Министерство образования Республики Беларусь

Учреждение образования “Белорусский государственный университет информатики и радиоэлектроники”

Факультет компьютерных систем и сетей

кафедра Информатики

Дисциплина: Объектно-ориентированное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

ФИРМА ПО ПРОДАЖЕ ЗАПЧАСТЕЙ

БГУИР КП 1-40 04 01

Выполнил: студент гр. 953505 Ефимчик П.В.

Проверил: Летохо А.С

Минск 2021

**Содержание**

1. Анализ предметной области

1.1Теория……………………………………….……………...3

1.2 Постановка задачи……...….……………………………...9

1. Логическое проектирование

2.1 Логическое проектирование…………………………….10

2.2 Отношения……………………………………………….12

2.3 Диаграмма классов………………………………………14

2.4 Описание классов ………………...…………………..…15

1. Методика работы с полученной программой…………....………………………...……………17
2. Программный код……………….…….……….……….…....22

Заключение……………..………………………………………47

Литература……………...……………………………………....48

**Анализ предметной области**

**1.1 Теория**

В вычислительных системах оконная система (или оконная система) - это программное обеспечение, которое отдельно управляет различными частями экранов дисплея. Это тип графического пользовательского интерфейса (GUI), который реализует парадигму WIMP (окна, значки, меню, указатель) для пользовательского интерфейса.

Каждому запущенному в данный момент приложению назначается обычно изменяемая по размеру и обычно прямоугольная поверхность дисплея для представления пользователю своего графического интерфейса; эти окна могут перекрывать друг друга, в отличие от мозаичного интерфейса, где они не могут перекрываться. Обычно вокруг каждого окна оформляется оконный декор. Программирование как оформления окна, так и доступных виджетов внутри окна, которые представляют собой графические элементы для прямого взаимодействия с пользователем, такие как ползунки, кнопки и т. д., упрощается и упрощается за счет использования наборов инструментов виджетов.

Главный компонент любой оконной системы обычно называется сервером отображения, хотя также используются альтернативные наименования, такие как оконный сервер или композитор. Любое приложение, которое запускается и представляет свой графический интерфейс в окне, является клиентом сервера отображения. Сервер дисплея и его клиенты взаимодействуют друг с другом по протоколу связи, который обычно называется протоколом сервера дисплея, причем сервер дисплея является посредником между клиентами и пользователем. Он получает весь ввод от ядра, который ядро ​​получает от всех подключенных устройств ввода, таких как клавиатура, указывающие устройства или сенсорный экран, и передает его правильному клиенту. Сервер отображения также отвечает за вывод данных клиентов на монитор компьютера. Вывод звука обычно не управляется сервером дисплея, но громкость звука обычно обрабатывается с помощью апплетов графического интерфейса пользователя, и именно сервер дисплея решает, какие приложения находятся наверху. Оконная система позволяет пользователю компьютера работать с несколькими программами одновременно. Каждая программа представляет свой графический интерфейс в собственном окне, которое обычно представляет собой прямоугольную область экрана.

С точки зрения программиста, оконная система реализует графические примитивы. Например: рендеринг шрифтов или рисование линии на экране. Это обеспечивает абстракцию графического оборудования для использования высокоуровневыми элементами графического интерфейса, такими как оконный менеджер.

Сервер отображения или оконный сервер - это программа, основная задача которой - координировать ввод и вывод своих клиентов в и из остальной части операционной системы, оборудования и друг друга. Сервер дисплея взаимодействует со своими клиентами по протоколу сервера дисплея, протоколу связи, который может быть прозрачным для сети или просто совместимым с сетью.

Сервер отображения является ключевым компонентом любого графического пользовательского интерфейса, в частности оконной системы.

Одним из примеров сервера отображения является сервер X.Org, который работает поверх ядра (обычно это Unix-подобное ядро, такое как Linux или BSD). Он получает данные, вводимые пользователем (например, от evdev в Linux), и передает их одному из своих клиентов. Сервер отображения также получает данные от своих клиентов; он обрабатывает данные, выполняет композицию, а в Linux передает данные в один из трех компонентов ядра - DRM, гем или драйвер KMS. Компонент записывает данные в буфер кадра, и содержимое буфера кадра передается на подключенный экран и отображается. X полагается на GLX.

Одной из реализаций концепции сервера отображения является система X Window, в частности ее фактически используемая версия - сервер X.Org и клиентские библиотеки Xlib и XCB. Сервер X.Org является сервером отображения, но в своей текущей реализации он полагается на вторую программу, оконный менеджер композитинга, для выполнения композитинга. Примеры: Mutter или KWin.

Известными примерами серверов отображения, реализующих протокол сервера отображения X11, являются X.Org Server, XFree86, XQuartz и Cygwin / X, а клиентскими библиотеками, реализующими протокол сервера отображения X11, являются Xlib и XCB.

**JavaFX**

JavaFX - это платформа клиентских приложений нового поколения с открытым исходным кодом для настольных, мобильных и встроенных систем, построенная на Java. Это плод совместных усилий многих людей и компаний с целью создания современного, эффективного и полнофункционального инструментария для разработки многофункциональных клиентских приложений.

JavaFX - это программная платформа для создания и доставки настольных приложений, а также многофункциональных веб-приложений, которые могут работать на самых разных устройствах. JavaFX поддерживает настольные компьютеры и веб-браузеры в Microsoft Windows, Linux и macOS, а также мобильные устройства под управлением iOS и Android.

На настольных компьютерах JavaFX поддерживает операционные системы Windows Vista, Windows 7, Windows 8, Windows 10, macOS и Linux. Начиная с JavaFX 1.2, Oracle выпустила бета-версии для OpenSolaris. На мобильных устройствах JavaFX Mobile 1.x может работать в нескольких мобильных операционных системах, включая Symbian OS, Windows Mobile и проприетарные операционные системы реального времени.

JavaFX был предназначен для замены Swing в качестве стандартной библиотеки графического интерфейса для Java SE, но он был исключен из новых стандартных выпусков, в то время как Swing и AWT остались включенными, предположительно из-за того, что рыночная доля JavaFX была «подорвана ростом популярности мобильных устройств в первую очередь» и первые приложения ". С выпуском JDK 11 в 2018 году Oracle сделала JavaFX частью OpenJDK в рамках проекта OpenJFX, чтобы ускорить темпы его разработки. Поддержка Oracle для JavaFX также доступна для Java JDK 8 до марта 2025 года.

JavaFXPorts с открытым исходным кодом работает для iOS (iPhone и iPad) и Android и встраивается (Raspberry Pi), а соответствующее коммерческое программное обеспечение, созданное под названием «Gluon», поддерживает те же мобильные платформы с дополнительными функциями плюс рабочий стол. Это позволяет из единой базы исходного кода создавать приложения для настольных компьютеров, устройств iOS и Android.

Приложения JavaFX создаются с помощью декларативного языка программирования JavaFX Script. Для разработки приложений на языке JavaFX Script необходимо установить JavaFX SDK, который входит в поставку с Java SE 7. Из кода, написанного на языке JavaFX Script, можно обращаться к любым библиотекам Java. Поэтому совместное использование языков Java и JavaFX Script позволяет решать разнообразные задачи, например, логика бизнес-приложения может быть написана на Java, а графический интерфейс пользователя — на JavaFX Script.

JavaFX предоставляет большие возможности по сравнению с рядом других подобных платформ, в частности, по сравнению со Swing. Это и большой набор элементов управления, и возможности по работе с мультимедиа, двухмерной и трехмерной графикой, декларативный способ описания интерфейса с помощью языка разметки FXML, возможность стилизации интерфейса с помощью CSS, интеграция со Swing и многое другое.

История JavaFX фактически началась в первой половине 2000-х годов, когда разработчик по имени Крис Оливер (Chris Oliver), будучи работником компании SeeBeyond, разработал для создания графических интерфейсов новый язык F3 (Froms Follows Functions). Впоследствии в 2005 году SeeBeyond была приобретена компанией Sun Microsystems (которая на тот момент развивала язык Java до покупки компанией Oracle). F3 был переименован в JavaFX, а Крис Оливер продолжил работу над новой платформой уже в рамках компании Sun. И в мае 2007 года Sun Microsystems публично анонсировала новую платформу для создания графических приложений. А 4 декабря 2008 года вышел JavaFX 1.0 SDK.

После приобретения Sun Microsystems компанией Oracle в 2010 году была анонсирована, а в 2011 году вышла в релиз версия JavaFX 2.0. В первой версии JavaFX фактически представлял скриптовый язык. Во второй версии был полностью изменен подход. Скриптовый язык был убран, а платформа была полностью переписана фактически с нуля. Теперь создавать приложения можно было с помощью любого языка, который поддерживала JVM. Были добавлены новые API, интеграция со Swing и много других вещей.

Следующими важными вехами в развитии платформы стали версии JavaFX 8 и особенно JavaFX 9, которая вышла в сентябре 2017 года вместе с Java 9 и привнесла в платформу модульность. И если раньше JavaFX поставлялась вместе с Java SE, то сейчас JavaFX отделена от основной функциональности Java SE и используется как отдельный модуль. Последняя версия фреймворка - JavaFX 15 - вышла в сентябре 2020 года.

На данный момент JavaFX представляет предпочтительный способ для создания графических приложений с помощью языка Java, который пришел на смену AWT и Swing. Также стоит отметить, что для работы с JavaFX вместо Java теоретически можно использовать любой язык программирования, который поддерживается JVM.

**Model-View-Controller** (**MVC**,«Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

* ***Модель*** (*Model*) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.
* ***Представление*** (*View*) отвечает за отображение данных модели пользователю, реагируя на изменения модели.
* ***Контроллер*** (*Controller*) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Основная цель применения этой концепции состоит в отделении бизнес-логики (*модели*) от её визуализации (*представления*, *вида*). За счёт такого разделения повышается возможность повторного использования кода. Наиболее полезно применение данной концепции в тех случаях, когда пользователь должен видеть те же самые данные одновременно в различных контекстах и/или с различных точек зрения. В частности, выполняются следующие задачи:

1. К одной *модели* можно присоединить несколько *видов*, при этом не затрагивая реализацию *модели*. Например, некоторые данные могут быть одновременно представлены в виде электронной таблицы, гистограммы и круговой диаграммы;
2. Не затрагивая реализацию *видов*, можно изменить реакции на действия пользователя (нажатие мышью на кнопке, ввод данных) — для этого достаточно использовать другой *контроллер*;
3. Ряд разработчиков специализируется только в одной из областей: либо разрабатывают графический интерфейс, либо разрабатывают бизнес-логику. Поэтому возможно добиться того, что программисты, занимающиеся разработкой бизнес-логики (*модели*), вообще не будут осведомлены о том, какое *представление* будет использоваться.

**1.2 Постановка задачи**

Курсовой проект является реализацией оконного приложения с работой с базой данных для реализации фирмы по продаже запчастей. Используется фреймворк JavaFX.

Функциональный минимум состоит из:

1. Возможности добавлять нового пользователя(клиента): указание полного имени, электронной почты, пароля.
2. Просмотр запчастей.
3. Поиск запчастей по поставщику, запчастям, названию, а также фильтр указывающий ограничение поиска.
4. Добавление запчасти в корзину и дальнейшая работа с ней.
5. Покупка запчасти.

Также курсовой проект должен иметь работу с базой данных, в которой хранятся данные о клиентах и заказах.

**Логическое проектирование**

**2.1. Логическое проектирование**

В данном курсовом проекте в качестве хранилища данных используется СУБД MySQL Workbench. **MySQL Workbench** — инструмент для визуального проектирования базы данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы данных MySQL. В то время как MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU General Public License, что стало одной из причин выбора данного продукта для выполнения работы с базой данных. Основные возможности MySQL Workbench – позволяет наглядно представить модель базы данных в графическом виде; функциональный механизм установки связей между таблицами; возможность редактирования данных в таблице в визуальном режиме; Reverse Engineering – восстановление структуры таблиц из уже существующей на сервере БД; удобный редактор SQL запросов, позволяющий сразу же отправлять их серверу и получать ответ в виде таблицы.

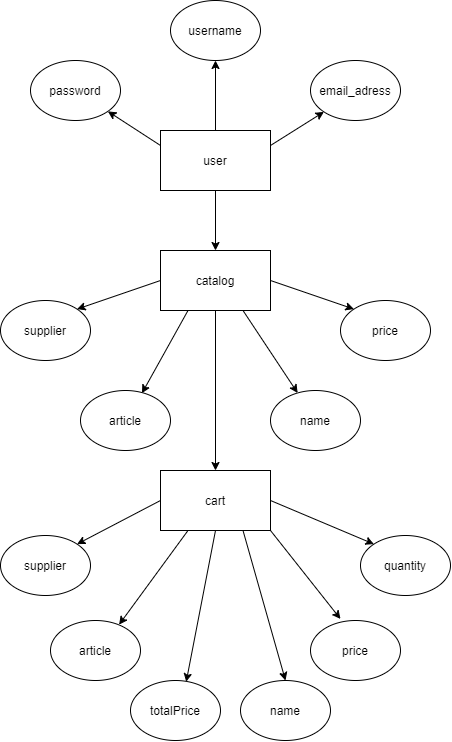
В базе данных данного проекта содержится две таблицы (users, catalog), а также папка с хранимыми процедурами для манипуляций над базой данных в приложении написанном на Java.

Таблица *users* хранит информацию о пользователях и содержит следующие поля:

* Имя клиента
* Электронная почта
* Пароль

Таблица *catalog* хранит данные о заказах и имеет следующие поля:

* Имя поставщика
* Тип запчасти
* Название запчасти
* Цену запчасти

**

*рис. 1 - связь между сущностями users и catalog*

**2.2 Отношения**

На основе всего выше перечисленного, можно составить следующие отношения:

**Таблица 1. Отношения “users”**

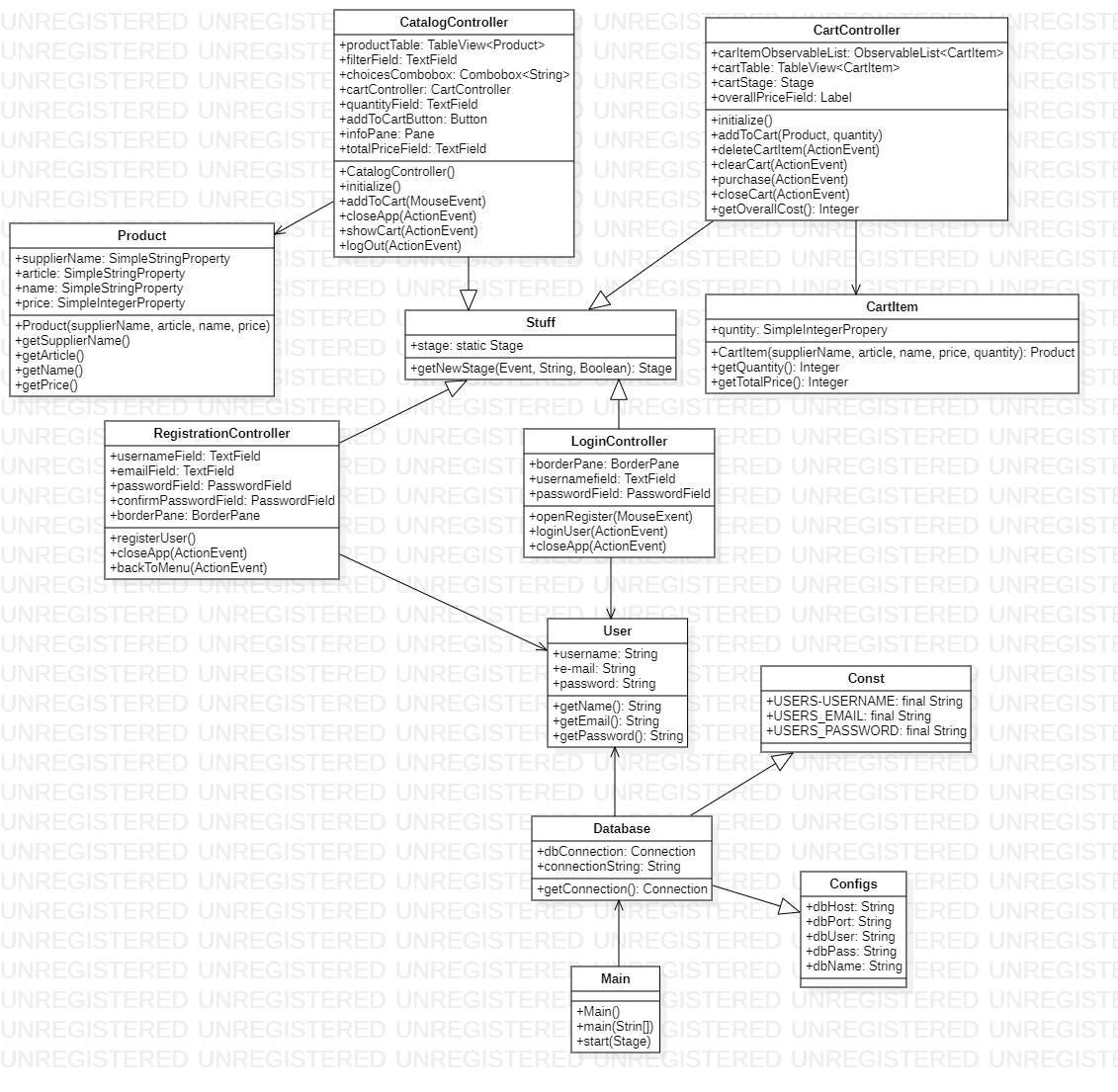
|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип** | **В программе** |
| Username | текстовый | username |
| E-mail | текстовый | email |
| Password | текстовый | password |

**Таблица 2. Отношения “catalog”**

|  |  |  |
| --- | --- | --- |
| **Имя атрибута** | **Тип** | **В программе** |
| Supplier | текстовый | supplier |
| Article | текстовый | article |
| Name | текстовый | name |
| Price | числовой | price |

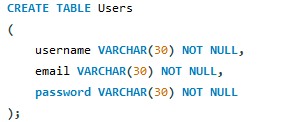
**2.3 Диаграмма классов**

UML-диаграмма классов:



**Заполнение таблиц**

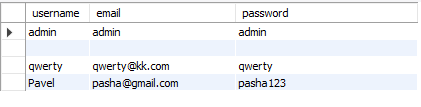
Создание таблицы *users*:



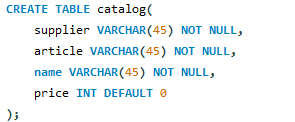
Заполнение таблицы *users*:



Результат заполнения:



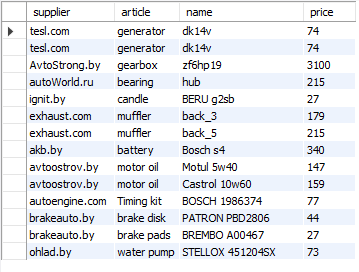
Создание таблицы *catalog*:



Заполнение таблицы *catalog*:



Результат заполнения:



**Описание классов программы**

**пакет controllers:**

класс **CatalogController**:

Контроллер представления со списком запчастей и окном создания заказа

класс **LoginController**:

Контроллер представления окна входа в аккаунт

класс **RegisterController:**

Контроллер представления окна регистрации аккаунта

класс **CartController:**

Контроллер представления окна корзины пользователя

класс **Stuff:**

Контроллер перехода окна на другую сцену

**пакет Models:**

класс **Product:**

Бизнес модель запчасти.

класс **CartItem:**

Бизнес модель корзины.

**пакет database:**

класс **DatabaseConnection**:

Бизнес модель базы данных, через которую осуществляется подключение к ней.

класс **Const**:

содержит константные поля для подключения пользователя.

класс **Configs**:

содержит данные для подключения программы к MySQL.

**пакет** **Sample**:

класс **Controller:**

Контроллер представления главного окна приложения

класс **Main:**

Точка входа в приложение.

**пакет Views:**

представление **Catalog.fxml:**

Представление окна с набором запчастей;

представление **Login.fxml:**

Представление окна входа в аккаунт;

представление **Main.fxml:**

Представление главного окна приложения;

представление **Cart.fxml:**

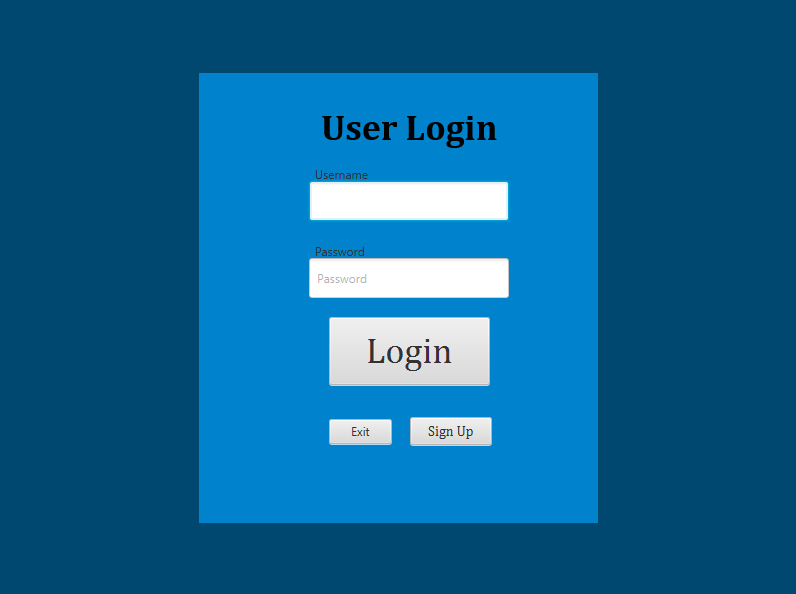
Представление окна с заказами пользователя;

представление **Register.fxml:**

Представление окна регистрации аккаунта;

**Методика работы с полученной программой**

Окно авторизации пользователя:



На нём находятся следующие элементы управления:

1. Поле ввода имени пользователя
2. Поле ввода пароля
3. Кнопка “Exit” для выхода из приложения
4. Кнопка “Sign Up” для перехода в окно регистрации
5. Кнопка “Login” для подтверждения ввода

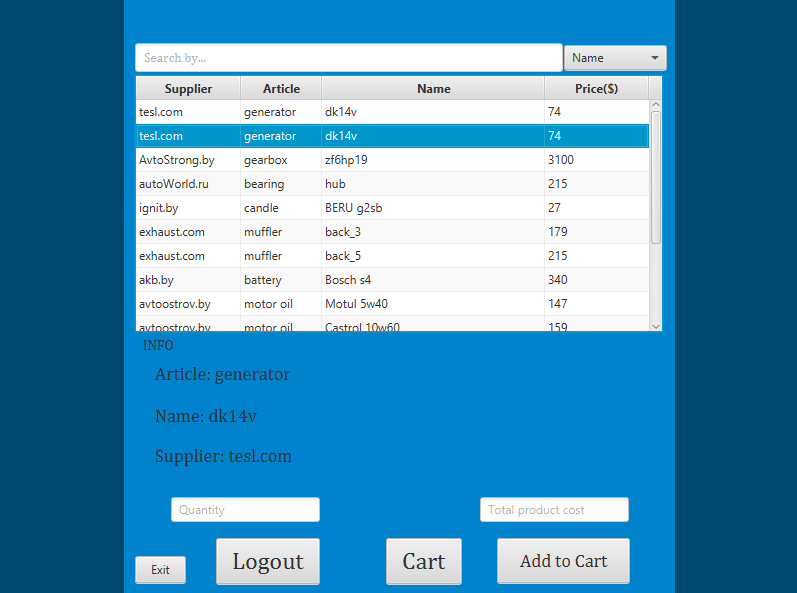
Окно регистрации нового пользователя:



На нём находятся следующие элементы управления:

1. Поле ввода имени пользователя
2. Поле ввода Электронной почты
3. Поле ввода пароля
4. Поле ввода подтверждения пароля
5. Кнопка “Cancel” для возврата к предыдущему окну
6. Кнопка “Exit” для выхода из программы
7. Кнопка “Register” для регистрации введенных пользователем данных

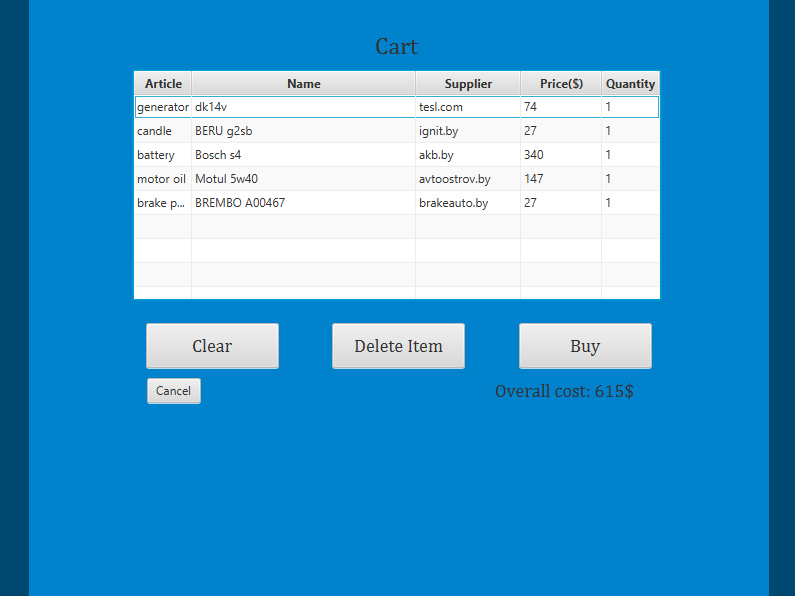
Окно после входа в аккаунт:



На нём находятся следующие элементы управления:

1. “TableView” для отображения доступных запчастей
2. Выпадающее меня для выбора критерия поиска по запчастям
3. При выборе элемента из “TableView” активизируется показ информации о данной запчасти
4. Поле “Quantity” ожидает ввода количества товаров
5. Поле “Total product cost” высчитывает общую стоимость этих товаров
6. Кнопка “Logout” для выхода из аккаунта пользователя
7. Кнопка “Exit” для выхода из программы
8. Кнопка “Add to Cart” для добавления запчасти в корзину
9. Кнопка “Cart” для перехода в окно корзины пользователя

Окно с корзиной текущего пользователя:



На нём находятся следующие элементы управления:

1. “TableView” для отображения добавленных запчастей
2. Кнопка “Cancel” для возврата в предыдущее окно
3. Кнопка “Clear” для удаления всех данных корзины
4. Кнопка “Delete Item” для удаления одного элемента корзины
5. Кнопка “Buy” для покупки добавленных в корзину запчастей
6. “Overall cost” показывает общую стоимость покупки

**Програмный код:**

**CartController.java**

package controllers;

import javafx.collections.FXCollections;

import javafx.collections.ListChangeListener;

import javafx.collections.ObservableList;

import javafx.event.ActionEvent;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.stage.Stage;

import models.CartItem;

import models.Product;

public class CartController {

public static ObservableList<CartItem> cartItemObservableList = FXCollections.observableArrayList();

public TableView<CartItem> cartTable;

public static Stage cartStage;

public Label overallPriceField;

public void initialize(){

cartTable.setItems(cartItemObservableList);

cartTable.setPlaceholder(new Label("Your requested products will appear here"));

cartItemObservableList.addListener(new ListChangeListener<>() {

@Override

public void onChanged(ListChangeListener.Change<? extends CartItem> c) {

overallPriceField.setText("Overall cost: " + String.valueOf(getOverallCost()) + "$");

}

});

TableColumn<CartItem, String> articleColumn = new TableColumn<>("Article");

articleColumn.setCellValueFactory(new PropertyValueFactory<>("article"));

TableColumn<CartItem, String> nameColumn = new TableColumn<>("Name");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<CartItem, String> supplierColumn = new TableColumn<>("Supplier");

supplierColumn.setCellValueFactory(new PropertyValueFactory<>("supplierName"));

TableColumn<CartItem, String> priceColumn = new TableColumn<>("Price($)");

priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

TableColumn<CartItem, Integer> quantityColumn = new TableColumn<>("Quantity");

quantityColumn.setCellValueFactory(new PropertyValueFactory<>("quantity"));

cartTable.setColumnResizePolicy( TableView.CONSTRAINED\_RESIZE\_POLICY );

articleColumn.setMaxWidth( 1f \* Integer.MAX\_VALUE \* 10);

nameColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 45);

supplierColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 20);

priceColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 15);

quantityColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 10);

cartTable.getColumns().add(articleColumn);

cartTable.getColumns().add(nameColumn);

cartTable.getColumns().add(supplierColumn);

cartTable.getColumns().add(priceColumn);

cartTable.getColumns().add(quantityColumn);

overallPriceField.setText("Overall cost: " + String.valueOf(getOverallCost()) + "$");

}

public void addToCart(Product product, int quantity){

int currQ =0;

int index = -1;

for(var cartItem : cartItemObservableList){

if(cartItem.getArticle().equals(product.getArticle()) &&

cartItem.getName().equals(product.getName()) &&

cartItem.getSupplierName().equals(product.getSupplierName()))

{

currQ = cartItem.getQuantity();

index = cartItemObservableList.indexOf(cartItem);

}

}

if(index != -1) {

cartItemObservableList.remove(cartItemObservableList.get(index));

}

cartItemObservableList.add(new CartItem(product.getSupplierName(),product.getArticle(), product.getName(), product.getPrice(), currQ+quantity));

}

public void deleteCartItem(ActionEvent actionEvent) {

var cartItem = cartTable.getSelectionModel().getSelectedItem();

if(cartItem != null) {

cartItemObservableList.remove(cartItem);

}

}

public void clearCart(ActionEvent actionEvent) {

cartItemObservableList.clear();

}

public void purchase(ActionEvent actionEvent) {

if(cartItemObservableList.size() != 0){

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Cart");

alert.setHeaderText(null);

alert.setContentText("Purchased successfully!");

alert.showAndWait();

}

}

public void closeCart(ActionEvent actionEvent) {

cartStage.close();

cartStage = null;

}

public Integer getOverallCost(){

int sum = 0;

for(var item: cartItemObservableList){

sum += item.getTotalPrice();

}

return sum;

}

}

**CatalogController.java**

package controllers;

import database.DatabaseConnection;

import javafx.beans.property.ReadOnlyStringWrapper;

import javafx.beans.value.ChangeListener;

import javafx.beans.value.ObservableValue;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.collections.transformation.FilteredList;

import javafx.collections.transformation.SortedList;

import javafx.event.ActionEvent;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.Pane;

import javafx.stage.Stage;

import models.Product;

import java.sql.\*;

import java.util.List;

import static controllers.CartController.cartStage;

import static controllers.CartController.cartItemObservableList;

public class CatalogController extends Stuff{

public TableView<Product> productsTable;

public Label articleLabel;

public Label nameLabel;

public Label supplierLabel;

public TextField filterField;

public ComboBox<String> choicesComboBox;

public CartController cartController;

public TextField quantityField;

public Button addToCartButton;

public Pane infoPane;

public TextField totalPriceField;

public CatalogController()

{

cartController = new CartController();

}

public void initialize() throws SQLException, ClassNotFoundException {

infoPane.setVisible(false);

addToCartButton.setVisible(false);

quantityField.setVisible(false);

totalPriceField.setVisible(false);

ObservableList<String> choices = FXCollections.observableArrayList("Supplier", "Article", "Name", "Price+", "Price-");

choicesComboBox.setItems(choices);

choicesComboBox.setValue("Name");

DatabaseConnection connectDB = new DatabaseConnection();

Connection connection = connectDB.getConnection();

String sql = "SELECT \* FROM userdata.catalogg";// Where supplier = ? and article = ? and name = ? and price = ?";

CallableStatement callableStatement = connection.prepareCall(sql);

callableStatement.execute();

ResultSet resultSet = callableStatement.getResultSet();

ObservableList<Product> products = FXCollections.observableArrayList();

while (resultSet.next()) {

var product = new Product(

resultSet.getString("supplier"),

resultSet.getString("article"),

resultSet.getString("name"),

resultSet.getInt("price")

);

products.add(product);

}

callableStatement.close();

productsTable.setItems(products);

productsTable.setPlaceholder(new Label("It seems, there are no products here!"));

TableColumn<Product, String> supplierColumn = new TableColumn<>("Supplier");

supplierColumn.setCellValueFactory(new PropertyValueFactory<>("supplierName"));

TableColumn<Product, String> articleColumn = new TableColumn<>("Article");

articleColumn.setCellValueFactory(new PropertyValueFactory<>("article"));

TableColumn<Product, String> nameColumn = new TableColumn<>("Name");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<Product, String> priceColumn = new TableColumn<>("Price($)");

priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

productsTable.setColumnResizePolicy(TableView.CONSTRAINED\_RESIZE\_POLICY);

supplierColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 20);

articleColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 15);

nameColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 45);

priceColumn.setMaxWidth(1f \* Integer.MAX\_VALUE \* 20);

productsTable.getColumns().add(supplierColumn);

productsTable.getColumns().add(articleColumn);

productsTable.getColumns().add(nameColumn);

productsTable.getColumns().add(priceColumn);

productsTable.getSelectionModel().selectedItemProperty().addListener(new ChangeListener() {

@Override

public void changed(ObservableValue observableValue, Object oldValue, Object newValue) {

if(productsTable.getSelectionModel().getSelectedItem() != null)

{

var sel = productsTable.getSelectionModel().getSelectedItem();

System.out.println("Selected Value " + sel.getName() + " " + sel.getSupplierName());

articleLabel.setText("Article: "+sel.getArticle());

nameLabel.setText("Name: "+sel.getName());

supplierLabel.setText("Supplier: "+sel.getSupplierName());

infoPane.setVisible(true);

addToCartButton.setVisible(true);

quantityField.setVisible(true);

totalPriceField.setVisible(true);

var selected = productsTable.getSelectionModel().getSelectedItem();

if(selected != null){

int cost = Integer.parseInt("0"+quantityField.getText()) \* selected.getPrice();

if(cost != 0){

totalPriceField.setText(String.valueOf(cost));

}else{

totalPriceField.clear();

}

}

}

else{

infoPane.setVisible(false);

addToCartButton.setVisible(false);

quantityField.setVisible(false);

quantityField.clear();

totalPriceField.setVisible(false);

}

}

}); choicesComboBox.getSelectionModel().selectedItemProperty().addListener(

(options, oldValue, newValue) -> {

filterField.setText("");

}

);

FilteredList<Product> filteredProducts = new FilteredList<>(products, b->true);

filterField.textProperty().addListener(((observableValue, oldValue, newValue) -> {

filteredProducts.setPredicate(product -> {

if (newValue == null || newValue.isEmpty()) {

return true;

}

String lowerCaseFilter = newValue.toLowerCase();

String choice = (String) choicesComboBox.getValue();

switch (choice) {

case "Supplier":

return product.getSupplierName().toLowerCase().contains(lowerCaseFilter);

case "Article":

return product.getArticle().toLowerCase().contains(lowerCaseFilter);

case "Name":

return product.getName().toLowerCase().contains(lowerCaseFilter);

case "Price+":

try {

return product.getPrice() >= Integer.parseInt(lowerCaseFilter);

} catch (Exception e) {

return false;

}

case "Price-":

try {

return product.getPrice() <= Integer.parseInt(lowerCaseFilter);

} catch (Exception e) {

return false;

}

default:

return false;

}

});

}));

SortedList<Product> sortedProducts = new SortedList<>(filteredProducts);

sortedProducts.comparatorProperty().bind(productsTable.comparatorProperty());

productsTable.setItems(sortedProducts);

quantityField.textProperty().addListener((observable, oldValue, newValue) -> {

if (!newValue.matches("^[1-9]\\d\*$|^[0]$")) {

System.out.println("not match1");

quantityField.setText(newValue.replaceAll("[^\\d]", ""));

if (newValue.matches("[0]\\d+"))

{

quantityField.setText(newValue.replaceAll("^[0]", ""));

}

}

else{

if (Integer.parseInt(newValue) > 1000){

quantityField.setText("1000");

}

}

var selected = productsTable.getSelectionModel().getSelectedItem();

if (selected != null){

int cost = Integer.parseInt("0"+quantityField.getText()) \* selected.getPrice();

if(cost != 0){

totalPriceField.setText(String.valueOf(cost));

}else{

totalPriceField.clear();

}

}

});

}

public void addToCart(MouseEvent mouseEvent) {

var selectedProduct = productsTable.getSelectionModel().getSelectedItem();

if (selectedProduct != null){

int quantity = Integer.parseInt("0"+quantityField.getText());

if (quantity != 0){

cartController.addToCart(selectedProduct, quantity);

}

}

}

public void closeApp(ActionEvent actionEvent) {

System.exit(0);

}

public void showCart(ActionEvent actionEvent) {

if(cartStage == null) {

cartStage = getNewStage(actionEvent, "../views/Cart.fxml", false);

if(cartStage != null) {

cartStage.show();

}

}

}

public void logOut(ActionEvent actionEvent) {

if(cartStage!= null){

cartStage.close();

cartStage = null;

}

cartItemObservableList.clear();

Stage login = getNewStage(actionEvent, "../views/Login.fxml", true);

if(login != null) {

login.show();

}

}

}

**LoginController.java**

package controllers;

import database.DatabaseConnection;

import javafx.event.ActionEvent;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.control.Alert;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.BorderPane;

import javafx.stage.Stage;

import java.io.IOException;

import java.sql.\*;

public class LoginController extends Stuff {

public BorderPane borderPane;

public TextField usernameField;

public PasswordField passwordField;

public void openRegister(MouseEvent mouseEvent) throws IOException {

Parent root = FXMLLoader.load(getClass().getResource("../views/Register.fxml"));

borderPane.getChildren().removeAll();

borderPane.getChildren().setAll(root);

}

public void loginUser(ActionEvent actionEvent) throws SQLException, ClassNotFoundException {

var username = usernameField.getText();

var password = passwordField.getText();

if(!username.equals("") && !password.equals(""))

{

DatabaseConnection connectDB = new DatabaseConnection();

Connection connection = connectDB.getConnection();

String sql = "SELECT \* FROM users Where username = ? and password = ?";

CallableStatement callableStatement = connection.prepareCall(sql);

callableStatement.setString(1, username);

callableStatement.setString(2, password);

callableStatement.execute();

ResultSet res = callableStatement.getResultSet();

if (res.next()) {

Stage mainStage = getNewStage(actionEvent, "../views/Catalog.fxml", true);

if(mainStage != null){

mainStage.show();

}

} else {

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(username);

alert.setHeaderText(null);

alert.setContentText("Wrong username/password combination");

alert.showAndWait();

}

callableStatement.close();

}

}

public void closeApp(ActionEvent actionEvent) {

System.exit(0);

}

}

**RegisterController.java**

package controllers;

import database.DatabaseConnection;

import database.Const;

import javafx.event.ActionEvent;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.control.Alert;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextField;

import javafx.scene.input.MouseEvent;

import javafx.scene.layout.BorderPane;

import java.io.IOException;

import java.sql.\*;

public class RegisterController extends Stuff{

public TextField usernameField;

public TextField emailField;

public PasswordField passwordField;

public PasswordField confirmPasswordField;

public BorderPane borderPane;

public void registerUser(MouseEvent mouseEvent) {

var username = usernameField.getText();

var email = emailField.getText();

var password1 = passwordField.getText();

var password2 = confirmPasswordField.getText();

if(!username.equals("") && !email.equals("") && !password1.equals("") && !password1.equals(""))

{

String textError = "";

if(!email.matches("^([a-z0-9\_-]+\\.)\*[a-z0-9\_-]+@[a-z0-9\_-]+(\\.[a-z0-9\_-]+)\*\\.[a-z]{2,6}$")){

textError = "Invalid E-mail address";

}

if(!password1.equals(password2)){

textError = "Passwords aren't equal";

}

if(!textError.equals("")){

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(username);

alert.setHeaderText(null);

alert.setContentText(textError);

alert.showAndWait();

return;

}

try{

DatabaseConnection connectDB = new DatabaseConnection();

Connection connection = connectDB.getConnection();

String insert = "INSERT Users(" +

Const.USERS\_USERNAME + ", " +

Const.USERS\_EMAIL + ", " +

Const.USERS\_PASSWORD + ")" +

"VALUES(?, ?, ?)";

try{

PreparedStatement prSt = connection.prepareStatement(insert);

prSt.setString(1, username);

prSt.setString(2, email);

prSt.setString(3, password1);

prSt.executeUpdate();

connection.close();

}catch (Exception e){

e.printStackTrace();

}

usernameField.clear();;

emailField.clear();

passwordField.clear();

confirmPasswordField.clear();

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle(username);

alert.setHeaderText(null);

alert.setContentText("Added " + username + " to database");

alert.showAndWait();

} catch (Exception e){

e.printStackTrace();

}

}

}

public void closeApp(ActionEvent actionEvent) {

System.exit(0);

}

public void backToMenu(ActionEvent actionEvent) throws IOException {

Parent root = FXMLLoader.load(getClass().getResource("../views/Login.fxml"));

borderPane.getChildren().removeAll();

borderPane.getChildren().setAll(root);

}

}

**Stuff.java**

package controllers;

import javafx.event.Event;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

import java.io.IOException;

public class Stuff {

private static double xOffset = 0;

private static double yOffset = 0;

public static Stage getNewStage(Event event, String pathToFxml, Boolean hideCurrent) {

try{

Parent root = FXMLLoader.load(Stuff.class.getResource(pathToFxml));

Stage stage = new Stage();

stage.setTitle("New Stage");

stage.initStyle(StageStyle.UNDECORATED);

stage.setScene(new Scene(root));

root.setOnMousePressed(event1 -> {

xOffset = event1.getSceneX();

yOffset = event1.getSceneY();

});

root.setOnMouseDragged(event2 -> {

stage.setX(event2.getScreenX() - xOffset);

stage.setY(event2.getScreenY() - yOffset);

});

if(hideCurrent) {

((Node)(event.getSource())).getScene().getWindow().hide();

}

return stage;

}

catch (IOException e){

e.printStackTrace();

}

return null;

}

}

**DatabaseConnection.java**

package database;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection extends Configs{

Connection dbConnection = null;

public Connection getConnection() throws ClassNotFoundException, SQLException {

String connectionString = "jdbc:mysql://" +

dbHost + ":" +

dbPort + "/" +

dbName + "?" + "autoReconnect=true&useSSL=false&useUnicode=true&characterEncoding=UTF-8&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

Class.forName("com.mysql.jdbc.Driver");

try {

dbConnection = DriverManager.getConnection(connectionString, dbUser, dbPass);

if(dbConnection!=null){

System.out.println("Connection success!");

}

else{

System.out.println("Connection fail!");

}

}catch(Exception e){

e.printStackTrace();

}

return dbConnection;

}

}

**Const.java**

package database;

public class Const {

public static final String USERS\_USERNAME = "username";

public static final String USERS\_EMAIL = "email";

public static final String USERS\_PASSWORD = "password";

}

**Configs.java**

package database;

public class Configs {

protected String dbHost = "127.0.0.1";

protected String dbPort = "3306";

protected String dbUser = "root";

protected String dbPass = "2013\_Pasha\_14";

protected String dbName = "userdata";

}

**Product.java**

package models;

import javafx.beans.property.\*;

public class Product {

protected SimpleStringProperty supplierName;

protected SimpleStringProperty article;

protected SimpleStringProperty name;

protected SimpleIntegerProperty price;

public Product(String supplierName, String article, String name, int price){

this.supplierName = new SimpleStringProperty(supplierName);

this.article = new SimpleStringProperty(article);

this.name = new SimpleStringProperty(name);

this.price = new SimpleIntegerProperty(price);

}

public String getSupplierName(){

return supplierName.get();

}

public void setSupplierName(String value){

supplierName.set(value);

}

public String getArticle(){

return article.get();

}

public void setArticle(String value){

article.set(value);

}

public String getName(){

return name.get();

}

public void setName(String value){

name.set(value);

}

public Integer getPrice(){

return price.get();

}

public void setPrice(Integer value){

price.set(value);

}

}

**CartItem.java**

package models;

import javafx.beans.property.SimpleIntegerProperty;

public class CartItem extends Product {

private SimpleIntegerProperty quantity;

public CartItem(String supplierName, String article, String name, int price, int quantity) {

super(supplierName, article, name, price);

this.quantity = new SimpleIntegerProperty(quantity);

}

public Integer getQuantity(){

return quantity.get();

}

public void setQuantity(Integer value){

quantity.set(value);

}

public Integer getTotalPrice(){

return getPrice() \* getQuantity();

}

}

**Cart.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
<?import javafx.scene.control.\*?>  
<?import javafx.scene.layout.\*?>  
<?import javafx.scene.text.\*?  
<AnchorPane fx:id="content\_area" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="740.0" style="-fx-background-color: #0083CC;" xmlns="http://javafx.com/javafx/11.0.2" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.CartController">  
 <children>  
 <TableView fx:id="cartTable" layoutX="104.0" layoutY="71.0" prefHeight="230.0" prefWidth="528.0" />  
 <Button layoutX="117.0" layoutY="324.0" mnemonicParsing="false" onAction="#clearCart" prefHeight="45.0" prefWidth="133.0" text="Clear">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Button>  
 <Button fx:id="deleteButton" layoutX="303.0" layoutY="324.0" mnemonicParsing="false" onAction="#deleteCartItem" prefHeight="45.0" prefWidth="133.0" text="Delete Item">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Button>  
 <Button layoutX="490.0" layoutY="324.0" mnemonicParsing="false" onAction="#purchase" prefHeight="45.0" prefWidth="133.0" text="Buy">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Button>  
 <Button fx:id="btn\_close" layoutX="118.0" layoutY="379.0" mnemonicParsing="false" onAction="#closeCart" prefHeight="25.0" prefWidth="54.0" text="Cancel" />  
 <Label fx:id="articleLabel" layoutX="346.0" layoutY="32.0" maxWidth="487.0" prefHeight="30.0" text="Cart">  
 <font>  
 <Font name="Cambria" size="24.0" />  
 </font>  
 </Label>  
 <Label fx:id="overallPriceField" layoutX="466.0" layoutY="377.0" maxWidth="487.0" prefHeight="30.0" prefWidth="157.0">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

**Login.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
<?import javafx.scene.\*?>  
<?import javafx.scene.control.\*?>  
<?import javafx.scene.layout.\*?>  
<?import javafx.scene.text.\*?>  
<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0" style="-fx-background-color: #004870;" xmlns="http://javafx.com/javafx/11.0.2" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.LoginController">  
 <center>  
 <AnchorPane fx:id="content\_area" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="450.0" prefWidth="399.0" style="-fx-background-color: #0083CC;" BorderPane.alignment="CENTER">  
 <children>  
 <TextField fx:id="usernameField" layoutX="110.0" layoutY="108.0" prefHeight="40.0" prefWidth="200.0" promptText="Username">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </TextField>  
 <PasswordField fx:id="passwordField" layoutX="110.0" layoutY="185.0" prefHeight="40.0" prefWidth="200.0" promptText="Password">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </PasswordField>  
 <Text layoutX="122.0" layoutY="67.0" strokeType="OUTSIDE" strokeWidth="0.0" text="User Login">  
 <font>  
 <Font name="Cambria Bold" size="36.0" />  
 </font>  
 </Text>  
 <Button layoutX="130.0" layoutY="244.0" mnemonicParsing="false" onAction="#loginUser" prefHeight="68.0" prefWidth="161.0" text="Login">  
 <font>  
 <Font name="Cambria" size="36.0" />  
 </font>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </Button>  
 <Button layoutX="211.0" layoutY="344.0" mnemonicParsing="false" onMouseClicked="#openRegister" prefHeight="28.0" prefWidth="82.0" text="Sign Up">  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 <font>  
 <Font name="Cambria" size="14.0" />  
 </font>  
 </Button>  
 <Button fx:id="btn\_close" layoutX="130.0" layoutY="346.0" mnemonicParsing="false" onAction="#closeApp" prefHeight="25.0" prefWidth="63.0" text="Exit" />  
 <Label layoutX="116.0" layoutY="91.0" prefHeight="20.0" prefWidth="100.0" text="Username" />  
 <Label layoutX="116.0" layoutY="168.0" prefHeight="20.0" prefWidth="100.0" text="Password" />  
 </children>  
 </AnchorPane>  
 </center>  
</BorderPane>

**sample.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
<?import javafx.scene.control.Button?>  
<?import javafx.scene.control.Label?>  
<?import javafx.scene.control.TextField?>  
<?import javafx.scene.layout.AnchorPane?>  
<?import javafx.scene.text.Font?>  
  
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="400.0" prefWidth="600.0" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1" fx:controller="sample.Controller">  
 <children>  
 <Button fx:id="connectButton" layoutX="250.0" layoutY="218.0" mnemonicParsing="false" onAction="#connectButton" text="Insert and show" />  
 <TextField fx:id="showUsernameLabel" layoutX="225.0" layoutY="107.0" />  
 <Label fx:id="textLabel" layoutX="265.0" layoutY="141.0" text="Label">  
 <font>  
 <Font size="30.0" />  
 </font>  
 </Label>  
 </children>  
</AnchorPane>

**MyCars.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
<?import javafx.scene.control.\*?>  
<?import javafx.scene.layout.\*?>  
<?import javafx.scene.text.\*?>  
<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0" style="-fx-background-color: #004870;" xmlns="http://javafx.com/javafx/11.0.2" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.CatalogController">  
 <center>  
 <AnchorPane fx:id="content\_area" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="551.0" style="-fx-background-color: #0083CC;" BorderPane.alignment="CENTER">  
 <children>  
 <TextField fx:id="filterField" layoutX="11.0" layoutY="46.0" prefHeight="29.0" prefWidth="428.0" promptText="Search by...">  
 <font>  
 <Font name="Cambria" size="13.0" />  
 </font></TextField>  
 <ComboBox fx:id="choicesComboBox" layoutX="440.0" layoutY="48.0" prefHeight="25.0" prefWidth="103.0" promptText="Search" />  
 <TableView fx:id="productsTable" layoutX="11.0" layoutY="78.0" prefHeight="257.0" prefWidth="528.0" />  
 <TextField fx:id="quantityField" layoutX="47.0" layoutY="500.0" promptText="Quantity" />  
 <Button fx:id="addToCartButton" layoutX="373.0" layoutY="541.0" mnemonicParsing="false" onMouseClicked="#addToCart" prefHeight="45.0" prefWidth="133.0" text="Add to Cart">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font></Button>  
 <Pane fx:id="infoPane" layoutX="11.0" layoutY="338.0" prefHeight="150.0" prefWidth="528.0">  
 <children>  
 <Label fx:id="articleLabel" layoutX="20.0" layoutY="24.0" maxWidth="487.0" prefHeight="30.0" text="Article: ">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Label>  
 <Label fx:id="nameLabel" layoutX="20.0" layoutY="70.0" text="Name: ">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Label>  
 <Label fx:id="supplierLabel" layoutX="20.0" layoutY="110.0" text="Supplier: ">  
 <font>  
 <Font name="Cambria" size="18.0" />  
 </font>  
 </Label>  
 <Pane prefHeight="20.0" prefWidth="49.0">  
 <children>  
 <Label layoutX="8.0" layoutY="1.0" text="INFO">  
 <font>  
 <Font name="Cambria" size="14.0" />  
 </font>  
 </Label>  
 </children>  
 </Pane>  
 </children>  
 </Pane>  
 <TextField fx:id="totalPriceField" editable="false" layoutX="356.0" layoutY="500.0" promptText="Total product cost" />  
 <Button layoutX="11.0" layoutY="559.0" mnemonicParsing="false" onAction="#closeApp" prefHeight="27.0" prefWidth="51.0" text="Exit" />  
 <Button layoutX="92.0" layoutY="541.0" mnemonicParsing="false" onAction="#logOut" text="Logout" textAlignment="CENTER">  
 <font>  
 <Font name="Cambria" size="24.0" />  
 </font>  
 </Button>  
 <Button layoutX="262.0" layoutY="541.0" mnemonicParsing="false" onAction="#showCart" prefHeight="33.0" prefWidth="76.0" text="Cart" textAlignment="CENTER">  
 <font>  
 <Font name="Cambria" size="24.0" />  
 </font>  
 </Button>  
 </children>  
 </AnchorPane>  
 </center>  
</BorderPane>

**Register.fxml**

<?xml version="1.0" encoding="UTF-8"?>  
<?import javafx.scene.\*?>  
<?import javafx.scene.control.\*?>  
<?import javafx.scene.layout.\*?>  
<?import javafx.scene.text.\*?>  
<BorderPane fx:id="borderPane" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="600.0" prefWidth="800.0" style="-fx-background-color: #004870;" xmlns="http://javafx.com/javafx/11.0.2" xmlns:fx="http://javafx.com/fxml/1" fx:controller="controllers.RegisterController">  
 <center>  
 <AnchorPane fx:id="content\_area" maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="450.0" prefWidth="390.0" style="-fx-background-color: #0083CC;" xmlns="http://javafx.com/javafx/16" xmlns:fx="http://javafx.com/fxml/1">  
 <children>  
 <TextField fx:id="usernameField" layoutX="97.0" layoutY="91.0" prefHeight="40.0" prefWidth="200.0" promptText="Username">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </TextField>  
 <TextField fx:id="emailField" layoutX="96.0" layoutY="156.0" prefHeight="40.0" prefWidth="200.0" promptText="Email">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </TextField>  
 <PasswordField fx:id="passwordField" layoutX="96.0" layoutY="218.0" prefHeight="40.0" prefWidth="200.0" promptText="Password">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </PasswordField>  
 <PasswordField fx:id="confirmPasswordField" layoutX="97.0" layoutY="276.0" prefHeight="40.0" prefWidth="200.0" promptText="Confirm Password">  
 <cursor>  
 <Cursor fx:constant="TEXT" />  
 </cursor>  
 </PasswordField>  
 <Button layoutX="121.0" layoutY="346.0" mnemonicParsing="false" onMouseClicked="#registerUser" prefHeight="54.0" prefWidth="149.0" text="Register">  
 <font>  
 <Font name="Cambria" size="24.0" />  
 </font>  
 <cursor>  
 <Cursor fx:constant="HAND" />  
 </cursor>  
 </Button>  
 <Text layoutX="46.0" layoutY="48.0" strokeType="OUTSIDE" strokeWidth="0.0" text="Registration Form">  
 <font>  
 <Font name="Cambria Bold" size="36.0" />  
 </font>  
 </Text>  
 <Button layoutX="121.0" layoutY="418.0" mnemonicParsing="false" onAction="#backToMenu" prefHeight="25.0" prefWidth="66.0" text="Cancel" />  
 <Button fx:id="btn\_close" layoutX="208.0" layoutY="418.0" mnemonicParsing="false" onAction="#closeApp" prefHeight="25.0" prefWidth="62.0" text="Exit" />  
 <Label layoutX="107.0" layoutY="74.0" prefHeight="20.0" prefWidth="100.0" text="Username" />  
 <Label layoutX="107.0" layoutY="139.0" prefHeight="20.0" prefWidth="100.0" text="E-mail" />  
 <Label layoutX="107.0" layoutY="201.0" prefHeight="20.0" prefWidth="100.0" text="Password" />  
 <Label layoutX="107.0" layoutY="258.0" prefHeight="20.0" prefWidth="100.0" text="Confirm password" />  
 </children>  
 </AnchorPane>  
 </center>  
</BorderPane>

**Main.java**

package sample;

import javafx.application.Application;

import javafx.event.EventHandler;

import javafx.fxml.FXMLLoader;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.scene.input.MouseEvent;

import javafx.stage.Stage;

import javafx.stage.StageStyle;

public class Main extends Application {

public static Stage stage = null;

private double xOffset = 0;

private double yOffset = 0;

@Override

public void start(Stage primaryStage) throws Exception{

Parent root = FXMLLoader.load(getClass().getResource("../views/Login.fxml"));

primaryStage.setTitle("Hello World");

primaryStage.initStyle(StageStyle.UNDECORATED);

primaryStage.setScene(new Scene(root, 800 , 600));

stage = primaryStage;

primaryStage.show();

root.setOnMousePressed(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

xOffset = event.getSceneX();

yOffset = event.getSceneY();

}

});

root.setOnMouseDragged(new EventHandler<MouseEvent>() {

@Override

public void handle(MouseEvent event) {

stage.setX(event.getScreenX() - xOffset);

stage.setY(event.getScreenY() - yOffset);

}

});

}

public static void main(String[] args) {

launch(args);

}

}

**Заключение**

В курсовом проекте были реализованы необходимые функции взаимодействия с ПО. Архитектура проекта соответствует принципам MVC - Model View Controller, что упрощает и структурирует написание программного кода. Использовались базовые концепции ООП, например, такие как наследование, абстракция и т.д. Был применен фреймворк JavaFX для создания графического интерфейса. В качестве системы управления базы данных выступала MySQL и приложение с графическим интерфейсом для отображения базы данных MySQL WorkBench.

**Литература**

1. Java. Методы программирования : уч.-мет. пособие / И. Н. Блинов, В. С. Романчик. — Минск : издательство «Четыре четверти», 2013 — 896 с.
2. Ресурсы по JavaFX на сайте Oracle (англ.) [ссылка] - https://www.oracle.com/java/technologies/javase/javafx-overview.html