UNIVERZITET METROPOLITAN BEOGRAD

CS230 - DISTRIBUIRANI SISTEMI

Razvoj JEE projekta

Lekcija 15

PRIRUČNIK ZA STUDENTE

CS230 - DISTRIBUIRANI SISTEMI

Lekcija 15

RAZVOJ JEE PROJEKTA

UVOD

Lekcija predstavlja rekapitulaciju gradiva kroz razvoj projekta i pripremu za intervju za posao.

U dosadašnjim lekcijama su prikazani i objašnjeni napredni *Jave EE* koncepti i principi, kao mehanizmi i alati za kreiranje distribuiranih veb aplikacija. Upravo, ovaj deo lekcije trebalo bi da posluži kao rekapitulacija prethodnih lekcija, uvodeći razvoj složenije distribuirane Java veb aplikacije, zajedno sa pratećom projektnom dokumentacijom. Aplikacija će biti u potpunosti funkcionalna i spremna za konkretnu upotrebu.

Dakle, kao što je rečeno, prvi deo lekcije će se baviti definisanjem projektnog zadatka i pripremom odgovarajuće dokumentacije. Dokumentacija će obuhvatiti sve potrebne tabele, dijagrame i izveštaje koji se sreću u izradi realnih projekata. Posebno, lekcija će se fokusirati na izradu i detaljno dokumentovanje programskog koda i podešavanja aplikacije koja je predmet projekta. Na kraju, ovog dela izlaganja, aplikacija će biti i demonstrirana, biće prikazane i slike koje demonstriraju funkcionalnosti ove aplikacije.

Budući da ovaj predmet predstavlja zaokruživanje izučavanja problematike razvoja softvera nad Java platformom i programskim jezikom, drugi deo lekcije će se fokusirati na pripremu za odlazak na intervju kod poslodavca koji zapošljava Java programere. U tu svrhu će detaljno biti obrazložena brojna pitanja, prikupljena sa većeg broja relevantnih izvora, i koja predstavljaju pitanja koja potencijalni kandidati dobijaju prilikom razgovora za posao.

Savladavanjem ove lekcije student će biti osposobljen da samostalno kreira i dokumentuje *Java EE* projekat, kao i da potpuno spreman se pojavi na intervjuu kod poslodavca.

Poglavlje 1

Zadatak i obim projekta

PROBLEM I CILJ PROJEKTA

Aplikacija bi trebalo da informiše korisnika o organskoj hrani.

Ovaj projekat rešava problem korisnika koji žele da se hrane zdravo, a pritom žele da brzo i lako dođu do organske hrane. Aplikacija bi trebalo da informiše korisnika o organskoj hrani, da mu omogući pregled svih proizvoda koji su dostupni za naručivanje, da omogući naručivanje, kao i da omogući adminu preko admin panela CRUD operacije nad proizvodima, korisnicima, receptima i farmama, kao i uvid u analitiku – tj. vizuelni prikaz najprodavanijih proizvoda, korisnika koji najviše koriste aplikaciju i koliko koji korisnik troši po korpi novca . Veb aplikacija ovakve namene nema puno u Srbiji tj. postoje dve veb stranice, te će naša aplikacija pružiti bolja rešenja od postojećih.

Kroz aplikaciju je potrebno realizovati niz funkcionalnosti koje se uglavnom odnose na mogućnost naručivanja proizvoda preko veb sajta, kao i mogućnost da se lako dođe do traženih informacija. U zavisnosti od tipa funkcionalnosti koju treba obezbediti putem ove aplikacije, moguće je razlikovati dva tipa korisnika- administrator i klijent. Na projektu će biti realizovane funkcionalnosti za klijenta, kao i za aministratora čija je osnovna namena da pristupa, kreira, menja i briše proizvode, farme, recepte i korisnike.

OPSEG I METODOLOGIJE

Akcenat je na agilnom metodu razvoja aplikacije.

Potrebno je da za svaki tip korisnika sistem ispunjava sve zahteve baš za taj tip. Dakle, radi se o kompleksnom sistemu koji se na osnovu arhitekture same aplikacije može definisati kao MVC arhitektura, u okviru koje će postojati dva modula od kojih svaki pruža funkcionalnosti koje su namenjene posebnim tipovima korisnika, odnosno rolama.

Aplikacija koja će biti razvijena namenjena veb korisnicima zato smo će biti akcenat na agilnom metodu razvoja aplikacije. Koncept predstavlja ujedno i dokumentaciju budućeg veb sistema.

Poglavlje 2

Opis sistema

PLAN I ODLIKE SISTEMA

Plan razvoja projekta je takav da se vrši raspodela zadataka

Pre svega je važno napomenuti da plan podrazumeva okolnost da su specifikacije i zahtevi koji će takođe biti detaljno analizirani u dokumentu već obezbeđeni tj. jasno definisani. Dakle, plan koji će ovde biti izložen se odnosi isključivo na proces razvoja i testiranja sistema.
Sistem ima dva tipa korisnika i to: *klijenta* i *administratora* koji ujedno i upravlja veb aplikacijom. Na projektu rade dve osobe, tako da je plan razvoja projekta takav da se vrši raspodela zadataka kako bi se za što kraće vreme, a podjednako kvalitetno razvio sistem.

Glavne odlike sistema su:

- istovremeno korišćenje sistema od strane većeg broja korisnika,
- sve promene vezane za proizvod se čuvaju i ažuriraju u bazi podataka.
- *Sistem* omogućava komunikaciju između *klijenta* i *admina* kroz formu za slanje poruka, a sve u cilju olakšavanja korišćenja aplikacije za klijenta.

KORISNIČKE KARAKTERISTIKE I OGRANIČENJA SISTEMA

Aplikacija će se sastojati iz dva modula, modul klijent i modul admin

Aplikacija će se sastojati iz dva modula, modul klijent i modul admin. Svaki modul predstavlja jedan tip korisnika. Aplikacije je prvenstveno namenjena za korisnike koji žele da kupe tj. poruče organsku hranu.

Sistem mora imati adekvatnu zaštitu što se tiče sistema za *online* plaćanje. Zamisao ekipe koja radi na projektu je da se koristi *PayPal* ali kao prikaz funkcionalnosti sistema postojaće obična forma za popunjavanje broja kartice, PIN broja itd.

Korisnički interfejsbi trebalo da bude jasan i interaktivan. Prilikom pojavljivanja grešaka, sistem treba da prikaže poruke o greškama i na taj način upozori i obavesti korisnika o tipu greške, kao i o tome gde je greška nastala. Najčešći tip greške je prilikom popunjavanja forme, te ćemo posebno raditi na obaveštenjima za takav tip greške.

Poglavlje 3

Specifikacija zahteva

FUNKCIONALNI ZAHTEVI - KLIJENT

Definisanje funkcionalnih zahteva za klijenta.

Klijent bi pomoću aplikacije mogao da pregleda sve organske proizvode u ponudi, kao i da naruči svaki od njih. Korisnik će prvo staviti proizvod u korpu, a potom naručiti proizvod. *Korisnik* ima mogućnost pregleda recepata, kao i osnovnih informacija o organskoj hrani i kontakt stranicu preko koje može poslati poruku ili mail zaposlenima u radnji I videte lokaciju o kontakt telefon.

- *Informacije o organskoj hrani* - klijent će moći da vidi sve informacije o organskoj hrani.
- *Pregled proizvoda* - klijent će moći da vidi sve proizvode u ponudi i to po kategorijama: voće, povrće, meso, mlečni proizvodi i jaja.
- *Odabir proizvoda* - klijent će moći da odabere određeni proizvod.

- Kontakt - klijentu će biti dostupna kontakt strana na kojoj može videti brojeve telefona, email adresu kao i adresu prodavnica u gradu.
- Korpa - klijent će moći da odabrani proizvod stavi u korpu u količini koju sam izabere.
- Naručivanje - klijent će iz korpe moći da naruči proizvod.
- *Uklanjanje iz korpe* - klijent će moći da ukloni proizvod iz korpe.

FUNKCIONALNI ZAHTEVI – ADMINISTRATOR

Administrator bi trebalo da upravlja aplikacijom.

Administrator bi trebalo da upravlja aplikacijom, tj. da vodi i održava stranu. Osnovna funkcija administratora je da unosi nove proizvode i informacije o njima, unosi nove kontakte, recepte i kontroliše analitiku poslovanja.

- Prijavljivanje - administrator može da se uloguje koristeći administratorski nalog.
- *Dodavanje proizvoda* - administrator može da doda proizvod u svakoj kategoriji.
- *Uklanjanje proizvoda* - administrator može da ukloni proizvod u svakoj kategoriji.
- *Dodavanje opisa proizvoda* - administrator može da doda opis proizvoda u svakoj kategoriji.

- *Dodavanje cene proizvoda* - administrator može da doda cenu proizvoda u svakoj kategoriji;
- *Promena kontakta* - administrator može da doda i izbriše kontakte firme (adresu, email, broj telefona).

- *Pregled narudžbina* - administrator može da pregleda narudžbine.
- *Dodavanje recepata* - administrator može da doda recept.
- *Brisanje recepata* - administrator može da izbriše recept.

NEFUNKCIONALNI ZAHTEVI

Preformanse, bezbednost i zaštita i raspoloživost

Performanse - kada su u pitanju performanse, potrebno je da se sadržaji iz baza učitavaju i prikazuju optimalnom brzinom. Zahtevi za sadržajem u bazi će se realizovati preko JEE servisa.
Bezbednost i zaštita - po pitanju bezbednosti treba voditi računa da se onemogući *neautentifikovan* i *neautorizovan* pristup korisničkim nalozima. Takođe veoma bitan aspekt plaćanja je bezbednost i kako je postići. Plan za dalji razvoj je korišćenje nekog sigurnog sistema za plaćanje kao što je *PayPal*. Baš iz razloga bezbednosti ne čuvamo broj kartice u bazi podataka korisnika.

Raspoloživost - *sistem* je raspoloživ samo korisnicima koji na njemu imaju otvoren nalog: klijent i administrator, ali moguće je koristiti aplikaciju bez prijavljivanja ili kreiranja naloga.

3.1 Tehnički opis funkcionalnosti

KLIJENT I ADMINISTRATOR

Tehnički opis funkcionalnosti za klijenta i administratora.

Što se tiče funkcionalnosti za *klijenta*, sa tehničke strane bitno je da se napomene da će on pristupiti svom funkcionalnom meniju tako što će se logovati na sistem i to unoseći svoju korisničku lozinku i ime. Pristup funkcionalnom meniju od strane *klijenta* moguć je i bez prijavljivanja, ali onda *klijentu* nije omogućeno da naruči tj. kupi neki proizvod *online*. Klijent može koristiti aplikaciju neprijavljen, ali bi trebalo da se prijavi kada odabere kupovinu proizvoda i onda će mu biti omogućeno da kupi proizvode.

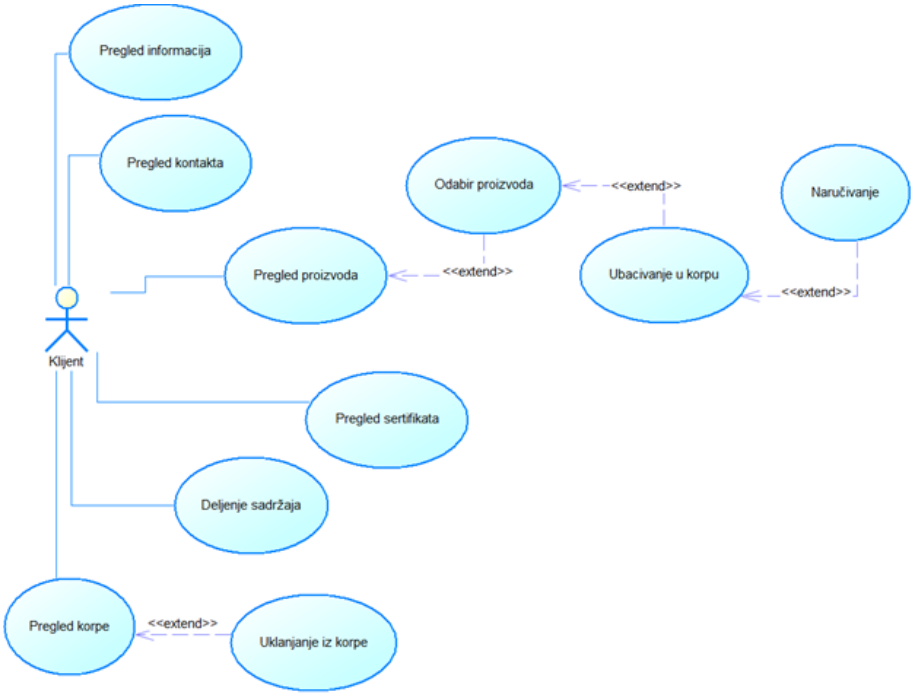
Prilikom kreiranja naloga potrebno je da se proveri da li nalog već postoji u bazi kako bi se izbegao dupli unos, a to će se izvršiti proverom postojanja unetog korisničkog imena u bazi korisnika.
Glavna funkcionalnost koja je administratoru data je dodavanje novih proizvoda, kontakata kao i izmena ili brisanje postojećih.

3.2 Slučajevi upotrebe

SLUČAJ UPOTREBE KORISNIKA KLIJENT - PREGLED INFORMACIJA

Prikaz i opisi korišćenja klijenta.

Na sledećoj slici prikazan je slučaju potrebe za korisnika klijent.



Slika 3.1.1 Slučaj korišćenja za korisnika klijent [izvor: autor]

Sledećom tabelom je prikazan pregled informacija.

Kratak opis:	Korisnik ima mogućnost da pregleda sve potrebne informacije o organskoj hrani kako bi bio upućen u to po čemu se ona razlikuje i ostale hrane i zašto je treba konzumirati. Takođe, korisnik može da pročita informacije o proizvođaču.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira opciju glavnog menija pod nazivom "O nama". 2. Sistem prikazuje informacije o organskoj hrani i proizvođaču.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	Lozinka je pogrešno uneta. Korisničko ime je pogrešno uneto
Rezultat:	Korisnik je dobio informacije o organskoj hrani i proizvođaču.

Slika 3.1.2 Tabela - pregled informacija [izvor: autor]

DODAVANJE U KORPU I NARUČIVANJE

Opis funkcionalnosti dodavanje u korpu i naručivanje.

Sledećom tabelom je prikazana funkcionalnost *dodavanje u korpu*.

Kratak opis:	Korisnik ima mogućnost da pregleda sve potrebne informacije o organskoj hrani kako bi bio upućen u to po čemu se ona razlikuje i ostale hrane i zašto je treba konzumirati. Takođe, korisnik može da pročita informacije o proizvođaču.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira opciju glavnog menija pod nazivom "O nama". 2. Sistem prikazuje informacije o organskoj hrani i proizvođaču.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	Lozinka je pogrešno uneta. Korisničko ime je pogrešno uneto
Rezultat:	Korisnik je dobio informacije o organskoj hrani i proizvođaču.

Slika 3.1.3 Tabela - ubacivanje u korpu [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *naručivanje*.

Kratak opis:	Korisnik bira opciju ‘Potvrdi sadržaj korpe‘‘ čime potvrđuje narudžbinu.
Učesnik:	Klijent
Preduslovi:	1.Korisnik ima pristup internetu. 2.Korisnik je ulogovan na nalog. 3.Korisnik je dodao određeni prozivod u korpu.
Osnovni scenario:	1. Korisnik u navigacionom meniju bira opciju “Potvrdi sadržaj/naruči“. 2. Sistem mu vraća formu kako bi korisnik obezbedio dodatne informacije o narudžbini.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	Podaci o kreditnoj kartici koje je korisnik uneo nisu validni i sistem mu ponovo obezbeđuje formu za unos informacija. Ukoliko korisnik i nakon toga unosi nepostojeće/pogrešne podatke sistem mu nudi rešenje "Plati po preuzimanju" i zahteva od korisnika broj telefona kako bi se izvršila validacija putem SMS-a.
Rezultat:	Korisnik je uspešno ubacio proizvode u korpu.

Slika 3.1.4 Tabela - naručivanje [izvor: autor]

PREGLED KORPE I UKLANJANJE IZ KORPE

Opis funkcionalnosti pregled korpe i uklanjanje iz korpe.

Sledećom tabelom je prikazana funkcionalnost *pregled korpe*.

Kratak opis:	Korisnik ima mogućnost da vidi sve proizvode koje je stavio u korpu.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira opciju pregleda korpe. 2. Sistem prikazuje sadržaj korpe.
Proširenje na drugi slučaj korišćenja:	Uklanjanje iz korpe
Izuzeci:	/
Rezultat:	Korisnik ima uvid u korpu.

Slika 3.1.5 Tabela - pregled korpe [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *uklanjanje iz korpe*.

Kratak opis:	Korisnik bira opciju "Izbriši iz korpe" kako bi uklonio konrektan proizvod iz svoje korpe.
Učesnik:	Klijent
Preduslovi:	1.Korisnik ima pristup internetu 2.Korisnik je dodao određeni prozivod u korpu. 3.Korisnik je u navigacionom meniju izabrao opciju ‘‘Pregled korpe‘‘.
Osnovni scenario:	1. Korisnik bira opciju da izbriše određeni proizvod iz svoje korpe. 2. Sistem mu vraća poruku "Da li želite da uklonite konkretan predmet iz svoje kopre"? 3. Korisnik bira opciju da. 4. Sistem vraća poruku da je proizvod uspešno izbrisan iz korpe.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	1. Nakon drugog koraka korisnik je izabrao opciju ne. 2. Sistem ga vraća na pregled korpe.
Rezultat:	Korisnik je uspešno izbrisao određeni proizvod iz svoje korpe.

Slika 3.1.6 Tabela - uklanjanje iz korpe [izvor: autor]

PREGLED I ODABIR PROIZVODA

Opis funkcionalnosti pregled i odabir proizvoda.

Sledećom tabelom je prikazana funkcionalnost *pregled proizvoda*.

Kratak opis:	Korisnik ima mogućnost da vidi sve proizvode u zavisnosti od toga koju vrstu proizvoda odabere (voće, povrće, meso, mlečni proizvodi i mleko).
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira opciju pregleda prozivoda. 2. Sistem prikazuje svo voće. 3. Korisnik može da promeni vrstu proizvoda. 4. Sistem prikazuje listu odabrane vrste proizvoda.
Proširenje na drugi slučaj korišćenja:	Odabir proizvoda
Izuzeci:	/
Rezultat:	Korisnik ima uvid u listu proizvoda.

Slika 3.1.7 Tabela - pregled proizvoda [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *odabir proizvoda*.

Kratak opis:	Korisnik ima mogućnost da iz liste proizvoda odabere konkretan proizvod. Pritom se pojavljuju opcije odabira količine, uvećanje slike i ubacivanja u korpu. Korisnik, takođe, može videti detaljnije informacije o odabranom proizvodu.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1.Korisnik bira određeni prozovod iz liste proizvoda. 2. Sistem prikazuje odabrani proizvod sa detaljnim informacijama i opcijama za uvećanje slike, odabira količine,i ubacivanje u korpu .
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	Ubacivanje u korpu
Rezultat:	Korisnik je odabrao željeni proizvod.

Slika 3.1.8 Tabela - odabir proizvoda [izvor: autor]

PREGLED KONTAKATA I DELJENJE SADRŽAJA

Opis funkcionalnosti pregled kontakata i deljenje sadržaja.

Sledećom tabelom je prikazana funkcionalnost *pregled kontakata*.

Kratak opis:	Korisnik može da odabere opciju pregled kontakta pri čemu može videti broj telefona, email adresu i adrese prodavnica u gradu.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira opciju "Kontakti". 2. Sistem prikazuje mapu lokacija u gradu na kojima se nalaze prodavnice, brojeve telefona i email adresu.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Korisnik je dobio informacije o kontaktima.

Slika 3.1.9 Tabela - pregled kontakata [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *deljenje sadržaja*.

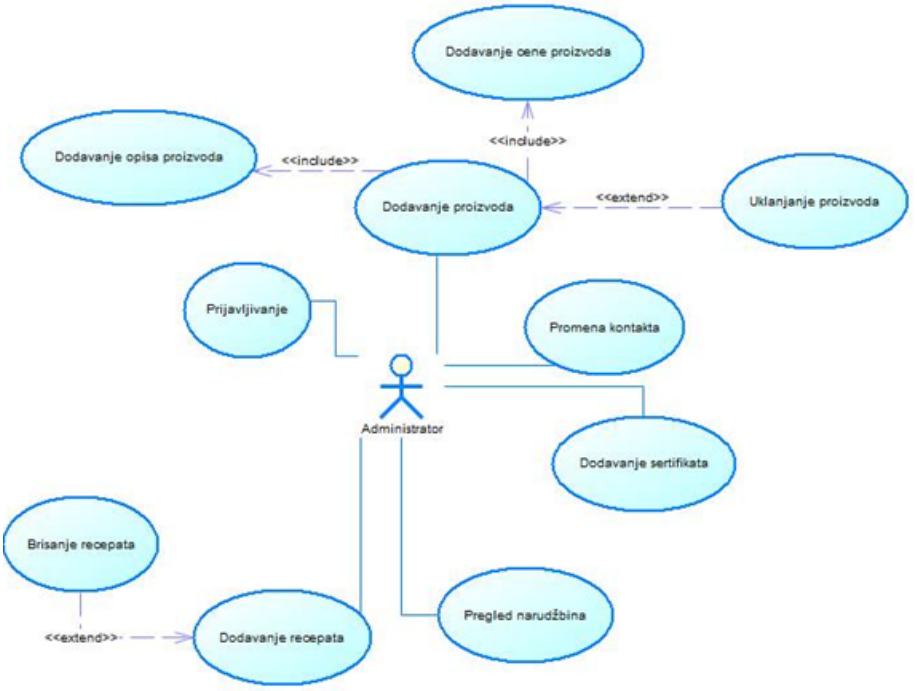
Kratak opis:	Korisnik može da deli sadržaj stranice na društvenim mrežama.
Učesnik:	Klijent
Preduslovi:	1. Korisnik ima pristup internetu.
Osnovni scenario:	1. Korisnik bira neku od ikonica za deljenje sadržaja na dnu strane u zavisnosti od toga na kojoj društvenoj mreži želi da deli sadržaj. 2. Sistem deli željeni sadržaj na odabranu društvenu mrežu.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Korisnik je uspešno podelio sadržaj.

Slika 3.1.10 Tabela - Deljenje sadržaja [izvor: autor]

SLUČAJ UPOTREBE KORISNIKA ADMINISTRATOR ZA PRIJAVLJIVANJE

Prikaz i opisi korišćenja administratora.

Na slici ispod prikazan je slučaj korišćenja za korisnika administrator.



Slika 3.1.11 Slučaj upotrebe korisnika administrator [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *prijavljivanje*.

Kratak opis:	Administrator ima mogućnost prijavljivanja na sistem.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Administrator ima otvoren nalog na sistemu.
Osnovni scenario:	1. Administrator šalje zahtev za prijavljivanje na sistem. 2. Sistem vraća formu za prijavljivanje na sistem. 3. Administrator popunjava formu unošenjem korisničkog imena i šifre.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator se uspešno prijavio na sistem.

Slika 3.1.12 Tabela - prijavljivanje [izvor: autor]

DODAVANJE PROIZVODA I OPISA PROIZVODA

Opis funkcionalnosti dodavanje proizvoda i opisa proizvoda.

Sledećom tabelom je prikazana funkcionalnost *dodavanje proizvoda*.

Kratak opis:	Administrator ima mogućnost dodavanja novih proizvoda u bazu proizvoda.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu.
Osnovni scenario:	1. Administrator bira opciju dodavanja novog proizvoda 2. Sistem prikazuje formu za dodavanje novog proizvoda. 3. Administrator popunjava formu za dodavanje novog proizvoda. 4.Sistem prikazuje poruku o uspešno dodatom proizvodu.
Proširenje na drugi slučaj korišćenja:	Uklanjanje proizvoda
Izuzeci:	/
Rezultat:	Administrator je uspešno dodao novi proizvod u bazu proizvoda.

Slika 3.1.13 Tabela - dodavanje proizvoda [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *dodavanje opisa proizvoda*.

Kratak opis:	Administrator ima mogućnost da doda opis proizvoda pri dodavanju novog proizvoda u bazu.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu. 3. Administrator je izabrao opciju dodavanja proizvoda.
Osnovni scenario:	1. Administrator je odabrao opciju dodavanja proizvoda. 2. Sistem prikazuje formu za dodavanje proizvoda. 3. Administrator unosi opis proizvoda koji želi da doda.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno dodao opis proizvoda.

Slika 3.1.14 Tabela - dodavanje opisa proizvoda [izvor: autor]

DODAVANJE CENE PROIZVODA I UKLANJANJE PROIZVODA

Opis funkcionalnosti dodavanje cene proizvoda i uklanjanje proizvoda.

Sledećom tabelom je prikazana funkcionalnost *dodavanje cene proizvoda*.

Kratak opis:	Administrator ima mogućnost da doda cenu proizvoda pri dodavanju novog proizvoda u bazu.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu. 3. Administrator je izabrao opciju dodavanja proizvoda.
Osnovni scenario:	1. Administrator je odabrao opciju dodavanja proizvoda. 2. Sistem prikazuje formu za dodavanje proizvoda. 3. Administrator unosi cenu proizvoda koji želi da doda.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno dodao cenu proizvoda.

Slika 3.1.15 Tabela - Dodavanje cene proizvoda [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *uklanjanje proizvoda*.

Kratak opis:	Administrator ima mogućnost da obriše proizvod iz baze.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu. 3. Administrator je izabrao opciju dodavanja proizvoda.
Osnovni scenario:	1. Administrator je odabrao opciju brisanja proizvoda. 2. Sistem prikazuje poruku o uspešnom brisanju proizvoda.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno obrisao proizvod.

Slika 3.1.16 Tabela uklanjanje proizvoda [izvor: autor]

DODAVANJE SERTIFIKATA I PREGLED NARUDŽBINA

Opis funkcionalnosti dodavanje sertifikata i pregled narudžbina.

Sledećom tabelom je prikazana funkcionalnost *dodavanje sertifikata.*

Kratak opis:	Administrator ima mogućnost dodavanja sertifikata.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu.
Osnovni scenario:	1. Administrator bira opciju dodavanja sertifikata. 2. Sistem prikazuje formu za dodavanje novog sertifikata (stavljanje slike). 3. Administrator unosi novi sertifikat i potvđuje unos. 4.Sistem prikazuje poruku o uspešno dodatom sertifikatu.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno dodao novi sertifikat.

Slika 3.1.17 Tabela - Dodavanje sertifikata [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *pregled narudžbina*.

Kratak opis:	Administrator ima mogućnost da pregleda narudžbine.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu.
Osnovni scenario:	1. Administrator je odabrao opciju pregleda narudžbina. 2. Sistem prikazuje narudžbine.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno pregledao narudžbine.

Slika 3.1.18 Tabela - pregled narudžbina [izvor: autor]

DODAVANJE I BRISANJE RECEPATA

Opis funkcionalnosti dodavanje i brisanje recepata

Sledećom tabelom je prikazana funkcionalnost *dodavanje recepata.*

Kratak opis:	Administrator ima mogućnost da doda recept koji se sastoji iz organske hrane.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu.
Osnovni scenario:	1. Administrator je odabrao opciju dodavanja recepata. 2. Sistem prikazuje formu za dodavanje recepata. 3. Administrator unosi recept i potvrđuje unos. 4.Sistem prikazuje recept.
Proširenje na drugi slučaj korišćenja:	Brisanje recepata
Izuzeci:	/
Rezultat:	Administrator je uspešno dodao recept.

Slika 3.1.19 Tabela - dodavanje recepata [izvor: autor]

Sledećom tabelom je prikazana funkcionalnost *brisanje recepata*.

Kratak opis:	Administrator ima mogućnost da obriše proizvod iz baze.
Učesnik:	Administrator
Preduslovi:	1. Administrator ima pristup internetu. 2. Admnistrator je uspešno pristupio svom nalogu. 3. Administrator je izabrao opciju dodavanja proizvoda.
Osnovni scenario:	1. Administrator je odabrao opciju brisanja proizvoda. 2. Sistem prikazuje poruku o uspešnom brisanju proizvoda.
Proširenje na drugi slučaj korišćenja:	/
Izuzeci:	/
Rezultat:	Administrator je uspešno obrisao proizvod.

Slika 3.1.20 Tabela - brisanje recepata [izvor: autor]

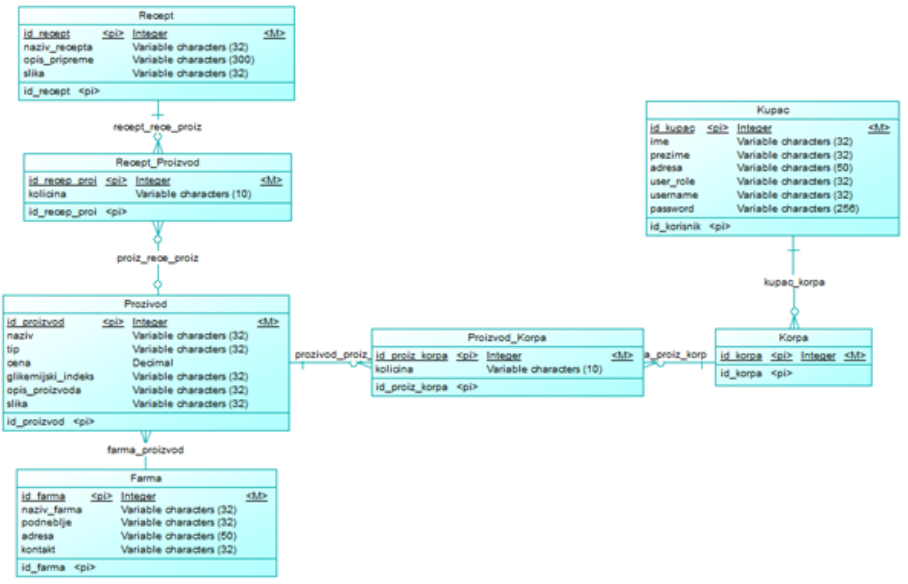
Poglavlje 4

Dizajn sistema

STRUKTURNI DIZAJN - DIZAJN BAZE PODATAKA

Prikaz konceptualnog modela baze podataka.

Na slici 1 nalazi se konceptualni model baze podataka. Na slici je moguće uočiti *Korisnika* i *Proizvod* sa vlastitim atributima. Svaki prozivod je potekao sa neke farme tako da se entitet *Farma* sadrži u *Proizvodu*. Svaki *kupac* može da kreira*korpu*, a u korpi može biti više proizvoda. Isto tako, jedan prozivod može da se nađe u više korpi u slučaju da dva kupca kupuju isti proizvod istovremeno, to je omogućeno kroz entitet *Proizvod_Korpa*. Entiteti *Admin* i *Kupac* nasleđuju entitet *Korisnik*.



Slika 4.1 Konceptualni model baze podataka [izvor: autor]

Poglavlje 5

Implementacija i testiranje

KORIŠĆENE TEHNOLOGIJE

Za izradu sistema korišćena je Java EE 7.

Za izradu sistema korišćena je JavaEE 7, stoga se struktura projekta može podeliti na dva sloja :

- na sloj java koda, zadužen za logiku same aplikacije, i
- na grafički sloj, koji sadrži JSF stranice, kao i CSS fajlove.

Frontend aplikacije je rađen u PrimeFace framework-u uz pomoć BootFaces.

NAVIGACIJA

Navigacija bi trebalo da bude krajnje jednostavna i intuitivna za sve korisnike.

Navigacija bi trebalo da bude krajnje jednostavna i intuitivna za sve korisnike. Što se tiče klijentske strane, korišćenje aplikacije se svodi na pregled organskih proizvoda, koji će biti razdvojeni po kategorijama, a korisniku će na raspolaganju za svaki proizvod biti i cena proizvoda po kilogramu, kao i mogućnost odabira količine određenog proizvoda. Takođe, korisniku će na raspolaganju biti i strana na kojoj će se nalaziti informacije o samoj firmi, kao i gde se mogu pročitati predloženi recepti, kao i strana za pregled farmi. Navigacija treba da bude takva da omogući da se korisnici lako i intuitivno mogu prilagoditi interfejsu kako bi izvršavali svoje funkcije na najbolji mogući način.

Korisnik će moći da pristupi svojoj korpi u bilo koje vreme i ona će kao glavni deo ove aplikacije biti posebno označena. Na kraju svake strane nalaziće se dugme koje će korisnika vratiti na početak te strane, što bi takođe trebalo da omogući korisniku lakše i intuitivnije korišćenje. *Korisnik* će takođe moći da koristi ovu veb aplikaciju putem neke od socijalnih mreža, što će mu takođe biti intuitivno. Što se tiče *administratorskog* dela, on ima uvid u sve delove aplikacije, kao i dodatne funkcionalnosti, poput, dodavanja novih proizvoda i izmenu sadržaja svega postojećeg na sajtu, kao i uvid u analitiku.

TESTIRANJE

Faza testirajna je ključna faza pre nego što se aplikacija okači na server i krene sa radom.

Nakon kodiranja aplikacije sledi faza testiranja (testiranje će biti korišćeno i prilikom kodiranja ali će se testirati direktno određeni deo koda-njegova funkcionalnost). Faza testirajna je ključna faza pre nego što se aplikacija okači na server i krene sa radom, stoga će dobar deo vremena biti odvojen za testiranje aplikacije.

Prvi deo testiranja se odnosi na testiranje funkcionalnosti. Nakon što je aplikacija završena treba prvo proveriti da li postoje takozvane "*mrtve strane*" i treba proveriti da li su validni linkovi ka ostalim stranama. Treba proveriti sve mogućeforme za upisivanje i testirati ih sa lošim podacima i praznim poljima. Nakon toga ide verifikacija toka podataka u okviru aplikacije kao i verifikacija integriteta podataka.

Drugi deo testiranja je testiranje korisnosti tj. koliko je "lako" snaći se i koristiti aplikaciju. Ovde treba proveriti navigaciju i kontrole. A potom uz pomoć nekih drugih lica koja nisu radila na projektu treba da testiraju komponente svake stranice kao i njihovu intuiciju odnosno korisničko zaključivanje ("Da li je korisniku odmah jasno čemu služi i šta treba da radi u okviru aplikacije?").

Treći deo testiranja bi bio testiranje interfejsa aplikacije. Postoji mnogo aplikacija koje služe za testiranje korisničkog interfejsa.

Pošto je u korisničkim zahtevima navedeno da će aplikacija biti prilagodljivog dizajna (*responsive web design*) sledeći korak u testiranju bi bilo testiranje kompatibilnosti. Testiranje kompatibilnosti podrazumeva kompatibilnost na raznim uređajima kao što su desktop računari, laptop računari, tablet i mobilni uređaji. U testiranje kompatibilnosti takođe spada testiranje aplikacije na više različitih pretraživača (*GoogleChrome, Mozilla Firefox, Opera, Explorer*...) kao i testiranje na više različitih operativnih sistema (*Windows, Mackintosh, Linux*...).

Peti deo testiranja je testiranje performansi to podrazumeva proces verifikacije vremena koje je potrebno serveru da odgovori u uobičajenim i ekstremnim uslovima. Jedan od mnogobrojnih testova za testiranje performansi je *stres test* koji utvrđuje koji je maksimum kapacitet aplikacije tj. servera, kao i utvrđivanje kako se aplikacija ponaša kada se pređe preko tog maksimuma. Takođe je moguće koristiti „*Spike test*" koji naglo poveća broj korisnika i prati reakciju aplikacije.

Poslednji deo testiranja je možda i najbitniji jer on predstavlja testiranje sigurnosti aplikacije. Ovaj deo testiranja je posebno bitan kod dela plaćanja proizvoda jer bi krađa informacija kao što su broj kreditne kartice i šifra kreditne kartice bili ogromni propusti.

Takođe je planirano snimanje testiranja sistema od strane trećeg lica.

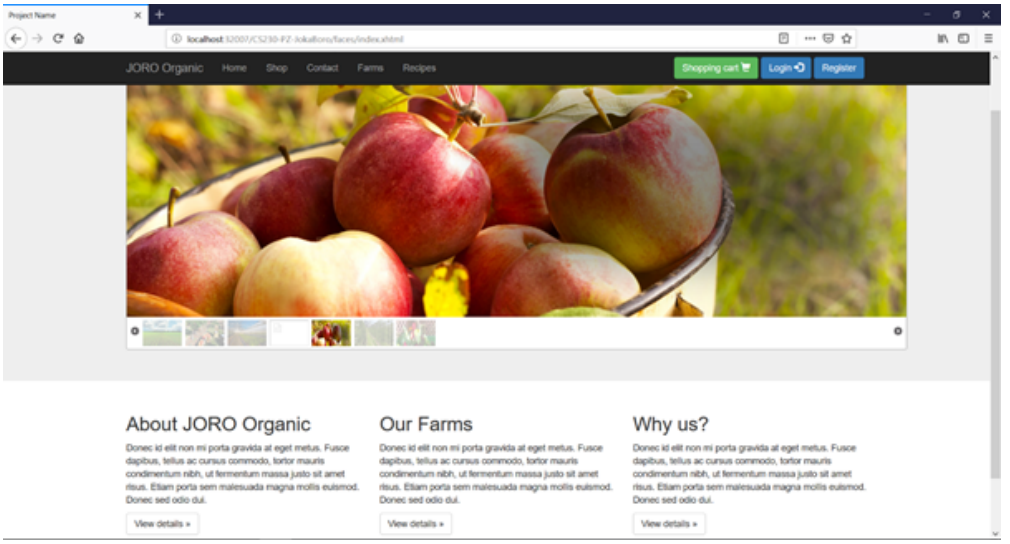
Poglavlje 6

Korisnički interfejs i funkcionisanje aplikacije

POČETNA STRANA

Demonstracija funkcionanosti započinje početnom stranicom.

Demonstracija funkcionalnosti aplikacije kroz navigaciju kroz *grafički korisnički interfejs* (GUI) započinje prikazivanjem početne stranice. Izgled navedene JSF stranice prikazan je sledećom slikom.

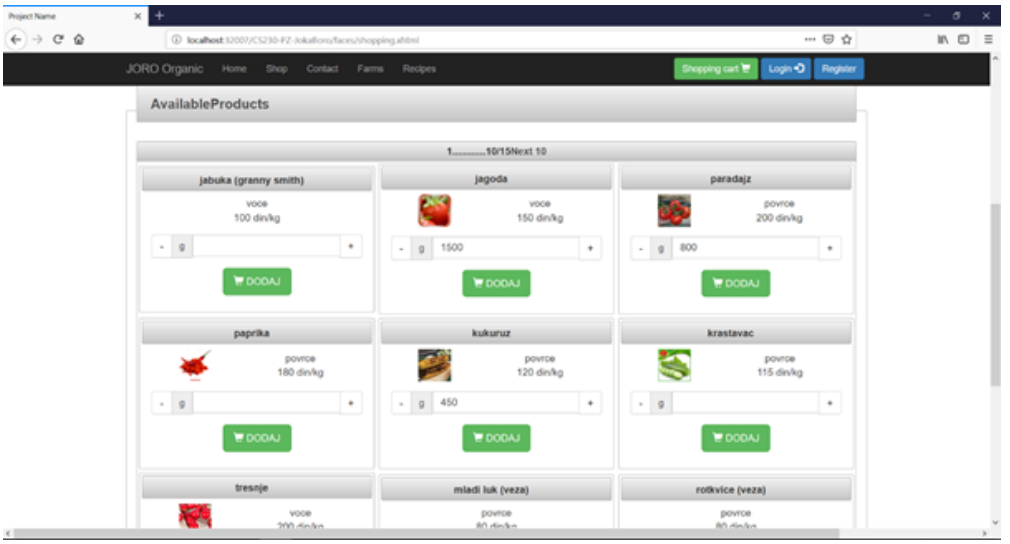


Slika 6.1 Početna strana veb aplikacije Organic JORO [izvor: autor]

PREGLED I NARUČIVANJE PROIZVODA

Sledeća stranica omogućava pregled i naručivanje proizvoda.

Nakon početne stranice, prikazuje se *JSF* stranica koja omogućava pregled i naručivanje proizvoda. Navedena stranica je prikazana sledećom slikom.

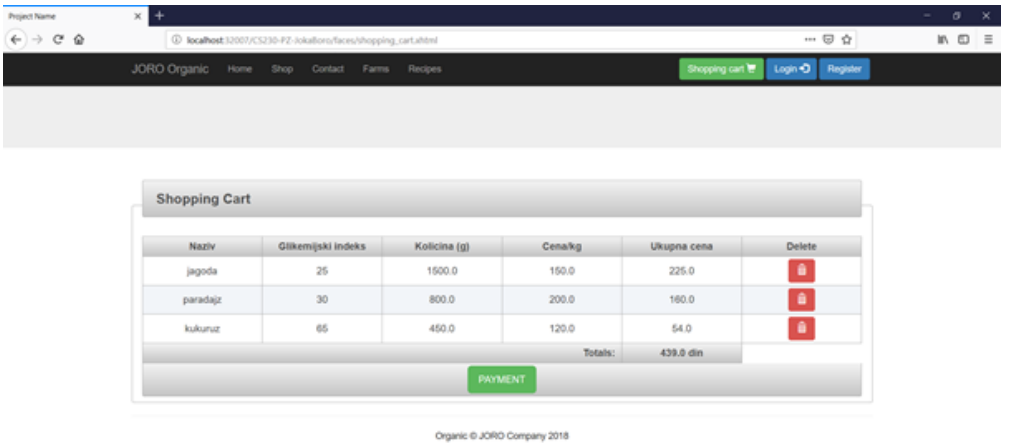


Slika 6.2 Pregled i naručivanje online proizvoda [izvor: autor]

STRANICA ZA PREGLED ONLINE KORPE KORISNIKA

Demo stranice za pregled potrošačke korpe.

Kao što je navedeno u zahtevima, aplikacija sadrži mogućnost pregleda potrošačke korpe. Odgovarajuća *JSF* stranica je prikazana sledećom slikom.

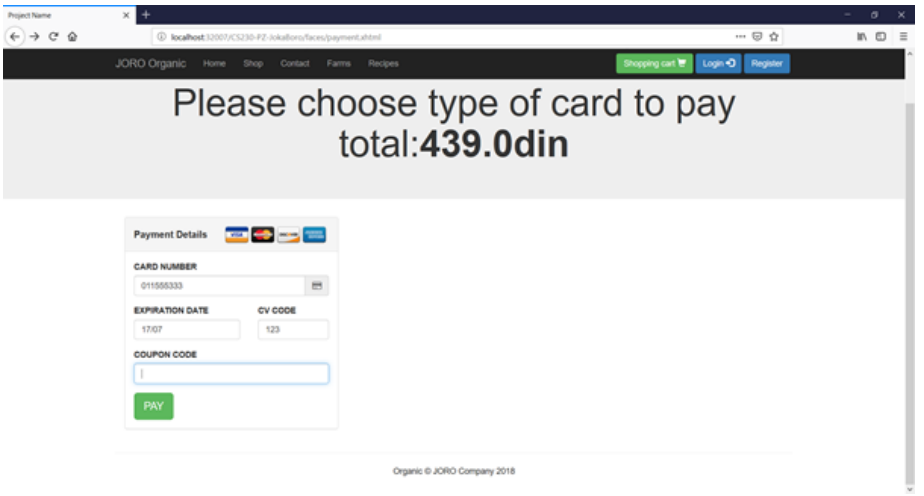


Slika 6.3 Pregled potrošačke korpe [izvor: autor]

PLAĆANJE I POTVRDA PLAĆANJA

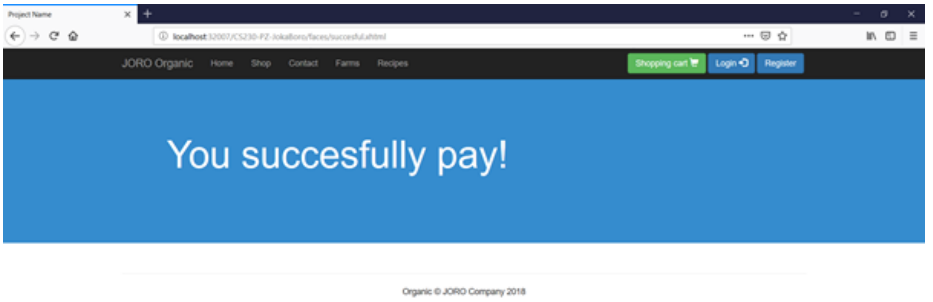
Aplikacija sadrži mogućnost plaćanja za izabranu potrošačku korpu.

Kao što je navedeno u zahtevima, aplikacija sadrži mogućnost plaćanja za izabranu potrošačku korpu. Odgovarajuća *JSF* stranica je prikazana sledećom slikom.



Slika 6.4 Stranica za plaćanje [izvor: autor]

Ishod obavljenog plaćanja prikazan je *JSF* stranicom sa sledeće slike.

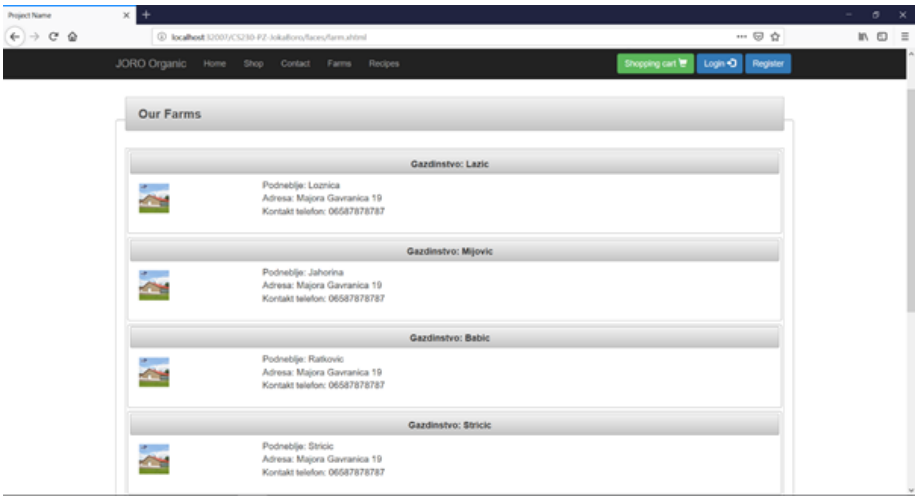


Slika 6.5 Ishod obavljenog plaćanja [izvor: autor]

PREGLED ORGANSKIH FARMI I RECEPATA

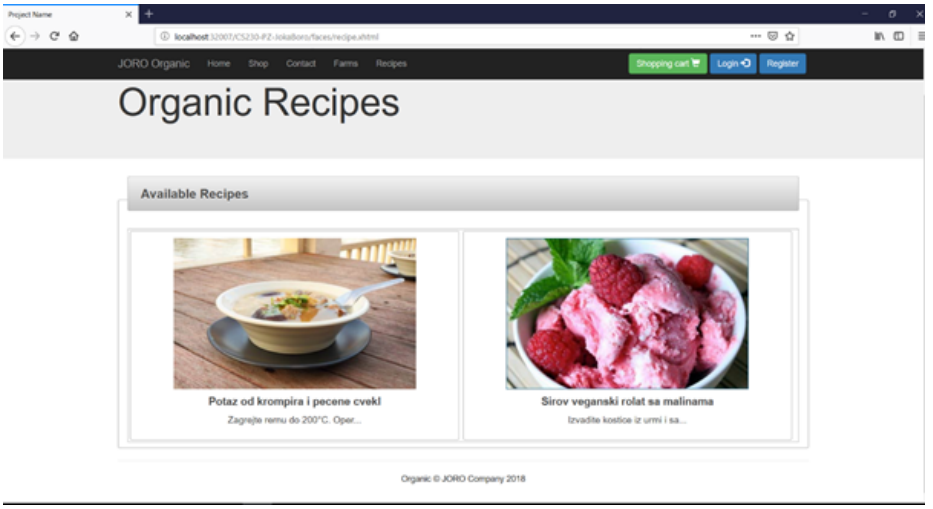
Aplikacija sadrži mogućnost pregleda organskih farmi i recepata za pripremu organske hrane.

Aplikacija sadrži mogućnost pregleda organskih farmi na način prikazan sledećom *JSF* stranicom.



Slika 6.6 Stranica za preglde organskih farmi [izvor: autor]

Aplikacija sadrži mogućnost pregleda organskih recepata na način prikazan sledećom *JSF* stranicom.

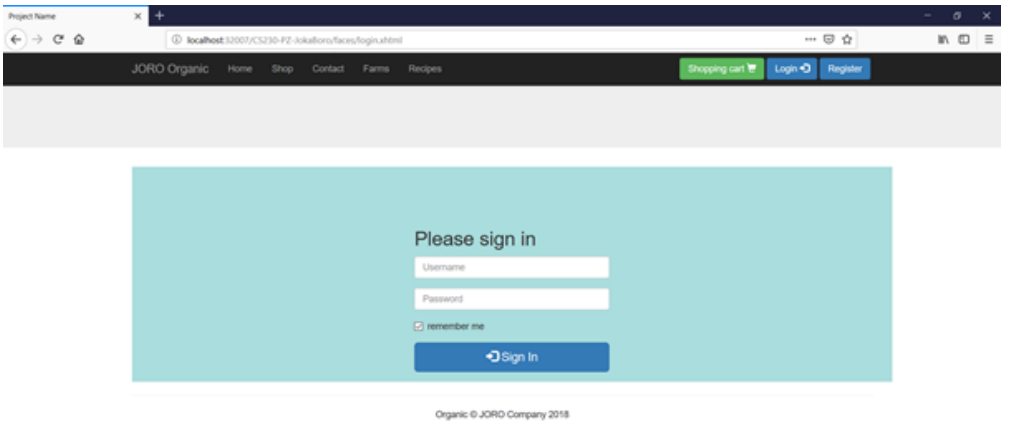


Slika 6.7 Stranica za preglde organskih recepata [izvor: autor]

PRIJAVA KORISNIKA NA SISTEM

Aplikacija je obezbeđene login sistemom.

Svaka ozbiljna veb aplikacija, pa tako i ova, mora da bude obezbeđena od neovlašćenog pristupa vlastitim resursima. Sledećom slikom je prikazana *JSF* stranica za prijavljivanje korisnika na aplikaciju.

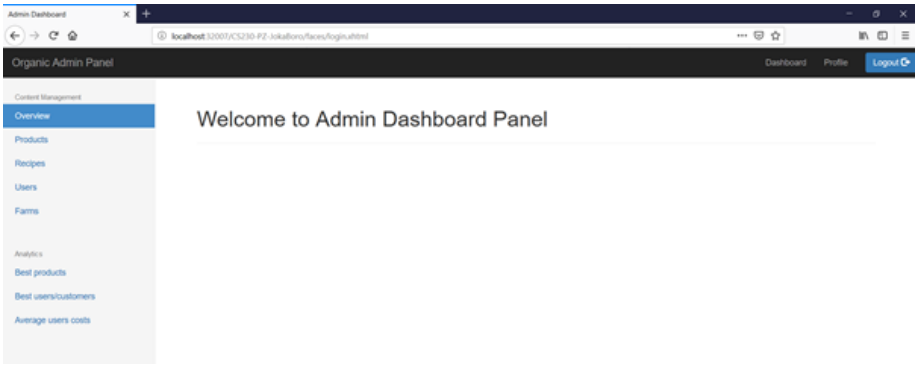


Slika 6.8 Stranica zaprijavljivanje korisnika/administratora [izvor: autor]

ADMIN PANEL I PREGLED PROIZVODA

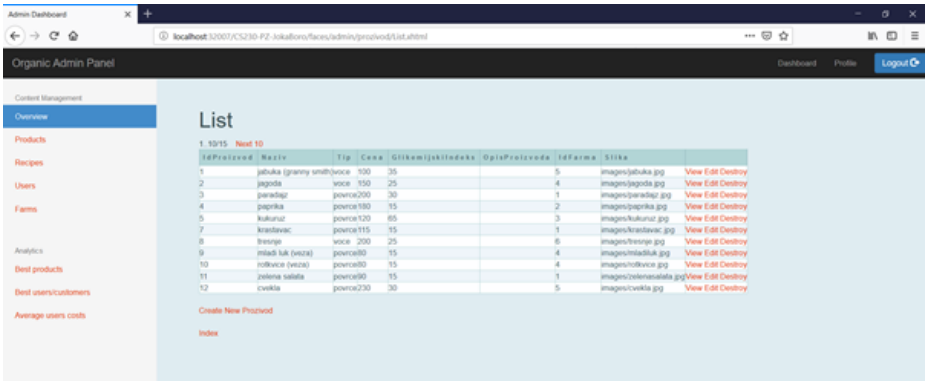
Administrator ima veća prava od standardnog korisnika.

Administrator ima veća prava od standardnog korisnika. Njegove aktivnosti u aplikaciji su olakšane kreiranjem *JSF* stranice - *admin panel.* Izgled navedene stranice je prikazan sledećom slikom.



Slika 6.9 Admin panel stranica [izvor: autor]

Manipulacija dostupnim proizvodima, administratoru je omogućena putem sledeće *JSF* stranice.

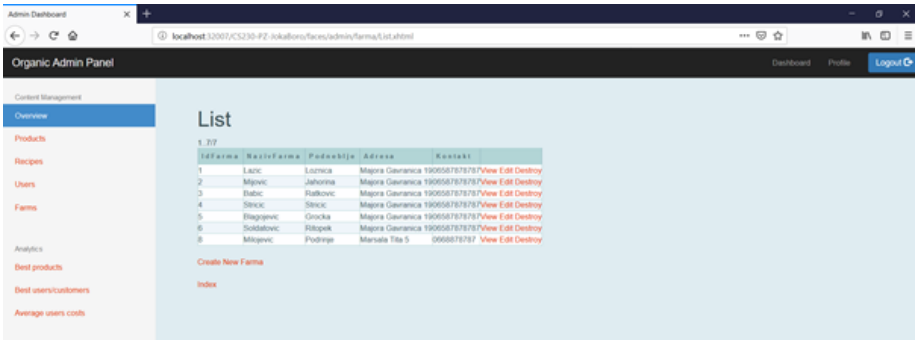


Slika 6.10 Pregled, dodavanje, izmena i brisanje proizvoda [izvor: autor]

FARME I RECEPTI

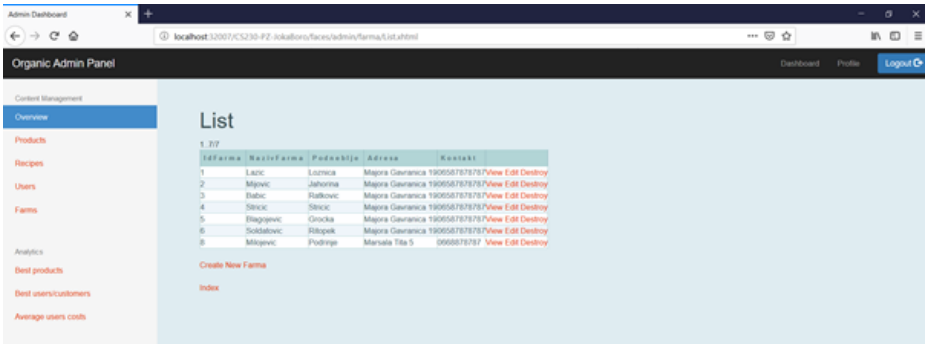
Administratorsko manipulisanje listom farmi i recepata moguće je na posebnim JSF stranicama.

Administratorsko manipulisanje listom farmi i recepata moguće je na posebnim *JSF* stranicama. Sledi izgled stranice na kojoj administrator ima mogućnost dodavanja, brisanja, izmene i pregleda organskih farmi.



Slika 6.11 Dodavanje, brisanje, izmena i pregled organskih farmi [izvor: autor]

Sledi izgled stranice na kojoj administrator ima mogućnost dodavanja, brisanja, izmene i pregleda organskih recepata.

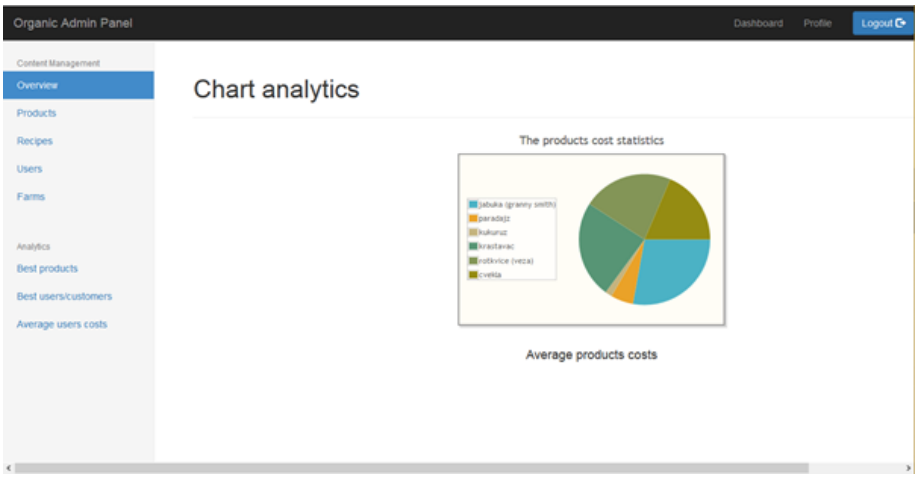


Slika 6.12 Dodavanje, brisanje, izmena i pregled organskih recepata [izvor: autor]

ANALITIKA

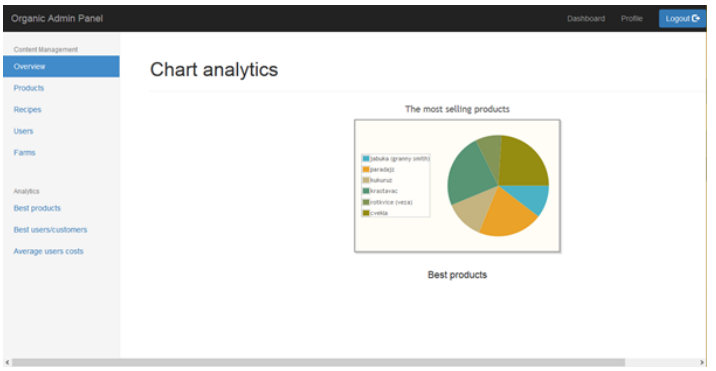
Administrator ima mogućnost pristupa stranicama za analitiku.

Administrator ima mogućnost pristupa stranicama za analitiku. Sledećom *JSF* stranicom je prikazana zarada po proizvodu.



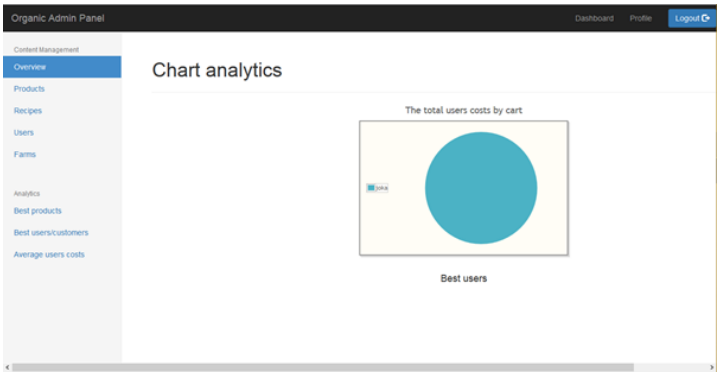
Slika 6.13 Zarada po proizvodu [izvor: autor]

Sledećom *JSF* stranicom je prikazana lista najprodavanijih proizvoda.



Slika 6.14 Lista najprodavanijih proizvoda [izvor: autor]

Sledećom *JSF* stranicom je prikazana potrošnja po potrošačkoj korpi po korisniku.

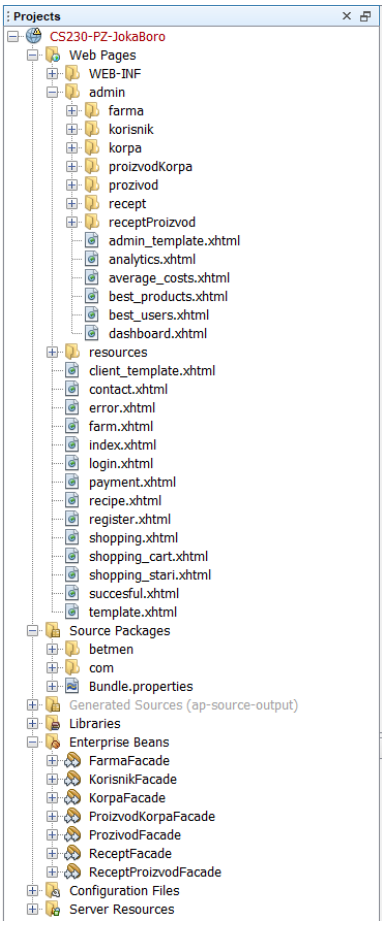


Slika 6.15 Potrošnja po potrošačkoj korpi po korisniku [izvor: autor]

KONAČNA STRUKTURA PROJEKTA

Konačno neophodno je prezentovati konačnu strukturu kreiranog JEE veb projekta.

Konačno neophodno je prezentovati konačnu strukturu kreiranog *JEE* veb projekta. Takođe, ovaj projekat, kao primer dobro urađenog projektnog zadatka, moguće je preuzeti u aktivnosti *Shared Resources* na kraju ovog objekta učenja.



Slika 6.16 Konačna struktura kreiranog JEE veb projekta [izvor: autor]

VIDEO MATERIJAL

Unapređenje razvoja JEE aplikacija kroz prelazak na novu JEE 8 platformu.

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

Poglavlje 7

Priprema za intervju kod Java poslodavca

GRUPE PITANJA ZA RAZGOVOR ZA POSAO JAVA PROGRAMERA

Prilikom razgovora za posao, kandidat dobija pitanja iz više kategorija.

Prilikom razgovora za posao, kandidat dobija pitanja iz više kategorija:

- *Java osnove* (Java Core);
- *Objektno - orijentisani koncepti* - osnovna grupa pitanja;
- *Objektno - orijentisani koncepti* - pitanja u vezi sa konstruktorom;
- *Objektno - orijentisani koncepti* - ***static*** rezervisana reč;
- *Objektno - orijentisani koncepti* - nasleđivanje;
- *Objektno - orijentisani koncepti* - agregacija i kompozicija;
- *Objektno - orijentisani koncepti* - overloading;
- *Objektno - orijentisani koncepti* - overriding;
- *Objektno - orijentisani koncepti* - rezervisana reč ***final***;
- *Objektno - orijentisani koncepti* - polimorfizam;
- *Objektno - orijentisani koncepti* - apstrakcija;
- *Objektno - orijentisani koncepti* - paketi;
- *Rukovanje izuzecima*;
- *Rukovanje Stringovima*;
- *Ugnježdene klase i interfejsi*;
- *Garbage Collection*;
- *Ulazno / izlazne vrednosti*;
- *Serijalizacija*;
- *Umrežavanje*;
- *Refleksija;*
- *Razno, AWT i Swing;*

- *Internacionalizacija;*
- *Java zrna;*
- *RMI;*
- *Višenitnost;*
- *Java kolekcije;*
- *JDBC*.

JAVA OSNOVE (JAVA CORE)

Sledi grupa pitanja u vezi sa JVM, JDK i JRE.

1. Koja je raztika između JDK, JRE i JVM?

ODGOVOR: *JVM* je skraćenica za Java virtuelnu mašinu, apstraktnu mašinu na kojoj se izvršava Java bajt kod (byte code). JVM je dostupna za sve poznate hardversko - softverske platforme.

JRE je skraćenica za *Java Runtime Environment* i predstavlja implementaciju Java virtuelne mašine.

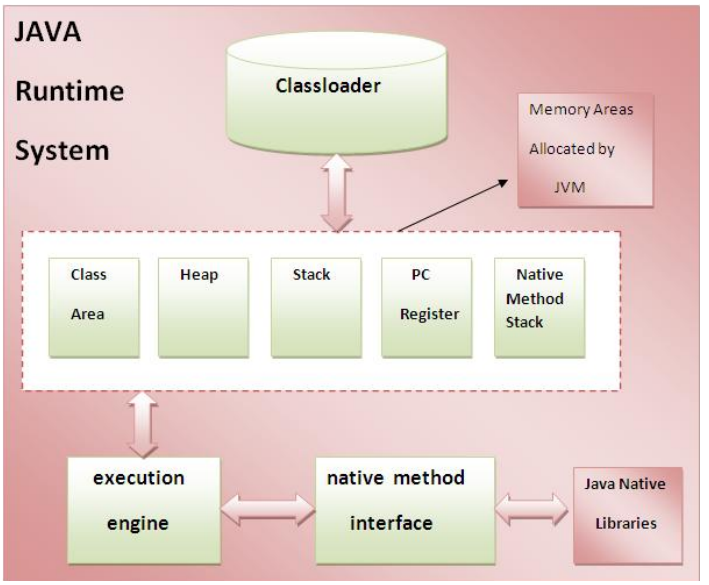
JDK je skraćenica za *Java Development Kit* i sadrži *JRE* i razvojne alate.

2. Koliko tipova memorijskih oblasti je alocirano Java virtuelnom mašinom?

ODGOVOR: JVM alocira više tipova memorijskih oblasti, poput:

- *Class(Method) Area*
- *Heap*
- *Stack*
- *Program Counter Register*
- *Native Method Stack*.

Unutrašnja arhitektura JVM je prikazana sledećom slikom:



Slika 7.1 Unutrašnja arhitektura JVM [Izvor: Oracle]

Classloader je JVM podsistem koji služi za učitavanje datoteka klasa. *Class(Method) Area* skladišti elemente strukture klase: polja, metode, kod metoda itako dalje. *Heap* predstavlja oblast podataka tokom vremena izvršavanja u kojem se alociraju objekti. *Stack* čuva lokalne varijable i međurezultate procesiranja. Svaka nit sadrži privatni JVM stek, kreiran istovremeno kad i nit. *Program Counter Register* sadrži adrese JVM instrukcija koje se trenutno izvršavaju.

Native Method Stack sadrži sve nativne metode koje se koriste u aplikaciji. Mašina za izvršavanje (Execution Engine) sadrži: virtuelni procesor, interpretator, JIT kompajler.

DODATNO O JAVA OSNOVAMA (JAVA CORE)

Sledi nova grupa obaveznih pitanja iz Osnova Jave.

3. Šta je JIT kompajler?

ODGOVOR: *Just-In-Time (JIT)* koristi za unapređivanje performansi. JIT kompajlira delove bajt koda koji imaju sličnu funkcionalnost u istom vremenu i, otuda, smanjuje količinu vremena neophodnog za kompajliranje. U ovom slučaju termin "kompajler" se odnosi na translatora seta instrukcija JVM u set instrukcija procesora.

4. Šta je platforma?

ODGOVOR: Platforma predstavlja hardversko i softversko okruženje na kojem se program izvršava. Java predstavlja softverski baziranu platformu.

5. Koja je osnovna razlika između Javine i drugih platformi?

ODGOVOR: Java predstavlja softverski baziranu platformu koja se izvršava na vrhu drugih hardverski baziranih platformi. Sadrži dve komponente:

- *Runtime Environment*
- *API (Application Programming Interface)*

6. Šta omogućava javinu prirodi "piši jednom, izvrši bilo gde"?

ODGOVOR: *Bajt kod* kao posrednik između izvornog i mašinskog koda.

7. Šta je classloader?

ODGOVOR: *Classloader* je podsistem Java virtuelne mašine koji učitava klase i interfejse. Postoje različiti tipovi za *classloader: Bootstrap classloader, Extension classloader, System classloader, Plugin classloader* i tako dalje.

8. Da li su delete, next, main, exit i null ključne reči u Javi?

ODGOVOR: Nisu!

9. Ako se ne obezbedi nijedan argument na komandnoj liniji da li će String niz Main metode bit prazan ili null?

ODGOVOR: Biće prazan ali ne ***null***.

10. Šta će se desiti ukoliko se napiše "static public void" umesto "public static void"?

ODGOVOR: Program će normalno funckionisati.

11 Koja je podrazumevana vrednost lokalnih varijabli?

ODGOVOR: Nisu inicijalizovane ni na kakvu podrazumevanu vrednost.

OSNOVNA PITANJA I KONSTRUKTOR KOD OBJEKNE ORIJENTISANOSTI

Sledeća grupa pitanja odnosi se na bazična pitanja u vezi sa OO konceptima.

1. U čemu je razlika između objektno - orijentisanih i objektno - baziranih programskih jezika?

REŠENJE: Objektno - bazirani programski jezici podržavaju sve objektno - orijentisane koncepte osim nasleđivanja. Primeri objektno - baziranih programskih jezika su: *JavaScript*, *VBScript* i tako dalje.

2. Koja je inicijalna vrednost reference na objekat koji je definisan kao varijabla instance?

REŠENJE: Sve objektne reference su inicijalizovane na ***null*** u Javi.

3. Šta je konstruktor?

REŠENJE: Objekat je metoda koja se koristi za inicijalizovanje stanja objekta. Konstruktor se poziva kada se objekat kreira. Konstruktor nosi isti naziv kao i klasa čiji se objekat kreira i ne mora da sadrži jasno povratni tip. Postoje dva tipa konstruktora: podrazumevani (bez argumenata) i konstruktori sa parametrima (Parameterized constructor).

4. Koja je svrha podrazumevanog (default) konstruktora?

REŠENJE: Podrazumevani konstruktor obezbeđuje podrazumevane (default) osobine objekta.

5. Da li konstruktor vraća vrednost?

REŠENJE: Da, vraća tekući objekat. Nije moguće koristiti povratni tip, ali ipak vraća vrednost.

6.Da li se konstruktor nasleđuje?

REŠENJE: Konstruktor nije moguće naslediti.

7.Da li se konstruktor može definisati kao final?

REŠENJE: Konstruktor ne može biti ***final***.

REZERVISANA REČ STATIC

Sledi diskusija u vezi sa primenom modifikatora static.

8.Šta je statička varijabla?

ODGOVOR: Statičkom varijablom se definiše opšte svojstvo objekata koji se kreiraju iz date klase. Statička varijabla uzima memoriju samo jednom u oblasti klase, kada se klasa učitava.

9. Šta je statička metoda?

ODGOVOR: Radi se metodi koja pripada klasi, umesto objektu. Statička metoda može biti pozvana bez prethodno kreiranog objekta. Statička metoda može da pristupa statičkim članovima klase i da im menja vrednosti.

10. Zašto je main metoda statička?

ODGOVOR: Zato što nije potreban objekat da je pozove. Ukoliko bi ***main*** metoda bila nestatička, morao bi prvo da bude kreiran objekat koji je poziva što bi dovelo do potrebe za alociranjem dodatne memorije.

11. Šta je statički blok?

ODGOVOR: Koristi se za inicijalizovanje statičkih podataka i izvršava se pre poziva main metode tokom vremena učitavanja klase.

12. Da li je moguće izvršiti program bez main metode?

ODGOVOR: U starijim verzijama Jave je moglo, od Java 7 verzije ne može. Na primer, testira se sledeći kod:

```
class A3{
      static{
      System.out.println("static block is invoked");
      System.exit(0);
      }
    }
```

Verzije Jave 7 i 8 bi javile sledeću grešku:

```
Output:Error: Main method not found in class A3, please define the main method as:
public static void main(String[] args)
```

Starije verzije Jave bi dale sledeći izlaz:

```
Output:static block is invoked (if not JDK7)
```

NASLEĐIVANJE

Nasleđivanje je ključna razlika između OO i ostalih programskih jezika.

13. Šta podrazumeva rezervisana reč "this" u Javi?

ODGOVOR: "***this***" je rezervisana reč koja ukazuje na aktuelni objekat. Može biti upotrebljena na šest različitih načina: kao referenca na instancu aktuelne klase, za poziv metode aktuelne klase, ***this()*** kao poziv konstruktora aktuelne klase, kao argument prosleđen metodi, kao argument u pozivu konstruktora i, na kraju, kao povratna vrednost (instanca tekuće klase) u pozivu metode.

14. Šta je nasleđivanje?

ODGOVOR: Nasleđivanje je objektno - orijentisani mehanizam pomoću kojeg neki objekat dobija sve osobine i ponašanja nekog drugog objekta koji pripada drugoj klasi. Ideja primene nasleđivanja je kreiranje novih klasa nad nekom postojećom klasom. Kaže se da potklasa proširuje (***extends***) roditeljsku (***super***) klasu. Nasleđivanje je relacija tipa "*IS - A*" ili "*roditelj - potomak*".

15. Koja klasa je super klasa svih klasa?

ODGOVOR: Klasa ***Object***.

16. Zašto višestruko nasleđivanje nije podržano u Javi?

ODGOVOR: Razlog je smanjenje kompleksnosti i pojednostavljenje jezika.

17. Šta je kompozicija?

ODGOVOR: Čuvanje reference na neku klasu u drugoj klasi naziva se kompozicija.

18. Koja je razlika između agregacije i kompozicije?

ODGOVOR: Agregacija podrazumeva slabu vezu, kompozicija je snažna veza.

19. Zašto Java ne podržava pointere?

ODGOVOR: Pointeri su varijable koji ukazuju na memorijsku adresu, nisu bezbedni i teški su za razumevanje.

20. Šta predstavlja super u Javi? Da li zajedno mogu da se koriste this i super u konstruktoru?

ODGOVOR: Ključna reč ***super*** zamenjuje roditeljsku klasu. Nije moguće koristiti ***this()*** i ***super()*** zajedno budući da moraju da budu na prvom mestu.

OVERRIDING I OVERLOADING

Sledi grupa pitanja u vezi sa primenom metoda u Javi?

21. Šta je kloniranje?

ODGOVOR: Kreiranje identične kopije nekog objekta.

22. Šta je overloading (preklapanje) metoda?

ODGOVOR: Ukoliko klasa poseduje više metoda sa istim nazivom, ali različitim potpisom (listom parametara) govori se o *overloading* - u. Na ovaj način se povećava čitljivost programa.

23. Da li je moguće u Javi vršiti overloading samo promenom povratnog tipa?

ODGOVOR: Nije moguće.

24. Da li je moguće preklopiti metodu main()?

ODGOVOR: Da! Moguće je u klasi imati više ***main()*** metoda preklapanjem podrazumevane.

25. Šta je overriding (redefinisanje) metode?

ODGOVOR: ***Overriding*** je redefinisanje logike metode super klase u potklasi.

26. Da li je moguće redefinisati statičku metodu?

ODGOVOR: Nije moguće. Statička metoda je deo klase, a ne objekta.

27. Da li je moguće redefinisati preklopljenu metodu?

ODGOVOR: Da, moguće je.

28. Da li je moguće imati virtuelne funkcije u Javi?

ODGOVOR: Da, moguće je. Sve funkcije u Javi su virtuelne po podrazumevanim podešavanjima.

29. Šta predstavlja covariant povratni tip?

ODGOVOR: Od verzije Java 5, moguće je redefinisati bilo koju metodu promenom povratnog tipa ukoliko povratni tip metode potklase predstavlja tip potklase. Ovo je poznato kao covariant povratni tip. Sledi primer:

```
class A{
A get(){return this;}
}

class B1 extends A{
B1 get(){return this;}
void message(){System.out.println("welcome to covariant return type");}

public static void main(String args[]){
new B1().get().message();
}
}
```

SLUŽBENA REČ FINAL

Grupa pitanja u vezi sa konstantama u Javi.

30. Šta je final varijabla?

ODGOVOR: Varijabla obeležena rezervisanom rečju ***final*** ne može da se menja tj. predstavlja konstantu.

31.Šta je final metoda?

ODGOVOR: Ovakva metoda ne može da bude redefinisana.

32. Šta je final klasa?

ODGOVOR: Ovakva klasa ne može da bude nasleđena.

33. Šta je "blanko" finalna varijabla?

ODGOVOR: Konstanta koja nije inicijalizovana tokom vremena deklarisanja.

34. Da li je moguće inicijalizovati "blanko" finalnu varijablu?
ODGOVOR: Da, u konstruktoru ako je nestatička. Ako je statička moguće je izvršiti inicijalizovanje samo u statičkom bloku.

35. Da li je moguće deklarisati main() metodu kao final?

ODGOVOR: Da, moguće je kao: *public static final void main(String[] args){}.*

POLIMORFIZAM

Polimorfizan je jedan od osnovnih objektno - orijentisanih koncepata.

36. Šta predstavlja Runtime polimorfizam?

ODGOVOR: *Runtime polimorfizam* ili dinamičko distribuiranje metoda (dynamic method dispatch) predstavlja proces u kojem se poziv redefinisane metode dešava tokom vremena izvršavanja umesto vremena prevođenja. Tokom ovog procesa, redefinisana metoda se poziva varijablom koja referencira na super klasu. Sledi primer:

```
class Bike{
      void run(){System.out.println("running");}
    }
    class Splender extends Bike{
      void run(){System.out.println("running safely with 60km");}

      public static void main(String args[]){
        Bike b = new Splender();//upcasting
        b.run();
      }
    }
```

Razni primeri Runtime polimorfizma su dostupni na linku: http://www.javatpoint.com/runtime-polymorphism-in-java.

37. Da li je moguće postići Runtime polimorfizam pomoću atributa klase?

ODGOVOR: Nije moguće.

38. Koja je razlika između statičkog (static binding) i dinamičkog (dynamic binding) povezivanja?

ODGOVOR: U slučaju statičkog povezivanja objekat se realizuje tokom kompajliranja, dok u slučaju dinamičkog povezivanja, objekat se kreira tokom vremena izvršavanja programa.

APSTRAKCIJA

Sledi grupa pitanja kojima je pokrivena apstrakcija u Java programskom jeziku.

39. Šta predstavlja apstrakcija?

ODGOVOR: *Apstrakcija* predstavlja skrivanje detalja implementacije od korisnika pri čemu se korisniku demonstrira samo funkcionalnost.

40. Koja je razlika između apstrakcije i enkapsulacije?

ODGOVOR: Apstrakcija sakriva delove implementacije dok enkapsulacija vrši omotavanje koda i podataka u jedan modul.

41.Šta je apstraktna klasa?

ODGOVOR: Klasa koja je deklarisana primenom rezervisane reči ***abstract naziva*** se apstraktnom klasom. Sve njene metode moraju da budu redefinisane i ona ne može da bude instancirana.

42. Da li postoji apstraktna metoda u neapstraktnoj klasi?

ODGOVOR: Ne postoji. Ukoliko je bar jedna metoda klase apstraktna, klasa mora biti apstraktna.

43. Da li postoji metoda definisana sa abstract i final?

ODGOVOR: Ne postoji. Apstraktna metoda mora da se redefiniše.

44. Šta je interfejs?

ODGOVOR: *Interfejs* je skica klase koja poseduje statičke konstante i apstraktne metode. Pomoću *interfejsa* je moguće postići punu *apstrakciju* i *višestruko nasleđivanje*, budući da klasa može da implementira više *interfejsa*.

45.Da li metoda interfejsa može da bude statička?

ODGOVOR: Ne može. Interfejs je sam po sebi apstraktan, a ***abstract*** i ***static*** ne mogu da se kombinuju.

46. Da li interfejs može biti final?

ODGOVOR: Ne može jer mora da bude implementiran nekom klasom.

47. Šta je maker interfejs?

ODGOVOR: Interfejs koji ne sadrži članove klase i metode je *maker interfejs,* na primer: ***Serializable***, ***Cloneable*** ...

APSTRAKTA KLASA I INTERFEJS

Razlikovanje apstraktne klase i interfejsa je često eliminaciono pitanje.

Gotovo da nije moguće zamisliti intervju za posao Java programera bez pitanja u vezi sa isticanjem razlike između apstraktne klase i interfejsa. Ovo je često i eliminaciono pitanje.

48. Koja je razlika između apstraktne klase i interfejsa?

ODGOVOR: Odgovor je dat sledećom tabelom.

Apstraktna klasa	Interfejs
Apstraktna klasa može da sadrži telo metoda (neapstraktne metode)	Interfejs sadrži samo apstraktne metode
Apstraktna klasa može da ima varijable instanci	Interfejs ne može da ima varijable instanci
Apstraktna klasa može da ima konstruktor	Interfejs ne može da ima konstruktor
Apstraktna klasa može da ima statičku metodu	Interfejs ne može da ima statičke metode
Moguće je naslediti jednu apstraktnu klasu	Moguće je implementirati više interfejsa

Slika 7.2 Razlike između apstraktne klase i interfejsa [izvor: autor]

49. Da li je moguće primeniti modifikatore private i protected na varijable interfejsa?

ODGOVOR: Nije moguće, one su podrazumevano ***public***.

50. U kojem slučaju je moguće objektnu referencu povezati (cast) sa referencom interfejsa?

ODGOVOR: Navedeno je moguće u slučaju kada objekat implementira referencirani interfejs.

PAKETI

Sledi grupa pitanja koja je u vezi sa Java paketima.

51. Šta je paket?

ODGOVOR: Paket je grupa klasa, interfejsa i potpaketa sličnog tipa.

52. Da li je neophodno uvesti paket java.lang?

ODGOVOR: Nije potrebno, učitava ga automatski po osnovnim podešavanjima *JVM*.

53. Šta je static import?

ODGOVOR: Na ovaj način je moguće direktno pristupiti statičkim članovima klase, na primer:

```
import static java.lang.System.*;
    class StaticImportExample{
      public static void main(String args[]){

       out.println("Hello");//Now no need of System.out
       out.println("Java");

     }
    }
```

54. Da li je moguće više puta uključiti (import) isti paket / klasu?

ODGOVOR: Moguće je. Ni kompajler, ni *JVM* se neće buniti zbog toga - moguće je da od razvojnog okruženja se dobije upozorenje da je isti ***import*** više puta ponovljen. Bitno je da će klasa iz uključenog paketa biti učitana samo jednom.

RUKOVANJE IZUZECIMA

Rukovanje izuzecima je veoma važna Java problematika

1. Šta predstavlja rukovanje izuzecima (Exception Handling)?

ODGOVOR: Rukovanje izuzecima je mehanizam za rukovanje greškama tokom vremena izvršavanja.

2. Koja je razlika između izuzetaka tipa Checked Exceptioni Unchecked Exception?

ODGOVOR: Klase koje proširuju klasu ***Throwable***, sa izuzetkom klasa ***RuntimeException*** i ***Error***, predstavljaju izuzetke ***Checked Exception***, na primer: ***IOException***, ***SQLException*** i tako dalje.

Klase koje proširuju klasu ***RuntimeException*** predstavljaju izuzetke ***Unchecked Exception***, na primer: ***ArithmeticException***, ***NullPointerException*** i tako dalje.

Checked Exception se proveravaju tokom prevođena, a ***Unchecked Exception*** ne.

3. Koja klasa je bazična za Errori Exception?

ODGOVOR: Klasa ***Throwable***.

4. Da li je neophodno da svaki try blok bude praćen catch blokom?

ODGOVOR: Nije neophodno. ***Try*** blok bi trebalo da bude praćen ***catch*** ili f***inally*** blokom.

5. Šta je finally blok?

ODGOVOR: Prilikom obrade izuzetaka, blok ***finally*** se uvek izvršava.

6. Da li je moguće koristiti finally bez catch?

ODGOVOR: Moguće je.

7. U kojem slučaju se finally blok neće izvršiti?

ODGOVOR: U slučaju izlaska iz programa - na primer u slučaju poziva *System.exit()*.

DODATNA PITANJA ZA IZUZETKE

Važno pitanje je razlikovanje throw i throws.

8. Koja je razlika između throw i throws?

ODGOVOR: Odgovor se nalazi u sledećoj tabeli.

throw	throws
Koristi se da eksplicitno izbaci izuzetak	Koristi se da deklariše izuzetak
checked izuzeci ne mogu biti istaknuti samo sa throw	checked izuzeci mogu biti istaknuti sa throws
praćen je instacnom	praćen je klasom
koristi se unutar metode	koristi se sa potpisom metode
Nije moguće izbaciti više izuzetaka	Moguće je deklarisati više izuzetaka, na primer public void method()throws IOException,SQLException.

Slika 7.3 Razlika između throw i throws [izvor: autor]

9. Da li je moguće ponovo izbaciti neki izuzetak (rethrown)?

ODGOVOR: Moguće je.

10. Da li može redefinisana metoda potklase da deklariše izuzetak ukoliko on nije deklarisan u odgovarajućoj metodi roditeljske klase?

ODGOVOR: Može ali samo izuzetke tipa *unchecked exception*.

11. Šta je širenje izuzetaka (exception propagation)?

ODGOVOR: Prosleđivanje izuzetka objektu koji poziva metodu je širenje izuzetaka.

RUKOVANJE STRINGOVIMA

Veoma važan segment izučavanja Jave su Stringovi.

1. Šta znači pojam "immutable" u kontekstu Stringova?

ODGOVOR: Znači nemodifikovan ili nepromenjiv. Jednom kada se kreira ***String*** objekat ne može da se menja već se kreira nov string objekat. Pogledati i testirati na linku: http://www.javatpoint.com/immutable-string.

2. Zašto su Stringovi u Javi nepromenjivi?

ODGOVOR: Zato što Java koristi koncept ***String*** literala. Ako, na primer, postoji više varijabli koje ukazuju na neki ***String*** objekat i ako jedna varijabla promeni stanje objekta to će imati uticaja i na preostale varijable. Zato su stringovi ne promenjivi u Javi.

3. Koliko je objekata kreirano sledećim kodom?

```
String s1="Welcome";
    String s2="Welcome";
    String s3="Welcome";
```

ODGOVOR: Samo jedan.

4. Zašto Java koristi koncept String literala?

ODGOVOR: Da bi Java bila više memorijski efikasna. Ne kreira se nov objekat ukoliko već postoji u pulu ***String*** konstanti.

5. Koliko objekata kreira sledeći kod?

```
String s = new String("Welcome");
```

ODGOVOR: Kreira dva objekta i jednu varijablu reference.

6. Koja je razlika između objekata String i StringBuffer?

ODGOVOR: ***String*** je nepromenjiv, ***StringBuffer*** nije.

7.Koja je razlika između objekata StringBuilder i StringBuffer?

ODGOVOR: ***StringBuffer*** je sinhronizovna, ***StringBuilder*** nije.

8. Kako je moguće kreirati "immutable" (nepromenjivu) klasu?

ODGOVOR: Klase omotači i String klase su nepromenjive. Pogledati: http://www.javatpoint.com/how-to-create-immutable-class

9. Koji zadatak ima metoda toString()?

ODGOVOR: Ova metoda vraća ***String*** reprezentaciju bilo kojeg objekta.

UGNJEŽDENE KLASE I INTERFEJSI

Sledi rezime u vezi sa ugnježenim klasama i interfejsima.

1. Šta je ugnježdena klasa?

ODGOVOR: Klasa deklarisana unutar druge klase je ugnježdena klasa.

2. Koja je razlika između unutrašnje i ugnježdene klase?

ODGOVOR: Unutrašnja klasa je nestatička ugnježdena klasa.

3. Da li je moguće pristupiti nekonstantnoj lokalnoj varijabli unutar lokalne unutrašnje klase?

ODGOVOR: Lokalna varijabla mora da bude konstantna ukoliko joj se želi pristupiti lokalnom unutrašnjom klasom.

4. Šta je ugnježdeni interfejs?

ODGOVOR: Bilo koji interfejs deklarisan u drugom interfejsu ili klasi.

5. Da li može klasa da poseduje interfejs?

ODGOVOR: Da, ugnježdeni interfejs.

6. Da li interfejs može da poseduje klasu?

ODGOVOR: Da. Takva klasa ima status statičke ugnježdene klase. Sledi primer:

```
interface M{
      class A{}
    }
```

GARBAGE COLLECTION

Sledi grupa pitanja u vezi sa upravljanjem isjkorišćenih objekata.

1. Šta je Garbage Collection?

ODGOVOR: Proces uklanjanja iskorišćenih objekata tokom vremena izvršavanja.

2. Šta je gc()?
ODGOVOR: Radi se o *deamon thread.gc()* metodi definisanoj u klasi ***System*** koja se koristi za slanje zahteva *JVM* da izvede skupljanje otpada.

3. Koja je svrha metode finalize()?

ODGOVOR: Ova metoda se izvodi neposredno pre smeštanja objekta u *Garbage Collection.* Vrši zadatak čišćenja.

4. Da li nereferencirani objekat može ponovo biti referenciran?

ODGOVOR: Da.

5. Kojeg tipa niti je nit Garbage Collection?

ODGOVOR: *Daemon thread.*

6. Koja je razlika između final, finally i finalize?

ODGOVOR: Razlika je sledeća: ***final*** označava konstantu, ***finally*** označava blok za upravljanje izuzecima koji se uvek izvršava, ***finalize()*** metoda vrši čišćenje objekata iz *Garbage Collection*.

7. Koja je svrha klase Runtime?

ODGOVOR: Ova klasa obezbeđuje pristup *Java runtime* sistemu.

8. Kako se poziva eksterni proces u Javi?

ODGOVOR: Pozivom metode *Runtime.getRuntime().exec(?).*

ULAZNO / IZLAZNE VREDNOSTI, SERIJALIZACIJA, UMREŽAVANJE

Sledi grupa I/O pitanja, o serijalizaciji i umrežavanju.

1. Šta je serijalizacija?

ODGOVOR: Proces zapisivanja stanja objekata u bajt tok. Najčešće se koristi za prenos stanja objekata kroz mrežu.

2. Šta je deserijalizacija?

ODGOVOR: Proces rekonstrukcije objekata iz serijalizovanog stanja.

3.Šta predstavlja ključna reč transient?

ODGOVOR: Podatak koji je definisan kao ***transient*** ne može biti serijalizovan.

4. Šta je Externalizable?

ODGOVOR: ***Externalizable*** interfejs se koristi za upisivanje stanja objekata u bajt tok u kompresovanom formatu. Nije *marker interfejs*.

5. Koja je razlika između interfejsa Serializalble i Externalizable?

ODGOVOR: ***Serializable*** je marker interfejs, a ***Externalizable*** nije. Kada se koristi ***Serializable***, klasa je serijalizovana automatski. Moraju se redefinisati metode ***writeObject()*** i ***readObject***() za kontrolu kompleksnih procesa serijalizacije. Kada se koristi ***Externalizable*** postoji potpuna kontrola nad procesom serijalizacije klase.

6. Koja je razlika između hijerarhije klasa Reader/Writer i InputStream / OutputStream?
ODGOVOR: ***Reader*** / Writer hijerarhija klase je orijentisana ka karakterima, ***InputStream*** / ***OutputStream*** je orijentisana ka bajtovima.

7. Šta je I/O filter?

ODGOVOR: *I/O filter* je objekat koji čita iz jednog toka i upisuje u drugi.

8. Kako se konvertuje numerička IP adresa, npr. 192.18.97.39, u naziv hosta , npr. java.sun.com?

ODGOVOR: *InetAddress.getByName("192.18.97.39").getHostName()*

REFLEKSIJA I RAZNO

Sledi nova grupa pitanja koja se skoro uvek pojavljuje na razgovoru za posao.

1. Šta je refleksija?

ODGOVOR: Refleksija je proces proučavanja ili modifikovanja ponašanja klase tokom vremena izvršavanja. Koriste je razvojna okruženja, dibager, test alati i tako dalje.

2. Da li je moguće pristupiti privatnoj metodi iz spoljašnje klase?

ODGOVOR: Da promenom runtime ponašanja klase ukoliko klasa nije obezbeđena. Primer: http://www.javatpoint.com/how-to-call-private-method-from-another-class-in-java

3. Šta je klasa omotač?
ODGOVOR: Klasa omotač omogućava primitivnim tipovima pristup kao objektima.

4. Šta je nativna metoda?
ODGOVOR: Radi se o metodi koja je implemetirana u jezik koji nije Java.

5. Koja je svrha klase System?
ODGOVOR: Klasa ***System*** obezbeđuje pristup resursima sistema.

6. Šta je singleton klasa?
ODGOVOR: Za *singleton* klasu je karakteristično da u svakom trenutku datog vremena je prisutna tačno jedna instanca klase.

AWT I SWING, INTERNACIONALIZACIJA, JAVA ZRNO, RMI

Pitanja u vezi sa grafičkim funkcionalnostima Jave, internacionaliazcijom, zrnima i RMI

1. Koji kontejneri koriste BorderLayout kao podrazumevani menadžer rasporeda?

ODGOVOR: ***Window***, ***Frame*** i ***Dialog.***

2. Koji kontejneri koriste FlowLayout kao podrazumevani menadžer rasporeda?

ODGOVOR: ***Panel*** i ***Applet*** (napušta se).

3.Koja je razlika između Scrollbari ScrollPane?

ODGOVOR: ***Scrollbar*** je komponenta, a ***ScrollPane*** kontejner.

4. Šta je laka (lightweight) komponenta?

ODGOVOR: *Swing* komponenta. Ne dolaze sa nativnim pozivom za dobijanje grafičke jedinice. Dele grafičke jedinice roditeljskih komponenata.

5. Šta je teška (heavyweight) komponenta?

ODGOVOR: Postoji nativni poziv za dobijanje grafičkih jedinica, na primer *AWT*:

6. Šta je Locale?

ODGOVOR: ***Locale*** objekat prestavlja specifičan geografski, politički ili kulturni region.

7. Kako se učitava specifični lokalizam?

ODGOVOR: Pozivom *ResourceBundle.getBundle(?).*

8.Šta je JavaBean?

ODGOVOR: Višestruko upotrebljiva softverska komponenta programirana u Javi dizajnirana da se njom manipuliše vizuelno okruženjem za razvoj softvera.

9. Da li mogu da sarađuju aplikacije bazirane na RMI i Corba?

ODGOVOR: Mogu. *RMI* je dostupan sa *IIOP* kao transportnim protokolom umesto *JRMP.*

VIŠENITNOST

Grupa pitanja za rad sa nitima u Javi.

1. Šta je Višenitnost?

ODGOVOR: Proces izvršavanja većeg broja niti simultano.

2. Šta je nit?

ODGOVOR: Lak potproces i odvojena putanja izvršavanja.

3. Šta radi metoda join()?

ODGOVOR: Metoda čeka da nit "umre". Izaziva da sve aktivne niti prestanu sa izvršavanjem dok nit kojoj se pristupa ne završi u potpunosti svoj zadatak.

4. Koja je razlika između wait() i sleep() metode?

ODGOVOR: Odgovor je sledećoj tabeli.

wait()	sleep()
Definisana u klasi Object	Definisana u klasi Thread
Oslobađa lock	Ne oslobađa lock

Slika 7.4 Razlika između wait() i sleep() metode [izvor: autor]

5. Da li je moguće pokrenuti nit dvaput?

ODGOVOR: Ne. Doći će do izbacivanja izuzetka.

6. Da li je moguće pokrenuti metodu run() umesto start()?

ODGOVOR: Da, ali neće raditi kao nit već kao normalan objekat i neće biti zamene konteksta između niti.

7. Šta su deamon niti?

ODGOVOR: Niti niskog prioriteta koje obezbeđuju pozadinsku podršku korisničkoj niti.

8. Kada je neophodno prekinuti (interupt) nit?

ODGOVOR: Koristi se ukoliko je neophodno probuditi nit iz stanja ***sleep*** ili ***wait***.

SINHRONIZACIJA NITI

Dodatna pitanja za rad sa nitima.

9. Šta je sinhronizacija?

ODGOVOR: Mogućnost kontrole pristupa većeg broja niti bilo kojem deljenom resursu. Koristi se za: sprečavanje mešanja niti i sprečavanje problema konzistentnosti.

10. Koja je svrha sinhronizovanog bloka?

ODGOVOR: Sinhronizovani blok se koristi za zaključavanje objekata za bilo koji deljeni resurs. Sinhronizovani blok ima užu oblast nego metoda.

11. Da li Java objekat može da bude zaključan za ekskluzivnu upotrebu datom niti?

ODGOVOR: Da, ukoliko se objekat spakuje u sinhronizovan blok. Zaključan objekat je nedostupan drugim nitima osim onoj koja eksplicitno ima privilegiju pristupa objektu.

12. Šta je statička sinhronizacija?

ODGOVOR: U slučaju statičke sinhronizacije vrši se zaključavanje klase umesto objekta.

13. Koja je razlika između notify() i notifyAll()?

ODGOVOR: Metodom ***notify()*** će biti odblokirana jedna nit na čekanju, dok će sa ***notifyAll()*** biti odblokirane sve niti koje čekaju.

14. Šta je deadlock?

ODGOVOR: To je situacija kada dve niti čekaju jedna drugu da oslobode resurse. Svaka nit čeka na resurs kojeg je zauzela druga nit na čekanju.

JAVA KOLEKCIJE

Veoma bitna grupa pitanja u svakom intervjuu.

1. Koja je razlika između ArrayList i Vector klasa?

ArrayList	Vector
Nije sinhronizovana	Sinhronizovana
Od početka pripada Java kolekcijama	Predstavlja tzv. "legacy" klasu
Po potrebi povećava svoju veličinu za 50% veličine niza	Po potrebi povećava svoju veličinu dupliranjem veličine niza

Slika 7.5 Razlika između ArrayList i Vector klasa [izvor: autor]

2. Koja je razlika između ArrayList i LinkedList klasa?

ArrayList	LinkedList
Koristi dinamički niz	Koristi dvostruko povezanu listu
Neefikasan za manipulisanje	Efikasan za manipulisanje
Bolji za čuvanje podataka	Bolji za manipulisanje podacima

Slika 7.6 Razlika između ArrayList i LinkedList klasa [izvor: autor]

3. Koja je razlika između Iteratota i ListIteratora?

Iterator	ListIterator
Iterator prolazi kroz elemente kolekcije u direktnom smeru unapred	ListIterator prolazi kroz elemente kolekcije u direktnom smeru unapred i unazad
Koriste ga liste, setovi u redovi	Koriste ga samo liste

Slika 7.7 Razlika između Iteratota i ListIteratora [izvor: autor]

4. Koja je razlika između Listei Seta?

ODGOVOR: Lista može da sadrži elemente sa ponavljanjem, set ne može.

5. Koja je razlika izmeću Set i Map?

ODGOVOR: Set sadrži vrednosti, a mapa parove (ključ, vrednost).

6. Koja je razlika između HashSeti TreeSet?

ODGOVOR: ***HashSet*** ne održava redosled, a ***TreeSet*** održava rastući redosled.

7. Koja je razlika između HashSet i HashMap?

ODGOVOR: ***HashSet*** sadrži vrednosti, a ***HashMap*** parove (ključ, vrednost). Moguće je vršiti iteraciju kroz ***HashSe***t, dok ***HashMap*** mora da bude konvertovana u ***Set*** pre iteracije.

8. Koja je razlika između HashMapi Hashtable?

HashMap	Hashtable
Sinhronizovana je	Nije sinhronizovan
Može da sadrži jedan null ključ i više null vrednosti	Može da sadrži bilo koji null ključ ili vrednost

Slika 7.8 Razlika između HashMap i Hashtable [izvor: autor]

9. Koja je razlika između Collectioni Collections?

ODGOVOR: ***Collection*** je interfejs, a ***Collections*** klasa. ***Collection*** obezbeđuje normalno funkcionisanje struktura podataka za ***List***, ***Set*** i ***Queue***. Klasa ***Collections*** obezbeđuje sortiranje i sinhronizaciju elemenata kolekcije.

SINHRONIZOVANE KOLEKCIJE

Sledi dodatna grupa pitanja u vezi sa kolekcijama.

10. Kako se sinhronizuju elementi List, Seti Map?

```
public static List synchronizedList(List l){}
public static Set synchronizedSet(Set s){}
public static SortedSet synchronizedSortedSet(SortedSet s){}
public static Map synchronizedMap(Map m){}
public static SortedMap synchronizedSortedMap(SortedMap m){}
```

Slika 7.9 Sinhronizacija elemenata liste, seta i mape [izvor: autor]

11. Koja je prednost primene generičke kolekcije?

ODGOVOR: Kada se koristi generička klasa nije potreban tzv. typecasting. Ona je bezbedna po tipu (typesafe) i proverena tokom kompajliranja.

12. Šta je klasa Dictionary?

ODGOVOR: Ova klasa obezbeđuje mogućnost skladištenja parova (ključ, vrednost).

13. Koja je podrazumevana veličina faktora učitavanja u heš kolekcijama?

ODGOVOR: Podrazumevana veličina faktora učitavanja 0.75.

JDBC

Sledi grupa pitanja u vezi sa bazama podataka u Javi.

1. Šta je JDBC?

ODGOVOR: *Java API* koji omogućava konektovanje i izvršavanje upita nad bazom podatak.

2. Šta je JDBC drajver?

ODGOVOR: Softverska komponenta koja omogućava JAVA aplikaciji da komunicira sa bazom podataka.

3.Šta su JDBCAPI komponente?

ODGOVOR:

Interfejsi: ***Connection***, ***Statement***, ***PreparedStatement***, ***ResultSet***, ***ResultSetMetaData***, ***DatabaseMetaData***, ***CallableStatement*** i drugi.

Klase: ***DriverManager***, ***Blob***, ***Clob***, ***Types***, ***SQLException*** i druge.

4. Koji su JDBC iskazi?

ODGOVOR: ***Statement***, ***PreparedStatement*** i ***CallableStatement.***

5. Koja je uloga klase JDBC DriverManager?

ODGOVOR: Upravlja registrovanim drajverima.

6. Šta radi JDBC Connection interfejs?

ODGOVOR: Održava sesiju sa bazom podataka.

7. Šta radi JDBC ResultSet interfejs?

ODGOVOR: ***ResultSet*** objekat reprezentuje vrstu u tabeli.

VIDEO MATERIJAL

Top 10 JAVA Interview Questions and Answers - video materijal

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

Poglavlje 8

Korisni linkovi za unapređenje Java EE znanja

UNAPREDITE ZNANJE

Ideja je da prolaskom kroz sledeće linkove budete korak bliže profesionalcima.

Ideja je da prolaskom kroz sledeće linkove budete korak bliže profesionalcima. Linkovi obuhvataju i Java platforme 9 - 12 koje. zapravo predstavljaju različite verzije jedne iste platforme, kao i nov IDE pod nazivom Apache NetBeans. Platforme i razvojno okruženje su još uvek u test fazi i u svetu profesionalaca Java 8 je i dalje dominantna verzija Java platforme.

Unapredite i proširite vaše znanje izučavanjem materijala sa sledećih linkova:

- Apache NetBeans (incubating) 11.0 Features - https://netbeans.apache.org/download/nb110/index.html
- Apache NetBeans 11.0 adds support for JDK 12, Java EE, Gradle, and more - https://jaxenter.com/apache-netbeans-11-0-157695.html
- Java EE with NetBeans 10 - https://www.agilejava.eu/2019/01/07/java-ee-with-netbeans-10/
- Apache NetBeans 11.1 Just released! - https://www.jee.gr/apache-netbeans-11-1-released-with-payara/

VIDEO MATERIJAL

Apache NetBeans 11.1, Payara, Java EE 8 - YouTube (trajanje:

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

Poglavlje 9

Pokazni primer intervjua sa Java poslodavcem

TEST 1 (30 MIN)

Primer testa dobijenog od poznatog poslodavca.

1. Koja je razlika između JDK, JRE i JVM?

ODGOVOR: JVM je skraćenica za Java virtuelnu mašinu, apstraktnu mašinu na kojoj se izvršava Java bajt kod (byte code). JVM je dostupna za sve poznate hardversko - softverske platforme.

JRE je skraćenica za Java Runtime Environment i predstavlja implementaciju Java virtuelne mašine.
JDK je skraćenica za Java Development Kit i sadrži JRE i razvojne alate.

2.Šta je platforma?

ODGOVOR: Platforma predstavlja hardversko i softversko okruženje na kojem se program izvršava. Java predstavlja softverski baziranu platformu.

3.Zašto su Stringovi u Javi nepromenjivi?

ODGOVOR: Zato što Java koristi koncept String literala. Ako, na primer, postoji više varijabli koje ukazuju na neki String objekat i ako jedna varijabla promeni stanje objekta to će imati uticaja i na preostale varijable. Zato su stringovi ne promenjivi u Javi.

4. Šta predstavlja rukovanje izuzecima (Exception Handling)?

ODGOVOR: Rukovanje izuzecima je mehanizam za rukovanje greškama tokom vremena izvršavanja

5. Šta predstavlja apstrakcija?
ODGOVOR: Apstrakcija predstavlja skrivanje detalja implementacije od korisnika pri čemu se korisniku demonstrira samo funkcionalnost.

6. Šta je Locale?

ODGOVOR: Locale objekat prestavlja specifičan geografski, politički ili kulturni region.

7. Šta je sinhronizacija?

ODGOVOR: Mogućnost kontrole pristupa većeg broja niti bilo kojem deljenom resursu. Koristi se za: sprečavanje mešanja niti i sprečavanje problema konzistentnosti.

8.Koja je razlika između HashSeti TreeSet?

ODGOVOR: HashSet ne održava redosled, a TreeSet održava rastući redosled.

9. Šta je JDBC drajver?

ODGOVOR: Softverska komponenta koja omogućava JAVA aplikaciji da komunicira sa bazom podataka.

Poglavlje 10

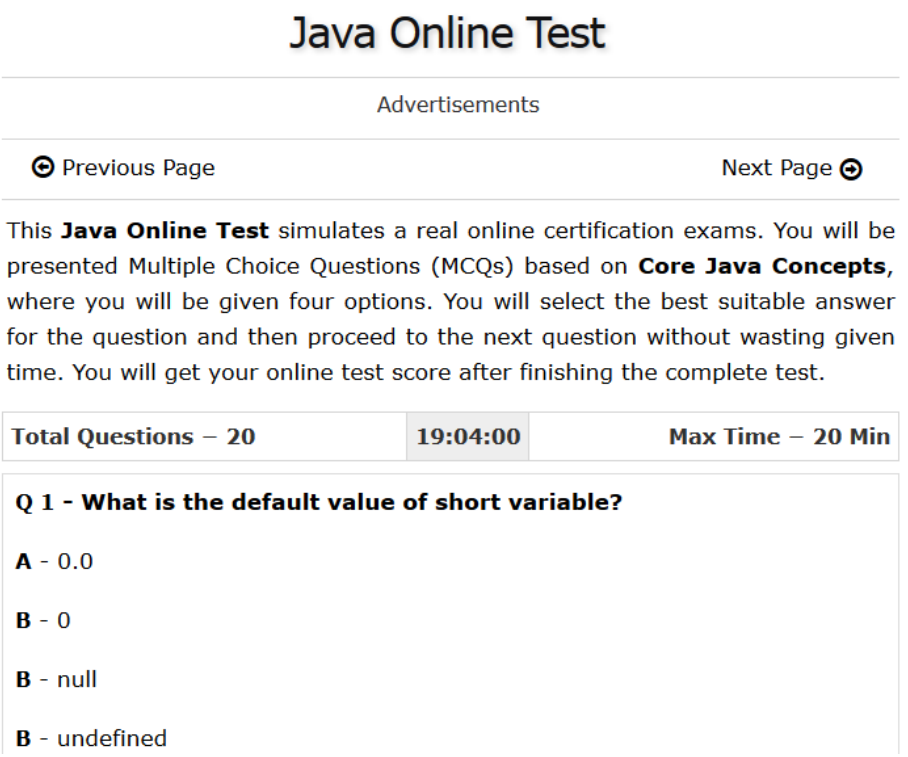
Individualna vežba 10

INDIVIDUALNA VEŽBA (135 MIN)

Pokušajte sami.

Pokušajte samostalno da proverite vaše znanje iz programskog jezika Java i Java tehnologija izradom simuliranog intervjua za posao sa linka:

https://www.tutorialspoint.com/java/java_online_test.htm



Slika 10.1 Simulirani intervju za posao [izvor: tutorialspoint]

Na kraju ovog objekta učenja preuzmite zadatke koji su dati u sekcijama "Zadaci za samostalni rad" u prethodnim lekcijama.

Poglavlje 11

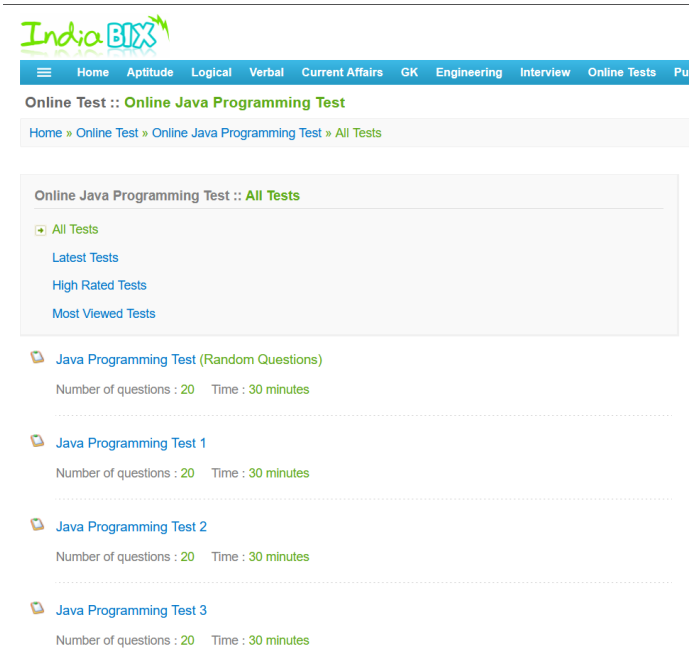
Domaći zadatak 15

ZADATAK 1 (120 MIN)

Popunjavanje online testa kao primera intervjua za posao

1. Popuniti sledeći online test za simuliranje razgovora sa poslodavcem: http://www.indiabix.com/online-test/java-programming-test/ (slika 1);
2. Potrebno je da uradite po tri različita testa iz ponuđene liste testova;
3. Rezultate testiranja dokumentovati kao pdf ili doc dokument.

Budući da postoji velika baza znanja i da će studenti dobijati različite testove za DZ u ovom slučaju nema potrebe za dobijanjem različitih zadataka na mail od asistenta.



Slika 11.1 Ponuđena lista testova [izvor: http://www.indiabix.com/]

Poglavlje 12

Zaključak

ZAKLJUČAK

Izradom JEE projekta je zaokruženo izučavanje razvoja JEE veb distribuiranih aplikacija

U dosadašnjim lekcijama su prikazani i objašnjeni napredni *Jave EE* koncepti i principi, kao mehanizmi i alati za kreiranje distribuiranih veb aplikacija. Upravo, ovaj deo lekcije je poslužio kao rekapitulacija prethodnih lekcija, uvodeći razvoj složenije distribuirane Java veb aplikacije, zajedno sa pratećom projektnom dokumentacijom. Razvijena aplikacija je u potpunosti funkcionalna i spremna za konkretnu upotrebu u poslovnom subjektu.

Dakle, kao što je rečeno,prvi deo lekcije se baviti definisanjem projektnog zadatka i pripremom odgovarajuće dokumentacije. Dokumentacija je obuhvatila primere potrebnih tabela, dijagrama i izveštaja koji se sreću u izradi realnih projekata. Posebno, lekcija se fokusirala na izradu i detaljno dokumentovanje programskog koda i podešavanja aplikacije koja je predmet projekta. Na kraju, ovog dela izlaganja, aplikacija je i demonstrirana, a prikazane su i slike koje demonstriraju funkcionalnosti ove aplikacije.

Budući da ovaj predmet predstavlja zaokruživanje izučavanja problematike razvoja softvera nad *Java* platformom i programskim jezikom, drugi deo lekcije je stavio akcenat na pripremu za odlazak na intervju kod poslodavca koji zapošljava Java programere. U tu svrhu su detaljno obrazložena brojna pitanja, prikupljena sa više relevantnih izvora, i koja predstavljaju pitanja koja potencijalni kandidati dobijaju prilikom razgovora za posao.
Savladavanjem ove lekcije student je osposobljen da samostalno kreira i dokumentuje Java EE projekat, kao i da potpuno spreman se pojavi na intervjuu kod poslodavca.

LITERATURA

Za izradu projekta korišćena je najnovija JEE pisana i elektronska literatura

1. Eric Jendrock, Ricardo Cervera - Navarro, Ian Evans, Kim Haase, William Markito. 2014. Java Platform, Enterprise Edition The Java EE Tutorial, Release 7, ORACLE
2. David R. HeffelFinger. 2015. Java EE7 Develpomnet With NetBeans 8, PACK Publishing
3. Yakov Fain. 2015. Java 8 programiranje, Kombib (Wiley)
4. Josh JUneau. 2015. Java EE7 Recipes, Apress
5. https://www.tutorialspoint.com/jsf/

6. https://www.tutorialspoint.com/jpa/
7. http://www.mysqltutorial.org/mysql-sample-database.aspx
8. Apache NetBeans (incubating) 11.0 Features - https://netbeans.apache.org/download/nb110/index.html
9. Apache NetBeans 11.0 adds support for JDK 12, Java EE, Gradle, and more - https://jaxenter.com/apache-netbeans-11-0-157695.html
10. Java EE with NetBeans 10 - https://www.agilejava.eu/2019/01/07/java-ee-with-netbeans-10/
11. Apache NetBeans 11.1 Just released! - https://www.jee.gr/apache-netbeans-11-1-released-with-payara/
12. http://www.indiabix.com/online-test/java-programming-test/