

UNIVERZITET ZA POSLOVNI INŽENJERING I MENADŽMENT BANJA LUKA
Dodiplomski studijski program Računarske nauke

Diplomski rad
PROJEKTOVANJE INFORMACIONOG SISTEMA UNIVERZITETA UPOTREBOM
JAVA PLATFORME – MODUL "STUDENTSKA SLUŽBA"

Mentor: doc. dr Saša Salapura

BANJA LUKA, JANUAR 2019.

IRENA MAJDANDŽIĆ

SAŽETAK

U sklopu ovog diplomskog rada opisan je proces izrade informacionog sistema univerziteta, čija bi implementacija trebala da omogući efikasnije funkcionisanje studentske službe. U radu su opisani osnovni principi korištenih tehnologija i data je kratka istorija razvoja. Za potrebe ovog rada napravljena je aplikacija Studentska služba koja je GUI (Graphical User Interface) aplikacija za vođenje evidencije o studentima, prvenstveno bazirana na prijavu ispita. Aplikacija omogućava dva nivoa pristupa, za studente i za studentsku službu. Aplikacija ima mogućnost pregleda, kreiranja, brisanja i mijenjanja podataka koji se nalaze u bazi. Predstavlja primjer jednostavne Java desktop aplikacije kreirane korištenjem Swing-a. Aplikacija je rađena u razvojnom okruženju NetbeansIDE uz Java development Kit. Pisana je Java programskim jezikom, baza aplikacije je SQL.

SUMMARY

The scope of this thesis is focused on the process of development of University Information Systems, whose implementation should enable more efficient functioning of the University Student Services. Within the thesis are described the basic principles the technologies used as well as a short history of development. An application was created for the purposes of this thesis entitled Student Service, which is a GUI (Graphical User Interface) developed for keeping the track records of students, primarily based on the student exam applications. This application allow for two levels of access, one for students and one for the university employees. The application has modules for viewing, creating, deleting and updating of the data stored in the database. It represent an example of simple Java desktop application created by using library Swing. The application is created in the development environment of the NetbeansIDE with Java development Kit. It was written in Java program language with the SQL as the base application.

SADRŽAJ

1.	UVOD.....	1
1.1.	Zadatak diplomskog rada.....	1
1.2.	Korištene tehnologije.....	2
2.	ANALIZA POSLOVNOG SISTEMA.....	3
3.	OSNOVE PROGRAMSKOG JEZIKA JAVA.....	4
3.1.	Razvoj Java programskog jezika.....	4
3.2.	Struktura Java programa.....	6
4.	BAZA PODATAKA.....	10
4.1.	SQL (Structured Query Language).....	11
4.1.1.	<i>Operatori u SQL-u</i>	12
4.2.	MySQL.....	13
4.2.1.	<i>Razvoj kompanije MySQL AB</i>	14
4.3.	Alati za izradu modela.....	14
4.3.1.	<i>MySQL Workbench</i>	15
4.4.	Kreiranje baze podataka.....	15
5.	RAZVOJ APLIKACIJE.....	17
5.1.	NetBeans razvojno okruženje.....	17
5.2.	Swing aplikacija.....	18
5.3.	Model-View-Controller arhitektura.....	19
6.	PRIKAZ REALIZOVANIH FUNKCIONALNOSTI IS-a.....	21
6.1.	Studentska služba.....	22
6.1.1.	<i>Novi predmet</i>	23
6.1.2.	<i>Novi student</i>	25
6.1.3.	<i>Prijave na ispit</i>	26
6.1.4.	<i>Unos rezultata ispita</i>	28
6.1.5.	<i>Izveštaj sa ispita</i>	30
6.2.	Student.....	32
6.2.1.	<i>Prijavljivanje ispita</i>	35
6.2.2.	<i>Izveštaj položenih ispita</i>	37
7.	ZAKLJUČAK.....	39
	LITERATURA.....	40

SADRŽAJ SLIKA

Slika 3.1.	Zavisnosti Java i C++ programskih jezika o operativnom sistemu.....	6
Slika 3.2.	Tipovi podataka u Java programskom jeziku.....	7
Slika 3.3.	Struktura klase "Student".....	8
Slika 3.4.	Struktura "Test" (main) klase.....	9
Slika 4.1.	Aritmetički operatori u SQL-u.....	12
Slika 4.2.	Operatori poređenja u SQL-u.....	12
Slika 4.3.	Logički operatori u SQL-u.....	13
Slika 4.4.	Baza podataka kreirana u DB Browseru.....	16
Slika 4.5.	ER dijagram izrađen u MySQL Workbench-u.....	16
Slika 5.1.	Model-View-Controller arhitektura.....	20
Slika 6.1.	Prvi prozor koji se pojavljuje pri pokretanju aplikacije.....	21
Slika 6.2.	Prozor "Opcije" sa svim funkcionalnostima studentske službe.....	22
Slika 6.3.	Prozor "Novi predmet".....	23
Slika 6.4.	Programski kod za dodavanje novog predmeta u bazu.....	23
Slika 6.5.	Prikaz tabele predmet u bazi podataka.....	24
Slika 6.6.	Prozor "Novi student".....	25
Slika 6.7.	Programski kod za upis novog studenta u bazu.....	25
Slika 6.8.	Prozor za izlistavanje prijave na određeni ispit.....	26
Slika 6.9.	Programski kod koji pokazuje kako da selektujemo podatke iz baze i upišemo ih u JTable.....	27
Slika 6.10.	Programski kod za izlistavanje prijave na ispit.....	27
Slika 6.11.	Prozor koji koristimo za unos rezultata ispita.....	28
Slika 6.12.	Programski kod pomoću kog selektujemo željene podatke iz baze i ispisujemo ih u naznačena polja.....	29
Slika 6.13.	Programski kod koji pokazuje kako da upisujemo podatke u bazu u željenu tabelu.....	29
Slika 6.14.	Prozor koji nam služi za pregled rezultata ispita.....	30
Slika 6.15.	Programski kod koji nam pokazuje kako da selektujemo određene elemente iz baze i upišemo ih u naznačena polja.....	31
Slika 6.16.	Programski kod koji pokazuje kako da selektujemo podatke iz baze podataka i upišemo u JTable.....	31
Slika 6.17.	Prozor za registraciju i prijavu studenta.....	32
Slika 6.18.	Programski kod za dugme "Prijavi se".....	32
Slika 6.19.	Prozor "Opcije".....	33
Slika 6.20.	Prozor za vraćanje zaboravljene šifre.....	34
Slika 6.21.	Programski kod za vraćanje zaboravljene šifre.....	34
Slika 6.22.	Prozor koji studentu služi za prijavu ispita.....	35

Slika 6.23. Programski kod koji nam pokazuje kako da upišemo podatke u bazu u željenu tabelu.....	36
Slika 6.24. Prozor pomoću kog student gleda izvještaj položenih predmeta i prosjek ocjena.....	37
Slika 6.25. Programski kod koji pokazuje računanje prosjeka ocjena i upis traženih rezultata u JTable.....	38

1. UVOD

Java je najpopularnija razvojna platforma koja ima široku primjenu u razvoju informacionih sistema. Koristi se za razvoj širokog spektra programskih rješenja, od desktop aplikacija, web aplikacija, do aplikacija za mobilne uređaje, pa čak i pametne kartice.

Programski jezik Java je objektno orjentisan te se temelji na principu da je klasa samostalna cjelina u programu, odnosno svaki Java program ima najmanje jednu klasu koja predstavlja neke tipove objekata ili obavlja određeni posao.

Jedan od glavnih razloga za uspjeh Jave je njena prilagodljivost. Java platforma dostupna je za većinu današnjih operativnih sistema (Windows, Unis, Linux, Mac) i potpuno je prenosiva između njih. Činjenica da je Java besplatan programski jezik, da se može izvršavati nezavisna od platforme, kao i to da je se u Java programskom jeziku mogu pisati mali programi (tzv. Applet) koji se ugrađuju u web stranice te im daju određene funkcionalnosti, doveli su do brze ekspanzije i današnje popularnosti ovog programskog jezika. Prema nekim statistikama u svijetu se nalazi oko 9 miliona Java programera, a ona se pokreće na oko 5 milijardi uređaja.

Ovaj diplomski rad se sastoji od šest poglavlja koja objašnjavaju osnove Java programskog jezika, njegovu strukturu, te razvoj Java aplikacije kao zadatka diplomskog rada.

1.1. Zadatak diplomskog rada

Zadatak ovog diplomskog rada je projektovanje informacionog sistema univerziteta upotrebom Java platforme. U sklopu ovog rada je urađena Java desktop aplikacija za efikasnije funkcionisanje Studentske službe.

Aplikacija Studentska služba je GUI (Graphical User Interface) aplikacija za vođenje evidencije o studentima, prvenstveno bazirana na prijavu ispita. Aplikacija ima mogućnost pregleda, kreiranja, brisanja i mijenjanja podataka koji se nalaze u bazi. Baza podataka se sastoji od 6 tabela sa jedinstvenim primarnim ključevima. Svaka tabela sadrži pripadajuće parametre zavisno o potrebama aplikacije.

Aplikacija omogućava dva nivoa pristupa, za studentsku službu i za studenta. Student da bi koristio aplikaciju, mora prvo da se registruje. Registracijom dobija korisničko ime i lozinku koji se čuvaju u bazi podataka, a koji su mu potrebni za rad sa aplikacijom. Studentskoj službi omogućava dodavanje novih studenata, novih predmeta, pregled prijava na ispit, unos rezultata ispita, evidenciju o studentima koji su položili određeni predmet, dok studentima omogućava prijavu ispita, pregled evidencije položenih predmeta, te računa prosjek na osnovu dobijenih ocjena iz položenih predmeta.

1.2. Korištene tehnologije

Aplikacija je pisana Java programskim jezikom. Rađena je u razvojnom okruženju NetBeansIDE uz Java development Kit. Baza aplikacije je SQL (Structured Query Language).

2. ANALIZA POSLOVNOG SISTEMA

Za izgradnju informacionog sistema neke poslovne organizacije potrebno je izvršiti adekvatnu analizu poslovnog sistema i na osnovu rezultata te analize napraviti model koji realizovan putem informacionih tehnologija čini informacioni sistem poslovne organizacije.

Jedna od početnih ideja zbog koje sam se odlučila na ovu temu diplomskog rada bila je izrada informacionog sistema koji bi omogućio studentima koji studiraju na daljinu online prijavu ispita.

Iako je u ovom radu opisana samo desktop aplikacija Studentske službe, u budućnosti je planirana i izrada web aplikacije i proširenje projekta koji je trenutno baziran na ispitu.

Za razradu ove ideje, bilo je potrebno uraditi određenu analizu zahtjeva. Prvo je bilo potrebno napraviti bazu podataka, koja bi sadržavala podatke o studentima, podatke o predmetima, podatke o ispitima, i napraviti veze između datih objekata kojima bi se dobila određena ograničenja, npr. ograničenje studentima na prijavljivanje njima dostupnih ispita.

Za razvoj uspješnog informacionog sistema potrebno je izvršiti i kategorizaciju korisnika koji imaju različita ovlaštenja. U ovom projektu izdvojene su dvije grupe korisnika, to su studenti i studentska služba. U okviru svake grupe nalaze se korisnici na koje se može primjeniti isti skup funkcionalnosti, privilegija i ograničenja.

U sklopu analize poslovnih procesa opisaćemo korake koje oni obavljaju. Studentska služba ima mogućnost upisa novog studenta u bazu podataka, ili brisanja postojećeg, i unos novog predmeta ili brisanje postojećeg. Nakon unosa novog studenta u bazu, na osnovu broja indeksa student može da napravi svoj korisnički nalog sa korisničkim imenom i lozinkom pomoću kojih će moći da se uloguje da bi prijavio ispit ili vidio svoju evidenciju položenih ispita i prosjek ocjena. Studentska služba takođe može da vidi evidenciju prijavljenih studenata na ispit iz određenog predmeta i spisak studenata koji su položili određeni predmet, ali i da unese rezultate ispita koji mu budu dostavljeni od strane profesora. Na ovaj način informacioni sistem nam nudi dodatnu bazu svih prijava ispita kao i ostvarenih rezultata (ocjena) studenata što znatno olakšava i automatizuje provjere. Da bismo postigli tačnost i automatizaciju informacionog sistema, potrebno je da informacioni sistem raspolaže tačnim informacijama vezanim za studente, predmete, ispitne rokove itd.

Uz izmjene i proširenja ovaj projekat se može usmjeriti u pravcu glavnog informacionog sistema Univerziteta sa interfejsom za online prijavu ispita.

3. OSNOVE PROGRAMSKOG JEZIKA JAVA

Java je jedan od najkorištenijih programskih jezika. Ima mnoštvo prednosti u odnosu na ostale programske jezike i okruženja te je odlučan alat za izradu mnogih projekata. Java je jezik opšte namjene i njime se mogu razvijati gotovo sve vrste aplikacija.

Najveća prednost, u odnosu na ostale programske jezike, je što se programi pisani u Java-i mogu izvoditi na svim operativnim sistemima koji posjeduju JVM (engl. Java Virtual Machine) bez ikakvih prilagođavanja. Java programski prevodilac, odnosno programski jezik koji zahtijeva da se izvorni kod prije izvođenja prevede. Java programi ne prevode se direktno u mašinski jezik za Java virtualnu mašinu, već u poseban međufORMAT poznat kao byte-code.¹

Unutar virtualne mašine postoji JIT (engl. Just In Time) prevodioc (eng. compiler) koji tokom izvršavanja programa prevodi Java bajt kod u procesorske instrukcije i sprema izvorni kod u privremenu memoriju. Takvo svojstvo omogućava da se jednom pisani program evaluiira jednako na svim podržanim platformama koje sadržavaju JVM.

Java virtualne mašine u današnje su vrijeme sastavni dio svih popularnijih operativnih sistema. Programi pisani u Javi se mogu smatrati potpuno platformski nezavisnima. Za razvijanje Java programa potrebno je također imati instaliranu Java razvojnu opremu čime se automatski kreira i Java razvojna okolina na računaru. Java platforma dijeli se na tri različite platforme sa specifičnim namjenama a to su Java standardno izdanje (engl. Java Standard Edition, Java SE) koje se koristi na kućnim računarima, Java mikro izdanje (engl. Java Micro Edition, Java JME) koje se koristi za mobilne uređaje, tablete i slične uređaje, te Java poslovno izdanje (engl. Java Enterprise Edition, Java EE) namijenjeno aplikacijama, serverima itd.

Postoje dvije osnovne vrste programa koji se mogu pisati u Javi: aplikacija i aplet. Aplikacija samostalni program za rješavanje jednog ili skupa problema, koji sadrži metodu main() i jednu ili više klasa. Aplet je dinamički interaktivni program koji se može izvršiti u okviru web stranice, a posredstvom nekoga od browsera.

3.1. Razvoj Java programskog jezika

Programski jezik Java osmišljen je u korporaciji Sun Microsystems početkom 90-tih godina. Razvoj jezika počeo je 1990. godine kao Stealth project (eng. nevidljivi, prikriveni projekt), poslije nazvan Green project (eng. zeleni projekt), a vodili su ga Bill Joy, Patrick Naughton, Mike Sheridan i James Gosling. U početku, Java je bila poznata

¹ Živković, Dejan. 2013. Java Programiranje, drugo izdanje. Beograd: Univerzitet Singidunum. str. 2. modifikovan tekst.

pod nazivom Oak (hrast), koji joj je dao Gosling prema starom hrastu koji je rastao u dvorištu ispred prozora njegove radne sobe. Od tog naziva kasnije se ipak odustalo zbog problema oko autorskih prava, nakon čega je usvojeno ime Java.² Jedna od osnovnih vodilja u razvoju ovog jezika, koja je dovela do opšte prihvaćenosti, je ideja napiši-jednom-pokreni-bilo gdje.³

Početna ideja projekta bila je povezivanje mnoštva elektroničkih naprava sa jednim centralnim uređajem, nalik daljinskom upravljaču, odnosno izgradnja sistema koji će omogućiti mrežno povezivanje i međusobnu komunikaciju velikog broja različitih elektroničkih uređaja. Odlučeno je da ni jedan programski jezik do tada, pa ni C++, ne zadovoljava potrebne uslove za razvoj takvog sistema, te se počelo raditi na novom jeziku. Zahtjevi koje je taj novi jezik trebalo da ispuni bili su jednostavnost, pouzdanost, sigurnost i prenosivost. Problem sa jezicima kao što su C i C++ (kao i većina drugih jezika), leži u tome što se pri prevođenju u binarni oblik oni moraju usmjeriti na određeni procesor. Iako se program pisan u jeziku C++ može prevesti za bilo koji procesor, za to je potreban potpun prevodilac namijenjen konkretnom procesoru. Problem je u tome što su prevodioci skupi, te se javila potreba za jednostavnijim i jeftinijim rješenjem. U pokušaju da dođu do njega Gosling i ostali počeli su da rade na prenosivom jeziku koji ne zavisi od platforme i čiji bi kod mogao da se izvršava na različitim procesorima i različitim okruženjima. Ovi naponi su konačno doveli do rađanja Jave.

Java je srodnik jezika C++, koji je direktan potomak jezika C. Veći dio svojih osobina Java je naslijedila od ova dva jezika. Iz jezika C java je naslijedila sintaksu. Mnoge objektno orijentisane osobine Jave nastale su pod uticajem jezika C++.⁴

Pojavom Interneta, u Sun-u su shvatili da je Java po svojim karakteristikama savršen izbor za izradu web aplikacija. Java je prilagođena novoj primjeni i službeno izdana 1995. godine. Omogućila je izradu dinamičkih web stranica i složenih web aplikacija, nezavisno o platformi na kojoj se izvodi, te je vrlo brzo postala sveprisutna na Internetu. Idući korak bio je ulazak na tržište mobilnih telefona što se također pokazalo uspješnim. Time je dokazano da se Java može koristiti u svim područjima tehnologije te je svojim uspješnim projektima postala sastavni dio današnjeg Interneta.

Za Javu danas možemo reći da je:

-specifikacija programskog jezika i standardni zbir klasa;

² Schildt, Herbert. 2012. Java JDK 7, prevod osmog izdanja knjige "Java: The Complete Reference". Beograd: Mikro knjiga. str. 6. modifikovan tekst.

³ Čupić, Marko. 2015. Programiranje u Javi. Zagreb: Fakultet elektrotehnike i računarstva. str.1. modifikovan tekst.

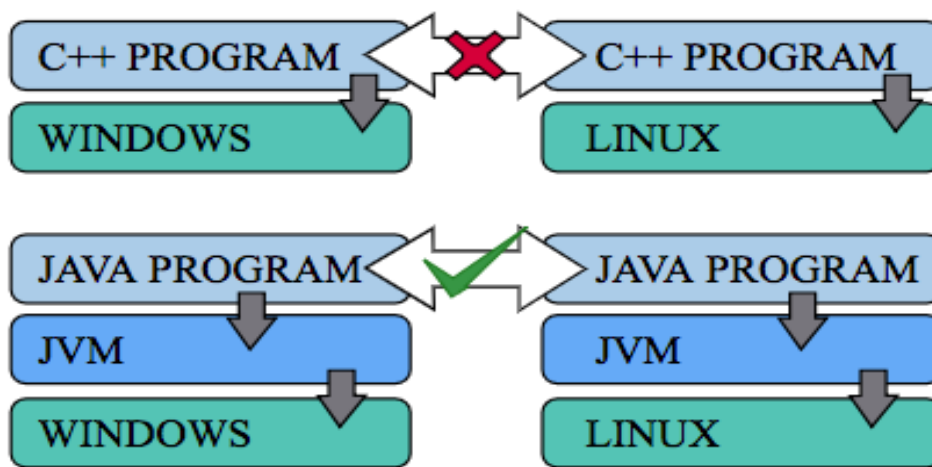
⁴ Schildt, Herbert. 2012. Java JDK 7, prevod osmog izdanja knjige "Java: The Complete Reference". Beograd: Mikro knjiga. str. 6. modifikovan tekst.

-implementacija navedenog programskog jezika i njegovih pratećih datoteka u okolini prevođenja i izvođenja (compile and run time environment) zaizradu i izvršavanje aplikacija;

-implementacija navedenog programskog jezika kao podskup ugrađenog koda u HTML stranicama (applet);

-implementacija navedenog programskog jezika kao dodatak animaciji i interakciji kod 3D objekata i scena.

Slika 3.1. Poređenje zavisnosti Java i C++ programskih jezika o operativnom sistemu



3.2 Struktura Java programa

Programski jezik Java je objektno orijentiran te se temelji na principu da je klasa samostalna cjelina u programu, odnosno svaki Java program ima najmanje jednu klasu koja predstavlja neke tipove objekata ili obavlja određeni posao.

Osnovni koncept u Javi je klasa (eng. class), tj. sav izvorni kod je napisan unutar klasa. U pravilu, svaka klasa je deklarirana unutar datoteke sa istim imenom i sufiksom .java. Ovo pravilo je donekle nametnuto kompajlerom koji zahtijeva da su imena klasa i datoteka ista (osim tzv. unutrašnjih klasa, eng. inner classes).⁵

Klasa je osnovna programska cjelina u Javi, tj. programski kod Java programa pisan je unutar klasa koje mogu imati metode (funkcije) i svojstva (atribute). Metode su radnje koje klasa obavlja, a atributi predstavljaju karakteristike navedene klase. Klasa je nacrt po kojem se objekti kreiraju a njome su definirane varijable i metode zajedničke za

⁵ Schildt, Herbert. 2012. Java JDK 7, prevod osmog izdanja knjige "Java: The Complete Reference". Beograd: Mikro knjiga. str. 106. modifikovan tekst.

objekte određenog tipa. Kreiranje objekta neke klase naziva se instanciranje objekta, a iste nazivamo instancama određene klase.

Za pokretanje programa pisanog u Javi, najprije se traži main metoda te se ona izvršava. Klasa Java programa koja sadrži main metodu naziva se izvršna klasa. Uobičajeno je da postoji samo jedna klasa s ovom metodom, a ostale se klase po potrebi pozivaju iz navedene glavne metode.

Metode su zapravo funkcije i tako se i definišu. Tip vrijednosti koje metod vraća može biti void ili proizvoljan tip (int, string...).

Postoji više načina pristupa metodama i varijablama. Imamo četiri vrste pristupnih modifikatora u Javi koji specifikuju dostupnost podataka u metodi, konstruktoru ili klasi a to su default, private, protected i public. Defaultni pristup omogućava pristupanje varijabli iz svih klasa unutar paketa. Ukoliko se varijabla unutar neke klase deklarira kao private, pristup toj varijabli omogućen je unutar pripadajuće klase. Korištenjem modifikatora protected pristup varijabli i/ili metodi dozvoljen je iz svih klasa u paketu i svih nasljednih klasa. Varijable i metode deklarirane modifikatorom public mogu se koristiti bez ograničenja, odnosno pristup istima dozvoljen je iz svih klasa koje instanciraju dani objekt. Uz spomenute glavne modifikatore mogu se još koristiti modifikatori final i static. Final modifikator koristi se za definiranje konstanti, odnosno za čuvanje vrijednosti koje se nikada ne mijenjaju, dok se modifikator static koristi za definiranje varijable koja je zajednička svim objektima koji instanciraju danu klasu.

U Java programskom jeziku razlikuju se dvije kategorije tipova podataka, a to su primitivni tipovi podataka (jednostavni) i reference (složeni).⁶

Slika 3.2. Tipovi podataka u Java programskom jeziku

PRIMITIVNI TIPOVI PODATAKA	REFERENCE
Cjelobrojni: byte, short, int, long	Nizovi (array)
Brojevi sa pokretnim zarezom: float, double	Klase(class)
Znakovni: char	Interfejsi (Interface)
Logički: boolean	

⁶ Čupić, Marko. 2015. Programiranje u Javi. Zagreb: Fakultet elektrotehnike i računarstva. str. 7. modifikovan tekst

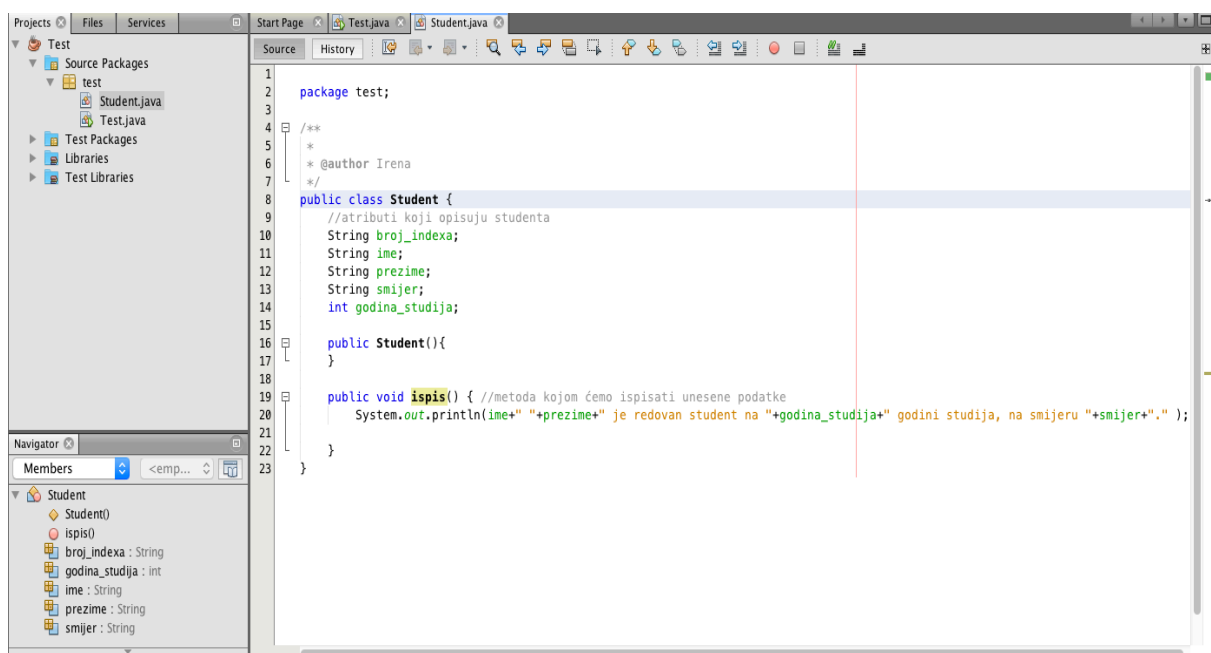
Java raspolaže primitivnim tipovima koji su na isti način definisani i u drugim programskim jezicima. Jedini izuzetak je tip char, koji zauzima dva bajta, umjesto uobičajenog jednog bajta. Radi se tome da se tipom char može predstaviti svaki karakter definisan Unicode standardom koji definiše kodni raspored koji obuhvata praktično sve današnje jezike. To znači da su Java programi osposobljeni da rade sa višejezičnim tekstom, ili u našim uslovima, ravnopravno sa srpskom ćirilicom i srpskom latinicom. Osnovni tipovi u Javi, poput int, boolean, byte i sl. su takodje predstavljeni u klasama i imaju odgovarajuće klasne omote poput java.lang.Integer, java.lang.Boolean i java.lang.Byte.

Referentni tipovi podataka sadrže referencu, ne mjesto u memoriji gdje je smješten podatak. Oni sadrže adresu memorijske lokacije na kojoj se objekat nalazi, odnosno ne sadrže pravu vrijednost. Niz je objekat koji sadrži više elemenata istog tipa podataka. Prvi element niza u Javi ima indeks 0, odnosno nizovi u Javi se prebrojavaju od nule. Klasa je nacrt po kojem se objekti kreiraju. Interfejs kao tip podatka predstavlja protokol ponašanja koji klasa može implementirati.

Iako je Java nastala od C++, ali za razliku od C++, koji dozvoljava strukturne kao i objektno-orjentisane principe, u Javi su objektno-orjentisani principi obavezni. Sve je u Javi objekat, a sav izvorni kod je pisan unutar klasa.

Na primjeru u nastavku je prikazana osnovna struktura jednostavnog Java programa koja je pisana u Netbeans razvojnom okruženju. U ovom programu korišten je samo jedan paket (eng. Package) pod nazivom "test", koji sadrži dvije klase.

Slika 3.3. Struktura klase "student"

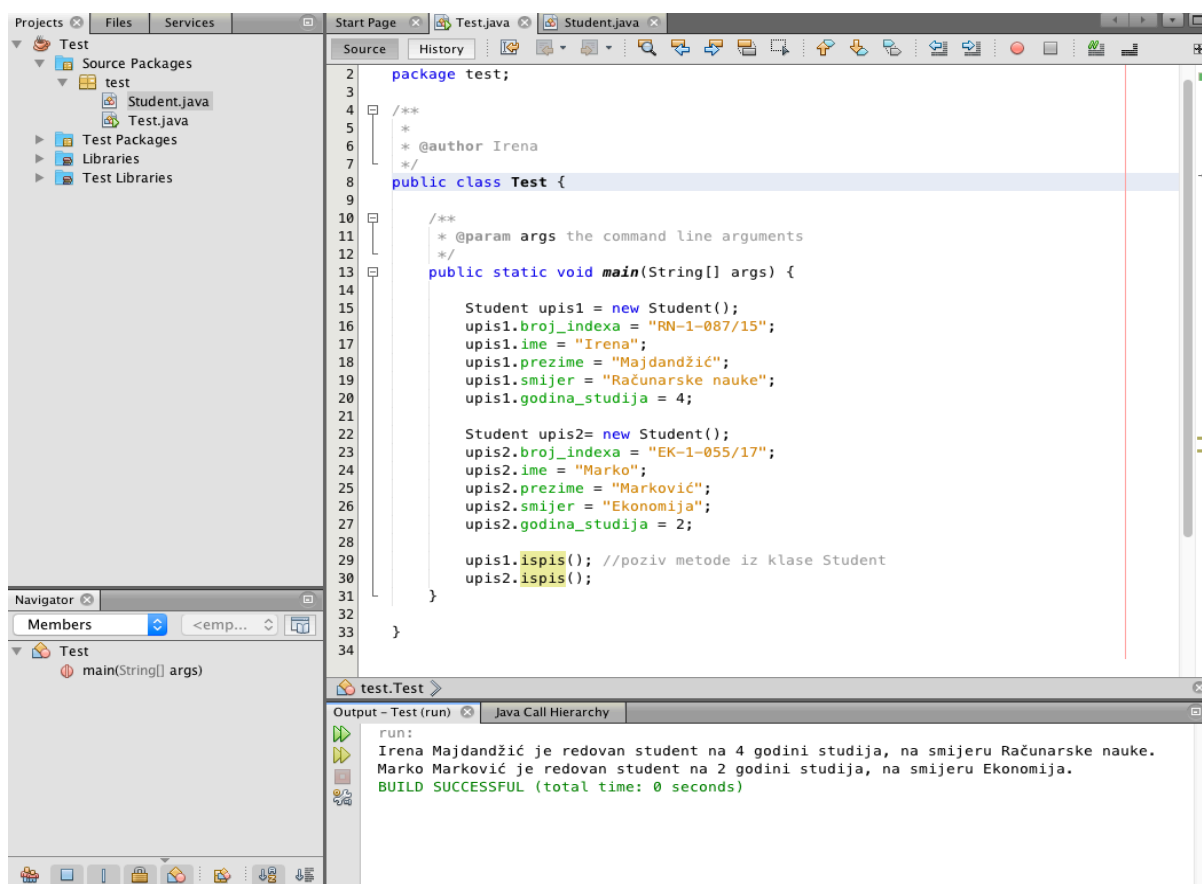


```
1 package test;
2
3
4 /**
5  *
6  * @author Irena
7  */
8 public class Student {
9     //atributi koji opisuju studenta
10    String broj_indexa;
11    String ime;
12    String prezime;
13    String smijer;
14    int godina_studija;
15
16    public Student(){
17    }
18
19    public void ispis() { //metoda kojom ćemo ispisati unesene podatke
20        System.out.println(ime+" "+prezime+" je redovan student na "+godina_studija+" godini studija, na smijeru "+smijer+"." );
21    }
22 }
23
```

Na slici je prikazana struktura klase Student. Sastoji se od nekoliko atributa koji opisuju studenta, konstruktora koji ima isti naziv kao i klasa, koji je javan (public) i omogućava nam kreiranje objekata, i metode kojom ćemo ispisati podatke kreiranog objekta, što nam omogućava izraz System.out.println.

U test (main) klasi nalazi se main metoda koja omogućava pokretanje napisanog Java programa. Programski kod pisan unutar main metode se izvršava liniju po liniju i ispisuje se u izlaznom dijelu korištenog razvojnog okruženja. U ovoj klasi prvo smo kreirali instance klase Student, odnosno nove objekte upis1 i upis2 kojima smo dodjelili željene vrijednosti unaprijed definisanih atributa. Zatim se izvršavaju metode koje smo pozvali, a koje su pisane u klasi Student. Rezultat tih metoda je vidljiv u izlaznom dijelu.

Slika 3.4. Struktura "test" (main) klase



4. BAZA PODATAKA

Baze podataka su postale sastavni dio modernog društva. Većina nas se svaki dan susreće s nekoliko aktivnosti koje uključuju neku interakciju s bazama podataka. Na primjer, svaki put kad odlazimo u banku podići novac, kada rezervišemo avionsku kartu, kupujemo nešto preko Interneta, velike su šanse da aplikacija na računaru kojem se pristupa za obradu pohranjuje i ažurira podatke u neku vrstu baze podataka. Ovo su većinom primjeri takozvanih tradicionalnih baza podataka koje spremaju informacije u tekstualni ili brojni tip podataka. Međutim razvoj sistema za upravljanje baza podataka nam danas omogućuje i pohranu slika, video i audio isječaka u digitalnom formatu, koji su važan dio složenih multimedijских baza podataka koje se koriste u pretraživačima Interneta, geografskim informacionim sistemima, skladištima podataka i sistemima za analitičko odlučivanje.

Dve ključne komponente u izgradnji baze podataka informacionog sistema su:

- Sistem za upravljanje bazama podataka (SUBP)
- Alati za izradu baze podataka

Sistemi za upravljanje bazama podataka (DBMS – DataBase Management System) predstavljaju aktivne komponente čije korištenje traje tokom cijelog životnog ciklusa informacionog sistema. SUBP predstavlja programski sistem koji se koristi za pristup, skladištenje, te manipulaciju podacima u bazi podataka. Koristi se za interakciju korisnika i baze podataka, a korisnik može sa SUBP imati direktnu interakciju, ili preko aplikacije programirane u nekim od programskih jezika poput Java, C++, Visual Basic-a... Sistemi za upravljanje bazama podataka mogu koristiti različitu organizaciju podataka (objektno-orijentisanu, hijerarhijsku...) ali je u upotrebi najčešće relacioni model koji je iskorišten i za potrebe ovog informacionog sistema.

Neophodni dijelovi svakog SUBP su jezici za upravljanje bazama podataka⁷:

- DDL (data description language) - jezik za definiciju ili deklaraciju objekata u bazi podataka;
- DML (data manipulation language) - jezik za manipulaciju objektima baze podataka;
- DCL (data control language) - jezik za postavljanje dozvola u bazi podataka;

⁷ Veinović, Mladen i Goran Šimić. 2010. Uvod u baze podataka. Beograd: Univerzitet Singidunum. str.120. modifikovan tekst.

4.1. SQL (Structured Query Language)

Za upravljanje bazom podataka koriste se adekvatni jezici, a najčešće SQL (Structured Query Language). SQL je najpoznatiji računalni jezik korišten pri izradi baza podataka, te manipulacijom podataka u bazi podataka. Napravljen je po uzoru na relacijski model Edgar F. Codd. Razvila ga je američka kompanija IBM, gdje je razvijena baza podataka pod nazivom „System R”, a zadatak tada zvanog Structured English Query Language ("SEQUEL") je bio upravljanje tom bazom podataka. Skraćenica SEQUEL je kasnije promijenjena u SQL. SQL je standardiziran preko standarda ANSI1 1986. godine i ISO2 1987. godine, te je postao referentni jezik za relacijske baze podataka. Glavna karakteristika mu je deklarativnost što znači da korisnik određuje što SQL treba napraviti, ali ne i kako doći do rezultata, te nema potrebe poznavati složene aktivnosti koje se događaju kad se unese SQL naredba. Iako je SQL standardiziran danas postoje male razlike u različitim SUBP koje možemo nazvati i „narječja” u SQL jeziku.

Osnovni tipovi podataka koje podržavaju svi SUBP su sljedeći⁸:

- NUMBER - brojevni tip podatka (cijeli, negativni, decimalni);
- INTEGER - cijeli brojevi;
- CHAR - niz podataka fiksne duljine;
- VARCHAR - niz podataka varijabilne dužine;
- DATE/TIME – tip podatka koji označava datum/vrijeme.

Naredbe u SQL-u se mogu podijeliti u četiri kategorije:

- DDL (data definition language) naredbe;
- DML (data manipulation language) naredbe;
- DCL (data control language) naredbe;
- TCL (transaction control language) naredbe.

DDL naredbe se koriste kod kreiranja i brisanja tablica, indeksa i pogleda. Osnovne DDL naredbe koje se koriste u SQL-u su CREATE TABLE, CREATE INDEX, CREATE VIEW, ALTER TABLE, DROP TABLE, DROP VIEW i DROP INDEX.

DML naredbe se koriste za dodavanje redova, izmjenu i brisanje podataka u tablicama, te se najviše koriste u radu s bazama podataka. U DML naredbe spadaju SELECT, INSERT, UPDATE i DELETE.

⁸ Veinović, Mladen i Goran Šimić. 2010. Uvod u baze podataka. Beograd: Univerzitet Singidunum. str.122. modifikovan tekst.

DCL naredbe se koristi kod dodjeljivanja dozvola za određene operacije nad bazom podataka. U njih spadaju GRANT koja dodjeljuje pravo, te REVOKE koja oduzima pravo nad određenom operacijom nad bazom podataka.

TCL naredbe se koriste kod upravljanja transakcijama u SQL-u. Transakcija predstavlja skup (obično DML) naredbi koje se izvršavaju u bazi podataka. U TCL naredbe spada COMMIT naredba kojom se spremaju promjene u bazi podataka i ROLLBACK kojom se ukidaju sve promjene od zadnje COMMIT naredbe.

4.1.1. Operatori u SQL-u

Aritmetički operatori se koriste u aritmetičkim operacijama u upitima.

Slika 4.1. Aritmetički operatori

<i>Operator</i>	<i>Svrha</i>	<i>Primer</i>
+, -	Definiše predznak broja (unarna operacija)	<code>select * from cena where min_cena < -10;</code>
+, -	Sabiranje i oduzimanje brojeva (binarni operator)	<code>select iznos+1000 from prodaja;</code>
*, /	Množenje i deljenje brojeva (binarni operator)	<code>select iznos/25.5 from prodaja;</code>
MOD,%	Moduo - daje ostatak pri deljenju (binarni operator)	<code>select mod(iznos,100) from prodaja;</code>

Izvor: <http://www.eknfak.ni.ac.rs/dl/informatika/SQL%20deo%202.pdf>

Operatori poređenja se koriste kod upoređivanja uslova u upitima.

Slika 4.2. Operatori poređenja

<i>Operator</i>	<i>Svrha</i>	<i>Primer</i>
=	poređenje jednakosti dve strane	<code>select * from radnik where prezime='JAMES';</code>
!=, <>	Operator nejednakosti	<code>select ime from kupac where grad<>'NEW YORK';</code>
<, >	Operator poređenja veće, manje	<code>select ime, kredit limit from kupac where kredit limit > 5000;</code>
<=, >=	Operatori: mane ili jednako i veće ili jednako	<code>select ime, kredit limit from kupac where kredit limit >= 5000;</code>
IN	Ispituje se jednakost sa svakim članom koji se testira u uslovu	<code>select ime, grad from kupac where drzava in ('NY', 'TX');</code>
NOT IN	Suprotno od IN operatora	<code>select ime, grad from kupac where drzava not in ('NY', 'TX');</code>
(NOT) BETWEEN - AND	Ispituje vrednosti koje se (ne) nalaze unutar traženog intervala	<code>select ime from kupac where kredit limit between 5000 and 10000;</code>
(NOT) LIKE	Upoređuje sličnost sa ispitivanim vrednostima tako da one odgovaraju nizu znakova koji je definisan zajedno sa znakom %.	<code>select * from radnik where prezime like 'A%';</code>

Izvor: <http://www.eknfak.ni.ac.rs/dl/informatika/SQL%20deo%202.pdf>

Logički operatori uspoređuju dva uslova istovremeno kako bi se utvrdilo da li red može biti dohvaćen.

Slika 4.3. Logički operatori

Operator	OPIS	Primer
NOT	Negacija. Vraća DA ako je uslov NE	<code>select * from artikal where not (kolicina < 250);</code>
AND	Logičko I. Vraća DA ako su svi uslovi koji se ispituju tačni	<code>select * from artikal where aktuelna_cena>9 and kolicina>250;</code>
OR	Logičko ILI. Vraća DA ako je bar jedan od uslova koji se ispituju tačan.	<code>select * from artikal where aktuelna_cena>50 or kolicina<400;</code>

Izvor: <http://www.eknfak.ni.ac.rs/dl/informatika/SQL%20deo%202.pdf>

Specijalni operatori se koriste zajedno s WHERE uslovom, u njih spadaju sljedeći operatori:

- BETWEEN – koristi se da bi se utvrdilo da li je vrijednost atributa unutar zadanih vrijednosti;
- IS NULL – koristi se da se utvrdi da li atribut ima vrijednost null;
- LIKE – koristi se da se utvrdi da li atribut ima isti uzorak znakova;
- IN – koristi se da se provjeri da li je atribut jednak nekoj od vrijednosti;
- EXISTS – koristi se da bi se provjerilo da li podupit vraća vrijednost;

Matematičke funkcije u SQL-u:

- COUNT – zbroj redova koji sadrže određenu vrijednost
- MIN – najmanja vrijednost od izabranog skupa atributa
- MAX – najveća vrijednost od izabranog skupa atributa
- SUM – zbroj svih vrijednosti za odabranu kolonu
- AVG – srednja vrijednost od odabrane kolone

4.2 MySQL

Jedan od najraširenijih sistema za upravljanje relacijskim bazama podataka koji se danas koristi je MySQL, sistem otvorenog koda koji se pokreće na serveru, te pruža pristup višestrukome broju korisnika i pohranu višestrukog broja baza podataka. MySQL

je odabran i za potrebe ovog informacionog sistema. Baza je takođe urađena i u DB Browseru koji koristi SQL.

MySQL je sistem za upravljanje SQL bazama podataka. MySQL je open-source proizvod, što znači da je dozvoljeno svakome da ga koristi i prilagođava za svoje potrebe. Svakome je omogućen download MySQL sa Interneta i korišćenje bez obaveze plaćanja. Takođe postoji MySQL Enterprise verzija programa koja nije besplatna i koja nudi dodatne opcije naprednim korisnicima.

MySQL je originalno razvijan za manipulaciju vrlo velikih baza podataka, mnogo je brži od postojećih rešenja i uspešno se koristi u visoko zahtevnim okruženjima. Pristupačnost, brzina i sigurnost čine MySQL vrlo pogodnim za pristupanje bazama podataka preko Interneta.

MySQL softver se sastoji od MySQL server-a, nekoliko dodatnih utility programa koji služe za olakšavanje administracije BP, takođe dolaze i dodatni programi koji su potrebni za rad servera.

Neke od osnovnih prednosti MySQL-a su brzina, pouzdanost, mala zahtevnost za sistemskim resursima, fleksibilno poboljšavanje performansi, rad na različitim platformama, besplatno ili povoljno licenciranje itd.

U navedenim sistemima se zapisi čuvaju u tablicama, a svaki zapis predstavlja se jednim redom tabele koji se sastoji od jedne ili više kolona. Kod ovakve vrste baze podataka, organizacija podataka zasniva se na racionalnom modelu, odnosno podaci se organiziraju u skupove između kojih se definišu određene veze. Svaka relacija/tabela mora imati definisan primarni ključ, koji određuje jedinstvenost svake tabele. Osim primarnog ključa, tabela u bazi može imati i vanjski ključ koji omogućava povezivanje tabela.

4.2.1. Razvoj kompanije MySQL AB

MySQL je izradila švedska kompanija MySQL AB 1995. godine, koja je osnovana iste godine. Osnivači su joj Michael Widenius, David Axmark i Allan Larsson. Od 2000. godine MySQL postaje software otvorenog koda, nakon čega već sljedeće godine broj aktivnih instalacija raste na 2 miliona. Kompanija se širi i na američko tržište, zapošljava oko 320 ljudi, te 2006. godine postiže 33% ukupnog svjetskog tržišta aktivnih instalacija i 0,2% ukupnih prihoda u tržištu baza podataka, a za njih se počinje zanimati Oracle. Međutim, kompanija je prodana Sun Microsystems-u za 1 bilijun dolara 2008. godine. Dvije godine kasnije Oracle kupuje većinski paket dionica Sun Microsystems-a te postaje jedna od vodećih svjetskih informatičkih kompanija. Danas je MySQL jedan od najpopularnijih SUBP s preko 100 milijuna aktivnih instalacija.

4.3 Alati za izradu modela

Alati za izradu modela baze podataka predstavljaju opcionu komponentu (izrada modela je moguća i putem ručnog unošenja komandi) ali je njihovo korištenje poželjno jer smanjuje mogućnost greške i vrijeme potrebno za izradu modela. Ova komponenta se koristi u fazi razvoja informacionog sistema i povremeno za potrebe izmene strukture baze podataka.

Za izradu ovog informacionog sistema koristili smo MySQL Workbench.

4.3.1. MySQL Workbench

MySQL Workbench je grafički alat za dizajniranje baza podataka koji integrira SQL razvoj, administraciju, dizajn i održavanje u jedan zajednički interfejs za MySQL baze podataka.

On nudi velik broj funkcija i sveukupna kvaliteta programa je vrlo dobra. Na početnom zaslonu ima opcije za izradu više dijagrama unutar pojedinog modela, jednostavan pristup svim tabelama, korisnicima i drugom.

Alat koristi standardnu grafiku, pravila i notacije za crtanje dijagrama baza podataka. MySQL Workbench entitet gleda i tretira kao tabelu, a dijagram entiteta i veza (Entity Relationship Diagram – ERD) gleda kao poboljšani dijagram entiteta i veza (Enhanced Entity Relationship Diagram – EER). Dodavanje entiteta u MySQL Workbench model vrlo je jednostavno. Treba pokrenuti MySQL Workbench, kreirati novi EER dijagram, dodati entitete i definirati im attribute. Elementi dijagrama mogu se uređivati tako da se dvaput klikne na njih.

Mana ovog alata je da omogućava samo dva tipa veze, 1:mного i 1:1. Zanimljivo je da pokušaj kreiranja veze mnogo:mного (n:m) MW odmah prepozna kao agregaciju, te automatski stvara novu tablicu, i dvije 1:mного veze.

MySQL Workbench je daleko najbolji alat za modeliranje baza podataka koji trenutno postoji. Može zadovoljiti potrebe čak i najzahtjevnijih developera i dizajnera baza podataka, nudeći odlične grafičke i tehnološke alate.

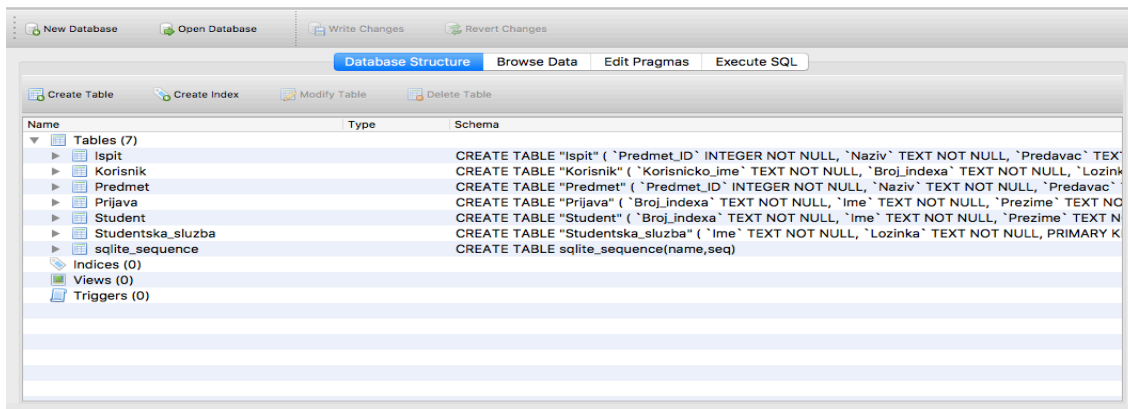
Otkad je alat predstavljen na tržištu, MySQL Workbench postao je vrlo popularan. Trenutno je drugi po redu preuzetih alata s MySQL web-stranice, s više od 250 000 preuzimanja mjesečno. Prva verzija je bila namijenjena za upotrebu na MS Windows operativnom sistemu, a od verzija 5.1 dostupna je i na drugim platformama.

4.4. Kreiranje baze podataka

Baza podataka korištena u izgradnji ovog informacionog sistema sastoji se od šest tabela, svaka s jedinstvenim primarnim ključem te pripadajućim redovima i kolonama. Za izradu

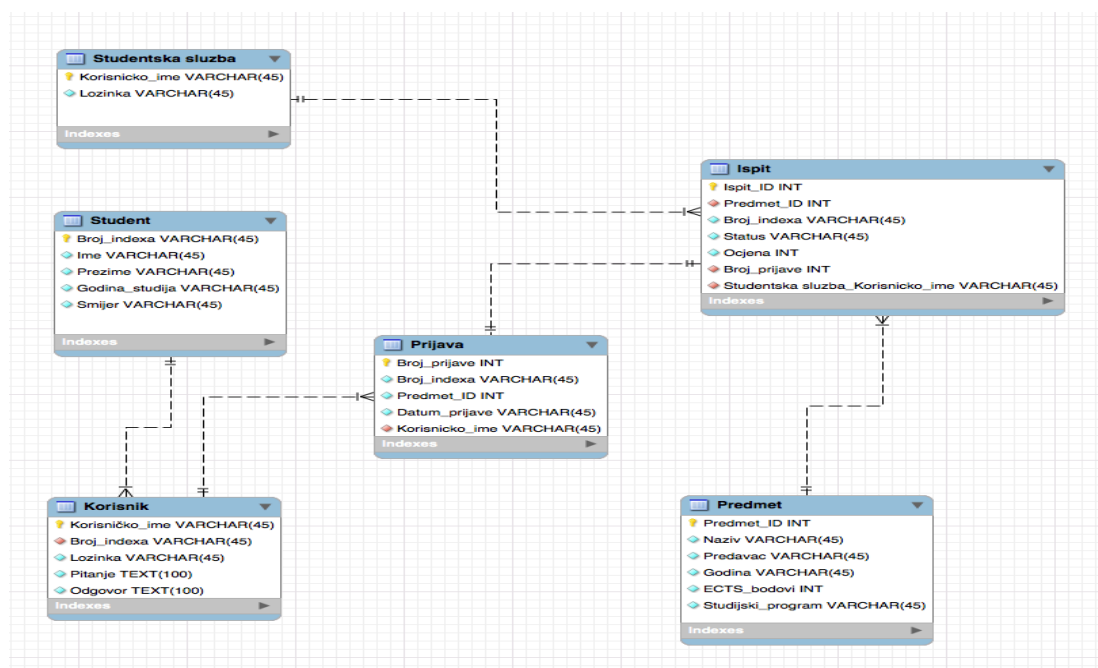
ove Java desktop aplikacije korišten je MySQL Workbench kao sistem za upravljanje i pristup bazi podataka. Baza podataka se također može kreirati korištenjem alata DB Browser koji se koristi za SQL, što je vidljivo na slici u nastavku.

Slika 4.4. Baza podataka kreirana u DB Browseru



Na slici ispod možemo vidjeti grafički prikaz ER modela s notacijom "vranino stopalo" izrađen u alatu MySQL Workbench. Baza podataka se sastoji od 6 tabelela i odgovarajućih atributa. Pokraj svakog atributa označen je i tip podatka koje polje može primiti. Entiteti koji se nalaze u bazi podataka su vidljivi u relacionoj šemi ispod, koja predstavlja tekstualni prikaz entiteta i atributa korištenih za izradu ove aplikacije.

Slika 4.5. ER dijagram izrađen u MySQL Workbench-u



5 RAZVOJ APLIKACIJE

Java desktop aplikacija rađena u okviru ovog diplomskog rada rađena je u NetBeansIDE razvojnom okruženju i temelji se Model-View-Controller arhitekturi, s bazom podataka pisanom u MySQL-u. Aplikacija ima mogućnost pregleda, unosa, uređivanja i brisanja podataka iz baze za što se ujedno koristi i naziv CRUD (Create, Read, Update, Delete). U ovom poglavlju biti će detaljno opisane tehnologije korištene u izradi aplikacije te detaljniji prikaz i pojašnjenje pojedinih dijelova aplikacije.

5.1. Netbeans razvojno okruženje

NetBeans IDE je modularno, integrirano razvojno okruženje (IDE) prvenstveno namijenjeno razvoju Java tehnologija, ali isto tako pruža dosta dodatnih mogućnosti koje mu omogućavaju da se jednako efikasno može koristiti za razvoj računarskih programa u ostalim programskim jezicima kao što su C, C++, PHP, Fortran, Python i drugi. Jednako dobro radi na različitim platformama. NetBeans projekt sastoji se od open source IDE-a i aplikacijske platforme koja se može koristiti kao generički okvir za izgradnju bilo koje vrste aplikacija.

Dizajn i struktura aplikacije rađena je u Netbeans razvojnom okruženju koji predstavlja jedan od najkvalitetnijih open source razvojnih okruženja (engl. Integrated Development Environment, IDE) na tržištu. NetBeans je razvijen pomoću Java programskog jezika te je njegova dostupnost omogućena svim platformama. Nastao je iz studentskog projekta zvanog Xelfi 1996. godine u Češkoj. Nakon postavljanja proizvoda na tržište, isti postiže veliki uspjeh da bi ga 1996. godine kupila već poznata kompanija Sun Microsystems i nakon godinu dana je NetBeans objavljen pod open source licencom. Ono što NetBeans čini posebno zanimljivim je set integriranih alata za izradu i razvoj grafičkih korisničkih interfejsa temeljenih 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 diplomskog rada.

Svaki program koji pravimo u NetBeansu mora da se nalazi u nekom projektu. Tako da kad god počinjemo nov program mi prvo počinjemo od novog projekta. NetBeans dozvoljava da imamo istovremeni pristup većem broju projekata dok je jedan od njih glavni.

NetBeans može da podesi i izgeneriše odgovarajuće fajlove u zavisnosti od toga da li razvijamo običnu java aplikaciju, web aplikaciju, plugin za NetBeans itd. NetBeans nam nudi i pomoć ako želimo da učitamo projekat rađen u drugom razvojnom okruženju.

Nakon što smo kreirali novi projekat u NetBeans-u će se pojaviti taj novi projekat sa već kreiranom klasom pod nazivom Main koja u sebi ima i main metodu. Naš novi projekat je automatski povezan i sa JDK-om kao i sa nekim plugin-ovima. Ako prilikom pisanja koda pogrešimo, napravimo sintaksnu grešku, razvojno okruženje će podvući crvenim dio koda gdje smo napravili grešku i staviti oznaku sa strane da se na tom mestu nalazi greška. Ovo se dešava u letu dok mi kucamo kod, tako da možemo odmah da vidimo gdje smo pogrešili. Ovakav pristup se zove intime kompajliranje i značajno povećava efikasnost prilikom kodovanja. Ponekad će razvojno okruženje dati predlog kako da otklonite grešku. Predlog će se pojaviti kao lampica na koju kad kliknete dobijate spisak predloga za rešavanje greške. Kada želimo da startujemo aplikaciju nije potrebno da posebno pokrećemo kompajler a posebno aplikaciju. Klikom na jedno dugme (play/run) na toolbar-u NetBeans će da izvrši kompajliranje programa, a nakon toga i njegovo pokretanje što je najčešći način pokretanja programa prilikom razvoja.

5.2 Swing aplikacija

Swing klase su dio opšteg skupa “alata” namenjenih izgradnji GUI-a koji se naziva JFC-om (Java Foundation Classes). Za izradu grafičkog korisničkog interfejsa (engl. Graphical User Interface, GUI) ove aplikacije korišten je skup alata, odnosno grafička biblioteka naziva Swing koja je dio Sun-ovog paketa grafičkih komponenti uz biblioteke java.awt te Java 2D. Kompanija Netscape Communications Corporation je prva razvila navedenu biblioteku pod nazivom Internet Foundation Classes 1996. godine, a godinu dana kasnije kompanija Sun Microsystems, koja je razvila Java programski jezik, uključuje biblioteku kao standardni dio Java jezika. Isti takođe omogućuje interakciju korisnika s aplikacijom putem miša, tastature ili drugih ulaznih uređaja. Komponente se nalaze u paketu javax.swing koji je prvi put uveden u Java 1.2 verziju, primarno kreiran kako bi ispravio nedostatke prethodne java.awt biblioteke. Kasnije, Swing biblioteka je kreirana s istim komponentama kao i njen prethodnik, ali joj je zavisnost o grafičkim interfejsima platformi manja. Swing biblioteka nastoji minimalno koristiti resurse platforme na kojoj se program izvršava i zato daje programskom interfejsu trajan izgled na različitim platformama. Pored toga, Swing takođe ima i znatno bogatiji izbor grafičkih komponenti, omogućava odabir grafičkih tema pod nazivom Look and Feel koje se jednostavno kreiraju i mijenjaju čime se povećava Swing-ova prilagodljivost svakoj platformi. Isti je danas standardni paket u Javi za kreiranje korisničkog interfejsa, međutim Swing biblioteka nije potpuna zamjena za *AWT* (Abstract Windowing Toolkit) biblioteku i obe biblioteke se često koriste zajedno.⁹

⁹ Schildt, Herbert. 2012. Java JDK 7, prevod osmog izdanja knjige ”Java: The Complete Reference”. Beograd: Mikro knjiga. str. 945. modifikovan tekst.

Swing je platformski nezavisna grafička biblioteka za izradu grafičkih korisničkih interfejsa Java programa te se velikim dijelom oslanja na Model-View-Controller arhitekturu. Taj tip arhitekture konceptualno razdvaja podatke predstavljene korisniku od interfejsa preko kojeg su mu podaci prikazani, odnosno olakšava prikazivanje vizuelnog dijela aplikacije i omogućava lakše razumijevanje same pozadine aplikacije, odnosno programskog koda. Platformska nezavisnost temelji se na kreiranju vizuelnih komponenti nezavisno od komponenti platforme na kojoj se program izvodi korištenjem Java 2D biblioteke. Prilagođavanje vizuelnih modela Swing komponenti korisničkim potrebama povećava prilagodljivost i jednostavnost upotrebe Swing-a. Vizuelni izgled komponenti određuje se kompozicijom standardnih grafičkih elemenata kao što su okviri prozora, klizači, ukrasi itd. te korisnik po potrebi može zadati grafičke elemente, boju, oblik, zavisno o ulozi pojedine komponente. Osim navedenog, korisnik također može mijenjati ili nadograđivati postojeće komponente i njihove funkcionalnosti. Važno je napomenuti da je sastavni dio svake grafičke komponente AWT-ov Container. To je osnovna klasa u AWT (Abstract Windowing Toolkit) okviru koju nasleđuju sve Swingove klase. Neki od Containera u Swing paketu su JFrame, JPanel, JContentPane itd. Klasa JFrame je glavni prozor aplikacije koji nasleđuje klasu Frame iz biblioteke java.awt, a JPanel predstavlja Container koji se koristi za grupisanje komponenti unutar prozora. Osnovna klasa javax.swing paketa je JComponent klasa te sve swing komponente nasleđuju spomenutu klasu. Neke od osnovnih komponenti koje su korištene u aplikaciji su JFrame, prozor koji je na vrhu hijerarhije Containera sa naslovnom trakom, trakom alate te minimize, maximize i close dugmima, JPanel, JButton koji predstavlja Swing dugme koji može sadržavati ikonu ili tekst te se koristi za otvaranje prozora ili vršenja nekih određenih akcija, JLabel komponenta koja omogućava prikazivanje teksta i ikone na odgovarajućem mjestu, JTextField komponenta koja predstavlja polje za unos teksta, JPasswordField, itd.

5.3 Model-View-Controller arhitektura

MVC je prvi put uveden u okviru SmallTalk-a i predstavlja ideju trodijelnog modela komponenti:

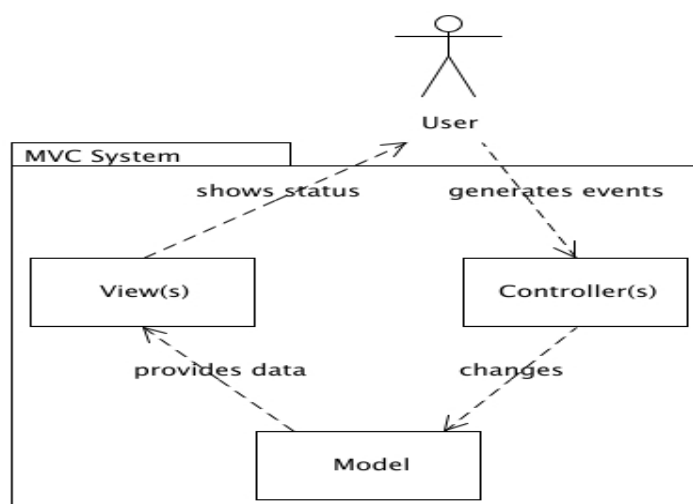
- Model dio – zadužen za čuvanje podataka kojima je komponenta definisana,
- View dio – koji je zadužen za prikaz same komponente,
- Controller deo – zadužen za interakciju sa korisnikom, tačnije kada korisnik ima bilo kakvu “komunikaciju” sa komponentom, Controller dio registruje akciju i mijenja Model i/ili View ako je tako nešto na osnovu akcije korisnika potrebno.

Model-View-Controller obrazac koristi se za odvajanje aplikacije na tri glavne logičke komponente poznate kao model, pogled (eng. View) i kontrolor (eng. Controller). Svaka od navedenih komponenti ima specifičnu namjenu i funkcionalnost.

Takva struktura aplikacije olakšava upravljanje složenim aplikacijama zbog mogućnosti posebnog razvoja svake od komponenti aplikacije i testiranja svakog dijela posebno. Svaka od navedenih komponenti izvršava određene zadatke. Model se sastoji od podataka, poslovnih pravila i informacija koji su ugrađeni u logiku programa. Isti odgovara svim logičkim podacima s kojima korisnik radi. To može predstavljati ili podatke koji se razmijenjuju između komponenti View-a i Controller-a ili bilo koji drugi podaci poslovne logike. Model predstavlja jednu ili više klasa sa svojim stanjima koji se prikazuju na zahtjeve View-a ili se mijenjaju zavisno o Controller-u.

Pogled (eng. View) je vizualni dio aplikacije, on zahtjeva od modela podatke/informacije potrebne za vizualni prikaz modela krajnjem korisniku, odnosno za prikaz podataka, te omogućava mijenjanje podataka. Kontrolor (engl. Controller) djeluje kao interfejs između modela i pogleda kako bi obradio poslovnu logiku i zahtjeve koji dolaze, manipulirao podacima pomoću modela te obavljao interakciju s pogledom s ciljem prikazivanja krajnjih podataka, odnosno kontrolor obavlja interakciju s modelom kako bi kreirao podatke koji će se prikazivati u pogledu. Prema slici 5.1 prikazana je MVC arhitektura i njezin princip rada. MVC aplikacija komunicira s korisnikom putem pogleda, gdje korisnik unosom zahtjeva šalje naredbe kontroloru, dok kontrolor od modela traži da obavi radnju i vrati mu rezultat. Rezultat radnje zatim kontroler šalje u pogled gdje ga korisnik može vidjeti.

Slika 5.1. Model-View-Controller arhitektura



Izvor: <http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html>

6 PRIKAZ REALIZOVANIH FUNKCIONALNOSTI IS

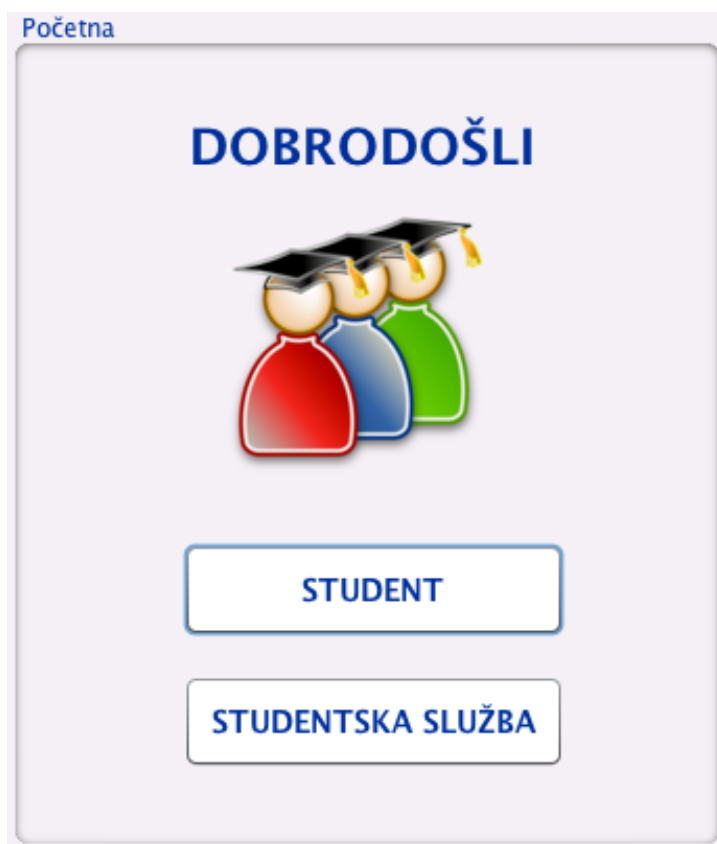
Nakon kraćeg navođenja alata i tehnologija koje smo koristili, u ovom poglavlju dolazimo i do razrade same aplikacije. U ovom dijelu rada su predstavljene funkcionalnosti informacionog sistema kao i način njihovog korištenja.

Aplikacija se sastoji od četrnaest Java okvira (eng. *jframe*). Za svaki kreirani prozor (jframe) napisan je programski kod kako bi dobio svoju funkcionalnost. U sklopu svakog od ponuđenih modula su realizovane različite funkcionalnosti.

Aplikacija je zamisljena da se sastoji od dva različita pogleda, to su studenti i službenici studentske službe. Bila je riječ i o još jednom, trećem pogledu, tzv. admin. Na kraju se ispostavilo da ova dva pogleda imaju dovoljno opcija koje pokrivaju većinu radnji koje službenik na tom radnom mjestu mora izvršiti, pa nije bilo nužno dodavati još jedan pogled.

Na slici ispod možemo vidjeti kako izgleda prvi okvir koji se otvara prilikom pokretanja aplikacije.

Slika 6.1. Prvi prozor koji se pojavljuje pri pokretanju aplikacije



6.1 Studentska služba

Unutar softverskog interfejsa namijenjenog studentskoj službi realizovane su sljedeće funkcionalnosti:

- Dodavanje novog predmeta
- Dodavanje novog studenta
- Unos rezultata ispita
- Pregled prijava na ispit
- Pregled izvještaja sa određenog ispita

Na slici ispod možemo vidjeti prozor "Opcije" koji se prvi otvara nakon uspješnog prijavljivanja službenika studentske službe. Prozor se sastoji od 6 jButton-a, od kojih nas 5 upućuje na neki od narednih okvira u kojima su realizovane gore navedene funkcionalnosti, dok nas dugme "Odjavi se" vraća na prethodni korak prijave.

Slika 6.2. Prozor "Opcije" gdje su prikazane sve funkcionalnosti Studentske službe

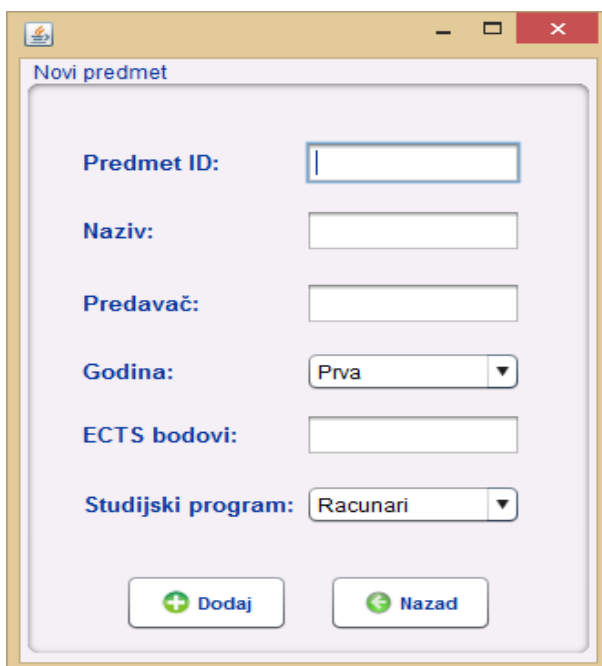


6.1.1 Novi predmet

Dodavanje novog predmeta u informacionom sistemu se vrši odabirom opcije Studentska Služba/Novi predmet.

Nakon što se upišu sve potrebne informacije za određeni predmet u odgovarajuća polja, klikom na dugme „Dodaj“ svi podaci se upisuju u bazu podataka u odgovarajuću tabelu. Klikom na dugme „Nazad“, vraćamo se na prethodni prozor gdje možemo izabrati neku drugu od 5 ponuđenih opcija.

Slika 6.3. Prozor "Novi predmet"



Slika 6.4. Programski kod za dodavanje novog predmeta

```
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    setVisible(false);  
    OpcijeSluzba oss = new OpcijeSluzba();  
    oss.setVisible(true);  
}  
  
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String sql = "INSERT INTO Predmet(Predmet_ID,Naziv,Predavac,Godina,ECTS_bodovi, Studijski_program) values (?, ?, ?, ?, ?)";  
    try{  
        pst=conn.prepareStatement(sql);  
        pst.setString(1, jTextField1.getText());  
        pst.setString(2, jTextField2.getText());  
        pst.setString(3, jTextField3.getText());  
        pst.setString(4, (String) jComboBox1.getSelectedItem());  
        pst.setString(5, jTextField4.getText());  
        pst.setString(6, (String) jComboBox2.getSelectedItem());  
  
        pst.execute();  
        JOptionPane.showMessageDialog(null, "Uspjesno ste dodali novi predmet!", "OBAVJESTENJE", JOptionPane.INFORMATION_MESSAGE);  
    }catch (Exception e ){  
        JOptionPane.showMessageDialog(null, e);  
    }  
}
```

Na slici ispod možemo vidjeti kako izgleda tabela u koju smo uz pomoć naše aplikacije, korištenjem SQL query-a, unosili potrebne podatke o predmetima, a koju ćemo kasnije koristiti za neke druge potrebe, npr. prijavu ispita.

Slika 6.5. Prikaz tabele Predmet u bazi podataka

Table: **Predmet** New Record Delete Record

	Predmet_ID	Naziv	Predavac	Godina	ECTS_bodovi	itudijski_progr
	Filter	Filter	Filter	Filter	Filter	Filter
1	11	Matematika	Vesna	Prva	6	Racunari
2	12	Fizika	Mirjana	Prva	6	Racunari
3	13	Elektronika	Nikolina	Druga	5	Racunari
4	14	Energetika	Nikolina	Treća	5	Racunari
5	15	Programiranje	Goran	Četvrta	7	Racunari
6	16	Istorija	Marko	Prva	4	Pravo
7	17	Marketing	Ana	Druga	4	Ekonomija
8	18	Engleski	Biljana	Treća	5	Ekonomija
9	19	Informatika	Biljana	Druga	5	Ekonomija
10	20	Baze podataka	Drazen	Četvrta	6	Racunari
11	21	Psihologija	Vania	Treća	3	Psihologija

6.1.2 Novi student

Na sličan način kao i kod prethodne opcije Novi predmet, upis novog studenta vrši se odabirom opcije Studentska Služba/Novi student.

Nakon što se upišu svi potrebni podaci u odgovarajuća polja, klikom na dugme „Dodaj“ svi podaci se upisuju u bazu podataka u odgovarajuću tabelu.

Slika 6.6. Prozor ”Novi student”

Slika 6.7. Programski kod za upis novog studenta u bazu

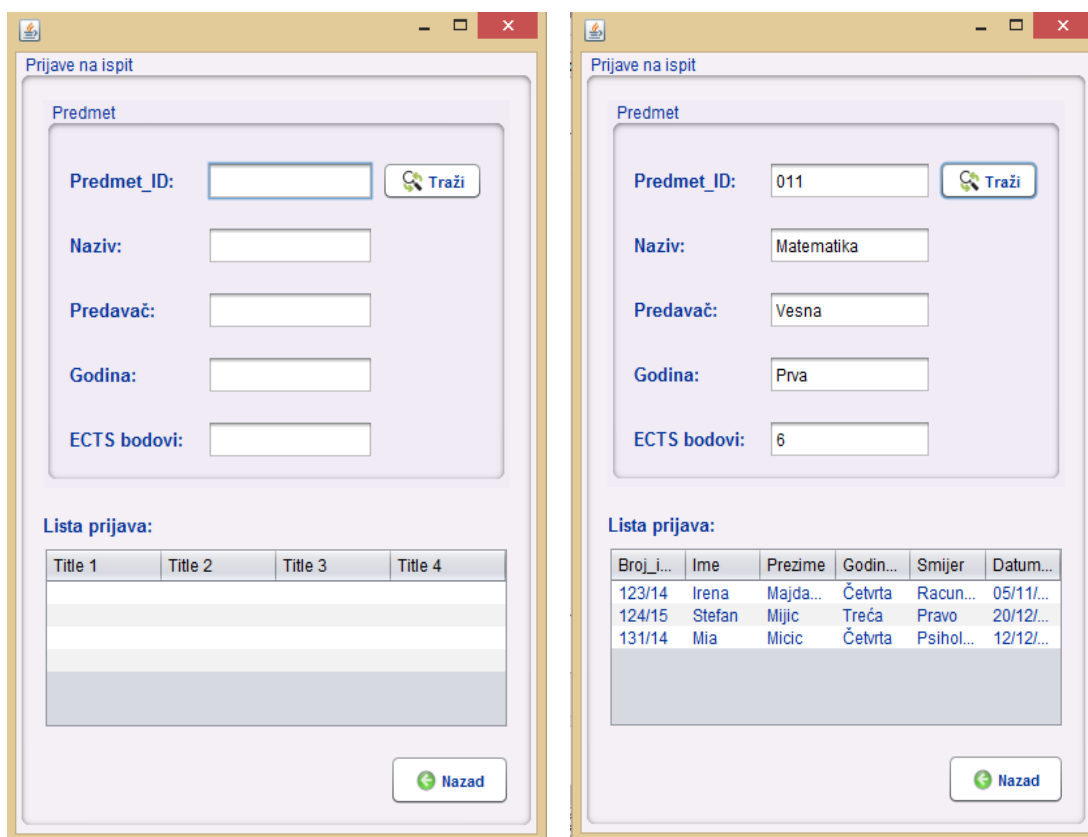
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String sql = "INSERT INTO Student(Broj_indexa,Ime,Prezime,Godina_studija,Smijer) values (?, ?, ?, ?, ?)";  
    try{  
        pst=conn.prepareStatement(sql);  
        pst.setString(1, jTextField1.getText());  
        pst.setString(2, jTextField2.getText());  
        pst.setString(3, jTextField3.getText());  
        pst.setString(4, (String) jComboBox1.getSelectedItem());  
        pst.setString(5, jTextField4.getText());  
  
        pst.execute();  
        JOptionPane.showMessageDialog(null, "Uspjesno ste dodali novog studenta!", "OBAVJESTENJE", JOptionPane.INFORMATION_MESSAGE);  
  
    }catch (Exception e){  
        JOptionPane.showMessageDialog(null, e);  
    }  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    setVisible(false);  
    OpcijeSluzba oss=new OpcijeSluzba();  
    oss.setVisible(true);  
}
```

6.1.3 Prijave na ispit

Odabirom opcije Studentska služba/Prijave na ispit, službenik u studentskoj službi može da izlista sve prijave za traženi predmet. Nakon unosa Predmet_ID-a potrebno je samo kliknuti na dugme „Traži“ kojim se šalje upit bazi podataka i izlistavaju se traženi podaci gdje se Predmet_ID podudara sa unešenim. Svi podaci će se automatski ispisati u odgovarajućim poljima.

Da bismo mogli izlistati sve prijave, odnosno sve studente koji su prijavili traženi predmet, potrebno je kreirati tabelu (JTable - swing komponenta koja omogućava prikazivanje i uređivanje dvodimenzionalnih celija tabela) u koju će se selektovani podaci ispisivati.

Slika 6.8. Prozor pomoću kog možemo izlistati prijave studenata na određeni ispit



U ovom slučaju možemo vidjeti da se preko SQL upita traži da se iz baze podataka, iz tabele Prijava selektuju Broj_indexa, Ime, Prezime, Godina_studija, Smijer i Datum_prijave gdje je Predmet_ID jednak unešenom, te se rezultati upita upisuju u tabelu (JTable). Klikom na dugme "Traži" u JTable će se automatski ispisati traženi podaci, tabela će sadržati sve tražene karakteristike.

Slika 6.9. Programski kod koji pokazuje kako da selektovane podatke iz baze upišemo u JTable

```
public void jTable1(){
    String tr = jTextField1.getText();
    try{
        String sql ="SELECT Broj_indexa, Ime, Prezime, Godina_studija, Smijer, Datum_prijave FROM Prijava WHERE Predmet_ID="+tr+"";
        pst=conn.prepareStatement(sql);
        rs=pst.executeQuery();
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Slika 6.10. Programski kod za izlistavanje prijava na ispit

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String tr = jTextField1.getText();
    String sql = "SELECT * FROM Predmet where Predmet_ID="+tr+"";
    try{
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        if(rs.next()){
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
            jTextField5.setText(rs.getString(5));
            rs.close();
            pst.close();
        }else
            JOptionPane.showMessageDialog(null, "Netačan unos!");
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
    jTable1();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    OpcijeSluzba oss=new OpcijeSluzba();
    oss.setVisible(true);
}
```


6.1.4 Unos rezultata ispita

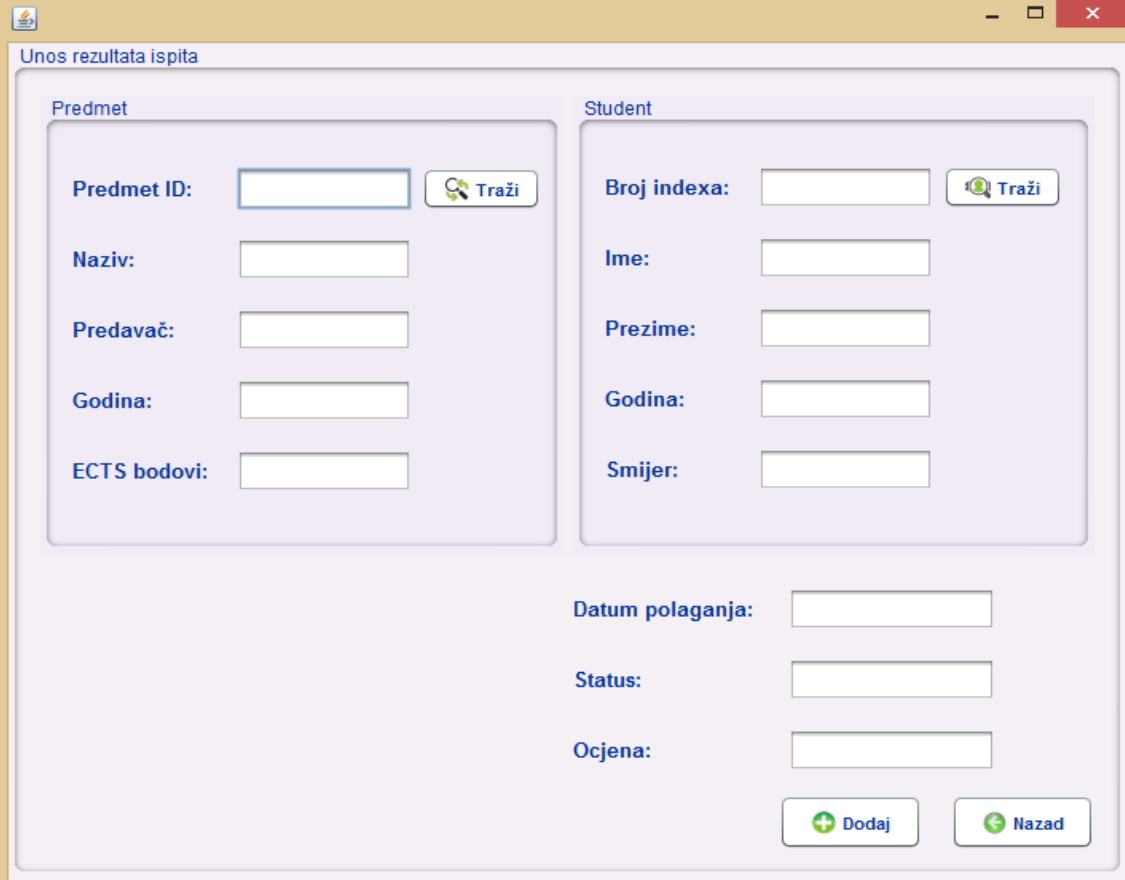
Kada u prozoru "Opcije" odaberemo "Unos rezultata ispita" otvori nam se prozor koji na prvi pogled izgleda kao da zahtjeva unos velikog broja podataka. Međutim, većina polja se automatski popunjava, dovoljno je samo da unesemo podatak na osnovu kog će povući tražene podatke iz baze.

Klikom na dugme "Traži", izvršice se traženi SQL upit i podaci će se upisati u naznačena polja. Da bismo dobili podatke o željenom predmetu, dovoljno je da upišemo ID predmeta. Slično, da bismo dobili ostale podatke o željenom studentu, dovoljno je upisati njegov broj indexa. Kada imamo sve potrebne podatke, od dodatnih informacija treba da unesemo datum kada je student polagao određeni predmet, da li ga je položio i sa kom ocjenom.

Kada su sva polja popunjena, klikom na dugme "Dodaj" podaci se upisuju u bazu podataka u tabelu Ispit. Poslije tim podacima studenti mogu da pristupe, da bi pregledali svoje položene ispite ili eventualno svoj prosjek ocjena.

Klikom na dugme "Nazad" vraćamo se ponovo na prozor "Opcije".

Slika 6.11. Prozor koji koristimo za unos rezultata ispita



Unos rezultata ispita

Predmet

Predmet ID: Traži

Naziv:

Predavač:

Godina:

ECTS bodovi:

Student

Broj indexa: Traži

Ime:

Prezime:

Godina:

Smijer:

Datum polaganja:

Status:

Ocjena:

+ Dodaj ← Nazad

Slika 6.12. Programski kod pomoću kog selektujemo željene podatke iz baze podataka i ispisujemo ih u naznačena polja

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String tx = jTextField1.getText();
    String sql = "SELECT * FROM Predmet where Predmet_ID='"+tx+"'";
    try{
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        if(rs.next()){
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
            jTextField5.setText(rs.getString(5));
            rs.close();
            pst.close();
        }else
        JOptionPane.showMessageDialog(null, "Metađan unos!");
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    String tx = jTextField6.getText();
    String sql = "SELECT * FROM Student where Broj_indexa='"+tx+"'";
    try{
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        if(rs.next()){
            jTextField7.setText(rs.getString(2));
            jTextField8.setText(rs.getString(3));
            jTextField9.setText(rs.getString(4));
            jTextField10.setText(rs.getString(5));
            rs.close();
            pst.close();
        }else
    }
}

```

Slika 6.13. Programski kod koji pokazuje kako da upisemo podatke u bazu u željenu tabelu

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    String sql= "INSERT INTO Ispite (Predmet_ID,Naiv,Predavac,Godina,ECTS_bodovi,Broj_indexa,Ime,Presime,Godina_studija,SmiJer,Datum_polaganja,Status,Ocjena) values (?,?,?,?,?,?,?,?,?,?,?,?)";
    try{
        pst=conn.prepareStatement(sql);
        pst.setString(1, jTextField1.getText());
        pst.setString(2, jTextField2.getText());
        pst.setString(3, jTextField3.getText());
        pst.setString(4, jTextField4.getText());
        pst.setString(5, jTextField5.getText());
        pst.setString(6, jTextField6.getText());
        pst.setString(7, jTextField7.getText());
        pst.setString(8, jTextField8.getText());
        pst.setString(9, jTextField9.getText());
        pst.setString(10, jTextField10.getText());
        pst.setString(11, jTextField11.getText());
        pst.setString(12, jTextField12.getText());
        pst.setString(13, jTextField13.getText());
        pst.execute();
        JOptionPane.showMessageDialog(null, "Uspješno ste unijeli rezultate ispita!");
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    OpcijeSluzba oa=new OpcijeSluzba();
    oa.setVisible(true);
}

```

6.1.5 Izvještaj sa ispita

Ako želimo da vidimo spisak svih studenata koji su položili određeni ispit, u opcijama biramo "Izvještaj sa ispita". Slično kao i kod pregleda prijave na ispit, nakon što unesemo Predmet_ID-a potrebno je samo kliknuti na dugme „Traži“ kojim se šalje upit bazi podataka i izlistava tražene podatke gdje se Predmet_ID podudara sa unešenim. Svi podaci će se automatski ispisati u odgovarajućim poljima.

Takođe je potrebno kreirati tabelu (JTable) u koju će se selektovani podaci ispisivati. Pomoću SQL upita tražimo da se iz baze podaka, iz tabele Ispit selektuju Broj_indexa, Ime, Prezime, Godina_studija, Smijer i Ocjena gdje je Predmet_ID jednak unešenom i Status = Položio, te se rezultati upita upisuju u tabelu (JTable).

Slika 6.14. Prozor koji nam služi za pregled rezultata ispita iz određenog predmeta

Izvještaj sa ispita

Predmet

Predmet ID:

Naziv:

Predavač:

Godina:

ECTS bodovi:

Ispit su položili:

Title 1	Title 2	Title 3	Title 4

Slika 6.15. Programski kod koji nam pokazuje kako da selektujemo određene elemente iz baze i upišemo ih u naznačena polja

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String tr = jTextField1.getText();
    String sql = "SELECT * FROM Predmet where Predmet_ID='"+tr+"'";
    try{
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        if(rs.next()){
            jTextField2.setText(rs.getString(2));
            jTextField3.setText(rs.getString(3));
            jTextField4.setText(rs.getString(4));
            jTextField5.setText(rs.getString(5));
            rs.close();
            pst.close();
        }else
            JOptionPane.showMessageDialog(null, "Netačan unos!");
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
    jTable1();
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    setVisible(false);
    OpcijeSluzba oss=new OpcijeSluzba();
    oss.setVisible(true);
}

```

Slika 6.16. Programski kod koji pokazuje kako da selektovane podatke iz baze upišemo u JTable

```

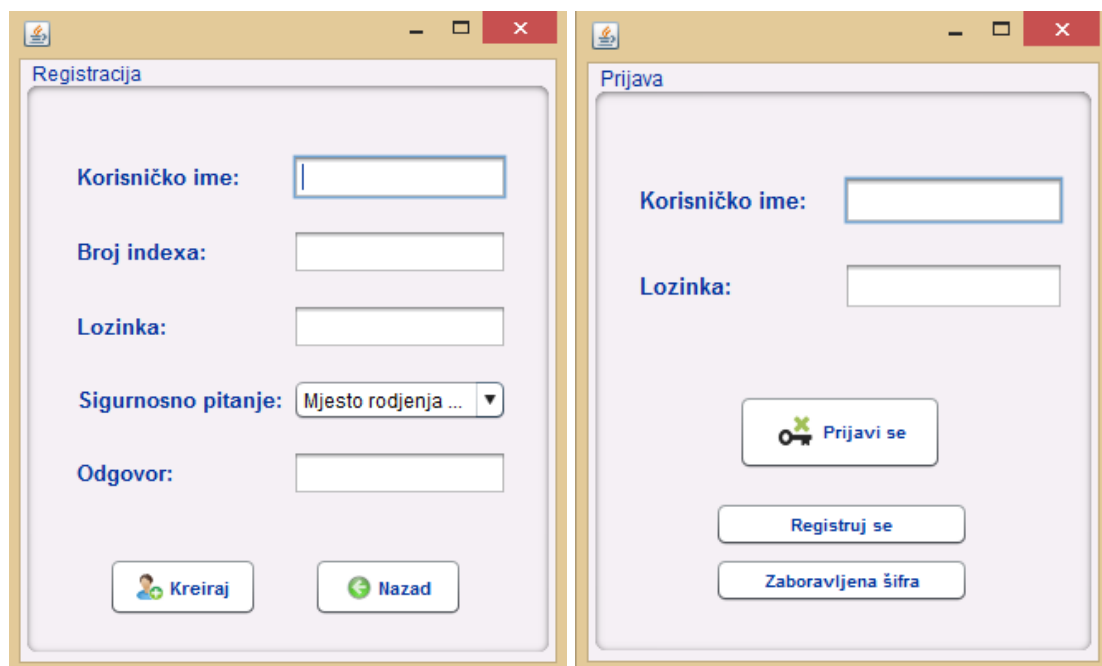
public void jTable1(){
    String tr = jTextField1.getText();
    try{
        String sql ="SELECT Broj_indexa, Ime, Prezime, Godina_studija, Smijer, Ocjena FROM Ispit WHERE Predmet_ID='"+tr+"' AND Status='Polozio'";
        pst=conn.prepareStatement(sql);
        rs=pst.executeQuery();
        jTable1.setModel(DbUtils.resultSetToTableModel(rs));
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

```

6.2 Student

Da bismo se prijavili kao Student, potrebno je da se prvo registrujemo, da bi naše korisničko ime klikom na dugme “Kreiraj” bilo upisano u odgovarajuću tabelu u bazi podataka. Da bi student mogao da se registruje, mora biti student datog fakulteta, tj. mora posjedovati broj indexa koji negdje u bazi već postoji. Nakon što smo se registrovali, klikom na dugme “Nazad” vraćamo se na prozor za prijavu gdje unosimo korisničko ime i lozinku, ti navedeni parametri su sada već definisani u bazi podataka.

Slika 6.17. Prozori za registraciju i prijavu studenta



Slika 6.18. Programski kod za dugme ”Prijavi se”

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String sql = "SELECT * FROM Korisnik where Korisnicko_ime=? and Lozinka=?";  
    try{  
        pst = conn.prepareStatement(sql);  
        pst.setString(1, jTextField1.getText());  
        pst.setString(2, jPasswordField1.getText());  
        rs = pst.executeQuery();  
        if(rs.next()){  
            rs.close();  
            pst.close();  
  
            setVisible(false);  
            OpcijeStudent os = new OpcijeStudent();  
            os.setVisible(true);  
        }  
        else{  
            JOptionPane.showMessageDialog(null, "Pogresno ste unijeli korisnicko ime ili šifru!", "UPOZORENJE" ,JOptionPane.INFORMATION_MESSAGE);  
        }  
    }catch (Exception e){  
        JOptionPane.showMessageDialog(null, e);  
    }finally{  
        try{  
            rs.close();  
            pst.close();  
        }catch (Exception e){  
        }  
    }  
}
```

U programskom kodu dugmeta "Prijavi se" možemo vidjeti da postoje dva parametra `Korisnicko_Ime` i `Lozinka` gdje se pomoću `get` metode traži pristup unešenim parametrima. Njih možemo gledati kao ulazne parametre koje se šalju upitom na bazu podataka s ciljem provjere parametara, odnosno da li unešeni podaci prilikom prijave odgovaraju onima u bazi podataka.

Za spajanje baze sa aplikacijom potreban je JDBC API (Java Database Connectivity) koji se u ovom slučaju naziva `sqlite-jdbc 3.8.11` i koji je dodan u aplikaciju u `lib` folder. Nakon uspostavljene veze sa bazom koristi se SQL izraz `SELECT` koji će pokupiti sve podatke iz baze podataka koje su vezane za uneseno korisničko ime i šifru.

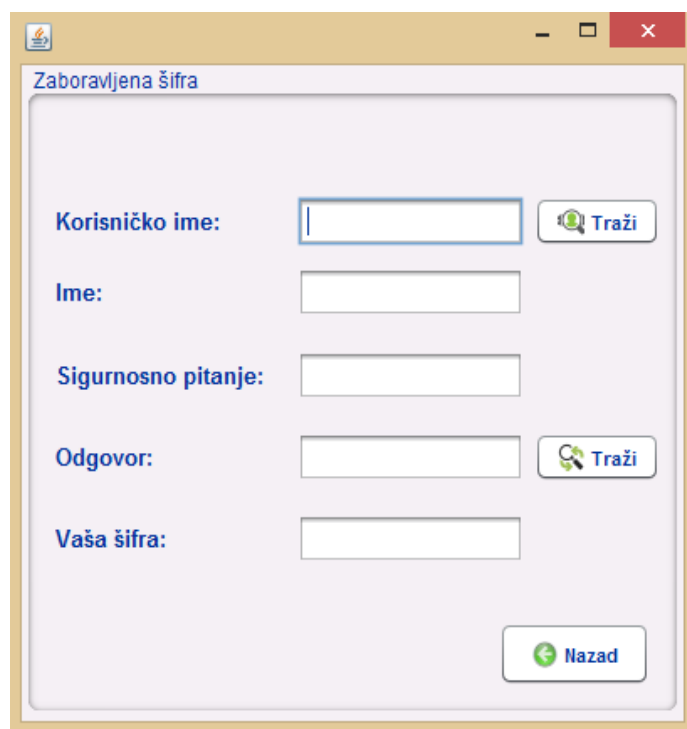
Ako su parametri tačni, odnosno isti kao u bazi podataka otvara se novi prozor "Opcije". Ukoliko unešeni parametri nisu tačni, odnosno ne podudaraju se sa onima u bazi podataka aplikacija nam vraća informaciju o grešci. Nakon uspješnog logovanja otvara se novi prozor pod nazivom "Opcije".

Slika 6.19. Prozor "Opcije"



Aplikacija nam takođe nudi i opciju za povrat lozinke u slučaju da smo je zaboravili, potrebno je samo dati tačan odgovor na sigurnosno pitanje. Klikom na dugme “Traži” šalje se upit bazi, ako se podaci podudaraju, korisnik će svoju šifru dobiti ispisanu u naznačenoj labeli.

Slika 6.20. Prozor za vraćanje zaboravljene šifre



Slika 6.21. Programski kod za vraćanje zaboravljene šifre

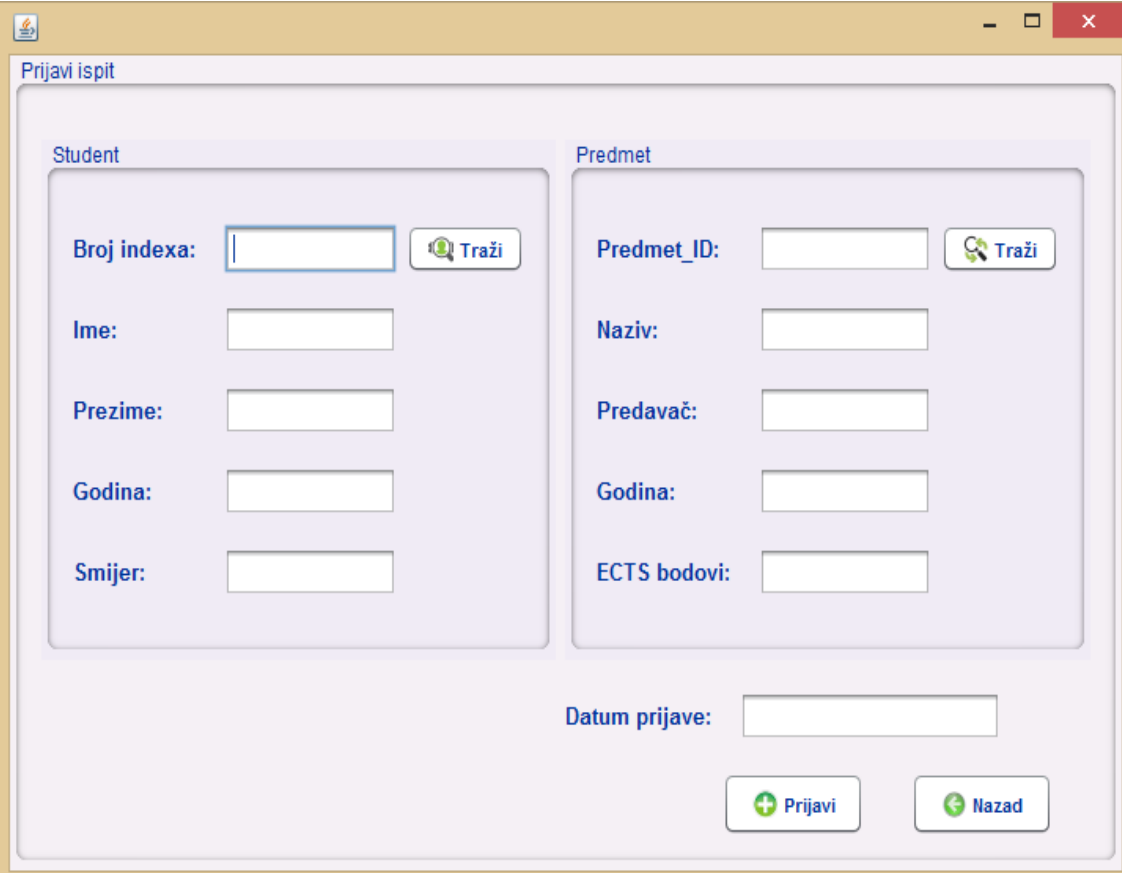
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    String tr = jTextField1.getText();  
    String sql = "SELECT * FROM Korisnik where Korisnicko_Ime='"+tr+"'";  
    try{  
        pst = conn.prepareStatement(sql);  
        rs = pst.executeQuery();  
        if(rs.next()){  
            jTextField2.setText(rs.getString(2));  
            jTextField3.setText(rs.getString(4));  
            rs.close();  
            pst.close();  
        }else  
        JOptionPane.showMessageDialog(null, "Netacno korisnicko ime");  
    }catch (Exception e){  
        JOptionPane.showMessageDialog(null, e);  
    }  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    String vr1 = jTextField1.getText();  
    String vr2 = jTextField4.getText();  
    String sql = "SELECT * FROM Korisnik where Odgovor='"+vr2+"' AND Korisnicko_Ime='"+vr1+"'";  
    try{  
        pst = conn.prepareStatement(sql);  
        rs = pst.executeQuery();  
        if(rs.next()){  
            jTextField5.setText(rs.getString(3));  
        }  
    }  
    }catch (Exception e){  
        JOptionPane.showMessageDialog(null, e);  
    }  
}
```

6.2.1 Prijavljivanje ispita

Kada u prozoru "Opcije" odaberemo "Prijavi ispit" otvori nam se prozor koji na prvi pogled izgleda kao da zahtjeva unos velikog broja podataka. Međutim, kao što smo već pokazali na opisu prozora za unos rezultata ispita, većina polja se automatski popunjava, dovoljno je samo da unesemo podatak na osnovu kog će povući tražene podatke iz baze.

Klikom na dugme "Traži", izvršice se traženi SQL upit i podaci će se upisati u naznačena polja. Da bismo dobili podatke o željenom predmetu, dovoljno je da upišemo ID predmeta. Slično, da bismo dobili ostale podatke o željenom studentu, dovoljno je upisati njegov broj indexa. Kada imamo sve potrebne podatke, od dodatnih informacija treba da unesemo datum prijave. Kada su sva polja popunjena, klikom na dugme "Dodaj" podaci se upisuju u bazu podataka u tabelu Prijava. Toj tabeli službenik studentske službe može da pristupi u svakom trenutku, takođe putem aplikacije. Klikom na dugme "Nazad" vraćamo se ponovo na prozor "Opcije".

Slika 6.22. Prozor koji studentu služi za prijavu ispita



Prijavi ispit

Student

Broj indexa:

Ime:

Prezime:

Godina:

Smijer:

Predmet

Predmet_ID:

Naziv:

Predavač:

Godina:

ECTS bodovi:

Datum prijave:

Slika 6.23. Programski kod koji nam pokazuje kako da upisemo podatke u bazu u željenu tabelu

```
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    String sql= "INSERT INTO Prijava(Broj_indexa,Ime,Prezime,Godina_studija,SmiJer,Predmet_ID,Nasiv,Predavac,Godina,ECTS_bodovi,Datum_prijave)values(?,?,?,?,?,?,?,?,?,?,?)";  
    try{  
        pst=conn.prepareStatement(sql);  
        pst.setString(1, jTextField1.getText());  
        pst.setString(2, jTextField2.getText());  
        pst.setString(3, jTextField3.getText());  
        pst.setString(4, jTextField4.getText());  
        pst.setString(5, jTextField5.getText());  
        pst.setString(6, jTextField6.getText());  
        pst.setString(7, jTextField7.getText());  
        pst.setString(8, jTextField8.getText());  
        pst.setString(9, jTextField9.getText());  
        pst.setString(10, jTextField10.getText());  
        pst.setString(11, jTextField11.getText());  
        pst.execute();  
        JOptionPane.showMessageDialog(null, "Uspješno ste prijavili ispit!");  
    }  
    catch (Exception e){  
        JOptionPane.showMessageDialog(null, e);  
    }  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    setVisible(false);  
    OpцијеStudent os=new OpцијеStudent();  
    os.setVisible(true);  
}
```

6.2.2 Izvještaj položenih ispita

Prozor koji studentu služi da pogleda svoj izvještaj, odnosno spisak svih predmeta koje je položio, kao i prosjek ocjena. Otvara se kada u prozoru "Opcije" odaberemo "Izvještaj".

Nakon što unesemo broj indexa potrebno je samo kliknuti na dugme „Traži“ kojim se šalje upit bazi podataka i izlistava tražene podatke gdje se Broj_indexa podudara sa unesenim. Svi podaci će se automatski ispisati u odgovarajućim poljima.

Takođe je potrebno kreirati tabelu (JTable) u koju će se selektovani podaci ispisivati. Pomoću SQL upita tražimo da se iz baze podaka, iz tabele Ispit selektuju Predmet_ID, Naziv, Godina, ECTS_bodovi i Ocjena gdje je Broj_indexa jednak unesenom i Status = Položio, te se rezultati upita upisuju u tabelu (JTable).

Slika 6.24. Prozor pomoću kog student gleda svoj izvještaj položenih predmeta i prosjek ocjena

The image shows two screenshots of a web application window titled "Izvještaj".

Left Screenshot: Shows the search form. The "Broj indexa:" field is empty. Below it are fields for "Ime:", "Prezime:", "Godina:", and "Smijer:". A "Traži" button is to the right of the "Broj indexa:" field.

Right Screenshot: Shows the search results. The "Broj indexa:" field contains "123/14". The "Ime:" field contains "Irena", "Prezime:" contains "Majdandzic", "Godina:" contains "Četvrta", and "Smijer:" contains "Racunari". Below the search form is a table titled "Položeni predmeti:".

Predmet...	Naziv	Godina	ECTS_b...	Ocjena
11	Matema...	Prva	6	9
15	Program...	Četvrta	7	8
12	Fizika	Prva	6	7
13	Elektron...	Druga	5	10

Below the table, the "Prosjeck ocjena:" field contains "8.5". Both screenshots have a "Nazad" button at the bottom right.

Slika 6.25. Programski kod koji pokazuje računanje prosjeka ocjena i upis traženih rezultata u JTable

```
public void jTable2(){
    String bri=jTextField2.getText();
    try{
        String sql ="SELECT Predmet_ID, Naziv, Godina, ECTS_bodovi, Ocjena FROM Ispit WHERE Broj_indexa='"+bri+"' AND Status='Polozio'";
        pst=conn.prepareStatement(sql);
        rs=pst.executeQuery();
        jTable2.setModel(DbUtils.resultSetToTableModel(rs));
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}

public void Prosjek(){
    String bri=jTextField2.getText();
    String sql = "SELECT AVG(Ocjena) FROM Ispit WHERE Broj_indexa='"+bri+"'";

    try{
        pst = conn.prepareStatement(sql);
        rs = pst.executeQuery();
        if(rs.next()){
            String avg=rs.getString("AVG(Ocjena)");
            jTextField1.setText(avg);

            rs.close();
            pst.close();
        }else
            JOptionPane.showMessageDialog(null, "Netačan unos!");
    }catch (Exception e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

7. ZAKLJUČAK

Aplikacija izrađena u okviru ovog diplomskog rada predstavlja primjer jednostavne Java desktop aplikacije korištenjem Swing-a. Iz prikazanog se može zaključiti da se upotrebom navedene biblioteke mogu brzo graditi korisne i prilagodljive aplikacije namijenjene korištenju na različitim računalnim platformama.

U ovom radu je opisan proces izgradnje informacionog sistema univerziteta. Realizacijom ovog informacionog sistema stvorili bi se uslovi za efikasnije funkcionisanje fakulteta.

Cilj prilikom izrade aplikacije bio je naučiti osnove objektno orijentiranog programiranja, kreiranje MySQL baze podataka te povezivanje aplikacije s bazom. Iz svega navedenog, može se zaključiti da odabir Jave kao jezika za izradu desktop aplikacija pruža najviše pogodnosti u smislu prenosivosti aplikacije, tj. mogućnosti korištenja na raznim platformama.

Desktop aplikacije su vrsta aplikacija koje je potrebno prenijeti a u velikom broju i instalirati na svakom računaru posebno. Ako dođe do promjene nekog dijela programa, na svakom računaru pojedinačno koji koristi ovu aplikaciju mora se izvršiti ažuriranje ili instalacija nove verzije aplikacije. Situacija se dodatno komplikuje ukoliko aplikacija radi sa podacima na lokalnom računaru. Ovo bi značilo da je osoba koja radi sa programom vezana za rad isključivo na svom računaru ili bi morala da sa sobom nosi sve podatke koji su joj potrebni za rad. Postoje i desktop aplikacije koje koriste udaljene ili centralizovane baze podataka koje se nalaze na serverima pa time olakšavaju pristup podacima i prevazilaze problem dostupnosti i tačnosti podataka. Današnje desktop aplikacije u najvećem broju slučajeva koriste ovakav pristup.

U prošlosti su gotovo sve aplikacije bile ovog tipa, međutim, razvojem globalne svijetske mreže i unapređenjem računarske moći i tehnologija sve više aplikacija se realizuju kao veb aplikacije.

Dalji plan razvoja je izrada veb aplikacije, kojom bi bila omogućena prijava ispita na daljinu za studente koji nemaju mogućnost da ispite prijave u zgradi fakulteta, kao i pregled rasporeda polaganja i ispitnih rezultata. Administrativnim službenicima fakulteta bi se olakšao rad posebno na aktivnostima izdavanja uverenja o položenim ispitima. Izloženim softverskim rešenjem operateri su oslobođeni obaveza mukotrpih proveravanja ocena u zapisniku, prijavi i matičnoj knjizi fakulteta, kada je često dolazilo do nepredviđenih grešaka oko datuma položenog ispita, broja indeksa, imena i prezimena i ostalog.

Na osnovu predstavljenih tehnologija, postupaka i primjera iz aplikacije možemo vidjeti da se upotrebom savremenih računarskih tehnologija mogu napraviti aplikacije koje značajno pojednostavljuju, ubrzavaju i olaksavaju svakodnevne poslovne procese i aktivnosti.

LITERATURA

- Eckel, Bruce. 2007. Misliti na Javi, prevod četvrtog izdanja knjige "Thinking in Java". Beograd: Mikro knjiga.
- Živković, Dejan. 2013. Java Programiranje, drugo izdanje. Beograd: Univerzitet Singidunum.
- Čupić, Marko. 2015. Programiranje u Javi. Zagreb: Fakultet elektrotehnike i računarstva.
- Schildt, Herbert. 2012. Java JDK 7, prevod osmog izdanja knjige "Java: The Complete Reference". Beograd: Mikro knjiga.
- Marić, Slavko i Dražen Brđanin. 2012. Relacione baze podataka. Banja Luka: Elektrotehnički fakultet.
- Veinović, Mladen i Goran Šimić. 2010. Uvod u baze podataka. Beograd: Univerzitet Singidunum.
- www.netbeans.org; <https://netbeans.org/features/index.html>; novembar 2018.
- www.mysql.com; <https://dev.mysql.com/doc/mysql-tutorial-excerpt/5.5/en/>; novembar, 2018.
- www.oracle.com; <https://docs.oracle.com/javase/tutorial/java/index.html>; novembar 2018
- www.javamagazine.com; <http://www.javamagazine.mozaicreader.com/NovemberDecember2018>; novembar 2018.
- www.w3schools.com; <https://www.w3schools.com/sql/>; decembar, 2018.
- www.codecademy.com; <https://www.codecademy.com/learn>; decembar, 2018.