Aplikacija za upravljanje uplatama i isplatama novčanih sredstava

Kušević, Tomislav

Undergraduate thesis / Završni rad

2018

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:816195

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKA, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij

APLIKACIJA ZA UPRAVLJANJE UPLATAMA I ISPLATAMA NOVČANIH SREDSTAVA

Završni rad

Tomislav Kušević

Osijek, 2018.

Sadržaj

1. UVOD 1

1.1. Zadatak završnog rada 1

2. PREGLED KORIŠTENIH TEHNOLOGIJA 2

2.1. Java programski jezik 2

2.2. MySQL (MariaDB) 3

2.3. NetBeans kao programsko okruženje 3

2.4. XAMPP 4

3. DEFINIRANJE KORISNIČKIH ZAHTJEVA 6

3.1. Korisnički zahtjevi 6

3.2. Arhitektura sustava 11

3.3. Korisnici, računi korisnika, radnici, radna mjesta sustava 11

3.4. Baza podataka 13

4. RAZVOJ APLIKACIJE UPLATE I ISPLATE NOVČANIH SREDSTAVA 15

4.1. Razvoj korisničkog sučelja 15

4.2. Povezivanje sa bazom podataka 15

4.3. Validacija podataka 16

4.4. Izgled aplikacije 17

5. ZAKLJUČAK 22

SAŽETAK 24

TITLE 24

ABSTRACT 24

ŽIVOTOPIS 25

1. UVOD

Prije same izrade aplikacije potrebno je istražiti dosad već postojeće tehnike i stanje tržišta koje se bavi razvojem aplikacija temeljenih na uplatama i isplatama novčanih sredstava. Za razliku od razvojnih tvrtki, banke i ostale financijske institucije vrlo su konzervativne pri odlučivanju o programskom jeziku za razvoj aplikacija. No, brojni izvještaji potvrđuju da je Java u širokoj upotrebi kao programski jezik visokih performansi za razvoj širokog spektra financijskih aplikacija. Stoga je Java programski jezik vrlo dobar odabir za ovakav tip aplikacije. Nadalje, postoje brojne slične aplikacije koje se koriste trenutno u bankarskim sustavima, no one su vrlo vjerojatno tehnološki zastarjele, a razlog tome je što takvi sustavi ne mijenjaju aplikacije godinama zbog često konzervativnog stava. Svrha ovog završnog rada je razviti aplikaciju koja će obradu podataka u sustavima kao što je banka učiniti bržom i efikasnijom. Izradom računalne *desktop* aplikacije potrebno je pojednostaviti korištenje i snalaženje naspram dosad postojećih aplikacija. Aplikacija ujedno čini kompletno rješenje od same isplate i uplate novčanih sredstava, mogućnost uvida svakog pojedinog korisnika u stanje njegova računa, sve isplate i uplate korisničkog računa, mogućnost izvoza pa i samog ispisa pomoću Excel tablice. Korisnik ove aplikacije, odnosno, radnik u takvom sustavu poput banke ili pošte, može kontrolirati radnike cijelog poslovnog sustava i imati uvid u raspored i količinu radnika u određenom sektoru sustava te uvid u međusobnu nadređenost među radnicima. Radnik takvog sustava ima pristup klijentima, njegovim računima, uplatama i isplatama te unosu i uređivanju istih.

1.1. Zadatak završnog rada

U sklopu završnog rada potrebno je napraviti aplikaciju koja će omogućiti unos i promjenu korisnika te uplatu i isplatu novčanih sredstva unesenim korisnicima. Novčane transakcije potrebno je zapisati u bazu podataka. Java i MySql su tehnologije koje su potrebne za korištenje.

2. PREGLED KORIŠTENIH TEHNOLOGIJA

2.1. Java programski jezik

Java programski jezik objektno je orijentirani programski jezik koji ima sintaksu sličnu programskim jezicima C i C++. Java se smatra višeplatformskim (engl. *Cross Platform*) programskim jezikom što bi značilo da je njen *bytekôd* moguće pokretati na svim operacijskim sustavima. To je moguće pomoću Java virtualnog stroja (engl. *Java Virtual Machine*) koji pokreće, koji se tada generira prevođenjem Java programskog kôda (Programski kôd 2.1.). Vrlo je čitljiv i jasan što je dovelo do stanja da je jedan od najpopularnijih jezika. Objektno je orijentiran jezik što daje više mogućnosti kod pisanja programskog kôda. Strogo je tipski jezik što znači da jasno definira tip svake varijable ili objekta. Koristi *garbage collection*, odnosno čisti memoriju ukoliko određeni dio nije potreban i više se ne koristi što kod nekih *bytekôd* drugih programskih jezika nije uvijek slučaj [1][2][3].

Neki od tipova podataka koji se koriste pri razvoju su *boolean*, *integer*, *float*, *double*. Poznat je po tome što koristi klase (engl. *class*) i sučelja (engl. *interface*).

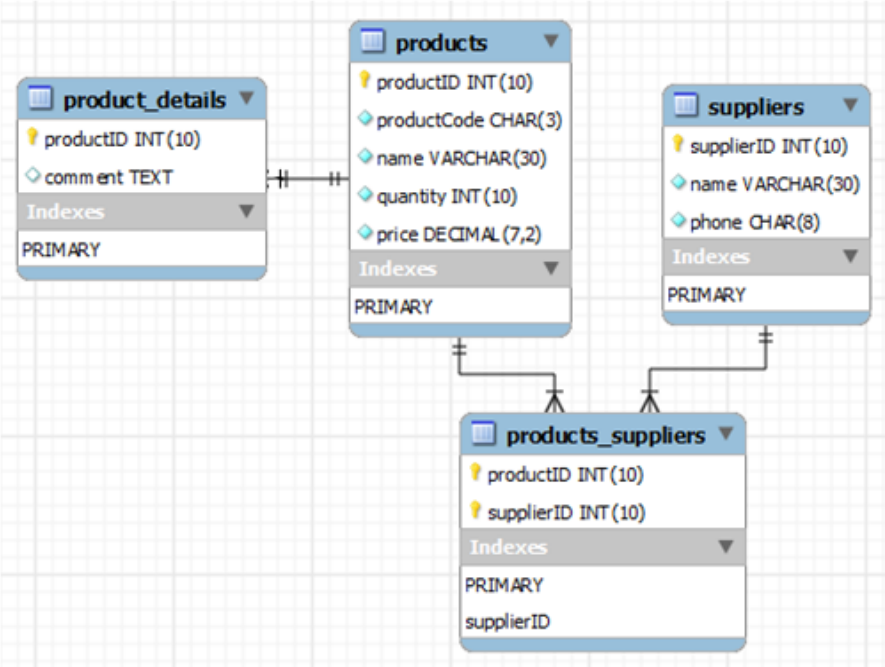
```
public class Customer {
    private String name;
    private ArrayList<Double> transactions;

    public Customer (String name, double initialAmount){
        this.name = name;
        this.transactions = new ArrayList<Double>();
        addTransaction(initialAmount);
    }

    public void addTransaction (double amount){
        this.transactions.add(amount);
    }
    public String getName() {
        return name;
    }
    public ArrayList<Double> getTransactions() {
        return transactions;
    }
}
```

Programski kôd 2.1. Primjer Java kôda

2.2. MySQL (MariaDB)

MySQL je svima besplatno dostupan (engl. *open source*) servis koji nudi mogućnost vrlo jednostavnog i efikasnog upravljanja bazom podataka. Budući da je jednostavan i raširen vrlo je lagano i bez puno predznanja doći do željene vlastite baze podataka. Spada pod relacijske baze budući da koristi relacije između tablica. Svaka tablica se sastoji od imena i entiteta poput primarnog ključa te ostalih entiteta koji definiraju tablicu [7].

Podaci unazad nekoliko godina pokazuju da postoji više od 10 milijuna instaliranih MySQL baza. Također postoji i grafičko sučelje koje olakšava korisnicima korištenje i manipuliranje podacima u bazi. Primjer relacije tablice prikazan je na slici 2.1.

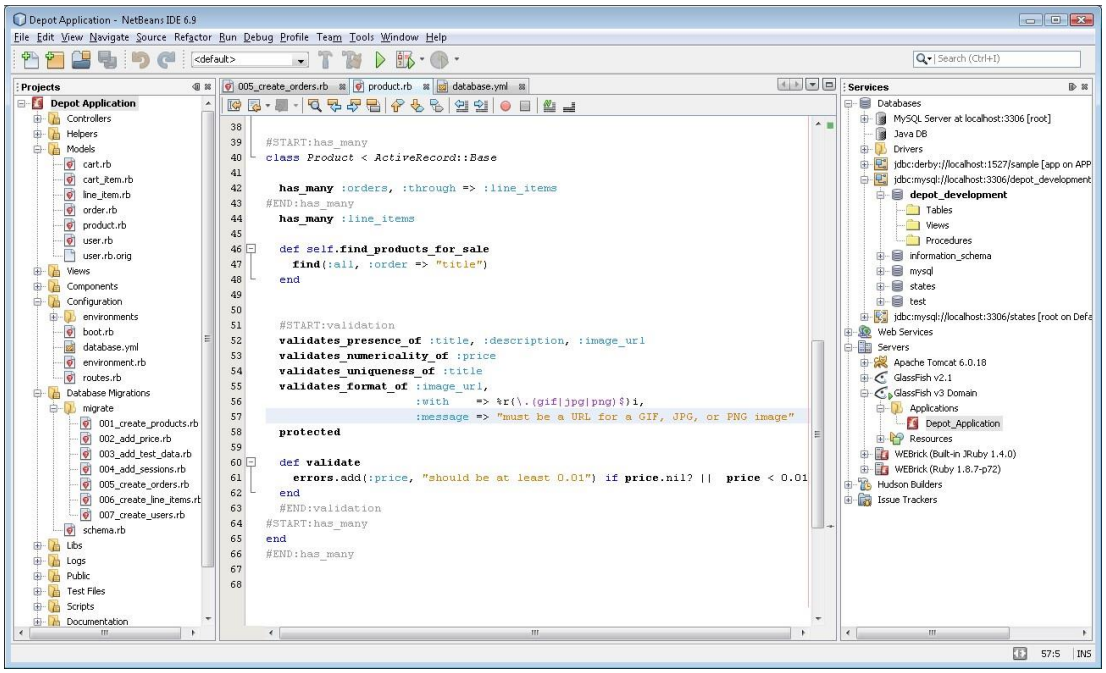


Slika 2.1. Primjer MySQL tablica [8]

2.3. NetBeans kao programsko okruženje

NetBeans je programsko okruženje (slika 2.2.) koje je i preporučeno kao primarno i najbolje okruženje Jave 8. Svojim alatima omogućuje lakšu implementaciju korisničkog sučelja (engl. *User Interface, UI*), a osim toga lagano povezuje kôd sa korisničkim sučeljem. Pogodan je za izradu desktop aplikacija, web aplikacija te mobilnih aplikacija. Potpuno je

besplatan i vrlo jednostavan pa je samim time često korišten alat za izradu aplikacija i programskih rješenja [5].

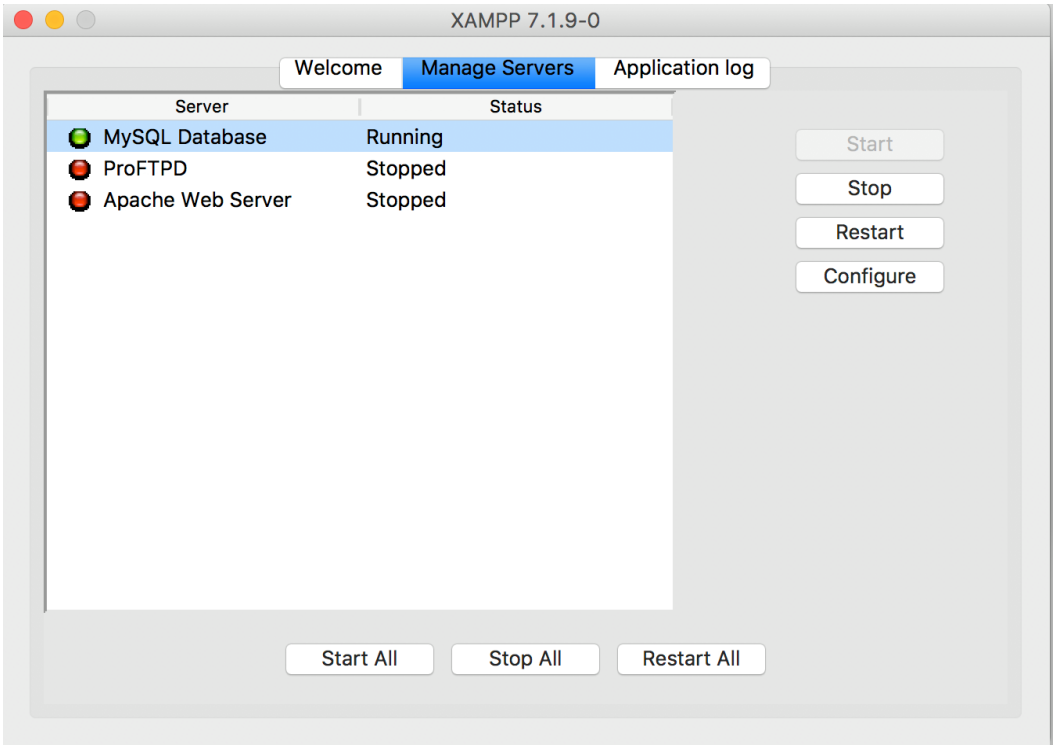


Slika 2.2. Izgled NetBeans korisničkog sučelja [10]

2.4. XAMPP

XAMPP (Slika 2.3.) je alat otvorenog kôda koji nudi rješenja web servisa. U imenu X označava višeplatformski alat, A označava Apache, M označava MariaDB, P označava Pearl i PHP. XAMPP je vrlo jednostavan alat koji pruža jednostavnu implementaciju lokanog web poslužitelja za razvoj i/ili testiranje **Error! Reference source not found.**.

XAMPP se redovito ažurira kako se ažuriraju i navedene inačice. Osim pune verzije, moguće je instalirati i manju verziju koja se sastoji od manje alata. Službeno, XAMPP-ovi inženjeri namjeravali su ga koristiti samo kao razvojni alat, kako bi ga programeri, web dizajneri, QA (engl. *Quality Assurance*) inženjeri mogli testirati svoj program na svojim računalima bez ikakvog pristupa internetu.



Slika 2.3. Prikaz XAMPP korisničkog sučelja

3. DEFINIRANJE KORISNIČKIH ZAHTJEVA

3.1. Korisnički zahtjevi

Ovakav tip aplikacije mogao bi se primijeniti na mnogim poslovnim sustavima koje uključuju uplate i isplate novčanih sredstava. Posebno se cilja na bankarske sustave kojima je primarno poslovanje temeljeno na uplatama i isplatama te praćenju stanja svih transakcija između korisničkih računa. Korisniku ovakvog sustava potrebno je omogućiti da što jednostavnije i brže zadovolji zahtjeve njegovih klijenata. U današnje vrijeme mnogi bankarski sustavi su komplicirani, imaju mnoštvo tablica koje sadrže velik broj vrijednosti. Pri razvoju ovakvih aplikacija često se ne pridodaje pažnja na izgled korisničkog sučelja te korisničkog iskustva (engl. *User Experience, UX*). Kako bi se ovaj problem razriješio, tablica 3.1. donosi pregled osnovnih korisničkih zahtjeva.

Tablica 3.1. Osnovni pregled korisničkih zahtjeva

ID	Naziv
1	Pregled, unos, brisanje, promjena korisnika
2	Pregled računa
3	Pregled transakcija
4	Pregled, unos, brisanje, promjena radnika
5	Unos i organizacija radnih mjesta
6	Eksport tablice transakcija u Excel tablicu
7	Više opcija filtracije liste transakcija

Tablica 3.2. opisuje korisnički zahtjev za pregled, unos, brisanje i promjenu korisnika. Pokretanjem aplikacije ponuđena je opcija „Korisnik“ koja pruža pristup navedenim opcijama. Preduvjeta za ovakvu akciju nema, ali da bi se korisnika unijelo u bazu podataka, potrebno je ispuniti određene uvjete kao što je unos imena, prezimena i OIB-a klijenta, a broj računa se generira automatski. Prije promjene postojećeg klijenta, potrebnoga je prvo označiti u tablici, a potom mijenjati podatke.

Tablica 3.3. opisuje korisnički zahtjev za pregled klijentskih računa. Pri pregledu popisa računa moguće je brisanje klijenta, a promjena računa nije potrebna.

Tablica 3.2. Detaljniji opis korisničkog zahtjeva Pregled, unos, brisanje, promjena korisnika

ID korisničkog zahtjeva	1
Naziv korisničkog zahtjeva	Pregled, unos, promjena, brisanje korisnika
Opis korisničkog zahtjeva	Mogućnost pregleda, unosa, promjene, brisanja korisnika
Preduvjet	Za pregled, promjena, brisanje korisnika - postoji korisnik Za unos korisnika – nema
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb "Korisnici" 3. Otvara se novi prozor koji pruža sve navedene opcije iznad
Alternativni scenarij	Nema

Tablica 3.3. Detaljniji opis korisničkog zahtjeva „Pregled računa"

ID korisničkog zahtjeva	2
Naziv korisničkog zahtjeva	Pregled računa
Opis korisničkog zahtjeva	Mogućnost pregleda popisa računa
Preduvjet	Postoji korisnik sa otvorenim računom
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb "Računi" 3. Otvara se novi prozor koji pruža pregled popisa računa
Alternativni scenarij	Nema

Tablica 3.4. opisuje korisnički zahtjev za pregled transakcija. Omogućuje se korisniku uvid u sve transakcije koje sadrže informacije o korisniku i njegovom računu, iznosu transakcije, datumu te radniku koji je obavio transakciju.

Tablica 3.4. Detaljniji opis korisničkog zahtjeva „Pregled transakcija”

ID korisničkog zahtjeva	3
Naziv korisničkog zahtjeva	Pregled transakcija
Opis korisničkog zahtjeva	Mogućnost za pregled liste transakcija
Preduvjet	Transakcija postoji
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb “Transakcije” 3. Otvara se novi prozor koji pruža navedene mogućnosti
Alternativni scenarij	Nema

Tablica 3.5. opisuje korisnički zahtjev za pregled, unos, brisanje i promjenu radnika. Ukoliko postoji potreba za novim radnikom, vrlo ga je lako unijeti. Ukoliko se moraju promijeniti informacije o određenom radniku, to je moguće učiniti putem ovog korisničkog zahtjeva. Još jedna od bitnih stavki ovog korisničkog zahtjeva je i brisanje radnika u slučaju da radnik prekida radni odnos.

Tablica 3.6. opisuje korisnički zahtjev za unos i organizaciju radnih mjesta. Budući da se sustav sastoji od mnogo radnika, bitno ih je organizirati po radnim mjestima kako bi imali organiziran cijeli radni sustav. Da bi radnika povezali sa sustavom prvo se unosi radno mjesto, a zatim sam radnik.

Tablica 3.7. opisuje korisnički zahtjev za izvoz tablice transakcija u Excel tablicu. Osim što korisnik ima mogućnost filtriranja tablice transakcija, bitno je omogućiti jednostavan ispis istih.

Tablica 3.8. opisuje korisnički zahtjev za filtraciju liste transakcija. Korisnicima je bitno da se u cijelom popisu transakcija izdvoji dio koji njima treba. Neki od slučajeva su izdvajanje transakcije samo određenog računa, određenog klijenta ili određenog radnika koji je obavio transakciju.

Tablica 3.5 Opis korisničkog zahtjeva „Pregled, unos, brisanje, promjena radnika”

ID korisničkog zahtjeva	4
Naziv korisničkog zahtjeva	Pregled, unos, brisanje, promjena radnika
Opis korisničkog zahtjeva	Mogućnost za pregled, unos, brisanje, promjena radnika
Preduvjet	Za pregled, promjena, brisanje radnika - postoji radnik Za unos radnika– nema
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb “Radnik” 3. Otvara se novi prozor koji pruža navedene mogućnosti
Alternativni scenarij	Nema

Tablica 3.6. Detaljniji opis korisničkog zahtjeva „Unos i organizacija radnih mjesta”

ID korisničkog zahtjeva	5
Naziv korisničkog zahtjeva	Unos i organizacija radnih mjesta
Opis korisničkog zahtjeva	Mogućnost za unos i organizaciju radnih mjesta
Preduvjet	Za unos - nema Za organizaciju – postoje radnici i radno mjesto
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb “Radno mjesto” 3. Otvara se novi prozor koji pruža navedene mogućnosti
Alternativni scenarij	Nema

Tablica 3.7. Detaljniji opis korisničkog zahtjeva „Eksport tablice transakcija u Excel tablicu”

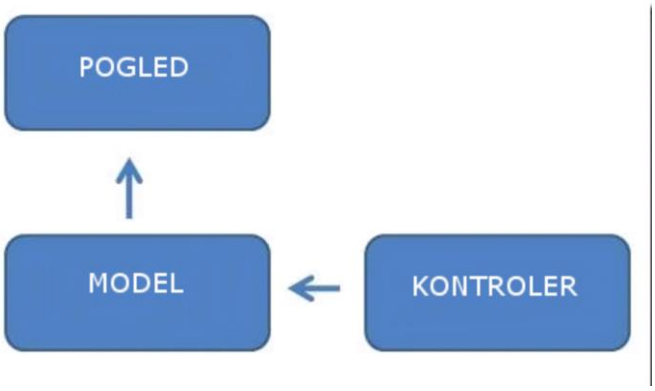
ID korisničkog zahtjeva	6
Naziv korisničkog zahtjeva	Eksport tablice transakcija u Excel tablicu
Opis korisničkog zahtjeva	Omogućuje eksport iz tablice u aplikaciji u Excel tablicu
Preduvjet	Prethodno izvršene transakcije
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb “Transakcije” 3. Pritisak na Excel ikonu 4. Odabir mjesta na računalu gdje se tablica treba spremiti
Alternativni scenarij	Nema

Tablica 3.8. Detaljniji opis korisničkog zahtjeva „Više opcija filtracije liste transakcija”

ID korisničkog zahtjeva	7
Naziv korisničkog zahtjeva	Više opcija filtracije liste transakcija
Opis korisničkog zahtjeva	Mogućnost za filtraciju liste transakcije na temelju više varijabli (račun, korisnik, djelatnik)
Preduvjet	Prethodno obavljene transakcije
Glavni scenarij	1. Otvori se glavni zaslon 2. Pritisak na gumb “Transakcije” 3. Odaberemo parametar prema kojem ćemo filtrirati listu (radnik, račun, djelatnik) 4. U predviđeno dio (*search)* upišemo vrijednosti koje želimo da postanu parametri filtracije
Alternativni scenarij	Nema

3.2. Arhitektura sustava

Kod arhitekture ovakvog sustava bitno je odvojiti koje ovlasti ima određeni korisnik. Korisnici sustava su ujedno i radnici, a ovlasti se dijele prema radnicima i administratorima. Radnici mogu pristupati informacijama te imaju kompletan uvid u sve novčane transakcije. Administrator ima pristup, mogućnosti promjene te unosa radnih mjesta i raspoređivanje radnika i njihovih nadređenih. Klijenti imaju mogućnost otvaranja računa, uplate i isplate na svoj račun ili račun drugog klijenta. Arhitektura samoga sustava, naziva se MVC (engl. *Model View Controller*), gdje M označava model, V označava korisničko sučelje (engl. *View*), a C označava upravljač (engl. *Controller*). Model zapravo predstavlja sloj aplikacije koji označava modele podataka i naglašava njihove promjene. Sloj korisničkog sučelja je zadužen za prikaz prethodno modeliranih podataka, a upravljač je zadužen za svu logiku aplikacije i povezuje korisničko sučelje s modelom. Slika 3.1. prikazuje shemu MVC arhitekture.



Slika 3.1. – Shema MVC arhitekture **Error! Reference source not found.**

3.3. Korisnici, računi korisnika, radnici, radna mjesta sustava

Glavni dijelovi sustava su stvaranje, promjena i brisanje klijentskih računa, informacije o klijentu ili radniku te radna mjesta. Moguće je odrediti i shematski prikazati hijerarhiju i odnose među radnicima u svakom trenutku te promijeniti iste. Nadalje, bitno je omogućiti uvid o datumu stvaranja računa, radniku koji je otvorio račun te količini novčanih sredstava prebačenih u određenim vremenskim periodima. Tijekom razvoja aplikacije, jedan od zahtjeva

koji predstavlja veći problem je odrediti u kakvom su odnosu navedeni dijelovi. Na početku je potrebno definirati koje su sve informacije nužne kako bi navedeni sustav mogao funkcionirati. Kada govorimo o računu, bitno je da svaki račun ima broj računa, datum otvaranja i informaciju o trenutnom stanju novčanih sredstava. Kada je u pitanju radnik, potrebne informacije su ime, prezime, OIB, odnos podređenog i nadređenog radnika te radno mjesto na kojem radi što prikazuje programski kôd 3.1.

```
private String ime,prezime, OIB;

@ManyToOne
private Radnik nadredeni;
@OneToMany(mappedBy = "nadredeni")
private List<Radnik> podredeni;
@OneToMany(mappedBy = "radnik")
private List<RadiNa> mjesta;
@OneToMany(mappedBy = "radnik")
private List<Transakcija> transakcije;
```

Programski kôd 3.1. Model radnika

Kada je u pitanju klijent, on se sastoji od imena, prezimena, OIB-a i broja računa prema programskom kôdu 3.2.

```
private String ime, prezime, oib, racun;
```

Programski kôd 3.2. Model klijenta

Model radnog mjesto korištenog u aplikaciji se sastoji od odjela, naziva i radnika koji rade na tom radnom mjestu, prema programskom kôdu 3.3.

```
private String odjel, naziv;
@OneToMany(mappedBy = "radnoMjesto")
private List<RadiNa> radnici;
```

Programski kod 3.3. Model radnog mjesta

Sve je to objedinjeno transakcijom (Programski kôd 3.4.) koja se sastoji od datuma, iznosa transakcije, računa na koji se odnosi, klijenta i radnika koji je obavio transakciju.

```
private Date datum;
private BigDecimal iznos;
@ManyToOne private Racun racun;
@ManyToOne private Korisnik korisnik;
@ManyToOne private Radnik radnik;
```

Programski kod 3.4. Model transakcije

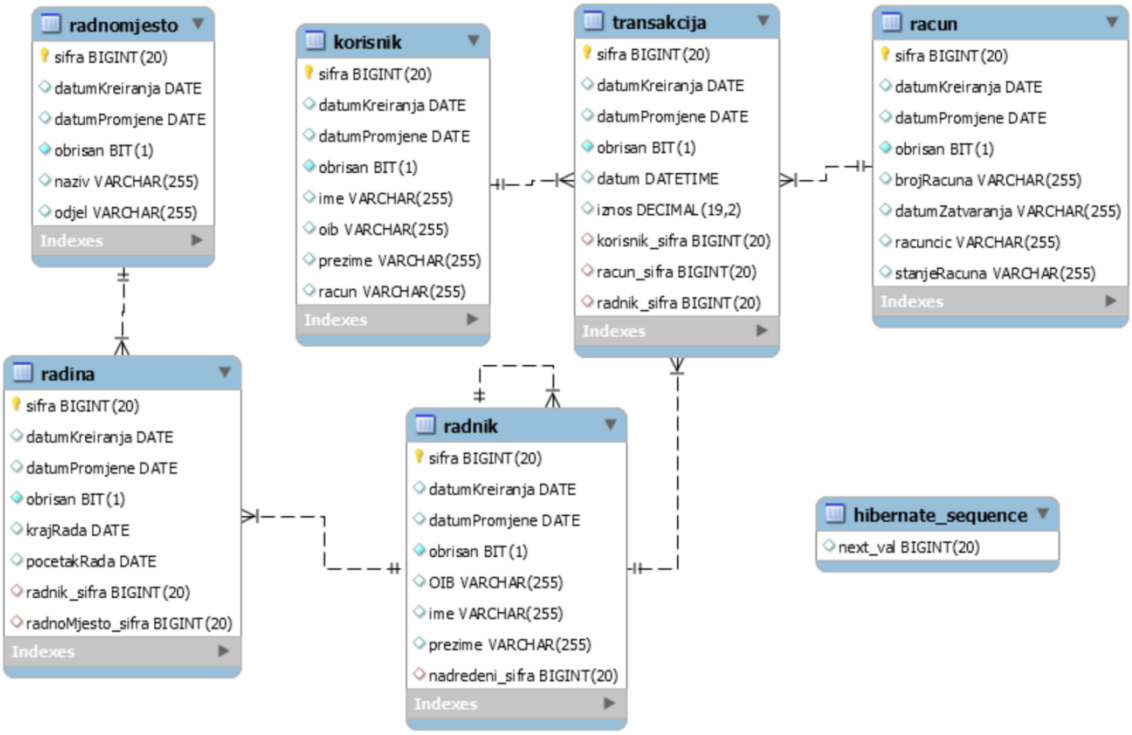
3.4. Baza podataka

Nakon utvrđivanja korisnika, svrhe te cilja izrade ovakve aplikacije definirana je arhitektura baze podataka. Tri glavna dijela ovakvog sustava, kao što je prethodno navedeno, čine korisnici, radnici i radna mjesta te je potrebno napraviti određene relacije među njima. Ovakvi sustavi najviše zahtijevaju dobru arhitekturu baze podataka jer je bitno da su informacije točne i da je vjerojatnost pogreške svedena na minimum. Slika 3.2. prikazuje relacije među tablicama.

Više naprema jedan relacije između tablica:

1. Tablica RadiNa sa tablicom Radnik i Korisnik
2. Tablica Transakcije sa tablicama Radnik, Račun, Korisnik
3. Tablica Radnik sa tablicom Radnik.

Svaki radnik može imati više nadređenih/podređenih radnika, takvom implementacijom može se postići hijerarhija radnika.



Slika 3.2. Relacije između tablica u bazi podataka

4. RAZVOJ APLIKACIJE UPLATE I ISPLATE NOVČANIH SREDSTAVA

4.1. Razvoj korisničkog sučelja

Prije implementacije sustava potrebno je definirati izgled korisničkog sučelja. U ovakvim sustavima nije potrebno mnogo dizajna, već se teži omogućiti korisnicima što preglednije i jednostavnije služenje sustavom. Bitno je odraditi raspored akcija koje otvaraju nova sučelja kako korisnik ne bi imao poteškoća s navigacijom u aplikaciji. U aplikaciji se koristi 6 sučelja prema tablici 4.1.

Tab. 4.1. Popis sučelja unutar aplikacije

Sučelje	Početno	Sučelje koje pruža glavnu navigaciju, raspored ostalih mogućih prozora
Sučelje	Korisnici	Sučelje koje pruža popis korisnika u poslovnom sustavu koji koristi aplikaciju
Sučelje	Radnici	Sučelje koje pruža mogućnost pregleda popisa radnika u sustavu koji koristi aplikaciju
Sučelje	Radna mjesta	Sučelje koje pruža mogućnost upravljanja radnih mjesta poslovnog sustava i radnika u njemu
Sučelje	Računi	Sučelje koje pruža popis računa otvorenih u poslovnom sustavu koji koristi aplikaciju
Sučelje	Transakcije	Sučelje koje pruža informacije o transakcijama, filtraciju transakcija i ispis

4.2.Povezivanje sa bazom podataka

Nakon izrade korisničkog sučelja, potrebna je implementacija kôda i povezivanje sa bazom podataka. Glavna svrha ovog sustava je spremanje podataka u bazu. Potrebno je dobro povezati kôd sa bazom podataka te svaku grešku prilikom spremanja podataka prikazati korisniku odgovarajućom porukom. Bitno je imati jednu instancu baze podataka, koju je moguće dobiti korištenjem tzv. *Singleton patterna*. *Singleton patternom* se pri kreiranju

zahtjeva dohvati samo jedna instanca baze podataka koja se koristi za vrijeme životnog vijeka aplikacije. Prilikom povezivanja s bazom podataka korišten je Hibernate ORM (engl, *Object/Relational Mapping*)[4] koji omogućuje lakše pretvaranje JSON kôda u podatke pogodne za korištenje pri razvoju. Hibernate ORM omogućuje programerima lakše pisanje aplikacija čiji su podaci promjenjivi unutar aplikacije te omogućuje trajno vjerodostojne podatke [1].

```
public class HibernateUtil {
    private static Session session = null;

    protected HibernateUtil() {
        // Exists only to defeat instantiation.
    }

    public static Session getSession() {
        if (session == null) {
            try {
                session = new
Configuration().configure().buildSessionFactory().openSession(
);
            } catch (Throwable ex) {
                JOptionPane.showConfirmDialog(null, "NEMA
KONEKCIJE!");
                throw new ExceptionInInitializerError(ex);
            }
        }
        return session;
    }
}
```

Programski kod 4.1. Konfiguracija Hibernate ORM-a

Budući da se u razvoju aplikacije koristila lokalna baza podataka, napisana je skripta koja dodaje početne vrijednosti u bazu podataka, kako bi podaci bili dostupni u bilo kojem trenutku razvoja aplikacije, a svrha takvih podataka je da tijekom faze razvoja budu što vjerodostojniji podacima koji će se koristiti kasnije u budućnosti u samoj aplikaciji.

4.3. Validacija podataka

Kod unosa podataka bitno je provjeriti jesu li podaci koje je potrebno unijeti točni kako bi zahtjev na bazu podataka za spremanje ili promjenu određenih podataka bio validan. Ukoliko takav zahtjev na bazu podataka nije validan, zahtjev neće proći te je iz tog razloga potrebno raditi provjere. Također je bitno utvrditi da polje za unos podataka korisnik nije ostavio prazno,

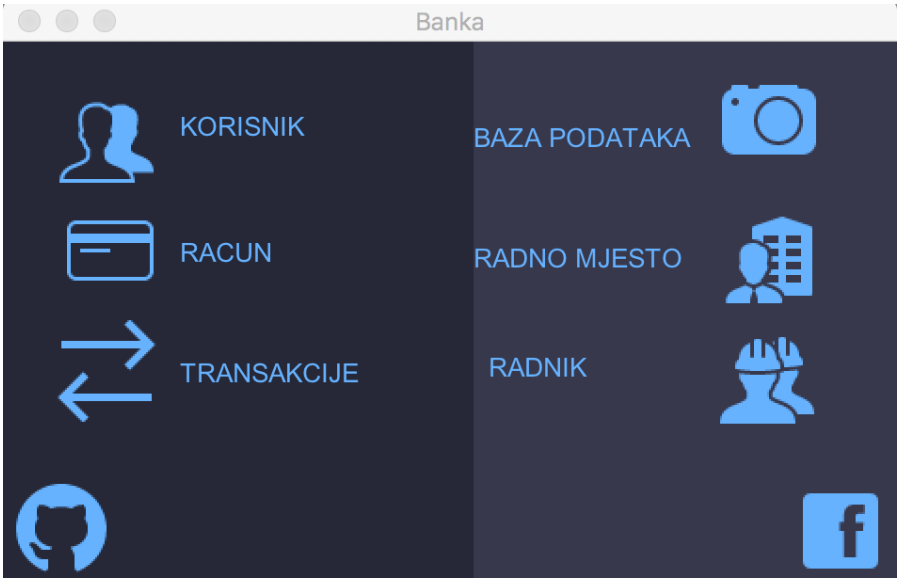
jer je unos potreban za nastavak korištenja sustava. Pri unosu OIB-a mora se sastojati od jedanaest brojeva te se uvodi provjera istog. Kasnije se OIB može i promijeniti, ali i u tom trenutku mora doći do provjere, sadrži li jedanaest brojeva. Iako se validacija odvija i na poslužiteljskoj strani, također je dobra praksa odraditi i validaciju direktno na sučelju tijekom unosa, kako bi se ostvarila što veća sigurnost koja je vrlo bitna na aplikacijama poput ove, iz razloga što uključuje brojne transakcije i radi neposredno s novčanim sredstvima.

4.4. Izgled aplikacije

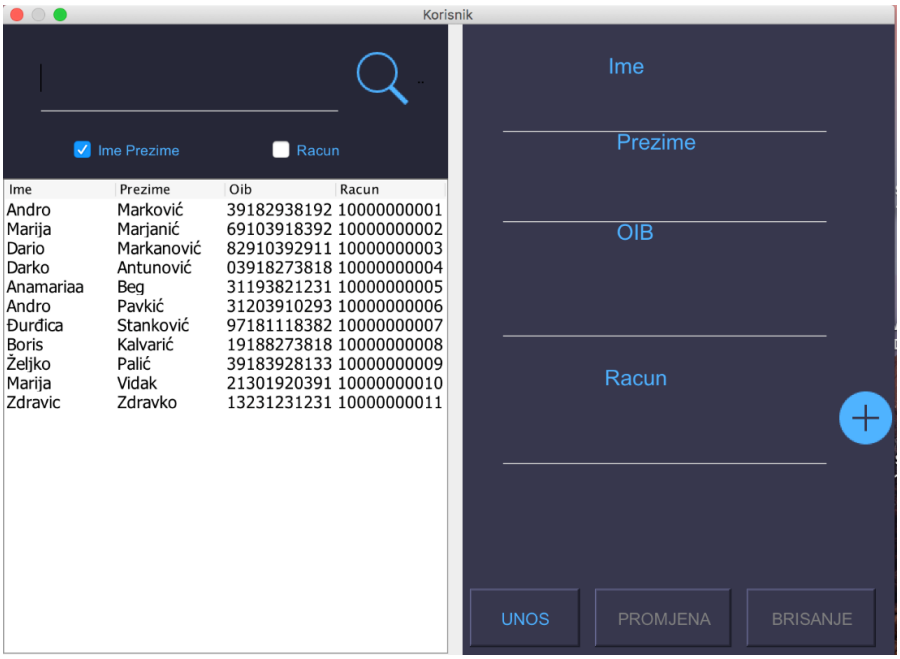
U nastavku ukratko je opisan izgled aplikacije. Na početku se nalazi glavno sučelje (Slika 4.1.) koje je, poželjno, što jednostavnije i što jasnije korisniku. Na njemu se nalazi 6 gumbova (engl. *button*), a pritiskom na gumb:

- Korisnik: otvara se sučelje koje pruža opciju unosa korisnika, promijene, pregleda te brisanja.
- Račun: otvara se sučelje koje nudi popis računa unutar sustava
- Transakcije: otvara se sučelje koje prikazuje popis, omogućuje filtraciju, izvoz i ispis transakcija
- Radnik: prikazuje se sučelje koje sadrži popis radnika i njihovu hijerarhiju unutar tvrtke (odnos nadređeni/podređeni)
- Radno mjesto: prikazuje sučelje koje omogućuje unos radnog mjesta, te raspoređivanje radnika po radnom mjestu
- Izlaz: gasi sučelja aplikacije

Nakon pritiska na gumb Korisnik otvara se novo sučelje (Slika 4.2.) koje se sastoji od polja za pretraživanje u koje se unosi tekst i pruža mogućnost pretraživanja klijenata. Također postoje dvije opcije pretraživanja, po imenu i po računu klijenta. U određenom trenutku samo jedna opcija može biti odabrana. Ispod polja za pretragu nalazi se lista klijenata. S desne strane, ovisno o odabranom klijentu na listi, prikazuju se njegove informacije. Osim toga, postoje i opcije brisanja, dodavanja i promjene klijenta.

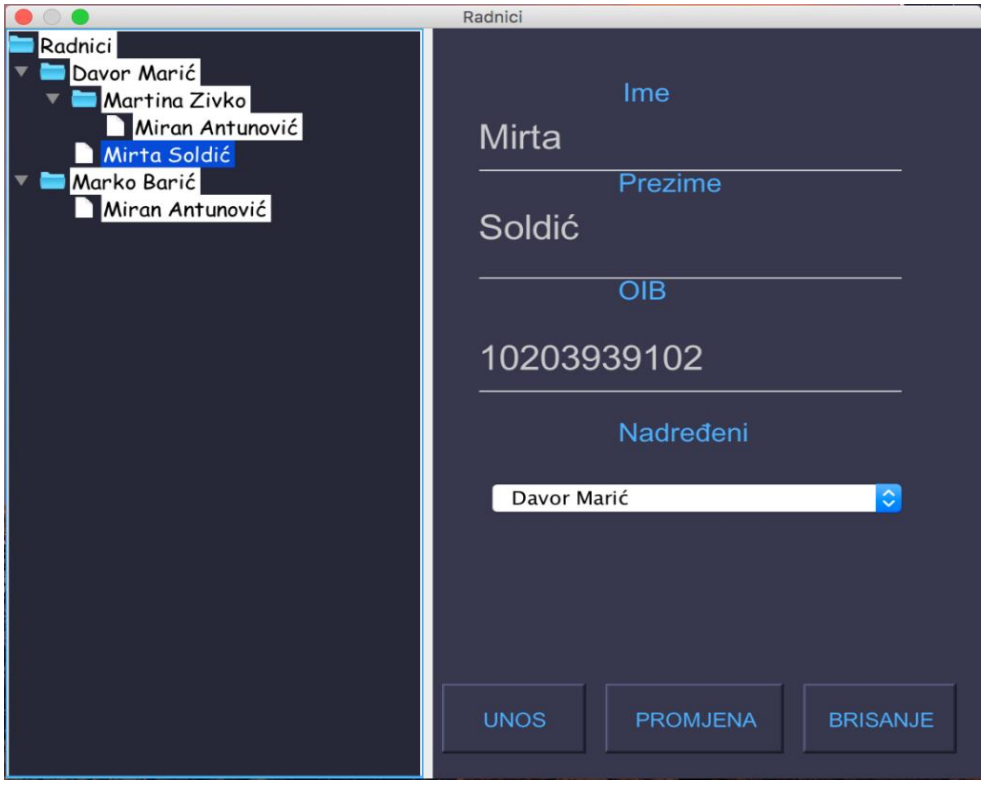


Slika 4.1. Početno korisničko sučelje



Slika 4.2. Korisničko sučelje *Korisnik*

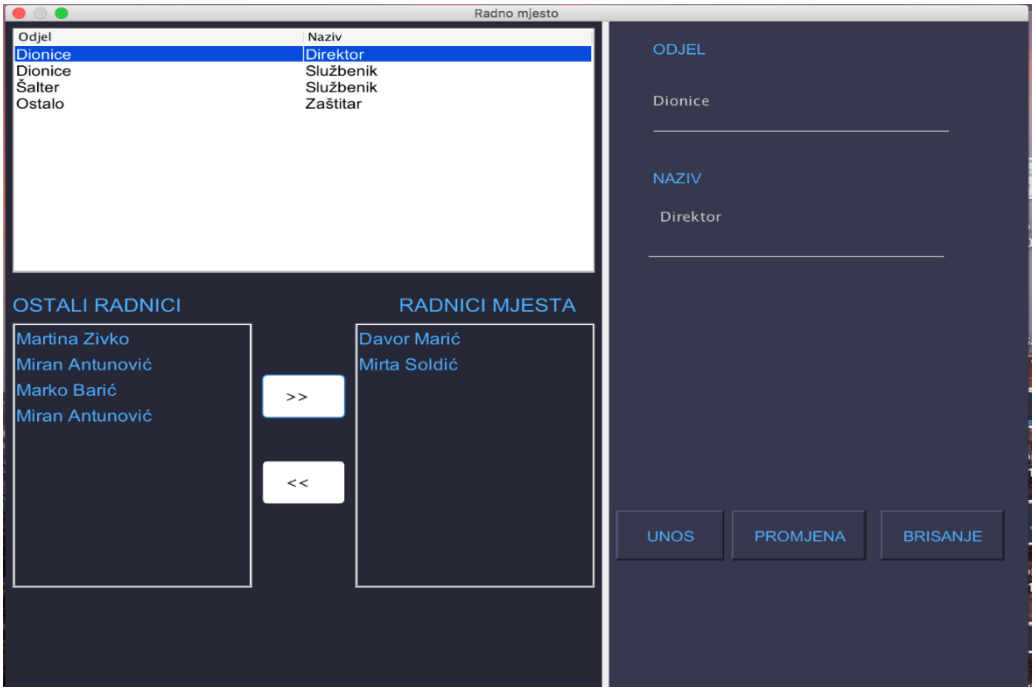
Na glavnom sučelju ukoliko se pritisne na gumb Radnik, otvara se sučelje radnik (Slika 4.3.). S lijeve strane sučelja nalazi se prikaz hijerarhije svih radnika te prikazuje kojem radniku je tko nadređeni. Pritiskom na radnika s desne strane prikazuju se njegove informacije. Nakon što je radnik odabran, pritiskom na padajući izbornik izabire mu se nadređena osoba. Također moguće mu je promijeniti ime, prezime i OIB.



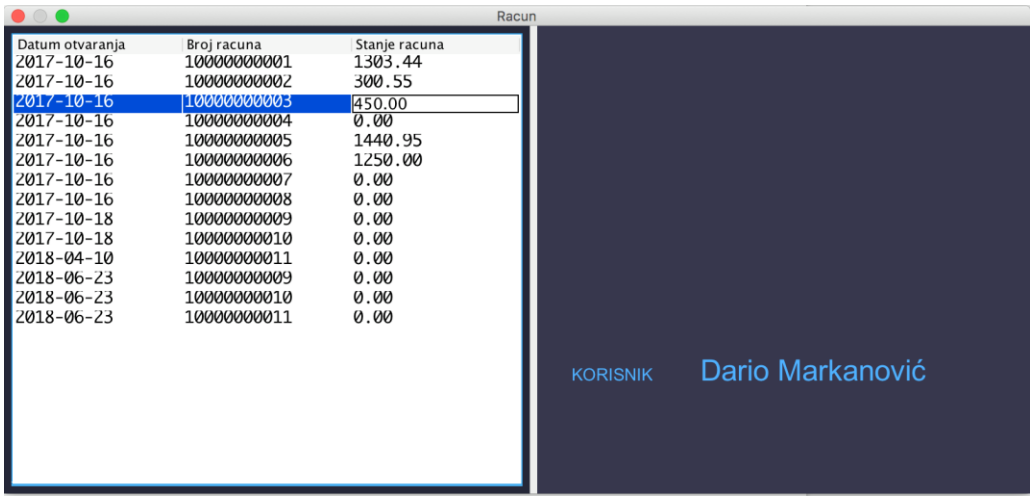
Slika 4.3. Korisničko sučelje *Radnik*

Ukoliko se pritisne na gumb Radno mjesto otvara se novo sučelje koje sa lijeve strane prikazuje listu radnih mjesta. Pritiskom na radno mjesto u listi, ispod liste se prikazuju radnici koji su prethodno dodani na izabrano radno mjesto. Sa desne strane moguće je brisanje, dodavanje i promjena odjela i naziva radnog mjesta što je moguće vidjeti na slici 4.4.

Na početnom zaslonu, pritiskom na gumb Račun, otvara se sučelje (Slika 4.5.) koje se s lijeve strane sastoji od liste računa, a pritiskom na određeni račun, s desne strane prikazuju se informacije o klijentu, vlasniku računa.

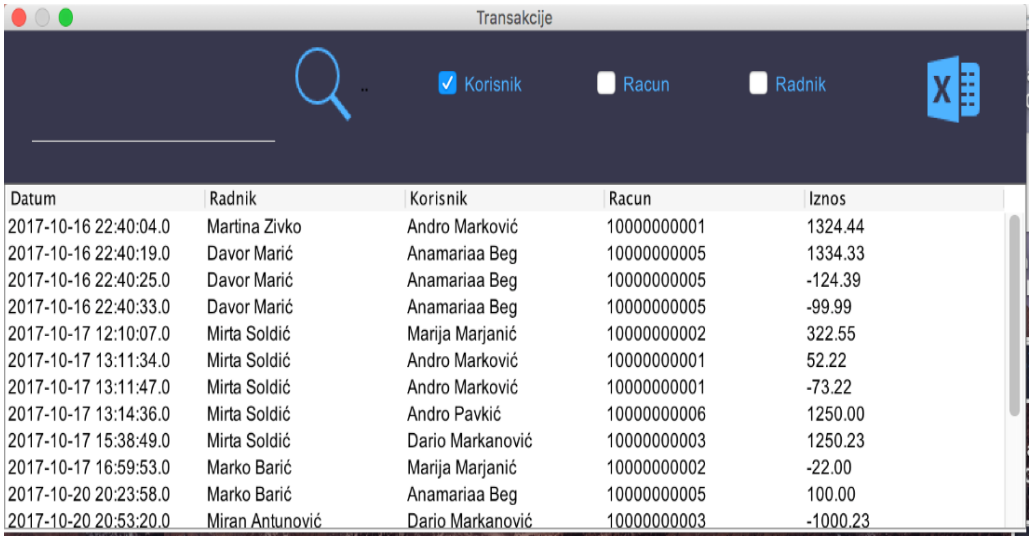


Slika 4.4. Korisničko sučelje *Radna mjesta*



Slika 4.5. Korisničko sučelje *Račun*

Sučelje transakcija (Slika 4.6.), na vrhu s lijeve strane, sadrži polje za unos teksta koje služi za pretraživanje transakcija. Ovisno o okviru (engl. *checkbox*) koji je odabran, pretraživati se mogu transakcije po klijentu, računu te radniku koji je izvršio transakciju. Glavni dio sučelja čini lista koja prikazuje transakcije, uz uvjet da su one prethodno dodane. U gornjem djelu s desne strane nalazi se gumb preko kojeg je prikazan Excel logo. Pritiskom na isti, odabire se mjesto gdje će se trenutno prikazane transakcije izvesti u obliku Excel datoteke.



Slika 4.6. Korisničko sučelje *Transakcije*

5. ZAKLJUČAK

Aplikacijom koja omogućuje uplatu i isplatu novčanih sredstava želi se omogućiti lakša obrada podataka djelatnicima u poslovnim sustavima čija je glavna svrha rukovanje novčanim sredstvima. Također se želi postići jednostavnost korištenja takve aplikacije i što veća preciznost kod transakcija, arhiviranje podataka i što lakši uvid i ispis istih. Pri izradi aplikacije za ovakav sustav mnogo se pozornosti moralo usmjeriti na samu edukaciju o poslovnoj logici koja stoji iza transakcija i obrade podataka. U fazi pripreme za izradu takve aplikacije bilo je potrebno uključiti ljude koji imaju iskustva u takvim sustavima. Na temelju istraživanja i edukacije, neophodno je saznati zahtjeve budućih korisnika kako bi se unaprijedile do sada postojeće aplikacije.

Nakon prikupljanja informacija uslijedio je razvoj aplikacije u kojoj je bio cilj omogućiti radnicima takvog sustava što bolje snalaženje među velikim brojem transakcija, klijenata, računa te radnika. Aplikacija je proširiva te je cilj nastaviti razvoj sve dok ne bude pogodna za produkciju u stvarnom poslovnom sustavu.

LITERATURA

[1] Java Documentation, About the Java technology, https://docs.oracle.com/javase/8/docs/, rujan 2018

[2] A. D., C. M., Downey, Mayfield, Think Java How to Think Like a Computer Scientist, Green Tea Press, Needham, Massachusetts, 2016

[3] B.B, K.S, Bates, Sierra, Head First Java, O'Reilly Media, Sebastopol, 2005

[4] Hibernate ORM, http://hibernate.org, rujan 2018

[5] XAMPP, https://www.apachefriends.org/index.html, rujan 2018

[6] Netbeans IDE Features, https://netbeans.org/features/, rujan 2018

[7] MySQL, https://dev.mysql.com, rujan 2018

[8] SAP, https://help.sap.com/doc/saphelp_uiaddon20/2.05/en-US/91/f233476f4d1014b6dd926db0e91070/loio1eb216151b1b41f1979b7b6c969670df_LowRes.png, rujan 2018

[9] MySQL Beginners, https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html, rujan 2018

[10] AlternativeTo, https://alternativeto.net/software/netbeans/, rujan 2018

SAŽETAK

Ova *desktop* aplikacija u Javi razvijala se sa ciljem pojednostavljenja uplate i isplate novčanih sredstava. Korisnik ima mogućnost otvoriti račun. Uplate i isplate bi vršio jedan od radnika koji je prethodno unesen u sustav. Svaka takva transakcija se sastoji od informacija o: radniku, datumu i vremenu, korisničkom računu te iznosu. Fokus je na transakcijama koje se mogu eksportirati u Excel tablicu i kao takva se može ispisati. Podaci transakcija se mogu filtrirati po radniku, korisniku i računu. Upravljanje hijerarhijom među radnicima (odnos nadređeni, podređeni) te određivanje radnih mjesta svakog radnika.

Ključne riječi: klijent, korisnik, račun, transakcija

TITLE

Application for money transaction management

ABSTRACT

This desktop application in Java has been developed to simplify payment and cash withdrawal. The user has ability to open an account. Payments and disbursements would be made by one of the employees who were previously entered into the system. Each such transaction consists of information about: worker, date and time, user account and amount. The focus is on transactions that can be exported to Excel tables and as such can be printed. Transaction data can be filtered by worker, user, and account. Managing the hierarchy among the workers (relationship of superior, subordinate) and determining the jobs of each worker.

Keywords: account, client, payment, user, transaction

ŽIVOTOPIS

Tomislav Kušević rođen je 28.5.1996. u Koprivnici. Pohađao je Osnovnu školu „Đure Estera“ u Koprivnici. Nakon toga pohađa Opću gimnaziju „Fran Galović“ u Koprivnici. Godine 2015. upisuje Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku – stručni studij informatike te je 2017. godine završio tečaj Jave programera u Osijeku. Nakon toga, odradio 4 je mjeseca stručne prakse u tvrtki COBE u Osijeku te je u trenutku pisanja završnog rada zaposlen u istoj.