

SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE

ZAVRŠNI RAD

**PROGRAMSKA PODRŠKA ZA
ADMINISTRACIJU I REZERVACIJU TERMINA
U SALONU LJEPOTE**

Marija Klarić

Mostar, listopad 2018.

SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE

ZAVRŠNI RAD

PROGRAMSKA PODRŠKA ZA ADMINISTRACIJU I REZERVACIJU TERMINA U SALONU LJEPOTE

Mentor:

Student:

Prof. dr. sc. Sven Gotovac

Marija Klarić

Mostar, listopad 2018.

IZJAVA

Ja, Marija Klarić, izjavljujem i svojim potpisom potvrđujem da sam u izradi završnog rada poštivala Pravilnik o izradi završnog rada Fakulteta strojarstva, računarstva i elektrotehnike, Sveučilišta u Mostaru. Završni rad sam pisala samostalno uz pomoć navedene literature. Zahvaljujem se mentoru prof.dr.sc. Svenu Gotovcu i asistentu Željku Šeremetu na pruženoj pomoći prilikom izrade završnog rada.

Posebnu zahvalnost iskazujem cijeloj svojoj obitelji, prijateljima i kolegama koji su na bilo koji način utjecali na moja postignuća.

Marija Klarić

SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA, RAČUNARSTVA I ELEKTROTEHNIKE

Preddiplomski studij: Računarstvo
Smjer: -
Ime i prezime: Marija Klarić
Broj indeksa: 1357/RR

ZADATAK ZAVRŠNOG RADA

Naslov: PROGRAMSKA PODRŠKA ZA ADMINISTRACIJU I REZERVACIJU
TERMINA U SALONU LJEPOTE

(Software for administration and booking in beauty salon)

Zadatak: Današnje informacijsko komunikacijske tehnologije zaziru u svim sferama poslovanja tako i u uslužnoj djelatnosti kao što je salon ljepote. Zadataka završnog rada je opisati postojeći način funkcioniranja salona ljepote. Potrebno je dati funkcijske i nefunkcijske zahtjeve koje će biti implementirane korištenjem Java programskom jezika i MySQL baze podataka.

Prijava rada: 05.03.2018. g.
Rok za predaju rada: 19.10.2018. g.
Rad predan: 12.10.2018. g.

Mentor:

Prof. dr.sc. Sven Gotovac

Programska podrška za administraciju i rezervaciju termina u salonu ljepote

Sažetak:

Završni rad opisuje i analizira desktop aplikaciju „Programska podrška za administraciju i rezervaciju termina u salonu ljepote“. Na konkretnom primjeru je prikazano moguće rješenje za aplikaciju koja olakšava administraciju i vođenje salona ljepote.

Osnovni cilj aplikacije jest pružiti mogućnost pregledavanja i rezerviranja termina krajnjim korisnicima. Uz to korisnici imaju i mogućnost pregleda vlastitih rezervacija, te otkazivanja istih.

Administrator ima ovlasti upravljanja aplikacijom. Može rukovati podacima o terminima, rezervacijama, kategorijama i korisnicima. Osim pregleda, to uključuje i mogućnost njihovog dodavanja, uređivanja i brisanja. Također, administrator može rezervirati određeni termin (umjesto običnog korisnika – klijenta), ali i otkazati rezervaciju koju je napravio.

U završnom radu su ukratko opisani studiji izvodljivosti, odnosno funkcionalni i nefunkcionalni zahtjevi korisnika i administratora. Nakon toga nam slijedi opis dizajna sustava, što uključuje opis UML dijagrama, USE-Case dijagrama i baze podataka. Prije implementacije opisane su sve tehnologije koje su korištene pri izradi ove desktop aplikacije. U samoj implementaciji, kroz prikaze i opise slika na kojim su prikazani različiti dijelovi aplikacije, korisniku se nastoji približiti način korištenja ove desktop aplikacije.

Aplikacija se sastoji od jednostavne MySQL baze podataka s četiri međusobno povezane tablice, a za izradu grafičkog korisničkog sučelja korišten je JavaFX Scene Builder unutar NetBeans razvojnog okruženja. Cijela aplikacija se oslanja na MVC (Model-View-Controller) arhitekturu, koja razdvaja podatke predstavljene korisniku od sučelja preko kojeg su podaci prezentirani.

Software for administration and booking in beauty salon

Abstract:

In this paper, software for administration and booking in beauty salon is analyzed and described. Possible solution for application that makes administration and salon management easier is shown on a particular example. Main function of this application is providing the possibility for customers to look over their appointments, cancel them or book the new ones. Administrator is the one who manages the data about appointments, customers, etc.

Main part of application is simple MySQL data base with four mutually connected tables. JavaFX Scene Builder in NetBeans environment is used for making the graphical interface. Whole application has MVC (Model –View – Controller) architecture that separates the data for customers from the interface where the data is presented.

Sadržaj

1	UVOD	1
2	STUDIJI IZVODLJIVOSTI	2
2.1	Funkcionalni zahtjevi	3
2.1.1	Funkcionalni zahtjevi za korisnički modul	3
2.1.2	Funkcionalni zahtjevi za administracijski modul	3
2.2	Nefunkcionalni zahtjevi	4
3	DIZAJN SUSTAVA	5
3.1	UML dijagram	5
3.1.1	USE-CASE dijagram	6
3.2	Baza podataka	8
4	OPIS KORIŠTENIH TEHNOLOGIJA	10
4.1	JAVA	10
4.2	JavaFX	10
4.3	Scene Builder	11
4.4	NetBeans	12
4.5	MVC arhitektura	12
4.6	MySQL	13
4.7	phpMyAdmin	14
5	IMPLEMENTACIJA	15
5.1	Izgled korisničkog sučelja običnog korisnika	17
5.2	Izgled korisničkog sučelja administratora	19
	ZAKLJUČAK	25
	LITERATURA	26
	POPIS SLIKA	27
	PRILOZI	28

1 UVOD

Kako vrijeme prolazi, u većem dijelu svijeta internet je preuzeo glavnu ulogu. Kao jedan od glavnih medija postao je vrlo važan za komunikaciju, prijenos informacija, ali isto tako i za poslovanje. Desktop aplikacije su također dobile svoju ulogu u svemu tome. Omogućile su određenom dijelu korisnika da na što brži, lakši i jednostavniji način ispune svoje korisničke zahtjeve, ali i zaposlenicima su pojednostavile i ubrzale dio posla. Primjer jedne takve desktop aplikacije je "Programska podrška za administraciju i rezervaciju termina u salonu ljepote". Implementacija ovakve aplikacije uvelike bi pojednostavila i olakšala dio posla u salonima ljepote. Naravno, osim ovog načina rezerviranja, preko desktop aplikacije, i dalje bi bilo moguće izvršavati rezervacije telefonskim putem, ili osobnim dolaskom u salon. U tom slučaju bi administrator rezervirao određeni termin u ime klijenta.

U ovom završnom radu, u pet poglavlja, opisana je ova već spomenuta aplikacija. Aplikacija krajnjem korisniku omogućuje pregled liste svih dostupnih termina koje salon nudi. Svaki termin popraćen je sa osnovnim podacima koje korisnik treba znati. Pruža mu mogućnost rezervacije termina, kao i otkazivanja rezervacije istih. Administrator ima mogućnost pregleda liste svih termina, rezervacija, kategorija i korisnika. Uz to, ovlašten je i za dodavanje, uređivanje i brisanje termina i kategorija, rezerviranje termina i otkazivanje rezervacija, uklanjanje završenih rezervacija sa liste rezervacija, kao i dodavanje ili uklanjanje administratora.

Nakon prvog, uvodnog dijela dolazimo do drugog poglavlja - studiji izvodljivosti. Tu su kratko opisani funkcionalni i nefunkcionalni zahtjevi za aplikaciju. Treći dio opisuje UML i USE-CASE dijagrame, te bazu podataka. Sljedeće, četvrto poglavlje, sadrži pregled i opis najvažnijih tehnologija koje su se koristile za razvoj ove aplikacije. Također, sadrži njihove glavne prednosti i značajke zbog kojih su izabrani za pisanje aplikacije. Nakon toga dolazimo do poglavlja implementacije aplikacije. Tu su detaljno opisani izgledi korisničkih sučelja administratora i krajnjeg korisnika. Radi lakše razumljivosti sve je popraćeno sa dosta slika. U posljednjem, petom poglavlju, dat je zaključak.

2 STUDIJI IZVODLJIVOSTI

Softverski zahtjevi se definiraju na razini sustava. Njima se ustanovljavaju aktivnosti koje korisnici zahtijevaju (očekuju) od sustava, te uvjeti pod kojima će se sustav razvijati kako bi udovoljio postavljenim zahtjevima. Sami zahtjevi predstavljaju opis aktivnosti sustava kao i opis pripadajućih ograničenja koja se generiraju tijekom procesa definiranja zahtjeva.

Zahtjevi se mogu definirati na najvišoj razini apstrakcije sustava ili detaljnim, matematičkim izraženim opisom funkcija sustava.

Evolucija softverskih zahtjeva razvija se kroz tri koraka:

- Prepoznavanje potreba (želja) korisnika
- Otkrivanje ključnih karakteristika identificiranih potreba
- Preoblikovanje identificiranih karakteristika u zahtjeve i njihovo dokumentiranje

Zahtjevi mogu biti: funkcionalni i nefunkcionalni.

Funkcionalni zahtjevi (ŠTO) odnose se na prikaze aktivnosti koje sustav treba izvršiti, kako sustav treba reagirati na određene ulaze, te kako će se sustav ponašati u određenim situacijama.

Nefunkcionalni sustavi (KAKO) se odnose na karakteristike (ograničenja) koje softver mora imati. Karakteristike koje sustav postavlja u odnosu na aktivnosti i funkcije koje sustav obavlja (vremenska ograničenja, ograničenja u razvojnem procesu, standardi ...). [1]

2.1 Funkcionalni zahtjevi

Kako bismo pojednostavili razumijevanje i izradu aplikacije, funkcionalne zahtjeve ćemo podijeliti na dva navedena modula: korisnički modul (krajnji, obični korisnici) i administracijski modul (administratori).

2.1.1 Funkcionalni zahtjevi za korisnički modul

- Registracija korisnika na sustav
- Prijava korisnika na sustav
- Prikaz liste svih dostupnih termina
- Mogućnost rezervacije termina
- Prikaz rezervacija prijavljenog korisnika
- Mogućnost otkazivanja rezervacije
- Odjava korisnika sa sustava

2.1.2 Funkcionalni zahtjevi za administracijski modul

- Prijava korisnika (administratora) u sustav
- Prikaz liste svih dostupnih termina
- Mogućnost rezervacije termina
- Mogućnost dodavanja novog termina, te uređivanja i brisanja postojećeg termina
- Prikaz rezervacija svih korisnika
- Mogućnost otkazivanja rezervacije
- Uklanjanje završene rezervacije sa liste rezervacija
- Prikaz liste svih kategorija
- Mogućnost dodavanja nove kategorije, te uređivanja i brisanja postojeće kategorije
- Prikaz liste svih registriranih korisnika
- Mogućnost dodavanja i uklanjanja administratora
- Odjava korisnika (administratora) sa sustava

2.2 Nefunkcionalni zahtjevi

- Aplikacija treba biti jednostavna za upotrebu
- Aplikacija mora imati jednostavan dizajn
- Aplikaciji se može pristupiti sa različitih platformi
- Dostupna samo na računalu na kojem je instalirana s lokalnom bazom podataka
- Sastoji se od MySQL baze podataka i formi korisničkog sučelja
- Mogućnost nadogradnje aplikacije dodatnim funkcionalnostima u budućnosti

3 DIZAJN SUSTAVA

3.1 UML dijagram

Ujedinjeni jezik za modeliranje (engl. Unified Modeling Language, kraće UML) je normirani jezik opće namjere koji se koristi za modeliranje računalnih sustava temeljenih na objektno orijentiranoj paradigmi. UML uključuje skup tehnika koje ostvaruju grafički prikaz objektno orijentiranih računalnih sustava. Dijagrami unutar UML-a mogu se podijeliti s obzirom na dinamičnost na statičke i dinamičke dijagrame.

Statički dijagrami ne razmatraju vremensku komponentu sustava, već daju sliku dijelova ili cijelog sustava kakav postoji u nekom trenutku. Težnja dinamičkih dijagrama je da uključe međudjelovanje sudionika i vremensku komponentu u opis sustava kako bi se modelirali sljedovi događaja unutar sustava. [2]

1. Strukturni dijagrami (structure diagram):

- Dijagram razreda (class diagram)
- Dijagram paketa (package diagram)
- Dijagram objekata (object diagram)
- Dijagram komponenti (component diagram)
- Dijagram razmještaja (deployment diagram)
- Dijagram profila (profile diagram)
- Dijagram složene strukture (composite structure diagram)

2. Ponašajni dijagrami (behavior diagram):

- Dijagram aktivnosti (activity diagram)
- Dijagram stanja (state diagram)
- Dijagram obrazaca uporabe (use-case diagram)
- Dijagram međudjelovanja (interaction diagram):
 - Dijagram vremena (timing diagram)
 - Dijagram pregleda međudjelovanja (interaction overview diagram)
 - Sekvencijski dijagram (sequence diagram)
 - Komunikacijski dijagram (communication diagram)

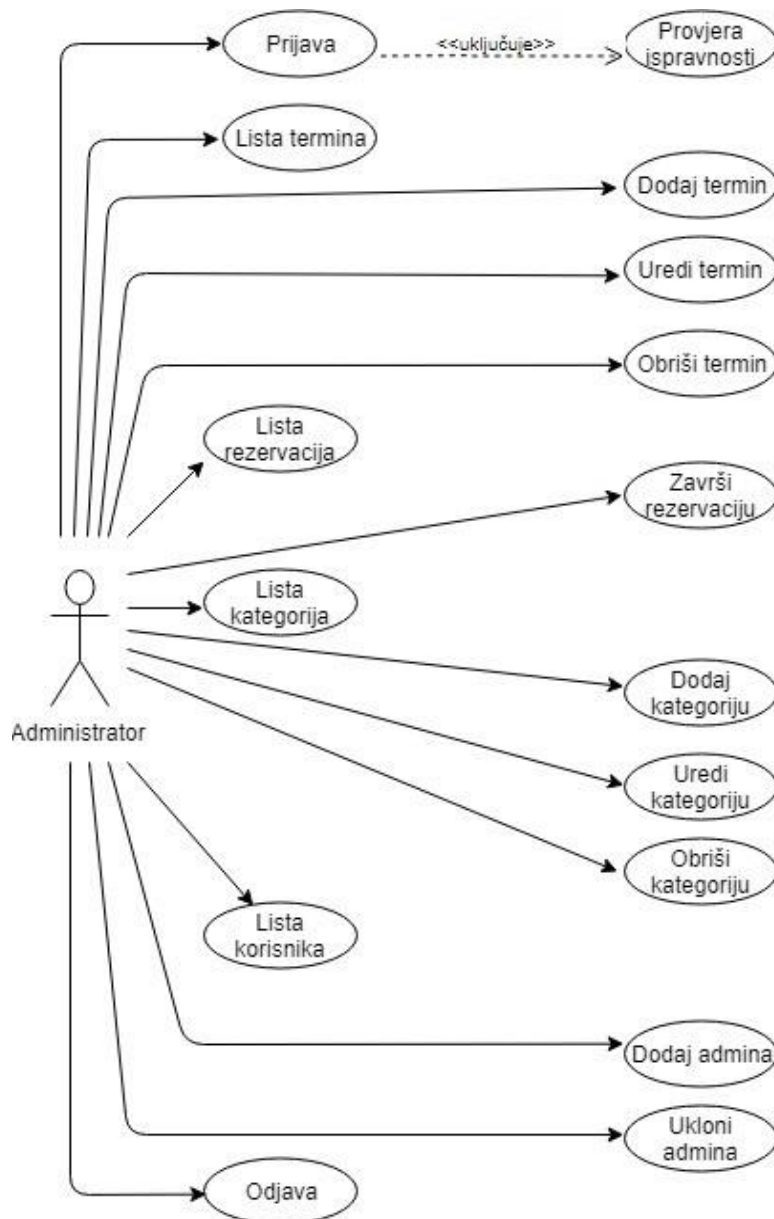
3.1.1 USE-CASE dijagram

Dijagrami obrazaca uporabe (engl. Use-case diagrams) koriste se da bi se prikazalo ponašanje sustava, dijelova sustava ili konkretnog razreda na način vidljiv korisniku sustava. Ponašanje sustava opisano je pomoću ciljeva i aktera koji predstavljaju apstrakciju korisnika sustava, odnosno općenitije nekog od dionika sustava.

Dijagrami obrazaca uporabe su statički UML-dijagrami (kao i dijagrami razreda, objekata, paketa i razmještaja), a također pripadaju i skupini ponašajnih dijagrama budući da modeliraju moguće ponašanje korisnika sustava. Prema UML-specifikaciji obrazaca uporabe sastoje se od aktera, obrazaca uporabe, te veza i odnosa među aktorima i obrascima uporabe. [2]

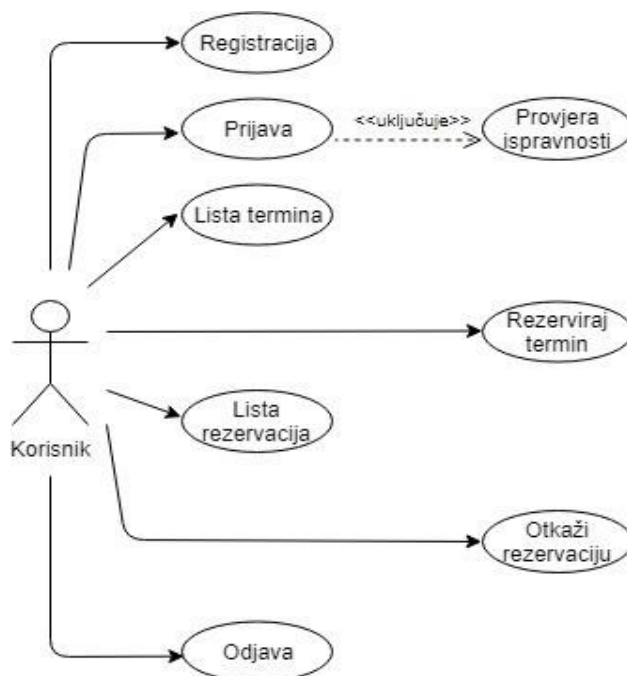
Aktori mogu biti ljudi (klijenti sustava), vanjski računalni sustavi, unutarnji računalni sustavi i sl.

U aplikaciji koju opisujemo razlikujemo dva aktera, dvije vrste korisnika: administrator i obični korisnik. Administrator ima sve ovlasti upravljanja aplikacijom. Nakon prijave u sustav ima mogućnost pregleda svih termina, rezervacija, kategorija i korisnika. Osim toga, omogućeno mu je uređivanje i brisanje postojećih termina i kategorija, te dodavanje novih termina i kategorija. Također, administrator može ukloniti završenu rezervaciju sa liste rezervacija, dodavati novog ili ukloniti već postojećeg administratora. Kao i prijava, omogućena mu je i odjava sa sustava, odnosno aplikacije.



Slika 2.1 Use-Case dijagram administratora

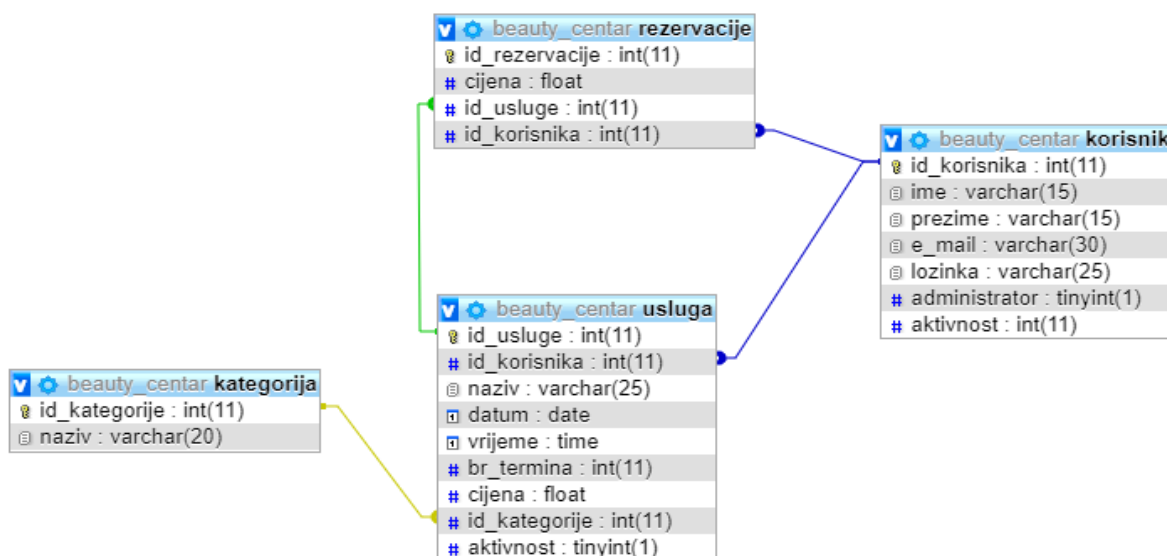
Drugi aktor je obični korisnik koji koristi sustav (aplikaciju). On se prvo mora registrirati, a potom prijaviti. Nakon prijave prikazuje mu se lista svih dostupnih termina koje može rezervirati. Također, korisnik može pregledati sve svoje rezervacije, kao i otkazati iste. Pri završetku ima mogućnost odjave sa aplikacije. Sekvencijalni dijagrami



Slika 2.2 Use-Case dijagram običnog korisnika

3.2 Baza podataka

Baza podataka je rađena u besplatnom alatu phpMyAdmin, napisanom u PHP-u, koji se koristi u web pregledniku. Sastoji se od četiri tablice: *kategorija*, *korisnik*, *rezervacije* i *usluga*.



Slika 2.3 Prikaz tablica u bazi podataka

Tablica *kategorija* sadrži dva atributa: *id_kategorije* i *naziv*. *Id_kategorije* je primarni ključ tablice i postavljen je na *auto increment* (vrijednost mu se automatski dodjeljuje). U „naziv“ se sprema naziv kategorije. Povezana je sa tablicom *usluga*.

Tablica *korisnik* sadrži sve potrebne podatke o korisniku kao što su: *id_korisnika*, *ime*, *prezime*, *e_mail*, *lozinka*, *administrator* i *aktivnost*. Povezana je sa tablicom *rezervacije* kako bi se znalo koji je korisnik izvršio rezervaciju, te sa tablicom *usluga* kako bi zapamtili tko je rezervirao određeni termin. Atribut *id_korisnika* je jedinstven, kao i atribut *e_mail*. Atribut *administrator*(boolean) vraća „true“ ako je prijavljeni korisnik admin, a u suprotnom slučaju vraća „false“.

U tablici *rezervacije* imamo attribute *id_rezervacije* i *cijena*, te attribute *id_usluge* i *id_korisnika*. Atributi *id_usluge* i *id_korisnika* su ujedno i strani ključevi, što nam govori da je ova tablica vezana sa tablicama *korisnik* i *usluga*. Ona pohranjuje informacije o napravljenj rezervaciji.

Tablica *usluga* je vezana sa sve tri preostale tablice. Primarni ključ tablice je *id_usluge*, a posjeduje i strane ključeve *id_korisnika* i *id_kategorije*. Uz njih imamo i attribute *naziv*, *datum*, *vrijeme*, *br_termina*, *cijena* i *aktivnost* u koje se spremaju informacije vezane za određeni termin.

4 OPIS KORIŠTENIH TEHNOLOGIJA

4.1 JAVA

Java je objektno-orijentiran programski jezik, koji je razvila firma Sun Microsystems početkom devedesetih godina. Mnogi koncepti Jave su bazirani na jeziku Oberon. [3]

Jedna od temeljnih vodilja u razvoju ovog jezika, koja je dovela do ovakve opće prihvatljivosti jest ideja *napiši-jednom-pokreni-bilo-gdje*. Centralna tema u razvoju je potpora za višeplatformnost, odnosno ideja da se programeru ponudi apstraktni pogled na računalo – pogled koji ne uključuje platformski specifične detalje poput širine podatkovne riječi centralnog procesora, direktan pristup upravljačkim programima operacijskog sustava ili podržanom sklopovlju. Java definira apstraktni računarski stroj kod kojeg je sve propisano i opisano specifikacijom. Za programera je u konačnici nebitno hoće li se njegov program izvoditi na 16-bitnom, 32-bitnom ili 64-bitnom procesoru. Da bi to bilo moguće, prilikom programiranja u Javi programi se pišu za Javin virtualni stroj. To je stroj koji je definiran specifikacijom i na kojem se izvode svi programi pisani u Javi. Zahvaljujući ovakvoj virtualizaciji programer doista ne treba razmišljati o specifičnostima platforme na kojoj će se izvoditi program koji je napisao, a to upravo i jest ideja izrade aplikacija za više platformi odjednom. [4]

Neke od karakteristika Jave: Jednostavnost, objektna orijentiranost, platformska neovisnost, sigurnost, robusnost, arhitekturna neutralnost, prenosivost, dinamičnost interpretiranost, visoke performanse, višenitnost, distribuiranost.

4.2 JavaFX

JavaFX je radni okvir za razvoj Java GUI programa. Na početku, Java je nudila biblioteku klasa za rad sa grafikom pod nazivom *Abstract Windows Toolkit*(AWT). Ove klase su omogućavale razvoj jednostavnog grafičkog korisničkog interfejsa (Graphis User Interface – GUI), ali ne i zahtjevnijih aplikacija. Pored toga, AWT je zavisio od izvršne platforme, te je bio izložen i njihovim bagovima.

AWT je kasnije zamijenjen sa novom, robusnijom, boljom i fleksibilnijom bibliotekom tzv. *Swing* komponenata. *Swing* komponente se direktno nanose i boje na „platnu“ (canvas) u

pozadini upotrebom Java koda. Swing komponente manje zavise od platforme na kojoj se izvršavaju, i upotrebljavaju manje vlastitih GUI resursa. Swing je razvijen za razvoj desktop GUI aplikacija.

Swing je sada zamijenjen potpuno novom GUI platformom koja je nazvana *JavaFX*. *JavaFX* sadrži moderne GUI tehnologije koje omogućavaju razvoj „bogatih Internet aplikacija“ (RIA – rich Internet applications). *JavaFX* može raditi isto i u desktop aplikaciji i u Web browser-u. *JavaFX* nudi i jedno novo svojstvo, a to je podrška radu uređaja koji reagiraju na dodir prstiju korisnika, kao što su smart telefoni i tableti. *JavaFX* podržava primjenu 2D i 3D grafike i animacija, prikaz video i audio snimaka, a radi ili kao nezavisna desktop aplikacija ili korištenjem web browser-a. *JavaFX* je lakša i jednostavnija za naučiti, te je samim tim lakša i izrada GUI programa. Nakon pojavljivanja *JavaFX*-a, Swing je „osuđen na smrt“, jer aplikacije izrađene u *JavaFX* rade na različitim platformama, web-u, stolnim i prenosivim uređajima (računalima).

4.3 Scene Builder

Scene Builder generira izvorni kod FXML dokumenta kako se definira grafičko korisničko sučelje za *JavaFX* aplikaciju.

Omogućava da se brzo i jednostavno napravi prototip interaktivne aplikacije koja povezuje vizualne komponente sa logikom aplikacije.

JavaFX Scene Builder je alat koji služi za dizajniranje *JavaFX* aplikacija. Omogućuje jednostavno prevlačenje (drag-and-drop) i pozicioniranje komponenti grafičkog korisničkog sučelja na *JavaFX* scenu. Kako se gradi scena, tako se automatski generira odgovarajući FXML dokument. [5]

Osnovni prozor Scene Buildera se sastoji iz nekoliko panela:

- Library Panel – omogućuje izbor, prevlačenje i pozicioniranje komponente u centralnu radnu površinu (Content Panel)
- Document Panel – sadrži hijerarhijski prikaz čvorova grafa scene (može se izabrati način prikaza hijerarhije). Također, sadrži i informacije o Controller klasi.
- Inspector Panel – podijeljen je na tri sekcije: Properties, Layout i Code

4.4 NetBeans

NetBeans je jedan od najkvalitetnijih *open source* razvojnih IDE-a (engl. Integrated Development Environment) na tržištu. Primarno je namijenjen razvoju Java aplikacija, no danas omogućava i razvoj aplikacija u mnogim drugim programskim jezicima. Razvijen je pomoću Java programskog jezika, a njegova dostupnost je omogućena na svim platformama. Nastao je iz studentskog projekta zvanog Xelfi 1996. godine u Češkoj. Nakon postavljanja na tržište postiže veliki uspjeh, a zbog toga ga iste godine kupuje tvrtka Sun Microsystems. Već nakon godinu dana objavljen je pod *open source* licencom. Ono što NetBeans čini posebno zanimljivim je set integriranih alata za izradu i razvoj grafičkih korisničkih sučelja baziranih na standardnim AWT (Abstract Windowing Toolkit) i JFC/SWING (Java Foundation Classes) komponentama. Po mogućnostima i osobinama NetBeans IDE parira ostalim komercijalnim alatima, a zbog svoje jednostavnosti izabran je kao alat za izradu Java desktop aplikacije ovog završnog rada.

4.5 MVC arhitektura

MVC je kodna arhitektura koja je, iako originalno razvijena za *desktop* platforme, danas popularna na suvremenim *web*-platformama. Glavne ideje ovoga modula su razdvajanje koda u zasebne cjeline, odnosno dosljedna struktura i mogućnost ponovnog korištenja istog koda (engl. *Code reusability*). Sama ideja je prvi put javno prezentirana već 1988. godine (Krasner, Pope, 1988), a do današnjeg dana doživjela je nekoliko iteracija pa je stvoreno nekoliko inačica originalne ideje (HMVC, MVA, MVP, MVVM, itd.).

MVC arhitektura sastoji se od određenih komponenti u kojima je svaka zadužena za obavljanje specifičnih funkcija. [6]

MVC tehnologija se koristi kako bi se razdvojilo korisničko sučelje (View) od poslovne logike (Controller) i modela podataka (Model).

Model je odgovoran za rad s podacima, poput konekcije na bazu podataka, izvršavanje upita, implementaciju poslovnih pravila itd. Podaci se prvotno prosljeđuju objektu Controller koji naposljetku podatke prosljeđuje objektu View. Ukratko, Model odgovara na zahtjeve koji stižu iz Controller objekta, priprema i obrađuje podatke, te obavještava registrirane View objekte da ažuriraju svoj prikaz sa novim podacima.

View uzima podatke od objekta Controller, formatira ih, te proslijeđuje pregledniku. Zadužen je za prikaz podataka i dodaje funkcionalnosti koje su potpuno izolirane od složenih opcija nad podacima.

Controller je zadužen za upravljanje akcijama. Kada korisnik šalje zahtjev sustavu preko korisničkog sučelja (View-a) Controller proslijeđuje zahtjev Modelu koji obrađuje i priprema podatke, te nakon toga vraća odgovor. [7]

Prednosti korištenja MVC arhitekture:

- Lakše održavanje, testiranje, te nadograđivanje aplikacije
- Jednostavnije dodavanje novih klijenata na vlastitim View i Controller objektima
- Fleksibilnost kod planiranja i implementiranja objekata Model, omogućena višestruka iskoristivost i modularnost
- Omogućen paralelni razvoj objekata Model, View i Controller
- To sve aplikaciju čini proširivom i skalabilnom

4.6 MySQL

MySQL je najrasprostranjeniji *open source* sustav za upravljanje bazama podataka. Njegov razvoj je započeo 1995. godine. [8] Proizvod je firme MySQL AB iz Švedske.

Jedan je od sustava za upravljanje relacijskim bazama podataka. Ovaj program se ponaša kao server sa multi-user funkcijom, odnosno dozvoljava pristup više korisnika. Svaka MySQL baza može imati nekoliko korisnika koji joj mogu pristupiti, a svaki korisnik ima predefinirane mogućnosti za rad. Ovakav pristup uz dobra podešavanja znatno umanjuje mogućnost greške. MySQL kao sustav može raditi na različitim operacijskim sustavima.

Neke karakteristike MySQL-a:

- Sustav za upravljanje bazama podataka
- Sustav za upravljanje relacijskim bazama podataka
- MySQL softver je softver otvorenog koda (Open Source)
- MySQL server je brz, pouzdan i jednostavan za korištenje
- Radi u klijent/server sustavima

4.7 phpMyAdmin

PhpMyAdmin je besplatan alat napisan u PHP-u koji služi za upravljanje i administraciju MySQL-a preko World Wide Web-a. Pomoću ovog alata moguće je izvršavati mnoge MySQL upite i operacije. Moguće je stvoriti novu bazu podataka, novu tablicu, pregledavati i mijenjati podatke i tipove podataka u tablicama, te izvršiti brojne druge akcije potrebne za rad s bazom podataka. Aplikacija se pokreće u web pregledniku. [9]

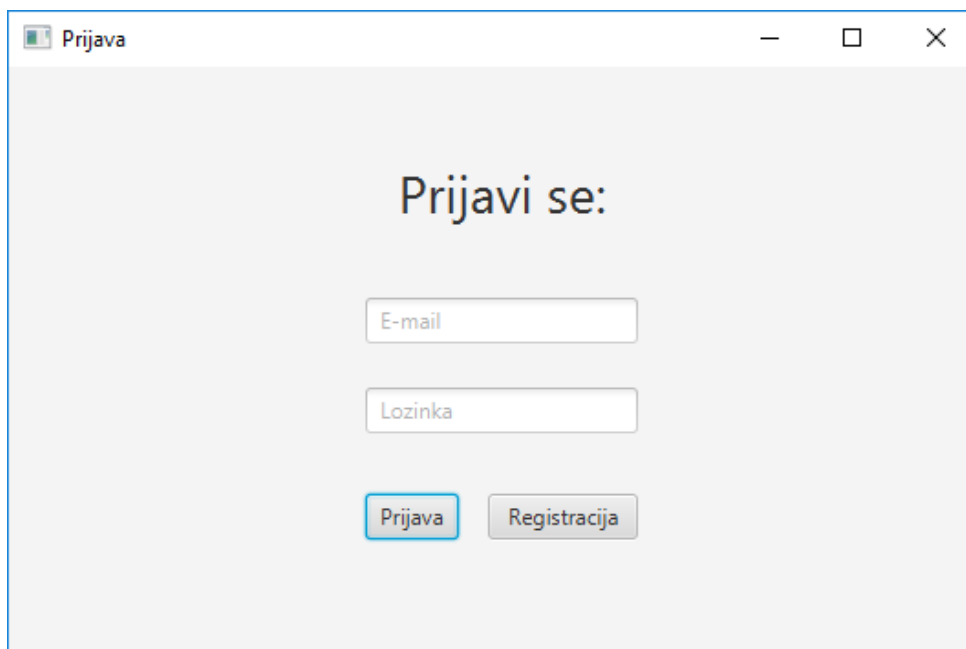
PhpMyAdmin je jedan od najpopularnijih PHP aplikacija i MySQL administracijskih alata, s velikim brojem korisnika i suradnika.

Neke od glavnih karakteristika PhpMyAdmin-a su:

- Web sučelje
- Upravljanje bazom podataka MySQL i MariaDB
- Uvoz podataka iz CSV i SQL
- Izvoz podataka u radnim formatima (CSV, SQL, XML, PDF, ...)
- Prikaz baza podataka u PDF-u

5 IMPLEMENTACIJA

Desktop aplikacija „Programska podrška za administraciju i rezervaciju termina u salonu ljepote“ je pisana u Java programskom jeziku. Pri pokretanju aplikacije otvara nam se prozor za prijavu korisnika u sustav. Korisnik se prijavljuje upisujući svoj e-mail i lozinku. Svoju prijavu potvrđuje button-om „Prijavi se“. Prozor je istog izgleda i za prijavu običnog korisnika i za prijavu administratora (28Prilog 1):



Slika 4.1 Prijava korisnika

Novi korisnici prije prijave u sustav moraju se registrirati. Pri registraciji popunjavaju svoje osnovne osobne podatke (ime, prezime, e-mail, lozinka).

Nakon uspješne registracije novog korisnika ponovno se otvara prozor za prijavu. Svaki novi, tek registrirani korisnik se vodi kao „običan korisnik“, klijent. Administrator može postati tek kada ga već postojeći administrator postavi kao administratora.

Izgled prozora registracije vidimo na sljedećoj slici (Slika 4.2.) (Prilog 2):

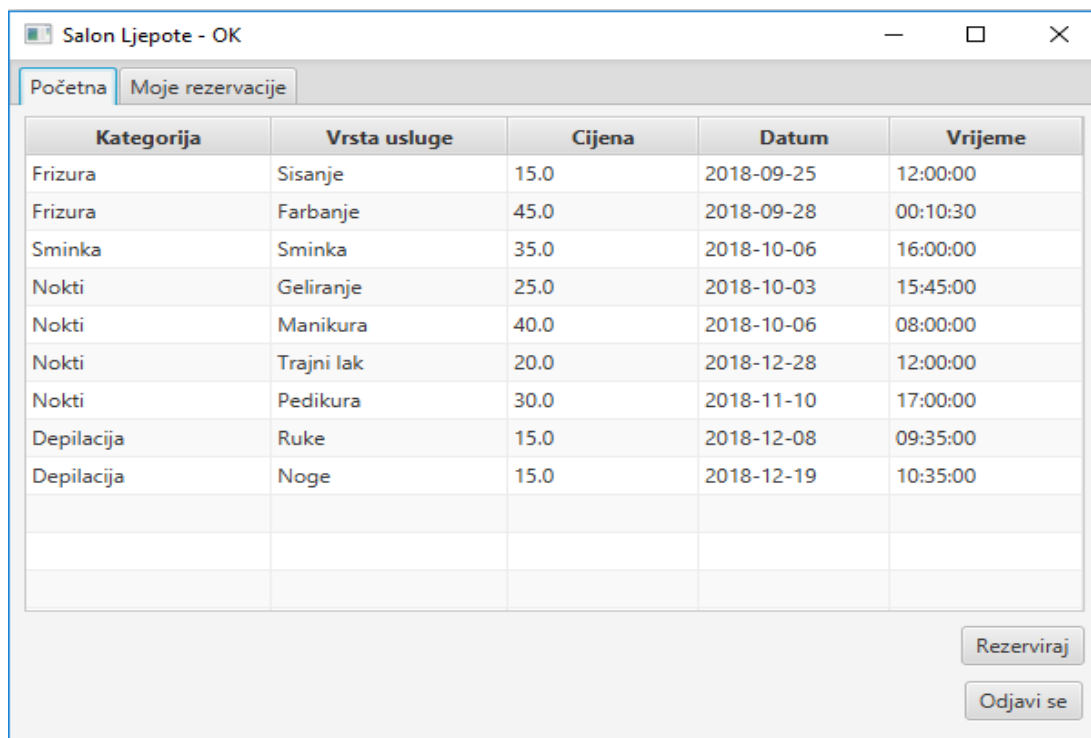
Slika 4.2 Registracija korisnika

Ukoliko bi se pri prijavi korisnika unijeli pogrešni podaci ili se neki podatak ne bi unio, program bi nam ispisao poruku o pogrešci. Isto bi se dogodilo i kod registracije korisnika. Ako bi se pokušao registrirati sa e-mailom koji se već nalazi u bazi, ako bi unio prekratke vrijednosti podataka, ili ako uopće ne bi unio podatke registracija bi bila neuspješna. Primjer poruke o pogrešci vidimo na sljedećim slikama:

Slika 4.3 Neuspješna prijava/registracija

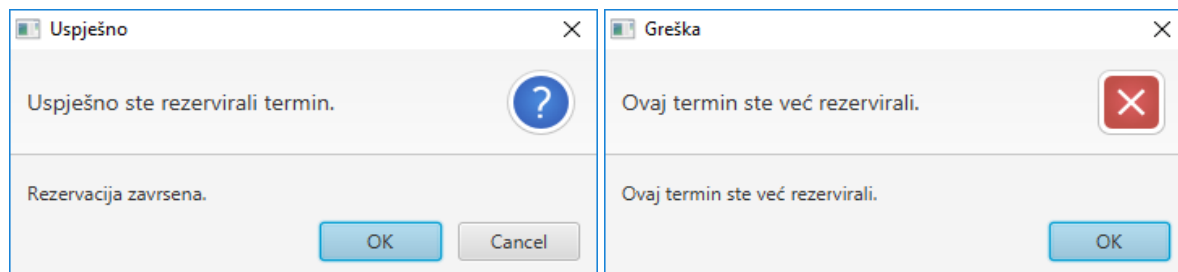
5.1 Izgled korisničkog sučelja običnog korisnika

Nakon uspješne prijave običnog korisnika u sustav otvara nam se prozor kao na slici 4.4. Korisniku je u tome prozoru vidljiva lista svih termina koje je moguće pregledati i rezervirati. U tablici se nalaze osnovni podaci koje korisnik mora znati o svakom terminu, a to su: kategorija, vrsta usluge, cijena, datum i vrijeme.



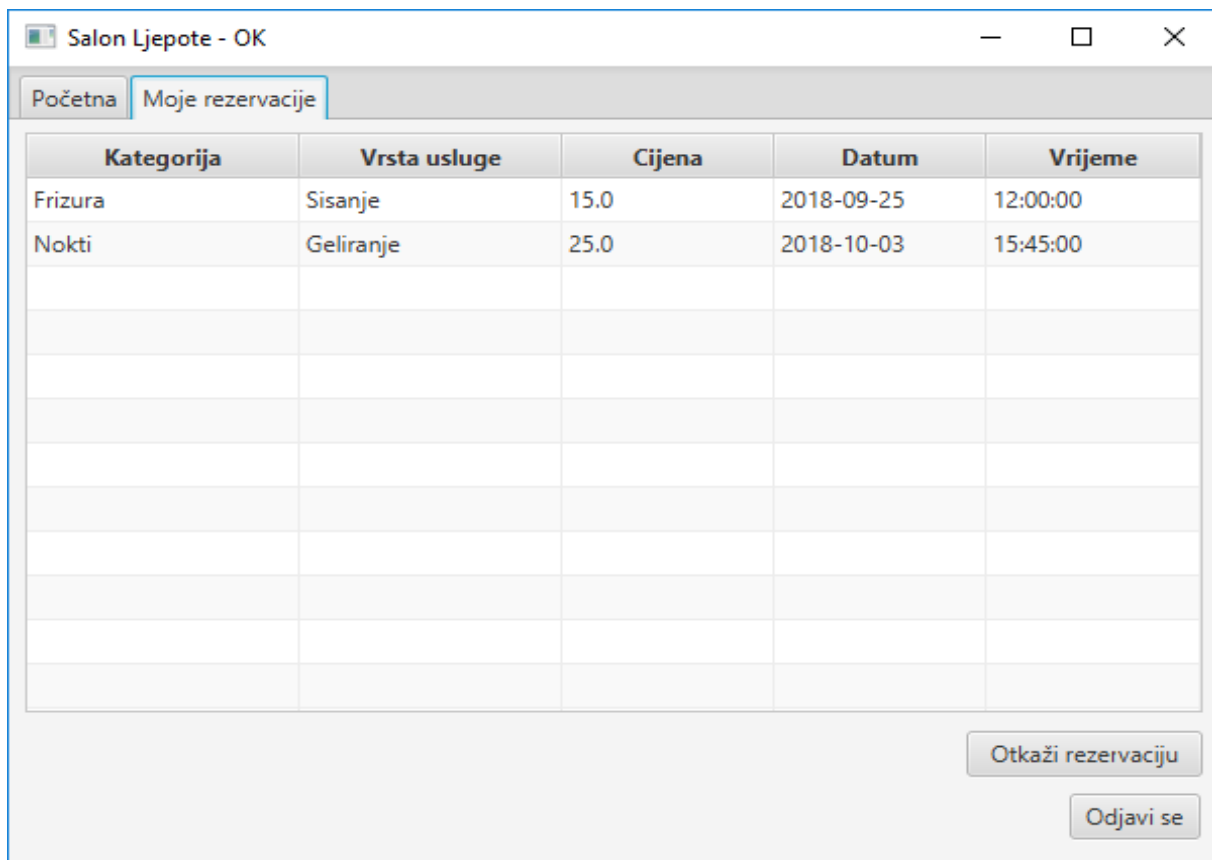
Slika 4.4. Korisničko sučelje običnog korisnika

Ukoliko korisnik odluči rezervirati neki od ponuđenih termina, odabire taj termin, a potom rezervaciju potvrđuje klikom na *button* „Rezerviraj“ (Prilog 3). Tada mu se na glavnom ekranu prikaže *alert* prozor sa porukom o uspješnoj ili neuspješnoj rezervaciji termina (Slika 4.5), a napravljena rezervacija se sprema u bazu podataka.



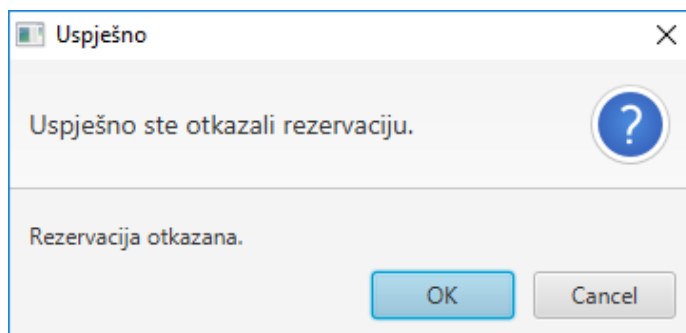
Slika 4.5 Informacijski prozori sa porukom o (ne)uspješnoj rezervaciji

Pri odabiru kartice „Moje rezervacije“ korisniku se otvara prozor sa prikazom liste svih rezervacija koje je napravio. U toj tablici su također sadržani svi bitni podaci o terminu, odnosno napravljennoj rezervaciji (kategorija, vrsta usluge, cijena, datum, vrijeme).



Slika 4.6 Prozor „Moje rezervacije“

Ukoliko se korisnik predomisli i odluči otkazati napravljenu rezervaciju, to mu je omogućeno klikom na *button* „Otkazi rezervaciju“ (Prilog 4). Tada mu se također prikaže *alert* prozor s porukom o uspješnosti otkazivanja rezervacije (Slika 4.7).



Slika 4.7 Otkazana rezervacija

Naravno, pri završetku korisniku je omogućena odjava sa sustava klikom na *button* „Odjavi se“ (Prilog 11), te ga to vodi ponovno na prozor „Prijava“.

5.2 Izgled korisničkog sučelja administratora

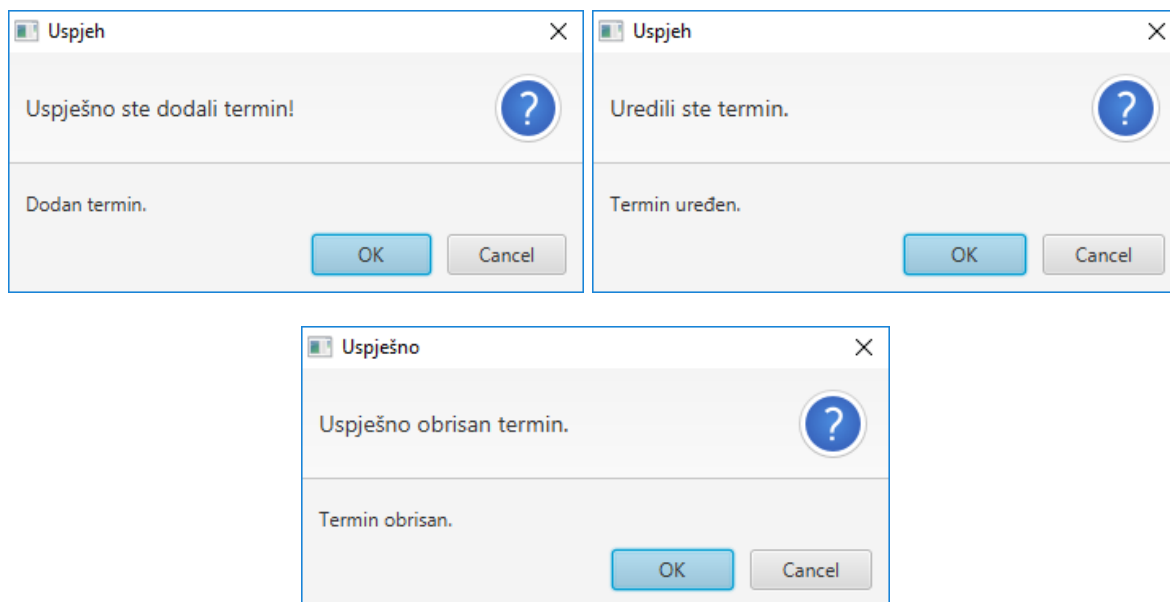
Korisničko sučelje administratora se uvelike razlikuje od korisničkog sučelja običnog korisnika. Nakon prijave u sustav glavni prikaz (početni prozor) koji administrator vidi izgleda ovako:

Kategorija	Vrsta usluge	Cijena	Datum	Vrijeme
Frizura	Sisanje	15.0	2018-09-25	12:00:00
Frizura	Farbanje	45.0	2018-09-28	00:10:30
Sminka	Sminka	35.0	2018-10-06	16:00:00
Sminka	Sminka	17.0	2018-10-16	09:30:00
Nokti	Geliranje	25.0	2018-10-03	15:45:00
Nokti	Manikura	40.0	2018-10-06	08:00:00
Nokti	Trajni lak	20.0	2018-12-28	12:00:00
Depilacija	Slobodan izbor	0.0	2018-11-10	08:00:00
Depilacija	Ruke	15.0	2018-12-08	09:35:00
Depilacija	Noge	15.0	2018-12-19	10:35:00

Slika 4.8 Korisničko sučelje administratora

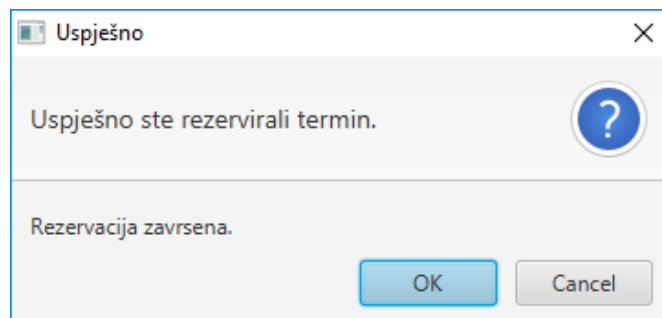
Osim pregleda liste svih termina, kao što to vidi i običan korisnik, administrator na početnoj stranici može izvršiti i dodavanja novog termina popunjavanjem navedenih *TextField*-ova, *choiceBox*-a i *DatePicker*-a, a dodavanje potvrđuje klikom na *button* „Dodaj“. Prilikom potvrde dodavanja novog termina izmjene se spremaju u bazu podataka. Isto tako,

administrator može uređivati i brisati postojeće termine, a radnje potvrđuje *button*-ima „Uredi“ i „Obriši“ (Prilog 5). Nakon svake izvršene radnje administrator prima obavijest o njenoj uspješnosti:



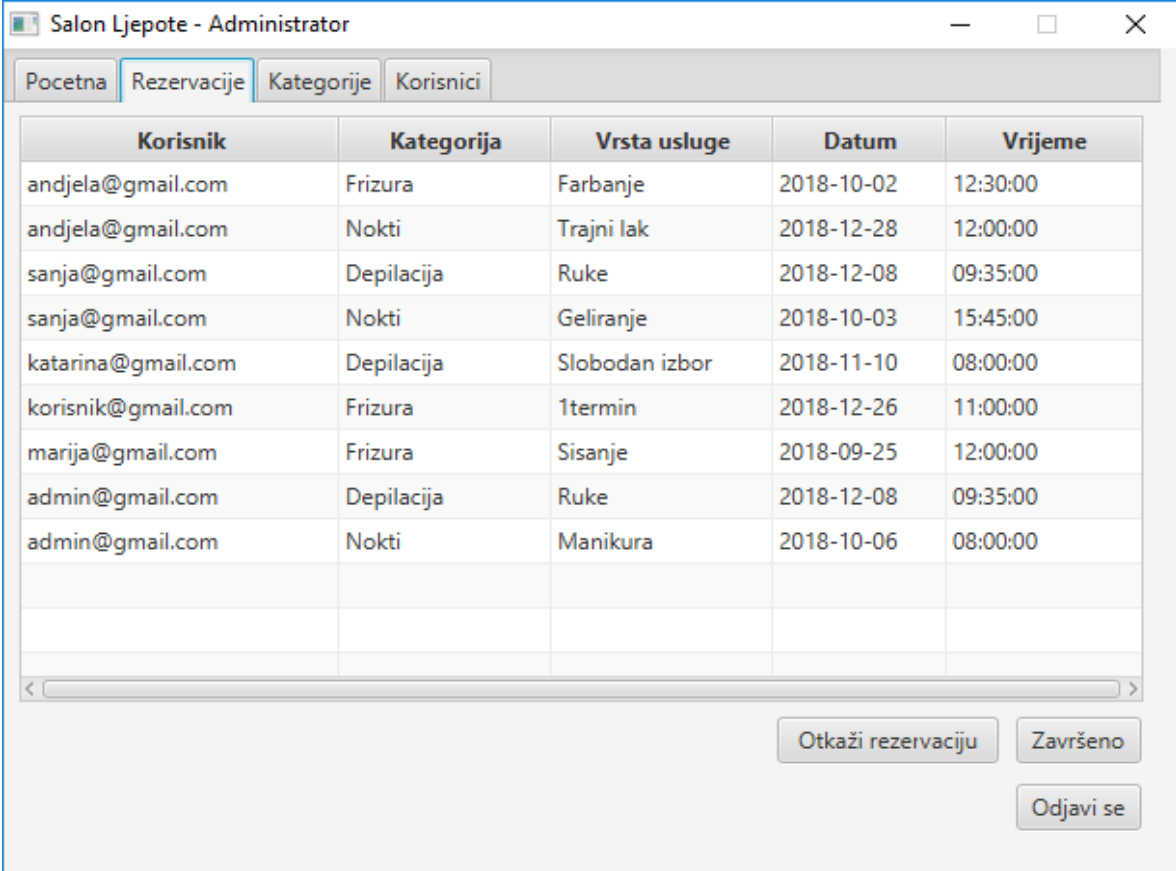
Slika 4.9 Informacijski prozor sa porukom o dodanom/uređenom/obrisanom terminu

U slučaju da neki od neregistriranih klijenata obavlja rezervaciju, bilo telefonskim putem ili osobnim dolaskom u salon, administrator ima mogućnost rezerviranja termina u njihovo ime klikom na *button* „Rezerviraj“.



Slika 4.10 Termin rezerviran

Sljedeći prozor koji administrator vidi je prozor „Rezervacije“ (Slika 4.11):



Salon Ljepote - Administrator

Pocetna Rezervacije Kategorije Korisnici

Korisnik	Kategorija	Vrsta usluge	Datum	Vrijeme
andjela@gmail.com	Frizura	Farbanje	2018-10-02	12:30:00
andjela@gmail.com	Nokti	Trajni lak	2018-12-28	12:00:00
sanja@gmail.com	Depilacija	Ruke	2018-12-08	09:35:00
sanja@gmail.com	Nokti	Geliranje	2018-10-03	15:45:00
katarina@gmail.com	Depilacija	Slobodan izbor	2018-11-10	08:00:00
korisnik@gmail.com	Frizura	1termin	2018-12-26	11:00:00
marija@gmail.com	Frizura	Sisanje	2018-09-25	12:00:00
admin@gmail.com	Depilacija	Ruke	2018-12-08	09:35:00
admin@gmail.com	Nokti	Manikura	2018-10-06	08:00:00

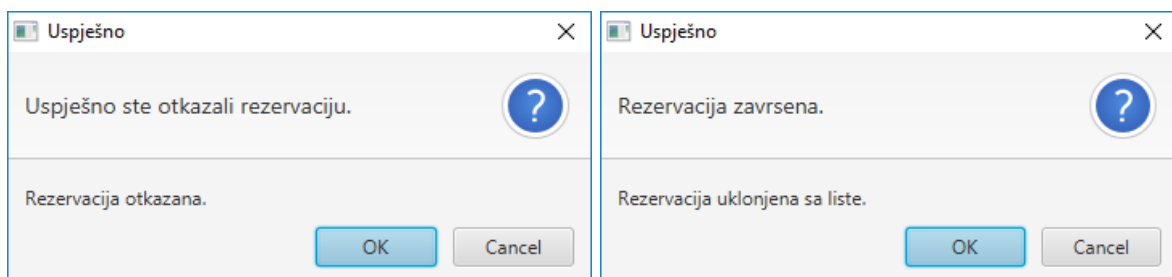
Otkazi rezervaciju Završeno

Odjavi se

Slika 4.11 Prozor „Rezervacije“

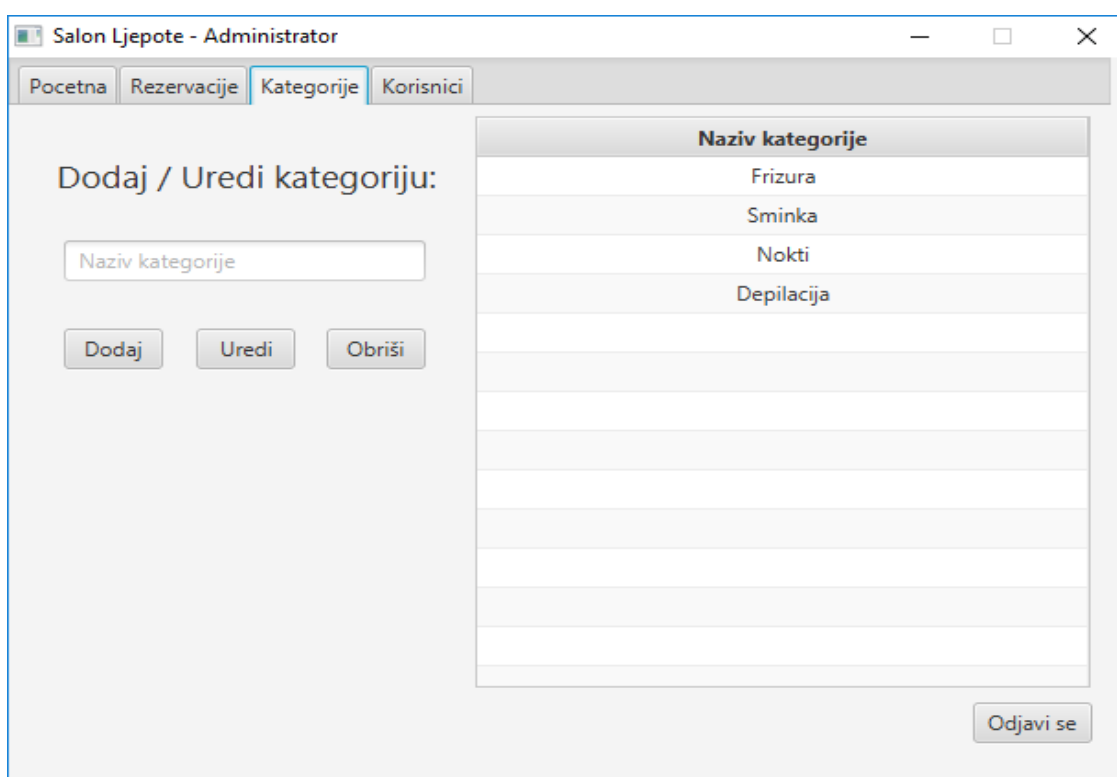
U ovom prozoru su prikazane rezervacije svih registriranih korisnika. U tablici se nalaze osnovni podaci o rezervaciji koje administrator mora znati. To su: korisnik (e-mail klijenta koji je izvršio rezervaciju), kategorija, vrsta usluge, datum i vrijeme.

Kao i običan korisnik i administrator ima mogućnost otkazivanja rezervacije koju je napravio klikom na *button* „Otkazi rezervaciju“. Isto tako, sve rezervacije koje su odrađene i završene može maknuti sa liste rezervacija pomoću *button*-a „Završeno“ (Prilog 6). Naravno, po izvršenju radnje prima obavijest u alert prozoru:



Slika 4.12 Informacijski prozori sa porukom o otkazanoj/završenoj rezervaciji

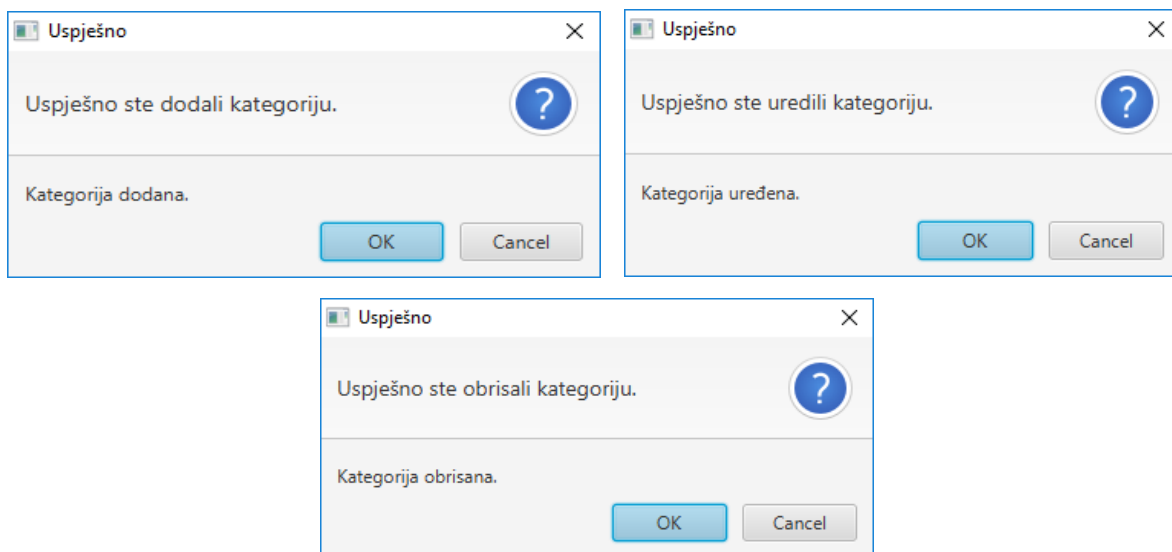
Kategorije su sljedeći prozor sa kojim se administrator susreće. Izgled prozora Kategorije:



Slika 4.13 Prozor „Kategorije“

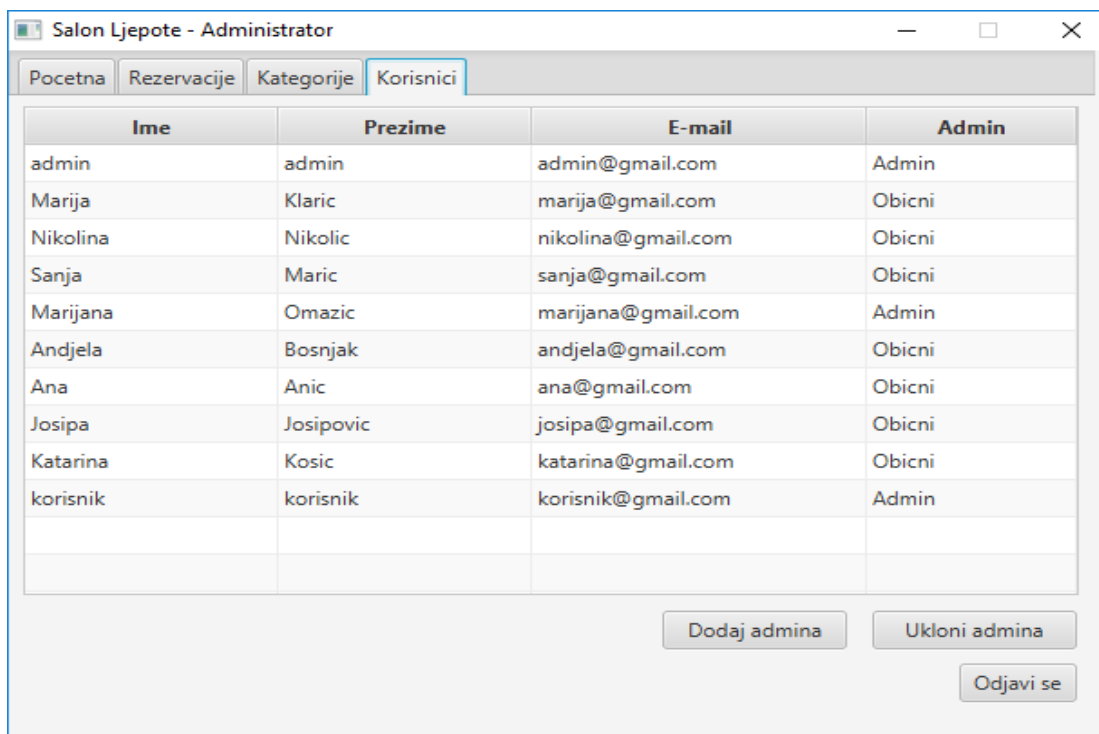
Uz prikaz tablice u kojoj su izlistane sve kategorije, administratoru je dozvoljeno i omogućeno dodavanje novih, te uređivanje i brisanje postojećih kategorija. Nakon izvršenja i potvrde neke od tih radnji promjene se spremaju i u bazi podataka.

Alert prozori sa obavijestima (porukom) o dodanoj/uređenoj/obrisanoj kategoriji (Slika 4.14) (Prilog 7) (Prilog 8):



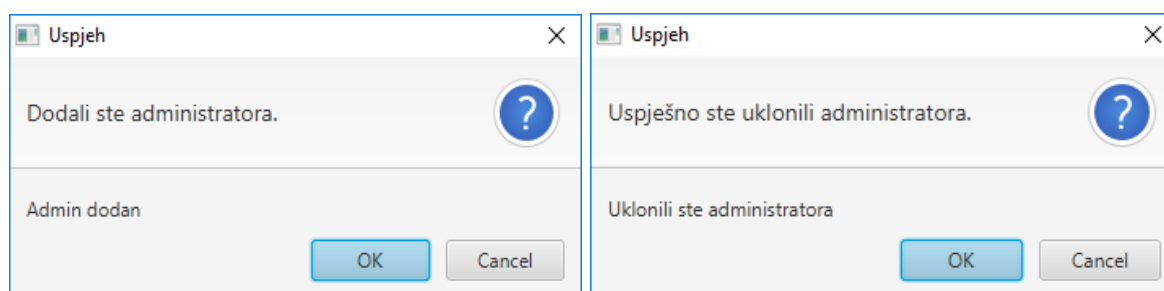
Slika 4.14 Informacijski prozor sa porukom o dodanoj/uređenoj/obrisanoj kategoriji

Posljednji prozor koji administrator vidi jest prozor „Korisnici“. Kao što vidimo na slici ispod, u tom prozoru imamo tablicu sa svim registriranim korisnicima. U tablici se nalaze osnovni korisnički podaci u koje administrator mora imati uvid. To su ime, prezime i e-mail.



Slika 4.15 Prozor „Korisnici“

Također, u tablici imamo i stupac „Admin“ koji daje informaciju o tome je li određeni korisnik administrator ili običan korisnik. Administrator ima mogućnost dodavanja novih administratora i uklanjanja postojećih pomoću *button*-a „Dodaj admina“ (Prilog 9) i „Ukloni admina“ (Prilog 10). Ukoliko izvrši neku od tih radnji napravljene promjene se spremaju u bazu podataka. Napomena: posljednjeg administratora nije moguće ukloniti. Alert poruka o dodavanju/uklanjanju administratora (Slika 4.16):



Slika 4.16 Informacijski prozor sa porukom o dodanom/uklonjenom administratoru

ZAKLJUČAK

Aplikacija „Programska podrška za administraciju i rezervaciju termina u salonu ljepote“ koja je napravljena u okviru ovog završnog rada primjer je jedne jednostavne Java desktop aplikacije, izrađene korištenjem JavaFX-a. Zaključujemo kako se korištenjem navedene biblioteke brzo i poprilično jednostavno mogu graditi funkcionalne aplikacije namijenjene korištenju na različitim računalnim platformama. Kako vrijeme prolazi upotreba desktop aplikacija je sve manja, a zamjenjuju ih web aplikacije koje preuzimaju „glavnu ulogu“. No, unatoč tome one se i dalje koriste npr. u velikim poduzećima kao što su Merkator za potrebe blagajni na stolnim računalima.

Iako su to aplikacije na stolnim računalima, one imaju i neke svoje prednosti u odnosu na web aplikacije. Nisu ovisne o internetu, mogu pristupiti podacima i u slučaju ako priključak na internet ne postoji.

Ukoliko bi se aplikacija dalje razvijala i nadograđivala, realizirala bi se tražilica, pregled termina po kategorijama, te prikaz korisničkog računa svakom od prijavljenih korisnika, kako bi mogao imati uvid u svoje osobne podatke, te mogao mijenjati iste po potrebi. Najvažnije od svega, bio bi razvoj web aplikacije, koja bi se povezala sa postojećom desktop aplikacijom. Tako bi aplikacija bila dostupna puno većem broju korisnika u svakom trenutku (naravno, ako postoji internet konekcija).

Prilikom odabira ove teme završnog rada i izrade aplikacije, cilj je bio proširiti dosadašnje znanje, te dodatno naučiti osnove objektno-orijentiranog programiranja, kao i usavršiti kreiranje MySQL baze podataka. Programski jezik „Java“ se pokazao kao odličan izbor za izradu aplikacije, jer osim mogućnosti korištenja na različitim platformama, nudi i niz biblioteka kojim se aplikacija može dodatno proširiti i poboljšati.

LITERATURA

- [1] Procesi karakteristični za inženjerstvo zahtjeva, preuzeto sa:
<https://www.scribd.com/document/370518438/T04-Softverski-zahtjevi>
- [2] UML-dijagrami: Zbirka primjera i riješenih zadataka, preuzeto sa:
<https://www.scribd.com/document/252441750/PrirucnikUML-dijagrami>
- [3] Saračević Muzafer: Objektno-orijentisano programiranje i modelovanje: Java i UML, Novi Pazar, 2011.
- [4] Čupić Marko, Programiranje u Javi, 2012.-2015.
- [5] JavaFX . FXML i Scene Builder, preuzeto sa:
<http://ns2.math.rs/~biljana/oop/cas12/FXML%20i%20SceneBuilder.pdf>
- [6] Izgradnja MVC modularnog radnog okvira, preuzeto sa:
<https://hrcak.srce.hr/file/190412>
- [7] MVC tehnologija, dostupno na: <http://docbook.rasip.fer.hr/ddb/res/35/Ch4.html>
- [8] Mujadžević Edin, Uvod u PHP i MySQL, Zagreb, 2007.
- [9] „XAMPP“, Apachedriends, dostupno na: www.apachefriends.org/index.html
- [10] Ekcel Bruce, Misliti na Javi, USA, 2006.
- [11] Schild Herbert, Java, A Beginner's Guide, Sixth Edition, USA, 2014.

POPIS SLIKA

Slika 2.1 Use-Case dijagram administratora.....	7
Slika 2.2 Use-Case dijagram običnog korisnika.....	8
Slika 2.3 Prikaz tablica u bazi podataka	8
Slika 4.1 Prijava korisnika	15
Slika 4.2 Registracija korisnika	16
Slika 4.3 Neuspješna prijava/registracija	16
Slika 4.4. Korisničko sučelje običnog korisnika.....	17
Slika 4.5 Informacijski prozori sa porukom o (ne)uspješnoj rezervaciji.....	17
Slika 4.6 Prozor „Moje rezervacije“	18
Slika 4.7 Otkazana rezervacija	18
Slika 4.8 Korisničko sučelje administratora.....	19
Slika 4.9 Informacijski prozor sa porukom o dodanom/uređenom/obrisanom terminu.....	20
Slika 4.10 Termin rezerviran.....	20
Slika 4.11 Prozor „Rezervacije“	21
Slika 4.12 Informacijski prozori sa porukom o otkazanoj/završenoj rezervaciji.....	22
Slika 4.13 Prozor „Kategorije“	22
Slika 4.14 Informacijski prozor sa porukom o dodanoj/uređenoj/obrisanoj kategoriji.....	23
Slika 4.15 Prozor „Korisnici“	23
Slika 4.16 Informacijski prozor sa porukom o dodanom/uklonjenom administratoru	24

PRILOZI

Prilog 1 (Prijava administratora)

```
if (PrijavljeniKorisnik.prijava(e_mail, lozinka)) {  
  
    if(PrijavljeniKorisnik.isAdmin(e_mail)){  
  
        try {  
  
            obavijest.setTextFill(Color.GREEN);  
  
            obavijest.setText("Uspješna prijava.");  
  
            Parent root;  
  
            root = FXMLLoader.load(getClass().getClassLoader().getResource  
                ("zavrsni/view/GlavniAdmin.fxml"));  
  
            stage stage = new Stage();  
  
            stage.setTitle("Salon Ljepote - Administrator");  
  
            stage.setScene(new Scene(root));  
  
            stage.resizableProperty().setValue(Boolean.FALSE);  
  
            stage.show();  
  
            obavijest.getScene().getWindow().hide();  
  
        } catch (IOException ex) {  
  
            Logger.getLogger(PrijavaController.class.getName()).  
                log(Level.SEVERE, null, ex);  
  
        }  
  
    }  
  
}
```

Prilog 2 (Registracija)

```
if(!PrijavljeniKorisnik.provjeriKorisnika(e_mail)){

    if(PrijavljeniKorisnik.registriraj(ime, prezime, e_mail, lozinka)){

        try{

            Parent root;

            root = FXMLLoader.load(getClass().getClassLoader().getResource

                ("zavrzni/view/Prijava.fxml"));

            stage stage = new Stage();

            stage.setTitle("Glavni ekran");

            stage.setScene(new Scene(root));

            stage.resizableProperty().setValue(Boolean.FALSE);

            stage.show();

            registracija_label.getScene().getWindow().hide();

            registracija_label.setTextFill(Color.GREEN);

            registracija_label.setText("Uspješna registracija.");

        } catch (IOException ex) {

            Logger.getLogger(RegistracijaController.class.getName()).

                log(Level.SEVERE, null, ex);

        }

    }else{

        registracija_label.setTextFill(Color.RED);

        registracija_label.setText("Greška.");

    }

}
```

Prilog 3 (Rezervacija termina)

```
public void rezervirajTermin(ActionEvent e){

    pocetnaOk.setEditable(true);

    int selected = pocetnaOk.getSelectionModel().getSelectedIndex();

    if(selected >= 0){

        if(Termini.rezervirajTermin(id2)){

            izlistajTermine();

            izlistajRezervacije();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Uspješno");

            alert.setHeaderText("Uspješno ste rezervirali termin.");

            alert.setContentText("Rezervacija završena.");

            alert.show();

        }else{

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("Greška");

            alert.setHeaderText("Ovaj termin ste već rezervirali.");

            alert.setContentText("Ovaj termin ste već rezervirali.");

            alert.show();

        }

    }

}
```

Prilog 4 (Otkazivanje rezervacije)

```
public void otkaziRezervaciju(ActionEvent e){

    rezervacije.setEditable(true);

    int selected = rezervacije.getSelectionModel().getSelectedIndex();

    if(selected >= 0){

        if(Rezervacije.otkaziRezervaciju(idRezervacije)){

            izlistajRezervacije();

            izlistajTermine();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Uspješno");

            alert.setHeaderText("Uspješno ste otkazali rezervaciju.");

            alert.setContentText("Rezervacija otkazana.");

            alert.show();

        }else{

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("Greška");

            alert.setHeaderText("Dogodila se greska pri otkazivanju rezervacije.");

            alert.setContentText("Dogodila se greska pri otkazivanju rezervacije.");

            alert.show();

        }

    }

}
```

Prilog 5 (Obriši termin)

```
public void obrisiTermin(ActionEvent e){

    pocetna.setEditable(true);

    int selected = pocetna.getSelectionModel().getSelectedIndex();

    if(selected >= 0){

        if(Termini.obrisiTermin(id2)){

            pocetna.getItems().remove(selected);

            izlistajTermine();

            izlistajRezervacije();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Uspješno");

            alert.setHeaderText("Uspješno obrisani termin.");

            alert.setContentText("Termin obrisani.");

            alert.show();

        }else{

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("Greška");

            alert.setHeaderText("Niste uspjeli obrisati termin.");

            alert.setContentText("Dogodila se greška prilikom

brisanja termina.");

            alert.show();

        }

    }

}
```

Prilog 6 (Završena rezervacija)

```
public void zavrsiRezervaciju(ActionEvent e){

    rezervacije.setEditable(true);

    int selected = rezervacije.getSelectionModel().getSelectedIndex();

    if(selected >= 0){

        if(Rezervacije.zavrsiRezervaciju(id1)){

            rezervacije.getItems().remove(selected);

            izlistajTermine();

            izlistajRezervacije();

            Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

            alert.setTitle("Uspješno");

            alert.setHeaderText("Rezervacija završena.");

            alert.setContentText("Rezervacija uklonjena sa liste.");

            alert.show();

        }else{

            Alert alert = new Alert(Alert.AlertType.ERROR);

            alert.setTitle("Greška");

            alert.setHeaderText("Dogodila se pogreška prilikom završetka rezervacije.");

            alert.setContentText("Niste uspjeli završiti rezervaciju.");

            alert.show();

        }

    }

}
```

Prilog 7 (Dodaj kategoriju)

```
public void dodajKategoriju(ActionEvent e){

    String nazivKategorije = naziv_kategorije.getText();

    if(Kategorija.dodajKategoriju(nazivKategorije)){

        izlistajKategorije();

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

        alert.setTitle("Uspješno");

        alert.setHeaderText("Uspješno ste dodali kategoriju.");

        alert.setContentText("Kategorija dodana.");

        alert.show();

    }else{

        Alert alert = new Alert(Alert.AlertType.ERROR);

        alert.setTitle("Greška");

        alert.setHeaderText("Niste uspjeli dodati kategoriju.");

        alert.setContentText("Dogodila se greška prilikom dodavanja kategorije.");

        alert.show();

    }

}
```

Prilog 8 (Uredi kategoriju)

```
public void urediKategoriju(ActionEvent e){

    String naziv = naziv_kategorije.getText();

    Kategorija k = new Kategorija(idKategorije, naziv, true);

    if(Kategorija.urediKategoriju(k)){

        izlistajKategorije();

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

        alert.setTitle("Uspješno");

        alert.setHeaderText("Uspješno ste uredili kategoriju.");

        alert.setContentText("Kategorija uređena.");

        alert.show();

    }else{

        Alert alert = new Alert(Alert.AlertType.ERROR);

        alert.setTitle("Greška");

        alert.setHeaderText("Niste uspjeli urediti kategoriju.");

        alert.setContentText("Dogodila se greška prilikom uređivanja kategorije.");

        alert.show();

    }

}
```

Prilog 9 (Dodaj administratora)

```
public void dodajAdmina (ActionEvent e){

    if (OstaliKorisnici.dodajAdmina(idKorisnika)){

        izlistajKorisnike();

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

        alert.setTitle("Uspjeh");

        alert.setHeaderText("Dodali ste administratora.");

        alert.setContentText("Admin dodan");

        alert.show();

    }else{

        Alert alert = new Alert(Alert.AlertType.ERROR);

        alert.setTitle("Greška");

        alert.setHeaderText("Korisnik je već administrator.");

        alert.setContentText("Administrator već postavljen.");

        alert.show();

    }

}
```

Prilog 10 (Ukloni administratora)

```
public void ukloniAdmina (ActionEvent e){

    if (OstaliKorisnici.ukloniAdmina(idKorisnika)){

        izlistajKorisnike();

        Alert alert = new Alert(Alert.AlertType.CONFIRMATION);

        alert.setTitle("Uspjeh");

        alert.setHeaderText("Uspješno ste uklonili administratora.");

        alert.setContentText("Uklonili ste administratora");

        alert.show();

    }else{

        Alert alert = new Alert(Alert.AlertType.ERROR);

        alert.setTitle("Greška");

        alert.setHeaderText("Korisnik nije administrator.");

        alert.setContentText("Korisnik nije administrator.");

        alert.show();

    }

}
```

Prilog 11 (Odjavi se)

```
public void odjava() throws IOException{

    try {

        Parent root;

        root = FXMLLoader.load(getClass().getClassLoader().getResource
            ("zavrsni/view/Prijava.fxml"));

        Stage stage = new Stage();

        stage.setTitle("Prijava se");

        stage.setScene(new Scene(root));

        stage.resizableProperty().setValue(Boolean.FALSE);

        stage.show();

        odjavaButton.getScene().getWindow().hide();

    }catch (IOException ex) {

        Logger.getLogger(PrijavaController.class.getName()).

            log(Level.SEVERE, null, ex);

    }

}
```