Федеральное государственное бюджетное образовательное

учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ**

**ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Департамент анализа данных и машинного обучения**

**2**

**Пояснительная записка к курсовой работе по дисциплине**

**«Современные технологии программирования»**

**на тему:**

**«Разработка информационно-справочной системы**

**магазина цифровой техники»**

Выполнил:

Студентка группы ПИ19-2

Петров Олег Игоревич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

Научный руководитель:

доцент, к.т.н.

Кублик Евгений Ильич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

2021

**Оглавление**

**Введение** ............................................................................................................... 3

**1. Постановка задачи** ......................................................................................... 4

**2. Описание предметной области** .................................................................... 5

**3. Актуальность автоматизации** ..................................................................... 5

**4.Описание программы** .................................................................................... 6

4.1. Алгоритмические решения .................................................................. 6

4.2 Описание интерфейса программы ....................................................... 8

4.3 Состав приложения ............................................................................... 15

**5.Назначение и состав программы** ................................................................ 16

5.1 Описание классов Сервера ................................................................... 16

5.2 Описание классов Клиента ................................................................... 20

**Заключение** ......................................................................................................... 25

**Список литературы** ........................................................................................... 26

**Приложение** ........................................................................................................ 27

2

**Введение**

Сейчас невероятно быстро и стремительно развиваются технологи, которые даже представить сложно. Технологиями являются не только инновационное научное оборудование или новые ракеты из программы Space X, но и цифровая техника, которую может приобрести практически каждый. Каждый год выпускается новые смартфоны, ноутбуки, smart-техника для умного дома и прочее. В связи с этим, при открытии собственного магазина отличным выбором будет продажа цифровой техники. Я считаю, тему «Информационно-справочная система магазина цифровой техники» **актуальной**, именно поэтому выбрал её для курсовой работы.

Для работы магазина необходимо решить много различных задач, одной из них является внутренняя система для сотрудников. Она необходима для хранения и обработки данных, например, о товарах или заказах и т. п. Этим я и занимался в процессе написания данной работы.

**Цель** данной работы:

Разработка информационно-справочной системы магазина цифровой техники.

**Задачи**, стоящие передо мной, для достижения цели:

• Проанализировать предстоящую работу и осуществить постановку задач.

• Разработать информационно-справочную систему

3

**1. Постановка задачи**

По выбранной теме мне необходимо разработать и написать клиент-серверное приложение, информационно-справочную систему магазина цифровой техники, которое предоставит сотрудникам следующий функционал:

• Просматривание данных, полученные из БД и занесенные в таблицы:

1. Категории товаров
2. Компании, производящие их
3. Модели товаров
4. Продукты
5. Заказы

• Возможность управлять записями в каждой таблице:

1. Добавлять
2. Редактировать
3. Удалять

• Сортировка и фильтрация записей

• Просмотр статистики по продуктам

4

**2. Описание предметной области**

Каждому магазину необходимо иметь быстрый доступ к данным о своих товарах. Наиболее удобными для этих целей являются информационно-справочные системы, которые предоставляют пользователю не только информацию о количестве единиц на складе, но и развёрнутое описание самого товара (его цвет, размер, комплектацию, информацию о модели и производителе).

Данная программа также может быть использована для создания, обработки и отслеживания онлайн-заказов. Сотрудники должны иметь возможность просмотреть и, в случае надобности, изменить комплектацию заказа и дату его выдачи.

Часто используемой функцией в подобных программах является сбор статистических данных о вашем бизнесе. К примеру, для нашего интернет-магазина будет актуальна статистика спроса на различные товары.

**3. Актуальность автоматизации**

Магазину важно иметь информационно справочную систему в том числе из-за удобства хранения большого количества данных. Функционал, предоставляемый приложением, позволяет сортировать и фильтровать записи, что ускоряет и упрощает работу с ними.

5

**4. Описание программы**

**4.1. Алгоритмические решения**

В моей информационно-справочной системе сущности имею между собой связи «один ко многим» (Рис. №1). Для каждой сущности был создан свой класс, со своими полями и конструкторами.

Изображение выглядит как текст

Автоматически созданное описание

Рис. №1 (Реляционная модель БД)

Для хранения списков Категорий и Компаний были созданы классы «Category» и «Company».

Оба класса имеют одни и те же поля:

* id (тип Long)
* name (имя Категории/Компании, тип String)

6

Для сущности Модель создан класс Model.

Атрибуты класса:

* id (тип Long)
* name (имя Модели, тип String)
* company\_id (id Компании, тип Long)
* category\_id (id Категории, тип Long)

Для хранения информации о Продукте реализован класс Product, который имеет поля:

* id (тип Long)
* price (цена продукта, тип Integer)
* color (цвет продукта, тип String)
* count (количество продукта, тип Integer)
* model (модель продукта, тип – Объект класса Model)

Последняя сущность Заказ имеет класс Order, для которой были созданы поля:

* id (тип Long)
* amount (количество заказанного продукта, тип Integer)
* date\_of\_create (дата создания заказа, тип LocalDate)
* date\_of\_ready (дата готовности к выдачи, тип LocalDate)
* product\_id (id Продукта, тип Long)

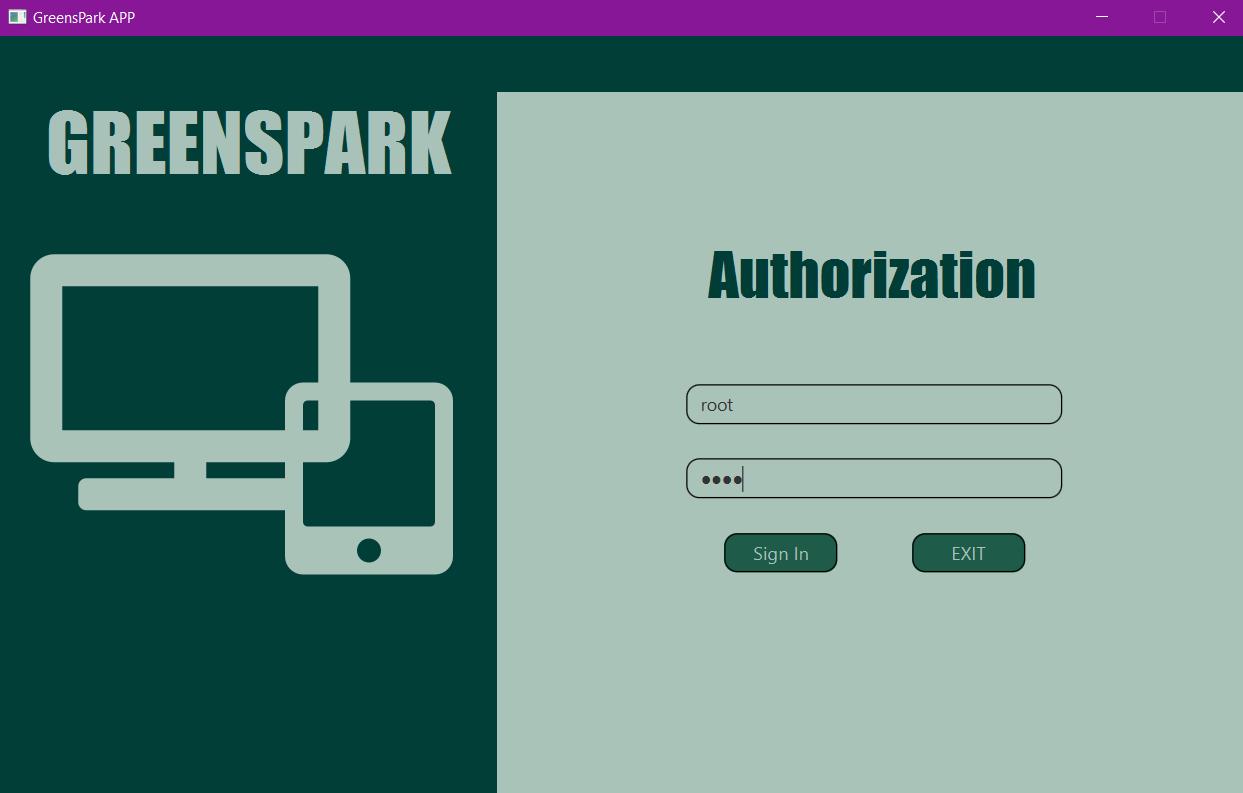
7

**4.2.** **Описание интерфейса программы**

Визуальная часть моего приложения в сумме состоит из 16 страниц. Такое количество необходимо для распределения и разбиения информации. В таком случае интерфейс становится интуитивно понятным и «дружелюбным».

При запуске первой страницей будет авторизация. Она содержит следующие элементы (Рис. № 2):

* 2 Label (“GREENSPARK”, “Authorization”)
* 2 Button (“Sign in”, “EXIT”)
* TextField (login)
* PasswordField (password)
* 2 SVG Path



**PasswordField**

**TextField**

**Button**

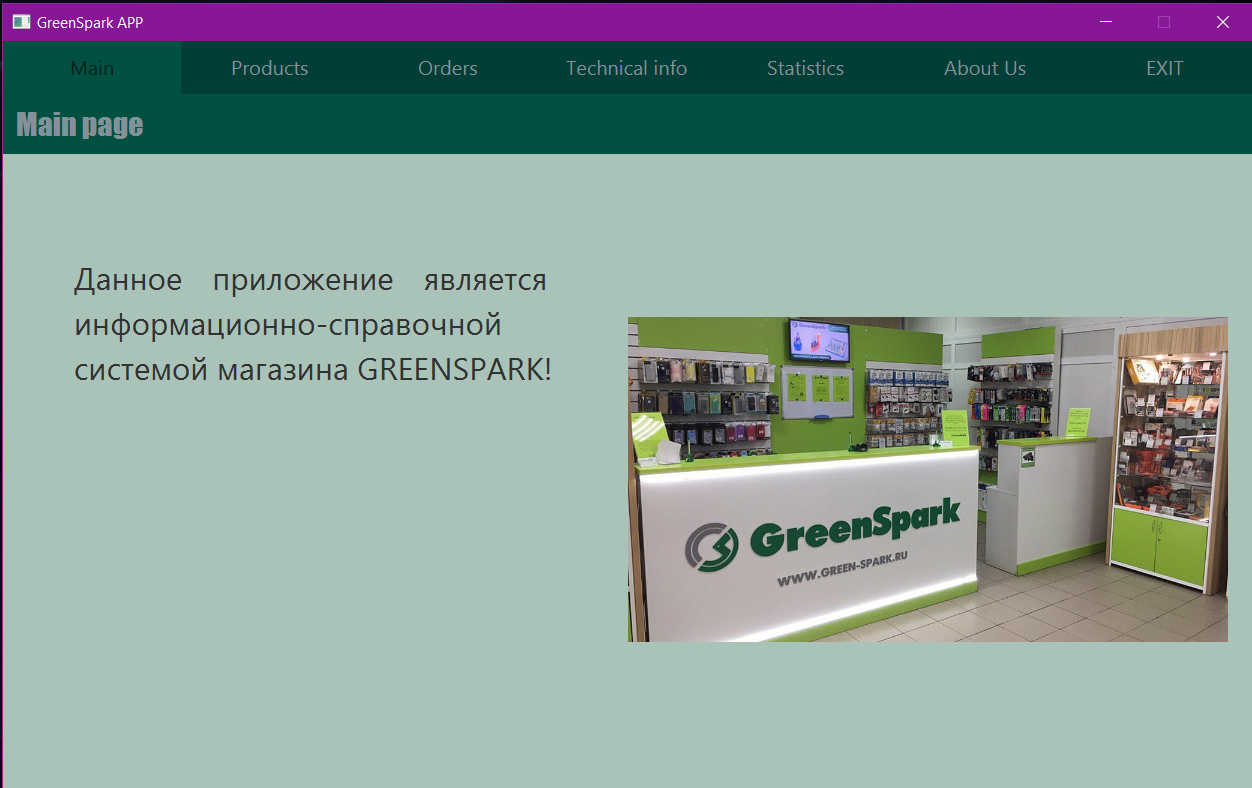
**SVG Path**

**Label**

Рис. №2 (Страница Авторизации)

8

Пользователя встречает главная страница сразу после авторизации. Эта страница имеет Label и одну фотографию (Рис. №3).



**MenuBar**

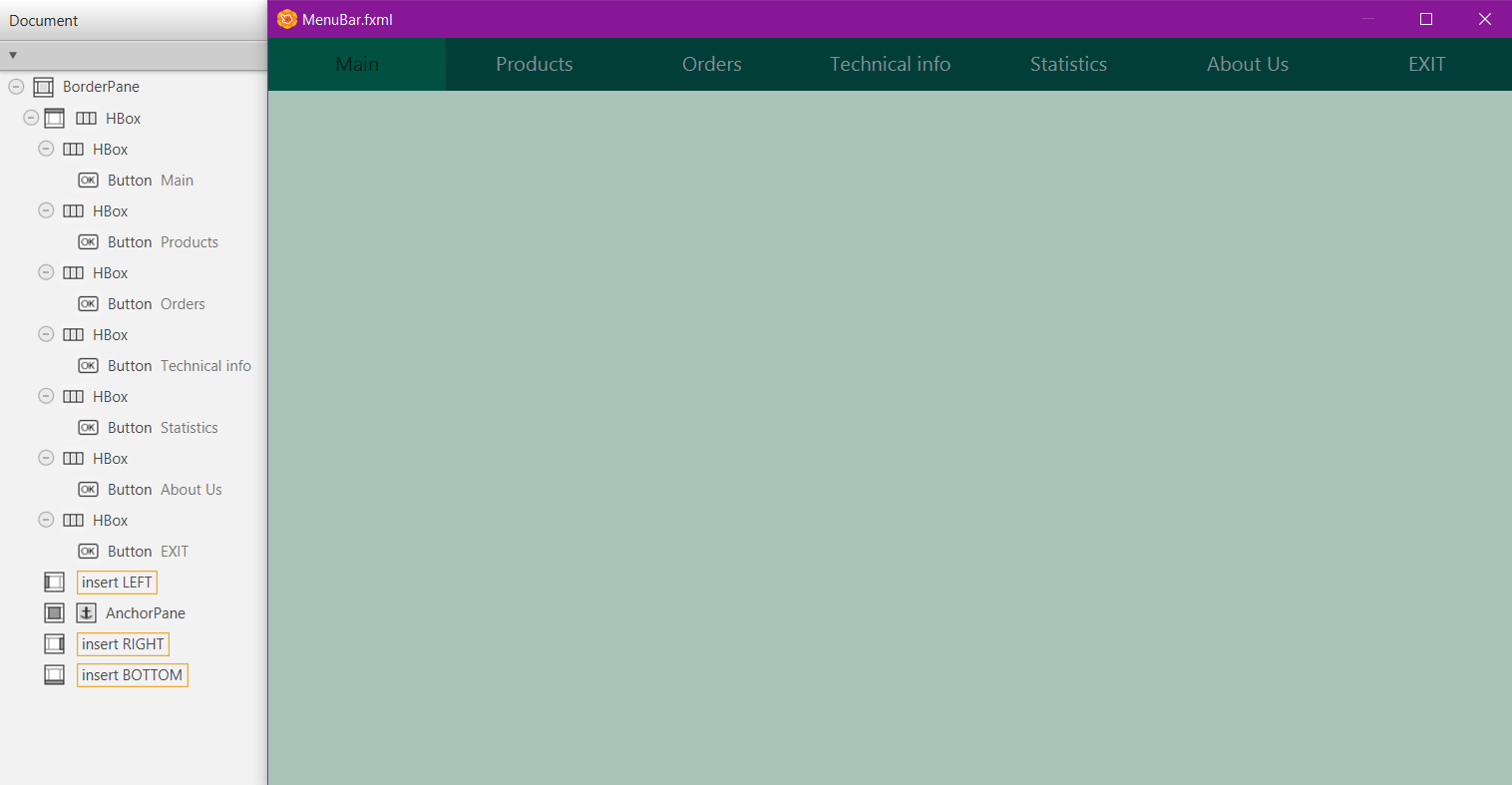
**Label**

**ImageView**

Рис. №3 (Главная страница)

Рисунок №3 сделан методом наложения страниц: главной и страницы меню (Menu Bar). На заднем плане расположено меню (Рис. №4). Оно состоит из кнопок, расположенных в верхней части BorderPane, в центральной части вставлен AnchorPane. В AnchorPane загружаются уже другие страницы, в зависимости от нажатия кнопки.

9



**AnchorPane**

**Buttons**

Рис. №4 (Страница Меню)

Для создания, редактирования и удаления Категории/Компании/Модели есть специальная страница «Техническая информация» (Рис. №5). Для каждой сущности из БД сделана таблица и 3 кнопки управления. В таблице Model можно увидеть связи «один ко многим» (каждая категория/компания может быть у одной или нескольких моделей). Окно добавления/изменения Модели более сложное, поэтому продемонстрирую его (Рис.№6).

Изображение выглядит как стол

Автоматически созданное описание

Рис. №5 (Страница Технической информации)

10

Изображение выглядит как текст, снимок экрана, зеленый, шоссе

Автоматически созданное описание

Рис. №6 (Окно добавления Модели)

На странице продуктов всего одна таблица, но она имеет расширенный функционал. Появилась поисковая строка (Рис. №8). Также эта страница отличается тем, что через таблицу продукты можно создать не только запись продукта (Рис. №9), но и запись заказа (Рис. №10). Для этого необходимо выбрать продукт в таблице и нажать кнопку «New Order».

Изображение выглядит как стол

Автоматически созданное описание

Рис. №7 (Страница продуктов)

11

Изображение выглядит как стол

Автоматически созданное описание

Рис. №8 (Пример работы поисковой строки)

Изображение выглядит как текст, снимок экрана, зеленый

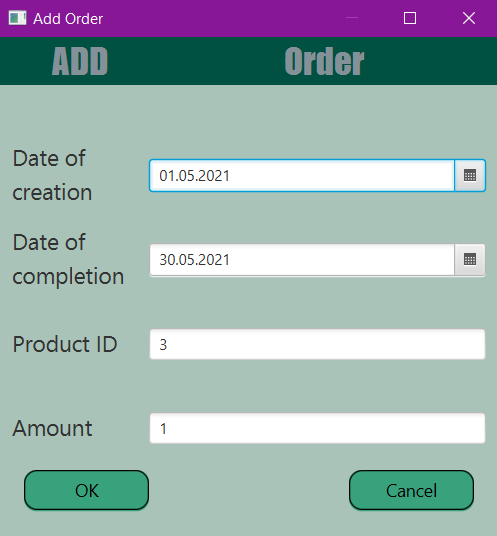
Автоматически созданное описание 

Рис. №9 (Окно добавления продуктов) Рис. №10 (Окно добавления заказа)

Страница заказов также, как страница продуктов, имеет всего одну таблицу, но справа еще есть отображение дополнительной информации, реализованное при помощи Label-ов. При нажатии на запись берётся id продукта и находится поэтому id полная информация. Если запись не выбрана, тогда Label-ы будут пустыми.

12

Изображение выглядит как стол

Автоматически созданное описание

Рис. №11 (Страница заказов)

Следующая страница – это Статистика (Рис. №12). На ней расположена диаграмма (PieChart). Она показывает какое количество продуктов имеется на складе. Данную диаграмму удобно анализировать. Разделение по цветам улучшает восприятие данных и позволяет быстро определить, какие товары в избытке, а какие в дефиците.

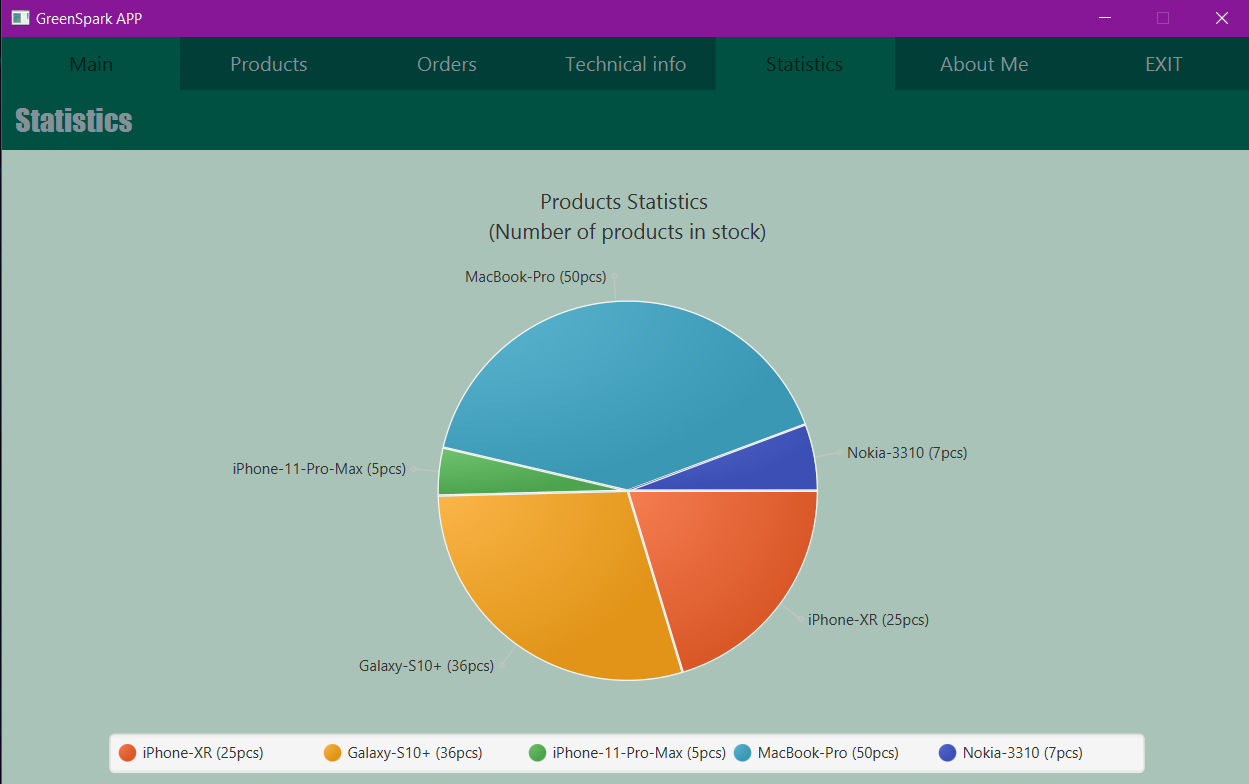


Рис. №12 (Страница статистики)

13

Осталась последняя страница моего приложения, страница «Обо мне». Тут при помощи Label была указана информация о том, кто выполнил данную курсовую работу, кто был научным руководителем этого студента. Мелким шрифтом прикреплена ссылка на GitHub, где можно найти весь код.

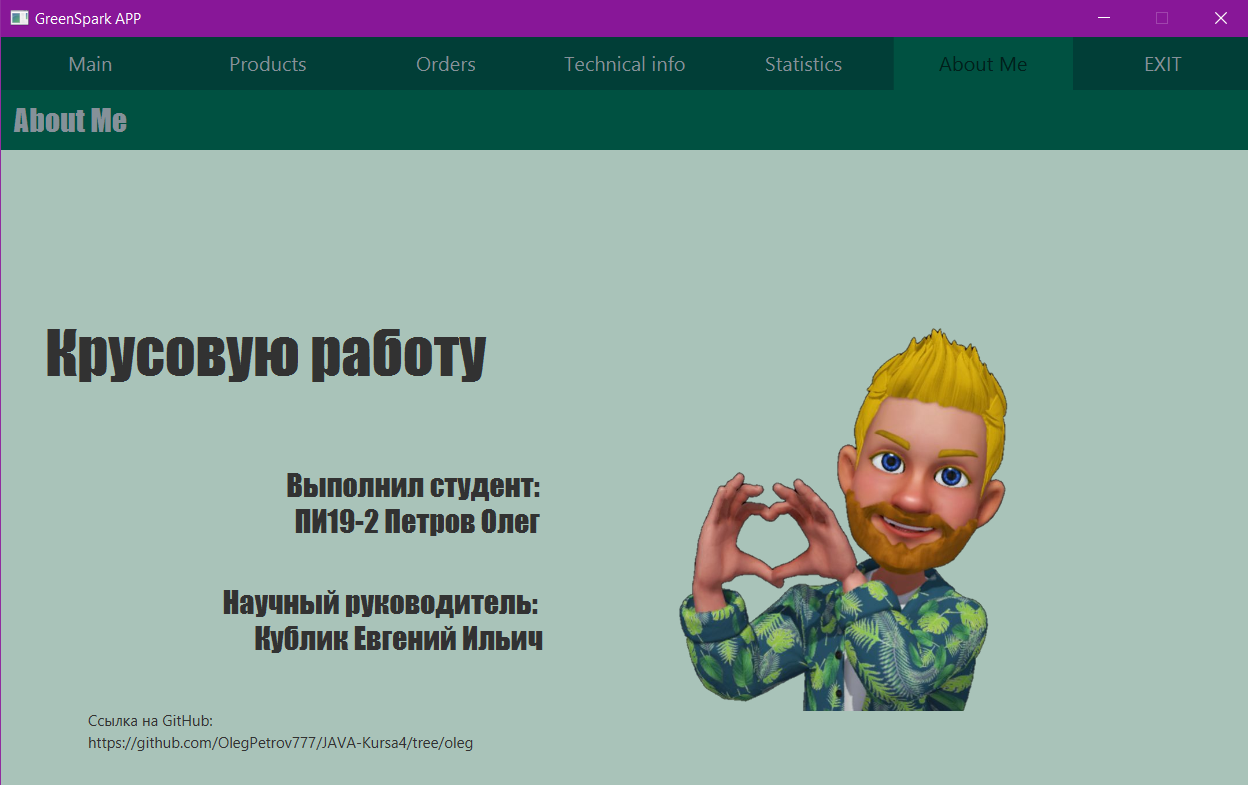


Рис. №13 (Страница «Обо мне»)

14

**4.3. Состав приложения**

Моё приложение является Клиент-Серверным. Программа реализована в трехзвенной архитектуре (клиент – сервера – база данных). Для написания курсовой работы я использовал объектно-ориентированный язык программирования Java, так как он лучше всего подходи для поставленной задачи.

Систему управления базами данных я выбрал PostgreSQL, потому что у меня уже был опыт работы с ней. Я понимал, как я использую её в своём проекте. Для работы с PostgreSQL нужно было скачать PgAdmin.

Просто базы данных недостаточно. Необходимо то, что могло бы ей управлять, вносить изменения. Для этого в моем проекте был написан Сервер. Создав новый проект Maven, я установил универсальный фреймворк для Java. Всё было готово к написанию Сервера.

Далее на очереди Клиентская часть. Чтобы написать Клиент, я решил воспользоваться JavaFX, платформа на основе Java для создания приложений с графическим интерфейсом.

Всех этих инструментов более чем достаточно, чтобы реализовать хорошее Клиент-Серверное приложение. Всё остальное в руках программиста.

15

**5. Назначение и состав классов программы**

**5.1. Описание классов Сервера**

В первую очередь стоит начать с обдумывания логической модели БД, определить какие сущности (таблицы) будут в данном проекте. Моя логическая модели изображена на Рисунке №1. Она состоит из 5 сущности со связями «один ко многим», следовательно должно быть 5 Java-классов. Они хранятся в моем проекте в папке «entity» (Рис. №14)

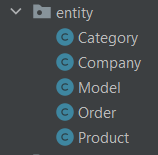


Рис. №14

Класс Category и Company хранят меньше всего информации, у них всего два поля:

* id
* name – название Категории / Компании

16

Класс Model кроме поля «name» имеет ещё информацию о Category и Company. Выглядит это так:

* id
* name – название Модели продукта
* category\_id – id Категории, с помощью которого можно получить полную информацию о Категории
* company\_id – id Компании, аналогично Категории

Класс Product имеет связь с Model (Рис. №1), поэтому можно сразу понять, что в Product будет хранить информацию о Model. Помимо этого, есть и другие поля:

* id
* price – цена продукта
* color – цвет продукта
* count – количество продукта на складе
* model – модель продукта

В классе Order содержится следующая информация о Заказах:

* id
* date\_of\_create – дата создания заказа
* date\_of\_ready – дата готовности к выдачи
* amount – количество заказанного продукта
* product\_id – id Продукта

У каждого из этих классов должен быть Контроллер. Его я разделил на сам Контроллер и Сервис, котором понадобиться свой небольшой Интерфейс (Рис. №15).

17

Интерфейс нам нужен исключительно для того, чтобы наследовать класс JpaRepository<?,?>. Покажу код на примере интерфейс-класса «OrderRepository»:

public interface OrderRepository extends JpaRepository<Order, Long> {

}

Изображение выглядит как текст

Автоматически созданное описание

Рис. №15

Эти интерфейсы играют важную роль в написании сервисов (Рис. №16). Используя объект класса Repository, я получил возможность использования крайне важных методов (Рис. №17). В итоге я для каждого сервиса я написал методы, названия которых говорят сами за себя:

* create
* findAll
* find
* update
* delete

Метод findByName используется не везде, только для тех, кто в Клиенте будет отображаться через ComboBox.

18

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Рис. №16 Рис. №18

Изображение выглядит как текст

Автоматически созданное описание

Рис. №17

Остался последний шаг, чтобы дописать Сервер. Этим шагом являются контроллеры (Рис. №18). Здесь прописываются забросы, которые мы будем делать через Клиент. Для каждого метода сервиса я делаю метод-запрос в контроллере. В моем проекте использовалось всего 4 вида запросов: GET, POST, PUT, DELETE. В коде обозначаются так:

* @GetMapping(value = "/api/order")
* @PostMapping(value = "/api/order")
* @PutMapping("/api/order/{id}")
* @DeleteMapping(value = "/api/order/{id}")

19

**5.2. Описание классов Клиента**

Интерфейс программы уже был описан в пункте 4.2 «Описание интерфейса программы». Каждая страница интерфейса – это fxml-файл. Для того чтобы привести в действие страницу необходимо для нее написать контроллер, поэтому в моём проекте количество fxml-файлов практически совпадает с количеством контроллеров (Рис. №19).

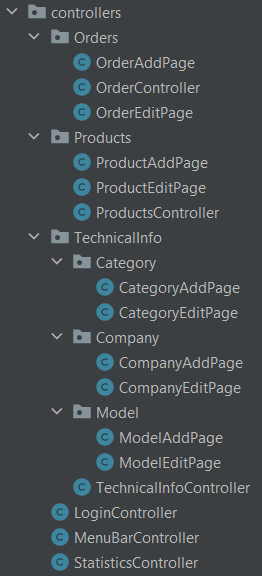
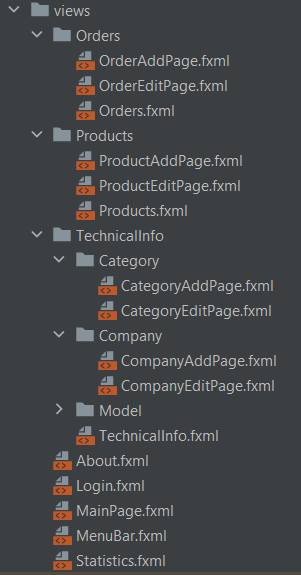


Рис. №19

Я не буду детально разбирать каждый Java-класс, потому что структура всех контроллеров очень схожа.

План написания контроллера для fxml-файла:

1. Объявление всех не статичных элементов интерфейса. Приведу пример объекта Button и TextField:

20

@FXML  
private TextField loginTextField;  
@FXML  
private Button loginButton;

1. Отображение страницы необходимо, чтобы пользователь мог видеть и управлять интерфейсом. Необходимо загрузить из папки resources нужный fxml-файл:  
     
   FXMLLoader loader = new FXMLLoader(LoginController.class.getResource("/views/Login.fxml"));  
   Parent root = loader.load();  
     
   Далее объекту класса Stage задаем сцену (Scene):  
     
   primaryStage.setScene(new Scene(root));  
     
   Чтобы при загрузке страницы вызывались другие методы, отвечающие за управление, нужно либо добавить контроллер, либо создать отдельный метода инициализации. Сначала рассмотрим первый вариант. В нём будет создан контроллер, в который добавлен метод для обработки нажатий кнопок («clickButtons()»):  
     
   LoginController controller = loader.getController();  
   controller.clickButtons();  
     
   Финальный штрих:  
     
   primaryStage.show();

21

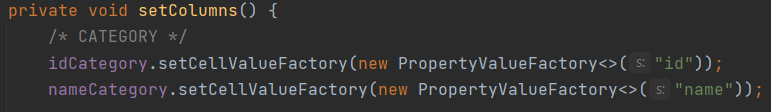
1. Остается написать метод(ы) для управления. Продолжу показывать на примере всё того же класса. Метод обработки нажатия кнопок – clickButtons (Рис. №20). Первые две строчки обозначают то, что кнопки «loginButton» и «exitButton» при активации вызовут методы «signIn» и «exit», соответственно.  
    Изображение выглядит как текст

   Автоматически созданное описание  
    Рис. №20
2. В курсовой работе часто встречающимся элементом, кроме кнопок, является таблица. Рассмотрим этот момент детально на Категории.  
   Для начала требуется создать ObservableList:  
     
   ObservableList < Category > categoryData = FXCollections.observableArrayList();  
     
   Далее в методе setTables чистим, заполняем ObservableList (1-2 строка, Рис. №21) и с помощью метода setItems предаём объекту класса TableView уже заполненный ObservableList.

22  
Изображение выглядит как текст

Автоматически созданное описание  
Рис. №21

Остается распределить значения по колонкам таблицы в методе setColumns (Рис. №22)

  
Рис. №22

Примерно так создается среднестатистический контроллер. Чтобы заполнять данными таблицы, их нужно как-то получить из БД. Для этого понадобиться связать Сервер и Клиент. Связывается со стороны Клиента в папке utils, в двух Java-классах: RestAPI и HttpClass.

Разберу реализацию GET-запроса. В методе «GetCompany» класса «RestAPI» создаю List<Company> для вывода результатов. Получаю в текстовом формате JSON объект из HttpClass.GetRequest и передаю в переменную buffer. После проверки, что buffer не пустой, перевожу buffer из String в JsonArray. Остается парсингом вытащить необходимые данные и создать объект Company, который отправится в List result. Именно этот лист передаётся при вызове данного объекта.

23

Все остальные GET-запросы реализованы аналогично.

Изображение выглядит как текст

Автоматически созданное описание

Рис. №23 (Get-запрос для Компании)

24

**Заключение**

В своей курсовой работе я выполнил главную и единственную цель – разработать информационно-справочную систему магазина цифровой техники. Вместе с этим поставленные задачи были выполнены. Весь функционал, который я хотел осуществить, был реализован. Программа работает стабильно, без авариного завершения. С её помощью действительно можно вести учет товаров, отслеживать заказы и наблюдать за статистикой.

Так как я разрабатывал и писал данный проект лично в малые сроки, некоторые моменты были упущены. К сожалению, на данный момент для реального магазина моё приложение не подойдет, потому что это всего лишь прототип, demo-версия. Однако в будущем, имея большее количество времени, я могу довести дело до конца и добиться полного успеха.

На данный момент я смог добиться выполнения требований курсовой работы. Это дало мне большой багаж знаний, опыт в разработке Клиент-Серверного приложения на языке программирования Java, понимание трехзвенной архитектуры и многое другое.

25

**Список литературы**

1. Козмина Ю., Харроп Р. Spring 5 для профессионалов. - Киев: Диалектика-Вильямс, 2019. - 1120 с.

2. Коузен К. Современный Java. Рецепты программирования . - М.: ДМК Пресс, 2018. - 274 с.

3. Мартин Роберт К. Чистый код. Создание анализ и рефакторинг. - СПб: Питер, 2019. - 464 с.

4. Прохоренок Н.А. JavaFX. - СПб: БХВ-Петербург, 2020. - 768 с.

5. Шилдт Г. Java. Полное руководство. - Киев: Диалектика, 2018. - 1488 с.

26

**Приложение**

27