# Java aplikacija za taxi službe

**Hlavati, Luka**





**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**
**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Stručni studij informatike**

# JAVA APLIKACIJA ZA TAXI SLUŽBE

**Završni rad**

**Luka Hlavati**

**Osijek, 2021.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju**

**Osijek, 30.06.2021.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

| | |
|---|---|
| **Ime i prezime studenta:** | Luka Hlavati |
| **Studij, smjer:** | Preddiplomski stručni studij Elektrotehnika, smjer Informatika |
| **Mat. br. studenta, godina upisa:** | AI 4608, 27.07.2017. |
| **OIB studenta:** | 35265447429 |
| **Mentor:** | Izv. prof. dr. sc. Ivica Lukić |
| **Sumentor:** | |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | Izv. prof. dr. sc. Krešimir Nenadić |
| **Član Povjerenstva 1:** | Izv. prof. dr. sc. Ivica Lukić |
| **Član Povjerenstva 2:** | Doc.dr.sc. Mirko Köhler |
| **Naslov završnog rada:** | Java aplikacija za taxi službe |
| **Znanstvena grana rada:** | **Informacijski sustavi (zn. polje računarstvo)** |
| **Zadatak završnog rada** | Napraviti Java aplikaciju za taxi službe koja omogućuje CRUD naredbe za vozače vozila i korisnike. U bazu podataka spremiti sve vožnje sa podacima o vozaču, vozilu i korisniku. Na kraju napraviti statistiku vožnji za vozača. Tema rezervirana za studenta: Luka Hlavati |
| **Prijedlog ocjene pismenog dijela ispita (završnog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene mentora:** | 30.06.2021. |

| | |
|---|---|
| *Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:* | Potpis: |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 12.07.2021.**

| | |
|---|---|
| **Ime i prezime studenta:** | Luka Hlavati |
| **Studij**: | Preddiplomski stručni studij Elektrotehnika, smjer Informatika |
| **Mat. br. studenta, godina upisa:** | AI 4608, 27.07.2017. |
| **Turnitin podudaranje [%]:** | 9% |

Ovom izjavom izjavljujem da je rad pod nazivom**: Java aplikacija za taxi službe**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

# SADRŽAJ

# 1. UVOD

U ovom radu prikazuje se proces izrade desktop aplikacije koja omogućuje korisniku CRUD baze podataka taxi službe. Namjena aplikacije je omogućiti korisniku unos, čitanje, promjenu i brisanje podataka iz baze podataka taxi službe. Na primjer, unos novog vozila i njegovih specifikacija u taxi službu, promjena postojećeg vozača, pregled vozila i vozača (koji vozač vozi koje vozilo) te brisanje vožnje i njezinih podataka.

Rješenje je ostvareno pomoću programskog jezika Java, baze podataka MariaDB te open-source web server paketa XAMPP. Izradu aplikacije je olakšao Maven, alat za projektni menadžment koji se bazira na POM-u (eng. Project Object Model). Koristi se za pokretanje projekta, dependency-a i dokumentacije. Za pisanje kod-a i pravljenje dizajna aplikacije sam koristio Apache NetBeans IDE 11.1 i JDK 11.

Isto tako mi je olakšao rad Hibernate, Java framework koji pojednostavlja povezivanje i rad Java aplikacije i baze podataka. Hibernate je open source te ORM (eng. Object Relational Mapping) alat. Implementira specifikacije od JPA (eng. Java Persistence API) za data persistence.

Kod je strukturiran pomoću MVC-a (eng. Model-View-Controller). MVC je model dizajna aplikacije, sastoji se od tri međusobno povezana dijela. Koristi se kako bi se pojedini dijelovi aplikacije odvojili u komponente radi boljeg pregleda i njihove namjene.

## 1.1. Zadatak završnog rada

U teorijskom dijelu rada potrebno je proučiti i opisati tehnologije za izradu Java aplikacija i povezivanje sa bazama podataka. U praktičnom dijelu rada potrebno napraviti Java aplikaciju te je povezati sa bazom podataka kako bi se omogućio CRUD (eng. Create Read Update Delete) baze podataka.

# 2. KORIŠTENE TEHNOLOGIJE

Opis korištenih tehnologija aplikacije će biti u ovom djelu rada. Korištene su MariaDB, XAMPP, Apache NetBeans IDE 11.1, programski jezik Java, MVC te alati Maven i Hibernate.

## 2.1. Baza podataka MariaDB

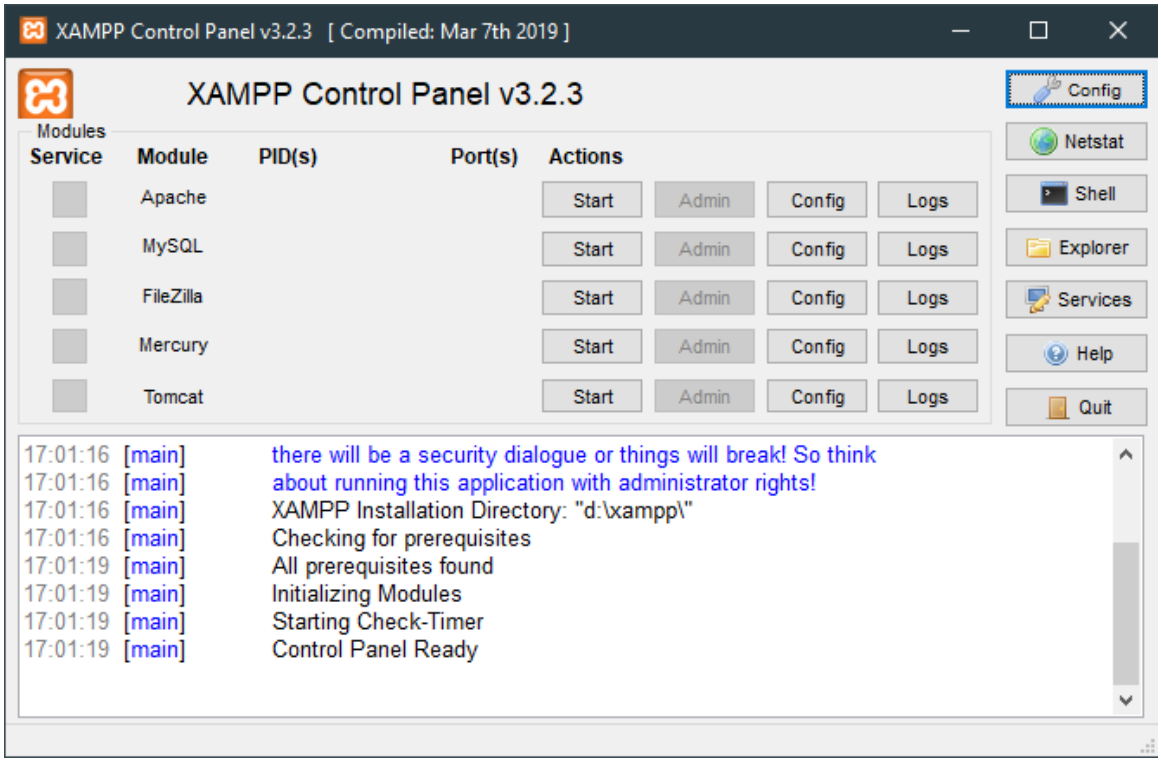
MariaDB server jedna je od najpopularnijih relacijskih baza podataka otvorenog koda. MariaDB je razvijena od zajednice, komercijalno podržana vilica MySQL relacijskog sustava upravljanja bazom podataka (eng. RDBMS), namijenjena da ostane besplatna i softver otvorenog koda pod GNU (eng. General Public License). Namjeravala je održavati visoku kompatibilnost s MySQL-om, osiguravajući mogućnost zamjenske zamjene s binarnim paritetom knjižnice i točno podudaranje s MySQL API-ima i naredbama. [1]

MariaDB je superiornija od MySQL-a jer pokazuje poboljšanu brzinu u usporedbi s MySQL-om. Pomoću mehanizma za pohranu memorije MariaDB, INSERT izjava može se dovršiti za 24% brže nego u standardnom MySQL-u.


**Slika 2.1.** Logo MariaDB

## 2.2. XAMPP

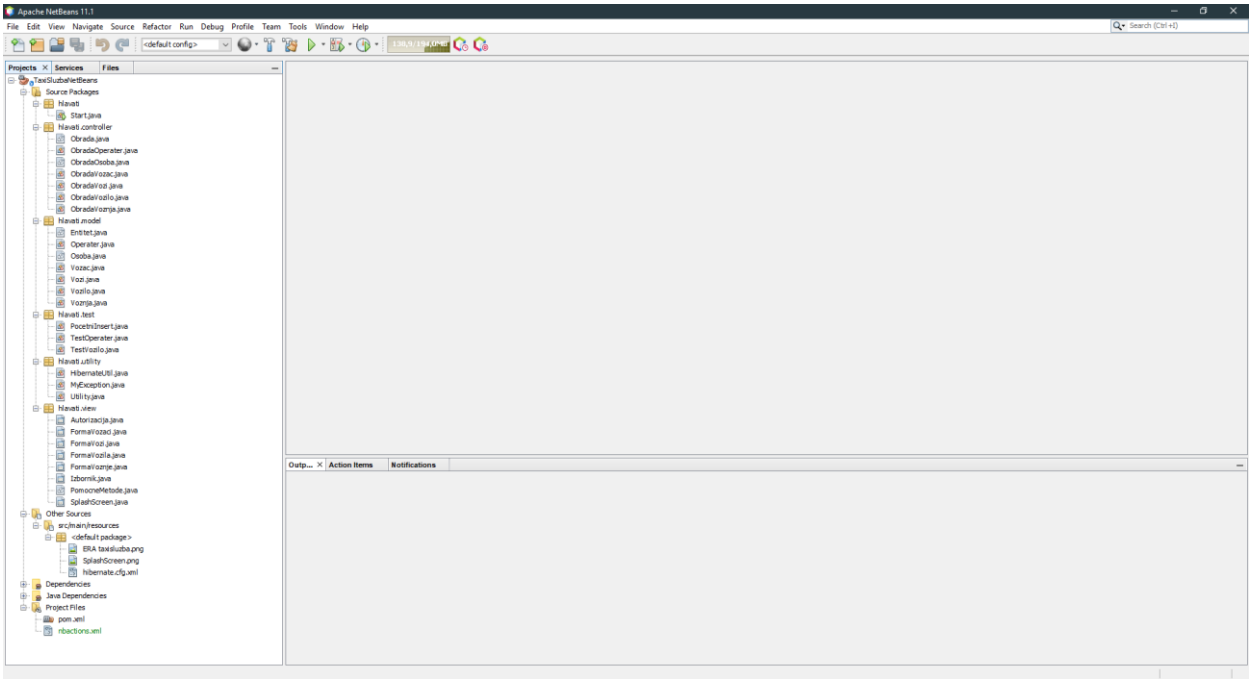
XAMPP je najpopularnije PHP razvojno okruženje. XAMPP je potpuno besplatna, lako instalirana Apache distribucija koja sadrži MariaDB, PHP i Perl. XAMPP je kratica za višeplatformske Apache, MySQL, PHP i Perl i omogućuje nam izradu WordPress web mjesta izvan mreže, na lokalnom web poslužitelju na računalu. Ovo jednostavno i lagano rješenje radi na sustavima Windows, Linux i Mac - otuda i dio "cross-platform". Vrlo je jednostavan za koristiti zbog svog preglednog control panel-a što se vidi na Slici 2.2. [2]




**Slika 2.2.** Control panel XAMPP-a

## 2.3. Apache NetBeans IDE

NetBeans je integrirano razvojno okruženje za Javu. NetBeans omogućava razvoj aplikacija iz modula. Moduli se nazivaju skupovi modularnih softverskih komponenti. Većina programera prepoznaje NetBeans IDE kao izvorni besplatni Java IDE. Apache NetBeans IDE pruža podršku za nekoliko jezika (Java, PHP, JavaFX, JavaScript, itd.) i framework-a, a uskoro će se uključiti i više (C / C ++). [3]



**Slika 2.3.** Razvojno okruženje Apache NetBeans-a IDE

## 2.4. Java

Java je objektno orijentirani programski jezik koji je objavljen u studenom 1995. Java se može koristiti za stvaranje cjelovitih desktop aplikacija između ostalog. Također se može koristiti za izradu malog aplikacijskog modula ili apleta (jednostavno dizajniranog, malog programa) za upotrebu kao dio web stranice. [4]

Programi u Javi mogu se pokrenuti na svim računalima na kojima postoji JVM (eng. Java Virtual Machine), isto tako java je jedan od popularnijih programskih jezika. Primjer programa koji ispisuje „Hello World!“ u konzolu možemo vidjeti na Slici 2.4.

```
public class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

**Slika 2.4.** Primjer programa „Hello World“

## 2.5. Maven

Maven je alat za automatizaciju izrade koji se prije svega koristi za Java projekte. Maven se također može koristiti za izradu i upravljanje projektima napisanim na C #, Ruby, Scala i drugim jezicima. Projekt Maven domaćin je softverske fondacije Apache, gdje je ranije bio dio projekta Jakarta. [5]

## 2.6. Hibernate

Hibernate ORM je objektno-relacijski alat za mapiranje za Javu. Hibernate brine o mapiranju Java klasa u tablice baze podataka pomoću XML datoteka i bez upisivanja bilo kojeg retka koda. Ako postoje promjene u bazi podataka ili bilo kojoj tablici, tada morate promijeniti samo svojstva XML datoteke. Abstraktira nepoznate SQL tipove i pruža način zaobilaženja poznatih Java objekata. [6]

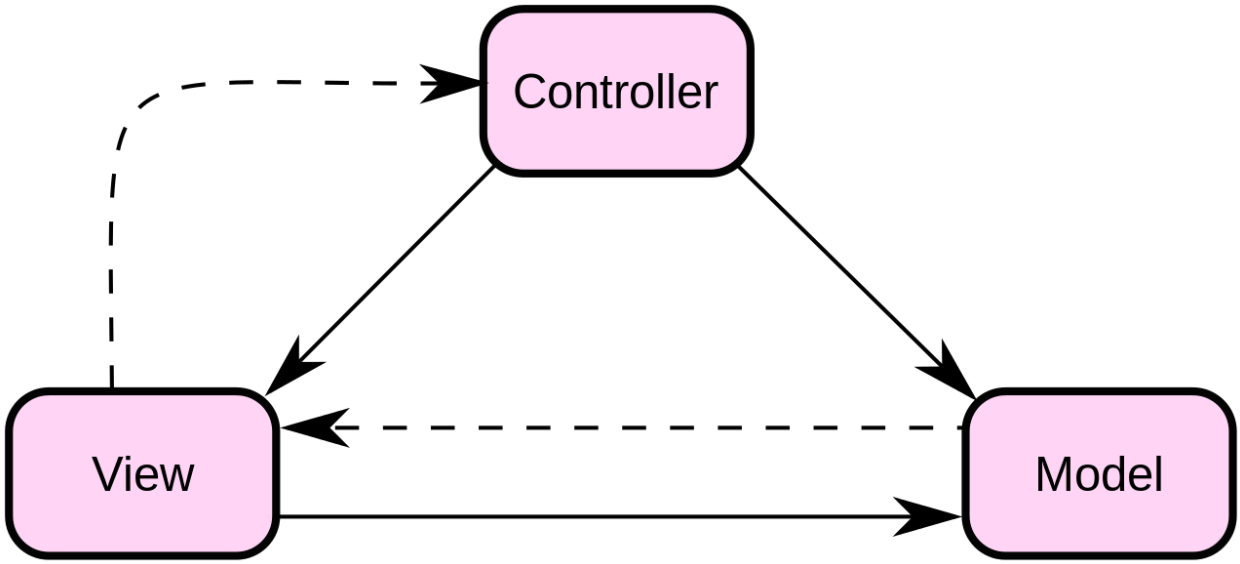
## 2.7. MVC

Model–View–Controller (MVC) je model dizajna aplikacije, sastoji se od tri međusobno povezana dijela. Koristi se kako bi se pojedini dijelovi aplikacije odvojili u komponente radi boljeg pregleda i njihove namjene.

Model je u principu klasa koja se sastoji od svojih privatnih ili javnih podataka. Iz modela se onda mogu izvlačiti ili postavljati željeni podaci.

View ili pogled može biti neki obrazac koji prikazivati izgled i pojedine radnje aplikacije.

Kontroler ima u sebi određene metode koje kontroliraju i obavljaju neke radnje na aplikaciji i šalje upite na bazu podataka. [7]

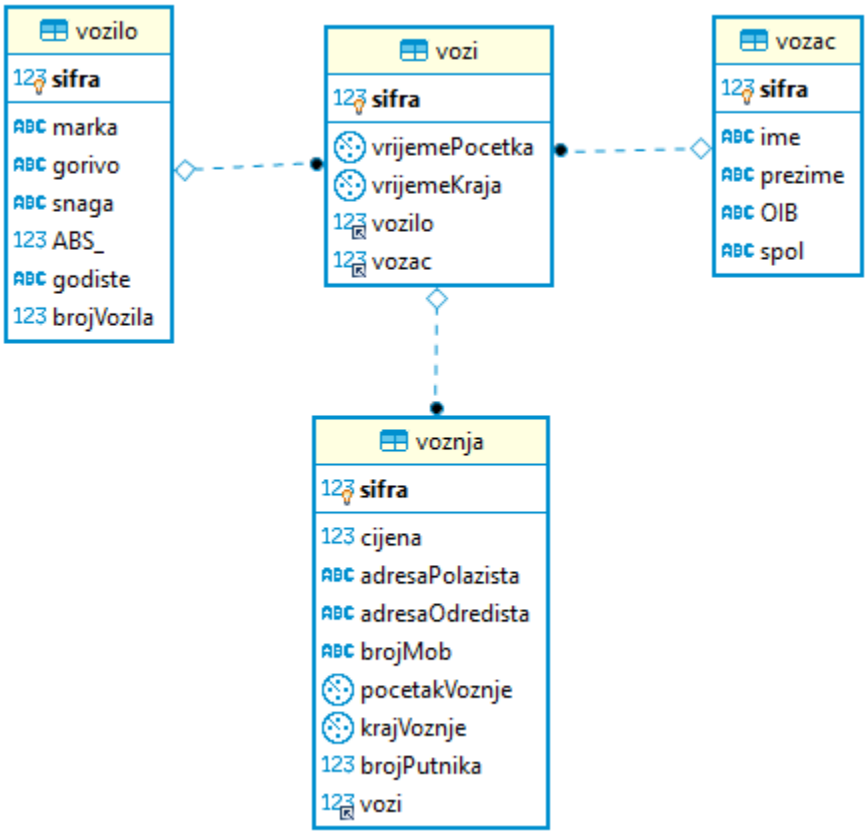



**Slika 2.5.** MVC koncept

# 3. STRUKTURA APLIKACIJE

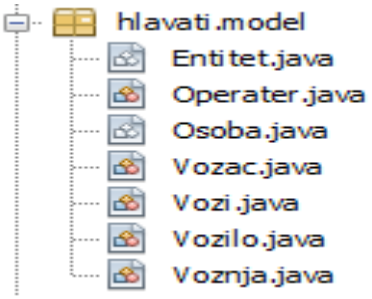
## 3.1. ERA dijagram

Prije samog početka pisanja aplikacije potrebno je osmisliti kako želimo da nam izgleda baza podataka. Nakon toga možemo napraviti ERA (eng. Entity Relationship Attribute) dijagram pomoću kojeg možemo bazu prikazati shematski što vidimo na Slici 3.1.




**Slika 3.1.** ERA dijagram aplikacije

## 3.2. Model




**Slika 3.2.** Model aplikacije

U paketu model se nalaze „entiteti“ kao java klase. Primjenjen je pristup učahurivanja/enkapsulacija u svakoj pojedinoj klasi. Varijable klase se ne može pristupiti nikako drugačije nego metodama jer ne postoje globalne varijable. Klase u modelu se sastoje od privatnih podataka, praznog i punog konstruktora te getter-a i setter-a koji nam služe kao „povlaćenje“ podataka iz baze i postavljanje podataka u bazu što vidimo sa Slike 3.3.

Osoba.java

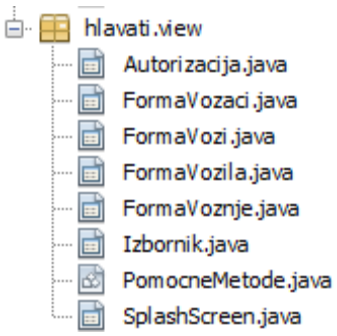
Source | History

```
18
19      private String ime;
20      private String prezime;
21      private String oib;
22      private String spol;
23
24      public Osoba() {
25          super();
26      }
27
28      public Osoba(String ime, String prezime, String oib, Integer sifra) {
29          super(sifra);
30          this.ime = ime;
31          this.prezime = prezime;
32          this.oib = oib;
33      }
34
35      public String getIme() {
36          return ime;
37      }
38
39      public void setIme(String ime) {
40          this.ime = ime;
41      }
42
43      public String getPrezime() {
44          return prezime;
45      }
46
47      public void setPrezime(String prezime) {
48          this.prezime = prezime;
49      }
50
51      public String getOib() {
52          return oib;
53      }
54
55      public void setOib(String oib) {
56          this.oib = oib;
57      }
58
59      public String getSpol() {
60          return spol;
61      }
62
63      public void setSpol(String spol) {
64          this.spol = spol;
65      }
```

**Slika 3.3.** Primjer učahurivanja podataka klase Osoba

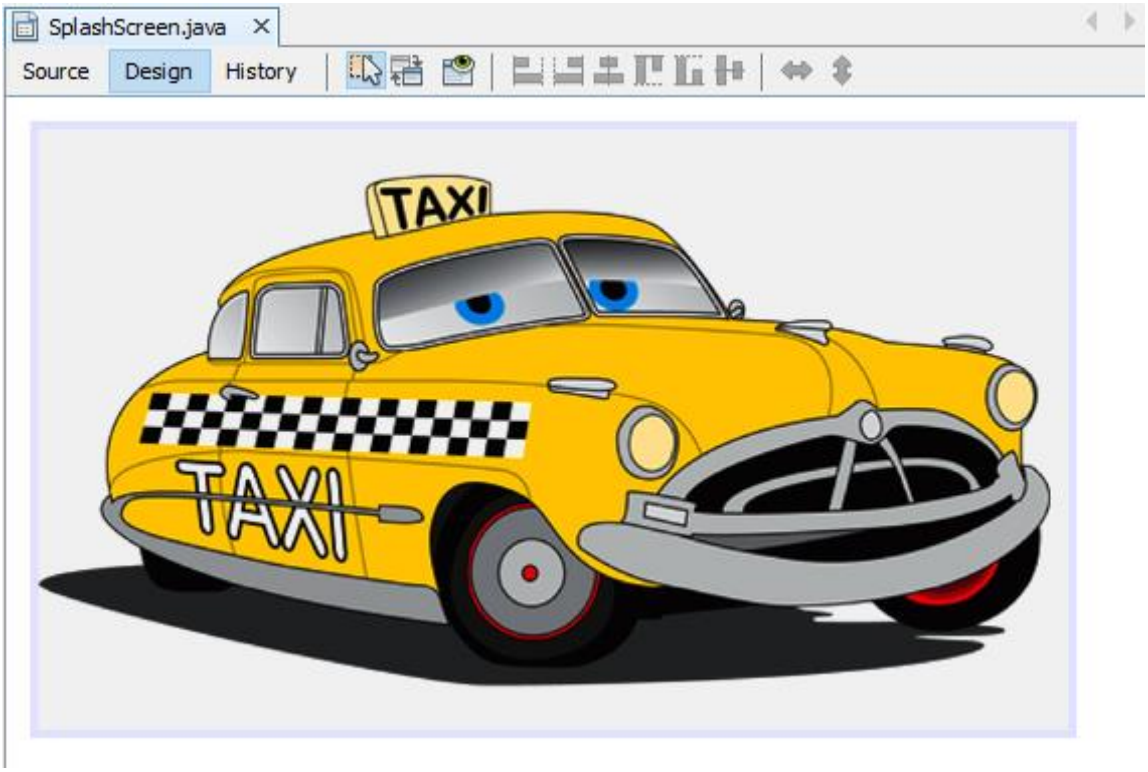
## 3.3. View

U paketu view se nalaze pogledi koji će služiti za prikaz aplikacije što vidimo sa slike 3.4.




**Slika 3.4.** View aplikacije

Kod pokretanja aplikacije prvi pogled koji se pokreće je SplashScreen, on se ponaša kao početni zaslon koji se sastoji od jednostavne slike (logotip) dok se aplikacija pokreće u pozadini što možemo vidjeti iz Slike 3.5.




**Slika 3.5.** Forma SplashScreen

SplashScreen.java

Source | Design | History

```
15  public class SplashScreen extends javax.swing.JFrame {
16
17      /**
18       */
19      public SplashScreen() {
20          initComponents();
21          Ucitavanje ucitavanje = new Ucitavanje();
            ucitavanje.start();
            setBackground(new Color(0,0,0,0));
24      }
25
26      private class Ucitavanje extends Thread{
27
28          @Override
            public void run() {
30              if(new ObradaVozac().getVozaci().size()>0){
31                  new Autorizacija().setVisible(true);
32                  dispose();
33              }
34          }
35
36      }
```

**Slika 3.6.** Source kod SplashScreen

Poslije splash screen-a se pokreće obrazac Autorizacija kao što možemo vidjeti iz slike 3.7. gdje se operater treba prijaviti svojim prezimenom i lozinkom kako bi se aplikacija mogla koristiti.


**Slika 3.7.** Forma Autorizacija

Također je implementirana provjera kod autorizacije, operater ne može ući u aplikaciju ako nije unio ispravno prezime i lozinku. Za svaku kombinaciju pogrešnog unosa se dobiva dijaloški okvir sa određenom porukom što je krivo unešeno što možemo vidjeti iz Slike 3.8.

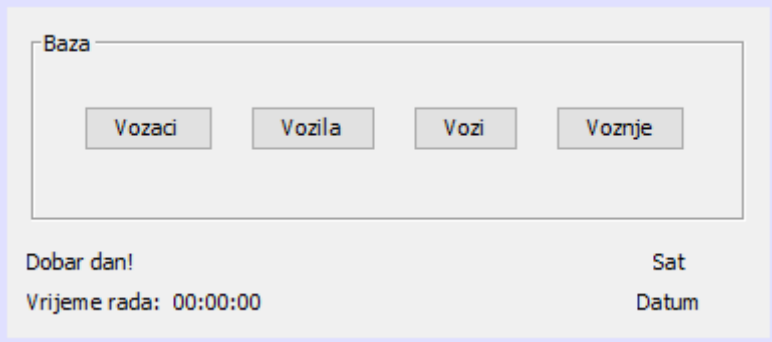
```
116     private void btnAutorizirajActionPerformed(java.awt.event.ActionEvent evt) {
117         if (txtPrezime.getText().trim().length() == 0) {
118             greska(txtPrezime, "Obavezan unos prezimena!");
119             txtPrezime.requestFocus();
120             return;
121         }
122 
123         if(txtPrezime.getText().matches(".*\\d.*")){
124             greska(txtPrezime, "Prezime ne može sadržavati broj!");
125             txtPrezime.requestFocus();
126             return;
127         }
128 
129         txtPrezime.setBorder(new LineBorder(Color.WHITE));
130 
131         if (pswLozinka.getPassword().length == 0) {
132             greska(pswLozinka, "Obavezan unos lozinke!");
133             pswLozinka.requestFocus();
134             return;
135         }
136 
137         pswLozinka.setBorder(new LineBorder(Color.WHITE));
138 
139         Operater o = new ObradaOperater().getOperater(txtPrezime.getText());
140 
141         if (o == null) {
142             greska(txtPrezime, "Nepostojeće prezime!");
143             txtPrezime.requestFocus();
144             return;
145         }
146 
147         if (!BCrypt.checkpw(new String(pswLozinka.getPassword()), o.getLozinka())) {
148             greska(pswLozinka, "Kombinacija prezimena i lozinka ne odgovara!");
149             pswLozinka.requestFocus();
150             return;
151         }
152 
153         new Izbornik().setVisible(true);
154         dispose();
155 
```

**Slika 3.8.** Provjera unešenih podataka nakon klika na gumb „Autoriziraj“

```
168     private void greska(JComponent komponenta, String poruka) {
169         JOptionPane.showMessageDialog(null, poruka);
170         komponenta.setBorder(new LineBorder(Color.RED));
171         komponenta.requestFocus();
172     }
```

**Slika 3.9** Metoda za prikaz dijaloškog okvira i poruke

Nakon uspješne autorizacije operatera, poziva se obrazac izbornik što možemo vidjeti sa slike 3.10 iz kojeg možemo izabrati neku tablicu iz baze te raditi CRUD (eng. Create Read Update Delete) baze. Poruka se prikazuje ovisno o tome koje je doba dana, isto tako radno vrijeme se prati od kad se operater ulogirao u aplikaciju, kao što je prikazano u klasi RadnoVrijeme na Slici 3.11.



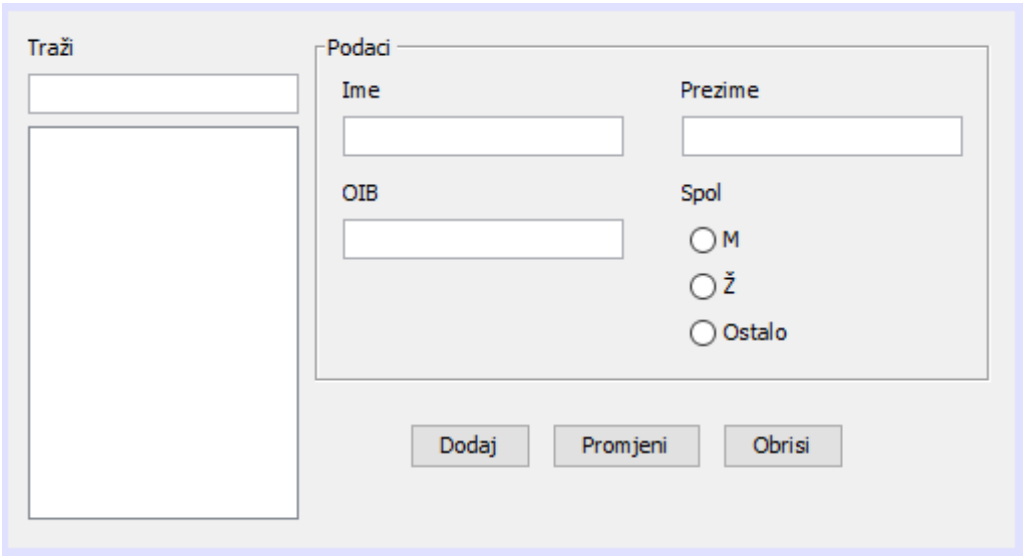

**Slika 3.10.** Obrazac Izbornik

```
private class RadnoVrijeme extends Thread {

    int h = 0;
    int m = 0;
    int s = 0;

    @Override
    public void run() {
        if (s > 59) {
            s = 0;
            m++;
        }
        if (m > 59) {
            m = 0;
            h++;
        }

        lblRadnoVrijeme.setText(String.format("%02d:%02d:%02d", h, m, s));
        s++;
        try {
            Thread.sleep(1000);
        } catch (InterruptedException e) {
        }

        run();
    }

}

private void Pozdrav() {
    if (Integer.parseInt(lblSat.getText().substring(0, 2)) < 12) {
        lblPozdrav.setText("Dobro jutro!");
    } else if (Integer.parseInt(lblSat.getText().substring(0, 2)) < 24
            && Integer.parseInt(lblSat.getText().substring(0, 2)) >= 18) {
        lblPozdrav.setText("Dobra večer!");
    } else {
        lblPozdrav.setText("Dobar dan!");
    }

}
```

**Slika 3.11.** Metode za poruku dobrodošlice i praćenje radnog vremena

### 3.3.1. Obrasci Tablica

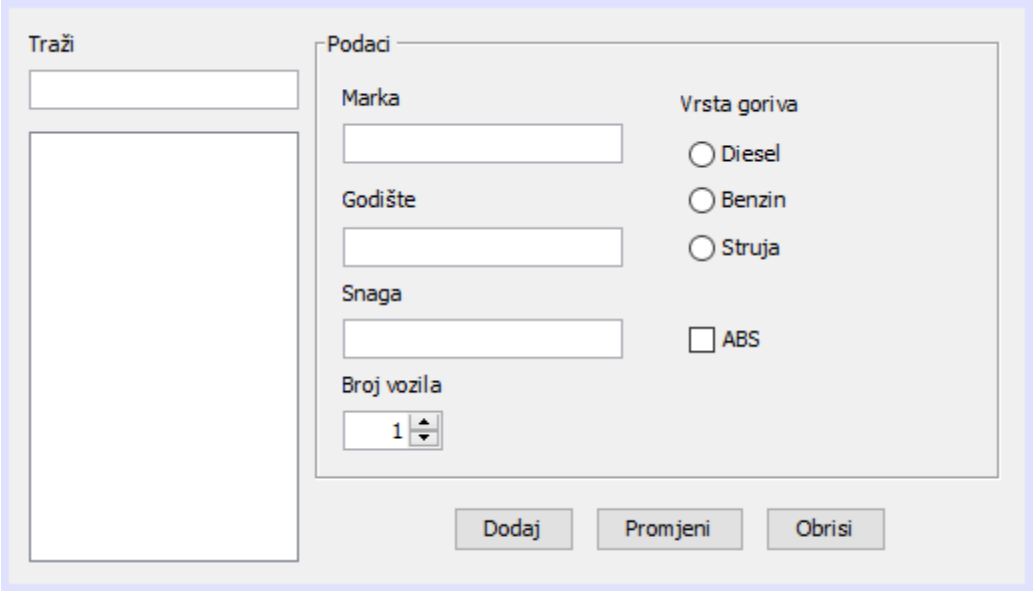
Svi obrasci (eng. Forms) tablica (FormaVozaci, FormaVozi, FormaVozila i FormaVoznje) se sastoje od tražilice, liste podataka, panela za unos podataka i gumbova za unos, promjenu i brisanje podataka iz tablice. Svaki podataka koji se unosi ima svoju provjeru valjanosti. Za primjer uzeti ćemo obrazac FormaVozaci što vidimo na Slici 3.12., kada bih u OIB upisali „12345“, metoda kontrolaOIB bi ispisala poruku u dijaloški okvir o nevažećem OIB-u što možemo vidjeti sa Slike 3.13.



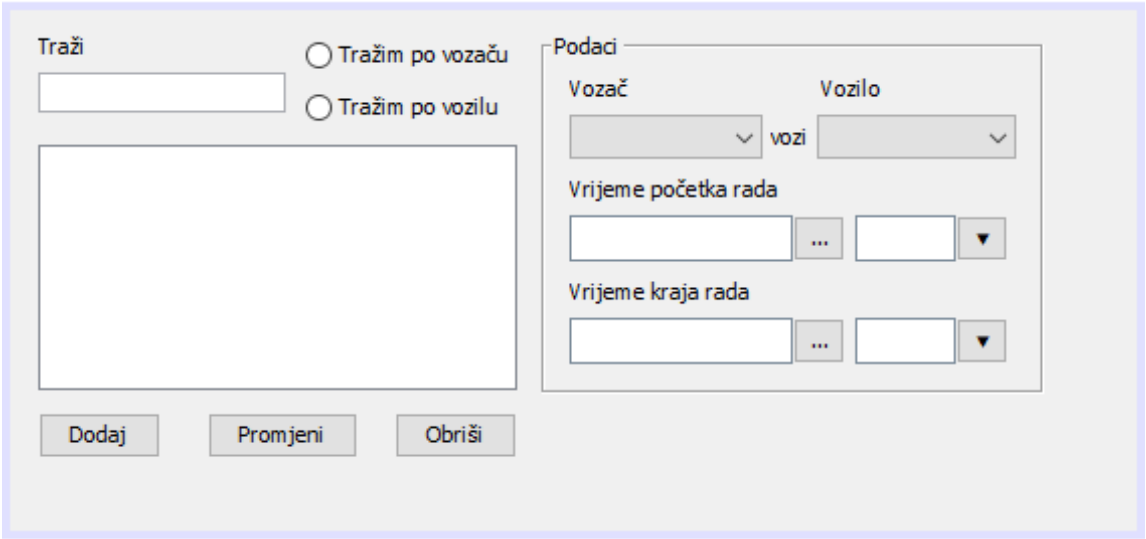

**Slika 3.12.** Obrazac FormaVozaci

```
private boolean kontrolaOIB(Vozac v) {
    if (!checkOIB(txtOIB.getText())) {
        JOptionPane.showMessageDialog(null, "OIB nije važeći!");
        return false;
    }
    v.setOib(txtOIB.getText());
    return true;
}

public static boolean checkOIB(String oib) {

    if (oib.length() != 11) {
        return false;
    }

    try {
        Long.parseLong(oib);
    } catch (NumberFormatException e) {
        return false;
    }

    int a = 10;
    for (int i = 0; i < 10; i++) {
        a = a + Integer.parseInt(oib.substring(i, i + 1));
        a = a % 10;
        if (a == 0) {
            a = 10;
        }
        a *= 2;
        a = a % 11;
    }
    int kontrolni = 11 - a;
    if (kontrolni == 10) {
        kontrolni = 0;
    }

    return kontrolni == Integer.parseInt(oib.substring(10));
}
```

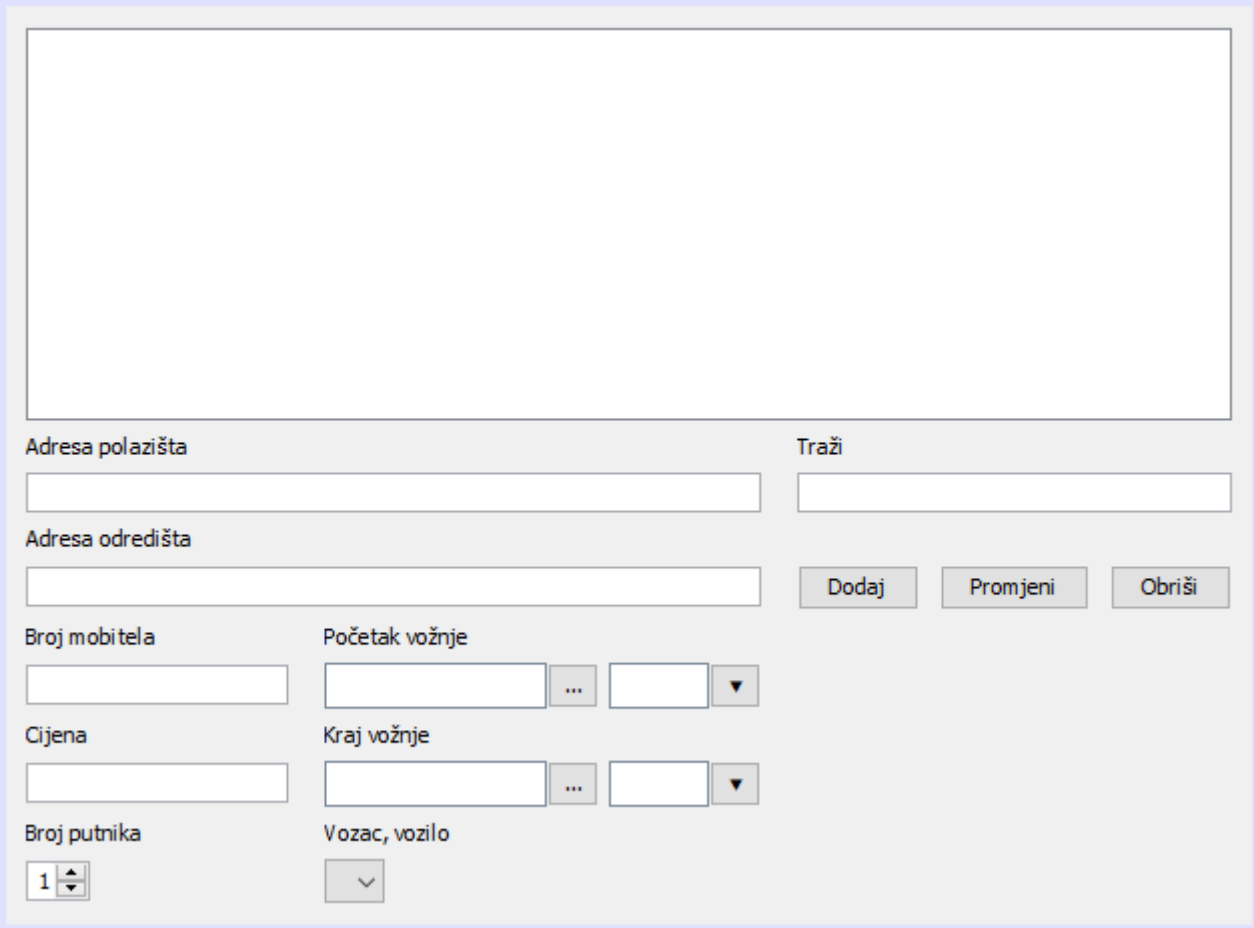
**Slika 3.13.** Metoda checkOIB koja kontrolira ispravnost OIB-a i metoda kontrolaOIB koja ispisuje poruku o ne važećem OIB-u




**Slika 3.14.** Obrazac FormaVozila



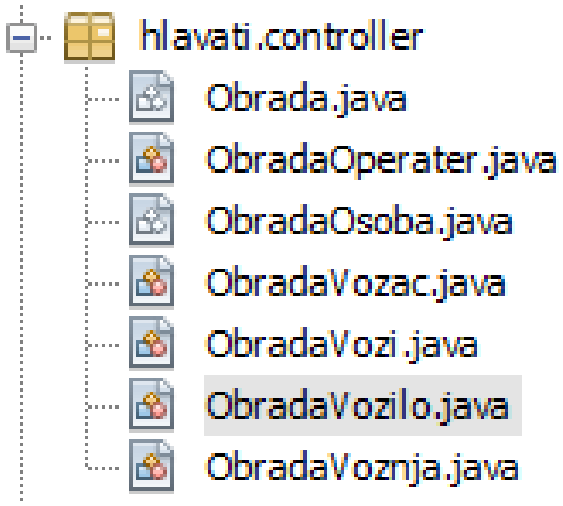

**Slika 3.15.** Obrazac FormaVozi




**Slika 3.16.** Obrazac FormaVoznje

## 3.4. Kontroler

U paketu kontroleru (eng. Controller) uspostavljamo sesiju sa bazom podataka i kontroliramo spremanje i brisanje entiteta u bazi. Sesija (eng. Session) je vremenski interval u kojem dva sustava (tj. klijent i poslužitelj) međusobno komuniciraju. Jednostavnije rečeno, sesija je stanje koje se sastoji od nekoliko zahtjeva i odgovora između klijenta i poslužitelja. Sve klase u kontroleru nasljeđuju klasu Obrada što možemo vidjeti sa Slike 3.18.




**Slika 3.17.** Paket controller

```
public abstract class Obrada<T> {

    protected abstract void kontrolaSpremi()throws MyException;
    protected abstract void kontrolaBrisi()throws MyException;

    protected Session session;

    public Obrada() {
        this.session = HibernateUtil.getSession();
    }

    public T spremi(T entitet) throws MyException{
        kontrolaSpremi();
        session.beginTransaction();
        session.save(entitet);
        session.getTransaction().commit();

        return entitet;
    }

    public void brisi(T entitet)throws MyException{
        kontrolaBrisi();
        session.beginTransaction();
        session.delete(entitet);
        session.getTransaction().commit();
    }

}
```

**Slika 3.18.** Klasa Obrada

## 3.5. Pom.xml

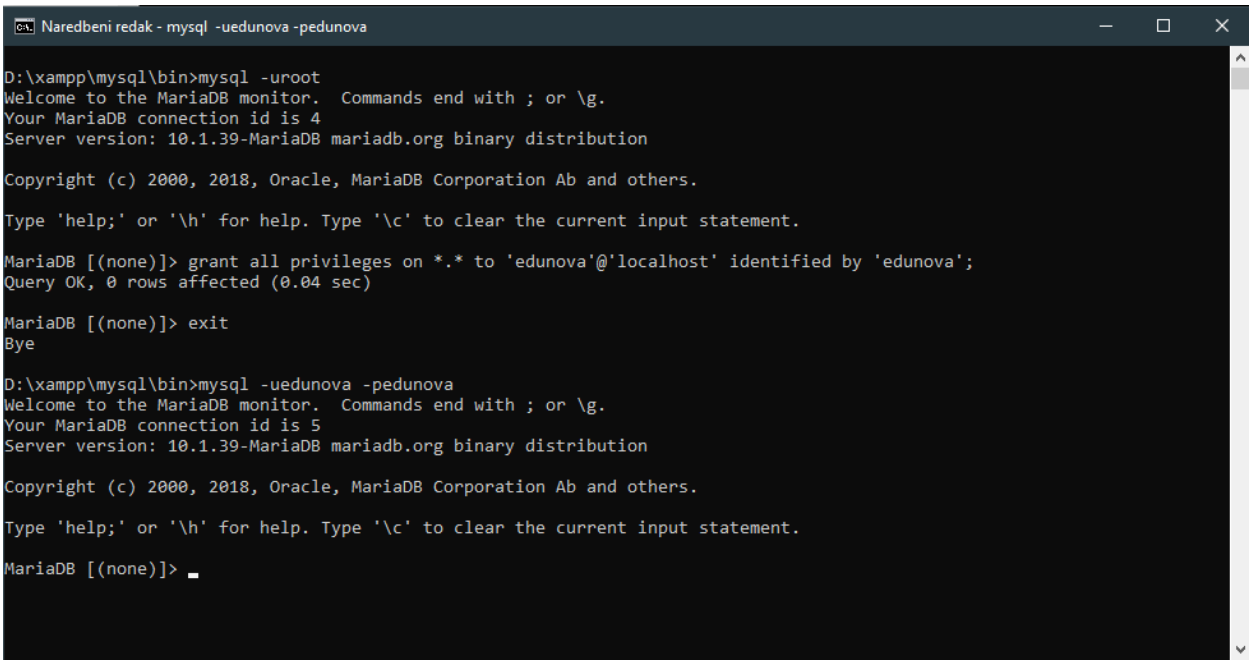
POM je kratica za eng. Project Object Model. Pom.xml datoteka sadrži informacije o projektu i informacije o konfiguraciji za maven za izgradnju projekta kao što su dependencies, direktorij gradnje, izvorni direktorij, testni direktorij izvora, plugin, ciljevi itd. što vidimo sa slike 3.19.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    <modelVersion>4.0.0</modelVersion>
    <groupId>mvc</groupId>
    <artifactId>TaxiSluzbaNetBeans</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>12</maven.compiler.source>
        <maven.compiler.target>12</maven.compiler.target>
    </properties>
    <name>TaxiSluzbaNetBeans</name>
    <dependencies>

        <!-- https://mvnrepository.com/artifact/org.mariadb.jdbc/mariadb-java-client -->
        <dependency>
            <groupId>org.mariadb.jdbc</groupId>
            <artifactId>mariadb-java-client</artifactId>
            <version>2.4.3</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>5.4.4.Final</version>
        </dependency>
        <dependency>
            <groupId>org.mindrot</groupId>
            <artifactId>jbcrypt</artifactId>
            <version>0.4</version>
            <type>jar</type>
        </dependency>
        <dependency>
            <groupId>com.github.lgooddatepicker</groupId>
            <artifactId>LGoodDatePicker</artifactId>
            <version>10.3.1</version>
        </dependency>
    </dependencies>
</project>
```

**Slika 3.19.** pom.xml

# 4. POKRETANJE APLIKACIJE

Prije svakog pokretanja aplikacije trebamo startati server naše baze pomoću XAMPP control panela kako bi aplikacija radila. Prvo trebamo dati sve ovlasti u MariaDB, u cmd-u (eng. Command Prompt) se trebamo pozicionirati na adresu gdje nam je xampp instaliran kako bi mogli dati sve ovlasti. Ulazak u MariaDB nam omogućuje naredba „mysql –uroot“. Komandom „grant all privileges“ dajemo sve ovlasti te postavljamo username i password s kojim ćemo ulaziti u bazu bez „-uroot“ što vidimo na Slici 4.1.




**Slika 4.1** Davanje svih ovlasti i postavljanje username-a i password-a

Nakon ulaska u MariaDB napravimo našu bazu podataka komandom „create database taxisluzba_hib character set utf8 collate utf8_general_ci;“. Struktura komande je sljedeća: create database [željeno ime baze] [kodiranje unicode-om]. Zatim možemo provjeriti je li baza uspješno kreirana sa komandom „show databases;“ što vidimo na Slici 4.2. nakon čega možemo pokrenuti Apache NetBeans.


**Slika 4.2** Prikaz baze u MariaDB-u

U Apache NetBeansu, kada prvi put postavljamo aplikaciju moramo otići u hibernate.cfg.xml datoteku te postaviti property „hbm2ddl.auto“ na create-drop što vidimo sa Slike 4.3.. kako bi hibernate napravio shemu baze. Isto tako moramo navesti naše mapirane klase što vidimo sa Slike 4.4.

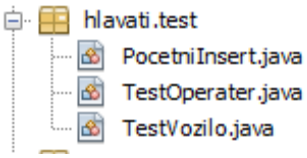
```
<!--
    validate: validate the schema, makes no changes to the database.
    update: update the schema.
    create: creates the schema, destroying previous data.
    create-drop: drop the schema at the end of the session.
    -->
<property name="hbm2ddl.auto">create-drop</property>
```

**Slika 4.3** Kreiranje sheme baze kod prvog pokretanja aplikacije

```
<mapping class="hlavati.model.Vozilo" />
<mapping class="hlavati.model.Vozac" />
<mapping class="hlavati.model.Vozi" />
<mapping class="hlavati.model.Voznja" />
<mapping class="hlavati.model.Operater" />
```

**Slika 4.4** Navođenje naših mapiranih klasa

U paketu test nalaze se tri testne klase prikazane Slikom 4.5. koje zapravo služe kod prvog pokretanja aplikacije tako što popunjavaju bazu nekim početnim podacima, njih je potrebno samo jednom pokrenuti u našoj Start.java klasi kao na Slici 4.6.




**Slika 4.5** Paket test

```
18    public class Start {
19
20        public static void main(String[] args) {
21            PocetniInsert.unesi();
22            TestVozilo.test();
23            TestOperater.test();
24    //        new SplashScreen().setVisible(true);
25
26        }
27    }
```

**Slika 4.6** Početno popunjavanje baze podacima pomoću klasa iz paketa test

Zatim je potrebno ponovo otići u hibernate.cfg.xml datoteku te promjeniti property „hbm2ddl.auto“ sa create-drop na update što vidimo sa Slike 4.7.. te u Start.java klasi zakomentirati/obrisati klase za unošenje podataka u bazu i pokrenuti aplikaciju sa SplashScreen klasom.

```
<property name="hbm2ddl.auto">update</property>
```

**Slika 4.7** U hibernate.cfg.xml property promjenjen sa create-drop na update

# 5. ZAKLJUČAK

Zadatak ovog rada bio je napraviti desktop aplikaciju za taxi službe koja omogućuje korisniku CRUD baze podataka. Aplikacija je pisana u objektno orijentiranom jeziku Java i pokreće se iz okruženja Apache NetBeans. Zadatak nije bio lagan, ali uz dosta proučavanja i istraživanja tehnologija se uspješno realizirao. Prvi dio ukratko opisuje uvod i ideju završnog rada. U drugom djelu završnog rada daje se uvid i opis u korištene tehnologije koje su se koristile za izradu ovog završnog rada. Unutar trećeg djela ovog završnog rada je objašnjena struktura same aplikacije, pregled klasa je olakšala MVC struktura te neki dodatni paketi. Naravno, mogla se koristiti bilo koja struktura, ali MVC mi se najviše svidjela zbog svoje jednostavnosti i popularnosti. Isto tako objašnjeno je korištenje svake klase i formi što ujedno objašnjava funkcionalnost i korištenje aplikacije. U četvrtom djelu detaljno je objašnjeno kako pripremiti aplikaciju za rad te kako je uspješno pokrenuti. Aplikacija bi se mogla više nadograditi tako što bi se kod pogrešnog unosa podatka obojao label u crveno i kada se podatak ispravi u zeleno. Mogao bi se unaprijediti sami dizajn cijele aplikacije, bolje oblikovati gumbove te dodati neke efekte kod korištenja aplikacije.

# LITERATURA


[1] »MariaDB Foundation,« lipanj 2021. [Mrežno]. Dostupno: https://mariadb.org/.

[2] »Apache Friends,« lipanj 2021. [Mrežno]. Dostupno: https://www.apachefriends.org/index.html.

[3] »Apache NetBeans,« lipanj 2021.. [Mrežno]. Dostupno: https://netbeans.apache.org/.

[4] »Java Oracle,« lipanj 2021.. [Mrežno]. Dostupno: https://www.java.com/en/.

[5] »Apache Maven,« lipanj 2021. [Mrežno]. Dostupno: https://maven.apache.org/.

[6] »Hibernate,« lipanj 2021. [Mrežno]. Dostupno: https://hibernate.org/.

[7] »Microsoft .NET MVC,« lipanj 2021. [Mrežno]. Dostupno: https://dotnet.microsoft.com/apps/aspnet/mvc.

# SAŽETAK

U ovom završnom radu razvijena je desktop aplikacija za taxi službe. Zadatak je realiziran pomoću programskog jezika Java, okruženja Apache NetBeans te baze MariaDB. Aplikacija omogućuje korisniku CRUD baze što znači da može napraviti, pročitati, promjeniti i obrisati podatke u bazi kao što su vozač, vozilo, vozi (koji vozač vozi koje vozilo) i vožnja. Ovaj završni rad objašnjava tehnologije koje su korištene tijekom izrade aplikacije, također je objašnjeno kako pripremiti i pokrenuti aplikaciju.

**Ključne riječi:** Apache NetBeans, CRUD, desktop aplikacija, Java, MariaDB

## ABSTRACT

## JAVA APPLICATION FOR TAXI SERVICES

In this final thesis, a desktop application for taxi services was developed. The task was realized by using the Java programming language, the Apache NetBeans environment and the MariaDB database. The application allows user to CRUD database which means it can create, read, update and delete data such as driver, vehicle, drive (which driver drives which vehicle) and drives. This final thesis explains the technologies used during the application development, it is also explained how to prepare and run the application.



**Keywords:** Apache NetBeans, CRUD, desktop application, Java, MariaDB

# ŽIVOTOPIS

Luka Hlavati rođen je u Našicama 23.8.1998. godine. Pohađao je osnovnu školu Kralja Tomislava u Našicama. Završio je srednju školu Isidora Kršnjavog Našice, Elektrotehnički smjer. Sudjelovao je na županijskom natjecanju iz Osnova Elektronike u Vukovaru tijekom drugog razreda. Završetkom srednje škole upisuje se na FERIT Osijek, preddiplomski stručni studij Elektrotehnika, smjer Informatika. Hobi mu je nogomet i tenis.