Java aplikacija za praćenje dostave poštanskih pošiljki

Antunović, Dražen

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:136084

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

JAVA APLIKACIJA ZA PRAĆENJE DOSTAVE POŠTANSKIH POŠILJKI

Završni rad

Dražen Antunović

Osijek, 2022.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 05.09.2022.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Dražen Antunović
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	4759, 07.10.2020.
OIB Pristupnika:	79078314721
Mentor:	Prof. dr. sc. Krešimir Nenadić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Java aplikacija za praćenje dostave poštanskih pošiljki
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Objasniti način rada dostave poštanskih pošiljki (pisma i paketi). Opisati slična postojeća rješenja i usporediti s traženim zahtjevima pred zadanim zadatkom iz ovog rada. Za potrebe aplikacije potrebno je osmisliti i dizajnirati bazu podataka u koju će se pohranjivati svi relevantni podaci (djelatnici, korisnici usluga, pošiljke, praćenje statusa pošilike). Predvidjeti više različitih korisničkih profila za potrebe
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	05.09.2022.
Datum potvrde ocjene od strane Odbora:	07.09.2022.

Potvrda mentora o predaji konačne verzije rada:	Mentor elektronički potpisao predaju konačne verzije.
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 07.09.2022.

Ime i prezime studenta:	Dražen Antunović
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	4759, 07.10.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Java aplikacija za praćenje dostave poštanskih pošiljki**

izrađen pod vodstvom mentora Prof. dr. sc. Krešimir Nenadić

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 1

1.1. Zadatak završnog rada 1

2. SLIČNA RJEŠENJA 3

2.1. DesktopShipper 3

2.2. Courier Management Software 3

2.3. Shipstation 4

2.4. ShippingEasy 5

2.5. Orderhive 5

3. KORIŠTENE TEHNOLOGIJE 7

3.1. MySQL baza podataka kao spremnik 7

3.2. Java kao jezik za izradu aplikacije 7

3.2.1. Swing arhitektura 8

3.2.2. Uporaba *java.sql* paketa 8

3.3. Eclipse IDE razvojno okruženje 8

4. IZRADA BAZE PODATAKA 9

4.1. Definiranje izgleda baze podataka 9

4.1.1. E-R model 9

4.1.2. Relacijski model 10

4.2. Atributi unutar tablica 10

5. DIZAJNIRANJE GRAFIČKOG KORISNIČKOG SUČELJA 13

5.1. Izrada početnog zaslona 14

5.1.1. Korištenje rasporeda 15

5.1.2. Dijaloški okvir prijave/registracije 17

5.1.3. Dijaloški okvir popunjavanja podataka korisnika usluge 17

5.2. Izrada zaslona korisnika usluge 18

5.2.1. Postupak slanja zahtjeva za dostavu 18

5.2.2. Prikaz dostava 18

5.3. Izrada zaslona djelatnika 19

5.4. Izrada zaslona završenih dostava 19

5.4.1. Korištenje *JTable* klase 19

5.5. Izrada zaslona upravitelja 21

6. INTERAKCIJA BAZE PODATAKA S APLIKACIJOM 22

7. PRIKAZ RADA APLIKACIJE 24

7.1. Početni zaslon 24

7.2. Zaslon djelatnika 25

7.3. Promjena i potvrda zaporke 27

7.4. Zaslon djelatnika 28

7.5. Zaslon upravitelja 30

8. ZAKLJUČAK 31

9. LITERATURA 32

SAŽETAK 33

ABSTRACT 34

1. UVOD

U ovom završnom radu opisan je način izrade desktop aplikacije korištenjem Java programskog jezika koja korisniku usluge omogućuje slanje zahtjeva za dostavu poštanskih pošiljki i praćenje statusa istih. Više korisničkih profila obuhvaćenom je aplikacijom, a to su:

Upravitelj (admin)

Djelatnik

Korisnik usluge

Za svakog pojedinog korisnika namijenjen je specifični dio aplikacije, pri čemu se korisnik mora prijaviti (ili registrirati) kako bi mu pristupio i izvršavao određene funkcije unutar svojeg dijela. Na taj način osiguran je ovlašteni pristup.

Bitne teme obuhvaćene završnim radom su:

Izrada baze podataka

Izrada grafičkog korisničkog sučelja

Interakcija baze podataka i aplikacije

Na početku glavnog dijela u drugom poglavlju rada uspoređena je izrađena aplikacija s ostalim sličnim rješenjima koja su dostupna. U sljedećem poglavlju opisane su korištene tehnologije za izradu aplikacije i ostalih komponenata koje omogućuju njen rad. Zatim u idućem poglavlju opisana je baza podataka odnosno izrada E-R dijagrama, izgled pojedinih tablica i značenje pojedinih atributa u tablicama. U poglavlju nakon toga opisan je izgled aplikacije, što uključuje opis korištene arhitekture unutar Java programskog jezika za postizanje željenog izgleda te komunikacija aplikacije sa bazom podataka odnosno kada i kako se izvršavaju određeni upiti koje aplikacija prosljeđuje bazi. Predzadnje poglavlje sadrži rad same aplikacije odnosno njen prikaz nakon čega slijedi zaključak.

1.1. Zadatak završnog rada

Objasniti način rada dostave poštanskih pošiljki (pisma i paketi). Opisati slična postojeća rješenja i usporediti s traženim zahtjevima pred zadanim zadatkom iz ovog rada. Za potrebe aplikacije potrebno je osmisliti i dizajnirati bazu podataka u koju će se pohranjivati svi relevantni podaci (djelatnici, korisnici usluga, pošiljke, praćenje statusa pošiljke). Predvidjeti više različitih

korisničkih profila za potrebe aplikacije. Potrebno je opisati postupak izrade aplikacije kao i tehnologije korištene u postupku izrade.

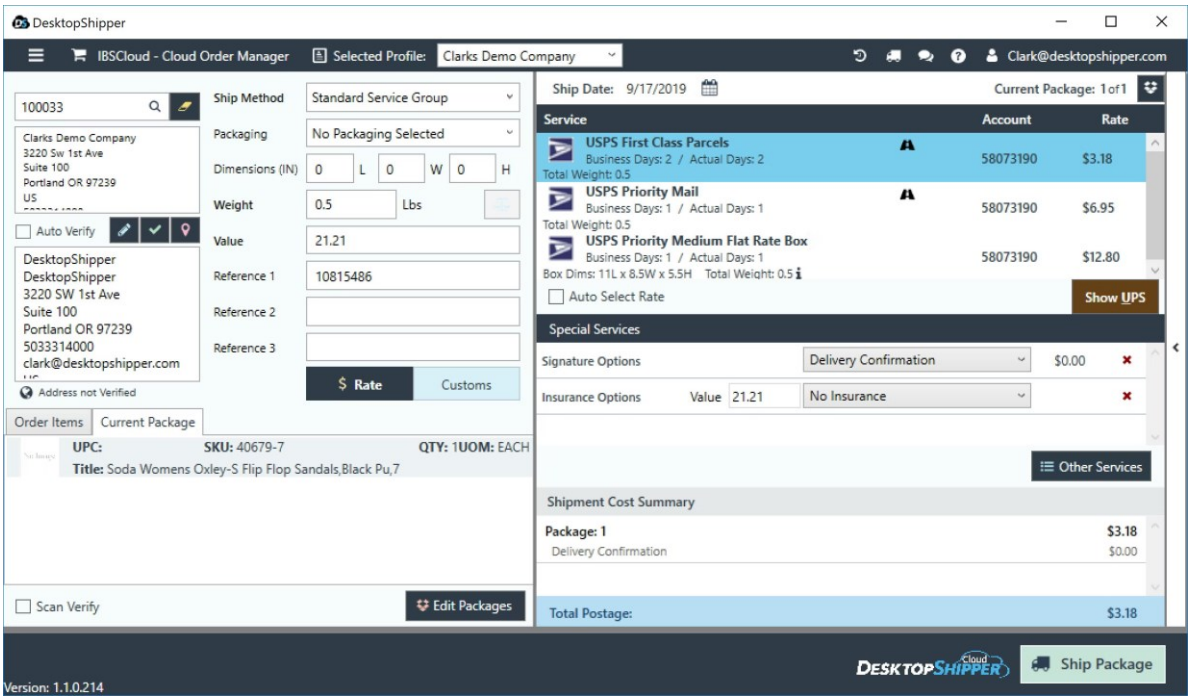
2. SLIČNA RJEŠENJA

U pregledu područja rada navedena su rješenja koja su slična temi ovog završnog rada. Sljedeća rješenja su navedena u poglavljima: 2.1., 2.2., 2.3., 2.4. i 2.5.

2.1. DesktopShipper

Rad aplikacije se temelji na oblaku računala. Tvrtkama je omogućeno upravljanje dostavama. Omogućeno je upravljanje narudžbama, provjeru valjanosti adrese, filtriranje i slično. Administratorima je omogućeno prikupljanje narudžbi s više tržišta u stvarnom vremenu. Menadžerima je omogućeno stvaranje računa prijevoznika sa određenim parametrima. Alati za usmjerivanje korisnicima omogućuje dodavanje postavki i mapiranje načina dostave sa specifičnim servisima prema [1].

Prikaz izgleda aplikacije je na slici 2.1.

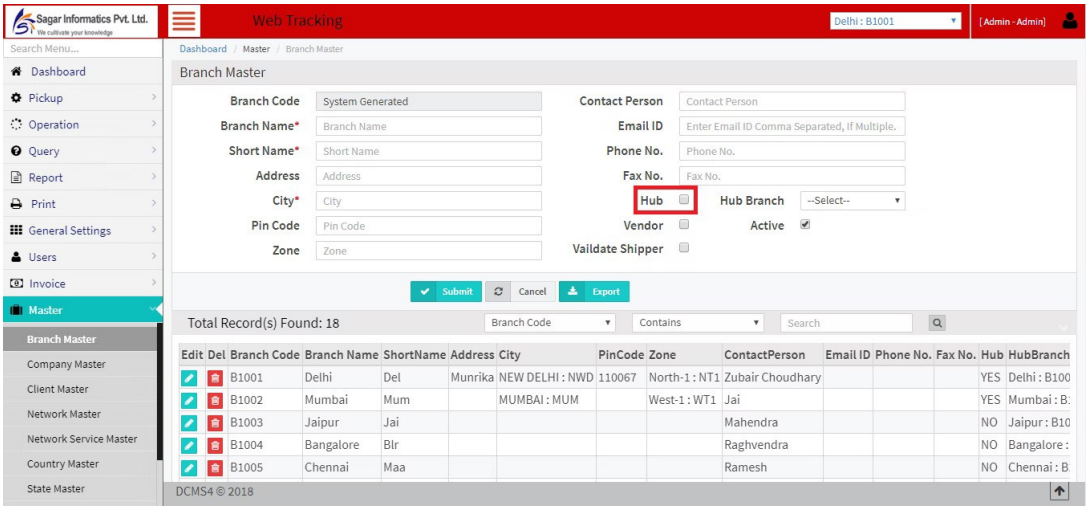


Sl. 2.1. Sučelje DesktopShipper aplikacije prema [1]

2.2. Courier Management Software

Unutar ove aplikacije omogućeno je upravljanje distribucijom paketa za kurirske tvrtke. Nalaze se opcije poput izvještaja, naplate i kontrole nad kurirskim operacijama. Pružen je jasan izvještaj menadžmentu kako koja kurirska tvrtka obavlja zadane poslove distribucije paketa prema [2].

Prikaz izgleda aplikacije je na slici 2.2.

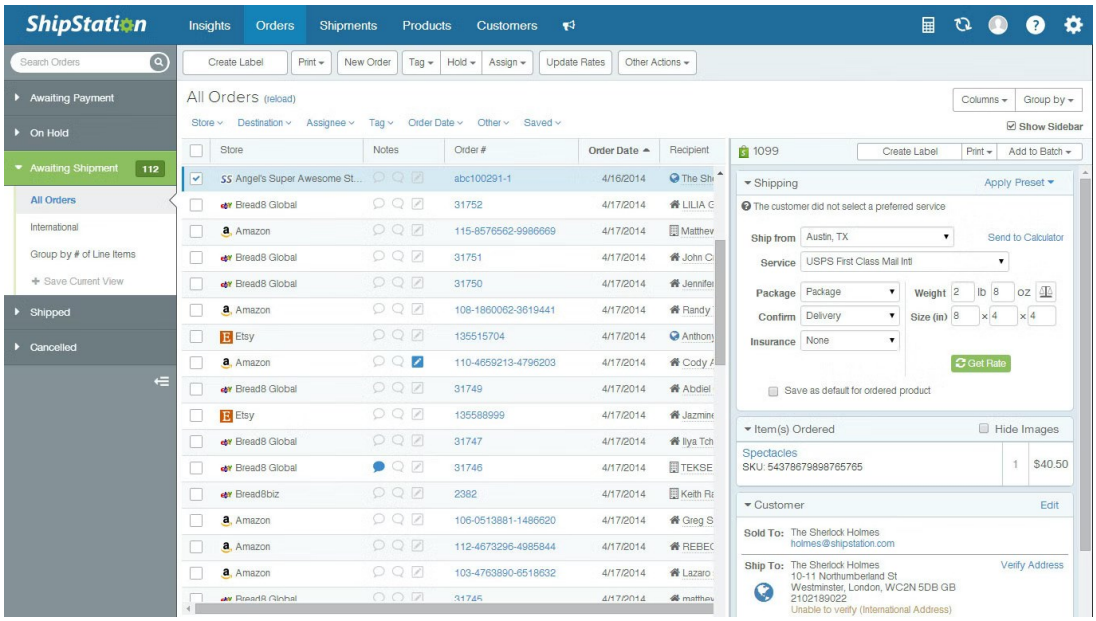


Sl. 2.2. Sučelje Courier Management Software aplikacije prema [2]

2.3. Shipstation

Omogućena je dostava sa više prijevoznika i više kanala. Prema tome, povećana je ušteda vremena i novca te učinkovitost za trgovce. Informacije o narudžbama su objedinjene iz više od 40 prodajnih kanala, pri čemu su postavke za svaku narudžbu unaprijed konfigurirane na temelju određenih parametara te se prave naljepnice za otpremu i otpremnice. Status dostave i podaci o praćenju pojedine trgovine su automatski ažurirani. Nakon izrade naljepnice, sva komunikacija s klijentima je automatizirana. Omogućen je trenutni pregled izvješća o narudžbama prema [3].

Prikaz izgleda aplikacije je na slici 2.3.

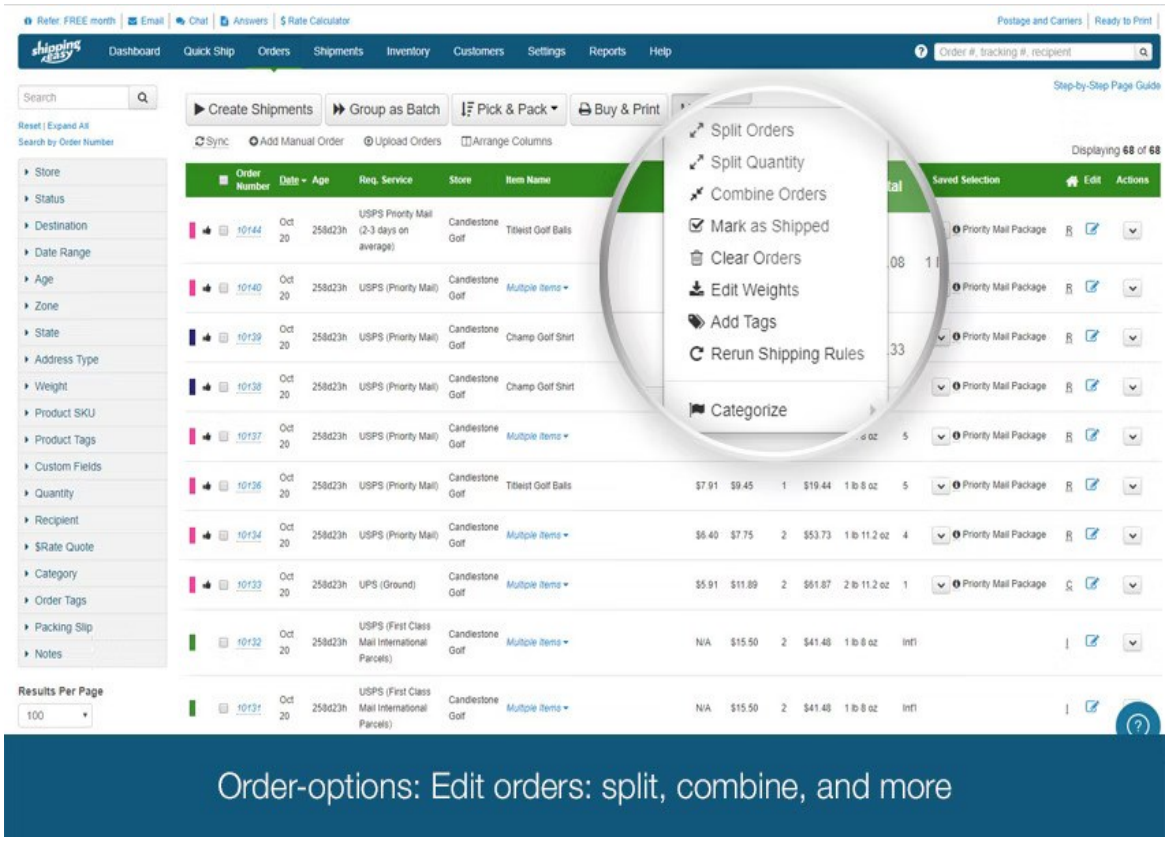


Sl. 2.3. Sučelje ShipStation aplikacije prema [3]

2.4. ShippingEasy

Napravljena je u svrhu korištenja za online prodavače robe i internetskih trgovina zbog automatiziranja brojnih pozadinskih procesa. Omogućena je primjena pravila dostave za plaćanje poštarine uobičajenim vrstama narudžbi, automatsko slanje poruke o statusu praćenja, pojednostavljen proces povrata, upravljanje zalihama i slično. ShippingEasy se integrira s platformama koje korisnici prodaju na mreži i trenutno preuzima narudžbe, obrada narudžbi je brza i točna prema [4].

Prikaz izgleda aplikacije je na slici 2.4.

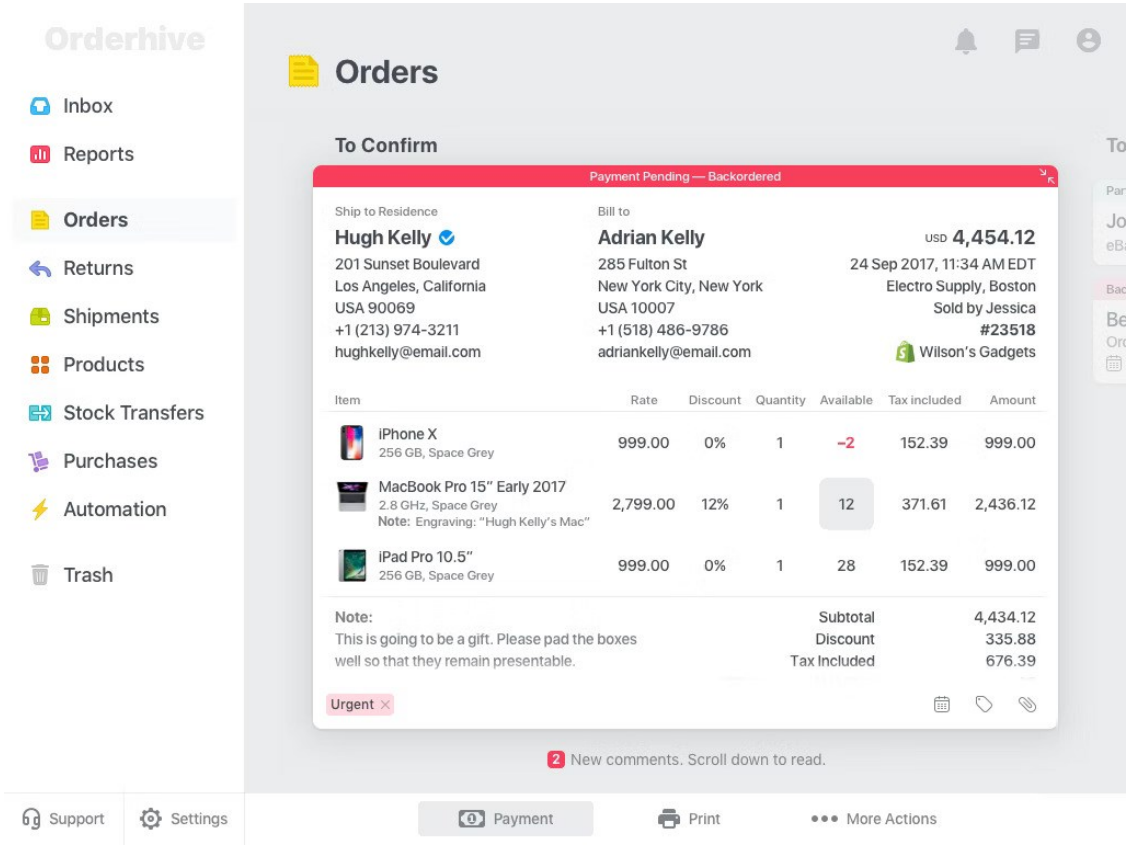


Sl. 2.4. Sučelje ShippingEasy aplikacije prema [4]

2.5. Orderhive

Unutar softvera omogućeno je upravljanje narudžbama, otpremom i zalihama. Posjeduje poslovna rješenja za višekanalnu prodaju i pozadinsko ispunjavanje narudžbi uz održavanje centraliziranog sustava praćenja inventara u stvarnom vremenu. Olakšana je besprijekorna integracija sa vodećim svjetskim tržištima. Rješenja se temelje na SaaS-u (eng. *Software as a Service*). Upravljanje zalihama je skalabilno, što omogućuje klijentima da zadovolje svoje sadašnje i buduće potrebe prema [5].

Prikaz izgleda aplikacije je na slici 2.5.



Sl. 2.5. Sučelje Orderhive aplikacije prema [5]

3. KORIŠTENE TEHNOLOGIJE

Za trajno spremanja podataka korištena je MySQL (eng. *My Structured Query Language*) baza podataka, za izradu same aplikacije korišten je Java programski jezik. Korišteno razvojno okruženje za prevođenje Java programskog koda je Eclipse IDE for Java Developers 2022-03.

3.1. MySQL baza podataka kao spremnik

Kako bi se moglo realizirati spremanje podataka i njihovo ponovno dohvaćanje nakon gašenja aplikacije te njenog ponovnog paljenja potrebno je ostvariti trajni spremnik podataka. Za te potrebe korištena je MySQL baza podataka pohranjena na serveru. Unutar baze podataka spremljeno je više tablica koje će biti pojašnjene u sljedećem poglavlju. Za pregled baze podataka korišten je program MySQL Workbench 8.0 CE. Kako bi se moglo pristupiti novom MySQL serveru preko sučelja potrebno je uspostaviti konekciju i predati joj parametre:

- Naziv konekcije (proizvoljno, ali ne smije biti duplicirano)
- Metoda konekcije (automatski zadana standardna (TCP/IP (eng. *Transmission Control Protocol/Internet Protocol*)))
- Naziv domaćina (eng. *host*)
- Port
- Korisničko ime
- Zaporka
- Zadana shema (opcionalno)

Nakon uspostave konekcije pristup serveru i svim njegovim bazama podataka je omogućen. Za izvršavanje naredbi potrebno je postaviti jednu od baza na "*default shema*" kako bi program znao nad kojom bazom se izvršavaju naredbe.

3.2. Java kao jezik za izradu aplikacije

Verzija *runtime* okruženja odabranog unutar Eclipse razvojnog okruženja je Java SE-17. Unutar Java programskog jezika korišteno je više različitih biblioteka i paketa za ostvarivanje željenih funkcionalnosti. Za potrebe izrade sučelja korištena je Swing arhitektura, za komunikaciju s bazom podataka korišten je paket *java.sql* zatim je korištena vanjska biblioteka pod nazivom mysql-connector-java-8.0.28 kako bi se ostvarila veza s bazom podataka.

3.2.1. Swing arhitektura

Set različitih klasa je ponuđen unutar paketa *javax.swing* za dizajniranje korisničkog sučelja. Unutar aplikacije najčešće su korištene klase poput:

- *javax.swing.JFrame* - za stvaranje i konfiguriranje prozora
- *javax.swing.JPan*el - služi kao kontejner unutar kojeg se dodavaju druge komponente
- *javax.swing.JDialog* - za stvaranje prozora koji su slični kao *JFrame* ali nemaju opcije poput minimiziranja i maksimiziranja prozora

3.2.2. Uporaba *java.sql* paketa

Kako bi se izvršile naredbe nad bazom podataka preko aplikacije korištene su klase unutar java.sql paketa. Neke od korištenih klasa uključuju:

- *java.sql.Connection* - za uspostavu veze s bazom podataka prosljeđujući joj određene parametre
- *java.sql.PreparedStatement* - za izvršavanje upita
- *java.sql.ResultSet* - za vraćanje rezultata upita poput SELECT

3.3. Eclipse IDE razvojno okruženje

Ovo razvojno okruženje korišteno je u svrhu pisanja programskog koda za aplikaciju te organiziranja klasa i paketa unutar projekta. Sučelje je jasno i praktično za korištenje što omogućuje lakše snalaženje unutar samog programa. Organiziranje projekta je omogućeno unutar projektnog stabla koje se po zadanome nalazi s lijeve strane, što nudi jednostavan pristup željenoj klasi ili paketu unutar projekta. Omogućeno je odabiranje slova na koja će se aktivirati predlošci za napisani tekst unutar *preferences* što pomaže prilikom pisanja koda. Također je omogućen prikaz mogućih rješenja na upozorenja ili greške unutar projekta.

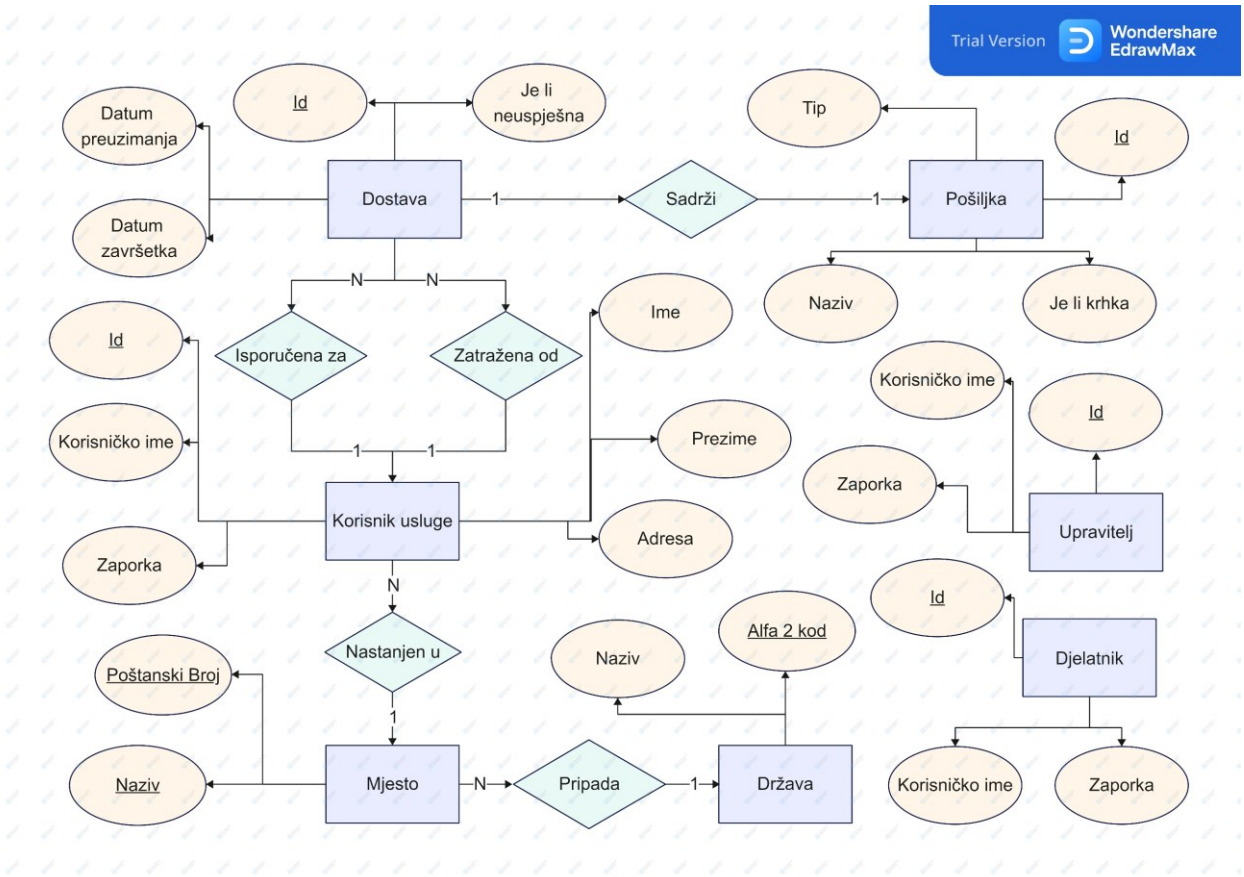
4. IZRADA BAZE PODATAKA

U ovom poglavlju opisana je izrada baze podataka što uključuje modeliranje i opisivanje atributa.

4.1. Definiranje izgleda baze podataka

Baza podataka sadrži podatke vezane za dostavu pošiljke te informacije vezane za prijavu i korištenje aplikacije. Entiteti korišteni u svrhu pohrane podataka su na slici 4.1.

4.1.1. E-R model



Sl. 4.1. E-R model baze podataka dostave pošiljki

Entitet *Dostava* povezan je s entitetima *Pošiljka* i *Korisnik usluge*. Entiteti *Dostava* i *Pošiljka* povezani su vezom 1-1 što znači da samo jedna pošiljka može pripadati jednoj dostavi i jedna dostava može sadržavati samo jednu pošiljku. Entitet *Dostava* povezana je sa 2 tipa entiteta *Korisnik usluge*, onaj koji je zahtjeva i onaj kojemu se isporučuje. Veze između entiteta *Dostava* i *Korisnik usluge* su N-1 i N-1, što znači da jedan pošiljatelj može zahtijevati više dostava, te jednom primatelju može stizati više dostava, no jedna dostava može sadržavati samo jednog pošiljatelja i jednog primatelja. *Korisnik usluge* je povezan s entitetom *Mjesto* što određuje mjesto

u kojemu on živi. Veza između entiteta *Korisnik usluge* i *Mjesto* je N-1 što znači da više korisnika usluge može živjeti u istom mjestu, ali jedan korisnik usluge može živjeti samo u jednom mjestu. *Korisnik usluge* sadržava još i atribute potrebne za autorizirano korištenje aplikacije. Entitet *Mjesto* je povezano s entitetom *Država* koji određuje državu tog grada. Veza između ta dva entiteta *Mjesto* i *Država* je N-1 jer više mjesta može pripadati jednoj državi, ali jedno mjesto pripada samo jednoj državi. Entiteti koji nisu povezani s ostalim entitetima su *Upravitelj* i *Djelatnik*, zbog toga što ne dijele podatke sa dostavom.

4.1.2. Relacijski model

Izgled relacijskog modela baze podataka opisan je ispod:

Dostava (Id, Id pošiljke, Id pošiljatelja, Id primatelja, Datum preuzimanja, Datum završetka, Je li neuspješna)

Pošiljka (Id, Tip, Naziv, Je li krhka)

Korisnik usluge (Id, Korisničko ime, Zaporka, Ime, Prezime, Adresa, Poštanski broj mjesta, Alfa 2 kod države, Naziv mjesta)

Mjesto (Poštanski broj, Alfa 2 kod države, Naziv)

Država (Alfa 2 kod, Naziv)

Djelatnik (Id, Korisničko ime, Zaporka)

Upravitelj (Id, Korisničko ime, Zaporka)

4.2. Atributi unutar tablica

Svi atributi *id* unutar tablica su nizovi od 36 znakova koji predstavljaju univerzalni jedinstveni identifikator. Generirani su na dva načina:

1. korištenjem Java programskog jezika pomoću klase *java.util.UUID***,** korištenjem njene statičke metode *randomUUID()* nakon koje se na objektu *UUID* klase poziva metoda *toString()* kako bi se objekt pretvorio u niz znakova
2. korištenjem MySQL funkcije *UUID()*

Tablica *Dostava* osim svog atributa *id* sadrži atribut *id* tablice *Pošiljka* i dva atributa *id* tablice *Korisnik usluge* kao strane ključeve. Atribut *Datum preuzimanja* predstavlja datum preuzimanja pošiljke od strane osoblja, a *Datum završetka* predstavlja datum kada je dostava stigla na odredište. Atribut *Je li neuspješno dostavljena* predstavlja *boolean* vrijednost koja je *true* ako je dostava

neuspješna. U tom slučaju ne smije postojati konkretna vrijednost atributa *Datum završetka* nego njena vrijednost mora biti *null.* Napravljen je okidač koji se brine za takve i slične nedozvoljene unose prilikom unošenja i ažuriranja podataka unutar te tablice. Datumi i logička varijabla služe za praćenje statusa dostave.

Primjer MySQL koda za kreiranje tablice *Dostava* je na slici 4.2.

Linija ***Kod***

```
1:      CREATE TABLE delivery(
2:          id CHAR(36),
3:          mail_id CHAR(36),
4:          sender_id CHAR(36),
5:          recipient_id CHAR(36),
6:          takeover_date DATE,
7:          completion_date DATE,
8:          is_unsuccessful BOOLEAN DEFAULT FALSE,
9:
10:         CONSTRAINT prkey_delivery_id PRIMARY KEY(id),
11:         CONSTRAINT frkey_delivery_mail_id FOREIGN KEY(mail_id)
12:             REFERENCES mail(id) ON DELETE SET NULL ON UPDATE CASCADE,
13:         CONSTRAINT frkey_delivery_sender_id FOREIGN KEY(sender_id)
14:             REFERENCES service_user(id) ON DELETE SET NULL ON UPDATE
15:             CASCADE,
16:         CONSTRAINT frkey_delivery_recipient_id
17:             FOREIGN KEY(recipient_id) REFERENCES service_user(id)
18:             ON DELETE SET NULL ON UPDATE CASCADE
19:     )
```

Sl. 4.2. Kod za kreiranje tablice *Dostava*

Tablica *Pošiljka* sadrži detalje o samoj pošiljci koja treba biti dostavljena. Sadržava svoj vlastiti atribut *id*. Atribut *Tip* predstavlja tip pošiljke, postoje dva konkretna tipa pošiljke: paket i pismo. *Naziv pošiljke* označava naziv same pošiljke, ako je pošiljka tipa pismo onda nema naziv, a ako je paket onda sadrži ime predmeta koji se šalje unutar paketa. Sljedeći atribut je logička varijabla *Je li krhka* koji označava da li je pošiljka koja se šalje u krhkom stanju, ako je riječ o pismu onda je vrijednost atributa *Je li krhka* uvijek *false*. Provjere korištenja jednog od dva tipa pošiljke te provjere vrijednosti naziv i vrijednost atributa *Je li krhka*, u slučaju da je pošiljka pismo (*Naziv* mora imati vrijednost *null* a atribut *Je li krhka* mora imati vrijednost *false*) ostvareno je pomoću okidača.

Primjer MySQL koda za kreiranje okidača za tablicu *Pošiljka* prilikom unošenja je na slici 4.3.

Linija ***Kod***

```
1:      DELIMITER **
2:      CREATE TRIGGER TR_mail_insert BEFORE INSERT
3:      ON mail
4:      FOR EACH ROW
5:          IF NEW.type != 'package' AND NEW.type != 'letter' THEN
6:              SIGNAL SQLSTATE '03102'
7:              SET MESSAGE_TEXT = 'Type must be strictly letter or
8:                   package.';
9:          ELSE IF NEW.type = 'letter' AND (NEW.name IS NOT NULL OR
10:             NEW.is_fragile = TRUE) THEN
11:             SIGNAL SQLSTATE '03103'
12:             SET MESSAGE_TEXT = 'Letter must not have a name and cannot
13:                 be fragile.';
14:         END IF;
15:     **
```

Sl. 4.3. Kod za kreiranje okidača za tablicu *Pošiljka* prilikom unošenja

Tablica *Korisnik usluge* također sadrži svoj vlastiti atribut *id* koji se dodjeljuje pri registraciji. S atributima *Korisničko ime* i *Zaporka* korisnik se prijavljuje u sustav. Ova 3 atributa se nalaze i kod tablica *Upravitelj* i *Djelatnik*. Prilikom unosa zaporke potrebno je izvršiti enkripciju s nekom od funkcija koje to omogućuju. U ovom slučaju korištena je funkcija *SHA2(str, hash_length).* Duljina kriptirane zaporke je 56 znakova zbog drugog argumenta koji je postavljen na 224, a on predstavlja željenu duljinu rezultata u bitovima prema [6]. Nad kriptiranom zaporkom na taj način, ne može se izvršiti dekripcija. Tablica sadrži atribute *Ime* i *Prezime* koji predstavljaju pravo ime i prezime korisnika, atribut *Adresa* koji predstavlja ulicu ili trg, broj građevine i naselje u slučaju da nije mjesto čiji se poštanski broj koristi ali koristi taj poštanski broj odnosno pripada tom poštanskom uredu, te strane ključeve *Poštanski broj mjesta*, *Naziv mjesta* te *Alfa 2 kod države* iz tablice *Mjesto*.

Tablica *Mjesto* sadrži atribute koji opisuju mjesto. Atribut *Poštanski broj* sastoji se od niza do 10 znakova koji predstavlja poštanski broj mjesta. *Alfa 2 kod države* predstavlja strani ključ iz tablice *Država* te *Naziv* koji predstavlja naziv mjesta. Ova tri atributa zajedno čine primarni ključ tablice jer se može dogoditi da mjesta unutar različitih država imaju isti poštanski broj kao i mjesta unutar iste države.

Tablica *Država* sadrži atribute *Alfa 2 kod* koji predstavlja primarni ključ tablice, a sastoji se od 2 znaka te atributa *Naziv* koji predstavlja naziv države.

5. DIZAJNIRANJE GRAFIČKOG KORISNIČKOG SUČELJA

Kao što je već ranije spomenuto za izradu sučelja koristila se Swing arhitektura. Klase koje su se koristile, a vezane su za sučelje nalaze se unutar sljedećih paketa:

- *javax.swing*
- *javax.swing.table*
- *java.awt*
- *java.awt.event*.

Bitno je napomenuti da sav programski kod vezan za izvođene operacija nad sučeljem se treba izvoditi unutar EDT, odnosno nit za slaganje događaja (eng. *Event Dispaching Thread*) inače može doći do neželjenog ponašanja unutar aplikacije. Postoje dva načina kako osigurati da se programski kod izvodi na EDT: korištenjem metoda *javax.swing.SwingUtilities.invokeLater* i *javax.swing.SwingUtilities.invokeAndWait*. Razlike između ove dvije metode su u tome što prva je neblokirajuća odnosno izvršava zadatak bez da čeka na kraj prijašnjih dok druga je blokirajuća i čeka da se završe prijašnji zadaci da bi se pokrenula prema [7].

Na slici 5.1. kreiran je *java.lang.Runnable* objekt te se metoda *run()* izvodi na EDT prema [7].

Linija ***Kod***

```
1:      javax.swing.SwingUtilities.invokeLater(new Runnable(
2:           public void run(){
3:                //Napisati zadatak
4:           }
5:      });
```

Sl. 5.1. Kod za pokretanje na EDT prema [7]

Runnable sučelje služi za ostvarivanje višenitnosti u Java programima tako što se definira metodu *run()* koju propisuje sučelje te se na temelju tog prilikom pozivanja metode *execute()* na Runnable objektu pokreće njena metoda *run()* na novoj niti.

Na početku programa, izgled i doživljaj se mogu promijeniti korištenjem *javax.swing.UIManager* klase. To se postiže na način da se pozove njena statička metoda *setLookAndFeel(String className)* koja prima jedan *String* parametar koji predstavlja ime klase čiji se izgled i doživljaj želi postaviti. Unutar ove aplikacije korišten je izgled i doživljaj klase *com.sun.java.swing.plaf.windows.WindowsLookAndFeel* koji predstavlja zadani izgled i doživljaj unutar operacijskih sustava Microsoft Windows-a. U slučaju bacanja greške aplikacije prilikom

postavljanja koristi se zadani izgled i doživljaj unutar Java programskog jezika koji pripada klasi *javax.swing.plaf.metal.MetalLookAndFeel*. Izgled i doživljaj mijenjaju izgled korisničkog sučelja i komponenata unutar sučelja.

5.1. Izrada početnog zaslona

Na početku aplikacije nalazi se zaslon koji predstavlja zaslon za prijavu ili registraciju. Kako bi se uspostavio prozor potrebno ga je kreirati pomoću klase *javax.swing.JFrame* prilikom čega se klasa *JFrame* prvo treba instancirati. Korištenjem vlastitih podklasa ove klase prilikom poziva konstruktora pozvana je i kreirana metoda koja služi za konfiguraciju prozora koja je na slici 5.2.

Linija ***Kod***

```
1:      private void configureWindow(int width, int height){
2:           setSize(width, height);
3:           setIconImage(new ImageIcon("resources\\images\\icon.png").
4:                getImage());
5:           setLocationRelativeTo(null);
6:           setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7:           setResizable(false);
8:           setVisible(true);
9:      }
```

Sl. 5.2. Kod za konfiguraciju prozora

Nakon što je prozor postavljen dodan mu je objekt koji nasljeđuje klasu *javax.swing.JPanel* koji služi kao kontejner za komponente (na njemu se komponente nalaze), iako *JFrame* ima isto tu funkcionalnost. Pošto je potrebno dodati pozadinsku sliku unutar te klase korištena je nadjačana (eng. *override*) metoda *paintComponent(Graphics g)*. Pomoću objekta *Graphics* g omogućeno je crtanje po zaslonu, stoga željena slika koja zauzima ekran nacrtana je prema načinu na slici 5.3.

Linija ***Kod***

```
1:      //unutar klase koja nasljeđuje klasu JPanel
2:      private Image image;
3:      //unutar konstruktora…
4:      image = ImageIO.read(new File(filePath)); /* učitavanje slike
5:           predavajući metodi objekt klase File sa konstruktorom koji
6:           sadrži putanju do slike */
7:      @Override
8:      public void paintComponent(Graphics g){
9:           g.drawImage(image, 0, 0, getWidth(), getHeight(), null);
10:     }
```

Sl. 5.3. Kod za crtanje pozadine

Unutar ove klase sadržane su dodatne komponente. Dvije od njih su objekti klase *javax.swing.JLabel*. Klasa *JLabel* predstavlja komponentu koja većinom sadrži neki tekst, iako može sadržavati komponentu poput slike. Jedan objekt predstavlja naslov aplikacije, a drugi predstavlja dodatne informacije o autoru. Kako bi se dodala funkcionalnost poput prelaska u novu liniju potrebno je predani tekst formatirati s HTML-om (eng. *Hypertext Markup Language*):

```
JLabel title = new JLabel("<html>Mail<br>Delivery<br>Manager</html>");
```

Ostali elementi klase su gumbi koji su objekti klase *javax.swing.JButton* koji služe za odabir prijave korisnika te registraciju korisnika usluge. Kako bi se ostvarila funkcionalnost odvijanja događaja prilikom pritiska na gumb dodana je metoda *addActionListener(ActionListener l)*. Tada nakon što se klikne gumb poziva se metoda *actionPerformed(ActionEvent e)* koja je definirana unutar klase koja nasljeđuje sučelje *java.awt.event.ActionListener*. U ovom slučaju pomoću ovih gumbova nakon klika instanciraju se objekti klase *javax.swing.JDialog*, koji služi kao dijaloški okviri prilikom prijave ili registracije. Primjer dodavanja osluškivača za prijavu dan je na slici 5.4.

Linija ***Kod***

```
1:      //unutar metode
2:      loginButton.addActionListener(new ActionListener(){
3:           @Override
4:           public void actionPerformed(ActionEvent e){
5:                new DefaultDialog(SwingUtilities. //DefaultDialog i
6:                getWindowAncestor(HomePanel.this), "Login", //LoginPanel
7:                ModalityType.APPLICATION_MODAL, 400, 300, //su klase
8:                new LoginPanel(UserType.SERVICE_USER)); //unutar projekta
           }                    //UserType je enumeracija unutar projekta
9:      }); //ovo je pokazni primjer, u aplikaciji je napravljeno drukčije
```

Sl. 5.4. Kod za dodavanje "osluškivača radnje" na gumb za prijavu korisnika usluge

5.1.1. Korištenje rasporeda

Nakon što su komponente konfigurirane (font, boja, obrub i slično) slijedi njihovo raspoređivanje. Moguće je postaviti zadani raspored za kontejner na *null* i redati komponente zadajući im fiksnu poziciju. Ovakav pristup nije prikladan jer će komponente sadržati istu poziciju nakon raširivanja ili sužavanja zaslona. Prema tome korišteni su neki od dostupnih upravitelja rasporedom (eng. *layout manager*). Postoje Swing i AWT (eng. *Abstract Window Toolkit*) klase koje predstavljaju upravitelje rasporedom, a to su:

- *BorderLayout*
- *BoxLayout*

- *CardLayout*
- *FlowLayout*
- *GridBagLayout*
- *GridLayout*
- *GroupLayout*
- *SpringLayout*

Za potrebe ove klase koristio se objekt klase *java.awt.GridBagLayout*. Komponente unutar ovog rasporeda su postavljene unutar ćelija prema [8]. Klasa *java.awt.GridBagConstraints* sadrži varijable za definiranje kako će elementi biti posloženi i kako će zauzimati prostor a to su:

- *gridx*, *gridy* - postavljanje koordinata ćelija
- *gridwidth*, *gridheight* - odlučuje koliko će ćelija uzduž ili poprijeko zauzimati određena komponenta
- *fill* - odlučuje kako da se komponenta raširi unutar ćelije
- *ipadx*, *ipady* - horizontalno i vertikalno ispunjenje
- *insets* - vanjsko ispunjenje
- *anchor* - položaj unutar ćelije

Za postavljanje komponenti unutar *GridBagLayout*-a postavljaju se varijable instance *GridBagConstraints*. Potrebno je paziti da su varijable postavljane onako kako je zamišljeno prilikom dodavanja više komponenti koristeći istu instancu *GridBagConstraints* objekta, inače može doći do neželjenog izgleda prema [9]. Primjer korištenja *GridbagLayout*-a je na slici 5.5.

Linija ***Kod***

```
1:      //unutar klase koja nasljeđuje JPanel
2:      private GridBagConstraints gbc = new GridBagConstraints();
3:      //unutar konstruktora
4:      setLayout(new GridBagLayout());
5:      //unutar metode
6:      gbc.gridx = 0; //postavljanje na ćeliju sa x-osi na poziciji 0
7:      gbc.gridy = 0; //postavljanje na ćeliju sa y-osi na poziciji 0
8:      gbc.weightx = 0.08; //omjer širine stupca u odnosu na ostale ćelije
9:      gbc.weighty = 0.5; //omjer širine retka u odnosu na ostale ćelije
10:     gbc.insets = new Insets(14, 0, 0, 0); //vanjsko ispunjenje
11:     gbc.anchor = GridBagConstraints.NORTH; //položaj
12:     add(managerLoginButton, gbc); //dodavanje objekta u kontejner
13;     //dodavanje ostalih komponenti...
```

Sl. 5.5. Kod za dodavanje komponente u kontejner pomoću *GridBagLayout*-a

5.1.2. Dijaloški okvir prijave/registracije

Nakon klika na gumb za prijavu ili registraciju otvara se novi dijaloški okvir. Dijalog služi za upisivanje podataka o korisniku koji planira koristiti aplikaciju za prijavu ili registraciju. Dijaloški okvir predstavljen je klasom *javax.swing.JDialog*. Komponente koje služe za upis podataka su *javax.swing.JTextField* čija se instanca koristi za unos podataka o korisničkom imenu i *javax.swing.JPasswordField* čija se instanca koristi za unos zaporke. Komponente su smještene unutar klase koja nasljeđuje klasu *JPanel* kao i u početnom zaslonu te se njena instanca nalazi unutar dijaloškog okvira. Prilikom unosa zaporke znakovi su predstavljeni točkama te se ne mogu kopirati. Tekst unutar objekta *JTextField* klase se dohvaća pomoću metode *getText()*, a unutar objekta *JPasswordField* klasa pomoću metode *getPassword()*. Izgled dijaloškog okvira prikazan je na slici ispod. Prilikom neispravnog unosa poput (prazan tekst unutar jedno od ova dva polja, korisničko ime ili zaporka nisu pronađeni kod prijave, zaporka ne ispunjava kriterij kod registracije) boja njihovih okvira mijenja boju u crvenu pomoću metode *setBorder(Border border)* kod onog polja kod kojeg nešto nije u redu. Također, izbacuje se novi dijaloški okvir greške pomoću metode unutar klase *javax.swing.JOptionPane*. Neke komponente su onemogućene tijekom interakcije sa bazom podataka kako ne bi došlo do neželjenog ponašanja.

Dodano je mijenjanje pozadine kod komponenti unosa ovisno o onoj na kojoj je trenutni fokus pomoću metode *addFocusListener(FocusListener l)* kao i pomicanje po komponentama pomoću tipki: *UP*, *DOWN*, *LEFT* i *RIGHT* korištenjem metode *addKeyListener(KeyListener l)*.

5.1.3. Dijaloški okvir popunjavanja podataka korisnika usluge

Ako je korisnik usluge odabrao gumb za registriranje nakon unošenja podataka unutar okvira registracije otvara se novi dijaloški okvir unutar kojeg se unose njegovi podaci, a to su: ime, prezime i adresa. Unutar ovog okvira pojavljuje se objekt klase *javax.swing.JComboBox* koja služi kao padajući izbornik unutar kojega se odabire jedan izbor. Prilikom instanciranja ove klase potrebno je odrediti koji tip vrijednosti će se nalaziti unutar padajućeg izbornika. U ovome slučaju služi za odabir države i mjesta iz te države pa se tako koristi tip vrijednosti *String* za oba objekta. Države se učitavaju iz baze podataka. Mjesta se sva učitavaju iz baze podataka i zatim se filtriraju u novoj listi ovisno o državi koju je korisnik odabrao te se filtrirana lista prikazuju u padajućem izborniku. Omogućene su provjere praznog teksta ili prekoračenja dozvoljene dužine teksta za pojedino polje. Nakon unošenja podataka i klika na gumb za potvrdu korisnik usluge je registriran te se dijaloški okvir zatvara i otvara se novi prozor.

5.2. Izrada zaslona korisnika usluge

Ovaj zaslon namijenjen je korisniku usluge unutar kojega je omogućeno slanje zahtjeva za dostavu drugim korisnicima usluge te pregled narudžbi koje je korisnik usluge poslao ili koje su njemu poslane. Unutar zaslona postavljena je traka izbornika u kojoj se nalazi izbornik koji sadrži opcije vezane za račun, a to su: pregled profila i mijenjanje podataka, mijenjanje zaporke, brisanje računa i odjavljivanje. Kod svih opcija osim odjave otvara se novi dijaloški okvir unutar kojeg se izvršavaju navedene radnje, a kod odjave se samo gasi trenutni zaslon i otvara početni. Kod mijenjanja zaporke unosi se stara zaporka te se dva puta unosi nova, omogućene su provjere ispravnosti. Kod brisanja računa potrebno je unijeti zaporku kako bi se račun izbrisao.

5.2.1. Postupak slanja zahtjeva za dostavu

Nakon što se klikne gumb "*Send new delivery request*" otvara se novi dijaloški okvir. Unutar tog okvira ispunjavaju se detalji o dostavi. Kada je odabrani tip pošiljke postavljen na *Letter* što i jest zadani odabir prilikom otvaranja dijaloškog okvira, tada su komponente za upisivanje imena pošiljke i odabira vrijednosti za krhkost pošiljke onemogućene za korištenje. Kada se promjeni tip na *Package* tada su omogućene. U slučaju da se opet vrati na zadanu vrijednost tada se i polja vezana za ime i krhkost pošiljke postavljaju na zadane bez obzira što je prije toga pisalo ili bilo odabrano unutar njih. Za raspored korišten je *java.awt.GridLayout* koji elemente prikazuje unutar rešetke, gdje se unutar konstruktora unosi broj stupaca i redaka te je moguće definirati horizontalni i vertikalni prostor između njih.

5.2.2. Prikaz dostava

Klasa koja omogućuje prikaz dostava sadrži unutar sebe instancu klase *javax.swing.JScrollPane* unutar koje se dodavaju komponente te je moguće vertikalno i horizontalno pomicanje, prilikom čega ako ova komponenta zauzima više prostora nego što je dostupno moguće je opet pregledati sve što se u njoj nalazi. Unutar nje se nalaze instance klase koja nasljeđuje klasu *JPanel*, a predstavljaju prikaz podataka. *JPanel* unutar kojega je smješten ovaj niz *JPanel*-a koristi *BoxLayout* za raspoređivanje elemenata koji elemente raspoređuje jedan iza drugog vertikalno ili horizontalno. Svaki puta kada se osvježava niz *JPanel*-a za prikaz podataka, niz se uklanja pomoću metode *removeAll()* te se dodaje novi niz. Niz se osvježava nakon dodavanja novih podataka, zatim na način da se klikne desni klik miša na instanci *JScrollPane*-a koji prikazuje te podatke te se odabere opcija *Refresh*. Instanca klase *javax.swing.Timer* služi kako bi se niz osvježio svakih pet minuta. Mjerač vremena se zaustavlja kada se izađe iz prozora korisnika usluge. Ova klasa je apstraktna jer se mora definirati u podklasama iz kojeg zaslona se što osvježava te koji podaci se

prikazuju unutar niza *JPanel*-a. Stoga se koristi objekt podklase ove klase te se na zaslonu korisnika usluge prikazuju podaci o dostavama u koju je uključen prijavljeni korisnik usluge. Status dostave može imati četiri stanja: u tijeku odobravanja, u toku, završena i neuspjela.

5.3. Izrada zaslona djelatnika

Nakon što se korisnik uspješno prijavio pomoću gumba za prijavu djelatnika zatvaraju se prijašnji prozor i dijaloški okvir i otvara se novi prozor. Unutar njega nalazi se instanca klase *JPanel* koja u sebi ima instancu klase *javax.swing.JTextArea* smještenu unutar *JScrollPane*-a koja služi kao tekstualno polje unutar kojega se prikazuju lokalne aktivnosti djelatnika i njihovo vrijeme događanja. Pored toga unutar zaslona nalazi se gumb za pregled završenih dostava te instancu klase *javax.swing.JTabbedPane* unutar koje se može nalaziti više objekta klase *JPanel* te se njihov prikaz mijenja ovisno o kliku gumbova koji se nalaze na vrhu *JTabbedPane*-a. *JPanel*-i smješteni unutar te instance su instance klasa koje nasljeđuju klasu koja omogućuje pregled podataka spomenutu u poglavlju 5.2.2. Prikazani podaci unutar prvog *JPanel*-a predstavljaju dostave koje su u tijeku odnosno aktivne dostave, a unutar drugog su prikazani podaci zahtijevanih dostava koje se još nisu odobrile. Zahtijevane dostave se mogu potvrditi te se odabire njihov datum preuzimanja i odbiti pri čemu su izbrisane iz baze podataka. Aktivne dostave se mogu završiti te se odabiru opcije s obzirom da li je dostava uspješna ili neuspješna, ako je uspješna također se odabire i datum završetka dostave. U traci izbornika, izbornik sadrži opcije za promjenu zaporke i odjavu.

5.4. Izrada zaslona završenih dostava

Nakon klika na gumb za pregled završenih dostava unutar zaslona djelatnika otvara se novi prozor unutar kojega se nalazi tablica sa podacima o završenim dostavama. Retke unutar tablice je moguće brisati označujući ih i zatim pritiskanjem gumba za brisanje prilikom čega se briše dostava iz baze podataka. Ostvarivanje tablice omogućeno je pomoću objekta klase *javax.swing.JTable*.

5.4.1. Korištenje *JTable* klase

U ovom slučaju korišten je *JScrollPane* objekt kao kontejner za *JTable* objekt. Za postavljanje širina stupca koristi se metoda *setPreferredWidth(int preferredWidth)*. Svaki objekt tablice koristi objekt modela tablice za upravljanje podacima unutar nje. Objekt modela tablice mora implementirati sučelje *javax.swing.table.TableModel*. Korištena instanca modela pripada podklasi klase *javax.swing.table.DefaultTableModel* koja nasljeđuje navedeno sučelje. Za potrebe aplikacije to svojstvo je onemogućeno na način prikazan na slici 5.6.

Linija ***Kod***

```
1:      //unutar klase koja nasljeđuje JFrame
2:      private DefaultTableModel tableModel;
3:      //unutar konstruktora
4:      tableModel = new DefaultTableModel(){
5:           @Override
6:           public boolean isCellEditable(int row, int column){
7:                return false;
8:           }
9:      };
```

Sl. 5.6. Kod za inicijalizaciju modela tablice prema [10]

Obrađivači se koriste sa crtanje ćelija. Moguće je koristiti ćelijski-specifične obrađivače, tipom-specifične obrađivače te je moguće zadati da ćelije u pojedinom stupcu koriste obrađivač prema [10].

Za potrebe aplikacije korišten je ćelijski-specifičan obrađivač. Primjer implementacije vlastitog obrađivača koji mijenja prednju boju i rub ćelija ovisno o tome da li je redak u kojem se nalaze označen ili ne, je na slici 5.7.

Linija ***Kod***

```
1:      DefaultTableCellRenderer renderer = new DefaultTableCellRenderer(){
2            @Override
3:           public Component getTableCellRendererComponent(JTable table,
4:           Object value, boolean isSelected, boolean hasFocus, int row,
5:           int column){
6:                Component component = super.getTableCellRendererComponent(
7:                table, value, isSelected, hasFocus, row, column);
8:                if(isSelected){
9:                     setForeground(new Color(0, 73, 181));
10:                    setBorder(BorderFactory.createLineBorder(Color.
11:                    BLACK));
12:               }
13:               else{
14:                    setForeground(Color.BLACK);
15:                    setBorder(null);
16:               }
17:               return component;
18:          }
19:     };
```

Sl. 5.7. Kod za inicijalizaciju obrađivača tablice prema [10]

Kako bi se postavio ćelijski-specifičan obrađivač potrebno je kreirati podklasu klase *JTable* koja će nadjačati metodu *getCellRenderer(int row, int column)* tako što će vratiti specifičan objekt klase koja nasljeđuje *TableCellRenderer* sučelje. Unutar podklasa klase *JTable* sadrži nadjačana je metoda *prepareRenderer(TableCellRenderer renderer, int row, int col)* pomoću koje se mijenja boja retka tablice u crvenu ako je dostava neuspješna i zelenu ako je dostava uspješna. Pošto tekst unutar tablica može zauzimati više mjesta nego što je dužina ćelije u kojoj se nalazi potrebno je također nadjačati metodu *getToolTipText(MouseEvent event)* prema [10].

Prilikom klika na stupac u zaglavlju omogućeno je sortiranje ovisno o kliknutom stupcu. Sortiranje se temelji na usporedbi pomoću objekta klase *java.util.Comparator*. Primjer uspoređivanja podataka jednog od stupaca dan je prema slici 5.8.

Linija ***Kod***

```
1:      Comparator<Delivery> comparator = new Comparator<Delivery>(){
2:           @Override
3:           public int compare(Delivery o1, Delivery o2){
4:                return o1.takeoverDate().compareTo(o2.
5:                takeoverDate());
6:           }
7:      };
```

Sl. 5.8. Kod za sortiranje prema datum preuzimanja

Model tablice sadrži jedan stupac koji nije dostupan unutar tablice a to je id dostave, a služi kako bi se znalo nad kojom dostavom se vrše naredbe u bazi podataka. Prilikom osvježavanja podataka unutar tablice brišu se svi redci unutar modela tablice te se učitavaju ponovo iz baze podataka.

5.5. Izrada zaslona upravitelja

Nakon prijave pomoću gumba za prijavu upravitelja unutar početnog zaslona zatvara se postojeći prozor te se otvara novi prozor koji služi za upravljanje djelatnicima. Jedini način za stvaranje upraviteljevog računa je od strane administratora baze podataka. Za prikaz popisa djelatnika korištena je podklasa apstraktne klase spomenute u poglavlju 5.2.2. Moguće je dodavati nove djelatnike klikom na gumb prilikom čega se otvara dijaloški okvir unutar kojega se unosi korisničko ime i zaporka djelatnika. Djelatnike je moguće brisati iz baze podataka, te uređivati im imena. Upravitelj može pomoću izbornika unutra trake izbornika birati opcije za promjenu svoje zaporke i odjavu.

6. INTERAKCIJA BAZE PODATAKA S APLIKACIJOM

Kako bi se komuniciralo s bazom podataka unutar aplikacije potrebno je koristiti metode i klase unutar *java.sql* paketa. Nakon pokretanja programa postavljaju se parametri koji će se koristiti za uspostavu konekcije sa bazom podataka. Također je potrebno potražiti klasu koja predstavlja JDBC pogonski program pomoću statičke metode *forName(String className)* unutar *java.lang.Class* klase. JDBC pogonski program omogućuje povezivanje s bazom podataka. Prva interakcija s bazom podataka odvija se prilikom prijave korisnika ili registracije korisnika usluge. Prilikom pozivanja metoda koje vrše interakciju s bazom podataka potrebno ih je pozivati na drugoj niti koja nije EDT, kako se ne bi blokirao UI (eng. *User Interface*). To se postiže korištenjem instance podklase klase *javax.swing.SwingWorker<T,V>*. Potrebno je implementirati njenu apstraktnu metodu *doInBackground()* koja ono što se odvija unutar nje stavlja na sporednu nit. Moguće je nadjačati metodu *done()* koja nakon što prijašnja metoda odradi posao, odrađuje posao na EDT. Nakon instanciranja podklase potrebno ju je pokrenuti metodom *execute()*. Primjer definicije metode za verifikaciju korisnika u bazi podataka je na slici 6.1.

Linija Kod

```
1:      //unutar klase DatabaseOperations
2:      public static String verifyUser(String verificationAttributeName,
3:      String verificationAttribute, String password, String table) throws
4:      SQLException{
5:           String id = null;
6:           Connection connection = DatabaseConnectionSetup.getConnection();
7:           String sql = "SELECT id FROM " + table + "WHERE " +
8:           verificationAttributeName + " = ? AND password = SHA2(?, 224)";
9:           PreparedStatament statement = connection.prepareStatement(sql);
10:          statement.setString(1, verificationAttribute);
11:          statement.setString(2, password);
12:          ResultSet resultSet = statement.executeQuery();
14:          if(resultSet.next()){
15:               id = resultSet.getString(1);
16:          }
17:          connection.close();
18:          return id;
19:     }
```

Sl. 6.1. Kod za autorizaciju korisnika

Parametri za konekciju predaju se na početku programa klasi unutar projekta pod nazivom *DatabaseConnectionSetup* te se pomoću nje dohvaća konekcija. Objekt konekcije kreiran je pomoću klase *java.sql.DriverManager*. Primjer kreiranja konekcije i vraćanja konekcije sa zadanima parametrima je na slici 6.2.

Linija ***Kod***

```
1:      public static Connection getConnection() throws SQLException{
2:           return DriverManager.getConnection("jdbc:mysql://" + serverName
3:                + ":" + port + "/" + databaseName, username, password);
4:      }
```

Sl. 6.2. Kod za kreiranje metode koja vraća konekciju sa zadanim parametrima prema [11]

Na svakom prozoru osim početnog dohvaća se lista podataka iz baze podataka. Ti se podaci prikazuju korisniku. Primjer dohvaćanja liste djelatnika prikazan je na slici 6.3.

Linija ***Kod***

```
1:     public static List<Employee> getAllEmployees() throws SQLException{
2:          Connection connection = DatabaseConnectionSetup.getConnection();
3:          String sql = "SELECT * FROM employee";
4:          PreparedStatement statement = connection.prepareStatement(sql);
5:          ResultSet resultSet = statement.executeQuery();
6:          List<Employee> employees = new LinkedList<>();
7:          while(resultSet.next()){
8:               employees.add(new Employee(resultSet.getString(1),
9:                    resultSet.getString(2)));
10:         }
11:         connection.close();
12:         return employees;
13:    }
```

Sl. 6.3. Kod za dohvaćanje svih djelatnika

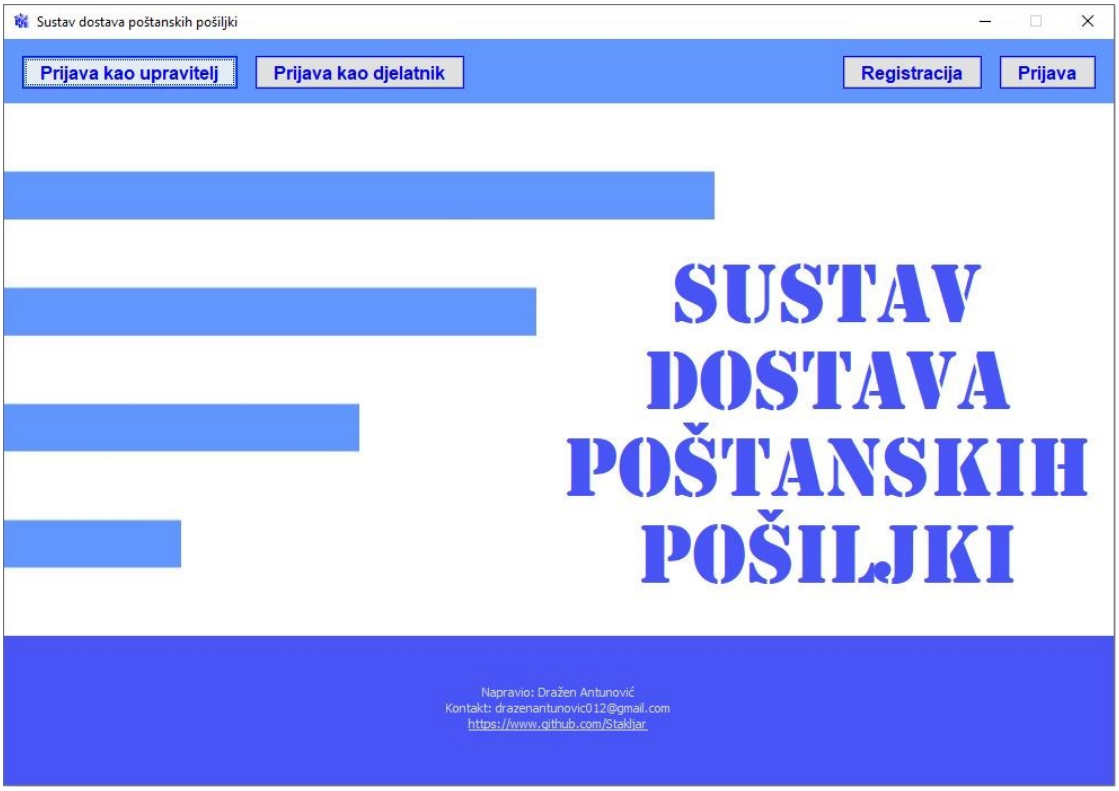
Prilikom odrađivanja operacija koje modificiraju podatke unutar baze podataka provjerava se je li korisnikov račun izbrisan, u slučaju da jest operacija se ne izvršava te se korisnika odjavljuje i vraća početni zaslon.

7. PRIKAZ RADA APLIKACIJE

U ovom poglavlju prikazan je način rada i izgled grafičkog sučelja aplikacije.

7.1. Početni zaslon

Nakon ulaska u aplikaciju izgled početnog zaslona prikazan je na slici 7.1.

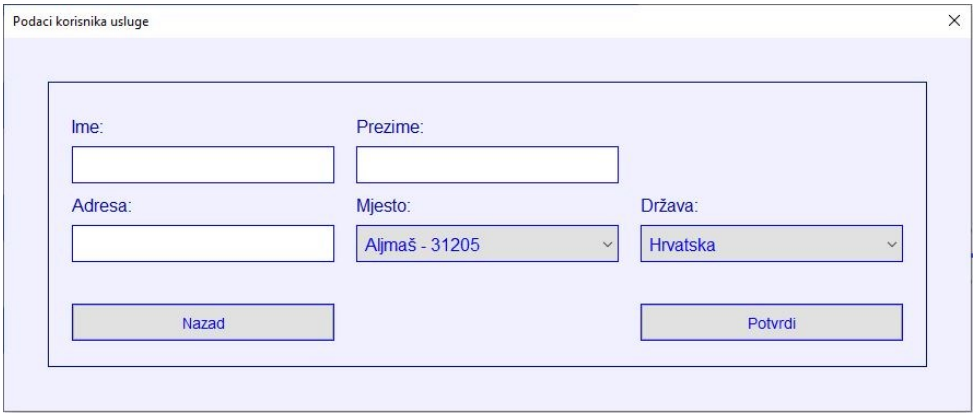


Sl. 7.1. Početni zaslon aplikacije

Nakon pritiska na gumb za prijavu što predstavlja prijavu korisnika usluge pojavljuje se dijaloški okvir gdje se popunjavaju podaci potrebni za prijavu čiji je izgled prikazan na slici 7.2.

Sl. 7.2. Izgled dijaloškog okvira prijave korisnika usluge

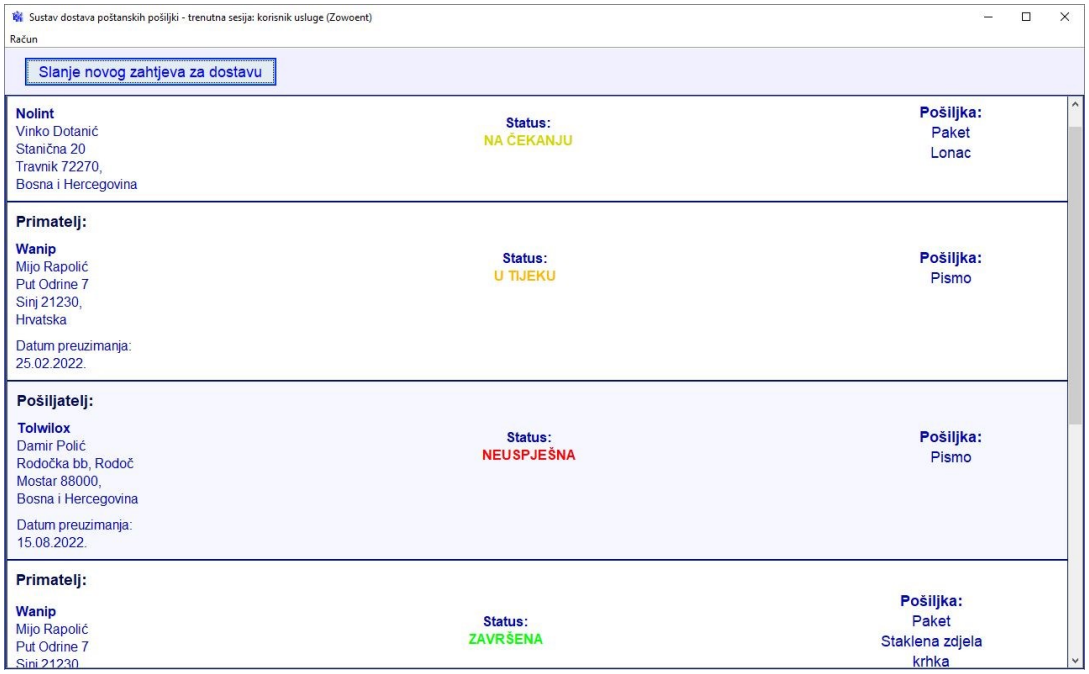
Dijaloški okviri za prijavu ostalih korisnika se razliku po naslovu, a okvir registracije je po izgledu isti kao za prijavu osim što se isto razlikuje po naslovu. Nakon što korisnik usluge unese korisničko ime i zaporku kod registracije pojavljuje se dijaloški okvir za unos ostalih podataka čiji je izgled prikazan na slici 7.3.



Sl. 7.3. Izgled dijaloškog okvira popunjavanja podataka korisnika usluge

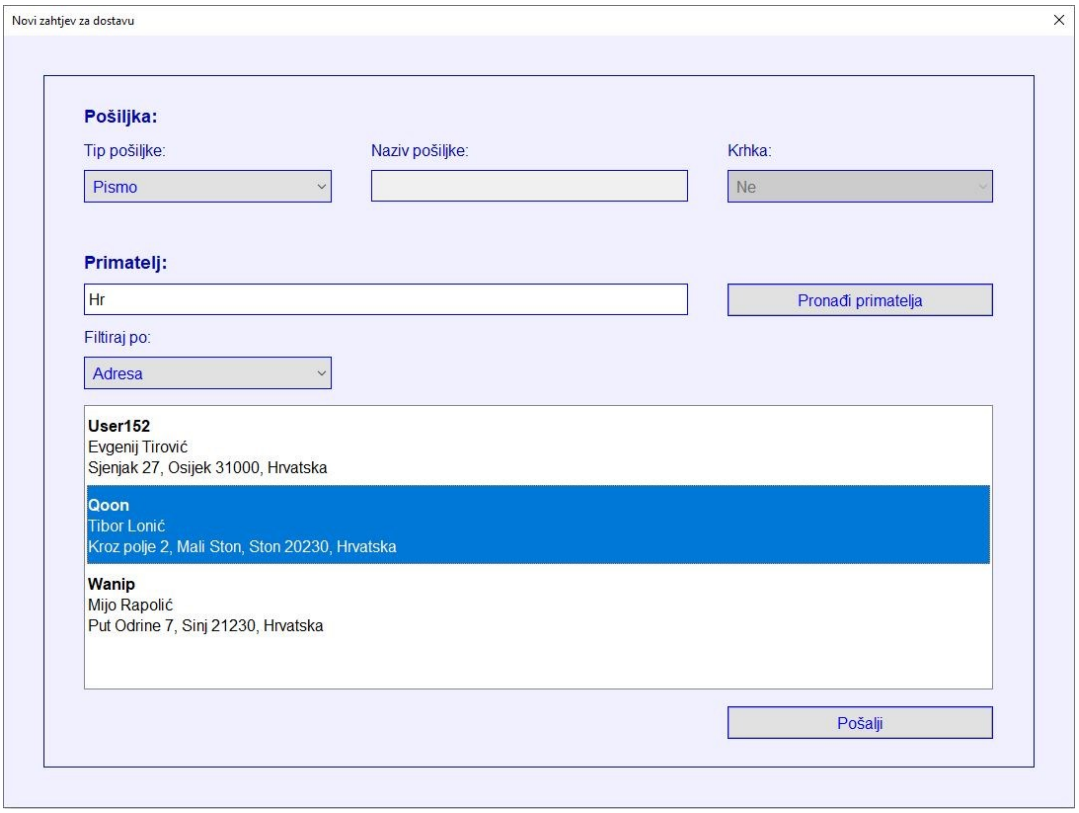
7.2. Zaslon djelatnika

Nakon unošenja podataka i klika na gumb za potvrdu, ako nema nekih grešaka u unosu, otvara se prozor gdje korisnik usluge može zahtijevati nove dostave poštanskih pošiljki te pratiti postojeće u kojima je uključen. Podaci o dostavama koji su prikazani na slikama su pokazni primjeri te su korisnici nasumični i nasumično su povezani s adresama. Ovisno je li korisnik primatelj ili pošiljatelj tako i panel te pošiljke mijenja boju. Prikaz tog prozora prikazan je na slici 7.4.



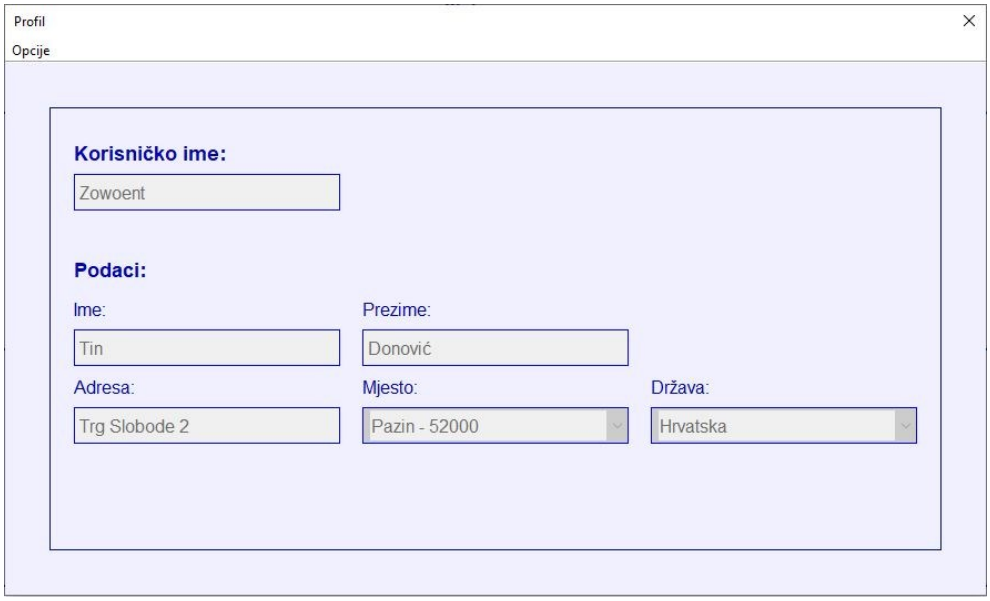
Sl. 7.4. Izgled prozora korisnika usluge

Primjer dijaloga za slanje novog zahtjeva za dostavu prikazan je na slici 7.5.



Sl. 7.5. Izgled dijaloškog okvira slanja zahtjeva za dostavu

Korisnik može pregledati svoj profil te izmjenjivati svoje pomoću podatke pomoću jedne opcije izbornika unutar trake izbornika. Primjer izgleda dijaloškog okvira profila prikazan je na slici 7.6.



Sl. 7.6. Izgled dijaloškog okvira profila korisnika usluge

7.3. Promjena i potvrda zaporke

Jedna od ostalih opcija izbornika na prozoru korisnika usluge je mijenjanje zaporke koja je omogućena i ostalim korisničkim profilima. Prikaz dijaloškog okvira promjene zaporke prikazan je na slici 7.7.

Sl. 7.7. Izgled dijaloškog okvira promjene zaporke

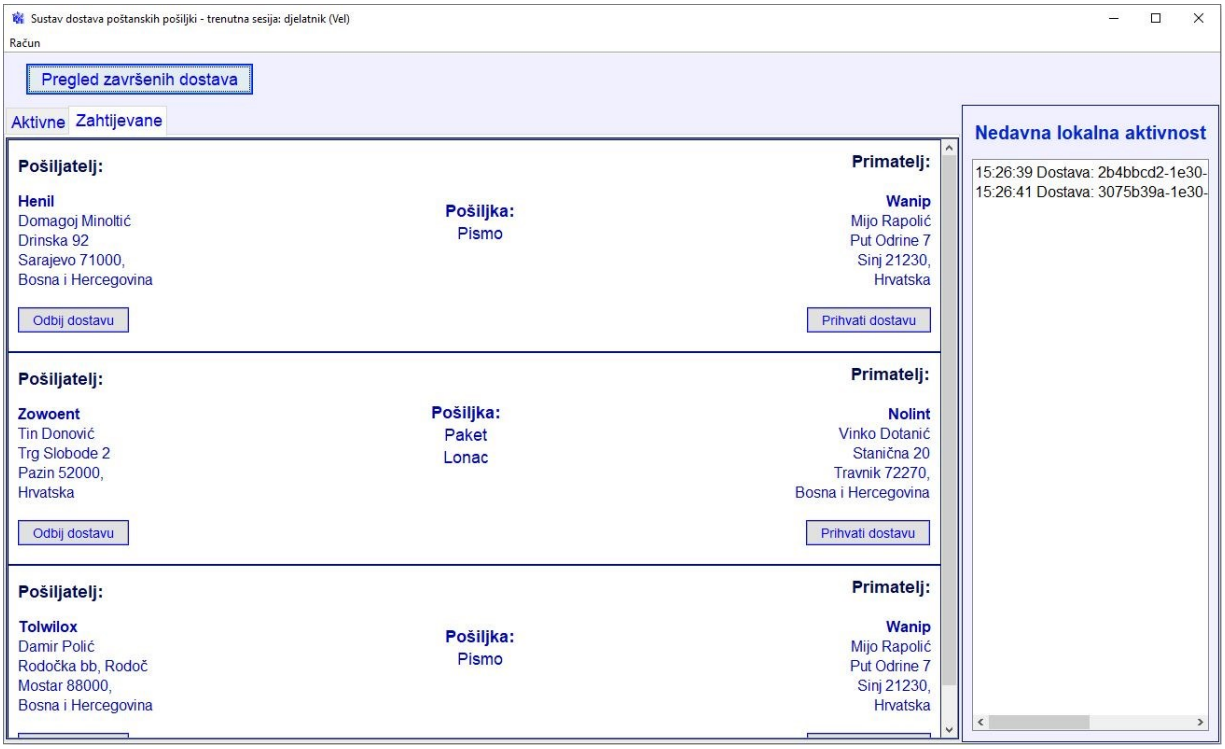
Sljedeća opcija je brisanje računa prilikom koje, ako se potvrdi unutar dijaloškog okvira da se želi obrisati račun otvara se dijaloški okvir unutar kojega je potrebno potvrditi zaporku. Prikaz dijaloškog okvira potvrde zaporke prikazan je na slici 7.8.

Sl. 7.8. Izgled dijaloškog okvira potvrde zaporke

Zadnja opcija je odjava nakon koje se otvara početni zaslon.

7.4. Zaslon djelatnika

Nakon prijave djelatnika otvara se prozor u kojemu se upravlja dostavama (prihvaćanje ili odbijanje novih zahtjeva za dostavu te završetak dostave). Prikaz zaslona djelatnika prikazan je na slici 7.9.

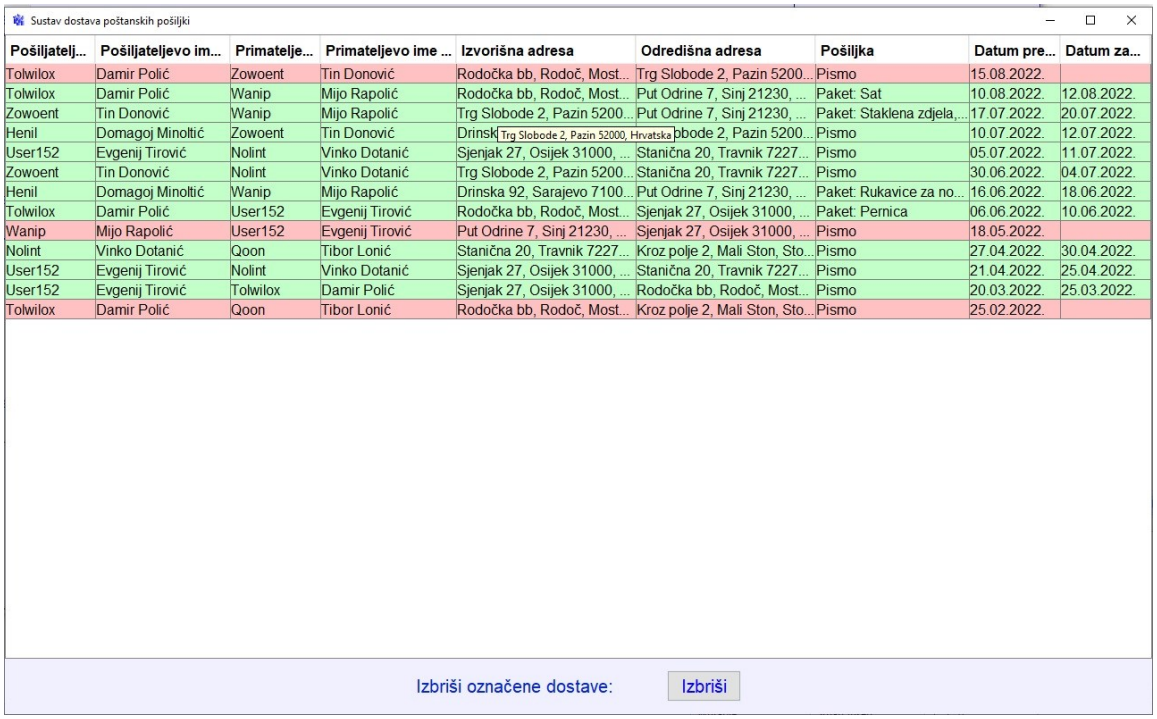


Sl. 7.9. Izgled prozora djelatnika

Primjer dijaloškog okvira prilikom prihvaćanja dostave gdje se unosi datum preuzimanja prikazan je na slici 7.10.

Sl. 7.10. Izgled dijaloškog prihvaćanja dostave

Prilikom pregleda završenih dostava otvara se novi prozor sa tablicom unutar kojega je moguće pregledati i brisati završene dostave. Prikaz tog prozora prikazan je na slici 7.11.



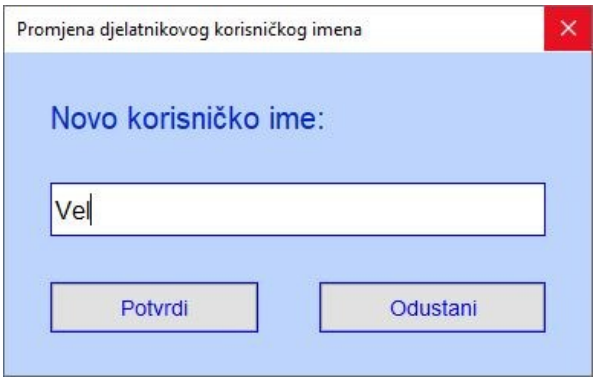
Sl. 7.11. Izgled prozora završenih dostava

7.5. Zaslon upravitelja

Prilikom prijave kao upravitelj otvara se prozor gdje se prikazuju djelatnici te je moguće dodavati, mijenjati korisničko ime i brisati djelatnikov račun. Prikaz tog prozora prikazan je na slici 7.12.

Sl. 7.12. Izgled prozora upravitelja

Izgled dijaloškog okvira prilikom dodavanja novog djelatnika isti je kao i kod prijave/registracije samo što je naslov drugačiji. Primjer dijaloškog okvira prilikom uređivanja korisničkog imena djelatnika prikazan je na slici 7.13.



Sl. 7.13. Izgled dijaloškog okvira mijenjanja djelatnikovog korisničkog imena

8. ZAKLJUČAK

Prilikom stvaranja baze podataka cilj je bio da se omogući stvaranje ključnih podataka vezanih za dostavu pošiljke i za korisnike. Korišteno je sedam tablica kako bi se podaci posložili u pripadajuće dijelove. Za pošiljku mogle su se koristiti dodatne informacije poput dimenzija pošiljke u slučaju da je paket te prioritet, za korisnike usluge moguće je bilo dodati informaciju o broju telefona, ali ti podaci su izostavljeni. Prilikom unošenja zaporke za korisnike u bazu podataka problem *hashiranja* zaporke je riješen korištenjem ugrađene metode unutar MySQL-a. Java programski jezik korišten je za dizajniranje sučelja i interakciju s bazom podataka. Zbog programskog dizajniranja sučelja programski kod može zauzimati nešto više linija što smanjuje njegovu čistinu. Prilikom dizajnirana sučelja potrebno je bilo onemogućiti korisniku pristup podacima sve dok on ne potvrdi da je ovlašten unosom valjanih podataka. To je ostvareno sučeljem za prijavu prilikom kojeg korisnik tek nakon unosa ostvaruje pravo korištenja određenog dijela aplikacije. Korištenjem više različitih prozora i dijaloških okvira omogućeno je prikladno izvršavanje naredbi nad bazom podataka unutar aplikacije te pridonosi višeznačnosti same aplikacije. Prikaz podataka korisnicima je omogućen kao vertikalni niz kojega je moguće listati. Interakcija s bazom podataka se vrši unutar kreiranih klasa koja sadrže sve statičke metode i atribute jer se sve naredbe vrše nad jednom bazom podataka pa je potrebno korištenje istih parametara postavljenih prilikom pokretanja programa.

9. LITERATURA

[1] »GetApp | Business Software, Reviews & Comparisons« [Mrežno]
Dostupno na: https://www.getapp.com/transportation-logistics-software/a/desktopshipper/
[Pokušaj pristupa: 10.09.2022.]

[2] »GetApp | Business Software, Reviews & Comparisons« [Mrežno]
Dostupno na: https://www.getapp.com/transportation-logistics-software/a/courier-management-software/
[Pokušaj pristupa: 10.09.2022.]

[3] »GetApp | Business Software, Reviews & Comparisons« [Mrežno]
Dostupno na: https://www.getapp.com/operations-management-software/a/shipstation/
[Pokušaj pristupa: 10.09.2022.]

[4] »GetApp | Business Software, Reviews & Comparisons« [Mrežno]
Dostupno na: https://www.getapp.com/operations-management-software/a/shippingeasy/
[Pokušaj pristupa: 10.09.2022.]

[5] »GetApp | Business Software, Reviews & Comparisons« [Mrežno]
Dostupno na: https://www.getapp.com/operations-management-software/a/orderhive/
[Pokušaj pristupa: 10.09.2022.]

[6] »MySQL :: Developer Zone« [Mrežno]
Dostupno na: https://dev.mysql.com/doc/refman/8.0/en/encryption-functions.html
[Pokušaj pristupa: 10.09.2022.]

[7] »Oracle Help Center« [Mrežno]
Dostupno na: https://docs.oracle.com/javase/tutorial/uiswing/concurrency/initial.html
[Pokušaj pristupa: 10.09.2022.]

[8] »Oracle Help Center« [Mrežno]
Dostupno na: https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
[Pokušaj pristupa: 10.09.2022.]

[9] »Oracle Help Center« [Mrežno]
Dostupno na: https://docs.oracle.com/javase/tutorial/uiswing/layout/gridbag.html
[Pokušaj pristupa: 10.09.2022.]

[10] »Oracle Help Center« [Mrežno]
Dostupno na: https://docs.oracle.com/javase/tutorial/uiswing/components/table.html
[Pokušaj pristupa: 10.09.2022.]

[11] »Oracle Help Center« [Mrežno]
Dostupno na: https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html
[Pokušaj pristupa: 12.09.2022.]

SAŽETAK

Izrađena je Java desktop aplikacija koja različitim korisničkim profilima pruža ulogu u postupku dostave poštanskih pošiljki. Uspostavljena je interakcija s MySQL bazom podataka te izvođenje naredbi nad njom. Unutar aplikacije korištene su CRUD (eng. *Create, Read, Update and Delete*) naredbe. Unutar aplikacije koriste se tri različita korisnička profila: korisnik usluge, djelatnik i upravitelj. Rad aplikacije se temelji na tome da korisnik usluge zahtjeva dostavu pošiljke drugom korisniku usluge prilikom čega djelatnik mora odobriti dostavu kako bi se isporučila ili je može odbiti. Djelatnik na kraju dostave označava da je dostava završena unoseći njezin datum završetka. Korisnici usluge mogu pratiti status dostave u kojima su uključeni. Dostava obuhvaća: podatke pošiljatelja i primatelja koji su ispunjeni tijekom registracije tih korisnika, podatke o pošiljci te atribute pomoću kojih se prati status dostave (datumi i logička varijabla o neuspješnosti). Završene dostave se nalaze u tablici gdje ih je moguće brisati iz baze podataka. Upraviteljeva uloga je: dodavanje, uklanjanje ili mijenjanje korisničkih imena računima djelatnika. Tijekom operacija s bazom podataka omogućeno je prikazivanje obavještenja prilikom pojave grešaka.

Ključne riječi: Java, MySQL, Swing

ABSTRACT

Java application for postal mail delivery tracking

Java desktop application which provides role for different user profiles in postal mail delivery procedure is made. Interaction with MySQL database is established as well as executing commands over it. Inside application CRUD (Create, Read, Update and Delete) operations are utilized. Inside application three different user profiles are used: service user, employee and manager. The work of application is based on service user requesting mail delivery to another service user during which an employee must accept delivery for it to be delivered or it can be denied. An employee marks delivery finished inputing it's completion date. Service users can track delivery status in which they are involved. A delivery includes: sender and recipient's data which is filled during registration of those users, mail data and atrributes which help track delivery status (dates and boolean value of delivery unsuccess). Finished deliveries can be found in the table from where they can be deleted from database. Manager's roles are to: add, delete or change username of employees' accounts. During database operations error report is established.

Key words: Java, MySQL, Swing