmeravanja manjim. Veza između poddomena se obezbeđuje pomoću usmerivača graničnih područja (engl. **Area border router** - ABR).

EXTERIOR GATEWAY PROTOCOL I BORDER GATEWAY PROTOCOL

EGP ne koristi metriku, nego samo vodi računa o tome koje mreže su trenutno dostupne preko datog usmerivača

Exterior Gateway Protocol (EGP) je interdomenski protokol usmeravanja koji se koristi za usmeravanje između različitih domena usmeravanja koji su povezani glavnim vezama kao što je Internet. EGP se od 1984. godine koristi za usmeravanje između najvažnijih usmerivača na Internetu. EGP ne koristi metriku, nego samo vodi računa o tome koje mreže su trenutno dostupne preko datog usmerivača.

Border Gateway Protocol (BGP) je takođe interdomenski protokol kreiran specijalno za glavne usmerivače i kičmu Interneta. U odnosu na EGP je superiorniji jer ima mogućnost otkrivanja petlji i koristi metriku.

PRIKAZ PRIMERA PROTOKOLA RUTIRANJA

Routing Protocols - TYPES of Routing Protocol - BGP, OSPF, EIGRP, static routing, dynamic routing

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 14

Pokazne vežbe: PHP šabloni

STANJA

Osnovni način čuvanja stanja je kroz GET query parametre

Očekivano vreme izrade zadatka: 45 minuta.

Jedna od osnovnih karakteristika HTTP protokola je to da je protokol „stateless“. Ovo znači da protokol ne čuva stanje izmedju poruka, tj. svaka poruka se tretira kao posebna.

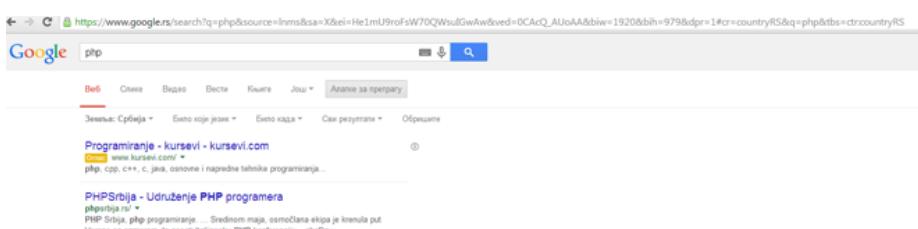
Ova karakteristika ima prednosti i mana. Osnovna prednost je da se stateless arhitektura može lako skalirati na više servera, a mana je dodatna kompleksnost u programiranju.

Generalno, životni ciklus PHP skripte se završi u deliću sekunde. Posle izvršenja skripte gube se vrednosti svih varijabli.

Medutim, često je potrebno da se pamti stanje u interakcijama sa korisnikom. Nekada je se ovo stanje čuva tokom kompletne interakcije, a nekada je samo čuva preko nekoliko stranica (conversational state), npr. u wizardima.

Osnovni način čuvanja stanja je kroz GET query parametre. Svaka dodatna opcija koju je korisnik izabrao se dodaje kao poseban query parametar. Ovo se često koristi kada je potrebno implementirati pretragu sa više parametara. Prednost ove metode je što generiše URL kakav je moguće kopirati ili bookmarkovati.

Dobar primer ove metode je Google pretraga. U URL baru na slici 1 vidi se veliki broj parametara. Kada je npr. izabrana zemlja, dodan je parametar **country=RS**.



Slika 14.1.1 Google pretraga [Izvor: Autor]

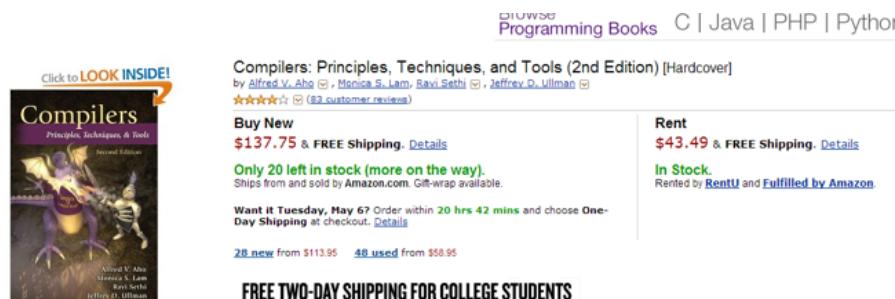
MASTER DETAIL NAVIGACIJA

Najlakši način da se postigne ovaj stil navigacije je da svaka stavka iz liste ima jedinstven ID u bazi

Čest slučaj u web programiranju je zahtev da se korisniku prikaže lista sa više elemenata, a da se klikom na neki element liste, na drugoj stranici prikažu detalji tog elementa



Slika 14.1.2 Master strana [Izvor: Autor, snapshot]



Slika 14.1.3 Detail strana [Izvor: Autor, snapshot]

Na slikama 2 i 3 prikazana je Master Detail navigacija sa sajta amazon.com.

Najlakši način da se postigne ovaj stil navigacije je da svaka stavka iz liste ima jedinstven ID u bazi. Taj ID se onda postavlja kao query parametar u svim elementima liste. Kada korisnik klikne na jedan element iz liste, preusmerava se na drugu stranu, koja čita ID iz query parametra, nalazi red sa tim identifikatorom u bazi i prikazuje ga.

KOLONE ZA MASTER/DETAIL NAVIGACIJU

Prikaz koda za primer Master/Detail navigacije

Za primer Master/Detail navigacije kreiraćemo listu filmova na master stranici i njihov detaljniji prikaz na detail stranici.

Prvo je potrebno kreirati bazu podataka i u njoj tri tabele koristeći phpMyAdmin. Tabele su prikazane na slici.

vezbe11_film	vezbe11_uloga	vezbe11_glumac
idFilm : int(11)	idUloge : int(11)	idGlumac : int(11)
nazivFilma : text	# idFilm : int(11)	imeGlumca : text
# godinalzlaska : int(11)	# idGlumac : int(11)	prezimeGlumca : text
reziserFilma : text		
slikaFilma : text		

Slika 14.1.4 Potrebne tabele i njihove kolone [Izvor: Autor]

Nakon kreiranja tabela i njihovih kolona potrebno je popuniti ih. Tabela uloga predstavlja među-tabelu između tabela film i glumac. Kako jedan film ima više glumaca, ali i jedan glumac može da glumi u više različitih filmova javlja se veza više-na-više pa se iz tog razloga kreira tabela uloga koja sadrži id filma i id glumca.

Prvo je potrebno povezati se na kreiranu bazu podataka pomoću PHP skripte koju ćemo nazvati connect.php.

```
<?php
try{
    $con = new PDO("mysql:host=localhost;dbname=vezbe11", "root","");
    $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e){
    echo "Error: " . $e->getMessage();
}
?>
```

Zatim kreiramo stranicu na kojoj će biti lista filmova iz baze.

```
<?php
    require_once("connect.php");
?>
<!DOCTYPE html>
<html>
<head>
    <title>Master</title>
</head>
<body>
    <h1>Lista filmova</h1>
    <div>
        <?php
            $data = $con->query("SELECT * FROM film");
            if($data) {
                $i = 0;
                while ($row = $data->fetch()) {
                    echo "<div style='float: left'><a href='detail.php?id=" .
$row["idFilm"] ."><img width='350px' height='400px' src='".$row["slikaFilma"] . "'/></a><div>Naziv: " . $row['nazivFilma'] . "<br/>Godina: " .
$row['godinaIzlaska'] . "<br/><a href='detail.php?id=" . $row["idFilm"] . "'>Read more...</a></div></div>";
                }
            }
        </?php>
    </div>
</body>
</html>
```

```
}
```

```
?>
```

```
</div>
```

```
</body>
```

```
</html>
```

Pre svega, sa require_once() bilo je neophodno učitati fajl koji kreira konekciju na bazu.

Potom se u while petlji kreiraju div elementi sa podacima o filmu redom. Svaki film na tekstu Read more... ima postavljen link ka detail.php stranici, ali pri čemu se šalje id tog filma.

```
<?php
    require_once("connect.php");
    $idFilma = $_GET["id"];
?>
<!DOCTYPE html>
<html>
<head>
    <title>Detail</title>
</head>
<body>
    <?php
        $stmt = $con->prepare("SELECT * FROM film JOIN uloga ON film.idFilm =
uloga.idFilm JOIN glumac ON uloga.idGlumac = glumac.idGlumac WHERE film.idFilm =
:id");
        $stmt->bindParam(":id", $idFilma);
        $stmt->execute();
        $glumci = "";
        echo "<center>";
        while ($data = $stmt->fetch()) {
            $film = "<h1>" . $data["nazivFilma"] . "</h1><br/>" . "<img src='"
. $data["slikaFilma"] . "' width='25%' height='25%' /><br/> Godina: " .
$data["godinaIzlaska"] ."<br/> Reziser: " . $data["reziserFilma"] ."<br/> Glumci: ";
            $glumci = $glumci . $data["imeGlumca"] . " " . $data[" prezimeGlumca"]
. " ";
        }
        echo $film . $glumci . "<br/><a href='master.php'>Nazad</a></center>";
    ?>
</body>
</html>
```

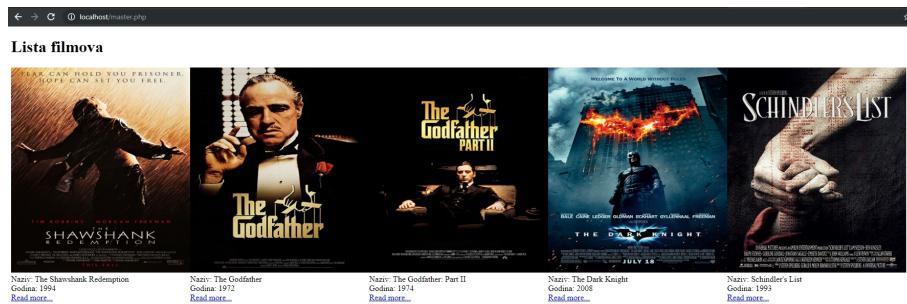
Na detail.php strani je potrebno uzeti id iz parametara i potom ga iskoristiti u upitu tako da se izvuku podaci o filmu sa tim id-jem.

Kako bi izvukli podatke iz svih tabela moramo koristiti JOIN. JOIN omogućava kombinovanje redova iz dve ili više tabela, a na osnovu povezanih kolona između njih. U ovom primeru potrebno je JOIN-ovati sve tabele, tabelu film kombinujemo sa tabelom uloga preko kolone idFilm iz jedne i druge tabele, a na sličan način se dalje povezuje sa glumac.

PRIMER MASTER/DETAIL NAVIGACIJE

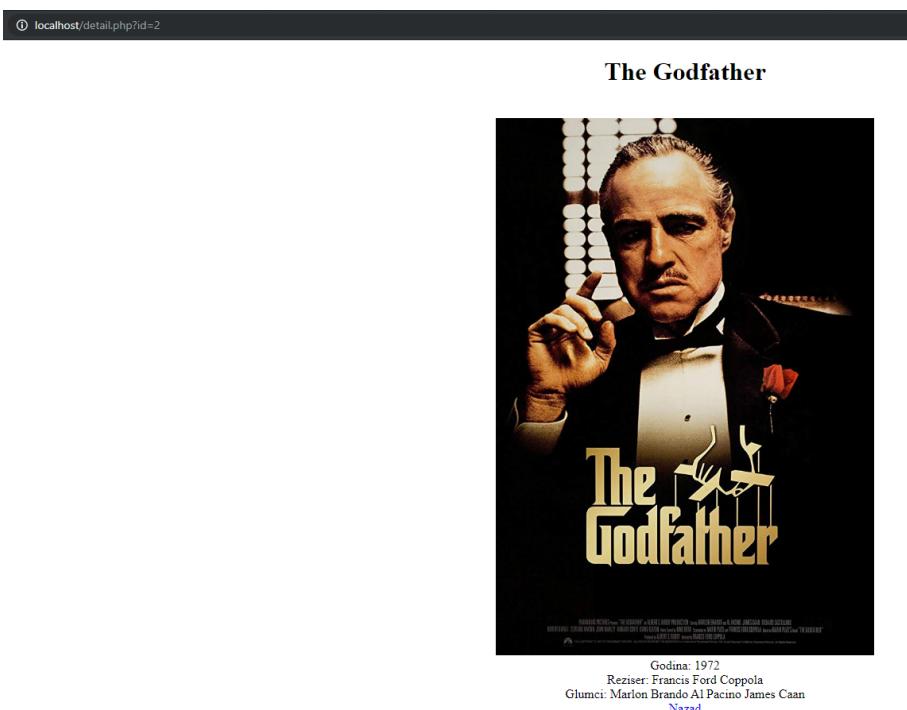
Prikaz izgleda rešenja za primer Master/Detail navigacije

Prikaz master.php strane se nalazi na slici 5.



Slika 14.1.5 Master strana [Izvor: Autor]

Na slici 6 je prikazan izgled detail strane.



Slika 14.1.6 Detail strana [Izvor: Autor]

REDIRECT AFTER POST

Pošto POST zahtev u većini slučajeva ima posledice, pojavljuje se problem u slučaju da korisnik slučajno pošalje jedan zahtev više puta

Po HTTP standardu, POST i PUT zahtevi nisu idempotentan, što u osnovni znači da zahtevi ovog tipa mogu prouzrokovati „nuspojave“ (side effects). GET zahtev je idempotentan.

Šta će server uraditi kao odgovor na bilo koji od ovih zahteva zavisi od programera, ali važno je poštovati pravilo da GET zahtev nikada ne bi trebao da pravi bilo kakve izmene na serveru, tj. u bazi podataka.

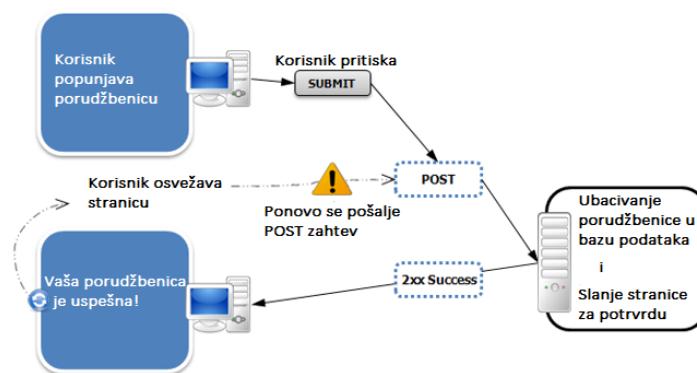
Pošto POST zahtev u većini slučajeva ima posledice, pojavljuje se problem u slučaju da korisnik slučajno pošalje jedan zahtev više puta, npr. korišćenjem refresh dugmeta.

U tom slučaju, browser će prikazati poruku upozorenja.

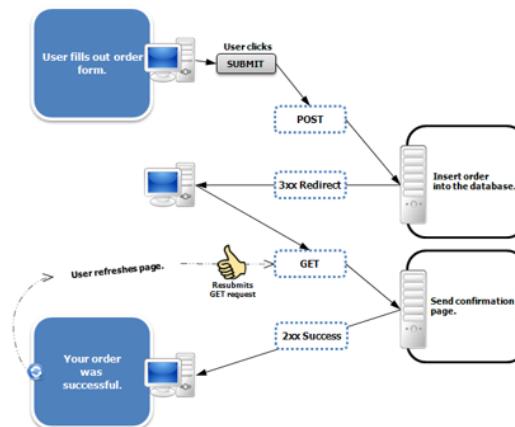
Rešenje se posle POST zahteva vrati redirect, koji će naterati browser da pošalje GET zahtev za sledeću stranu.

Redirect se može poslati na sledeći način:

```
header( 'Location: http://www.yoursite.com/new_page.html' );
```



Slika 14.1.7 POST bez preusmerenja [Izvor: Autor]



Slika 14.1.8 POST sa preusmerenjem [Izvor: Autor]

ORGANIZACIJA KODA

Najjednostavniji način je da se kod logike aplikacije, i kod za pristup bazi odvoji u posebne fajlove, sa funkcijama i klasama koje mogu da se koriste na više mesta

Kako codebase aplikacije raste, tako postaje jasno da je potrebno organizovati kod aplikacije na neki način. Veliki fajlovi koji u sebi sadrže PHP, SQL i HTML nisu čitljivi.

Postoji mnogo PHP biblioteka (framework) za organizaciju koda, ali za početak je dovoljno izdvojiti što više HTML koda od logike aplikacije, i pristupa bazi.

Naime, potrebno je voditi računa da kod HTML stranica sadrži samo onoliko PHP koda koliko je potrebno da se dinamički renderuje sadržaj strane, ali ne i logiku aplikacije.

Najjednostavniji način je da se kod logike aplikacije, i kod za pristup bazi odvoji u posebne fajlove, sa funkcijama i klasama koje mogu da se koriste na više mesta.

Takvi fajlovi se mogu uključiti u trenutni fajl sledećim funkcijama: **include**, **require** i **require_once**.

▼ 14.1 Zadaci za samostalnu vežbu

OPIS ZADATKA ZA SAMOSTALNU VEŽBU

Kreirati HTML stranu sa formom za unos podataka

Očekivano vreme izrade zadatka: 45 minuta.

Koristeći deo zadatka sa prethodnih vežbi za samostalni rad kreirati:

- HTML stranu sa formom za unos podataka o igračkama koje se prodaju u prodavnici.
- Svaka igračka ima sledeće podatke: ID, ime, ime proizvođača, datum proizvodnje, cenu.
- Svaka uneta igračka mora se upisati u bazi koja je unapred kreirana.
- Na novoj strani kreirati tabelu u kojoj će se ispisati ime igračke i cena, a pored toga treba da postoji i kolona detaljnije u kojoj će se nalaziti link ka detail strani za tu igračku.
- Na detail strani prikazati sve podatke o odabranoj igrački. Ukoliko je cena igračke veća od 2000 dinara ispisati je crvenom bojom, u suprotnom ispisati je zelenom bojom.

▼ 14.2 Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ11

Kreirati bazu podataka sa podacima o artiklima i korisnicima na web aukciji

Očekivano vreme izrade zadatka: 45 minuta.

- Kreirati bazu podataka sa podacima o artiklima i korisnicima na web aukciji.
- Kreirati formu za unos korisnika i upis podataka o artiklima.

- Unutar forme za novi artikl treba da se nalazi padajuća lista sa korisnicima tako da se može definisati koji artikl je od kog korisnika.
- Kreirati stranu na kojoj će se prikazati nazivi svih artikala, a klikom na naziv prikazuje se strana sa detaljima o izabranom artiklu.
- Na strani gde se prikazuju svi artikli dodati opciju za njihovo brisanje.
- Koristiti prikazane šablone u izradi zadatka.

Potrebne datoteke snimiti pod imenom:

IT210-DZ11_Ime_Prezime_brojIndexa

gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovane datoteke poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210-DZ11)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovoj lekciji bilo je reči o osnovnim konceptima računarskih mreža. Pored toga što je objašnjeno šta je to računarska mreža i koji su njeni osnovni delovi, obrađena su dva ključna mrežna referentna modela, ISO OSI i TCP/IP. Iako slični, među njima postoji razlika koja na drugačiji način opisuje višeslojni rad mreže. Kao bitan segment računarskih mreža, opisani su osnovni principi rutiranja, kao i osnovni algoritmi i protokoli koji se koriste prilikom rutiranja paketa kroz mrežu.

Literatura

- [1] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall, 2003.
- [2] Werner Feibel, Enciklopedija računarskih mreža, Mikro kniga, 1995.
- [3] James F. Kurose, Keith W. Ross, Computer networking, Addison Wesly, 2001.



IT210 - SISTEMI IT

RAČUNARSKE MREŽE I
BEZBEDNOST

Lekcija 12

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 12

RAČUNARSKE MREŽE I BEZBEDNOST

- ▼ RAČUNARSKE MREŽE I BEZBEDNOST
- ▼ Poglavlje 1: Značaj mreže
- ▼ Poglavlje 2: Aplikaciono područje mreža - Multimedijalne tehnologije
- ▼ Poglavlje 3: Osiguranje informacija i bezbednost
- ▼ Poglavlje 4: Model za osiguranje informacija
- ▼ Poglavlje 5: Osnove kriptografije i kriptosistema
- ▼ Poglavlje 6: Pokazne vežbe: SQL injection napad
- ▼ Poglavlje 7: Zadaci za dodatnu vežbu
- ▼ Poglavlje 8: DOMAĆI ZADATAK
- ▼ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Fokus ove lekcije je na bezbednosti informacija i multimedijalnim aplikacijama u računarskim mrežama

U ovom predavanju biće reči o:

- Multimedijalnim tehnologijama
- Osiguranju informacija i bezbednost
- Fundamentalnim aspektima bezbednosti
- Model za osiguranje informacija (pretnje, ranjivost, napadi, protivmere)
- Bezbednosni mehanizmi - Kriptografija i kriptosistmi
- Autentifikacija

✓ Poglavlje 1

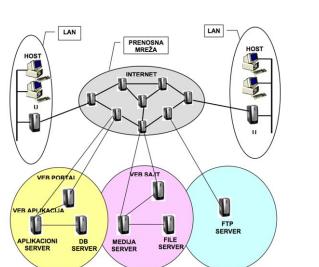
Značaj mreže

ZNAČAJ KORIŠĆENJA RAČUNARSKE MREŽE

Danas se Internet i WWW koristi ne samo za pregled HTTP strana, nego i za komunikaciju, pristup različitim sadržajima, korišćenje veb servisa, pristup bazama podataka, pristup deljenim

Internet je prvo bitno razvijen sa idejom da se omogući prenos podataka između udaljenih računara. Ideja koja je pokrenula razvoj WWW je bila da se omogući čitanje hipertekstualnih dokumenata. Internet i WWW su u međuvremenu evoluirali u mnogo korisnije okruženje. Danas se Internet i WWW koristi ne samo za pregled HTTP strana, nego i za komunikaciju, pristup različitim multimedijalnim sadržajima, korišćenje veb servisa, pristup bazama podataka, pristup deljenim datotekama, pristup veb i izvršavanje veb aplikacija itd.

U svim ovim slučajevima računarske mreže su neophodna infrastruktura. Značaj mreže će se ilustrovati na primerima njihovog korišćenja prikazanim na slici 1. Tipična veb aplikacija ima zadatak da bude raspoloživa svim autorizovanim korisnicima gde god da se oni nalazili. Oni mogu biti u okviru iste lokalne mreže, u okviru virtualne privatne mreže ili na nekoj drugoj udaljenoj mreži.



Slika 1.1 Primer značaja korišćenja računarske mreže [Izvor: Autor]

Korisnik može da koristi stacionarni računar, mobilni računar ili neki drugi mobilni uređaj koji ima pristup mreži. Zahvaljujući računarskoj mreži udaljeni korisnici mogu sa bilo koje tačke na Internetu da koriste aplikaciju.

Bez obzira gde se nalazi korisnik pristupa aplikaciji preko veb portala, koji između ostalog vrši autentikaciju korisnika i dozvoljava mu da počne da koristi aplikaciju. Sama aplikacija se izvršava na aplikacionom serveru, koji treba da ima dovoljnu procesorsku snagu da istovremeno opslužuje više klijenata.

Gotovo po pravilu, sve veb aplikacije koriste neku bazu podataka. Aplikacija priprema upit i šalje ga DB serveru na kome se nalazi sistem za upravljanje bazom podataka. Uobičajeno je ova baza na posebnom DB serveru, posebno konfigurisanim da može da obradi veliki broj

upita u jedinici vremena. Kao što se vidi, svi ovi računari imaju potrebe da komuniciraju, pa su takođe povezani u mrežu. Ovo je najčešće jedna lokalna mreža, ali nisu retki slučajevi veb aplikacija kada su ovi računari na različitim LAN-ovima. Baza podataka, dalje može biti na jednom računaru, a može biti i distribuirana. Bez pouzdanije i brze mrežne komunikacije, korišćenje veb aplikacija je neefikasno.

PRIMER KORIŠĆENJA MREŽA

Za datoteke koje više korisnika treba da koristi formiraju se serveri koji rade sa FTP-om, pa se oni nazivaju FTP serveri

Primer korišćenja mreža za aplikacije može da bude veb sajt koji korisnicima nudi emitovanje multimedijalnih sadržaja. I u ovom slučaju se celokupan posao deli na specijalizovane računare. Za čuvanje ogromne količine podataka koja predstavlja multimedijalnu arhivu koriste se specijalni fajl serveri koji pored brzih diskova imaju i magnetne trake, na kojima se nalazi celokupna arhiva. Kada korisnik odabere sadržaj koji želi, veb sajt prosleđuje njegov zahtev medija serveru. Ukoliko u memoriji medija servera nema tog sadržaja, on ga traži od fajl servera. Zatim medija server počinje stvaranje traženog multimedijalnog zapisa. U ovom slučaju se od mreže očekuje da ima dovoljan kapacitet da obezbedi emitovanje zapisa u realnom vremenu.

Jedan od najstarijih servisa na Internetu je bio prenos datoteka. Za datoteke koje više korisnika treba da koristi formiraju se serveri koji rade sa **File Transfer Protocol-om**, pa se oni nazivaju FTP serveri. U principu FTP serveri pružaju dve usluge: prijem datoteke (engl. **upload**) i slanje datoteke (engl. **download**). Pored toga, autorizovani korisnici imaju mogućnost promena imena datoteka, njihovo premeštanje i brisanje.

Korisnik koji želi da koristi usluge FTP servera treba da ima instaliran neki FTP klijent.

Većina današnjih operativnih sistema i veb čitača ima FTP klijent. FTP server osluškuje mrežu kako bi prihvatio zahteve FTP klijenata. U svim prethodno navedenim primerima neophodno je dobro funkcionisanje mreže ne samo lokalne mreže, nego i celog Interneta, jer su korisnici često na drugoj strani zemljine kugle. Zbog toga je važno da okosnica Interneta (engl. **backbone**) i svi usmerivači neprekidno funkcionišu. Postoje mnogi razlozi zbog kojih može da dođe do poremećaja saobraćaja na Internetu. Nestanak napajanja, prekid u kablovima, kvar na usmerivaču, atmosferske smetnje kod bežičnog prenosa, hakerski napadi na mrežu i drugi razlozi mogu da učine prenos podataka neefikasnim i time ugroze funkcionisanje aplikacija. Prestanak rada jednog usmerivača, na primer zbog nestanka struje, može da se prevaziđe time što će se izmeniti tabele usmeravanja i naći će se alternativni put. Ovo će, razume se povećati opterećenje pojedinih linija na Internetu. Ako bi, međutim, više okolnih usmerivača prekinulo rad može doći do potpunog zagrušenja u saobraćaju. Neki od programa kontrolišu učestalost komunikacije sa klijentima i ako se ona ne događa dovoljno dugo prekidaju sesiju. U ekstremnim slučajevima može da se dogodi da zbog otkaza više usmerivača saobraćaj između lokalnih i regionalnih mreža u nekim delovima Interneta potpuno zamre.

▼ Poglavlje 2

Aplikaciono područje mreža - Multimedijalne tehnologije

MULTIMEDIJALNE APLIKACIJE

Pod multimedijalnim aplikacijama se podrazumevaju aplikacije koje kombinuju različite tipove informacija koje se emituju, uobičajeno audio i video.

Računarske mreže predstavljaju infrastrukturu za mnoge multimedijalne aplikacije. Pod multimedijalnim aplikacijama se podrazumevaju aplikacije koje kombinuju različite tipove informacija koje se emituju, uobičajeno audio i video. Ukoliko korisnik informacija ima mogućnost interakcije, onda se radi o interaktivnoj multimediji.

Tipične multimedijalne aplikacije su:

- audio na zahtev
- Internet radio
- Internet telefonija
- video na zahtev
- video konferencije

Ovde se može postaviti pitanje da li Internet telefonija i radio spadaju u multimedijalne aplikacije? Striktno gledano, pošto multimedija znači više paralelnih medija, ove aplikacije ne bi trebalo da spadaju u kategoriju multimedijalnih, ali se one u većini literature [1, 2, 3] svrstavaju u multimedijalne aplikacije.

▼ 2.1 Audio na zahtev

PRINCIP RADA AUDIA NA ZAHTEV

Audio na zahtev je tehnologija kojom se odnosi na emitovanje zvučnih zapisa preko Interneta

Audio na zahtev je tehnologija kojom se odnosi na emitovanje zvučnih zapisa preko Interneta.
Provajder audio zapisa ima medija server čiji je zadatak da strimuje tražene audio zapise.
Audio zapis može biti neki govor, snimljeni zvuk, a najčešće su muzičke numere. Tipična

muzička numera u mp3 formatu ima veličinu od oko 4 MB. Oni se mogu nalaziti na istom računaru ili na nekom fajl serveru.

Tipičan scenario se odigrava tako što korisnik preko neke veb strane odabere audio zapis koji želi da čuje, a on se zatim sa nekog medija servera emituje (odnosno strimuje) korisniku koji sluša zapis u realnom vremenu. Korisnik ne može da zapiše audio zapis na svom računaru, već ih interpretira iz bafera koji se nalazi u RAM memoriji.

Da bi korisnik mogao da interpretira primljeni zvuk treba da ima neki program za čitanje, dekomprimovanje i interpretaciju zvučnih zapisa tipa **RealOne Player, Windows Media Player ili Winamp**. Za prenos podataka se koristi RTP protokol koji radi nad UDP protokolom, tako da neki paketi mogu biti izgubljeni tokom transporta. Plejeri raspolažu logikom koja prevazilazi ove probleme.

Audio plejeri čitaju podatke iz bafera koji se puni emitovanim strimom podataka od strane servera. **Funkcija bafera je da obezbedi minimalnu rezervu podataka, tako da se interpretacija zapisa vrši bez zastoja**, čak i u slučaju da dođe do zastoja u strimovanju. Na taj način korisnik ne mora da čeka da mu pristigne ceo zapis da bi počeo njegovu interpretaciju. Zašto je ovo važno, može se ilustrovati na primeru korisnika koji želi da čuje jednu mp3 muzičku numeru, a ima modemsku vezu. On bi trebao da čeka 10 minuta da ceo zapis pristigne do njegovog računara.

U radu sa baferom je moguće da se javi jedan od dva problema. Interpretacija zapisa može da bude sporija od emitovanja od strane servera. U tom slučaju bi se bafer prepunio. Drugi problem može da se javi kada je pristizanje strima u bafer sporije od emitovanja. Ukoliko to potraje, bafer će se isprazniti. Zbog toga se **uvodi gornja i donja granica popunjenoosti bafera**. Kada se dostigne gornja granica, od servera se zahteva da privremeno prekine strimovanje. Kada se dostigne donja granica, od servera se zahteva da nastavi strimovanje. Donja granicu u baferu treba da bude usaglašena sa stanjem veza. Ukoliko su veze slabije, donja granica treba da bude na višem nivou.

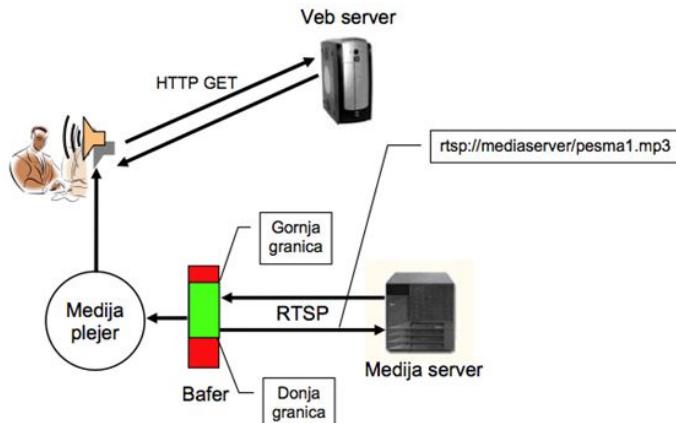
PROCEDURA EMITOVARJA AUDIO ZAPISA NA ZAHTEV

Incijalizacija audia na zahtev se obavlja HTTP GET zahtevom

Procedura emitovanja audio zapisa na zahtev bi mogla da se opiše sledećim koracima:

1. Korisnik **pomoću veb klijenta i preko HTTP protokola** pristupa veb serveru i zahteva stranicu sa listom audio zapisa
2. **Veb server šalje traženu stranicu klijentu**
3. **Veb strana se prikazuje u veb čitačuna** klijentovom računaru
4. Klijent izabere audio zapis koji želi da sluša čime je u stvari **poslao zahtev medija serveru da mu emituje neki audio zapis preko RTSP protokola**
5. Medija **server počinje da šalje audio strim** klijentu
6. Audio **strim** sena klijentskoj strani **smešta u bafer**
7. **Kada se bafer napuni iznad donje granice, veb čitač aktivira medija plejerna** klijentskom računaru koji interpretira zapis iz bafera i interpretirani deo briše iz bafera. Za to vreme medija server nastavlja da puni bafer.

8. Ako količina podataka u baferu dostigne gornju granicu, medija server dobija informaciju da prekine strimovanje
9. Kada količina podataka u baferu padne ispod donje granice, od medija servera se zahteva da nastavi strimovanje.



Slika 2.1.1 Princip rada audia na zahtev [Izvor: Autor]

✓ 2.2 Internet radio

NAČINI EMITOVARJA INTERNET RADIJA

Za emitovanje radija program se može snimiti unapred i slušaoci tom programu mogu da pristupe kasnije ili emitovanje može da se obavlja uživo

Za emitovanje radija postoje dva pristupa. U prvom pristupu, program se snima unapred i zapisuje na medija serveru. Slušaoci kasnije mogu da pristupe programu i da slušaju odgovarajuću emisiju. Ovo je praktično radio na zahtev, ali se po tehnologiji ne razlikuje od strimovanja audio zapisa. Snimljeni program, može biti i arhiva radijskog programa koji se normalno emituje uživo, tako da je slušaocima ostavljena mogućnost da neku emisiju čuju u vreme kada njima odgovara.

Drugi pristup je emitovanje programa uživo. I ovde postoji potreba za baferom koji bi čuvao u rezervi dvadesetak sekundi programa kojim bi se prevazišli problemi koji nastaju zbog neredovnog pristizanja paketa.

Razlika između audia na zahtev i Internet radija je u tome što se emitovanje radio programa vrši u realnom vremenu, pa brzina reprodukovanja treba da bude ista. Na žalost, to ne mora uvek da bude slučaj, tako da pri manjoj brzini reprodukovanja od emitovanja može doći do prepunjivanja bafera, čime se neki podaci gube.

Druga razlika je u tome što Internet radio može imati na hiljade slušaoca istovremeno pa se prenos vrši po principu multikasta, dok se audio na zahtev vrši samo za jednog

slušaoca, pa se prenos vrši po principu unikasta. Zbog toga Internet radio koji radi na principu multikasta koristi Real-time Transport Protocol za komunikaciju sa slušaocima. Na žalost, mnogi provajderi Internet usluga ne podržavaju ovaj protokol, pa se prenos vrši po unikast principu preko TCP protokola.

▼ 2.3 Internet telefonija

MODEL INTERNET TELEFONIJE

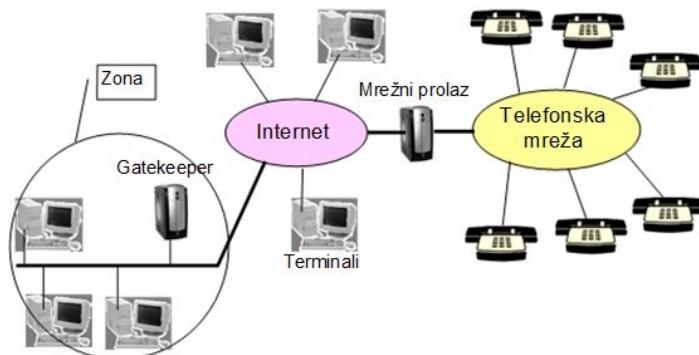
H.323 definiše arhitekturu Internet telefonije , a referencira se na skupu drugih protokola koji služe za kodiranje govora, pozivanje, slanje signala, prenos podataka i dr.

Nekada se javni komutirani telefonski sistem koristio pre svega za prenos glasa, tj. za telefoniranje, a nešto manje i za prenos podataka. Međutim, provajderi Internet usluga su primetili da bi takođe mogli da pruže usluge telefoniranja koristeći postojeću mrežnu infrastrukturu koju nudi Internet. Ova vrsta usluge se naziva Internet telefonija ili Voice over IP(VoIP). Pošto cena prenosa podataka na Internetu ne zavisi od rastojanja između izvorišne i odredišne strane, telefoniranje preko Interneta postalo je mnogo jeftinije od telefoniranja korišćenjem klasične telefonske mreže. Druga prednost Internet telefonije je u tome da se poziv automatski preusmerava ka mestu na mreži gde se korisnik prijavio.

Za Internet telefoniju se koriste dva standarda: H.323 koji predstavlja preporuku UTI-ja i SIP koga je razvila IETF.

H.323 preporuka, koja je razvijena od strane pružaoca telefonskih usluga 1996. godine, ima originalni naziv: "Visual Telephone Systems and Equipment for Local Area Networks Which Provide a NonGuaranteed Quality of Service". Prvi sistemi za telefoniranje preko Interneta su bili zasnovani na ovim preporukama. H.323 definiše arhitekturu Internet telefonije , a referencira se na skupu drugih protokola koji služe za kodiranje govora, pozivanje, slanje signala, prenos podataka i dr.

Na slici 1 je prikazan model na kome je zasnovana Internet telefonija. U centru modela je mrežni prolaz koji spaja Internet i klasičnu telefonsku mrežu. Sa jedne strane su korisnici Interneta, a sa druge, korisnici telefonske mreže. Zbog toga mrežni prolaz koristi dva protokola: H.323 za Internet stranu i PSTN(engl. Public Switched Telephone Network) za stranu telefonske mreže. Komunikacioni uređaji se nazivaju terminali. Neka lokalna mreža može da ima čuvara (engl. gatekeeper) koji upravlja krajnje tačke pod njegovom jurisdikcijom, koje se zajedno nazivaju zona.



Slika 2.2.1 Model na kome je zasnovana Internet telefonija [Izvor: Autor]

PROTOKOLI INTERNET TELEFONIJE

Za prenos podataka se koriste RTP/RTCP protokoli. SIP može da bude implementiran nad UDP ili TCP protokolu

Svi H.323 sistemi za kodiranje i dekodiranje govora se koristi protokol [G.711](#). On koristi 8-bitno kodiranje da kodira 8000 uzoraka govora u sekundi, što znači da je za prenos ovakvog nekomprimovanog govora potreban kapacitet veze od 64 Kb/s. Pored ovog, postoji i drugi protokoli za kodiranje govora, kao što je [G.723](#).

Protokol [H.245](#) definiše kojom se brzinom prenosi govor i koja vrsta kodiranja se koristi. Standard [ITU Q.931](#) se koristi kao protokol kojim se definiše uspostavljanje i oslobođanje veze, pozivni signal, zvuk zvonjenja telefona itd. Za komunikaciju sa čuvarem zone terminali koriste [H.225](#) protokol. Čuvar zone nije obavezna komponenta Internet telefonije. On se postavlja iz administrativnih razloga, kako bi se u jednoj lokalnoj mreži dozvoljavao pristup Internoj telefoniji.

Govor	Upravljanje				
	G.7xx	RTCP	H.225 (RAS)	Q.932 (pozivni signal)	H.245 (upravljanje pozivom)
RTP	UDP				
	IP				
	Protokoli sloja veze				
	Protokoli fizičkog sloja				

Slika 2.2.2 Protokoli internet telefonije [Izvor: Autor]

Internet zajednica je preko svog tela IETF donela svoj protokol za Internet telefoniju koji se zove **Session Initiation Protocol- SIP**. On je opisan dokumentom RFC 3261. SIP definiše Internet telefoniju, video konferencije i druge multimedijalne veze. On definiše brojeve telefona kao URL-ove. SIP definiše tri vrste sesija:

- dvo-partnersku sesiju, koja odgovara običnom telefonskom pozivu
- multipartnersku sesiju, što omogućuje da više korisnika istovremeno govori i čuje i
- multikast sesiju, kada jedan korisnik emituje podatke, a ostali ih primaju.

U bilo kojoj sesiji se mogu razmenjivati audio, video i podaci. Za prenos podataka se koriste RTP/RTCP protokoli. SIP može da bude implementiran nad UDP ili TCP protokolu.

❖ 2.4 Video na zahtev

NAČIN PRENOSA PODATAKA ZA VIDEO NA ZAHTEV

Prenos podataka se može obaviti strimovanjem i „set-top box“-ovima

Video na zahtev (engl. **video on demand (VOD)**) omogućava korisnicima da pristupe interaktivnom televizijskom sistemu i izaberu i gledaju video zapis. Korisnik je u mogućnosti da tokom gledanja zapisa koristi klasične VCR komande kao što su **pause**, **fast forward** ili **rewind**. Radi se o relativno novoj tehnologiji koja još tehnološki nije sazrela.

VOD sistem koristi dva metoda za prenos podataka do korisnika. Prvi metod je klasično strimovanje koje je vrlo zahtevno sa aspekta medija servera i kapaciteta prenosnih puteva, a posebno je kritično izvršavanje VCR komandi. Drugi način je da se ceo film prethodno prevuče u takozvane „**set-top box**“-e, koji su u stvari specijalizovani PC računari, a tek onda gleda. U ovom slučaju, pošto su podaci kod korisnika, implementacija VCR komandi je vrlo jednostavna. I u jednom i u drugom slučaju, za svakog korisnika se emituje poseban strim, a korisnik treba da ima jako brze veze sa Internetom.

Da bi se prevazišao problem ogromnih prenosnih kapaciteta koje zahteva VOD, uvedena je tehnologija „**near video on demand**“. U ovom slučaju televizijska kuća emituje različite popularne video zapise, ali tako što jedan isti video zapis počinje da emituje svakih 10 minuta.

Ovde korisnici takođe biraju koji će zapis gledati, ali nemaju mogućnost da prekinu emitovanje ili da koriste druge VCR komande. Umesto toga, oni mogu da prekinu gledanje videa, a zatim da nastave na nekom od strimova istog zapisa koji je počeo kasnije. Sa aspekta servera, ovaj pristup je mnogo racionalniji jer umesto velikog broja strimova, emituje samo 10-tak strimova za svaki video zapis.

Poslednja tehnologija koja se pojavila vezano za multimediju na Internetu je **Over IP Video (OIPV)**. U ovom slučaju se za prenos video zapisa koristi Internet ili 3G telefonske mreže. Oba sistema koriste IP protokol za pristup Internetu.

▼ Poglavlje 3

Osiguranje informacija i bezbednost

ISTORIJA I TERMINOLOGIJA

Sa pojavom personalnih računara, računarskih mreža, a kasnije i Interneta, osiguranje informacija i bezbednost postaju mnogo veći problem

Osiguranje informacija i bezbednost informacionih sistema su se pojavili kao problem sa prvim komercijalnim primenama računarskih sistema. Međutim, 50-tih i 60-tih godina prošlog veka, broj korisnika računarskih sistema je bio relativno mali i svodio se na timove stručnjaka u računarskim centrima i istraživačkim laboratorijama. Računarski sistemi su bili nepovezani i nije postojao način za spoljni upad u sistem. Stoga se osiguranje bezbednosti svodilo na osiguranje fizičkog pristupa računarskim sistemima.

Sa pojavom personalnih računara, računarskih mreža, a kasnije i Interneta, osiguranje informacija i bezbednost postaju mnogo veći problem.

Inženjeri u Xerox-ovom Paolo Alto istraživačkom centru su 1979. godine napisali kratak program, koga su nazvali „crv“, čiji je zadatak bio da se šeta po mreži i traži neuposlene procesore. Iako je razlog za pisanje ovakvog programa bio efikasnije korišćenje računara, ovaj crv je postao prethodnik modernih crva - destruktivnih kompjuterskih virusa koji menjaju ili brišu podatke ili čine datoteke neupotrebljivim.

Pojavljivanjem prvih personalnih računara i modemskih komunikacija stvoreni su i novi mehanizmi za ugrožavanje sigurnosti. Tako je 1983 FBI otkrio grupu mladih hakera koja je pomoću modema i Apple II računara uspela da izvrši upad u nekoliko vladinih računarskih mreža.

Doktorant Univerziteta u Južnoj Kaliforniji, Fred Cohen, prvi je upotrebio termin računarski virus (engl. computer virus) da opiše računarski program koji može da menja druge programe tako što u njih umeće svoju kopiju.

Jedan od prvih PC virusa, „The Brain“ kreiran je 1986. godine u Pakistanu. Godine 1988. mladi programer Robert Moris ubacio je crva u ARPANET mrežu i uspeo da onesposobi oko 6000 računara. Crv je popunjavao memoriju napadnutih računara svojim kopijama. Moris, koji je bio uhvaćen i priznao da je crva kreirao iz dosade je kažnjen sa 10.000 \$ i još sa tri godine uslovne kazne.

Pojava virusa i crva izazvala je potražnju odbrambenih programa. Tako je Symantec 1991. objavio Norton Anti-Virus softver.

PRIMERI NAPADA

Jedan od prvih masovnih napada koji se dogodio 1999. korišćenjem elektronske pošte bio je virus Melissa, koji je zarazio na hiljade računara širom cele planete za kratko vreme

Jedan od većih napada sa ozbiljnim posledicama se dogodio 1998. godine kada su nepoznati napadači provalili u 500 vojnih, vladinih i industrijskih računara. Incident je nazvan „Solar Sunrise“ zbog ranjivosti računara koji su koristili Sun Solaris operativni sistem. Smatralo se da su napad izvršili operativci Iraka. Istraživači su kasnije otkrili da su napad izvršila dva tinejdžera. Ministarstvo odbrane SAD je dugo proučavalo ovaj slučaj kako bi saznali šta bi sve neprijateljski suparnici sa velikim iskustvom i resursima mogli da urade sa nacionalnim komandnim centrom, posebno u tandemu sa fizičkim napadom.

Jedan od prvih masovnih napada koji se dogodio 1999. korišćenjem elektronske pošte bio je virus Melissa, koji je zarazio na hiljade računara širom cele planete za neverovatno kratko vreme. Procenjena šteta koju je izazvao ovaj virus je oko 80 miliona dolara. S druge strane virus je izazvao ogromnu zabrinutost korisnika i rekordnu prodaju antivirusnih programa. Virus se širio kroz mrežu tako što bi sa računara na kome se našao poslao svoju kopiju na prvih 50 imena iz adresara programa Outlook. Takođe je inficirao Word dokumente i zatim ih slao preko Outlooka osobama čije su adrese bile u adresaru programa. Autor virusa David L. Smith je 2002. godine osuđen na 20 meseci zatvora.

Virus koji se najviše i najbrže proširio do sada je verovatno poznati „I Love You“ virus, koji je takoreći preko noći zarazio na milione računara. On je koristio iste principe kao i Melisa, ali je tvorcu virusa slao i korisničke lozinke koje je nalazio na zaraženim računarima. Istraga je otkrila da je autor virusa mladi Filipinac, ali on nije mogao da bude kažnjen jer zakoni Filipina ne prepoznavaju ovo krivično delo. Ovo je podstaklo Evropsku Uniju da pokrene globalni ugovor protiv sajberkriminala.

Veliki portali, među kojima i Yahoo, eBay, Amazon, Datek i mnogi drugi su 2000. godine neupotrebljivi nekoliko sati nakon serije napada poznatih pod imenom **distributed denial-of-service attacks**- DDOS. Napadnuti sistem je preplavljen saobraćajem koji dolazi od hiljade računara simultano. Napad je izveo haker koji je upao u računarsku mrežu Univerziteta Kalifornije u Santa Barbari i sa svih računara orkestirano vršio napad.

Svakog dana se javljaju novi virusi i crvi, ali i varijante starih virusa. Njihovo vreme rasprostiranja je sve kraće. Crv Slammer je obišao zemljinu kuglu za manje od 3 sata. Neki crvi, kao na primer Klez, su uspevali da onesposobe čak i antivirusne programe.

✓ 3.1 Napadi i pretnje

RAZLIKA IZMEĐU NAPADA I PRETNJI

Napadi se mogu javiti u različitim oblicima i sa različitim ciljevima.

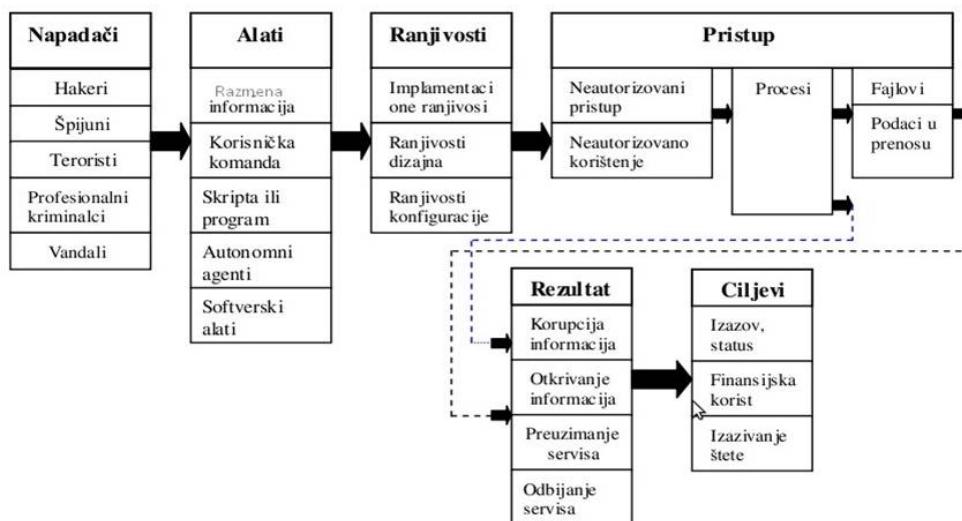
Bezbednost informacija je disciplina informatike koja se bavi izučavanjem metoda sprečavanja, detekcije i sanacije posledica, uglavnom elektronskog napada. Napadi se mogu javiti u različitim oblicima i sa različitim ciljevima. U literaturi, termini pretnja (engl. **threat**) i napad (engl. **attack**) često znače isto. Slede definicije ovih termina preuzete su iz RFC 2828, Internet Security Glossary.

- **Pretnja:** Potencijal za kršenje bezbednosti, koji postoji kada imamo okolnost, pogodnost, radnju ili događaj koji bi mogli izazvati kršenje bezbednosti i štetu; to jest, pretnja je moguća opasnost sa namerom da se iskoristi ranjivost sistema.
- **Napad:** Napad na sistem bezbednosti koji proizlazi iz inteligentnih pretnji; to jest, inteligentni akt sa namerom (posebno u smislu metode ili tehnike) izbegavanja bezbednosnih usluga i narušavanja bezbednosne politike sistema.

KLASIFIKACIJA NAPADA

Napadi se mogu podeliti na aktivne i pasivne.

Postoje različite vrste napada i mnogi načini njihove kategorizacije. Jedna od kategorizacija je prikazana na slici 1. Prema tome kako utiču na resurse sistema, **napadi se mogu podeliti na aktivne i pasivne**. **Aktivni napad** je pokušaj da se promene resursi sistema, ili da se utiče na njihov rad. **Pasivan napad** je pokušaj da se saznaju ili koriste informacije iz sistema, ali se ne utiče na resurse sistema.



Slika 3.1.1 Klasifikacija i komponente napada na računarski sistem [Izvor: Autor]

INTERNI I EKSTERNI NAPADI

U zavisnosti da li je subjekat napada unutar ili sa spoljne strane informacionog sistema, razlikujemo napade iznutra i napade spolja

U zavisnosti da li je subjekat napada unutar ili sa spoljne strane informacionog sistema, razlikujemo **napade iznutra i napade spolja**. **Napad iznutra** je napad pokrenut od strane

entiteta unutar bezbednosnog domena (nekog "zatvorenog kruga"), tj. jedna osoba koja je ovlašćena za pristup resursima sistema, ali njih koristi na način koji nije odobren od strane onih od koji je dobio ovlašćenje.

Napad spolja je pokrenut sa spoljnje strane domena, ili od strane neovlašćenog nelegitimnog korisnika sistema. Danas, potencijalni spoljni napadači na mrežu su u rasponu od amatera do organizovanih kriminalaca, međunarodnih terorista i neprijateljska vlada. Osoba koja napada računar na mreži često se naziva haker.

▼ 3.2 Bezbednosni mehanizmi

STADIJUMI KOJE PROLAZI ORGANIZACIJA PRILIKOM SAGLEDAVANJA BEZBEDNOSTI

Stadijumi: ignorantski stadijum, stadijum buđenja, stadijum spoznaje, stadijum odbrane, stadijum isleđivanja, stadijum praktične sigurnosti.

Cilj organizacije koja ima neki informacioni sistem je da on bude pouzdan i u funkciji poslovanja. Ukoliko informacioni sistem nije bezbedan, može da bude ugrožen, čime je istovremeno ugroženo i poslovanje organizacije. Zbog toga jedna organizacija treba da ima izgrađen bezbednosni način razmišljanja.

Razume se da jedna organizacija ne mora da ima izgrađen bezbednosni način razmišljanja. U tom slučaju će organizacija, ukoliko opstane proći kroz različite stadijume gledanja na bezbednost. Nadalje će biti opisane karakteristike ovih stadijuma:

- Ignorantski stadijum
- Stadijum buđenja
- Stadijum spoznaje
- Stadijum odbrane
- Stadijum isleđivanja
- Stadijum praktične sigurnosti.

KARAKTERISTIKE STADIJUMA

Kao posledica analize sigurnosti sistema većina zaposlenih postaje svesna ranjivosti sistema

Ignorantski stadijum. Ovaj stadijum je tipičan za organizaciju koja je tek uvela informacioni sistem i nije imala iskustva sa ugrožavanjem bezbednosti sistema. Zaposleni nisu svesni unutrašnjih i spoljašnjih opasnosti koje vrebaju njihov informacioni sistem. Niko ne zna koliko je sistem ranjiv i koje bi bile posledice narušavanja bezbednosti sistema. Postoji verovanje da je sistem siguran sam po sebi.

Stadijum buđenja. Prelazak iz ignorantskog u stadijum buđenja se događa nakon nekog događaja kojim je narušena bezbednost, a posledice su bile očigledne. Menadžment

organizacije postaje svestan opasnosti i pokreće akcije vezane za analizu sigurnosti sistema. Ljudi koji su zaduženi za sigurnost dobijaju podršku za svoj rad.

Stadijum spoznaje. Kao posledica analize sigurnosti sistema većina zaposlenih postaje svesna ranjivosti sistema. Otkrivaju se novi bezbednosni propusti i zahtevaju se nove analize koje pokazuju da je ranjivost sistema veća nego što se smatralo ranije. Menadžment donosi odluke koje treba da doprinesu bezbednosti ali shvata da realizacija odluka zahteva materijalna ulaganja. Ključni stručnjaci se šalju na dodatnu obuku kako bi se bolje planirala bezbednost.

Stadijum odbrane. Preduzimaju se potrebne mere da se sistem zaštitи. Postoje kvalifikovani ljudi koji planiraju i implementiraju mere sigurnosti. Uobičajeno se u ovoj fazi formira odeljenje za bezbednost informacionog sistema, koje će u budućnosti biti konsultovano pri implementiranju novih aplikacija. Sistem za otkrivanje upada (engl. **Intrusion Detection System**) je implementiran.

Stadijum isledivanja. U ovom stadijumu bezbednosni sistem je podignut na takav nivo da postoji jasna evidencija o tome ko je i šta radio unutar informacionog sistema. Na osnovu ove evidencije je moguće izvršiti istragu i nedvosmisленo dokazati ko je i kada ugrozio bezbednost sistema. Dokazi su do te mere pouzdani da ne mogu da se pobiju i mogu da posluže za definisanje optužbe. Pravnici imaju osnovu da deluju.

Stadijum praktične sigurnosti. U ovoj fazi u organizaciji postoji dovoljan broj iskusnih ljudi zaduženih za bezbednost koji saradjuju sa drugim IT odeljenjima i ostatom poslovnog sistema. Svi članovi organizacije su svesni potrebe da bezbednost ne sme da bude narušena, kao i činjenice da zahtevi bezbednosti ne treba da ugroze poslovanje organizacije. Dolazi se do praktičnih rešenja koja predstavljaju kompromis između bezbednosnih potreba i mogućnosti organizacije. Svi članovi organizacije prihvatili bezbednosni način razmišljanja.

▼ Poglavlje 4

Model za osiguranje informacija

PRETNJE

Jedna od definicija računarske bezbednosti je da je to nauka koja se bavi pretnjama i ranjivošću računarskih sistema.

Jedna od definicija računarske bezbednosti je da je to nauka koja se bavi pretnjama i ranjivošću računarskih sistema.

Današnji računarski sistemi su uglavnom vezani na Internet. Većina pretnji računarskim sistemima dolazi baš sa Interneta. Najčešće pretnje su:

- Virusi
- Trojanci
- Crvi
- Špijunski programi
- Uskraćeno opsluživanje

PRETNJE: VIRUSI, TROJANCI, CRVI

Računarski virusi su samoreplicirajući programi koji se šire umećući svoje kopije u druge programe ili dokumente

Virusi. Računarski virusi su samoreplicirajući programi koji se šire umećući svoje kopije u druge programe ili dokumente. Po svojoj prirodi mogu biti destruktivni ili dobroćudni. Neki se aktiviraju odmah, a neki u nekom vremenskom trenutku, pa se nazivaju vremenska bomba. Postoje i virusi koji se aktiviraju nekom akcijom u računarskom sistemu. Oni se nazivaju logičke bombe. Nekada su se virusi prenosili samo kopiranjem zaraženih datoteka, a danas se prenose i preko elektronske pošte i WWW, čime postaju slični crvima. Virusi mogu da se upgrade u aplikacije ili delove operativnog sistema, but sektore, skript datoteke i dokumente.

Trojanci. Trojanci su maliciozni programi maskirani u neku aplikaciju ili datoteku. Za razliku od virusa, nemaju osobinu samorepliciranja. Kada se lažno predstavljajući se useli u neki računarski sistem i aktivira ponaša se uglavnom destruktivno ili omogućuje neautorizovani pristup hostu od strane udaljenog korisnika. Uglavnom se predstavljaju kao fotografije u prilogu elektronske pošte, aplikacije, skrin sejveri, pa čak i kao antivirusni programi.

Crvi. Crv je destruktivni računarski program koji se kopira sa jednog računarskog sistema na drugi uobičajeno bez eksplicitne akcije korisnika. Širi se preko elektronske pošte, trenutnih poruka i kanala za časkanje. Kada jednom zaraze neki sistem, koriste podatke iz adresara

elektronske pošte i šalju svoje kopije drugim korisnicima. Neki crvi, koji se nazivaju mrežni, koriste siguronosne propuste u veb serverima i veb čitačima i inficiraju host sistem na potpuno nevidljivi način. Kada uđu u jednu lokalnu mrežu mogu da zaraže i ostale računare u toj mreži. Poznatiji crvi su CodeRed, Klez, LoveLetter, MyWife i Nimda.

Skript virusi. Ovi virusi su napisani u nekom od skript jezika kao što su Visual Basic Script ili JavaScript i ugnježdeni su u HTML strane. Prilikom čitanja zaražene veb strane u veb čitaču, virus počne da se izvršava.

PRETNJE: ŠPIJUNSKI PRORGAMI, USKRAĆENO OPSLUŽIVANJE

Napadi tipa uskraćeno opsluživanje zatravaju računar ili mrežu nepotrebnim podacima ili porukama, izazivaju preopterećenje sistema u cilju izazivanja degradacije

Špijunski programi. Špijunski programi (engl. **spyware**) snimaju celokupan rad korisnika, beleže ga u datoteku i šalju autoru programa. Ovakvi programi mogu da snime adrese odlazeće i dolazeće elektronske pošte, sadržaj pošte i poruka, adrese posećenih veb strana itd. Koristeći ove podatke autor špijunskog programa može da dođe do poverljivih podataka i do socijalne slike špijuniranih osoba. Postoje i špijunski programi koji se koriste za nadgledanje zaposlenih ili dece.

Uskraćeno opsluživanje. Napadi tipa uskraćeno opsluživanje (engl. **Denial of service- DoS**) zatravaju računar ili mrežu nepotrebnim podacima ili porukama, izazivaju preopterećenje sistema u cilju izazivanja degradacije karakteristika ili prestanka rada sistema. Poruke mogu da budu elektronska pošta ili poruke koje koristi neki Internet protokol. Da bi se poslao veliki broj poruka ciljanom računaru, koristi se više računara - zombija. Vlasnici ovih računara nisu ni svesni da se sa njihovih računara emituje napad.

RANJIVOST

Termin ranjivost se u računarskoj sigurnosti koristi da označi slabost sistema koji napadaču dozvoljava da povredi integritet sistema, tajnovitost, kontrolu pristupa, raspoloživost, konzistentnost

Termin **ranjivost** se u računarskoj sigurnosti koristi da označi slabost sistema koji napadaču dozvoljava da povredi integritet sistema, tajnovitost, kontrolu pristupa, raspoloživost, konzistentnost ili mehanizam nadgledanja. Ranjivost programa generalno nastaje zbog grešaka u softverskim sistemima. U nekim slučajevima su to slučajne greške u kodiranju, a u drugim, greške u projektovanju. Međutim, ranjivost ne nastaje samo zbog slabe kontrole kvaliteta koda, nego i zbog genijalnosti hakera koji uvek iznova nalaze nove načine za upade u sistem. Najčešći uzroci ranjivosti programskih sistema su:

- prepunjavanje bafera

- greška u rutinama za validaciju podataka
- greška u rutinama za rasčlanjivanje URL-a
- slabosti i propuste u sistemu lozinki
- pogrešna interpretacija standarda
- slabosti i propuste u modelu sigurnosti
- loša konfiguracija podrazumevajućih dozvola pristupa
- loše rukovanje izuzecima.

NAPADI

Napadi predstavljaju metode koje jedan računarski sistem koristi da bi povredio integritet drugog sistema

Postoje različiti metode koje jedan računarski sistem koristi da bi povredio integritet drugog sistema. Ove metode zajednički se nazivaju napadi. Postoje različite vrste napada. Ovde se navode samo neke najčešće:

- Napad brutalnom silom, je oblik programa koji koristi metod proba i greška da otkrije lozinke, ključeve šifriranja ili PIN-ove. Pri tom se karakteri menjaju po slučajnom zakonu
- Napad rečnikom, je po ciljevima sličan kao napad brutalnom silom, samo se za lozinke koriste reči iz nekog rečnika
- Uskraćeno opsluživanje
- Napad „čovek u sredini“ se vrši programom koji se postavlja između pošiljaoca informacija i primaoca, predstavljajući se pošiljaocu kao klijent, a primaocu kao server. Dok je „u sredini“ napadač preuzima podatke i zamenjuje ih lošim ili destruktivnim informacijama.
- TCP/IP otmica je vrsta napada kojom se preuzima kontrola nad komunikacionom sesijom tokom njenog trajanja.
- Njuškala (engl. sniffers) su programi koji mogu da preuzmu pakete koji putuju Internetom, kako bi se pronašli podaci o lozinkama ili PIN-ovima.
- Neželjena pošta (engl. spam) je oblik elektronskog reklamiranja korišćenjem elektronske pošte
- Napadi obmane su posebna vrsta napada kojom se napadač lažno predstavlja kako bi ukrao identitet napadnute osobe.

Kao što se vidi, brojne su pretnje po sigurnost računarskog sistema. Zbog toga je potrebno preduzeti odgovarajuće protivmere. Postoje opšte protivmere, ali se njima ne može odbraniti od svih pretnji. Zbog toga je pored njih potrebno preduzeti protivmere za svaku pretnju posebno.

“ZERO DAY” RANJIVOST I PRETNJA

Zero day pretnja je prednja koja eksplandiše do tog trenutka nepoznatu, bezbednosnu ranjivost

Pojam "**zero day**" se može odnositi na ranjivost i na pretnju. Zero day pretnja je prednja koja eksplatiše takvu, do tog trenutka nepoznatu, bezbednosnu ranjivost. Termin "**zero day**" potiče od toga da je na ovakvu ranjivost potrebno što pre reagovati, odnosno da programeri imaju "nula dana" da isprave ovaj propust, odnosno da u datom trenutni ni ne znaju da ovakva ranjivost ili napad postoje u njihovom sistemu. Zero day ranjivost se može iskoristiti kako bi se obavilo više vrsta napada. Napad može biti u formi virusa, crva, trojanca ili nekog drugog malvera. Zero-day pretnja se još naziva i zero-hour napad i day-zero napad. Zero-day napad eksplatiše ranjivost koja nije poznata ni programerima ni korisnicima. Kada napadači otkriju ranjivost, naprave crva ili virus koji će eksplatisati tu ranjivost i napraviti štetu. Koraci koje napadači prave kod zero-day napada obično uključuje sledeće faze:

- Traženje ranjivosti. Napadači proučavaju kod tražeći ranjivosti. U nekim slučajevima, informacije o zero-day ranjivosti hakeri mogu prodati ili kupiti.
- Ranjivost je pronađena. Napadači su pronašli "rupu" u bezbednosnom sistemu koja je nepoznata programerima konkretne aplikacije.
- Kreiranje koda za eksplataciju.
- Infiltracija. Napadači zaobilaze odbranu bez znanja programeri.
- Lansiranje napada. Naoružani kodom za eksplataciju, napadači ubacuju virus ili malver.

Tehnike detekcije uključuju sledeće:

- Tehnika zasnovana na statistici. Ovo je pristup otkrivanja napada u realnom vremenu koji se zasniva na profilima napada koji su se ranije događali, a koji su napravljeni na osnovu istorijskih podataka.
- Tehnika zasnovana na potpisu. Tehnika je zasnovana na "potpisima" koji su ostali od poznatih napada.
- Tehnika zasnovana na ponašanju. Ovaj model bazira se na analizi interakcije eksplatacije sa metom napada.
- Hibridna tehnika. Kao što joj i ime kaže, ova tehnika je mešavina različitih pristupa.

Zero day tržište je mesto gde se kupuju i prodaju informacije o zero-day ranjivostima i načinima za njihovu eksplataciju. Ovo tržište trenutno cveta. Pošto su zero-day ranjivosti i eksplatacije u suštini retka pojava, ovi kodovi su od izuzetne vrednosti ne samo za sajber-kriminalce, nego i za državne obaveštajne agencije.

Izvor: <http://www.it-klinika.rs/blog/sta-je-zero-day>

PRETNJE I NAPADI - VIDEO

CompTIA Network+ - Network Security (Threats and Attacks)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 5

Osnove kriptografije i kriptosistema

KRIPTOGRAFIJA I KRIPTOSISTEMI

Kriptografija je metod kojim se pomoću hardvera i softvera osetljive informacije šifriraju u nečitljivi oblik

Kriptografija je metod kojim se pomoću hardvera i softvera osetljive informacije šifriraju u nečitljivi oblik. Kriptografija se koristi kada je potrebno osetljive informacije preneti preko nebezbedne mreže, kao što je Internet, ili kada ih je potrebno čuvati na bezbedan način.

Većina kriptografskih sistema obezbeđuje način da se šifrirane informacije vrate u prvočitno, nešifrirano stanje, ali postoje i ireverzibilni načini šifriranja kada je to nemoguće.

Postoje različiti kriptografski sistemi, ali se u praksi uglavnom koriste varijacije jedne od dve metode za šifriranje:

- simetrični sistem ili sistem privatnog ključa, koji koristi samo jedan ključ
- asimetrični sistem ili sistem javnog ključa, koji koristi kombinaciju javnog i privatnog ključa

Izraz kriptografija potiče iz grčkih reči *kryptos* koji znači sakriven i *graphein* što znači pisati. Kriptografija se koristi hiljada godina, a u početku se koristila samo da obezbedi vojnu komunikaciju. Na primer, poznati rimski vojskovođa Julije Cezar je koristio kod da bi komunicirao sa svojim trupama. U okviru kriptografije se izdvajaju dve oblasti:

- kriptologija
- kriptoanaliza.

SIMETRIČNI SISTEM ŠIFRIRANJA

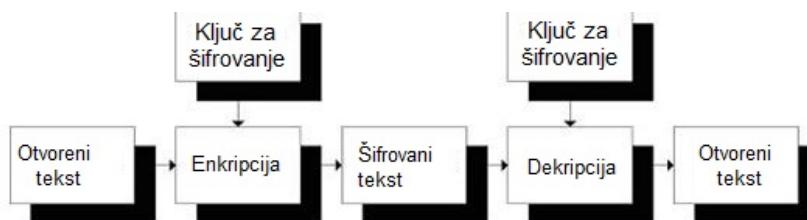
U simetričnom sistemu šifriranja pošiljalac i primalac koriste jedan ključ, pa se metod naziva simetričnim

Simetrični sistem šifriranja je originalno razvio IBM 1975. godine pod nazivom **Data Encryption Standard**(DES). Po ovoj metodi i pošiljalac i primalac koriste jedan ključ, pa se metod naziva simetričnim. Sam ključ se naziva privatni ključ.

Pošiljalac koristi primarni ključ da šifrira podatke na svom računaru pre slanja. Kroz mrežu se podaci prenose u šifriranom stanju. Primalac koristi isti ključ prilikom dešifriranja.

Na primer, IBM-ov DES deli informacije na pakete veličine 64 bita i šifrira ih jedan po jedan. Za šifriranje se koristi 56-bitni ključ. Za normalne komercijalne potrebe DES je dovoljno siguran metod, mada se primenom napada brutalne sile DES može razbiti. Da bi se ovakav sistem učinio dovoljno bezbednim, potrebna je česta promena **simetričnog ključa**.

Jedan od problema koji se javlja prilikom korišćenja simetričnog ključa je da obe strane treba da ga imaju. Slanje informacija o ključu između pošiljaoca i primalaca, takođe mora biti bezbedno



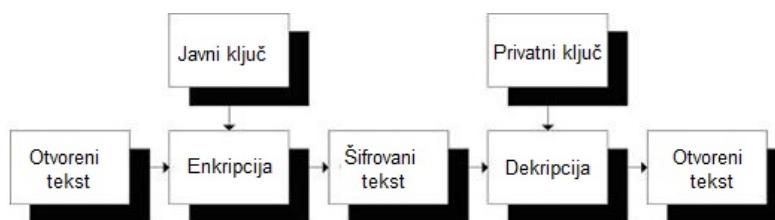
Slika 5.1 Simetrični sistem šifriranja [Izvor: Autor]

ASIMETRIČNI SISTEM ŠIFRIRANJA

Asimetrični sistem šifriranja koristi javni ključ prilikom šifriranja i privatni prilikom dešifriranja

Asimetrični metod kriptovanja je projektovan baš u cilju rešavanja problema slanja simetričnog ključa. Whitfield Diffie i Martin Hellman su 1976. projektovali sistem koji koristi dva, asimetrična, ključa: javni i privatni ključ. **Javni ključ se koristi prilikom šifriranja, a privatni ključ prilikom dešifriranja.**

Organizacija koja koristi asimetrični način šifriranja ima jedan privatni ključ koga čuva za sebe i distribuira odgovarajući javni ključ svojim partnerima. Svi partneri koriste isti javni ključ za šifrovanje informacija koje šalju organizaciji. Informacije se mogu dešifrirati samo posedovanjem privatnog ključa. Poznavanjem samo jednog ključa, informacije se ne mogu dešifrovati. na taj način se broj ključeva u opticaju smanjuje, a sigurnost povećava.



Asimetrična enkripcija

Slika 5.2 Asimetrični sistem šifriranja [Izvor: Autor]

DEŠIFROVANJE

Metode dešifrovanja delimo na reverzibilne i ireverzibilne

Sa aspekta mogućnosti dešifrovanja, odnosno vraćanja informacija u oblik pre šifriranja postoje:

- reverzibilne
- ireverzibilne metode.

Kod **reverzibilnih metoda** postoji način da se informacije povrate u stanje pre šifriranja. Nijedan specijalista za kriptografiju neće tvrditi da su podaci šifrirani reverzibilnom metodom apsolutno sigurni, jer se svako reverzibilno šifriranje pre ili kasnije može razbiti. Zadatak kriptografije je da proces dešifriranja učini toliko dugotrajnim da se dešifriranje jednostavno ne isplati.

Kod **ireverzibilnih metoda** ne postoji način da se šifrirane informacije vrate u stanje pre šifriranja. Ova metoda se, na primer, koristi za čuvanje lozinki u računarskom sistemu. Za šifriranje se koriste haš tehnike. Kada se korisnik ponovo loguje na sistem i unese lozinku ona se istim algoritmom šifrira, a zatim se tako šifrirana upoređuje sa ranije šifriranom lozinkom koja se čuva u sistemu. Ovo čini sistem teoretski neprobojnim za dešifriranje.

AUTENTIKACIJA

Autentikacija je proces provere identiteta osobe koja se prijavljuje računarskom sistemu.

Autentikacija je proces provere identiteta osobe koja se prijavljuje računarskom sistemu. Pri tom osoba može da koristi različite podatke da se autentikuje. To može biti neki od sledećih autentikacionih faktora:

- nešto što osoba zna
- nešto što osoba ima
- nešto što je osoba
- kombinacije prethodne tri stvari.

Što se više autentikacionih faktora koristi, autentikacija je sigurnija.

Za autentikaciju se koriste različite šeme. Svaka šema koristi neke određene autentikacione faktore. Nadalje će biti prikazane neke šeme autentikacije:

- Kolačići
- Identifikator i lozinka
- Sistem lozinka-odziv
- Token
- Digitalni sertifikati
- Biometrijski uređaji
- Pametne kartice.

KOLAČIĆI, IDENTIFIKATOR I LOZINKA, SISTEM LOZINKA ODZIV

Kolačić sadrži informacije koje je definisao server i koje mogu, ali ne moraju da sadrže lične informacije o korisniku

Kolačići. Kolačići (engl. **cookies**) predstavljaju najčešći oblik provere identiteta na Internetu. Uglavnom se koriste za pristup nekim veb sajtovima ili portalima na serverima. Prilikom prvog prijavljivanja na sajt, udaljeni korisnik dobija kolačić koga čuva na svom računaru. Kolačić sadrži informacije koje je definisao server i koje mogu, ali ne moraju da sadrže lične informacije o korisniku. On ne može da se izvršava ili da pristupa računarskim resursima klijenta. Prilikom narednog pristupa istom serveru kolačić se šalje serveru radi autentikacije.

Identifikator i lozinka. Kombinacija identifikatora i lozinke je vrlo česta autentikaciona šema koja se koristi za pristup računarskim sistemima. Kao identifikator se obično koristi neka kartica koja može da memoriše podatke. Korišćenjem identifikatora osoba zahteva identitet, a lozinka se koristi da joj se taj identitet odobri na sistemu. Na žalost, identifikatori su podložni neovlašćenom kopiranju.

Sistem lozinka-odziv. Ovaj sistem je povezan sa sistemom lozinki. U ovom slučaju server generiše slučajni string karaktera - lozinku, a od korisnika se zahteva da prema unapred definisanom tajnom algoritmu, koga imaju obe strane, obradi string i vrati ga serveru kao odziv. Algoritam može da uključi i primenu tajnog ključa. Ako se odziv korisnika poklapa sa očekivanjem servera, smatra se da je autentikacija uspešna. Ovaj sistem ima prednost u odnosu na sistem identifikator i lozinka, jer algoritam može da bude dovoljno kompleksan da ga je teško probiti. Nedostatak je što se algoritam nalazi i na korisnikovom računaru, pa ga je moguće ukrasti.

TOKEN, DIGITALNI SERTIFIKATI

Token i server koji vrši autentikaciju sinhronizuju vreme tako da server može da zna koje brojeve generiše token u nekom vremenskom intervalu

Token. Jedna od varijacija sistema lozinka-odziv je primena specijalnih računarskih uređaja koji se nazivaju tokeni. Token može da generiše seriju pseudoslučajnih brojeva koji zavise od vremena i menjaju se recimo svakih 60 sekundi. Token i server koji vrši autentikaciju sinhronizuju vreme tako da server može da zna koje brojeve generiše token u nekom vremenskom intervalu. Ovi broevi mogu da se iskoriste na različite načine. Na primer, u sistemu lozinka-odziv generisani broevi mogu na neki način da se upotrebe za manipulaciju stringom koji se dobija od servera. To je mnogo sigurnije nego unapred definisan algoritam.

Digitalni sertifikati. Digitalni sertifikat objedinjuje informacije o identitetu i javni ključ koji je potpisano od treće strane. Najjednostavniji način primene digitalnog sertifikata za autentikaciju je u sistemu lozinka-odziv. Pri tom se umesto tajnog algoritma koristi algoritam digitalnog potpisa. Server generiše slučajni string i šalje ga korisniku. Korisnik digitalno potpisuje slučajni string i vraća ga serveru. Server može da proveri da je vraćeni string tačan i da je potpisano od strane vlasnika sertifikata.

BIOMETRIJSKI UREĐAJI, PAMETNE KARTICE

Prednost biometrijskih metoda autentikacije je u tome što se autentikuje osoba, a ne neko ko zna korisničko ime i lozinku

Biometrijski uređaji. Biometrijski uređaji se koriste za snimanje neke biometrijske osobine osoba. Jednom snimljene i sačuvane, ove osobine se mogu upoređivati sa snimkom iste osobine prilikom autentikacije. Na taj način sistem može da prepozna osobu koju je prethodno upisao. Biometrijsko snimanje se kombinuje sa nekim drugim autentifikacionim faktorom kao što je identifikator ili token. Pošto se zna kako se osoba predstavila, potrebno je samo uporediti izmerene i memorisane biometrijske osobine kako bi se osoba autentifikovala. Prednost biometrijskih metoda autentikacije je u tome što se autentikuje osoba, a ne neko ko zna korisničko ime i lozinku. Nedostatak biometrijskog metoda je u tome što neke od njih mogu biti kopirane ili ukradene.

Pametne kartice. Pametne kartice imaju ugrađeni čip i jednostavni interfejs za pristup memoriji ili procesoru na čipu. Većina kartica ima i set kontakata, preko kojih se, kada je kartica u čitaču, obezbeđuje napajanje i prenos podataka. Pametne kartice se mogu upotrebiti na različite načine. Jedna od mogućnosti je da se memorija na kartici koristi za čuvanje digitalnog sertifikata. Procesor se može upotrebiti za zadatke šifriranja i dešifriranja i za verifikaciju autorizovanog pristupa privatnom ključu. Na taj način nema potrebe da se ključ prenosi van kartice, pa se smanjuju šanse da on bude otkriven. Pametne kartice takođe mogu biti upotrebljene i u scenariju lozinka-odziv. U tom slučaju algoritmi za manipulaciju lozinkom su memorisani na kartici. Neke pametne kartice imaju ugrađen i skener otiska prsta. Najčešća primena pametnih kartica je u obliku SIM modula koji se umeće u mobilne telefone. Ova kartica sadrži identitet vlasnika telefonskog broja.

ISTORIJA KRIPTOGRAFIJE - VIDEO

Cryptography: Crash Course Computer Science #33

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 6

Pokazne vežbe: SQL injection napad

SQL INJECTION

SQL injection je napad u kome maliciozni korisnik izvršava SQL kod

Očekivano vreme izrade zadatka: 25 minuta.

U aplikacijama sa bazom podataka, često je potrebno konstruisati upit sa koristeći podatke koje je korisnik zadao kao ulaz. Npr. aplikacija će često imati polje za pretragu u koje korisnik može da upiše svoj upit.

Uzmimo za primer aplikaciju koja dozvoljava korisniku da pretražuje bazu studenata po imenu. Upit koji će se izvršiti u bazi će biti sličan sledećem, gde će se znak ? zameniti tekstrom iz ulaza:

SELECT * FROM Student WHERE Student.ime = ?

Naivni način da se u PHP jeziku konstruiše ovakav upit bi bio sledeći:

```
$ime=$_POST['ime'];
$con=odbc_connect('lista','','');
$sql="SELECT * FROM Student WHERE Student.Ime= '$ime'";
```

Na izgled, ovaj kod radi bez problema, ali je izuzetno ranjiv na napade. Maliciozni korisnik bi mogao da u polje za pretragu upiše sledeći kod:

'a ; DROP DATABASE Studenti;

Rezultujući SQL upit bi bio:

SELECT * FROM Studenti WHERE Student.Ime='a';DROP DATABASE Studenti;

Šta se tačno ovde desilo? Napadač je koristio single quote znak da zatvori navodnike koji označavaju string u where delu upita. To znači da se ostatak teksta izvršava kao SQL kod. Dobijamo sledeće upite:

SELECT * FROM Studenti WHERE Student.Ime='a';

DROP DATABASE Studenti;

PREVENCIJA SQL INJECTION NAPADA

Svi podaci koji dolaze od krajnjeg korisnika su potencijalno maliciozni.

Prvi upit verovatno neće selektovati ništa, dok će drugi upit izbrisati kompletну bazu studenata. Ovakav propust u aplikaciji bi imao katastrofalne posledice.

SQL Injection napadi dolaze u mnogo oblika, i obično nisu očigledni kao u ovom primeru. Parametri za upit mogu doći iz forme, URL parametara kroz cookie, itd. Svi podaci koji dolaze od krajnjeg korisnika su potencijalno maliciozni.

Rešenje je da se nedozvoljeni karakteri filtriraju iz ulaza. Postoje funkcije koje rade upravo ovo kao npr. `mysql_real_escape_string()`, ali ovo ipak stavlja preveliko opterećenje na programera.

Najbolje rešenje je korišćenje tzv. pripremljenih izraza (engl. [Prepared Statement](#)).

▼ 6.1 Pokazne vežbe: PHP Cookies i sesije

ŠTA JE PHP COOKIE?

Cookie je tekstualni fajl koji cuva neke podatke o korisniku ili trenutnoj sesiji

Očekivano vreme izrade zadatka: 35 minuta.

Cookie je tekstualni fajl (uglavnom oko 4KB) koji se čuva na klijentskom kompjuteru i čuva neke podatke.

Uglavnom se koristi za praćenje sesija, ili čuvanje informacija o korisniku kao što su njegovo korisničko ime koje posećeni web sajt može koristiti za personalizaciju stranice ili brže učitvanje sadržaja svakom narednom posetom.

Najčešće se koriste u sledećim slučajevima:

- Pamti username i password tako da korisnik ne mora da se loguje svaki put kad poseti sajt (remember me opcija)
- Za pamćenje korisničkog imena
- Da prati progress korisnika ukoliko postoji potreba
- Pamćenje teme korisnika

Kolačići mogu da se kreiraju koristeći **setcookie()** funkciju. Sintaksa je sledeća:

setcookie(name, value, expire, path, domain, secure, httponly);

name - obavezan atribut. Definiše naziv cookie-ja;

value - opcioni atribut. Definiše vrednost cookie-ja.

expire - opcioni atribut. Definiše kada će cookie da istekne. Za definisanje ovog atributa se uglavnom koristi funkcija **time()**. Ukoliko se kao vrednost definiše 0 ili se izostavi vrednost, cookie će automatski da traje do kraja sesije, odnosno dok korisnik ne zatvori browser.

path - opcioni atribut. Označava putanju na serveru na kome se nalazi cookie. Ukoliko se vrednost definiše kao '/', cookie će biti dostupan na celom domenu. Ako se definiše neka putanja , npr. 'php', biće dostupan samo u php direktorijumu i svim pod-direktorijumima.

domain - opcioni atribut. Označava u kom domenu će cookie biti dostupan. Ukoliko se kao vrednost definiše pod-domén www.example.com, biće dostupan samo u tom pod-domenu i svim njegovim pod-doménima. Ako se vrednost definiše kao example.com, onda će cookie biti dostupan u čitavom domenu.

secure - opcioni atribut. Određuje da li će se cookie prenositi samo putem sigurne HTTPS konekcije. Ukoliko je podešeno na 'true' cookie će se kreirati samo ukoliko postoji sigurna konekcija.

httponly - opcioni atribut. Ukoliko je podešeno na 'true', cookie-ju se može pristupiti samo preko HTTP protokola.

PHP COOKIE PRIMER

Definisanje cookie-ja

Očekivano vreme izrade zadatka: 25 minuta.

Korišćenje funkcije za postavljanje vrednosti:

```
<!DOCTYPE html>
<?php
    $cookie_name = "user";
    $cookie_value = "John";
    setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?

<body>
</body>
</html>
```

Pomoću **isset()** funkcije se proverava da li je uspešno setovano.

```
<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie sa nazivom '" . $cookie_name . "' nije podešen.";
} else {
    echo "Cookie sa nazivom '" . $cookie_name . "' je uspešno podešen.<br/>";
    echo "Vrednost je: " . $_COOKIE[$cookie_name];
}
?>
```

Za izmenu podataka ne postoji posebna funkcija, već se cookie ponovo setuje sa izmenjenim podacima.

```
<?php
$cookie_name = "user";
$cookie_value = "Jane";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
```

Brisanje je najsigurnije uraditi podešavanjem isteka za vreme koje je prošlo.

```
<?php
setcookie("user", "", time() - 3600); //pre jednog sata
?>
```

U ovom primeru, ponovo je setovan cookie, ali je podešeno da ističe presat vremena, što znači da će biti obrisan.

PHP SESIJE

Kod sesije cookie sadrži samo automatski generisan identifikator sesije, dok se ostatak podataka vezanih za identifikator čuva na serveru, u memoriji, ili u nekoj brzoj bazi podataka

Glavni nedostatak prethodnog pristupa je što se cookie šalje prilikom svakog zahteva, što je pogodno za samo malu količinu podataka. Svaki put kada pretraživač zatraži URL od servera, podaci svih kolačića za sajt se automatski šalju server u sklopu zahteva. Ukoliko je na korisničkom računaru sačuvano 5 kolačića od kojih svaki zauzima po 4KB memorije, pretraživač onda mora da učita 20KB podataka svaki put kada korisnik otvorí stranicu. Što može da utiče na performance sajta.

Takođe, cookie se nalazi na klijentskoj strani, što znači da ga maliciozni korisnik može lako promeniti.

Rešenje za ove probleme su sesije. Kod sesije cookie sadrži samo automatski generisan identifikator sesije, dok se ostatak podataka vezanih za identifikator čuva na serveru, u memoriji, ili u nekoj brzoj bazi podataka.

Sesije su glavni mehanizam za čuvanje podataka trenutno ulogovanog korisnika, pošto će svaki korisnik imati svoju posebnu sesiju.

Sesija se započinje jednostavnom funkcijom **session_start()** koja se uglavnom stavlja na početak dokumenta. Ova funkcija prvo proverava da li postoji neka otvorena sesija pa ukoliko nije započinje novu.

Varijable sesije se podešavaju globalnom PHP varijablom **\$_SESSION** i čuvaju se u nizu **\$_SESSION[]**.

PHP SESIJE PRIMERI

Kreiranje, pristupanje i brisanje sesije

Očekivano vreme izrade zadatka: 25 minuta.

Sesija se kreira na sledeći način:

```
<?php
    session_start();
?>
```

Kada je sesija kreirana, neophodno je podesiti podatke. Podesićemo podatke o korisniku odnosno ime i prezime korisnika na sledeći način:

```
<?php
    session_start();

    $_SESSION["ime"] = "Marko";
    $_SESSION["prezime"] = "Markovic";
?>
```

Da bismo pristupili podacima sesije, otvorićemo sesiju i zatim pristupiti nizu u kome su sačuvani podaci:

```
<?php
    session_start();

    echo 'Cao, ' . $_SESSION["ime"] . ' ' . $_SESSION["prezime"];
?>
```

Brisanje podataka sesije se vrši funkcijom **unset()**:

```
<?php
    session_start();

    if(isset($_SESSION["prezime"])){
        unset($_SESSION["prezime"]);
    }
?>
```

Međutim, ukoliko želimo da potpuno obrišemo sesiju, koristi se funkcija **session_destroy()**.

KAKO ZAŠTITITI OD SQL INJECTION NAPADA - VIDEO

*Protect your database against SQL injection using MySQLi | PHP tutorial
| Learn PHP programming*

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 7

Zadaci za dodatnu vežbu

OPIS ZADATAKA ZA SAMOSTALNU VEŽBU

Kreirati HTML stranu sa formom za unos podataka

Očekivano vreme izrade zadatka: 45 minuta.

Prethodna vežba za samostalni rad glasila je:

Kreirati HTML stranu sa formom za unos podataka o igračkama koje se prodaju u prodavnici.

Svaka igračka ima sledeće podatke: ID, ime, ime proizvođača, datum proizvodnje, cenu.

Svaka uneta igračka mora se upisati u bazi koja je unapred kreirana.

Na novoj strani kreirati tabelu u kojoj će se ispisati igračke koje su upisane u bazi.

Igračke čija je cena veća od 2000 dinara biće ispisane crvenom bojom, ostale zelenom bojom.

Dopunite ove zahteve sa sledećim i doradite vašu stranu:

- Kreirati formu za registraciju korisnika i logovanje korisnika.
- Ukoliko je korisnik administrator može uneti novu igračku na sajt.
- Ukoliko nije može samo kupiti igračku.
- Takođe administrator može da obriše igračke.

✓ Poglavlje 8

DOMAĆI ZADATAK

DOMAĆI ZADATAK- DZ12

Opis domaćeg zadatka

Očekivano vreme izrade zadatka: 45 minuta.

- Kreirati formu za registraciju korisnika.
- Kreirati formu za login korisnika.
- Ukoliko se korisnik uspešno uloguje, otvara se nova stranica (welcome.php).
- Ukoliko je logovanje neuspešno, korisnik se vraća na stranicu za login uz poruku o neuspešnom logovanju.
- Ukoliko korisnik pokuša da pristupi stranici welcome.php bez logovanja, pristup mu ne sme biti dozvoljen.
- Kada je logovanje uspešno, na stranici welcome.php prikazati korisnikovo ime.
- Forme moraju biti otporne na SQL injection.

Potrebne datoteke snimiti pod imenom:

IT210-DZ12-Ime_Prezime_brojIndexa, gde su Ime, Prezime i broj indeksa vaši podaci.
Tako imenovane datoteke poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210 - DZ12)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovom predavanju je opisana ulogu mreže u aplikacijama sa bazama podataka i file serverima. Opisano je šta bi se dogodilo sa WWW delom Interneta ako bi većina rutera prestala da radi. Takođe, dat je pregled istorije osiguranja informacija i bezbednosti, kao i osnovne karakteristike autentifikacije i osnovnih bezbednosnih mehanizama kao što su kriptografija i kriptosistemi.

Literatura

- [1] Andrew S. Tanenbaum, Computer Networks, Fourth Edition, Prentice Hall.
- [2] Werner Feibel, Enciklopedija računarskih mreža, Mikro kniga.
- [3] James F. Kurose, Keith W. Ross, Computer networking, Addison Wesly.



IT210 - SISTEMI IT

OSIGURANJE INFORMACIJA I BEZBEDNOST

Lekcija 13

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 13

OSIGURANJE INFORMACIJA I BEZBEDNOST

- ✓ OSIGURANJE INFORMACIJA I BEZBEDNOST
- ✓ Poglavlje 1: NAPADI
- ✓ Poglavlje 2: VRSTE NAPADA
- ✓ Poglavlje 3: BEZBEDONOSNI DOMENI
- ✓ Poglavlje 4: FORENZIKA
- ✓ Poglavlje 5: PRAVNI SISTEM
- ✓ Poglavlje 6: Pokazna vežba: Enkripcija i dekripcija teksta
- ✓ Poglavlje 7: Pokazne vežbe: Šifrovanje koristeći Cryptool
- ✓ Poglavlje 8: Zadaci za samostalni rad
- ✓ Poglavlje 9: Domaći zadatak
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije da se obrade bezbednosni domeni iz ugla računarstva i pravnog sistema

U ovom predavanju biće reči o:

- Vrstama napada
- Bezbednosnim domenima
- Pravnom sistemu koji se primenjuje na računarstvo
- Digitalnoj istrazi i njenom odnosu sa drugim istragama

▼ Poglavlje 1

NAPADI

MOTIVI NAPADAČA

Pod napadima se podrazumeva zlonamerna akcija koju jedan proces vrši na drugom računarskom sistemu bez dozvole vlasnika sistema

Pod **napadima** se podrazumeva zlonamerna akcija koju jedan proces vrši na drugom računarskom sistemu bez dozvole vlasnika sistema. Napadi se uglavnom izvode korišćenjem Interneta. Različiti su motivi napadača. Tipični motivi napada su:

- **Zabava.** Tipičan motiv napada kod mladih ljudi.
- **Izazov.** Upad u dobro branjene sisteme je jak izazov za osobe koje žele da dokažu sebi ili svojoj okolini da su sposobne da upadnu u bilo koji sistem. Tipični ciljevi su serveri ili mreže vladinih ili vojnih institucija.
- **Osveta.** Mnogi ljudi su nezadovoljni svojim statusom u organizaciji. Oni poznaju rad informacionog sistema i nije im teško da organizuju napad. Posebno su opasni kada budu otpušteni.
- **Industrijska ili privredna špijunaža.** Mnoge firme organizuju upade u konkurentske sisteme kako bi došle do poverljivih dokumenata, otkrile ko su klijenti konkurenčije i došle do drugih informacija koje imaju prirodu intelektualne svojine.
- **Terorizam.** Odbrana mnogih zemalja je zasnovana na različitim informacionim tehnologijama. Sajber teroristi organizuju napade na informacione sisteme koji su u funkciji odbrane kako bi onesposobili ili degradirali funkcije odbrane.
- **Krađa.** Vrlo jak i čest motiv za napad je krađa. Krađa se izvodi u dva koraka. U prvom koraku se kradu poverljive informacije sa računara, kao što su korisnička imena, lozinke, brojevi kreditnih kartica, brojevi naloga kod ISP provajdera i slično. U drugom koraku se ovi podaci koriste za pribavljanje određene materijalne dobiti.

TENDENCIJE NAPADA I OPASNOSTI

"zero-day" ranjivost predstavlja propuste u softveru firme za koje firma nije svesna.

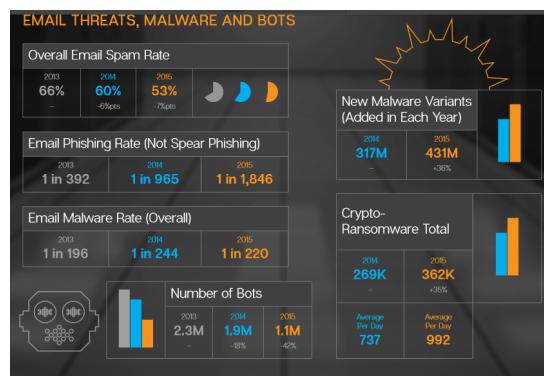
Symantec je otkrio više od 430 miliona novih jedinstvenih malvera u 2015-oj godini, što je oko 36% više od prethodne godine. Ono što je možda zadržalo najviše jeste da nas ovakva statistika možda više i ne iznenađuje. U 2015-oj broj "zero-day" ranjivost je porasla za 125% od prethodne godine. "zero-day" ranjivost predstavlja propuste u softveru firme za koje firma nije svesna. Drugim rečima, ova ranjivost je u proseku nađena svake nedelje u 2015-oj godini.

Pri kraju 2015-te godine, svet je iskusio najveću krađu ličnih zapisa koji su prijavljeni javno. Neverovatnih 191 milion zapisa je otkriveno. U 2015-oj godini je otkriveno i najveće mega-probijanje (engl. **mega breach**). Jedno mega probijanje predstavlja probijanje 10 miliona zapisa.

Ukupan broj ukradenih identiteta je zabeležen na 429 miliona, što je porast od 23%. Međutim, ovaj broj krije veću priču iza sebe. U 2015., veliki broj kompanija je odlučio da ne objavljuje javno u potpunosti nivo probijanja koje su imale. Symantec procenjuje da je ovaj broj ipak veći od pola milijarde.



Slika 1.1 Symantec izveštaj o probijanjima, Izvor: Internet Security Threat Report, Symantec [Izvor: Semantec]



Slika 1.2 Symantec izeštaj o pretnjama, malveru i botovima, Izvor: Internet Security Threat Report, Symantec [Izvor: Semantec]

PET PET NAJRAZORNIJIH “CYBER” NAPADA

5 Most Devastating Cyber Attacks | Cybersecurity Insights #18

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 2

VRSTE NAPADA

NAPAD BRUTALNOM SILOM

Napad brutalnom silom , je oblik programa koji koristi metod proba i greška

Napad brutalnom silom je oblik programa koji koristi metod proba i greška da otkrije lozinke, ključeve šifriranja ili PIN-ove. Cilj napada je da se izvrši lažna autentikacija i upadne u sistem. Pri tom se karakteri menjaju po slučajnom zakonu. Ova metoda je pouzdana jer je verovatno da će se posle dovoljnog broja pokušaja pogoditi očekivani string za autentikaciju. Međutim broj pokušaja može da bude izuzetno veliki pa se očekuje da će napadač odustati.

Varijanta ovog napada je napad rečnikom. Koristi se isključivo za otkrivanje lozinke. U ovo slučaju se, umesto slučajnog generisanja karaktera, koriste reči iz nekog rečnika. Metod nije tako uspešan samo ako je lozinka sastavljena od pravih reči. Ako sadrži i brojeve ili stringove bez značenja, ovaj metod je neuspešan.

DOS NAPAD

Napad uskraćivanja opsluživanja ima za cilj da zatrpa računarski sistem ili mrežu nepotrebnim porukama ili podacima kako bi preopteretio sistem

Jedna od najčešćih vrsta sajber pretnji je **Denial of Service (DoS)**, koja kako samo ime kaže, čini sajtove i ostale mrežne resurse i servise koji su namenjeni korisnicima nedostupnim. Postoji mnogo različitih formi DoS pretnji, od kojih su neki direktno usmereni na osnovnu infrastrukturu servera. Drugi koriste ranjivost u aplikaciji i komunikacionim protokolima.

Za razliku od drugih vrsta napada i pretnji, koji se obično uspostavljaju radi dugoročnih ometanja i prikupljanja osetljivih informacija, Denial of Service napad ne pokušava da probije bezbednosni domen. Umesto toga, ovaj napad pokušava da sajt i servere učini nedostupnim legitimnim korisnicima. U nekim slučajevima, DOS se koristi kao paravan za druge zlonamerne aktivnosti kako bi poremetio bezbednosne alate i metode kao što je firewall.

Uspešan DoS napad je veoma primetan, pošto utiče na čitavu bazu korisnika, napadnutog servisa.

DoS napadi često traju danima, nedeljama ili čak mesecima, čineći ih veoma destruktivnim na bilo koje online aktivnosti organizacija. Oni mogu dovesti do gubitka prihoda, narušavaju poverenje potrošača, da dovedu do toga da preduzeća moraju da isplate velike odštete

i uzrokovati dugoročnu lošu reputaciju. Napad uskraćivanja opsluživanja (engl. denial of service- DoS) ima za cilj da zatrpa računarski sistem ili mrežu nepotrebnim porukama ili podacima kako bi preopteretio sistem i redukovao ili potpuno uskratio opsluživanje legalnih korisnika. Napad može da se vrši sa jednog ili istovremeno sa više računara koji se nazivaju zombiji. Ukoliko se za napad koristi više zombi računara radi se o distribuiranom napadu uskraćivanja opsluživanja (DDoS). Cilj napada može biti bilo koji mrežni uređaj uključujući usmerivače, veb servere, mejl servere i DNS servere.

VRSTE DOS NAPADA

Postoji više metoda kojim se vrši DoS napad: stalnog pingovajne, ping smrti, slanjem podataka, SYN poplava

Postoji više metoda kojim se vrši DoS napad, kao što su:

- Stalno pingovanje. Pingovanjem se proverava veza između dva sistema koja rade sa TCP/IP protokolom, tako što se odredišnom klijentu šalje Internet Control Message Protocol (ICMP) Echo Request. Odredišni klijent vraća pošiljaocu poruku Echo Reply zajedno sa vremenom koje je bilo potrebno da se poruka vrati do pošiljaoca. Odredišni klijent mora da odvoji neko vreme da odgovori na ping. Ako se ovom klijentu pošalje veliki broj pingova njegova sposobnost da opslužuje druge klijente će biti degradirana.
- Ping smrti. Ova metoda napada šalje žrtvi Internet Control Message Protocol pakete koji su loše oblikovani ili fragmentirani, a nekada mogu da aktiviraju poznati bag u operativnom sistemu i obore računar žrtve. Za slanje paketa se koristi ping komanda sa nekim modifikatorom.
- Slanjem elektronske pošte. U ovom slučaju se klijentu žrtvi šalje ogromna količina elektronske pošte. Kao rezultat toga mreža klijenta postaje zagušena, a mejl server se pri ili kasnije prepuni. Uobičajeno se pošta šalje sa više zombi računara na mreži.
- Slanjem podataka. U ovom slučaju se žrtvi šalju velike količine podataka kako bi se premašio kapacitet njene mreže i blokirao njen rad.
- SYN poplava. Napadač šalje ogromnu količinu TCP/SYN paketa često sa lažnom adresom pošiljaoca. Svaki paket se kod žrtve tretira kao zahtev za konekcijom, što izaziva stvaranje poluotvorenih konekcija. Žrtva šalje TCP/SYN-ACK paket i očekuje TCP/ACK paket kao odgovor, ali pošto je adresa bila lažna odgovor nikad ne stiže. Poluotvorene sesije popunjavaju raspoloživi broj konekcija koje server može da ostvari, tako da ne može da odgovori na legitimne zahteve.

DOS NAPADI - VIDEO

Denial of Service ATTACK | DOS ATTACK Mitigation Steps | Types of DOS Attack | DOS Attack Examples

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

TCP/IP OTMICA

Napadač presretne trenutnu konekciju ka ciljanom računaru i korišćenjem odgovarajućeg programa pogađa naredni broj u nizu

TCP/IP otmica je vrsta napada kojom se preuzima kontrola nad komunikacionom sesijom tokom njenog trajanja. Ova vrsta napada se zasniva na činjenici da su konekcije preko Interneta označene brojevima u slučajnom rastućem redosledu. Napadač presretne trenutnu konekciju ka ciljanom računaru i korišćenjem odgovarajućeg programa pogađa naredni broj u nizu. Ako napadač pogodi broj, stvara se nova konekcija kojom napadač šalje žrtvi podatke ili instrukcije. Specijalni slučaj ovog napada je metod „čovek u sredini“. Pošiljalac ima utisak da primalac prima pakete, a u stvari ih prima napadač. On može da promeni pakete i da ih dalje prosleđuje ka primaocu koji veruje da ih dobija od servera sa kojim je započeo komunikaciju.

Posebnu vrstu krađe paketa koji putuju Internetom vrše **njuškala** (engl. **snifers**). Svrha ovog presretanja je samo da se prikupe podaci o lozinkama i PIN-ovima koji se nalaze u nešifrovanim paketima.

TCP/IP OTMICA - VIDEO

TCP/IP Hijacking

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

SOCIJALNI INŽENJERING

Prevara se zasniva na naivnosti korisnika, koji veruje sagovorniku i zaboravlja na bezbednosne principe

Socijalni inženjering je metod dobijanja poverljivih informacija obmanom legitimnih korisnika. Napadači se uglavnom koriste elektronskom poštom ili telefonom. Predstavljajući se lažno oni od korisnika zahtevaju da izvrši neku radnji koja im omogućuje da dođu do poverljivih informacija. Prevara se zasniva na naivnosti korisnika, koji veruje sagovorniku i zaboravlja na bezbednosne principe. Tipično se napadač predstavlja kao administrator i zahteva od žrtve da potvrdi lozinku ili PIN.

U klasu napada koja koristi socijalni inženjering spada **pecanje** (engl. **phishing**). Cilj napadača je da dođe do lozinke i detalja o kreditnoj kartici. Korisniku se šalje elektronska pošta ili trenutna poruka koja izgleda oficijelno. Korisniku se u tekstu poruke navede link banke koji je vizuelno tačan, ali je iza tog URL aplikacije koju koristi napadač. Kada se korisnik autorizuje, misleći da je na serveru banke, napadač snimi njegove autorizacione detalje.

PRIMERI PHISHING NAPADA

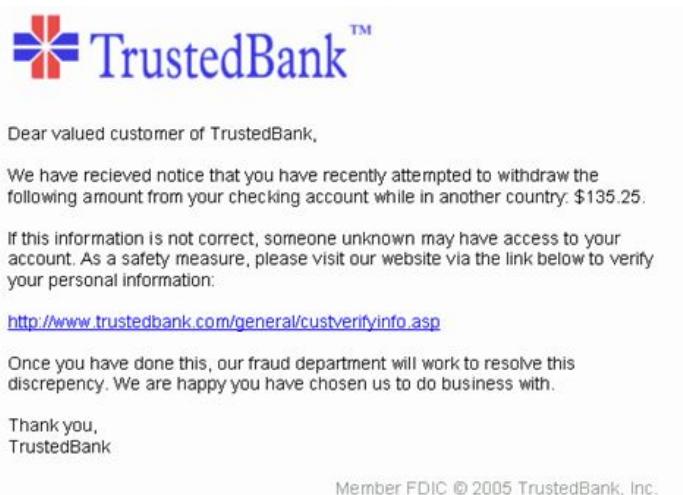
Napadač se predstavlja kao banka kod koje korisnik ima račun i moli ga da potvrdi svoje podatke

Na narednoj slici je prikazan jedan tipičan primer elektronske poruke koju dobija korisnik prilikom pecanja. Napadač se predstavlja kao banka kod koje korisnik ima račun i moli ga da potvrdi svoje podatke. Adresa banke deluje ispravno i korisnik se može lako „upecati“.



Slika 2.1 Phishing email [Izvor: Autor, snapshot]

U narednom primeru se „banka“ obraća korisniku sa informacijom da je neko podigao u inostranstvu 135 \$ sa računa, ostavljajući mogućnost da je to neko drugi uradio. Da bi korisnik proverio svoje stanje na računu treba da klikne na ponuđeni link što će otvoriti lažnu stranicu za autentikaciju. Ova stranica je na veb serveru napadača, tako da će korisnik ostaviti svoje podatke napadaču.



Slika 2.2 Phishing email [Izvor: Autor, snapshot]

▼ Poglavlje 3

BEZBEDONOSNI DOMENI

OSNOVA BEZBEDNOSTI U PRAKSI

Bezbednosni domen je jedan domen poverenja koji ima jedinstvena bezbednosna pravila i zajednički menadžment

Bezbednost računarskih sistema zavisi od mnogo hardverskih, softverskih, organizacionih, ljudskih i drugih faktora. Uticajem na pojedine faktore se može poboljšati bezbednost u pojedinim domenima. Ovi domeni se u praksi preklapaju pa nije očigledno šta je to što utiče na bezbednost. Zbog toga se definišu bezbednosni domeni.

Bezbednosni domen je jedan domen poverenja koji ima jedinstvena bezbednosna pravila i zajednički menadžment. Nekada se pod bezbednosnim domenom smatrao jedan izolovani sistem. Međutim, razvoj mreža i Interneta često zahtevaju da bezbednosni domen obuhvata više sistema.

Da bi obezbedio zajedničke termine i znanja International Information Systems Security Certification Consortium (ISC) je definisao deset bezbednosnih domena.

Ovi domeni predstavljaju osnovu bezbednosti u praksi. Nazivi ovih domena su:

- Bezbednosni menadžment
- Sistemi za kontrolu pristupa i metodologija
- Bezbednost telekomunikacija i mreže
- Kriptografija
- Bezbednosna arhitektura i modeli
- Operaciona (radna) sigurnost
- Bezbednost razvoja aplikacija i sistema
- Fizička bezbednost
- Planiranje kontinualnosti poslovanja i oporavak
- Zakoni, istraga i etika

BEZBEDNOSNI MENADŽMENT

Trijada poverljivost, integritet i raspoloživost su tri cilja na osnovu kojih se meri bezbednost.

Domen bezbednosnog menadžmenta uspostavlja osnove za rad bezbednosnih profesionalaca identificujući ključne koncepte, upravljanje i definicije. National Institute of Standards and Technology definiše računarsku bezbednost kao „zaštitu informacionog sistema u cilju

očuvanja integriteta, raspoloživosti i poverljivost resursa informacionog sistema. Pod resursima IS se smatra hardver, softver, firmver, podaci, i telekomunikacije.



Slika 3.1 Trijada bezbednosti [Izvor: Autor]

Trijada poverljivost, integritet i raspoloživost su tri cilja na osnovu kojih se meri bezbednost.

Poverljivost je zahtev da lične i poverljive informacije ne treba da budu otkrivene neautorizovanim pojedincima.

Integritet podataka je zahtev da se podaci i programi mogu menjati samo na specifikovani i autorizovani način. Integritet sistema je zahtev da sistem izvršava planirane funkcije na nesmetan način, bez namernog ili slučajnog neautorizovanog manipulisanja sistemom.

Raspoloživost je zahtev da sistem odgovara promptno i da ne uskraćuje pružanje usluga autorizovanim korisnicima.

DOMEN BEZBEDONOSNOG MENADŽMENTA

Domen bezbednosnog menadžmenta uključuje klasifikaciju podataka na obične, osetljive, poverljive i strogo poverljive

Ključni korak u bezbednosnom menadžmentu je analiza rizika, tj. identifikacija pretnji i ranjivosti i njihovo balansiranje sa bezbednosnim kontrolama i merama. Kroz analizu rizika organizacija može da sagleda potencijalne gubitke. Na osnovu ove vrednosti određuje se najpogodnije mere bezbednosti.

Domen bezbednosnog menadžmenta uključuje klasifikaciju podataka na obične, osetljive, poverljive i strogo poverljive. Proces klasifikacije podataka pomaže organizaciji da identificuje kritične informacije, obezbedi osnovu za upravljanje pristupom i napravi razliku između tipova zaštite koje su joj potrebni.

Komponente bezbednosnog menadžmenta su i dokumentacija i svest. Organizacija treba da održava dokumente kojima se definišu politika, procedure, smernice i standardi vezani za bezbednost. Zaposleni treba da budu svesni bezbednosne politike koju organizacija vodi i njihove uloge u realizaciji bezbednosti.

UPRAVLJANJE PRISTUPOM

Upravljanje pristupom podrazumeva sprečavanje neautorizovanih korisnika da pristupe sistemu i/ili koriste i menjaju podatke.

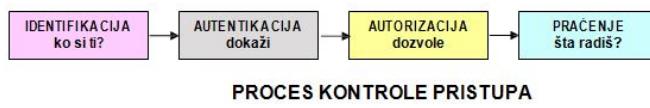
U cilju održavanja poverljivosti, integriteta i raspoloživosti važno je upravljati pristupom informacionom sistemu. Upravljanje pristupom podrazumeva sprečavanje neautorizovanih korisnika da pristupe sistemu i/ili koriste i menjaju podatke. Pored toga, upravljanje pristupom sprečava i autorizovane korisnike da vrše neautorizovane promene podataka. Kada se planira tip potrebnog upravljanja pristupom, organizacija treba da odredi rizik, pretnje i ranjivost sistema.

Mere vezane za pristup se mogu podeliti na: preventivne, upozoravajuće i korektivne.

Preventivne mere treba da spreče štetan događaj.

Upozoravajuće mere otkrivaju da se štetan događaj desio. Korektivne mere se koriste za restauraciju sistema nakon štetnog događaja.

Proces kontrole pristupa se sastoji od 4 koraka. Ključ uspešne kontrole pristupa je dobar sistem za identifikaciju i autentikaciju korisnika. Nakon toga se izvodi autorizacija, odnosno dodeljivanja prava korisniku da koristi određene računarske resurse i podatke. Svo vreme dok je korisnik prijavljen na sistemu vrši se praćenje rada korisnika.



Slika 3.2 Proces kontrole pristupa [Izvor: Autor]

SIGURNOST TELEKOMUNIKACIJA I MREŽE

Domen sigurnosti telekomunikacije i mreža utiče na sva tri elementa trijade bezbednosti

Domen sigurnosti telekomunikacija i mreže se bavi mrežnom strukturom, metodima komunikacije, formatima za transport i merama za obezbeđenje mreže i prenosa podataka. Ovaj domen utiče na sva tri elementa trijade bezbednosti. U narednoj tabeli je prikazano kojim metodama domen sigurnosti telekomunikacija i mreže utiče na pojedine elemente trijade bezbednosti.

Poverljivost	Integritet	Raspoloživost
<ul style="list-style-type: none"> • Protokoli mrežne sigurnosti • Usluge mrežne autentikacije • Usluge šifriranja podataka 	<ul style="list-style-type: none"> • Usluge zaštitnih zidova • Upravljanje sigurnošću komunikacija • Usluge detekcije upada 	<ul style="list-style-type: none"> • Tolerancija i otkaza za raspoloživost podataka (sistemi redundantnih diskova, rezervne kopije) • Prijhvatljivo prijavljivanje i performanse u drugih procesima • Pouzdani i interoperabilni bezbednosni mehanizmi za procese i mrežu

Slika 3.3 Metode kojim se služi domen sigurnosti telekomunikacija mreža [Izvor: Autor]

SIGURNOST RAZVOJA APLIKACIJA I SISTEMA

Uključivanje bezbednosnih profesionalaca treba da počne još u konceptualnoj fazi, a njihovo angažovanje treba da se nastavi i kroz faze razvoja, implementacije, testiranja i održavanja.

Bezbednosni profesionalci treba da budu uključeni u proces razvoja softvera da bi osigurali da se tokom celog razvojnog procesa dovoljno pažnje posveti bezbednosti aplikacije ili sistema. Uključivanje bezbednosnih profesionalaca treba da počne još u konceptualnoj fazi, a njihovo angažovanje treba da se nastavi i kroz faze razvoja, implementacije, testiranja i održavanja. Sledeća lista identificuje ključna pitanja u pojedinim fazama razvojnog ciklusa:

- **Izvodljivost sistema:** Identifikovati bezbednosne zahteve, pravila, standarde i druge dokumente koji će biti potrebni
- **Planiranje zahteva i softvera:** Identifikovati ranjivosti, pretnje i rizike. Planirati odgovarajući nivo zaštite. Izvršiti analizu troškovi-dobit.
- **Projektovanje proizvoda:** Ugraditi specifikaciju mehanizama bezbednosti u projekat proizvoda (upravljanje pristupom, šifriranje, itd.)
- **Detaljno projektovanje:** Projektovati detaljno bezbednosne module saglasno poslovnim potrebama i zakonskim obavezama.
- **Kodiranje:** Napisati bezbednosne module i odgovarajuću dokumentaciju
- **Integracija proizvoda:** Testirati bezbednosne mere ugrađene u softver i izvršiti potrebna poboljšanja
- **Implementacija:** Implementirati bezbednosne mere i testirati ih pre svakodnevne upotrebe
- **Rad i održavanje:** Nadgledati bezbednosni softver. Predlagati promene, vršiti testiranja za odgovarajuće pretnje i implementirati adekvatne promene kada je potrebno.

KRIPTOGRAFIJA I KRIPTOANALIZA

Domen kriptografije je zadužen za šifrovanje informacija pre njihovog slanja i dešifrovanja samo autorizovanim osobama.

Domen kriptografije obezbeđuje bezbednosne mere koje osiguravaju da prenesene informacije mogu da budu čitane i razumevane samo od strane autorizovanih osoba. Drugim rečima, domen kriptografije je zadužen za šifrovanje informacija pre njihovog slanja i dešifrovanja samo autorizovanim osobama.

Kriptografija je naučna disciplina koja se bavi proučavanjem metoda za slanje poruka u takvom obliku da ih samo onaj kome su namenjene može pročitati. Neki elementi kriptografije bili su prisutni već kod starih Grka. Naime, Spartanci su u 5. veku p.n.e upotrebljavali napravu za šifrovanje, koja se zvala skital. Skital je bio drveni štap oko kojeg se namotavala vrpca od pergamenta, pa se na nju uspravno pisala poruka. Nakon upisivanja poruke, vrpca bi se odmotala, a na njoj bi ostali izmešani znakovi koje je mogao pročitati samo onaj ko je imao štap jednake debljine.

Osnovni zadatak kriptografije je omogućiti dvema osobama (mi ih nazivamo pošiljalac i primalac - u kriptografskoj literaturi su za njih rezervisana imena Alice i Bob) da komuniciraju preko nesigurnog komunikacionog kanala (telefonska linija, računarska mreža, ...) na način tako da treća osoba (njihov "napadač" ili osluškivač - u literaturi se najčešće zove Eva ili Oskar), koja može nadzirati komunikacioni kanal, ne može da razume njihove poruke. Poruku koju pošiljalac želi da pošalje primaocu nazivamo otvoreni tekst (engl. plaintext). To može biti tekst na njihovom maternjem jeziku, numerički podaci ili bilo šta drugo. Pošiljalac transformiše otvoreni tekst koristeći unapred dogovoren ključ. Taj postupak se naziva **šifriranje**, a dobijeni rezultat šifrovani tekst (engl. ciphertext). Nakon toga pošiljalac šalje šifrovani tekst preko nekog komunikacionog kanala. Napad prisluškivanjem može da sazna sadržaj šifrovanog teksta, ali ne može odrediti otvoreni tekst. Za razliku od njega, primalac koji zna ključ kojim je šifrirana poruka, može da dešifruje preneti tekst i da odredi otvoreni tekst.

Za razliku od dešifrovanja, **kriptoanaliza** ili dekriptovanje je naučna disciplina koja se bavi proučavanjem postupaka za čitanje skrivenih poruka bez poznavanja ključa. **Kriptologija** je grana nauke koja obuhvata kriptografiju i kriptoanalizu.

BEZBEDONOSNA ARHITEKTURA I MODELI

Bezbednosni profesionalac treba da sagleda sve probleme arhitekture sistema i primeni adekvatne mere bezbednosti

Bezbednosni profesionalci treba da razumeju celokupan informacioni sistem da bi razvili odgovarajuću **bezbednosnu arhitekturu**. Na primer, informacioni sistem baziran na klijent-server arhitekturi zahteva različit bezbednosni pristup u odnosu na sisteme bazirane na troslojnoj arhitekturi tipičnoj za veb aplikacije. Bezbednosni profesionalac treba da sagleda sve probleme arhitekture sistema i primeni adekvatne mere bezbednosti.

- Modeli bezbednosti informacija se koriste da se organizuju i formalizuju pravila bezbednosti. Modeli bezbednosti obezbeđuju pri tom koncept i okvir rada. Postoje tri glavna tipa bezbednosnih modela:
- Upravljanje pristupom: Ovaj model je čest u zdravstvu i sličnim organizacijama gde se radi sa poverljivim ličnim podacima. Model omogućuje da se podaci klasifikuju da bi se ograničio pristup svim neautorizovanim osobama.

- **Integritet:** Ovaj model ne štiti samo poverljivost, nego i integritet podataka. Model bezbednosti baziran na integritetu sprečava da informacije budu modifikovane od neautorizovanih osoba i sprečava autorizovane osobe da vrše neautorizovane promene.
- **Informacioni tok:** U ovom modelu informacije su klasifikovane i prenose se od osobe do osobe na način specifikovan bezbednosnom politikom i pravilima.

DOMEN RADNE BEZBEDNOSTI

Domen radne ili operacione bezbednosti brine o zaštiti hardvera, softvera i informacionih resursa, vršeći ispitivanje i nadgledanje, kao i određivanje pretnji i ranjivosti sistema.

Domen radne ili operacione bezbednosti brine o zaštiti hardvera, softvera i informacionih resursa , vršeći ispitivanje i nadgledanje, kao i određivanje pretnji i ranjivosti sistema. Na raspolaganju su mnoge mere koje organizacija može da preduzme kako bi osigurala bezbedan rad. Organizacije koje brinu o radnoj bezbednosti primenjuju sledeće metode:

- Preventivne mere čiji je zadatak da spreče pretnju koja nastaje od nemamernih grešaka ili pristupa sistemu od neautorizovanih korisnika
- Mere otkrivanja kojima se identificuje pojava greške
- Odvajanje obaveza dodeljivanjem zadataka različitim osobama, kako jedna osoba ne bi imala potpunu kontrolu nad sistemom bezbednosti
- Pravljenje rezervnih kopija i restauracija sistema
- Mere za praćenje i odobravanje rekonfiguracije sistema
- Praćenje pristupa zaposlenih osetljivim podacima ili sistemu za očuvanje bezbednosti
- Čuvanje podataka tokom dužeg vremenskog perioda saglasno politici organizacije
- Zaštita hardvera, softvera i informacionih resursa.

Pored preduzimanja ovih mera bezbednosti, domen radne bezbednosti podrazumeva ocenjivanje i praćenje bezbednosti sistema. Za to se koriste tri tehnike: otkrivanje upada, testiranje probosa u sistem i analiza kršenja bezbednosnih pravila.

DOMEN FIZIČKE SIGURNOSTI

Domen fizičke sigurnosti se stara o pristupu prostorijama u kojima se nalaze računarski resursi, video nadzoru, alarmnim sistemima itd.

Domen fizičke sigurnosti sistema se odnosi na okolinu koja okružuje informacioni sistem i njegove komponente. Ključni zadatak ovog domena je da identificuje i spreči pretnje i ranjivost sistema primenjujući adekvatne kontramere koje će fizički zaštititi sistem.

Domen fizičke sigurnosti se stara o pristupu prostorijama u kojima se nalaze računarski resursi, video nadzoru, alarmnim sistemima itd. U moguće pretnje i ranjivosti, pored ostalog, treba uzeti u obzir i sabotaže, nestanak električne energije, kvarovi na instalacijama, pregrevanje prostora, požari i prirodne katastrofe, kao što su na primer poplave, zemljotresi, itd.

PLANIRANJE KONTINUALNOSTI POSLOVANJA I OPORAVAK

Plan oporavka sadrži mere koje je potrebno preduzeti da bi se, nakon incidenta, sistem potpuno sanirao i vratio u stanje pre incidenta.

Ovaj domen obuhvata izradu dve vrste planova:

- Planiranje kontinualnosti poslovanja
- Planiranje oporavka posle incidenta

Iako su ova dva koncepta slična, postoje i značajne razlike. Plan kontinualnosti poslovanja treba da sadrži mere koje će obezbititi neprekidno funkcionisanje glavnih poslovnih procesa i u slučaju incidenta. Plan oporavka sadrži mere koje je potrebno preduzeti da bi se, nakon incidenta, sistem potpuno sanirao i vratio u stanje pre incidenta.

ZAKONI, ISTRAGA I ETIKA

Bezbednosni profesionalci se moraju ponašati u skladu sa specifičnim etičkim principima.

Poslednji bezbednosni domen se odnosi na poznavanje zakona, načina istrage i etike. Od bezbednosnih profesionalaca se očekuje da dobro poznaju lokalne i međunarodne zakone koji su vezani za bezbednost informacionih sistema. Takođe se očekuju da oni mogu da prepoznaju različite oblike računarskog kriminala, kao i da izvrše istragu i sačuvali evidenciju o kriminalnom delu.

Bezbednosni profesionalci su u prilici da pristupaju najosetljivijim delovima informacionog sistema i zaštićenim podacima. Zbog toga moraju da se ponašaju u skladu sa specifičnim etičkim principima.

Među najvažnijim etičkim principima koji se odnose na bezbednosne profesionalce su:

- Zaštita društva, zajedničkog dobra i infrastrukture
- Delovati časno, iskreno, pravedno, odgovorno i legalno
- Raditi marljivo i kompetentno
- Unapredijevati i štititi profesiju.

▼ Poglavlje 4

FORENZIKA

ISTRAGA I FORENZIKA

Računarska istraga je proces prikupljanja, zaštite, analize i prezentacije dokaza koji su dovoljno pouzdani i ubedljivi kao dokazni materijal na sudu

Elektronska evidencija i prikupljanje informacija postaju sve važnija tema sa povećanjem broja računarskih napada i narastanjem računarskog kriminala. Zvanični istražni organi i sudovi prihvataju u redovnim zakonskim procedurama elektronsku evidenciju i ravnopravno je tretiraju kao i ostali klasični dokazni materijal. Metodima prikupljanja elektronskih dokaza se bavi nauka koja se naziva računarska istraga (engl. computer forensic) ili računarska forenzika. Osobe koje se bave računarskom istragom nazivaju se računarski forenzičari.

Računarska istraga je proces prikupljanja, zaštite, analize i prezentacije dokaza koji su dovoljno pouzdani i ubedljivi kao dokazni materijal na sudu. Računarska istraga se zasniva na metodološkom ispitivanju memorijskih medija, kao što su diskovi, diskete, trake, kompakt diskovi i drugi, radi otkrivanju dokaza o neovlašćenom ili nedozvoljenom korišćenju informacionih resursa. Na osnovu prikupljenih dokaza se mogu rekonstruisati aktivnosti korisnika. Računarski dokazi se mogu koristiti na sudu, ali i u disciplinskim postupcima u organizaciji.

Uspeh računarskih istraga se zasniva na činjenici da računarski sistemi čuvaju mnogo više podataka nego što obični korisnici zamišljaju. Obrisana datoteka ne znači da je nemoguće rekonstruisati podatke koje je ona sadržala. Mnoge log datoteke sadrže informacije o aktivnostima pojedinih korisnika. Iskusni računarski forenzičar može na osnovu zatečenih podataka na medijima da pronađe potrebne podatke i da ih interpretira.

▼ Poglavlje 5

PRAVNI SISTEM

ZAKONI U SRBIJI

Krivični zakonik Republike Srbije definiše krivična dela vezana za korišćenje računara.

Pravnu osnovu za računarsku istragu čini pravni sistem zemlje u kojoj je počinjen incident. Nepostojanje nekog zakona ne oslobađa krvice počinioца incidenta. Pored zakona postoje i pravilnici organizacija koji takođe mogu da sankcionišu svoje zaposlene za neautorizovano korišćenje računarskih resursa. Treba imati na umu da postoje i međunarodni zakoni koje ipak treba poštovati, bez obzira što ne važe na teritoriji naše zemlje.

Kada se radi o pravnom sistemu Republike Srbije, treba imati na umu da postoji nekoliko zakona koji se manje ili više odnose na bezbednost informacionih sistema. U ove zakone spadaju:

- Krivični zakonik Republike Srbije, objavljen 2005. godine
- Zakon o organizaciji i nadležnosti državnih organa za borbu protiv visoko-tehnološkog kriminala, objavljen 2005. godine
- Zakon o telekomunikacijama, objavljen 2003. Godine
- Zakon o elektronskom potpisu
- Zakon o oglašavanju
- Zakon o autorskim i srodnim pravima

Krivični zakonik Republike Srbije definiše krivična dela vezana za korišćenje računara. Glava dvadeset sedma Krivičnog zakonika se posebno bavi oblašću Krivična dela protiv bezbednosti računarskih podataka. Krivični zakonik sankcioniše dela kao što su: oštećenje računarskih programa i podataka, računarska sabotaža, pravljenje i unošenje računarskih virusa, računarska prevara, neovlašćeni pristup zaštićenom računaru, računarskoj mreži i elektronskoj obradi podataka, sprečavanje i ograničavanje pristupa javnoj računarskoj mreži i neovlašćeno korišćenje računara ili računarske mreže.

OŠTEĆENJE RAČUNARSKIH PODATAKA I PROGRAMA

Zbog velike važnosti ovog zakona, navode se u celini članovi koji se odnose na dela protiv bezbednosti računarskih podataka.

Zbog velike važnosti ovog zakona, navode se u celini članovi koji se odnose na dela protiv bezbednosti računarskih podataka.

Oštećenje računarskih podataka i programa

Član 298.

1. Ko neovlašćeno izbriše, izmeni, ošteti, prikrije ili na drugi način učini neupotrebljivim računarski podatak ili program, kazniće se novčanom kaznom ili zatvorom do jedne godine.
2. Ako je delom iz stava 1. ovog člana prouzrokovana šteta u iznosu koji prelazi četristopedeset hiljada dinara, učinilac će se kazniti zatvorom od tri meseca do tri godine.
3. Ako je delom iz stava 1. ovog člana prouzrokovana šteta u iznosu koji prelazi milion i petsto hiljada dinara, učinilac će se kazniti zatvorom od tri meseca do pet godina.
4. Uređaji i sredstva kojima je učinjeno krivično delo iz st. 1. i 2. ovog člana, ako su u svojini učinioca, oduzeće se.

Računarska sabotaža

Član 299.

Ko unese, uništi, izbriše, izmeni, ošteti, prikrije ili na drugi način učini neupotrebljivim računarski podatak ili program ili uništi ili ošteti računar ili drugi uređaj za elektronsku obradu i prenos podataka sa namerom da onemogući ili znatno omete postupak elektronske obrade i prenosa podataka koji su od značaja za državne organe, javne službe, ustanove, preduzeća ili druge subjekte, kazniće se zatvorom od šest meseci do pet godina.

VIRUSI I RAČUNARSKA PREVARA

Ko napravi računarski virus u nameri njegovog unošenja u tuđ računar ili računarsku mrežu,kazniće se novčanom kaznom ili zatvorom do šest meseci.

Pravljenje i unošenje računarskih virusa

Član 300.

1. Ko napravi računarski virus u nameri njegovog unošenja u tuđ računar ili računarsku mrežu,kazniće se novčanom kaznom ili zatvorom do šest meseci.
2. Ko unese računarski virus u tuđ računar ili računarsku mrežu i time prouzrokuje štetu, kazniće se novčanom kaznom ili zatvorom do dve godine.
3. Uređaj i sredstva kojima je učinjeno krivično delo iz st. 1. i 2. ovog člana oduzeće se.

Računarska prevara

Član 301.

1. Ko unese netačan podatak, propusti unošenje tačnog podatka ili na drugi način prikrije ili lažno prikaže podatak i time utiče na rezultat elektronske obrade i prenosapodataka u nameri da sebi ili drugom pribavi protivpravnu imovinsku korist i time drugom prouzrokuje imovinsku štetu,kazniće se novčanom kaznom ili zatvorom do tri godine.
2. Ako je delom iz stava 1. ovog člana pribavljenim imovinskim korist koja prelazi iznos od četristopadeset hiljada dinara,učinilac će se kazniti zatvorom od jedne do osam godina.
3. Ako je delom iz stava 1. ovog člana pribavljenim imovinskim korist koja prelazi iznos od milion i petsto hiljada dinara, učinilac će se kazniti zatvorom od dve do deset godina.
4. Ko delo iz stava 1. ovog člana učini samo u nameri da drugog ošteti, kazniće se novčanom kaznom ili zatvorom do šest meseci.

NEOVLAŠĆENI PRISTUP I KORIŠĆENJE

Ko se, kršeći mere zaštite, neovlašćeno uključi u računar ili računarsku mrežu, kazniće se novčanom kaznom ili zatvorom do šest meseci.

Neovlašćeni pristup zaštićenom računaru, računarskoj mreži i elektronskoj obradi podataka

Član 302.

1. Ko se, kršeći mere zaštite, neovlašćeno uključi u računar ili računarsku mrežu, ili neovlašćeno pristupi elektronskoj obradi podataka,kazniće se novčanom kaznom ili zatvorom do šest meseci.
2. Ko upotrebi podatak dobijen na način predviđen u stavu 1. ovog člana, kazniće se novčanom kaznom ili zatvorom do dve godine.
3. Ako je usled dela iz stava 1. ovog člana došlo do zastoja ili ozbiljnog poremećaja funkcionsanja elektronske obrade i prenosa podataka ili mreže ili su nastupile druge teške posledice,učinilac će se kazniti zatvorom do tri godine.

Neovlašćeno korišćenje računara ili računarske mreže

Član 304.

1. Ko neovlašćeno koristi računarske usluge ili računarsku mrežu u nameri da sebi ili drugom pribavi protivpravnu imovinsku korist, kazniće se novčanom kaznom ili zatvorom do tri meseca.

2. Gonjenje za delo iz stava 1. ovog člana preduzima se po privatnoj tužbi.

Sprečavanje i organičavanje pristupa javnoj računarskoj mreži

Član 303.

1. Ko neovlašćeno sprečava ili ometa pristup javnoj računarskoj mreži, kazniće se novčanom kaznom ili zatvorom do jedne godine.
2. Ako delo iz stava 1. ovog člana učini službeno lice u vršenju službe, kazniće se zatvorom do tri godine.

Pravljenje, nabavljanje i davanje drugom sredstava za izvršenje krivičnih dela protiv bezbednosti računarskih podataka

Član 304 a.

1. Ko poseduje, pravi, nabavlja, prodaje ili daje drugom na upotrebu računare, računarske sisteme, računarske podatke i programe radi izvršenja krivičnog dela iz čl. 298. do 303. ovog zakonika, kazniće se zatvorom od šest meseci do tri godine.
2. Predmeti iz stava 1. ovog člana oduzeće se.

DEFINICIJE IZRAZA

Računarskim programom smatra se uređeni skup naredbi koji služe za upravljanje radom računara, kao i za rešavanje određenog zadatka pomoću računara.

U zakonu se posebno, u članu 112, u stavovima 16 do 20 definiše i značenje pojedinih izraza.

- Pokretnom stvari se smatra i svaka proizvedena ili sakupljena energija za davanje svetlosti, toploće ili kretanja, telefonski impuls, kao i računarski podatak i računarski program.
- Računarskim podatkom se smatra predstavljena informacija, znanje, činjenica, koncept ili naredba koji se unosi, obrađuje ili pamti ili je unet, obrađen ili zapamćen u računaru ili računarskoj mreži.
- Računarskom mrežom smatra se skup međusobno povezanih računara koji komuniciraju razmenjujući podatke.
- Računarskim programom smatra se uređeni skup naredbi koji služe za upravljanje radom računara, kao i za rešavanje određenog zadatka pomoću računara.
- Računarski virus je računarski program ili neki drugi skup naredbi unet u računar ili računarsku mrežu koji je napravljen da sam sebe umnožava i deluje na druge programe ili podatke u računaru ili računarskoj mreži dodavanjem tog programa ili skupa naredbi jednom ili više računarskih programa ili podataka.

ZAKON O ORGANIZACIJI I NADLEŽNOSTI DRŽAVNIH ORGANA ZA BORBU PROTIV VISOKOTEHNOLOŠKOG KRIMINALA

Suština zakona je da se za navedena dela u Okružnom javnom tužilaštvu u Beogradu obrazuje posebno odeljenje za borbu protiv visokotehnološkog kriminala

Drugi zakon koji je posebno važan za računarsku istragu je Zakon o organizaciji i nadležnosti državnih organa za borbu protiv visokotehnološkog kriminala. Njime se uređuje se obrazovanje, organizacija, nadležnost i ovlašćenja posebnih organizacionih jedinica državnih organa radi otkrivanja, krivičnog gonjenja i suđenja za krivična dela određena ovim zakonom.

Zakon pod visokotehnološkim kriminalom smatra vršenje krivičnih dela kod kojih se kao objekat ili sredstvo izvršenja krivičnih dela javljaju računari, računarske mreže, računarski podaci, kao i njihovi proizvodi u materijalnom ili elektronskom obliku.

Pod proizvodima u elektronskom obliku posebno se podrazumevaju računarski programi i autorska dela koja se mogu upotrebiti u elektronskom obliku.

- Ovaj zakon primenjuje se radi otkrivanja, krivičnog gonjenja i suđenja za:
- krivična dela protiv bezbednosti računarskih podataka određena krivičnim zakonom;
- krivična dela protiv intelektualne svojine, imovine i pravnog saobraćaja kod kojih se kaoobjekat ili sredstvo izvršenja krivičnih dela javljaju računari, računarske mreže, računarski podaci, kao i njihovi proizvodi u materijalnom ili elektronskom obliku, ako broj primeraka autorskih dela prelazi 500 ili nastala materijalna šteta prelazi iznos od 850.000 dinara.

Suština zakona je da se za navedena dela u Okružnom javnom tužilaštvu u Beogradu obrazuje posebno odeljenje za borbu protiv visokotehnološkog kriminala, kako bi se stručno obradili sve češći slučajevi kriminala u oblasti računarstva.

DIGITALNA ISTRAGA I NJEN ODнос SA DRUGIM ISTRAGAMA

Računarska istraga se u nekim slučajevima kombinuje sa drugim istragama.

Računarska istraga može da pruži mnoge dokaze vezane za zloupotrebu računarskih sistema. Međutim, to ne mora da bude samo za sebe dovoljno za pokretanje krivičnog postupka. Zbog toga se računarska istraga u nekim slučajevima kombinuje sa drugim istragama.

Ako je, na primer, kreker izvršio napad iz internet kafea vrlo teško će se otkriti njegov identitet samo na osnovu digitalnih dokaza. Klasična istraga, međutim, može da dokaže da je kreker baš u vreme napada bio u Internet kafeu i koristio neki računar.

U drugim slučajevima digitalna istraga se može koristiti kao dopuna klasične istrage. Tipičan primer je upoređivanje biometrijskih osobina sumnjive osobe sa podacima u bazi podataka. Vrlo često se tražena osoba identificuje na osnovu fotografije ili otiska prstiju.

Pored saradnje sa kriminalnim tužilaštvom, računarski forenzičari sarađuju i u istragama koje vode civilni parničari, detektivi osiguravajućih društava, vojni istražni organi ili obezbeđenje kompanija.

PRAVILA EVIDENCIJE

Da bi se došlo do valjanih dokaza u računarskoj istrazi treba se držati pravila koje propisuju lokalni i međunarodni zakoni

Da bi se došlo do valjanih dokaza u računarskoj istrazi treba se držati pravila koje propisuju lokalni i međunarodni zakoni. U tom smislu računarska istraga treba da pruži dokaze za koje važi isto što i za kriminalnu istragu. Dokazi moraju da budu:

- Prihvatljivi za sud,
- Autentični,
- Tačni odnosno pouzdani,
- Kompletni,
- Ubedljivi za sudije,
- U saglasnosti sa zakonom i pravom.

Ovo nije lako ako se zna da je prikupljanje računarskih dokaza vezano za sledeće činjenice:

- Računarski podaci mogu da se menjaju iz časa u čas,
- Računarski podaci su nevidljivi za ljudsko oko, a postaju vidljivi indirektno samo ako se pokrenu određene procedure
- Proces prikupljanja podataka može znatno da promeni podatke. Na primer, proces otvaranja datoteke može da promeni neka polja u datoteci i metapodatke o datoteci.

Imajući na umu sve navedene činjenice može se doći do nekih opštih pravila koja važe za proces prikupljanja evidencije o zloupotrebljavanju računarskih sistema:

- Scena kriminalne radnje treba da bude potpuno očuvana, dokazi se moraju prikupiti što je pre moguće i bez kontaminacije sistema.
- Mora se održati kontinualnost i nepromenljivost dokaza od trenutka njihovog prikupljanja do prezentovanja na sudu.
- Sve procedure korišćene u istrazi treba da budu podložne proveri (uditabilne), tj. druga strana na sudu može da se uveri da su dokazi skupljeni valjano.

VERODOSTOJNOST DOKAZA

Računarski forenzičar treba da preduzme više pažljivih koraka kako bi identifikovao i pokušao da prikupi moguće dokaze koji možda postoje na računarskom sistemu napadača

Računarski forenzičar treba da preduzme više pažljivih koraka kako bi identifikovao i pokušao da prikupi moguće dokaze koji možda postoje na računarskom sistemu napadača. Da bi dokazi bili **verodostojni** treba preuzeti sledeće korake:

- Zaštititi napadačev računar tokom računarske istrage od bilo kakvog brisanja, promene, gubljenja podataka ili računarskih virusa.
- Otkriti sve datoteke na napadačevom sistemu. Pored normalnih datoteka, to podrazumeva i obrisane datoteke koje se još mogu rekonstruisati, skrivene datoteke, datoteke zaštićene lozinkom i šifrovane datoteke.
- Restaurirati sve ili bar one koje je moguće obrisane datoteke.
- Otkriti koliko je to moguće sadržaj skrivenih datoteka, kao i temporarnih datoteka ili svap datoteka koje se koriste od strane aplikacija ili operativnog sistema.
- Pristupiti, ukoliko je moguće i zakonski dozvoljeno, sadržaju svih zaštićenih i šifriranih datoteka.
- Analizirati sve potencijalno relevantne podatke pronađene na specijalnim područjima diska. Ovo se odnosi na nealocirani prostor na disku koji trenutno nije zauzet datotekama, a nekada je možda na istom prostoru bila datoteka koja sadrži dokaze.
- Odštampati celokupnu analizu napadačevog sistema zajedno sa listom svih potencijalno relevantnih datoteka.
- Obezbediti opis konfiguracije sistema, korišćenog sistema datoteka, strukturu datoteka, informacije o autorima, pokušaja sakrivanja, brisanja, zaštite i šifriranja informacija 9. Obezbediti konsultacije sa ekspertima i/ili svedoke ako je potrebno.

▼ Poglavlje 6

Pokazna vežba: Enkripcija i dekripcija teksta

ENKRIPCIJA I DEKRIPCIJA TEKSTA

Enkripcija je proces pretvaranja teksta u nerazumljivu formu dok je dekripcija proces pretvaranja enkriptovanog teksta u originalnu formu.

Očekivano vreme izrade zadatka: 45 minuta.

Šifrovanje je proces transformacije teksta tako da on nije razumljiv nikome osim primaocu koji ga dobija. Dešifrovanje je proces pretvaranja šifrovanog teksta u originalno stanje kako bi bio razumljiv. Kriptografski algoritam, koji se naziva i šifra, je matematička funkcija koja se koristi za šifrovanje ili dešifrovanje. U većini slučajeva koriste se dve povezane funkcije, jedna za šifrovanje i druga za dešifrovanje. Kriptografija se uglavnom zasniva na određenom broju koji se naziva ključ koji se mora koristiti uz algoritam kako bi se proizveo šifrovani rezultat. Isti ključ se zatim koristi za dešifrovanje teksta. Dešifrovanje ispravnim ključem je jednostavno. Dešifrovanje bez ispravnog ključa veoma je teško, a u nekim slučajevima i nemoguće.

Enkripcija simetričnim ključem

Kod šifriranja ključa sa simetričnim ključem, ključ za šifrovanje može se izračunati iz ključa za dešifrovanje i obrnuto. Kod većine simetričnih algoritama isti se ključ koristi i za šifrovanje i za dešifrovanje. Implementacija simetričnog šifriranja ključeva može biti vrlo efikasna, zato što ne dovodi do značajnog vremenskog kašnjenja prilikom šifriranja i dešifrovanja. Šifriranje teksta sa simetričnim ključem takođe omogućava stepen autentičnosti, jer se informacije šifrovane jednim simetričnim ključem ne mogu dešifrovati bilo kojim drugim simetričnim ključem.

Dakle, sve dok obe strane koje komuniciraju drže ključeve u tajnosti, svaka strana može biti sigurna da komunicira s drugom sve dok dešifrirane poruke i dalje imaju smisla.

Šifriranje simetričnim ključem igra važnu ulogu u SSL protokolu, koji se koristi za autentifikaciju i šifrovanje preko TCP / IP mreža. SSL takođe koristi tehniku šifriranja javnim ključem, koje su opisane u nastavku.

Enkripcija javnim ključem

Enkripcija javnim ključem (asimetrična enkripcija) uključuje par ključeva, javni i privatni, povezani sa entitetom koji treba da potvrdi svoj identitet elektronskim putem, da potpiše ili šifrira podatke. Svaki javni ključ je objavljen, a odgovarajući privatni ključ čuva se u tajnosti. Podaci šifrovani javnim ključem mogu se dešifrovati samo privatnim ključem.

U poređenju sa enkripcijom simetričnim ključem, šifriranje javnim ključevima zahteva više računanja i zato nije uvek pogodno za velike količine podataka. Međutim, moguće je koristiti

Šifriranje javnim ključem za slanje simetričnog ključa koji se zatim može koristiti za šifriranje dodatnih podataka. Ovo je pristup koji koristi SSL protokol.

CEZAROVA ŠIFRA

Šifrovanje Cezarovom šifrom koristi celobrojnu vrednost kao ključ pomeraja slova u abecedi

Očekivano vreme izrade zadatka: 10 minuta.

Cezarov šifra jedna je od najranijih poznatih i najjednostavnijih šifra. To je vrsta šifre za zamenu u kojoj svako slovo u otvorenom tekstu „pomera“ određeni broj mesta prema abecedi. Na primer, sa smenom od 1, A bi bio zamenjen sa B, B bi postao C, i tako dalje.

Dakle, za šifriranje datog teksta potreban nam je jedan broj koji predstavlja ključ i označava za koliko pozicija će svako slovo biti pomereno.

Da bi šifrovana poruka bila uspešno prosleđena, prvo je potrebno da obe strane imaju 'ključ' za šifru, kako bi ih pošiljalac mogao da šifrira i primalac može da ga dešifruje.

Primer:

Koristeći ključ 4, enkriptovati sledeći tekst:

the quick brown fox jumps over the lazy dog

Rezultat:

Kako je ključ pomeraja 4, svako slovo u šifrovanom tekstu biće pomereno za 4 pozicije. Tako će T preći u X, H će preći u L, itd.

Koristeći ključ 4, dobićemo sledeći rezultat:

xli uymgo fvsar jsb nyqtw sziv xli pedc hsk

VIGENERE ŠIFRA

Vigenere šifra je metoda šifriranja teksta koja koristi jednostavan oblik supstitucije polialfabetom.

Očekivano vreme izrade zadatka: 15 minuta.

Vigenere šifra je metoda šifriranja teksta koja koristi jednostavan oblik supstitucije polialfabetom. Polifabetska šifra je svaka šifra zasnovana na supstituciji, pomoću višestrukih alfabetova za zamenu. Šifriranje originalnog teksta vrši se pomoću Vigenere kvadrata ili Vigenere tabele.

Tabela se sastoji od abecede ispisane 26 puta u različitim redovima, a svaka abeceda se ciklično pomera uлево u poređenju s prethodnom abecedom, što je ekvivalentno svim kombinacijama Cezarovi šifri (26). Na slici je prikazana tabela.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Slika 6.1 Vigenere tabela [Izvor: Autor]

Pored teksta za dešifrovanje i tabele, Vigenere šifra zahteva i ključnu reč, koja se ponavlja tako da je njena ukupna dužina jednak dužini teksta koji se dešifruje. Način šifrovanja objašnjen je na sledećem primeru:

Tekst za šifrovanje: Desifrovanje je veoma zanimljivo

Ključ: vezbe

Sada treba da izjednačimo dužine teksta sa ključem pa će to izgledati ovako:

Desifrovanje je veoma zanimljivo
vezbevezbez ez bevez bevezbevez

Sada uzimamo prvo slovo teksta i u tabeli nađemo kolonu koja odgovara tom slovu. Isto tako pronađemo red koji odgovara prvom slovu ključa i zatim nađemo slovo koje se nalazi na preseku kolone i reda. U ovom slučaju to će biti slovo Y. Prateći isti obrazac dobijamo sledeći **rezultat:**

Yirjjmsubrei if zzslb dvrhnpemup

PLAYFAIR ŠIFRA

Playfair šifra koristi tehniku zamene parova slova

Playfair šifra koristi tehniku zamene parova slova umesto pojedinačnih slova kao u prethodnim šiframa. Da bi se kodirala, poruka se razdvaja na parove od dva slova. Ukoliko su jedno do drugog dva ista slova, jedno se menja slovom X. Na primer, reč HELLO bi bila razdvojena na sledeći način: HE LX LO. Takođe, ukoliko tekst nema paran broj slova, na kraju se dodaje X.

Sledeći korak je kreiranje ključa i tabele alfabetika. Tabela je veličine 5x5. Kreira se tako što se prvo popune polja slovima ključa, a ostatak polja se popunjava ostalim slovima alfabetika. Kako bi tabela bila odgovarajuće veličine, uglavnom se izbacuje slovo J.

Primer:

Ključ: monarchy

Tabela:

M	O	N	A	R
C	H	Y	B	D
E	F	G	I	K
L	P	Q	S	T
U	V	W	X	Z

Slika 6.2 Tabela alfabetika [Izvor: Autor]

Tekst za šifrovanje: instruments

Podeljeni tekst: 'in' 'st' 'ru' 'me' 'nt' 'sx'

Pravila šifriranja:

1. Ukoliko su slova u istoj koloni, uzeti slovo ispod slova iz teksta ("me" -> "cl")
2. Ukoliko su slova u istom redu, uzeti slova na desno ("st" -> "tl")
3. Ako ne važe prva dva pravila, formirati kvadrat od dva slova i zatim uzeti slova koja su na druga dva čoška kvadrata ("nt" -> "rq")

Rezultat: gatlmzclrqtx

Očekivano vreme izrade zadatka: 15 minuta.

"BRUTE FORCE" ANALIZA

Brute force napad predstavlja uzastopne pokušaje različitih kombinacija kako bi se dešifrovaо tekstu

Očekivano vreme izrade zadatka: 5 minuta.

"Brute force" napad predstavlja uzastopne pokušaje različitih kombinacija kako bi se dešifrovaо dati tekstu. Ovi napadi su jednostavnii i pouzdani. Napadači puste računar da radi posao - isprobavajući, na primer, različite kombinacije korisničkih imena i lozinki - sve dok ne pronađu onu koja funkcioniše.

Najosnovnija vrsta brute force napada je napad pomoću rečnika, gde napadač prolazi kroz rečnik mogućih lozinki i isproba ih sve. Napadi sa rečnikom počinju sa nekim prepostavkama o uobičajenim lozinkama koje biste pokušali pogoditi sa liste u rečniku.

Na isti način, napadač može pokušati da pogodi ključ koji je potreban za dešifrovanje neke poruke. Napadač će probati sve moguće ključeve kako bi pronašao jedan koji će uspešno dešifrovati poruku.

▼ Poglavlje 7

Pokazne vežbe: Šifrovanje koristeći Cryptool

POKRETANJE CRYPTOOL-A

Prikaz inicijalnog izgleda Cryptool-a

Očekivano vreme izrade zadatka: 45 minuta.

U okviru vežbi ćemo koristiti alat CrypTool. Cryptool možete preuzeti sa sajta www.cryptool.org

Korak 1: Pokrenite Cryptool

Korak 2: Izaberite New Workspace (slika 1)



Slika 7.1 Izgled Cryptool alata [Izvor: Autor]

CEZAROVA ŠIFRA - KODIRANJE

Prikaz enkripcije teksta pomoću Cezarove metode šifriranja

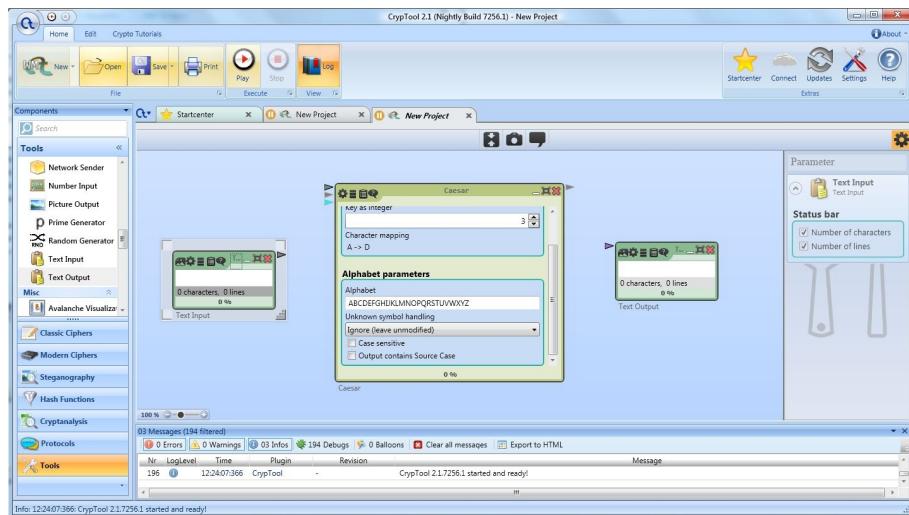
Korak 1: [Components] ->[Tools] ->[Text Input/Text Output]

Korak 2: [Components] -> [Classic Ciphers] -> [Caesar]

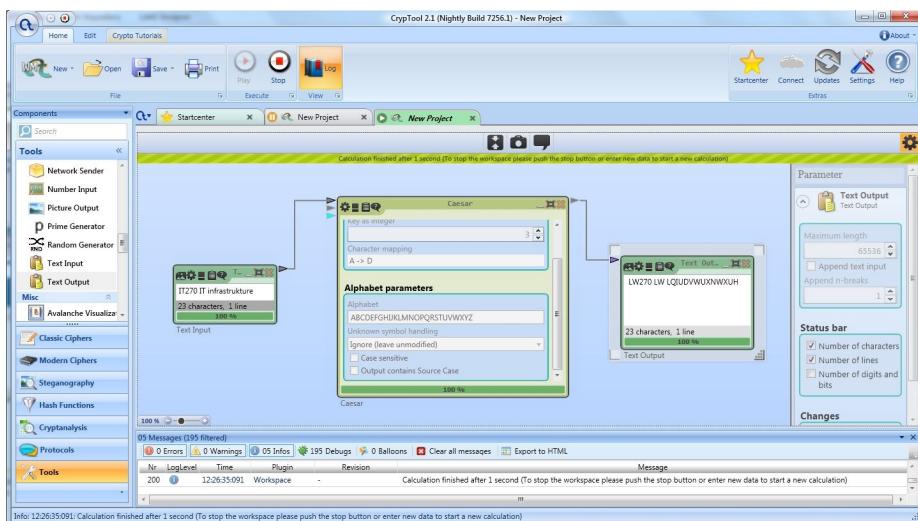
Korak 3: Povežite komponente

Korak 4: Napišite tekst u Text Input prozoru “IT270 IT infrastrukture”

Korak 5: Kliknite na dugme “play”



Slika 7.2 Prikaz dodatih komponenti [Izvor: Autor]



Slika 7.3 Prikaz povezanih komponenti i šifrovane poruke [Izvor: Autor]

CEZAROVA ŠIFRA - DEKODIRANJE

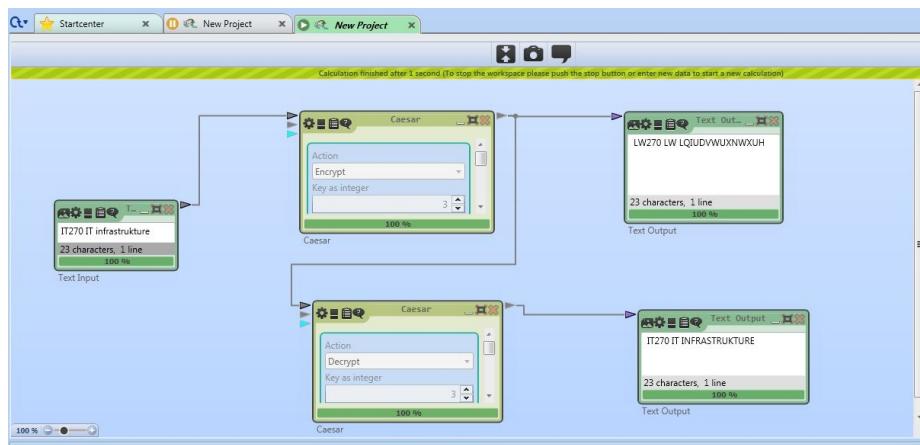
Prikaz dekripcije teksta pomoću Cezarove metode šifriranja

Korak 6: Dodajte još jednu komponentu [Ceaser Cipher] sa opcijom Decrypt

Korak 7: Povežite izlaz iz prve komponente [Ceaser Cipher] sa opcijom Encrypt, sa novokreiranom komponentom

Korak 8: Povežite izlaz iz [Ceaser Cipher] sa Decrypt sa novom komponentom Text Output

Korak 9: Kliknite na dugme "play"



Slika 7.4 Rezultat dekripcije [Izvor: Autor]

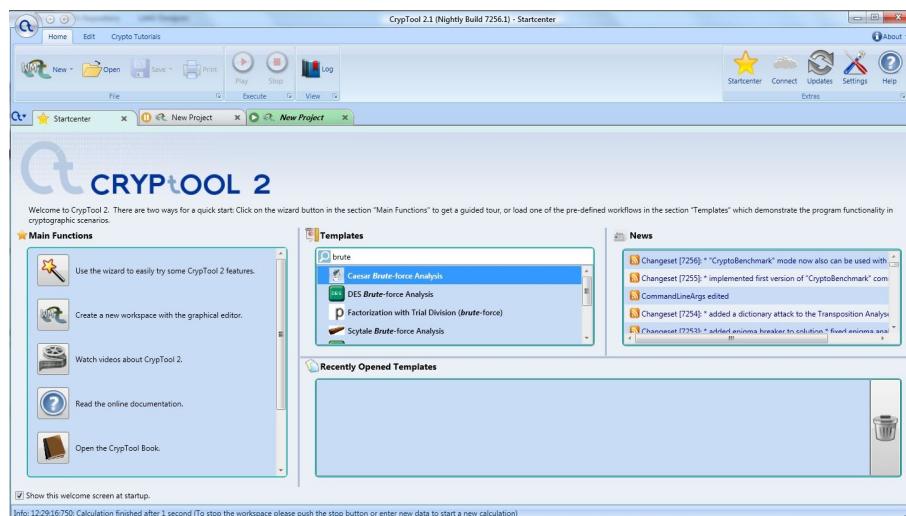
“BRUTE FORCE” ANALIZA

Enkripcija pomoću brute force

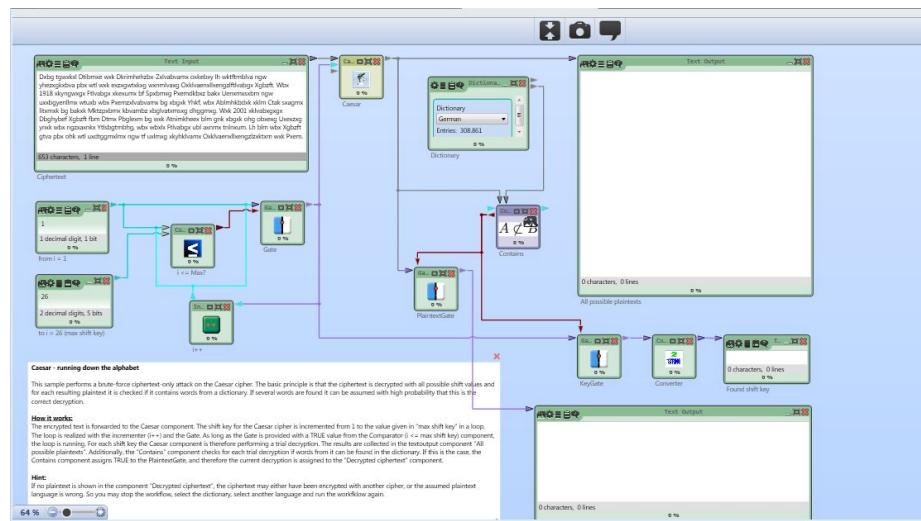
Koristeći postojeći templejt pokrenite “Brute force” analizu. Izlaz će vam dati sve moguće kombinacije za uneti tekst, odnosno za sve moguće ključeve A-Z. Na vama je da pronađete pravu.

Korak 1: [Templates]->[Caesar Brute-force Analysis]

Korak 2: Unesite šifrovani tekst



Slika 7.5 Pronalaženje templejta za Cesear Brute Force Analysis [Izvor: Autor]



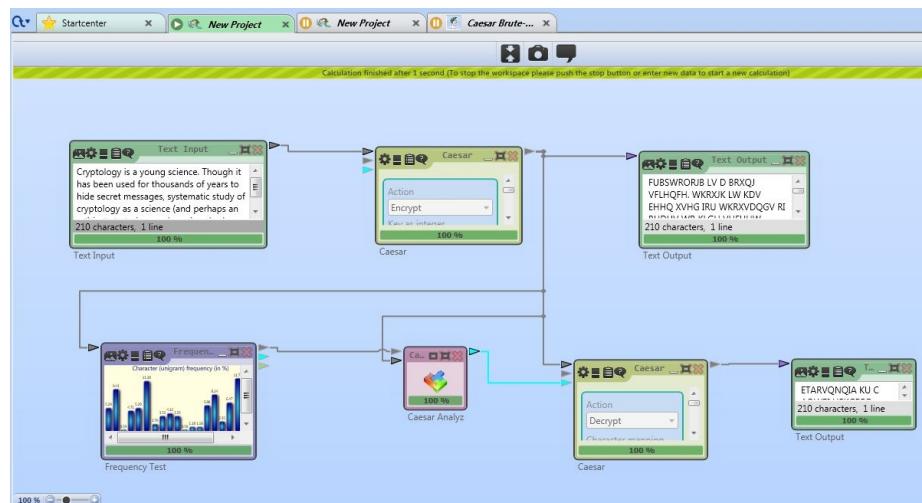
Slika 7.6 Izgled templejta [Izvor: Autor]

ANALIZA FREKVENTNOSTI

U kriptoanalizi, analiza frekventnosti proučava koliko se puta pojavljuje određeni karakter ili set karaktera, kako bi zaključili o kom ključu se radi.

U kriptoanalizi, analiza frekventnosti proučava koliko se puta pojavljuje određeni karakter ili set karaktera, kako bi zaključili o kom ključu se radi. CrypTool će kreirati histogram koji se sastoji od slova u šifrovanom tekstu i pokušaće da uporedi histogram sa engleskim tekstrom.

1. [Components]->[Cryptanalysis]->[Frequency Test]
2. [Components]->[Cryptanalysis]->[Caesar Analyser]
3. [Components] -> [Classic Ciphers] -> [Caesar]
4. Dodati jedan Text Input u [Caesar]
5. Iz [Caesar] enkriptovan tekst povezati sa Text Output-om i
6. [Frequency Test] komponentama
7. Enkriptovan tekst povezati u ulaz "Frequency Test"
8. Enkriptovan tekst i izlaz [Frequency Test] povezati u ulaz [Caesar Analyser]
9. Enkriptovan tekst i izlaz iz [Caesar Analyser] povezati u ulaz druge [Caesar] komponente koja jeza dekripciju
10. Izlaz iz [Caesar] dekripcije je ulaz u "Text Output"
- Start



Slika 7.7 Prikaz postavke za analizu frekventnosti [Izvor: Autor]

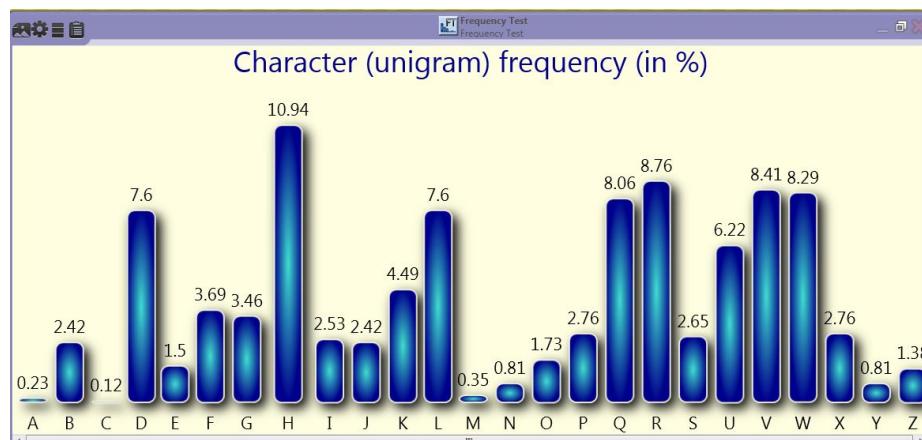
PRIMER - ANALIZE FREKVENTNOSTI

Analiza frekventnosti teksta

Za analizu sledećeg teksta, dobijamo histogram prikazan na slici 8.

Cryptology is a young science. Though it has been used for thousands of years to hide secret messages, systematic study of cryptology as a science (and perhaps an art) just started around one hundred years ago.

The first known evidence of the use of cryptography (in some form) was found in an inscription carved around 1900 BC, in the main chamber of the tomb of the nobleman Khnumhotep II, in Egypt. The scribe used some unusual hieroglyphic symbols here and there in place of more ordinary ones. The purpose was not to hide the message but perhaps to change its form in a way which would make it appear dignified. Though the inscription was not a form of secret writing, but incorporated some sort of transformation of the original text, and is the oldest known text to do so. Evidence of some use of cryptography has been seen in most major early civilizations. "Arthshashtra", a classic work on statecraft written by Kautalya, describes the espionage service in India and mentions giving assignments to spies in "secret writing" - sounds like an ancient version of James Bond?



Slika 7.8 Analiza frekventnosti teksta iz primera [Izvor: Autor, snapshot]

TRANSPOZICIONE ŠIFRE

Koristeći postojeći templejt pokrenite “Transposition Cipher”.

Koristeći postojeći templejt pokrenite “Transposition Cipher”.

Korak 1: [Templates]->[Transposition Cipher]



Slika 7.9 Izgled templejta [Izvor: Autor]

PRIMER TRANSPOZICIONE ŠIFRE

Analiza transpozicionog šifrovanja

Ključ: ZEBRA = 53241

Običan tekst:

Cryptography prior to the modern age was effectively synonymous with encryption, the conversion of information from a readable state to apparent nonsense. The originator of an encrypted message (Alice) shared the decoding technique needed to recover the original information only with intended recipients (Bob), thereby precluding unwanted persons (Eve) from doing the same. The cryptography literature often uses Alice ("A") for the sender, Bob ("B") for the intended recipient, and Eve ("eavesdropper") for the adversary. Since the development of rotor cipher machines in World War I and the advent of computers in World War II, the methods used to carry out cryptology have become increasingly complex and its application more widespread. Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary. It is theoretically possible to break such a system, but it is infeasible to do so by any known practical means.

Šifrovan tekst:

thotdg t ywnihvnrrtlans rt cdals egnn cti m wndp ,eenaps) ahpptrese)tnBBriei,Eer" drneltoimeWWaence I ddaupy eelpn c wrM oye tiham cc oiosdeumihsunkugmdrntyaaathi b aetsa nranyarom aelos p oi m aatoroe noets()e iceerroaooy dctbepdndovon

rlltfsiArs,(fenend(seo sStve rindltvfull hhs rovocno tloes.rpp lema yceereplhegrcalnsim ata a yr elstahsb fldb ie
 rrpt nwfenuhy,csfrn dstansei pe eredeud e nfilhnenoh uuesErie.cg aoul"o r heren epfer emfrehil dop rrttuoyclacnicxipredeyasis mlr tipcyatraia tadt g ihokrene.srls ecy tnb yntm ppi oascyn ettnoiaf ba en.oafnesA ddnh de rln ieis)riiw nemgsTyaiutec" e "o dct "drraaihenrc n he tndleoecoygemrgmasin p nthhydatt orna;thgm noot esoaslhrbicb slttyiok tuieeykpcayoerhheeismicoee fioeeepteTioar gihtc ietohgjaoit i(bchgne dtmetheen hdo" ndp vao)tvyc o tpasoan torWW,ms rtt b ayldaamieocg abohcenpsetcgcr ednpoasmsicos e i dr ecplbs m stsaoolsCg er fvyotrn roooma p nhgrnymecahotqoeveintntereBtyl tr(fohe oyr A(feeb)tt iaevp he.edpoohc rrda msoa e tr oheis e ptodadrivaneaoduc irr ias dunr p,nhr tapcavyioaoerus,iiionwc .

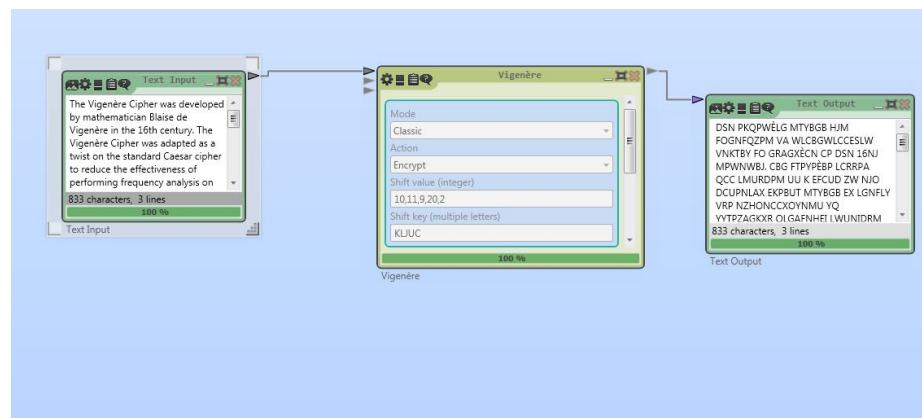
VIGENERE ŠIFRA

Vigenere šifra koristi polialfabetsku šifru zamene

Vigenere šifra je nastala kao poboljšanje Cezarove šifre koja je postala previše laka za dešifrovanje.

Prvi red Vižnerovog kvadrata predstavlja početni alfabet, a ostali redovi predstavljaju 26 šifrovanih alfabetova, od kojih je svaki pomeren za jedno mesto udesno. Tako je prvi šifrovani alfabet Cezarova šifra pomeranja za jedan, drugi - Cezarova šifra pomeranja za dva i tako redom. Vižnerova šifra podrazumeva da se za šifrovanje različitih slova iz poruke koriste različiti redovi Vižnerovog kvadrata (odnosno različiti šifrovani alfabeti).

Ključ za šifrovanje predstavlja bilo koju reč koju odredi onaj koji piše poruku. Primalac poruke mora znati koji je ovaj ključ kako bi uspešno dešifrovaо šifru.



Slika 7.10 Primer šifrovanja Vegenere šifrom [Izvor: Autor]

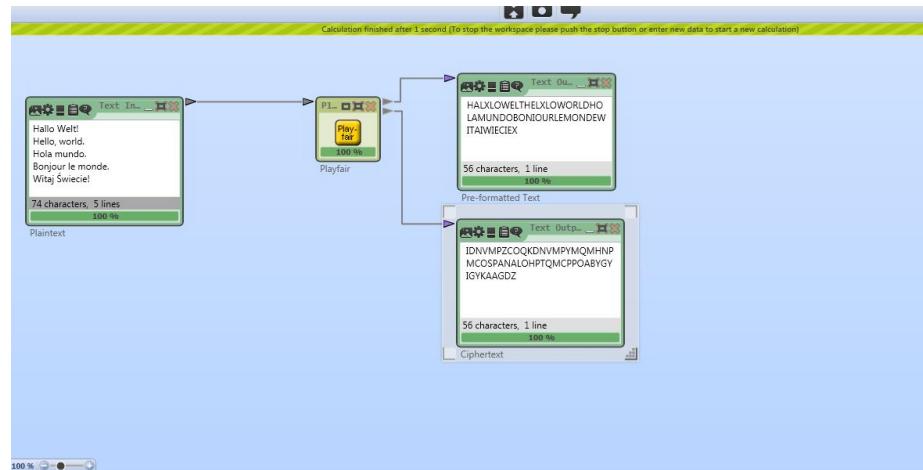
PLAYFAIR ŠIFRA

Postupak šifrovanja sa plejfer šifrom zasniva se na zameni blokova od dva slova ili simbola

Postupak šifrovanja sa plejfer šifrom zasniva se na zameni blokova od dva slova ili simbola (bigrami).

Ukoliko bismo kao ključ koristili 'hello world' ova slova će biti postavljena prva u tabelu. Pre toga, potrebno je izbaciti sva slova koja se ponavljaju, a zatim popuniti tabelu sa ostalim slovima alfabeta redom.

Kada se poruka enkriptuje, tekst se deli u parove od po dva slova, a zatim se vrši zamena slova. Ukoliko poslednji par slova nema dva slova, na kraj se dodaje slovo Z.



Slika 7.11 Šifrovanje pomoću Playfair šifre [Izvor: Autor]

✓ Poglavlje 8

Zadaci za samostalni rad

OPIS ZADATAKA ZA SAMOSTALNU VEŽBU

Dešifrijte sledeći tekst

Očekivano vreme izrade zadatka: 45 minuta.

Zadatak 1 - predviđeno vreme trajanja 20 minuta

Sledeći tekst je enkriptovan transpozicionom šifrom. Dešifrijte sledeći tekst, ako je korišćeni ključ: SISTEMIT (5,2,6,7,1,4,3,8).

rpi rmmropsfett dr.e,orcnnnoecii o nga l feoc sldt oauais roectrmnedpyhntpnp irwipcoacAnlm p.i infni yhpai iyrte fe f o r mmitcr onocrlo ramr ienele nl d tsnseThn eriho r igfonnra - hbw rlv tem rootatrecacsoorc y odsaeepetecenee d to .t,e hegtdoiensalsenaeoih pesvo pc

Zadatak 2 - predviđeno vreme trajanja 25 minuta

Implementirati funkciju za enkripciju i dekripciju teksta pomoću Cezrove šifre u PHP programskom jeziku.

✓ Poglavlje 9

Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ13

Šifrovanje korišćenjem Cryptool alata

Očekivano vreme izrade zadatka: 45 minuta.

Zadatak 1:

Koristeći Cezarovu šifru, šifrirajte svoje ime, prezime i broj indeksa. Kao ključ koristiti poslednju cifru u broju indeksa.

Zadatak 2:

Sledeća poruka je šifrovana Cezarovom šifrom:

DKD YT YTSCP DS CPYYTSCDHIPKCXYXW BTIDSP ZDSXGPCYP EDGJZP

Odgovorite na pitanja:

1. Koja je tajna poruka?
2. Koliko je mogućih ključeva koje treba probati da bi dešifrovali Cezarovu šifru?
3. Kako bi sproveli napad "brute force", šta je potrebno da znate? Drugim rečima, šta Vam je omogućilo da uradite "brute-force" napad u ovom slučaju?

OPIS DOMAĆEG ZADATKA

Zadatak 3

Zadatak 3:

Za sledeću poruku uradite analizu frekventnosti:

J Ixwcnwc vjwjpvnwc bhbcnv rb j bxocfjan jyyurlcrxw cqjc Ijw kn dbnm cx vjwjpn cqn lanjcrxw jwm vxmrorljcrxw xo mrprcju Ixwcnwc. LVBb jan chyrljuuh dbnm oxa nwcnayarbn Ixwcnwc vjwjpvnwc jwm fnk Ixwcnwc vjwjpvnwc. NLV chyrljuuh bdyyxacb vducryun dbnab rw j Ixuujkxajcren nweraxvvnwc kh rwcnpajcrwp mxldvnwc vjwjpvnwc, mrprcju jbbnc vjwjpvnwc jwm anlxam ancnwcrxw. Jucnawjcrenuh, FLV rb cqn Ixuujkxajcren jdcqxarwp oxa fnkbrcnb jwm vjh rwludmn cngc jwm nvknm pajyqrlb, yqxcxb, ermnx, jdmrx, vjyb jwm yaxpajvvn Ixmn cqjc mrbyujh Ixwcnwc jwm rwcnjlc frcq cqn dbna. NLV chyrljuuh rwludmnb j FLV odwlcrxw.

Za prethodno uneti tekst, odgovorite na sledeća pitanja:

Koji se ključ koristio?

Da li je šifriran tekst dešifrovan ispravno?

Kako glasi prva rečenica dešifrovanog teksta?

Objasnite kako je CrypTool mogao da uradi dekripciju na osnovu šifrovanog teksta na osnovu frekventnosti slova.

Potrebne datoteke snimiti pod imenom: **IT210-DZ13-Ime_Prezime_brojIndexa**

gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovane datoteke poslati na adresu predmetnog asistenta. (napomena: u Subject-u e-mejla napisati IT210 - DZ13)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovom predavanju su obrađeni sledeće ključne teme:

- Dat je pregled mogućih napada na mrežne i računarske resurse
- Dati su primeri zajedničkih mera za set različitih bezbednosnih domena
- Dati su primeri mera koje su specifične za jedan bezbednosni domen
- Dat je pregled zakonskih odredbi
- Prikazan je odnos digitalne istrage i drugih istraga

Literatura

[1] Harlan Carvey, Windows Forensics and Incident Recovery, Addison Wesley, 2004

[2] John R. Vacca, Computer Forensics: Computer Crime Scene Investigation, Charles River Media, 2002

[3] Hal Tipton and Micki Krause, Handbook of Information Security Management, CRC Press LLC, 1998

[4] Michael Miller, Apsolutna zaštita PC-ja i privatnosti, Komputer biblioteka, Čačak, 2003.

[5] Matt Bishop, Introduction to Computer Security, Prentice Hall PTR, 2004

[6] FBI, Calling All Business Professionals, What's the Current State of Computer Network Security?, http://www.fbi.gov/news/stories/2005/july/ccyber_072505

[7] National Institute of Standards and Technology, An Introduction to Computer Security: The NIST Handbook, Special Publication 800-12, 1995



IT210 - SISTEMI IT

**OSIGURANJE INFORMACIJA I
BEZBEDNOST - 2. Deo**

Lekcija 14

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 14

OSIGURANJE INFORMACIJA I BEZBEDNOST - 2. DEO

- ✓ OSIGURANJE INFORMACIJA I BEZBEDNOST - 2. Deo
- ✓ Poglavlje 1: Stanja informacija
- ✓ Poglavlje 2: Bezbednosne usluge
- ✓ Poglavlje 3: Korišćenje kriptografije u bezbednosti
- ✓ Poglavlje 4: Modeli sigurnosti i poverljivosti
- ✓ Poglavlje 5: Model za analizu opasnosti
- ✓ Poglavlje 6: Pokazne vežbe: PHP aplikacije primeri
- ✓ Poglavlje 7: Domaći zadatak
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Cilj ove lekcije je upoznavanje sa bezbednosnim uslugama i modelom za procenu rizika i troškova

U ovom predavanju biće reči o:

1. Stanju informacija

- Prenos
- Čuvanje
- Obrada

2. Bezbednosnim uslugama

- Raspoloživost Integritet
- Tajnost
- Autentifikacija
- Neprihvatanje nepriznavanja

3. Model za analizu opasnosti

- Procena rizika
- Troškovi

▼ Poglavlje 1

Stanja informacija

STANJA U KOJIMA SE INFORMACIJA MOŽE NAĆI

Postoje tri različita stanja u kojima se informacija može naći: obrada, prenos, čuvanje

Informacija se tokom svog životnog ciklusa može naći u različitim stanjima. Postoje tri različita stanja u kojima se informacija može naći. Informacija, ako postoji, mora biti najmanje u jednom od sledeća tri stanja:

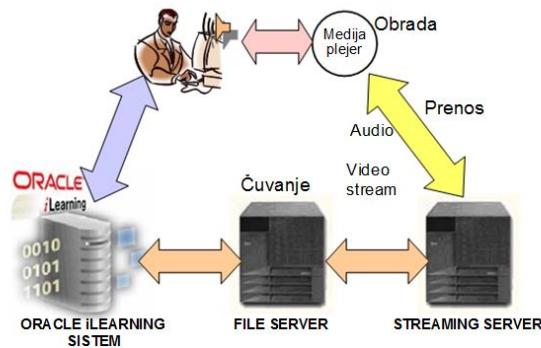
- Obrada
- Prenos
- Čuvanje

Međutim ista informacija može istovremeno biti i u dva ili tri različita stanja.

Razmotrimo primer jednog Word dokumenta. Prepostavimo da je on već kreiran i zapisan na disku. Informacije koje sadrži taj dokument su u stanju čuvanja. Isti dokument se može otvoriti ponovo da bi se recimo izvršila provera ispravnosti pisanja reči (engl. **spell check**). U trenutku kada taj proces započne, može se reći da su informacije u dva stanja: čuvanje i obrada. Zamislimo da smo istovremeno isti dokument poslali kao prilog elektronskom poštovom preko Interneta našem poslovnom partneru. Onda će, u istom trenutku dokument biti u sva tri stanja: čuvan na disku, obrađivan u Wordu i prenošen kroz mrežu.

Drugi primer koji pokazuje da informacija može u jednom trenutku biti u sva tri stanja je u jednom sistemu za učenje na daljinu. Posmatrajmo jedan video zapis koji predstavlja jednu lekciju. Ovaj zapis se čuva na fajl serveru. Ako se student odlučio da sluša to predavanje medija server će uzeti ovo predavanje i korišćenjem RTSP preneti studentu. Dobijeni podaci će biti obrađeni od strane medija plejera na studentovom računaru kako bi bili interpretirani na izlaznim uređajima.

Informacije su ranjive u sva tri stanja. Međutim, u različitim stanjima postoje različite pretnje po informacije. Takođe i ranjivost informacija zavisi od stanja u kome se nalaze informacije. Na poslednjem primeru će biti pokazana ranjivost informacija u pojedinim stanjima.



Slika 1.1 Primer stanja informacija u sistemu učenja na daljinu [Izvor: Autor]

ČUVANJE

Najveća ranjivost kada je informacija u stanju čuvanja potiče od neautorizovanog pristupa

Dok su informacije u stanju **čuvanja** izložene su različitim opasnostima, pa su zavisno od sigurnosti sistema, manje ili više ranjivi. Informacije se čuvaju u datotekama i bazama podataka. Za **čuvanje** se mogu koristiti različiti mediji, kao što su diskovi, diskete, trake, optički mediji, fleš memorije itd.

Najveća ranjivost potiče od neautorizovanog pristupa. Ukoliko se omogući neautorizovan pristup informacijama može doći do:

- Krađe informacija, odnosno nelegalno kopiranje informacija
- Brisanje informacija
- Izmena informacija, nemerna ili namerna, uključujući i kontaminaciju virusima
- Kompromitacija informacija, odnosno pristup poverljivim informacijama od strane neautorizovane osobe

Druga ranjivost informacija dok su u stanju **čuvanja** može da bude zbog kvara i fizičkog **oštećenja medija**. Ovo može da se dogodi zbog:

- prirodnih katastrofa kao što su poplave i zemljotresi
- požara
- izlaganja medija jakom elektromagnetskom značenju

PRENOS

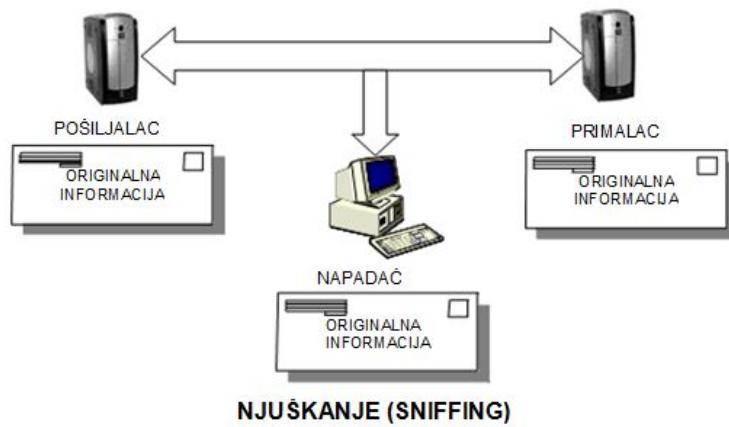
Najveća opasnost koja preti informacijama tokom transporta je njihova krađa ili kopiranje

Ranjivost informacija je možda najizraženija tokom prenosa podataka. Tokom **prenosa** informacije više nisu pod kontrolom vlasnika, i on ne može da utiče na njihovu sudbinu. **Najveća opasnost koja preti informacijama tokom transporta je njihova krađa ili**

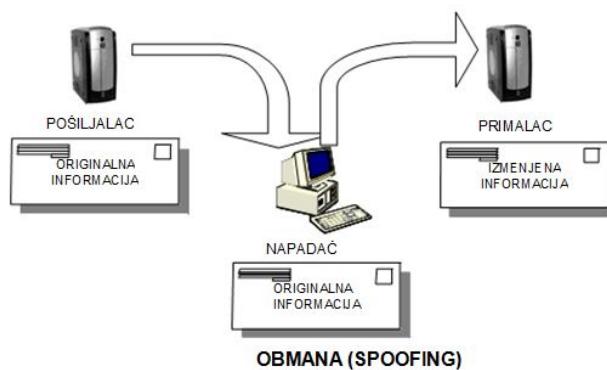
kopiranje. U slučaju krađe, informacije ne stignu na odredišnu stranu, već ih preuzme napadač. U slučaju kopiranja, informacija stigne do odredišta, ali je kopiju informacija preuzeo i napadač. Međutim, informacije tokom prenosa mogu biti i kontaminirane nekim virusom.

Razmotrimo primer slanja elektronske pošte. Informacije koje se šalju na ovaj način su podložne njuškanju (engl. **sniffing**). Napadač koristi neki softver za njuškanje preuzima informacije iz elektronske pošte i pokušava da u njima pronađe poverljive informacije kao što su korisnička imena i lozinke, brojevi kreditnih kartica, PIN-ovi ili važne informacije o organizaciji.

Ista elektronska pošta može biti podvrgнутa drugoj vrsti napada koja se naziva obmana (engl. **spoofing**). U ovom slučaju se informacije koje su u stadijumu prenosa preuzimaju sa namenom da se unutar njih ugrade maliciozne informacije ili kod. Onda se iste informacije prenose dalje, a odredišna strana veruje da je primila originalne informacije.



Slika 1.2 Njuškanje [Izvor: Autor]



Slika 1.3 Spoofing [Izvor: Autor]

OBRADA

Tipična opasnost koja se javlja prilikom obrade podataka dolazi od programa za špijuniranje

Informacije su ranjive i tokom obrade. Tipična opasnost koja se javlja prilikom obrade podataka dolazi od programa za špijuniranje (engl. **spyware**). Postoji čitav niz različitih programa za špijuniranje žrtve. Neki programi prate ulaz sa tastature kako bi ukrali identitet žrtve. Drugi programi prate koje sajtove posećuje žrtva kako bi otkrili oblast njegovog interesovanja i kasnije ga zasipali reklamnim porukama.

▼ Poglavlje 2

Bezbednosne usluge

PET VRSTA BEZBEDNOSNIH USLUGA MODELA OSIGURANJA INFORMACIJA

Pet vrsta bezbednosnih usluga su raspoloživost, integritet, tajnost, autentikacija i nepobitnost

U modelu osiguranja informacija definisano je pet vrsta bezbednosnih usluga:

- Raspoloživost
- Integritet
- Tajnost
- Autentikacija
- Nepobitnost

U daljem tekstu će biti više reči o svakoj od ovih usluga.

RASPOLOŽIVOST

Raspoloživost je bezbednosni servis koji garantuje blagovremeni i pouzdani pristup informacijama i informacionim uslugama autorizovanim osobama.

Raspoloživost je bezbednosni servis koji garantuje blagovremeni i pouzdani pristup informacijama i informacionim uslugama autorizovanim osobama. Ovo znači da smisao ovog servisa da informacioni resursi treba da budu dostupni autorizovanim osobama i u uslovima napada. Na žalost, u nekim slučajevima zahtevi za raspoloživost informacionih resursa se mogu ispuniti samo po cenu drugih bezbednosnih usluga. Posmatrajmo primer pojave novog virusa za koji posmatrani informacioni sistem nema odbranu. Ukoliko se želi potpuna bezbednost sistema, odnosno zaštita integriteta sistema, računare bi trebalo isključiti sa mreže dok se ne nađe adekvatna kontramera. U tom slučaju će usluga raspoloživosti biti u velikoj meri degradirana, jer su mnogi podaci i aplikacije na drugim serverima u mreži. Mrežni štampači i drugi zajednički hardver takođe ne bi bio raspoloživ. Bezbednosni inženjeri treba da izvrše procenu rizika i da donešu odluku o tome hoće li se i koliko degradirati ostale bezbednosne usluge, da bi se raspoloživost održala na zadovoljavajućem nivou.

Postoje različiti razlozi zbog kojih se usluga raspoloživosti u praksi degradira. Jedan od tipičnih razloga je napad uskraćivanja usluga (engl. *denial of service attack*) kada je neki od servisa

u mreži zatrpan porukama ili podacima tako da ne može da pruži usluge dobromamernim korisnicima.

Drugi primer degradiranja raspoloživosti može da bude napad kojim su korisnicima obrisani ili promjenjeni nalozi za pristup podacima ili aplikacijama. Sve dok administratori ne rekonstruišu staro stanje, korisnicima neće na raspolaganju biti informacioni resursi za koje je potrebna autentifikacija.

U nekim slučajevima se raspoloživost može da zavisi i od faktora koji striktno ne spadaju u domen bezbednosti. To su slučajevi u kojima poremećaji dolaze pod uticajem okoline informacionog sistema, pa nisu pod kontrolom bezbednosnih profesionalaca. Tipičan primer ovih poremećaja su: nestanak električne energije, kvar na uređajima za neprekidno napajanje, prekid saobraćaja na Internetu, požar, zemljotres itd.

Ipak, bezbednosni inženjeri treba da sačine kontra mere i za ovakve slučajnosti kojim će se minimizirati vreme tokom koga će informacioni resursi biti nerasploživi. Plan kontramera treba da sadrži mere za nastavak poslovanja, alternativne informacione resurse na drugoj lokaciji, hardversku redundantnost, diskove sa rezervnim kopijama podataka itd.

WEB SERVISI I RASPOLOŽIVOST

Raspoloživost zavisi od rada bezbednosnih profesionalaca ne samo u organizaciji korisnika, nego i od rada bezbednjaka kod provajdera Internet usluga i kod provajdera veb servisa

U slučaju korišćenja veb servisa **raspoloživost** se ne odnosi samo na raspoloživost informacija, nego i na raspoloživost Internet komunikacija i raspoloživost samog servisa. U ovom slučaju raspoloživost, dakle, zavisi od rada bezbednosnih profesionalaca ne samo u organizaciji korisnika, nego i od rada bezbednjaka kod provajdera Internet usluga i kod provajdera veb servisa. Napad uskraćivanja usluga koji se dogodio na serveru organizacije koja pruža neki veb servis ili napad na neki od rutera je sigurno van oblasti odgovornosti bezbednosnih profesionalaca organizacije korisnika. Instaliranjem alternativnih sajtova na geografski različitim pozicijama sa kojih se u slučaju napada mogu pružati isti veb servisi se može povećati raspoloživost.

INTEGRITET

Integritet se definiše kao kvalitet informacionog sistema koji odražava logičku korektnost i pouzdanost operativnog sistema, logičku kompletnost hardvera i softvera koji implementira zaštitu

U užem smislu, kada se posmatraju samo podaci, **integritet** se definiše kao zaštita protiv neautorizovanog menjanja i uništavanja podataka. Integritet podataka je na određeni način stvar stepena poverenja.

U širem smislu, kada se posmatra ceo informacioni sistem, integritet se definiše kao kvalitet informacionog sistema koji odražava logičku korektnost i pouzdanost operativnog sistema, logičku kompletност hardvera i softvera koji implementira zaštitni mehanizam i konzistentnost struktura podataka i stanje čuvanih podataka.

Integritet može da bude kompromitovan zlonamernim delovanjem neautorizovanih osoba, nesvesnim delovanjem autorizovanih osoba ili dejstvom virusa i trojanaca. Osnovna tri principa koja se koriste za uspostavljanje kontrole integriteta su:

- **Dodeljivanje pristupa na bazi „potrebno-da-zna“** (engl. **need-to-know**). Korisnicima treba da bude dodeljen pristup samo onim informacionim resursima koji su im potrebni da vršenje dodeljenih poslova. Korisnicima kojima je odobren unos i promena podataka na raspolaganju su programi koji kontrolisu podatke i održavaju njihov integritet. Svi podaci o transakcijama i izmenama podataka se upisuju u log datoteke tako da se kasnije, ukoliko je došlo do narušavanja integriteta, može izvršiti istraga. Podacima se može pristupati samo dozvoljenim setom programa koji su testirani, dobro instalirani i koje je zabranjeno menjati od strane neautorizovanih osoba.
- **Odvajanjem zaduženja.** Da bi se obezbedilo da samo jedna osoba nema kontrolu nad transakcijom od početka do kraja, odvajaju se zaduženja za njeno izvršavanje. Na primer, jednoj osobi se dozvoljava da kreira i sertificuje transakciju, a druga osoba je izvršava. Na taj način transakcija ne može da bude izvršena radi lične dobiti neke osobe.
- **Rotacijom zaduženja.** Osobama koje imaju osetljiva zaduženja treba povremeno menjati zaduženja kako se više osoba ne bi povezalo u cilju preuzimanja potpune kontrole nad transakcijama sa nepoštenim namerama.

MODELI INTEGRITETA

Modeli integriteta se koriste za opis potrebnih radnji koje je potrebno preduzeti da bi se osigurao integritet informacija

Modeli integriteta se koriste za opis potrebnih radnji koje je potrebno preduzeti da bi se osigurao integritet informacija. Postoje tri cilja integriteta, koje pojedini modeli ispunjavaju na različite načine:

- Sprečavanje neautorizovanih osoba da menjaju podatke ili programe
- Sprečavanje autorizovanih osoba da vrše neispravne ili neautorizovane promene
- Održavanjem interne i eksterne konsistencije podataka i programa.

U praksi se koristi nekoliko modela za očuvanje integriteta informacija prilikom čuvanja i obrade, a najčešći su:

1. Biba,
2. Goguen-Meseguer,
3. Sutherland,
4. Clark-Wilson,
5. Brewer-Nash.

TAJNOST

Tajnost je bezbednosna usluga koja obezbeđuje da informacije ne budu obelodanjene ili prikazane neautorizovanim osobama, procesu ili uređaju

Tajnost je bezbednosna usluga koja obezbeđuje da informacije ne budu obelodanjene ili prikazane neautorizovanim osobama, procesu ili uređaju. Ova usluga nije samo važna za zaštitu informacija u vojnim i državnim institucijama. Iz razloga bezbednosti, ona je podjednako važna u bilo kom informacionom sistemu. Podjednako je važno zaštititi tajnost zdravstvenih podataka pacijenata, podataka o poslovanju neke kompanije ili podataka o korisničkim imenima, lozinkama, javnim i privatnim ključevima. Pitanje privatnosti je poslednjih godina postalo aktuelno baš zbog toga što je tajnost kao bezbednosna usluga u mnogim slučajevima bila degradirana. Tako su lični podaci o osobama bili zloupotrebljavani pre svega u marketinške svrhe.

Krucijalni aspekt tajnosti je autentikacija i autorizacija svih korisnika sistema koji sadrže informacije sa određenim stepenom tajnosti.

U najveće pretnje zaštitu tajnosti podataka spadaju hakeri, prenos nešifrovanih podataka preko interneta, programi za špijuniranje i *trojanci*.

Da bi se opisale akcije koje treba preduzeti u cilju osiguranja tajnosti podataka prave se modeli tajnosti. Oni opisuju kako treba koristiti bezbednosne alate da bi se postigao određeni stepen zaštite tajnosti podataka.

AUTENTIFIKACIJA I NEPOBITNOST

Nepobitnost je bezbednosna usluga kojom se pošiljaocu podataka obezbeđuje dokaz da su podaci isporučeni

Autentifikacija

Autentikacija je bezbednosna usluga koja obezbeđuje validnost prenosa, poruka ili pošiljaoca i obezbeđuje mehanizam kojim se verifikuje da li je osoba autorizovana da koristi određene informacione resurse.

Nepobitnost

Nepobitnost ili neprihvatanje nepriznavanja (engl. *non-repudiation*) je bezbednosna usluga kojom se pošiljaocu podataka obezbeđuje dokaz da su podaci isporučeni, a primaocu se obezbeđuje dokaz o identitetu pošiljaoca, tako da kasnije nijedan ne može da pobije da je imao te podatke.

Usluga nepobitnosti je posebno značajna u elektronskoj trgovini i elektronskom bankarstvu, kada je nepobitno potrebno dokazati da su pošiljalac i primalac izvršili neke radnje predviđene poslovним procesom ili ugovorom. Jedan od metoda koji se koristi za obezbeđivanje nepobitnosti je digitalni potpis.

▼ Poglavlje 3

Korišćenje kriptografije u bezbednosti

PRIMENA KRIPTOGRAFIJE

Da bi se osigurao integritet podataka prilikom prenosa mogu se koristiti tajni i javni ključevi

Mnogo veći problem predstavlja očuvanje integriteta informacija prilikom prenosa. Zlonamerni napadač može poruku, na primer poslovnu ponudu, prepraviti tokom prenosa Internetom tako da ona ne bude povoljna za kupca. Dovoljno je cenu prepraviti sa 1.000 na 10.000, pa da pošiljalac poruke ima izgubljen posao. Zbog toga se ovakve poruke pre slanja šifriraju. Kriptografija može da pomogne da se detektuje da li je poruka modifikovana. Ali, šifriranje poruka ne znači da je poruka zaštićena od menjanja. Da bi se osigurao integritet podataka prilikom prenosa mogu se koristiti tajni i javni ključevi. Mada su noviji metodi sa javnim ključevima mnogo fleksibilniji, verifikacija integriteta korišćenjem tajnih ključeva je ugrađena u mnoge aplikacije.

Kada se koristi kriptografija bazirana na tajnom ključu, izračunava se *autentikacioni kod poruke* (engl. *message authentication code - MAC*). Ovaj autentikacioni kod se dodaje podacima koji se prenose. Kada se na prijemnoj strani poruka primi vrši se ponovno izračunavanje autentikacionog koda poruke i upoređuje se sa originalnim. Na taj način se može zaključiti da li je poruka menjana.

Kada se koristi kriptografija bazirana na javnom ključu, verifikacija integriteta poruke se vrši potpisivanjem poruke javnim ključem i korišćenjem heš funkcija za šifriranje.

Korišćenjem heš algoritma se kreira *sažetak poruke* (engl. *message digest*) koja se naziva i heš. Sažetak poruke se zatim potpisuje privatnim ključem čime se dobija digitalni potpis. Na prijemnoj strani se ponovo izračunava sažetak poruke korišćenjem javnog ključa pošiljaoca i upoređuje sa sažetkom koji se dobija izračunavanjem iz dešifrovane poruke. Ako se oba sažetka poklapaju, poruka nije menjana. Na taj način, odgovarajućim javnim ključem se praktično verifikuje integritet poruke. Istovremeno se verifikuje i autentičnost pošiljaoca jer javni ključ može da dešifruje samo one poruke koje odgovaraju privatnom ključu pošiljaoca.

▼ Poglavlje 4

Modeli sigurnosti i poverljivosti

MODELI SIGURNOSTI INFORMACIJA

Modeli sigurnosti informacija se mogu zasnivati na kontroli pristupa, integritetu i toku informacija

1. Modeli kontrole pristupa (engl. **access control models**)

- Bell LaPadula model,
- Model matrice pristupa (engl. **access matrix**)
- Model preuzmi dodeli (engl. **take-grant model**)

2. Modeli integriteta, tj. celovitosti (engl. **integrity models**)

- Biba model integriteta,
- Clark Wilson model integriteta

3. Modeli toka informacija (engl. **information flow models**)

- Model bez preplitanja (engl. **non-interference model**),
- Teorije kompozicije (engl. **composition theories**)

BELL-LAPADULA MODEL POVERLJIVOSTI

Svakom objektu je dodeljen nivo tajnovitosti, a svakom subjektu nivo pristupa. Relacije su opisane dodeljenim nivoom pristupa i privilegija subjektu i nivou tajnovitosti objekta

Da bi se opisale akcije koje treba preduzeti u cilju osiguranja tajnosti podataka prave se modeli tajnosti. Oni opisuju kako treba koristiti bezbednosne alate da bi se postigao određeni stepen zaštite tajnosti podataka.

Najčešće korišćeni model za opis zaštite tajnovitosti je **Bell-LaPadula** model. On definije relacije između informacionih objekata, kao što su datoteke, zapisi, programi i oprema koja sadrži ili prenosi informacije, i subjekata, kao što su osobe, procesi ili uređaji koji izazivaju prenos informacija između objekata. Svakom objektu je dodeljen nivo tajnovitosti, a svakom subjektu nivo pristupa. Relacije su opisane dodeljenim nivoom pristupa i privilegija subjektu i nivou tajnovitosti objekta.

Subjekti pristupaju objektima da bi čitali, pisali ili čitali i pisali informacije.

Bell-LaPadula model radi na principu, koji specificira da je subjektima dozvoljen:

- pristup pisanju objekata koji su na istom ili višem nivou od subjekta
- pristup čitanju objekta na istom ili nižem nivou
- pristup čitanju i pisanju samo onih objekata koji su istog nivoa kao subjekt.

Ovakav pristup onemogućuje da se informacije višeg nivoa tajnovitosti upišu u objekte nižeg nivoa tajnovitosti. Takođe, informacije višeg nivoa tajnovitosti nisu dostupne subjektima nižeg nivoa pristupa. Bell-LaPadula model garantuje samo zaštitu tajnosti informacija, a ne i integritet. Na primer, osobi nižeg nivoa pristupa je dozvoljeno da piše informacije višeg nivoa tajnovitosti.

Drugi tip modela koji se često koristi je model upravljanja pristupom, koji organizuje sistem u:

- *objekte* - resurse na koje se deluje
- *subjekte* - osobe ili programi
- *operacije* - procesi interakcije

Set pravila specificira koje operacije mogu da budu izvedene nad nekim objektom od strane nekog subjekta. Ovaj model istovremeno garantuje i tajnovitost i integritet informacija.

BIBA MODEL INTEGRITETA

Biba model integriteta je analogan Bell-LaPadula modelu poverljivosti

Biba model integriteta je objavljen 1977. godine od strane MITRE korporacije, godinu dana nakon što je objavljen Bell-LaPadula model (Koen). Bell-LaPadula model garantuje tajnost podataka, ali ne i njen integritet. Kao rezultat, Biba je stvorio model za rešavanje potreba za sprovođenje integriteta u računarskom sistemu. Biba model integriteta je analogan Bell-LaPadula modelu poverljivosti. Deo koji je analogan se prevashodno odnosi na sličnost klasifikacija različitih nivoa osetljivosti, koje imamo u Bell-LaPadula modelu, s tim što Biba model klasificiše objekte u različite nivoje integriteta. Biba model integriteta definiše tri aksioma:

1. Jednostavna aksioma integriteta (engl. **simple integrity axiom**) - podrazumeva da subjektu na jednom nivou integriteta nije dozvoljeno da vidi (čita) objekat nižeg integriteta (nema čitanja nadole – engl. **no read down**).
2. Zvezda (*) aksiom integriteta (engl. **star integrity axiom**) - podrazumeva da objektu na jednom nivou integriteta nije dozvoljeno da izmeni tj. modifikuje objekat višeg nivoa integriteta niti da upisuje u njega (engl. **no write up**)
3. Subjekat na jednom nivou integriteta ne može pozivati (engl. **invoke**) subjekat na višem nivou integriteta.

Biba model se sastoji od grupe pristupnih režima. Pristupni režimi su slični onima koji se koriste u drugim modelima, mada treba naglasiti da se ne koristi svuda ista terminologija za njihov opis.

Pristupni režimi koje podržava Biba su sledeći:

1. **Modifikovanje** (engl. **Modify**): omogućava da subjekt napiše objekat. Ovaj režim je sličan "write" režimu u ostalim modelima

Režim u ostalim modelima.

2. *Posmatranje* (engl. *Observe*): omogućava da subjekt čita objekat. Ova komanda je sinonim za "read" komandu u drugim modelima.
3. *Pozivanje* (engl. *Invoke*): omogućava da subjekt komunicira sa drugim subjektom.
4. *Izvršavanje* (engl. *Execute*): omogućava da subjekt izvrši objekat. Ova komanda u stvari omogućava subjektu da izvrši program, koji predstavlja objekat.

UPOREĐENJE BELL-LAPADULA I BIBA MODELA - VIDEO

Security Models Pt 1 Bell-La Padula and Biba (CISSP Free by Skillset.com)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 5

Model za analizu opasnosti

PROCENA RIZIKA

Procena rizika je postupak kojim se identifikuju potencijalne opasnosti za kompromitovanje zaštićenog informacionog sistema

Procena rizika je postupak kojim se identifikuju potencijalne opasnosti za kompromitovanje zaštićenog informacionog sistema. Na najosnovnijem nivou procena rizika daje verovatnoću narušavanja sigurnosti sistema. Narušavanje sigurnosti može biti rezultat unutrašnjih ili spoljašnjih napada ili potencijalne ranjivosti odbrane sistema.

Procena rizika se vrši, a njeni rezultati koriste, za definisanje bezbednosnih mera koje će biti preduzete u cilju održavanja sigurnosti sistema. Kako su mere bezbednosti skopčane sa troškovima, procena rizika pomaže da se definiše efektivni bezbednosni sistem.

Rezultati analize rizika mogu da se predstave kvantitativno ili kvalitativno. Kvantitativna mera rizika može biti izražena finansijskim gubicima koji će nastati zbog narušavanja bezbednosti. Oni mogu biti izraženi kao očekivani gubici na godišnjem nivou ili kao gubici za neki konkretno očekivani napad.

Kvalitativna procena rizika je deskriptivna. Ona se izražava terminima kao što su nizak, srednji ili visoki rizik.

Tokom procesa procene rizika potrebno je odgovoriti na sledeća pitanja:

- Koje su pretnje bezbednosti sistema?
- Koje su posledice pojedinih slučajeva narušavanja bezbednosti?
- Koliko često bi moglo da se dogode pojedini slučajevi narušavanja bezbednosti tokom nekog perioda vremena, a uobičajeno tokom jedne godine?
- Koliko su pouzdani odgovori na prethodna pitanja?

Odgovori na gornja pitanja su osnova za analizu rizika. Na osnovu analize rukovodstvo organizacije i bezbednosni profesionalci treba da odgovore na još tri pitanja:

- Kako i kojim bezbednosnim merama smanjiti rizik
- Koliki su troškovi bezbednosnih mera na godišnjem nivou
- Da li su bezbednosne mere efektivne sa aspekta troškova, što se može zaključiti analizom troškovi-dobit.

Prikazani proces daje odgovore koji zavise od mnogo faktora kao što su vrste pretnji, trenutno stanje informacionog sistema u organizaciji, obim i vrsta poslovanja. Svi ovi faktori su

dinamičke prirode i menjaju se tokom vremena. Ovo navodi na zaključak da procenu rizika treba vršiti stalno, a mere za smanjenje rizika prilagođavati trenutnoj situaciji.

NIST-OVA METODOLOGIJA ZA PROCENU RIZIKA

NIST je propisao proceduru za procenu rizika koje se sastoji iz 9 koraka

Zbog velike važnosti i složenosti zadatka NIST (engl. National Institute of Standards and Technology) je propisao proceduru za procenu rizika. Ova procedura se sastoji od 9 koraka, koji će nadalje biti objašnjeni.

- Određivanje domena za koji se procenjuje rizik i metodologije. U prvom koraku se određuje da li će se vršiti procena rizika za ceo informacioni sistem ili samo za deo sistema. Određuje se metodologija koja će se primeniti i nivo detaljnosti.
- Skupljanje i analiziranje podataka. Tokom analize je potrebno ispitati različite komponente rizika. Ispitivanje uključuje prikupljanje podataka o delovima sistema koji su izloženi pretnjama, sintezu i analizu informacija. Ovo podrazumeva sledeće korake:
- Određivanje vrednosti imovine. Ovo uključuje informacije, softver, osoblje, hardver i prostor. Vrednost neke imovine se sastoji od njene stvarne (tržišne) vrednosti i kratkoročnih i dugoročnih troškova koji će nastati kao posledica njenog kompromitovanja.
- Procena posledica. Procena posledica podrazumeva određivanje gubitaka koji bi mogli da nastanu.
- Identifikacija pretnji. Pretnja je entitet ili događaj koji ima mogućnost da ošteti sistem. Tipične pretnje su: greške, obmana, nezadovoljni radnik, požar, poplava, kreker ili virus. Pretnje treba identifikovati i analizirati da bi se odredila verovatnoća njihovog događanja i njihovih potencijala da oštete sistem.
- Analiza bezbednosti. Analiza bezbednosti treba da uključi ispitivanje efektivnosti postojećeg bezbednosnog sistema.
- Analiza ranjivosti. Ranjivost je stanje, nepostojanje ili slabost procedura, tehničkog nadzora, fizičkog nadzora ili drugog nadzora koje može biti iskorišćeno od strane pretnje.
- Procena verovatnoće. Verovatnoća je ocena frekvencije ili slučajnosti da će se pretnja ostvariti. Procena verovatnoće razmatra prisutnost, ukorenjenost i snagu pretnje, kao i efektivnost bezbednosnih mera u odnosu na datu pretnju (ukoliko postoji).
- Interpretacija rezultata ocene rizika. Procedura ocene rizika treba da proizvede jasne smernice o tome šta je stvarno važno za organizaciju u pogledu bezbednosti kako bi se prihvatio rizik i izabrale bezbednosne mere čiji su troškovi opravdani.

TROŠKOVI

Troškovi se mogu odnositi na troškove za upravljanje bezbednošću i za radni nadzor

Planiranje, uvođenje i održavanje bezbednosnih mera iziskuje ljudske i materijalne resurse. Oni su vezani za pojedine bezbednosne aktivnosti, pa će se i ovde tako tretirati. Ukupni troškovi potrebni za osiguranje bezbednosti sistema se mogu podeliti na:

- Troškove upravljanja bezbednošću
- Troškove radnog nadzora

UPRAVLJANJE BEZBEDNOŠĆU

Upravljanje bezbednošću jedne organizacije podrazumeva definisanje politike bezbednosti, upravljanje programom bezbednosti, upravljanje bezbednosnim rizikom, obezbeđenje sigurnosti i

Upravljanje bezbednošću jedne organizacije podrazumeva definisanje politike bezbednosti, upravljanje programom bezbednosti, upravljanje bezbednosnim rizikom, obezbeđenje sigurnosti i planiranje tokom životnog ciklusa računara i sigurnost. Svaka od ovih aktivnosti stvara troškove organizaciji.

- **Politika bezbednosti.** Troškovi vezani za politiku bezbednosti se sastoje od troškova za razvoj politike bezbednosti i troškova za implementaciju politike unutar organizacije. Razvoj politike bezbednosti zahteva angažovanje ne samo bezbednosnih profesionalaca, nego i veliki broj rukovodilaca organizacije kako bi se predložila, usvojila, publikovala politika bezbednosti. U nekim slučajevima, implementiranje politike bezbednosti utiče na poslovne procese, pa se indirektno mogu javiti i zavisni troškovi.
- **Upravljanje programom bezbednosti.** Ovde je najveći deo troškova vezan za plate osoblja koje se bavi upravljanjem bezbednošću u organizaciji.
- **Upravljanje bezbednosnim rizikom.** Analiza bezbednosti sistema je kontinualan zadatak koji zahteva angažovanje eksperata i softvera za analizu bezbednosti, pa su i troškovi vezani za plate i potrebne softverske alate.
- **Sigurnost tokom životnog ciklusa računara.** Životni ciklus jednog računara počinje od njegove nabavke pa do njegovog odlaganja. Od trenutka kada se novi računar uvodi u organizaciju potrebno je planirati i sprovoditi niz mera vezan za taj računar, softver koji je instaliran na njemu i podatke koji se u njemu čuvaju. Svaki računar u organizaciji je na neki način specifičan i zahteva posebne mere. Najvažniji trošak u ovoj kategoriji su plate osoblja.
- **Sigurnost.** Bezbednosna sigurnost je stepen poverenja korisnika da bezbednosne mere rade ispravno i štite sistem kao što je planirano. Da bi se postigla i održala sigurnost potrebne su stalne provere i testiranje bezbednosnog sistema. Troškovi ove kategorije su vezani za plate osoblja i softverske alate za testiranje sigurnosti.

RADNI NADZOR

Radni nadzor u jednom bezbednosnom sistemu obuhvata čitav niz bezbednosnih aktivnosti koje se svakodnevno vrše

Radni nadzor u jednom bezbednosnom sistemu obuhvata čitav niz bezbednosnih aktivnosti koje se svakodnevno vrše. Troškovi vezani za radni nadzor se mogu klasifikovati na sledeći način:

- **Administriranje korisnika.** Jedan od zadataka administratora sistema je da registruju korisnike, dodeljuju im prava pristupa informacionim resursima i nadgledaju njihov rad. Korisnici sistema su najčešće iz iste organizacije, ali u nekim sistemima korisnici mogu biti anonimni ili iz partnerske organizacije. Troškovi vezani za ovu aktivnost se odnose na inicijalno i povremeno testiranje moralnih i etičkih karakteristika korisnika i administratora, obuku, identifikaciju i autentikaciju korisnika, kao i softver potreban za nadgledanje korisnika.
- **Pripreme za otkaze i nesreće.** Jedna od obaveza bezbednosnog sistema je da obezbedi da informacioni sistem nastavi da radi i u slučaju otkaza različitih sistema ili nesreća kao što su požari ili poplave. Za ovakve slučajevе se razvija poseban plan dejstava kako bi se obezbedilo potpuno funkcionisanje IS ili bar onih njegovih delova koji su krucijalni za rad organizacije. Troškovi vezani za ovu aktivnost mogu biti znatni pogotovu ako je predviđeno instaliranje i održavanje rezervnih sistema na udaljenim geografskim lokacijama, rezervno napajanje, rezervne veze ka Internetu itd. I testiranje za rad u kritičnim situacijama znatan trošak.
- **Obrada incidenata.** Tokom rada informacionog sistema događaju se različiti incidenti, od prostih kao što je nenamerno brisanje datoteke, do složenih kao što je upad virusa u LAN. Svaki incident zahteva angažovanje kako bi se informacioni sistem doveo u stanje pre napada. Troškovi vezani za ovu aktivnost potiču od plata osoblja, softverskih alata i troškova za edukaciju osoblja.
- **Edukacija korisnika.** Informacioni sistem ni uz najbolje mere neće biti bezbedan ako korisnici nisu svesni pretnji i ranjivosti sistema i ako nisu trenirani i obučeni da ispravno koriste sistem. Ova aktivnost zahteva materijalna sredstva za nabavku literature, predavače i putne troškove.
- **Podrška i održavanje.** Određena grupa poslova vezana za rad sistema ne spada striktno u bezbednosne aktivnosti ali se delom ipak odnosi i utiče na bezbednost. U ove aktivnosti spadaju podrška korisnicima, upravljanje konfiguracijama, izrada rezervnih kopija, upravljanje medijima i dokumentacijom i održavanje. Za ovu aktivnost su vezani različiti troškovi kao što su troškovi osoblja, memorijskih medija, obuka i slično.

Fizička sigurnost i okruženje . Fizička sigurnost se odnosi na obezbeđenje sistema, zgrade i odgovarajuće infrastrukture. Fizička sigurnost se stara o pretnjama kao što su: pristup neautorizovanim osobama u prostorije u kojima su smešteni resursi informacionog sistema, požar, rušenje zgrade, prskanje vodovodnih cevi itd. Troškovi su vezani za nabavku i održavanje sistema za detekciju požara, video nadzor, kontrolu pristupa prostorijama, kao i osoblje.

PROCENA RIZIKA - VIDEO

Cyber security Risk Assessment [A step by step method to perform cybersecurity risk assessment]

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

✓ Poglavlje 6

Pokazne vežbe: PHP aplikacije primeri

PRIMER APLIKACIJE: TURISTIČKE PONUDE

Potrebno je kreirati sajt za praćenje ponuda turističkih agencija

Očekivano vreme izrade zadatka: 50 minuta.

Potrebno je kreirati sajt za praćenje ponuda turističkih agencija upotrebom HTML-a, CSS-a, PHP-a i baze podataka.

U MySQL ili MS Access je potrebno kreirati bazu podataka pod nazivom "agencija".

U okviru baze agencija kreirane su tabele prikazane na slici 1.

The screenshot shows two tables in MySQL Workbench:

turistica korisnik	
korisnik_id	int(11)
username	text
password	text
ime	text
prezime	text
email	text
admin	int(11)

turistica putovanje	
putovanje_id	int(11)
odrediste	text
opis	text
cena	int(11)
slika	text

Slika 6.1.1 Prikaz tabela unutar baze agencija [Izvor: Autor]

U PHP-u prvo kreiramo fajl konekcija.php za konekciju na kreiranu bazu.

```
<?php
    function povezi(){
        try {
            $con = new PDO("mysql:host=localhost;dbname=agencija", "root", "");
            $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $con;
        } catch(PDOException $e){
            echo "Error: " . $e->getMessage();
        }
    }
?>
```

Sajt treba da ima navigacioni bar koji treba da bude konzistentan na svim stranama pa se zato kreira header.php fajl koji će kasnije biti pozivan u ostalim fajlovima pomoću require funkcije. Na početku ovog fajla nalazi se skripta koja uzima podatke iz sesije i proverava da li je korisnik admin ili ne.

```
<?php
session_start();
require_once('konekcija.php');
$con = povezi();
$stmt = $con->prepare('SELECT * FROM korisnik WHERE username=:username');
$stmt->bindParam(":username", $_SESSION['username']);

$stmt->execute();
$USER = $stmt->fetch();
$is_mod = (intval($USER["admin"]) == 1 ? true : false);
?>
<!DOCTYPE html>
<html>
<head>
    <title>Turisticka agencija</title>
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <link rel="icon" href="pic/logo.png">
</head>
<body>
<header>
    <nav>
        <ul>
            <li><a href="index.php">Početna</a></li>
            <li><a href="ponude.php">Ponude</a></li>
            <li><a href="kontakt.php">Kontakt</a></li>
            <?php echo ($is_mod ? '<li><a href="dodaj.php">Dodaj novu ponudu</a></li>' :
: '') ;?>
            <?php
                if($USER["ime"] != ""){
                    echo "<li><a href='logout.php'>Odjavi se - " . $USER['ime'] .
                "</a></li>";
                }
                else{
                    echo "<li><a href='loginstrana.php'>Login</a></li>";
                }
            ?>
        </ul>
    </nav>
</header>
```

POČETNA STRANA

Na početnoj strani se nalazi slideshow nekih ponuda

Na početnoj strani nalazi se slideshow nekih destinacija koje mogu da se smenjuju. Prikaz početne strane prikazan je na slici 2.



Slika 6.1.2 Početna strana [Izvor: Autor]

U nastavku se nalazi kod za početnu stranu (index.php).

```
<?php
require_once ("header.php");
?>
<div class="slideshow-container">
    <div class="mySlides fade">
        <div class="numbertext">1 / 3</div>
        
        <div class="text">Marko</div>
    </div>
    <div class="mySlides fade">
        <div class="numbertext">2 / 3</div>
        
        <div class="text">Aranzman Indija i Nepal</div>
    </div>
    <div class="mySlides fade">
        <div class="numbertext">3 / 3</div>
        
        <div class="text">Las Vegas</div>
    </div>
    <a class="prev" onclick="plusSlides(-1)">#10094;</a>
    <a class="next" onclick="plusSlides(1)">#10095;</a>
</div>
<br/>
<div style="text-align:center">
    <span class="dot" onclick="currentSlide(1)"></span>
    <span class="dot" onclick="currentSlide(2)"></span>
    <span class="dot" onclick="currentSlide(3)"></span>
</div>
<script>
    var slideIndex = 1;
    showSlides(slideIndex);

    function plusSlides(n) {
        showSlides(slideIndex += n);
    }
</script>
```

```
function currentSlide(n) {
    showSlides(slideIndex = n);
}

function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("dot");
    if (n > slides.length) {slideIndex = 1}
    if (n < 1) {slideIndex = slides.length}
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
}
</script>
</body>
</html>
```

LOGIN I REGISTER STRANA

U nastavku su kreirane login i register strana i php skripte koje to realizuju

Na strani login (loginstrana.php) se nalazi forma sa poljima za username i password.

```
<?php
require_once ("header.php");
?>
<div class="container">
    <center>
        <form action="login.php" method="POST">
            <h2>Ulogujte se</h2>
            <div class="col-50">
                <input type="text" name="username" placeholder="Username"/>
            </div>
            <div class="col-50">
                <input type="password" name="password" placeholder="Password"/>
            </div>
            <input type="submit" value="Login"/>
            <br/>
            <a href="registerstrana.php">Nemate nalog? Registrujte se</a>
        </form>
    </center>
</div>
```

```
</body>  
</html>
```

Ova forma poziva login.php skriptu koja uzima podatke iz forme proverava ih sa podacima u bazi i ukoliko se poklapaju otvara sesiju.

```
<?php  
    require_once('konekcija.php');  
    $con = povezi();  
    $username = $_POST["username"];  
    $password = $_POST["password"];  
    $password = md5($password);  
  
    if(isset($_POST['username']) && isset($_POST['password'])){  
        $stmt = $con->prepare("SELECT * FROM korisnik WHERE username=:username AND password=:password");  
        $stmt->bindParam(":username", $username);  
        $stmt->bindParam(":password", $password);  
        $stmt->execute();  
  
        $row = $stmt->fetch();  
        if($row){  
            session_start();  
            $_SESSION["username"] = $row["username"];  
            $_SESSION["ime"] = $row["ime"];  
            $_SESSION["email"] = $row["email"];  
            header('Location: index.php');  
        }  
        else {  
            header('Location: loginstrana.php');  
        }  
    }  
?>
```

Na strani register (registerstrana.php) se nalazi forma sa poljima za email, ime, prezime, username i password korisnika.

```
<?php  
require_once ("header.php");  
?>  
<div class="container">  
    <center>  
        <form action="registracija.php" method="POST">  
            <h2>Registrujte se</h2>  
            <div class="col-50">  
                <input type="text" name="email" placeholder="E-mail*"/>  
            </div>  
            <div class="col-50">  
                <input type="text" name="ime" placeholder="Ime*"/>  
            </div>  
            <div class="col-50">  
                <input type="text" name="prezime" placeholder="Prezime*"/>  
            </div>
```

```
</div>
<div class="col-50">
    <input type="text" name="username" placeholder="Username*" />
</div>
<div class="col-50">
    <input type="password" name="password" placeholder="Password*" />
</div>
<input type="submit" value="Register"/>
<br/>
<br/>
<a href="loginstrana.php">Već posedujete nalog? Idite na login</a>
</form>
</center>
</div>
</body>
</html>
```

Ova forma poziva register.php skriptu koja uzima podatke iz forme i unosi ih u tabelu korisnik. Vrednost kolone admin mora biti po default-u 0 jer bi u suprotnom bilo koji korisnik bio administrator.

```
<?php
    require_once('konekcija.php');
    $con = povezi();

    if(isset($_POST['ime']) && isset($_POST['prezime']) &&
    isset($_POST['username']) && isset($_POST['password']) && isset($_POST['email'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $password = md5($password);
        $ime = $_POST['ime'];
        $prezime = $_POST['prezime'];
        $email = $_POST['email'];

        $stmt = $con->prepare("INSERT INTO korisnik (username, password, ime,
prezime, email, admin) VALUES (:username, :password, :ime, :prezime, :email, 0)");
        $stmt->bindParam(":username", $username);
        $stmt->bindParam(":password", $password);
        $stmt->bindParam(":ime", $ime);
        $stmt->bindParam(":prezime", $prezime);
        $stmt->bindParam(":email", $email);

        $stmt->execute();
        header('Location: loginstrana.php');
    }
?>
```

Pored login potrebno je kreirati i logout.php kako bi korisnik mogao da se izloguje i na taj način uništi sesija.

```
<?php
    session_start();
```

```
session_destroy();
header("Location:index.php");
?>
```

PRIKAZ LOGIN I REGISTER STRANE

Na slikama su prikazane strane login i register

Na slici 3 je prikaz login strane, a na slici 4 forma na strani register.



Slika 6.1.3 Strana login [Izvor: Autor]



Slika 6.1.4 Strana register [Izvor: Autor]

STRANA SA PONUDAMA

Strana sa ponudama treba da sadrži listu putovanja u ponudi

Na ovoj strani je potrebno uzeti podatke iz baze i prikazati ih na stranici. Kako bi to postigli konektujemo se na bazu, postavljamo upit za podatke iz tabele putovanje i potom te podatke pomoću while petlje prikazujemo na strani.

```
<?php
require_once("header.php");
require_once('konekcija.php');
$con = povezi();

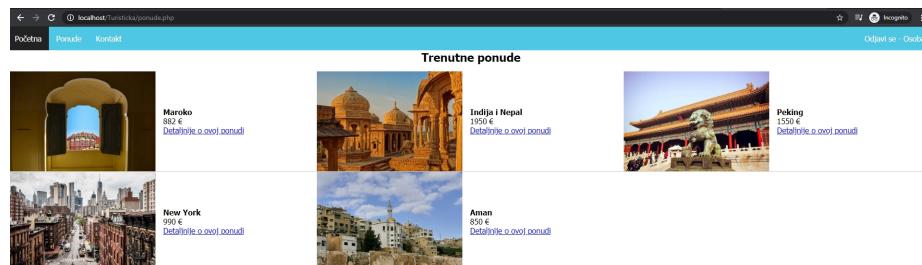
$sql = "SELECT * FROM putovanje";
$result = $con->query($sql);
```

```

echo "<center><h2 style='margin-bottom: 15px'>Trenutne ponude</h2></center>";
while ($row = $result->fetch()) {
    echo "<div id='pattern' class='pattern'>
        <ul class='list img-list'>
            <li>
                <div class='li-img'>
                    <img src='".$row["slika"]."' alt='".$row["odrediste"]."'/>
                </div>
                <div class='li-text'>
                    <h4 class='li-head'>" . $row["odrediste"] . "</h4>
                    <p class='li-sub'>" . $row["cena"] . " €</p>
                    <a href='#'>Detaljnije o ovoj ponudi</a>
                </div>
            </li>
        </ul>
    </div>";
}
?>
</body>
</html>

```

Na slici 5 se nalazi prikaz strane sa ponudama.



Slika 6.1.5 Strana ponude [Izvor: Autor]

KONTAKT STRANA

Kontakt strana treba da sadrži formu za slanje dodatnih pitanja

Kontakt strana sadrži formu za slanje dodatnih pitanja. Na slici 6 je prikaz ove strane, a u nastavku i kod.

Slika 6.1.6 Kontakt strana [Izvor: Autor]

```
<?php
require_once ("header.php");
?>
<div class="container">
    <center>
        <form>
            <h2>Forma za dodatna pitanja</h2>
            <div class="col-50">
                <input type="text" name="ime" placeholder="Ime"/>
            </div>
            <div class="col-50">
                <input type="text" name=" prezime" placeholder="Prezime"/>
            </div>
            <div class="col-50">
                <input type="text" name="email" placeholder="E-mail"/>
            </div>
            <div class="col-50">
                <input type="text" name="naslov" placeholder="Naslov"/>
            </div>
            <div class="col-50">
                <textarea name="poruka" placeholder="Tekst poruke...."></textarea>
            </div>
            <input type="submit" value="Submit"/>
        </form>
    </center>
</div>
</body>
</html>
```

STRANA SA FORMOM ZA DODAVANJE NOVE PONUDE

Potrebno je kreirati stranu za dodavanje novih ponuda kojoj može pristupiti samo admin korisnik

Kao što je na početku navedeno prilikom login-a korisnika započinje se sesija u kojoj se čuvaju podaci korisnika koji je ulogovan. Pomoću ovih podataka se može proveri da li je korisnik admin ili ne. Kako pristup strani za kreiranje ovih ponuda treba omogućiti samo admin korisnicima, na početku ove strane nalazi se provera koja to kontroliše.

```
<?php
    require_once("header.php");
    if(!$is_mod){
        header('Location: index.php');
        exit(0);
    }
?>
<div class="container">
```

```
<center>
    <form action="dodajPonudu.php" method="POST">
        <h2>Dodaj novu ponudu</h2>
        <div class="col-50">
            <input type="text" name="odrediste" placeholder="Odrediste"/>
        </div>
        <div class="col-50">
            <textarea name="opis" placeholder="Opis odredista"></textarea>
        </div>
        <div class="col-50">
            <input type="text" name="cena" placeholder="Cena"/>
        </div>
        <div class="col-50">
            <input type="text" name="slika" placeholder="Slika putovanja"/>
        </div>
        <input type="submit" value="Dodaj"/>
    </form>
</center>
</div>
</body>
</html>
```

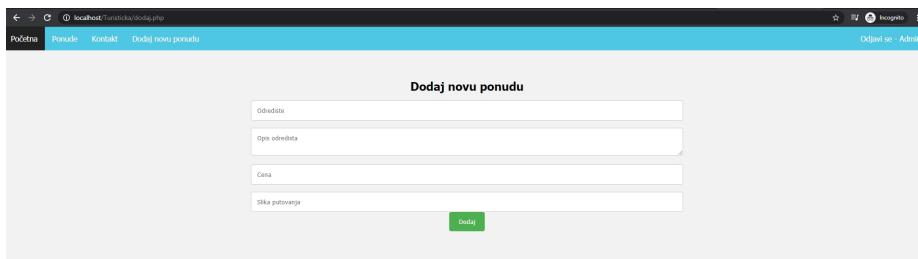
Prethodna forma poziva skriptu dodajPonudu.php koja podatke iz forme upisuje u bazu podataka. U nastavku se nalazi kod za dodavanje ponude.

```
<?php
    require_once('konekcija.php');
    $con = povezi();

    $odrediste = $_POST["odrediste"];
    $opis = $_POST["opis"];
    $cena = $_POST["cena"];
    $slika = $_POST["slika"];

    if(isset($_POST['odrediste']) && isset($_POST['opis']) && isset($_POST['cena'])
    && isset($_POST['slika'])){
        $stmt = $con->prepare("INSERT INTO putovanje (odrediste, opis, cena, slika)
VALUES (:odrediste, :opis, :cena, :slika)");
        $stmt->bindParam(":odrediste", $odrediste);
        $stmt->bindParam(":opis", $opis);
        $stmt->bindParam(":cena", $cena);
        $stmt->bindParam(":slika", $slika);

        $stmt->execute();
    }
    header('Location: dodaj.php');
?>
```



Slika 6.1.7 Prikaz strane za dodavanje nove ponude [Izvor: Autor]

STILIZOVANJE STRANA

Strane su stilizovane pomoću CSS-a

```
/* Start General */
* {
    margin:0;
    padding:0;
    font-family: Tahoma;
    box-sizing:border-box;
    -o-box-sizing:border-box;
    -ms-box-sizing:border-box;
    -moz-box-sizing:border-box;
    -webkit-box-sizing:border-box;
}
body{
    background-color: white;
}
nav ul li a {
    color: #FFF;
    text-decoration: none;
}
/* End General */

/* Start Navbar */
nav ul {
    background-color: #4dc7e3;
}
nav ul > li {
    padding: 15px;
    display: inline-block;
    transition: all .3s ease;
    margin-left: -5px
}
nav ul > li:not(:last-of-type):hover {
    background-color: #222
}
nav ul > li:first-of-type {
    background-color: #222
}
nav ul > li:last-of-type {
```

```
float: right;
}

nav ul > li:last-of-type a .fa {
    font-size: 21px
}
/* End Navbar */

/* Start Menue Right */
nav ul > ol {
    position: absolute;
    top: 49px;
    right: 0;
    background: #333;
    text-align: center;
    list-style: none;
    display: none
}
nav ul > ol > li {
    width: 100vw;
    color: #FFF;
    margin: 0;
    padding: 0;
    padding-top: 10px;
    padding-bottom: 10px;
    transition: all .3s ease;
}
nav ul > ol > li:hover a {
    margin: 20px;
}
nav ul > ol > li:hover {
    background: #000;
    cursor: pointer
}
nav ul input {
    opacity: .7;
    padding: 5px;
    float: right;
    display: none
}
/* Start Menue Right */

/* Start Media Query */
@media screen and (max-width:680px){
    nav ul > li:not(:first-child) {
        display:none
    }
    nav ul > li:last-of-type {
        display: block
    }
    nav ul input {
        display: inline
    }
}
```

```
@media screen and (min-width:680px) {  
    nav ul > ol > li {  
        display:none  
    }  
}  
/* End Media Query */  
  
input[type=text], select, textarea {  
    width: 100%;  
    padding: 12px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
    resize: vertical;  
}  
  
input[type=password], select, textarea {  
    width: 100%;  
    padding: 12px;  
    border: 1px solid #ccc;  
    border-radius: 4px;  
    resize: vertical;  
}  
  
input[type=submit] {  
    background-color: #4CAF50;  
    color: white;  
    padding: 12px 20px;  
    border: none;  
    border-radius: 4px;  
    cursor: pointer;  
}  
  
input[type=submit]:hover {  
    background-color: #45a049;  
}  
  
.container {  
    border-radius: 5px;  
    background-color: #f2f2f2;  
    padding: 60px;  
}  
  
.col-50 {  
    width: 50%;  
    margin-top: 15px;  
}  
  
/* Clear floats after the columns */  
.row:after {  
    content: "";  
    display: table;  
    clear: both;  
}
```

```
.mySlides {  
    display: none  
}  
  
/* Slideshow container */  
.slideshow-container {  
    max-width: 1000px;  
    position: relative;  
    margin: auto;  
}  
  
/* Next & previous buttons */  
.prev, .next {  
    cursor: pointer;  
    position: absolute;  
    top: 50%;  
    width: auto;  
    padding: 16px;  
    margin-top: -22px;  
    color: white;  
    font-weight: bold;  
    font-size: 18px;  
    transition: 0.6s ease;  
    border-radius: 0 3px 3px 0;  
    user-select: none;  
}  
  
/* Position the "next button" to the right */  
.next {  
    right: 0;  
    border-radius: 3px 0 0 3px;  
}  
  
/* On hover, add a black background color with a little bit see-through */  
.prev:hover, .next:hover {  
    background-color: rgba(0,0,0,0.8);  
}  
  
/* Caption text */  
.text {  
    color: #f2f2f2;  
    font-size: 15px;  
    padding: 8px 12px;  
    position: absolute;  
    bottom: 8px;  
    width: 100%;  
    text-align: center;  
}  
  
/* Number text (1/3 etc) */  
.numbertext {  
    color: #f2f2f2;
```

```
font-size: 12px;
padding: 8px 12px;
position: absolute;
top: 0;
}

/* The dots/bullets/indicators */
.dot {
    cursor: pointer;
    height: 15px;
    width: 15px;
    margin: 0 2px;
    background-color: #bbb;
    border-radius: 50%;
    display: inline-block;
    transition: background-color 0.6s ease;
}

.active, .dot:hover {
    background-color: #717171;
}

/* Fading animation */
.fade {
    -webkit-animation-name: fade;
    -webkit-animation-duration: 1.5s;
    animation-name: fade;
    animation-duration: 1.5s;
}

@-webkit-keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

@keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

/* On smaller screens, decrease text size */
@media only screen and (max-width: 300px) {
    .prev, .next,.text {font-size: 11px}
}

.list li {
    border-bottom: 1px solid #ccc;
    display: table;
    border-collapse: collapse;
    width: 100%;
}
.inner {
    display: table-row;
```

```
    overflow: hidden;
}
.li-img {
    display: table-cell;
    vertical-align: middle;
    width: 50%;
    padding-right: 1em;
}
.li-img img {
    display: block;
    width: 100%;
    height: auto;
}
.li-text {
    display: table-cell;
    vertical-align: middle;
    width: 70%;
}
.li-head {
    margin: 0;
}
.li-sub {
    margin: 0;
}

@media all and (min-width: 45em) {
    .list li {
        float: left;
        width: 50%;
    }
}

@media all and (min-width: 75em) {
    .list li {
        width: 33.333333%;
    }
}
```

✓ 6.1 Zadaci za samostalni rad: PHP aplikacije

ZADATAK 1 - APLIKACIJA ZA TURISTIČKE PONUDE

Iskoristiti prethodni primer i dodati nove funkcionalnosti

Ukupno vreme trajanja: 90 minuta

Očekivano vreme izrade zadatka: 15 minuta.

Zadatak

Nadograditi prethodno objašnjen primer turističke agencije tako da se osposobi link na strani ponude ka detaljima za svaku od ponuda.

Za svaku ponudu već postoji definisan link detaljnije o ovoj ponudi koji treba izmeniti tako da se otvara detail strana sa opisom odabrane ponude.

ZADATAK 2 - IGRA

Kreirati PHP aplikaciju koja sa korisnikom igra igru papir, kamen, makaze.

Očekivano vreme izrade zadatka: 15 minuta.

Zadatak

Kreirati PHP aplikaciju koja sa korisnikom igra igru makaze, papir, kamen. Potrebno je kreirati formu u kojoj će korisnik izabrati jednu od opcija, dok server za svoj izbor koristi random. Kreirati stranu na kojoj će se ispisati rezultat poteza, i trenutni skor. Za čuvanje skora koristiti sesije.

ZADATAK 3 - ŠIFROVANJE

Kreirati aplikaciju koja će omogućiti korisniku da šifruje tekst pomoću Cezarove šifre, ili neke njene varijacije.

Očekivano vreme izrade zadatka: 25 minuta.

Kreirati aplikaciju koja će omogućiti korisniku da šifruje tekst pomoću Cezarove šifre, ili neke njene varijacije.

Cezarova šifra je tip šifre zamjene (supstitucije), u kome se svako slovo otvorenog teksta zamjenjuje odgovarajućim slovom abecede, pomaknutim za određeni broj mesta. Na primjer, s pomakom 3, A se zamjenjuje slovom D, B slovom E itd.

ZADATAK 4 - LOGIN FORMA

Po uzoru na sliku, kreirati formu za registraciju i logovanje na sajt

Očekivano vreme izrade zadatka: 30 minuta.

- Po uzoru na sliku, kreirati formu za registraciju i logovanje na sajt.
- Sajt registrovane korisnike mora da upisuje u bazu.
- Svaki korisnik mora da ima korisničku sesiju.
- Potrebno je za bazu sa korisnicima kreirati formu za pretragu korisnika.
- Sajt mora biti otporan na SQL injection.



Slika 6.2.1 Sajt za vežbu [Izvor: Autor]

✓ Poglavlje 7

Domaći zadatak

OPIS DOMAĆEG ZADATKA - DZ14

Kreirati PHP aplikaciju koja treba da predstavlja forum.

Očekivano vreme izrade zadatka: 45 minuta.

Kreirati PHP aplikaciju koja treba da predstavlja forum. Potrebno je kreirati:

- Login i register stranu za korisnike
- Glavnu stranu sa listom tema koje postoje
- Na glavnoj strani treba da se nalazi i opcija za kreiranje nove teme, kao i opcija logout
- Na glavnoj strani, za teme koji je ulogovani korisnik kreirao, treba da ima opciju njihovog brisanja
- Odabirom teme na glavnoj strani otvara se nova strana gde se mogu videti odgovori/komentari na temu
- Korisnik treba da ima opciju na strani određene teme da ostavi odgovor/komentar

Aplikacija treba da ispunjava:

- Svaki korisnik mora da ima korisničku sesiju.
- Otporna je na SQL Injection napade i pogrešno korišćenje.

Potrebne datoteke snimiti pod imenom:

IT210-DZ14-Ime_Prezime_brojIndexa

gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovane datoteke poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210 - DZ14)

▼ Zaključak

ZAKLJUČAK

U ovom predavanju :

- Dati su primeri datoteka koje reprezentuju svako od tri stanja informacija. Pokazano je da jedan dokument može istovremeno da bude u sva tri stanja
- Dati su primeri ranjivosti za specifična stanja
- Opisano je kako redundantnost i geografska disperzija utiču na raspoloživost
- Definisani su autentifikacija, integritet, neprihvatanje, nepriznavanja i tajnost kao bezbednosne usluge
- Opisano je kako se koristi jednosmerna kriptografija za implementaciju integriteta dokumenta pri prenosu
- Opisano je kako se kriptografski algoritmi koriste za zaštitu tajnosti dokumenta pri prenosu
- Opisano je kako se jednosmerne funkcije koriste za usluge autentikacije i neprihvatanje nepriznavanja
- Identifikovani su aspekti poslovanja koji mogu biti ugroženi bezbednosnim propustima. Kvantifikovani su finansijske gubici izazvani bezbednosnim propustima
- Identifikovano je i opisano devet koraka za ocenu rizika vezanog za bezbednost definisanih od strane NIST-a

Literatura

1. Harlan Carvey, Windows Forensics and Incident Recovery, Addison Wesley, 2004
2. John R. Vacca, Computer Forensics: Computer Crime Scene Investigation, Charles River Media, 2002
3. Hal Tipton and Micki Krause, Handbook of Information Security Management, CRC Press LLC, 1998
4. Michael Miller, Apsolutna zaštita PC-ja i privatnosti, Kompjuter biblioteka, Čačak, 2003.
5. Matt Bishop, Introduction to Computer Security, Prentice Hall PTR, 2004
6. National Institute of Standards and Technology, An Introduction to Computer Security: The NIST Handbook, Special Publication 800-12, 1995
7. Marianne Swanson, Barbara Guttman, Generally Accepted Principles and Practices for Securing Information Technology Systems, National Institute of Standards and Technology, Special Publication 800-14, 1995



IT210 - SISTEMI IT

INTEGRACIJA

Lekcija 15

PRIRUČNIK ZA STUDENTE

IT210 - SISTEMI IT

Lekcija 15

INTEGRACIJA

- ✓ INTEGRACIJA
- ✓ Poglavlje 1: INTEGRACIJA POSLOVNIH APLIKACIJA
- ✓ Poglavlje 2: ARHITEKTURA EAI SISTEMA
- ✓ Poglavlje 3: TEHNIKE INTEGRACIJE
- ✓ Poglavlje 4: DATA WAREHOUSES
- ✓ Poglavlje 5: ARHITEKTURA DATA WAREHOUSE SISTEMA
- ✓ Poglavlje 6: Pokazne vežbe: PHP APLIKACIJE 2
- ✓ Poglavlje 7: Zadaci za samostalni rad: PHP APLIKACIJE 2
- ✓ Poglavlje 8: DOMAĆI ZADATAK
- ✓ ZAKLJUČAK

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

✓ Uvod

UVOD

Fokus ove lekcije su integracije sistema i tehnike integracije

Na ovom predavanju biće reči o:

1. Integraciji

- Komponente, interfejsi i integracija
- Infrastruktura, srednji sloj (**middleware**) i platform

2. Tehnike integracije

- Data warehouses, extending frameworks, wrappers, glue, facades

▼ Poglavlje 1

INTEGRACIJA POSLOVNIH APLIKACIJA

INTEGRACIJA POSLOVNIH APLIKACIJA

Integrисано реšење се развија kreiranjem tzv. EAI zona - група aplikacija slične funkcije, односно група aplikacija čijim zajedničким функционисањем se obezбеђује ostvarenje jedног циља

Kada različiti sistemi ne mogu da efikasno dele i razmenjuju podatke, u informacionom sistemu se stvaraju uska grla kroz koje se protok ne može obezbediti bez ljudske intervencije u obliku donošenja odluka ili unosa podataka. EAI predstavlja disciplinu koja je nastala zahvaljujući godinama razvoja aplikacionih sistema za podršku poslovanju, tokom kojih se nije vodilo računa o sposobnosti tih sistema da međusobno komuniciraju.

Integrисање poslovnih aplikација (engl. **EAI** - Enterprise Application Integration) predstavlja disciplinu koja obuhvata skup procesa, softverskih i hardverskih alata, metodologija i tehnologija, чijом se integrисањом implementацијом постиже konsolidација, povezивање и организовање свих poslovnih računarsких aplikација, података и процеса у integrисано okruženje koje obezbeđuje razмену, управљање и reformулацију информација и зnanja iz domena poslovanja preduzeća, u realnom vremenu.

Integrисано реšење се развија kreiranjem tzv. EAI zona - група aplikacija slične funkcije, односно група aplikacija čijim zajedničким функционисањем se obezбеђује ostvarenje jedног poslovnog циља. Svaka od pojedinačних aplikација sa осталима u okviru своје EAI zone treba da komunicira putem jedinstvenog protokola, pri čemu je поželjно да se тaj protokol користи i за комуникацију између различитих EAI zona. Svaku od aplikација unutar EAI решења treba da karakterише не зависност - интерфејс integrисања представља само један програмски слој над том aplikацијом, а eventualne измене aplikacije, односно njena kompletна замена другом не sme da utiče na funkcionalnost ukupnog integrisanog okruženja.

OSNOVNE KARAKTERISTIKE APLIKACIONOG OKRUŽENJA

Osnovне karakterистике aplikacionog okruženja integrisanog primenom EAI principa su: interoperabilnost, sinhrone i asinhrone interakcije, transformacija podataka i procesna logika

Osnovne karakteristike aplikacionog okruženja integrisanog primenom EAI principa su:

- Interoperabilnost
- Sinhrone i asinhrone interakcije
- Preslikavanje i transformacija podataka
- Procesna logika (deterministička i nedeterministička)
- Kompenzacija transakcija

INTEROPERABILNOST APLIKACIJA

Interoperabilnost jedne aplikacije u odnosu na njeno okruženje karakteriše sposobnost slanja odgovarajuće poruke odgovarajućoj aplikaciji

Interoperabilnost jedne aplikacije u odnosu na njeno okruženje karakteriše sposobnost slanja odgovarajuće poruke odgovarajućoj aplikaciji, bez obzira na to gde se nalazi ta aplikacija, da li je aktivna ili ne i, bez obzira na to u kom okruženju (operativni sistemi, hardver) funkcioniše.

Interoperabilnost aplikacija može da se sagleda sa dva aspekta, od kojih svaki karakterišu različiti principi. Prvi se odnosi na internu komunikaciju dve aplikacije, unutar granica poslovnog sistema. Drugi je sposobnost integrisanog informacionog sistema, odnosno odgovornih aplikacija, da komunicira sa spoljnjim svetom - aplikacijama koje se nalaze izvan granica poslovnog sistema. nosioci komunikacije između različitih aplikacija u EAI okruženju se nazivaju *porukama* (engl. *message*), pri čemu se interakcija dve aplikacije ostvaruje njihovom *razmenom* (engl. *messaging*). Razmena poruka se ostvaruje putem odgovarajućih *kanala* (engl. *channels*), duž kojih se kreću poruke paketi podataka jedinstvene, definisane strukture. Poruka se sastoji iz *zaglavlja* (engl. *header*), u kojem se nalaze identifikacije pošiljaoca i primaoca poruke i drugi meta podaci, i *tela* (engl. *body*), koje čini sam sadržaj poruke.

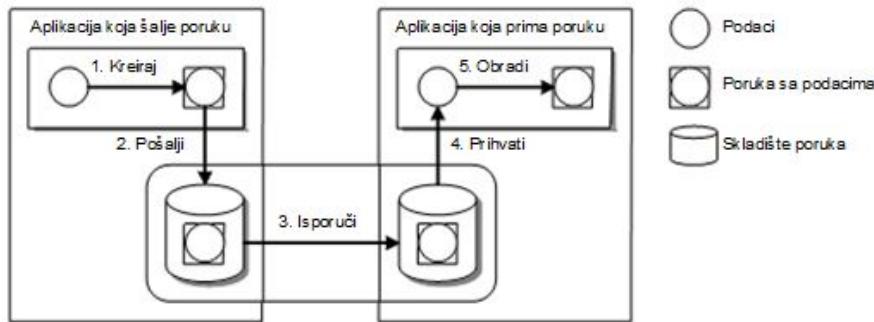
STRUKTURA INTERAKCIJE DVE APLIKACIJE U EAI OKRUŽENJU

Interakcija dve aplikacije u EAI okruženju se obavlja u 5 koraka

Na slici je prikazana **struktura interakcije dve aplikacije u EAI okruženju**, putem razmene poruka. Ona se obavlja u 5 koraka:

- *Kreiranje poruke*. Aplikacija - pošiljalac kreira poruku i popunjava podatke koji čine njen sadržaj;
- *Slanje poruke*. Aplikacija - pošiljalac šalje poruku u kanal komunikacije. Nakon slanja poruke, aplikacija - pošiljalac nastavlja sa radom, pri čemu nema više nikakvih odgovornosti u procesu razmene poruka;
- *Isporuka poruke*. EAI sloj za interakciju koji implementira kanal komunikacije se stara o tome da se poruka prenese duž mrežne infrastrukture, sa računara na kome se izvršava aplikacija pošiljaoca, na računar na kojem se izvršava aplikacija - primaoca poruke.

- **Prijem poruke.** EAI sloj za interakciju se stara o tome da je poruka primljena, skladištena na računar na kome se izvršava aplikacija - primaoca, i dostupna njoj u svakom ili odgovarajućem trenutku;
- **Obrada poruke.** Aplikacija - primalac vrši ekstrakciju podataka iz poruke i koristi ih za obradu.



Slika 1.1 Razmena poruka izmedju dve aplikacije [Izvor: Autor]

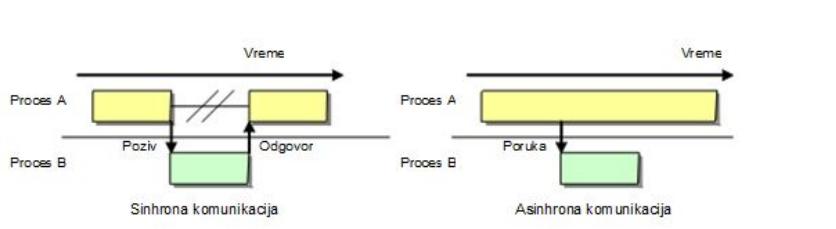
SINHRONE I ASINHRONE INTERAKCIJE

Sinhrona interakcija predstavlja razmenu poruka koja se obavlja u realnom vremenu, pri čemu aplikacija koja šalje poruku drugoj aplikaciji, očekuje od nje odgovor u realnom vremenu

Sinhrona interakcija predstavlja razmenu poruka koja se obavlja u realnom vremenu, pri čemu aplikacija koja šalje poruku drugoj aplikaciji, očekuje od nje odgovor u realnom vremenu. Komunikaciju karakteriše dvosmerni tok informacija u jednom komunikacionom kanalu, duž kojeg se kreću **zahtev** (engl. *request*) i **njegov odgovor** (engl. *response*).

Ipak, veliki broj poslovnih okolnosti karakterišu problemi zbog kojih se sinhronom komunikacijom ne može zaokružiti jedan poslovni proces na pouzdan i efikasan način. Naime, u velikom broju slučajeva, aplikacija koja upućuje zahtev mora da čeka na njegov odgovor, zbog vremena potrebnog za procesiranje zahteva, proisteklog iz potrebe za reakcijom operatera ili zbog opterećenja računarskih resursa. Ovo stanje čekanja blokira rad aplikacije koja je uputila poziv i zauzima mrežne resurse. Sa druge strane, nepouzdanost i sporost komunikacione infrastrukture može da uvede veliki rizik narušavanja integriteta poslovnog sistema usled prekinute sinhronne interakcije aplikacija.

Asinhrona komunikacija se ostvaruje u dva komunikaciona kanala, duž kojih se kreću zahtev i odgovor u različitim smerovima. Ona se ne vrši u realnom vremenu, već u konačno dugom vremenskom periodu, pri čemu se dobijanje odgovora obezbeđuje novim zahtevom ili otvaranjem kanala za prijem odgovora, koji funkcioniše uporedno u odnosu na ostale funkcije aplikacije i ne angažuje značajne aplikacione resurse.



Slika 1.2 Sinhrona i asinhrona komunikacija [Izvor: Autor]

PRESLIKAVANJE I TRANSFORMACIJA PODATAKA

Transformacija podataka predstavlja aktivnost konverzije podataka u prihvatljive formate sa stanovišta njihovog korisnika.

Aplikacije u jednom heterogenom okruženju nisu projektovane, razvijene i instalirane sa ciljem da funkcionišu integrисано. One uobičajeno koriste različite formate podataka i protokole interfejsa za eksternu komunikaciju. Zbog toga, neophodno je da EAI sistem karakterише sposobnost preslikavanja i transformacije podataka u realnom vremenu, prilikom njihovog transfera i njihovog preslikavanja iz izvornog u ciljni format.

Preslikavanje podataka (engl. **data mapping**) podrazumeva uspostavljanje referenci između formata podataka koji se koristi za njihovo skladištenje (najčešće, relacioni model) i formata podataka koji se koristi za obradu unutar jedne aplikacije.

Transformacija podataka predstavlja aktivnost konverzije podataka u prihvatljive formate sa stanovišta njihovog korisnika. Osnovna karakteristika uobičajene strategije integrisanja informacionog sistema preduzeća je korišćenje jedinstvenog - kanoničnog formata podataka u njihovom transportu od jedne do druge aplikacije. Kanoničan model podataka je model koji je nezavisan od bilo koje aplikacije u okruženju. Danas, najkorišćeniji kanoničan model podataka je **XML** (engl. **eXtensible Markup Language**).

PROCESNA LOGIKA

Poslovne informacione sisteme karakterиše složena poslovna logika.

Poslovne informacione sisteme karakterиše složena **poslovna logika**. EAI sistem treba da karakterиše sposobnost izražavanja i automatizacije ove logike i to primenom paradigmе poslovnih procesa, odnosno **radnih tokova** (engl. **workflow**).

KOMPENZACIJA TRANSAKCIJA

Osnovno sredstvo za očuvanje integriteta jedne interakcije je njena transakciona priroda.

S obzirom na zahtev za asinhronom komunikacijom unutar jednog integrisanog poslovnog sistema, od izuzetne je važnosti da se očuva integritet podataka koji se razmenjuju u toku jedne interakcije dve aplikacije, odnosno integritet same interakcije. Osnovno sredstvo za očuvanje integriteta jedne interakcije je njena transakcionalna priroda.

Transakcionalna interakcija dve aplikacija u okviru EAI okruženja je izvršena samo i jedino nakon eksplisitne potvrde (engl. **commit**) njenog izvršenja. Pre potvrde, interakcija se može otkazati, bez obzira na fazu u kojoj se nalazi njen izvršenje. Otkazivanje transakcione interakcije nakon potvrde nije moguće ni u kom slučaju. U slučaju da je izvršenje jedne interakcije dovelo aplikacioni sistem u nestabilno ili generalno, neželjeno stanje, on se može povratiti u stabilno stanje primenom kompenzacije transakcije.

Kompenzacija transakcione interakcije obuhvata njen zaustavljanje i reviziju definisanog, već izvršenog skupa operacija transfera i obrada. Iako se često prepostavlja da je uloga kompenzacije povraćaj aplikacionog sistema u prethodno stanje, to nije uvek tačno - kompenzacija može da podrazumeva i dovođenje sistema u balansirano i konzistentno stanje, umanjenjem ili poništavanjem dejstva interakcije ili interakcija koje su ga dovele u neželjeno stanje.

OSNOVNI PRINCIPI IMPLEMENTACIJE EAI

EAI predviđa različite načine implementacije

EAI predviđa različite načine implementacije, od kojih su najčešće korišćeni:

- *Integriranje informacija*. Obezbeđivanje konzistentnosti informacije koja se koristi u različitim sistemima.
- *Procesna integracija*. Projektovanje procesa koji prožimaju rad različitih aplikacionih sistema
- *Nezavisnost vendor-a* (engl. **Vendor independence**). Ekstrahovanje poslovne logike iz aplikacionih sistema i njihovo implementiranje u EAI sistem, tako da i u situacijama kada se aplikacioni sistem zameni drugim sistemom različitog vendor-a, poslovna logika ne mora da se ponovo implementira.
- *Fasadni pristup*. EAI sistem igra ulogu front-enda - fasade za grupu aplikacija, obezbeđujući jedinstven i konzistentan interfejs za sve aplikacione sisteme.

KOJI PROIZVODI IMPLEMENTIRAJU ILI KORISTE EAI PATERNE?

Paterni pružaju programerima i projektantima, nezavisno od tehnologija, da opišu i razviju robusna integraciona rešenja.

Paterni pružaju programerima i projektantima, nezavisno od tehnologija, da opišu i razviju robusna integraciona rešenja.

Integracija preduzeća je previše složena da bi se mogla rešiti jednostavnim pristupom „recepta iz kuvara“. Umesto toga, paterni mogu pružiti smernice dokumentovanjem različitih

iskustava, koja obično žive samo u glavama arhitekata, a mogu biti korisna rešenja za ponavljajuće probleme u datom kontekstu. Paterni su dovoljno apstraktni da se primenjuju na većinu integracionih tehnologija, ali dovoljno specifični da pružaju praktične smernice projektantima. Paterni takođe pružaju vokabular programerima da efikasno opišu svoje rešenje.

Paterni nisu vezani za konkretnu implementaciju. Oni pomažu da se projektuju bolja rešenja, bez obzira da li se koristi neka od sledećih platformi:

- EAI i SOA platforme, kao što su IBM WebSphere MQ, TIBCO, Vitria, Oracle Service Bus, WebMethods (now Software AG), Microsoft BizTalk, ili Fiorano.
- Open source ESB's kao što su Mule ESB, JBoss Fuse, Open ESB, WSo2, Spring Integration, ili Talend ESB
- Message Brokeri kao što su ActiveMQ, Apache Kafka, ili RabbitMQ
- Veb servisi ili integracije bazirane na REST-u, uključujući Amazon Simple Queue Service (SQS) ili Google Cloud Pub/Sub
- JMS messaging sistemi
- Microsoft tehnologije kao što su MSMQ ili Windows Communication Foundation (WCF)

BUDUĆNOST INTEGRACIJA - VIDEO

Future of Enterprise Application Integration

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 2

ARHITEKTURA EAI SISTEMA

MIDDLEWARE TEHNOLOGIJA ZA REALIZACIJU EAI SISTEMA

Middleware predstavlja osnovnu tehnologiju realizacije jednog EAI sistema, pri čemu se veoma često koristi i za podršku funkcionisanju kompleksnih distribuiranih aplikacija.

Middleware predstavlja osnovnu tehnologiju realizacije jednog EAI sistema, pri čemu se veoma često koristi i za podršku funkcionisanju kompleksnih distribuiranih aplikacija.

Middleware predstavlja bilo koji softver ili deo softvera koji povezuje dve ili više aplikacija na način koji obezbeđuje razmenu podataka. Iako se termin koristi još od 1968 (NATO Software Engineering Conference), ova tehnologija je stekla popularnost osamdesetih godina, kao rešenje za povezivanje savremenih aplikacionih sistema sa legacy softverom. U poslednje vreme, zahvaljujući razvoju distribuiranih softverskih sistema, middleware tehnologije dobijaju značajnu pažnju. U tom smislu, jedna od najznačajnijih primena middleware koncepta, danas su aplikacioni serveri.

Neke od značajnih primena middleware koncepta su sledeće tehnologije:

- Poziv udaljene procedure (engl. **Remote Procedure Call - RPC**). Tehnologija omogućava da klijentski softver vrši pozive procedura koje se izvršavaju na udaljenom računaru - serveru.
- **Message Oriented Middleware (MOM)**. Softver koji omogućava asinhronu komunikaciju između različitih aplikacija korišćenjem poruka kao formata za slanje i prijem podataka koje aplikacije razmenjuju.
- **Object Request Broker (ORB)**. Tehnologija koja omogućava razmenu podataka između dva objektno-orientisana sistema, i to u formi objekata.
- SQL-orientisan pristup podacima. Deo softvera koji se koristi za razmenu podataka između aplikacije i sistema za upravljanje bazama podataka.
- Aplikacioni serveri. Softver koji se koristi za izvršavanje drugih aplikacija, najčešće distribuiranih.

ARHITEKTURA EAI SISTEMA

Postoje dva osnovna pristupa projektovanju EAI sistema: hub-and-spoke i enterprise service bus (ESB).

Postoje dva osnovna pristupa projektovanju EAI sistema: **hub-and-spoke** i **enterprise service bus** (ESB). Prvi pristup karakteriše centralna pozicija EAI sistema, koji igra ulogu hub-a, posrednika sa aplikacijama koje se nalaze u zoni integracije. U drugom pristupu, EAI sistem ima ulogu magistrale, ili je implementiran kao rezidentan modul postojeće magistrale ili middleware-a.

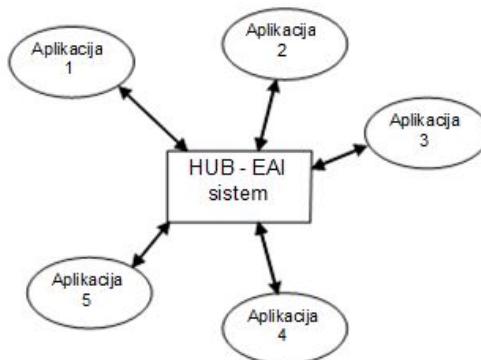
HUB-AND-SPOKE ARHITEKTURA EAI SISTEMA

Hub-and-spoke arhitektura predstavlja realizaciju topologije zvezde u oblasti integrisanja poslovnih aplikacija

Hub-and-spoke arhitektura predstavlja realizaciju topologije zvezde u oblasti integrisanja poslovnih aplikacija, pri čemu EAI sistem ima centralnu ulogu posrednika u interakciji između bilo koje dve aplikacije iz zone integracije.

Osnovna prednost ovakve topologije je veoma jednostavno dodavanje novih aplikacija u zonu integracije, implementacijom jedinstvenog interfejsa za komunikaciju sa EAI sistemom.

Ipak, **hub-and-spoke** arhitektura se relativno retko koristi za integrisanje velikog broja aplikacija. Razlozi za to su relativno mala fleksibilnost i uobičajeno visoko opterećenje hub-a.



Slika 2.1 Hub arhitektura [Izvor: Autor]

Centralizovani model karakteriše relativno mala fleksibilnost. Izmene u funkcionisanju hub-a mogu dovesti do neočekivanih posledica u komunikaciji sa aplikacijama iz zone integracije. Sa druge strane, periodično veliki saobraćaj na pojedinačnim krovovima, nasuprot malom ili nikakvom opterećenju na drugim krovovima, može značajno usporiti performanse EAI sistema, odnosno, dovesti u pitanje njegovo funkcionisanje.

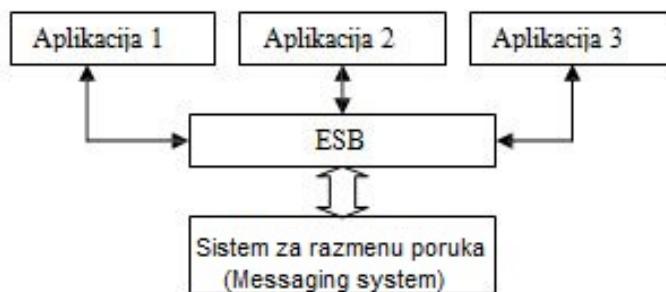
Hub predstavlja usko grlo ukupne zone integracije. Naime, on je posrednik za sve saobraćaj koji je namenjen integrisanju bilo koje dve aplikacije u zoni integracije. U tom smislu, ukupan kapacitet komunikacije svih aplikacija u zoni integracije je ograničen kapacitetom hub-a. Očigledno je da bilo kakav zastoj u funkcionisanju hub-a, usled njegove preopterećenosti, utiče na ukupne performanse sistema, naročito ukoliko je u projektovanju EAI sistema korišćen patern federacije.

ESB PRISTUP

Generalno, ESB predstavlja apstraktni sloj messaging sistema informacionog sistema preduzeća, postavljen tako da može da koristi njegove funkcije, bez dodatnog razvoja.

Enterprise Service Bus (ESB) predstavlja oblik arhitekture softvera, zasnovan na standardima složenijih **event-driven** arhitektura i **messaging** sistema, koji se implementira primenom middleware tehnologija.

Generalno, ESB predstavlja apstraktni sloj messaging sistema informacionog sistema preduzeća, postavljen tako da može da koristi njegove funkcije, bez dodatnog razvoja.



Slika 2.2 ESB [Izvor: Autor]

KARAKTERISTIKE ESB PRISTUPA

ESB obuhvata skup standardnih karakteristika - sposobnosti integrisanog sistema, koje se mogu implementirati na proizvoljan način.

ESB obuhvata skup standardnih karakteristika - sposobnosti integrisanog sistema, koje se mogu implementirati na proizvoljan način. Opšte prihvaćene funkcije enterprise service bus-a su:

- Podrška sinhronim i asinhronim transportnim protokolima za poziv funkcija poslovnih aplikacija.
- Rutiranje poruka kroz zonu integracije, na osnovu definisanog adresnog prostora, odnosno dodeljenih adresa se vrši na osnovu predviđenih putanja poruka ili u realnom vremenu - na osnovu sadržaja poruka.
- Medijacija aplikacija u zoni integracije se vrši uz pomoć adaptera, pri čemu su obezbeđene funkcije translacije protokola, odnosno transformacije i prevođenja podataka.
- Na osnovu predviđene definicije poslovnih procesa, može se vršiti orkestracija aplikacija u zoni integracije. Oko ove funkcije ESB, ne postoji industrijski konsenzus. Dok svi ostali vendori smatraju da je ona važna karakteristika enterprise service bus-a, IBM ovu

funkciju realizuje posebni sistemom - WebSphere Process Server, u odnosu na svoj - WebSphere ESB.

- Obrada događaja u zoni integracije se vrši primenom mehanizama interpretacije događaja, korelacije podataka, upoređivanje paterna, itd.
- Sigurnost (enkripcija i digitalno potpisivanje), pouzdanost isporuke poruka i implementirani mehanizmi za transakcije
- Upravljanje ukupnim sistemom se vrši primenom funkcija monitoringa, auditinga, logovanja, itd.

TEHNOLOŠKI ZAHTEVI ESB PRISTUPA

ESB se uobičajeno implementira nezavisno od korišćenih operativnih sistema i programskih jezika.

Pored ovih, osnovnih funkcija ESB, primenu ovakvog oblika arhitektura karakterišu i tehnološki zahtevi, kao što su:

- Korišćenje XML-a, kao standardnog jezika za komunikaciju između aplikacija,
- Primenu transformacionih servisa (XSLT, XQuery) za prilagođavanje formata podataka i vrednosti u različitim kontekstima njihovog korišćenja,
- Validacija poruka, naspram predviđene šeme poruka prilikom njihovog slanja i prijema,
- Uobičajeno se implementira nezavisno od korišćenih operativnih sistema i programskih jezika. Naime, ESB treba da obezbedi ravnopravnu interakciju svih poslovnih aplikacija, bez obzira na programski jezik koji je korišćen za njihov razvoj i platformu na kojoj se one izvršavaju.
- Podrška zadržavanju poruka, ukoliko su poslovne aplikacije kojima su one namenjene trenutno nedostupne, i drugi.

PREDNOSTI I MANE ESB PRISTUPA

Osnovna prednost ESB pristupa je velika fleksibilnost u implementaciji novih zahteva ukupnog sistema

Osnovne **prednosti** primene **enterprise service bus-a** u integrisanju poslovnih aplikacija su:

- velika fleksibilnost u implementaciji novih zahteva ukupnog sistema;
- zasnovanost na standardima;
- mogućnost primene u različitim opsezima integracije - veličinama zona integracije;
- značajno umanjena potreba za razvojem prilikom implementacije - ona se primarno zasniva na konfigurisanju ESB-a, i;
- implementacija ESB-a, odnosno njegovih pojedinačnih funkcija i veza se može vršiti bez dodatnog opterećenja poslovnih aplikacija i uz održavanje potpunog kontinuiteta njihovog funkcionisanja.

Ključne **mane** ovog pristupa su:

- ESB se može primeniti samo u poslovnom informacionom sistemu u kojem postoji messaging arhitektura;
- Iako je osnova za fleksibilnost ovog pristupa tzv. slaba integracija, bez značajnog i pažljivog planiranja implementacije ESB, postoji rizik za uspostavljanje upravo čvrste integracije;
- Za implementaciju, konfigurisanje i administraciju ESB, potrebno je angažovati dodatne ljudske resurse;
- Za efikasnu primenu, neophodno je da u preduzeću postoji uređen sistem procesa i detaljno definisana strategija poslovanja.

ENTERPRISE SERVICE BUS - VIDEO

Enterprise Service Bus

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 3

TEHNIKE INTEGRACIJE

PROGRAMSKI PRISTUPI INTEGRISANJA POSLOVNIH APLIKACIJA

Direktnu integraciju poslovnih aplikacija na nivou programskog koda, karakterišu najbolje performanse interfejsa, odnosno međusobne komunikacije.

U teoriji i praksi oblasti integrisanja poslovnih aplikacija, postoji veliki broj, funkcionalno i logički različitih pristupa, koji se koriste za kreiranje jedinstvenog poslovnog informacionog sistema, a čijom se implementacijom prevazilazi monolitnost konvencionalnog softvera.

U ovom predavanju, predstavljena su tri pristupa, odnosno paterna za programiranje čijom se primenom može ostvariti integrisanje poslovnih aplikacija:

- *Korišćenje vezivnog koda*
- *Korišćenje adaptera*
- *Korišćenje fasada*

Sva tri pristupa se odnose na programsku realizaciju interfejsa, odnosno posrednika između dve poslovne aplikacije otvorenog izvornog koda. Dakle, oni nisu vezani za paterne višeg nivoa, već načine direktne programske realizacije interfejsa između dve aplikacije, u cilju njihovog integrisanja.

Primenu bilo kojeg od navedenih pristupa karakteriše neophodnost neograničenog i potpunog pristupa programskom kodu aplikacije. Iako je to normalan slučaj kod poslovnih aplikacija koje su naručene od strane njihovih korisnika i razvijene na osnovu detaljnih zahteva, segment custom-made poslovnih aplikacija nije toliko zastupljen na tržištu njihovih korisnika. Razvoj ponude enterprise softvera, njegov sve veći kvalitet u smislu generičnosti i fleksibilnosti, utiče na to da se ovaj segment smanjuje.

Drugi problem predstavlja uobičajeno visoka cena naknadnog razvoja postojeće **custom-made** aplikacije, kao i održavanja nastalog koda, naročito ukoliko se zanemare određeni standardi kvaliteta razvoja u uslovima neophodnosti brzog izvođenja.

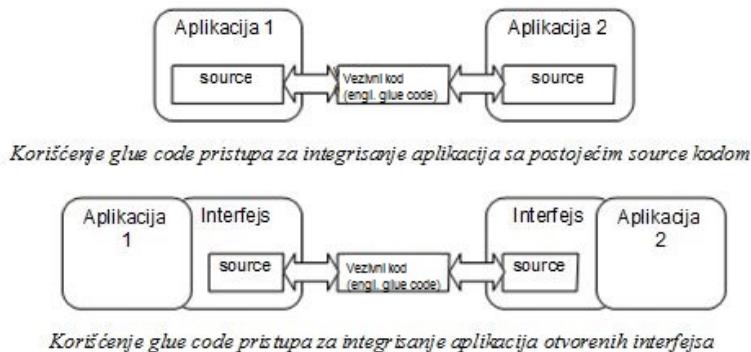
Sa druge strane, direktnu integraciju poslovnih aplikacija na nivou programskog koda, karakterišu najbolje performanse interfejsa, odnosno međusobne komunikacije.

PRISTUP KORIŠĆENJA VEZIVNOG KODA

Pristup korišćenja vezivnog koda se najčešće sreće u praksi kod obezbeđivanja interoperabilnosti postojećih programske biblioteka ili celih softverskih aplikacija.

U teoriji i praksi programiranja, **glue code** (**vezivni kod**) je programski kod koji ne predstavlja realizaciju neke funkcije, odnosno konkretnog zahteva, već se koristi za "lepljenje", ili vezivanje različitih delova koda koji, inače nisu kompatibilni. Pristup korišćenja vezivnog koda se najčešće sreće u praksi kod obezbeđivanja interoperabilnosti postojećih programske biblioteka ili celih softverskih aplikacija.

Ovaj pristup se, na primer, može koristiti kao most između API funkcija koda poslovnih aplikacija, ali i nekompatibilnih interfejsa različitih aplikacija kao medijator saobraćaja, sa osnovnom funkcijom transformacije poruka koje se kreću između dva interfejsa.



Slika 3.1 Korišćenje vezivnog koda [Izvor: Autor]

SWIG

Jedan od primera veoma efikasnog korišćenja pristupa vezivnog koda u praksi je SWIG

Jedan od primera veoma efikasnog korišćenja pristupa vezivnog koda u praksi je **SWIG** (engl. **Simplified Wrapper and Interface Generator**), softver otvorenog koda koji se koristi za povezivanje programa ili biblioteka, napisanih u C/C++ jeziku, sa skript jezicima, kao što su Tcl, Perl, Python, Ruby, PHP, Lua, ali i drugim jezicima (Java, C#, Scheme, Ocaml). Osnovna funkcija SWIG softvera je da ostvari vezu između različitih aplikacija uz minimalan napor u implementaciji. Naime, uz definisanje veoma malog broja direktiva u header datotekama koda, SWIG alat može da generiše vezivni kod koji se može koristiti za integriranje C/C++ funkcija i funkcija izabranog jezika. U zavisnosti od izabranog ciljnog jezika, vezivni kod se može pojaviti u tri oblika:

- Ukoliko je ciljni jezik - neki od skript jezika (Perl, Python, PHP, itd.), vezivni kod se generiše kao izvršni kod koji se ponaša kao originalni program, sa integrisanim interpreterom za izabrani skript jezik;

- Biblioteka funkcija koju izabrani interpreter može da koristi kao modul;
- Biblioteka funkcija koja se može povezati sa drugim programima, kompajliranim u ciljnom jeziku (na primer, korišćenje JNI - Java Native Interface-a u Java jeziku).

SWIG je napisan u C/C++ jeziku, i dostupan je od 1996. godine. Trenutno ga podržava i proširuje veliki broj programera volontera u sourceforge zajednici, a izdat je pod BSD licencom, što znači da se može koristiti, distribuirati, kopirati i modifikovati potpuno besplatno, u komercijalne i nekomercijalne svrhe.

U uslovima, u kojima u jednom poslovnom informacionom sistemu postoje dve monolitne aplikacije izuzetnog značaja, napisane u različitim programskim jezicima, pristup korišćenja vezivnog koda najčešće predstavlja najbrži put ka integraciji. Ukoliko preduzeće nije spremno da investira u nabavku novih sistema čija se integracija može izvršiti primenom neke od drugih metoda, vezivni kod predstavlja brzo i efikasno rešenje.

Sa druge strane, iako SWIG okruženje omogućava brzu i jednostavnu integraciju aplikacija, napisanih u različitim programskim jezicima, diverzitet tehnologija u jednom poslovnom informacionom sistemu uvek predstavlja ozbiljan problem za njegovo održavanje, fleksibilnost korišćenja i unapređivanje.

ADAPTER PATTERN

Adapter omogućava dvema klasama sa nekompatibilnim interfejsima da funkcionišu integrisano

Adapter patern (u literaturi se veoma često koristi i izraz **wrapper patern**, ili samo **wrapper**) predstavlja pristup programskoj realizaciji integrisanja, koji karakteriše adaptacija interfejsa klase jedne aplikacije u interfejs koji druga aplikacija (**klijent**), očekuje. Adapter omogućava dvema klasama sa nekompatibilnim interfejsima da funkcionišu integrisano.

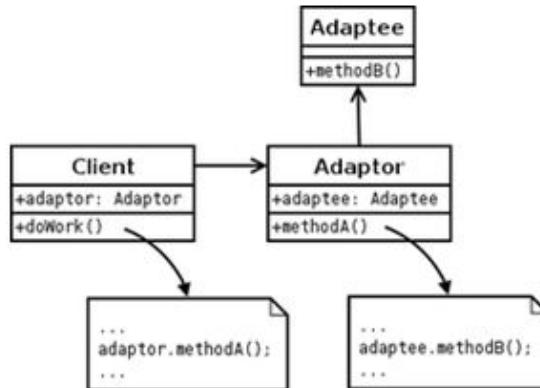
U realnom životu, primena pristupa korišćenja adapter paterna se u praksi sreće izuzetno često. Jedan od primera je predstavljen na slici 2. Kada je potrebno izvršiti zatezanje nekog elementa ključem, očigledno, prečnik glave ključa mora da odgovara prečniku gnezda. Ukoliko ne postoji ključ sa odgovarajućom glavom, koristi se adapter.

Pored "obuhvatanja" (engl. **wrapping**) postojećeg interfejsa, adapter je takođe odgovoran za primenu logike neophodne za sve potrebne transformacije podataka, u komunikaciji dve klase. Na primer, u programerskoj praksi se, za definisanje binarnih tipova, uobičajeno koriste **boolean** tipovi podataka, ali i **single integer** tipovi. U ovom slučaju, adapter bi bio odgovoran za transformaciju tipova u transferu podataka između dveju klase.

TIPOVI ADAPTER ŠABLONA

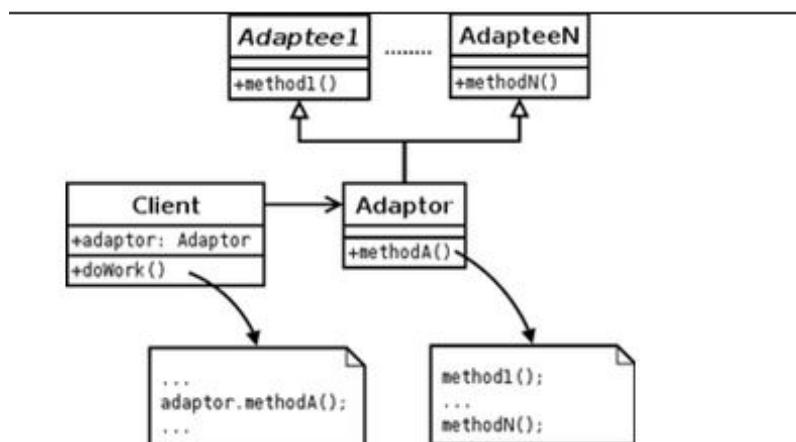
Postoje dva tipa adapter paterna: adapter objekata i adapter klasa

Postoje dva tipa adapter paterna: adapter objekata i adapter klasa. UML šema funkcionisanja adaptera objekata je prikazana na slici 2. U ovom slučaju, adapter sadrži instancu klase koju "obuhvata" i poziva njene metode.



Slika 3.2 Adapter objekata [Izvor: <https://medium.com/better-programming/design-patterns-adapter-efb73c5090e6>]

Adapter klasa koristi osobinu višestrukog nasleđivanja u cilju integrisanja. Adapter je kreiran, tako da nasleđuje postojeći interfejs, ali istovremeno i interfejs koji klijentska aplikacija očekuje. Očigledno je da se ovaj tip adaptera - adapter klasa, ne može koristiti u programskim jezicima koji ne podržavaju višestruko nasleđivanje, kao što je Java.



Slika 3.3 Adapter klasa [Izvor: <https://medium.com/better-programming/design-patterns-adapter-efb73c5090e6>]

PREDNOSTI I MANE ADAPTER ŠABLONA

Pristup korišćenja adaptera je veoma koristan u situacijama, u kojima postojeća klasa sadrži sve metode koje su potrebne za integrisanje sa drugom aplikacijom, ali ne sadrži adekvatan interfejs.

Pristup korišćenja adaptera je veoma koristan u situacijama, u kojima postojeća klasa sadrži sve metode koje su potrebne za integrisanje sa drugom aplikacijom, ali ne sadrži adekvatan interfejs.

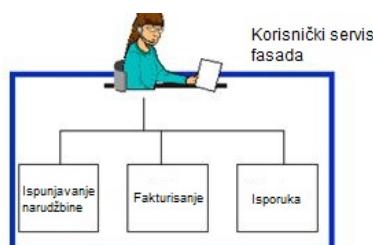
U odnosu na pristup korišćenja vezivnog koda, mana primene adapter klase je ograničenost na integrisanje aplikacija, koje su napisane u istom programskom jeziku. Sa druge strane, njena primena predstavlja bolji pristup sa stanovišta očuvanja integriteta funkcija, odnosno klasa koje se integrišu. U tom smislu, održavanje aplikacija nakon integracije nije ni na koji način otežano.

PRISTUP KORIŠĆENJA FASADA I INTERGACIJI

Fasada predstavlja objekat koji obezbeđuje pojednostavljeni interfejs ka jednoj, relativno velikoj celini programskog koda, kao što je biblioteka klasa, ili većem skupu interfejsa

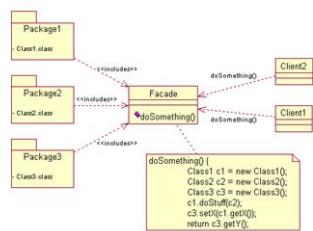
Fasada predstavlja objekat koji obezbeđuje pojednostavljeni interfejs ka jednoj, relativno velikoj celini programskog koda, kao što je biblioteka klasa, ili većem skupu interfejsa. Ona se koristi u slučajevima kada je potrebno obezbediti pojednostavljenje velikog broja komplikovanih interakcija objekata, lakše korišćenje API funkcija i uopšte - razumevanje funkcija biblioteke softvera, naročito ukoliko je ona napisana nekonzistentno. Naime, fasada može predstavljati jedinstveni, konzistentan interfejs prema različitim delovima koda, istih ili različitih softverskih aplikacija.

Sa stanovišta primene u rešavanju neprogramerskih problema, jedan od najčešćih primera koji u praksi predstavlja apstraktну realizaciju fasade predstavlja korisnički servis. On predstavlja jedinstveno mesto pristupa za kontakt klijenata sa preduzećem, pri čemu njegovi službenici imaju ulogu da upit, zahtev ili komentar klijenta na odgovarajući način obrade, koristeći sve poslovne funkcije preduzeća, kao što su obrada narudžbina, naplata, isporuka, itd.



Slika 3.4 Primer šablona fasade [Izvor: Autor]

Na slici 5, prikazan je UML model prvog kojim se demonstrira korišćenje fasada u integriranju. U ovom slučaju fasada ima ulogu da obezbedi komunikaciju klijenata 1 i 2 - klasa iste ili druge aplikacije sa paketima klasa 1, 2 i 3. S obzirom na složenost paketa klasa i njihovu različitost, umetanje posredničkog sloja komunikacije, u vidu fasade, predstavlja značajno pojednostavljenje za realizaciju integrisanog sistema.



Slika 3.5 UML model šabloni fasade [Izvor: <https://www.learn-it-with-examples.com/development/java/java-design-patterns/java-facade.html>]

▼ Poglavlje 4

DATA WAREHOUSES

DATA WAREHOUSING

Data warehousing tehnologije predstavljaju jedan deo korporativne IT arhitekture, nosilac arhitekture tzv. sistema poslovne inteligencije

Data warehousing tehnologije predstavljaju jedan deo korporativne IT arhitekture, nosilac arhitekture tzv. sistema poslovne inteligencije, nastale kao rezultat evolucije fokusa razvoja informacionih sistema preduzeća, od rešavanja operacionih problema, do elemenata za podršku donošenju odluka. Naime, iako informacioni sistemi omogućavaju obradu velikog broja raznorodnih informacija i predstavljanje rezultata na različite načine, nemoguće je predvideti sve poslovne situacije u kojima je računarskipodržana podrška neophodna savremenim menadžerima. Iako postojeće funkcije neke poslovne aplikacije podržavaju veliki deo domena poslovanja, u određenim slučajevima je neophodno angažovanje konsultanata i programera, za prilagođavanje aplikacije npr. novonastalim poslovnim okolnostima. Dalje, informacija, potrebna u realnom vremenu, može predstavljati sintezu drugih informacija, koje se obrađuju u različitim delovima poslovnog informacionog sistema - aplikacijama koje nisu međusobno povezane, odnosno integrisane. U ovom slučaju, potrebna informacija se dobija manuelnim radom ili angažovanjem treće aplikacije.

Očigledno je da monolitnost poslovnih aplikacija, u funkcionalnom smislu, kao i sa aspekta mogućnosti integracije sa drugim aplikacijama predstavlja ozbiljan problem za dinamično poslovanje, koje karakteriše brzo donošenje odluka na osnovu realnih, aktuelnih podataka o tekućem poslovanju.

DATA WAREHOUSING - VIDEO

Data Warehousing - An Overview

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

KARAKTERISTIKE SKLADIŠTA PODATAKA

Skladišta podataka predstavljaju "skup integrisanih, subjektno orijentisanih baza podataka, projektovanih sa ciljem da snabdevaju menadžment informacija potrebnim u procesu donošenj

Skladišta podataka (engl. **data warehouses**) predstavljaju komponentu poslovnog informacionog sistema čiji je osnovni cilj da prevaziđe navedene probleme. Ona predstavljaju "skup integrisanih, subjektnoorijentisanih baza podataka, projektovanih sa ciljem da snabdevaju menadžment informacija potrebnim u procesu donošenja odluka". Osnovne karakteristike skladišta podataka su:

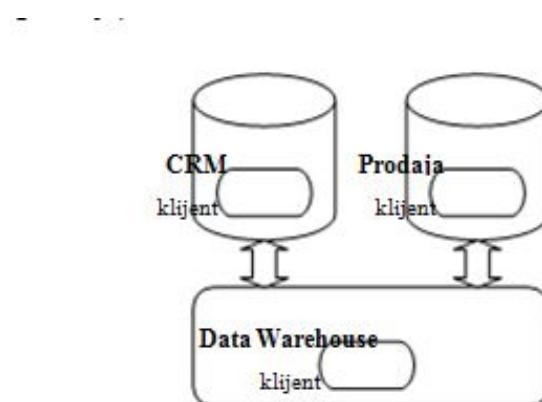
- Integriranost informacija;
- Orijentisanost na poslovne subjekte;
- Obavezno korišćenje vremenske dimenzije; i
- Kontinuiranost.

Integriranost skladišta podataka se odnosi na činjenicu da ona sadrže informacije koje su sublimirane iz različitih poslovnih aplikacija jednog preduzeća, odnosno na osnovu njihovih podataka. U nekim slučajevima, u skladištu podataka su integrisane i informacije čije je poreklo van granica preduzeća. U situacijama u kojima poslovne aplikacije nisu integrisane, one mogu sadržati različite vrste informacija o različitim poslovnim objektima, pri čemu, u ovakvom slučaju može doći i do redundantnosti podataka - konfliktnih podataka o nekom poslovnom objektu. Skladišta podataka se koriste za stvaranje jedinstvenog ili parcijalnog pogleda na sve informacije vezane za određeni poslovni subjekat.

PRIMER DATA WAREHOUSE TEHNOLOGIJE

Na primer, klijent preduzeća predstavlja poslovni subjekat, kojeg karakterišu različite informacije koje se skladište u različitim poslovnim aplikacijama

Na primer, klijent preduzeća predstavlja poslovni subjekat, kojeg karakterišu različite informacije koje se skladište u različitim poslovnim aplikacijama. Primenom tehnologija skladišta podataka, može se obezbediti jedinstven pogled na sve informacije i ostvarene relacije klijenta unutar granica preduzeća, npr. osnovne informacije, karakteristike demografske grupe kojoj pripada, obavljene kontakte (CRM), ali i otpremnice i fakture (prodaja) (vidi sliku 1.).



Slika 4.1 Primer data warehouse tehnologije [Izvor: Autor]

DENORMALIZACIJA I IZVEŠTAJI

Denormalizacija znači da skladišta podataka u okviru repozitorijuma čuvaju iste podatke na različitim nivoima detaljnosti

Za razliku od konvencionalne baze podataka, skladište podataka predstavlja repozitorijum informacija koje se mogu samo čitati, ali ne i menjati. Uobičajeno izuzetno veliki, on obuhvata sve podatke iz svih poslovnih aplikacija preduzeća, ali i eksterne podatke, nastale u komunikaciji sa partnerima preduzeća, klijentima, itd. Pored toga, skladište podataka sadrži i meta podatke - informacije koje opisuju sadržaj skladišta podataka.

Za razliku od relacionih baza podataka, koje su projektovane sa ciljem da obezbede i održe integritet podataka, osnovni cilj projektovanja skladišta podataka je obezbeđenje njegove korisnosti, odnosno korisnosti informacija koje sadrži. Karakteristike relacionih baza podataka koje koriste poslovne aplikacije su jednostavnost modela podataka i brzina modifikacije, čime se obezbeđuju uslovi za *obradu transakcija u realnom vremenu* (engl. *online transaction processing - OLTP*). Sa druge strane, skladišta podataka se projektuju sa ciljem da se stvore uslovi za izveštavanje i analizu u realnom vremenu (engl. *online analytical processing - OLAP*).

Iz navedenih razloga, **struktura informacija u skladištu podataka, za razliku od relacionih baza, nije normalizovana**. Ovakav pristup omogućava menadžeru da sužava domen informacije koju želi (**drill-down**), analizirajući podatke o nekom poslovnom subjektu ili oblasti, od najopštijeg nivoa poslovanja, ka njihovim detaljnim reprezentacijama. Na taj način, on može da dobije elemente za donošenje odluka, koje su u skladu sa operativnim zahtevima trenutnih okolnosti, ali i najširom strategijom preduzeća.

Denormalizacija znači da skladišta podataka u okviru repozitorijuma čuvaju iste podatke na različitim nivoima detaljnosti. Iako se podaci na najdetaljnijim nivoima koriste za izvođenje agregatnih podataka, i oni sami se čuvaju u repozitorijumu, da bi se omogućio brži odziv na upite na opštijim nivoima.

DINAMIČKI IZVEŠTAJI

Svaki od kreiranih izveštaja može lako da modifikuje, uključujući u njega detaljniji pregled informacija, ili generalizujući ga na agregatni nivo, bez ikakvog programiranja.

Jedna od značajnih karakteristika data warehousing tehnologija je i mogućnost kreiranja **dinamičkih izveštaja**. Ona predstavlja jedno od osnovnih sredstava za rešavanje problema monolitnosti konvencionalnih poslovnih aplikacija. Naime, zahvaljujući data warehouse sistemu, menadžer može, u svakom trenutku da samostalno kreira svoj skup izveštaja, na osnovu svih potrebnih podataka. Takođe, svaki od kreiranih izveštaja može lako da modifikuje, uključujući u njega detaljniji pregled informacija, ili generalizujući ga na agregatni nivo, bez ikakvog programiranja.

PRIMERI DINAMIČKIH IZVEŠTAJA

Bez ikakvog programiranja, jednostavnim izborom opcije za pregled na nižem nivou detaljnosti, menadžer može da dobije izveštaj

Primeri dinamičkog izveštaja su prikazani u tabelama 1. i 2. Na primer, u tabeli 1, prikazane su ukupne performanse prodaje preduzeća, na teritoriji Republike Srbije.

Region	Planirana prodaja	Ostvarena prodaja
Beograd	3.000.000,00	3.125.982,00
Vojvodina	2.800.000,00	2.810.000,00
Zapadna Srbija	2.100.000,00	2.550.000,00
Južna Srbija	1.100.000,00	825.000,00

Slika 4.2 Tabela 1 - Primer izveštaja ukupne prodaje po regionima [Izvor: Autor]

Bez ikakvog programiranja, jednostavnim izborom opcije za pregled na nižem nivou detaljnosti, menadžer može da dobije izveštaj, čiji je primer prikazan u tabeli 2. Uvidom u njega, on može da lokalizuje uočeni problem i stekne uslove za donošenje određene odluke, odnosno inicira rešavanja problema.

Ovaj izveštaj predstavlja najopštiji izveštaj o poslovanju preduzeća, na osnovu kojeg se može krenuti u dublje analize, ukoliko je to potrebno. Na primer, menadžer uočava da je u regionu Južna Srbija, ostvarena prodaja manja od planirane. U cilju analize očiglednog problema, on želi da izvrši uvid u performanse prodaje preduzeća, na nižem nivou detaljnosti, npr. da razmotri podatke o prodaji po gradovima.

Region	Grad	Planirana prodaja	Ostvarena prodaja
Beograd		3.000.000,00	3.125.982,00
Vojvodina	Novi sad	1.200.000,00	1.210.000,00
	Vrbas	800.000,00	800.000,00
	Subotica	800.000,00	800.000,00
Zapadna Srbija	Čačak	1.000.000,00	1.150.000,00
	Kraljevo	600.000,00	650.000,00
	Užice	500.000,00	750.000,00
Južna Srbija	Niš	600.000,00	200.000,00
	Vranje	350.000,00	400.000,00
	Surdulica	150.000,00	225.000,00

Slika 4.3 Tabela 2 - Primer izveštaja ukupne prodaje po regionima i gradovima [Izvor: Autor]

PREDNOSTI I NEDOSTACI DATA WAREHOUSING TEHNOLOGIJA

Primenom data warehousing tehnologija, menadžeri mogu donositi brze i adekvatne poslovne odluke, bez ikakve stručne pomoći u potreboj obradi informacija

Osnovne prednosti data warehousing tehnologija se ogledaju u izuzetnim mogućnostima analitičke obrade velikog skupa podataka iz svih domena poslovanja preduzeća. Primenom data warehousing tehnologija, menadžeri mogu donositi brze i adekvatne poslovne odluke, bez ikakve stručne pomoći u potreboj obradi informacija, neophodnih za stvaranje konteksta za donošenje odluka. Jedinstvenim sagledavanjem performansi preduzeća, ostvaruje se i izuzetan stepen integracije poslovnih informacionih sistema, koja doduše, nije izvršena na transakcionom nivou, jer ne postoji neposredna veza između aplikacija u realnom vremenu.

Sa druge strane, obavezno je naglasiti da postoje izuzetno veliki problemi u korišćenju ovih tehnologija, proistekli, između ostalog, iz sledećih činjenica:

- Učitavanje, prikupljanje informacija u skladište podataka i njihovo "čišćenje" predstavlja aktivnost izuzetno dugog trajanja, koja opterećuje funkcionisanje poslovnog informacionog sistema sa stanovišta intenzivnog korišćenja računarskih i infrastrukturnih resursa;
- Projekat uvođenja data warehousing tehnologija se mora planirati veoma pažljivo, naročito sa aspekta definisanja okvira, odnosno ciljeva koje je potrebno ostvariti predviđenom dinamikom;
- Veoma često se pojavljuje problem sa kompatibilnošću data warehouse sistema sa poslovnim aplikacijama, odnosno bazama podataka koje oni koriste;
- Direktan pristup data warehouse sistema relacionim bazama podataka poslovnih aplikacija otvara problem njihove bezbednosti, kao i bezbednosti samog skladišta - naročito ukoliko je ono bazirano na veb tehnologijama, što je čest slučaj u praksi;

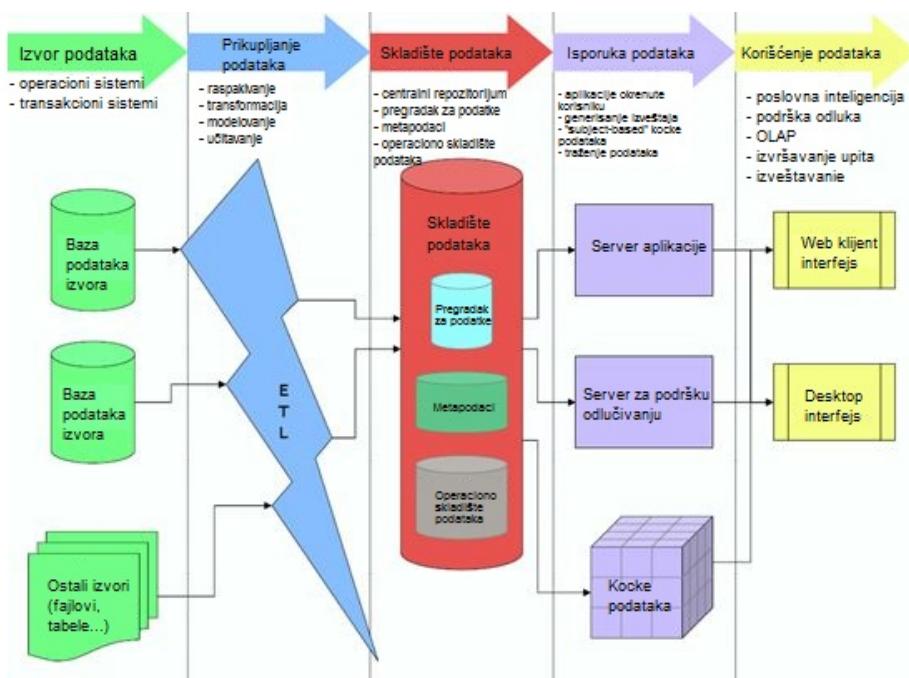
✓ Poglavlje 5

ARHITEKTURA DATA WAREHOUSE SISTEMA

GENERIČKA ARHITEKTURA DATA WAREHOUSE SISTEMA

Osnovni ciljevi projekta generičke arhitekture su da se obezbedi proširivost sistema, njegova fleksibilnost, višestruko korišćenje njegovih komponenti, ali pre svega - produktivnost u korišćenju

Na slici 1, prikazana je **generička arhitektura** data warehouse sistema, predstavljena na funkcionalnim nivoima. Oni obuhvataju funkcije skladištenja izvornih podataka, funkcije prikupljanja podataka, funkcije skladištenja u data warehouse sistemu, funkcije isporuke i korišćenja podataka. Osnovni ciljevi projekta generičke arhitekture su da se obezbedi proširivost sistema, njegova fleksibilnost, višestruko korišćenje njegovih komponenti, ali pre svega - produktivnost u korišćenju.



Slika 5.1 Arhitektura data warehouse sistema [Izvor: Autor]

KOMPONENTE SKLADIŠENJA IZVORNIH PODATAKA I NJIHOVOG PRIKUPLJANJA

Na nivou funkcije skladištenja izvornih podataka, nalaze se poslovne aplikacije, odnosno njihove relacione baze podataka, kao i drugi izvori podataka

Na nivou funkcije skladištenja izvornih podataka, nalaze se poslovne aplikacije, odnosno njihove relacione baze podataka, kao i drugi izvori podataka - datoteke koje se koriste u poslovanju, interfejsi prema informacionim sistemima partnera, itd.

Na nivou funkcija prikupljanja podataka, nalaze se komponente za ekstrakciju podataka iz skladišta izvornih podataka, njihova transformacija u predviđene modele, kao i učitavanje u samo data warehouse skladište.

KOMPONENTE SKLADIŠENJA PODATAKA U DATA WAREHOUSE SISTEMU

Skladište podataka predstavlja centralni repozitorijum data warehouse sistema. Ono se sastoji iz datamart-ova, repozitorijuma meta podataka i operativnog skladišta podataka

Skladište podataka predstavlja centralni repozitorijum data warehouse sistema. Ono se sastoji iz datamart-ova, repozitorijuma meta podataka i *operativnog skladišta podataka* (engl. *Operational Data Store*).

Dok skladište podataka sadrži ogromnu količinu podataka prikupljenih iz svih poslovnih aplikacija, datamart predstavlja samo jedan njihov deo, koji se može odnositi na funkcionisanje jedne organizacione celine preduzeća, njegove teritorijalne celine, i sl.

Skladište podataka sadrži i **meta podatke** - informacije koje opisuju sadržaj skladišta podataka, i alate za njihovo održavanje.

Konačno, poslednja komponenta funkcije skladištenja podataka je operativno skladište podataka (engl. *Operational Data Store*). Funkcije ove komponente su veoma slične ukupnoj funkciji skladišta podataka. Ipak, za razliku od skladišta podataka čiji je osnovni cilj podrška u donošenju strateških odluka, cilj operativnog skladišta je da obezbedi operativno praćenje poslovanja. Ono sadrži samo informacije o trenutnom poslovanju, na mnogo detaljnijem nivou nego što je to slučaj u skladištu podataka. Takođe, za razliku od skladišta podataka, podaci u operativnom skladištu se kontinuirano osvežavaju novim podacima iz poslovnih aplikacija, tako da njihove vrednosti uvek odražavaju aktuelno stanje performansi poslovnog sistema.

KOMPONENTE ISPORUKE I KORIŠĆENJA PODATAKA IZ DATA WAREHOUSE SISTEMA

Kocke podataka (data cubes) predstavljaju višedimenzionalne poglede na performanse preduzeća, proistekle iz potrebe da se poslovanje sagleda u odnosu na više od 2 parametra

U generičkoj arhitekturi jednog data warehouse sistema, osnovne funkcije komponente isporuke podataka su da obezbedi prezentaciju podataka i odgovarajućih korisničkih interfejsa korisnicima sistema, generisanje izveštaja i tzv. kocki podataka (engl. **data cubes**), kao i data mining funkcije.

Kocke podataka (engl.**data cubes**) predstavljaju višedimenzionalne poglede na performanse preduzeća, proistekle iz potrebe da se poslovanje sagleda u odnosu na više od 2 parametra (tabelarni prikaz), pri čemu je broj dimenzija kocke podataka jednak broju parametara čiji se uticaji analiziraju.

Sa stanovišta fizičke arhitekture data warehouse sistema, na nivou funkcije isporuke podataka se nalaze razni aplikacioni serveri, serveri za podršku odlučivanju itd.

Konačno, na nivou funkcije korišćenja podataka, u generičkoj arhitekturi data warehouse sistema, nalaze se komponente za podršku odlučivanju, *analitičku obradu u realnom vremenu* (engl. **online analytical processing**), definisanje upita i kriterijuma za izgradnju izveštaja i drugi alati iz domena poslovne inteligencije. Ove komponente se koriste posredstvom klijentskih aplikacija, koje mogu biti zasnovane na web interfejsima ili tankim klijentima (engl. **thin clients**).

▼ Poglavlje 6

Pokazne vežbe: PHP APLIKACIJE 2

PRIMER APLIKACIJE: INTERNET AUKCIJA

Potrebno je kreirati sajt online aukcije

Očekivano vreme izrade zadatka: 65 minuta.

Potrebno je kreirati sajt online aukcije upotreboom HTML-a, CSS-a, PHP-a i baze podataka. U MySQL ili MS Access kreirati bazu podataka pod nazivom "aukcija". Baza aukcija treba da sadrži tabele kao na slici 1.

aukcija korisnik	artikl	ponuda
korisnik_id : int(11)	artikl_id : int(11)	ponuda_id : int(11)
username : text	naziv : text	korisnik_id : int(11)
password : text	opis : text	artikl_id : int(11)
ime : text	slika : text	iznos : int(11)
prezime : text	# cena : int(11)	
email : text	trajanje : text	
	# korisnik_id : int(11)	

Slika 6.1 Prikaz tabela baze aukcija [Izvor: Autor]

U PHP-u prvo kreiramo fajl konekcija.php za konekciju na kreiranu bazu.

```
<?php
    function povezi(){
        try {
            $con = new PDO("mysql:host=localhost;dbname=aukcija", "root", "");
            $con->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $con;
        }
        catch(PDOException $e){
            echo "Error: " . $e->getMessage();
        }
    }
?>
```

Sajt treba da ima navigacioni bar koji treba da bude konzistentan na svim stranama pa se zato kreira header.php fajl koji će kasnije biti pozivan u ostalim fajlovima pomoću require funkcije.

Na početku ovog fajla nalazi se skripta koja uzima podatke iz sesije i čuva podatke korisnika koji će kasnije biti korišćeni prilikom unosa artikala i davanja ponuda.

```
<?php
    session_start();
    require_once('konekcija.php');
    $con = povezi();
    $stmt = $con->prepare('SELECT * FROM korisnik WHERE username=:username');
    $stmt->bindParam(":username", $_SESSION['username']);

    $stmt->execute();
    $USER = $stmt->fetch();
    $id = $USER["korisnik_id"];
?>
<!DOCTYPE html>
<html>
<head>
    <title>Aukcije</title>
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <link rel="icon" href="pic/logo.png">
</head>
<body>
<header>
    <nav>
        <ul>
            <li><a href="index.php">Početna</a></li>
            <li><a href="aukcije.php">Aukcije</a></li>
            <?php
                if(isset($_SESSION["username"])){
                    echo "<li><a href='dodaj.php'>Dodaj novu aukciju</a></li>";
                }
                else{
                    echo "";
                }
            >
            <li><a href="kontakt.php">Kontakt</a></li>
            <?php
                if($USER["ime"] != ""){
                    echo "<li><a href='logout.php'>Odjavi se - " . $USER['ime'] . ";
                }
                else{
                    echo "<li><a href='loginstrana.php'>Login</a></li>";
                }
            >
        </ul>
    </nav>
</header>
```

POČETNA STRANA

Početna strana aplikacije

Na slici 2 dat je prikaz početne strane aplikacije i kod.



Slika 6.2 Početna strana aplikacije [Izvor: Autor, snapshot]

```
<?php
    require_once ("header.php");
?>
<div class="slideshow-container">

<div class="mySlides fade">
    <div class="numbertext">1 / 3</div>
    
    <div class="text">Aukcija</div>
</div>

<div class="mySlides fade">
    <div class="numbertext">2 / 3</div>
    
    <div class="text">Aukcija</div>
</div>

<div class="mySlides fade">
    <div class="numbertext">3 / 3</div>
    
    <div class="text">Aukcija</div>
</div>

<a class="prev" onclick="plusSlides(-1)">#10094;</a>
<a class="next" onclick="plusSlides(1)">#10095;</a>

</div>
<br/>

<div style="text-align:center">
    <span class="dot" onclick="currentSlide(1)"></span>
    <span class="dot" onclick="currentSlide(2)"></span>
    <span class="dot" onclick="currentSlide(3)"></span>
</div>
<script>
var slideIndex = 1;
showSlides(slideIndex);

function plusSlides(n) {
```

```
    showSlides(slideIndex += n);
}

function currentSlide(n) {
    showSlides(slideIndex = n);
}

function showSlides(n) {
    var i;
    var slides = document.getElementsByClassName("mySlides");
    var dots = document.getElementsByClassName("dot");
    if (n > slides.length) {slideIndex = 1}
    if (n < 1) {slideIndex = slides.length}
    for (i = 0; i < slides.length; i++) {
        slides[i].style.display = "none";
    }
    for (i = 0; i < dots.length; i++) {
        dots[i].className = dots[i].className.replace(" active", "");
    }
    slides[slideIndex-1].style.display = "block";
    dots[slideIndex-1].className += " active";
}
</script>
</body>
</html>
```

LOGIN I REGISTER STRANA

U nastavku su kreirane login i register strana i php skripte koje to realizuju

Na strani login (loginstrana.php) se nalazi forma sa poljima za username i password.

```
<?php
    require_once ("header.php");
    if(isset($_SESSION["username"])){
        header("Location: index.php");
    }
?>
    <div class="container">
        <center>
            <form action="login.php" method="POST">
                <h2>Ulogujte se</h2>
                <div class="col-50">
                    <input type="text" name="username" placeholder="Username"/>
                </div>
                <div class="col-50">
                    <input type="password" name="password"
placeholder="Password"/>
                </div>
```

```
<input type="submit" value="Login"/>
<br/>
<a href="registerstrana.php">Nemate nalog? Registrujte se</a>
</form>
</center>
</div>
</body>
</html>
```

Ova forma poziva login.php skriptu koja uzima podatke iz forme proverava ih sa podacima u bazi i ukoliko se poklapaju otvara sesiju.

```
<?php
    require_once('konekcija.php');
    $con = povezi();
    $username = $_POST["username"];
    $password = $_POST["password"];
    $password = md5($password);

    if(isset($_POST['username']) && isset($_POST['password'])){
        $stmt = $con->prepare("SELECT * FROM korisnik WHERE username=:username AND password=:password");
        $stmt->bindParam(":username", $username);
        $stmt->bindParam(":password", $password);
        $stmt->execute();

        $row = $stmt->fetch();
        if($row){
            session_start();
            $_SESSION["username"] = $row["username"];
            $_SESSION["ime"] = $row["ime"];
            $_SESSION["email"] = $row["email"];
            header('Location: index.php');
        }
        else {
            header('Location: loginstrana.php');
        }
    }
?>
```

Na strani register (registerstrana.php) se nalazi forma sa poljima za email, ime, prezime, username i password korisnika.

```
<?php
    require_once ("header.php");
?>
<div class="container">
    <center>
        <form action="registracija.php" method="POST">
            <h2>Registrujte se</h2>
            <div class="col-50">
                <input type="text" name="email" placeholder="E-mail*" />
```

```
</div>
<div class="col-50">
    <input type="text" name="ime" placeholder="Ime*"/>
</div>
<div class="col-50">
    <input type="text" name="prezime" placeholder="Prezime*"/>
</div>
<div class="col-50">
    <input type="text" name="username" placeholder="Username*"/>
</div>
<div class="col-50">
    <input type="password" name="password"
placeholder="Password*"/>
</div>
<input type="submit" value="Register"/>
<br/>
<br/>
<a href="loginstrana.php">Već posedujete nalog? Idite na
login</a>
</form>
</center>
</div>
</body>
</html>
```

Ova forma poziva register.php skriptu koja uzima podatke iz forme i unosi ih u tabelu korisnik.

```
<?php
    require_once('konekcija.php');
    $con = povezi();

    if(isset($_POST['ime']) && isset($_POST['prezime']) &&
    isset($_POST['username']) && isset($_POST['password']) && isset($_POST['email'])){
        $username = $_POST['username'];
        $password = $_POST['password'];
        $password = md5($password);
        $ime = $_POST['ime'];
        $prezime = $_POST['prezime'];
        $email = $_POST['email'];

        $stmt = $con->prepare("INSERT INTO korisnik (username, password, ime,
        prezime, email) VALUES (:username, :password, :ime, :prezime, :email)");
        $stmt->bindParam(":username", $username);
        $stmt->bindParam(":password", $password);
        $stmt->bindParam(":ime", $ime);
        $stmt->bindParam(":prezime", $prezime);
        $stmt->bindParam(":email", $email);

        $stmt->execute();
        header('Location: loginstrana.php');
    }
?>
```

Pored login potrebno je kreirati i logout.php kako bi korisnik mogao da se izloguje i na taj način uništi sesija.

```
<?php
    session_start();
    session_destroy();
    header("Location:index.php");
?>
```

PRIKAZ LOGIN I REGISTER STRANE

Na slikama su prikazane strane login i register

Na slici 3 je prikaz login strane, a na slici 4 forma na strani register.



Slika 6.3 Login strana [Izvor: Autor]



Slika 6.4 Register strana [Izvor: Autor]

STRANA SA FORMOM ZA DODAVANJE NOVE AUKECIJE

Treba kreirati stranu za dodavanje novih aukcija od strane korisnika

Ulogovani korisnici imaju mogućnost da dodaju nove aukcije. Na početku ove strane se prvo proverava pomoću sesije da li je korisnik ulogovan.

```
<?php
    require_once("header.php");
    if(!isset($_SESSION["username"])){

```

```

        header('Location: index.php');
        exit(0);
    }
?>
<div class="container">
    <center>
        <form action="dodajAukciju.php" method="POST">
            <h2>Dodaj novu aukciju</h2>
            <div class="col-50">
                <input type="text" name="naziv" placeholder="Naziv"/>
            </div>
            <div class="col-50">
                <textarea name="opis" placeholder="Opis artikla"></textarea>
            </div>
            <div class="col-50">
                <input type="text" name="cena" placeholder="Minimalna
cena"/>
            </div>
            <div class="col-50">
                <input type="text" name="trajanje" placeholder="Trajanje"/>
            </div>
            <div class="col-50">
                <input type="text" name="slika" placeholder="Slika
artikla"/>
            </div>
            <?php
                echo "<input type='hidden' name='korisnikId' value='"
                    . $id
                    . "' />";
            ?>
                <input type="submit" value="Dodaj"/>
            </form>
        </center>
    </div>
</body>
</html>
```

Prethodna forma poziva skriptu dodajAukciju.php koja podatke iz forme upisuje u bazu podataka. U nastavku se nalazi kod za dodavanje aukcije.

```

<?php
    require_once('konekcija.php');
    $con = povezi();

    $naziv = $_POST["naziv"];
    $opis = $_POST["opis"];
    $slika = $_POST["slika"];
    $cena = $_POST["cena"];
    $trajanje = $_POST["trajanje"];
    $korisnikId = $_POST["korisnikId"];

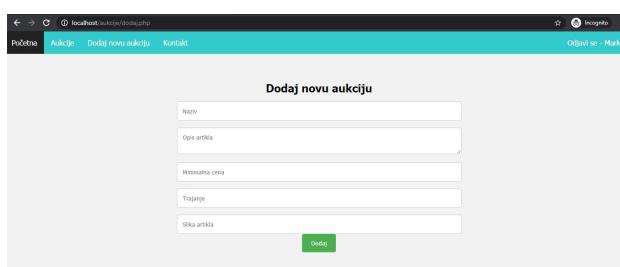
    if(isset($_POST['naziv']) && isset($_POST['opis']) && isset($_POST['cena']) &&
    isset($_POST['trajanje'])){
        $stmt = $con->prepare("INSERT INTO artikl (naziv, opis, slika, cena,
```

```

trajanje, korisnik_id) VALUES (:naziv, :opis, :slika, :cena, :trajanje,
:korisnik_id);
$stmt->bindParam(":naziv", $naziv);
$stmt->bindParam(":opis", $opis);
$stmt->bindParam(":slika", $slika);
$stmt->bindParam(":cena", $cena);
$stmt->bindParam(":trajanje", $trajanje);
$stmt->bindParam(":korisnik_id", $korisnikId);

$stmt->execute();
}
header('Location: dodaj.php');
?>

```



Slika 6.5 Forma za dodavanje aukcije [Izvor: Autor]

STRANA SA AUKEIJAMA

Strana sa aukcijama treba da sadrži listu trenutnih aukcija

Ova strana treba da preuzme podatke iz baze i prikaže ih na stranici. Kako bi to postigli konektujemo se na bazu, postavljamo upit za podatke iz tabele artikli i potom te podatke pomoću while petlje prikazujemo na strani.

SQL upit sadrži JOIN kako bi mogli da povežemo tabele i izvučemo podatke koji su nam potrebni.

```

<?php
require_once("header.php");
require_once('konekcija.php');
$con = povezi();

$sql = "SELECT * FROM artikl JOIN korisnik ON artikl.korisnik_id =
korisnik.korisnik_id";
$result = $con->query($sql);
echo "<center><h2 style='margin-bottom: 15px'>Trenutne aukcije</h2></center>";
while ($row = $result->fetch()) {
    echo "<div id='pattern' class='pattern'>
        <ul class='list img-list'>
            <li>
                <div class='li-img'>
                    <img src='".$row["slika"]."' alt='".$row["naziv"]."'
                </div>
            </li>
        </ul>
    </div>";
}
?>

```

```

        </div>
        <div class='li-text'>
            <h4 class='li-head'>" . $row["naziv"] ."</h4>
            <p class='li-sub'> Minimalna cena: " . $row["cena"] . "
$ /p>
            <p class='li-sub'> Trajanje aukcije jos: " .
$row["trajanje"] . " dana</p>
            <p class='li-sub'> Aukciju dodao/la: " . $row["ime"] .
" " . $row["prezime"] . "</p>
            <a href='detail.php?id=" . $row["artikl_id"] .
"'>Detaljnije o ovoj aukciji</a>
        </div>
    </li>
</ul>
</div>
</div>";
}
?>
</body>
</html>

```

Na slici 6 je prikaz strane sa aukcijama.



Slika 6.6 Stranica sa aukcijama [Izvor: Autor]

Svaka aukcija ima link ka detaljima o toj aukciji.

U nastavku se nalazi kod detail strane.

```

<?php
require_once("header.php");
require_once('konekcija.php');
$con = povezi();
$idArtikla = $_GET["id"];

$stmt = $con->prepare("SELECT * FROM artikl JOIN korisnik ON artikl.korisnik_id =
korisnik.korisnik_id WHERE artikl_id=:id");
$stmt->bindParam(":id", $idArtikla);
$stmt->execute();
echo "<center>";
while ($data = $stmt->fetch()) {
    echo "<h1>" . $data["naziv"] . "</h1><br/>" . "<img src='"
. $data["slika"] .
" width='25%' height='25%' /><br/> Kratak opis: <br/>" . $data["opis"] . "
<br/> Minimalna cena: " . $data["cena"] . " € <br/> Trajanje aukcije: " .
$data["trajanje"] . " dan/a <br/> Postavio/la: " . $data["ime"] . " " .
$data["prezime"] . "<br/> Kontakt mail: " . $data["email"];
    if($id == $data["korisnik_id"]){

```

```

        echo "<br/><a href='#'>Izmeni</a><br/><a href='#'>Izbrisni</a>";
    }
    else{
        echo "<br/><a href='ponudi.php?getid=" . $data["artikl_id"]
." '>Ponudi</a>";
    }
}

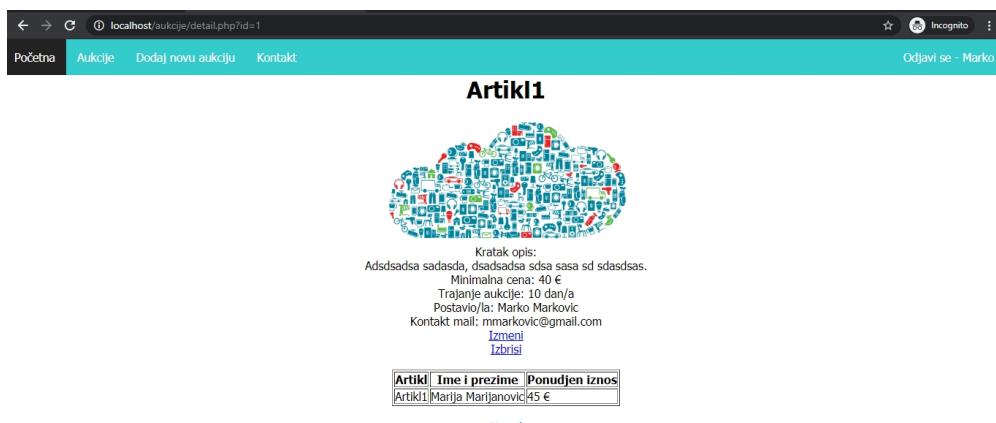
$stmt = $con->prepare("SELECT * FROM ponuda JOIN artikl ON ponuda.artikl_id =
artikl.artikl_id JOIN korisnik ON ponuda.korisnik_id = korisnik.korisnik_id WHERE
ponuda.artikl_id=:id");
$stmt->bindParam(":id", $idArtikla);
$stmt->execute();
echo "<center><table border='1'><th>Artikl</th><th>Ime i
prezime</th><th>Ponudjen iznos</th></th>";
while ($row = $stmt->fetch()) {
    echo "<tr> <td>" . $row["naziv"] . " </td> <td> " . $row["ime"] . " " .
$row["prezime"] . " </td><td> " . $row["iznos"] . " € </td></tr><br/>";
}
echo "</table>";
echo "<br/><a href='aukcije.php'>Nazad</a></center>";
?>
</body>
</html>

```

DETAIL STRANA

Svaka aukcija ima detail stranu na kojoj se nalaze detaljniji podaci

Na detail strani se nalaze detaljni podaci odabrane aukcije. Takođe, ispod podataka o aukciji, nalazi se tabela sa njenim trenutnim ponudama. Na stranici detail postoje opcije za izmenu i brisanje ili za davanje ponude na odabranu aukciju. Ukoliko je trenutno ulogovani korisnik kreirao odabranu aukciju, na stranici se postoje opcije za izmenu i brisanje ponude. Ako ulogovani korisnik nije kreirao odabranu aukciju onima opciju ponudi.



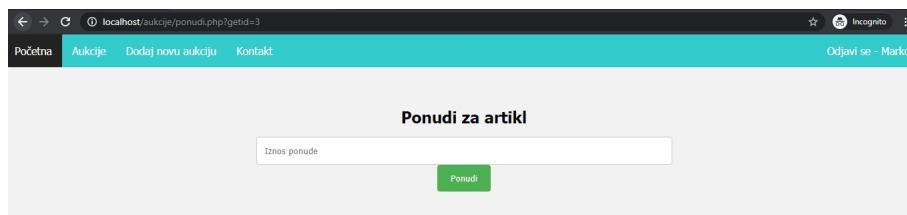
Slika 6.7 Prikaz detail strane aukcije [Izvor: Autor]

POSTAVLJANJE PONUDE ZA AUKCIJU

Za odabranu aukciju može se postaviti ponuda

Ukoliko korisnik želi da da ponudu na aukciju otvara se strana sa formom u koju se unosi ponuda, odnosno, iznos koji korisnik nudi. Podatke o tome koji korisnik nudi ponudu se uzimaju iz sesije, a pomoću HTTP-a GET zahteva se prosledjuje ID artikla za koji se nudi ponuda.

Na slici 8 je prikazana forma za unos ponude.



Slika 6.8 Strana za unos ponude [Izvor: Autor]

```
<?php
    require_once("header.php");
    if(!isset($_SESSION["username"])){
        header('Location: loginstrana.php');
        exit(0);
    }
    else{
        $getid = $_GET["getid"];
    }
?>
<div class="container">
    <center>
        <form action="dodajPonudu.php" method="POST">
            <h2>Ponudi za artikl</h2>
            <div class="col-50">
                <input type="text" name="iznos" placeholder="Iznos ponude"/>
            </div>
            <?php
                echo "<input type='hidden' name='korisnikId' value='". $_SESSION['korisnikId']."' />";
                echo "<input type='hidden' name='artiklId' value='". $getid ."' />";
            <?>
                <input type="submit" value="Ponudi"/>
            </form>
        </center>
    </div>
</body>
</html>
```

```

<?php
    require_once('konekcija.php');
    $con = povezi();

    $iznos = $_POST["iznos"];
    $artiklId = $_POST["artiklId"];
    $korisnikId = $_POST["korisnikId"];

    if(isset($_POST['iznos']) && isset($_POST['artiklId']) &&
    isset($_POST['korisnikId'])){
        $stmt = $con->prepare("INSERT INTO ponuda (korisnik_id, artikl_id, iznos)
VALUES (:korisnik_id, :artikl_id, :iznos)");
        $stmt->bindParam(":iznos", $iznos);
        $stmt->bindParam(":artikl_id", $artiklId);
        $stmt->bindParam(":korisnik_id", $korisnikId);

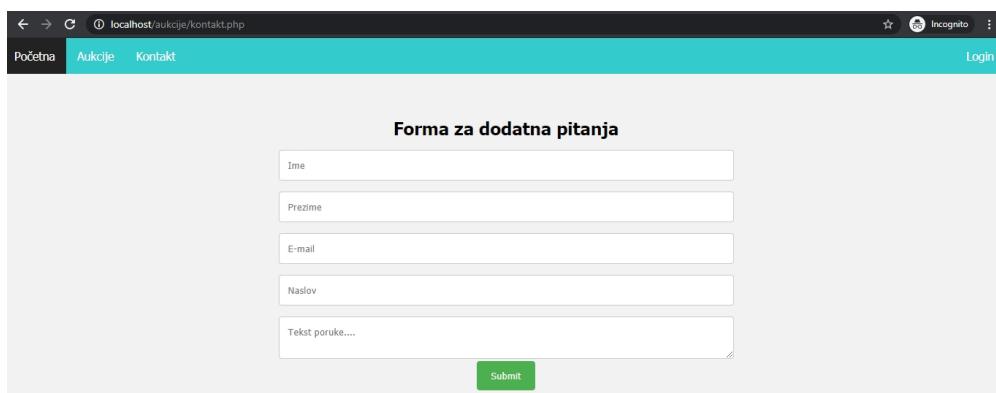
        $stmt->execute();
    }
    header('Location: aukcije.php');
?>

```

KONTAKT STRANA

Kontakt strana sadrži formu za slanje dodatnih pitanja

Kontakt strana sadrži formu za slanje dodatnih pitanja. Na slici 9 je prikaz ove strane, a u nastavku i kod.



Slika 6.9 Strana kontakt [Izvor: Autor]

```

<?php
    require_once ("header.php");
?>
    <div class="container">
        <center>
            <form>
                <h2>Forma za dodatna pitanja</h2>
                <div class="col-50">
                    <input type="text" name="ime" placeholder="Ime"/>

```

```
</div>
<div class="col-50">
    <input type="text" name="prezime" placeholder="Prezime"/>
</div>
<div class="col-50">
    <input type="text" name="email" placeholder="E-mail"/>
</div>
<div class="col-50">
    <input type="text" name="naslov" placeholder="Naslov"/>
</div>
<div class="col-50">
    <textarea name="poruka" placeholder="Tekst
poruke...."></textarea>
</div>
<input type="submit" value="Submit"/>
</form>
</center>
</div>
</body>
</html>
```

STILIZOVANJE STRANA

Strane su stilizovane pomoću CSS-a

```
/* Start General */
* {
margin:0;
padding:0;
font-family: Tahoma;
box-sizing:border-box;
-o-box-sizing:border-box;
-ms-box-sizing:border-box;
-moz-box-sizing:border-box;
-webkit-box-sizing:border-box;
}
body{
background-color: white;
}
nav ul li a {
color: #FFF;
text-decoration: none;
}
/* End General */

/* Start Navbar */
nav ul {
background-color: #33cccc;
}
nav ul > li {
```

```
padding: 15px;
display: inline-block;
transition: all .3s ease;
margin-left: -5px
}
nav ul > li:not(:last-of-type):hover {
background-color: #222
}
nav ul > li:first-of-type {
background-color: #222
}
nav ul > li:last-of-type {
float: right;
}
nav ul > li:last-of-type a .fa {
font-size: 21px
}
/* End Navbar */

/* Start Menue Right */
nav ul > ol {
position: absolute;
top: 49px;
right: 0;
background: #333;
text-align: center;
list-style: none;
display: none
}
nav ul > ol > li {
width: 100vw;
color: #FFF;
margin: 0;
padding: 0;
padding-top: 10px;
padding-bottom: 10px;
transition: all .3s ease;
}
nav ul > ol > li:hover a {
margin: 20px;
}
nav ul > ol > li:hover {
background: #000;
cursor: pointer
}
nav ul input {
opacity: .7;
padding: 5px;
float: right;
display: none
}
/* Start Menue Right */
```

```
/* Start Media Query */
@media screen and (max-width:680px){
    nav ul > li:not(:first-child) {
        display:none
    }
    nav ul > li:last-of-type {
        display: block
    }
    nav ul input {
        display: inline
    }
}
@media screen and (min-width:680px) {
    nav ul > ol > li {
        display:none
    }
}
/* End Media Query */

input[type=text], select, textarea {
    width: 100%;
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    resize: vertical;
}

input[type=password], select, textarea {
    width: 100%;
    padding: 12px;
    border: 1px solid #ccc;
    border-radius: 4px;
    resize: vertical;
}

input[type=submit] {
    background-color: #4CAF50;
    color: white;
    padding: 12px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

input[type=submit]:hover {
    background-color: #45a049;
}

.container {
    border-radius: 5px;
    background-color: #f2f2f2;
    padding: 60px;
}
```

```
.col-50 {  
    width: 50%;  
    margin-top: 15px;  
}  
  
/* Clear floats after the columns */  
.row:after {  
    content: "";  
    display: table;  
    clear: both;  
}  
  
.mySlides {  
    display: none  
}  
  
/* Slideshow container */  
.slideshow-container {  
    max-width: 1000px;  
    position: relative;  
    margin: auto;  
}  
  
/* Next & previous buttons */  
.prev, .next {  
    cursor: pointer;  
    position: absolute;  
    top: 50%;  
    width: auto;  
    padding: 16px;  
    margin-top: -22px;  
    color: white;  
    font-weight: bold;  
    font-size: 18px;  
    transition: 0.6s ease;  
    border-radius: 0 3px 3px 0;  
    user-select: none;  
}  
  
/* Position the "next button" to the right */  
.next {  
    right: 0;  
    border-radius: 3px 0 0 3px;  
}  
  
/* On hover, add a black background color with a little bit see-through */  
.prev:hover, .next:hover {  
    background-color: rgba(0,0,0,0.8);  
}  
  
/* Caption text */  
.text {
```

```
color: #f2f2f2;
font-size: 15px;
padding: 8px 12px;
position: absolute;
bottom: 8px;
width: 100%;
text-align: center;
}

/* Number text (1/3 etc) */
.numbertext {
    color: #f2f2f2;
    font-size: 12px;
    padding: 8px 12px;
    position: absolute;
    top: 0;
}

/* The dots/bullets/indicators */
.dot {
    cursor: pointer;
    height: 15px;
    width: 15px;
    margin: 0 2px;
    background-color: #bbb;
    border-radius: 50%;
    display: inline-block;
    transition: background-color 0.6s ease;
}

.active, .dot:hover {
    background-color: #717171;
}

/* Fading animation */
.fade {
    -webkit-animation-name: fade;
    -webkit-animation-duration: 1.5s;
    animation-name: fade;
    animation-duration: 1.5s;
}

@-webkit-keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

@keyframes fade {
    from {opacity: .4}
    to {opacity: 1}
}

/* On smaller screens, decrease text size */
```

```
@media only screen and (max-width: 300px) {  
    .prev, .next,.text {font-size: 11px}  
}  
  
.list li {  
    border-bottom: 1px solid #ccc;  
    display: table;  
    border-collapse: collapse;  
    width: 100%;  
}  
.inner {  
    display: table-row;  
    overflow: hidden;  
}  
.li-img {  
    display: table-cell;  
    vertical-align: middle;  
    width: 50%;  
    padding-right: 1em;  
}  
.li-img img {  
    display: block;  
    width: 100%;  
    height: auto;  
}  
.li-text {  
    display: table-cell;  
    vertical-align: middle;  
    width: 70%;  
}  
.li-head {  
    margin: 0;  
}  
.li-sub {  
    margin: 0;  
}  
  
@media all and (min-width: 45em) {  
    .list li {  
        float: left;  
        width: 50%;  
    }  
}  
  
@media all and (min-width: 75em) {  
    .list li {  
        width: 33.33333%;  
    }  
}
```

▼ Poglavlje 7

Zadaci za samostalni rad: PHP APLIKACIJE 2

ZADATAK 1 - INTERNET AUKEIJA

Iskoristiti prethodni primer i dodati nove funkcionalnosti

Očekivano vreme izrade zadatka: 30 minuta.

Zadatak

Nadograditi prethodno objašnjen primer internet aukcije tako da se osposobe funkcije za izmenu i brisanje aukcije na detail strani.

ZADATAK 2 - TIMELINE

Klonirati Facebook sajt koji će samo imati Timeline.

Očekivano vreme izrade zadatka: 50 minuta.

- Potrebno je klonirati Facebook sajt koji će samo imati Timeline.
- Na prvoj strani se nalazi forma za registraciju i logovanje - podaci o registrovanim korisnicima se čuvaju u tabelu Korisnik. Nakon toga je potrebno kreirati Timeline.
- Timeline sadrži textbox za unos statusa, dugme za dodavanje slike, dugme za dodavanje mape (synchronizovati sa gmaps ili neki drugi servis).
- Ukoliko se pritisne dugme pošalji status se upisuje u bazu i otvara se naredna strana koja prikazuje taj status iščitavanjem iz baze.

✓ Poglavlje 8

DOMAĆI ZADATAK

OPIS DOMAĆEG ZADATKA - DZ15

Kreirati malu PHP aplikaciju po vašem izboru.

Očekivano vreme izrade zadatka: 45 minuta.

Kreirati malu PHP aplikaciju po vašem izboru.

Aplikacija se smatra kompletnom ako sadrži sledeće elemente:

- Dve vrste korisnika (admin i obične korisnike) i korisničke sesije
- Omogućiti login, register i logout
- Web forme za unos i izmenu podataka u bazi
- Web forme za pretragu baze ili master/detail navigaciju
- Admin korisnik treba da ima određene privilegije u odnosu na običnog korisnika
- Otporna je na SQL Injection napade i pogrešno korišćenje

Potrebne datoteke snimiti pod imenom:

IT210-DZ15-Ime_Prezime_brojIndexa

gde su Ime, Prezime i broj indeksa vaši podaci. Tako imenovane datoteke poslati na adresu predmetnog asistenta.

(napomena: u Subject-u e-mejla napisati IT210 - DZ15)

▼ ZAKLJUČAK

ZAKLJUČAK

U ovom predavanju:

- Definisana je integracija u funkciji komponenata i interfejsa
- Dat je primer platforme srednjeg sloja i navedene su prednosti i nedostaci
- Prikazani su neki najvažniji aspekti izbora integracionih platformi organizacija
- Dati su primjeri integracije korišćenjem **wrapper** i **glue code** pristupa
- Objasnjeno je kako **data warehouse** koncept utiče na integraciju informacija organizacije

Literatura

1. Humphries Hawkins, Data Warehousing Architecture and Implementation, Prentice Hall PTR, 1998.