|  |  |
| --- | --- |
|  | ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА  ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ |

**КУРСОВ ПРОЕКТ**

на

Хакан Нихат Мехмед, фак.№ 19621700

Илиян Кирилов Костов, фак.№ 19621609

спец.: СИТ, III курс, 4б група,

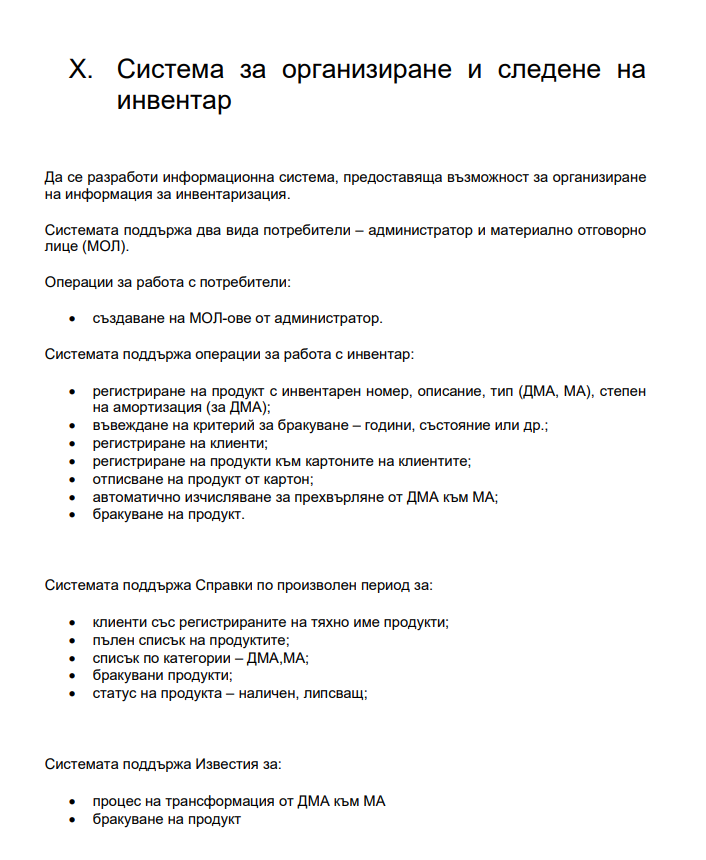
**Тема №20:**

Система за организиране и

следене на инвентар

В А Р Н А

2021/2022 год.



**Анализ на проблема**

Тъй като системата съдържа два вида потребители- Админ и МОЛ, ще трябва да имаме два отделни панела за всеки потребител. В началния панел за логване потребителят ще избере между „МОЛ“ и „ Админ“ и след това да си въведе потребителското име и паролата. При некоректни данни или празни полета, логването ще е неуспешно и потребителят ще бъде уведомен за грешката си.

При успешно логване се отваря контролният панел на съответния потребител (единствената разлика между панелите е, че админът има бутон за създаване на материално отговорни лица).

Контролният панел ще съдържа бутони, които при кликване ни пренасочват към други панели за регистриране, справка и т.н.

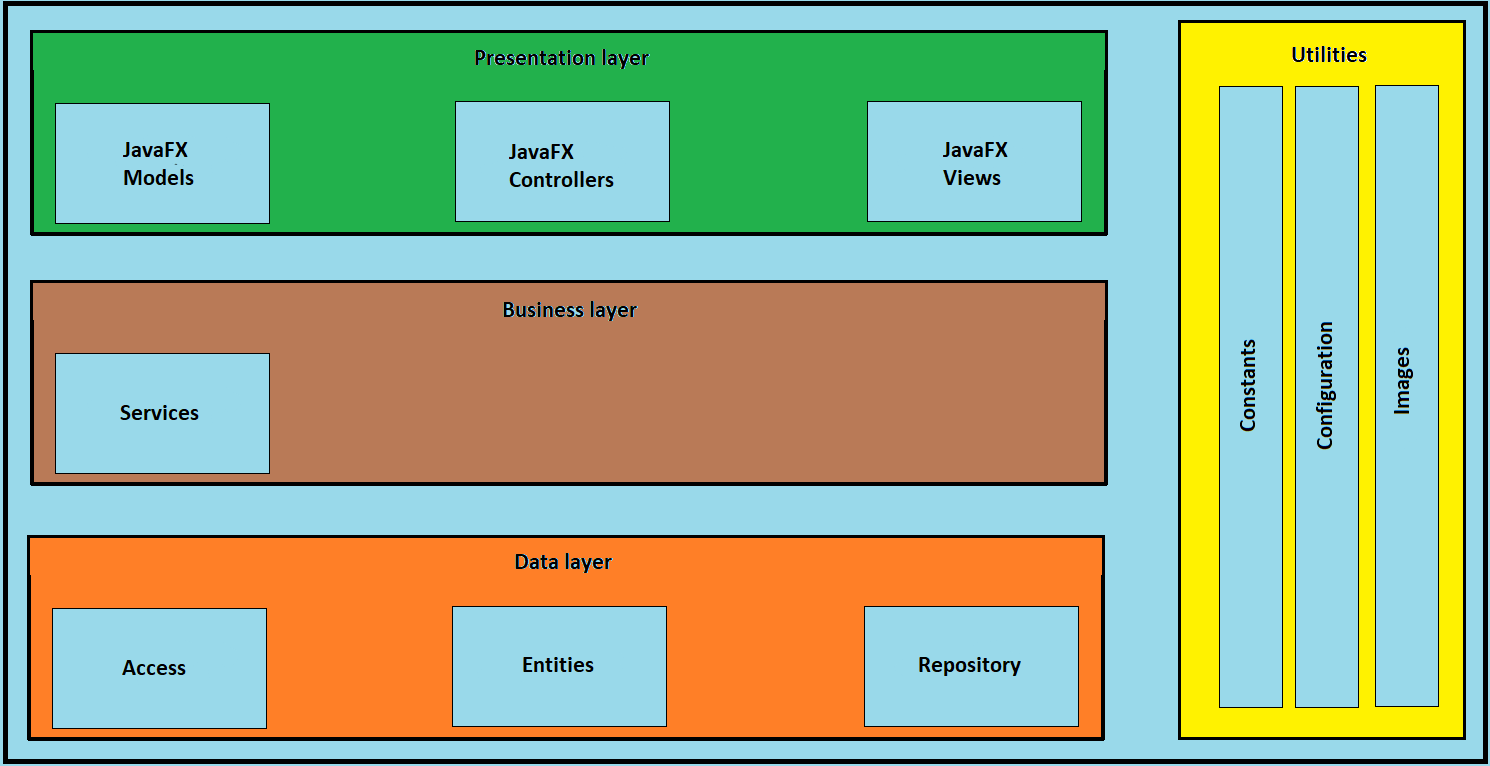
Всички панели ще имат бутон за връщане назад, а контролните панели - бутон за изход.

1. Регистриране на продукт – попълва се формулярът за регистриране на продукт. При наличие на празни полета регистрирането е неуспешно. Ако за тип се избере МА, комбобоксът за степен на амортизация става недостъпен.
2. Регистриране на клиент - попълва се формулярът за регистриране на клиент. При наличие на празни полета регистрирането е неуспешно.
3. Регистриране на продукт към картон на клиент и отписване – избира са Id на клиент и инвентарен номер на продукт и ако продуктът не е регистриран, се регистрира, в противен случай потребителят се уведомява, че продуктът не наличен или ако е бракуван, се уведомява, че е бракуван. За отписване се избира ред от таблицата и ако продуктът не е отписан се отписва, при което автоматично получава дата на връщане. Върнати са тези продукти, които имат дата на връщане. Ако потребителя се опита да върне върнат продукт, получава съобщение за грешка.
4. Въвеждане на състояние – тук се въвеждат причините за бракуване на продукта. При наличие на празни полета регистрирането е неуспешно.
5. Справка за клиенти с регистрираните на тяхно име продукти – избира се периодът, за който ще се направи справката. Ако не са посочени дати, потребителят получава съобщение за грешка.
6. Справка за всички продукти - избира се периодът, за който ще се направи справката. Ако не са посочени дати, потребителят получава съобщение за грешка.
7. Справка за продукти по категории - избира се периодът, за който ще се направи справката и типът на продукта( по подразбиране е избран „ДМА“). Ако не са посочени дати, потребителят получава съобщение за грешка.
8. Справка за продукти по наличност - избира се периодът, за който ще се направи справката и типът на продукта( по подразбиране е избран „налични“). Ако не са посочени дати, потребителят получава съобщение за грешка.

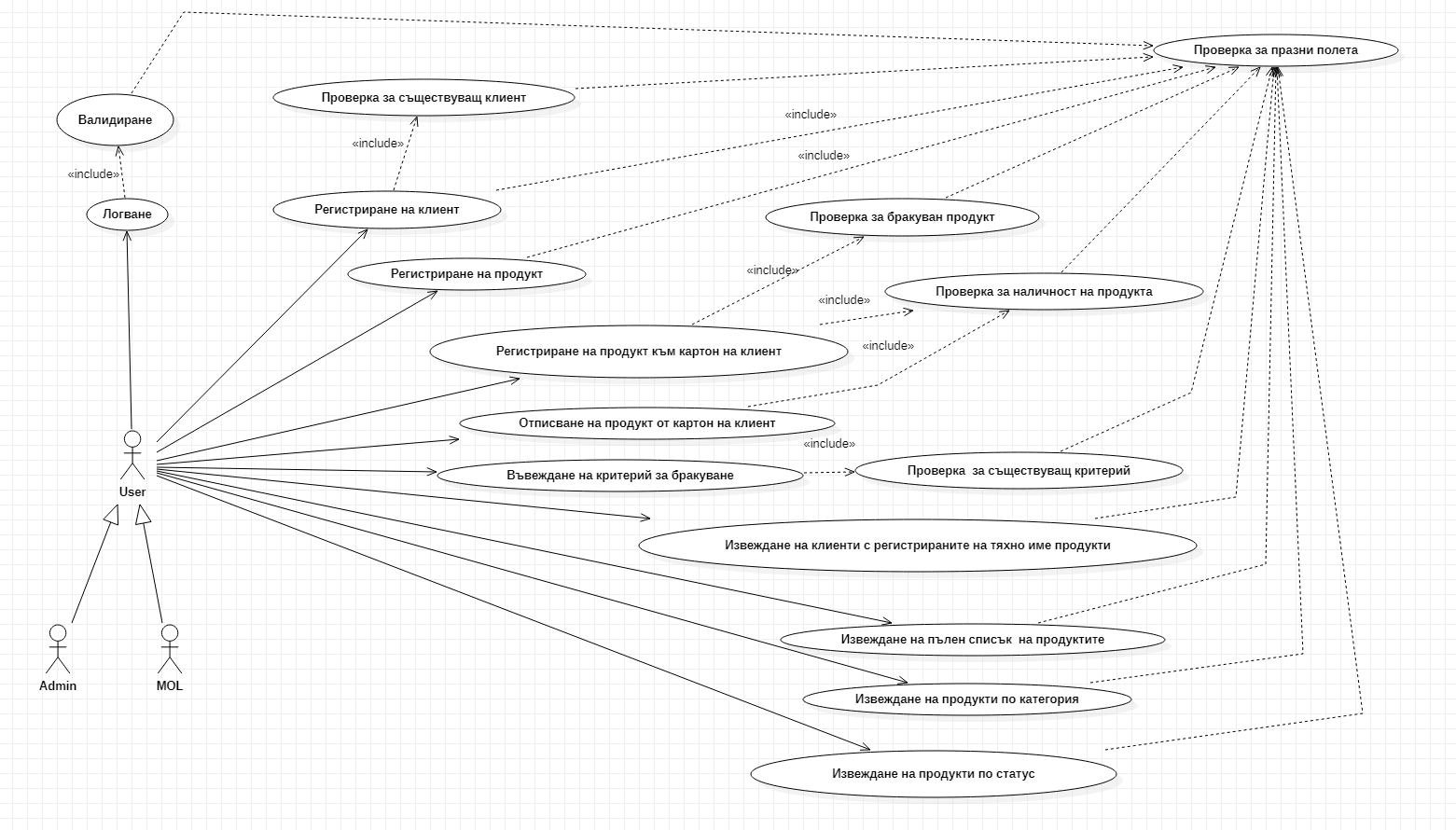
**Структура на проекта**

За реализиране на системата ще се използва многослойната архитектура, тъй като този модел може да се използва за структуриране на програми , които могат да бъдат разложени на групи от подзадачи, всяка от които е на определено ниво на абстакция и всеки слой предоставя услуги на следващия по-висок слой.

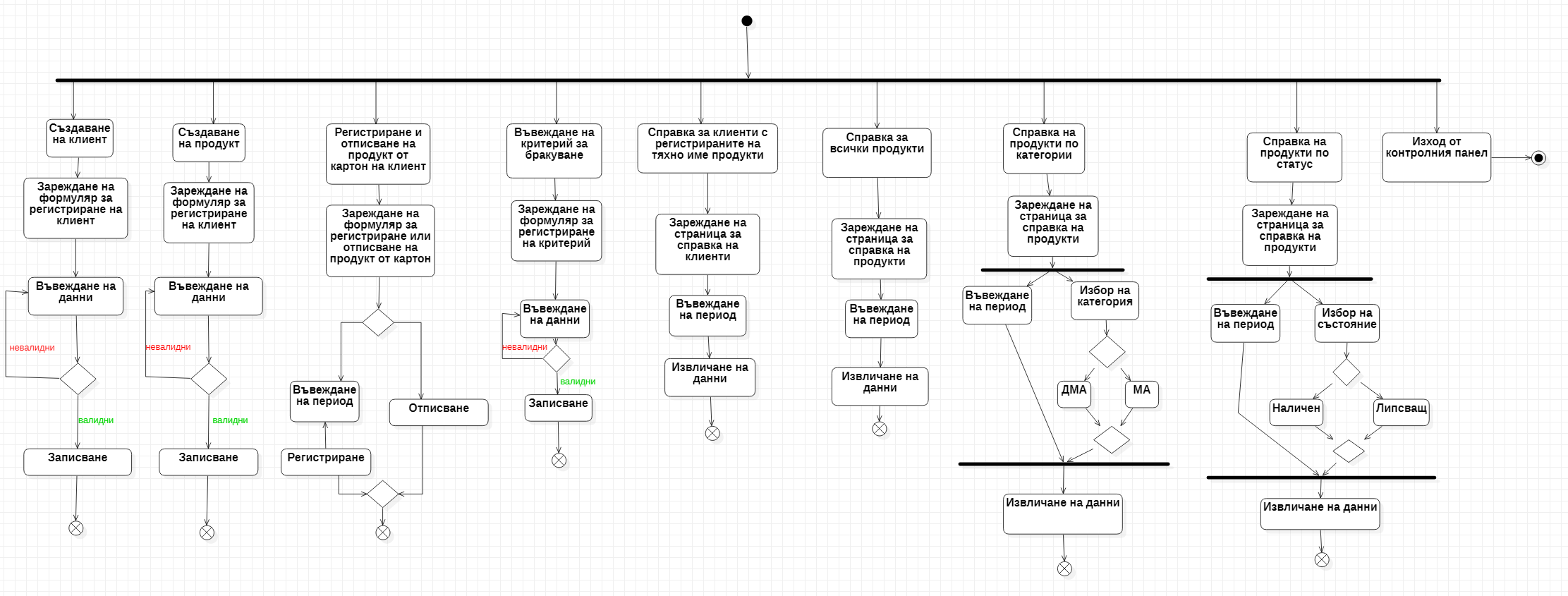
Не схемата се виждат всички модули в системата:



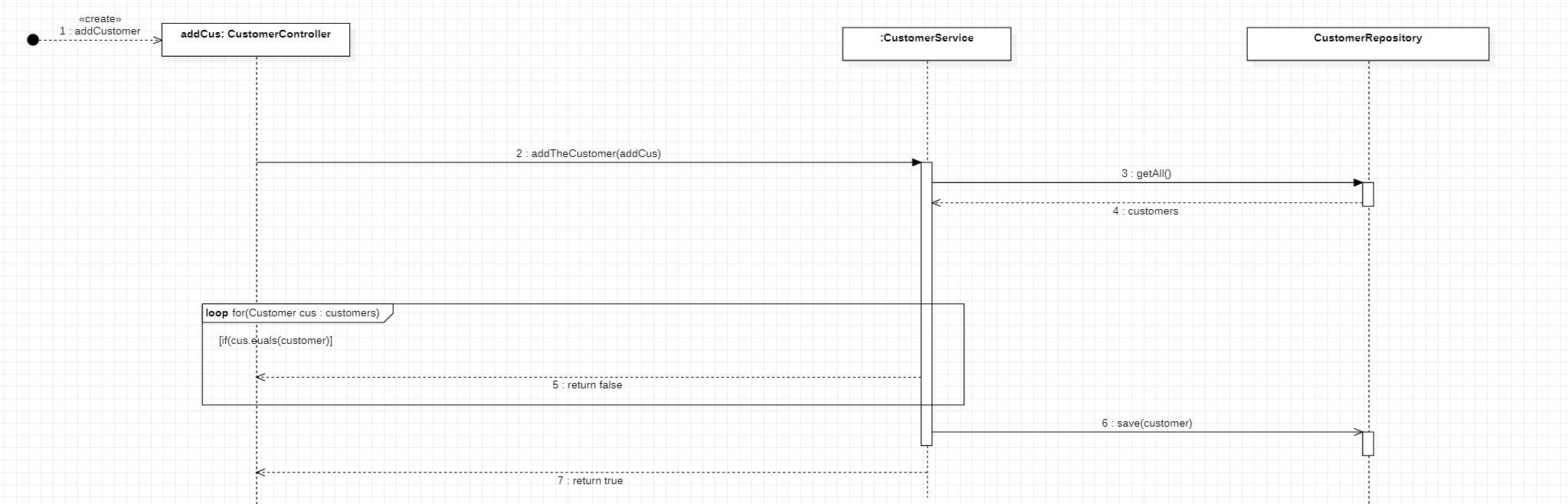
**Проектиране на системата**

Uml Diagram

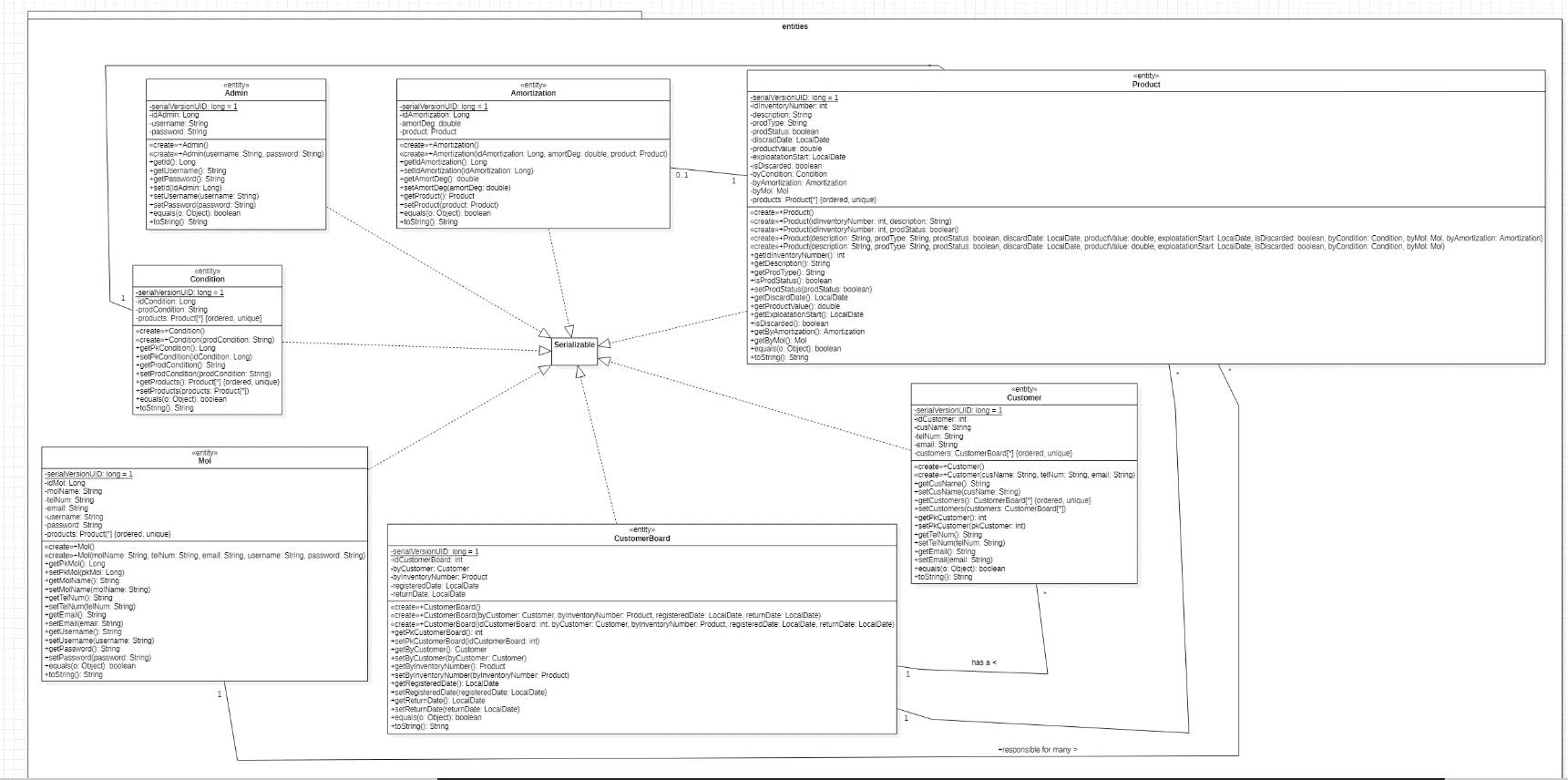
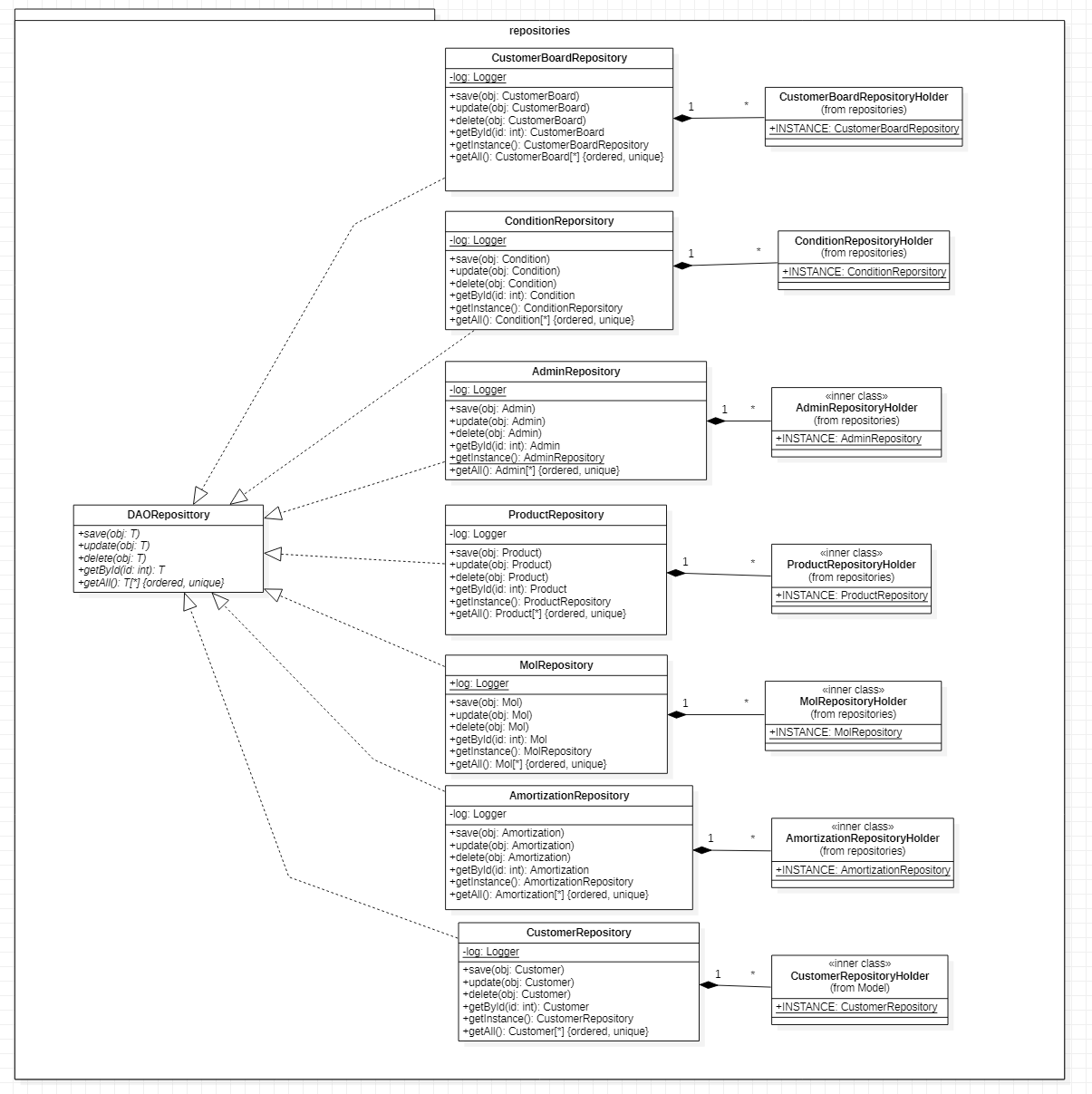
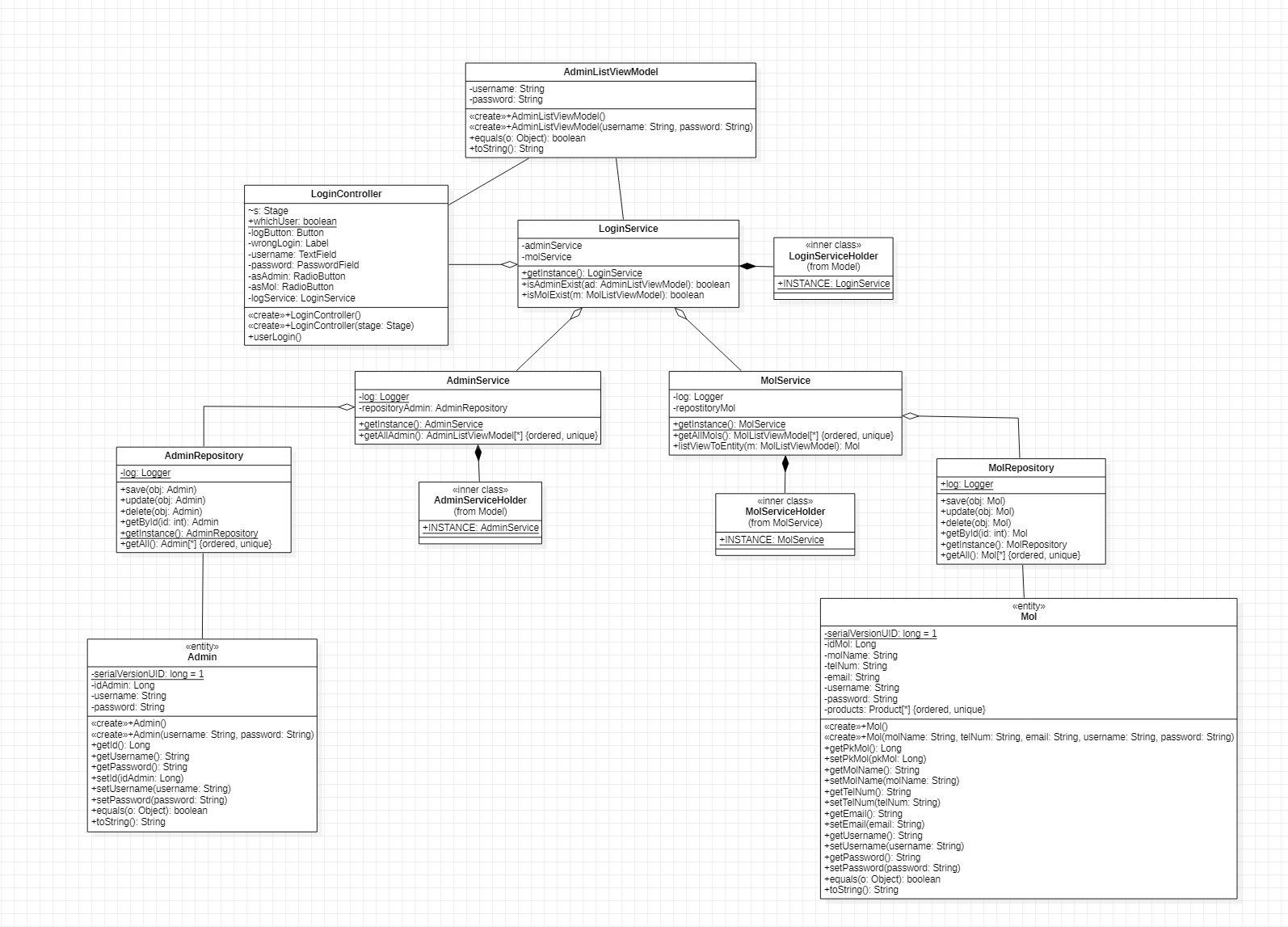
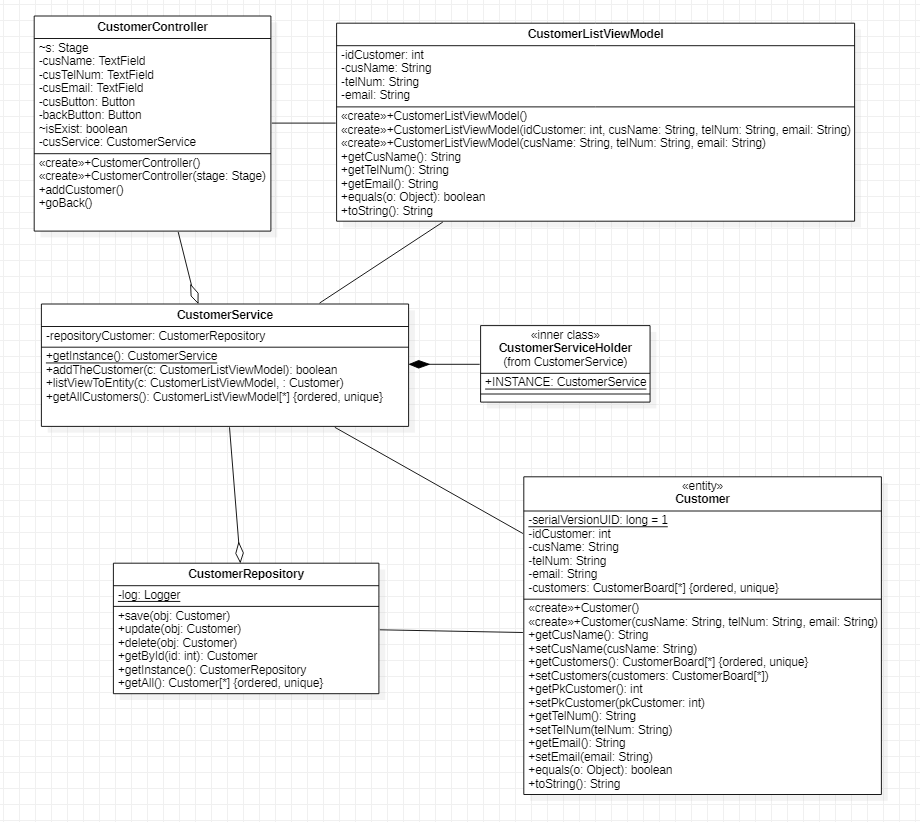
Activity diagram

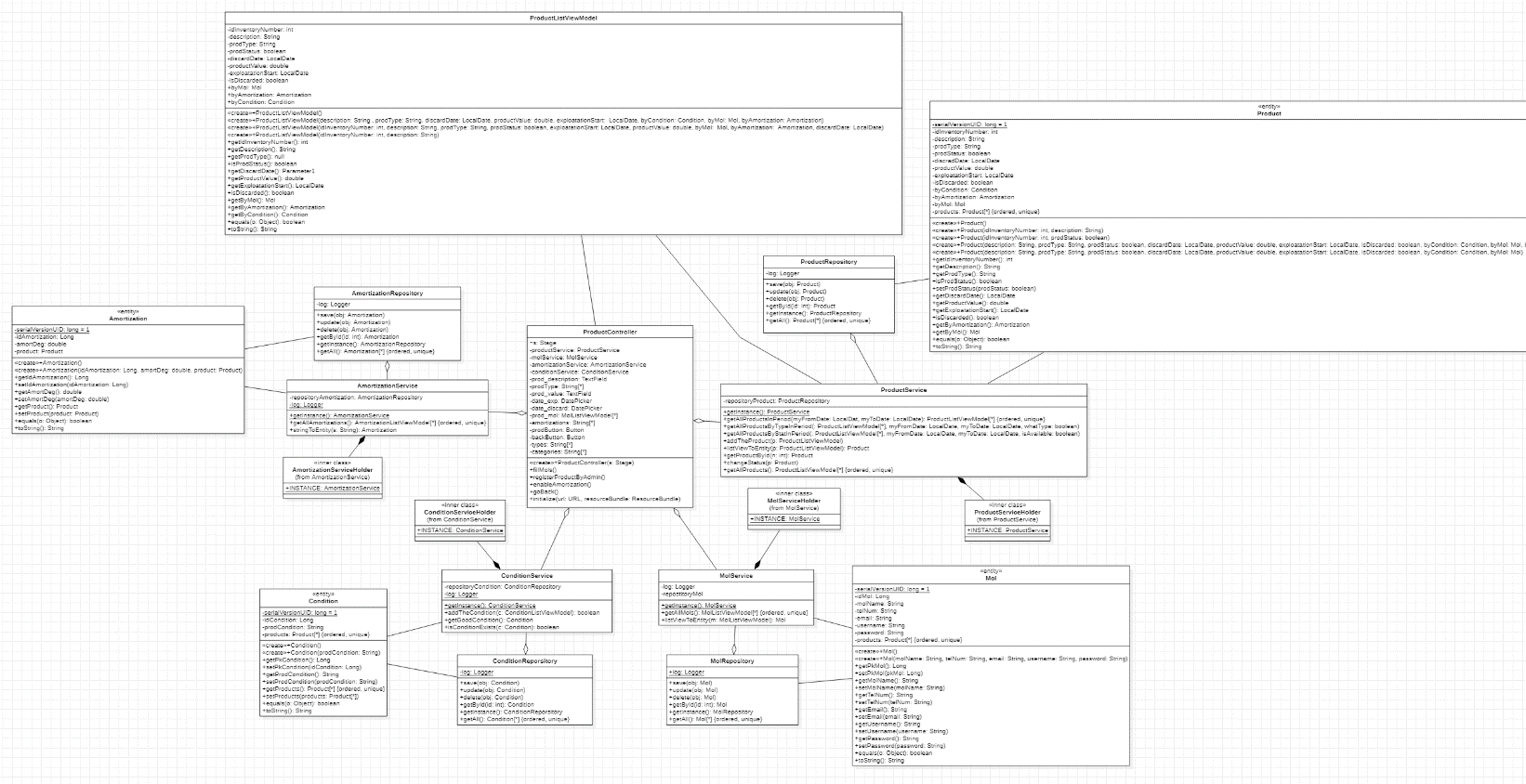


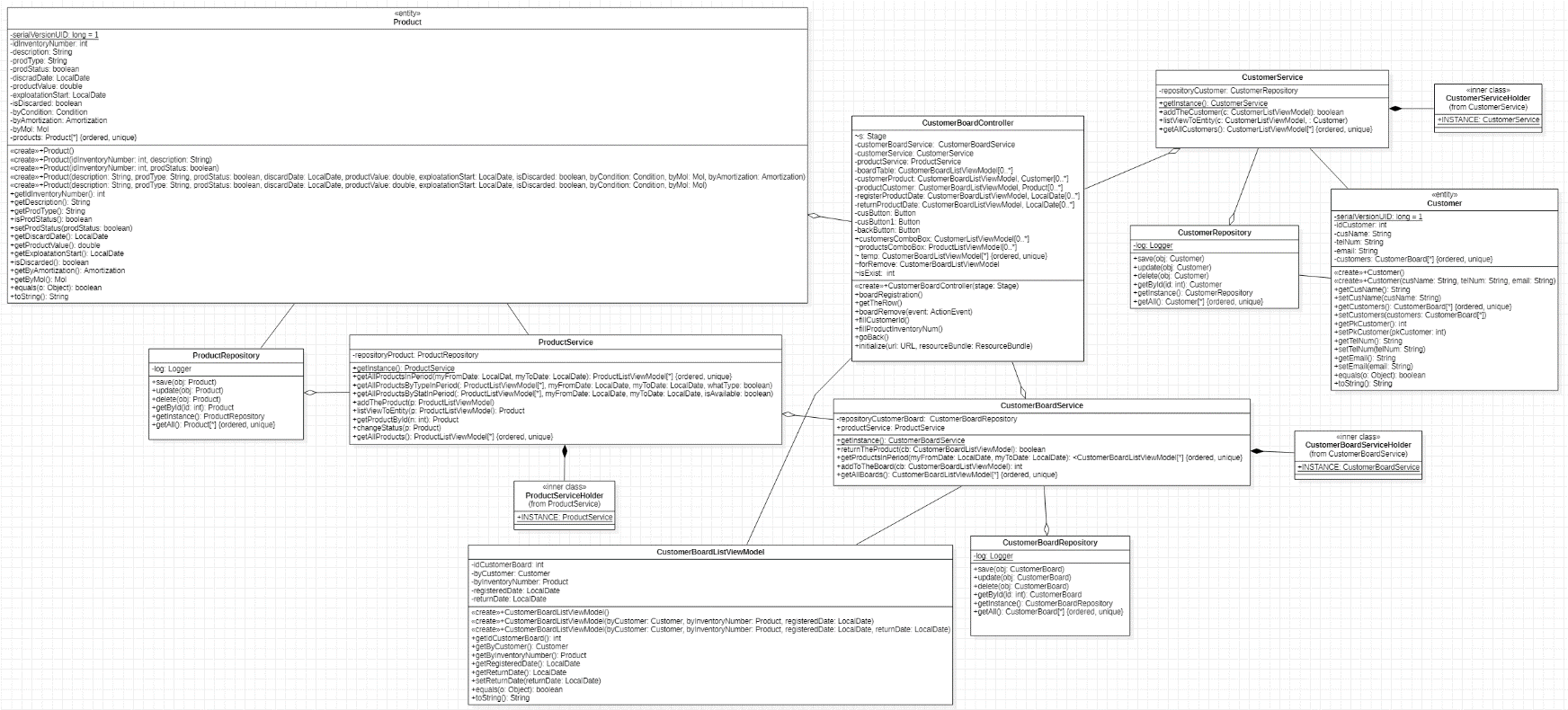
Sequence diagram

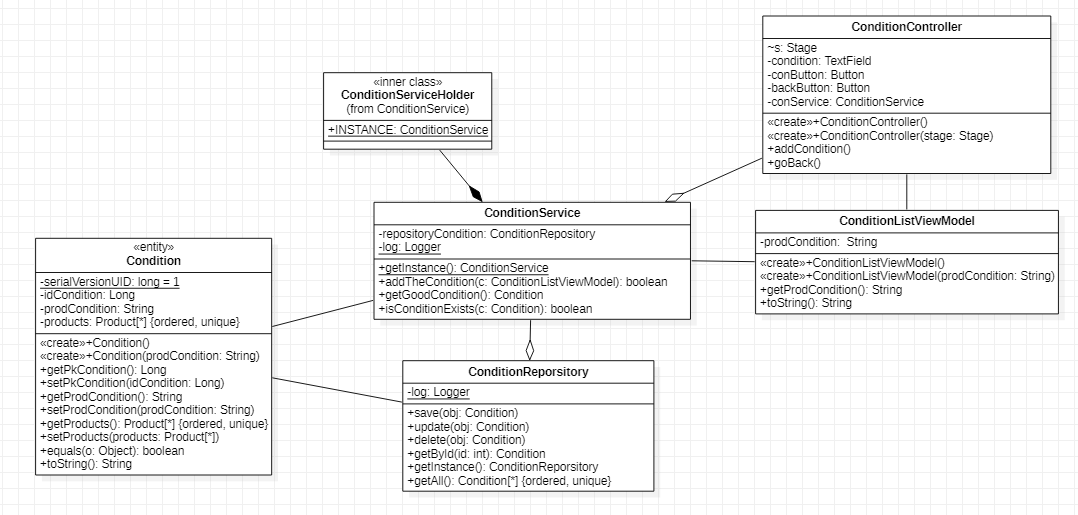


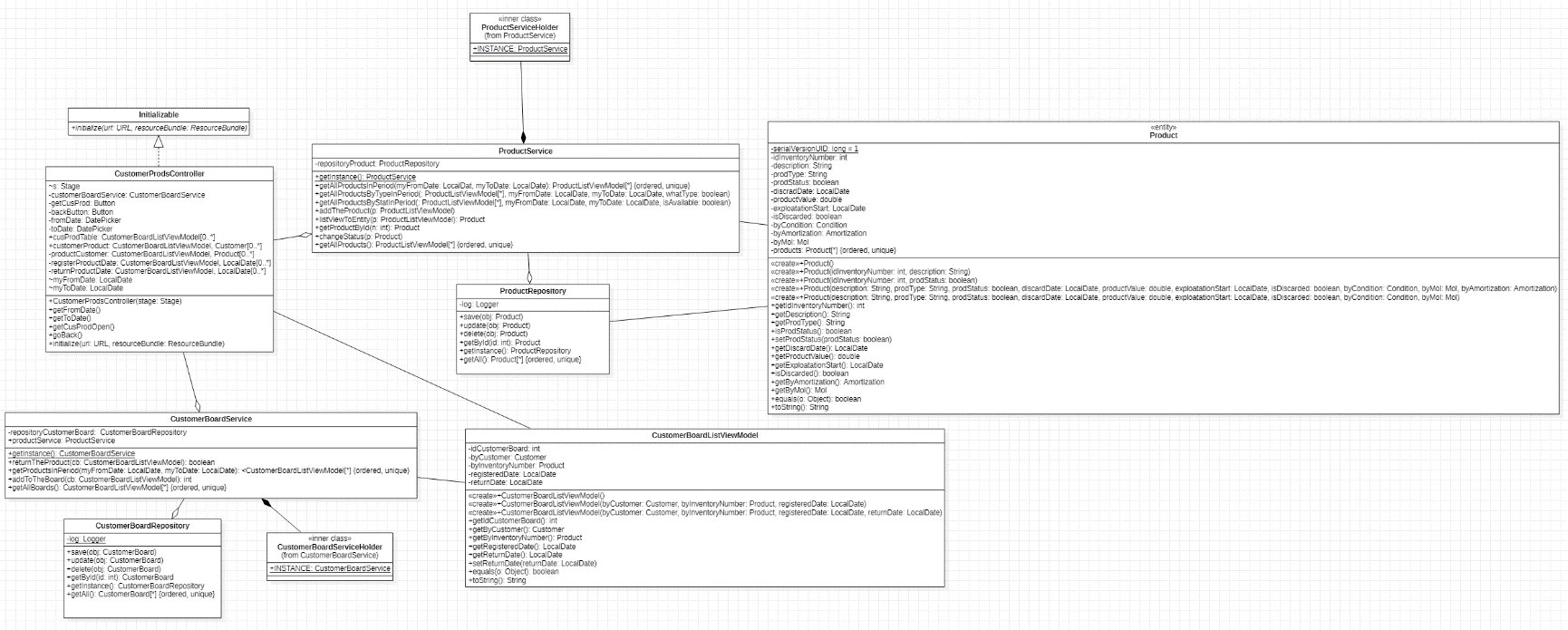
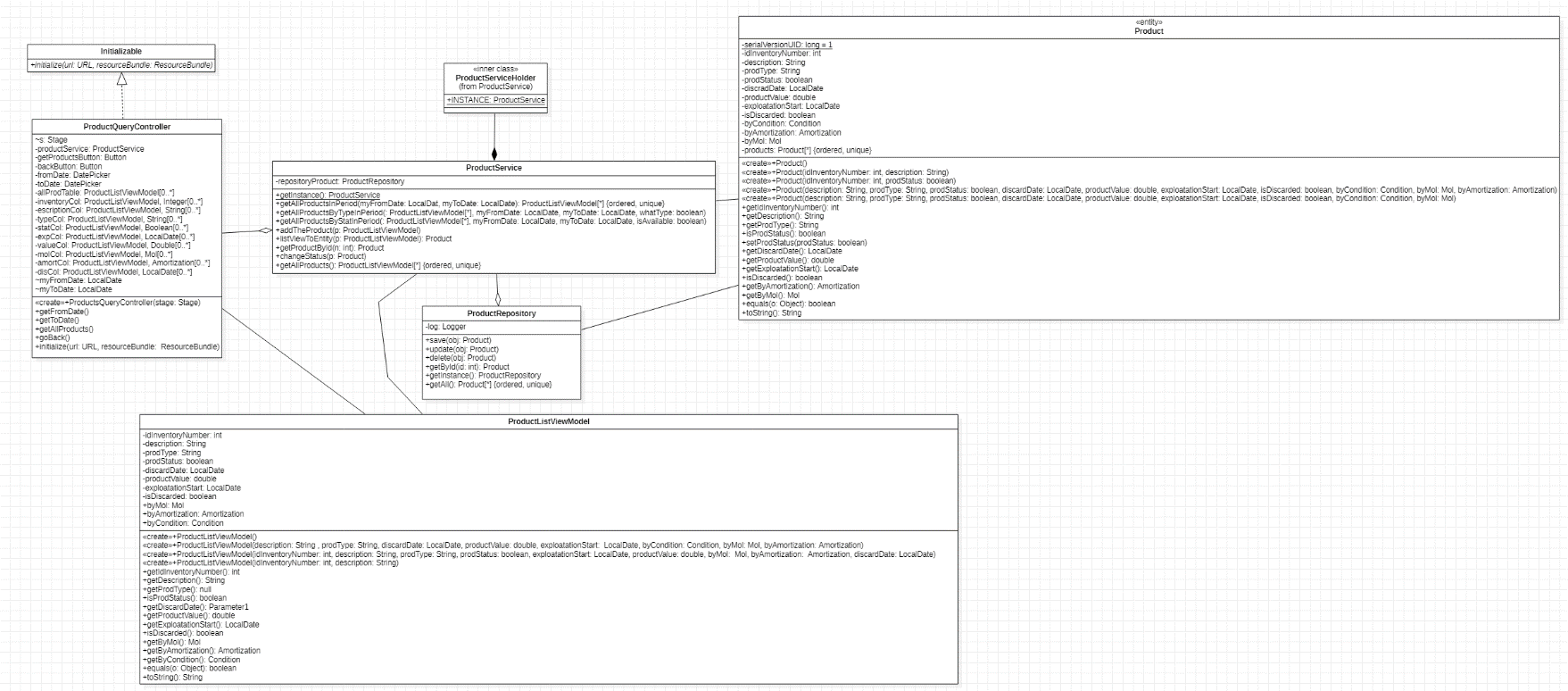
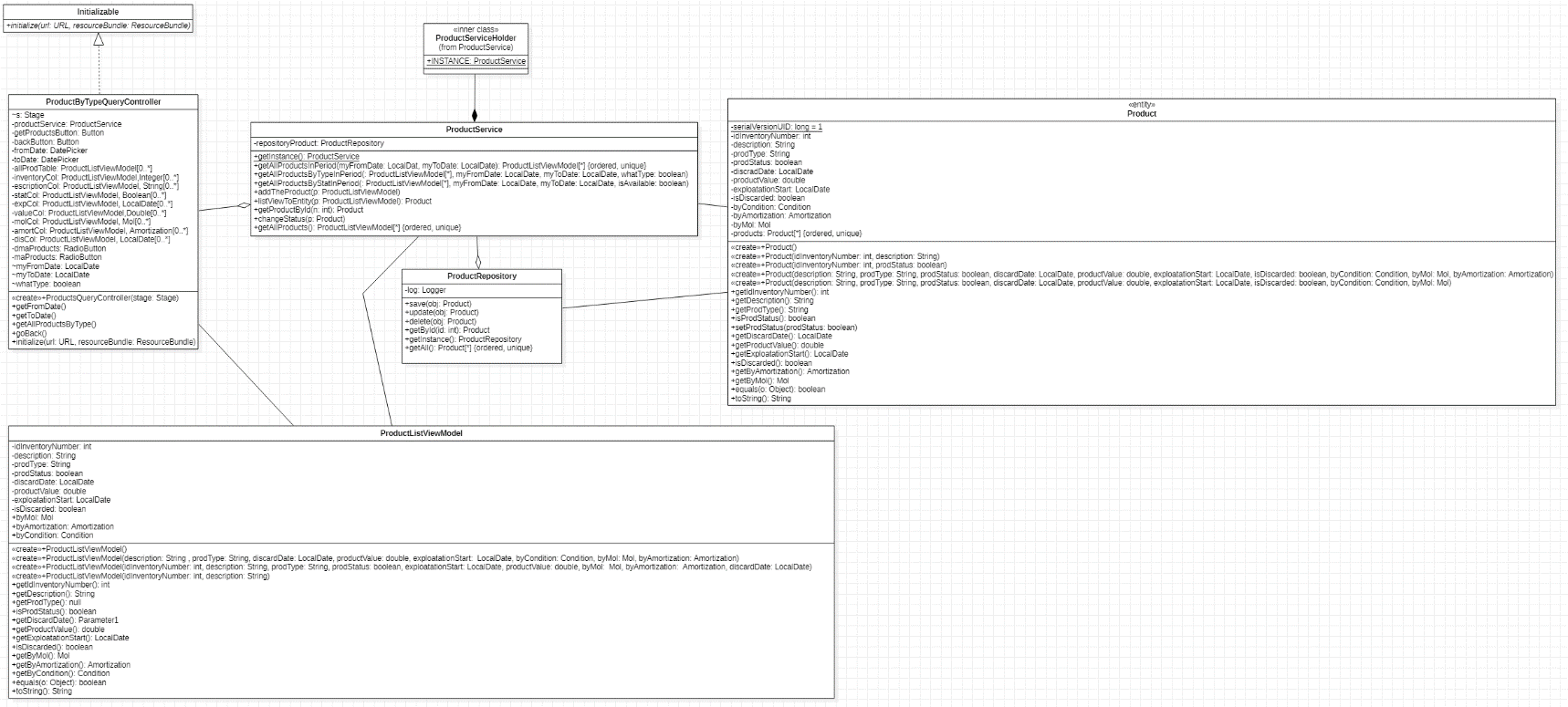
Class diagrams

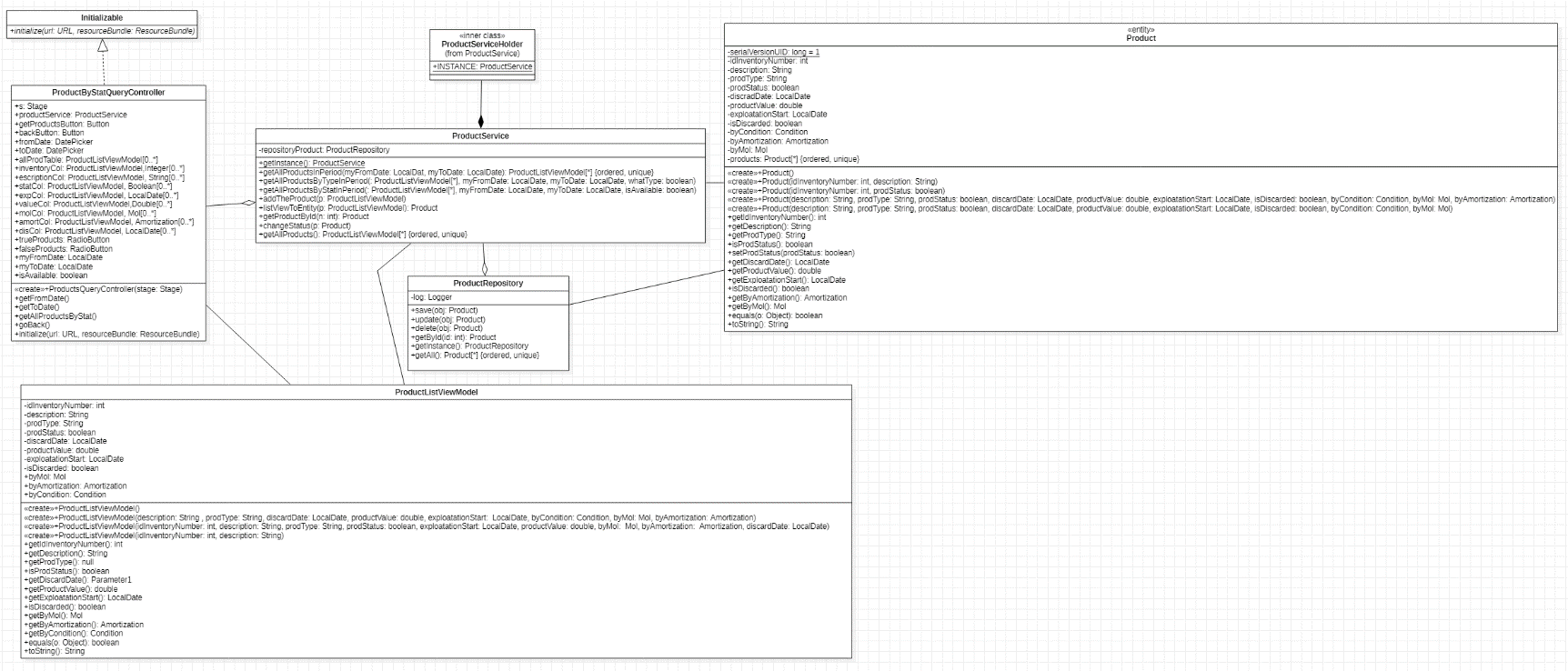
* Entities
* Repositories
* UserLogin
* RegisterCustomer
* RegisterProduct



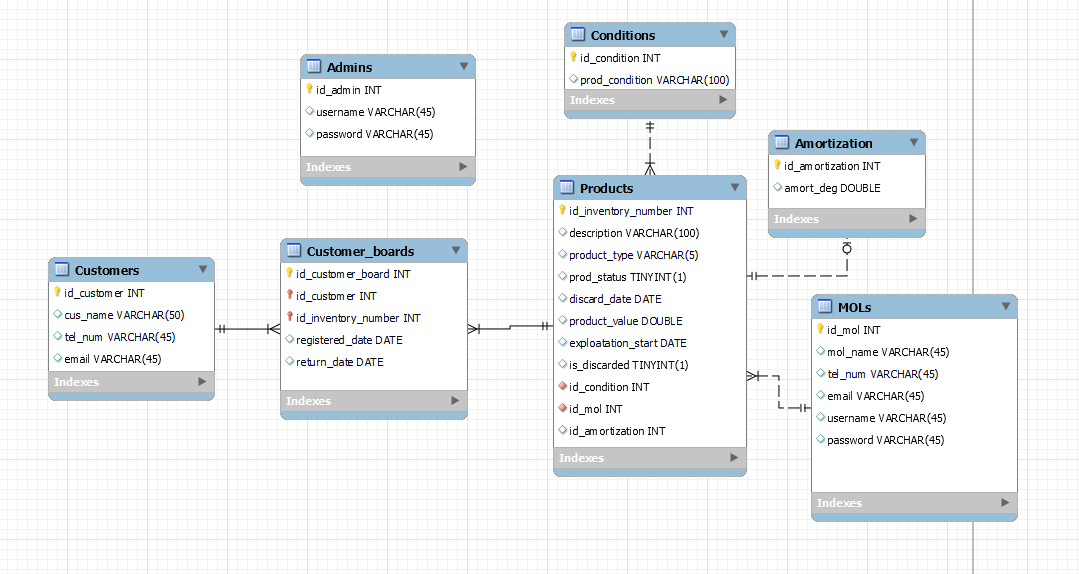
* RegisterProductToCustomer
* addCondition



* CustomerQuery
* ProductQuery
* ProductByTypeQuery
* ProductByStatQuery



ER diagram



**Реализация на проекта**

Имаме общо 7 таблици. Колоните на всички таблици са именовани така, че лесно да се разбере каква информация ще съдържат. На възможните места са използвани съкращения ( примерно: prod\_status вместо product\_status).

За статус на продукт и за бракуван продукт се използва тип Boolean(tinyint(1)), понеже има само два възможни случая(наличен/не наличен, бракуван/ не бракуван). За всички пароли, описания, имена се използва Varchar, понеже тектовете няма да имат фиксирана дължина.

* Заявки за създаване на някои от таблиците:

-- -----------------------------------------------------

-- Table `inventory`.`Products`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `inventory`.`Products` (

`id\_inventory\_number` INT NOT NULL AUTO\_INCREMENT,

`description` VARCHAR(100) NULL,

`product\_type` VARCHAR(5) NULL,

`prod\_status` TINYINT(1) NULL,

`discard\_date` DATE NULL,

`product\_value` DOUBLE NULL,

`exploatation\_start` DATE NULL,

`is\_discarded` TINYINT(1) NULL,

`id\_condition` INT NOT NULL,

`id\_mol` INT NOT NULL,

`id\_amortization` INT NULL,

PRIMARY KEY (`id\_inventory\_number`),

INDEX `fk\_Products\_MOL1\_idx` (`id\_mol` ASC) VISIBLE,

INDEX `fk\_Products\_Conditions1\_idx` (`id\_condition` ASC) VISIBLE,

INDEX `fk\_Products\_Amortization1\_idx` (`id\_amortization` ASC) VISIBLE,

CONSTRAINT `fk\_Products\_MOL1`

FOREIGN KEY (`id\_mol`)

REFERENCES `inventory`.`MOLs` (`id\_mol`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Products\_Conditions1`

FOREIGN KEY (`id\_condition`)

REFERENCES `inventory`.`Conditions` (`id\_condition`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Products\_Amortization1`

FOREIGN KEY (`id\_amortization`)

REFERENCES `inventory`.`Amortization` (`id\_amortization`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `inventory`.`Customer\_boards`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `inventory`.`Customer\_boards` (

`id\_customer\_board` INT NOT NULL AUTO\_INCREMENT,

`id\_customer` INT NOT NULL,

`id\_inventory\_number` INT NOT NULL,

`registered\_date` DATE NULL,

`return\_date` DATE NULL,

PRIMARY KEY (`id\_customer\_board`, `id\_customer`, `id\_inventory\_number`),

INDEX `fk\_Customer\_boards\_Customers\_idx` (`id\_customer` ASC) VISIBLE,

INDEX `fk\_Customer\_boards\_Products1\_idx` (`id\_inventory\_number` ASC) VISIBLE,

CONSTRAINT `fk\_Customer\_boards\_Customers`

FOREIGN KEY (`id\_customer`)

REFERENCES `inventory`.`Customers` (`id\_customer`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Customer\_boards\_Products1`

FOREIGN KEY (`id\_inventory\_number`)

REFERENCES `inventory`.`Products` (`id\_inventory\_number`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

**Реализация на слоя за работа с базата данни**

За целта ще се ползва Hibernate имплементация на JPA, понеже има следните предимства:

1. По-бързо написване на програмата за достъп до базата данни;
2. Може да генерира SQL в движение и след това автоматично извършва необходимите SQL отчети (спестява време);
3. Независим от базата данни( Пример: За преминаване от MySQL на Oracle е нужно да се смени само конфигурационният файл на Hibernate, а не всички класове);
4. Подкрепа за анотация на JPA;
5. С отворен код и безплатен софтуер и много други

Има и леки недостатъци :

1. Малко по-бавно от чисто JDBC
2. Генерира много SQL изявления по време на изпълнение
3. Много API(Application Programming Interface) за научаване и др.;

Конфигурационният файл на Hibernate:

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE hibernate-configuration PUBLIC  
 "-//Hibernate/Hibernate Configuration DTD 3.0//EN"  
 "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">  
<hibernate-configuration>  
 <session-factory>  
 <property name="connection.driver\_class">com.mysql.cj.jdbc.Driver</property>  
 <property name="connection.url">jdbc:mysql://localhost:3306/inventory</property>  
 <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>  
 <property name="connection.username">root</property>  
 <property name="connection.password">root1</property>  
 <property name="connection.pool\_size">3</property>  
 <property name="current\_session\_context\_class">thread</property>  
 <property name="show\_sql">true</property>  
 <property name="format\_sql">true</property>  
 <property name="hbm2ddl.auto">update</property>  
  
  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Admin"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Condition"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Customer"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.CustomerBoard"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Mol"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Product"/>  
 <mapping class="bg.tu\_varna.sit.inventorymanagement.data.entities.Amortization"/>  
  
 </session-factory>  
</hibernate-configuration>

Клас Connection за връзка с базата:

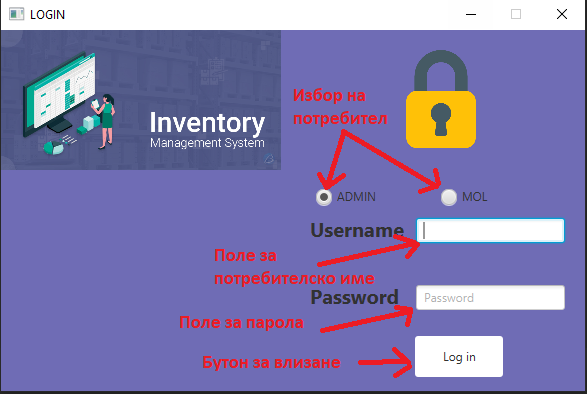
package bg.tu\_varna.sit.inventorymanagement.data.access;  
  
import org.apache.log4j.Logger;  
import org.hibernate.Session;  
import org.hibernate.SessionFactory;  
import org.hibernate.cfg.Configuration;  
  
public class Connection {  
 private static final Logger *log* = Logger.*getLogger*(Connection.class);  
 private static SessionFactory *sessionFactory*;  
 static {  
 try{  
 *sessionFactory* = new Configuration().configure().buildSessionFactory();  
 }catch (Throwable ex){  
 *log*.error("Initial SessionFactory created failed" + ex);  
 }  
 }  
  
 public static Session openSession(){  
 return *sessionFactory*.openSession();  
 }  
  
 public static void openSessionClose(){  
 *sessionFactory*.close();  
 }  
}

Заявки за промяна на продукт и извличане на всички продукти :

public void update(Product obj) {  
 Session session = Connection.*openSession*();  
 Transaction transaction = session.beginTransaction();  
 try{  
 session.update(obj);  
 *log*.info("Product updated successfully");  
 }catch (Exception e)  
 {  
 *log*.error("Product update error" + e.getMessage());  
 }finally {  
 transaction.commit();  
 session.close();  
 }  
}

public List<Product> getAll() {  
 Session session = Connection.*openSession*();  
 Transaction transaction = session.beginTransaction();  
 List<Product> products = new LinkedList<>();  
 try{  
 String jpql = "SELECT p FROM Product p ORDER BY p.description";  
 products.addAll(session.createQuery(jpql, Product.class).getResultList());  
 *log*.info("Successfully got all Products!");  
 }catch (Exception e){  
 *log*.error("Get Products error: " +e.getMessage());  
 }finally {  
 transaction.commit();  
 session.close();  
 }  
 return products;  
}

**Реализация на бизнес логика и графичен интерфейс**

* **Логин панел:**

**FXML код на панела:**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.PasswordField?>  
<?import javafx.scene.control.RadioButton?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.control.ToggleGroup?>  
<?import javafx.scene.image.Image?>  
<?import javafx.scene.image.ImageView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.text.Font?>  
  
<BorderPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" style="-fx-background-color: #6F6CB5;" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1">  
 <left>  
 <AnchorPane prefHeight="400.0" prefWidth="235.0" BorderPane.alignment="CENTER">  
 <children>  
 <ImageView fitHeight="172.0" fitWidth="280.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../../../../../images/68747470733a2f2f7777772e62697a34736f6c7574696f6e732e636f6d2f626c6f672f77702d636f6e74656e742f75706c6f6164732f323032302f30372f62616e6e65725f395f4a756c2e6a7067.jpg" />  
 </image>  
 </ImageView>  
 </children>  
 </AnchorPane>  
 </left>  
 <center>  
 <AnchorPane prefHeight="400.0" prefWidth="365.0" BorderPane.alignment="CENTER">  
 <children>  
 <ImageView fitHeight="119.0" fitWidth="98.0" layoutX="111.0" layoutY="20.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@../../../../../images/lock.png" />  
 </image>  
 </ImageView>  
 <Label layoutX="29.0" layoutY="185.0" text="Username">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <Label layoutX="29.0" layoutY="252.0" text="Password">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Label>  
 <TextField fx:id="username" layoutX="135.0" layoutY="188.0" promptText="Username" />  
 <PasswordField fx:id="password" layoutX="135.0" layoutY="255.0" promptText="Password" />  
 <Button fx:id="logButton" layoutX="134.0" layoutY="306.0" mnemonicParsing="false" onAction="#userLogin" prefHeight="40.0" prefWidth="88.0" style="-fx-background-color: #FFFFFF;" text="Log in" />  
 <Label fx:id="wrongLogin" layoutX="35.0" layoutY="127.0" prefHeight="18.0" prefWidth="266.0" textFill="RED" />  
 <RadioButton fx:id="asAdmin" layoutX="35.0" layoutY="158.0" mnemonicParsing="false" selected="true" text="ADMIN">  
 <toggleGroup>  
 <ToggleGroup fx:id="ch" />  
 </toggleGroup>  
 </RadioButton>  
 <RadioButton fx:id="asMol" layoutX="160.0" layoutY="158.0" mnemonicParsing="false" text="MOL" toggleGroup="$ch" />  
 </children>  
 </AnchorPane>  
 </center>  
</BorderPane>

**Комуникация на логин панела с бизнес логиката:**

@FXML  
public void userLogin()  
{  
 if(username.getText().equals("") && password.getText().equals(""))  
 {  
 wrongLogin.setText("Insert your username and password!");  
 }  
 else if(password.getText().equals(""))  
 {  
 wrongLogin.setText("Insert your password!");  
 username.setText("");  
 password.setText("");  
 }  
 else if(username.getText().equals("") )  
 {  
 wrongLogin.setText("Insert your username!");  
 username.setText("");  
 password.setText("");  
 }  
 else  
 {  
 if (asAdmin.isSelected())  
 {  
 AdminListViewModel adminFind = new AdminListViewModel(username.getText(), password.getText());  
 if (logService.isAdminExist(adminFind))  
 {  
 try  
 { *whichUser*=false;  
 s.close();  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource(*ADMIN\_VIEW*));  
 Stage stage = new Stage();  
 fxmlLoader.setController(new AdminController(stage));  
 Parent root1 = (Parent) fxmlLoader.load();  
 stage.setScene(new Scene(root1));  
 stage.show();  
 } catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 else  
 {  
 wrongLogin.setText("No such Admin!");  
 username.setText("");  
 password.setText("");  
 }  
 } else if (asMol.isSelected())  
 {  
 MolListViewModel molFind = new MolListViewModel(username.getText(), password.getText());  
  
 if (logService.isMolExist(molFind))  
 {  
 try  
 {  
 *whichUser*=true;  
 s.close();  
 FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource(*MOL\_VIEW*));  
 Stage stage = new Stage();  
 fxmlLoader.setController(new MolController(stage));  
 Parent root2 = fxmlLoader.load();  
 stage.setScene(new Scene(root2));  
 stage.show();  
 } catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 else  
 {  
 wrongLogin.setText("No such Mol!");  
 username.setText("");  
 password.setText("");  
 }  
 }  
 }  
}

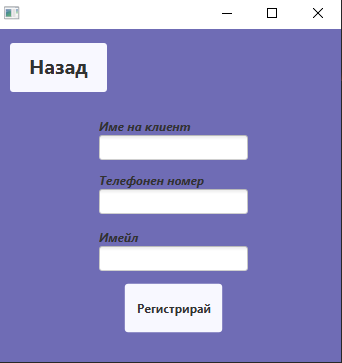
Методът проверява за празни полета и некоректни данни. При натискане на бутона за влизане, ако има грешка, то тя се показва на екрана като червен текст.

Ако данните са коректни, което се проверява в логин сървиса чрез методите *IsAdminExist* и *IsMolExist,* логин панелът се скрива и на негово място се зарежда Админ панелът или Мол панелът, в зависимост от избрания радио бутон.

*  **Админ панел**

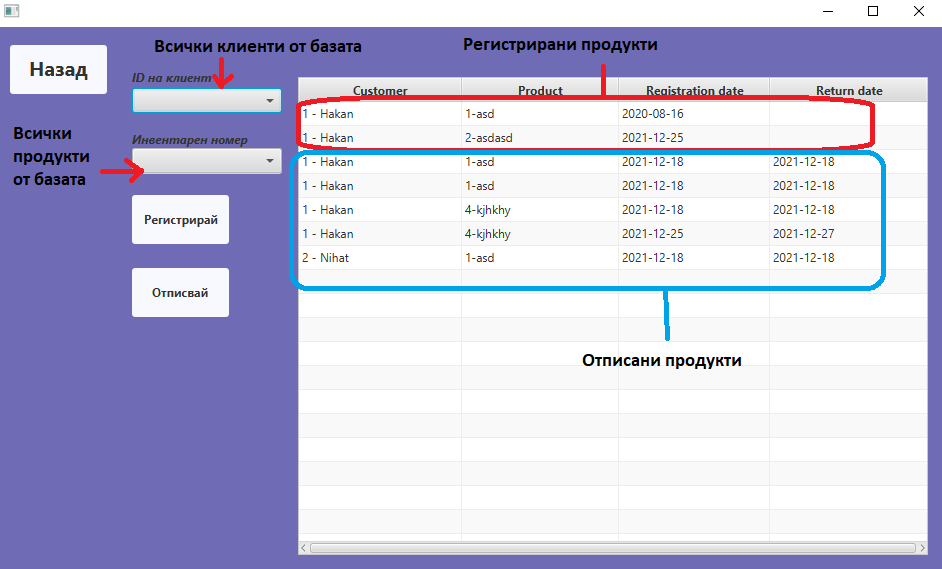
Всички бутони пренасочват потребителя към панела за съответната операция

* **Панел за регистриране на клиент**

****

При натискане на бутона „Регистрирай“ се проверяват полетата. При празни полета или при наличие на клиента в базата регистрацията е неуспешна. Проверката за същестуващ клиент се прави в сървиса чрез метода *addTheCustomer.*

@FXML  
public void addCustomer()  
{ CustomerListViewModel addCus = new CustomerListViewModel(cusName.getText(),cusTelNum.getText(),cusEmail.getText());  
 if(cusName.getText().equals("") || cusTelNum.getText().equals("") || cusEmail.getText().equals("")) {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*,"Please,fill all fields!", ButtonType.*OK*);  
 alert.show();  
 }  
 else {  
 isExist = cusService.addTheCustomer(addCus);  
 if (isExist) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*, "The customer has been registered successfully!", ButtonType.*OK*);  
 alert.show();  
 } else {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*, "The customer has been already added to customer's board!", ButtonType.*OK*);  
 alert.show();  
 }  
 }  
  
}

* **Панел за регистриране и отписване на продукт от картон на клиент**

В комбобоксовете са заредени всички клиенти и продукти, а в таблицата може да видим всички регистрирани и отписани продукти.

Ако не са избрани клиент и продукт, потребителят получава съобщение за грешка.

**FXML на панела:**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.ComboBox?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane prefHeight="549.0" prefWidth="946.0" style="-fx-background-color: #6F6CB5;" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1">  
 <children>  
 <Button fx:id="cusButton" layoutX="136.0" layoutY="168.0" mnemonicParsing="false" onAction="#boardRegistration" prefHeight="48.0" prefWidth="97.0" style="-fx-background-color: #F8F8FF;" text="Регистрирай">  
 <font>  
 <Font name="System Bold" size="12.0" />  
 </font>  
 </Button>  
 <Label layoutX="136.0" layoutY="42.0" prefHeight="17.0" prefWidth="97.0" text="ID на клиент">  
 <font>  
 <Font name="System Bold Italic" size="12.0" />  
 </font>  
 </Label>  
 <Label layoutX="103.0" layoutY="95.0" prefHeight="17.0" prefWidth="108.0">  
 <font>  
 <Font name="System Bold Italic" size="12.0" />  
 </font>  
 </Label>  
 <ComboBox fx:id="customersComboBox" layoutX="136.0" layoutY="61.0" prefWidth="150.0" />  
 <ComboBox fx:id="productsComboBox" layoutX="136.0" layoutY="121.0" prefWidth="150.0" />  
 <Label layoutX="136.0" layoutY="104.0" prefHeight="17.0" prefWidth="129.0" text="Инвентарен номер">  
 <font>  
 <Font name="System Bold Italic" size="12.0" />  
 </font>  
 </Label>  
 <TableView fx:id="boardTable" layoutX="302.0" layoutY="50.0" onMouseClicked="#getTheRow" prefHeight="478.0" prefWidth="630.0">  
 <columns>  
 <TableColumn fx:id="customerProduct" prefWidth="163.0" text="Customer" />  
 <TableColumn fx:id="productCustomer" prefWidth="157.0" text="Product" />  
 <TableColumn fx:id="registerProductDate" prefWidth="151.0" text="Registration date" />  
 <TableColumn fx:id="returnProductDate" prefWidth="158.0" text="Return date" />  
 </columns>  
 </TableView>  
 <Button fx:id="cusButton11" layoutX="14.0" layoutY="18.0" mnemonicParsing="false" onAction="#goBack" prefHeight="48.0" prefWidth="97.0" style="-fx-background-color: #F8F8FF;" text="Назад">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Button>  
 <Button fx:id="cusButton1" layoutX="136.0" layoutY="241.0" mnemonicParsing="false" onAction="#boardRemove" prefHeight="48.0" prefWidth="97.0" style="-fx-background-color: #F8F8FF;" text="Отпиши">  
 <font>  
 <Font name="System Bold" size="12.0" />  
 </font>  
 </Button>  
 </children>  
</AnchorPane>

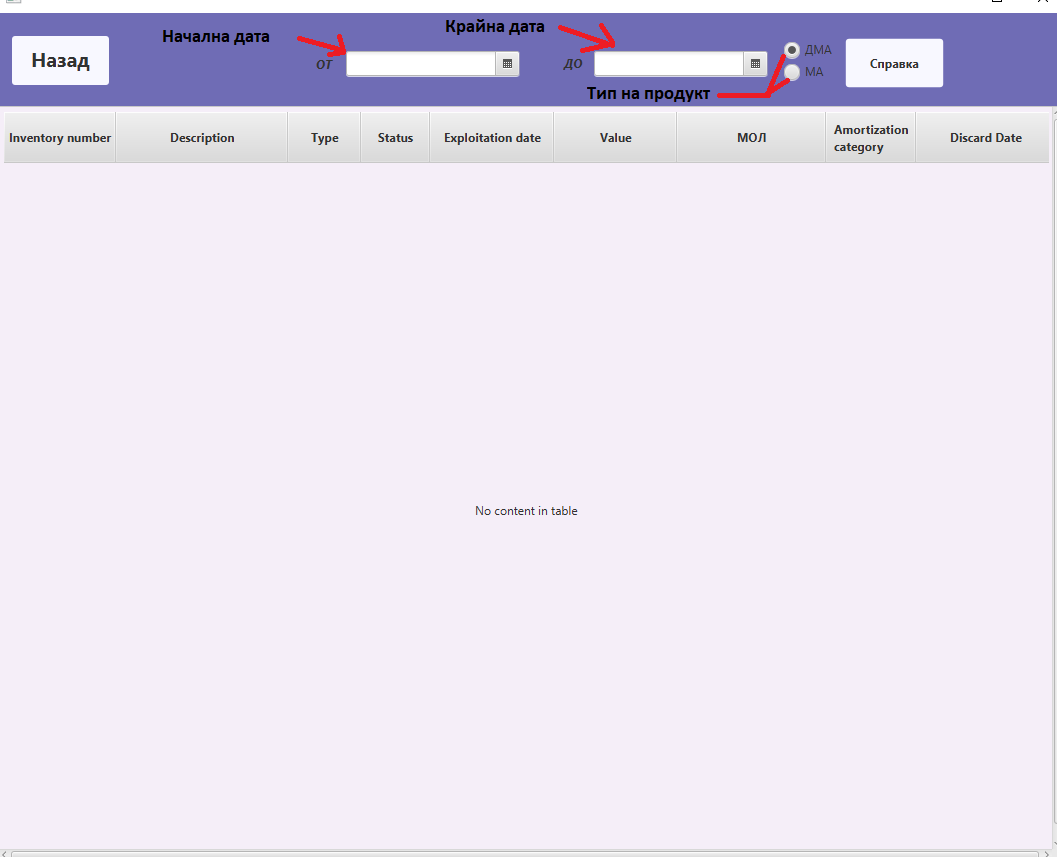
**Част от контролера на панела:**

1. Метод за регистриране на продукт към картон на клиент:
2. @FXML  
   public void boardRegistration(){  
     
    if(customersComboBox.getValue()==null || productsComboBox.getValue()==null)  
    {  
    Alert alert=new Alert(Alert.AlertType.*ERROR*,"Please,fill all fields!", ButtonType.*OK*);  
    alert.show();  
    }  
     
    else{  
    temp.clear();  
    boardTable.setItems(temp);  
    LocalDate localDate = LocalDate.*now*();  
    CustomerBoardListViewModel addToBoard = new CustomerBoardListViewModel(customerService.listViewToEntity(customersComboBox.getValue()),productService.listViewToEntity(productsComboBox.getValue()),localDate);  
    isExist= customerBoardService.addToTheBoard(addToBoard);  
     
    if(isExist==0) {  
    Alert alert=new Alert(Alert.AlertType.*ERROR*,"The product is not available!",ButtonType.*OK*);  
    alert.show();  
    }  
    else if(isExist==1)  
    {  
    Alert alert=new Alert(Alert.AlertType.*ERROR*,"The product is discarded!",ButtonType.*OK*);  
    alert.show();  
    }  
    else  
    {  
    Alert alert=new Alert(Alert.AlertType.*INFORMATION*,"The product has been added successfully!",ButtonType.*OK*);  
    alert.show();  
    }  
    temp = customerBoardService.getAllBoards();  
    boardTable.setItems(temp);  
    }  
   }

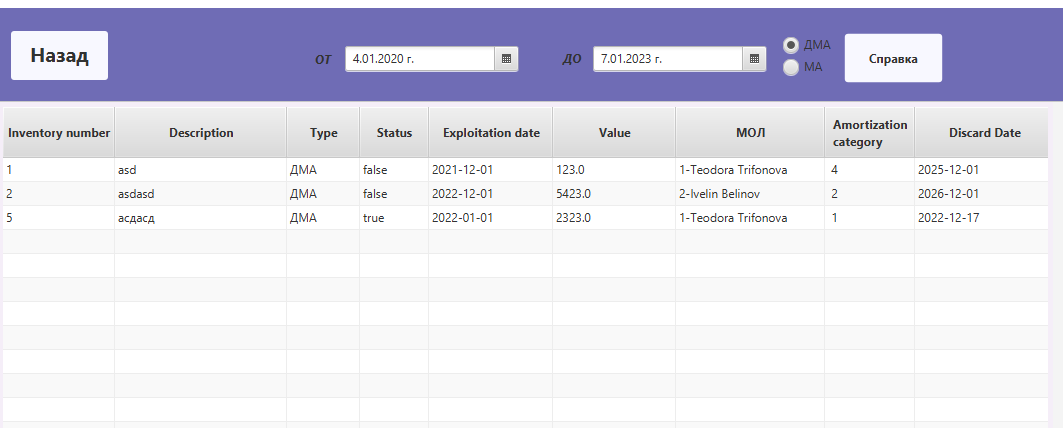
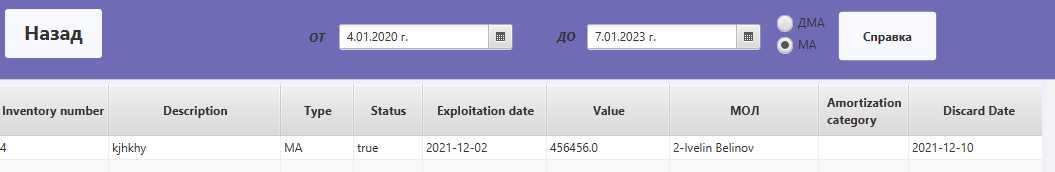
2.Метод за отписване на продукт от картон на клиент:

@FXML  
public void boardRemove(ActionEvent event){  
 LocalDate localDate2 = LocalDate.*now*();  
 forRemove.setReturnDate(localDate2);  
 temp.clear();  
  
 if(customerBoardService.returnTheProduct(forRemove))  
 {  
 Alert alert=new Alert(Alert.AlertType.*INFORMATION*,"The product has been returned successfully!",ButtonType.*OK*);  
 alert.show();  
 }  
  
 else  
 {  
 Alert alert=new Alert(Alert.AlertType.*WARNING*,"The product is already returned!",ButtonType.*OK*);  
 alert.show();  
 }  
  
 temp = customerBoardService.getAllBoards();  
 boardTable.setItems(temp);  
  
}

* Панел за справка на продукти по категории:



След избор на периода и категорията се натиска бутонът „Справка“ и в таблицата се показват продуктите за този период:



**FXML на панела:**

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.DatePicker?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.RadioButton?>  
<?import javafx.scene.control.ScrollPane?>  
<?import javafx.scene.control.TableColumn?>  
<?import javafx.scene.control.TableView?>  
<?import javafx.scene.control.ToggleGroup?>  
<?import javafx.scene.layout.BorderPane?>  
<?import javafx.scene.layout.Pane?>  
<?import javafx.scene.text.Font?>  
  
<BorderPane prefHeight="850.0" prefWidth="1068.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1">  
 <top>  
 <Pane prefHeight="93.0" prefWidth="1162.0" style="-fx-background-color: #6F6CB5;" BorderPane.alignment="CENTER">  
 <children>  
 <Button fx:id="getProductsByTypeButton" layoutX="848.0" layoutY="26.0" mnemonicParsing="false" onAction="#getAllProductsByType" prefHeight="48.0" prefWidth="97.0" style="-fx-background-color: #F8F8FF;" text="Справка">  
 <font>  
 <Font name="System Bold" size="12.0" />  
 </font>  
 </Button>  
 <DatePicker fx:id="fromDate" layoutX="348.0" layoutY="38.0" onAction="#getFromDate" />  
 <Label layoutX="318.0" layoutY="42.0" prefHeight="18.0" prefWidth="19.0" text="ОТ">  
 <font>  
 <Font name="System Bold Italic" size="12.0" />  
 </font>  
 </Label>  
 <Label layoutX="566.0" layoutY="41.0" prefHeight="18.0" prefWidth="19.0" text="ДО">  
 <font>  
 <Font name="System Bold Italic" size="12.0" />  
 </font>  
 </Label>  
 <DatePicker fx:id="toDate" layoutX="596.0" layoutY="38.0" onAction="#getToDate" />  
 <RadioButton fx:id="dmaProducts" layoutX="786.0" layoutY="28.0" mnemonicParsing="false" selected="true" text="ДМА">  
 <toggleGroup>  
 <ToggleGroup fx:id="type" />  
 </toggleGroup>  
 </RadioButton>  
 <Button fx:id="backButton" layoutX="14.0" layoutY="23.0" mnemonicParsing="false" onAction="#goBack" prefHeight="48.0" prefWidth="97.0" style="-fx-background-color: #F8F8FF;" text="Назад">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </Button>  
 <RadioButton fx:id="maProducts" layoutX="786.0" layoutY="50.0" mnemonicParsing="false" text="МА" toggleGroup="$type" />  
 </children>  
 </Pane>  
 </top>  
 <center>  
 <ScrollPane prefHeight="757.0" prefWidth="1102.0" BorderPane.alignment="CENTER">  
 <content>  
 <TableView fx:id="allProdByTypeTable" prefHeight="756.0" prefWidth="1055.0" style="-fx-background-color: #F5EEF8;">  
 <columns>  
 <TableColumn fx:id="inventoryCol" prefWidth="112.0" text="Inventory number" />  
 <TableColumn fx:id="descriptionCol" prefWidth="172.0" text="Description" />  
 <TableColumn fx:id="typeCol" prefWidth="73.0" text="Type" />  
 <TableColumn fx:id="statCol" prefWidth="69.0" text="Status" />  
 <TableColumn fx:id="expCol" prefWidth="124.0" text="Exploitation date" />  
 <TableColumn fx:id="valueCol" prefWidth="123.0" text="Value" />  
 <TableColumn fx:id="molCol" prefWidth="149.0" text="МОЛ" />  
 <TableColumn fx:id="amortCol" minWidth="0.0" prefWidth="90.0" text="Amortization&#10;category&#10;&#10;" />  
 <TableColumn fx:id="disCol" prefWidth="140.0" text="Discard Date" />  
 </columns>  
 </TableView>  
 </content>  
 </ScrollPane>  
 </center>  
</BorderPane>

**Част от контролера на панела:**

@FXML  
public void getFromDate()  
{  
 myFromDate=fromDate.getValue();  
}  
  
@FXML  
public void getToDate(){  
  
 myToDate=toDate.getValue();  
}  
  
  
  
@FXML  
public void getAllProductsByType()  
{  
 if(dmaProducts.isSelected())  
 whatType=true;  
 else if(maProducts.isSelected())  
 whatType=false;  
 if(myFromDate==null || myToDate==null)  
 {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*,"Please,fill all fields!", ButtonType.*OK*);  
 alert.show();  
 }  
 else {  
 ObservableList<ProductListViewModel> productsInPeriod = productService.getAllProductsByTypeInPeriod(myFromDate, myToDate, whatType);  
 allProdByTypeTable.setItems(productsInPeriod);  
 }  
}

**Използване на log4J**

Файл за свойствата на log4J:

log4j.rootLogger=ERROR, console, file  
  
log4j.appender.console=org.apache.log4j.ConsoleAppender  
log4j.appender.console.layout=org.apache.log4j.PatternLayout  
log4j.appender.console.layout.ConversionPattern=[%t] %-5p %c %x - %m%n  
  
log4j.appender.file=org.apache.log4j.RollingFileAppender  
log4j.appender.file.layout=org.apache.log4j.PatternLayout  
log4j.appender.file.layout.ConversionPattern=[%t] %-5p %c %x - %m%n  
log4j.appender.file.File=application.log

В програмата log4J е използван най-често за грешки при обработване на заявките и в методите за проверка на съществуващ елемент (продукт, клиент и т.н)

@Override  
public void save(Product obj) {  
 Session session = Connection.*openSession*();  
 Transaction transaction = session.beginTransaction();  
 try{  
 session.save(obj);  
 *log*.info("Product saved successfully");  
 }catch (Exception e)  
 {  
 *log*.error("Product save error" + e.getMessage());  
 }finally {  
 transaction.commit();  
 session.close();  
 }  
}

@Override  
public List<Product> getAll() {  
 Session session = Connection.*openSession*();  
 Transaction transaction = session.beginTransaction();  
 List<Product> products = new LinkedList<>();  
 try{  
 String jpql = "SELECT p FROM Product p ORDER BY p.description";  
 products.addAll(session.createQuery(jpql, Product.class).getResultList());  
 *log*.info("Successfully got all Products!");  
 }catch (Exception e){  
 *log*.error("Get Products error: " +e.getMessage());  
 }finally {  
 transaction.commit();  
 session.close();  
 }  
 return products;  
}

**Junit тестове**

Направени са тестове за цялата бизнес логика и и за едно репоситори( понеже всички репоситорита са с еднакви заявки).

* **Част от направените тестове:**

Проверява се дали за дати в бъдеще,които не са регистрирани, ще върне празен списък.

@Test  
void getAllProductsInPeriod() {  
 List<Product> temp = new ArrayList<>();  
 *assertEquals*(temp,productService.getAllProductsInPeriod(LocalDate.*of*(2030,10,10),LocalDate.*of*(2035,5,12)));  
}

Проверява дали получаваме точно продукта, чийто инвентарен номер сме подали.

@Test  
void getProductById() {  
 product= productRepository.getById(1);  
 *assertEquals*(product,productRepository.getById(1));  
  
}

Проверява дали въведеният админ се запазва успешно в базата:

void save() {  
 List<Admin> admins =adminRepository.getAll();  
 adminRepository.save(admin);  
 Assertions.*assertNotEquals*(admins,adminRepository.getAll());  
}

След направените тестове можем да потвърдим, че програмата работи коректно и е готова за използване!