****

**VILNIAUS KOLEGIJA**

**ELEKTRONIKOS IR INFORMATIKOS FAKULTETAS**

**Programinės įrangos katedra**

**Antroji programavimo praktika**

Ataskaita

**Programų sistemos (PI20S grupė)**

STUDENTAS M. Marcinkevič

LEKTORIUS V. Liubinas

Vilnius

2021

**Turinys**

[ĮVADAS 3](#_Toc61363526)

[1. MySQL WORKBENCH DUOMENŲ BAZĖ 4](#_Toc61363527)

[2. Programos klasės 6](#_Toc61363528)

[2.1. „RegisterFromController“ 6](#_Toc61363529)

[2.2. „LogInController“ 8](#_Toc61363530)

[2.3. „MainWindowController “ 8](#_Toc61363531)

[2.4. „AdminWindow “ 11](#_Toc61363532)

[2.5. „Wishlist “ 13](#_Toc61363533)

[2.6. „ItemPublicViewWishlist “ 14](#_Toc61363534)

[2.7. „UserListViewController“ 15](#_Toc61363535)

[2.8. „ItemsRepository“ 15](#_Toc61363536)

[2.9. „UserRep“ 17](#_Toc61363537)

[3. Naudotojo instrukcija 20](#_Toc61363538)

[3.1. Registracija ir prisijungimas 20](#_Toc61363539)

[3.2. Prekės komentavimas 22](#_Toc61363540)

[3.3. Prekės įsiminimas 23](#_Toc61363541)

[3.4. Administratoriaus paskyros kūrimas ir valdymas 24](#_Toc61363542)

[IŠVADOS 27](#_Toc61363543)

# ĮVADAS

**Darbo tikslas:** Papildyti trečios užduoties programą - elektroninės prekybos programą taip, kad anoniminis naudotojas galėtų tik peržiūrėti turinį, registruotas naudotojas galėtų peržiūrėti, komentuoti ir įsiminti patinkančią prekę, administratorius galėtų kurti prekių kategorijas bei įkelti ar šalinti atitinkamas prekes.

**Darbo uždaviniai:**

* Sukurti naują duomenų bazę;
* Perkelti užduoties saugomą informaciją į sukurtą duomenų bazę;
* Sukurti funkcijas administratoriui, kurios leistų: sukurti prekių kategoriją, pridėti prekę, šalinti prekę, įsiminti prekę, filtruoti prekes pagal kategorijas, komentuoti, šalinti komentarus, peržiūrėti įsimintų sąrašą, šalinti prekę iš įsimintų sąrašo;
* Sukurti funkcijas registruotam naudotojui, kurios leistų: įsiminti prekes, komentuoti, peržiūrėti įsimintų sąrašą, šalinti prekę iš įsimintų sąrašo;
* Sukurti funkciją neprisiregistravusiems naudotojams, kuri leistų peržiūrėti prekių sąrašą ir filtruoti jas pagal kategorijas.

**Darbo priemonės:**

* Java Intellij
* MySQL Workbench

# MySQL WORKBENCH DUOMENŲ BAZĖ

Naudojantis MySQL Workbench buvo sukurta duomenų bazė, kuri susideda iš 5 lentelių:

* „User“: Saugo registruoto vartotojo duomenis (ID, Vardą, Pavardę, Gimimo datą, prisijungimo vardą, slaptažodį)
* „Comments“: Saugo vartotojų komentarus apie prekę (ID, Komentaras, Vartotojo vardas, aprašymas)
* „Items“: Saugo el. parduotuvės prekes (ID, Prekės pavadinimas, Kaina, Aprašymas, Paveikslėlio pavadinimas, Kategorijos ID)
* „Wishlist“: Saugo registruotų vartotojų įsimintas prekes (ID, Prekės pavadinimas, Kaina, Aprašymas, Paveikslėlio pavadinimas, Vartotojo vardas)
* „Catagories“ Saugo kategorijas (ID, Kategorijos pavadinimas)

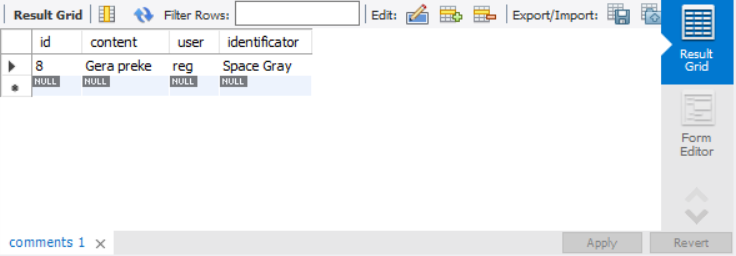
**Duomenų bazė:**

* **User**

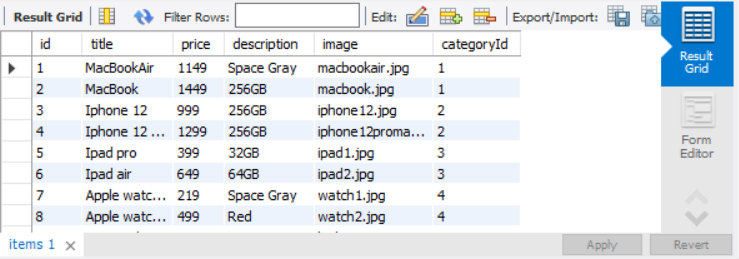
Graphical user interface, text, application, email

Description automatically generated

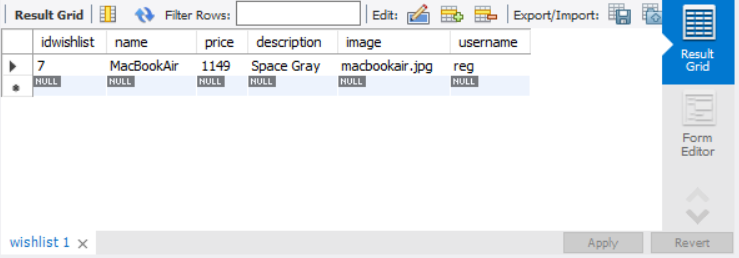
* **Comments**



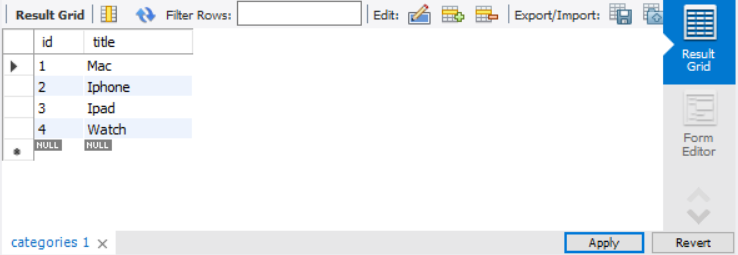
* **Items**



* **Wishlist**



* **Categories**



# Programos klasės

## „RegisterFromController“

**Kodo fragmentas:**

public class RegisterFormController {  
 private UserRep repository= new UserRep();  
 private Boolean admin;  
  
 @FXML  
 TextField NameInput;  
 @FXML  
 TextField PassInput;  
 @FXML  
 TextField SurnameInput;  
 @FXML  
 TextField DateInput;  
 @FXML  
 TextField NickInput;  
 @FXML  
 Label LabelPopUp;  
 @FXML  
 Label LabelPopUp1;  
 @FXML  
 Label LabelPopUp2;  
 @FXML  
 Label NickLabel;  
 @FXML  
 Label PassLabel;  
 @FXML  
 Button buttonA;  
 @FXML  
 Button buttonU;  
  
  
 public RegisterFormController() throws Exception {  
 }  
  
 public void User() throws Exception {  
 admin = false;  
 Submitted();  
 }  
 public void Admin() throws Exception {  
 admin = true;  
 Submitted();  
 }  
 public void Submitted() throws Exception {  
 Admin temp = new Admin();  
 try {  
 temp.setName(NameInput.getText());  
 }catch (Exception exc){  
 LabelPopUp.setText(exc.getMessage());  
 }  
  
 try {  
 temp.setSurname(SurnameInput.getText());  
 }catch (Exception exc){  
 LabelPopUp2.setText(exc.getMessage());  
 }  
  
 try {  
 temp.setBirthDate(DateInput.getText());  
 }catch (Exception exc){  
 LabelPopUp1.setText(exc.getMessage());  
 }  
  
 try {  
 temp.setNickName(NickInput.getText());  
 }catch (Exception exc){  
 NickLabel.setText(exc.getMessage());  
 }  
  
 try {  
 temp.setPassword(PassInput.getText());  
 }catch (Exception exc){  
 PassLabel.setText(exc.getMessage());  
 }  
  
 Stage stage;  
 if (admin) {  
 repository.RegisterAdmin(NameInput.getText(), SurnameInput.getText(), DateInput.getText(), NickInput.getText(), PassInput.getText());  
 stage = (Stage) buttonA.getScene().getWindow();  
 stage.close();  
 }  
  
 if (!admin) {  
 try{  
 Connection conn;  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 conn= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop","root","root");  
 Statement statement=conn.createStatement();  
 String query = "INSERT INTO eshop.user (name, surname, birthday, nick, password)"+ " values (?, ?, ?, ?, ?)";  
 PreparedStatement preparedStmt = conn.prepareStatement(query);  
 preparedStmt.setString(1, NameInput.getText());  
 preparedStmt.setString(2, SurnameInput.getText());  
 preparedStmt.setString(3, DateInput.getText());  
 preparedStmt.setString(4, NickInput.getText());  
 preparedStmt.setString(5,PassInput.getText());  
 preparedStmt.execute();  
 int rowsaffected= statement.executeUpdate(query);  
 System.*out*.println("Rows effected "+ rowsaffected);  
 }  
 catch(Exception exc) {  
 System.*out*.println(exc.getMessage());}  
 repository.Register(NameInput.getText(), SurnameInput.getText(), DateInput.getText(), NickInput.getText(), PassInput.getText());  
 stage = (Stage) buttonU.getScene().getWindow();  
 stage.close();  
 }  
  
 }  
 }

**Klasės paskirtis:**

Suteikia galimybę prisiregistruoti prie sistemos, registruoto vartotojo duomenis siunčia į duomenų bazę.

## „LogInController“

**Kodo fragmentas:**

public class LogInController {  
 private UserRep repository= new UserRep();  
 @FXML  
 TextField NickInputCheck;  
 @FXML  
 TextField PassInputCheck;  
  
 public LogInController() throws Exception {  
 }  
  
 public void Logged(){  
 String Nick = NickInputCheck.getText();  
 String Pass = PassInputCheck.getText();  
 System.*out*.println("LOGIN - > " + Nick);  
 System.*out*.println("LOGIN - > " +Pass);  
  
 try {  
 repository.Login(Nick, Pass);  
 FXMLLoader fxmlLoader = new FXMLLoader();  
 fxmlLoader.setLocation(getClass().getResource("LoggedInWindow.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 Stage stage = new Stage();  
 stage.initModality(Modality.*APPLICATION\_MODAL*);  
 stage.setTitle("Logged window");  
 stage.setScene(scene);  
 stage.show();  
 }catch (Exception exc){  
 }  
 }  
}

**Klasės paskirtis:**

Suteikia galimybę prisijungti prie sistemos.

## „MainWindowController “

**Kodo fragmentas:**

public class MainWindowController implements Initializable {  
 private Connection conn;  
 private UserRep repository1= new UserRep();  
 @FXML  
 HBox header;  
 @FXML  
 Button ADMIN;  
 @FXML  
 VBox sideMenu;  
 @FXML  
 FlowPane content;  
 @FXML  
 Button wishlist;  
 @FXML  
 ListView<Item> LISTV = new ListView<>();  
  
 private ItemsRepository repository;  
 private List<Item> itemsList;  
 private List<Category> categoriesList;  
 private List<Item> itemsList2;  
  
 private static boolean *isAdmin* = false;  
  
 public MainWindowController() throws Exception {  
 repository = new ItemsRepository();  
 itemsList = repository.getItems();  
 categoriesList = repository.getCategories();  
 itemsList2=repository.getItems2();  
 }  
  
 public void displayCategories() {  
 for (Category c : categoriesList) {  
 Button categoryButton = new Button(c.getTitle());  
 categoryButton.setPrefWidth(Double.*MAX\_VALUE*);  
 CategoryButtonEventHandler handler = new CategoryButtonEventHandler(c, this);  
 categoryButton.setOnAction(handler);  
 sideMenu.getChildren().add(categoryButton);  
 }  
 for(Item i : itemsList2){  
 System.*out*.println(i.getTitle());  
 }  
 }  
  
 public void displayItemsOnStartup() throws IOException {  
 content.getChildren().clear();  
 for (Item item : itemsList) {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("ItemPublicView.fxml"));  
 Parent root = loader.load();  
 ItemPublicViewController controller = loader.getController();  
 controller.displayItem(item);  
  
 content.getChildren().add(root);  
  
 }  
 }  
  
 public void displayItems(List<Item> items) throws IOException {  
 content.getChildren().clear();  
 for (Item item : items) {  
 System.*out*.println(item.getId());  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("ItemPublicView.fxml"));  
 FXMLLoader loader1 = new FXMLLoader(getClass().getResource("ItemPublicViewWishList.fxml"));  
 Parent root = loader.load();  
 Parent root1 = loader1.load();  
 ItemPublicViewController controller = loader.getController();  
 ItemPublicViewWishList controller1= loader1.getController();  
 controller1.displayItem(item);  
 controller.displayItem(item);  
 content.getChildren().add(root);  
 }  
 }  
  
  
  
 public void onAdminClicked(ActionEvent actionEvent) {  
 if (repository1.GetClassName().equals("app.backend.models.Admin")) {  
 System.*out*.println("Admin");  
 }  
 *isAdmin* = !*isAdmin*;  
 }  
  
 public void AdminThings(ActionEvent actionEvent) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader();  
 fxmlLoader.setLocation(getClass().getResource("/app/ui/AdminWindow.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 Stage stage = new Stage();  
 stage.initModality(Modality.*APPLICATION\_MODAL*);  
 stage.setTitle("Logged window");  
 stage.setScene(scene);  
 stage.show();  
  
 }  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
  
 if (repository1.GetClassName().equals("app.backend.models.Admin")) {  
 ADMIN.setVisible(true);  
 wishlist.setVisible(false);  
 }  
 if (repository1.GetClassName().equals("app.backend.models.User")) {  
 ADMIN.setVisible(false);  
 wishlist.setVisible(true);  
 }  
 for(Item i : itemsList2){  
 System.*out*.println(i.getTitle());  
 }  
  
 }  
  
 public void WishlistOpen(ActionEvent actionEvent) throws IOException {  
 FXMLLoader fxmlLoader = new FXMLLoader();  
 fxmlLoader.setLocation(getClass().getResource("/app/ui/WishList.fxml"));  
 Scene scene = new Scene(fxmlLoader.load());  
 Stage stage = new Stage();  
 stage.setTitle("E-Shop Application");  
 stage.setScene(scene);  
 stage.show();}  
  
 }

**Klasės paskirtis:**

Suteikia galimybę naudotis pagrindinio lango funkcijomis.

## „AdminWindow “

**Kodo fragmentas:**

public class AdminWindow implements Initializable {  
 private ItemsRepository repository;  
 private List<Category> categoryList;  
 private Connection Conn;  
 @FXML  
 private ObservableList<String> categories = FXCollections.*observableArrayList*();  
 @FXML  
 private ObservableList<String> ForPoP = FXCollections.*observableArrayList*();  
 @FXML  
 ListView ListBox1;  
 @FXML  
 TextField TitleGet;  
 @FXML  
 TextField PriceGet;  
 @FXML  
 TextField DescriptionGet;  
 @FXML  
 TextField ImageGet;  
 @FXML  
 TextField CategoryId;  
 @FXML  
 TextField CategoyInput;  
 @FXML  
 ChoiceBox Boxie;  
  
 public void AddCategory(ActionEvent actionEvent) {  
 try{  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 Conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 Statement statement = Conn.createStatement();  
 String query = "INSERT INTO eshop.categories (title)" + " values ( ?)";  
 PreparedStatement preparedStmt = Conn.prepareStatement(query);  
 preparedStmt.setString(1, CategoyInput.getText());  
 if(!CategoyInput.getText().equals(null)){  
 Label Text = new Label();  
 Pane pane= new Pane();  
 Text.setText("Added");  
 Stage stage = new Stage();  
 pane.getChildren().add(Text);  
 Scene scene = new Scene(pane, 200 ,200);  
 stage.setScene(scene);  
 stage.show();  
 preparedStmt.execute();}  
  
 }  
 catch(Exception exc) {  
 System.*out*.println(exc.getMessage());}  
  
 }  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
 try {  
 repository = new ItemsRepository();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 categoryList = repository.getCategories();  
 for(Category c: categoryList){  
 System.*out*.println(c.getId());  
 ForPoP.add(String.*valueOf*((c.getId())));  
 }  
  
 Boxie.setItems(ForPoP);  
  
 ListBox1.setItems(categories);  
  
  
 try {  
 Conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 System.*out*.println("All good connected");  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 System.*out*.println("Something went wrong.");  
 }  
  
 Statement stmt = null;  
 ResultSet rs = null;  
 try {  
 stmt = Conn.createStatement();  
 rs = stmt.executeQuery("SELECT id,title FROM eshop.categories");  
  
 while (rs.next()) {  
 categories.add("Category id." + ( rs.getInt("id")) + " " + "Category title - " + " "+rs.getString("title"));  
 }  
 } catch (SQLException e) {  
 }}  
  
 public void AddItem(ActionEvent actionEvent) {  
 try{  
  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 Conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 Statement statement = Conn.createStatement();  
 String query = "INSERT INTO eshop.items (title,price,description,image,categoryId)" + " values ( ?,?,?,?,?)";  
 PreparedStatement preparedStmt = Conn.prepareStatement(query);  
 preparedStmt.setString(1, TitleGet.getText());  
 preparedStmt.setString(2, PriceGet.getText());  
 preparedStmt.setString(3, DescriptionGet.getText());  
 preparedStmt.setString(4, ImageGet.getText());  
 preparedStmt.setString(5, ((String) Boxie.getSelectionModel().getSelectedItem()));  
  
 if(!TitleGet.getText().equals(null)){  
 Label Text = new Label();  
 Pane pane= new Pane();  
 Text.setText("Added");  
 Stage stage = new Stage();  
 pane.getChildren().add(Text);  
 Scene scene = new Scene(pane, 200 ,200);  
 stage.setScene(scene);  
 stage.show();  
 preparedStmt.execute();}  
  
 }  
 catch(Exception exc) {  
 System.*out*.println(exc.getMessage());}  
 }  
}

**Klasės paskirtis:**

Suteikia galimybę naudotis administratoriaus lango funkcijomis, pridėti arba pašalinti kategoriją bei prekę.

## „Wishlist “

**Kodo fragmentas:**

public class WishList implements Initializable {  
 private Connection Conn;  
private UserRep repo= new UserRep();  
private ItemsRepository repository = new ItemsRepository();  
private List<Item> itemList2;  
@FXML  
FlowPane Content;  
  
 public WishList() throws Exception {  
 }  
  
 @Override  
 public void initialize(URL url, ResourceBundle resourceBundle) {  
  
  
 itemList2=repository.getItems2();  
 for (Item item : itemList2) {  
 FXMLLoader loader = new FXMLLoader(getClass().getResource("/app/ui/ItemPublicViewWishList.fxml"));  
 Parent root = null;  
 try {  
 root = loader.load();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 ItemPublicViewWishList controller = loader.getController();  
 controller.displayItem(item);  
 Content.getChildren().add(root);  
  
 }  
 }  
  
  
}

**Klasės paskirtis:**

Registruoto vartotojo įsimintinų prekių sąrašas

## „ItemPublicViewWishlist “

**Kodo fragmentas:**

public class ItemPublicViewWishList {  
 private UserRep repository1 = new UserRep();  
 private Connection Conn;  
 @FXML  
 ImageView image;  
 @FXML  
 Label titleLabel;  
 @FXML  
 Label descriptionLabel;  
 @FXML  
 Label priceLabel;  
 @FXML  
 Button Add;  
 @FXML  
 Button Remove;  
 @FXML  
 AnchorPane Anchor;  
  
 public ItemPublicViewWishList() throws Exception {  
 }  
  
 public void displayItem(Item item) {  
 titleLabel.setText(item.getTitle());  
 descriptionLabel.setText(item.getDescription());  
 priceLabel.setText(String.*valueOf*(item.getPrice()));  
 File file = new File("pics/" + item.getImage());  
 image.setImage(new Image(file.toURI().toString()));  
 }  
  
 public void DeleteItem(ActionEvent actionEvent) {  
  
 try {  
 Conn= DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 Statement stmt= Conn.createStatement();  
 String query1 = "DELETE FROM eshop.wishlist WHERE name= '"+ titleLabel.getText() +"'";  
  
 int rowsaffected= stmt.executeUpdate(query1);  
 System.*out*.println("Rows effected "+ rowsaffected);  
  
  
 }catch (Exception exc){  
 exc.printStackTrace();  
  
 }  
  
 }  
}

**Klasės paskirtis:**

Prideda arba pašalina prekę iš įsimintinų prekių sąrašo

## „UserListViewController“

**Kodo fragmentas:**

public class UserListViewController {  
 private UserRep repository = new UserRep();  
  
 @FXML  
 ListView<User> Listbox1 = new ListView<>();  
 @FXML  
 Label TTT;  
 @FXML  
 Button button;  
 @FXML  
 protected void initialize(){  
 List<User> userList= repository.GetUsers();  
 Listbox1.getItems().addAll(userList);  
 }  
 public UserListViewController() throws Exception {  
 }  
  
 public void DeleteUser() {  
 int i = Listbox1.getSelectionModel().getSelectedIndex();  
 try {  
 repository.RemoveUser(i);  
 Listbox1.getItems().remove(i);  
 }catch (Exception exc){  
 TTT.setText(exc.getMessage());  
 }  
 }  
}

**Klasės paskirtis:**

Suteikia galimybe peržiūrėti vartotojus ir juos ištrinti

## „ItemsRepository“

**Kodo fragmentas:**

public class ItemsRepository {  
  
 private Connection conn;  
private UserRep repo = new UserRep();  
 public ItemsRepository() throws Exception {  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 } catch (Exception exc) {  
 System.*out*.println(exc.getMessage());  
 }  
 }  
  
 public List<Item> getItems() {  
 List<Item> itemsList = new ArrayList<Item>();  
 try {  
 String sql = "select id, title, price, description, image from items";  
 PreparedStatement ps = conn.prepareStatement(sql);  
 ResultSet rs = ps.executeQuery();  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String title = rs.getString("title");  
 double price = rs.getDouble("price");  
 String description = rs.getString("description");  
 String image = rs.getString("image");  
 itemsList.add(new Item(id, title, description, price, image));  
 }  
 } catch (Exception exc) {  
 System.*out*.println(exc.getMessage());  
 } finally {  
 return itemsList;  
 }  
 }  
  
 public List<Category> getCategories() {  
 List<Category> categoriesList = new ArrayList<Category>();  
 try {  
 String sql = "select id, title from categories";  
 PreparedStatement ps = conn.prepareStatement(sql);  
  
 ResultSet rs = ps.executeQuery();  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String title = rs.getString("title");  
 categoriesList.add(new Category(id, title));  
 }  
 } catch (Exception exc) {  
 System.*out*.println(exc.getMessage());  
 } finally {  
 for (Category c : categoriesList) {  
 c.setItems(getItems(c.getId()));  
 }  
 return categoriesList;  
 }  
 }  
  
 private List<Item> getItems(int categoryId) {  
 List<Item> itemsList = new ArrayList<Item>();  
 try {  
 String sql = "select id, title, price, description, image from items " +  
 "where categoryId=?";  
 PreparedStatement ps = conn.prepareStatement(sql);  
 ps.setInt(1, categoryId);  
  
 ResultSet rs = ps.executeQuery();  
 while (rs.next()) {  
 int id = rs.getInt("id");  
 String title = rs.getString("title");  
 double price = rs.getDouble("price");  
 String description = rs.getString("description");  
 String image = rs.getString("image");  
 itemsList.add(new Item(id, title, description, price, image));  
 }  
 } catch (Exception exc) {  
 System.*out*.println(exc.getMessage());  
 } finally {  
 return itemsList;  
 }  
 }  
 public List<Item> getItems2() {  
 List<Item> itemsList2 = new ArrayList<Item>();  
 try {  
 Class.*forName*("com.mysql.cj.jdbc.Driver");  
 conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 String sql = "select idwishlist, name, price, description, image from wishlist " +  
 "where username=?";  
 PreparedStatement ps = conn.prepareStatement(sql);  
 ps.setString(1, repo.GetLoggedInUser().GetNickname());  
 ResultSet rs = ps.executeQuery();  
 while (rs.next()) {  
 int id = rs.getInt("idwishlist");  
 String title = rs.getString("name");  
 double price = rs.getDouble("price");  
 String description = rs.getString("description");  
 String image = rs.getString("image");  
 itemsList2.add(new Item(id, title, description, price, image));  
 }  
 } catch (Exception exc) {  
 System.*out*.println(exc.getMessage());  
 } finally {  
 return itemsList2;  
 }  
 }  
}

**Klasės paskirtis:**

Talpina prekes į duomenų bazę.

## „UserRep“

**Kodo fragmentas:**

public class UserRep {  
 private static List<User> *userList*;  
 private static User *loggedInUser*;  
private Connection Conn;  
 public UserRep() throws Exception{  
 try {  
 Conn = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/eshop", "root", "root");  
 System.*out*.println("All good connected");  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 System.*out*.println("Something went wrong.");  
 }  
  
 if (*userList* == null) {  
 *userList* = new ArrayList<>();  
 try {  
 Statement stmt;  
 ResultSet rs;  
 stmt = Conn.createStatement();  
 rs = stmt.executeQuery("SELECT \* FROM eshop.user");  
 while (rs.next()) {  
  
 String name1= String.*valueOf*(rs.getString("name"));  
 String surename1= String.*valueOf*(rs.getString("surname"));  
 String birth= rs.getString("birthday");  
 String nick1= rs.getString("nick");  
 String pass1= rs.getString("password");  
 *userList*.add(new User(name1,surename1,birth,nick1,pass1));  
  
 }  
 } catch (SQLException exc) {  
 exc.printStackTrace();  
 }  
 *userList*.add(new Admin("Deivid", "Marcinkevic", "2000-02-23", "admin", "admin"));  
 for (User user : *userList*) {  
 System.*out*.println(user);  
 }  
 }  
 }  
  
 public void Register(String name, String surname, String birthdate, String nickname, String password) throws Exception{  
 *userList*.add(new User(name, surname, birthdate, nickname, password));  
 *userList*.stream().forEach(System.*out*::println);  
 }  
 public void RegisterAdmin(String name, String surname, String birthdate, String nickname, String password) throws Exception{  
 *userList*.add(new Admin(name, surname, birthdate, nickname, password));  
 *userList*.stream().forEach(System.*out*::println);  
  
 }  
  
 public void Login(String nickName, String password) throws Exception {  
 *loggedInUser* = null;  
 for (User u : *userList*){  
  
 if(u.GetNickname().equals(nickName) && u.GetPass().equals(password)){  
 *loggedInUser*=u;  
 System.*out*.println(u);  
 break;  
 }  
 }  
 if (*loggedInUser* == null){  
 throw new Exception("wrong credentials");  
 }  
 }  
  
 public void RemoveUser(int i) throws Exception{  
 if (i == -1)  
 throw new Exception("nothing was selected");  
  
 if (*userList*.get(i).equals(*loggedInUser*)){  
 throw new Exception("cannot delete your self");  
 }  
 *userList*.remove(i);  
 }  
  
 public List<User> GetUsers() {  
 return *userList*;  
 }  
  
 public String GetClassName(){  
 if(*loggedInUser*==null){  
 return "Out";  
 }  
 else{  
 return *loggedInUser*.getClassname();  
 }  
 }  
 public User GetLoggedInUser ()  
 {  
 return *loggedInUser*;  
 }  
 public void LogOut()  
 {  
 *loggedInUser* = null;  
 }  
public String SetClassName(String s){  
 if(*loggedInUser*==null){  
 return "Out";  
 }  
 else{  
 return *loggedInUser*.setClassnname(s);  
 }  
}  
  
}

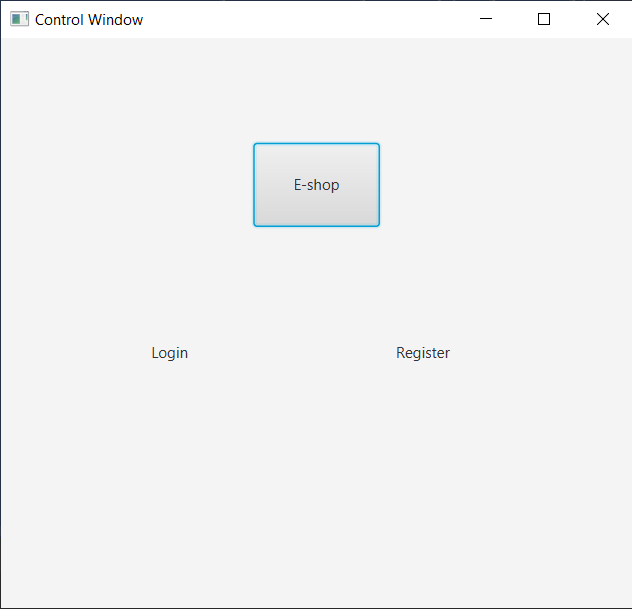
**Klasės paskirtis:**

Valdyti vartotojus, pridėti administratoriu

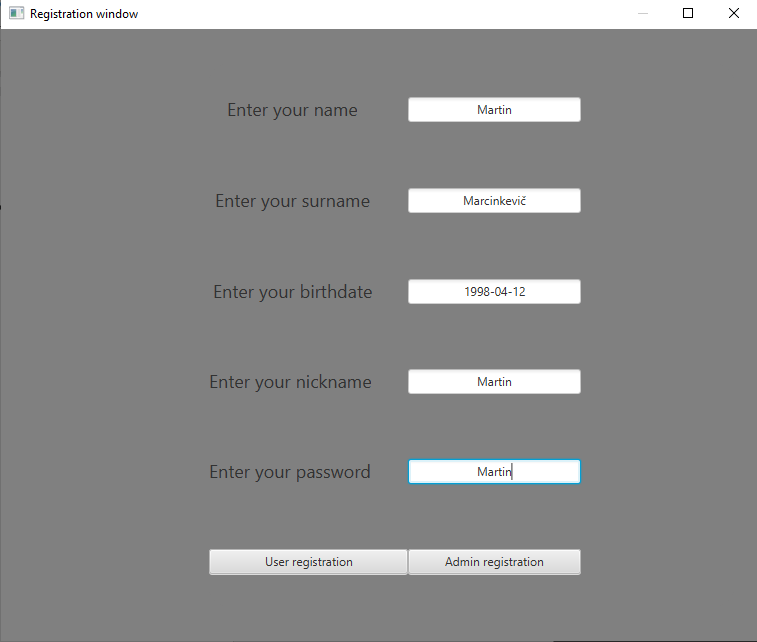
# Naudotojo instrukcija

## Registracija ir prisijungimas

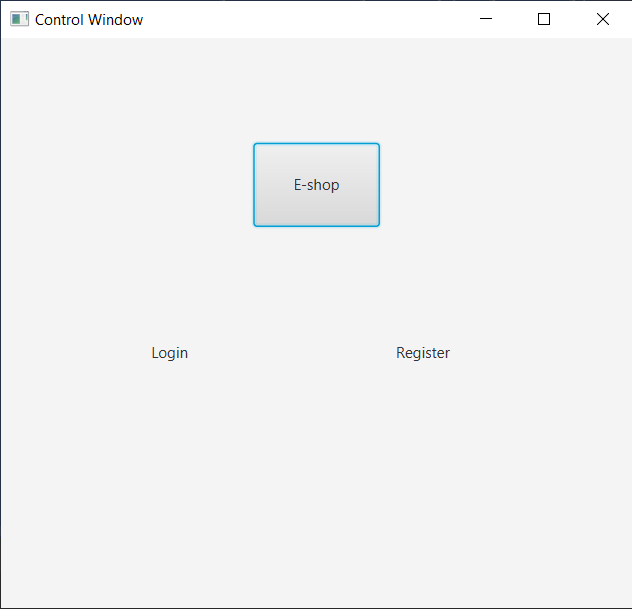
* Spaudžiame „register“ mygtuką.

****

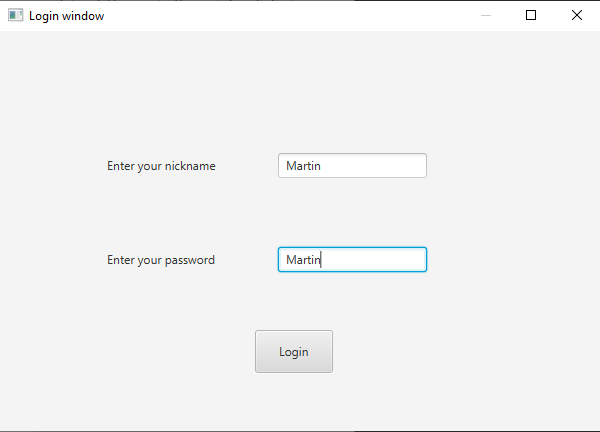
* Užpildome tuščius laukus ir spaudžiame „User registration“



* Spaudžiame „login“

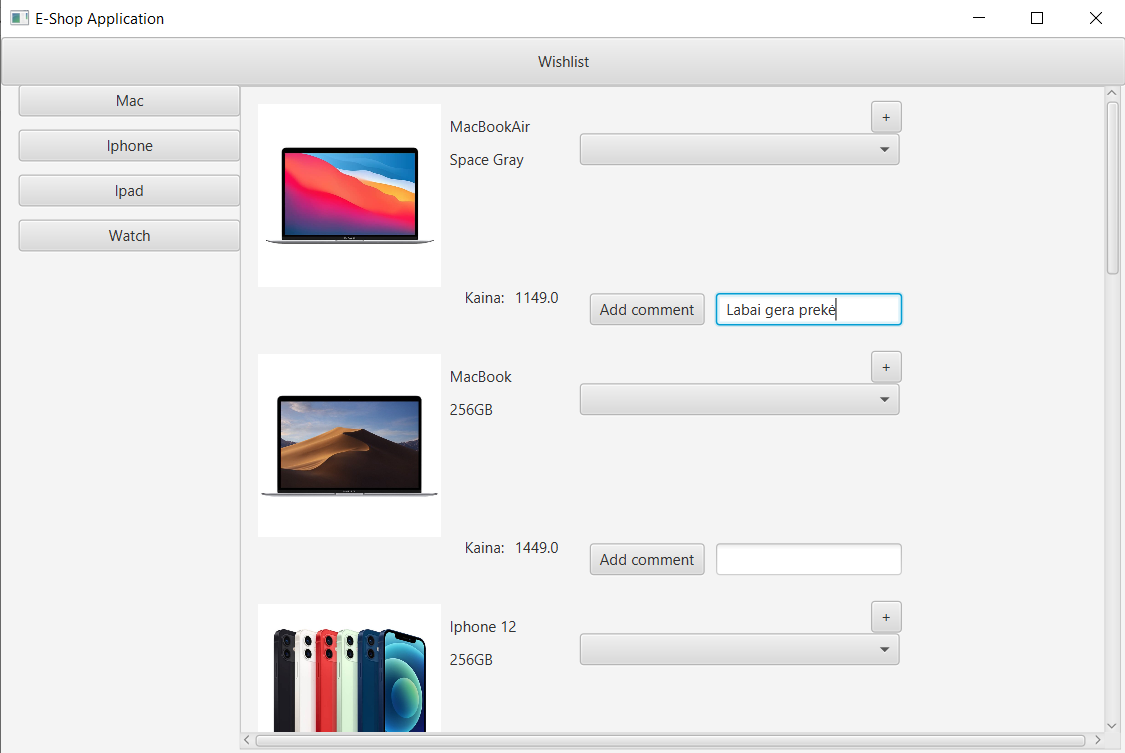
****

* Įvedame prisijungimo duomenis ir spaudžiame „login“

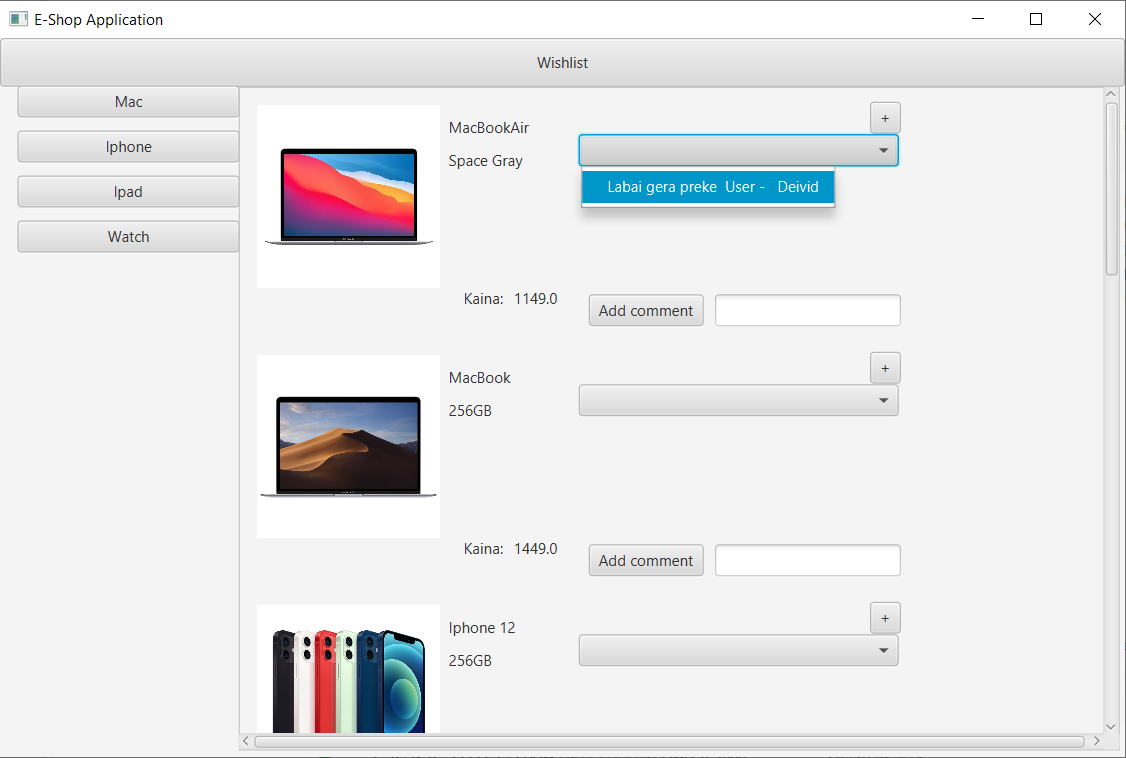


## Prekės komentavimas

* **Įrašome į tuščia laukelį savo komentarą ir spaudžiame „Add comment“**

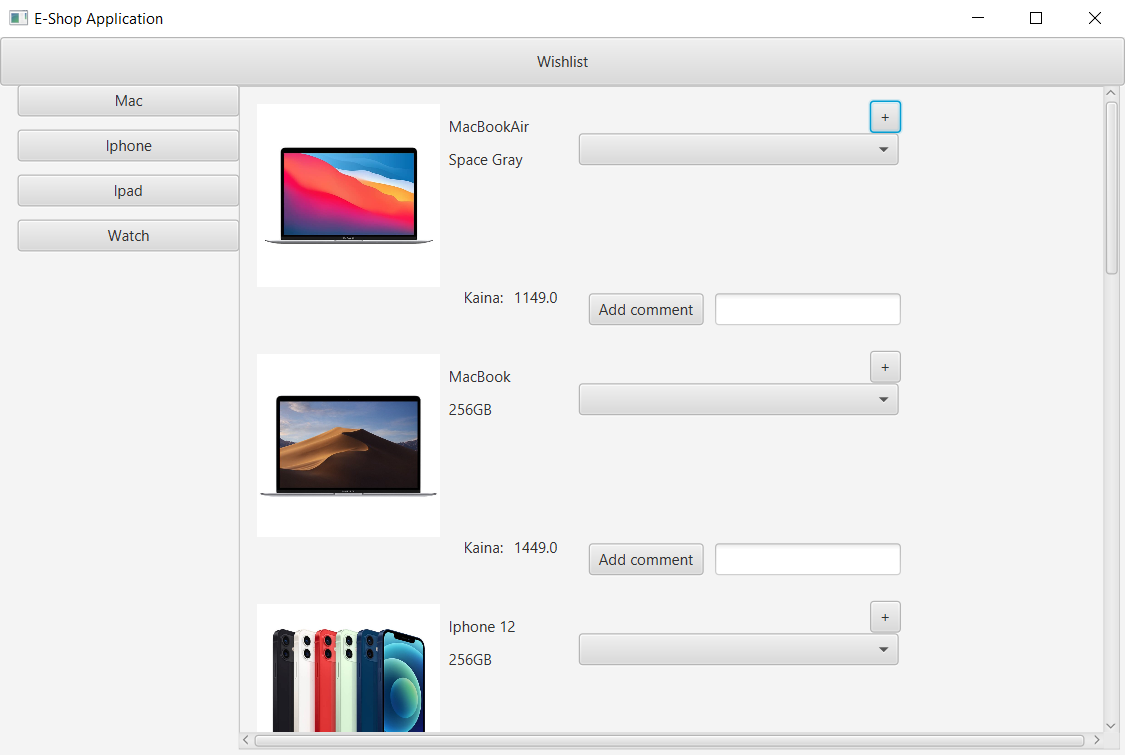
****

* **Prekės komentarą galima peržiūrėti paspaudus ant mygtuko prie prekės.**

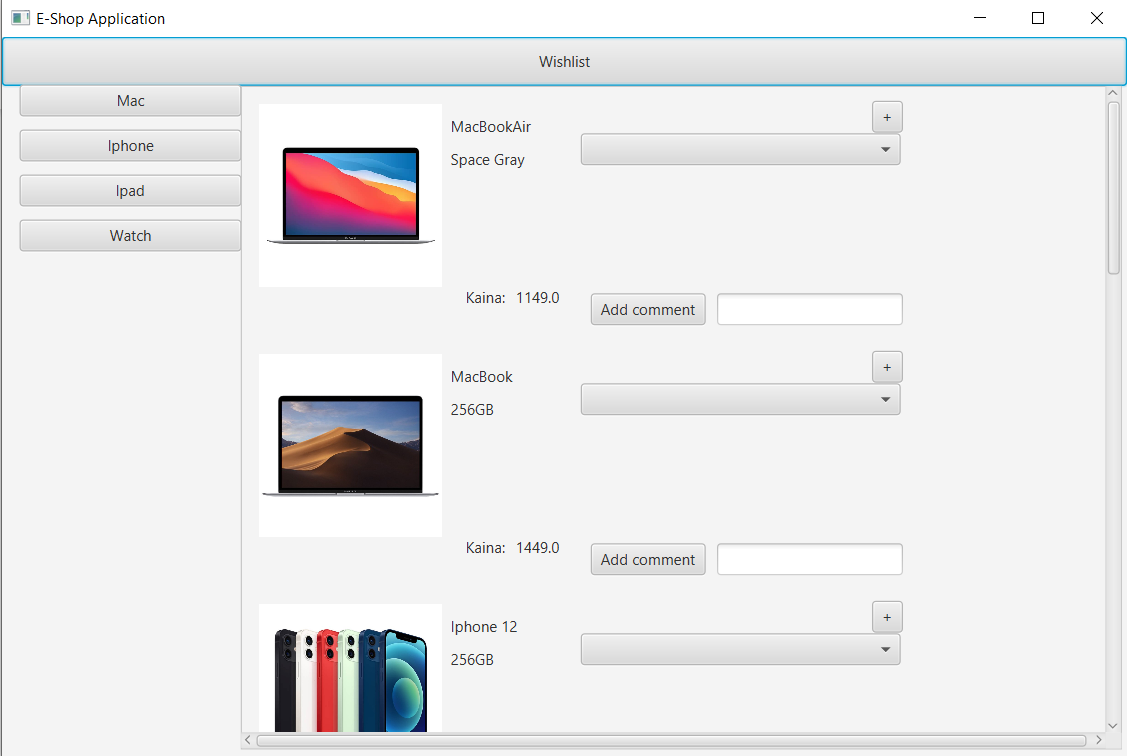
****

## Prekės įsiminimas

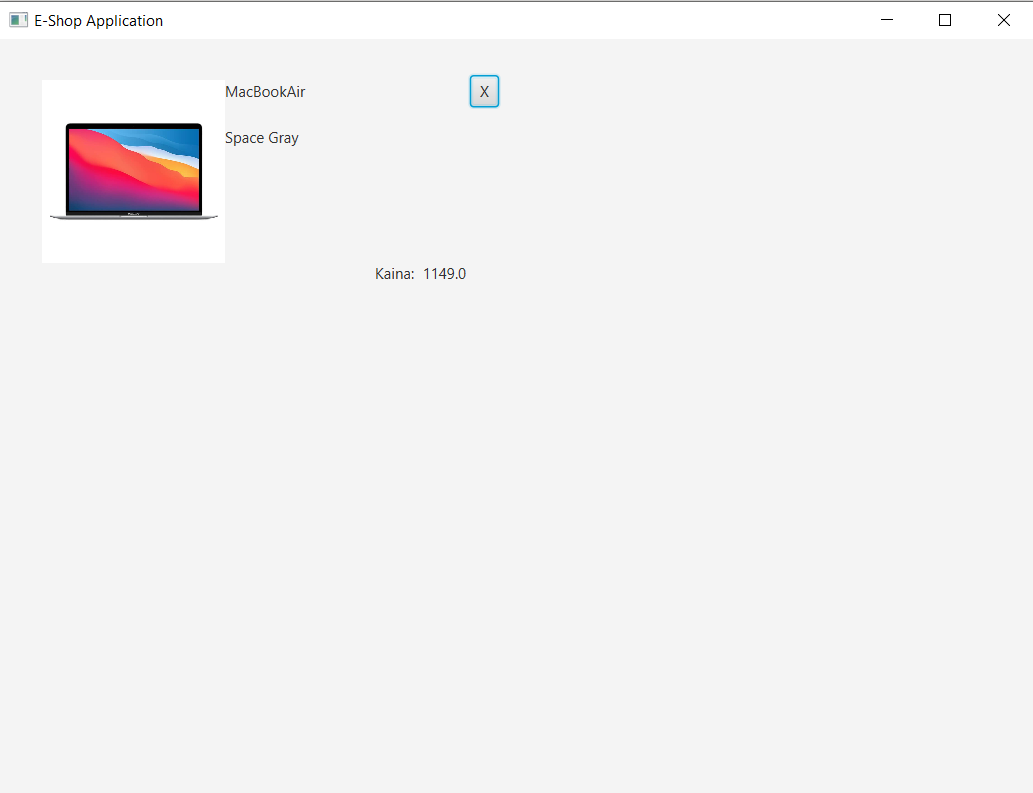
* **Paspaudžiame ant „+“ virš prekių komentarų norint įsiminti prekę**

****

* **Norint peržiūrėti įsimintas prekes spaudžiame ant „Wishlist“ mygtuko esančio viršuje**

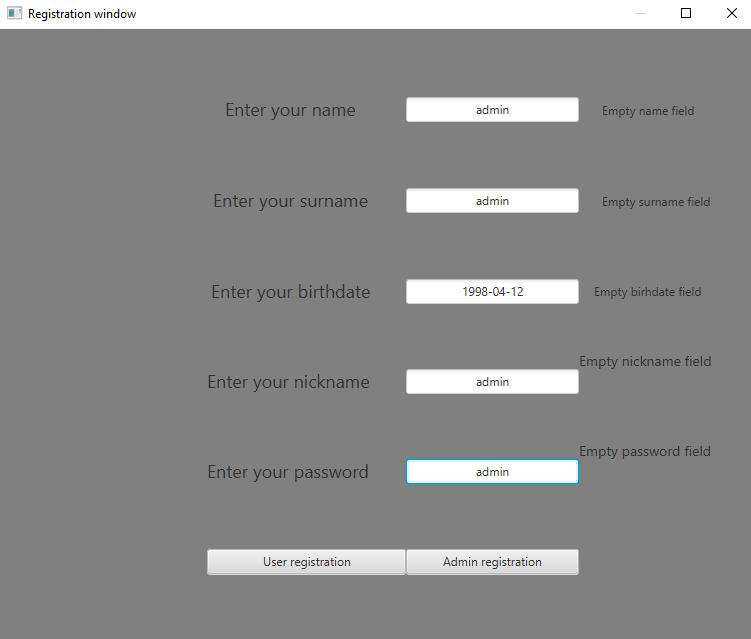
****

* **Šiame lange matysime įsimintas prekes, bei galime jas pašalinti paspaudus „X“ mygtuką**

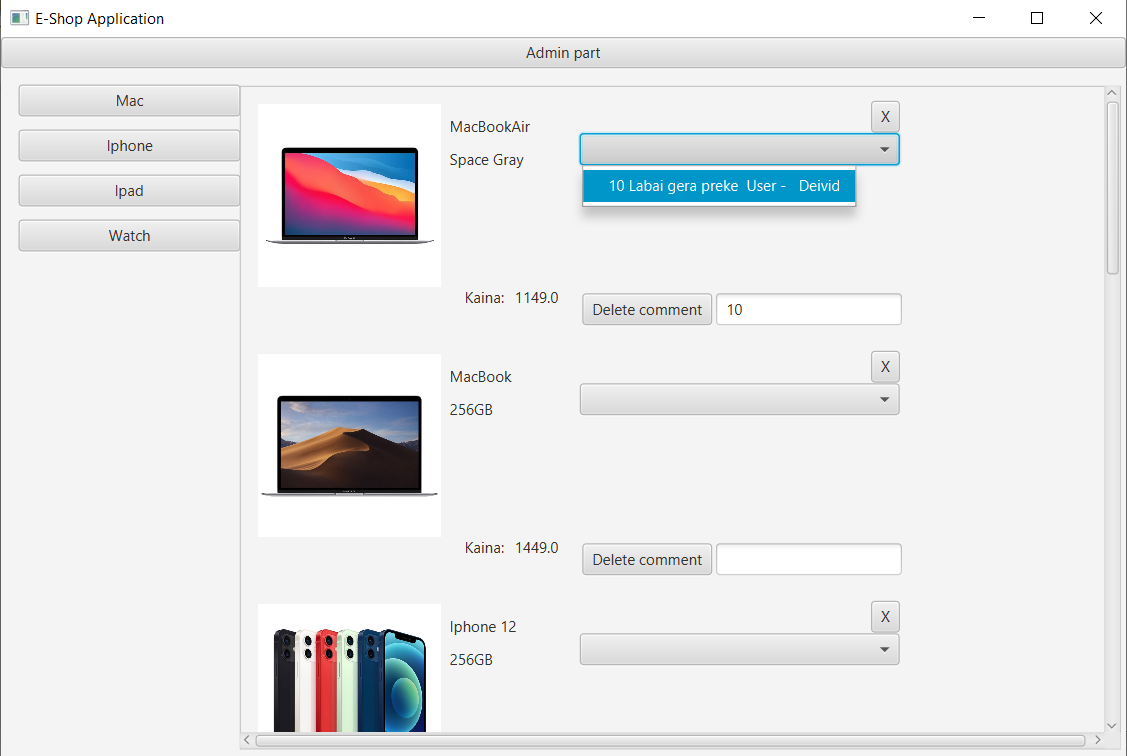
****

## Administratoriaus paskyros kūrimas ir valdymas

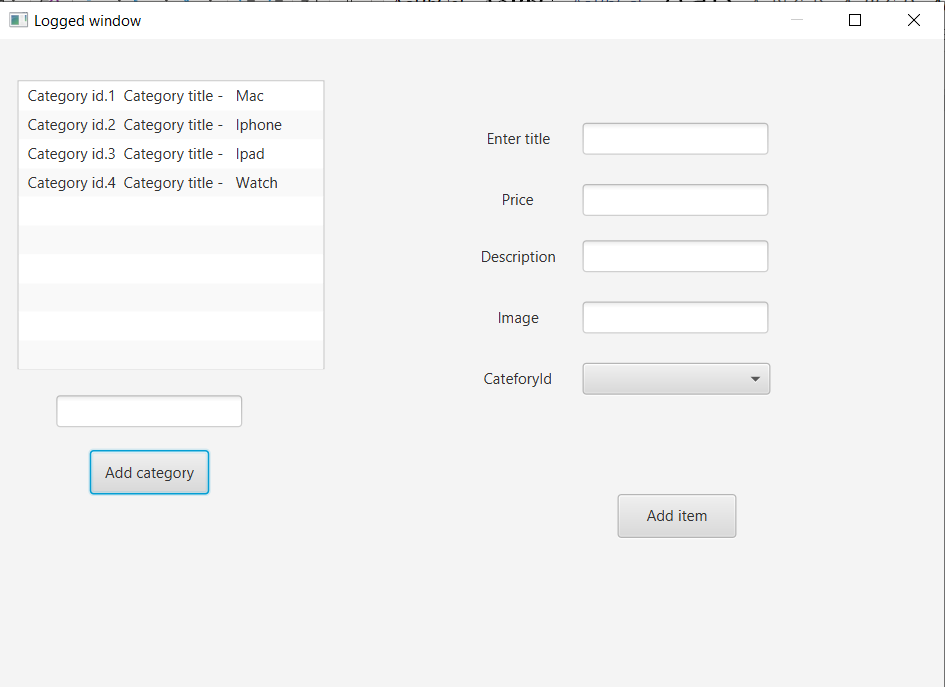
* **Norint susikurti administratoriaus paskyrą, įvedame duomenis ir spaudžiame „Admin registration“**

****

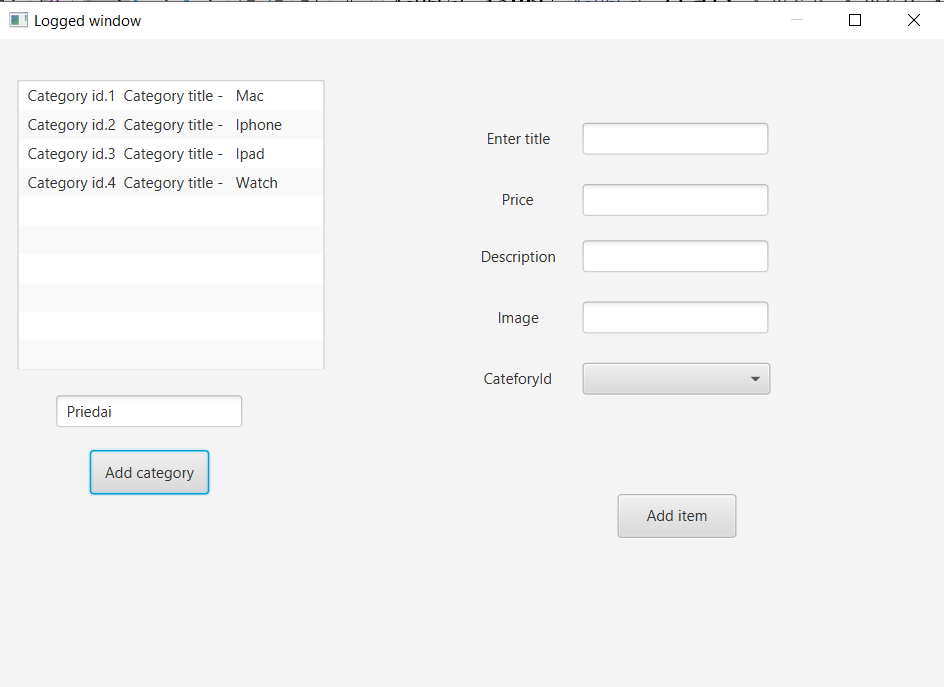
* **Prisijungus prie administratoriaus paskyros galima peržiūrėti komentarus, bei juos pašalinti. Prie komentaro rašo komentaro ID. norint ištrinti komentarą įvedame jo ID ir spaudžiame „Delete comment“ mygtuką**

****

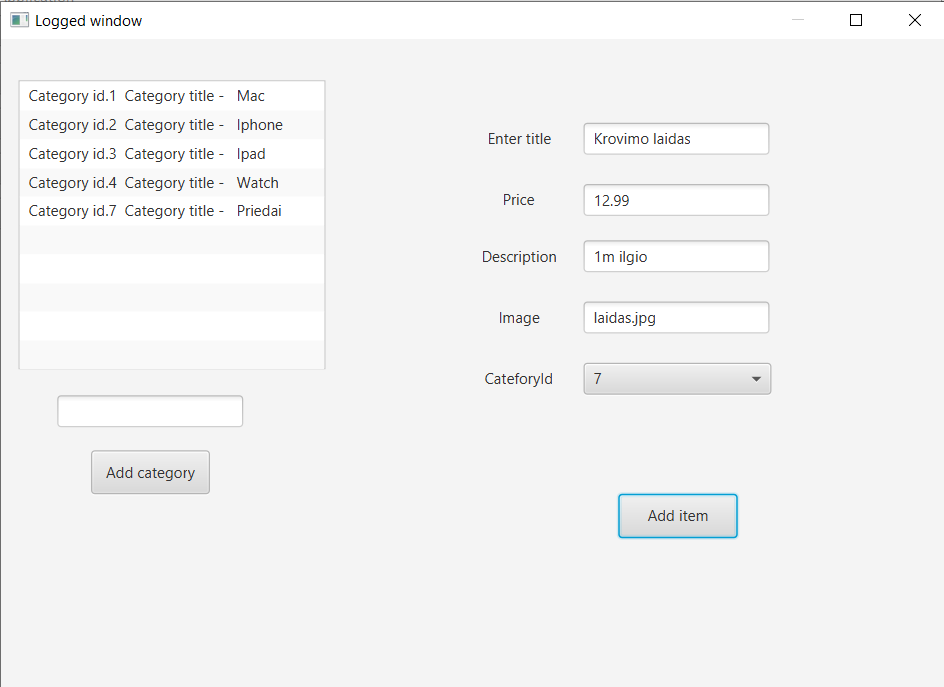
* **Viršuje matome „Admin part“ mygtuką, jį paspaudus Atidarys langą, kuriame galima pridėti prekių kategoriją bei pridėti prekes.**

****

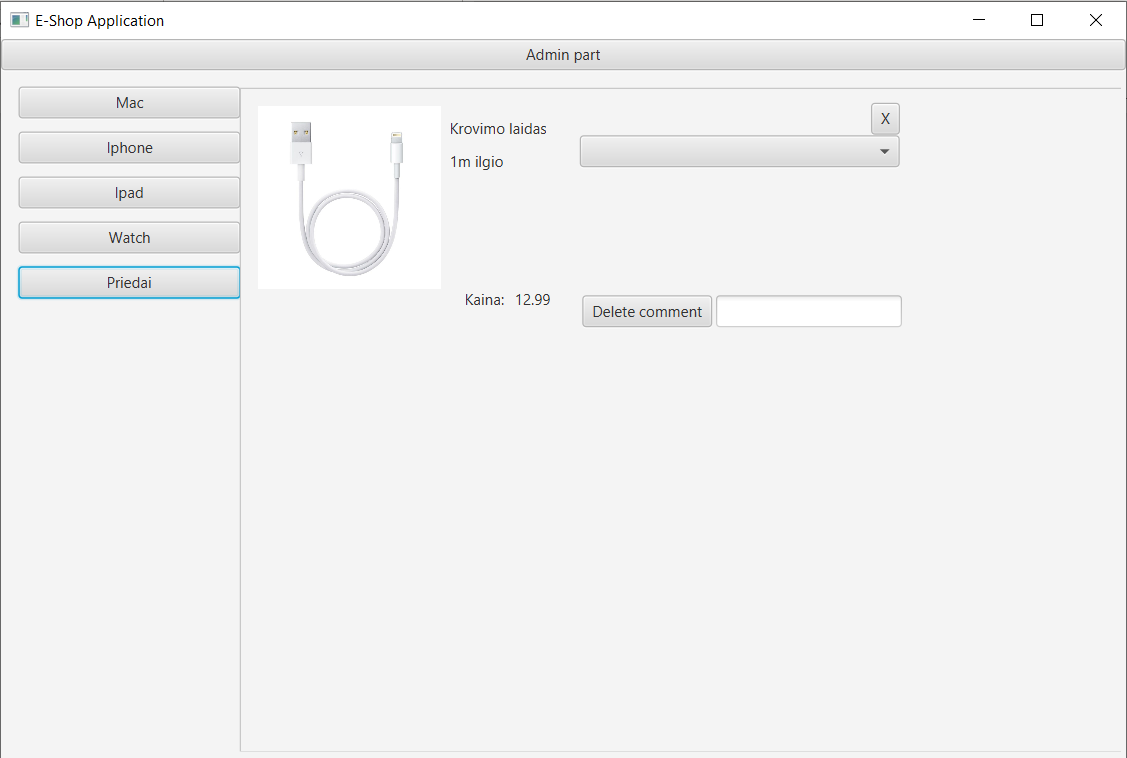
* **Norint pridėti kategoriją įvedame į kategorijos laukelį naujos kategorijos pavadinimą ir spaudžiame „Add category“ mygtuką**

****

* **Norint pridėti prekę, įvedame prekės duomenis, pasirenkame į kurią kategoriją norim pridėti ją ir spaudžiame „Add item“**

****

* **Matome, jog atsirado nauja kategorija ir prekė kurią pridėjome**

****

# IŠVADOS

Šio darbo metu buvo išmokta kurti duomenų bazes naudojant „MySQL Workbench“ programą, susieti duomenų bazę su programa ir valdyti duomenų bazės duomenis naudojant Java kalbą, valdyti išimtinius atvejus ir dirbti su objektų kolekcijomis.