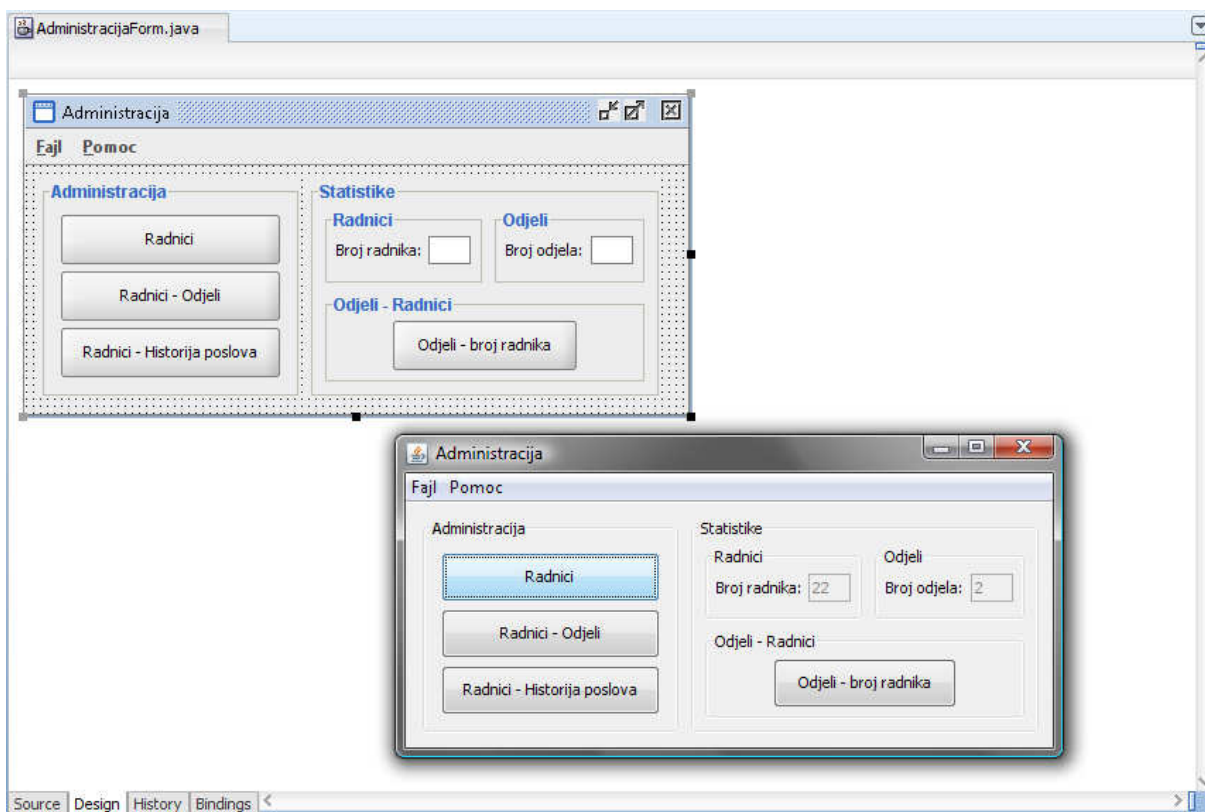


Kreiranje Desktop aplikacije uz pomoć ADF Swing i ADF Biznis Komponenti

Glavna forma je prikazana na slijedećoj slici:



Slika 113. Izgled forme *Administracija*

Na glavnoj formi dodana su 2 panela i to:

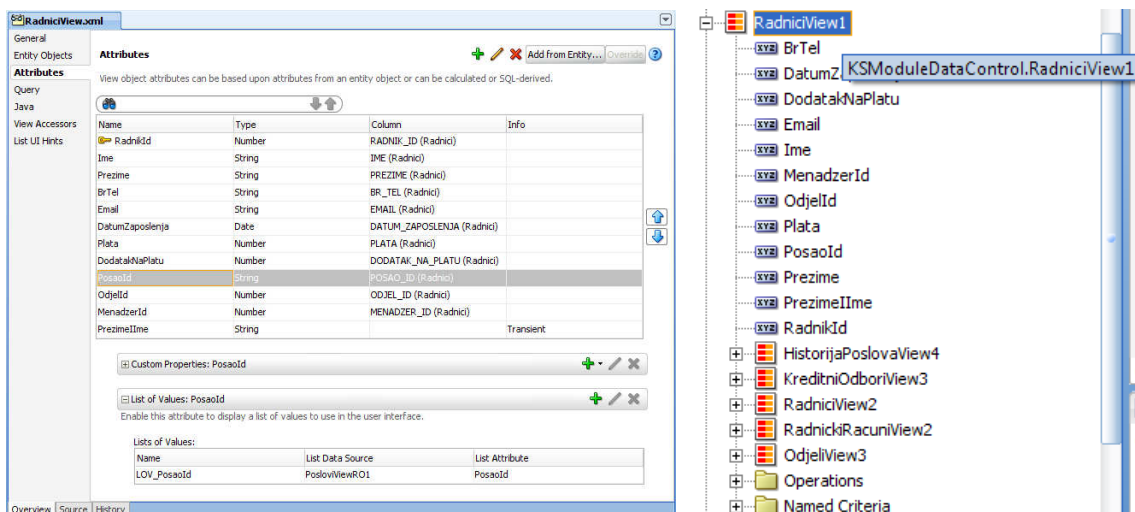
- ✓ *Administracija* (sadrži tri dugmeta *Radnici*, *Radnici – Odjeli*, *Radnici – Historija Poslova*)
- ✓ *Statistike* (sadrži tri panela, panel *Radnici* – u kojem je prikazan podatak o ukupnom broju radnika u svim odjelima; panel *Odjeli* – u kojem je prikazan podatak o ukupnom broju odjela u bazi podataka; panel *Odjeli – Radnici* koji sadrži dugme *Odjeli – broj radnika*).

Da bi kreirali prethodno spomenuta dugmad, morali smo kreirati 4 panela. Prevlačenjem kreiranih panela na glavnu formu, automatski se kreiraju dugmad tako da dugmad *Radnici*, *Radnici – Odjeli*,

Radnici – Historija poslova i Odjeli – broj radnika odgovaraju panelima: *DodavanjeRadnikaPanel.java*, *OdjeliRadniciPanel.java*, *RadniciHistorijaPoslova.java* i *StatistikePanel.java*, respektivno.

DodavanjeRadnikaPanel

Nakon klika na dugme *Radnici* otvorit će se panel *DodavanjeRadnikaPanel.java* koji se kreira na osnovu *RadniciView* objekta (*RadniciView.xml*) odnosno *ModuleDataControl* objekta *RadniciView1*, što je prikazano na slici 114.



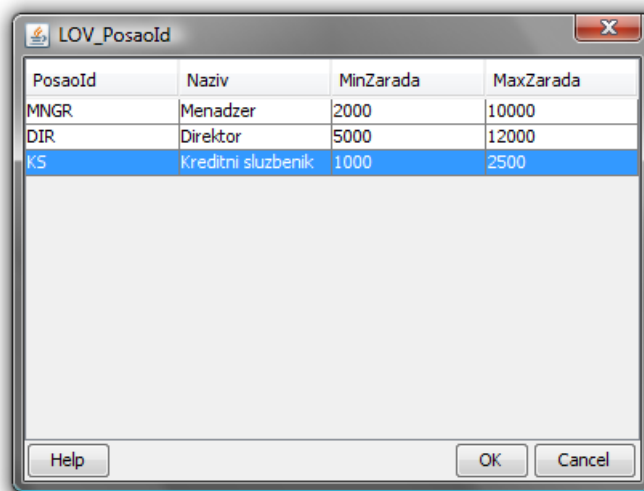
Slika 114. *RadniciView.xml* (lijevo) i *RadniciView1* Data Control objekat (desno)

Prevlačenjem komponenti iz *RadniciView1* Data Control objekta, dobijamo slijedeći izgled *DodavanjeRadnikaPanel* panela:

Slika 115. Izgled panela *DodavanjeRadnikaPanel* nakon klika na dugme *Radnici*

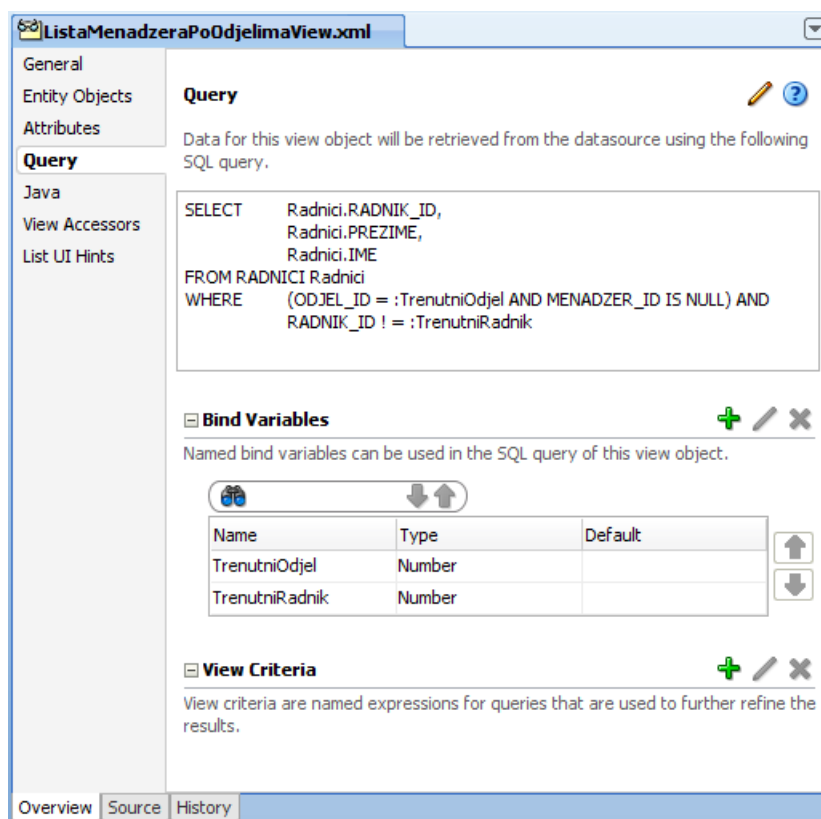
Funkcija ovog panela jeste **kreiranje novog radnika** (*dodavanje novog reda*), **brisanje/ažuriranje već postojećeg**. Sve ove funkcionalnosti su omogućene od strane *Navigation Bar-a* . Opcije koje pruža ova komponenta su: kretanje kroz slogove tabele *Radnici* u oba smjera , pozicioniranje na posljednji slog , pozicioniranje na prvi slog , dodavanje i brisanje sloga odnosno reda u tabeli *Radnici*, *Commit* i poništenje svih promjena koje su se desile od posljednjeg commita kao i pretraga tabele po već unaprijed definisanom SQL upitu (u našem slučaju, kriterij je definisan tako da se pretraga vrši po imenu i/ili prezimenu; moguće je dodavanje više kriterija pretrage). Dodavanje novog radnika vrši se klikom na tek nakon što se pozicioniramo na posljednji slog.

Kao što možemo vidjeti olakšan je *unos novih podataka* u polja radnika. Polje u koje se upisuje id posla (*POSJO_ID*) predstavlja *LOV (List Of Value)* polje koje se definiše na nivou atributa *Posaoid* unutar *RadniciView Object-a*. Klikom na prikazat će nam se slijedeći prozor gdje možemo izabrati željeni posao:

Slika 116. Izgled *LOV_PosaoId* panela

U *LOV_PosaoId* su prikazane slijedeće kolone sa podacima: *PosaoId*, *Naziv*, *MinZarada* i *MaxZarada* koji su mapirani u *POSAO_ID*, *NAZIV*, *MIN_ZARADA*, *MAX_ZARADA* respektivno (mapiranje se vrši zato što je interakcija korisnika sa bazom podataka realizovana preko pogleda). Na taj način korisniku je olakšan posao unosa ID-a i korisnik unaprijed zna u kojim granicama se kreće plata radnika. Odabirom jednog od redova i klikom na dugme *OK* kreirali smo novi unos u polje *ID Posla*. Klikom na *Cancel* akcija se poništava.

Vrijednost plate je ograničena na opseg od 800 – 10000 KM, korištenjem *JSpinner* komponente (korak = 50). Unos vrijednosti u polje označeno labelom *Dodatak na platu* vrši se preko *Combo Box-a* u kojem su unaprijed definisane vrijednosti od 0.10 – 0.75. Unos id-a za odjel i menadžera se vrši preko *Combo Box-a* i da se primijetiti da je i u ovom slučaju unos pojednostavljen (mogućnost greške pri unosu je smanjen). Polje za unos ID menadžera je realizovano pomoću *LOV* polja u obliku *Combo Box-a*. Prethodno je bilo potrebno kreirati pogled *ListaMenadzeraPoOdjelimaView* za koji smo iskoristili SQL upit prikazan na slici 117.



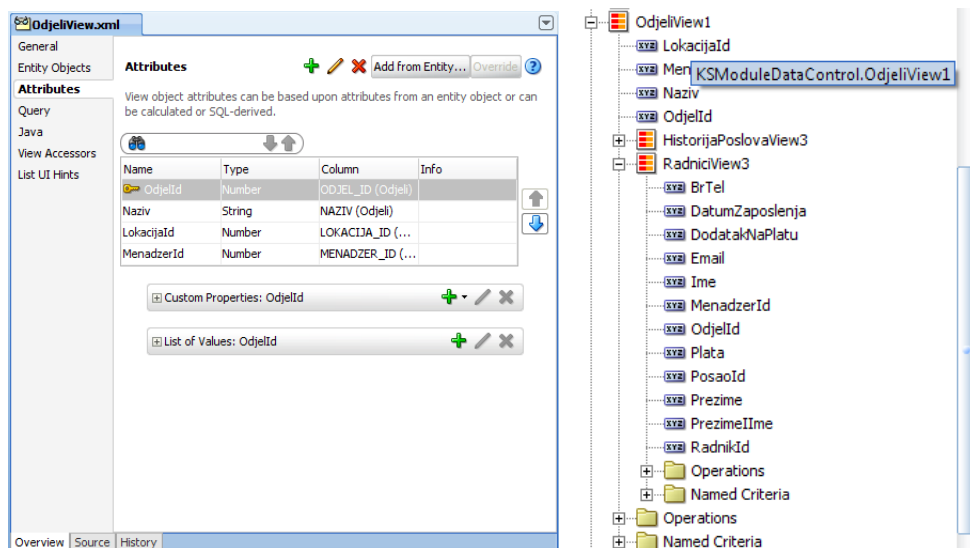
Slika 117. ListaMenadzeraPoOdjelimaView.xml

U pogledu su iskoristene tzv. *bind varijable* *TrenutniRadnik* i *TrenutniOdjel* koje će dobiti svoju vrijednost tek nakon što korisnik klikne na *Combo box*. Nakon što korisnik klikne na *Combo Box* prikazat će mu se svi menadžeri istog odjela, nakon čega korisnik može izabrati željenog menadžera.

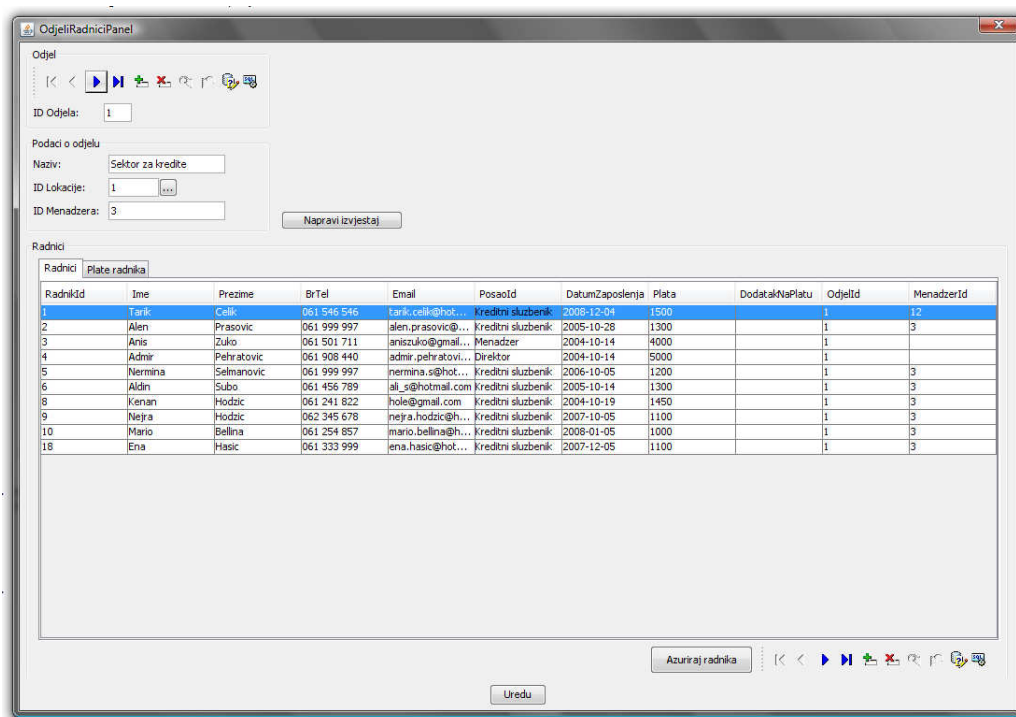
Dugme *Uredi* je automatski dodano nakon prevlačenja panela na glavnu formu. Klikom na *Uredi* zatvara se prozor, sve akcije koje su prethodno urađene bit će commit-ovane.

OdjeliRadniciPanel

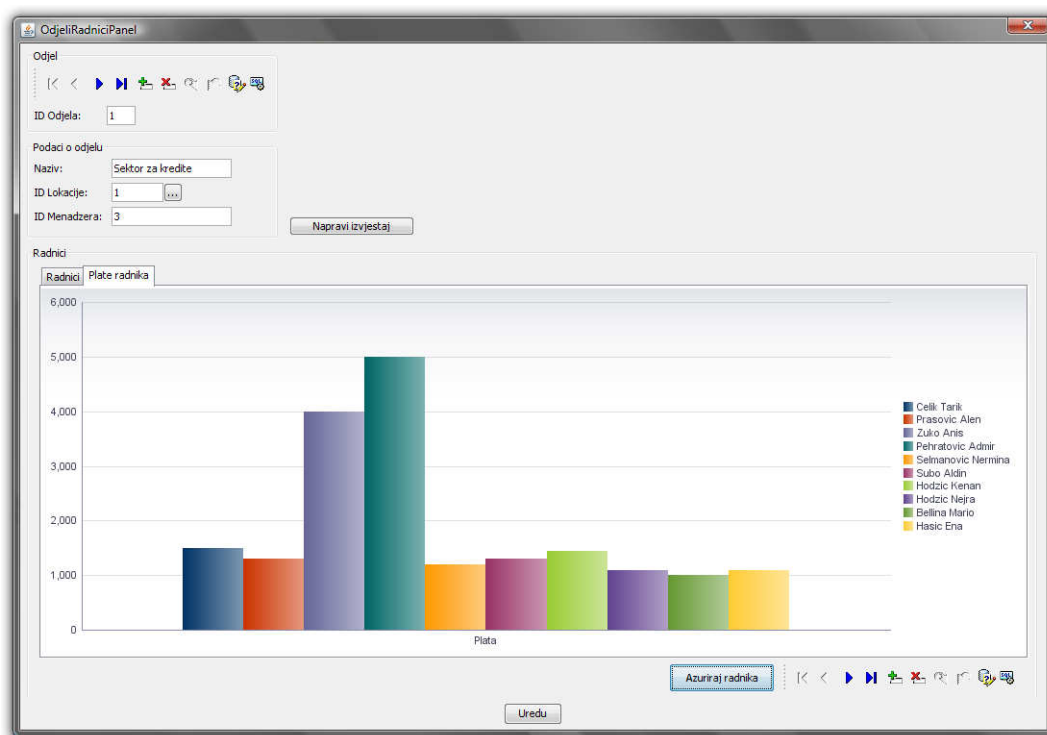
Nakon klika na dugme *Radnici – Odjeli* otvorit će se panel *OdjeliRadniciPanel.java* koji se kreira na osnovu *OdjeliView* objekta odnosno *ModuleDataControl* objekta *OdjeliView1*, što je prikazano na slici 118.

Slika 118. *OdjeliView.xml* (lijevo) i *OdjeliView1* Data Control objekat (desno)


Prevlaćenjem komponenti iz *OdjeliView1* Data Control objekta, dobijamo slijedeći izgled *OdjeliRadniciPanel* panela:

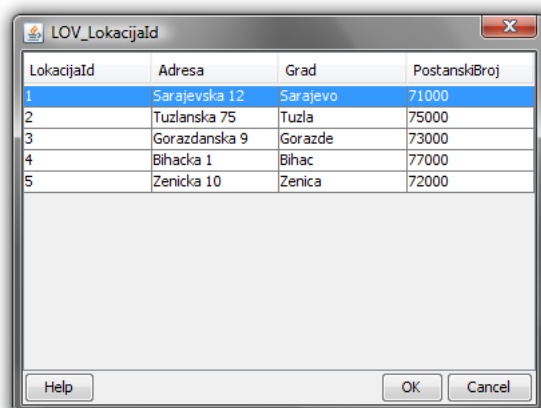
Slika 119. Izgled *OdjeliRadniciPanel* panela (selektovan *Radnici* tab)

U panelu *OdjeliRadniciPanel* je omogućeno dodavanje, brisanje, ažuriranje odjela, radnika, prikaz radnika po odjelima kao, grafički prikaz njihovih plata unutar jednog odjela, kreiranje izvještaja za selektovani odjel kao i pretraživanje radnika po imenu i/ili prezimenu. Za te funkcionalnosti iskorištene su navigacijske komponente (*Navigation Bar*) kao i *JTabbedPane* komponenta u kojoj se kreiraju tabela radnika (tab *Radnici*), graf komponenta (tab *Plate radnika*). Izgled *OdjeliRadniciPanel* panela kada je selektovan tab *Plate Radnika* prikazan je na slijedećoj slici:



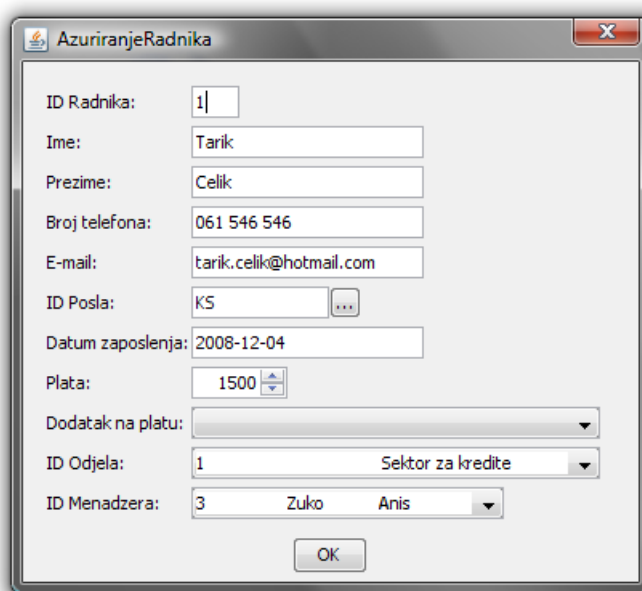
Slika 120. Izgled *OdjeliRadniciPanel* panela (selektovan *Plate radnika* tab)

Polje u koje se upisuje id lokacije predstavlja LOV (*List Of Value*) polje koje se definiše na nivou atributa *Lokacijald* unutar *OdjeliView* objekta. Klikom na  prikazat će nam se slijedeći prozor (panel) gdje možemo izabrati željenu lokaciju:



Slika 121. Izgled LOV_LokacijaId panela

Odabirom jednog od redova i klikom na dugme *OK* kreirali smo novi unos u polje *id lokacije*. Klikom na *Cancel* akcija se poništava. U panelu *OdjeliRadniciPanel* kreirano je još jedno dugme *Azuriraj radnika*. Klikom na *Azuriraj radnika* otvara nam se novi panel u kojem možemo izvršiti željene promjene na poljima selektovanog radnika (slika 122).



Slika 122. Izgled AzuriranjeRadnika panela

Panel *AzuriranjeRadnika* je sličan *DodavanjeRadnikaPanel* panelu. Nakon što smo izvršili odgovarajuće izmjene u poljima radnika klikom na *OK* vraćamo se na *OdjeliRadniciPanel* panel.

Klikom na dugme *Napravi izvještaj* kreira se izvještaj za trenutno selektovani odjel nakon čega je moguć pregled, spasavanje, printanje istog (slika 123).



sarajevobank

Id odjela: 1
 Naziv: Sektor za kredite
 Kanton: Kanton Sarajevo
 Postanski broj: 71000
 Grad: Sarajevo
 Adresa: Sarajevska 12

ID	Prezime	Ime	E-mail	Plata (KM)	Posao
1	Celik	Tarik	tarik.celik@hotmail.com	1500	Kreditni sluzbenik
2	Prasovic	Alen	alen.prasovic@gmail.com	1300	Kreditni sluzbenik
3	Zuko	Anis	aniszuko@gmail.com	4000	Menadzer
4	Pehratovic	Admir	admir.pehratovic@gmail.com	5000	Direktor
5	Selmanovic	Nermina	nermina.s@hotmail.com	1200	Kreditni sluzbenik
6	Subo	Aldin	ali_s@hotmail.com	1300	Kreditni sluzbenik
8	Hodzic	Kenan	hole@gmail.com	1450	Kreditni sluzbenik
9	Hodzic	Nejra	nejra.hodzic@hotmail.com	1100	Kreditni sluzbenik
10	Bellina	Mario	mario.bellina@hotmail.com	1000	Kreditni sluzbenik
18	Hasic	Ena	ena.hasic@hotmail.com	1100	Kreditni sluzbenik

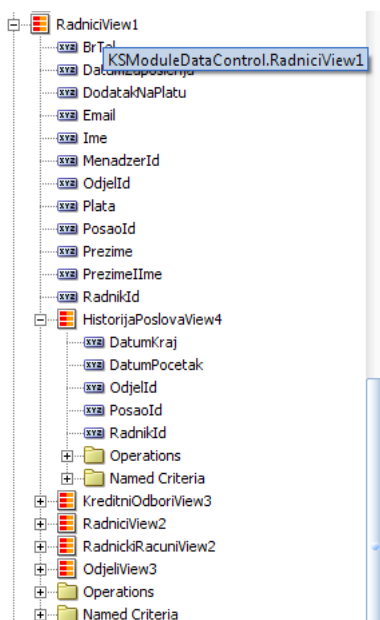
Stranica 1 of 1

Slika 123. Izgled kreiranog izvještaja za selektovani odjel

Kreiranja izvještaja će na kraju ovog rada biti opisano.

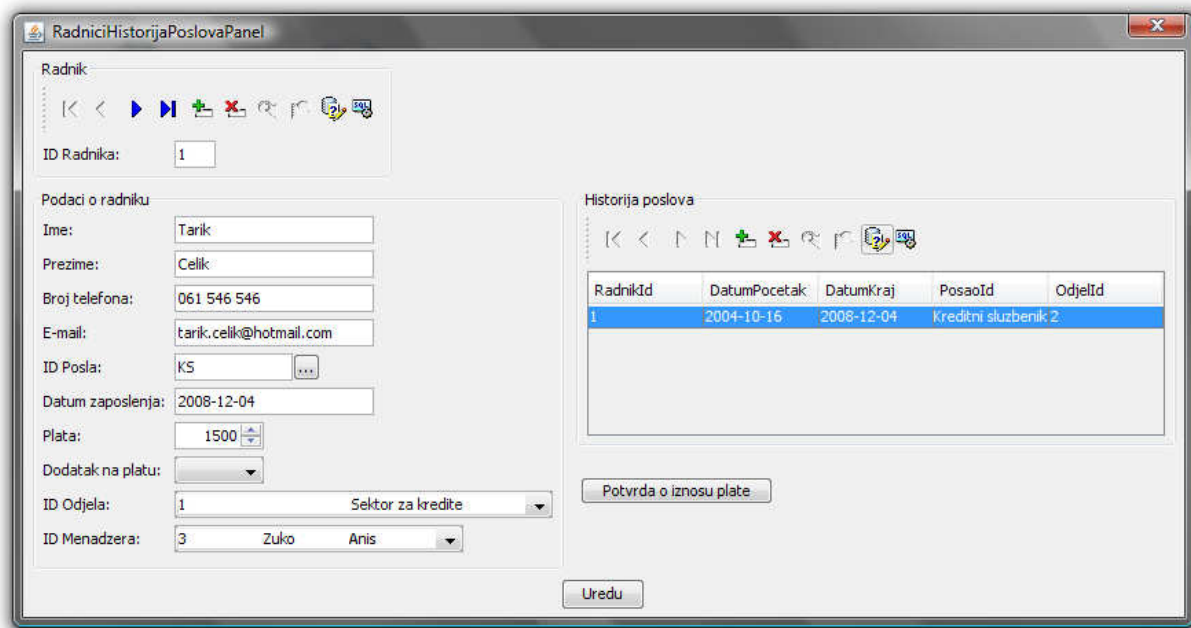
RadniciHistorijaPoslovaPanel

Nakon klika na dugme *Radnici – Historija poslova* otvorit će se panel *RadniciHistorijaPoslovaPanel.java* koji se kreira na osnovu *RadniciView* objekta odnosno *ModuleDataControl* objekta *RadniciView1* koji je prikazan na slijedećoj slici:



Slika 124. *RadniciView1* Data Control objekat koji se nalazi unutar *KSMModuleDataControl*

Prevlaćenjem komponenti iz *RadniciView1* Data Control objekta, dobijamo slijedeći izgled *RadniciHistorijaPoslovaPanel* panela:

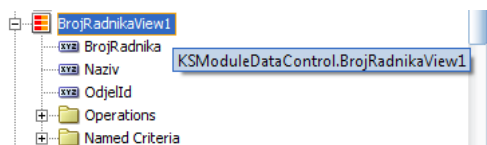


Slika 125. Izgled *RadniciHistorijaPoslovaPanel* panela

RadniciHistorijaPoslovaPanel je sličan prethodno kreiranim panelima. Moguće je kreiranje novih radnika, brisanje, ažuriranje već postojećih. U tabeli *Historija poslova* se čuvaju podaci o historiji poslova za selektovanog radnika. Svako ažuriranje polja koje predstavlja id odjela ili id posla rezultirat će ažuriranjem tabele *Historija poslova*. Također, dodano je dugme *Potvrda o iznosu plate*. Klikom na ovo dugme generisat će se Jasper Repot (izvještaj). Na kraju rada će biti opisano kreiranje Jasper izvještaja.

StatistikePanel

Nakon klika na dugme *Odjeli – broj radnika* otvorit će se panel *StatistikePanel.java* koji se kreira na osnovu prethodno kreiranog *BrojRadnikaView* objekta, odnosno *ModuleDataControl* objekta *BrojRadnikaView1* koji je prikazan na slijedećoj slici:



Slika 126. *BrojRadnikaView1* Data Control objekat koji se nalazi unutar *KSMModuleDataControl*

Za keriranje *BrojRadnikaView* objekta iskorišten je slijedeći upit:

```
SELECT      Odjeli.ODJEL_ID, Odjeli.NAZIV,
            COUNT(Radnici.RADNIK_ID)

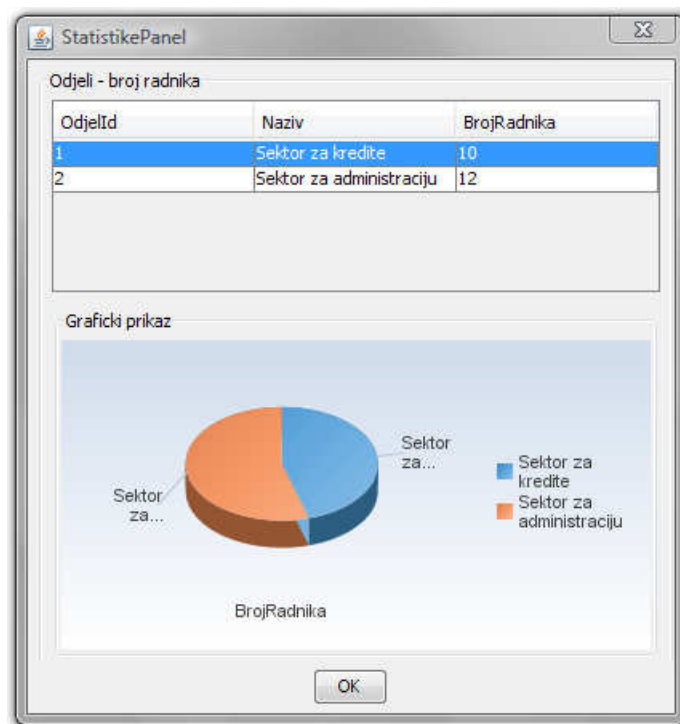
FROM        Odjeli, Radnici

WHERE       Radnici.ODJEL_ID = Odjeli.ODJEL_ID

GROUP BY   Odjeli.ODJEL_ID, Odjeli.NAZIV

ORDER BY   Odjeli.ODJEL_ID
```

Prevlaćenjem komponenti iz *BrojRadnikaView1* Data Control objekta, dobijamo slijedeći izgled *StatistikePanel* panela:

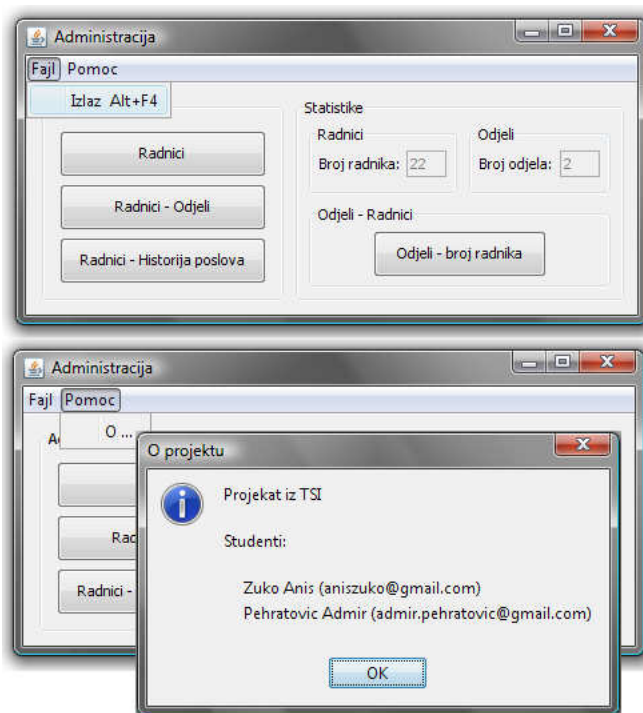
Slika 127. Izgled *StatistikePanel* panela

Nakon što kliknemo na *OK*, prikazat će nam se ponovo glavna forma *Administracija*. Za polja u kojima su prikazani broj radnika i broj odjela kreirana su dva pogleda i to:

- ✓ BrRadnikaView (`SELECT COUNT(Radnici.RADNIK_ID) FROM Radnici`) i
- ✓ BrOdjelaView (`SELECT COUNT(Odjeli.ODJEL_ID) FROM Odjeli`)

Menu Bar

Izgled Menu Bar-a je prikazan na slijedećoj slici:

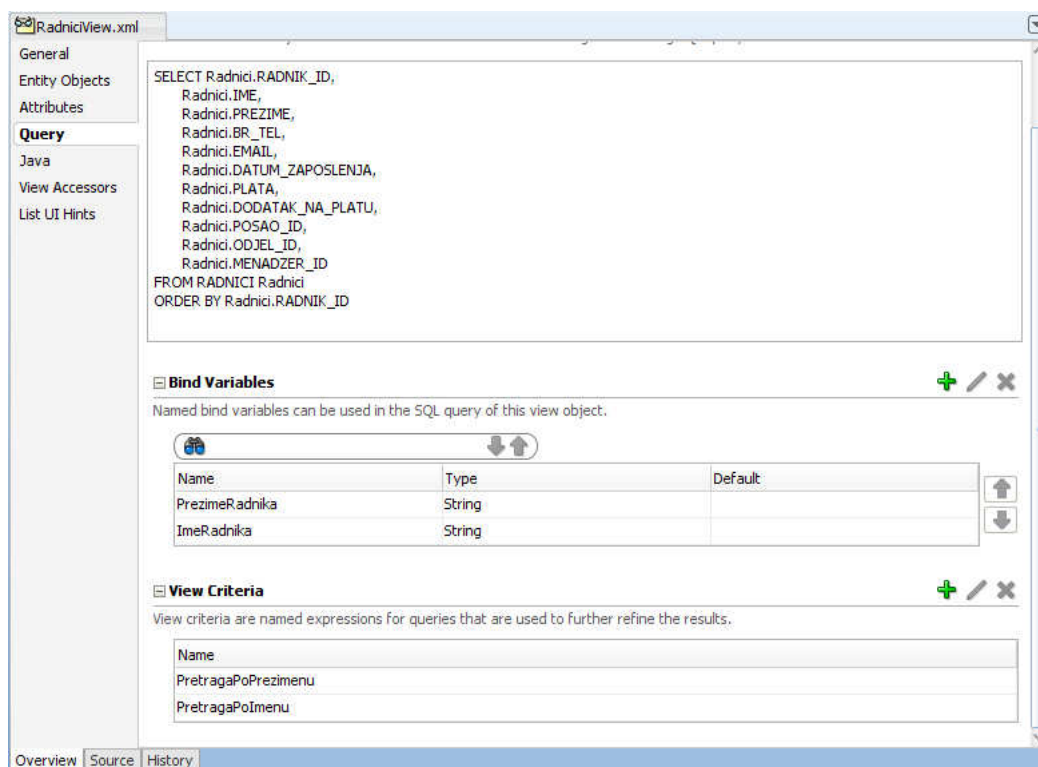


Slika 128. Menu Bar

IZVJEŠTAJI

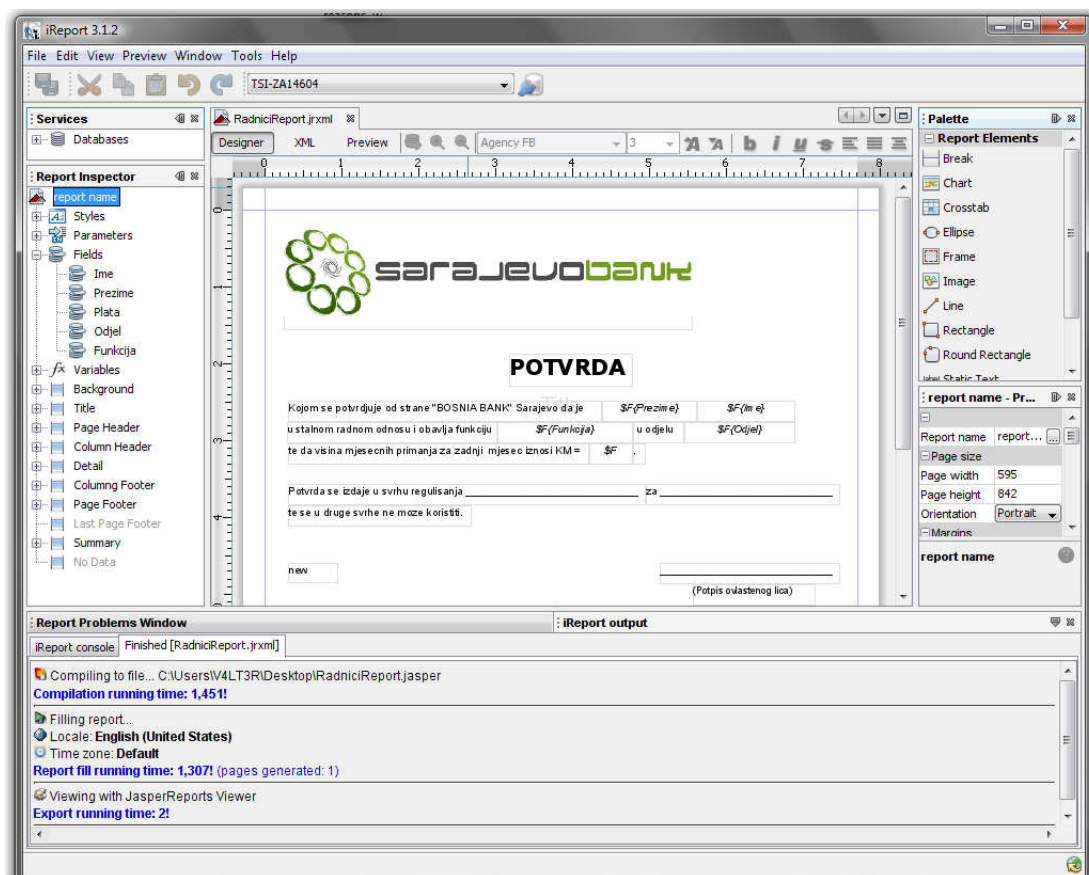
Za izradu ovog projektnog zadatka korištena je Oracle 10g baza podataka, a za izradu prezentacijskog sloja korištena je Oracle ADF tehnologija. Cijeli sistem za upravljanje podacima o jednoj poslovnoj kompaniji (u pogledu kadrovske službe) napravljen je u razvojnom okruženju Oracle JDeveloper.

Kao jedna od tačaka koje smo morali ispuniti u ovom projektnom zadatku bila je i generisanje izvještaja. Izvještaji ne postoje unutar samog ADF-a tako da smo za rješavanje ove tačke morali koristiti neka druga rješenja izvan ovog frameworka. Prije nego što pređemo na opis izvještaja potrebno je reći da je prije kreiranja izvještaja bilo potrebno na nivou RadniciView objekta kreirati dva kriterija pretrage i to: *PretragaPoPrezimeni* i *PretragaPoImenu* kao što je prikazano na slici 129.



Slika 129. Upit iskorišten za kreiranje RadniciView objekta i unaprijed definisani kriteriji pretrage za isti

U Java svijetu programiranja postoji veliki broj open source projekata koji nam omogućavaju generisanje izvještaja. Jedan od tih open source projekata je i *JasperReports* baziran na xml strukturi izvještaja. Dakle, ukoliko želimo kreirati izvještaj sa ovim frameworkom neophodno je da kao njegov ulaz obezbjedimo xml datoteku koja će opisivati izgled izvještaja. Važno je napomenuti da se i sql upit, koji će se koristiti za dohvaćanje podataka iz baze koje je neophodno prikazati izvještajem, također nalazi unutar xml datoteke i to specificiran kao jedan od njenih elemenata. Pri kreiranju xml datoteke koje će biti korištene od strane JasperReports-a možemo se poslužiti alatom *iReport* koji nam služi za grafičko editovanje datoteke, te tako na jednostavan način prevlačenjem komponenti „slažemo“ izvještaj.



Slika 130. Kreiranje xml datoteke upotrebom alata iReport

Kada kreiramo datoteku možemo je spasiti kao xml datoteku ili kao kompajliranu jasper dizajn xml datoteku (jrxml). Ukoliko se odlučimo za prvu varijantu onda ćemo istu unutar Java koda morati učitati i na osnovu nje kreirati jrxml. U nastavku je u potpunosti naveden Java kod koji na osnovu xml datoteke kreira i prikazuje izvještaj upotrebom JasperReports-a.

```
//kreiranje jasper report-a
```

```
InputStream input = null;
try {
    input = new FileInputStream(new File(path));
} catch (FileNotFoundException e) {
    System.out.println("Greska pri ucitavanju XML datoteke");
}
```



```
JasperDesign design = null;

try {
    design = JRXmlLoader.load(input);
} catch (JRException e) {
    System.out.println("Greska pri učitavanju xml datoteke u dizajner");
}

JasperReport report = null;

try {
    report = JasperCompileManager.compileReport(design);
} catch (JRException e) {
    System.out.println("Greska pri kompajliranju xml datoteke");
}

Map parameters = new HashMap();
parameters.put("ReportTitle", "PDF JasperReport");

InitialContext initialContext = null;
try {
    initialContext = new InitialContext();
} catch (NamingException e) {
    System.out.println("Greska initialContext = new InitialContext()");
}

Connection kon = null;
try {
    DriverManager.registerDriver (new oracle.jdbc.OracleDriver());
    kon = DriverManager.getConnection("jdbc:oracle:thin:@konekcioniString",ime, sifra);
} catch (SQLException e) {
    System.out.println("Greska pri uspostavljanju konekcije");
}

JasperPrint print = null;
try {
    print = JasperFillManager.fillReport(report, parameters, kon);
} catch (JRException e) {
    System.out.println("Greska pri popunjavanju izvjestaja");
}

// zatvori konekciju
try {
    kon.close();
} catch (SQLException e) {
    System.out.println("Greska pri zatvaranju konekcije");
}
```

```
// prikazi izvjestaj
```

```
JasperViewer.viewReport(print);
```

Konkretno u našoj aplikaciji kreirali smo dvije vrste izvještaja i to:

- (1) Potvrda o iznosu primanja nekog radnika za zadnji mjesec.
- (2) Podaci o uposlenim jednog odjela kompanije.

Stoga smo na odgovarajućim mjestima u aplikaciji morali dodati *JButton* komponente koje će pozivati kreiranje ovih izvještaja. Prvu vrstu izvještaja smo stavili da se kreira na panelu *RadniciHistorijaPoslovaPanel* gdje smo dodali dugme *Potvrda o iznosu plate* koje za radnika kreira izvještaj prikazan na slici ispod:



Slika 131. Izvještaj o iznosu mjesečne plate radnika. Izvještaj se otvara u JasperViewer-u gdje imamo opcije za ispis izvještaja ili za pohranjivanje istog kao pdf datoteke

```
//-----
TRIGGER UBACI_U_OBAVEZE_APLIKANTA AFTER INSERT OR UPDATE ON ZAHTJEVI_ZA_KREDIT
FOR EACH ROW
WHEN (new.odobren = 'DA')
declare
    brojac number;
    datum date;
    brojac2 number;

begin
    insert into tbl_obaveze_aplikanta (id, id_zajtjeva, rok, uplaceno)
    values (1, :new.id_zajtjeva, sysdate, 'NE');
    brojac := 2;
    loop
        brojac2 := brojac-1;
        select add_months(sysdate, brojac2) into datum from dual;
        insert into tbl_obaveze_aplikanta (id, id_zajtjeva, rok, uplaceno)
        values (brojac, : new.id_zajtjeva, datum, 'NE');
        brojac := brojac + 1;
        exit when brojac > :new.period_otplate;
    end loop;
END;
//-----

i ovaj

//-----
TRIGGER UBACI_U_OBAVEZE_BANKE AFTER UPDATE OR INSERT ON ZAHTJEVI_ZA_KREDIT
FOR EACH ROW

begin

    if : new.odobren = 'DA' then
        insert into tbl_obaveze_banke (id, rok, datum_isplate)
        values (:new.id_zajtjeva, sysdate, null);
    end if;
end;
//-----
```

BAZE PODATAKA [PROJEKTNI ZADATAK: SISTEM ZA IZDAVANJE KREDITA U BANCIMA]

OPIS SISTEMA POMOĆU DIJAGRAMA

Konceptualni model

Razvoj softverskog proizvoda započinje sa zahtjevom za specifičnim tipom programa i sa specifikacijom o tome šta bi taj program trebao biti. Zahtjev se može shvatiti kao pitanje, a specifikacija kao odgovor ili na ovaj odnos možemo gledati kao da je zahtjev postavljeni problem, a specifikacija je predloženo rješenje. U literaturi se navode različite definicije ovih pojmova u odnosu na SE, međutim, treba naglasiti da se eksperti na polju razvoja računarskih igara drže definicije da je zahtjev potreba za određenom vrstom softvera, te da je specifikacija opis tog softvera.

Pored toga što govore šta bi softverski proizvodi trebali da rade, zahtjevi također mogu definisati i listu osobina koje bi taj proizvod trebao imati. Nekad su zahtjevi veoma detaljizirani, a često predstavljaju samo sažet dokument. Još jedan važan aspekt dokumenta zahtjeva je namjena softverskog proizvoda koja treba da nam odgovori na pitanja: „Zašto je softverski proizvod vrijedan razvoja?“ i „Zašto će ljudi željeti da imaju razvijeni softverski proizvod?“.

Konceptualni model predstavlja rječnik projekta, rječnik svih pojmova koji se koriste u projektu. Međutim, konceptualni model je bolji od samog rječnika, jer može grafički da prikaže odnos između ovih pojmova projekta. U praksi to je pojednostavljeni klasni dijagram sa linijama koje povezuju odgovarajuće klase (konceptualne objekte) kako bi prikazao odnos između njih. Tipovi veza na dijagramu konceptualnog modela su agregacija i generalizacija.

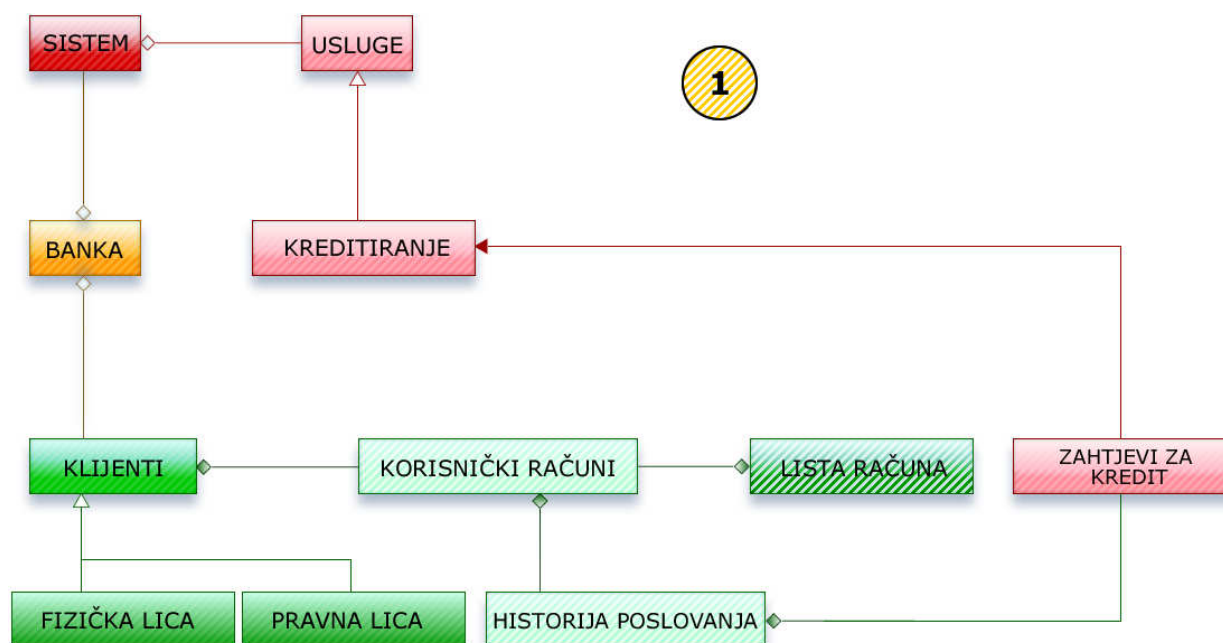
Upravo zahtjevi predstavljaju glavni izvor konceptualnih objekata i imenice iz tih zahtjeva predstavljaju naše konceptualne objekte. Sami zahtjevi sistema za izdavanje kredita u banci opisani su u zadaći broj 2 *Opis funkcionalnosti*, ali ćemo na ovome mjestu dati pregled istih kako bi mogli izdvojiti konceptualne objekte.

Tabela 1. Konceptualni objekti na osnovu zahtjeva sistema.

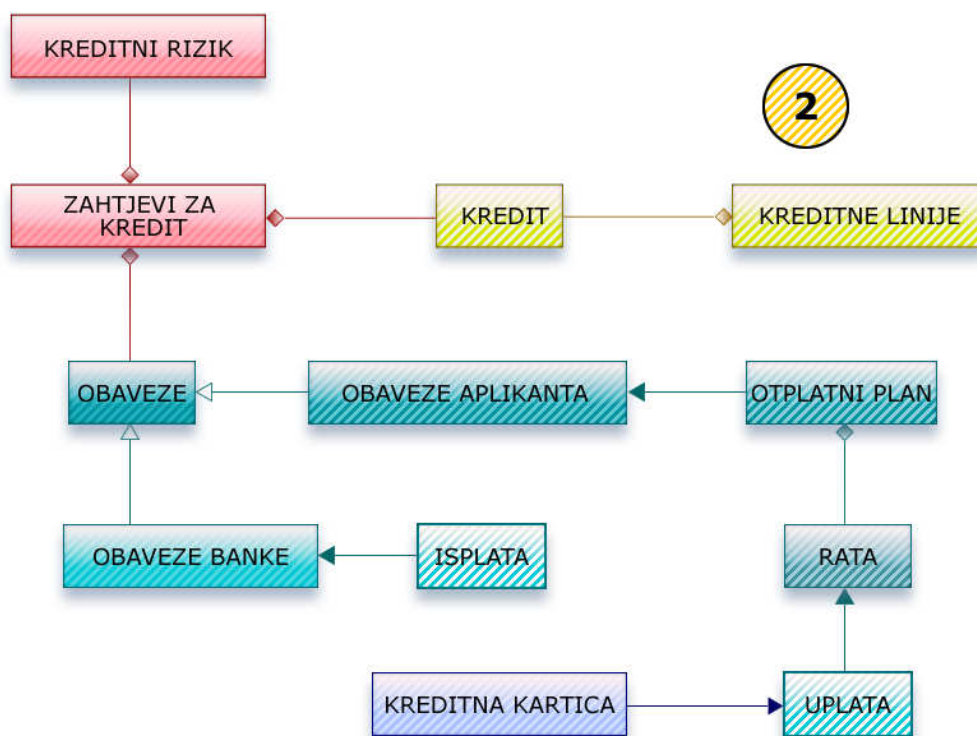
Zahtjev	Konceptualni objekat
Sistem će biti prvenstveno web baziran, ali će morati imati fleksibilnu arhitekturu koja će omogućiti i razvijanje alternativnih front-end rješenja (dektop aplikacija upotrebom SWING tehnologije itd)	Sistem
Sistem mora pružiti usluge kreditiranja sa mogućnošću podnošenja zahtjeva za kredit putem Interneta.	Usluga kreditiranja, zahtjev za kredit
Korisnicima usluga banke moraju biti dostupne sve informacije o vrstama kredita, kao i o načinu njihove otplate.	Vrste kredita, otplata, korisnici (fizička i pravna lica), banka
Za svaki odobreni zahtjev za kredit sistem mora kreirati obaveze. Postoje dvije vrste obaveza (one koje su određene upotrebom REA modela) i to: obaveze aplikanta i obaveze banke.	Obaveze: obaveze banke, obaveze aplikanta
Za svaki odobreni kredit sistem mora kreirati jedinstveni otplatni plan, koji je generalizacija obaveza aplikanta.	Otplatni plan

Online otplata kredita, tj. uplaćivanje rata kredita putem web aplikacija.	Rata
Prije nego što i počne otplata kredita logično je da banka mora ispuniti svoju obavezu. Ta obaveza banke ogleda se u isplati kredita.	Isplata
Korisnici koji otplaćuju kredit putem interneta moraju imati korisnički račun, ali isto tako i kreditnu karticu kojom će se vršiti otplata.	Korisnički račun, kreditna kartica.
Svaki korisnik mora biti u mogućnosti pregledati historiju poslovanja sa bankom, ali isto tako i uposlenik banke koji je zaposlen u monitoring timu ili kao kreditni službenik mora biti u mogućnosti pregledati tu istu historiju poslovanja kako bi mogao procijeniti kreditni rizik.	Historija poslovanja, kreditni rizik, uposlenik.
Da bi uposlenik mogao pristupiti sistemu i on mora imati svoj račun.	Računi uposlenika
Sistem mora obezbjediti i usluge za kadrovsku službu: informacije o uposlenicima, o odjelima u kojim oni rade, koji posao obavljaju, historiju poslova, lokaciji odjela ...	Odjeli, lokacije, posao, historije poslova

Dijagram konceptualnog modela radi lakšeg pregleda predstavljen je pomoću niza slika kako slijedi u nastavku ovog dokumenta. Moramo napomenuti da dijagram sadrži i neke koncepte koji se ne nalaze u tabeli 1.



Slika 132. (1) dio konceptualnog modela na kojem jasno vidimo da banka ima sistem za izdavanje kredita. Taj sistem pruža(ima) određene usluge. Na slici (1) prikazana je samo usluga kreditiranja koja sa konceptom „usluge“ ima odnos generalizacije. Banka također ima i svoje klijente i to pravna i fizička lica. Svaki od klijenata može a i ne mora da ima korisnički račun koji je pohranjen u listi računa. Za svaki korisnički račun veže se i historija poslovanja koja uključuje(ima) zahtjev za kredit. Zahtjev za kredit je koncept koji realizuje uslugu kreditiranja.



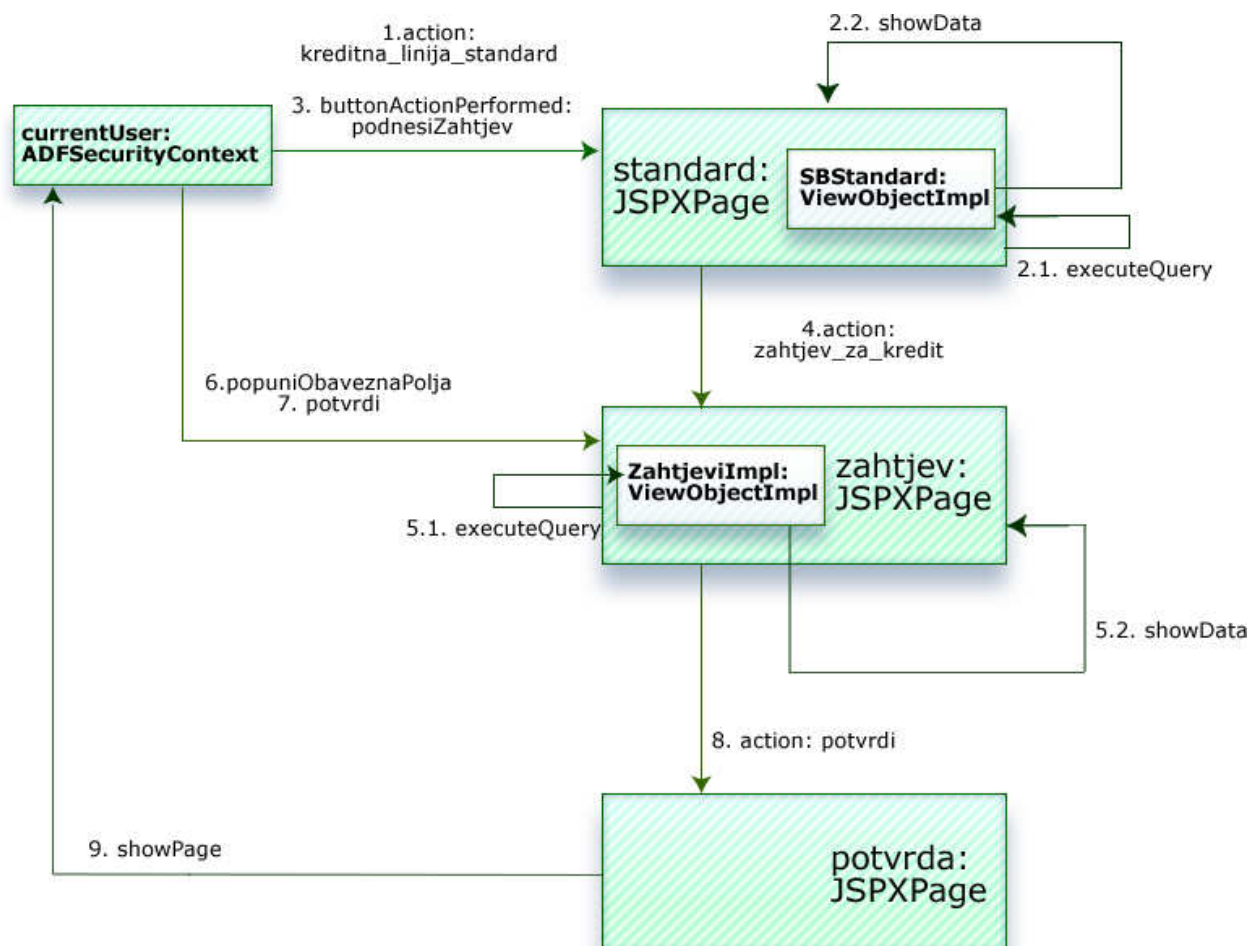
Slika 133. (2) dio konceptualnog modelana kojem je prikazan odnos kocepta „Zahtjevi za kredit“ za dijela (1) konceptualnog modela. Možemo uočiti da zahtjev za kredit ima određeni kreditni rizik koji predstavlja vjerovatnoću da će isti biti i odobren. Svaki zahtjev odnosi se na specifični kredit (koncept koji sadrži informacije o visini kredita, vremenskom periodu otplate itd). Svi krediti predstavljaju dio koncepta „Kreditne linije“. Ukoliko se kredit odobri sistem mora kreirati „Obaveze“ gdje imamo dva tipa obaveza „Obaveze banke“ koje realizuje „Isplata“, te „Obaveze aplikanta“ koje se prikazuju u vidu „Otplatnog plana“. Otplatni plan ima mjesečnu ratu koju je korisnik dužan uplatiti (uplata) upotrebom njegove kreditne kartice, ako je riječ o uplatu putem web aplikacije.



Slika 134. (3) dio konceptualnog modela koja nam prikazuje da zahtjev za kredit mora imati kreditni odbor koji će ga odobriti ili ne. Svaki član kreditnog odbora ima svoj račun uposlenika kojim će pristupati sistemu. Svaki uposlenik ima historiju poslova koja nam govori koje to sve poslove uposlenik obavljao u banci, te u kojem odjelu banke. Svaki odjel nalazi se na specifičnoj lokaciji koja pripada nekom od kantona Bosne i Hercegovine.

Dijagram saradnje

Svrha dijagrama saradnje je da prikaže razmjenu poruka između objekata radi postizanja određenih ciljeva. Na ovom mjestu dat je primjer komunikacije objekata pri izvođenju akcija za podnošenje zahtjeva za kredit.



Slika 135. Dijagram kolaboracije(saradnje): ovde je prikazan slučaj kada je korisnik već prijavljen sa svojim korisničkim računom i gdje on odabirom opcije na menu baru zahtjeva otvaranje stranice sa kreditnom linijom Standard. Dolazi do proslijeđivanja poruke (1) koja inicira prikazivanje stranice o kreditnoj liniji. Ta stranica prvo mora izvršiti upit kojim će dohvatiti podatke iz baze (poruka 2.1), te ih nakon toga i prikazati u nekoj od svojih komponenti (poruka 2.2). Nakon toga korisnik pritiskom na dugme „Podnesi zahtjev za kredit“ šalje poruku 3 stranici standard, nakon čega se proslijeđuje poruka 4 koja rezultuje otvaranjem stranice za podnošenje zahtjeva. Ova stranica mora prikazati formu za podnošenje zahtjeva, tako da prvo mora izvršiti upit (poruka 5.1) kako bi

popunila neka polja zahtjeva (id, korisnik, datum) te onda prikazati formu nakon slanja poruke 5.2. Nakon toga korisnik upisuje neophodne podatke (poruka 6 – ova poruka se zapravo može smatrati kao grupa poruka kojim se ostvaruje komunikacija sa inputText komponentama forme za podnošenje zahtjeva). Nakon toga korisnik potvrđuje zahtjev i stranici se šalje poruka 7. Akcija 7 otvara stranicu potvrda.aspx (rezultat rada poruke 8) te se ista stranica prikazuje korisniku (poruka 9).

Klasni dijagram

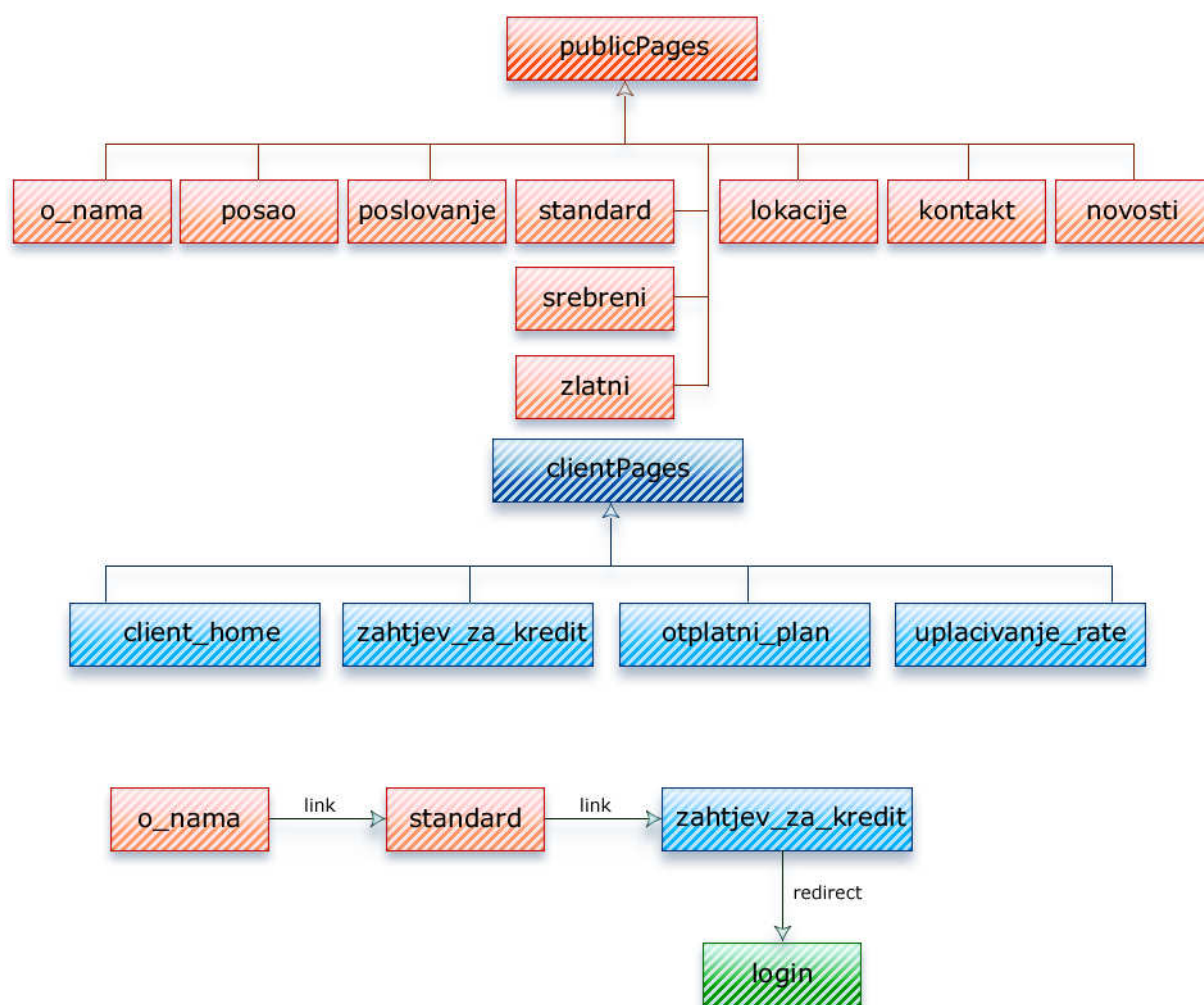
Kako smo sistem radili upotrebom ADF tehnologije to nije bilo potrebe za mapiranjem tabela iz baze u klase. U izradi sistema ipak smo morali napraviti neke klase koje su nam olakšavale rad i izradu sistema, tako da ćemo prikazati te iste klase klasnim dijagramom.



Slika 136. Klasni dijagram: *AppModuleImpl* klasa koja ima metode za dohvaćanje svih view objekata; *DohvatildKorisnikaRowImpl* služi za dohvaćanje id aplikanta na osnovu korisničkog naloga; *DohvatildKorisnikaImpl* služi za postavljanje bind varijable u where klauzuli upita ovog view objekta.

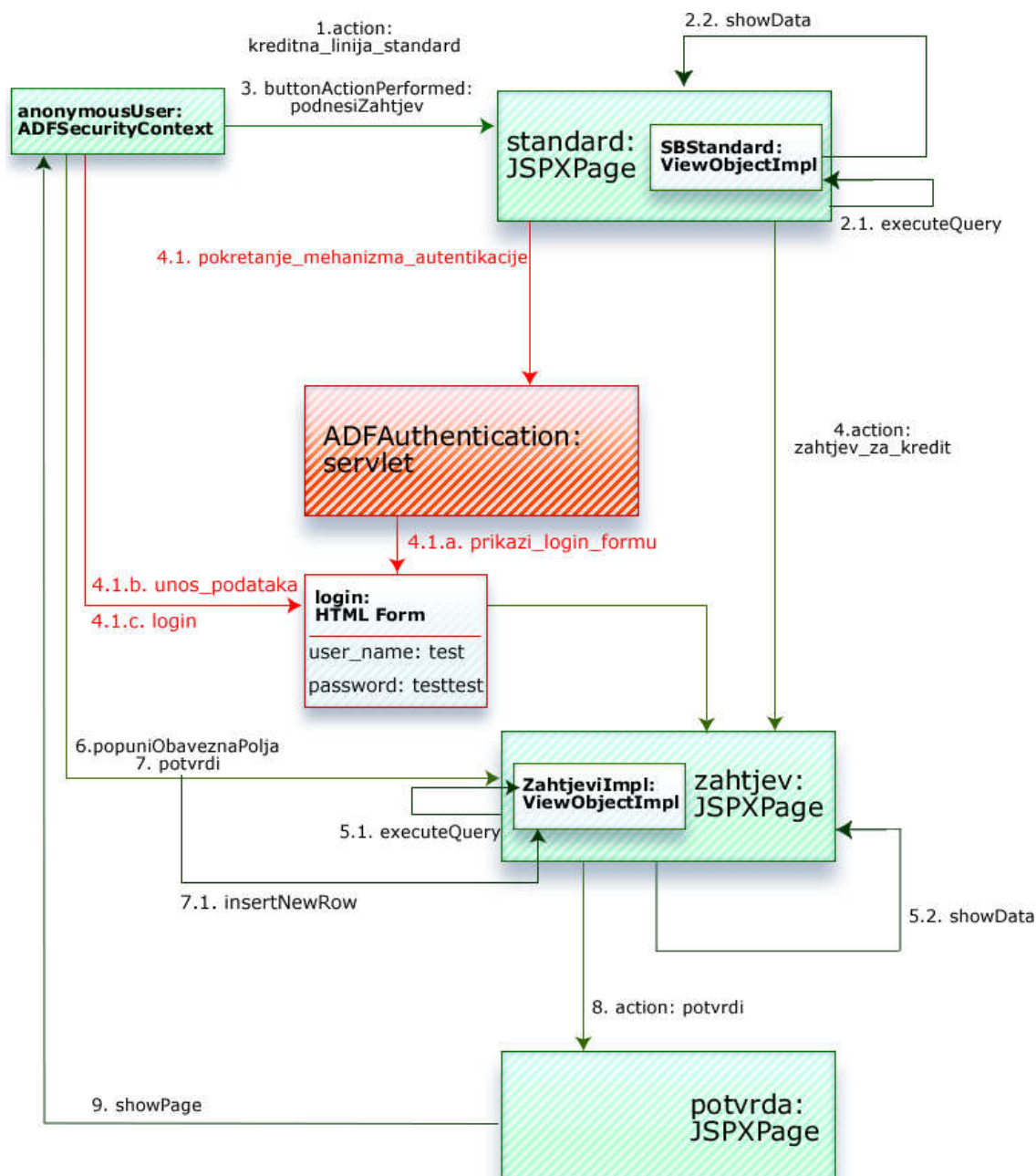
Dijagram objekata

Objektni dijagram je slika objekata u sistemu u jednom vremenskom trenutku. Na slici ispod predstavljen je dijagram objekata korisničkog okruženja.



Slika 137a. Kratak opis: Svaki objekat predstavlja web stranicu koja se prikazuje korisniku. Objekti su izvedeni iz dva različita šablona u zavisnosti od namjene stranice. Jedan uobičajan scenario podnošenja zahtjeva za kredit jeste sljedeći: korisnik sa glavne stranice (stranica „o_nama“) prelazi (link) na stranicu za pregled detalja o kreditnoj liniji SB Standard (stranica „standard“). Na toj stranici nalazi se odgovarajući link na stranicu za podnošenje zahtjeva za kredit. Ako korisnik nije logovan, link za podnošenje zahtjeva će ga automatski preusmjeriti na login stranicu (redirect).

Što se tiče onih objekata koji učestvuju u transakcijama nad bazom, na slici 135 su prikazani ključni objekti koji učestvuju u realizaciji najbitnije transakcije nad bazom, a to je podnošenje zahtjeva za kredit (makar što sa tačke gledišta poslovne logike). U nastavku ćemo priložiti dijagram koji detaljno opisuje šta se sve dešava sa objektima u ovom procesu.



Slika 137b. Kratak opis: na početku korisnik nije logovan tako da objekat `ADFSecurityContext` sadrži vrijednost „anonymous“ kao user name trenutnog korisnika. Anonimni korisnik poduzima akciju 1 i otvara stranicu kreditne linije SB Standard. U tom trenutku dolazi do dohvaćanja podataka iz baze podataka (poruka 2.1), jer stranica sadrži `read only` formu koja ispisuje podatke na osnovu rezultata rada `view` objekta. Vrijednosti tog objekta prikazane su na slici 6b1. Nakon uspješnog dohvaćanja podaci, tj objekat se prikazuje korisniku (poruka 2.2). Nakon toga korisnik želi podnijeti zahtjev za kredit (poruka 3), te tako dolazi do kreiranja poruke 4 i pokušaja da se otvori stranica za zahtjev. Ali, kako korisnik nije autentikovao (anonimni korisnik nema privilegije za pristup stranici) dolazi do pokretanja mehanizma autentikacije (poruka 4.1). Mehanizam autentikacije obavlja `ADFAuthentication Servlet` koji kreira login formu (4.1.a) . Zatim korisnik unosi neophodne podatke i potvrđuje iste klikom na dugme Login (4.1.b i 4.1.c). Nakon ovih koraka, kao što je prikazano na slici, login forma koja se proslijeđuje servletu kao objekat ima podatke `userName:test` i `password:testtest`. Nakon uspješno obavljene autentikacije korisnik je **test**. Dolazi do automatskog preusmjeravanja na stranicu za podnošenje zahtjeva. Ova stranica mora prikazati formu za podnošenje zahtjeva, tako da prvo mora izvršiti upit (poruka 5.1) kako bi popunila neka polja zahtjeva (id, korisnik, datum) te onda prikazati formu nakon slanja poruke 5.2. Nakon toga korisnik upisuje neophodne podatke (6) i potvrđuje podnošenje zahtjeva (7). Klikom na dugme za potvrdu dolazi do akcije koja se događa nad objektom `ZahtjevImpl` i ona pokušava dodati novi slog u tabelu u bazi podataka koju ovaj `view` objekat i mapira (7.1). Nakon što se insertuje novi slog u tabelu, korisnik se preusmjerava na stranicu `potvrda.jspx` na kojoj dobiva poruku o uspješno podnešenom zahtjevu za kredit.

SB Standard		TipId
Ko može aplicirati ?	Fizicka lica	
Opis kreditne linije	SB Standard kredit je namijenjen za finansiranje tekućih obaveza klijenata i poboljšanja likvidnosti.	
Minimalna vrijednost kredita	1000	
Maksimalna vrijednost	30000	
Kamatna stopa	13,99	
Obaveznost ucesca ziranata	da	
Minimalni period otplate	12	
Maksimalni period otplate	60	

Podnesi zahtjev za kredit


Slika 137.b.1. Stranica *standard.jspx* i prikaz objekta *SBStandard*.



Korisnicko ime:

Lozinka:

Slika 137.b.2. Preusmjeravanje na login stranicu i unošenje podataka u login formu.



Zahtjev za kredit

Aplikant:


Tip kredita:

* Vrijednost kredita:

* Prirod otplate:

Iznos mjesečne rate:

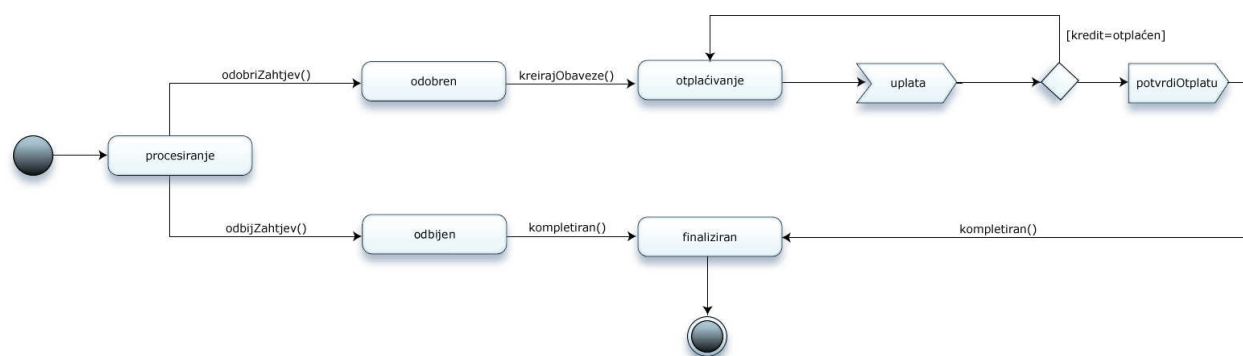
Odobren:

Datum podnošenja zahtjeva: 

Slika 137.b.3. Stranica *zahtjev.jspx* i prikaz objekta *ZahtjeviZaKredit*. Zatamljena polja su rezultat rada metoda koje izvršavaju određene upite kako bi na primjer za korisnika *test* dohvatili *aplikantId* polje ili tip kredita. I na kraju vidimo da je korisnik unio podatke o iznosu kredita kao i o vremenskom periodu otplate.

Dijagram stanja

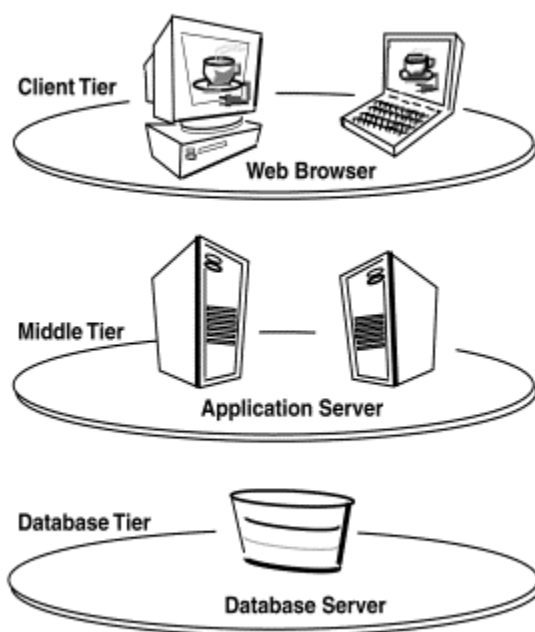
Dijagrami stanja modeliraju ciklus života objekta, ili stanja kroz koja prolazi tokom svog života. Dijagrami stanja su korisni za modeliranje objekata koji se ponašaju različito zavisno od svog stanja. Na slici 138 je prikazan životni ciklus `ZahtjevZaKredit` objekta kako prolazi iz stanja procesiranja u stanje odobren ili odbijen. Ukoliko je zahtjev odbijen prelazi se u stanje finaliziran, a ukoliko je zahtjev odobren isti prelazi u stanje otplaćivanja. Vidimo da se nakon svakog signala „uplata“ provjerava da li je kredit otplaćen, ukoliko nije opet se ostaje u stanju otplaćivanja i ponovo se vrši uplata (šalje se signal) sve dok se kredit ne otplati kada potvrđujemo otplaćivanje kredita i onda se prelazi u stanje finaliziran.



Slika 138. Dijagram stanja objekta `ZahtjevZaKredit`.

Aplikativna arhitektura

Sistem za izdavanje kredita u banci će se realizovati kao tronivoska arhitektura. Za razliku od dvonivoske arhitekture organizacije i implementacije poslovnih aplikacija, i sistema uopće, tronivoska arhitektura kao treći subjekt u jednoj takvoj organizaciji uvodi *Application Server*, kao što je prikazano na narednoj slici.



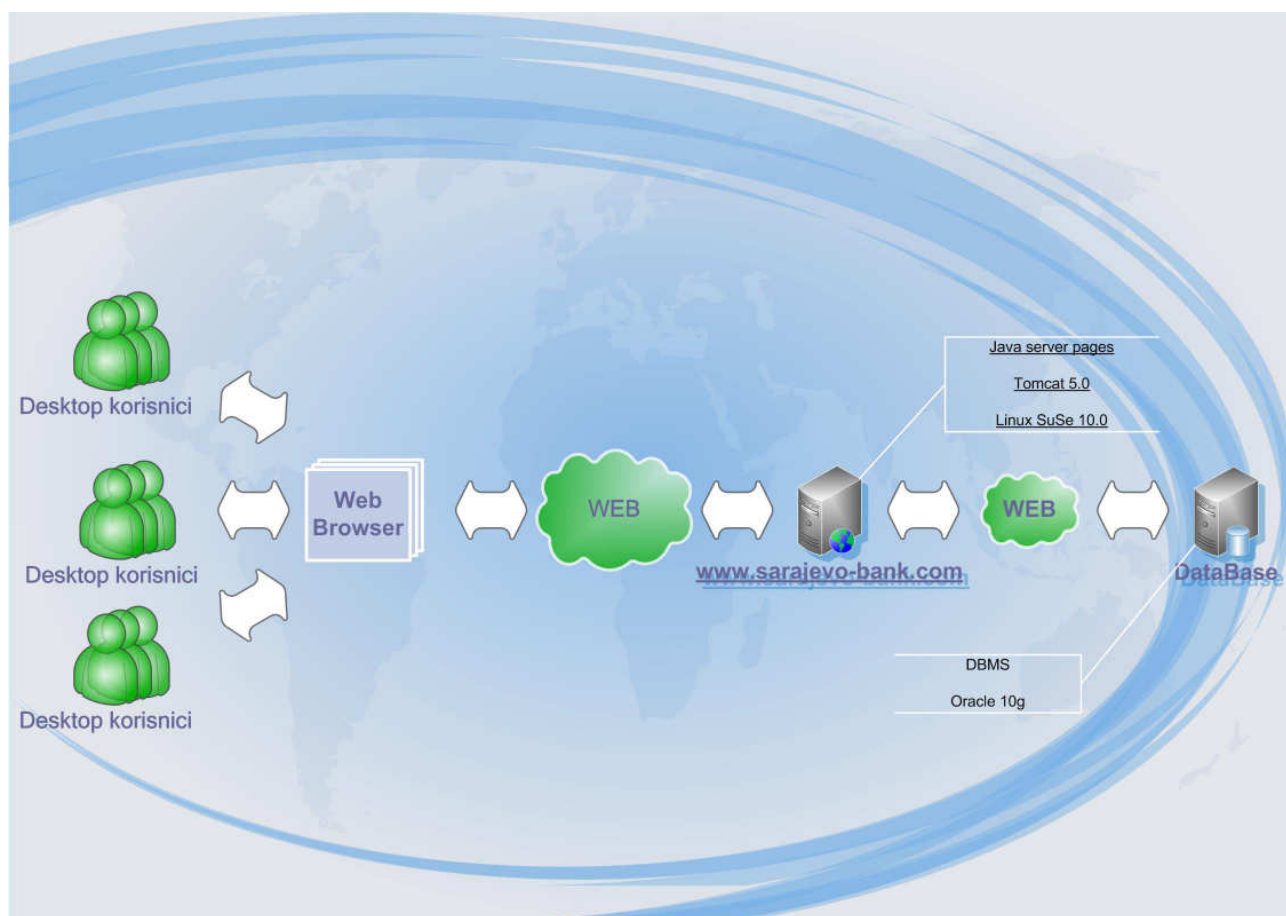
Slika 139. Tronivoska arhitektura

Dakle, ovu arhitekturu sačinjavaju slijedeći nivoi:

1. **Client computer** – pristup podacima i informacijama vezanih za poslovnu organizaciju postaje dostupan i na desktop računarima postojećih korisnika usluga jedne poslovne organizacije, kao i svih onih koji to nastoje postati. Dakle, javila se potreba za web prezentacijom proizvodnog programa proizvoda i usluga u velikim razmjerama. Ovo podrazumijeva komunikaciju korisnika (client) sa *application server-om* putem dijela aplikativnog programa. Prednost ovakve organizacije jeste što se sistem organizovan na ovaj način može izvršavati paralelno na više računara.
2. **Application server** – izrada poslovne logike sada se bazira na implementaciji i upotrebi web aplikacija, koje dolaze kao logičan slijed web prezentacija. Svrha web aplikacija je osiguravanje i pružanje podrške poslovnim partnerima i korisnicima, ali ne samo u tome, već i u olakšavanju

samog procesa poslovanja jedne poslovne organizacije. Application server se sastoji od usluga (servisa), a to su: aplikacijski program, kod za konektovanje na bazu podataka i *data caching* koji obezbjeđuje offline rad sa podacima koji se najčešće koriste.

3. **Server baze podataka** – kao i kod ostale dvije arhitekture (mainframe i client/server) baza podataka ima istu ulogu. Jedina razlika je u tome što komunikacija sa korisnikom ne ide direktno preko baze, već se sada komunikacija klijent-baza usmjerava preko *application servera* koji obezbjeđuje poslovnu logiku.



Slika 140. Arhitektura sarajevo-bank.

Aplikativnu arhitekturu prema postavci zadatke trebalo je opisati dijagramom komponenti. Komponente koristimo kako bismo organizovali sistem u upravljive, ponovno upotrebljive i zamjenljive dijelove softvera.



Slika 141. Kratak opis: Dijagram komponenti prati dijagram paketa, s tim da komponente sistema predstavljaju i web stranice aplikacije, kao i odgovarajući bibliotečni fajlovi korišteni u konstrukciji sistema. Svaka komponenta obuhvata širok skup datoteka (*.java, *.obj, *.exe, *.dll, *.jspx, *.html).