Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
по лабораторной работе
№10 Дисциплина «СПП»

Выполнил:
Студент гр. ПО-3

Будяков В.В.

Проверил:
Крощенко А. А.

Брест 2021

Цель работы: приобрести практические навыки разработки многооконных приложений на JavaFX для работы с базами данных

Задание:

На основе БД, разработанной в лабораторной работе No9, реализовать многооконное приложение-клиент, позволяющее выполнять основные операции над таблицей в БД (добавление, удаление, модификацию данных).

База данных «Торгово-закупочная деятельность фирмы»

Ход работы

Текст программы:

Main.java

```
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;

public class Main extends Application {
  @Override
  public void start(Stage primaryStage) throws Exception{
     Parent root = FXMLLoader.load(getClass().getResource("sample.fxml"));
primaryStage.setTitle("СПП 3 лаба");
     primaryStage.setScene(new Scene(root, 600, 400));
     primaryStage.setMinHeight(400);
     primaryStage.setMinWidth(600);
     primaryStage.show();
  }
  public static void main(String[] args) {
     launch(args);
  }
}
```

DatabaseController.java

```
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class DatabaseController {
  Connection connection;
  Statement state;
  public DatabaseController () throws SQLException {
     connection=

DriverManager.getConnection("jdbc:postgresql://localhost:5432/lab3?user=postgres&password=1111&ssl
=false");
     state = connection.createStatement();
  }
  public List<Element> getProductsClients() throws SQLException
  {
     ResultSet result = state.executeQuery("SELECT * FROM products_clients;");  List<Element> list =
new ArrayList<Element>();
     while(result.next()) {
        Statement state2= connection.createStatement();
        Statement state3= connection.createStatement();
        String client1="null", product1="null", contr="null";
        ResultSet client = state2.executeQuery("SELECT * FROM clients WHERE
idclients="+result.getString(3)+";");
        if(client.next())
           client1=client.getString(3)+ " " + client.getString(2);
        ResultSet product = state3.executeQuery("SELECT * FROM products WHERE
idproducts="+result.getString(2)+";");
        if(product.next())
           product1=product.getString(2);
        String idProduct = result.getString(2);
        Statement state4 = connection.createStatement();
        ResultSet result4 = state4.executeQuery("SELECT name FROM providers WHERE idproviders =
(SELECT idproviders FROM products_providers WHERE idproducts ="+idProduct+");");
        if(result4.next())
           contr = result4.getString(1);
        list.add(new Element(product1, client1, result.getInt(4), contr, result.getInt(1)));
     }
     return list;
  }
  public int getProductInProductsClientsById(int id) throws SQLException  {
     String query=String.format("SELECT * FROM products_clients WHERE idproducts_clients= %d;",
id);
     ResultSet result = state.executeQuery(query);
```

```
        if(!result.next()) {
            System.out.println("-1!!!!!!!!!!!!!!!!!!!!");
            return -1;
        }
        return Integer.parseInt(result.getString(2));
    }
    public int getClientInProductsClientsById(int id) throws SQLException  {
        String query=String.format("SELECT * FROM products_clients WHERE idproducts_clients= %d;",
id);
        ResultSet result = state.executeQuery(query);
        if(!result.next()) {
            System.out.println("-1!!!!!!!!!!!!!!!!!!!!");
            return -1;
        }
        return Integer.parseInt(result.getString(3));
    }
    public int getContrInProductsClientsById(int id) throws SQLException  {
        String query=String.format("SELECT idproviders FROM providers WHERE idproviders= %d;", id);
        ResultSet result = state.executeQuery(query);
        if(!result.next()) {
            System.out.println("-1!!!!!!!!!!!!!!!!!!!!");
            return -1;
        }
        return Integer.parseInt(result.getString(1));
    }
    public int getCountInProductsClientsById(int id) throws SQLException  {
        String query=String.format("SELECT * FROM products_clients WHERE idproducts_clients= %d;",
id);
        ResultSet result = state.executeQuery(query);
        if(!result.next()) {
            System.out.println("-1!!!!!!!!!!!!!!!!!!!!");
            return -1;
        }
        return Integer.parseInt(result.getString(4));
    }
    public void deleteRow(int id) throws SQLException
    {
        String query=String.format("DELETE FROM products_clients WHERE  idproducts_clients='%s';",
String.valueOf(id));
        state.executeUpdate(query);
    }
    public List<Product> getProducts() throws SQLException {
        ResultSet result=state.executeQuery("SELECT * FROM products;");  List<Product> list=new
ArrayList<Product>();
        while(result.next())
        {
```

```
        Product product=new Product(result.getInt(1), result.getString(2),  result.getDouble(3), result.getInt(4));
        list.add(product);
    }
    return list;
  }
  public List<Client> getClients() throws SQLException {
    ResultSet result=state.executeQuery("SELECT * FROM clients;");  List<Client> list=new ArrayList<Client>();
    while(result.next())
    {
        Client client=new Client(result.getInt(1), result.getString(2),  result.getString(3), result.getString(4));
        list.add(client);
    }
    return list;
  }
  public List<Contr> getContrs() throws SQLException {
    ResultSet result=state.executeQuery("SELECT * FROM providers;");
    List<Contr> list=new ArrayList<Contr>();
    while(result.next())
    {
        Contr contr=new Contr(result.getInt(1), result.getString(2),  result.getString(3), result.getString(4));
        list.add(contr);
    }
    return list;
  }
  public boolean addNewRecord(int idProduct, int idClient, int count, int idContr) throws SQLException
  {
    String query=String.format("INSERT INTO products_clients (idproducts, idclients,  count) VALUES ('%d', '%d', '%d');", idProduct, idClient, count);
    String query1=String.format("UPDATE products_providers SET idproviders='%d' WHERE idproducts='%d';", idContr, idProduct);
    try {
        state.executeUpdate(query);
        state.executeUpdate(query1);
        return true;
    }
    catch(SQLIntegrityConstraintViolationException E)
    {
        return false;
    }
  }
  public boolean updateRecord(int id, int idProduct, int idClient, int count, int idContr) throws SQLException
  {
```

```
        String query=String.format("UPDATE products_clients SET idproducts='%d', idclients='%d',
count='%d' WHERE idproducts_clients='%d';", idProduct, idClient, count,  id);
        String query1=String.format("UPDATE products_providers SET idproviders='%d' WHERE
idproducts='%d';", idContr, idProduct);
        try {
            state.executeUpdate(query);
            state.executeUpdate(query1);
            return true;
        }
        catch(SQLIntegrityConstraintViolationException E)
        {
            return false;
        }
    }
}
```

Controller.java:

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.FXMLLoader;
import javafx.fxml.Initializable;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.awt.event.ActionEvent;
import java.net.URL;
import java.util.Comparator;
import java.util.List;
import java.util.ResourceBundle;

public class Controller implements Initializable {
    @FXML
    public TableView<Element> table;
    @FXML
    public TableColumn<Element, String> tableItemProduct;
    @FXML
    public TableColumn<Element, String> tableItemClient;
    @FXML
    public TableColumn<Element, Integer> tableItemCount;
    @FXML
    public TableColumn<Element, Integer> tableItemId;
    @FXML
    public TableColumn<Element, String> tableItemContr;
    @FXML
    public CheckBox checkBox;
    @FXML
```

```
    public TextField textArea;
    @FXML
    public Button searchButton;
    DatabaseController DBController;
    boolean searchState=false;
    @Override
    public void initialize(URL location, ResourceBundle resources)  {
        tableItemProduct.setCellValueFactory(new PropertyValueFactory<>("product"));
        tableItemClient.setCellValueFactory(new PropertyValueFactory<>("client"));
        tableItemCount.setCellValueFactory(new PropertyValueFactory<>("count"));
        tableItemContr.setCellValueFactory(new PropertyValueFactory<>("contr"));
        try {
            DBController = new DatabaseController();
        }
        catch(Exception e){System.out.println("Exception "+e.toString());}
        updateTable();
    }
    void updateTable()
    {
        try {
            List<Element> list=DBController.getProductsClients();  if(checkBox.isSelected());
            {
                class comparator implements Comparator<Element> {
                    public int compare(Element a, Element b) {
                        return a.getProduct().compareTo(b.getProduct());
                    }
                }
                list.sort(new comparator());
            }
            if(searchState)
                for(int i=0; i<list.size(); i++)
                    if(!list.get(i).getProduct().contains(textArea.getText())) {
                        list.remove(i);
                        i--;
                    }
            ObservableList<Element> tableItemsList = FXCollections.observableArrayList();  for(int i = 0; i<list.size();
i++)
                tableItemsList.add(list.get(i));
            table.setItems(tableItemsList);
        }
        catch(Exception e)
        {
            System.out.println("Exception "+e.toString());
        }
    }
    public void clickCheckBox(javafx.event.ActionEvent event)
    {
        updateTable();
    }
    public void clickSearchButton(javafx.event.ActionEvent event)
    {
        if(textArea.getText().isEmpty())
            searchState=false;
        else
            searchState=true;
        updateTable();
    }
    public void clickDeleteButton(javafx.event.ActionEvent event)
    {
        Alert alert = new Alert(Alert.AlertType.WARNING);
```

```
        alert.setTitle("Внимание");
        alert.setHeaderText("Подтвердите удаление");
        alert.setContentText("Вы точно хотите удалить запись?");
        alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
                try {

                    DBController.deleteRow(table.getSelectionModel().getSelectedItem().getId());  updateTable();
                }
                catch(Exception e)
                {
                    System.out.println("Exception "+e.toString());
                }
            }
        });
    }
    public void clickAddButton(javafx.event.ActionEvent event)
    {
        initializeAddWindow(0, 0);
    }
    public void clickUpdateButton(javafx.event.ActionEvent event)
    {
        try{
            int id=table.getSelectionModel().getSelectedItem().getId();  initializeAddWindow(1, id);
        }
        catch(Exception e)
        {
            Alert alert = new Alert(Alert.AlertType.ERROR);
            alert.setTitle("Ошибка");
            alert.setHeaderText("Произошла ошибка");
            alert.setContentText("Проверьте, указали ли вы запись для обновления.");
alert.showAndWait().ifPresent(rs -> {
            if (rs == ButtonType.OK) {
            }
        });
            e.printStackTrace();
        }
    }
    public void initializeAddWindow(int mode, int id)
    {
        try {
            FXMLLoader loader= new FXMLLoader(this.getClass().getResource("adder.fxml"));  Parent root =
(Parent)loader.load();
            Stage addWindow = new Stage();
            addWindow.initModality(Modality.APPLICATION_MODAL);
            if(mode==0)
                addWindow.setTitle("Добавить");
            else
                addWindow.setTitle("Обновить");
            addWindow.setScene(new Scene(root, 300, 300));
            addWindow.setMinHeight(300);
            addWindow.setMinWidth(300);
            addWindow.setMaxHeight(300);
            addWindow.setMaxWidth(300);
            addWindow.setOnHiding(event1 -> {
                System.out.println("CLOSED");
                updateTable();
            });
            AdderController adderController = loader.getController();
            adderController.start(mode, id);
```

```
            addWindow.show();
        }
        catch(Exception e)
        {
            System.out.println("Exception"+e.toString());
            e.printStackTrace();
        }
    }
}
```

AdderController.java:

```
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.Node;
import javafx.scene.control.*;
import javafx.stage.Stage;
import javafx.util.Callback;

import java.net.URL;
import java.util.List;
import java.util.ResourceBundle;

class Item
{
    public int id;
    public String text;
    public Item(int id, String text)
    {
        this.id=id;
        this.text=text;
    }
}
public class AdderController implements Initializable {
    @FXML
    public ComboBox comboBoxProducts;
    @FXML
    public ComboBox comboBoxClients;
    @FXML
    public ComboBox comboBoxContr;
    @FXML
    public TextField textAreaCount;
    @FXML
    public Button buttonAdd;
    private int id;
    private int mode;
    private List<Product> products;
    private List<Client> clients;
    private List<Contr> contrs;
    @Override
    public void initialize(URL location, ResourceBundle resources)
    {}
    public void start(int mode, int id)
    {
        this.id=id;
        this.mode=mode;
```

```
        try {
            DatabaseController DBController = new DatabaseController();
            products=DBController.getProducts();
            clients=DBController.getClients();
            contrs = DBController.getContrs();
            ObservableList<Item> listProducts = FXCollections.observableArrayList();
            ObservableList<Item> listClients = FXCollections.observableArrayList();
            ObservableList<Item> listContrs = FXCollections.observableArrayList();
            for(int i=0; i<products.size(); i++)
                listProducts.add(new Item(products.get(i).getId(),  products.get(i).getName()));
            for(int i=0; i<clients.size(); i++)
                listClients.add(new Item(clients.get(i).getId(),  clients.get(i).getSurname()+" "+clients.get(i).getName()));
            for(int i=0; i<contrs.size(); i++)
                listContrs.add(new Item(contrs.get(i).getId(),  contrs.get(i).getName()));
            comboBoxProducts.setItems(listProducts);
            comboBoxClients.setItems(listClients);
            comboBoxContr.setItems(listContrs);
            Callback<ListView<Item>, ListCell<Item>> factory = new  Callback<ListView<Item>, ListCell<Item>>() {
                @Override
                public ListCell<Item> call(ListView<Item> l) {
                    return new ListCell<Item>() {
                        @Override
                        protected void updateItem(Item item, boolean empty) {
                            super.updateItem(item, empty);
                            if (item == null || empty) {
                                setGraphic(null);
                            } else {
                                setText(item.text);
                            }
                        }
                    } ;
                }
            };
            comboBoxProducts.setCellFactory(factory);
            comboBoxProducts.setButtonCell(factory.call(null));
            comboBoxClients.setCellFactory(factory);
            comboBoxClients.setButtonCell(factory.call(null));
            comboBoxContr.setCellFactory(factory);
            comboBoxContr.setButtonCell(factory.call(null));
            if(mode==1) {
                buttonAdd.setText("Обновить");
                List<Element> list = DBController.getProductsClients();
                for (int i = 0; i < list.size(); i++)
                    if (list.get(i).getId() == id) {
                        int idProduct =
                                DBController.getProductInProductsClientsById(id);
                        for (int j = 0; j < products.size(); j++)  if (products.get(j).getId() == idProduct) {
comboBoxProducts.getSelectionModel().select(j); break;
                        }
                        int idClient =
                                DBController.getClientInProductsClientsById(id);
                        for (int j = 0; j < clients.size(); j++)  if (clients.get(j).getId() == idClient) {
comboBoxClients.getSelectionModel().select(j);  break;
                        }

                        int idContr =
                                DBController.getContrInProductsClientsById(id);
                        for (int j = 0; j < contrs.size(); j++)  if (contrs.get(j).getId() == idContr) {
comboBoxContr.getSelectionModel().select(j);  break;
                        }
```

```
                    textAreaCount.setText(String.valueOf(DBController.getCountInProductsClientsById(id)));  break;
                }
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
    public void addButtonClick(javafx.event.ActionEvent actionEvent) {  try {
        Item product = (Item) comboBoxProducts.getValue();
        Item client = (Item) comboBoxClients.getValue();
        Item contr = (Item) comboBoxContr.getValue();
        int count=Integer.parseInt(textAreaCount.getText());
        if(count<=0)
            throw new Exception();
        DatabaseController DBController = new DatabaseController();
        boolean result;
        if(mode==0)
            result=DBController.addNewRecord(product.id, client.id, count, contr.id);
        else
            result=DBController.updateRecord(id, product.id, client.id, count, contr.id);
            if(!result)
                throw new Exception();
        Node node=(Node)actionEvent.getSource();
        Stage stage=(Stage)node.getScene().getWindow();
        stage.close();
    }
    catch(Exception e)
    {
        Alert alert = new Alert(Alert.AlertType.ERROR);  alert.setTitle("Ошибка");
        alert.setHeaderText("Произошла ошибка");
        alert.setContentText("Указаны неверные данные.");  alert.showAndWait();
        e.printStackTrace();
    }
    }
}
```

Client.java:

```
public class Client {
    private int id;
    private String name;
    private String surname;
    private String phone;
    public Client(int id, String name, String surname, String phone) {  this.id = id;
        this.name = name;
        this.surname = surname;
        this.phone = phone;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
```

```
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getSurname() {
        return surname;
    }
    public void setSurname(String surname) {
        this.surname = surname;
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
}
```

Contr.java:

```
public class Contr {
        private int id;
        private String name;
        private String adres;
        private String phone;
        public Contr(int id, String name, String adres, String phone) {
            this.id = id;
            this.name = name;
            this.adres = adres;
            this.phone = phone;
        }
        public int getId() {
            return id;
        }
        public void setId(int id) {
            this.id = id;
        }
        public String getName() {
            return name;
        }
        public void setName(String name) {
            this.name = name;
        }
        public String getAdres() {
            return adres;
        }
        public void setAdres(String adres) {
            this.adres = adres;
        }
        public String getPhone() {
            return phone;
        }
        public void setPhone(String phone) {
            this.phone = phone;
        }
}
```

Element.java:

```
import javafx.beans.property.SimpleIntegerProperty;
import javafx.beans.property.SimpleStringProperty;

public class Element {
    private SimpleStringProperty product;
    private SimpleStringProperty client;
    private SimpleIntegerProperty count;
    private SimpleStringProperty contrName;
    private SimpleIntegerProperty id;
    public Element(String product, String client, int count, String contrName, int id) {
        this.product = new SimpleStringProperty(product);
        this.client = new SimpleStringProperty(client);
        this.count = new SimpleIntegerProperty(count);
        this.contrName = new SimpleStringProperty(contrName);
        this.id = new SimpleIntegerProperty(id);
    }
    public String getProduct() {
        return product.get();
    }
    public void setProduct(String product) {
        this.product.set(product);
    }
    public String getClient() {
        return client.get();
    }
    public void setClient(String client) {
        this.client.set(client);
    }
    public int getCount() {
        return count.get();
    }
    public void setCount(int count) {
        this.count.set(count);
    }
    public int getId() {
        return id.get();
    }
    public void setId(int id) {
        this.count.set(id);
    }
    public String getContr() { return contrName.get(); }
    public void setContr(String contr) { this.contrName.set(contr); }
}
```

Product.java:

```
public class Product {
    private int id;
    private String name;
    private double cost;
    private int type;
    public Product(int id, String name, double cost, int type) {  this.id = id;
        this.name = name;
        this.cost = cost;
        this.type = type;
    }
    public int getId() {
```

```
        return id;
    }
    public String getName() {
        return name;
    }
    public double getCost() {
        return cost;
    }
    public int getType() {
        return type;
    }
    public void setId(int id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setCost(double cost) {
        this.cost = cost;
    }
    public void setType(int type) {
        this.type = type;
    }
}
```

Sample.fxml:

```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.text.Font?>
<VBox maxHeight="-Infinity" minHeight="400.0" minWidth="600.0" prefHeight="400.0" prefWidth="600.0"
xmlns="http://javafx.com/javafx/10.0.2-internal" xmlns:fx="http://javafx.com/fxml/1" fx:controller="Controller">
<children>
    <HBox maxWidth="1.7976931348623157E308" prefHeight="30.0">
        <children>
            <Button maxWidth="1.7976931348623157E308" minWidth="0.0"  mnemonicParsing="false"
onAction="#clickAddButton" text="Добавить запись"  HBox.hgrow="ALWAYS">
                <HBox.margin>
                    <Insets right="10.0" />
                </HBox.margin>
            </Button>
            <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false"
onAction="#clickUpdateButton" text="Изменить запись" HBox.hgrow="ALWAYS">  <HBox.margin>
                <Insets right="10.0" />
            </HBox.margin></Button>
            <Button maxWidth="1.7976931348623157E308" mnemonicParsing="false"
onAction="#clickDeleteButton" text="Удалить запись" HBox.hgrow="ALWAYS" />  </children>
    </HBox>
    <CheckBox fx:id="checkBox" alignment="CENTER" maxWidth="1.7976931348623157E308"
mnemonicParsing="false" onAction="#clickCheckBox" text="сортировать по продукту">  <font>
        <Font name="Bell MT" size="14.0" />
    </font>
```

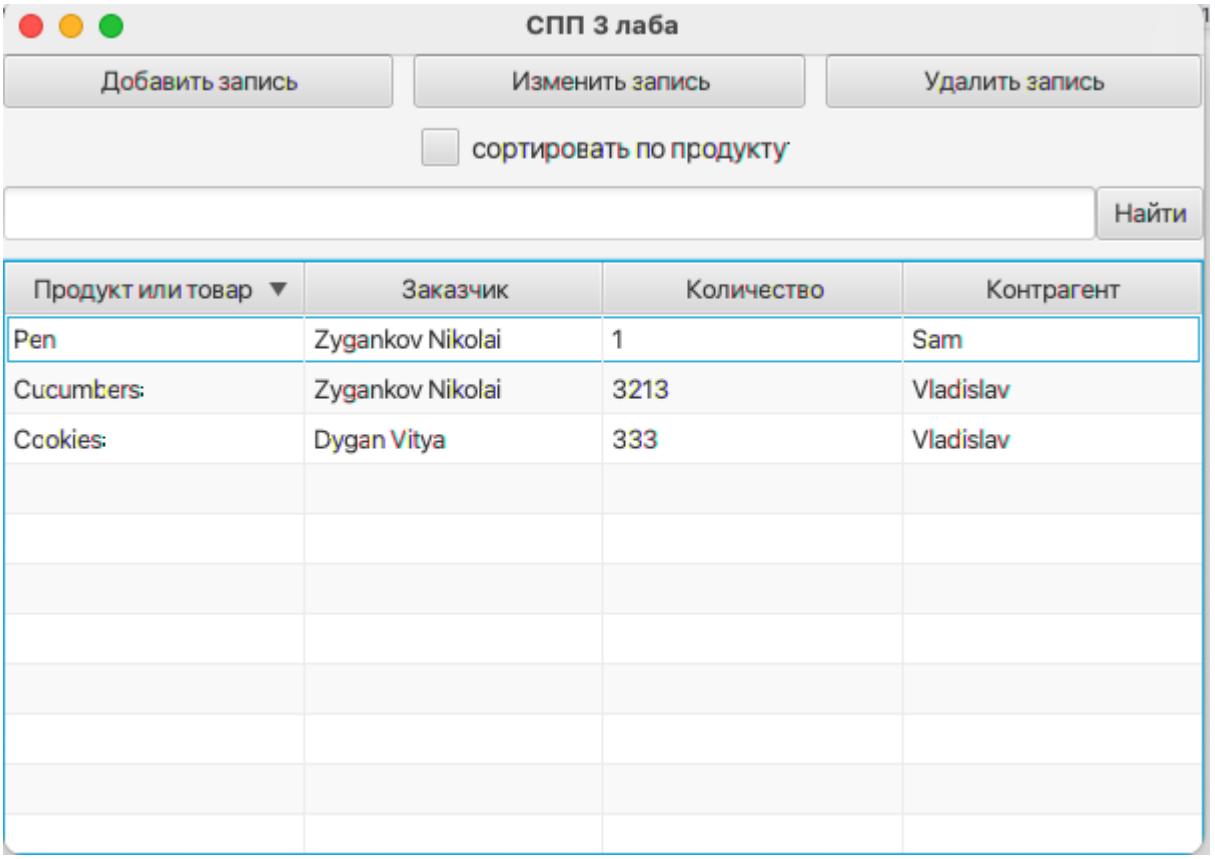
```
      <VBox.margin>
         <Insets top="10.0" />
      </VBox.margin>
   </CheckBox>
   <HBox prefHeight="30.0">
      <children>
         <TextField fx:id="textArea" HBox.hgrow="ALWAYS">
            <HBox.margin>
               <Insets />
            </HBox.margin>
         </TextField>
         <Button fx:id="searchButton" alignment="CENTER" mnemonicParsing="false"
onAction="#clickSearchButton" text="Найти" />
      </children>
      <VBox.margin>
         <Insets bottom="10.0" top="10.0" />
      </VBox.margin></HBox>
   <TableView fx:id="table" maxHeight="1.7976931348623157E308">  <columns>
      <TableColumn fx:id="tableItemProduct" text="Продукт или товар" />  <TableColumn fx:id="tableItemClient"
text="Заказчик" />  <TableColumn fx:id="tableItemCount" text="Количество" /> <TableColumn
fx:id="tableItemContr" text="Контрагент" /> <TableColumn fx:id="tableItemId" text="id" visible="false" />
</columns>
      <columnResizePolicy>
         <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />  </columnResizePolicy>
   </TableView>
</children>
</VBox>
```

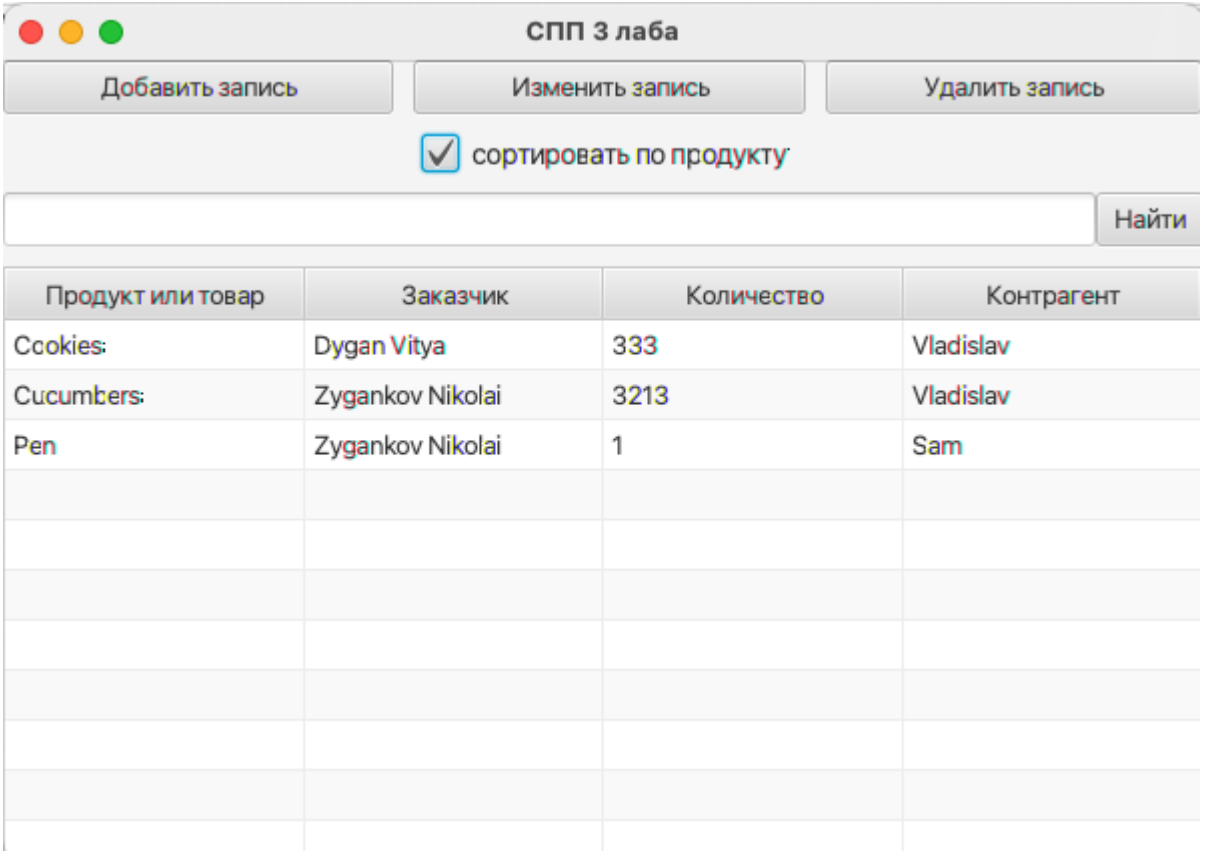
adder.fxml:

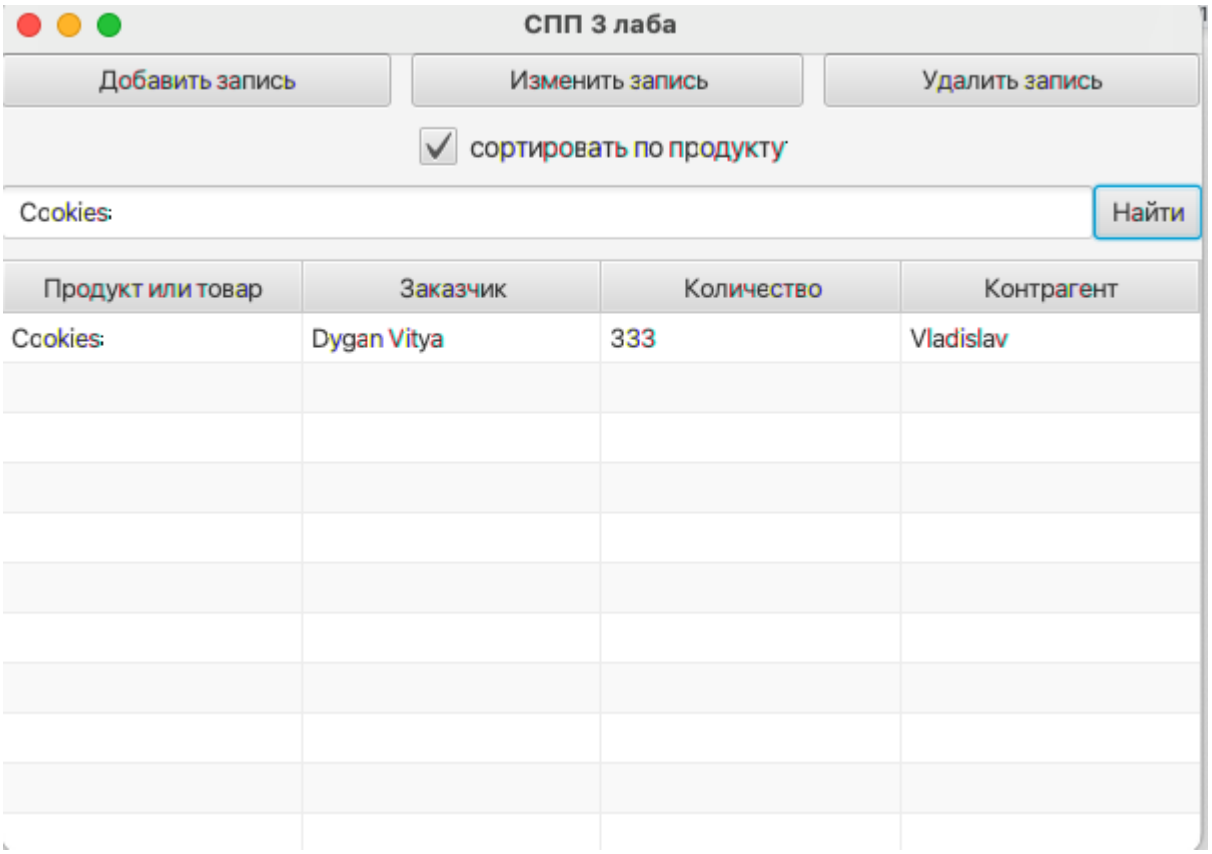
```
<?xml version="1.0" encoding="UTF-8"?>

<?import java.lang.*?>
<?import java.util.*?>
<?import javafx.scene.*?>
<?import javafx.scene.control.*?>
<?import javafx.scene.layout.*?>

<?import javafx.geometry.Insets?>
<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity"
prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/10.0.2- internal"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="AdderController">  <children>
   <ComboBox fx:id="comboBoxProducts" maxWidth="1.7976931348623157E308"  promptText="Выберите
продукт или товар">
      <VBox.margin>
         <Insets bottom="30.0" />
      </VBox.margin>
   </ComboBox>
   <ComboBox fx:id="comboBoxClients" maxWidth="1.7976931348623157E308"  promptText="Выберите
заказчика">
      <VBox.margin>
         <Insets bottom="30.0" />
      </VBox.margin>
   </ComboBox>
   <ComboBox fx:id="comboBoxContr" maxWidth="1.7976931348623157E308"  promptText="Выберите
контрагента">
      <VBox.margin>
         <Insets bottom="30.0" />
      </VBox.margin>
```

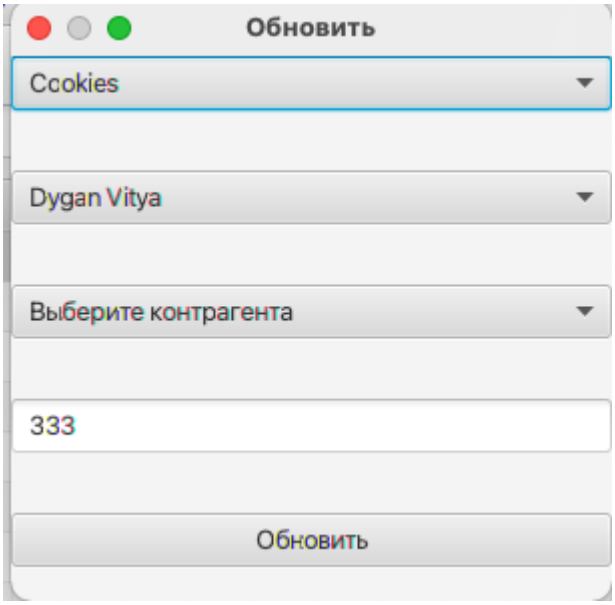
```
    </ComboBox>
    <TextField fx:id="textAreaCount" maxWidth="1.7976931348623157E308"  promptText="Введите количество">
      <VBox.margin>
        <Insets bottom="30.0" />
      </VBox.margin>
    </TextField>
    <Button fx:id="buttonAdd" maxWidth="1.7976931348623157E308" mnemonicParsing="false"
onAction="#addButtonClick" text="Добавить" />
</children>
</VBox>
```

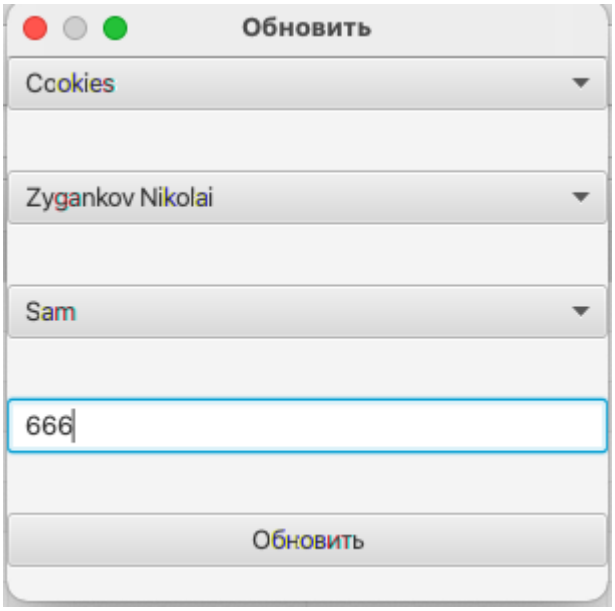
Результат выполнения:

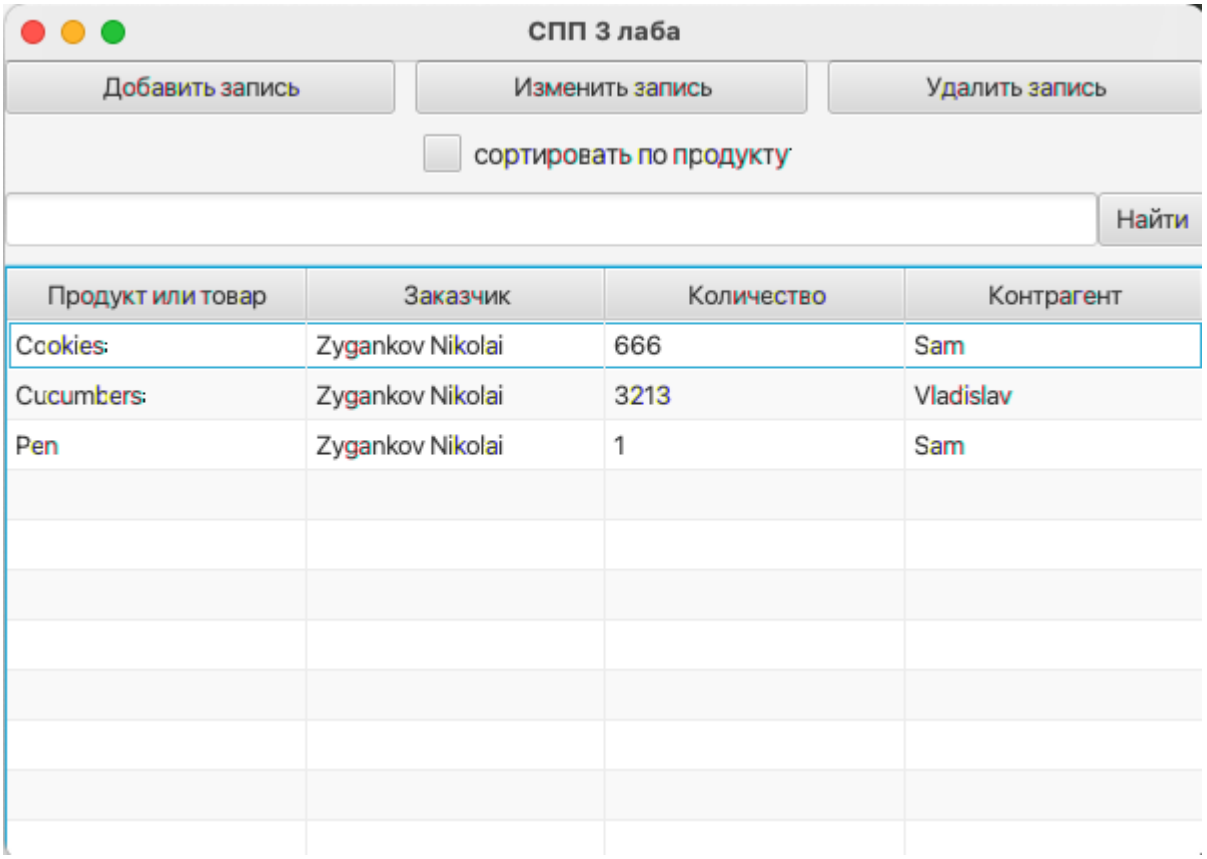


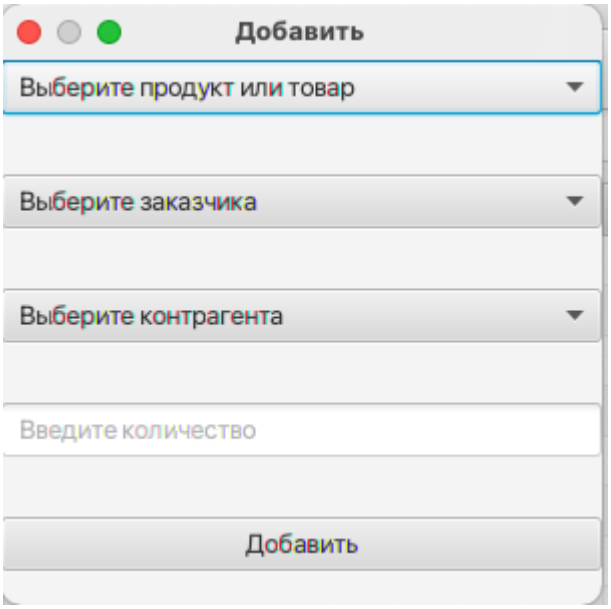

СПП 3 лаба
Добавить запись
Изменить запись
Удалить запись
сортировать по продукту
Найти
Продукт или товар
Заказчик
Количество
Контрагент
Ccokies:
Dygan Vitya
333
Vladislav
Cucumbers:
Zygankov Nikolai
3213
Vladislav
Pen
Zygankov Nikolai
1
Sam

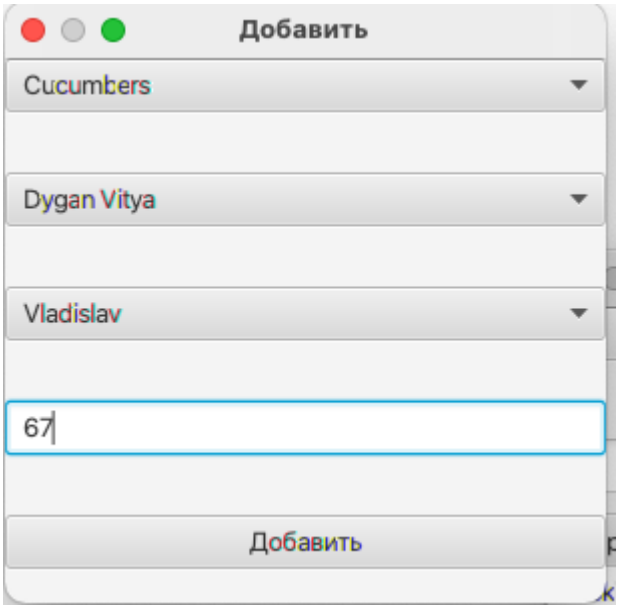

СПП 3 лаба
Добавить запись
Изменить запись
Удалить запись
сортировать по продукту
Ccokies:
Найти
Продукт или товар
Заказчик
Количество
Контрагент
Ccokies:
Dygan Vitya
333
Vladislav

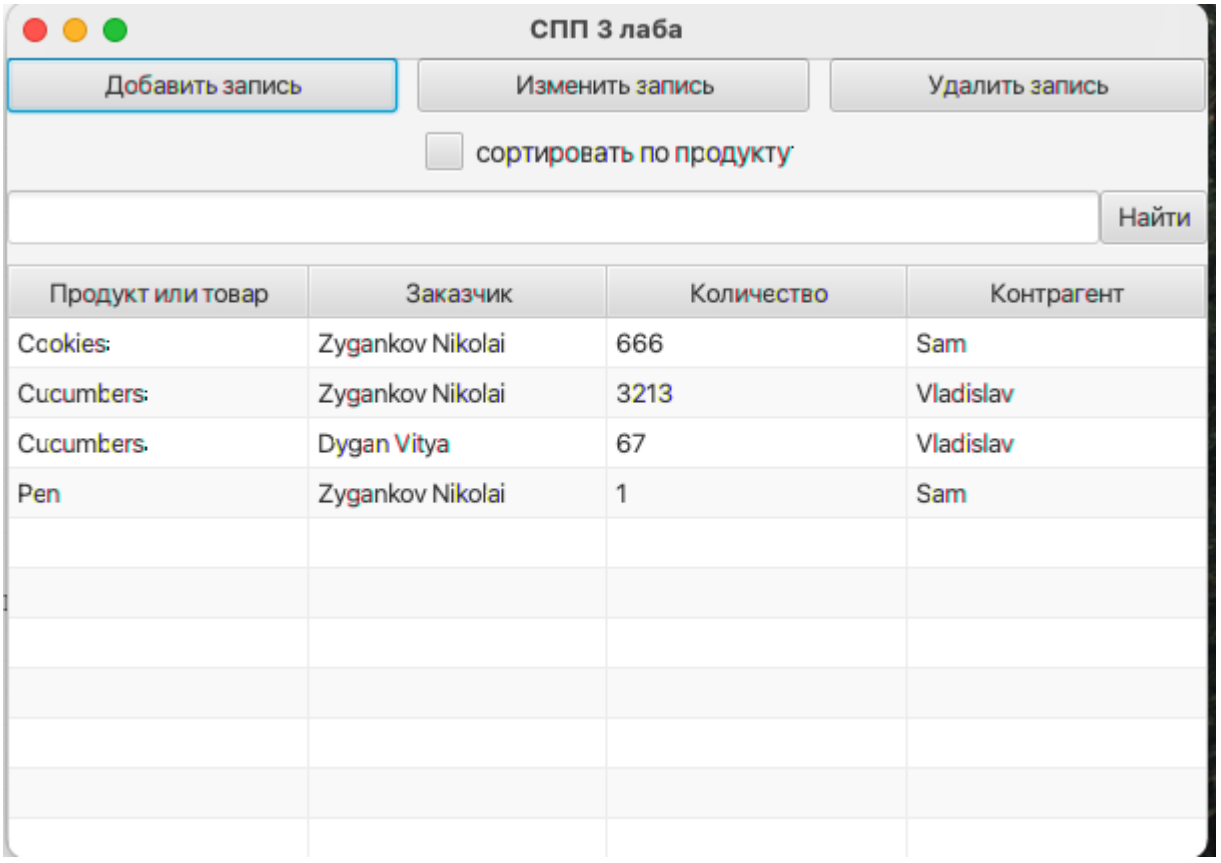

Обновить
Ccokies
Dygan Vitya
Выберите контрагента
333
Обновить

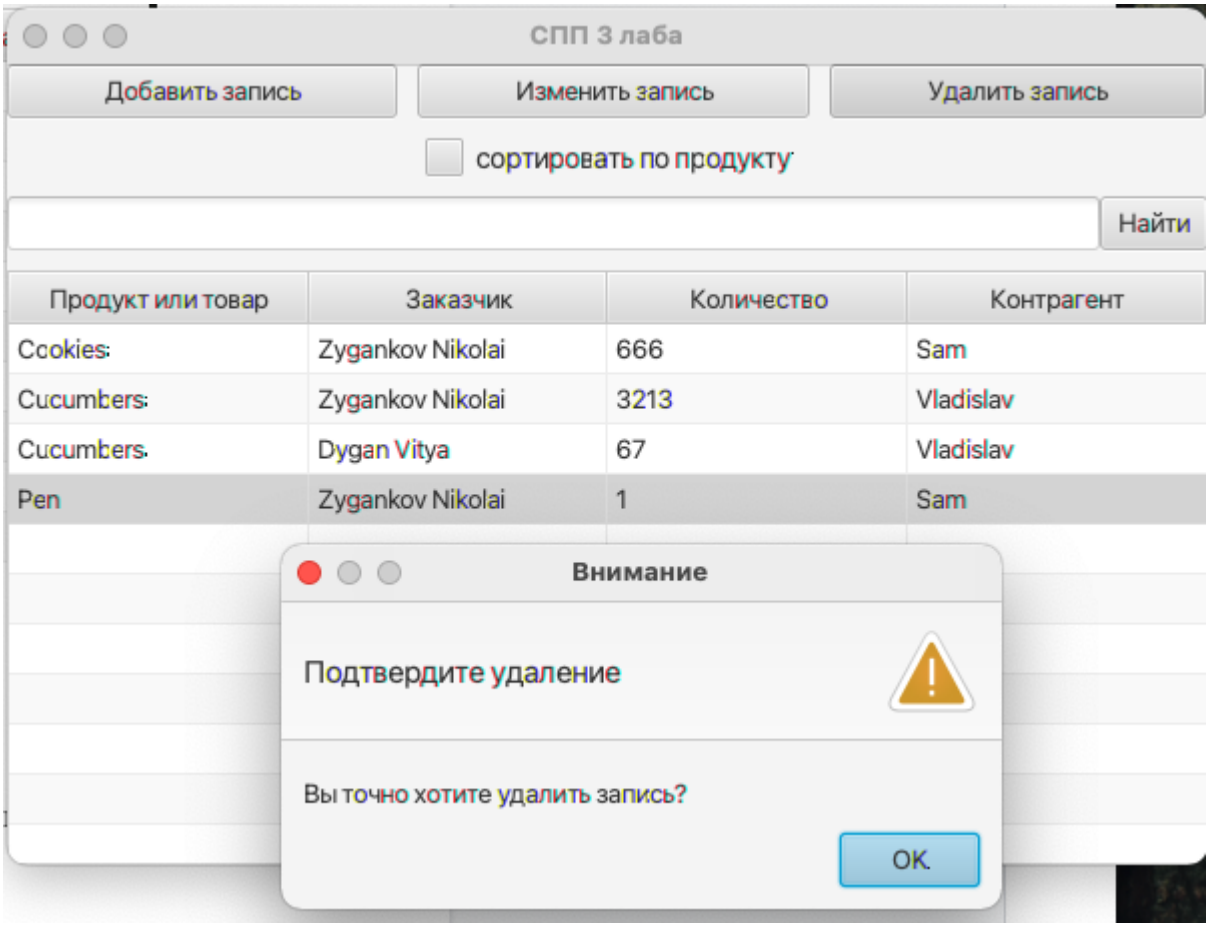

Обновить
Ccokies
Zygankov Nikolai
Sam
666
Обновить

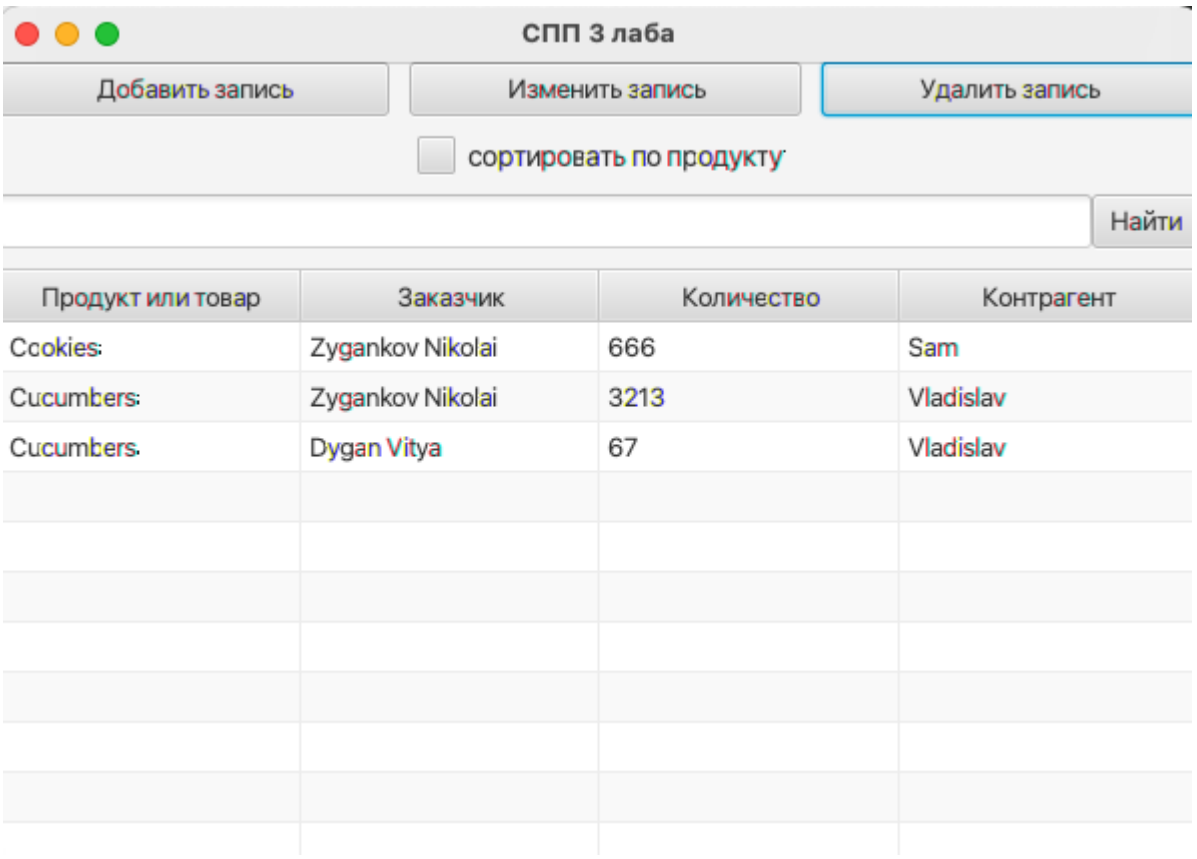

СПП 3 лаба
Добавить запись
Изменить запись
Удалить запись
сортировать по продукту
Найти
Продукт или товар
Заказчик
Количество
Контрагент
Ccokies:
Zygankov Nikolai
666
Sam
Cucumbers:
Zygankov Nikolai
3213
Vladislav
Pen
Zygankov Nikolai
1
Sam


Добавить
Выберите продукт или товар
Выберите заказчика
Выберите контрагента
Введите количество
Добавить


Добавить
Cucumbers
Dygan Vitya
Vladislav
67
Добавить


СПП 3 лаба
Добавить запись
Изменить запись
Удалить запись
сортировать по продукту
Найти
Продукт или товар
Заказчик
Количество
Контрагент
Ccokies
Zygankov Nikolai
666
Sam
Cucumbers
Zygankov Nikolai
3213
Vladislav
Cucumbers
Dygan Vitya
67
Vladislav
Pen
Zygankov Nikolai
1
Sam





Вывод: приобрел практические навыки разработки многооконных приложений на JavaFX для работы с базами данных.