***Logo

Description automatically generated***

Технически Университет – Варна

ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ

КАТЕДРА „СОФТУЕРНИ И ИНТЕРНЕТ ТЕХНОЛОГИИ“

Специалност „Софтуерни и интернет технологии“

Проект по Обектно – ориентирано програмиране 2 част

Тема 10

Система за организиране и следене на

инвентар

**Изготвили:**

**Герман Петев Копчев ф.н. 19621656, СИТ, III курс, 2 б групa**

**Тиодор Веселинов Бурилков ф.н. 19621650, СИТ, III курс, 2 б група**

**Условие на проекта**

X. Система за организиране и следене на

Инвентар

Да се разработи информационна система, предоставяща възможност за организиране на информация за инвентаризация.

Системата поддържа два вида потребители – администратор и материално отговорно

лице (МОЛ).

Операции за работа с потребители:

* създаване на МОЛ-ове от администратор.

Системата поддържа операции за работа с инвентар:

* регистриране на продукт с инвентарен номер, описание, тип (ДМА, МА), степен

на амортизация (за ДМА);

* въвеждане на критерий за бракуване – години, състояние или др.;
* регистриране на клиенти;
* регистриране на продукти към картоните на клиентите;
* отписване на продукт от картон;
* автоматично изчисляване за прехвърляне от ДМА към МА;
* бракуване на продукт.

Системата поддържа Справки по произволен период за:

* клиенти със регистрираните на тяхно име продукти;
* пълен списък на продуктите;
* списък по категории – ДМА,МА;
* бракувани продукти;
* статус на продукта – наличен, липсващ;

Системата поддържа Известия за:

* процес на трансформация от ДМА към МА
* бракуване на продукт

Анализ на проблема

Системата трябва да поддържа два вида потребители, които имат различни нива на достъп до различните функции в програмата. Всеки потребител ще има уникални данни за идентификация.

Потребител от тип „Администратор“ ще има достъп до всички функционалности на системата, а те са:

* създаване на нов потребител от тип материално отговорно лице (МОЛ);
* Регистриране на нов продукт към конкретния МОЛ;
* Регистриране на нов продукт към картона на потребител;
* Регистриране на нов продукт който е за бракуване
* Регистриране на нов клиент.
* Справки по произволен период за клиенти със регистрираните на тяхно име продукти;
* Справки по произволен период пълен списък на продуктите;
* Справки по произволен период списък по категории – ДМА,МА;
* Справки по произволен период бракувани продукти;
* Справки по произволен период статус на продукта – наличен, липсващ;

Потребители от тип „Материално отговорно лице“ ще има достъп до следните функционалности но системата:

* Регистриране на нов продукт към конкретния МОЛ;
* Регистриране на нов продукт към картона на потребител;
* Регистриране на нов продукт който е за бракуване
* Регистриране на нов клиент.
* Справки по произволен период за клиенти със регистрираните на тяхно име продукти;
* Справки по произволен период пълен списък на продуктите;
* Справки по произволен период списък по категории – ДМА,МА;
* Справки по произволен период бракувани продукти;
* Справки по произволен период статус на продукта – наличен, липсващ;

Системата следи кой вид потребител се е вписал. При вписване на потребител ще се извършва проврка за коректност на данните и при липса на съвпадение ще се извежда съощение, и ще бъде отказан достъп. Потребителите ще имат достъп до функционалностите на системата чрез интуитивен потребителски интерфейс.

Структура и модули на проекта

Graphical user interface, diagram

Description automatically generated

За реализацията на възложения проект е избрана многослойна архитектура, представена от слоевете посочени на горната схема, предоставената от нея възможност за чсното разграничаване на нейните модули и по-лесна имплементация на нови функционалности и корекции по системата.

Презентационният слой(Presentation layer) е изграден от три модула – Models, Views и Controllers.

* В модул Views се намират JavaFX файловете, които изграждат потребителския интерфейс и изгледа на системата. Те съдържат основните компуненти на прозорците като диалогови кутии, бутони, менута и т.н.
* В модул Controllers се намират класовете, които осъществяват връзката между въведените от потребителя данни и тяхната обработка в бизнес логиката. Те управляват отделните елементи в JavaFX файловете.
* В модул Models се намират обектите в които се записва въведената от потребителя информация, която се предава на биснес логиката. Чрез тях се постига индиректната връзка между данните от потребителя и записите в базата данни.

Слоят бизнес логика (Business layer) обработва информацията и реализара отделните функционалнои изисквания на системата. Той преобразява модулите в entity обекти, които се използват от слоя за връзката с базата данни.

Слоят за връзката данни се разделя на три модула – Access, Entities и Repositories.

* В модула Access се осъществява същинската връзка с базата данни, чрез която се достъпват отделните записи.
* В модъла Entities се съдържат класове съотверстващи на таблиците от базата данни и връзките между тях. Във всеки един клас са реализирани методи, които позволяват достъването, сравнението и визуализацията на отделните записи от съответната таблица.
* В модула Repositories се намират класове в които се имплементират различните заявки към базта данни.

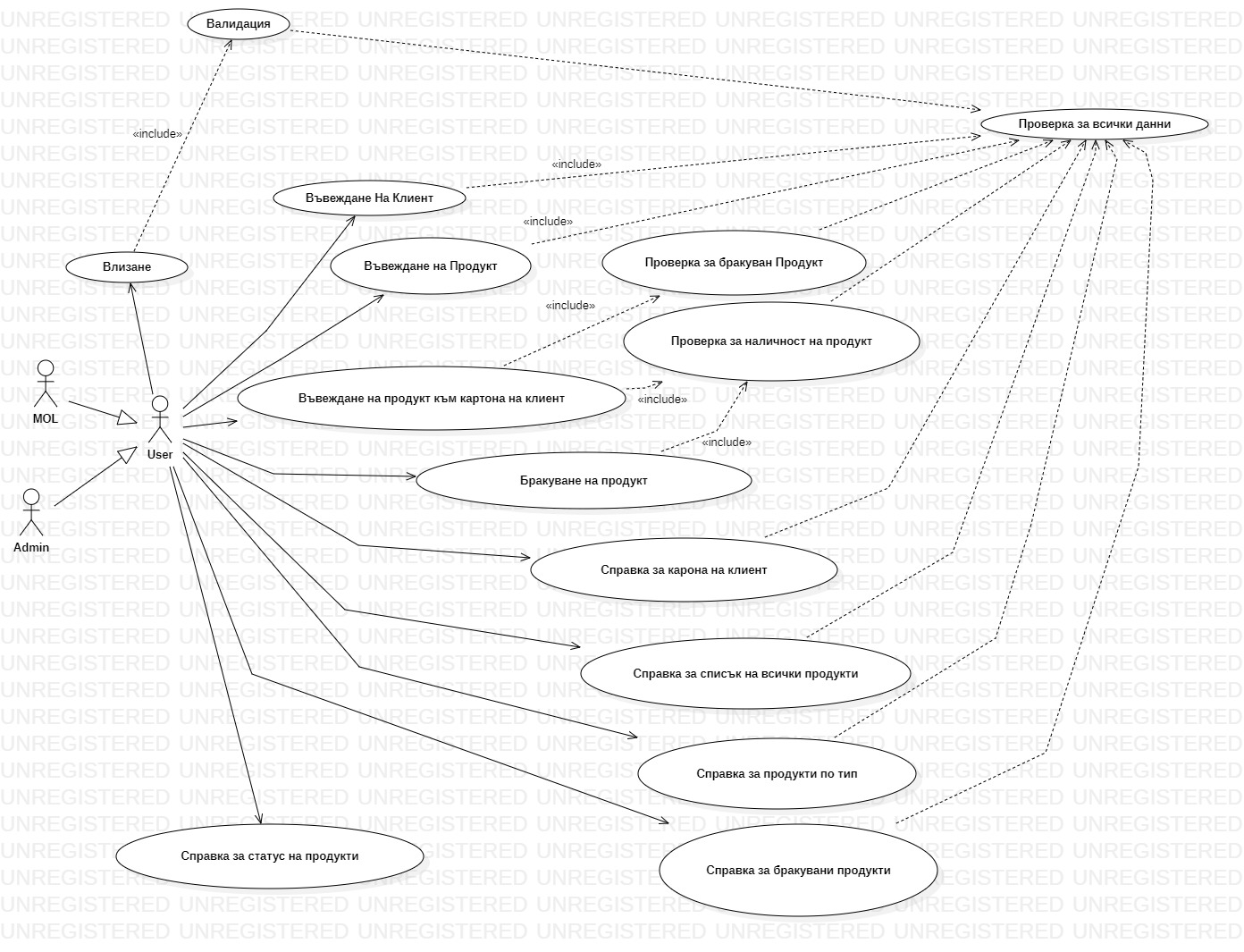
Връзката с базата данни е реализирана чрез Hibernate. Избран е този подход, защото за разлика от JDBC Hibernate се свързва със самата база данни и използва HQL (Hibernate Query Language) за изпълнение на заявките и след това ги свързва резултатите към Java обектите. За реализация на базата данни е избран Microsoft SQL Server, защото е с по-добра производителност, лесен за първоначално настройване и с по-добра защита.

В слоя Utilities са поместени три модула – Configuration, Constants, Images.

* В модула Configuration се намира класът използван за указване на пътя до конфигурацията на логъра за събития.
* В модула Constants се използва за зареждане на различни страници от потребителския интерфейс, класът който се използва за следен на конкретния потребител използващ системата.

Проектиране и реализация на системата

UML Use Case Diagram

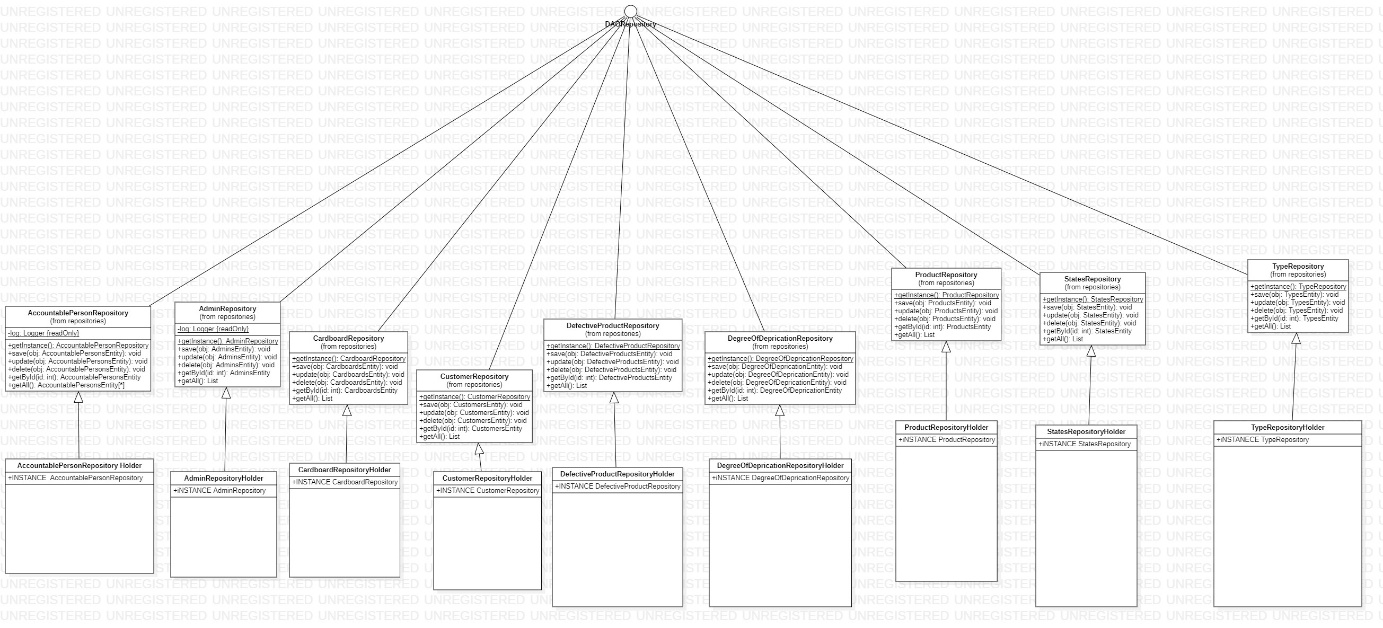


UML Class diagrams

Entities

Diagram, engineering drawing, schematic

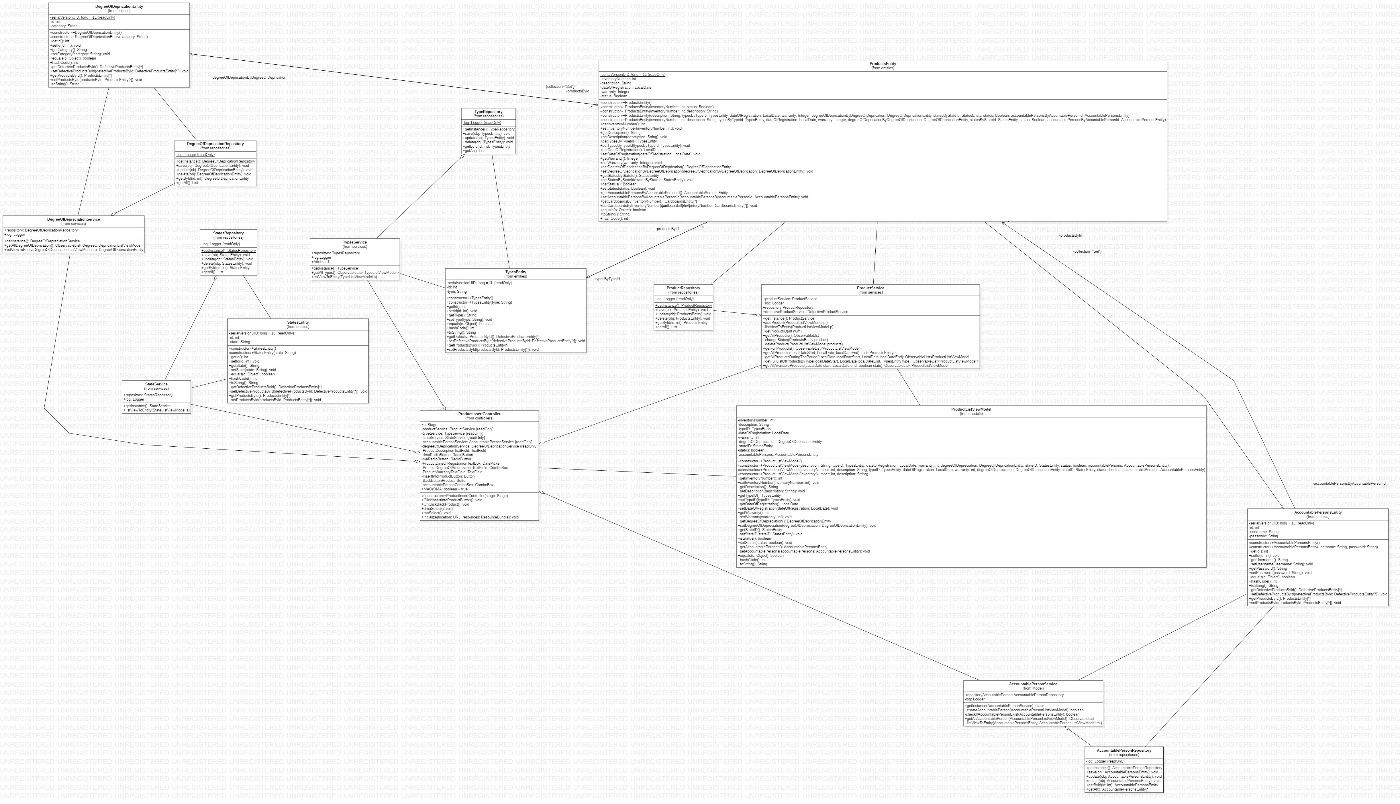
Description automatically generated

Repositories

User LoginDiagram

Description automatically generated

Register Product

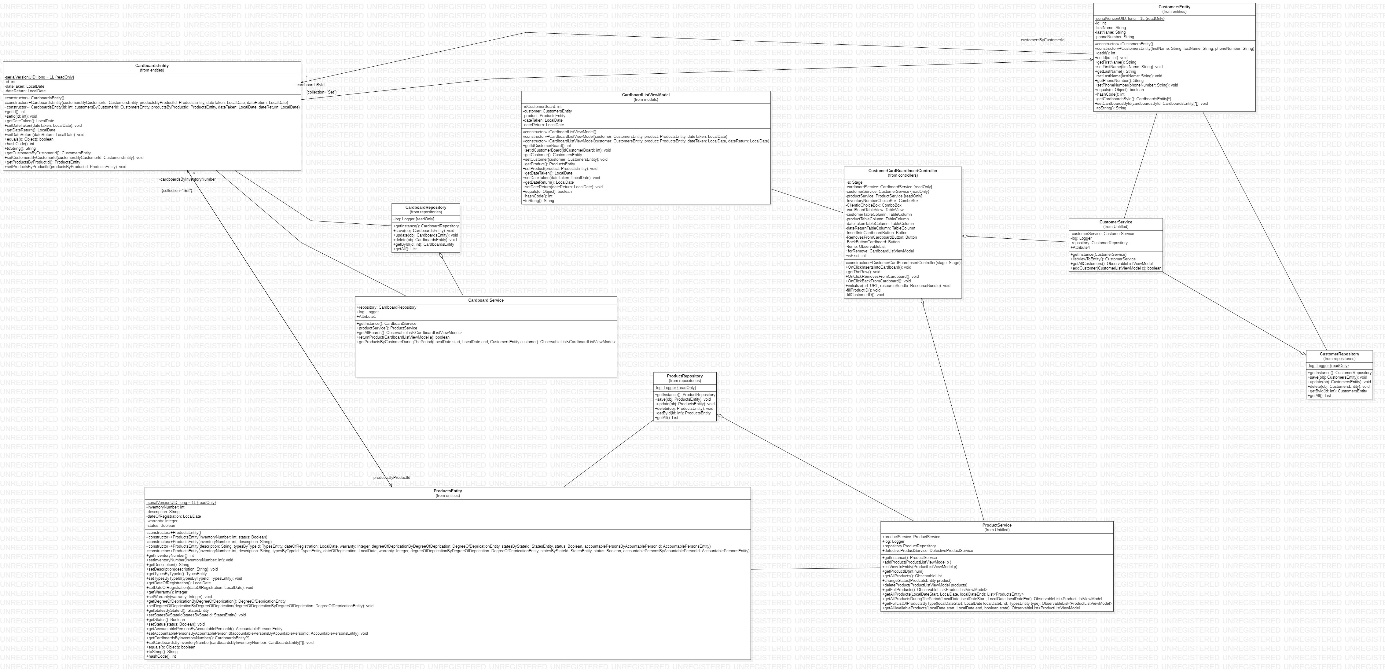


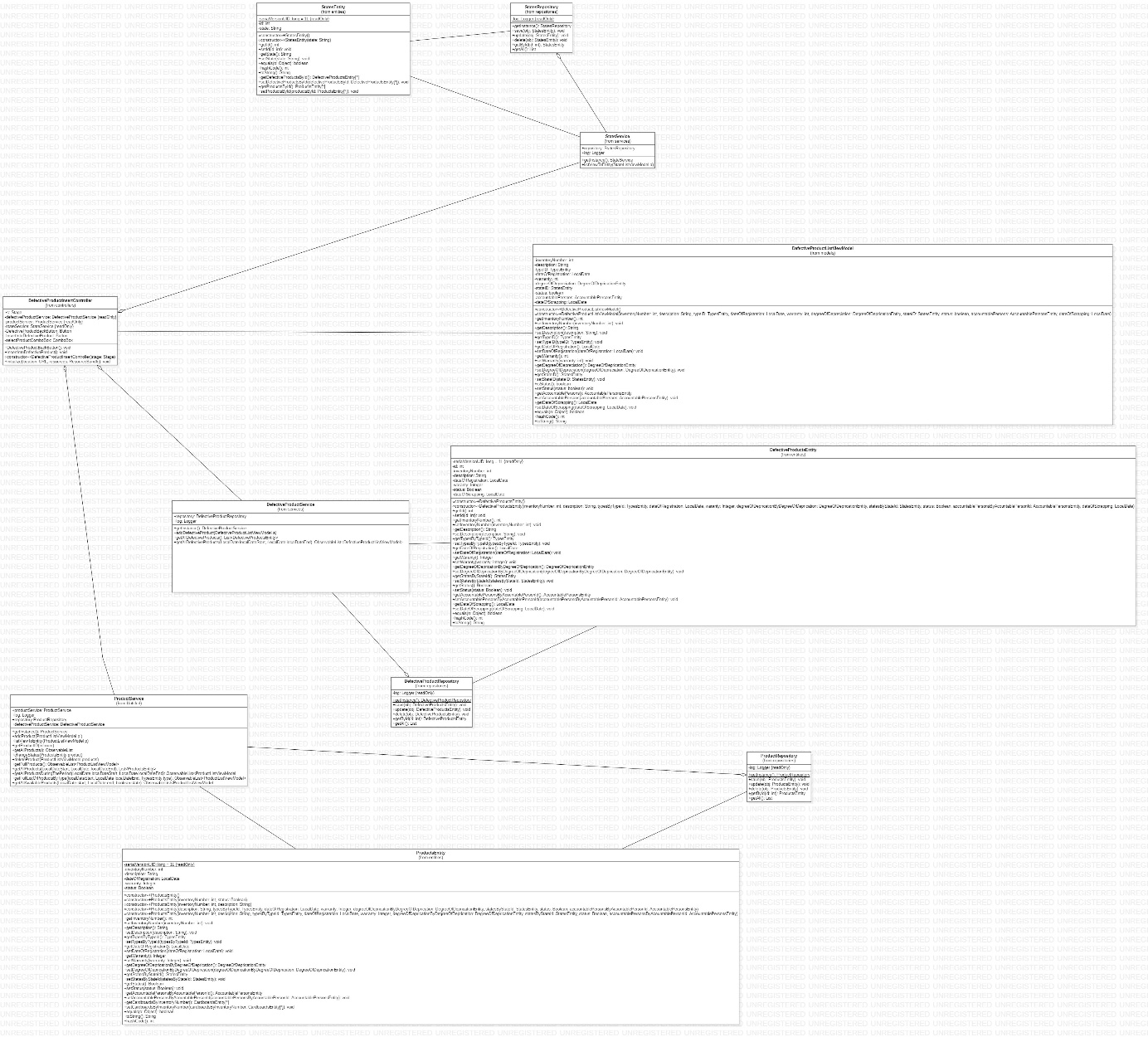
Register Customer

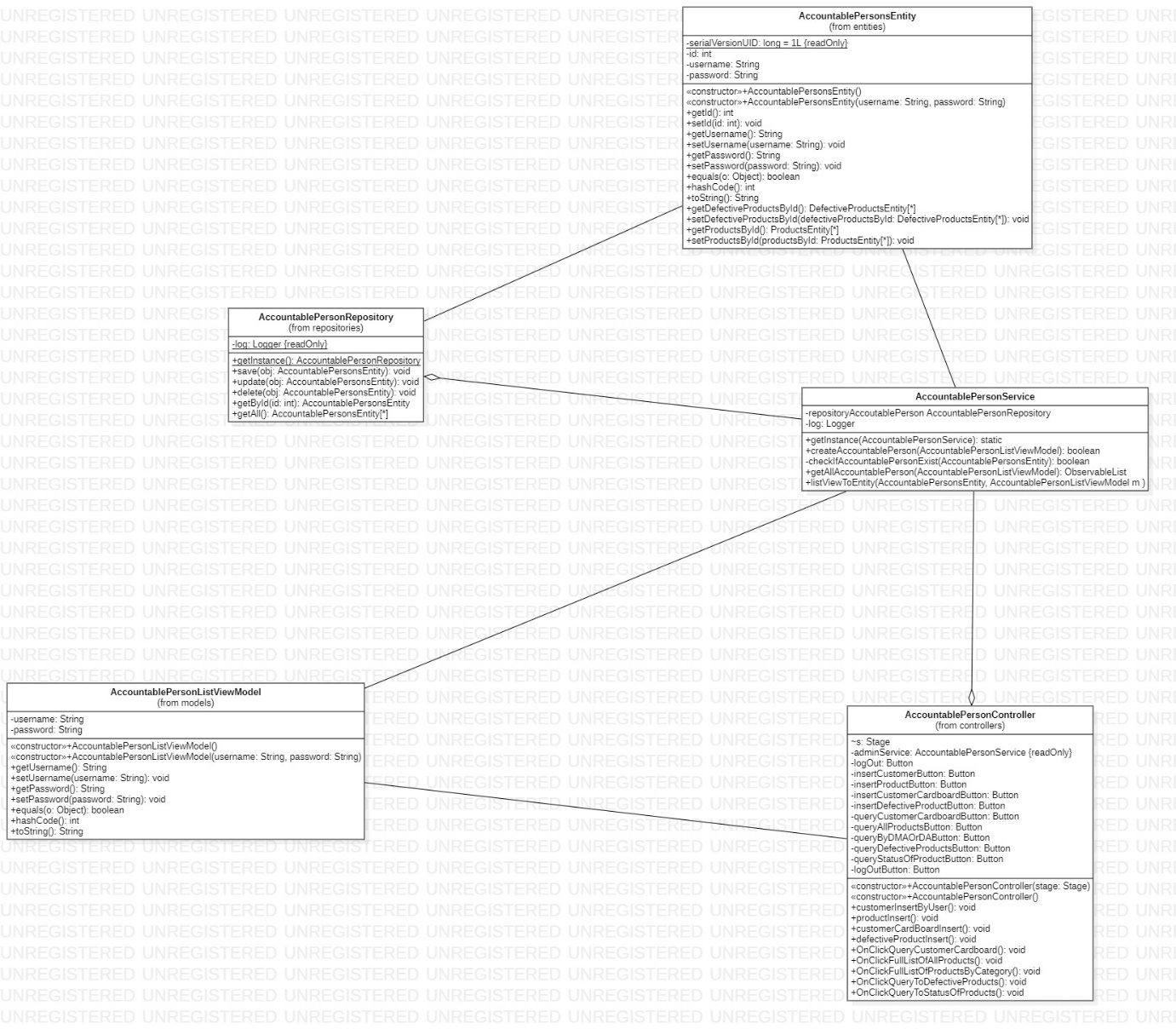
Diagram

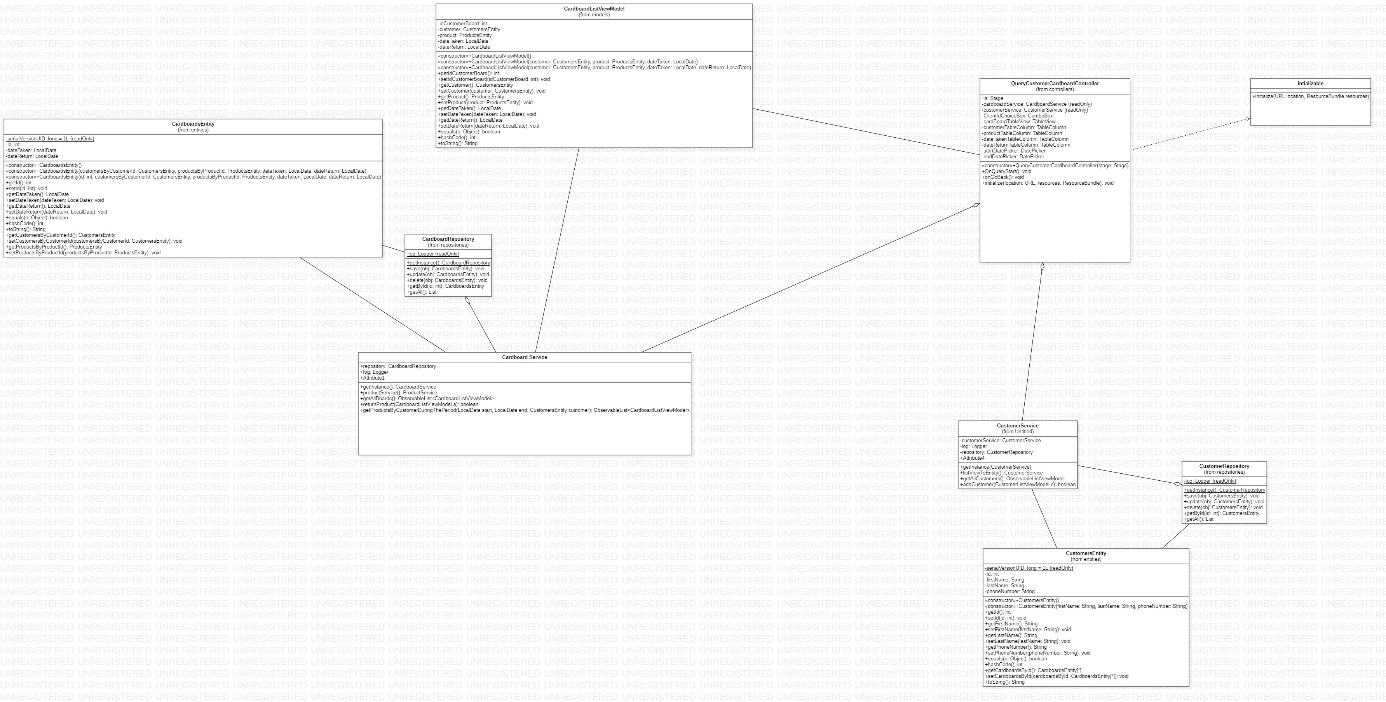
Description automatically generated

Register Cardboard



Register Defective product

Register Accountable person

Query all products in period by customer

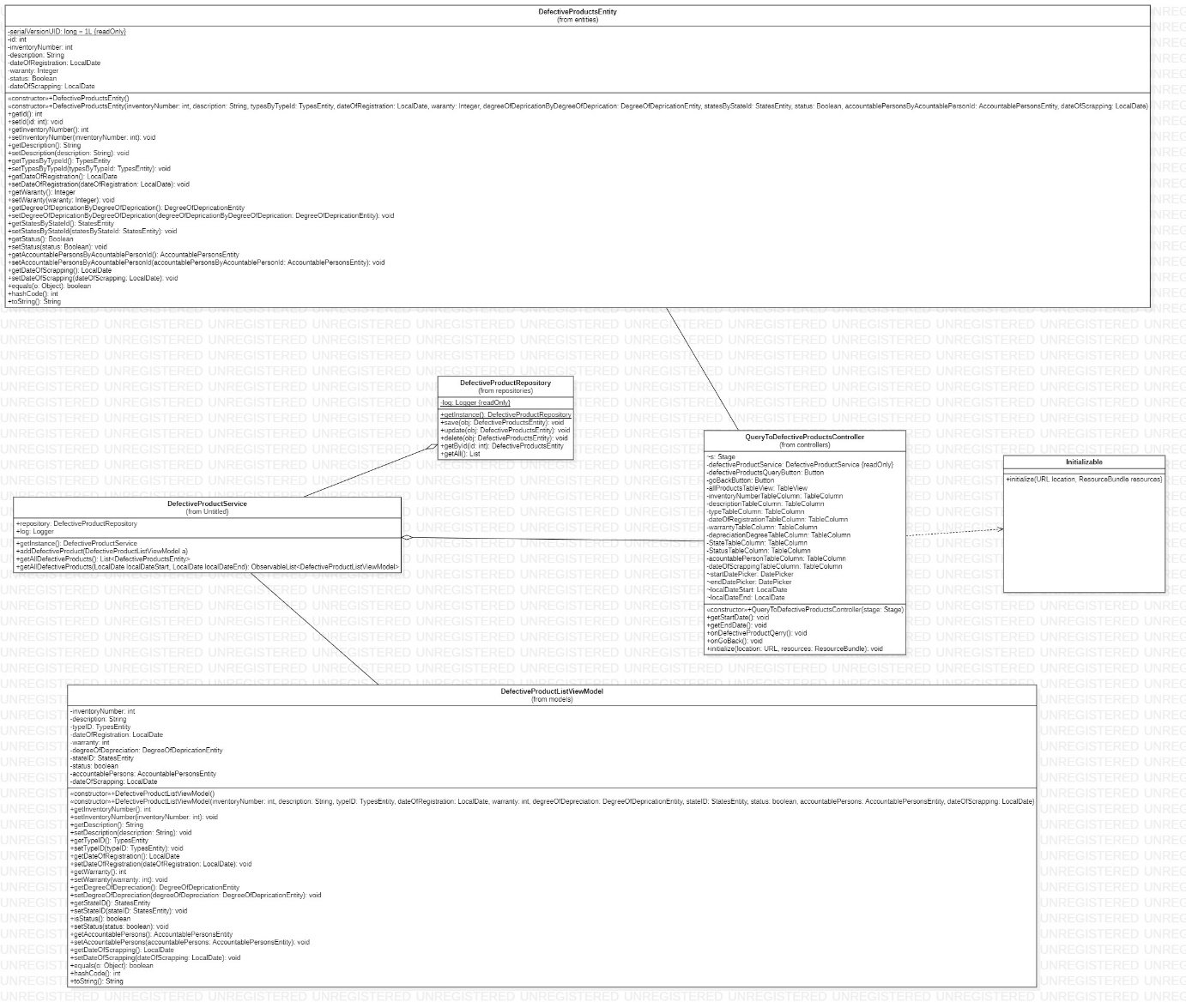
Query all products in period

Diagram

Description automatically generated with low confidence

Query all products in period by type Diagram

Description automatically generated

Query all defective products in period

Query all product in period by status

A picture containing diagram

Description automatically generated

ER диаграма

Graphical user interface, diagram

Description automatically generated

Проектиране и реализация на базата данни:

Тя съдържа общо 9 таблици. Колоните на всичките таблици са наименовани така че лесно да се разбира каква информация ще се съдържа в тях. За статус на продукта се използва тип Boolean(bit), понеже има само два възможни случая (дали е наличен или не). За всички пароли описания, имена се използва varchar, а за полетата които са primer key се използа тип int.

Проектиране и реализация на слоя с база данни:

Слоя за работа с бази от данни е реализиран чрез Hibernate, който симплементиран чрез пакета data, съдържащ три подпакета – access, entities и repositories. В подпапката access се осъществява конфигурацията необходима за връзка с базата данни, чрез класа Connection

Text

Description automatically generated

В подпапкет entities се напират класовете, които са свързни с отделните таблици в базата данни. В тях се описва името на таблицата, отделните колони и типовете данни, както и връзките между таблиците.

Entity за таблицa Customers:

Text

Description automatically generated

Entity за таблица AccountablePersons:Text

Description automatically generated

Entity за таблица Admins:Text

Description automatically generated

По аналогичен начин се създадени класовете описващи останалите таблици от базата данни.

Подпаката repositories съдържа класове, които осъществяват директна връзка до базата данни за всяко едно entity. Всички те имплементират интерфейса DAORepository, който служи за дефениция за методите необходими за комуникация, като например save, update, delete, getById и getAll.

DAORepository:

Text

Description automatically generated

AdminRepository: Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated Text

Description automatically generated

Всеки един клас в подпакет repositories съдържа вграден статичен клас, който се използва в статияния метод getInstance, необходим за създаването на нова инстанция за този клас. И по аналогичен начин са реализирани останалите класове от подпакета

Проектиране и реализация на бизнеслогика и графичен интерфейс:

За реализацията на графияния потребителски интерфейс е използван JavaFX. За целта fxml файловете необходими за визуализацията на раличните прозорци на програмата са поместени в поддиректория presentation.views, част от директория resources.

Във всеки един fxml файл са описани необходимите текстови полета, бутони и изгледи с които взаимодейства потребителя.

Логин страница

Graphical user interface, application

Description automatically generated

Логин страница fxml код

<?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.layout.AnchorPane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane id="LoginScreen" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="310.0" prefWidth="454.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="bg.tu\_varna.sit.inventory.presentation.models.controllers.LoginScreenController">  
 <children>  
 <TextField fx:id="username" layoutX="123.0" layoutY="90.0" />  
 <Label layoutX="168.0" layoutY="116.0" prefHeight="30.0" prefWidth="60.0" text="Парола">  
 <font>  
 <Font name="System Bold" size="14.0" />  
 </font>  
 </Label>  
 <Label layoutX="126.0" layoutY="59.0" prefHeight="30.0" prefWidth="142.0" text="Потребителско Име">  
 <font>  
 <Font name="System Bold" size="14.0" />  
 </font>  
 </Label>  
 <Button fx:id="loginButton" layoutX="155.0" layoutY="177.0" mnemonicParsing="false" onAction="#userLogin" prefHeight="44.0" prefWidth="94.0" text="Log in" />  
<!-- <ImageView fitHeight="257.0" fitWidth="500.0" layoutX="152.0" layoutY="14.0" pickOnBounds="true" preserveRatio="true">  
 <image>  
 <Image url="@Blog\_What-is-Inventory.png" />  
 </image>  
 </ImageView>-->  
 <RadioButton fx:id="asAdmin" layoutX="307.0" layoutY="62.0" mnemonicParsing="false" selected="true" text="Админ">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 <toggleGroup>  
 <ToggleGroup fx:id="ch" />  
 </toggleGroup>  
 </RadioButton>  
 <RadioButton fx:id="asAccountablePerson" layoutX="307.0" layoutY="94.0" mnemonicParsing="false" text="МОЛ" toggleGroup="$ch">  
 <font>  
 <Font name="System Bold" size="20.0" />  
 </font>  
 </RadioButton>  
 <PasswordField fx:id="password" layoutX="123.0" layoutY="142.0" />  
 </children>  
</AnchorPane>

Комуникация на логин страницата с бизнес логиката:

public class LoginScreenController {  
 public Stage s;  
 public static boolean *user*;  
 @FXML  
 private Button loginButton;  
 @FXML  
 private RadioButton asAdmin;  
 @FXML  
 private RadioButton asAccountablePerson;  
 @FXML  
 private TextField username;  
 @FXML  
 private PasswordField password;  
  
 public LoginScreenController(){ }  
 public LoginScreenController(Stage stage) {  
 s = stage;  
 }  
 private final LoginService logService=LoginService.*getInstance*();  
  
 @FXML  
 public void userLogin(ActionEvent actionEvent){  
 AccountablePersonListViewModel molFind = new AccountablePersonListViewModel();  
 AdminListViewModel adminFind = new AdminListViewModel();  
 if(username.getText() == "" || password.getText() == "") {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*,"Please,fill all fields!", ButtonType.*OK*);  
 alert.show();  
 }  
 else {  
  
 if(asAdmin.isSelected()){  
 adminFind = new AdminListViewModel(username.getText(), password.getText());  
 *user* = true;  
 Constants.User.*mol* = null;  
 }  
 else if(asAccountablePerson.isSelected()) {  
 molFind = new AccountablePersonListViewModel(username.getText(), password.getText());  
 *user* = false;  
 Constants.User.*mol* = molFind;  
 }  
 if (logService.isAdminExist(adminFind) || logService.isAccountablePersonExist(molFind))  
 {  
 try {  
 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.setResizable(false);  
 stage.show();  
 } catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }  
 else {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*,"User not found!", ButtonType.*OK*);  
 alert.show();  
 username.setText("");  
 password.setText("");  
 }  
 }  
 }  
}

Метод userLogin() проверява за празни полета и некоректни въведени данни. При натискане на бутона влизане, ако има грешка, то тя се показва на екрана като известие. Ако данните са коректни, което се проверява в LoginService чрез IsAdminExist и isAccountablePersonExist, логин страницата се скрива и на нейно място се зарежда Началната страница. Тя бива ограниена в зависимост какъв потребител е влязал.

Начална страница

Graphical user interface, website

Description automatically generated

От менюто потребителя може да избере какво действие иска да извърши и съответно бива пренасочван към съответната страница.

Основен изглед за потребител - AdminListView.fxml:

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.Menu?>  
<?import javafx.scene.control.MenuBar?>  
<?import javafx.scene.control.MenuItem?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane id="AdminListView" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="367.0" prefWidth="485.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1">  
 <children>  
 <Label layoutX="162.0" layoutY="97.0" prefHeight="34.0" prefWidth="160.0" text="Добре дошли">  
 <font>  
 <Font name="System Bold" size="23.0" />  
 </font>  
 </Label>  
 <Button fx:id="logOutButton" layoutX="172.0" layoutY="288.0" mnemonicParsing="false" onAction="#onLogOutClick" prefHeight="34.0" prefWidth="142.0" text="Изход" />  
 <MenuBar layoutX="-1.0" layoutY="1.0" prefHeight="26.0" prefWidth="485.0">  
 <menus>  
 <Menu mnemonicParsing="false" text="Въведи">  
 <items>  
 <MenuItem fx:id="insertCustomerMenuItem" mnemonicParsing="false" onAction="#customerInsertByUser" text="Клиент" />  
 <MenuItem fx:id="insertProductMenuItem" mnemonicParsing="false" onAction="#productInsert" text="Продукт" />  
 <MenuItem fx:id="insertCardboardMenuItem" mnemonicParsing="false" onAction="#customerCardBoardInsert" text="Картони на клиентите" />  
 <MenuItem fx:id="insertDefectiveProductMenuItem" mnemonicParsing="false" onAction="#defectiveProductInsert" text="Бракуване на продукт" />  
 <MenuItem fx:id="insertAccountablePersonMenuItem" mnemonicParsing="false" onAction="#accountablePersonInsert" text="Добавяне на МОЛ" />  
 </items>  
 </Menu>  
 <Menu mnemonicParsing="false" text="Справки">  
 <items>  
 <MenuItem fx:id="queryCustomerCardboardButton" mnemonicParsing="false" onAction="#OnClickQueryCustomerCardboard" text="Картон на клиента за период" />  
 <MenuItem fx:id="queryAllProductsButton" mnemonicParsing="false" onAction="#OnClickFullListOfAllProducts" text="Всички продукти за период" />  
 <MenuItem fx:id="queryByDMAOrDAButton" mnemonicParsing="false" onAction="#OnClickFullListOfProductsByCategory" text="Продукти по тип за период" />  
 <MenuItem fx:id="queryDefectiveProductsButton" mnemonicParsing="false" onAction="#OnClickQueryToDefectiveProducts" text="Бракувани продукти за период" />  
 <MenuItem fx:id="queryStatusOfProductButton" mnemonicParsing="false" onAction="#OnClickQueryToStatusOfProducts" text="Проверка на продукти по статус за период" />  
 </items>  
 </Menu>  
 </menus>  
 </MenuBar>  
 </children>  
</AnchorPane>

При избиране от менюто Въведи->Клиент се отваря този панел:

Graphical user interface, website

Description automatically generated

Тук се попълват данните за клиента. При избиране назад се връща в Началната страница. При избиране на бутон Въведи се изпълнява функцията onInsertIntoCustomers() която проверява дали всички полета са попълнени и ако няма проблем записва данните в таблицата.

Метод onInsertIntoCustomers():

@FXML  
public void onInsertIntoCustomers() {  
 CustomerListViewModel customer = new CustomerListViewModel(firstNameTextField.getText(),lastNameTextField.getText(),phoneNumberTextField.getText());  
 if(firstNameTextField.getText().equals("") || lastNameTextField.getText().equals("") || phoneNumberTextField.getText().equals("")) {  
 Alert alert=new Alert(Alert.AlertType.*ERROR*,"Please,fill all fields!", ButtonType.*OK*);  
 alert.show();  
 }  
 else {  
 if (customerService.addCustomer(customer)) {  
 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();  
 }  
 }  
}

Подобно на тази страница работят и всички останали Въвеждания от менюто въведи.

Graphical user interface

Description automatically generated

Table

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

Справки:

Картон на клиента за период

Graphical user interface, application

Description automatically generated

Всички продукти за период

Graphical user interface, text

Description automatically generated

Продукти по тип за периодText

Description automatically generated

Бракувани продукти за периодText

Description automatically generated

Проверка на продукти по статус за периодText

Description automatically generated

Реализация на модула за регистриране на събития в системата

За реализацията на модела за регистриране на събития в системата е използван log4j.

Конфигурацията на логъра е поместена в файла log4j.properties, намиращ се в поддиректорията configuration, директория resources. В него са зададени начинът на логване, форматът, както и файлът, в който логовете да се записват – application.log.

Във всички класове от пакета business.services е инициализирана променлива от тип Logger, която служи за записване логове за информация и грешки в отделните методи. Логър също така има и в класа Connection, част от пакета data.access, който служи за инициализация на връзката с базата данни от hibernate.

log4J.properties

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

Пример за инициализация и използване на логър за логване на информация и грешки при изтриване на потребител от тип AdminsEntity в AdminRepository:

Text

Description automatically generated

Тестове

Тестовете са реализирани чрез framework-a Junit. За целта са включени необходимите dependency-та във файла module-info.java. Тестовете са поместени в директория tests, кото за всеки тестван пакен от системата е създаден отделен пакет с тестове.

Част от тестовете:

AdminServiceTest

Graphical user interface

Description automatically generated

CardboardServiceTest

Graphical user interface, text

Description automatically generated

CustomerServiceTest

Graphical user interface, text, application

Description automatically generated

AdminRepositoryTest

Graphical user interface, text

Description automatically generated