Blagajna za maloprodajnu trgovinu

Šušnjara, Antonio

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:200:929218

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Stručni studij

BLAGAJNA ZA MALOPRODAJNU TRGOVINU

Završni rad

Antonio Šušnjara

Osijek, 2022.

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

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

Osijek, 02.12.2022.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

Ime i prezime Pristupnika:	Antonio Šušnjara
Studij, smjer:	Preddiplomski stručni studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AI 4639, 27.07.2017.
OIB Pristupnika:	18346305487
Mentor:	Robert Šojo, mag. ing. comp.
Sumentor:	Doc. dr. sc. Tomislav Galba
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Marina Peko, dipl. ing.
Član Povjerenstva 1:	Robert Šojo, mag. ing. comp.
Član Povjerenstva 2:	mr.sc. Željko Štanfel
Naslov završnog rada:	Blagajna za maloprodajnu trgovinu
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada	Kreirati aplikaciju blagajne za maloprodajnu trgovinu. Aplikacija ima mogućnost korištenja od strane administratora i regularnog blagajnika. Administrator može odobravati vraćanje proizvoda, unos novih proizvoda, korisnika (blagajnika), izmijene cijena, radne sate pojedinog blagajnika i ostala analitika. Regularni blagajnik ima mogućnost prijave, kreiranje računa, pridruživanje popusta i tome slično. Opisati tehnologije u procesu izrade aplikacije, opisati pojedine dijelove aplikacije, te kreirati potpuno funkcionalnu aplikaciju. Tema rezervirana za: Antonio Šušnjara Sumentor s FERIT-a: Tomislav Galba
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 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:	02.12.2022.

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, 14.12.2022.

Ime i prezime studenta:	Antonio Šušnjara
Studij:	Preddiplomski stručni studij Računarstvo
Mat. br. studenta, godina upisa:	AI 4639, 27.07.2017.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Blagajna za maloprodajnu trgovinu**

izrađen pod vodstvom mentora Robert Šojo, mag. ing. comp.

i sumentora Doc. dr. sc. Tomislav Galba

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. PREGLED PODRUČJA 2
2.1 Fiskalna.hr 2
2.2 Loyverse POS 2
2.3 eHopper POS 3
3. KORIŠTENE TEHNOLOGIJE 4
3.1 Java 4
3.2 JavaFX 5
3.3 IntelliJ IDEA 5
3.4 Scene Builder 6
3.5 SQL 7
4. FUNKCIONALNOST APLIKACIJE 8
4.1 Baza podataka 8
4.2 MVC arhitektura 10
4.3 Funkcionalnost prijave 12
4.4 Funkcionalnost administratora 13
4.5 Funkcionalnost zaposlenika 21
5. KORIŠTENJE APLIKACIJE 28
5.1 Administrator 28
5.1.1 Upravljanje zaposlenicima 28
5.1.2 Upravljanje računima 30
5.1.3 Upravljanje artiklima 30
5.2 Zaposlenik 31
6. ZAKLJUČAK 33
LITERATURA 34

SAŽETAK.....35
ABSTRACT.....36

1. UVOD

Ovaj rad prolazi kroz kreiranje i korištenje GUI (engl. *Graphic User Interface*) aplikacije za korištenje u sustavu blagajne maloprodajne trgovine zajedno s bazom podataka u kojoj se spremaju podaci o artiklima u trgovini i njihovo stanje. Aplikacija omogućava dvije vrste korisnika: administrator i zaposlenik. Obje vrste korisnika imaju drugačije obaveze i s time drugačije mogućnosti pri korištenju iste aplikacije.

U drugom poglavlju su istražene aplikacije i sustavi u istom području. U trećem poglavlju su opisane tehnologije korištene u aplikaciji. U četvrtom poglavlju je opisana funkcionalnost aplikacije i u petom poglavlju je objašnjeno korištenje aplikacije.

1.1 Zadatak završnog rada

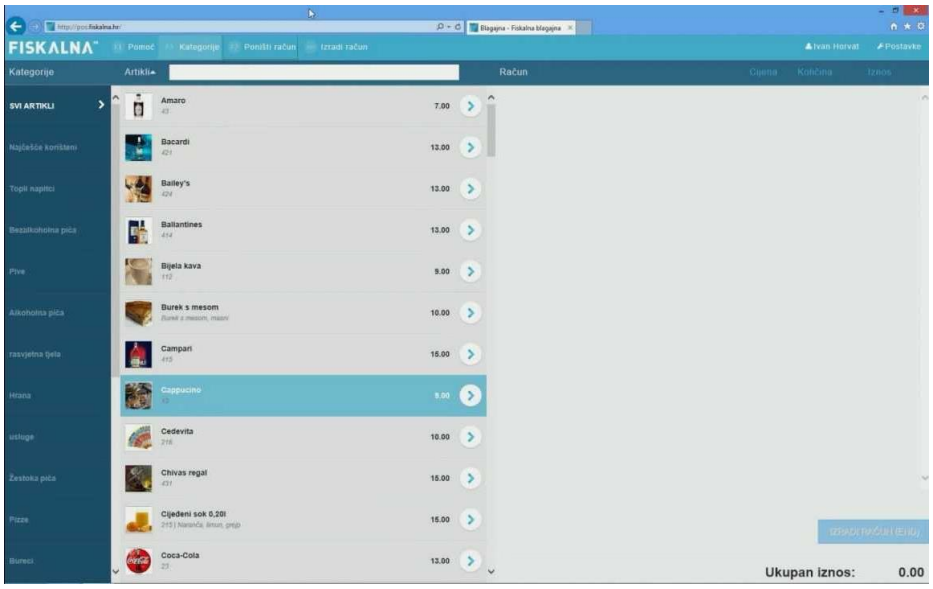
Kreirati aplikaciju blagajne za maloprodajnu trgovinu miješane robe. Aplikacija ima mogućnost korištenja od strane administratora i regularnog blagajnika. Administrator može odobravati vraćanje proizvoda, unos novih proizvoda, kreiranje blagajnika, izmijene cijena, radne sate pojedinog blagajnika i ostala analitika. Regularni blagajnik ima mogućnost prijave, izdavanje računa, pridruživanje popusta i tome slično. Opisati tehnologije u procesu izrade aplikacije, opisati pojedine dijelove aplikacije, te kreirati potpuno funkcionalnu aplikaciju.

2. PREGLED PODRUČJA

U ovom poglavlju su ukratko istražene funkcionalnosti i dizajn aplikacija u istom području. Iz prikazanih sučelja aplikacija se može vidjeti da je jednostavan dizajn zajednička odlika svih aplikacija za blagajnu.

2.1 Fiskalna.hr

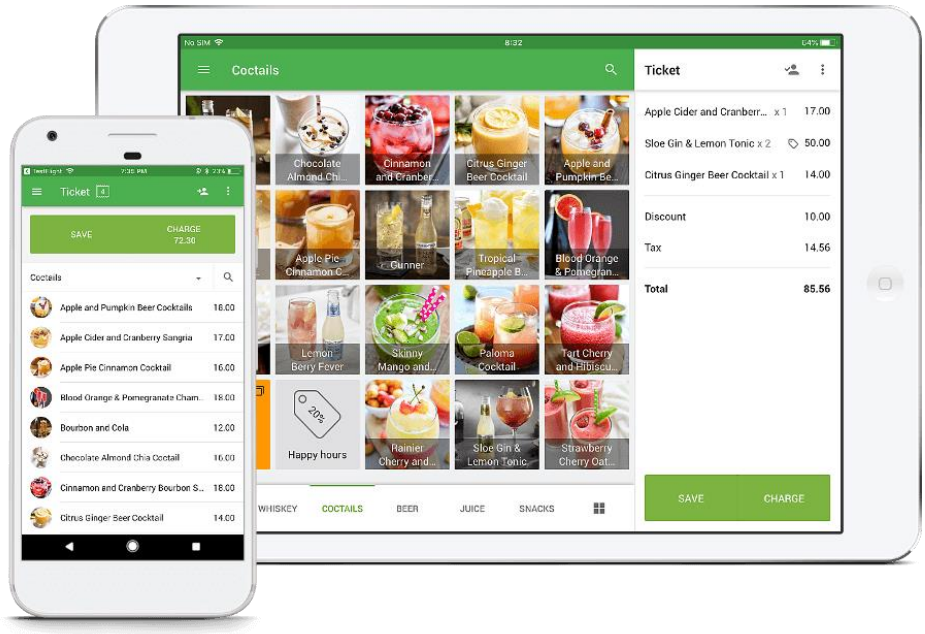
Fiskalna.hr je tvrtka koja prodaje opremu za blagajnu i uz to prodaje aplikacije za mobitel, tablet, računalo i internet preglednik s kojima se koristi ta oprema. Sučelje aplikacije Fiskalna.hr u internet pregledniku je prikazano u slici 2.1 [1]. Aplikacija Fiskalna.hr ima jednostavan dizajn namijenjen za lagano korištenje ali i dalje omogućava sve potrebne funkcionalnosti.



Slika 2.1. *Sučelje aplikacije Fiskalna.hr u internet pregledniku.*

2.2 Loyverse POS

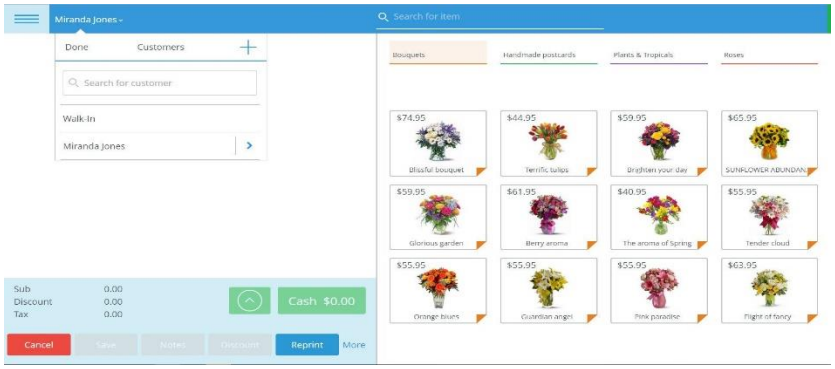
Loyverse POS (engl. *Point-Of-Sale*) je aplikacija za blagajnu koja sve osnovne funkcionalnosti nudi besplatno, primarno za mobilne uređaje ali naplaćuju dodatne funkcionalnosti. Uz to omogućuje i poseban način rada za restorane gdje više računala može biti povezano što omogućuje da se u kuhinji na računalu prikazuju narudžbe čim su zaprimljene. Sučelje Loyverse POS aplikacije na mobitelu i tabletu je prikazano u slici 2.2 [2].



Slika 2.2. *Sučelje Loyverse POS aplikacije.*

2.3 eHopper POS

eHopper POS je aplikacija za blagajnu koja ima besplatnu verziju ali s jako malo funkcionalnosti i kompatibilna je s više uređaja. Sučelje eHopper aplikacije je prikazano slikom 2.3 [3].



Slika 2.3. *Sučelje eHopper POS aplikacije.*

3. KORIŠTENE TEHNOLOGIJE

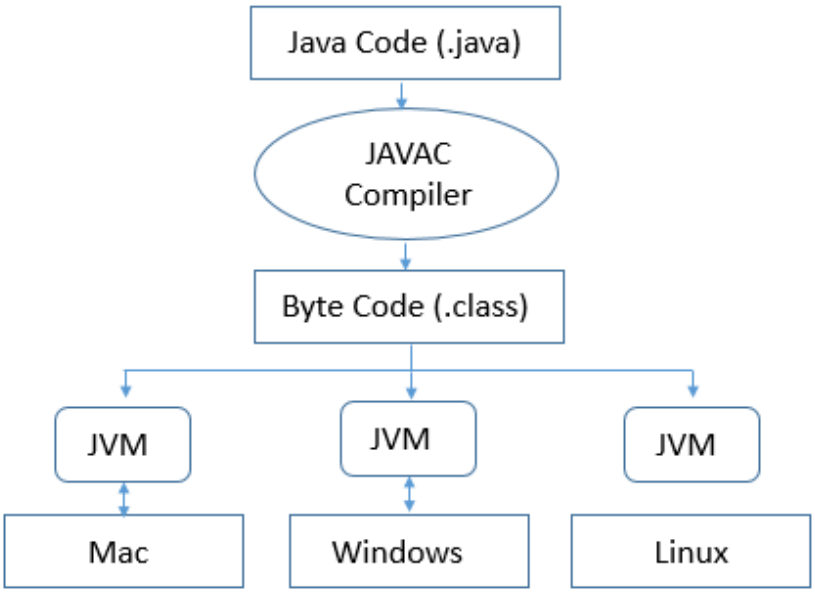
U ovom poglavlju su opisane korištene tehnologije što podrazumijeva korištene programske jezike i programe za pisanje i pravljenje aplikacije.

3.1 Java

Java je besplatni programski jezik koji je 1995. godine napravila tvrtka Sun Microsystems, a od 2009. godine je u vlasništvu Oracle Corporation tvrtke. Java je ažuran jezik koji je jedan od najkorišteniji i najpopularnijih jezika u svijetu. Razlog popularnosti je sličnost sintakse C/C++ jeziku ali ne zahtjeva prevođenja koda za svaku platformu zasebno zato što koristi JVM (engl. *Java Virtual Machine*) koji omogućava izvršavanje istog Java koda na više platformi [4].

Java je objektno-orijentirani programski jezik što je popularniji način programiranja jer omogućava definiranje klasa i njihovih metoda i s time omogućava kreiranja objekata i ostvarivanje apstrakcije, polimorfizma, nasljeđivanja i enkapsulacije [5].

U slici 3.1 je prikazano kako se Java kod prevodi u Byte Code i kako se isti Byte Code izvršava na različitim platformama tako što svaka platforma ima svoj JVM [6].



Slika 3.1. *Prikaz izvršavanja istog koda na više platformi.*

3.2 JavaFX

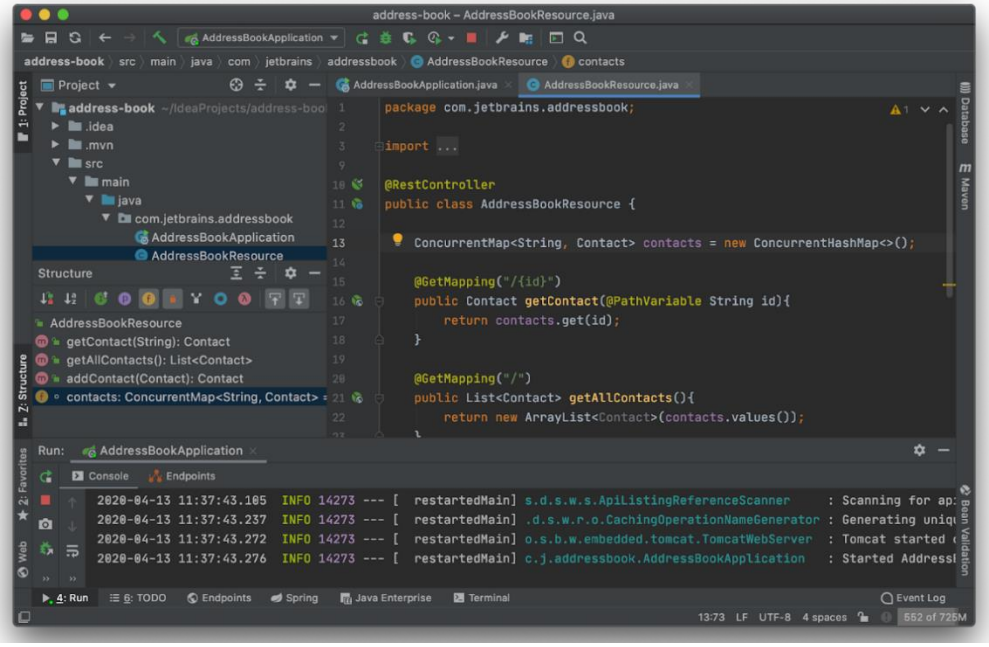
JavaFX je dodatak otvorenog koda (engl. *open-source code*) Java programskom jeziku koji omogućava kreiranje grafičkog korisničkog sučelja i definiranje komunikacije između Java programa i objekata sučelja. Kao dodatak Java jeziku zadržava mogućnost izvršavanja i pokretanja na više platformi koristeći JVM. JavaFX je napravljen kao zamjena za Swing koji je stariji način kreiranja sučelja [7].

JavaFX sučelje se može kreirati vizualno načinom premještanjem elemenata (engl. *drag-and-drop*) koristeći program Scene Builder ili pisanjem koda u FXML jeziku, koji je baziran na XML-u (engl. *Extensible Markup Language*). Izgled FXML koda je prikazan u slici 3.2.

Slika 3.2. *Prikaz FXML koda.*

3.3 IntelliJ IDEA

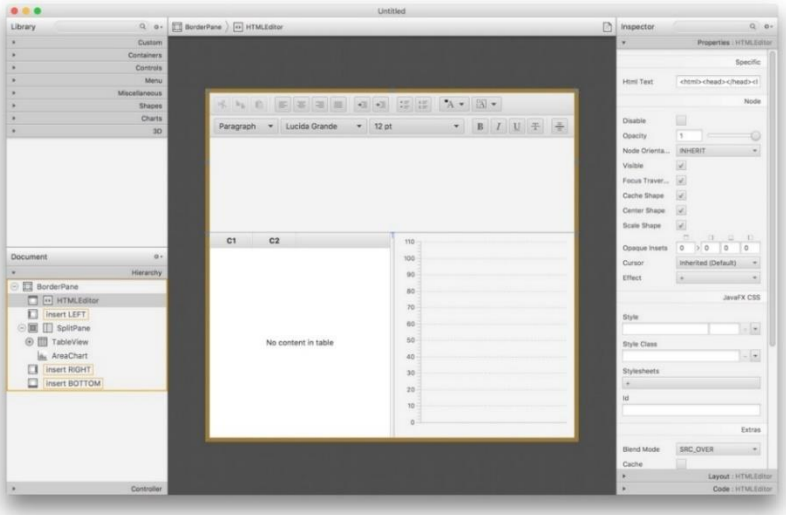
IntelliJ IDEA je razvojno okruženje za Java programski jezik koji je napravila tvrtka JetBrains. IDEA je besplatan program za korištenje u osobne svrhe i kompatibilan je sa SceneBuilder programom. Izgled sučelja IDEA okruženja je prikazano slikom 3.3 [8].



Slika 3.3. *Prikaz IntelliJ IDEA sučelja.*

3.4 Scene Builder

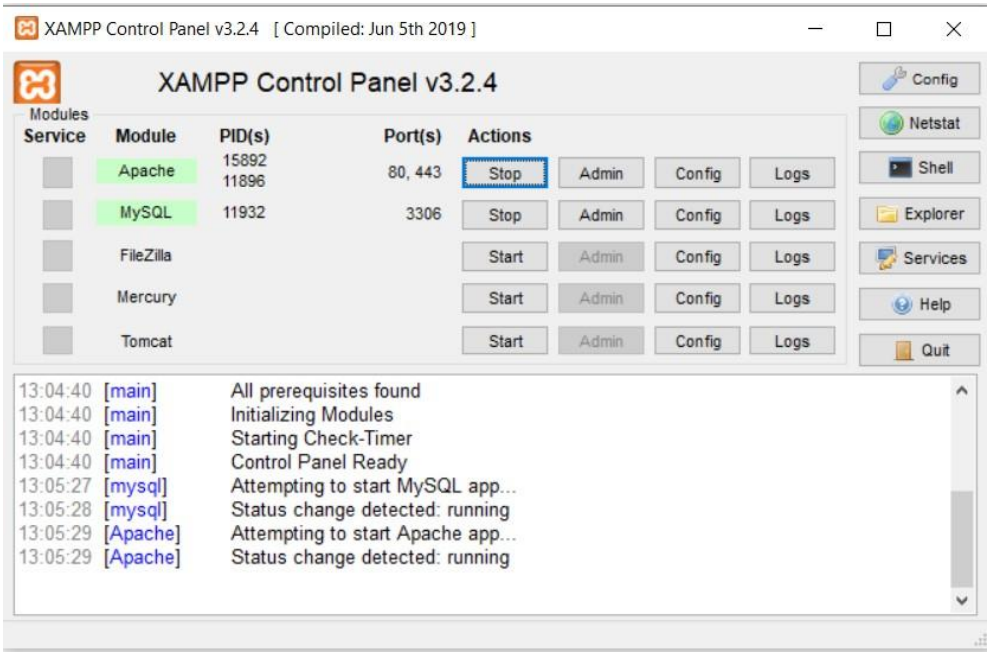
Scene Builder je besplatan i program otvorenog koda za vizualno kreiranje grafičkog sučelja. Radi načinom premještanja elemenata koji automatski generira FXML kod za korištenje u programu [9]. Prikaz sučelja programa Scene Builder za premještanje elemenata je prikazano slikom 3.4.



Slika 3.4. *Prikaz Scene Builder sučelja.*

3.5 SQL

SQL (engl. *Structured Query Language*) je programski jezik za relacijske baze podataka koji se zasniva na postavljanju „upita“ poslužitelju. Omogućava kreiranje i korištenje baze podataka koja sadržava tablice i definira njihove odnose [10]. Komunikacija između baze podataka i Java programa se radi postavljanjem „upita“ od strane Java programa SQL poslužitelju. Za rad baze podataka je korišten program XAMPP koji sadrži phpMyAdmin. phpMyAdmin je program u kojem se koristi MySQL i omogućava vizualno sučelje u Internet pregledniku. Sučelje programa XAMPP s pokrenutim SQL poslužiteljem je prikazano slikom 3.5.



Slika 3.5. *Prikaz XAMPP sučelja.*

4. FUNKCIONALNOST APLIKACIJE

U ovom poglavlju su opisane funkcionalnosti i dizajn programa, opis programskog koda aplikacije i dizajn baze podataka s kojom aplikacija radi.

4.1 Baza podataka

Baza podataka sastoji se od 5 tablica*: „article“, „articles_in_invoice“, „invoice“, „employee“, „work_hours“*. Tablica *„article“* sadrži artikle u ponudi, primarni ključ, naziv, kategoriju, cijenu i trenutno stanje tog artikla u trgovini. Tablica *„article“* prikazana je slikom 4.1.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(8)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	category	enum('food', 'cleaning', 'tools', 'drinks', 'schoo...	utf8mb4_croatian_ci		No	undefined			Change Drop More
☐	3	name	varchar(50)	utf8mb4_croatian_ci		No	*None*			Change Drop More
☐	4	price	decimal(9,2)			No	*None*			Change Drop More
☐	5	storage	int(5)			No	*None*			Change Drop More

Slika 4.1. *Tablica „article“.*

Tablica *„invoice“* sadrži račune koji se naprave tokom rada u aplikaciji. Tablica *„invoice“* sadrži primarni ključ, strani ključ zaposlenika koji je primarni ključ tablice *„employee“*, vrijeme izdavanja računa i popust primijenjen na taj račun u postotcima. Tablica *„invoice“* prikazana je slikom 4.2.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(8)			No	*None*		AUTO_INCREMENT	Change Drop More
☐	2	idEmployee	int(8)			No	*None*			Change Drop More
☐	3	time	timestamp			No	current_timestamp()			Change Drop More
☐	4	discount	smallint(2)			No	0			Change Drop More

Slika 4.2. *Tablica „invoice“.*

Tablica *„articles_in_invoice“* je tablica koja služi kao relacija između tablica *„article“* i *„invoice“* tako što nema primarni ključ, nego sadrži dva strana ključa koji su primarni ključevi tablica između kojih pravi relaciju. Uz to sadrži i broj za količinu artikla u računu. Tablica *„articles_in_invoice“* prikazana je slikom 4.3.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	idArticle	int(8)			No	None			Change Drop More
☐	2	idInvoice	int(8)			No	None			Change Drop More
☐	3	count	int(5)			No	None			Change Drop More

Slika 4.3. *Tablica „articles_in_invoice“.*

Tablica *„employee“* sadrži podatke o zaposlenicima, primarni ključ, ime, prezime, korisničko ime, lozinku i oznaku za administratora sustava. Tablica *„employee“* prikazana je slikom 4.4.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(8)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	name	varchar(30)	utf8mb4_croatian_ci		No	None			Change Drop More
☐	3	surname	varchar(30)	utf8mb4_croatian_ci		No	None			Change Drop More
☐	4	user	varchar(30)	utf8mb4_croatian_ci		No	None			Change Drop More
☐	5	pass	varchar(30)	utf8mb4_croatian_ci		No	None			Change Drop More
☐	6	admin	tinyint(1)			No	0			Change Drop More

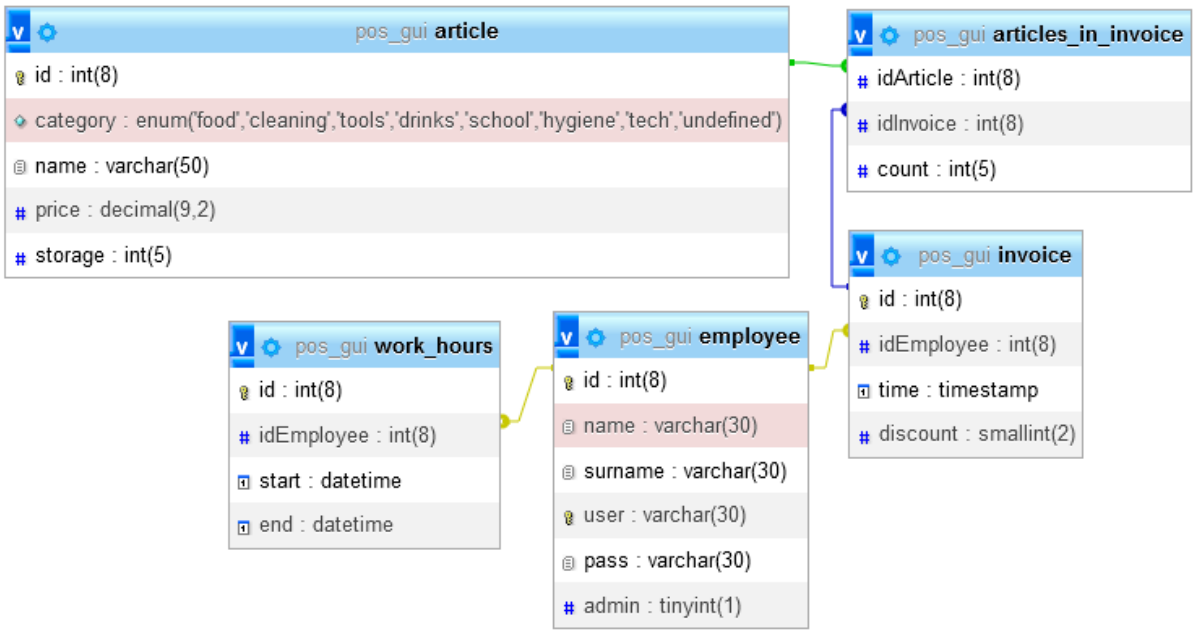
Slika 4.4. *Tablica „employee“.*

Tablica *„work_hours“* sadrži podatke o početku i kraju rada zaposlenika koji se prate prijavom i odjavom u aplikaciju. Tablica *„work_hours“* sadrži primarni ključ, strani ključ zaposlenika, početak i kraj rada u vremenskom zapisu. Tablica *„work_hours“* prikazana je slikom 4.5.

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(8)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	idEmployee	int(8)			No	None			Change Drop More
☐	3	start	datetime			No	None			Change Drop More
☐	4	end	datetime			No	current_timestamp()			Change Drop More

Slika 4.5. *Tablica „work_hours“.*

PhpMyAdmin omogućuje vizualni prikaz relacija u bazi podataka. Dijagram relacija je prikazan slikom 4.6.



Slika 4.6. *Dijagram relacija baze podataka.*

Kako bi koristili bazu podataka u programu potrebno je spojiti aplikaciju s bazom podataka. Spajanje s bazom podataka koje se koristi na potrebnim mjestima je prikazano slikom 4.7.

```
public Connection getConnection() {

    Connection conn;
    try {
        conn = DriverManager.getConnection( url: "jdbc:mysql://localhost:3306/pos_gui", user: "root", password: "");
        return conn;
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return null;
}
```

Slika 4.7. *Programski kod za dohvaćanje konekcije s bazom podatak.*

4.2 MVC arhitektura

MVC (engl. *Model - View - Controller*) je arhitektura s kojom se lagano mogu odrediti komponente programskog koda i njihove odgovornosti. „*Model*“ komponente sadrže opis složenih podataka, konstruktor i metode postavljanja i dohvaćanja vrijednosti. Najčešće se koriste kao prikaz tablice iz baze podataka u programskom kodu. „*View*“ je komponenta koja definira vizualni prikaz podataka u prozoru aplikacije. „*View*“ komponente u ovoj aplikaciji su napravljene koristeći JavaFX i FXML *„markup“* jezik. „*Controller*“ komponenta upravlja zadanom „*View*“ komponentom i koristi podatke iz „*Model*“ komponente za rad.

Korištene „*Model*“ komponente u ovoj aplikaciju su: „*Article.java*“ za prikaz tablice „*article*“, „*Invoice.java*“ za prikaz tablice „*invoice*“, „*ArticleInInvoice.java*“ koja je drugačija od tablice „*articles_in_invoice*“ po tome što sadrži ime artikla, količinu i izračunatu cijenu jer se u programu koristi samo kao element unutar konteksta računa, „*Employee.java*“ za prikaz tablice „*employee*“ i „*WorkHours.java*“ za prikaz tablice „*work_hours*“. Primjer „*Model-a*“ „*ArticleInInvoice.java*“ prikazan je slikom 4.8.

```java
public class ArticleInInvoice {
    private String article;
    private Integer amount;
    private Float price;

    public ArticleInInvoice(String article, Integer amount, Float price) {
        this.article = article;
        this.amount = amount;
        this.price = price;
    }

    public String getArticle() { return article; }
    public void setArticle(String article) { this.article = article; }
    public Integer getAmount() { return amount; }
    public void setAmount(Integer amount) { this.amount = amount; }
    public Float getPrice() { return price; }
    public void setPrice(Float price) { this.price = price; }
}
```

Slika 4.8. *Model „ArticleInInvoice.java“.*

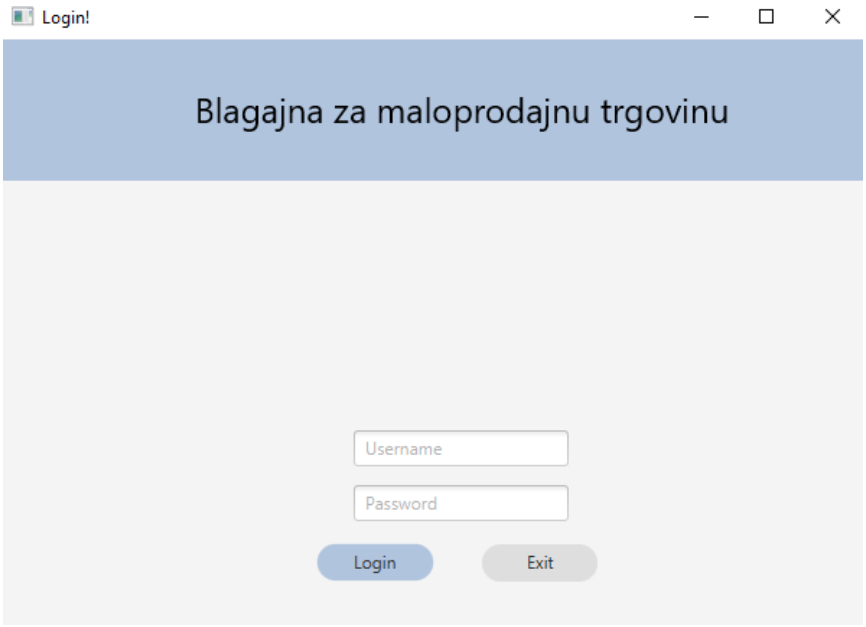
Korištene „*View*“ komponente i pripadajuće „*Controller*“ komponente su:

- loginView.fxml – LoginController.java
- adminView.fxml – AdminController.java
    - adminArticleView.fxml – AdminArticleController.java
    - adminEmployeeView.fxml – AdminEmployeeController.java
    - adminInvoiceView.fxml – AdminInvoiceController.java
- employeeView.fxml – EmployeeController.java
    - employeeMainView.fxml – EmployeeMainController.java
    - employeeInventoryView.fxml – EmployeeInventoryController.java
    - employeePersonalView.fxml – EmployeePersonalController.java

„*View*“ i „*Controller*“ komponente su podijeljene jer se unutar jednog „*View-a*“ može instancirati neki drugi „*View*“ koji ima svoj pripadajući „*Controller*“.

4.3 Funkcionalnost prijave

Pri pokretanju programa koristi se JavaFX metoda *„start()“* koja instancira *„loginView.fxml“* prikazan na slici 4.9.



Slika 4.9. *Sučelje za prijavu.*

Pri instanciranju *„loginView-a“* se kreira i *„LoginController“* objekt za kontrolu *„View-a“* za prijavu. *„LoginController“* sadrži programski kod koji nakon pritiska na tipku *„Login“* pokreće metodu *„login()“*. Metoda *„login()“* nakon provjere jesu li potrebna polja popunjena, pravi upit bazi podataka i dohvaća rezultat zaposlenika ako su uneseni točni podaci. Na slici 4.10 je prikazano dohvaćanje zaposlenika iz baze podataka.

```
private void login() {
    if (tfUsername.getText() == null || tfUsername.getText().trim().isEmpty()
            || tfPassword.getText() == null || tfPassword.getText().trim().isEmpty()) {
        lbMsg.setText("Fill both fields!");
    } else {
        Connection conn = getConnection();
        String query = "SELECT * FROM employee " +
                "WHERE user ='" + tfUsername.getText() + "' " +
                "AND pass='" + tfPassword.getText() + "'";
        PreparedStatement st;
        ResultSet rs;
        try {
            Stage stage = (Stage) btnLogin.getScene().getWindow();
            st = conn.prepareStatement(query);
            rs = st.executeQuery();
```

Slika 4.10. *Programski kod za dohvaćanje zaposlenika.*

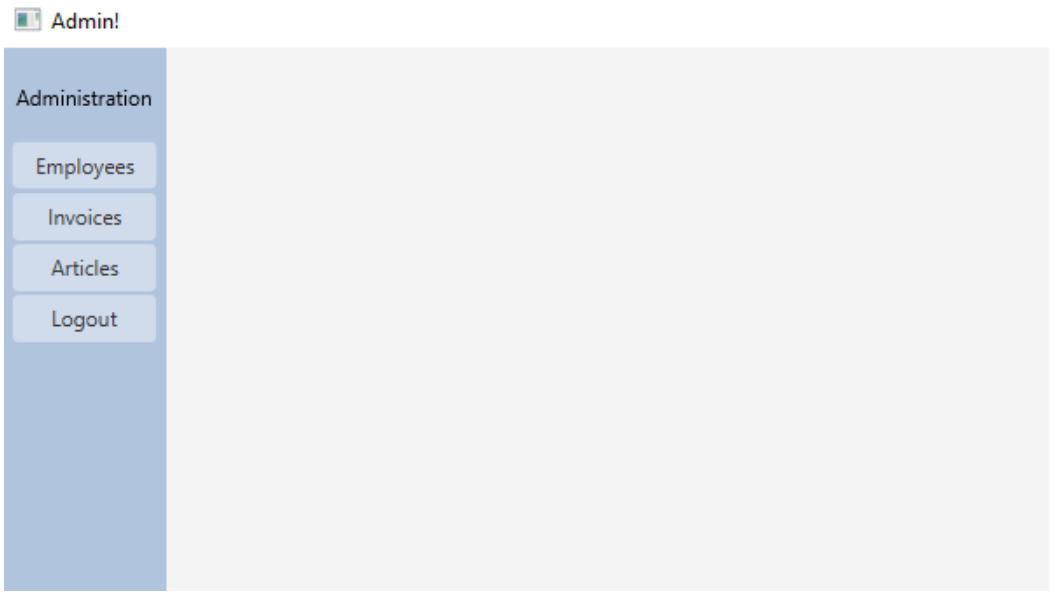
Ako dohvaćeni zaposlenik nije administrator otvara se „*employeeVew*“ i „*EmployeeController-u*“ se prosljeđuje „*Employee*“ zbog korištenja podataka zaposlenika u daljnjim „*Controller-ima*“. Ako je dohvaćeni zaposlenik administrator, programski kod se grana i otvara „*adminView*“. Grananje i otvaranje pripadajućih „*View-ova*“ je prikazano na slici 4.11.

```
    if (rs.getInt("admin") == 1) {
        FXMLLoader fxmlLoader = new FXMLLoader(MainApp.class.getResource("adminView.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1200, 700);
        stage.setTitle("Admin!");
        stage.setScene(scene);
        stage.show();
    } else {
        FXMLLoader fxmlLoader = new FXMLLoader(MainApp.class.getResource("employeeView.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 1200, 700);
        EmployeeController employeeController = fxmlLoader.getController();
        employeeController.setUser(
                new Employee(rs.getInt("id"),
                        rs.getString("name"),
                        rs.getString("surname"),
                        rs.getString("user"),
                        rs.getString("pass")));
        stage.setTitle("POS");
        stage.setScene(scene);
        stage.show();
    }
}
```

Slika 4.11. *Programski kod za otvaranje pripadajućih „View-ova“ nakon uspješne prijave.*

4.4 Funkcionalnost administratora

Prvo sučelje koje se prikazuje administratoru nakon prijave je prikazano na slici 4.12.



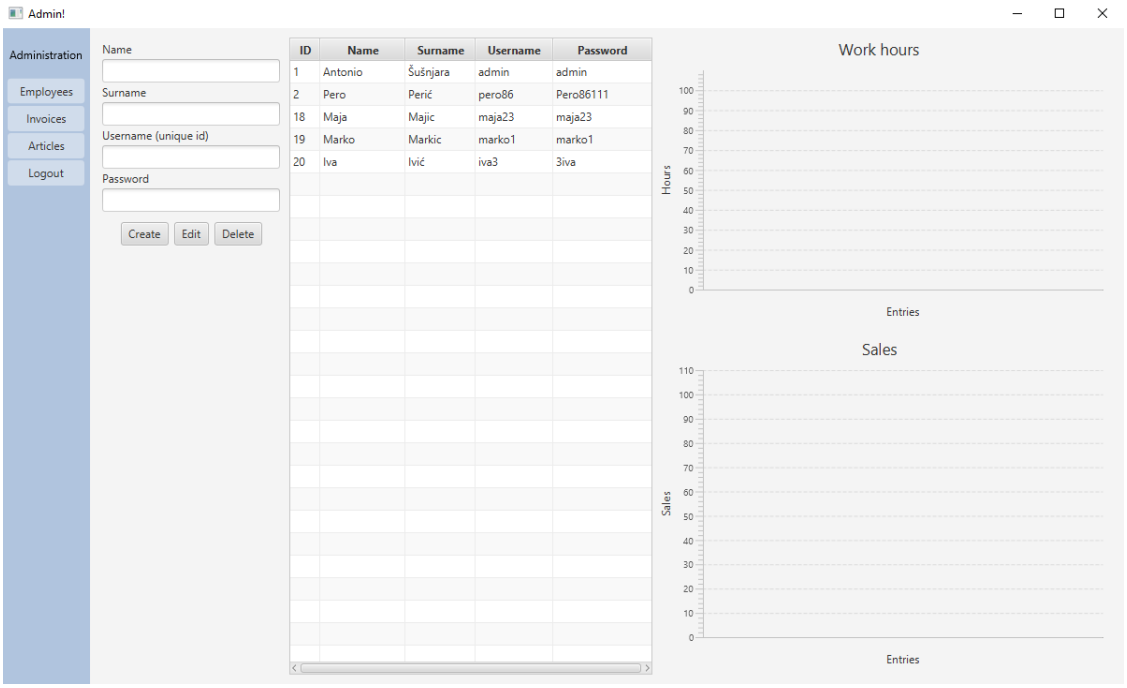
Slika 4.12. *Sučelje aplikacije za administratora.*

Sučelje za administratora nakon prijave je prazno dok se ne odabere jedna od opcija za rad. Pritiskom tipke s lijeve strane prazni prostor se popuni jednim od „*View-ova*“ pod „*adminView*“. Tipka „*Logout*“ vrača aplikaciju na sučelje za prijavu. Programski kod za otvaranje pripadajućih „*View-ova*“ je prikazan slikom 4.13.

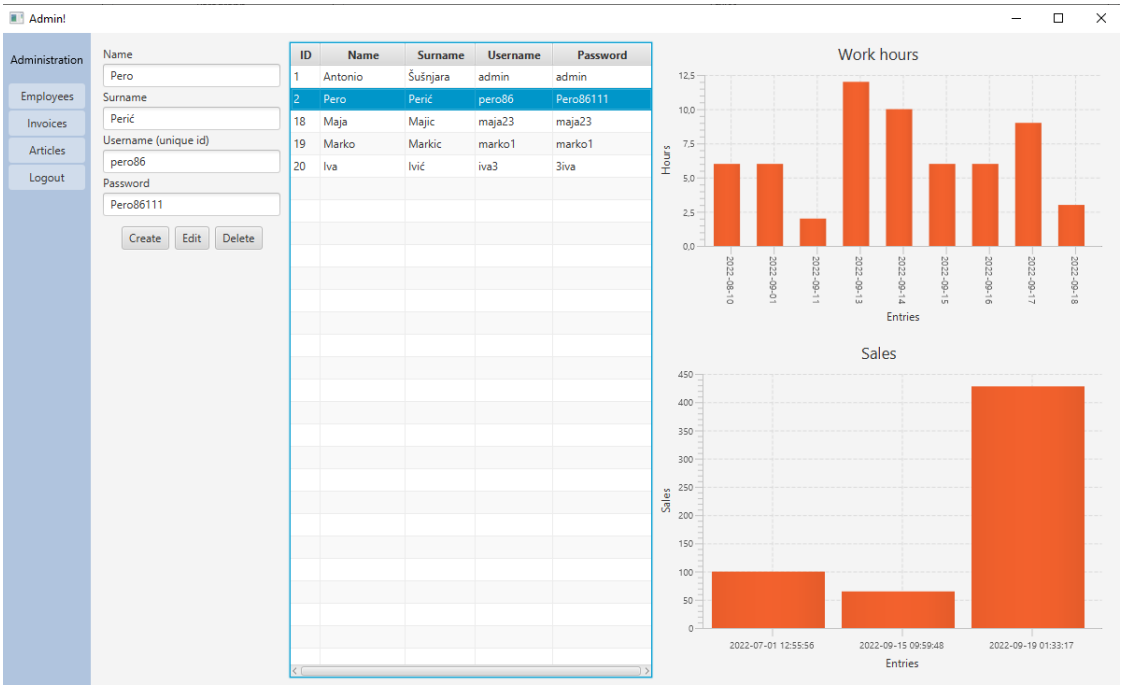
```
@FXML
protected void showEmployees() throws IOException {
    AdminController.contentPane = FXMLLoader.load(getClass().getResource("adminEmployeeView.fxml"));
    try {
        this.pane.getChildren().clear();
        this.pane.getChildren().add(AdminController.contentPane);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
@FXML
protected void showArticles() throws IOException {
    AdminController.contentPane = FXMLLoader.load(getClass().getResource("adminArticleView.fxml"));
    try {
        this.pane.getChildren().clear();
        this.pane.getChildren().add(AdminController.contentPane);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
@FXML
protected void showInvoices() throws IOException {
    AdminController.contentPane = FXMLLoader.load(getClass().getResource("adminInvoiceView.fxml"));
    try {
        this.pane.getChildren().clear();
        this.pane.getChildren().add(AdminController.contentPane);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Slika 4.13. *Programski kod odabira sučelja za rad administratora.*

Aplikacija nakon pritiska tipke „*Employees*“ otvara *„adminEmployeeView*“. Sučelje aplikacije za administraciju zaposlenika je prikazano na slici 4.14. Sučelje za administraciju zaposlenika sadrži formu za unos novih zaposlenika, uređivanje podataka i brisanje postojećih zaposlenika. Uz to sadrži tablicu svih zaposlenika i grafove koji prikazuju zadnjih deset unosa za radne sate i prodaju. Forma i grafovi se popune dvostrukim klikom na redak željenog zaposlenika. Na slici 4.15 je prikazano sučelje s popunjenim podacima.



Slika 4.14. *Sučelje za administraciju zaposlenika.*



Slika 4.15. *Popunjeno sučelje za administraciju zaposlenika.*

Na slici 4.16 je prikazana metoda za popunjavanje tablice pri inicijalizaciji „*View-a*“ podacima zaposlenika koje dohvati metodom *„getEmployees()“*. Na slici 4.17 je prikazana metoda *„getEmployees()“* koja dohvaća sve zaposlenike iz baze podataka.

```
private void updateTable() {
    ObservableList<Employee> employees = getEmployees();

    colId.setCellValueFactory(new PropertyValueFactory<>("id"));
    colName.setCellValueFactory(new PropertyValueFactory<>("name"));
    colSurname.setCellValueFactory(new PropertyValueFactory<>("surname"));
    colUser.setCellValueFactory(new PropertyValueFactory<>("username"));
    colPass.setCellValueFactory(new PropertyValueFactory<>("password"));

    tvEmployees.setItems(employees);
}
```

Slika 4.16. *Programski kod popunjavanja tablice.*

```
public ObservableList<Employee> getEmployees() {
    ObservableList<Employee> employeeList = FXCollections.observableArrayList();
    Connection conn = getConnection();
    String query = "SELECT * FROM employee";
    Statement st;
    ResultSet rs;
    try {
        st = conn.createStatement();
        rs = st.executeQuery(query);
        Employee employee;
        while (rs.next()) {
            employee = new Employee(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("surname"),
                    rs.getString("user"),
                    rs.getString("pass"));
            employeeList.add(employee);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return employeeList;
}
```

Slika 4.17. *Programski kod dohvaćanja zaposlenika.*

Na slici 4.18 je prikazana metoda za unos novih zaposlenika. Metode za uređivanje i brisanje rade na označenom zaposleniku. Prikaz upita za bazu podataka pri uređivanju i brisanju su na slikama 4.19 i 4.20.

```
@FXML
private void onCreateEmployee() {
    if (checkEmpty()) {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setContentText("Fill all of text fields");
        a.show();
    } else {
        String query = "INSERT INTO employee VALUES(NULL," +
                "'" + tfName.getText() + "'," +
                "'" + tfSurname.getText() + "'," +
                "'" + tfUser.getText() + "'," +
                "'" + tfPass.getText() + "', '0')";

        Connection conn = getConnection();
        Statement st;
        try {
            st = conn.createStatement();
            st.executeUpdate(query);
            updateTable();
        } catch (Exception e) {
            Alert a = new Alert(Alert.AlertType.WARNING);
            a.setContentText("Could not create a new Employee");
            a.show();
            e.printStackTrace();
        }
    }
}
```

Slika 4.18. *Programski kod unosa novog zaposlenika.*

```
Employee employee = tvEmployees.getSelectionModel().getSelectedItem();

String query = "UPDATE employee SET " +
        "name='" + tfName.getText() + "'," +
        "surname='" + tfSurname.getText() + "'," +
        "user='" + tfUser.getText() + "'," +
        "pass='" + tfPass.getText() + "'" +
        "WHERE id='" + employee.getId() + "'";
```

Slika 4.19. *Upit za uređivanje zaposlenika.*

```
Employee employee = tvEmployees.getSelectionModel().getSelectedItem();

String query = "DELETE FROM employee " +
        "WHERE user='" + employee.getUsername() + "' " +
        "AND id='" + employee.getId() + "'";
```

Slika 4.20. *Upit za brisanje zaposlenika.*

Grafovi se popunjavaju dohvaćanim podatcima iz baze podataka za odabranog zaposlenika. Upiti za bazu podataka su prilagođeni tako da dohvaćaju podatke u traženom obliku. Upit za

dohvaćanje radnih sati je prikazan na slici 4.21, a upit za dohvaćanje podataka o prodaji je prikazan na slici 4.22.

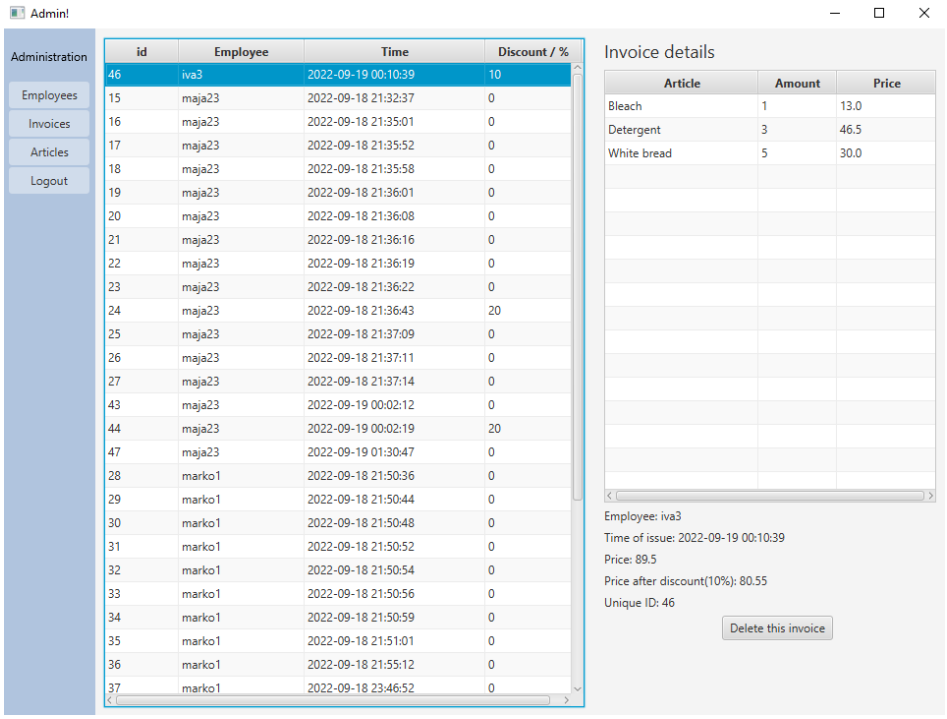
```
String query = "SELECT * FROM " +
        "(SELECT TIMESTAMPDIFF(MINUTE, start, end) AS min, CONVERT(start, DATE) AS date " +
        "FROM work_hours " +
        "WHERE idEmployee=" + clickedRow.getId() + " " +
        "ORDER BY start DESC limit 0,10) AS selection " +
        "ORDER BY date ASC";
```

Slika 4.21. *Upit za dohvaćanje radnih sati zaposlenika.*

```
String query = "SELECT * FROM " +
        "(SELECT SUM(a.price*ai.count) AS price, CONVERT(i.time, DATETIME) AS date, i.discount AS discount " +
        "FROM article a " +
        "INNER JOIN articles_in_invoice ai ON a.id = ai.idArticle " +
        "INNER JOIN invoice i ON ai.idInvoice = i.id " +
        "WHERE i.idEmployee = '" + clickedRow.getId() + "' " +
        "GROUP BY i.id " +
        "ORDER BY date DESC limit 0,10) AS selection " +
        "ORDER BY date ASC";
```

Slika 4.22. *Upit za dohvaćanje podataka o prodaji zaposlenika.*

Pritiskom tipke *„Invoices“* u lijevom stupcu aplikacija otvara *„adminInvoicesView“* koji sadrži tablicu svih računa iz baze podataka, detaljan ispis odabranog računa i tipku za brisanje odabranog računa. Na slici 4.23 je prikazano sučelje nakon što je popunjeno dvostrukim klikom na željeni račun u tablici.



Slika 4.23. *Sučelje za administraciju računa.*

Na slici 4.24 je prikaz metode *„getInvoices()“* s upitom za bazu podataka koji dohvaća i korisničko ime zaposlenika koji je napravio račun. Na slici 4.25 je prikaz metode *„getArticles()“* kojom se popunjava lista *„Invoice details“* s artiklima iz odabranog računa.

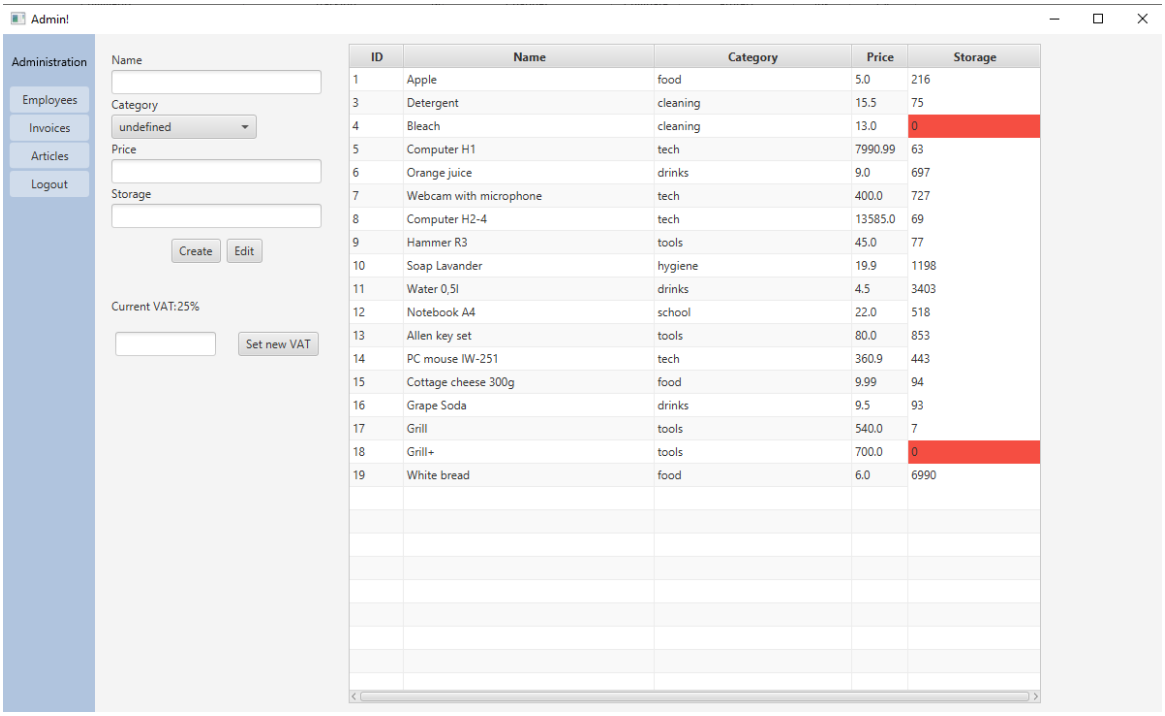
```
public ObservableList<Invoice> getInvoices() {
    ObservableList<Invoice> invoiceList = FXCollections.observableArrayList();
    Connection conn = getConnection();
    String query = "SELECT i.id, e.user, i.time, i.discount FROM invoice i " +
            "INNER JOIN employee e ON e.id = i.idEmployee";
    Statement st;
    ResultSet rs;
    try {
        st = conn.createStatement();
        rs = st.executeQuery(query);
        Invoice invoice;
        while (rs.next()) {
            invoice = new Invoice(
                    rs.getInt("id"),
                    rs.getString("user"),
                    rs.getString("time"),
                    rs.getInt("discount"));
            invoiceList.add(invoice);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return invoiceList;
}
```

Slika 4.24. *Programski kod metode „getInvoices()“.*

```
private ObservableList<ArticleInInvoice> getArticles(Invoice clickedRow) {
    ObservableList<ArticleInInvoice> articleList = FXCollections.observableArrayList();
    Connection conn = getConnection();
    String query = "SELECT a.name AS article, a.price AS price, ai.count AS amount FROM articles_in_invoice ai " +
            "INNER JOIN article a ON a.id = ai.idArticle " +
            "WHERE ai.idInvoice = '" + clickedRow.getId() + "'";
    Statement st;
    ResultSet rs;
    try {
        st = conn.createStatement();
        rs = st.executeQuery(query);
        ArticleInInvoice article;
        while (rs.next()) {
            article = new ArticleInInvoice(
                    rs.getString("article"),
                    rs.getInt("amount"),
                    rs.getFloat("price") * rs.getInt("amount"));
            articleList.add(article);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return articleList;
}
```

Slika 4.25. *Programski kod metode „getArticles()“.*

Pritiskom tipke *„Articles“* u lijevom stupcu aplikacija otvara *„adminArticlesView“* koji sadrži tablicu svih artikala iz baze podataka i formu za unos novih artikala ili uređivanje postojećih artikala. Uz to sadrži i tipku „*Set new VAT*“ za promjenu poreza na dodanu vrijednost (engl. *value added tax*). U tablici se crvenom bojom označavaju ćelije stupca „*Storage“* kad je količina tog artikla „0“. Forma se na isti način popunjava dvostrukim klikom na redak u tablici. Na slici 4.26 je prikazano sučelje za administraciju artikala.



Slika 4.26. *Sučelje za administraciju artikala.*

Na slici 4.27 je prikazana metoda *„onCreateArticle()“* koja unosi novi artikl u bazu podataka. Na slici 4.28 je prikazana metoda *„onEditArticle()“* koja uređuje označeni artikl u tablici.

```
@FXML
private void onCreateArticle() {
    if (checkEmpty()) {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setContentText("Fill all of text fields");
        a.show();
    } else {
        String query = "INSERT INTO article VALUES(NULL," +
                "'" + choiceCategory.getSelectionModel().getSelectedItem().toString() + "'," +
                "'" + tfName.getText() + "'," +
                "'" + tfPrice.getText() + "'," +
                "'" + tfStorage.getText() + "')";

        Connection conn = getConnection();
        Statement st;
        try {
            st = conn.createStatement();
            st.executeUpdate(query);
            updateTable();
        } catch (Exception e) {
            Alert a = new Alert(Alert.AlertType.WARNING);
            a.setContentText("Could not create a new Article");
            a.show();
            e.printStackTrace();
        }
    }
}
```

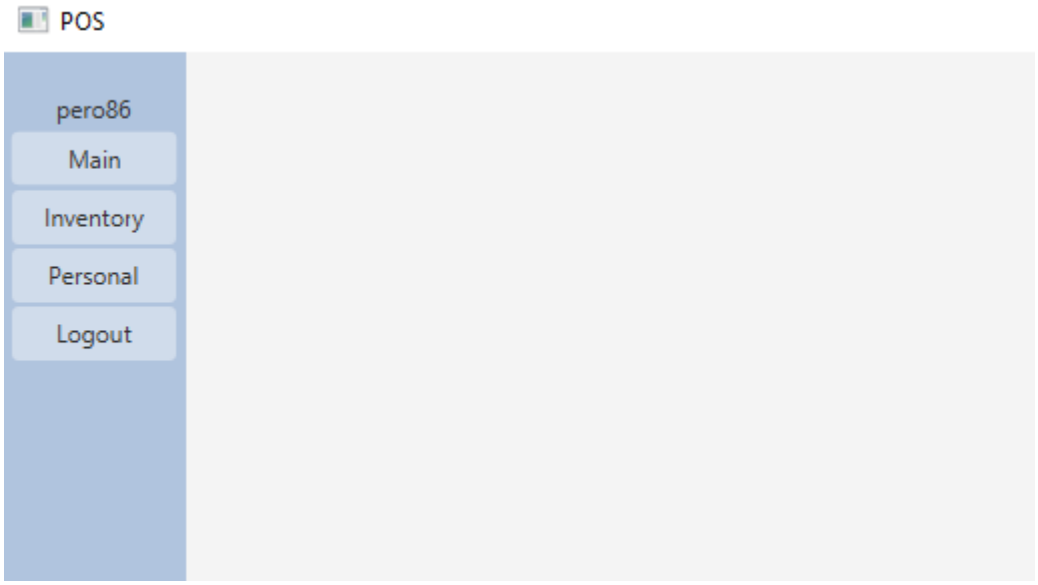
Slika 4.27. *Programski kod metode „onCreateArticle()“.*

```
@FXML
private void onEditArticle() {
    if (checkEmpty()) {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setContentText("Fill all of text fields");
        a.show();
        return;
    }
    Article article = tvArticle.getSelectionModel().getSelectedItem();

    String query = "UPDATE article SET " +
            "name='" + tfName.getText() + "'," +
            "category='" + choiceCategory.getSelectionModel().getSelectedItem().toString() + "'," +
            "price='" + tfPrice.getText() + "'," +
            "storage='" + tfStorage.getText() + "'" +
            "WHERE id='" + article.getId() + "'";

    Connection conn = getConnection();
    Statement st;
    try {
        st = conn.createStatement();
        st.executeUpdate(query);
        updateTable();
    } catch (Exception e) {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setContentText("Could not edit Article");
        a.show();
        e.printStackTrace();
    }
}
```

Slika 4.28. *Programski kod metode „onEditArticle()“.*

4.5 Funkcionalnost zaposlenika

Prvo sučelje koje se prikaže zaposleniku nakon prijave je prikazano na slici 4.29.



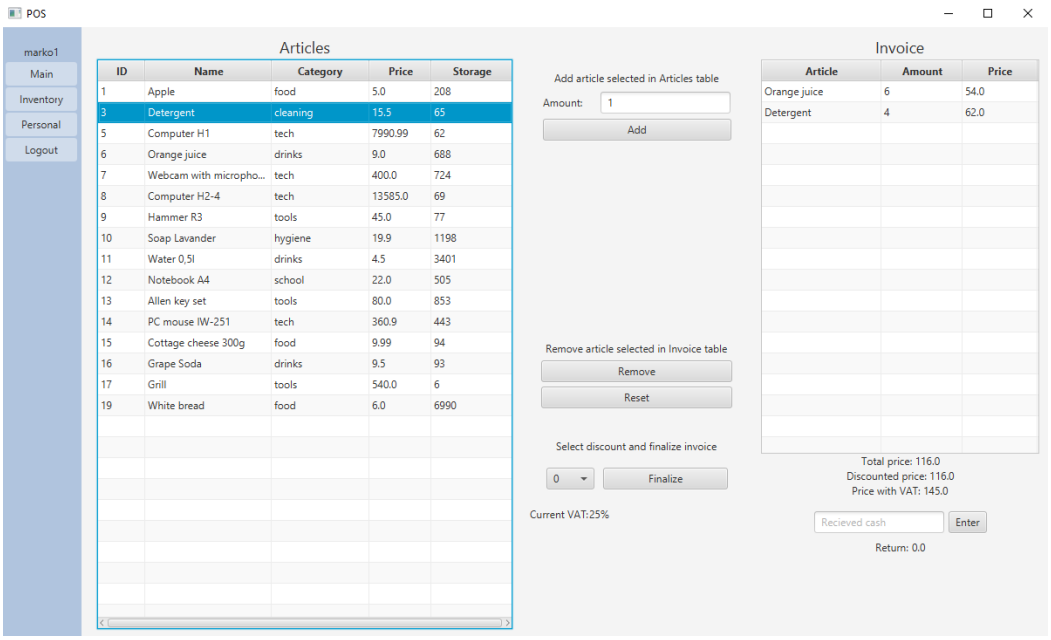
Slika 4.29. *Sučelje zaposlenika nakon prijave.*

Tipka *„Logout“* pokreće metodu *„logout()“* koja uzima vrijeme koje je spremljeno pri prijavi zaposlenika i unosi novi element u tablicu *„work_hours“* za praćenje radnog vremena u bazi podataka, nakon toga vrača aplikaciju na sučelje za prijavu. Upit za unos novog elementa radnog vremena je prikazan na slici 4.30.

```
protected void logout() throws IOException {
    String query = "INSERT INTO work_hours " +
            "VALUES(NULL, '" + user.getId() + "', '"+ timeFormat.format(startTime) +"', current_timestamp())";
```

Slika 4.30. *Upit za unos radnog vremena.*

Tipka *„Main“* pokreće metodu *„showMain()“* koja otvara novo sučelje za rad zaposlenika i *„employeeMainController-u“* prosljeđuje podatke o prijavljenom zaposleniku. Na slici 4.31 je prikaz sučelja za rad zaposlenika. Zaposleniku se u tablici „*Articles“* ne prikazuju artikli kojih trenutno nema u trgovini.



Slika 4.31. *Sučelje za rad zaposlenika.*

Tablica *„Articles“* se popunjava na isti načina kao i tablica u sučelju za administraciju artikala. Pritiskom tipke *„Add“* se poziva metoda *„addItem()“* koja dodaje iznad navedenu količinu označenog artikla u račun, koji je s desne strane ispisan u trenutnom stanju. Metoda *„addItem()“* je prikazana na slici 4.32.

```
@FXML
private void addItem() {
    try {
        String article = tvArticle.getSelectionModel().getSelectedItem().getName();
        Integer amount = Integer.parseInt(tfAmount.getText());
        Float price = tvArticle.getSelectionModel().getSelectedItem().getPrice() * amount;
        Integer storage = tvArticle.getSelectionModel().getSelectedItem().getStorage();
        if (amount > storage) {
            Alert a = new Alert(Alert.AlertType.WARNING);
            a.setContentText("Can't add amount more than in storage");
            a.show();
        } else {
            articleInInvoice.add(new ArticleInInvoice(article, amount, price));
            articles.forEach((item) -> {
                if (item.getName() == article) {
                    item.setStorage(storage - amount);
                }
            });
            updateArticles();
        }
    } catch (Exception e) {
        Alert a = new Alert(Alert.AlertType.WARNING);
        a.setContentText("Error with article selection");
        a.show();
        e.printStackTrace();
    }
    tvInvoice.setItems(articleInInvoice);
    updatePrice();
    refreshInputs();
}
```

Slika 4.32. *Programski kod metode „addItem()“.*

Metoda *„addItem()“* poziva dodatne metode za osvježavanje podataka na trenutno stanje, slične metode se koriste u skoro svim tablicama. Pri dodavanju artikala u račun se smanjuje broj

artikala na stanju i aplikacija ne dopušta stanje ispod. Pritiskom tipke *„Remove“* poziva se metoda *„removeItem()“* kojom se iz računa briše označeni artikl i ta količina se vrati u tablicu artikala. Tipka *„Reset“* poziva metodu *„removeItems()“* koja obriše sve iz računa i vrati sučelje u početno stanje. Na slici 4.33 su prikazane metode *„removeItem()“* i *„removeItems()“*.

```
@FXML
private void removeItem() {
    articles.forEach((item) -> {
        if (item.getName() == tvInvoice.getSelectionModel().getSelectedItem().getArticle()) {
            item.setStorage(item.getStorage() + tvInvoice.getSelectionModel().getSelectedItem().getAmount());
        }
    });
    updateArticles();
    articleInInvoice.remove(tvInvoice.getSelectionModel().getSelectedItem());
    tvInvoice.setItems(articleInInvoice);
    updatePrice();
    refreshInputs();
}

@FXML
private void removeItems() {
    articleInInvoice.clear();
    tvInvoice.setItems(articleInInvoice);
    refresh();
}
```

Slika 4.33. *Programski kod metoda „removeItem()“ i „removeItems()“.*

Pokraj tipke *„Finalize“* se može odabrati željeni popust na cijeli račun. Pritiskom tipke *„Finalize“* poziva se metoda *„finalizeInvoice()“* koja čita podatke iz tablice računa i iz tih podataka pravi upite za kreiranje elemenata u *„invoice“* i *„articles_in_invoice“* tablicama u bazi podataka. Nakon toga se generira tekstualna datoteka koja u imenu sadrži *„id“* računa i unutar datoteke su ispisani svi artikli i detalji računa. Na slikama 4.34 i 4.35 je prikazan programski kod metode *„finalizeInvoice()“*. Na slici 4.36 je prikazan izgled ispisa računa u tekstualnoj datoteci.

```
for (ArticleInInvoice item : articleInInvoice) {
    for (Article itemArticle : articles) {
        if (item.getArticle() == itemArticle.getName()) {
            String query = "INSERT INTO articles_in_invoice " +
                    "VALUES('" + itemArticle.getId().toString() + "'," +
                    "'" + invoice.getId().toString() + "'," +
                    "'" + item.getAmount().toString() + "')";

            String query2 = "UPDATE article " +
                    "SET storage=storage-" + item.getAmount().toString() + " " +
                    "WHERE id=" + itemArticle.getId().toString();
            Statement st;
            try {
                st = conn.createStatement();
                st.executeUpdate(query);
                st.executeUpdate(query2);
            } catch (Exception e) {
                Alert a = new Alert(Alert.AlertType.WARNING);
                a.setContentText("Error making an Invoice");
                a.show();
                e.printStackTrace();
            }
        }
    }
}
```

Slika 4.34. *Programski kod metode „finalizeInvoice()“ za unos u bazu podataka.*

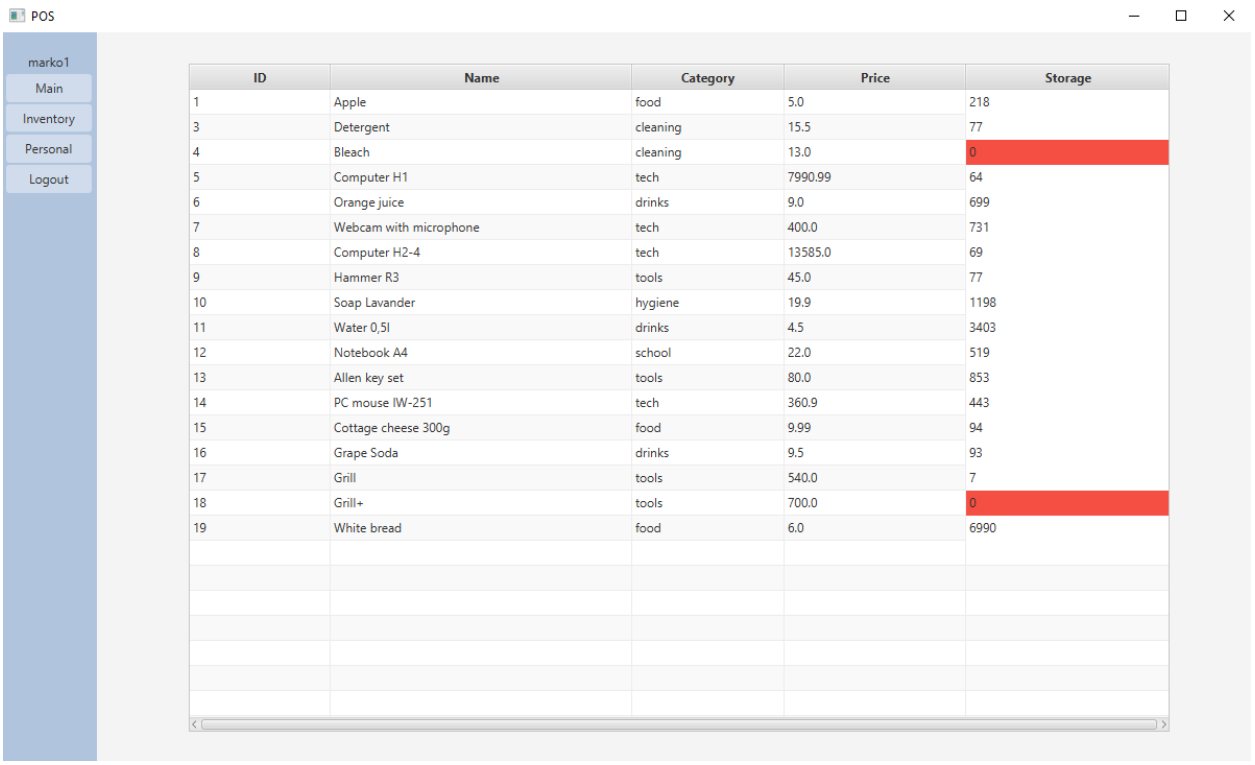
```
try {
    float invoicePrice = getInvoicePrice();
    float invoiceDiscount = invoice.getDiscount();
    File file = new File("d:/Projects/GuiPos/output","invoice-" + invoice.getId().toString() + "-output.txt");
    if(!file.exists()){
        file.createNewFile();
    }
    PrintWriter printer = new PrintWriter(file);
    printer.println("Store GuiPos");
    printer.println("Adresa: 5th Street 42, Old Town");
    printer.println();
    printer.println("Invoice ID: " + invoice.getId().toString());
    printer.println("Time of issue: " + invoice.getTime());
    printer.println();
    printer.println("Price: " + getVatApplied(invoicePrice));
    if(invoiceDiscount != 0) {
        float invoicePriceAfterDiscount = invoicePrice - ((invoiceDiscount/100)*invoicePrice);
        printer.printf("Discount: %.0f\n", invoiceDiscount);
        printer.printf("Price after discount: %.2f", getVatApplied(invoicePriceAfterDiscount));
        printer.println();
    }
    printer.println("Employee: " + user.getName() + " " + user.getSurname());
    printer.println();
    printer.println("List of items in invoice:");
    printer.println();
    printer.printf("%-30s %-10s %-10s %-10s %-10s\n", "Article", "Amount", "Price/Unit", "Price/Total", "Price with VAT");
    for (ArticleInInvoice itemInInvoice : articleInInvoice) {
        printer.printf("%-30s %-10d %-10.2f %-10.2f %-10.2f \n",
                itemInInvoice.getArticle(),
                itemInInvoice.getAmount(),
                itemInInvoice.getPrice()/itemInInvoice.getAmount(),
                itemInInvoice.getPrice(),
                getVatApplied(itemInInvoice.getPrice()));
    }
    printer.close();
} catch (Exception e) {
```

Slika 4.35. *Programski kod metode „finalizeInvoice()“ za ispis tekstualnog računa.*

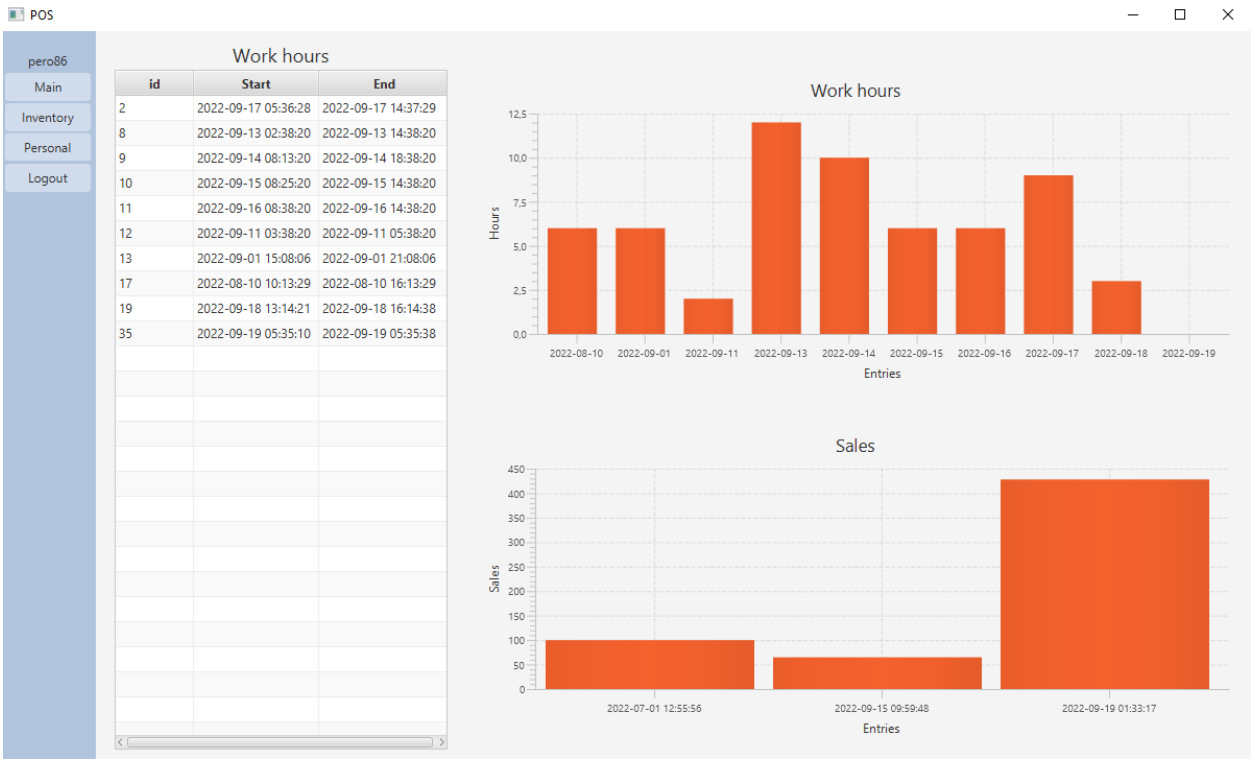
```
Store GuiPos
Adresa: 5th Street 42, Old Town

Invoice ID: 58
Time of issue: 2022-11-27 23:24:45

Price: 1071.88
Discount: 10
Price after discount: 964.69
Employee: Marko Markic

List of items in invoice:

Article                    Amount     Price/Unit Price/Total Price with VAT
Detergent                  1          15.50      15.50       19.38
Webcam with microphone     1          400.00     400.00      500.00
Water 0,5l                 1          4.50       4.50        5.63
Notebook A4                1          22.00      22.00       27.50
Webcam with microphone     1          400.00     400.00      500.00
Detergent                  1          15.50      15.50       19.38
```

Slika 4.36. *Izgled ispisanog računa.*

Pritiskom tipke *„Inventory“* u lijevom stupcu se otvara sučelje koje prikazuje stanje artikala u bazi podataka. Pritiskom *„Personal“* se otvara sučelje koje prikazuje unose radnih sati, graf radnih sati i graf prodaja prijavljenog zaposlenika. Na slici 4.37 je prikazano sučelje *„Inventory“* i na slici 4.38 je prikazano sučelje *„Personal“*. Oba sučelja dohvaćaju podatke istim metodama kao i u prijašnjim sučeljima za te podatke.



Slika 4.37. *Prikaz sučelja „Inventory“.*



Slika 4.38. *Prikaz sučelja „Personal“.*

5. KORIŠTENJE APLIKACIJE

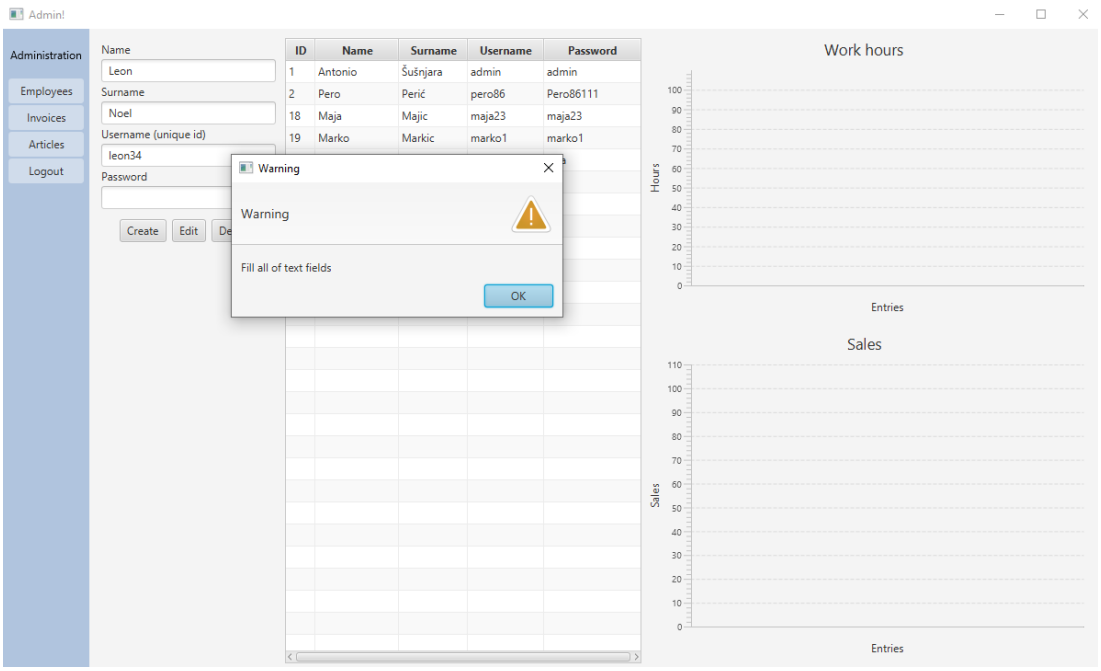
U ovom poglavlju bit će objašnjeno kako koristiti aplikaciju, koje su mogućnosti rada i koja su ograničenja.

5.1 Administrator

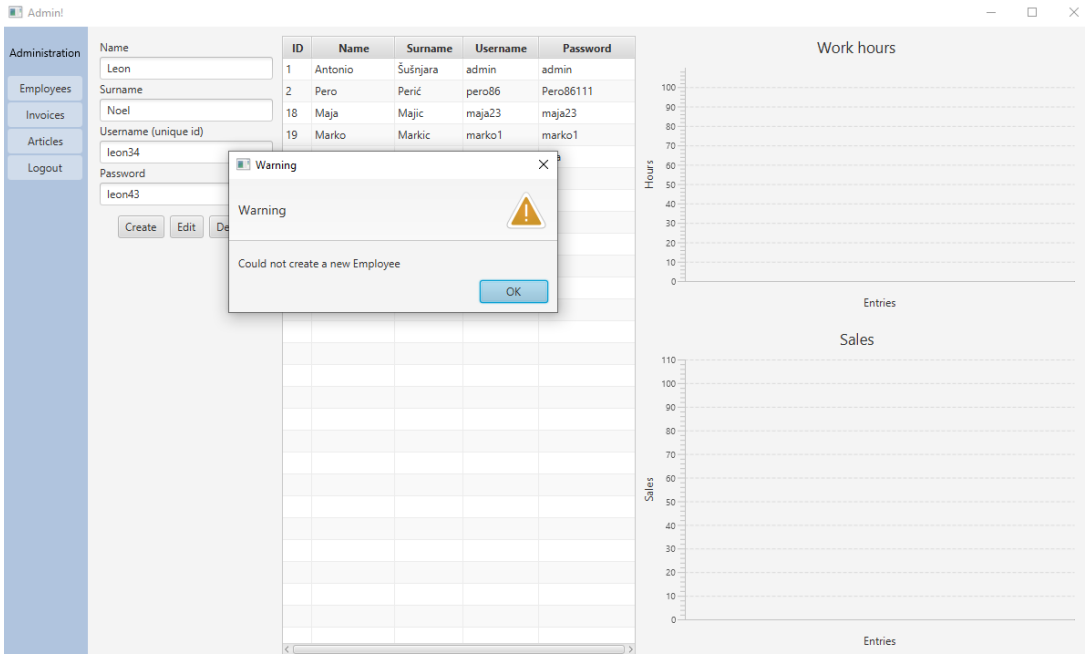
Administrator može upravljati zaposlenicima, računima i artiklima. Svaka kategorija ima svoje zasebno sučelje. Pritiskom tipke *„Logout“* administrator se odjavljuje i aplikacije otvara sučelje za prijavljivanje.

5.1.1 Upravljanje zaposlenicima

Administrator može unosit nove, uređivati i brisati postojeće zaposlenike. Unošenje novog zaposlenika se izvodi tako da se u formu upišu svi podaci i zatim pritisne tipka *„Create“*. Ako pri kreiranju zaposlenika neki podatak nije unesen aplikacija će prikazati upozorenje na slici 5.1. U slučaju greške na strani baze podataka će se prikazati upozorenje na slici 5.2.

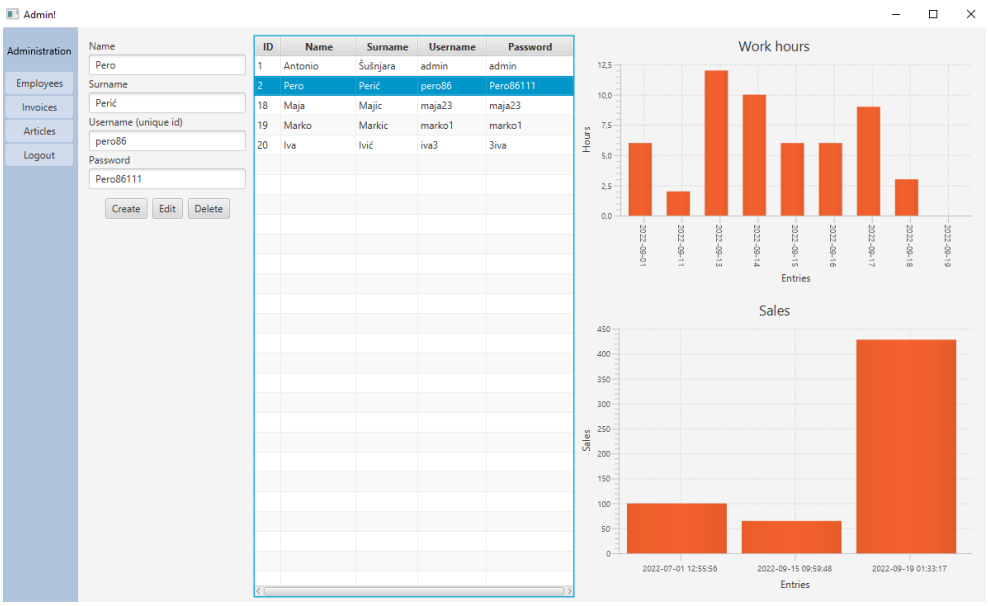


Slika 5.1. *Sučelje aplikacije pri unosu zaposlenika bez svih podataka.*



Slika 5.2. *Sučelje aplikacije nakon greške u bazi podataka.*

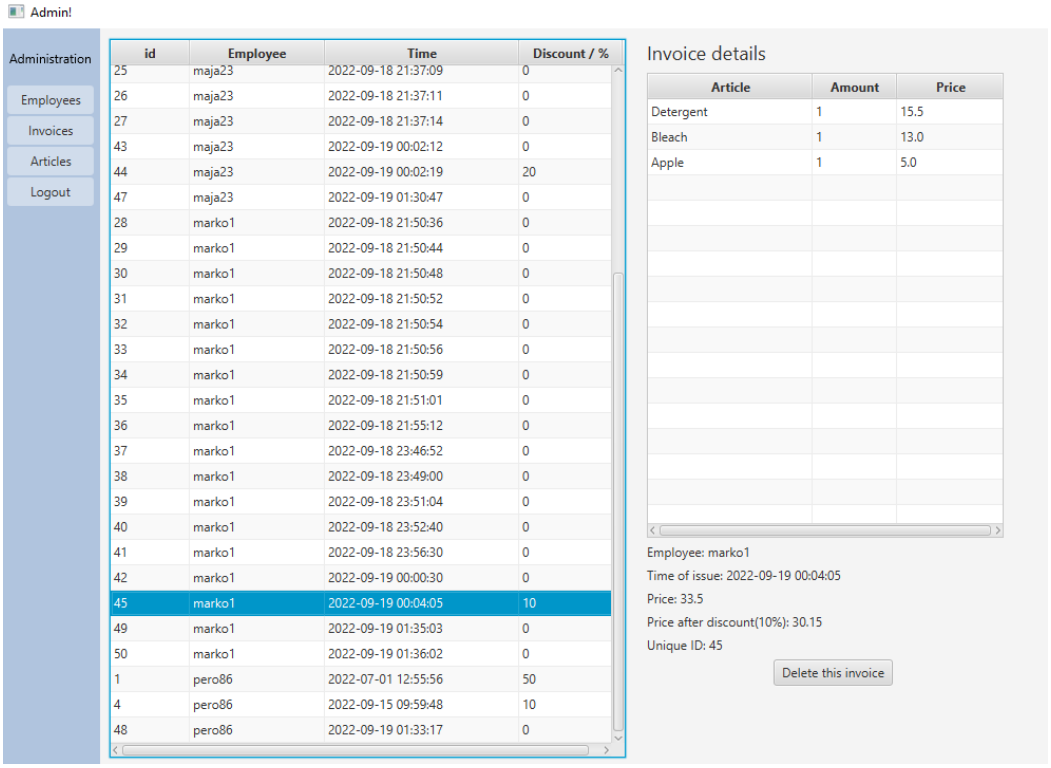
Dvostrukim klikom na redak željenog zaposlenika se ispuni forma i grafovi s podacima tog zaposlenika. Administrator može dvostrukim klikom ispuniti formu i zatim promijeniti željenu vrijednost pritiskom tipke *„Edit“*. Uređivanje zaposlenika ima istu provjeru kao i kreiranje za unesene podatke. Označavanjem zaposlenika u tablici i pritiskom tipke *„Delete“* se obriše zaposlenik. Zaposlenik se ne može obrisati ako u bazi podataka još postoje računi u kojima je taj zaposlenik naveden, jer bi to dovelo do ne planiranog gubitka računa. Na slici 5.3 je prikazano sučelje s označenim zaposlenikom i popunjenim podacima.



Slika 5.3. *Sučelje aplikacije popunjeno podacima odabranog zaposlenika.*

5.1.2 Upravljanje računima

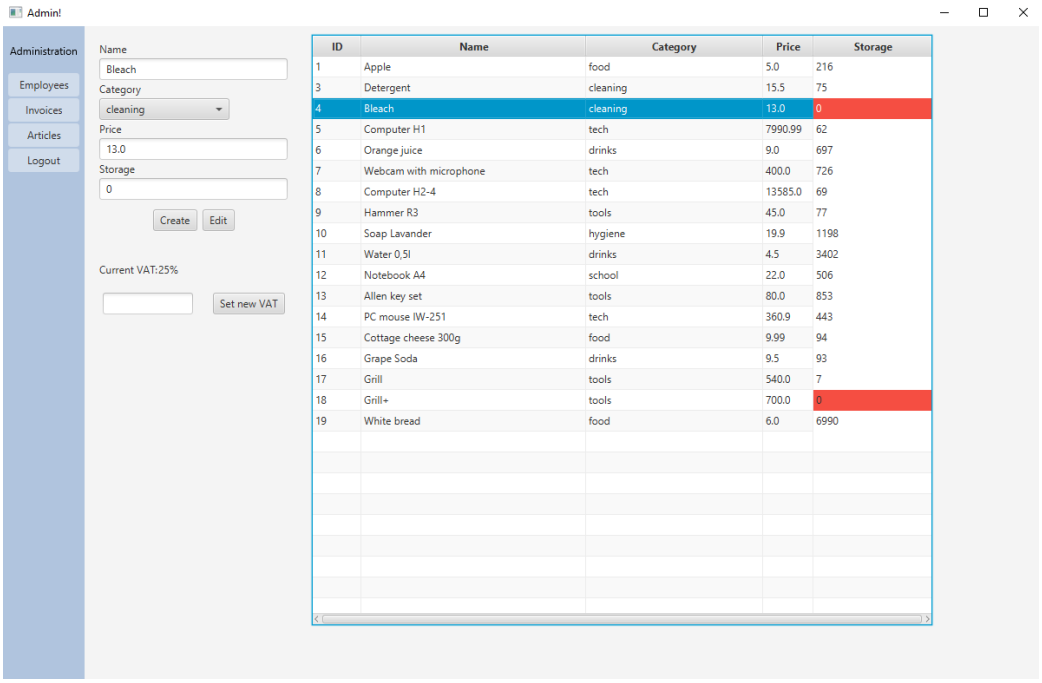
Administrator može vidjeti sve račune i njihov sadržaj, uz to može obrisati račun. Računi su prikazani u tablici gdje administrator može dvostrukim klikom na željeni račun popuniti tablicu sadržaja i detalje tog računa. Tipkom *„Delete this invoice“* može obrisati račun koji je označen u tablici. Na slici 5.4 je prikaz sučelja za upravljanje računima popunjeno podacima računa.



Slika 5.4. *Sučelje aplikacije popunjeno podacima odabranog računa.*

5.1.3 Upravljanje artiklima

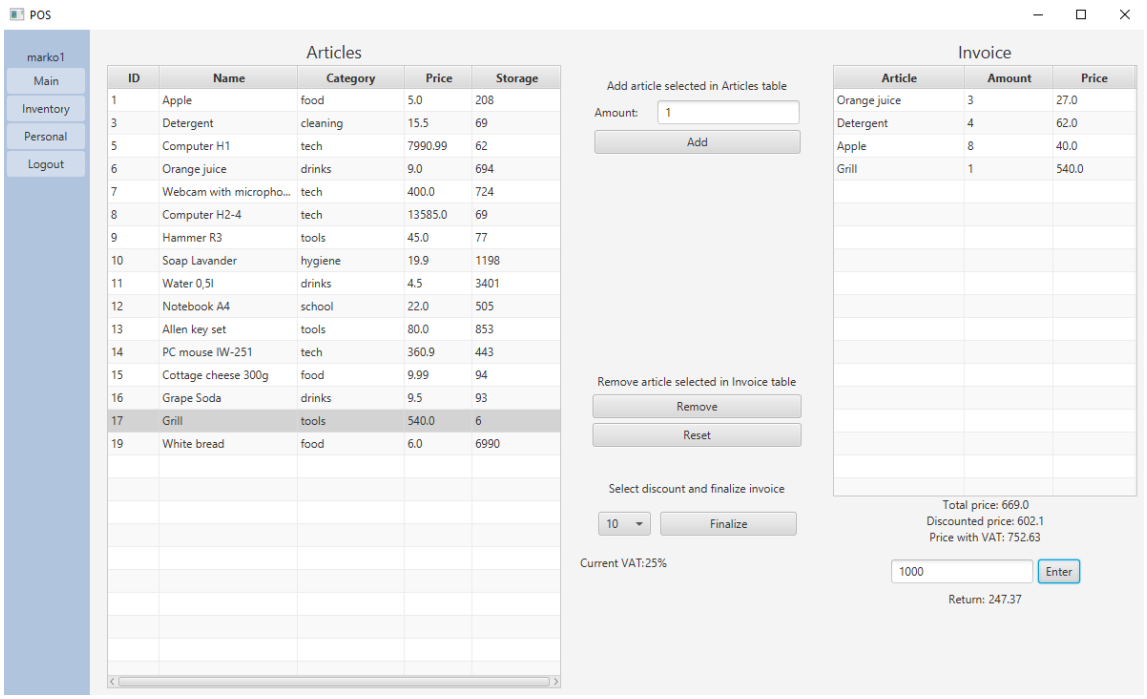
Administrator može vidjeti sve artikle i njihovo stanje u bazi podataka. Može unositi nove i uređivati postojeće artikle. Dvostrukim klikom na artikl u tablici se popuni forma gdje se onda može promijeniti podatak i pritiskom tipke *„Edit“* izvršiti promjena. Unošenjem svih traženih podataka i pritiskom tipke *„Create“* se mogu unijeti novi proizvodi. Artikli se ne mogu brisati iz aplikacije zbog postojećih računa u kojima su sadržani što bi dovelo do promjene izdanih računa. Pritiskom tipke „*Set new VAT*“ se ažurira vrijednost koja je spremljena u bazi podataka. Na slici 5.5 je prikaz sučelja za upravljanje artiklima popunjeno podacima odabranog artikla.



Slika 5.5. *Sučelje aplikacije popunjeno podacima odabranog artikla.*

5.2 Zaposlenik

Zaposlenik može izdavati račune, vidjeti stanje artikala i vidjeti osobne podatke rada i prodaje. Pritiskom tipke *„Logout“* zaposlenik se odjavljuje i s time unosi novi element radnih sati u bazu podatak u rasponu vremena prijavljivanja i odjavljivanja. Pritiskom tipke *„Inventory“* zaposlenik može vidjeti opis i stanje artikala. Pritiskom tipke *„Personal“* zaposlenik može vidjeti osobne unose radnih sati i grafove radnih sati i prodaje. Pritiskom tipke *„Main“* se otvara glavno sučelje za rad zaposlenika gdje može izdavati račune. Zaposlenik klikom na artikl odabere koji artikl želi dodati u račun, zatim unosom broja u *„Amount“* podatak odrediti količinu za dodati u račun. Zaposlenik može klikom odabrati artikl u tablici računa i pritiskom tipke *„Remove“* maknuti taj artikl iz računa. Pritiskom tipke *„Reset“* se tablica račun isprazni i sučelje se vrati na prvotno stanje. Pritiskom tipke „*Enter“* nakon unosa primljenog novca ispisati će se količina koju je potrebno vratiti. Pored tipke *„Finalize“* zaposlenik može odabrati popust za primijeniti na račun, odabiri mogu biti 0%, 5%, 10% ili 20%. Na slici 5.6 je prikaz sučelja tokom unošenja artikala u račun. Pritiskom tipke *„Finalize“* se kreira novi račun ako ima barem jedan artikl naveden u računu, nakon toga se kreira tekstualna datoteka koja u imenu sadrži unikatni *„id“* računa i sve potrebne detalje. Ispisani računa sadrži cijene s porezom na dodanu vrijednost. Na slici 5.7 je prikaz računa u tekstualnoj datoteci.



Slika 5.6. *Sučelje aplikacije pri kreiranju računa.*

```
Store GuiPos
Adresa: 5th Street 42, Old Town

Invoice ID: 59
Time of issue: 2022-11-28 00:10:10

Price: 836.25
Discount: 10
Price after discount: 752.63
Employee: Marko Markic

List of items in invoice:

Article                 Amount     Price/Unit Price/Total Price with VAT
Orange juice            3          9.00       27.00       33.75
Detergent               4          15.50      62.00       77.50
Apple                   8          5.00       40.00       50.00
Grill                   1          540.00     540.00      675.00
```

Slika 5.7. *Prikaz ispisanog računa.*

6. ZAKLJUČAK

Cilj ovog završnog rada bio je izrada aplikacije kao programsko rješenje za blagajnu u maloprodajnoj trgovini. Aplikacija izrađena u sklopu ovog završnog rada omogućava jednostavan rad administratoru sustava i zaposleniku trgovine, no uz manjak visoke razine sigurnosti i velikog broja značajki koje su jako bitne za sustav upravljanja trgovinom i novcem. Aplikacija je napravljena u JavaFX okviru što je omogućilo korištenje Java programskog jezika i mnoge dodatke koji su korisno za kreiranje aplikacije s grafičkim sučeljem i povezivanje s MySQL bazom podataka. MVC arhitektura je jako korisna za lakše shvaćanje povezanosti komponenti i organizaciju koda što je bila jako velika prednost za pisanje aplikacije. Pogledi (engl. *Views*) za aplikaciju su pravljeni u SceneBuilderu koji je nakon teškog vremena učenja postao jako lagan i intuitivan alat za koristit. SQL upiti i dizajniranje relacija za bazu podataka je zahtijevalo puno planiranja unaprijed što je prouzročilo otežano snalaženje po povezanosti tablica u bazi podataka.

Aplikacija omogućava jednostavan način kreiranja računa s odabranim artiklima i količinama uz spremljeni račun u tekstualnoj datoteci i u bazi podataka, uz to omogućava kreiranje profila zaposlenika i praćenje njihovog rada i prodaje.

Moguća poboljšanja aplikacije je bolji prikaz i korisnost uvida u radno vrijeme zaposlenika trgovine. Nova funkcionalnost za administratora koja prikazuje radno vrijeme u određenom vremenskom rasponu. Uz to moguće poboljšanje je korištenje dvostrukog cijelog broja (engl. *integer*) za prikaz cijene gdje prvi broj označava cjelobrojnu vrijednost cijene i drugi označava decimalni dio, zato što postoji nepreciznost pri zaokruživanju decimalnog broja s pomičnom točkom (engl. *float*).

LITERATURA

[1] »Upute-online-aplikacija,« https://fiskalna.hr/dokumenti/Upute-online-aplikacija.pdf [7.7.2021.]

[2] »Bar, Brewery Software - Loyverse POS System,« https://loyverse.com/bar-pos [7.7.2021.]

[3] »eHopper POS Features,« https://ehopper.com/product-features [7.7.2021.]

[4] »What is Java? Definition, Meaning & Features of Java Platforms,« https://www.guru99.com/java-platform.html [7.7.2021.]

[5] »Java OOP,« https://www.w3schools.com/java/java_oop.asp [7.7.2021.]

[6] »Introduction to Java Virtual Machine (JVM) – SUNIXI,« https://sunixi.com/introduction-to-java-virtual-machine-jvm [7.7.2021.]

[7] »JavaFX vs Swing: The Key Differences | Career Karma,« https://careerkarma.com/blog/javafx-vs-java-swing [7.7.2021.]

[8] »IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains,« https://www.jetbrains.com/idea [7.7.2021.]

[9] »Scene Builder - Gluon,« https://gluonhq.com/products/scene-builder [7.7.2021.]

[10] »What Is SQL And How Does it Work?,« https://in.springboard.com/blog/what-is-sql-and-how-does-it-workn [7.7.2021.]

SAŽETAK

Tema: Blagajna za maloprodajnu trgovinu

GUI aplikacija za korištenje računala na blagajni maloprodajne trgovine. Aplikacija omogućava rad zaposlenika za prodaju artikala. Aplikacija omogućava administratoru sustava upravljanje artiklima i praćenje rada zaposlenika kroz jednostavni prikaz podataka koji su spremljeni u bazi podataka. Aplikacija je napravljena u Java programskom jeziku kako bi se omogućilo korištenje na različitim računalima neovisno o arhitekturi.

Ključne riječi: GUI, JavaFX, MySQL, trgovina, prodaja

ABSTRACT

Title: Retail store cash register

GUI application for use on a cash register computer in a retail store. Application enables employees to sell items. The application enables the system administrator to manage items and monitor the work of employees through a simple display of data from a database. The application is made in the Java programming language to enable use on different computers regardless of architecture.

Ključne riječi: GUI, JavaFX, MySQL, sale, store