# Razvoj aplikacije "LabReport"

**Magdalenić, Zlatko**

**VELEUČILIŠTE U RIJECI**

Zlatko Magdalenić

# RAZVOJ APLIKACIJE „LabReport“

(završni rad)

Rijeka, 2020.

# VELEUČILIŠTE U RIJECI

Poslovni odjel

Stručni studij informatika


# RAZVOJ APLIKACIJE „LabReport“

(završni rad)


MENTOR

Vlatka Davidović, v. pred.

STUDENT

Zlatko Magdalenić

MBS: 2422000001/17


Rijeka, rujan 2020.

**VELEUČILIŠTE U RIJECI** **Prilog 1.**

**Poslovni odjel**

**Rijeka, 14. svibnja 2020.**

# ZADATAK
## za završni rad

**Pristupnik ZLATKO MAGDALENIĆ** **MBS: 2422000001/17**

**Studentu preddiplomskog stručnog studija Informatika izdaje se zadatak završni rad – tema završnog rada pod nazivom:**

**RAZVOJ APLIKACIJE "LabReport"**

**Sadržaj zadatka:** Razviti aplikaciju „LabReport" za upravljanje uzorcima u laboratoriju te izradu izvještaja s rezultatima analiza. Aplikacija pohranjuje informacije o uzorcima, te olakšava njihovo praćenje. U radu treba opisati funkcionalnosti sustava, te napraviti analizu, dizajn i implementaciju tih funkcionalnosti u Java programskom jeziku.

**Preporuka** _____

_____

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

**Zadano: 14. svibnja 2020.** **Predati do: 15. rujna 2020.**

**Mentorica:**
**Vlatka Davidović, v.pred.**

**Pročelnica odjela:**
**mr.sc.Anita Stilin, v.pred.**

**Zadatak primio dana: 14.svibnja 2020.**

_____
**Zlatko Magdalenić**

**Dostavlja se:**
- **mentoru**
- **pristupniku**

# I Z J A V A

Izjavljujem da sam završni rad pod naslovom: Razvoj aplikacije „LabReport“ izradio samostalno pod nadzorom i uz stručnu pomoć mentora Vlatke Davidović.

Ime i prezime

(potpis studenta)

# SAŽETAK

Ovaj rad opisuje izradu Java aplikacije „LabReport“ koja služi za upravljanje uzorcima u laboratoriju te izradu izvještaja s rezultatima analiza, koji se predaju klijentima. Aplikacija se sastoji od MySQL baze podataka koja je povezana sa samom aplikacijom i služi za pohranjivanje informacija o uzorcima. Aplikacija je osmišljena da olakša praćenje uzoraka i smanji mogućnost pogreške prilikom izrade izvještaja. Za izradu aplikacije je korišten NetBeans IDE, upotrijebljene su dodatne biblioteke za izgled same aplikacije – *FlatLaf* koji daje moderniji izgled elementima sučelja od standardnog *Swing-a,* te *JDateChooser* koji omogućava jednostavan odabir datuma. U radu su opisane funkcionalnosti sustava te analiza, dizajn i implementacija tih funkcionalnosti. Također su opisane i tehnologije koje su korištene za implementaciju opisanih funkcionalnosti.

**Ključne riječi:** Java, MySQL, laboratorij, uzorak, izvještaj

# SADRŽAJ

# 1. UVOD

Rad u laboratoriju podrazumijeva izvođenje različitih analiza, obradu rezultata te izdavanje izvješća o tim rezultatima. Upravo ova zadnja faza, izdavanje izvješća, je najpodložnija greškama analitičara. „LabReport“ je desktop aplikacija namijenjena djelatnicima koji rade u laboratoriju, a koji su zaduženi za pisanje i izdavanje izvješća svojim klijentima. Kako bi se smanjila mogućnost pogreške prilikom pisanja izvješća, potrebno je taj proces automatizirati. Upravo to je namjena aplikacije, uz neke dodatne mogućnosti, kao što su pregled klijenata, uzoraka i samih izvještaja.

LabReport aplikacija je pisana u Java jeziku, sastoji se od baze podataka i same aplikacije koja ima dio za djelatnike te dio za administratora/voditelja laboratorija. Sve radnje se odvijaju unutar jednog prozora, koji ima tematski odijeljene kartice.

# 2. OPIS RADA U LABORATORIJU

Rad u laboratoriju ima nekoliko faza – uzorkovanje, prijem i upis uzoraka, odabir analiza, izvršavanje analiza, obrada rezultata te izrada izvještaja. Uzorke koji se uzorkuju ili su dostavljeni u laboratorij potrebno je jednoznačno označiti laboratorijskim brojem, pomoću kojega se onda uzorak jednostavno prati tijekom analiza. Prilikom dodijele tog broja, upisuju se podaci o samom uzorku (datum i vrijeme uzorkovanja, kupac, djelatnik koji je izvršio uzorkovanje) te se odabiru analize koje će se izvršiti. Nakon provedenih analiza potrebno je obraditi rezultate te ih upisati kako bi se moglo napraviti laboratorijsko izvješće koje se predaje kupcu. Izvještaj se sastoji od podataka o djelatniku koji je uzorak uzorkovao, podataka o klijentu – naziv, adresa, OIB, šifra, zatim od podataka o uzorku – laboratorijski broj, oznaka uzorka, mjesto i vrijeme uzorkovanja, početak i završetak analize te od rezultata analize – naziv i šifra parametra, mjerna jedinica i granična vrijednost te rezultat za pojedini parametar.

# 3. ANALIZA SUSTAVA

U ovom projektu korištena je objektno orijentirana analiza koja prikazuje korisnički pogled na sustav. Kako bi se ovakav model izgradio, potrebno je prethodno prikupiti zahtjeve korisnika i klijenata. Na ovaj način se rješava pitanje ŠTO sustav treba raditi, te se ne rješavaju tehnološka pitanja same aplikacije.

Cilj ove faze je prikupiti što više znanja o projektu i o sustavu, te se ciljevi mogu podijeliti na:

- Prikupljanje i specifikacija zahtjeva,
- Nalaženje aktera sustava,
- Popis funkcionalnosti koje sustav treba imati (use-cases). (Davidović, 2016)

## 3.1 UML dijagrami korištenja sustava

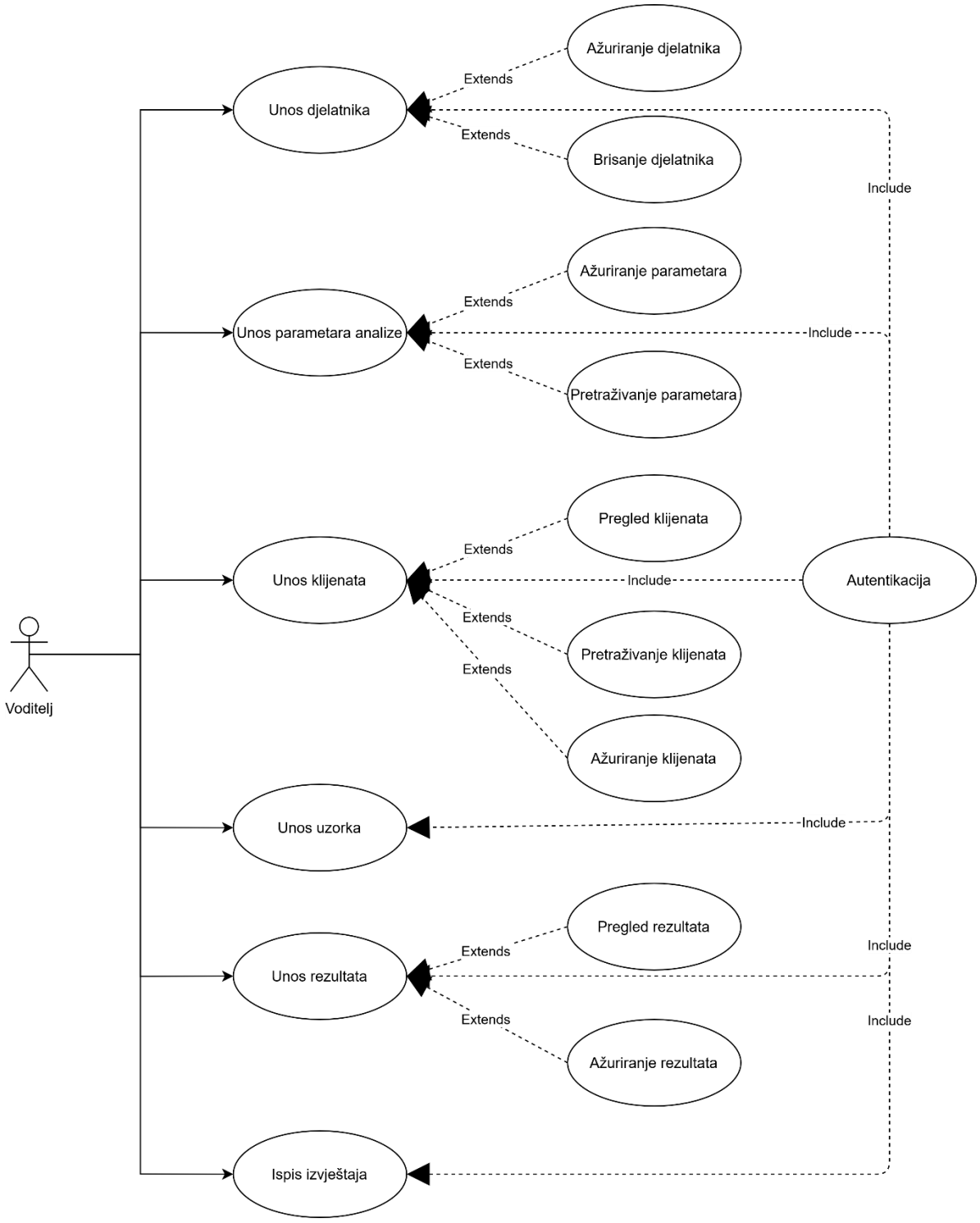
UML dijagrami korištenja sustava (engl. *Use-case Diagram*) prikazuju granice sustava, te odnose između sustava, aktera i slučajeva korištenja. Kroz ove dijagrame se nastoji odgovoriti na pitanje ŠTO sustav treba raditi. (Davidović, 2016)

Aplikacija LabReport je namijenjena djelatnicima laboratorija i voditelju laboratorija/administratoru. Njihove uloge u laboratoriju su različite, stoga su im i ovlasti različite. U nastavku je prikaz ovlasti i uloga pojedinih aktera te objašnjenja mogućih scenarija korištenja.

Voditelj laboratorija:

- Unos djelatnika
- Ažuriranje djelatnika
- Brisanje djelatnika
- Unos parametara analiza
- Ažuriranje parametara analiza
- Pretraživanje parametara analiza
- Unos klijenata
- Pregled klijenata
- Ažuriranje klijenata
- Pretraživanje klijenata
- Unos uzorka
- Unos rezultata
- Pregled rezultata
- Ažuriranje rezultata
- Izrada/ispis analitičkog izvješća

*Slika 1 Dijagram korištenja za ulogu voditelj*

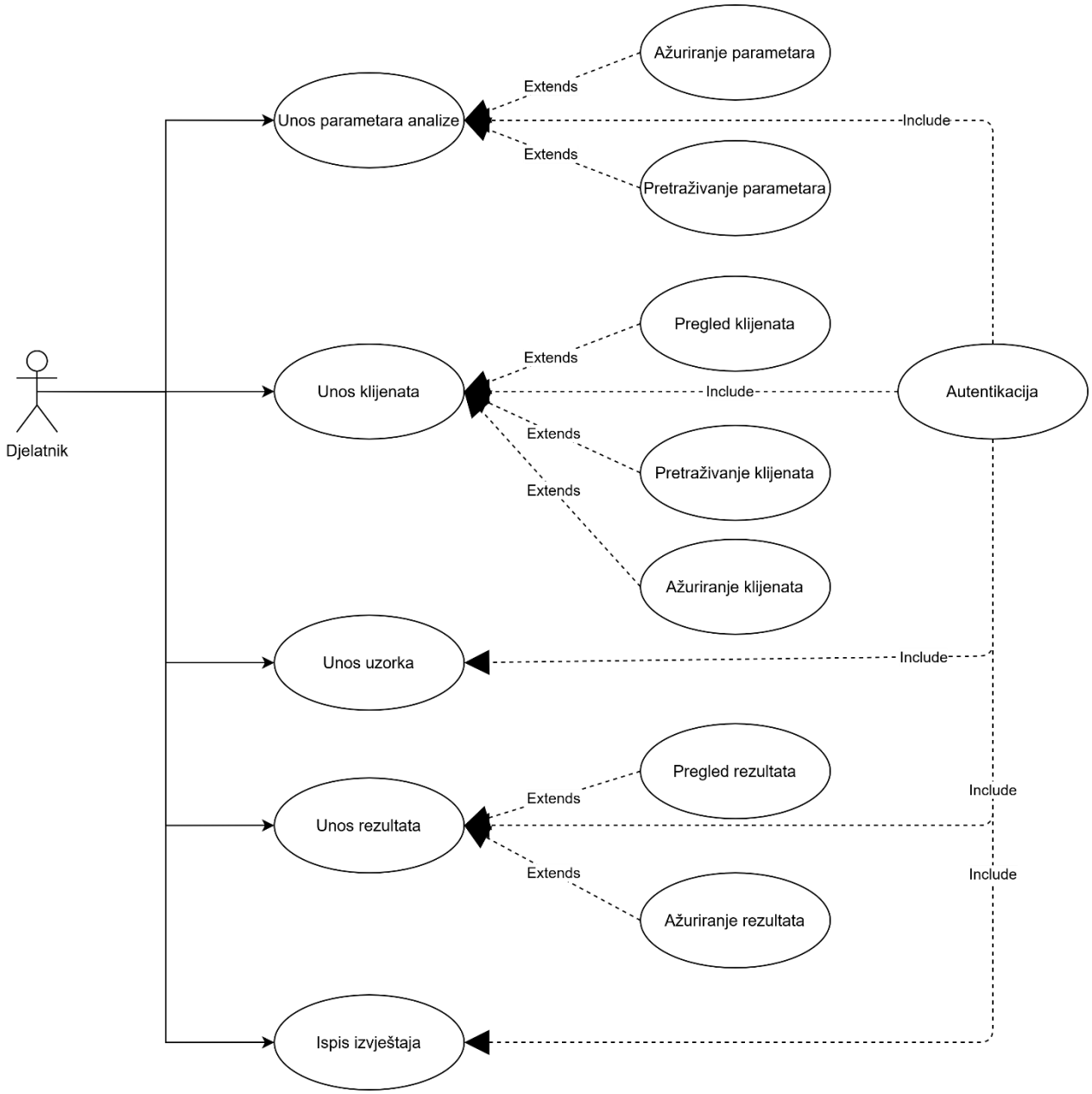



Izvor: autor

Djelatnik:

- Unos parametara analiza
- Ažuriranje parametara analiza
- Pretraživanje parametara analiza
- Unos klijenata
- Pregled klijenata
- Ažuriranje klijenata
- Pretraživanje klijenata
- Unos uzorka
- Unos rezultata
- Pregled uzoraka
- Pretraživanje uzoraka
- Pregled rezultata
- Ažuriranje rezultata
- Izrada/ispis analitičkog izvješća

*Slika 2 Dijagram korištenja za ulogu djelatnik*




Izvor: autor

## 3.2 Scenariji korištenja sustava

### 3.2.1 Autentifikacija

Nakon pokretanja aplikacije otvara se glavni prozor gdje korisnik vrši prijavu upisom korisničkog imena i lozinke, te odabirom vrste korisnika – Voditelj/administrator ili djelatnik. Aplikacija je postavljena s administratorskim računom i lozinkom koji se mogu promijeniti nakon ulaska u sustav. Postupak je isti za obje uloge.

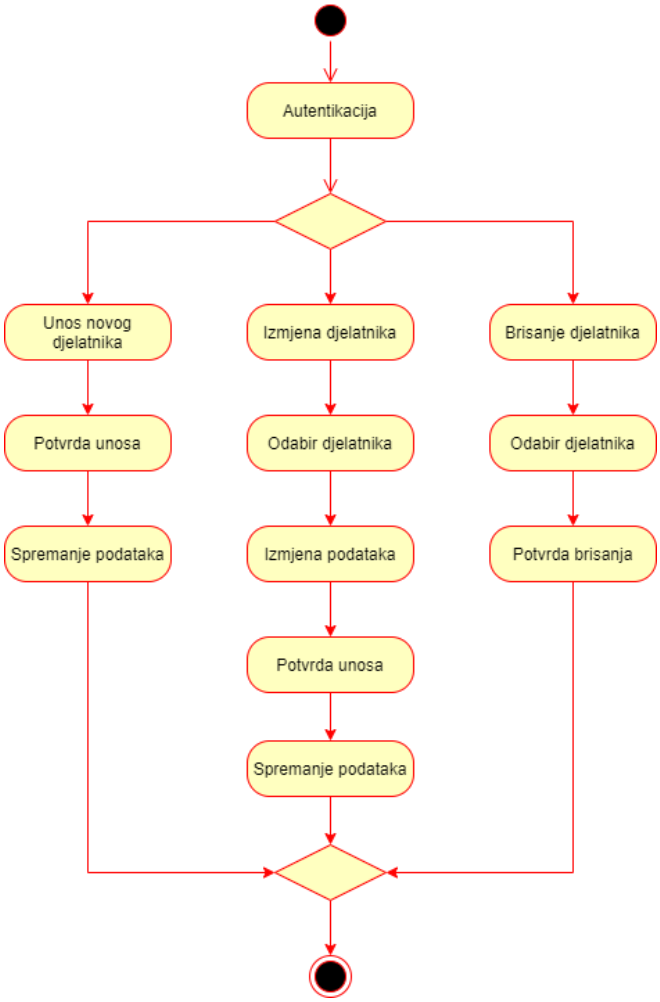
*Slika 3 Dijagram aktivnosti autentifikacije*


Izvor: autor

### 3.2.2 Unos, izmjena i brisanje djelatnika

Ova aktivnost je moguća samo kroz ulogu voditelja. Nakon uspješne autentifikacije kao voditelj, moguće je unijeti novog djelatnika, izmijeniti postojećeg ili obrisati postojećeg. Za unos novog je potrebno unijeti ime i prezime osobe i OIB, te joj dodijeliti korisničko ime i lozinku za ulazak u sustav. Sustav sam generira šifru djelatnika koja se koristi u daljnjem radu. Za izmjenu podataka potrebno je odabrati djelatnika kojega želimo izmijeniti, upisati nove podatke te spremiti izmjene. Brisanje djelatnika se vrši odabirom djelatnika i potvrdom akcije brisanja.

*Slika 4 Dijagram aktivnosti za unos novog djelatnika*

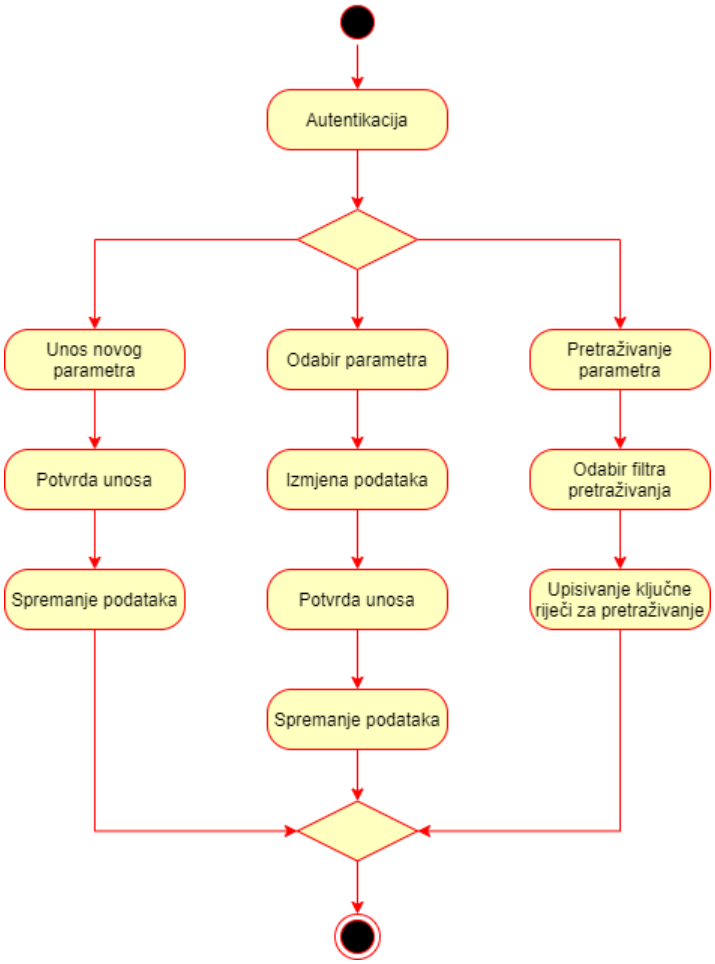



Izvor: autor

### 3.2.3 Unos, ažuriranje i pretraživanje parametara analize

Ova aktivnost je moguća kroz uloge voditelja i djelatnika. Nakon uspješne autentifikacije kao voditelj ili djelatnik, moguće je unijeti novi parametar, izmijeniti postojeći te pretraživati postojeće. Za unos novog je potrebno unijeti naziv parametra, mjernu jedinicu, maksimalnu dozvoljenu vrijednost. Sustav sam generira šifru parametra koja se koristi u daljnjem radu. Za izmjenu podataka potrebno je parametar koji želimo izmijeniti, upisati nove podatke te spremiti izmjene. Pretraživanje se vrši tako da se odabere filtar za pretraživanje (šifra, naziv, mjerna jedinica ili MDK) te se upisuje upit za pretraživanje.

*Slika 5 Dijagram aktivnosti za unos novog parametra*

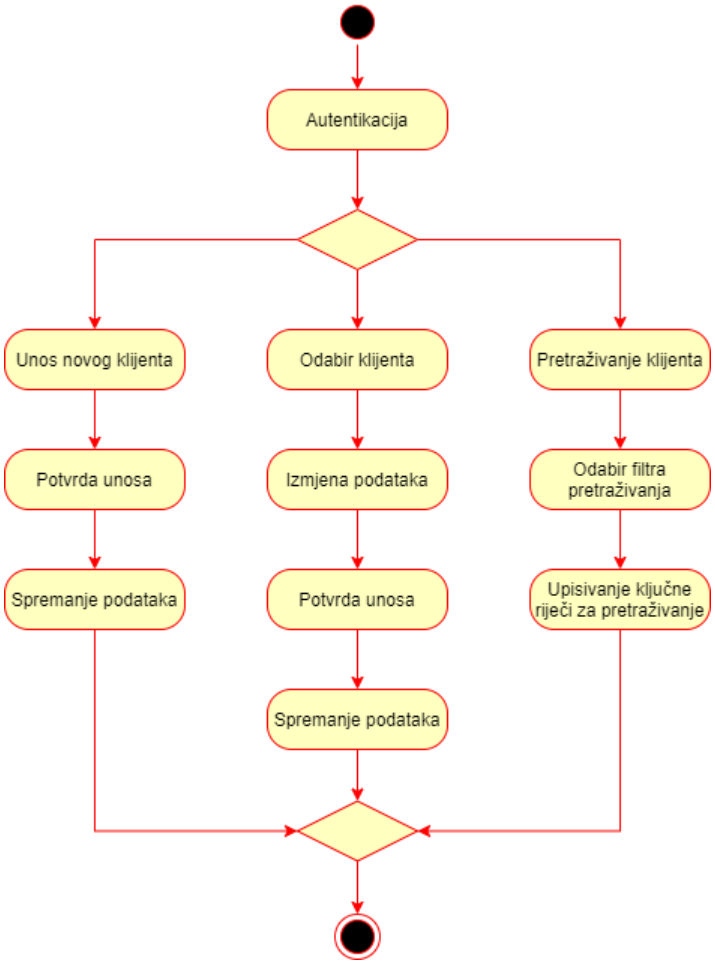



Izvor: autor

### 3.2.4 Unos, ažuriranje i pretraživanje klijenata

Ova aktivnost je moguća kroz uloge voditelja i djelatnika. Nakon uspješne autentifikacije kao voditelj ili djelatnik, moguće je unijeti novog klijenta, izmijeniti postojećeg te pretraživati postojeće. Za unos novog je potrebno unijeti naziv klijenta, adresu, poštanski broj, grad i OIB. Sustav sam generira šifru klijenta koja se koristi u daljnjem radu. Za izmjenu podataka potrebno je odabrati klijenta kojega želimo izmijeniti, upisati nove podatke te spremiti izmjene. Pretraživanje se vrši tako da se odabere filtar za pretraživanje (šifra, naziv, adresu, poštanski broj, grad ili OIB) te se upisuje upit za pretraživanje.

*Slika 6 Dijagram aktivnosti za unos novog klijenta*

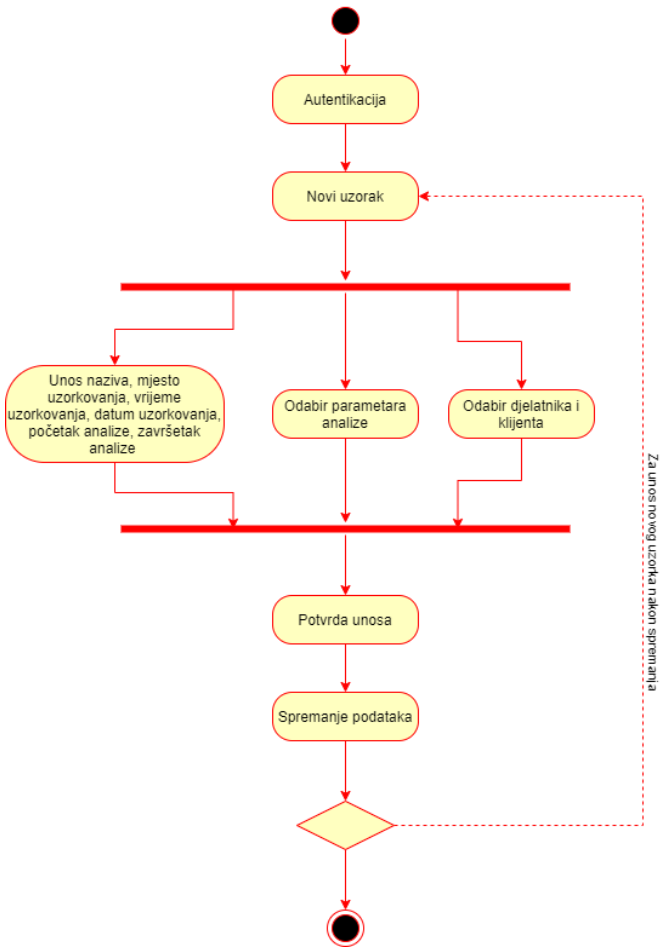



Izvor: autor

### 3.2.5 Unos uzorka

Ova aktivnost je moguća kroz uloge voditelja i djelatnika. Nakon uspješne autentifikacije kao voditelj ili djelatnik, moguće je unijeti novi uzorak. Za unos novog je potrebno unijeti naziv uzorka, mjesto uzorkovanja, datum uzorkovanja, vrijeme uzorkovanja, datum početka analize te datum završetka analize. Sustav sam generira šifru klijenta koja se koristi u daljnjem radu. Također ovdje se odabiru parametri koji će se analizirati u uzorku. Na kraju se odabire klijent čiji je uzorak te djelatnik koji je izvršio uzorkovanje. Ukoliko se unosi više uzorka, nakon spremanja uzorka, može se unijeti novi uzorak sa novom šifrom.

*Slika 7 Dijagram aktivnosti za unos uzorka*

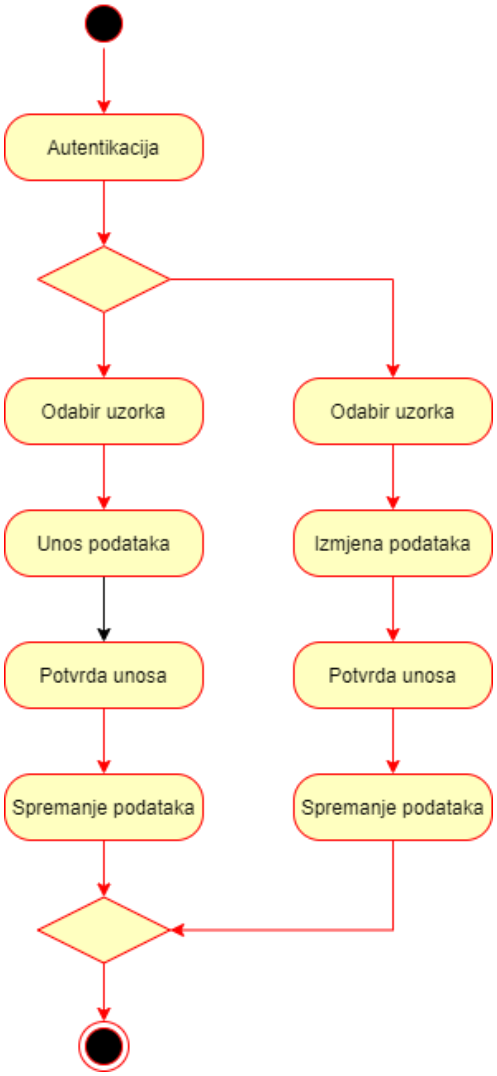



Izvor: autor

### 3.2.6 Unos rezultata

Ova aktivnost je moguća kroz uloge voditelja i djelatnika. Nakon uspješne autentifikacije kao voditelj ili djelatnik, moguće je unijeti rezultate za odabrani uzorak ili izmijeniti postojeće podatke za odabrani uzorak. Za unos novog je potrebno odabrati uzorak, nakon čeka sustav prikazuje samo parametre za koje je potrebno unijeti rezultate analize. Za izmjenu podataka potrebno je odabrati uzorak kojega želimo izmijeniti, upisati nove podatke te spremiti izmjene.

*Slika 8 Dijagram aktivnosti za unos rezultata*

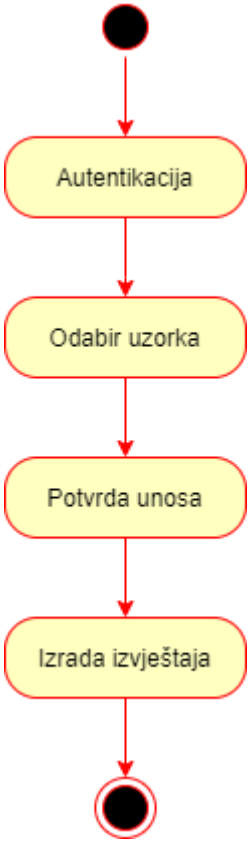



Izvor: autor

### 3.2.7 Izrada izvješća

Ova aktivnost je moguća kroz uloge voditelja i djelatnika. Nakon uspješne autentifikacije kao voditelj ili djelatnik, moguće je odabrati uzorak te preko sistemskog izbornika ispisati izvješće u pdf oblik ili direktno na papir. Prije ispisa izvještaja moguće ga je pregledati.

*Slika 9 Dijagram aktivnosti za ispis izvješća*




Izvor: autor

# 4. DIZAJN SUSTAVA

## 4.1 Odabir arhitekture sustava

Arhitektura sustava opisuje njegove glavne komponente, njihove odnose (strukture) i način međusobne interakcije, te služi kao njegov nacrt. U objektno orijentiranom dizajnu radi se dekompozicija sustava particioniranjem u podsustave koji su odgovorni za pojedinu funkcionalnost, a podsustavi se grupiraju u srodne usluge, koje se često koriste zajedno.

Na softverskom nivou najčešće korišteni način dekomponiranja je onaj korištenjem MVC uzoraka dizajna (engl. Model-View-Controller), te je taj model korišten i za izradu aplikacije LabReport. MVC dizajn podrazumijeva razvijanje podsustava koji se klasificiraju u 3 skupine:

- podsustav **modela** - predstavlja podatke i pravila koja pristupaju i ažuriraju te podatke,
- podsustav **pogleda** – pogled upravlja grafičkim izgledom, područjem prikaza i odgovoran je za prezentaciju podataka korisniku, korištenjem grafike i teksta,
- podsustav **kontrole** - prevodi korisničke interakcije s pogledom u akcije koje će model izvesti. (Davidović, 2016)

Prilikom izrade aplikacije korišten je NetBeans IDE, aplikacija je pisana u Java programskom jeziku, te su korištene dodatne biblioteke za moderniji izgled aplikacije i bolju funkcionalnost. Također korištena je MySQL baza podataka, kojom se upravlja iz MySQL Workbench aplikacije dok je veza između LabReport aplikacije i baze podataka standardni Java konektor (JDBC, 2020).

### 4.1.1 Java

Java je objektno orijentirani programski jezik, koji se može izvršavati na bilo kojoj platformi koja ga podržava bez potrebe za ponovnim kompajliranjem. To se postiže pretvaranjem svog koda napisanog u javi u bytecode koji se izvršava na Java Virtual Machine, neovisno o OS-u na kojem se nalazi. Java se temelji na C i C++ programskim jezicima, tako da ima i njima sličnu sintaksu. Razvoj je započeo u tvrtki Sun Microsystems 1991. pod vodstvom Jamesa Goslinga, a prva verzija je izašla 1995. (Wikipedia, 2014)

### 4.1.2 NetBeans IDE

NetBeans IDE je besplatno, open source, integrirano razvojno okruženje (IDE) koje omogućuje razvoj stolnih, mobilnih i web aplikacija. NetBeans IDE podržava razvoj aplikacija u raznim programskim jezicima, uključujući Java, HTML5, PHP i C ++. NetBeans IDE pruža integriranu podršku za cijeli razvojni ciklus, od stvaranja projekata do uklanjanja pogrešaka, profiliranja i implementacije. NetBeans IDE se izvodi na Windows, Linux, Mac OS X i drugim sustavima koji se temelje na UNIX-u. (Bai, 2011)

### 4.1.3 FlatLaf

FlatLaf je moderan i open-source izgled (Look and feel) za Java Swing desktop aplikacije. Korišten je u aplikaciji kako bi zamijenio izgled standardnih Swing kontrola (gumbi, padajući izbornici), te dao aplikaciji moderan izgled. (FlatLaf - Flat Look and Feel | FormDev, 2020)

### 4.1.4 JCalendar

JCalendar je Java klasa (bean) za grafički odabir datuma. JCalendar se sastoji od nekoliko različitih komponenti - JDayChooser, JMonthChooser i JYearChooser. Komponente se jednostavno mogu modificirati tako da se uklope u aplikaciju. Ovaj dodatak je freeware, odnosno besplatan za upotrebu. (JCalendar, 2020)

### 4.1.5 SQL

SQL (Structured Query Language) je jezik za upravljanje podacima koji se nalaze u sustavu upravljanja relacijskim bazama podataka (RDBMS). Koristi se za rukovanje strukturiranim podacima, tj. podacima koji uključuju odnose između entiteta i varijabli. SQL izrazi se koriste za izvršavanje zadataka kao što su ažuriranje podataka u bazi podataka ili za preuzimanje podataka iz baze podataka. Razvijen je 1970.g u IBM, a najnovije izdanje je iz 2016.g. (MySQL : MySQL 8.0 Reference Manual : 1.3.1 What is MySQL?,2020)

### 4.1.6 MySQL Server

MySQL je open source sustav upravljanja relacijskim bazama podataka (RDBMS) s modelom klijent-poslužitelj. Relacijska baza podataka organizira podatke u jednu ili više tablica podataka u kojima se tipovi podataka mogu međusobno povezati. SQL je jezik koji se koristi za stvaranje, izmjenu i izdvajanje podataka iz relacijske baze podataka, kao i za kontrolu pristupa korisniku baze podataka. Razvija ga tvrtka Oracle. (MySQL : MySQL 8.0 Reference Manual : 1.3.1 What is MySQL?, 2020)

### 4.1.7 MySQL Connector/J

Java Database Connectivity (JDBC) je aplikacijsko programsko sučelje (API) za programski jezik Java, koje definira kako klijent može pristupiti bazi podataka. Connector / J implementira Java Database Connectivity (JDBC) API, kao i njegova proširenja. MySQL Connector/J je JDBC Type 4 upravljački program (driver), koji implementira JDBC 4.2 specifikacije. Oznaka tipa 4 znači da je upravljački program čista Java implementacija MySQL protokola i da se ne oslanja na MySQL knjižnice klijenata. (Oracle, 2020)

### 4.1.8 MySQL Workbench

MySQL Workbench je alat koji u sebi integrira modeliranje podataka, razvoj SQL-a te alate za administraciju i konfiguraciju poslužitelja, administraciju korisnika, izradu sigurnosnih kopija i drugog. MySQL Workbench dostupan je na Windowsima, Linuxu i Mac OS X. (Oracle, 2020)

## 4.2 UML dijagram klasa

Dijagram klasa opisuje statičku strukturu sustava a sastoje se od notacija klasa i sučelja te veza među njima. Dijagram klasa ne koristi se samo za vizualizaciju, opisivanje i dokumentiranje različitih aspekata sustava, već i za konstrukciju izvršnog koda aplikacije. Grafički prikaz dijagram klasa se sastoji od naziva klase, liste atributa, liste metoda te popisa unutarnjih klasa. Dijagramom klasa opisujemo objekte iz stvarnog svijeta ali i veze između njih. Asocijacija je najvažnija veza korištena u dijagramu klasa, a označava sposobnost objekta da pošalje poruku drugom objektu.

Na dijagramu klasa aplikacije „LabReport“ se nalaze klase koje možemo podijeliti na klase modela (*djelatnik, Rezultat, Uzorak, Parametar, autentikacija, Klijent i Izvjesce*) te klase kontrolera (*DBaseControll*). Klase modela su povezane asocijacijom sa klasama kontrolera, njihova kardinalnost je 1..1, što znači da jedan objekt iz jedne klase je povezan s jednim objektom u drugoj klasi. Klase modela primaju podatke o djelatniku (*djelatnik*), rezultatu (*Rezultat*), uzorku (*Uzorak*), parametrima (*Parametar*), klijentima (*Klijent*), izvješću (*Izvjesce*) i autentikaciji (*autentikacija)* te instanciraju objekte prilikom njihovog poziva. Klasa *DBaseControll* sadrži metode koje služe komunikaciji sa bazom podatka – spremanje i dohvaćanje podataka iz baze. Primjer takve metode je `spremiDjelatnikaUBazu` koja omogućuje spremanje podataka o djelatniku u bazu, u tablicu „djelatnik“.

*Slika 10 Dijagram klasa*


Izvor: autor

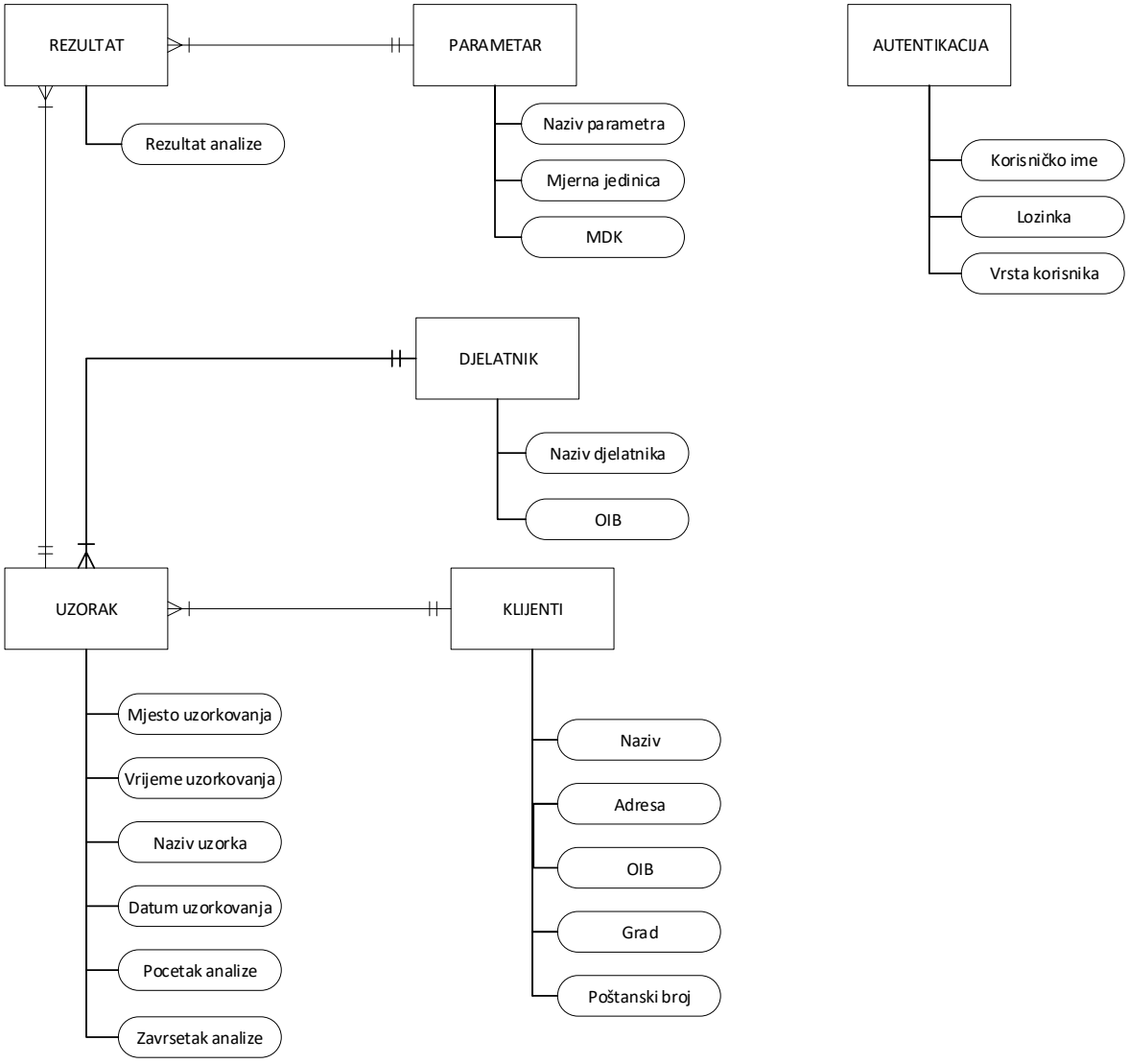
## 4.3 Modeliranje podataka

Modeliranje podataka je proces kojim se opisuju skupovi podataka i njihove međusobne veze zasnovan na nekoj metodi. Modeliranje je proces razvoja modela. Model nastaje procesom apstrakcije u kome se prvo biraju relevantni koncepti koje reprezentacija treba sadržati, a zatim se svakome konceptu pridružuju relevantne osobine (atributi) koje se žele prikazati u okviru modela. (Pavlić, 2011)

Model podataka je skup pravila koja određuju kako sve može izgledati logička struktura baze podataka. Model čini osnovu za projektiranje i implementiranje baze, odnosno podaci u bazi moraju biti logički organizirani u skladu s modelom koji podržava odabrani DBMS. Model se izražava grafičkim prikazom pomoću dogovorene notacije. Model podataka sustava je pojednostavljena reprezentacija o relevantnim karakteristikama sustava preko skupa entiteta (objekata), veza među entitetima i atributa entiteta i agregacija entiteta. (Pavlić, 2011)

Model entiteti-veze koristi se za simboličan, konceptualni opis podataka (model podataka), te je lako razumljiv svima (korisniku i izrađivaču aplikacije) i služi za izgradnju relacijskog modela. Metoda entiteti - veze je grafički prikaz međusobno povezanih grupa podataka promatranoga sustava. Metoda entiteti - veze je semantički bogata metoda za modeliranje podataka jer raspolaže ljudski bliskim konceptima te se odlikuje prirodnošću opisa a njezini koncepti su bliski korisniku, pa je shema modela podataka laka za razumijevanje i komunikaciju korisnika i projektanta. (Pavlić, 2011)

*Slika 11 Model entiteti-veze*



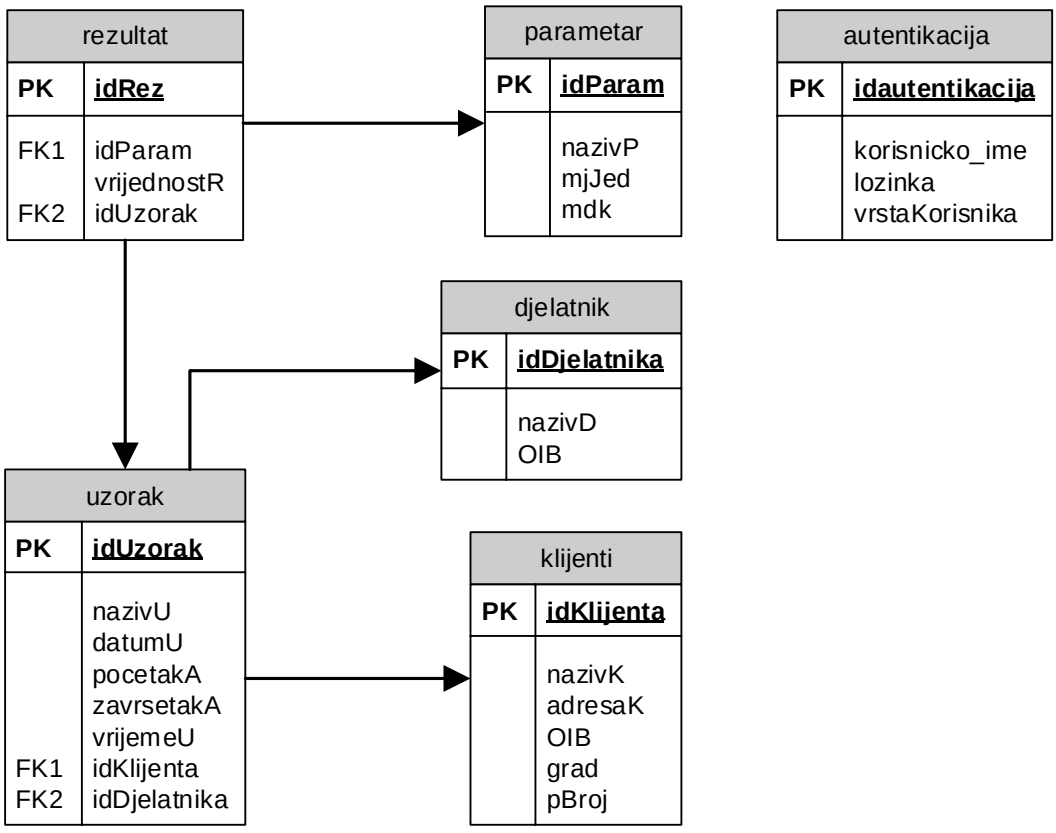

Izvor: autor

Iz dijagrama se vidi da tablica „djelatnik“ mora u sebi sadržavati ime djelatnika i OIB djelatnika, te da djelatnik može biti povezan sa jednim ili više uzoraka. Tablica „klijenti“ mora sadržavati naziv klijenta, adresu klijenta, grad, poštanski broj te OIB, a klijent je povezan sa jednim ili više uzoraka. Tablica „uzorak“ mora sadržavati podatke o mjestu i vremenu uzorkovanja, naziv uzroka, datum uzorkovanja, vrijeme početka i završetka analize, a uzorak može imati jedan ili više rezultata analize. Tablica „rezultat“ sprema podatke za rezultate analize, a može biti povezana sa jednim i samo jednim parametrom. Tablica „parametar“ spreman podatke o nazivu parametra, mjernoj jedinici i maksimalno dozvoljenoj koncentraciji (MDK), te može biti povezan sa jednim ili više rezultata. Tablica „autentikacija“ sprema podatke o korisničkom imenu, lozinki i vrsti korisnika, no nije direktno povezana sa ostalim tablicama, jer nam ona služi samo za ulaz u sustav.

## 4.4 Relacijski model

Relacijski model bio je teoretski zasnovan još krajem 60-tih godina 20. stoljeća, u radovima Edgara Codda. Relacijska shema manje je razumljiva korisnicima od konceptualne, jer su u njoj i entiteti i veze među entitetima pretvoreni u relacije pa je teško razlikovati jedno od drugog. Ipak, važno svojstvo relacijske sheme je da se ona može više-manje izravno implementirati pomoću današnjih DBMS-a. Relacijski je model prikaz svih tablica iz baze podataka koji koristi pripadnu relacijsku algebru koja djeluje korištenjem relacijskih operatora. Svaka se tablica u tom modelu naziva relacija i njezin je cilj pokazati svaku relaciju u obliku potpune ne ponavljajuće tablice kako bi se postigla normalizacija. (Manger, 2010)

*Slika 12 Relacijski model*



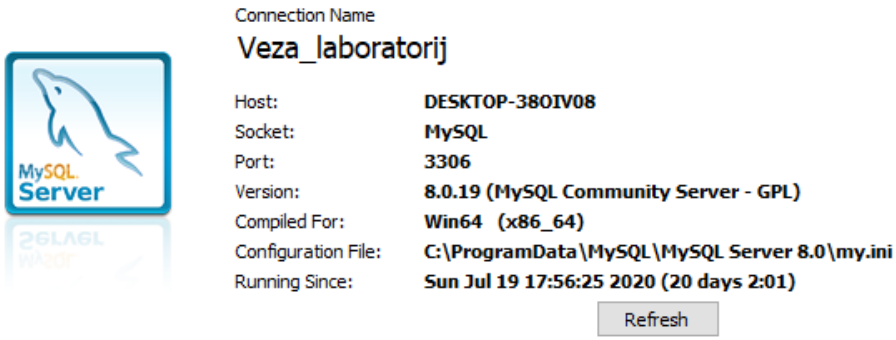

Izvor: autor

# 5. IMPLEMENTACIJA SUSTAVA

## 5.1 Baza podataka

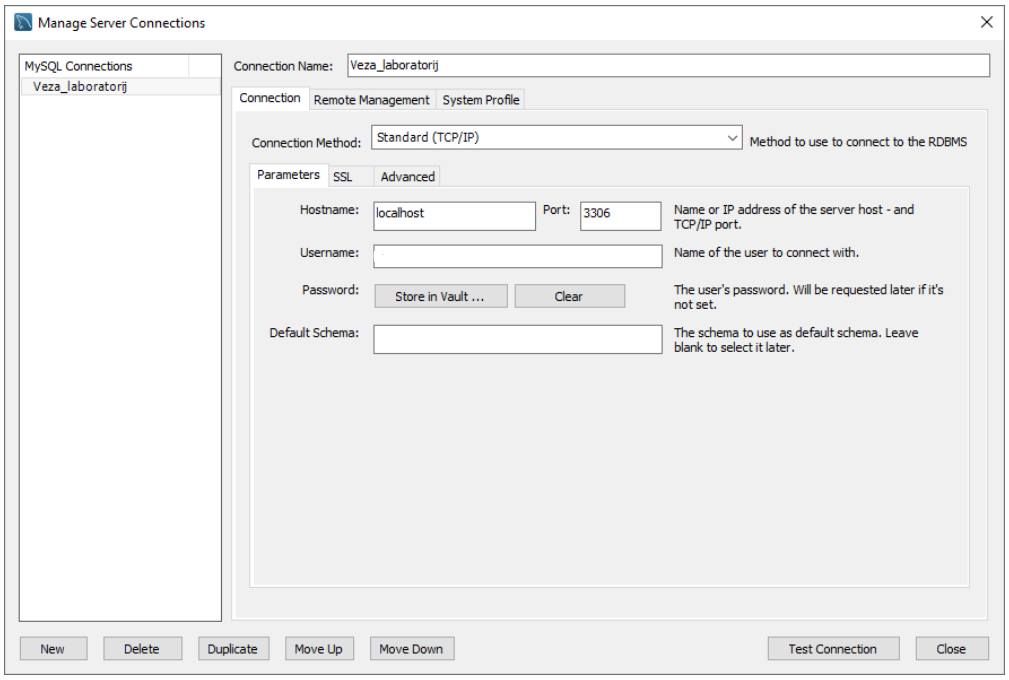
Aplikacija koristi MySQL verziju 8.0.19. pokrenutu na lokalnom računalu, te se povezuje preko „Veza_laboratorij“ konekcije, prikazano na slici 12 i 13.

*Slika 13 Status i verzija servera*




Izvor: autor

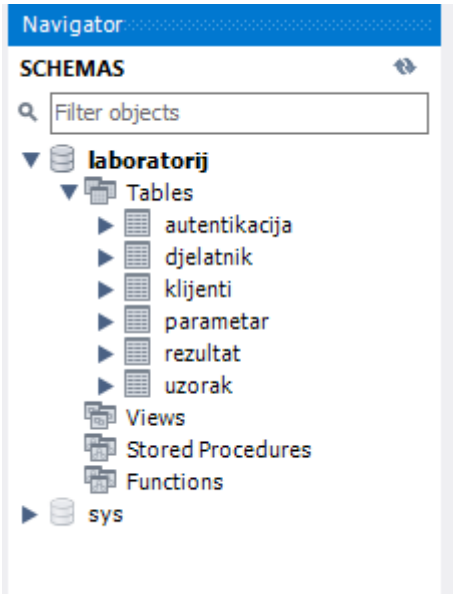
*Slika 14 Konekcija s informacijama za spajanje*




Izvor: autor

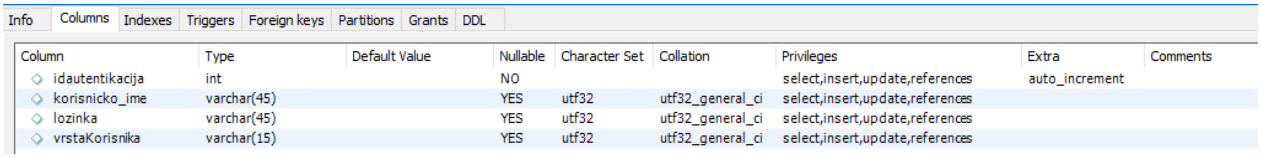
Baza podataka se nalazi u shemi „laboratorij“ (slika 14), a svaka tablica sadrži kolumne i primarne i vanjske ključeve koji su opisani u prethodnim poglavljima (slika 15).

*Slika 15 Shema baze podataka „laboratorij“*




Izvor: autor

*Slika 16 Tablica „autentikacija“*

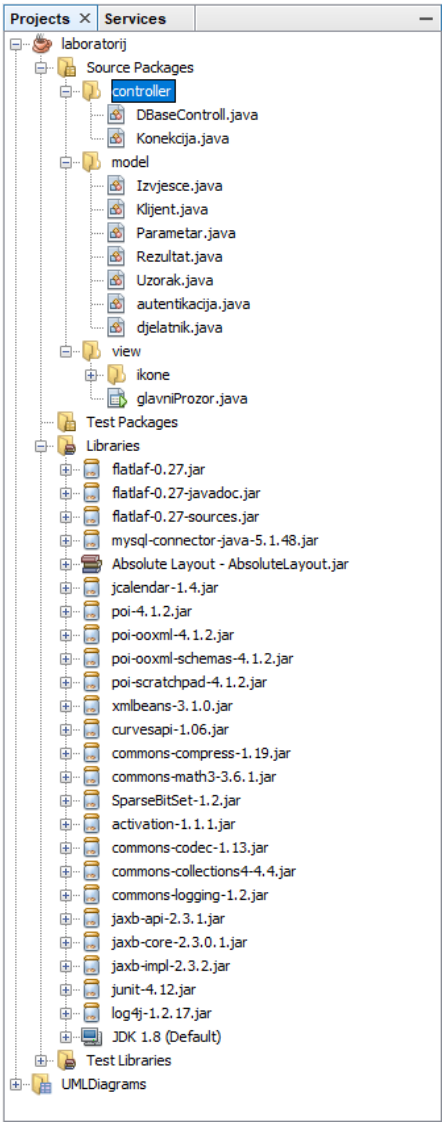



Izvor: autor

## 5.2 Java izvorni kod

U ovom poglavlju je opisan programski kod aplikacije. Cijela aplikacija je izrađena u NetBeans razvojnom okruženju, naziv projekta je „laboratorij“, a aplikacija prati MVC načelo te je podijeljena na tri dijela – MODEL, VIEW i CONTROLLER koji će biti opisani u narednim poglavljima. U aplikaciji se koriste i dodatne biblioteke koje su opisane u poglavlju 2.

*Slika 17 Projekt "laboratorij" u NetBeans-u*




Izvor: autor

### 5.2.1 Model

Model je kompletna reprezentacija objekta koji se koristi u aplikaciji, samostalan je i njegova reprezentacija bi trebala biti neovisna od ostatka programa. Model sadrži metode (setteri i getteri) koje omogućuju vanjskom svijetu da mu pristupe, naprave izmjenu, te dohvate izmijenjene objekte. Također u modelu se nalaze konstruktori koji instanciraju objekte prilikom njihova pozivanja. Aplikacija LabReport koristi više različitih modela, ovisno o podacima koji će se dohvaćati i spremati.

### 5.2.2 Controller

Controller je odgovoran za prevođenje korisničkih akcija mišem i tipkovnicom u upute modelu i pogledu (view). Unutar controllera se nalaze metode koje se pozivaju kada korisnik želi spremiti nove podatke u bazu podataka ili dohvatiti podatke da bi se prikazali u sučelju. U nastavku će biti prikazane neke metode unutar klase DBaseControll koje se koriste za spremanje, dohvaćanje, izmjenu i brisanje podataka iz baze podataka. Metoda za spajanje na bazu podataka je izdvojena u zasebnu klasu, te je pozivana po potrebi u metodama unutar klase DBaseControll.

Klasa „konekcija“ se sastoji od varijabli za korisničke podatke tipa `String`, te od `Connection` tipa varijable koja predstavlja objekt za povezivanje sa bazom. Klasa `DriverManager` prilikom inicijalizacije učitava drivere iz klase `jdbc.drivers`, te se koristi metoda `getConnection` za uspostavljanje veze sa bazom.

Programski kod metode „konekcija“ za spajanje na bazu:

```
public class Konekcija {
    private static String url = "localhost";
    private static String baza = "laboratorij";
    private static String user1 = "****";
    private static String pass1 = "**";
    private static Connection conn = null;
    public static Connection konekcija(){
try {
Class.forName("com.mysql.jdbc.Driver").newInstance();
conn=DriverManager.getConnection("jdbc:mysql://"+url+"/"+baza,user1,pass1);
}catch(ClassNotFoundException | InstantiationException |
IllegalAccessException | SQLException ex)
{Logger.getLogger(Konekcija.class.getName()).log(Level.SEVERE, null, ex);
}return conn;}}
```

Nakon povezivanja s bazom podataka, potrebno je poslati upit prema bazi što se izvodi pomoću *Statment* sučelja koje predstavlja SQL upit. U dolje navedenom programskom kodu je vidljivo da se koristi preparedStatment tip Statment objekta, koji može sadržavati korisnički unos u bazu podataka. Koristi se standardni SQL upit za unos podataka u batu pomoću naredbe INSERT. Statment se izvršava naredbom execute.

Programski kod metode „spremiDjelatnika“ za spremanje podataka o djelatniku u bazu:

```
public static void spremiDjelatnikaUBazu(djelatnik djelatnik) {
  Connection conn = null;
  PreparedStatement stmt;

try {
  conn = Konekcija.konekcija();
  stmt = conn.prepareStatement("INSERT INTO `laboratorij`.`djelatnik`
(`nazivD`, `OIB`)" + "VALUES (?,?);");
  stmt.setString(1, djelatnik.getNaziv());
  stmt.setString(2, String.valueOf(djelatnik.getOIB()));
  stmt.execute();
} catch (SQLException ex) {
 Logger.getLogger(DBaseControll.class.getName()).log(Level.SEVERE, null, ex);
 }
 }
```

Za dohvaćanje određenih podataka iz baze potrebno je izvršiti SQL upit `SELECT * FROM`, također pomoću `Statment` sučelja. Dohvaćeni podaci se spremaju u `ResultSet` objekt, koji je tablica sa podacima iz baze. Za kretanje kroz podatke u bazi se koristi pokazivač koji se pomiče sa metodom `next`. U dolje navedenoj metodi se dohvaćeni podaci spremaju u `ArrayList` tipa „djelatnik“.

Programski kod metode „ucitajDjelatnika“ za dohvaćanje podataka o djelatniku iz baze podataka:

```
public static ArrayList<djelatnik> ucitajDjelatnika() throws SQLException,
ClassNotFoundException, InstantiationException, IllegalAccessException {
ArrayList<djelatnik> listaDjelatnika = new ArrayList<djelatnik>();
  Connection conn = null;
  PreparedStatement stmt;
  conn = Konekcija.konekcija();
  stmt = conn.prepareStatement("SELECT * FROM djelatnik");
  ResultSet rs = stmt.executeQuery();
  while (rs.next()) {
     int sifra = rs.getInt("idDjelatnika");
     String naziv = rs.getString("nazivD");
     String oib = rs.getString("OIB");

      djelatnik djelatnik = new djelatnik();
      djelatnik.setSifraD(sifra);
      djelatnik.setNaziv(naziv);
      djelatnik.setOIB(oib);
       listaDjelatnika.add(djelatnik);
      }
    return listaDjelatnika;
    }
```

Prilikom izmjene podataka u bazi, koristi se SQL naredba `UPDATE`, dok je sve drugo isto kao i ostalim metodama za povezivanje s bazom.

Programski kod metode „izmjeniDjelatnika“ za izmjenu podataka o djelatniku:

```
public static void izmjeniDjelatnikaUBazi(djelatnik djelatnik) {
   Connection conn = null;
   PreparedStatement stmt;
   conn = Konekcija.konekcija();
try {
stmt=conn.prepareStatement("UPDATE `laboratorij`.`djelatnik` SET `nazivD`=?,
`OIB`=? WHERE `idDjelatnika`=?;");

    stmt.setString(1, djelatnik.getNaziv());
    stmt.setString(2, String.valueOf(djelatnik.getOIB()));
    stmt.setString(3, String.valueOf(djelatnik.getSifraD()));
    stmt.executeUpdate();
    conn.close();
} catch (SQLException ex) {
Logger.getLogger(DBaseControll.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

U metodi za brisanje djelatnika iz baze se koristi SQL naredba `DELETE FROM`.
Ovdje je autor naišao na problem sa dohvaćanjem šifre djelatnika kojeg je potrebno obrisati iz baze, što je riješeno unutar programskog koda za gumb Obriši djelatnika:

```
int sifra = (int) tableModel.getValueAt(tblDjelatnikNovi.getSelectedRow(), 0);
```

Šifra se dobiva iz prvog stupca označenog retka u tablici djelatnik, te se prosljeđuje u metodu „izbrisiDjelatnikaUBazi“:

```
djelatnik Djelatnik = new djelatnik();
Djelatnik.setSifraD(sifra);
```

Također, dodan je skočni prozor koji korisnika obavještava da se neki djelatnik ne može obrisati iz baze jer se njegova šifra još uvijek koristi:

```
Cannot delete or update a parent row: a foreign key constraint fails
```

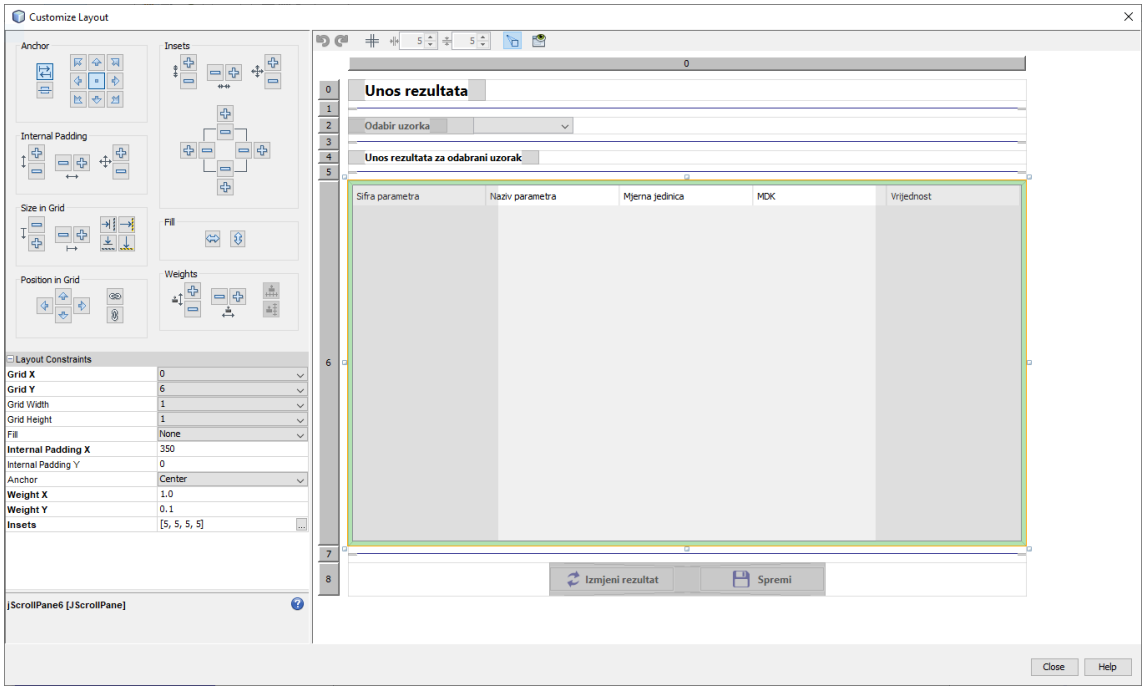
Programski kod metode „izbrisiDjelatnikaUBazi“ za brisanje podataka o djelatniku:

```
public static void izbrisiDjelatnikaUBazi(djelatnik djelatnik){
      Connection conn = null;
      PreparedStatement stmt;
      conn = Konekcija.konekcija();
try {
      stmt = conn.prepareStatement("DELETE FROM `laboratorij`.`djelatnik`
WHERE `idDjelatnika`=?;");
      stmt.setString(1,String.valueOf(djelatnik.getSifraD()));
      stmt.executeUpdate();
      conn.close();
} catch (SQLException  ex) {
if(ex instanceof SQLIntegrityConstraintViolationException )
{
JOptionPane.showMessageDialog(null, "Nije moguće obrisati djelatnika jer se
njegova šifra \n koristi na postojećim izvještajima!!\n"
+ "(Cannot delete or update a parent row: a foreign key constraint fails)");
System.out.println(ex);
};
}}
```

### 5.2.3 View

View upravlja grafičkim izgledom, područjem prikaza i odgovoran je za prezentaciju podataka korisniku, korištenjem grafike i teksta. Specificira kako bi trebali podaci iz modela biti prikazani, te kada se model promijeni, view prilagođava i svoj prikaz. Kako bi se dobio ujednačen prikaz sučelja aplikacije, korišten je `GridBagLayout manager`, koji je najfleksibilniji ali i najkompleksniji način za upravljanje komponentama sučelja, što je predstavljalo izazov za autora. `GridBagLayout` smješta komponente u mrežu redaka i stupaca, omogućujući navedenim komponentama da obuhvaćaju više redaka ili stupaca. Nisu svi redovi nužno iste visine. Slično tome, nemaju svi stupci nužno jednaku širinu. U osnovi, `GridBagLayout` smješta komponente u pravokutnike u mrežu, a zatim koristi željene veličine komponenata kako bi odredio koje veličine ćelije trebaju biti. Nakon proučavanja dokumentacije, autor je odlučio da je najbolje rješenje koristiti ugrađeni *layout manager*, umjesto upisivanja koda za svaku komponentu, što se vidi na slici 18.

*Slika 18 GridBagLayout manager*




Izvor: autor

Za ispis dokumenata koristi se `PrinterJob` klasa, koja koristi `printDialog` metodu za pozivanje sistemskog izbornika za ispis (ovdje se može birati ispis na printer ili u pdf oblik). Također korisiti se i `PageFormat` klasa koja omogućuje odabir veličine stranice i njen položaj (*landscape ili portrait*). `Graphics2D` je temeljna klasa za prikazivanje dvodimenzionalnih oblika, teksta i slika na Java platformi, te su korištene njene `scale` i `translate` metode kako bi ispis izvještaja bio prilagođen ISO A4 stranici.

Programski kod metode za ispis izvješća:

```
private void btnIspisActionPerformed(java.awt.event.ActionEvent evt) {//GEN-
FIRST:event_btnIspisActionPerformed
PrinterJob ispis =PrinterJob.getPrinterJob();
ispis.setJobName("Ispis izvještaja");

   ispis.setPrintable(new Printable(){
   public int print(Graphics pg,PageFormat pf, int pageNum){
   pf.setOrientation(PageFormat.PORTRAIT);
   Paper papir =pf.getPaper();
   papir.setSize(595, 842);
   papir.setImageableArea(17, 15, papir.getHeight(), papir.getWidth());
   pf.setPaper(papir);

     if(pageNum>0){
     return Printable.NO_SUCH_PAGE;
     }

    Graphics2D g2 = (Graphics2D)pg;
    g2.translate(pf.getImageableX(), pf.getImageableY());
    g2.scale(1.20,1.20);
    panelIspis.paint(g2);
    return Printable.PAGE_EXISTS;
   }
  });
    boolean ok = ispis.printDialog();
    if(ok){
    try{ispis.print();
    }
    catch (PrinterException ex){}
}
```

Prilikom upisa novog uzorka potrebno mu je dodijeliti analitički broj, međutim baza podataka sadrži samo spremljene uzorke sa njihovim brojevima. Kako bi se riješio ovaj problem iskorištena je SQL naredba `SELECT AUTO_INCREMENT FROM INFORMATION_SCHEMA.TABLES.` `INFORMATION_SCHEMA` pruža pristup metapodacima baze podataka, informacijama o MySQL poslužitelju, poput naziva baze podataka ili tablice, vrste podataka stupca ili privilegija pristupa. Međutim prilikom korištenja ove naredbe broj koji se dodjeljuje `AUTO_INCREMENT` naredbom se povlači iz memorije uređaja, a nije zapisan u bazi podataka, pa je potrebno prije izvršavanja ove naredbe resetirati podatke i povući svježe iz baze. Za to se koristi naredba `ANALYZE TABLE` koja briše podatke tablice uzorak, te se oni ažuriraju prilikom slijedećeg pristupanja toj tablici. Kako se vidi u priloženom kodu, prvo se izvršava `PreparedStatement stmt1` a nakon toga se izvršava `PreparedStatement stmt` koja zapravo povlači podatke iz baze. (Thakur, 2018)

Programski kod za dohvaćanje slijedećeg broja uzorka:

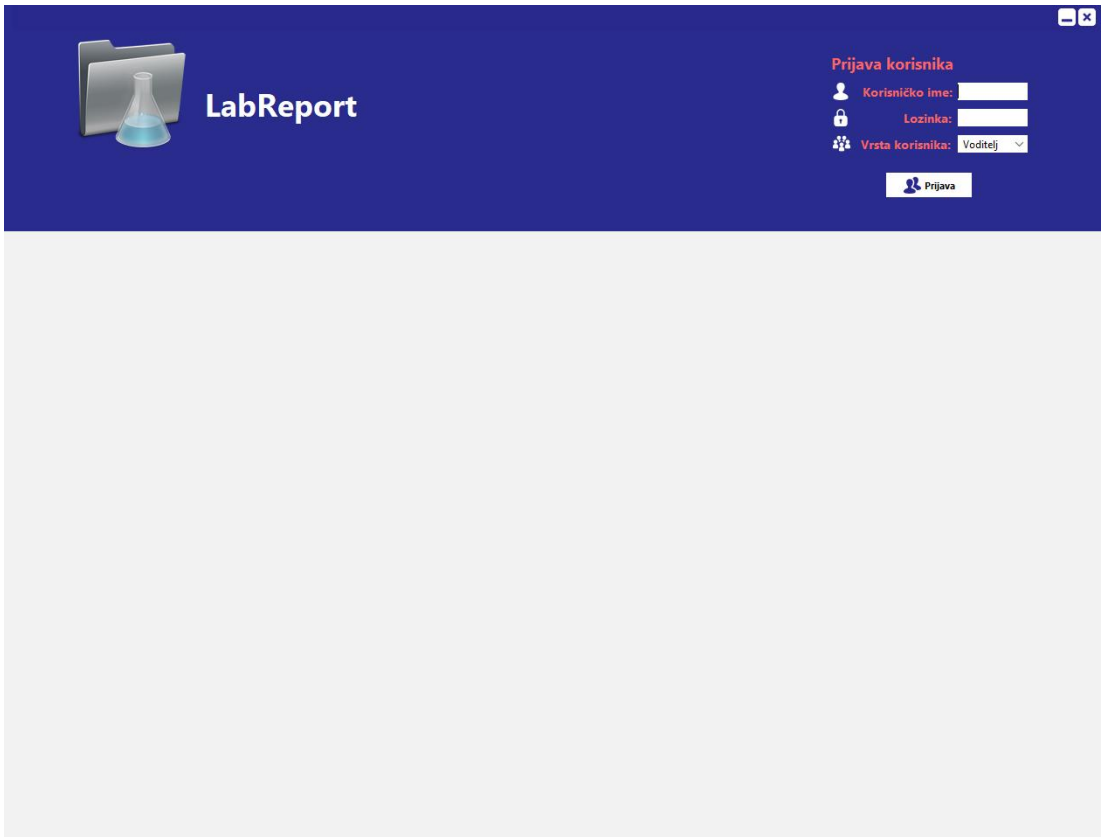
```
private void prikaziBrojU() {
 try {
    Connection conn = null;
    PreparedStatement stmt;
    PreparedStatement stmt1;
    conn = Konekcija.konekcija();
    stmt = conn.prepareStatement("SELECT AUTO_INCREMENT FROM
information_schema.TABLES WHERE TABLE_SCHEMA = \"laboratorij\" AND TABLE_NAME
= \"uzorak\";");
    stmt1 = conn.prepareStatement("Analyze table uzorak");
    stmt1.executeQuery();
    ResultSet rs = stmt.executeQuery();
        if (rs.next()) {
        int sifraU = rs.getInt(1);
        lblSifraUz.setText(String.valueOf(sifraU));
        }
   } catch (SQLException ex) {
Logger.getLogger(DBaseControll.class.getName()).log(Level.SEVERE, null, ex);
}}
```

# 6. KORIŠTENJE APLIKACIJE

## 6.1 Glavni prozor

Nakon pokretanja aplikacije otvara se glavni prozor na kojem su vidljiva dva dijela – gornji dio sa poljima za prijavu u sustav, te donji dio na kojem se prikazuje radni dio aplikacije nakon prijave. Sama prijava se vrši upisom korisničkog imena i lozinke, te odabirom vrste korisnika – Voditelj/administrator ili djelatnik.

*Slika 19 Glavni prozor aplikacije*



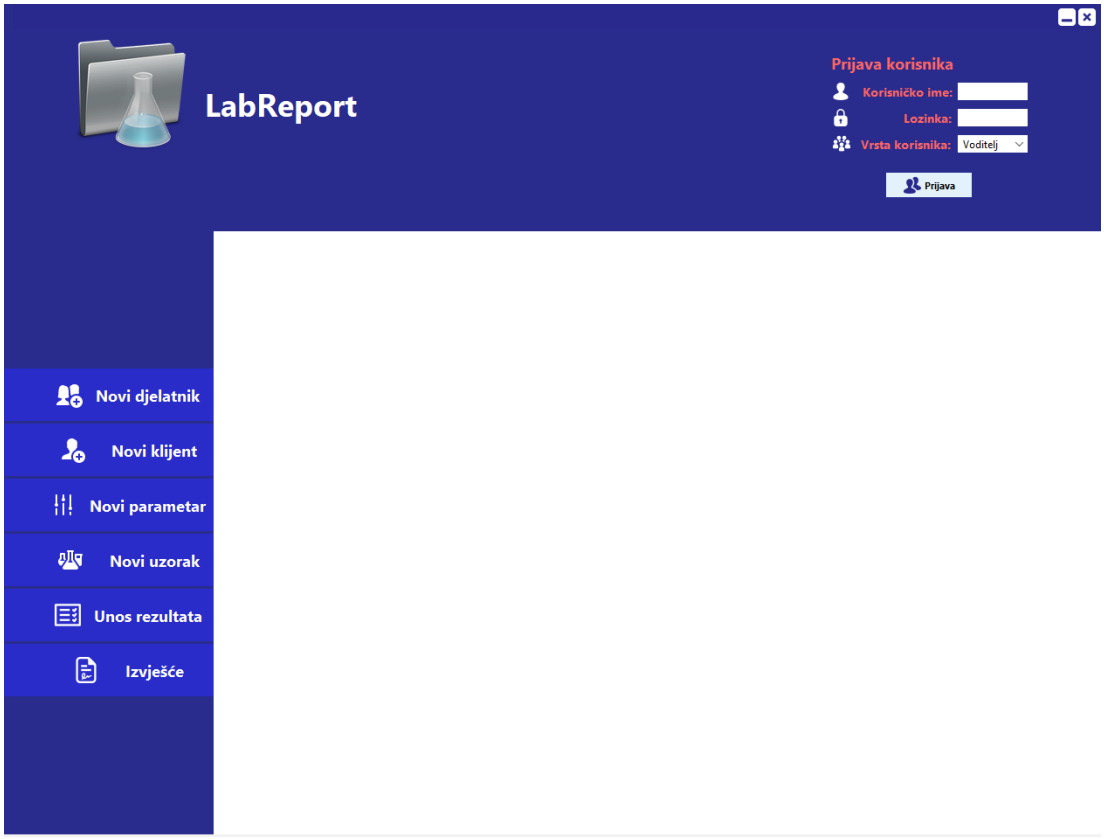

Izvor: autor

Ukoliko se upiše krivo korisničko ime ili lozinka, pojavljuje se poruka korisniku da je unio krive podatke, te se klikom na gumb OK, polja isprazne za ponovni unos podataka.

## 6.2 Bočni izbornik

Nakon uspješne prijave, u donjem dijelu glavnog prozora prikazuje se bočni izbornik koji se sastoji od nekoliko kartica, čijim pritiskom se otvaraju podizbornici u koje je moguće upisivati podatke.

*Slika 20 Bočni izbornik*




Izvor: autor

### 6.2.1 Podizbornik novi djelatnik

Ovaj podizbornik služi za upis novih djelatnika, te njihovu izmjenu i brisanje. Upisuje se ime i prezime djelatnika, OIB, korisničko ime, lozinka i odabire vrsta ovlaštenja djelatnika.

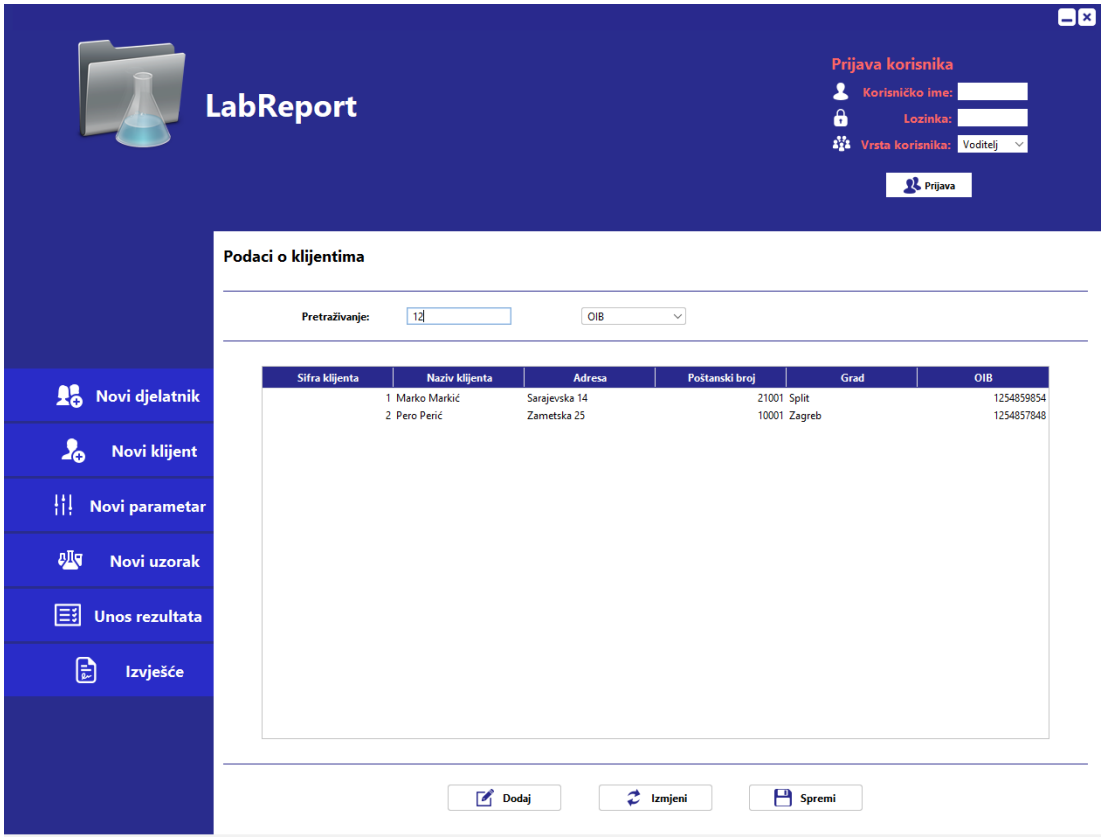
*Slika 21 Podizbornik novi djelatnik*


Izvor: autor

### 6.2.2 Podizbornik novi klijent

Ovaj podizbornik je namijenjen unosu novih klijenata, te njihovom ažuriranju. U gornjem dijelu prozora se nalazi polje za pretraživanje i padajući izbornik za odabir po kojem polju tablice želimo pretraživati upisane klijente.

*Slika 22 Podizbornik novi klijent*

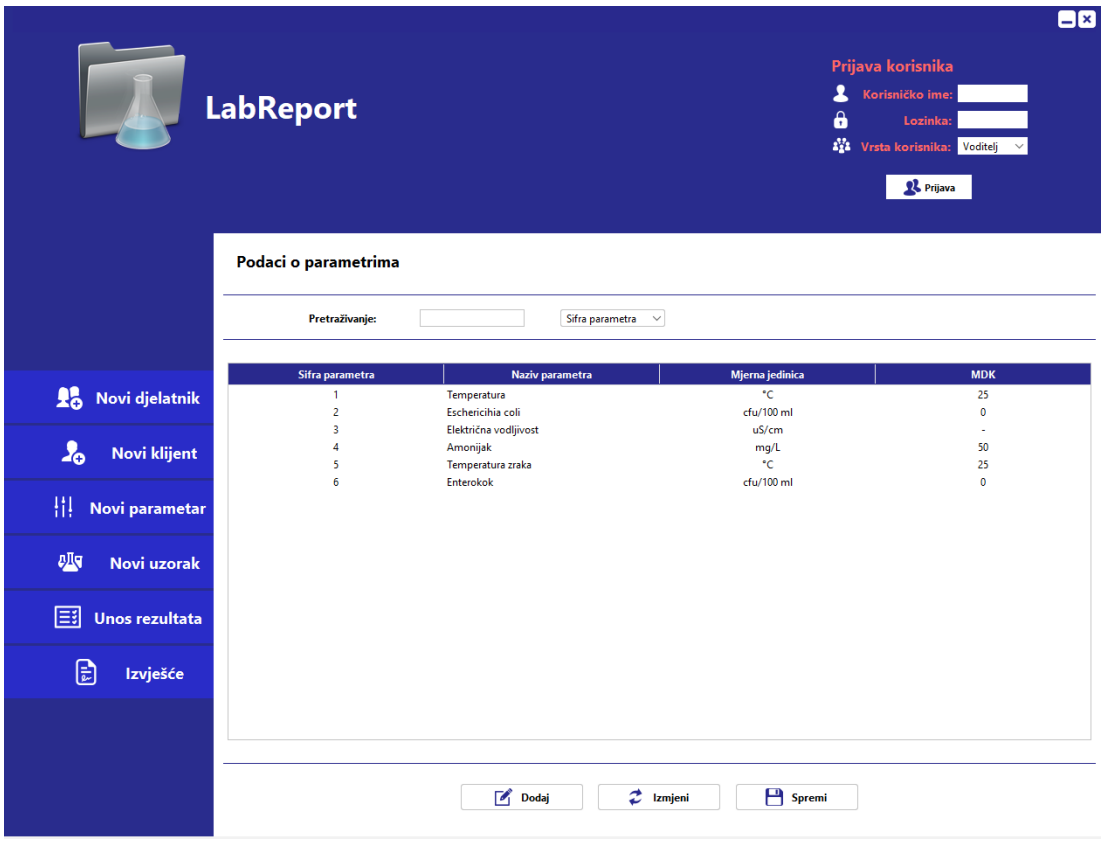



Izvor: autor

### 6.2.3 Podizbornik novi parametar

Ovaj podizbornik je namijenjen unosu novih parametara analize, te njihovom ažuriranju. U gornjem dijelu prozora se nalazi polje za pretraživanje i padajući izbornik za odabir po kojem parametru želimo pretraživati.

*Slika 23 Podizbornik novi parametar*

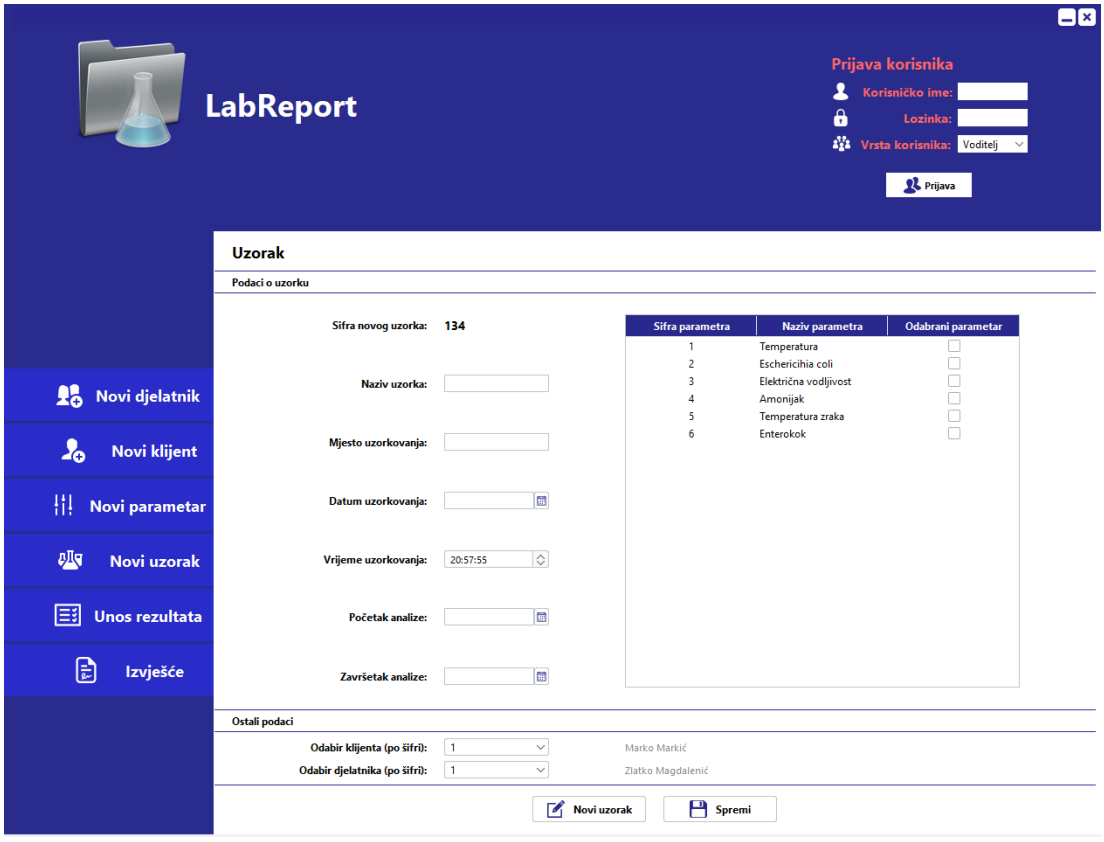



Izvor: autor

### 6.2.4 Podizbornik novi uzorak

Ovaj podizbornik omogućuje unos podataka o uzorku. Aplikacija automatski dodjeljuje laboratorijski broj uzorku, te se u polja upisuju podaci o mjestu i vremenu uzorkovanja, početku i završetku analize. S desne strane je tablica s popisom svih mogućih parametara koji se mogu analizirati te se oni koje želimo analizirati u tom uzorku odabiru u tablici. Na dnu prozora je odabir za kojeg klijenta je uzorak uzorkovan te koji djelatnik ga je uzorkovao.

*Slika 24 Podizbornik novi uzorak*

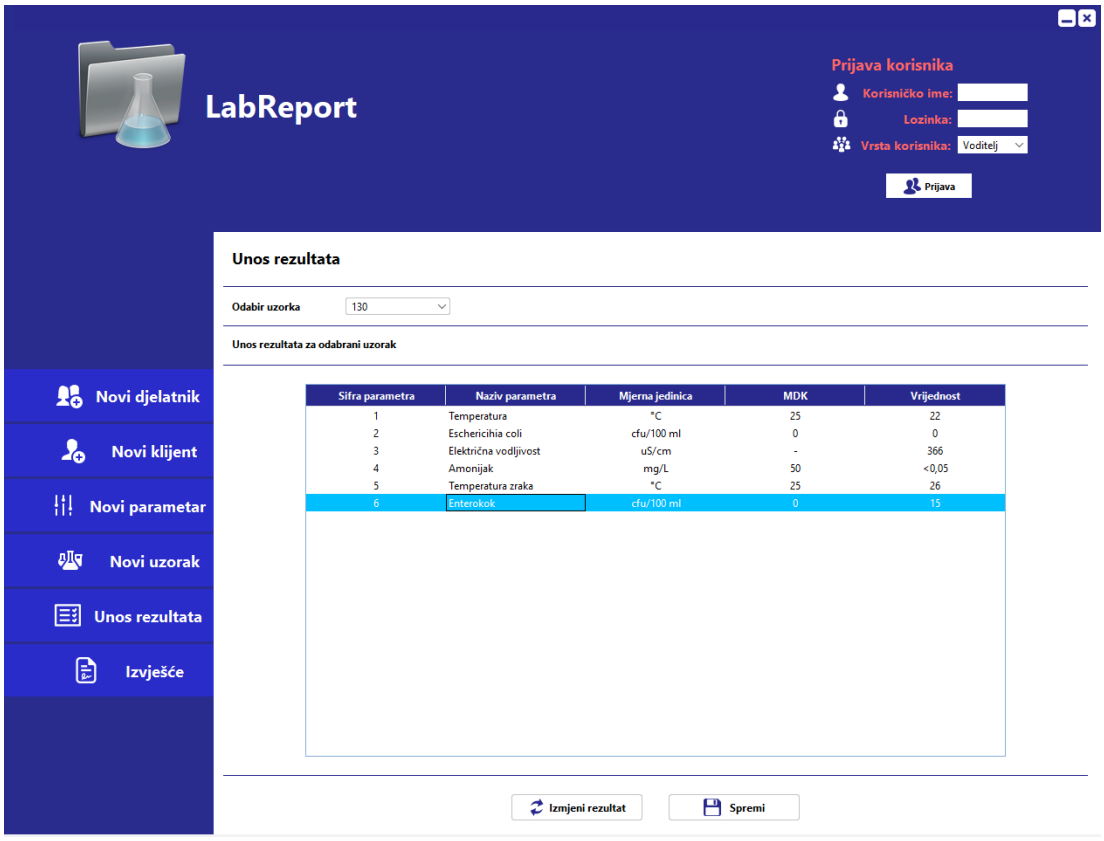



Izvor: autor

### 6.2.5 Podizbornik unos rezultata

U ovom podizborniku korisnik unosi rezultate analize za odabrani uzorak. Prilikom odabira uzorka u tablici se prikazuju samo parametri odabrani u prethodnom koraku, kada se upisivao uzorak, te se samo za njih mogu unijeti rezultati.

*Slika 25 Podizbornik unos rezultata*

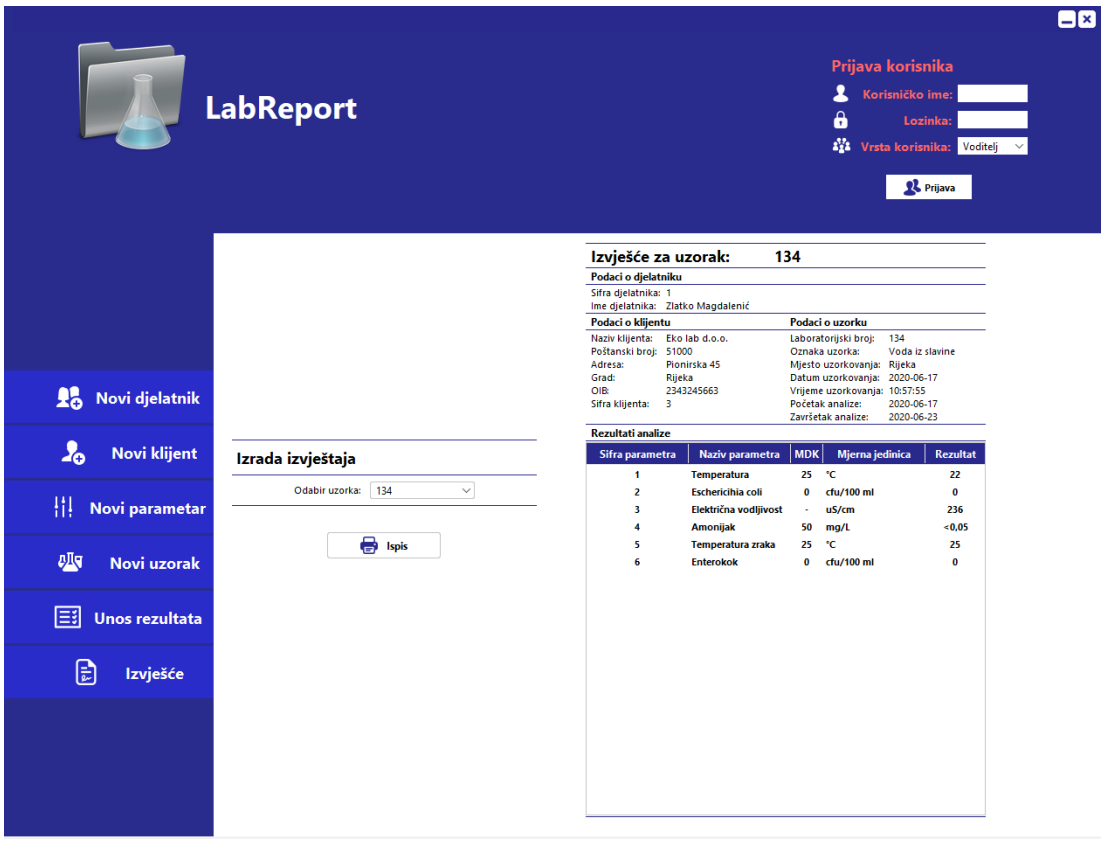



Izvor: autor

### 6.2.6 Podizbornik izvješće

Ovaj podizbornik služi za ispis gotovih izvještaja. Za ispis se koristi sistemski izbornik koji omogućuje ispis na pisač ili u pdf oblik, ovisno o odabiru. Na lijevoj strani prozora se odabere uzorak za koji želimo izvještaj, te se pritiskom na gumb ispis pozove izbornik za odabir vrste ispisa. Na desnoj strani prozora vidimo kako će izvještaj izgledati (*preview*).

*Slika 26 Podizbornik izvješće*




Izvor: autor

# 7. ZAKLJUČAK

U ovom radu je opisana izrada aplikacije „LabReport“ koja je namijenjena djelatnicima laboratorija za brže i učinkovitije praćenje uzoraka i pisanje izvještaja. Za razvoj aplikacije je korišteno NetBeans integrirano razvojno okruženje (NetBeans IDE 8.2) te JDK verzija 1.8.0_211. Razvoj aplikacije je tekao od ideje i izrade relacijskog modela za bazu podataka, do implementacije modela u bazu podataka sa svim tablicama i ključevima. Nakon toga je slijedila razrada korištenja aplikacije kako bi se izradilo grafičko sučelje, pri čemu su korištene i dodatne biblioteke kako bi aplikacija imala moderniji izgled, a paralelno s izradom sučelja se i testirao rad aplikacije i njeno povezivanje s bazom.

Razvoj aplikacije je trajao 5 mjeseci, i usput su stalno nalažena bolja rješenja od već implementiranih, što je usporilo izradu. Izrada ove aplikacije je bila izazovna, međutim bila je i iznimno korisna jer je autor stekao cijeli niz novih znanja iz programiranja u Java programskom jeziku. Obzirom na ograničeno vrijeme za izradu aplikacije, nisu implementirane sve značajke koje su prvotno bile zamišljene, poput naprednijeg sustava za ispis izvještaja, koji bi omogućio da se ispis grafički uredi. Obzirom da aplikacija ima praktičnu primjenu, ona će u budućnosti biti dalje nadograđivana.

# LITERATURA


1. Bai, Y., Practical Database Programming with Java, Wiley-IEEE Press, 2011, https://books.google.hr/books?id=ZGmzqnnu904C&printsec=frontcover&dq=Practical+Database+Programming+with+Java&hl=hr&sa=X&ved=2ahUKEwj3oMuzlunrAhW7QEEAHenDA0kQ6AEwAHoECAIQAg#v=onepage&q=Practical%20Database%20Programming%20with%20Java&f=false (23. 6. 2020.)
2. Davidović, V., Objektno orijentirane tehnologije II, Veleučilište u Rijeci, 2016, https://www.veleri.hr/?q=system/files/nastavni_materijali/k_informatika_3/OOT2-skripta-final.pdf (23. 6. 2020.)
3. FlatLaf - Flat Look and Feel | FormDev., https://www.formdev.com/flatlaf/ (23. 6. 2020.)
4. JCalendar, https://toedter.com/jcalendar/, (pristupljeno 23. lipnja 2020.)
5. Manger, R., Osnove projektiranja baza podataka, Sveučilišni računski centar, Sveučilište u Zagrebu, 2010, https://www.srce.unizg.hr/tecajevi/popis-osnovnih-tecajeva/D310 (23. 6. 2020.)
6. MySQL : MySQL 8.0 Reference Manual : 1.3.1 What is MySQL?, https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html, (23. 6. 2020.)
7. MySQL :: MySQL Connector/J 5.1 Developer Guide :: 1 Overview of MySQL Connector/J, https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-overview.html, (23. 6. 2020.)
8. MySQL :: MySQL Workbench Manual, https://dev.mysql.com/doc/workbench/en/, (23. 6. 2020.)
9. Pavlić, M., Oblikovanje baza podataka, Odjel za informatiku Sveučilišta u Rijeci, 2011, https://library.foi.hr/pdf/web/viewere.php?fil=https://bib.irb.hr/datoteka/556923.OBP7_skraceno.pdf, (24. 6. 2020.)
10. Thakur, A., How to get the next auto-increment id in MySQL?, https://www.tutorialspoint.com/how-to-get-the-next-auto-increment-id-in-mysql, (23. 6. 2020.)

11. Wikipedia, the free encyclopedia, Java (programming language), https://en.wikipedia.org/wiki/Java_(programming_language), (23. 6. 2020.)

# POPIS SLIKA