Spletne tehnologije, UL, FRI (VSP)
ST 9 - Spletno načrtovanje-arhitektura

doc.dr. Mira Trebar

Uvod

- Programska arhitektura
  - Kaj je programska arhitektura?
  - Osnovni principi
  - Arhitekturni vzorci in stili
  - Arhitekturne tehnike in snovanje
- Načrtovanje spletnih rešitev

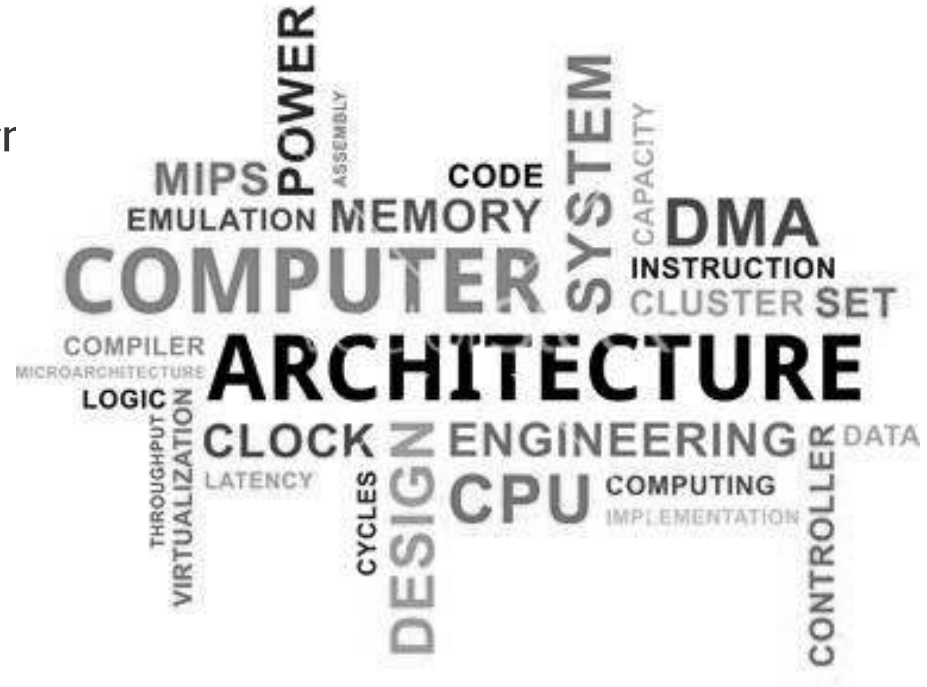
- Literatura:
  - Microsoft Application Architecture Guide, Microsoft, 2009 (Poglavja: 1, 4, 5, 21)
    https://msdn.microsoft.com/en-us/library/ff650706.aspx

Uvod

- Programska arhitektura

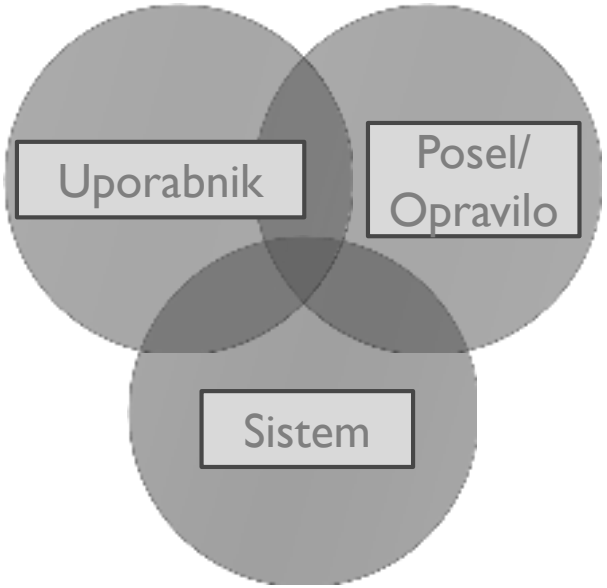
- Razvijalci – izdelava aplikacij:
  - Okolje Microsoft (Microsoft platforr
  - Ogrodje .NET

- Aplikacijska arhitektura in snovanje:
  - Funkcionalnosti se delijo na:
    - Nivoji
    - Komponente
    - Storitve



Programska arhitektura

- je proces za določanje strukturirane rešitve, ki zadošča
  - tehničnim in operativnim zahtevam, medtem ko
  - optimizira običajne atribute kvalitete.
- vključuje odločitve na osnovi različnih faktorjev, ki vplivajo na:
  - kvaliteto,
  - zmogljivost,
  - vzdržljivost in
  - celoten uspeh aplikacije.
- Kompleksna struktura
- Slaba arhitektura - tveganja:
  - Napačno ocenjeni problemi
  - Napačne dolgoročne odločitve
- Uporabnik, sistem, poslovni cilji

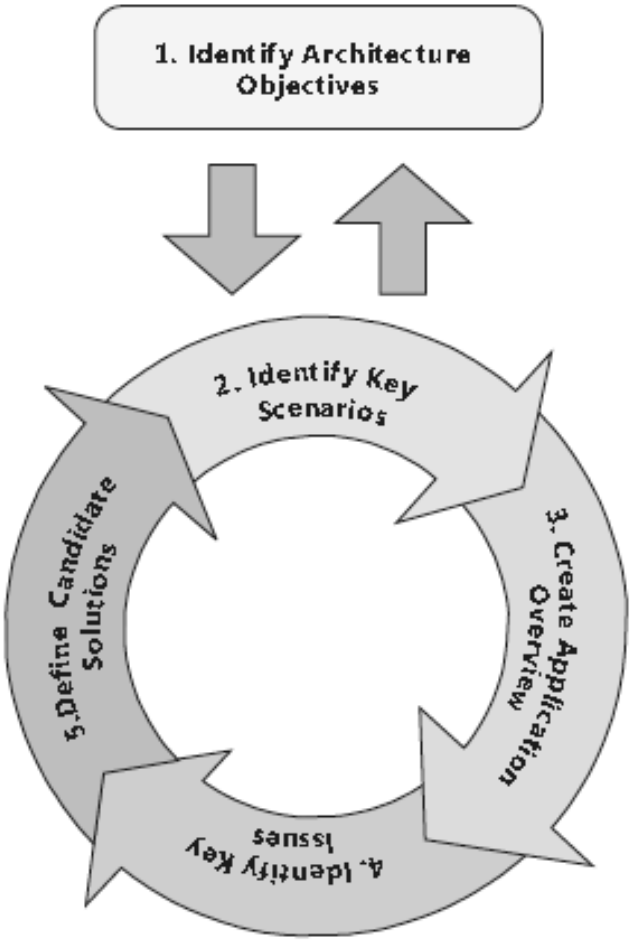


- Kako bo uporabnik uporabljal aplikacijo?
- Kako bo aplikacija umeščena v proizvodnjo in kako bo potekalo njeno upravljanje?
- Kaj so zahteve kvalitetnih atributov za aplikacijo?
  - Varnost
  - Zmogljivost
  - Skladnost/Koordiniranost
  - Internacionalizacija
  - Konfiguracija
- Kako bo aplikacija načrtovana, da bo fleksibilna in jo bo mogoče vzdrževati skozi čas?
- Kakšni so trendi v arhitekturah, ki bi lahko vplivali na aplikacijo zdaj ali pa potem ko bo nameščena?

- Cilji:
  - Prikazati strukturo sistema, skriti izvedbene podrobnosti
  - Realizirati vse interakcije s sistemom (use case) in primere uporabe (scenarios)
  - Upoštevati zahteve vseh različnih deležnikov
  - Obravnavati vse funkcionalne in kvalitativne zahteve
- Arhitekturna ureditev
  - Uporabnikovo odločanje
  - Zrelost tržišča
  - Fleksibilna sestava
  - Trendi prihodnosti
- Principi arhitekturne zasnove/sestave
  - Zgraditi in omogočiti spremembe
  - Modelirati rešitev za analizo in zmanjšati tveganje
  - Uporabiti modele, vizualizacijo kot orodje za komunikacijo in sodelovanje
  - Identifikacija glavnih inženirskih odločitev

Arhitektura in snovanje

- Iterativna tehnika za načrtovanje arhitekture
- Vhodi in izhodi:
  - Primeri uporabe (use case)
  - Scenarij uporabe (usage scenario)
  - Funkcionalne zahteve
  - Ne-funkcionalne zahteve (zmogljivost, varnost, zanesljivost)
  - Tehnološke zahteve
  - Namestitveno okolje
- Koraki:
  1. "Identify Architecture Objectives"
  2. " Key Scenarios"
  3. " Application Overview "
  4. " Key Issues."
  5. " Candidate Solutions."

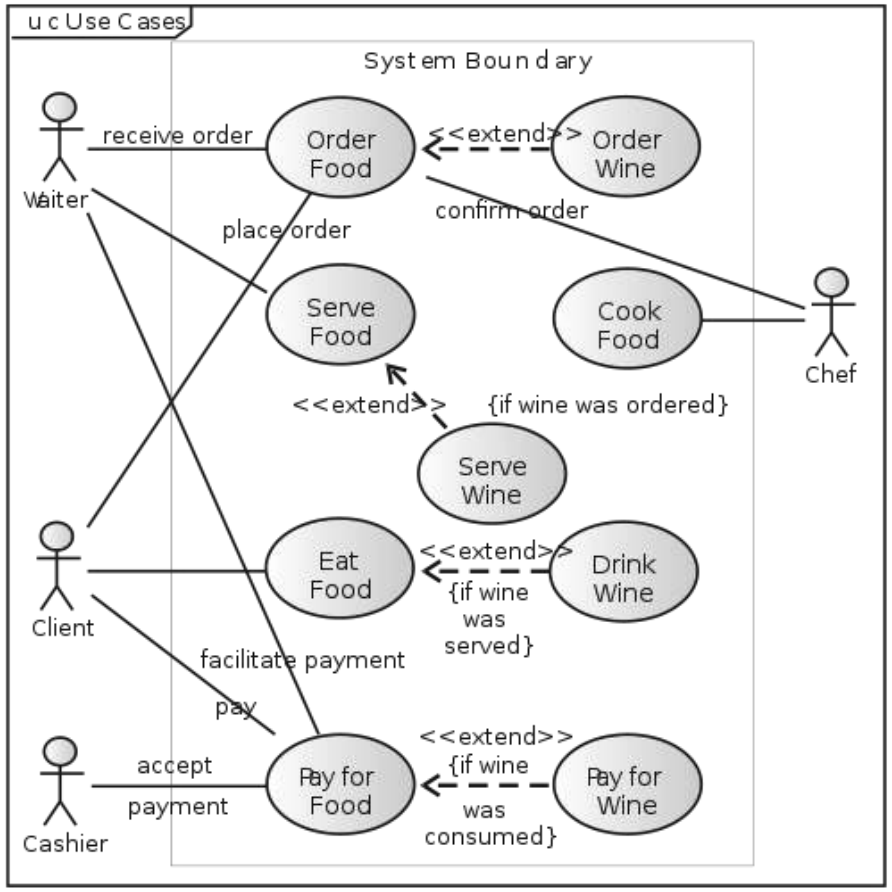


1. Identifikacija arhitekturnih ciljev (Identify Architecture Objectives)

- Naloge in omejitve, ki oblikujejo:
  - arhitekturo in proces razvoja,
  - okvir delovanja in
  - določajo kdaj bomo končali.
- Identifikacija ciljev arhitekture:
  - Izdelava prototipa (DA/NE)
  - Testiranje možnih poti
  - Arhitekturni proces nove aplikacije
- Komu je namenjena?
  - Drugim načrtovalcem/arhitektom, razvijalcem
  - Operativno osebje
  - Vodstvo, uprava
- Identifikacija omejitev
  - Tehnologije
  - Omejitve uporabe
  - Omejitve namestitve
- Določitev časovnega obsega aktivnosti (število v dan/mesec/leto)

2. Ključni scenariji (Key Scenarios)

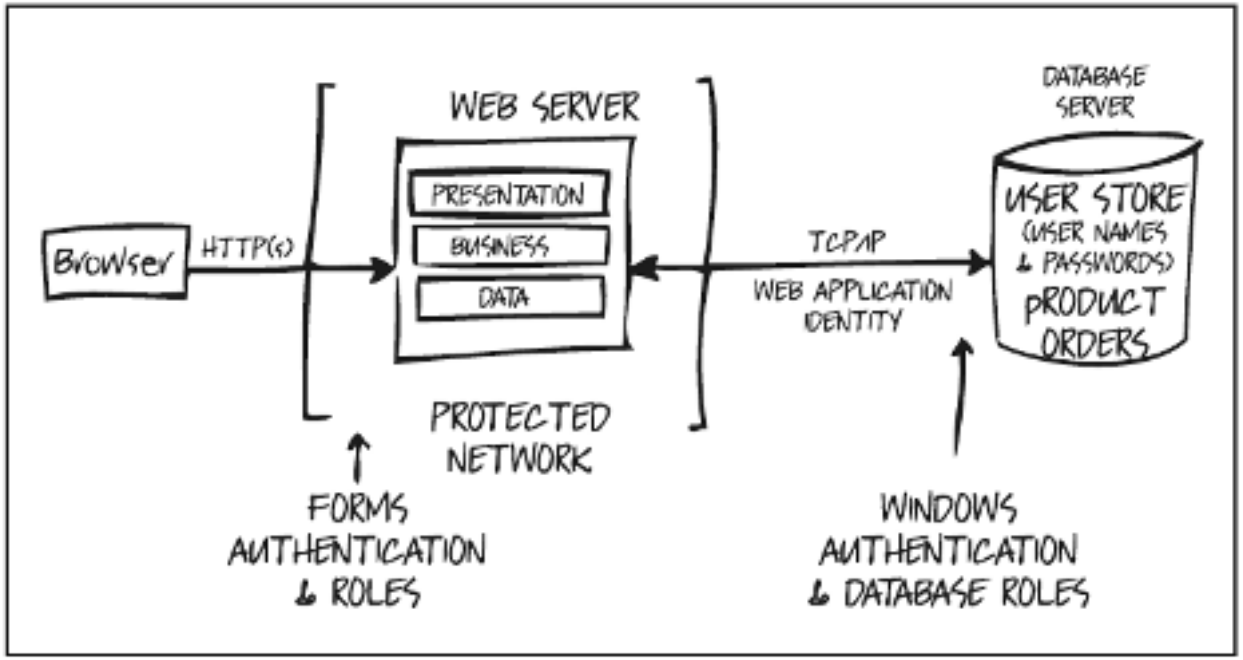
- Primer uporabe (Use case) – opis interakcij med sistemom in uporabnikom ali drugim sistemom.
- Scenarij (Scenario) – širši in bolj obsežen opis uporabnikove interakcije s sistemom.
- Identificirati različne ključne scenarije za določitev arhitekture:
  - avtentikacija uporabnika- povezava med atributom kvalitete (varnost) in uporabnostjo (kako se uporabnik prijavi)
  - UML diagram: Restavracija



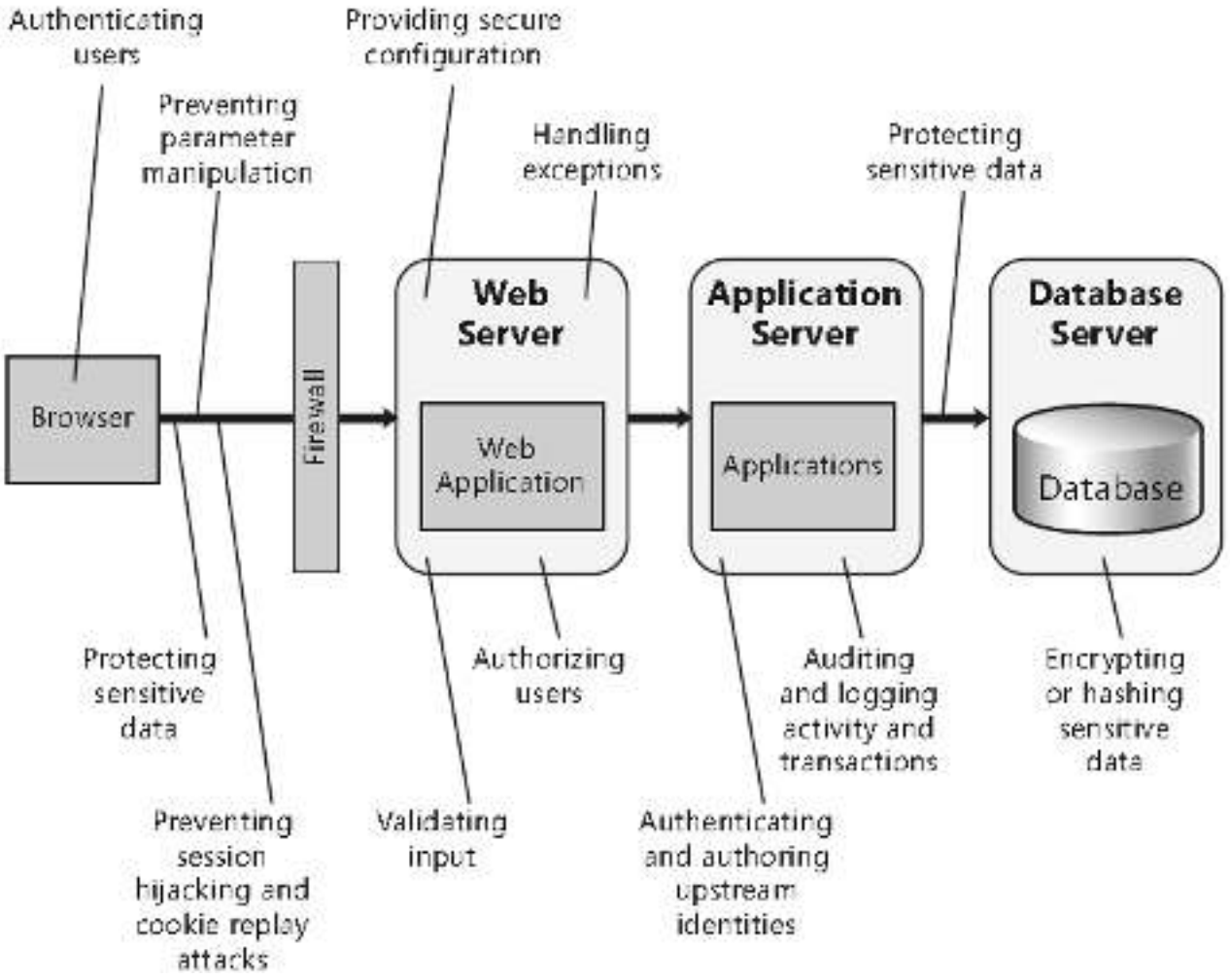
http://www.modelio.org/tutorials/how-to-create-uml-use-case-diagram.html

3. Pregled aplikacije (Application Overview)

- Izgled aplikacije in povezava z realnim svetom
- Tip aplikacije: mobilna, storitev, spletna, kombinacija
- Namestitvene omejitve – infrastruktura: varnost, zanesljivost, protokoli, ...
- Izbira stila: Client/Server, nivojska, SOA, ...
- Relevantne tehnologije
- Shema arhitekture



4. Ključna vprašanja (Key Issues)



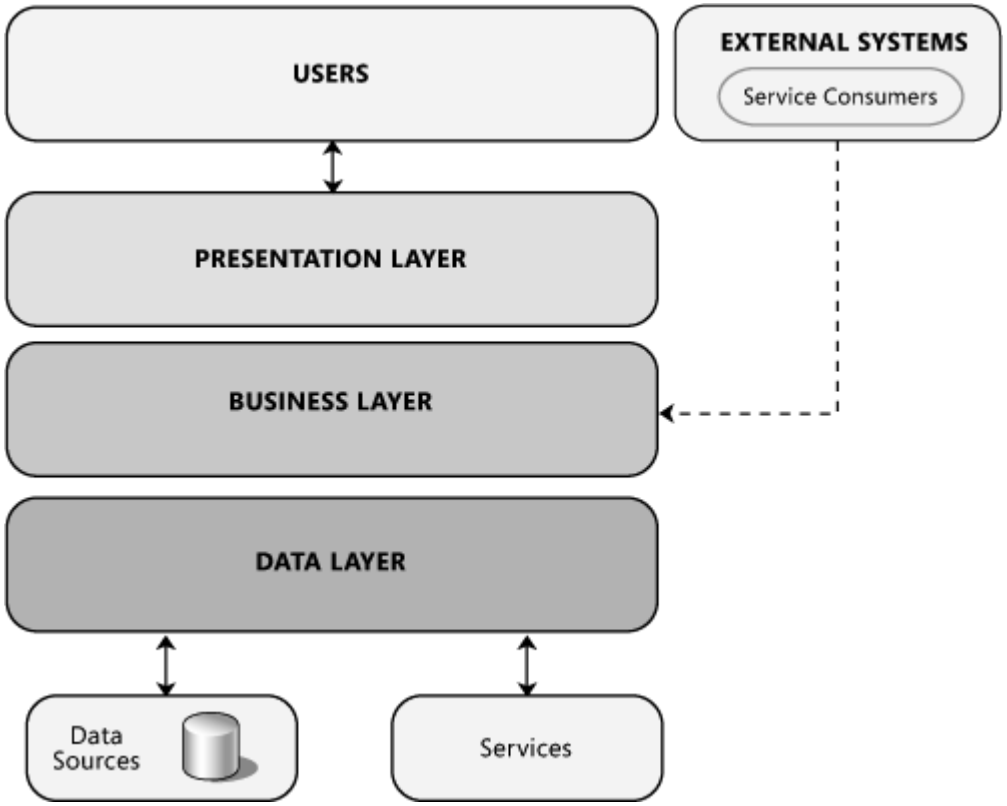
5. Različica rešitve (Candidate Solutions)

- Kreiranje osnovne izvedbene arhitekture (‘baseline‘ arhitektura)
- Dodajanje podrobnosti za izdelavo arhitekture pred končno izdajo (‘candidate‘ arhitektura)
- Pregled posameznih področij s testnimi implementacijami
- Pregled veljavnosti glede na ključne scenarije in zahteve
- ‘Baseline‘ arhitektura:
  - obstoječ sistem – to je takšen izgled sistema, kot ga vidimo danes
  - če je arhitektura nova – začetna ‘baseline‘ je prva arhitekturna zasnova iz katere bo zgrajena ‘candidate‘ arhitektura.
- ‘Candidate‘ arhitektura vključuje:
  - tip aplikacije in postavitveno arhitekturo,
  - arhitekturni stil,
  - izbiro tehnologij,
  - atribute kvalitete in druge povezave s sistemom (cross-cutting).

Nivojska arhitektura

- Logično združevanje komponent v nivoje (layers)

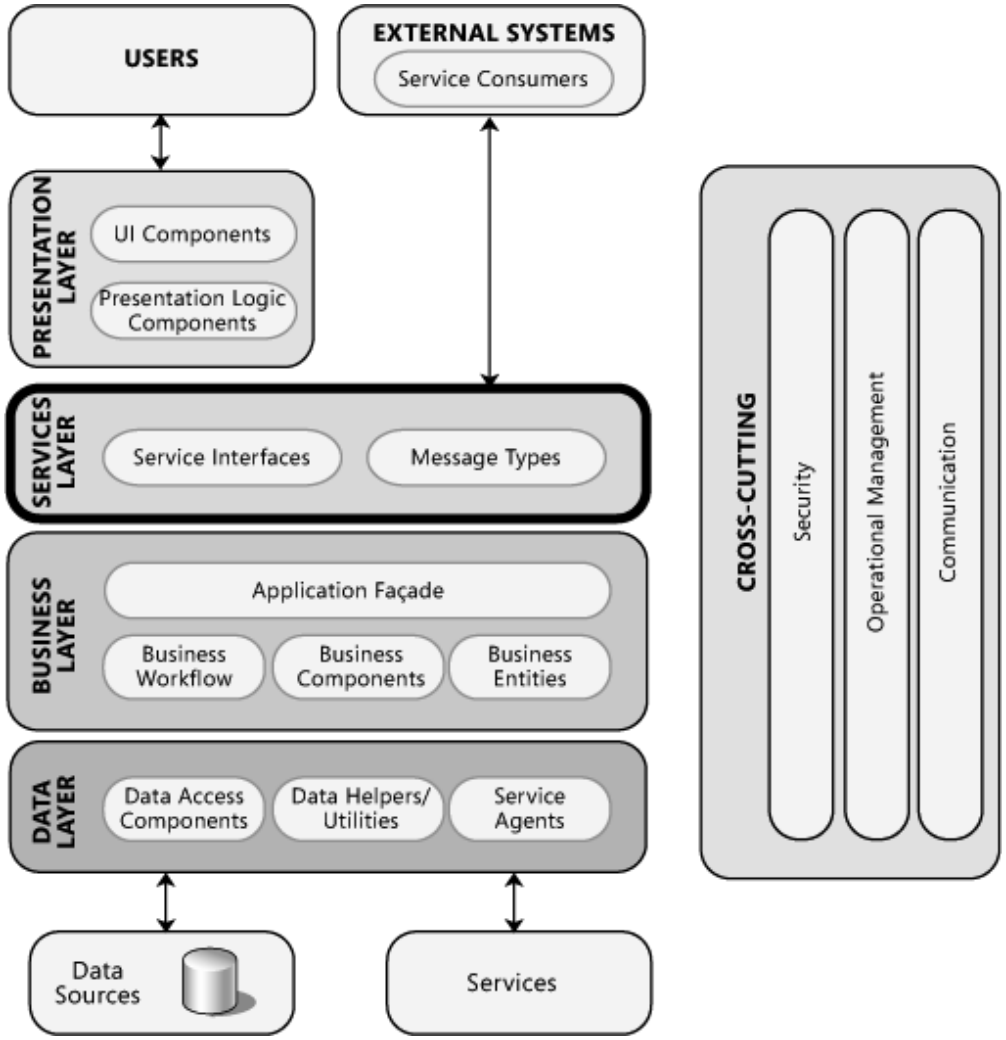
- Predstavitveni nivo
  - interakcija uporabnika s sistemom
- Poslovni nivo:
  - funkcionalnosti sistema
  - storitve drugih aplikacij
- Podatkovni nivo:
  - dostop do podatkov v sistemu
  - dostop do podatkov drugih sistemov v omrežju preko zunanjih storitev



http://msdn.microsoft.com/en-us/library/ee658109.aspx

Osnovna aplikacijska arhitektura

- Uporabniki
- Predstavitveni nivo
  - Uporabniški vmesniki
  - Komponente predst. logike
- Zunanji sistemi
  - Storitve potrošnikov
- Storitveni nivo
  - Storitveni vmesniki
  - Tipi sporočil
- Poslovni nivo
  - Aplikacijska zunanjost
- Podatkovni nivo
  - Podatkovni viri
  - Storitve
- Cross-Cutting



http://msdn.microsoft.com/en-us/library/ee658109.aspx#Step1

Razvojni koraki nivojske strukture

1. Izbrati nivojsko strategijo
2. Določiti zahtevane nivoje
3. Odločitev o porazdelitvi nivojev in komponent
4. Določiti zmanjševanje, sproščanje nivojev
5. Določiti pravila za interakcijo med nivoji
6. Identifikacija funkcionalnosti, ki povezujejo nivoje
7. Definicija vmesnikov med nivoji
8. Izbira razvojne strategije
9. Izbira komunikacijskih protokolov

Načrtovanje spletne aplikacije

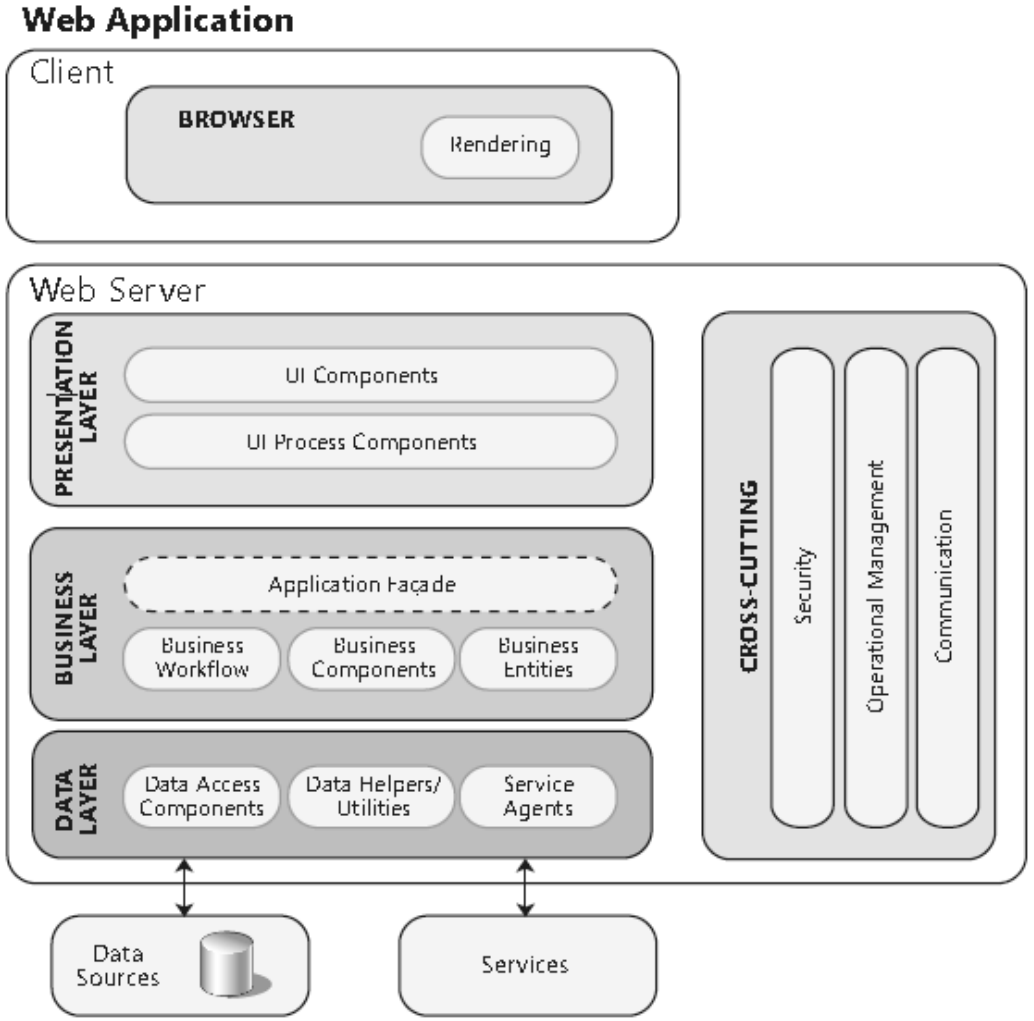


Figure 1
The typical structure of a Web application

Posebna vprašanja načrtovanja

- Procesiranje aplikacijske zahteve (Application Request Processing)
- Avtentikacija (Authentication)
- Avtorizacija (Authorization)
- Predpomnjenje (Caching)
- Upravljanje izjem (Exception Management)
- Beleženje in nadzorovanje (Logging and Instrumentation)
- Navigacija (Navigation)
- Razporeditev strani (Page Layout)
- Upodabljanje strani (Page Rendering)
- Upravljanje sej (Session Management)
- Preverjanje (Validation)

- PROCESIRANJE ZAHTEVE (Application Request Processing)
  - 'post back approach':
    - omogoča večji nadzor nad aplikacijo na strani strežnika
    - primeren za aplikacije, ki vsebujejo **veliko spletnih obrazcev**
  - 'RESTful service approach':
    - omogoča večji nadzor nad uporabniškim vmesnikom
    - zagotavlja večjo prilagodljivost pri testiranju in navigaciji
- Načrtovanje spletne aplikacije - logika procesiranja zahtev in aplikacijska logika sta ločeni od uporabniškega vmesnika

- Dve programski zasnovi:
  - MVP (Model-View-Presenter):
    - pri načinu 'post back approach'
    - uporaba predvsem za gradnjo uporabniškega vmesnika
  - MVC (Model-View-Controller):
    - pri načinu 'RESTful service approach'
    - loči predstavitev informacij od interakcije z uporabnikom

- AVTENTIKACIJA (Authentication)
  - predstavlja varnost in zanesljivost aplikacije
  - neprimerna ali šibka avtentikacija lahko izpostavi aplikacijo različnim napadom (npr. kraja seje)
  - zagotavljanje varnosti: potek gesel, minimalna dolžina gesla, uporaba različnih znakov v geslu, zaklepanje računov itd.
  - gesla naj bodo v podatkovni bazi shranjena v kriptirani obliki, ne kot berljivo besedilo (plaintext)

- AVTORIZACIJA (Authorization)
  - zagotovitev varnosti in zanesljivosti aplikacije
  - neprimerna ali šibka avtorizacija lahko pomeni razkritje informacij, dostop do podatkov
  - pooblaščeni uporabniki morajo biti zanesljivi in vredni zaupanja
  - prava mera avtorizacije: preveč razdrobljena avtorizacija lahko upočasni upravljanje, premalo razdrobljena avtorizacija zmanjšuje prilagodljivost

- PREDPOMNJENJE (Caching)
  - shranjevanje podatkov izboljšuje in optimizira delovanje aplikacije
  - nepravilno shranjevanje vpliva na odzivnost aplikacije
  - pri implementaciji se je potrebno odločiti, kdaj naložiti podatke v predpomnilnik
  - shranjevanje le tistih podatkov, ki se redko spreminjajo (npr. statične spletne strani)
  - shranjevanje le kripriranih občutljivih podatkov
- UPRAVLJANJE Z IZJEMAMI (Exception management)
  - načrtovanje strategije upravljanja z izjemami zaradi varnosti in zanesljivosti aplikacije
  - pravilno upravljanje z izjemami pri spletnih straneh preprečuje uporabnikom prikaz podrobnejših podatkov o napaki
  - uporabniku prijazno sporočilo z obvestilom o napaki (lepša podoba aplikacije)
  - preprečevanje prikazovanja podatkov pri napakah na strani, napakah v sporočilih

- BELEŽENJE IN NADZOROVANJE (Loging and Instrumentation)
  - Zaznavanje napadov na vseh nivojih: kritični uporabniški in sistemski dogodki, kritične poslovne operacije, neobičajne aktivnosti.
  - Kreiranje politike varnega upravljanja z datoteko 'log': omejen dostop do datotek za beleženje, omogočanje branje/pisanje.
  - Ne shranjujte občutljive informacije v datoteke **log in audit**.

- NAVIGACIJA (Navigation)
  - Omogočiti uporabnikom enostaven pregled na zaslonu oz. na spletni strani.
  - Stran naj ima konsistentno strukturo zaradi zmanjšanja zmede in kompleksnosti.
  - Spletni obrazec 'post back approach' (uporabi MVP)
  - 'RESTful service approach' (uporabi MVC)
  - Vključiti navigacijo v glavno stran
  - Uporabiti zemljevid strani (pomoč uporabniku pri ogledu, brskalniku pri pregledu strani)

- RAZPOREDITEV STRANI (Page Layout)
  - Uporabite CSS, kjer je možno in tabele le tam, kjer je potrebno za prikaz podatkov.
  - Izogibajte se obsežnim stranem z veliko nalogami
  - Uporabite skupen izgled za strani, kjer je možno, za največjo dostopnost in enostavno uporabo
  - Ne mešajte skripte na odjemalčevi strani s HTML (ločeni datoteki)

- UPODABLJANJE STRANI (Page Rendering)
  - učinkovitost in največja uporabnost vmesnika
  - Uporabite skripto na odjemalčevi strani (ASP.NET AJAX) za boljšo uporabniško izkušnjo in boljšo odzivnost.
  - Povezovanje podatkov (zbirke, objekti DataReader, tabele DataSet)
  - Načrtovanje podpore za lokalizacijo v komponentah UI.

- UPRAVLJANJE SEJ (Session Management)
  - Shranjevanje stanja seje – uporaba stanja seje pomeni dodatek k vsaki zahtevi strani.
  - Zagotovite zadržanje podatkov seje, če je zahtevano.
  - Shranjevanje stanja seje poteka na drugem strežniku – zaščitite sejo s SSL (Secure Socket Layer) ali IPSec (Internet Protocol Security).

- PREVERJANJE (Validation)
  - Preveri vse podatke, ki prehajajo med zaupnimi mejami aplikacije. (vsi uporabnikovi podatki so škodljivi - se preverjajo)
  - Načrtuj strategijo za zavrnitev in saniranje zlonamernih vhodov.
  - Uporabi preverjanje strani odjemalca – vedno na strežniku.
  - Razišči rešitve druge strani, vzorce načrtovanja in knjižnic.

Pogoji načrtovanja za nivoje

- **Predstavitveni sloj** – omogoča prikaz uporabniškega vmesnika (UI) in olajšanje uporabnikove interakcije:
  - Ločitev interakcijske logike od komponent uporabniškega vmesnika
  - Sestoji iz:
    - Strežniških komponent (vrnejo HTML)
    - Odjemalčevih komponent (brskalnik izvaja skripte in prikaže HTML)

- **Poslovni sloj** – izboljša vzdrževanje in preizkušanje aplikacije:
  - Implementacija poslovne logike
  - Izvajanje delovnega toka
  - Centralizacija in ponovna uporaba skupnih poslovnih logičnih funkcij
  - Razvoj poslovnih entitet z realnimi podatki, ki se uporabijo za prenos podatkov med komponentami.

- **Podatkovni sloj -** logika za dostop do podatkovne baze
  - Omogoča enostavnejšo konfiguracijo in vzdrževanje.
  - Skrije podrobnosti podatkovne baze drugim nivojem.
  - Omogoča razvoj strategije za obravnavo izjem pri napakah ob dostopu podatkov in sporočanje izjem poslovnemu nivoju.

- **Storitveni sloj**
  - Namestitev poslovnega sloja na oddaljenem nivoju
  - Odprtje poslovne logike za uporabo spletnih storitev
  - Razvoj storitev za dosego večkratne uporabnosti brez posebnih podrobnosti odjemalcev, ki jih bodo uporabljali.
  - Ne vključuje poslovnih funkcij
  - Zahteve intoperabilnosti z izbiro protokolov in prenosnih mehanizmov

Testiranje in pogoji testabilnosti

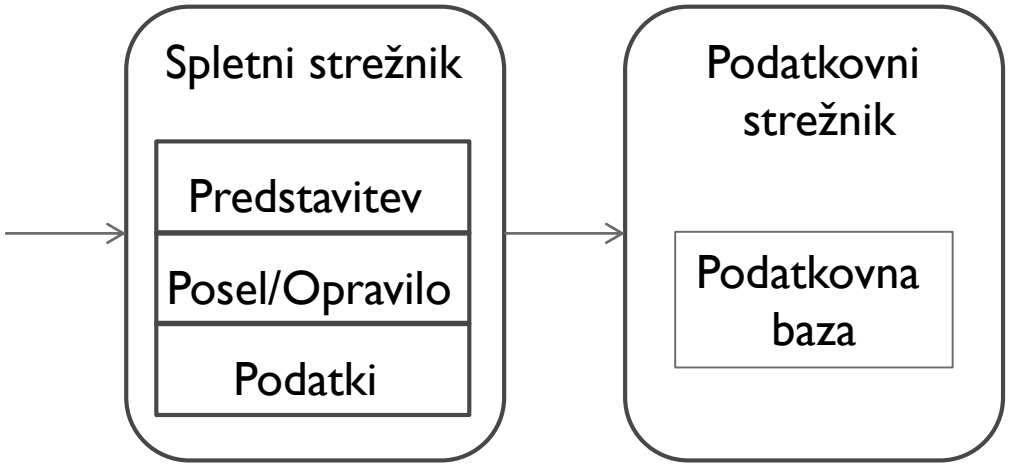
- Testabilnost - Kako dobro sestavimo kriterije testiranja in izvajamo teste, da ugotovimo ali so kriteriji izpolnjeni.
- Upoštevati testabilnost med načrtovanjem arhitekture za enostavnejše zgodnje diagnosticiranje problemov.
- Napotki:
  - Jasna določitev vhodov in izhodov aplikacijskih slojev in komponent v razvojni fazi.
  - Ločevanje predstavitvenih vzorcev kot sta MVC ali MVP
  - Ločen poslovni nivo od ostalega
  - Šibka povezanost med komponentami tako, da jih je možno ločeno testirati.
  - Razvij učinkovite strategije sistema beleženja za detekcijo ali odpravljanje napak, ki jih je sicer težko odkriti.

Tehnološki vidiki

- Microsoft okolje (ASP.NET)– kombiniramo model ASP.NET Web Forms z drugimi tehnologijami: AJAX, ASP.NET MVC, Silverlight, ASP.NET Dynamic Data
- Napotki:
  - Aplikacija z dostopom preko brskalnika – ASP.NET
  - Interakivnost in procesiranje v ozadju s čim manj nalaganj strani – ASP.NET in AJAX
  - Aplikacije z veliko količino multimedijske vsebine - ASP.NET in kontrola Silverlight
  - Ob ASP.NET upoštevati glavne strani za izvedbo konsistentnega UI na vseh straneh
  - Podatkovno upravljana aplikacija - ASP.NET Dynamic Data
  - Testno zasnovan razvojni pristop – MVC pattern in ASP.NET MVC

Pogoji umestitve/namestitve

- Premislek: vpliv na zmogljivost, varnost aplikacije
- **Neporazdeljena umestitev** – poveča zmogljivost z zmanjšanjem števila klicev preko fizičnih meja:
  - Nivoji so fizično nameščeni na istem spletnem strežniku, razen podatkovna baza.
  - Upoštevati zahteve dostopa večjega števila hkratnih uporabnikov.
  - Kako zavarovati sloje na istem strežniku?



- **Porazdeljena umestitev:**
  - izboljšana skalabilnost (omogoča večji obseg dela) in
  - ločena zaščita slojev
  - predstavitveni sloj je fizično ločen od poslovnega in podatkovnega – oddaljena komunikacija.