SVEUČILIŠTE U MOSTARU
FAKULTET STROJARSTVA RAČUNARSTVA I ELEKTROTEHNIKE

Programirnje u Javi

Projektni zadatak: HAIR SALON

Jelena Petrušić

Mila Lovrić

Laura Jerkušić

Akademska godina 2021/2022.

PROJEKTNI ZADATAK:

Opis problema:

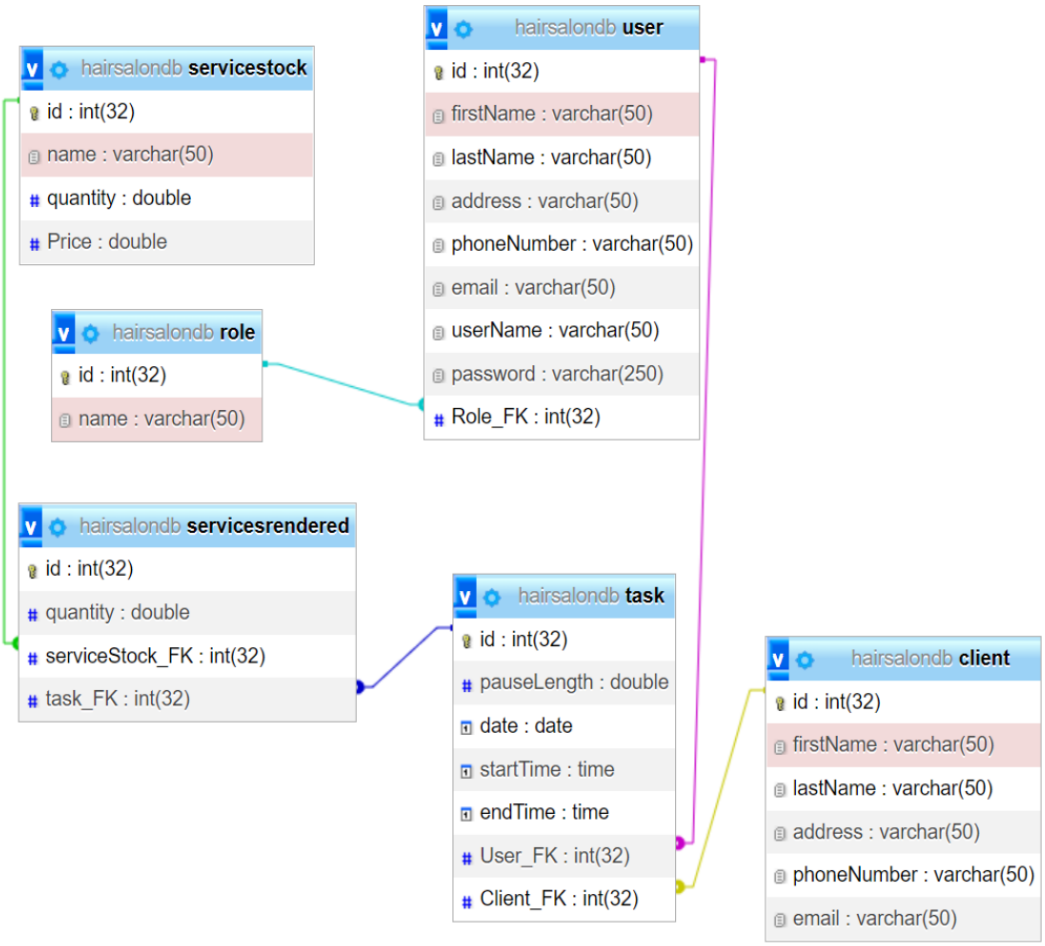
Problem s kojim se susreću mnogobrojne tvrtke, pa tako i saloni poput Frizerskog salona koji želimo obraditi u ovom primjeru je evidencija rezerviranih termina.
Kako bi došli do idealnog rješenja potrebno je isti gledati sa stajališta osobe, tj. Vlasnika koji posjeduje jedan takav salon.
Zamisao ovog projekta je da se olakša poslovanje jednog manjeg salona koji ima veliku zainteresiranost za uslugama koje nudi.
Tako smo napravile i zamislile da bude jedan Admin(Vlasnik) koji vodi računa o poslovanju svog salona, vodi brigu o zadacima koji su zamišljeni da njegovi radnici(u ovom slučaju frizeri) odrade, te o samim radnicima.
Naša aplikacija omogućava olakšano vođenje takvog vida posla. Admin ima infomacije o uslugama, korisnicima, klijentima i zadacima, te ih je moguće kao admin uređivati/brisati/dodavai. Dok User(Radnici) mogu pregledati i urediti svoj profil, te pregledati prošlost svojih zadataka.

1. Modeliranje podataka

Prvi dio zadatka se radi u .

- *Poglavlje 1.1. (Relacijski model podataka) - potrebno je staviti sliku relacijskog modela kreiranog u MS Accessu.*
- *Poglavlje 1.2. (Opis tablica u relacijskom modelu) - potrebno je navesti i opisati sve tablice (i atribute unutar tablica) iz dobivenog modela te na kraju priložiti sliku*
- *Poglavlje 1.3. (Aplikacija u Javi) - sadrži screenshotove kreiranih formi Javi.*

1.1. Relacijski model podataka



1.2. Opis tablica u relacijskom modelu

role

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
Id	PK,NOT NULL	*Jedinstveni indetifikator*
name	NOT NULL	*Naziv role*

user

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
id	PK, NOT NULL	*Jedinstveni indetifikator*
firstName	NOT NULL	*Ime korisnika*
lastName	NOT NULL	*Prezime*
address	NOT NULL	*Adresa*
phoneNumber	NOT NULL	*Broj telefona*
email	NOT NULL	*E-mail*
userName	NOT NULL	*Korisničko ime*
password	NOT NULL	
Role_FK	FK, NOT NULL	*Rola koja je dodjeljena*

task

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
Id	PK,NOT NULL	*Jedinstveni indetifikator*
pauseLength	NOT NULL	*Duzina pauze*
date	NULL	*Datum izvršenja naloga*
startTime	NOT NULL	*Početak pauza*
endTime	NOT NULL	*Kraj pauze*
User_FK	FK, NOT NULL	*ID radnika koji je zadužen za usluga*
Client_FK	FK, NOT NULL	*ID klijenta kod kojeg se usluga obavlja*

client

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
Id	PK,NOT NULL	*Jedinstveni indetifikator*
firstName	NOT NULL	*Ime klijenta*
lastName	NOT NULL	*Prezime klijenta*
address	NOT NULL	*Adresa*
phoneNumber	NOT NULL	*Broj telefona*
email	NOT NULL	*E-mail*

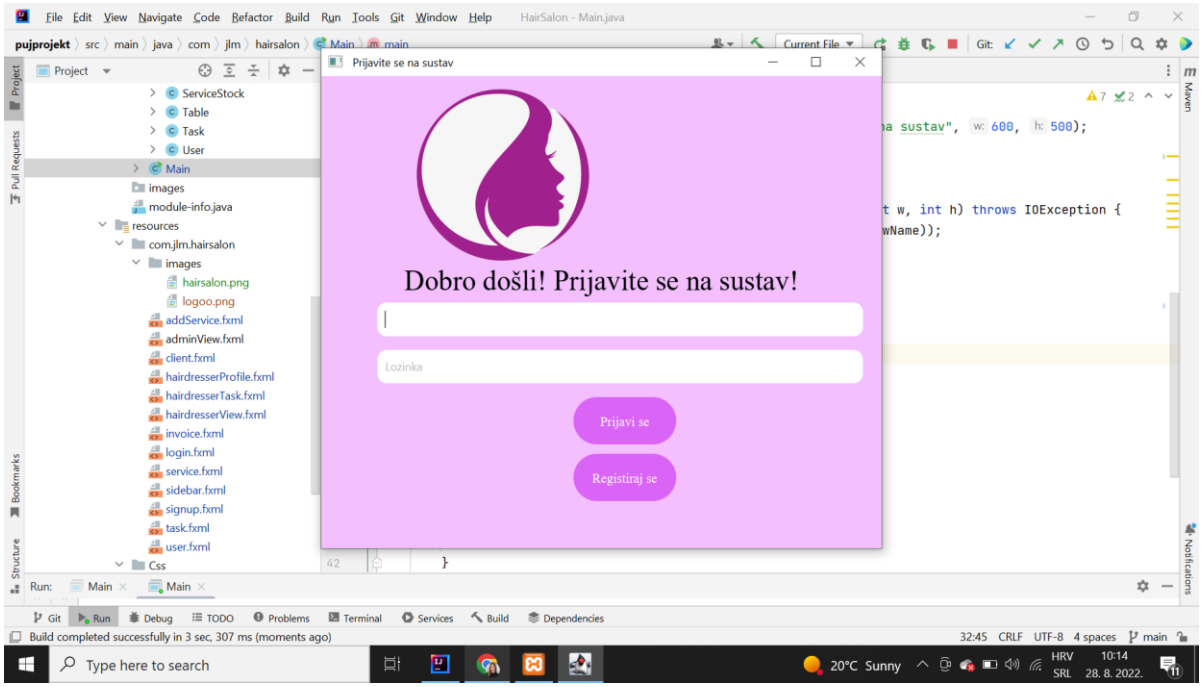
servicestock

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
Id	PK,NOT NULL	*Jedinstveni indetifikator*
name	NOT NULL	*Naziv usluge*
quantity	NOT NULL	*Količina usluga*
Price	NOT NULL	*Cijena*

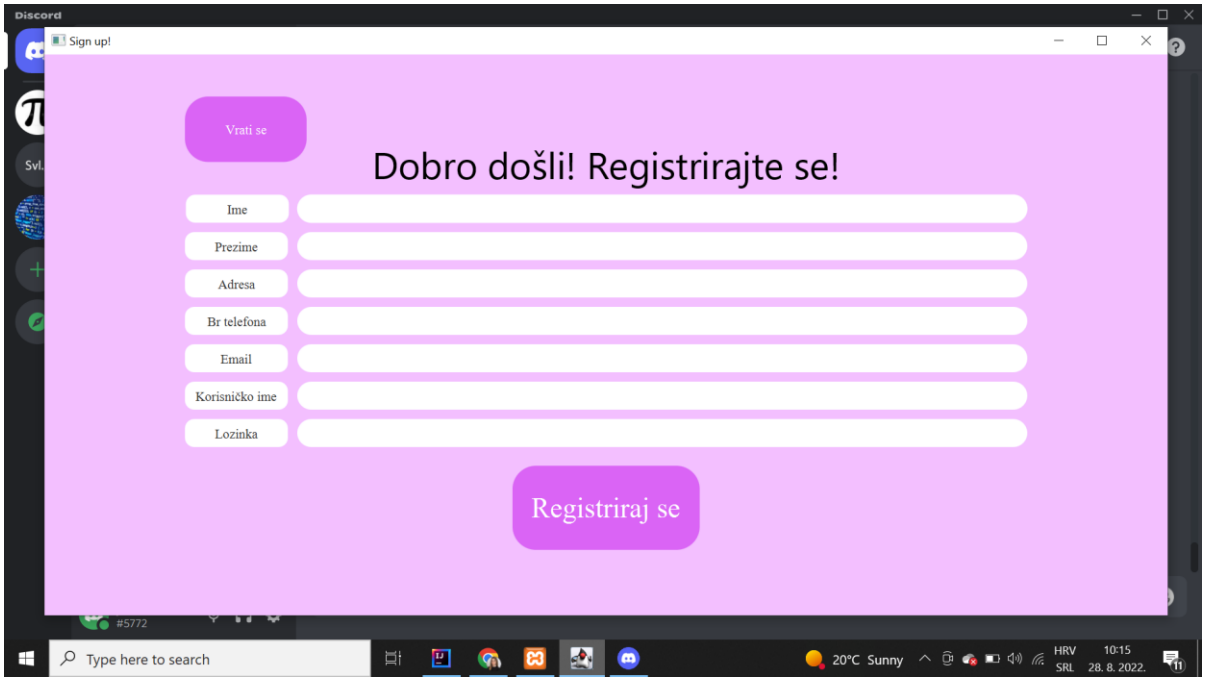
servicesrendered

Naziv atributa	Integritet *(PK/FK, NULL / NOT NULL)*	Kratki opis atributa
Id	PK,NOT NULL	*Jedinstveni indetifikator*
quantity	NOT NULL	*Naziv role*
serviceStock_FK	FK,NOT NULL	*Pružene usluge*
task_FK	FK,NOT NULL	*ID zadatka na kojem se usluga vrši*

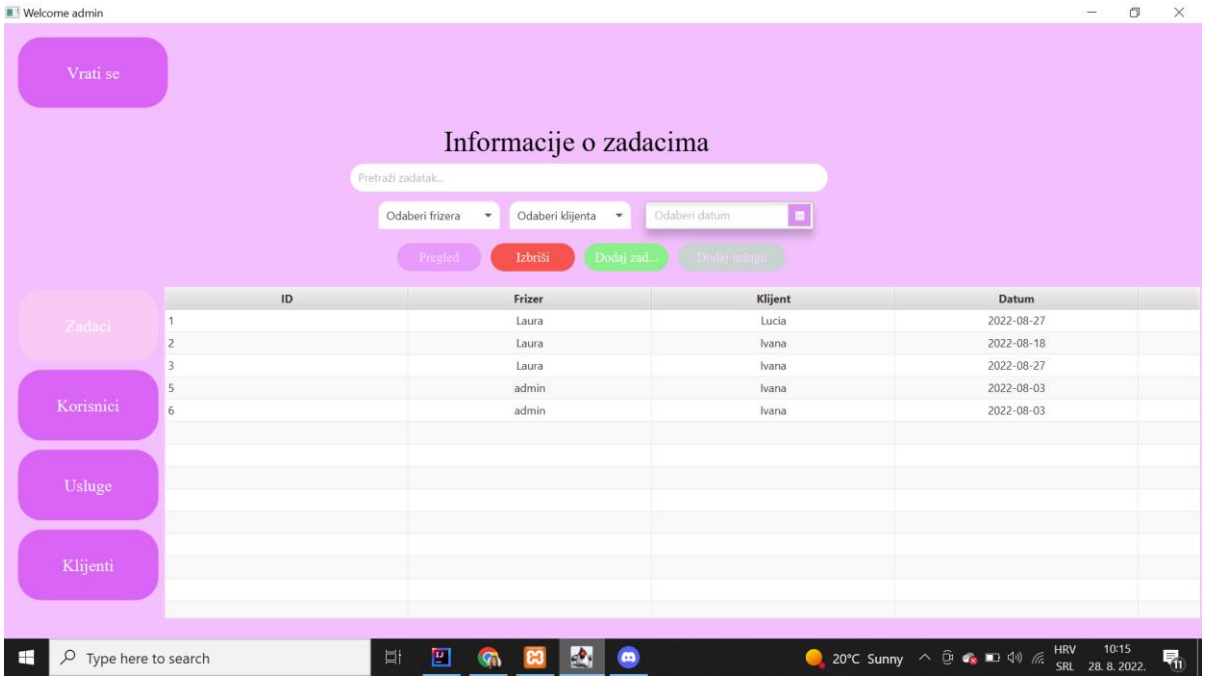
1.3. Aplikacija u Javi



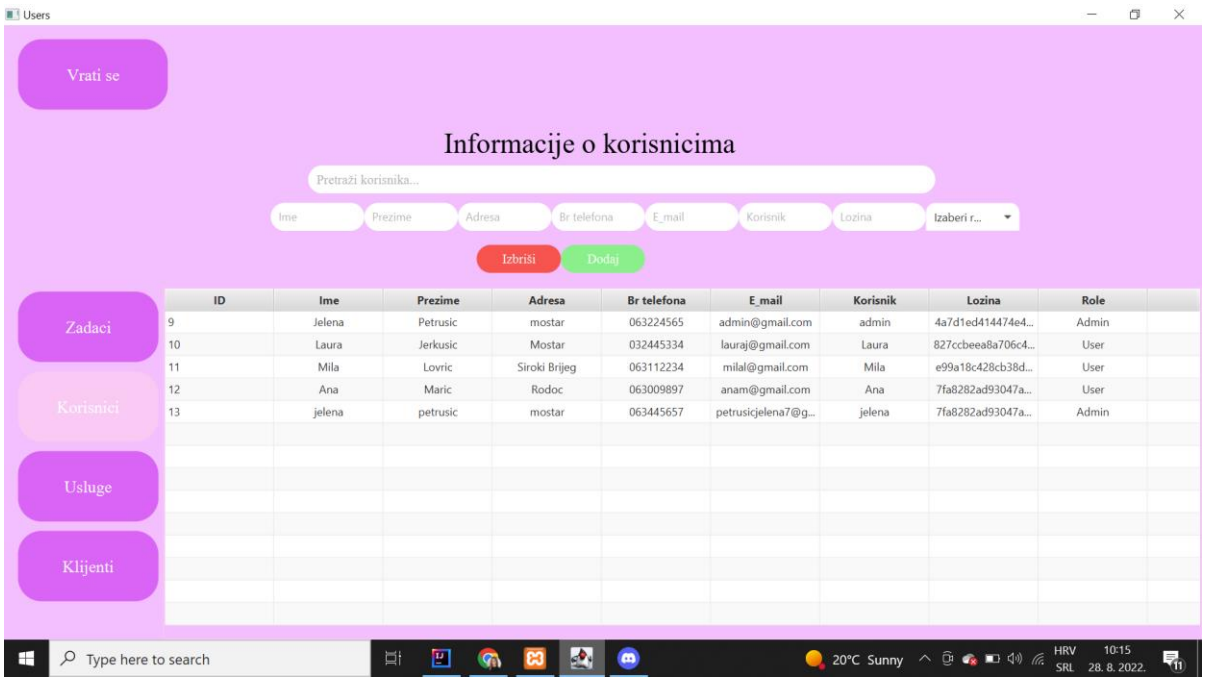
Slika 1. Prijava na sustav



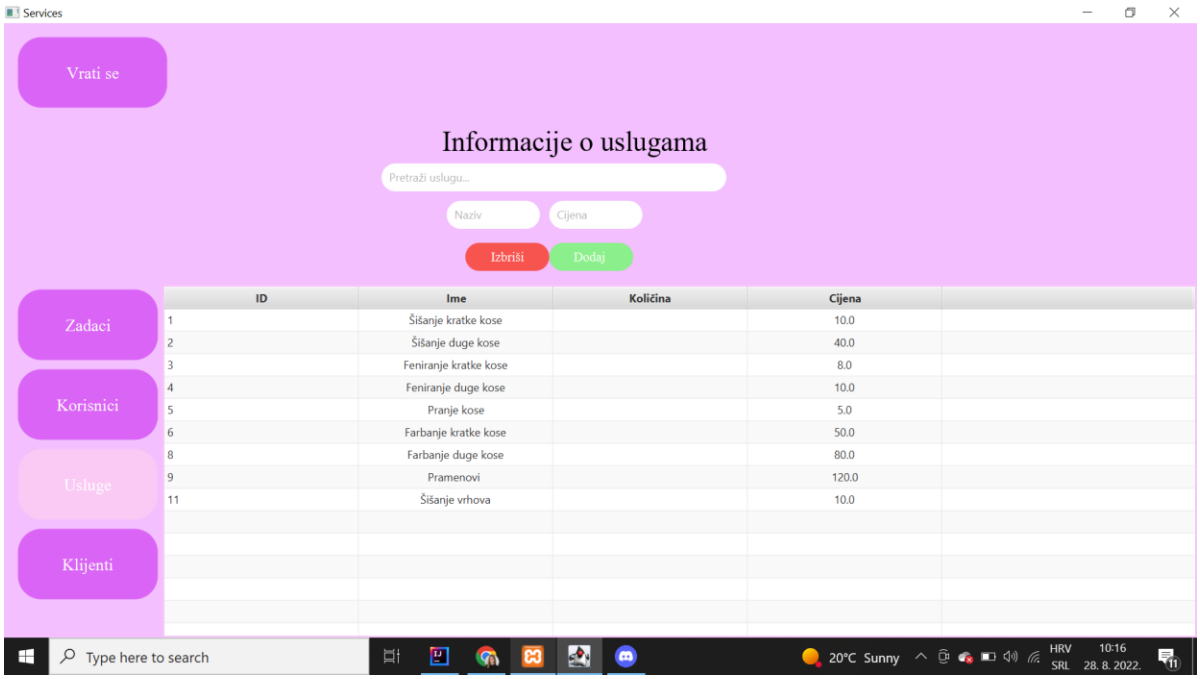
Slika 2. Registracija korisnika



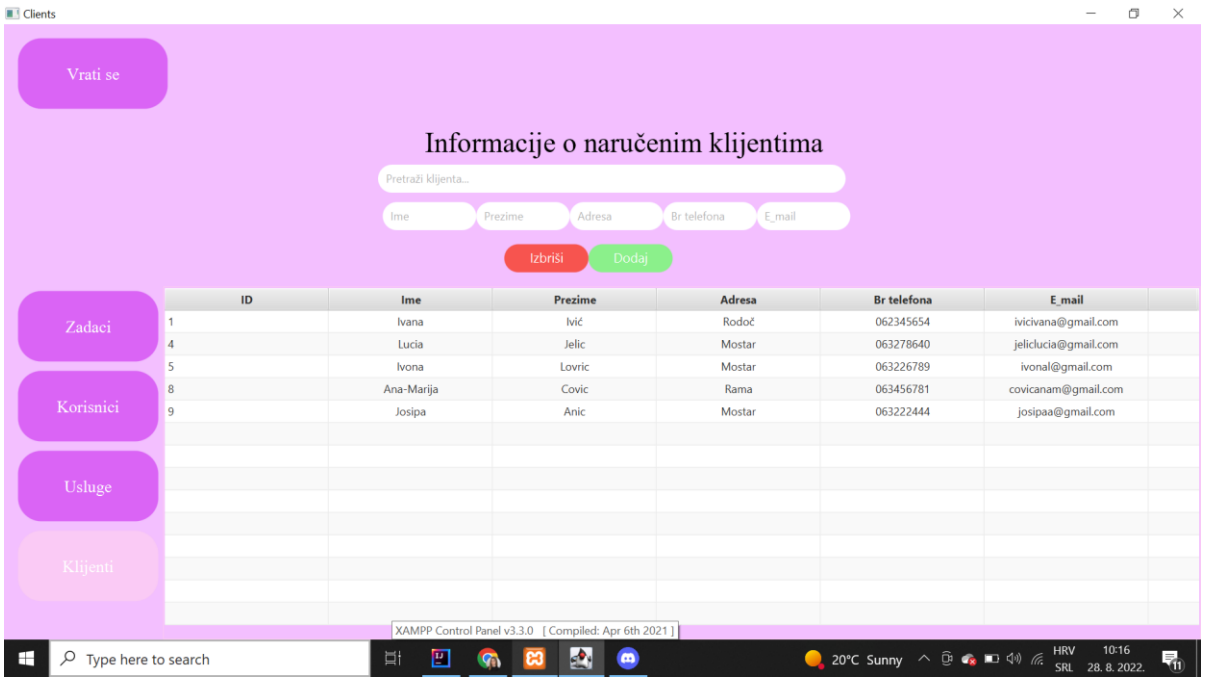
Slika 3. Pogled admina..informacije o zadacima



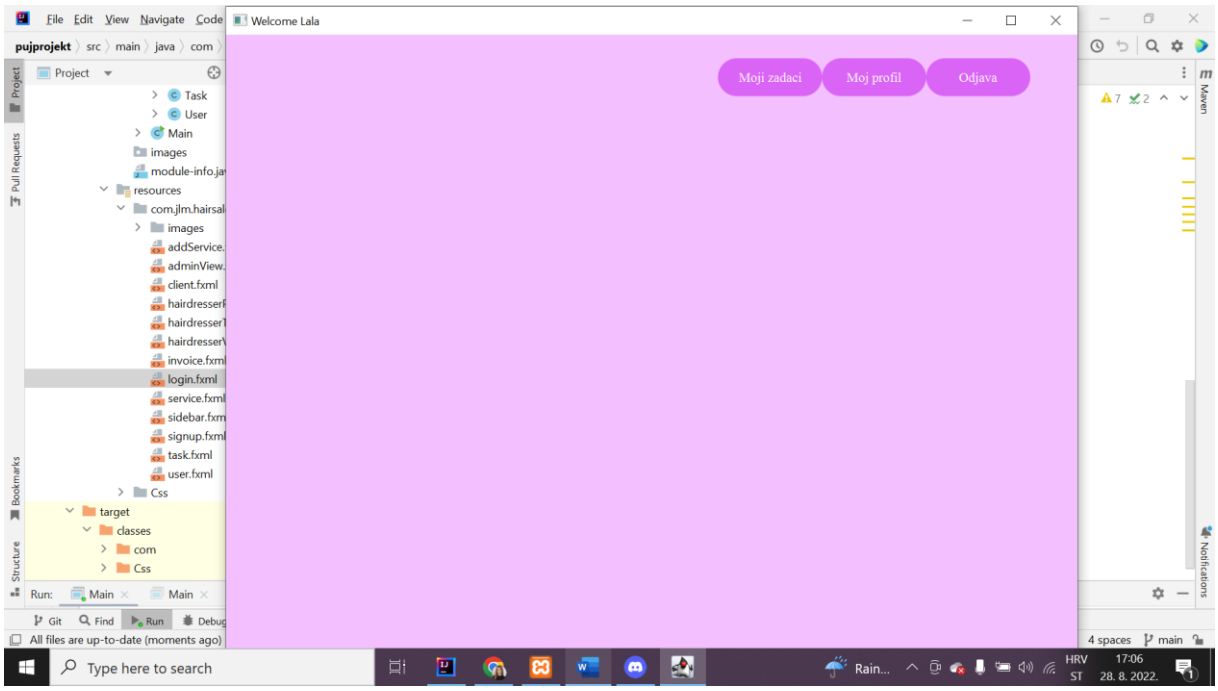
Slika 4. Pogled admina..informacije o korisnicima



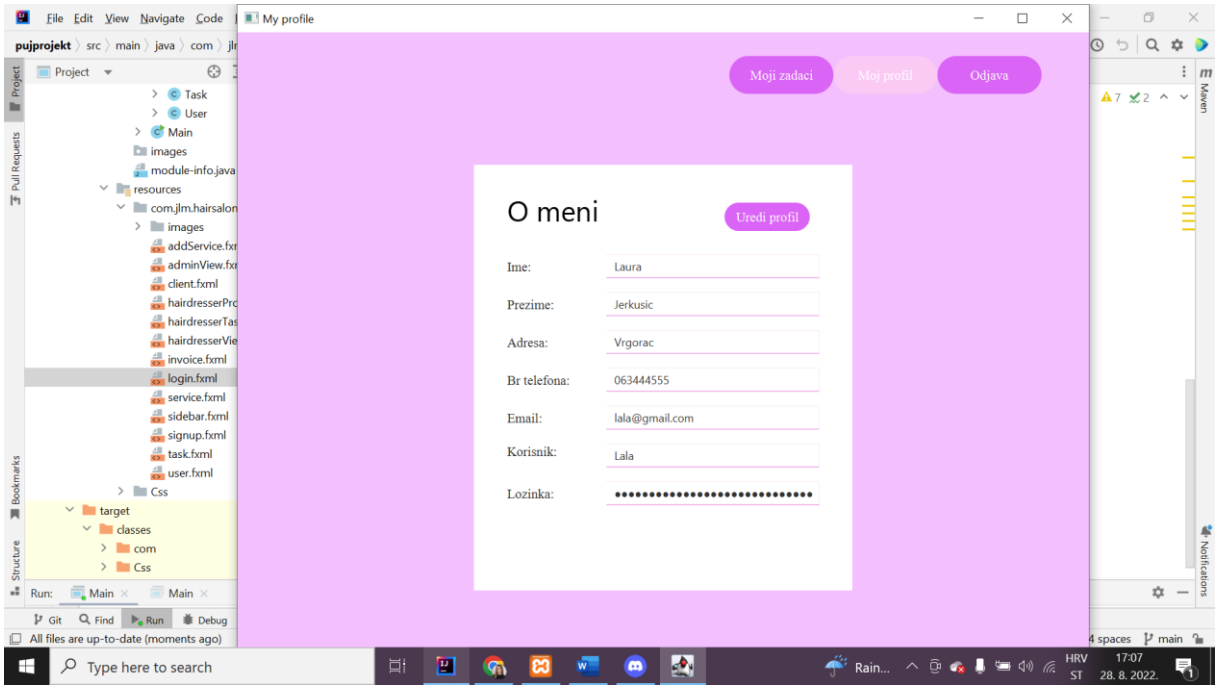
Slika 5. Pogled admina..informacije o uslugama



Slika 6. Pogled admina..informacije o naručenim klijentima



Slika 7. Pogled Usera (radnika)



Slika 8. O korisniku

2. Model programske podrške

Drugi dio zadatka se odnosi na modeliranje programske podrške, detaljno čete raspisati sve klase i na kraju priložiti dijagram klase koji opisuje izgled programske podrške:

- *Napraviti class dijagram*
- *Opisati metode koje se nalaze unutar svake klase*

U navedenim klasama potrebno je opisati :

1) *Paketi – kratko opisati paket vaše programske podrške*
2) *Klase – kratko opisati klase koje koristite (ukratko za što će se klasa koristiti*
   a. *Atribute – kratko pisati atribute koje se nalaze unutar klase;*
   b. *Metode – kratko opisati metode koje se koriste za što služe i opišite njihov povratni tip;*

2.1. Paketi

Ovaj projekt se sastoji od dva paketa (com.jlm.hairsalon i resource).
Svaki od tih paketa imaju podpaketa:
1.com.jlm.hairsalon: controller i model, te klasa Main koja pokreće program
2.resource: css i com.jlm.hairsalon

2.2. Klase

Dijagram klasa
...

2.2.1. Klasa Client

Koristi se za pohranu podataka vezanih za klijente i atribute koje njima pripadaju.

2.2.1.1. Atributi

```
. @Entity(type = "INTEGER",size = 32,primary = true)
int id;

@Entity(type = "VARCHAR",size = 50)
String firstName;

@Entity(type = "VARCHAR",size = 50)
String lastName;

@Entity(type = "VARCHAR",size = 50)
String address;

@Entity(type = "VARCHAR",size = 50)
String phoneNumber;

@Entity(type = "VARCHAR",size = 50)
String email;
```

2.2.1.2. Metode

```
public int getId() { return id; }

public void setId(int id) {
    this.id = id;
}

public String getFirstName() {
    return firstName;
}

public void setFirstName(String firstName) {
    this.firstName = firstName;
}

public String getLastName() {
    return lastName;
}

public void setLastName(String lastName) {
    this.lastName = lastName;
}

public String getAdress() {
    return address;
}

public void setAdress(String adress) {
    this.address = adress;
}

public String getPhoneNumber() {
    return phoneNumber;
}

public void setPhoneNumber(String phoneNumber) {
    this.phoneNumber = phoneNumber;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String toString()
{
    return this.firstName;
}
```

2.2.2 Klasa CryptMD5

Koristi se za enkripciju stringova.(pohrana stringova)

2.2.2.1. Atributi klase

```
private static MessageDigest md;
```

2.2.2.2. Metode klase

```
public static String cryptWithMD5(String pass)-enkriptira prosljeđeni
string i vraća enktriptirani
```

2.2.3 Klasa HairdresserView

Ova klasa služi za pohranu zadataka za pojedine radnike.

2.2.3.1. Atributi klase

```
private int taskId;
private int clientId;
private String clientName;
private String clientAddress;
private String date;
```

2.2.3.2. Metode klase

```
public int getTaskId() {
    return taskId;
}

public void setTaskId(int taskId) {
    this.taskId = taskId;
}

public int getClientId() {
    return clientId;
}

public void setClientId(int clientId) {
    this.clientId = clientId;
}

public String getClientName() {
    return clientName;
}

public void setClientName(String clientName) {
    this.clientName = clientName;
}

public String getClientAddress() {
    return clientAddress;
}

public void setClientAddress(String clientAddress) {
    this.clientAddress = clientAddress;
}

public String getDate() {
    return date;
```

```
}

public void setDate(String date) {
    this.date = date;
}
```

2.2.4 Klasa Role

Koriste se za pohranu rola aplikacije.

2.2.4.1. Atributi klase

```
@Entity(type = "INTEGER", size = 32, primary = true)
int id;

@Entity(type = "VARCHAR",size = 50)
String name;
```

2.2.4.2. Metode klase

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String toString()
{
    return this.name;
}
```

2.2.5 Klasa ServicesRendered

Nasljeđuje klasu Table. Služi nam za operacije nad bazom. Nasljeđuje klasu Table.

2.2.5.1. Atributi klase

```
@Entity(type = "INTEGER",size=32,primary = true)
int id;

@Entity(type = "DOUBLE",size = 25)
Double quantity;

@Entity(type = "INTEGER",size=32)
@ForeignKey(table = "ServiceStock",attribute = "id")
int serviceStock_FK;
```

```
@Entity(type = "INTEGER",size = 32)
@ForeignKey(table = "Task",attribute = "id")
int task_FK;
```

2.2.5.2. Metode klase

```
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public Task getTask()throws Exception { return (Task)
Table.get(Task.class,task_FK); }

public void setTask_FK(int Task_FK)
{
    this.task_FK=Task_FK;
}

public ServiceStock getServiceStock()throws Exception { return
(ServiceStock) Table.get(ServiceStock.class,serviceStock_FK);  }

public void setServiceStock_FK(int serviceStock_FK) {
    this.serviceStock_FK = serviceStock_FK;
}

public Double getQuantity() { return quantity; }

public void setQuantity(Double quantity) { this.quantity = quantity; }

public int getServiceStock_FK() {
    return serviceStock_FK;
}

public int getTask_FK() {
    return task_FK;
}
```

2.2.6 Klasa ServiceStock

Također nasljeđuje klasu Table. Služi za pohranjivanje svih usluga koje su obavljene.

2.2.6.1. Atributi klase

```
@Entity(type="INTEGER", size=32, primary = true)
int id;

@Entity(type="VARCHAR", size=50)
String name;

@Entity(type="DOUBLE", isnull = false)
Double quantity;
```

```
@Entity(type="DOUBLE", size=25)
Double Price;
```

2.2.6.2. Metode klase

```
public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Double getQuantity() {
        return quantity;
    }

    public void setQuantity(Double quantity) {
        this.quantity = quantity;
    }

    public Double getPrice() {
        return Price;
    }

    public void setPrice (Double price) {
        this.Price = Price;
    }

    public String toString()
    {
        return this.name;
    }
}
```

2.2.7 Klasa Table

2.2.7.1. Atributi klase

2.2.7.2. Metode klase

```
private static String  getTableName(Class cls) – vraća ime tablice
private static String getAttributeName(Field field) – vraća ime atributa
public static boolean create(Class cls) throws SQLException – kreira
tablicu u bazi podataka
public void save() throws Exception – sprema redak u tablici
public void update() throws Exception – ažurira redak u  tablici
public static Object get(Class cls, int id) throws Exception – vraća
objekat iz neke tablice u bazi
```

```
public static List<?> list(Class cls) throws Exception – vraća listu redaka
u nekoj tablici
public static List<Invoice> listTempTable(ResultSet rs) throws Exception -
vraća listu svih redaka u tablici HairdresserView
public static List<ServicesRendered> servicesRendered(ResultSet rs) throws
Exception – vraća listu svih redaka u tablici ServicesRendered
public static List<HairdresserView> HairdresserTasks(ResultSet rs) throws
Exception – vraća listu redaka u tablici HairdresserView
```

2.2.8 Klasa Task

Nasljeđuje klasu Table. Koristi se za pohranu podataka vzanih za oređeni zadatak.

2.2.8.1. Atributi klase

```
@Entity(type="INTEGER",size = 32, primary = true)
int id;

@Entity(type="DOUBLE",size = 25)
Double pauseLength;

@Entity(type = "DATE",isnull = false)
Date date;

@Entity(type = "TIME")
Time startTime;

@Entity(type = "TIME")
Time endTime;

@Entity(type="INTEGER",size=32)
@ForeignKey(table ="User", attribute = "id")
int User_FK;

@Entity(type = "INTEGER",size = 32)
@ForeignKey(table = "Client",attribute = "id")
int Client_FK;
```

2.2.8.2. Metode klase

```
public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public Double getPauseLength() {
        return pauseLength;
    }

    public void setPauseLength(Double pauseLength) {
        this.pauseLength = pauseLength;
    }

    public Date getDate() {
        return date;
```

```
    }

    public void setDate(Date date) {
        this.date=date;
    }

    public Time getStartTime() {
        return startTime;
    }

    public void setStartTime(Time startTime) {
        this.startTime = startTime;
    }

    public Time getEndTime() {
        return endTime;
    }

    public void setEndTime(Time endTime) {
        this.endTime = endTime;
    }

    public User getUser()throws Exception { return (User)
Table.get(User.class,User_FK); }

    public int getUser_FK() {
        return User_FK;
    }

    public void setUser_FK(int user_FK) {
        User_FK = user_FK;
    }

    public Client getClient()throws Exception { return (Client)
Table.get(Client.class,Client_FK); }

    public void setClient_FK(int client_FK) {
        Client_FK = client_FK;
    }
```

2.2.9 Klasa User

Koristi se za pohranu podataka vezanih za korisnike ove aplikacije. Nasljeđuje klasu Table.

2.2.9.1. Atributi klase

```
@Entity(type="INTEGER", size=32, primary = true)
int id;

@Entity(type="VARCHAR", size=50)
String firstName;

@Entity(type="VARCHAR", size=50)
String lastName;

@Entity(type="VARCHAR", size=50)
String address;
```

```
@Entity(type="VARCHAR", size=50)
String phoneNumber;

@Entity(type="VARCHAR", size=50)
String email;

@Entity(type="VARCHAR", size=50)
String userName;

@Entity(type = "VARCHAR",size=250)
String password;

@Entity(type = "INTEGER",size = 32)
@ForeignKey(table = "Role",attribute = "id")
int Role_FK;
```

2.2.9.2. Metode klase

```
public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getPhoneNumber()
    {
        return this.phoneNumber;
    }

    public void setPhoneNumber(String phoneNumber)
{this.phoneNumber=phoneNumber; }

    public String getEmail()
    {
        return this.email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public int getRole_FK() { return Role_FK; }

    public void setRole_FK(int Role_FK) {
        this.Role_FK=Role_FK;
    }

    public Role getRole()throws Exception { return (Role)
Table.get(Role.class,Role_FK); }

    public String toString() { return this.userName; }

}
```