**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Чорноморський національний університет імені Петра Могили**

**Факультет комп’ютерних наук**

**Кафедра інженерії програмного забезпечення**

**КУРСОВА РОБОТА**

з дисципліни «Об’єктно - орієнтоване програмування»

на тему: **Автоматизована система  «Аптека»**

122 – КР.ПЗ.00 – 202.1910211

**Виконав: студент 2 курсу, групи 202**

**спеціальності 122 «Комп'ютерні науки»**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.Ю.Карпенко

(підпис, ініціали та прізвище)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата)

**Керівник: ст. викладач**

**Швед А.В.**

Національна   шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_ \_\_\_\_\_\_\_\_

**м. Миколаїв – 2021 рік**

Оглавление

[ВСТУП 2](#_Toc72201194)

[Розділ 1 Розробка користувацького інтерфейсу 4](#_Toc72201195)

[1.1 Вікна авторизації 4](#_Toc72201196)

[1.2 Головні вікна застосунку 7](#_Toc72201197)

[Розділ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ 8](#_Toc72201198)

[2.1 Основні моделі представлення 8](#_Toc72201199)

[2.2 Реєстрація користувача 9](#_Toc72201200)

[2.3 Зберігання даних користувача 9](#_Toc72201201)

[2.4 Створення замовлення 10](#_Toc72201202)

[2.5 Зберігання даних 13](#_Toc72201203)

[2.6 Авторизація 16](#_Toc72201204)

[2.7 Успішна авторизація 17](#_Toc72201205)

[2.8 Додавання товарів 18](#_Toc72201206)

[2.9 Додавання проекту на GitHub 20](#_Toc72201207)

[Для того, щоб зручно переглянути проект, його було завантажено на GitHub за посиланням(“https://github.com/TENOVOD/CourseWork”): 20](#_Toc72201208)

[Розділ 3 Тестування застосунку 21](#_Toc72201209)

[3.1 Авторизація 21](#_Toc72201210)

[3.2 Головні вкладки 25](#_Toc72201211)

[висновки 30](#_Toc72201212)

[перелік джерел посилання 31](#_Toc72201213)

# 

# ВСТУП

Головними функціями будь якої програми є гармонійне, зручне та ефективне середовище. В даній курсовій роботі розглядається приклад створення автоматизованої програми, а саме комп’ютерного застосунку. Програма надає можливість реєструвати нових користувачів, додавати та замовляти медикаменти. Для програмної реалізації використовується JavaFX, це безкоштовна графічна платформа на базі мови програмування Java, для розробки програм з якісним, багатогранним графічним інтерфейсом. Можна використовувати на ОС , інтернет-додатках та гаджетах. Розроблено для того, щоб замінити бібліотеку Swing.

Завданням курсової роботи є створення якісної програми, основною метою якою є автоматичний продаж продукції з інтерактивним інтерфейсом, що надасть змогу самостійно користуватись застосунком.

Мета роботи – побудова програми з можливістю збереження даних в БД та автоматизація продажу продукції.

Для досягнення поставленої мети необхідно виконати наступні завдання:

* Обрання мови програмування для розробки застосунку
* Обрати бібліотеку, яка надасть можливість реалізувати всі функції автоматизованого застосунку
* Пошук необхідної літератури
* Пошук та підбір кольорів для візуального та практичного комфорту користувача
* Розробка функціоналу , моделей та БД
* Розробка графічного інтерфейсу
* Тестування

Курсова робота складається з 31 сторінок, містить 28 ілюстрацій, 10 інтернет і літературних джерел.

# Розділ 1 Розробка користувацького інтерфейсу

Користувацький інтерфейс – це область, яка включає в себе функціональність, зручність, ефективність та візуальну складову. В добре розробленому інтерфейсі повинно бути гармонійно поєднано візуальну, функціональну та програмну складову. Для побудови якісного дизайну необхідно проаналізувати інтерфейси популярних та зручних програм, та підкреслити основні положення і від цього відштовхуватись.

Для розробки автоматизованої системи «Аптека», я керувався стилем графічного дизайну Material Design 2. Реалізував за допомогою Scene Builder, дане програма допомогла пришвидшити розробку файлів інтерфейсів fxml. Програмне забезпечення є безкоштовним та належить Oracle Corporation.

## Вікна авторизації

Для того, щоб розділити інформацію за рівнем доступу в базах даних необхідно створити окремі сторінки авторизації. Інтерфейс було розроблено в Scene Builder

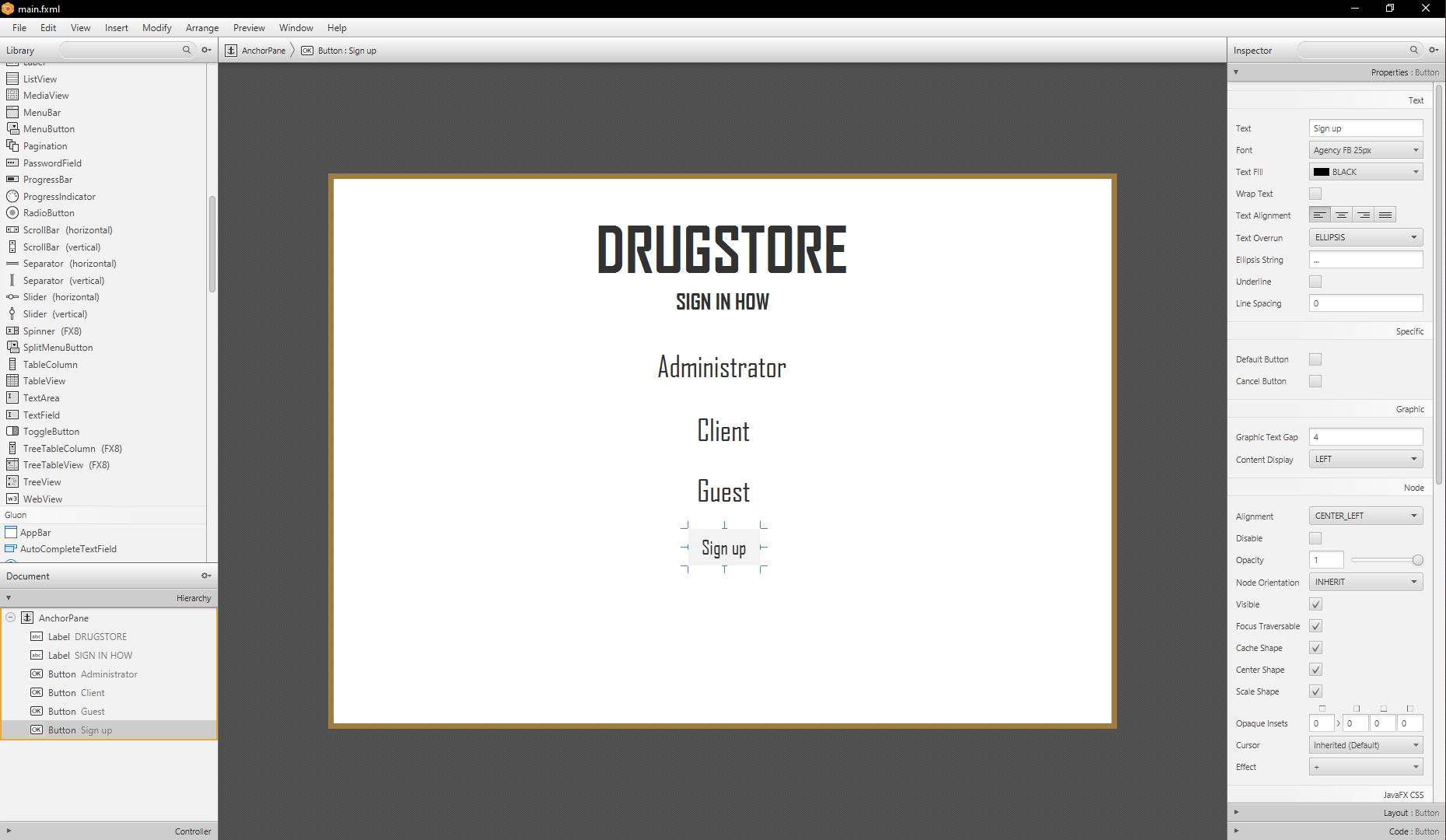


Рисунок 1.1 – Головне вікно

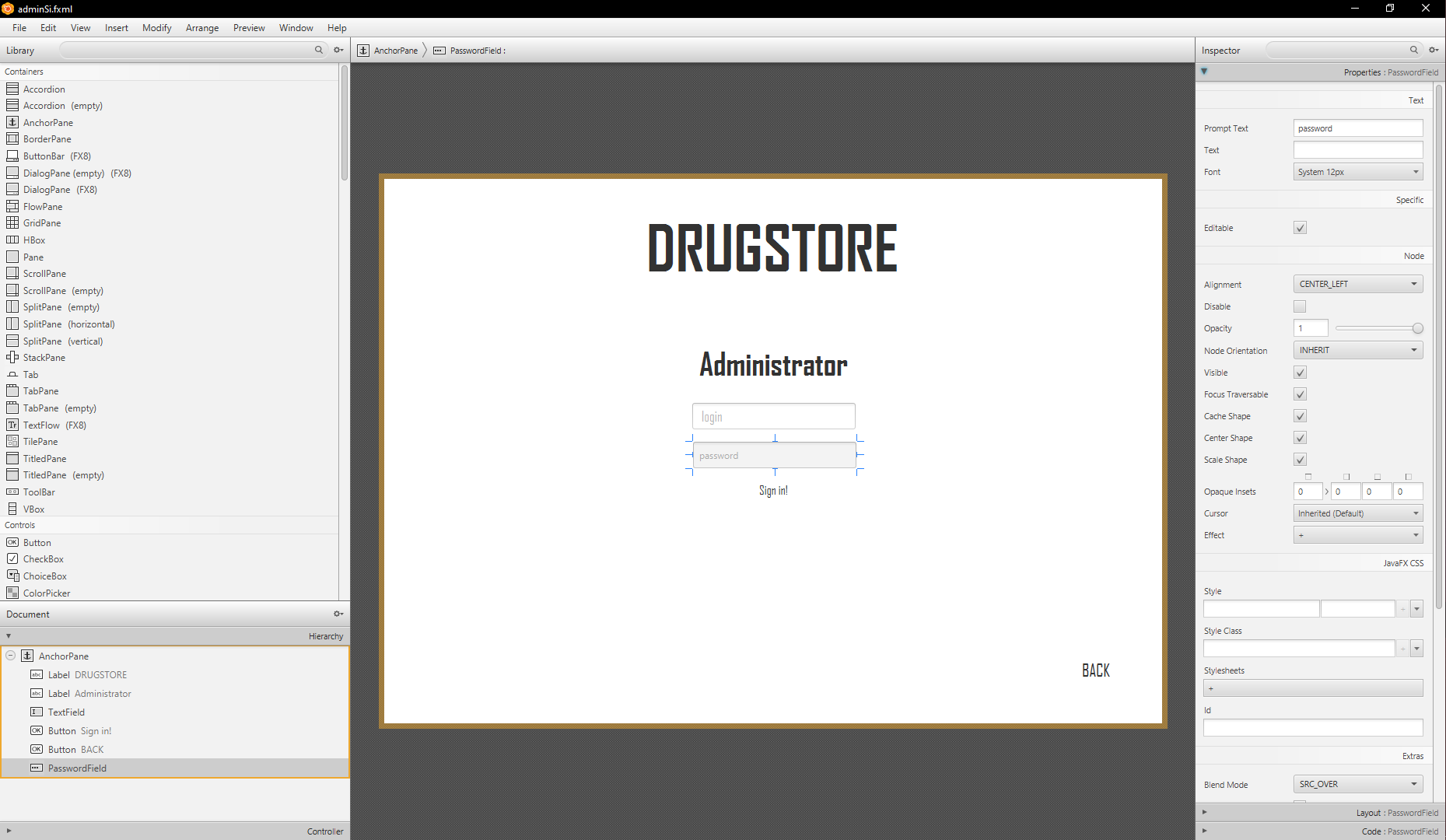


Рисунок 1.2 – Вікно авторизації адміністратора

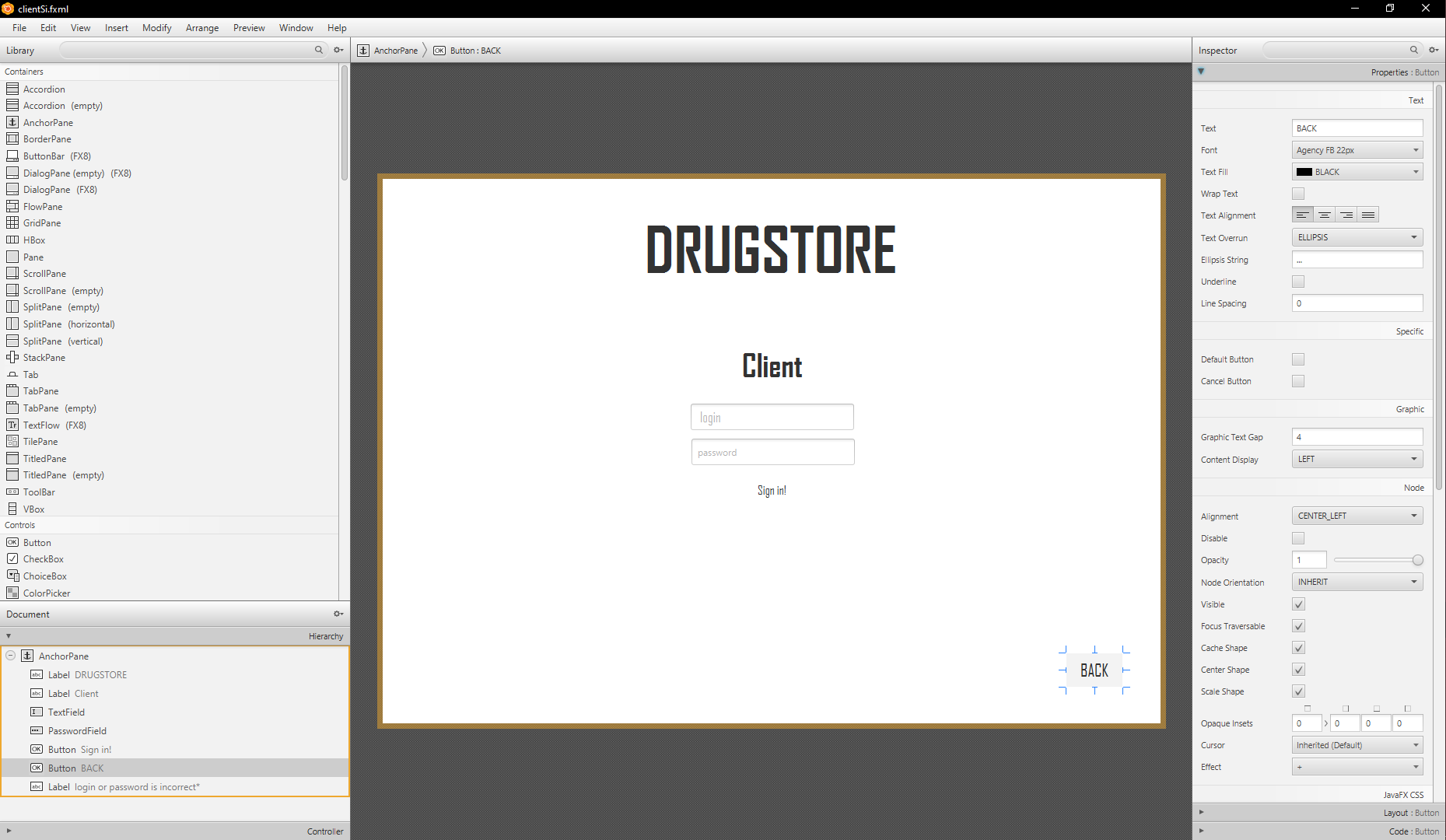


Рисунок 1.3 – Вікно авторизації клієнта

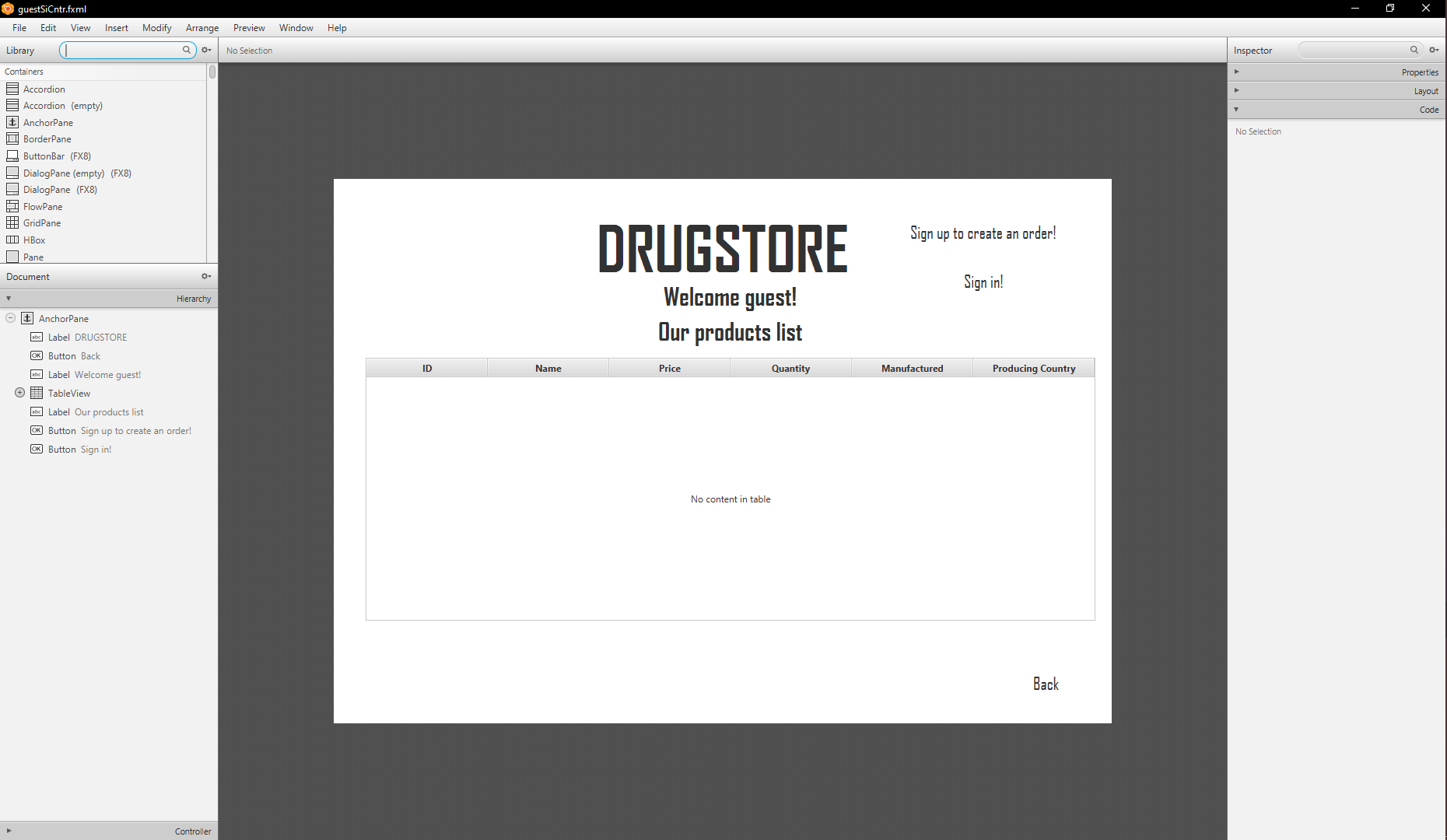


Рисунок 1.4 – Вікно гостя

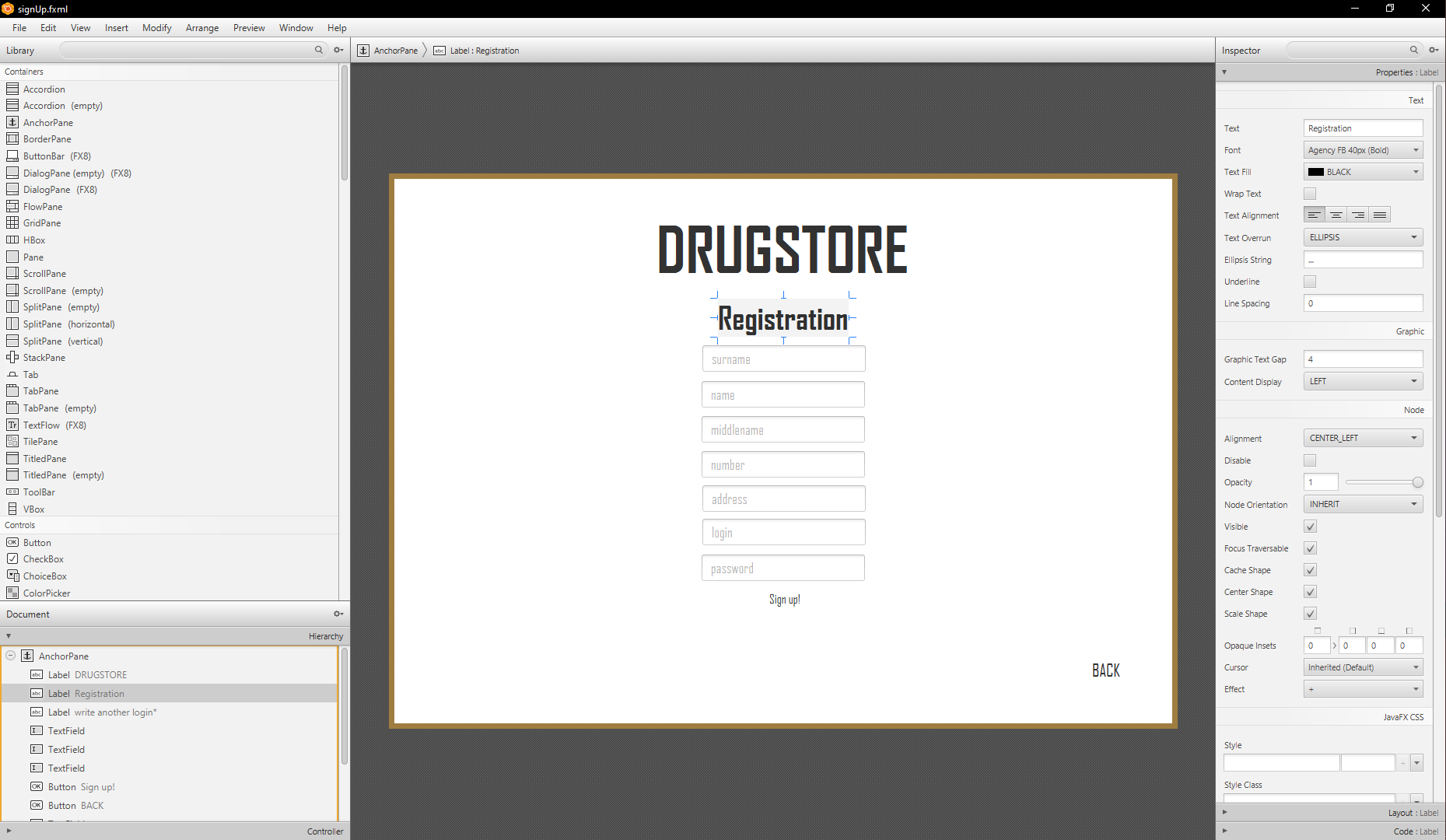


Рисунок 1.5 – Вікно реєстрації

## 

## Головні вікна застосунку

Для зручного моніторингу наявності товарів та замовлень було

використано TableView з бібліотеки JavaFx .

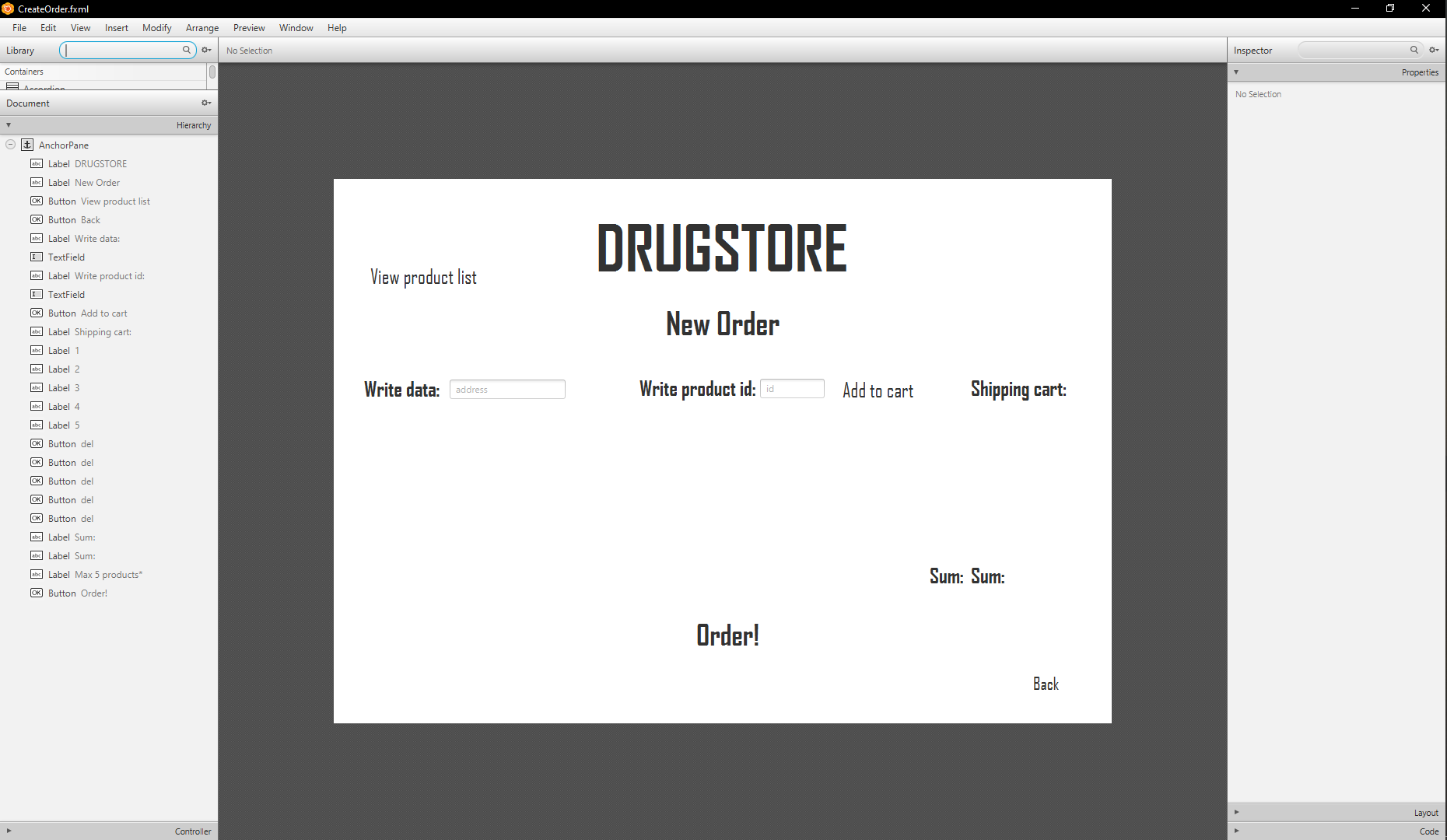


Рисунок 1.6 –Замовлення продукції

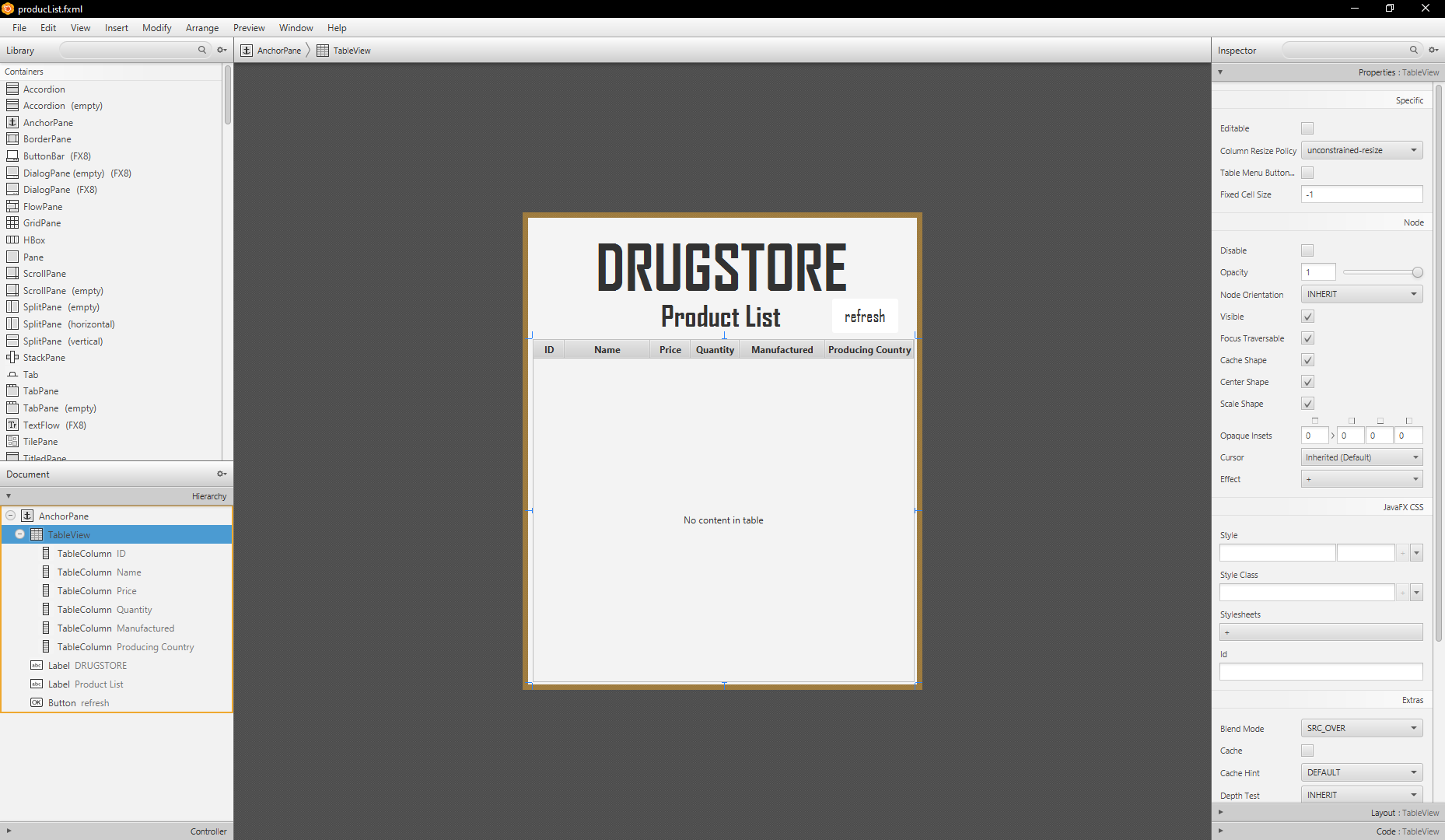


Рисунок 1.7 – Список продукції аптеки

# Розділ 2 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОЄКТУ

Програмне реалізація виконана у IntelliJ IDEA з використанням SQLiteStudio, SDK-16 та JDBS-3.34.0 . Все було завантажено з репозиторія Maven.

## 2.1 Основні моделі представлення

Клас Client містить поля необхідні для створення, авторизації аккаунту та зчитування з бази даних.

public class Client {  
 private int id=-1;  
 private String surname;  
 private String name;  
 private String middleName;  
 private int phoneNumber;  
 private String address;  
 private String login;  
 private String password;

Клас Order містить поля необхідні для створення замовлення, так як поля приватні, додатково було створено getters and setters.

public class Order {  
 private String date;  
 private int number;  
 private double sum;  
 private String nameProducts;  
 private String address;  
 private int idClient;

Клас Production містить поля необхідні для додавання товару до замовлення, додаваня нового товару, так як поля приватні, додатково було створено getters and setters.

public class Production {  
 private int num;  
 private String name;  
 private double price;  
 private int quantity;  
 private String manufactured;  
 private String producingCountry;

## 2.2 Реєстрація користувача

Реєстрація клієнта в графічному додатку, відбувається за допомогою метода :

public void registration() {  
 signUpRBtn.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 List<Client> clients=new ArrayList<>(cldao\_client.find(loginRFX.getText(),passwordRFX.getText()));  
 if(clients.size()>0) {  
 loginRFX.clear();  
 passwordRFX.clear();  
 writeAnothLogFX.setVisible(true);  
 }  
 else {  
 writeAnothLogFX.setVisible(false);  
 Client client=new Client(nameRFX.getText(),surnameRFX.getText(),middleRFX.getText(),Integer.*parseInt*(numberRFX.getText()),addressRFX.getText(),loginRFX.getText(),passwordRFX.getText());  
 cldao\_client.create(client);  
 clearField();  
 }  
 }  
 });  
}

## 2.3 Зберігання даних користувача

Для того щоб зберегти дані користувача у базі даних, було створено класс CLDAO\_Client, який зберігає його за допомогою метода create:

@Override  
public Client create(Client entity) {  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "INSERT INTO Clients (Surname,Name,MiddleName,Number,Address,Login,Password) VALUES (?,?,?,?,?,?,?);";  
 PreparedStatement preparedStatement = connect.prepareStatement(sql);  
 preparedStatement.setString(1, entity.getSurname());  
 preparedStatement.setString(2, entity.getName());  
 preparedStatement.setString(3, entity.getMiddleName());  
 preparedStatement.setInt(4, entity.getPhoneNumber());  
 preparedStatement.setString(5, entity.getAddress());  
 preparedStatement.setString(6, entity.getLogin());  
 preparedStatement.setString(7, entity.getPassword());  
 preparedStatement.execute();  
 ResultSet rs = preparedStatement.getGeneratedKeys();  
 rs.next();  
 int key = rs.getInt(1);  
 entity.setId(key);  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return entity;  
}

## 2.4 Створення замовлення

Клас CLDAO\_Order в якому містяться такі методи, як: create(відповідає за створення):

public Order create(Order entity) {  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "INSERT INTO orders (date,sum,nameProducts,address,idClient) VALUES ( ?,?,?,?,?)";  
 PreparedStatement preparedStatement = connect.prepareStatement(sql);  
 preparedStatement.setString(1, entity.getDate());  
 preparedStatement.setDouble(2, entity.getSum());  
 preparedStatement.setString(3, entity.getNameProducts());  
 preparedStatement.setString(4, entity.getAddress());  
 preparedStatement.setInt(5, entity.getIdClient());  
 preparedStatement.execute();  
 ResultSet rs = preparedStatement.getGeneratedKeys();  
 rs.next();  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return entity;  
}

Для того щоб клієнт мав змогу створити замовлення був створений спеціальний метод:

public void newOrder() {  
 CLDAO\_Production cldao\_production = new CLDAO\_Production();  
 ArrayList<Production> productions = new ArrayList<>(cldao\_production.findAll());  
 viewProductList1.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 int checkId = Integer.*parseInt*(id.getText());  
 Production production = new Production();  
 for (int i = 0; i < productions.size(); i++) {  
 production = productions.get(i);   
 if (production.getNum() == checkId){ //перевіряємо чи дорівнює введене id товару до товару на складі, якщо так, то виконується наступне   
 if (counter < 5) {  
 counter++; //лічильник +1 так як додали товар до кошика  
 SUM += production.getPrice(); //до загальної суми додаємо вартість обраного товару  
 production.setQuantity(production.getQuantity()-1); //забираємо 1 товар зі складу до кошика  
 cldao\_production.update(production);   
 sum.setText(String.*valueOf*(SUM));  
 maxProducts.setVisible(false);  
 //якщо це перший товар в кошику, то виконається 1 кейс і т.д.  
 switch (counter) {  
 case 1: {  
 prod1.setVisible(true); //напис стає видимим  
 delProd1.setVisible(true); //кнопка стає видима  
 prod1.setText(production.getName()); //на напис встановлюється назва продукту який було додано  
 first = production.getPrice();   
 num1=production.getNum();  
 }  
 break;  
 case 2: {  
 prod2.setVisible(true);  
 prod2.setText(production.getName());  
 delProd2.setVisible(true);  
 second = production.getPrice();  
 num2=production.getNum();  
 }  
 break;  
 case 3: {  
 prod3.setVisible(true);  
 delProd3.setVisible(true);  
 prod3.setText(production.getName());  
 third = production.getPrice();  
 num3=production.getNum();  
 }  
 break;  
 case 4: {  
 prod4.setVisible(true);  
 delProd4.setVisible(true);  
 prod4.setText(production.getName());  
 forth = production.getPrice();  
 num4=production.getNum();  
 }  
 break;  
 case 5: {  
 prod5.setVisible(true);  
 delProd5.setVisible(true);  
 prod5.setText(production.getName());  
 fifth = production.getPrice();  
 num5=production.getNum();  
 }  
 break;  
 }  
 }else {  
 maxProducts.setVisible(true);  
 }  
 }  
 }  
 }  
  
 });  
 id.clear();  
 //при натискані на певну кнопку видаляється назва товару та мінусується ціна  
 delProd1.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 prod1.setVisible(false);  
 SUM -= first; //від всієї суми мінусується вартість першого товару  
 delProd1.setVisible(false);  
 sum.setText(String.*valueOf*(SUM));  
 counter--; //лічильник кількості товарів -1  
 production=cldao\_production.find(num1).get(0);//у об'єкт додається товар, який ми видаляємо  
 production.setQuantity((production.getQuantity()+1));//та додаємо до кількості товару +1 так як ми видяляємо його з кошика  
 cldao\_production.update(production);//оновлюємо кількість продукції  
 }  
 });  
 delProd2.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 prod2.setVisible(false);  
 SUM -= second;  
 delProd2.setVisible(false);  
 sum.setText(String.*valueOf*(SUM));  
 counter--;  
 production=cldao\_production.find(num2).get(0);  
 production.setQuantity((production.getQuantity()+1));  
 cldao\_production.update(production);  
 }  
 });  
 delProd3.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 prod3.setVisible(false);  
 SUM -= third;  
 delProd3.setVisible(false);  
 sum.setText(String.*valueOf*(SUM));  
 counter--;  
 production=cldao\_production.find(num3).get(0);  
 production.setQuantity((production.getQuantity()+1));  
 cldao\_production.update(production);  
 }  
 });  
 delProd4.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 prod4.setVisible(false);  
 SUM -= forth;  
 delProd4.setVisible(false);  
 sum.setText(String.*valueOf*(SUM));  
 counter--;  
 production=cldao\_production.find(num4).get(0);  
 production.setQuantity((production.getQuantity()+1));  
 cldao\_production.update(production);  
 }  
 });  
 delProd5.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 prod5.setVisible(false);  
 SUM -= fifth;  
 delProd5.setVisible(false);  
 sum.setText(String.*valueOf*(SUM));  
 counter--;  
 production=cldao\_production.find(num5).get(0);  
 production.setQuantity((production.getQuantity()+1));  
 cldao\_production.update(production);  
 }  
 });  
 orderBtn.setOnAction(new EventHandler<ActionEvent>() {  
 @Override  
 public void handle(ActionEvent event) {  
 CLDAO\_Order cldao\_order=new CLDAO\_Order();  
 CLDAO\_Id cldao\_id=new CLDAO\_Id();  
 ArrayList<SaveID> saveIDS=new ArrayList<>(cldao\_id.find());  
 String time = new SimpleDateFormat("yyyy.MM.dd\_HH:mm").format(Calendar.*getInstance*().getTime());  
 String prodName=(prod1.getText()+" "+prod2.getText()+" "+prod3.getText()+" "+prod4.getText()+" "+prod5.getText());  
 Order order=new Order(time,SUM,prodName,address.getText(),saveIDS.get(0).getId());  
 cldao\_order.create(order);  
 clear();  
 SUM=0;  
 sum.setText(String.*valueOf*(SUM));  
 }  
 });  
}

## 2.5 Зберігання даних

Для роботи з базою даних було завантажено SQLiteStudio , дана програма містить всі типи даних, які знадобились, та функціонал, яких повністю підходить да даного проекту:

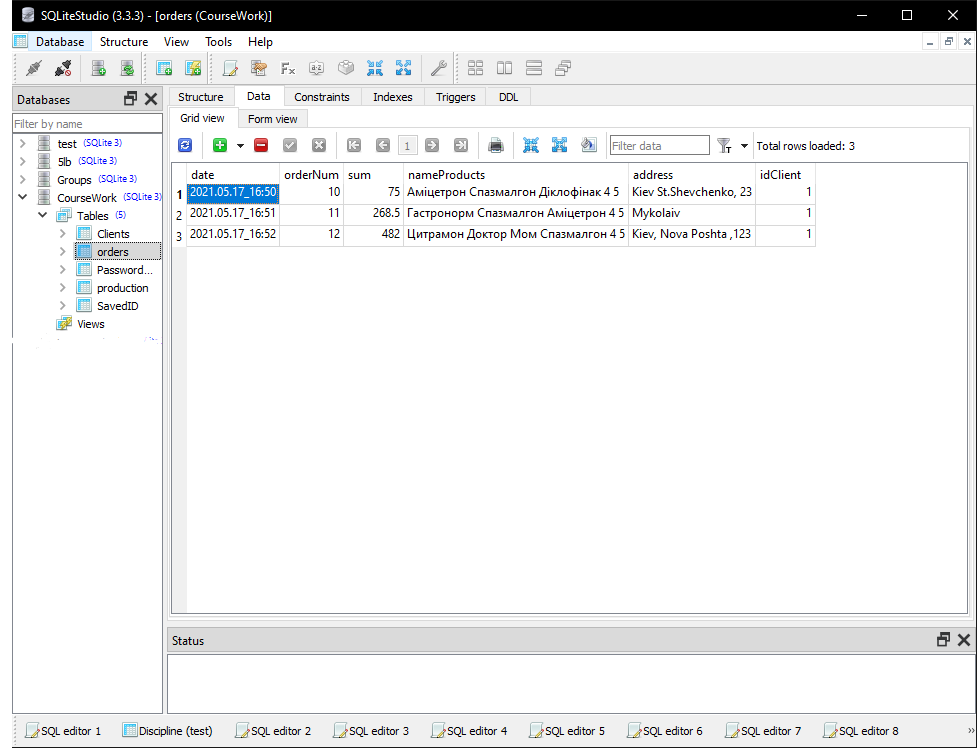


Рисунок 2.1 – База даних. Вікно даних таблиці orders(замовлення)

Таблиця клієнтів, можна побачити, що всі логіни різні, так як программа не дозволяє створювати користувачів з однаковими логінами:

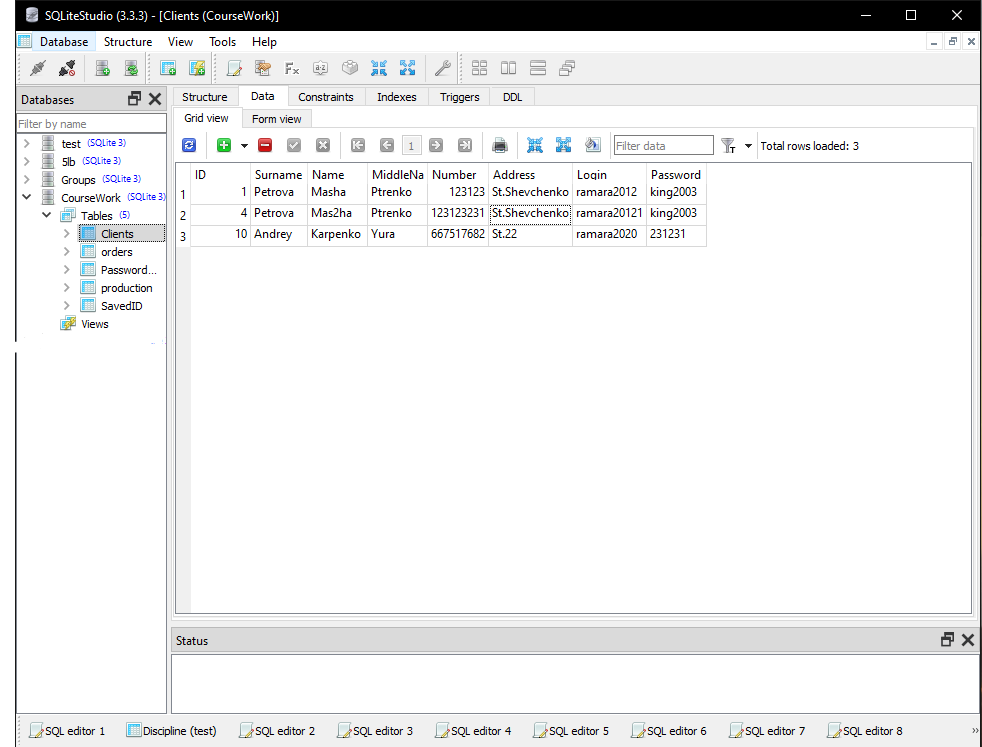


Рисунок 2.2 – База даних. Вікно даних таблиці Clients (клієнти)

Для того, щоб зберігати продукцію було створено таблицю production з ключем num з автоінкрименцією:

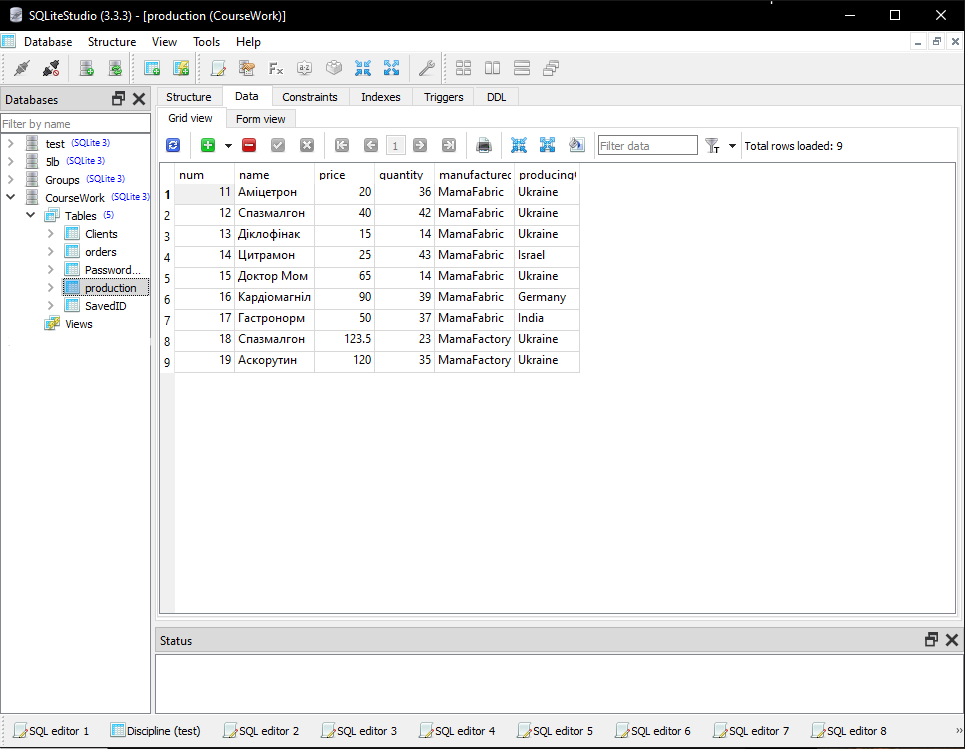


Рисунок 2.3 – База даних. Вікно даних таблиці production (продукція)

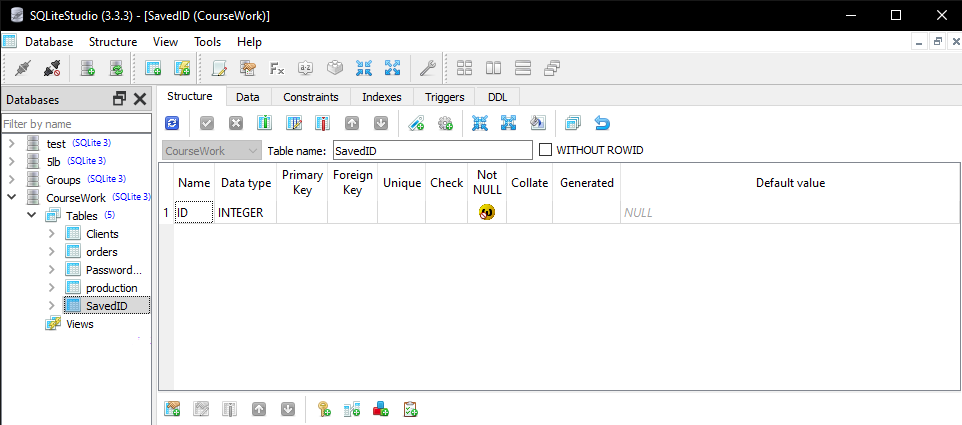


Рисунок 2.4 – База даних. Вікно структури таблиці SavedID

## 2.6 Авторизація

Для авторизації було спеціально створено 2 методи, авторизація адміністраторів:

public void adminSignIn() {  
 CLDAO\_Admin cldao\_admin=new CLDAO\_Admin();  
 ArrayList<Admin> admins=new ArrayList<>(cldao\_admin.find());  
 Admin checkAdmin=new Admin();  
 for(int i=0;i<admins.size();i++) {  
 checkAdmin=admins.get(i);  
 if(loginAdmTF.getText().equals(checkAdmin.getLogin())&&passAdmTF.getText().equals(checkAdmin.getPassword())) {  
 setNewScene("AdminPage.fxml");  
 }  
 else {  
 loginAdmTF.clear();  
 passAdmTF.clear();  
 }  
 }  
}

Aвторизація клієнтів:

public void signInBtn() {  
  
 ArrayList<Client> clients = new ArrayList<>(cldao\_client.findAll());  
 Client checkClient = new Client();  
 for (int i = 0; i < clients.size(); i++) {  
 checkClient = clients.get(i);  
 if (loginClientTF.getText().equals(checkClient.getLogin()) && passClientTF.getText().equals(checkClient.getPassword())) {  
 incorrectFx.setVisible(false);  
 CLDAO\_Id cldao\_id = new CLDAO\_Id();  
 SaveID saveID = new SaveID(checkClient.getId());  
 cldao\_id.delete();  
 cldao\_id.create(saveID);  
 setNewScene("ClientWelcomePage.fxml");  
 } else {  
 incorrectFx.setVisible(true);  
 }  
 }  
 loginClientTF.clear();  
 passClientTF.clear();  
}

У випадку з клієнтами, було додатково створено класс SaveID та певну таблицю в БД. При успішній авторизації ID клієнта буде збережено в таблиці за допомогою методу create, а при виході з аккаунту чи з програми таблиця буде очищена за допомогою методу delete( в таблиці зберігається лише поточне ID клієнта):

class CLDAO\_Id implements DAO\_ID<SaveID> {  
 Connection connect;  
  
 @Override  
 public SaveID create(SaveID entity) {  
  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "INSERT INTO SavedID (ID) VALUES (?);";  
 PreparedStatement preparedStatement = connect.prepareStatement(sql);  
 preparedStatement.setInt(1, entity.getId());  
 preparedStatement.execute();  
 ResultSet rs = preparedStatement.getGeneratedKeys();  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return entity;  
 }  
  
  
 @Override  
 public boolean delete() {  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "DELETE FROM SavedID";  
 PreparedStatement preparedStatement = connect.prepareStatement(sql);  
 preparedStatement.execute();  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return false;  
 }  
  
}

## 2.7 Успішна авторизація

Завдяки, тому що ID користувача, при вході було збережено до БД, ми використовуємо метод find(клас CLDAO\_Id):

@Override  
public List<SaveID> find() {  
 List<SaveID> result = new ArrayList<>();  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "SELECT \*FROM SavedID";  
 Statement statement = connect.createStatement();  
 ResultSet resultSet = statement.executeQuery(sql);  
 while (resultSet.next()) {  
 int id = resultSet.getInt(1);  
 SaveID saveID = new SaveID(id);  
 saveID.setId(id);  
 result.add(saveID);  
  
 }  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return result;  
}

Буде виведено ім’я користувача та таблицю за допомогою методів:

public void showTable() {  
 CLDAO\_Order cldao\_order = new CLDAO\_Order();  
 ArrayList<Order> arrayList = new ArrayList<>(cldao\_order.findAll());  
 ArrayList<Order> redyForShow = new ArrayList<>();  
 Order order = new Order();  
 ArrayList<SaveID> arrList = new ArrayList<>(cldao\_id.find());  
 ArrayList<Client> clientArrayList = new ArrayList<>(cldao\_client.foundForId(arrList.get(0).getId()));  
 for (int i = 0; i < arrayList.size(); i++) {  
 order = arrayList.get(i);  
 if (order.getIdClient() == clientArrayList.get(0).getId()) {  
 redyForShow.add(order);  
 }  
 }  
 ObservableList<Order> show = FXCollections.*observableArrayList*(redyForShow);  
 date.setCellValueFactory(new PropertyValueFactory<Order, String>("date"));  
 number.setCellValueFactory(new PropertyValueFactory<Order, Integer>("number"));  
 sum.setCellValueFactory(new PropertyValueFactory<Order, Double>("sum"));  
 nameProducts.setCellValueFactory(new PropertyValueFactory<Order, String>("nameProducts"));  
 address.setCellValueFactory(new PropertyValueFactory<Order, String>("address"));  
 table.setItems(show);  
}

Ім’я:

public void setNameToLabel() {  
 ArrayList<SaveID> arrList = new ArrayList<>(cldao\_id.find());  
 ArrayList<Client> clientArrayList = new ArrayList<>(cldao\_client.foundForId(arrList.get(0).getId()));  
 welcomeLabel.setText(" " + clientArrayList.get(0).getName() + " " + clientArrayList.get(0).getSurname() + " " + clientArrayList.get(0).getMiddleName());  
  
}

## 2.8 Додавання товарів

Для того щоб додавати товари та додавати їх на склад(БД) було створено спеціальний клас CLDAO\_Production та сворено певні методи:

@Override  
public Production create(Production entity) {  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "INSERT INTO production (name,price,quantity,manufactured,producingCountry) VALUES (?,?,?,?,?);";  
 PreparedStatement preparedStatement = connect.prepareStatement(sql);  
 preparedStatement.setString(1, entity.getName());  
 preparedStatement.setDouble(2, entity.getPrice());  
 preparedStatement.setInt(3, entity.getQuantity());  
 preparedStatement.setString(4, entity.getManufactured());  
 preparedStatement.setString(5, entity.getProducingCountry());  
 preparedStatement.execute();  
 ResultSet rs = preparedStatement.getGeneratedKeys();  
 rs.next();  
 int key = rs.getInt(1);  
 entity.setNum(key);  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return entity;  
}

Та додавання на склад:

@Override  
public boolean update(Production entity) {  
 boolean result = false;  
 try {  
 if (connect == null) {  
 connect = getConnection();  
 }  
 String sql = "UPDATE production SET quantity=? WHERE num=?";  
 PreparedStatement prepareStatement = connect.prepareStatement(sql);  
 prepareStatement.setInt(1, entity.getQuantity());  
 prepareStatement.setInt(2, entity.getNum());  
 result = prepareStatement.execute();  
  
 } catch (SQLException throwables) {  
 throwables.printStackTrace();  
 }  
 return result;  
}

Для того, щоб адміністратор міг додати новий товар було створено наступний метод в контролері(AddProduct):

public void addProd() {  
 CLDAO\_Production cldao\_production=new CLDAO\_Production();  
 Production production=new Production(name.getText(),Double.*valueOf*(price.getText()),Integer.*valueOf*(quantity.getText()),manufactured.getText(),producingCountry.getText());  
 cldao\_production.create(production);  
 name.clear();  
 price.clear();  
 manufactured.clear();  
 quantity.clear();  
 producingCountry.clear();  
}

## 2.9 Додавання проекту на GitHub

## Для того, щоб зручно переглянути проект, його було завантажено на GitHub за посиланням(“https://github.com/TENOVOD/CourseWork”):

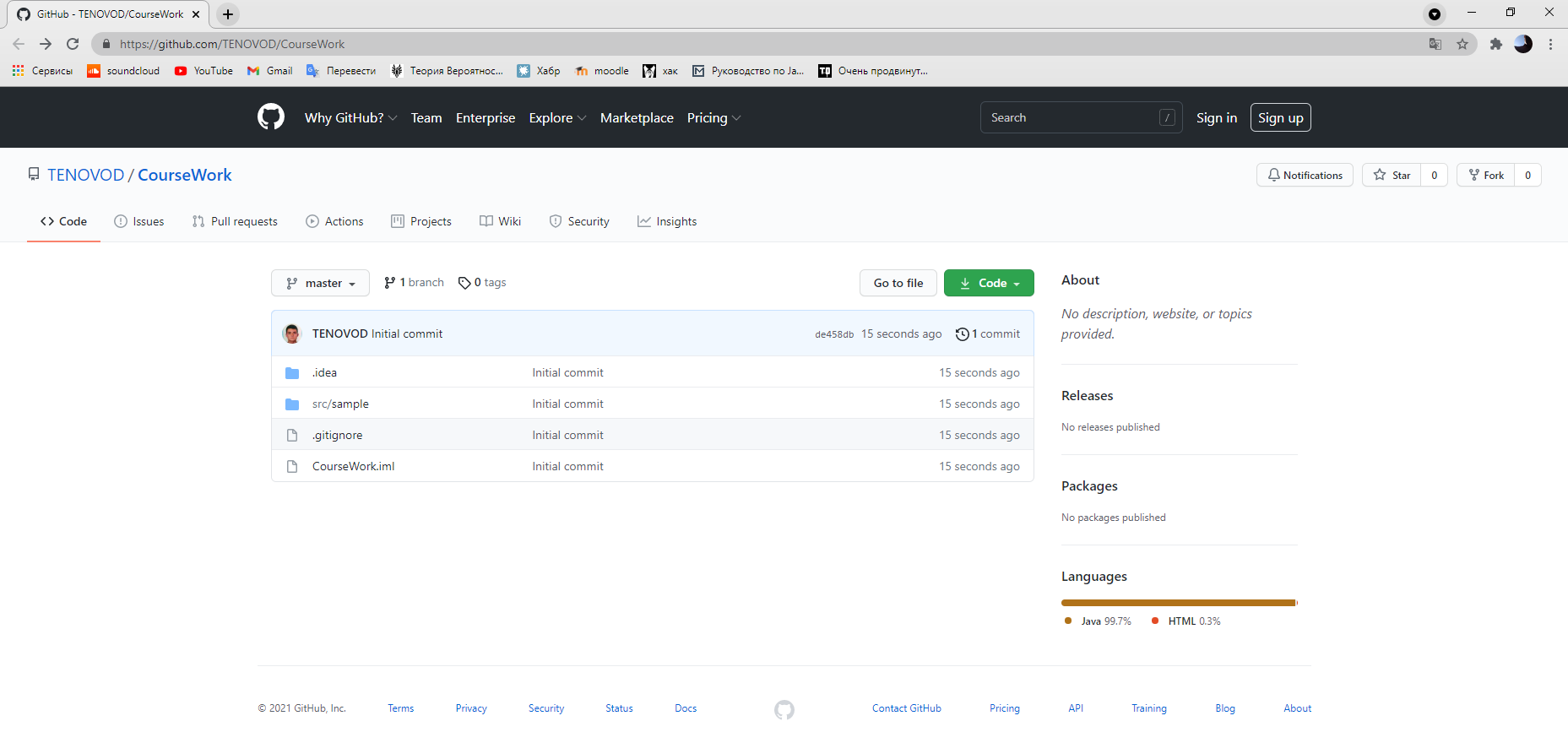


Рисунок 2.5 – Розміщення на GitHub.

# Розділ 3 Тестування застосунку

## 3.1 Авторизація

Рисунок 3.1 – Головне меню.

Для безпеки, для паролю було використано PasswordField

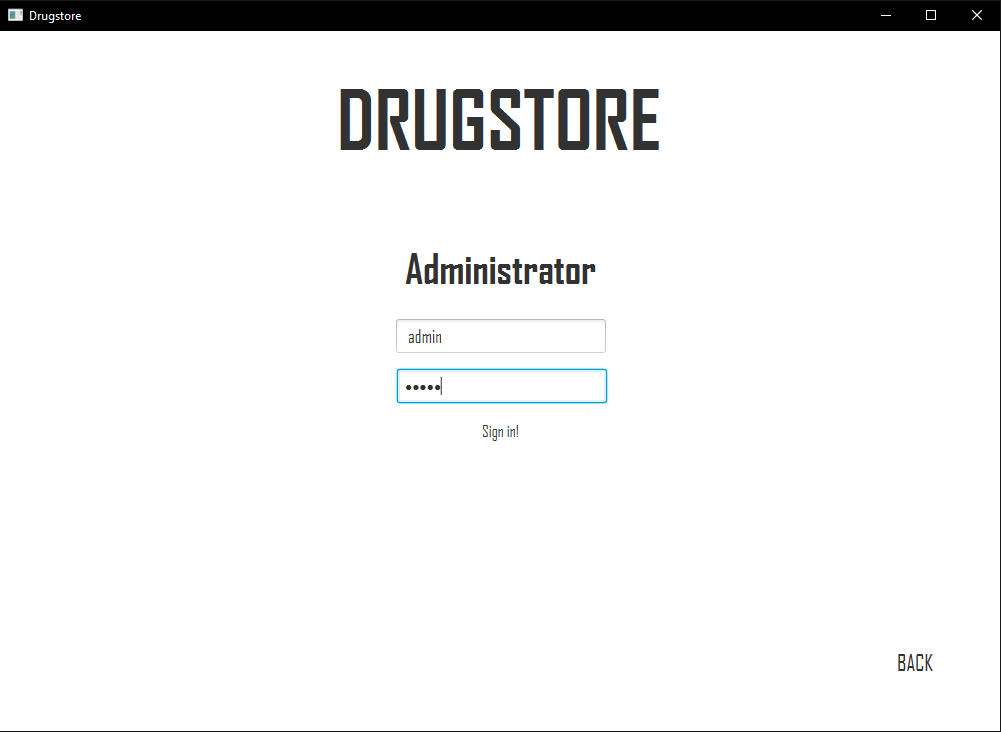


Рисунок 3.2 –Введених даних .

Лише для адміністратора, при невірному введені даних, немає пояснювального наппису

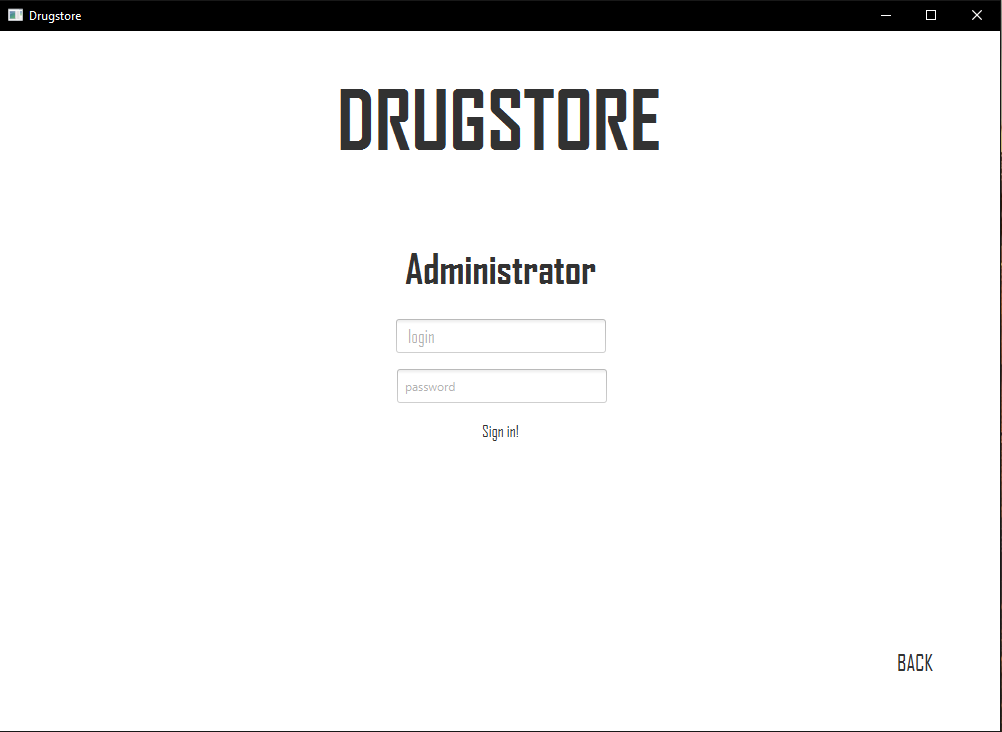


Рисунок 3.3 – Невірне введення даних.

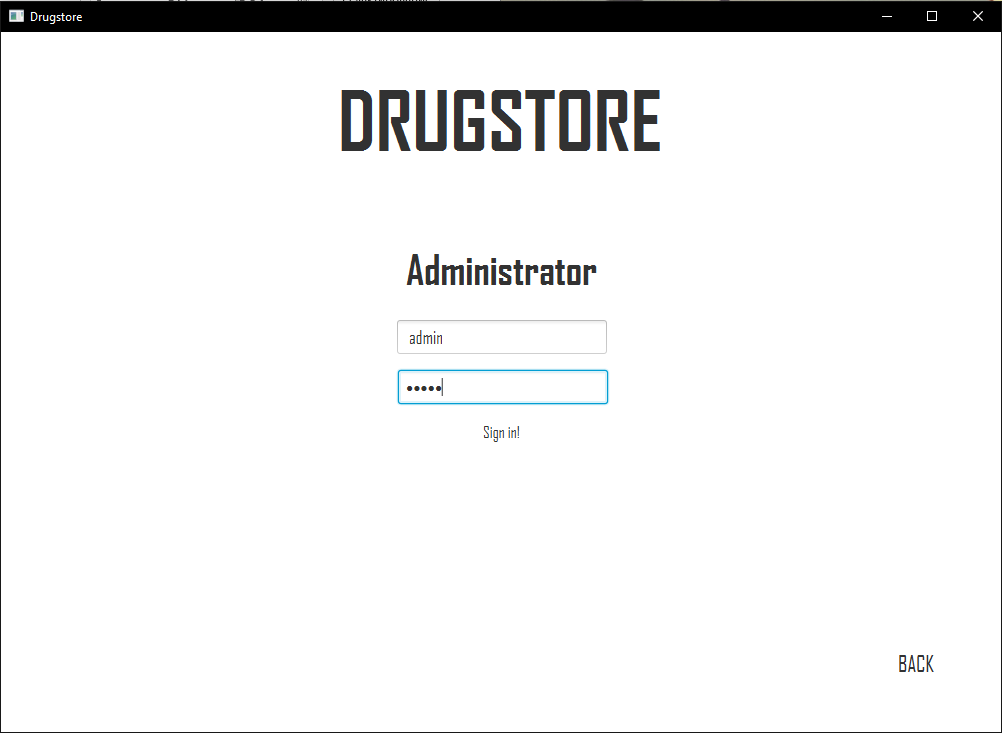


Рисунок 3.4 – Введення паролю, який вказаний в БД.

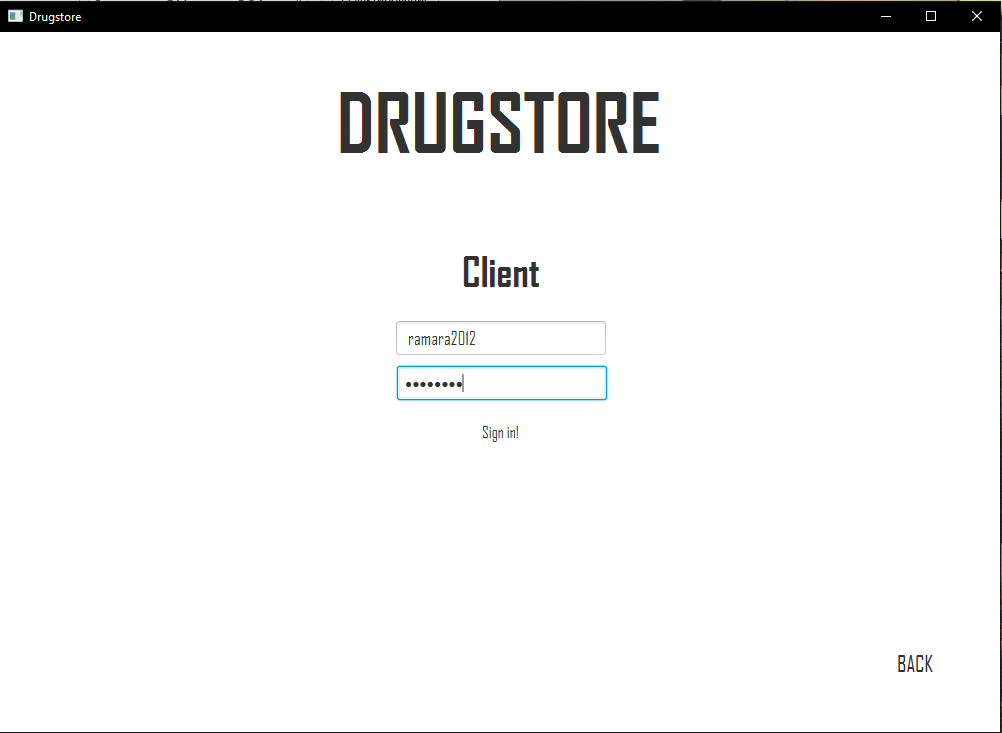


Рисунок 3.5 – При введення невірних даних.

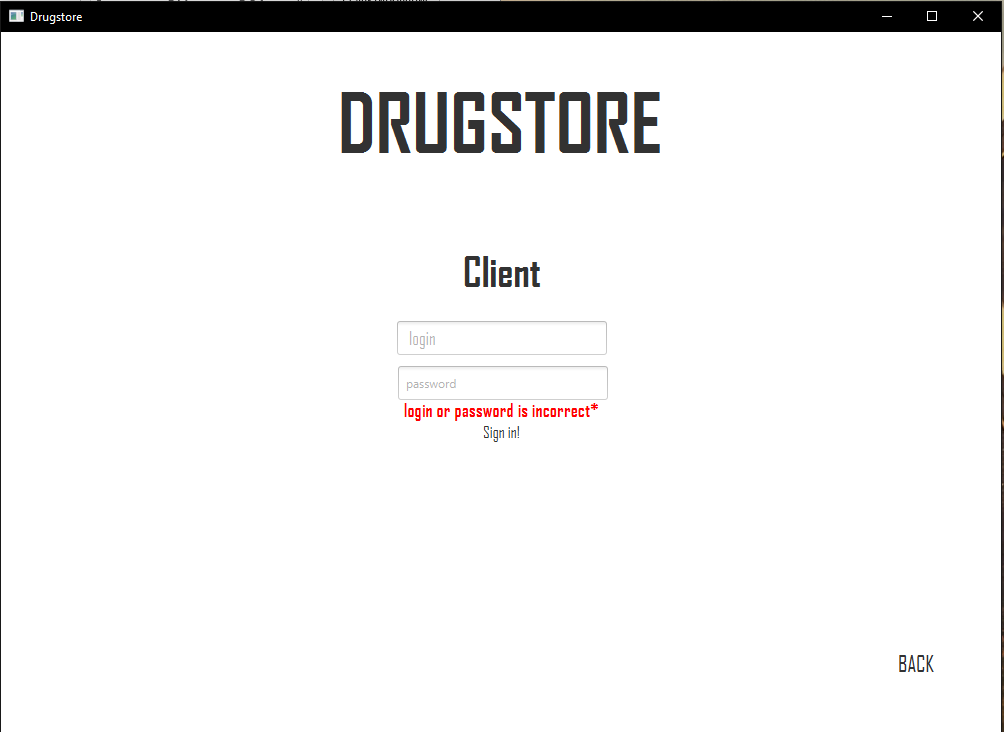


Рисунок 3.6 – При невірному введені данних.

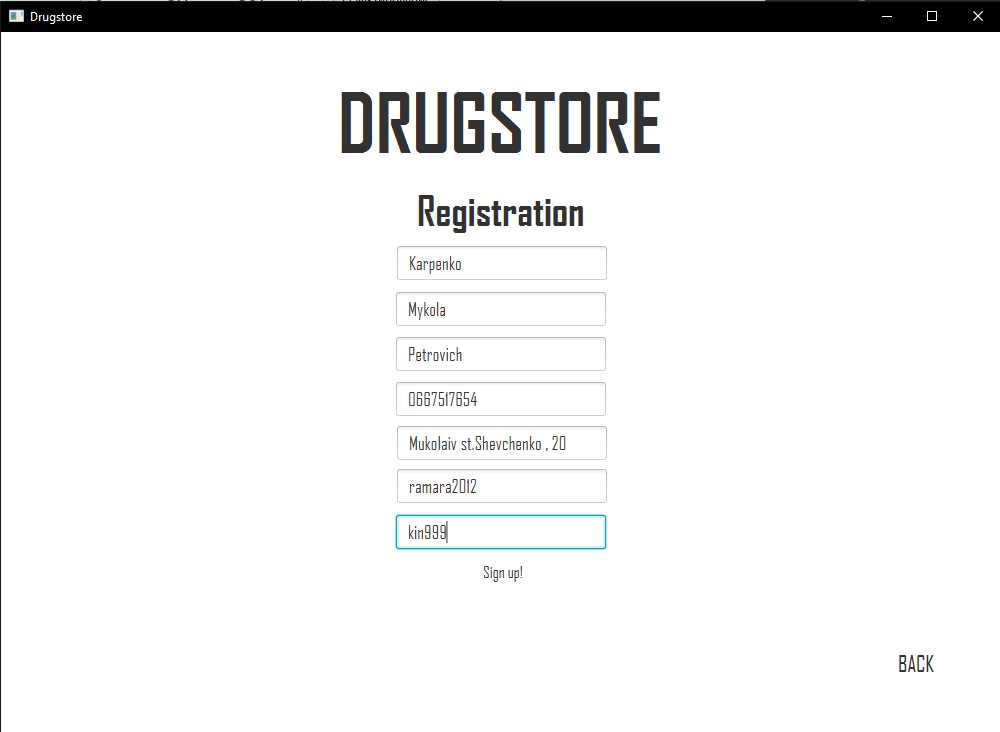


Рисунок 3.7 – Введення даних при реєстрації .

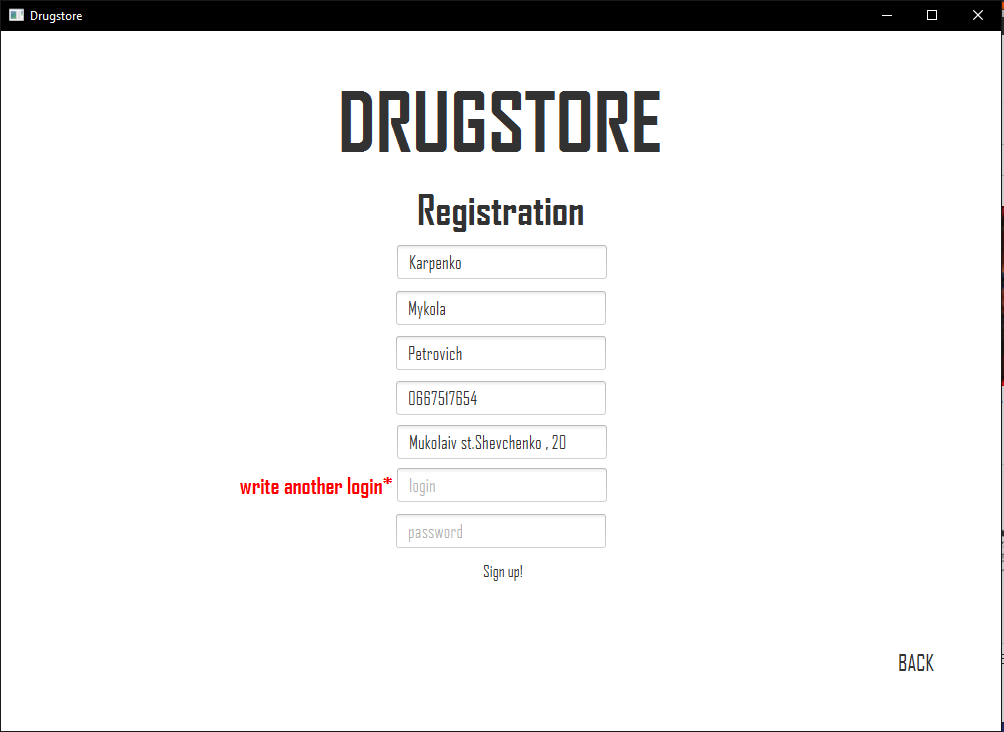


Рисунок 3.8 – Перевірка введених даних при реєстрації(існуючий логін).

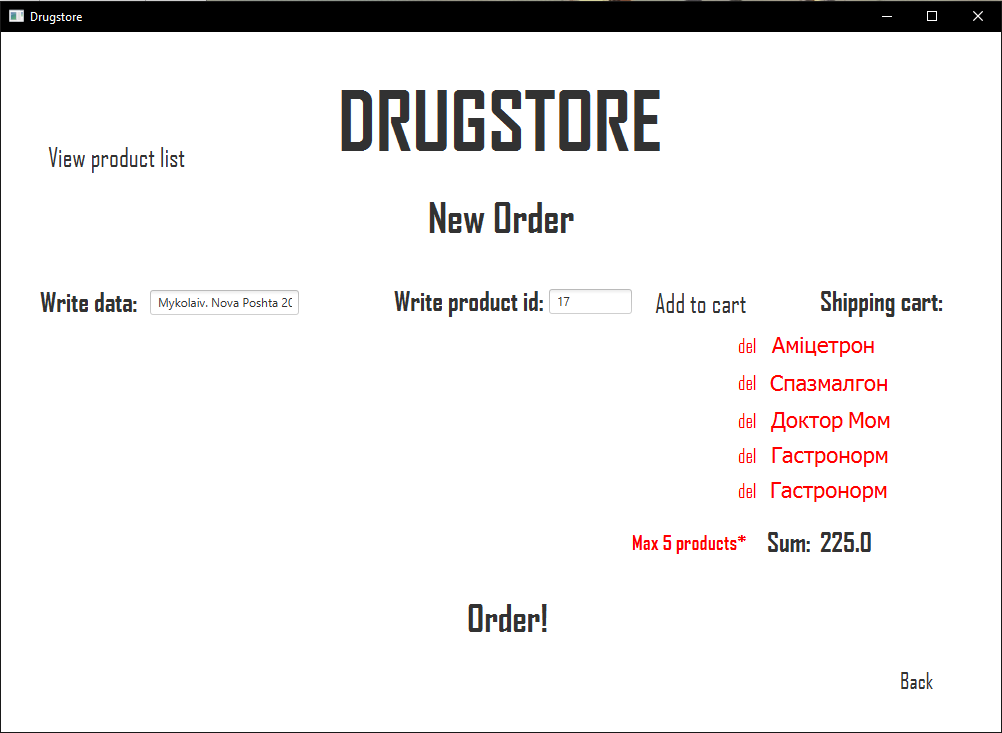


Рисунок 3.9 – Перевірка кількості товару у кошику.

## 3.2 Головні вкладки

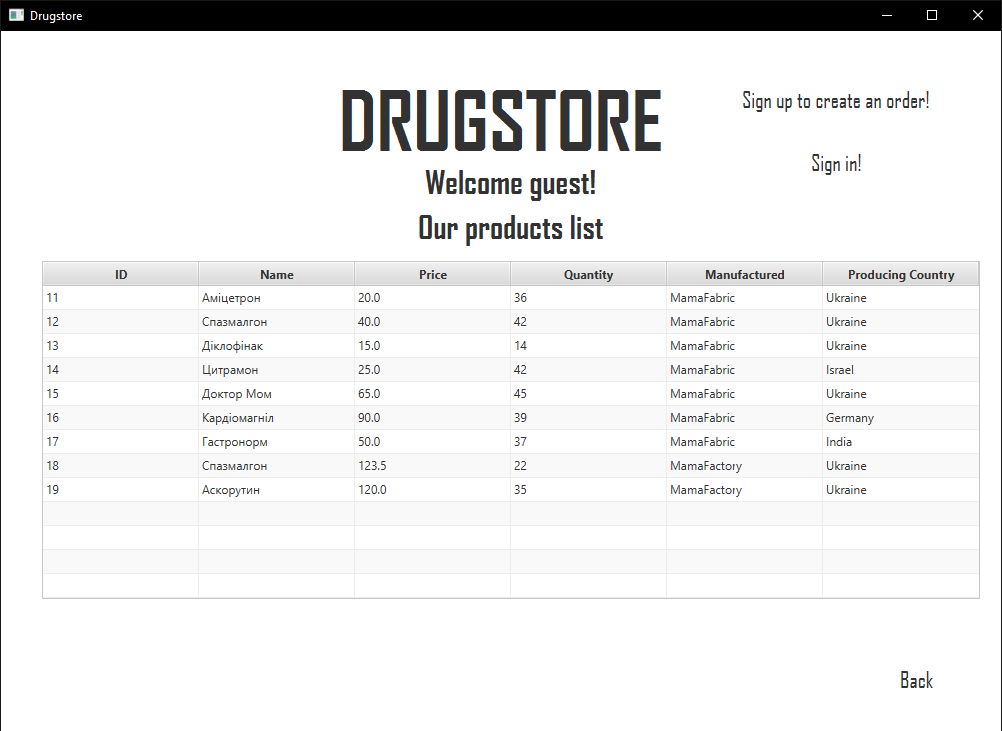


Рисунок 3.10 – Сторінка незареєстрованого користувача.

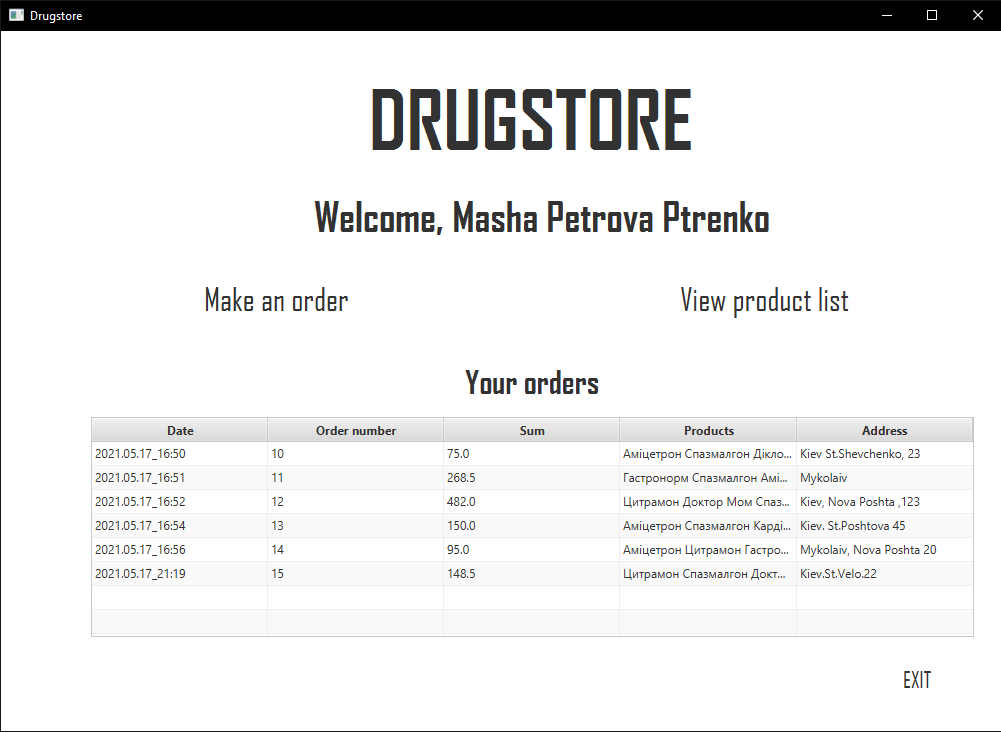


Рисунок 3.11 – Сторінка користувача при успішному вході до аккаунту.

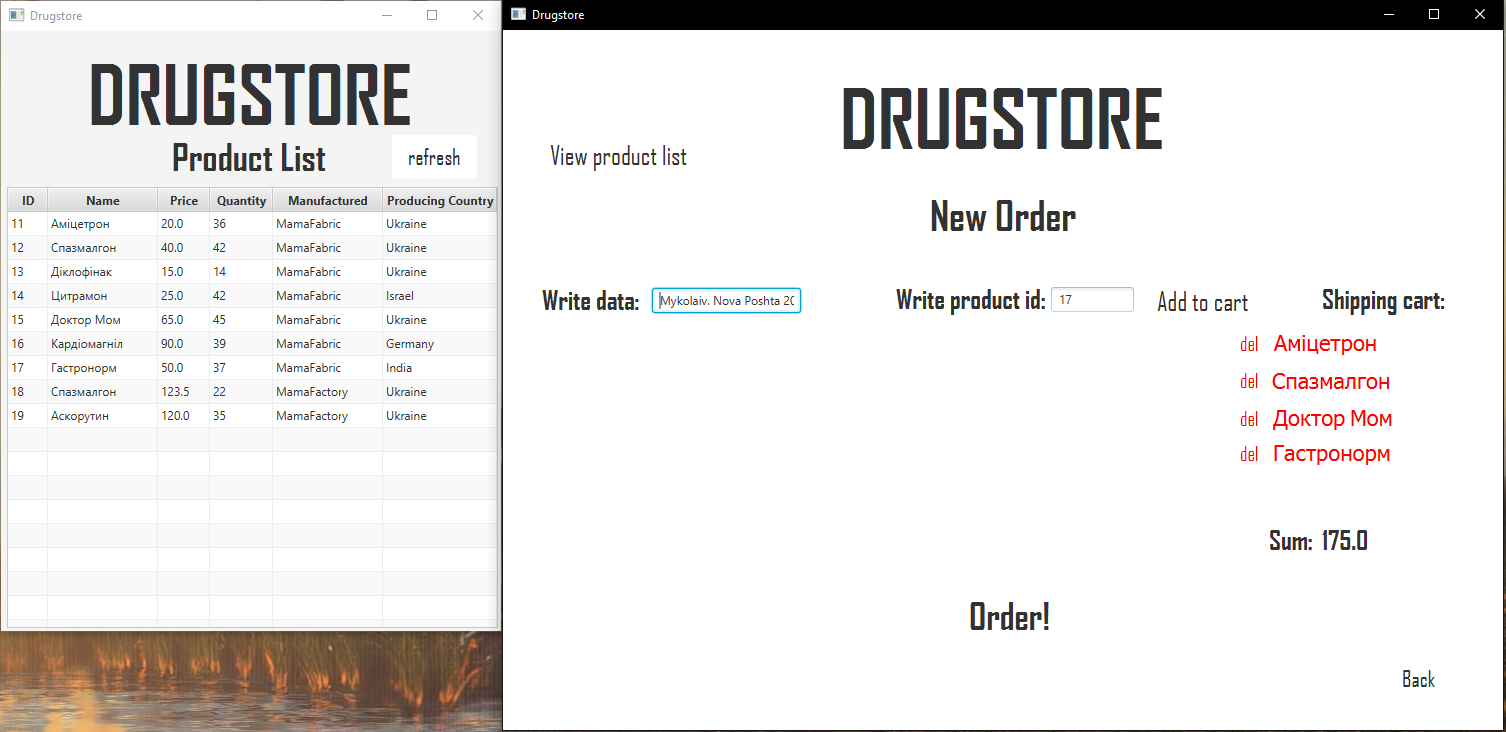


Рисунок 3.12 – Сторінка з товарами та вікно замовлення

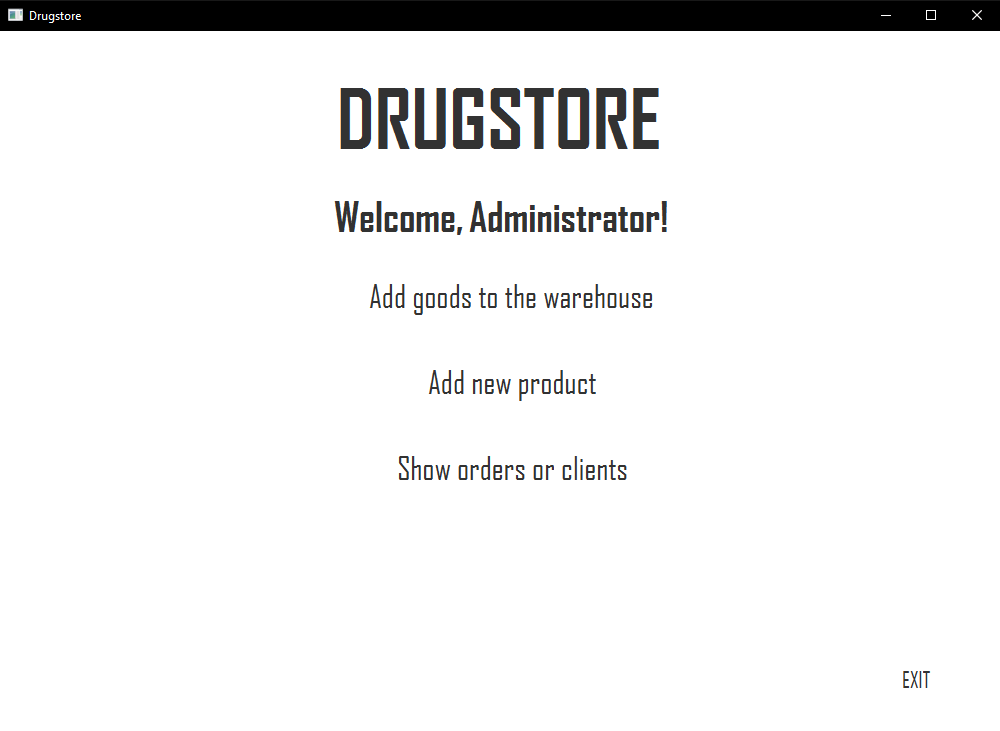


Рисунок 3.13 – Сторінка з функціями адміністратора .

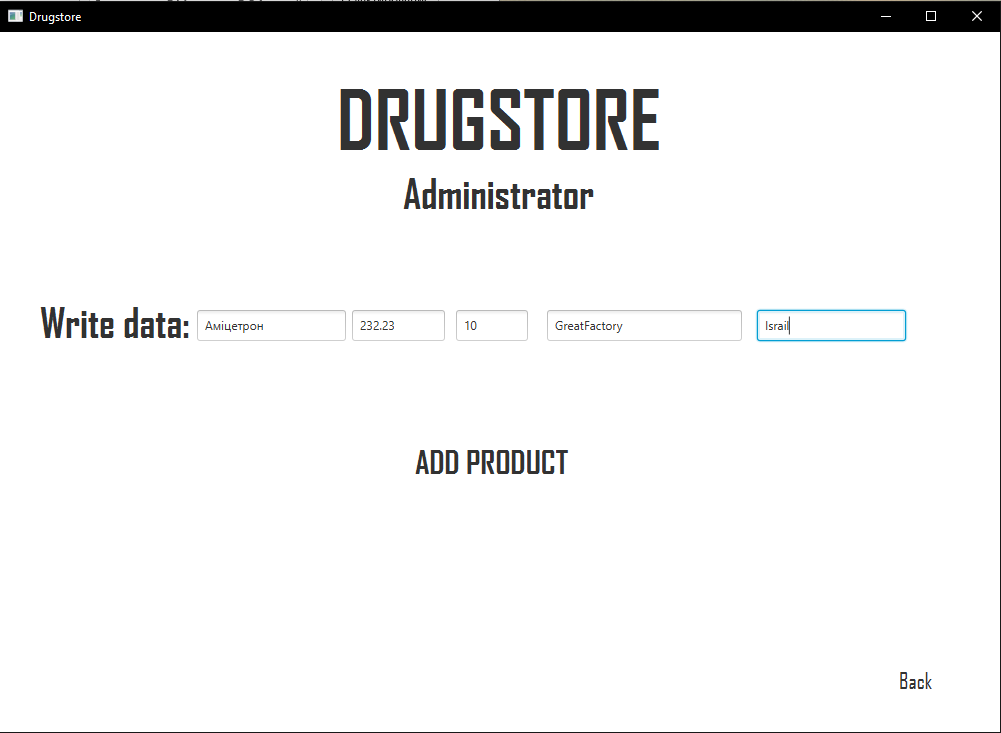


Рисунок 3.14 – Сторінка для додавання нового товару.

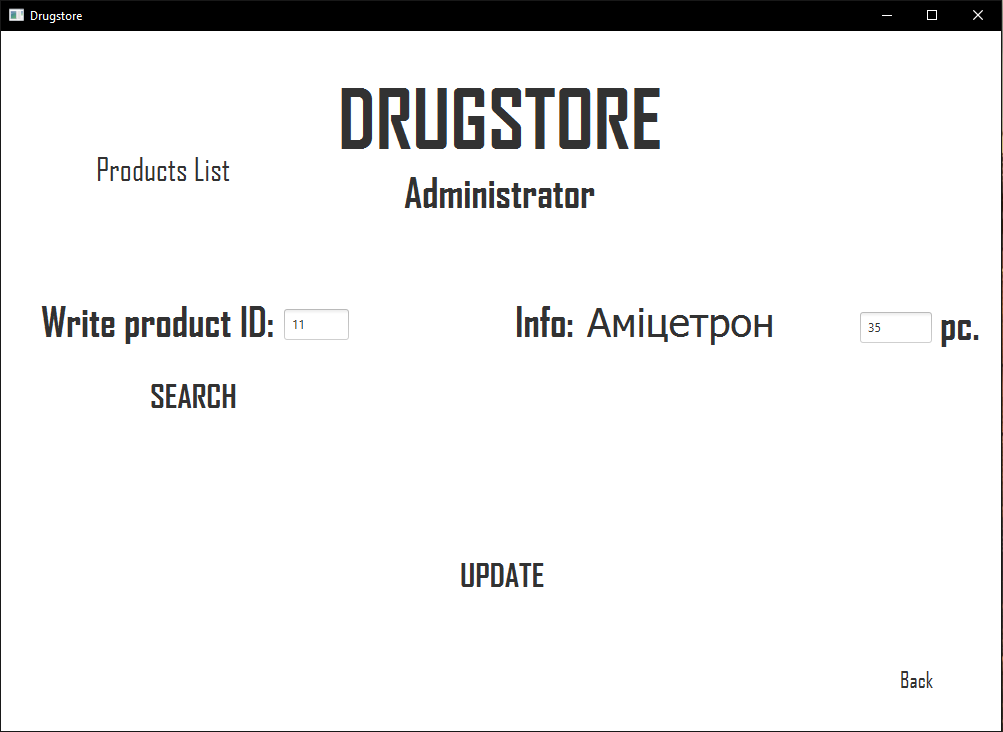


Рисунок 3.15 – Сторінка для додавання товару на склад.

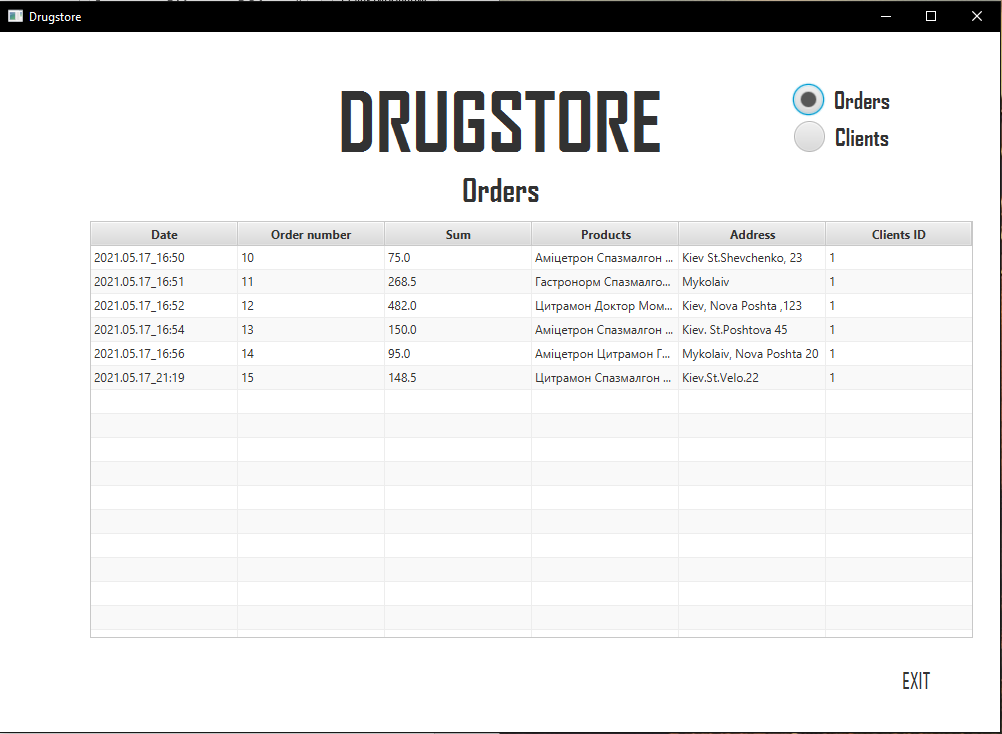


Рисунок 3.16 – Сторінка зі списком всіх замовлень.

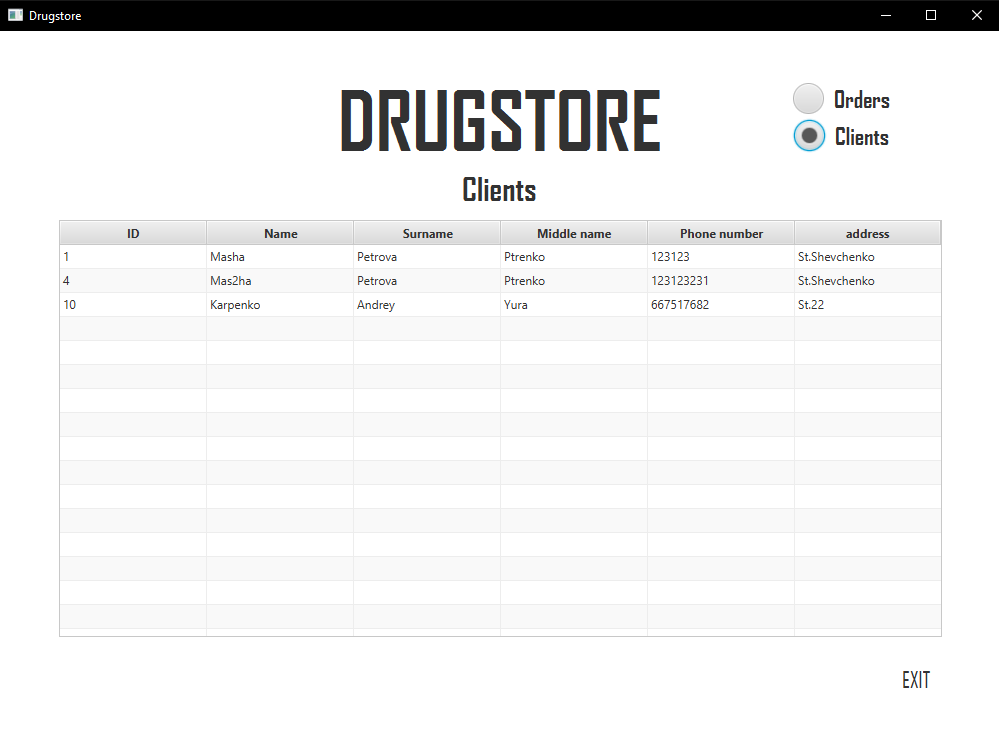


Рисунок 3.17 – Сторінка зі списком всіх клієнтів.

# 

# висновки

Під час виконання курсової роботи було створено застосунок, за допомогою якого можна купувати медикаменти, оновлювати асортимент та додавати товари на склад. Для візуального представлення даних використано платформу JavaFX на основі Java. У ролі збереження даних використано SQLite, а саме безкоштовну програму SQLiteStudio.

Повністю виконана поставлена мета роботи та пройдено усі етапи розробки програмного забеспечення(складання моделей, розробка макетів для користувацького інтерфейсу, розробки БД, розробка графічного інтерфейсу та тестування готового продукту)

Для розробки використано навички отримані в результаті вивчення дисципліни «Об’єктно – орієнтоване програмування», навички які були набуті самостійно і інтернет ресурси(книги і курси). Результатом є готова автоматизована система «Аптека», яка може стати основою більш серйозного застосунку.

Застосунок було розміщено в системі контролю версій GitHub, щоб скористатися ним, необхідно лише скомпілювати код.

URL: https://github.com/TENOVOD/CourseWork

# перелік джерел посилання

1.JavaFX Documentation. URL: https://openjfx.io/ .

2. Руководство по JavaFX. URL: https://metanit.com/java/javafx/ .

3.SQLiteStudio. URL: https://sqlitestudio.pl/ .

4.Scene Builder. URL: https://gluonhq.com/products/scene-builder/ .

5. GitHub. URL: <https://github.com/> .

6. Robert C.Martin. Clean Code. Apress 2019. 464p.

7. Bruce Eckel. Thinking in Java. Apress 2001. 1168р.

8. Drue Coles. The Practice and Philosophy of Object-Oriented Programming in Java. Apress 2020. 454p.

9. Katie Sierra. Learning Java. Apress 2013. 720p.

10. Herbert Schildt. Java. The Complete Guide, 10th Edition.Apress 2018. 1488p.

**ЗВІТ**

про унікальність курсової роботи на тему:

Автоматизована система «Турагенція»

студента спеціальності 122 Комп’ютерні науки, 202 групи

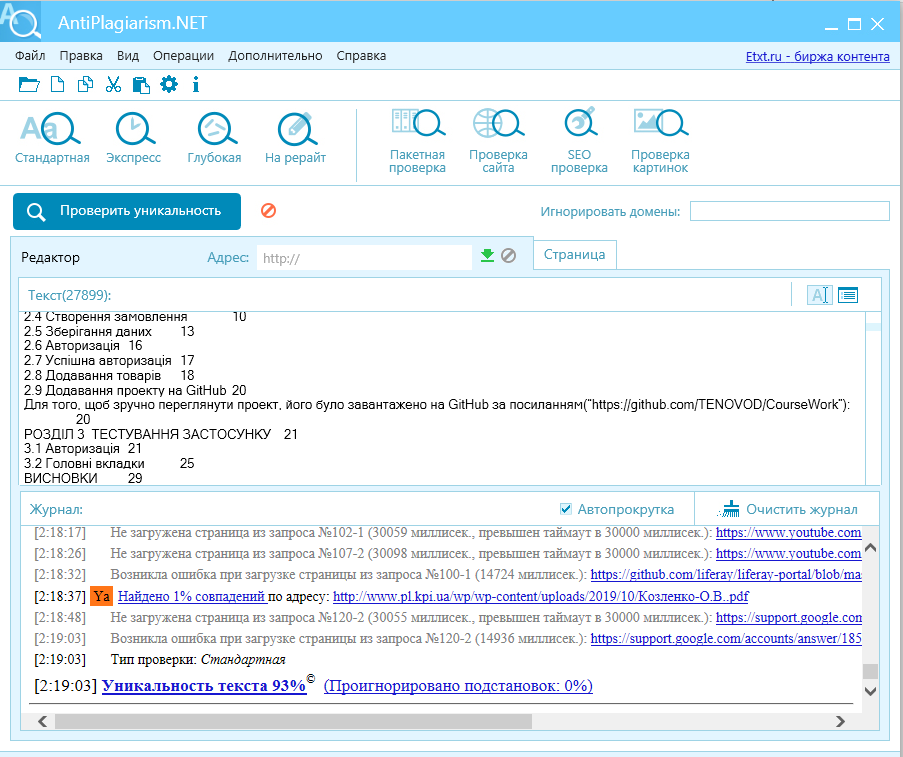
|  |
| --- |
| Карпенка Андрія Юрійовича |
| прізвище, ім’я, по-батькові |

|  |  |
| --- | --- |
| Шифр роботи: | 122 – КР.ПЗ.00 – 202.1910211 |

|  |  |
| --- | --- |
| Перевірку тексту здійснено сервісом: | Програма «AntiPlagiarism.NET» |
|  |  |

Результат перевірки тексту курсового проекту на унікальність складає 93%.

**Скріншот** результату  наведений на рисунку нижче.



Ст. викладач кафедри ІПЗ                                                  А. В. Швед

|  |  |  |  |
| --- | --- | --- | --- |
| Дата: | « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | 2021 р. |  |