МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 “Программное обеспечение информационной

безопасности мобильных систем”

Специализация Инженер-программист

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Магазин запчастей»

Выполнил студент Парчевская Виктория Сергеевна

(Ф.И.О.)

Руководитель проекта ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ст. преподаватель Северинчик Н.А. (учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc59059054)

[1 Обзор аналогов приложений «Магазин запчастей» 4](#_Toc59059055)

[1.1 Приложение «APZ.BY – сервис поиска автозапчастей в РБ» 4](#_Toc59059056)

[1.2 Приложение «ЕвроАвто: автозапчасти, сервис» 5](#_Toc59059057)

[1.3 Выводы по главе «Обзор аналогов приложений «Магазин запчастей» 6](#_Toc59059058)

[2 Проектирование программного продукта 7](#_Toc59059059)

[2.1 Проектирование мобильного приложения 7](#_Toc59059060)

[2.2 Проектирование базы данных 10](#_Toc59059061)

[2.3 Выводы по главе «Проектирование программного продукта» 12](#_Toc59059062)

[3 Программная реализация приложений 13](#_Toc59059063)

[3.1 Технические средства для разработки 13](#_Toc59059064)

[3.2 Разработка баз данных 13](#_Toc59059065)

[3.3 Разработка мобильного приложения 13](#_Toc59059066)

[3.4 Вывод по главе «Программная реализация приложений» 18](#_Toc59059067)

[4 Руководство пользователя 19](#_Toc59059068)

[4.1 Выводы по главе «Руководство пользователя» 25](#_Toc59059069)

[ЗАКЛЮЧЕНИЕ 26](#_Toc59059070)

[Список использованных источников 27](#_Toc59059071)

[ПРИЛОЖЕНИЕ А 28](#_Toc59059072)

[ПРИЛОЖЕНИЕ Б 29](#_Toc59059073)

[ПРИЛОЖЕНИЕ В 31](#_Toc59059074)

# **ВВЕДЕНИЕ**

Цель курсового проекта: обеспечение продаж, улучшение и укрепление позиций на рынке услуг и товаров, повышение популярности бренда, увеличение прибыли, привлечение большой целевой аудитории.

Задачи курсового проекта:

1. Изучить аналогичные мобильные приложения.

В рамках этой задачи будет рассмотрено несколько мобильных приложений, представляющих собой онлайн-магазины. Обзор этих приложений позволит нам сориентироваться в необходимом функционале, а также выявить недостатки, которые мы постараемся избежать.

1. Спроектировать приложение.

Проектирование приложения позволит нам увидеть картину предстоящей работы целиком, четко разграничить функции и взаимосвязи сущностей. В результате мы получим некоторый план действий, отталкиваясь от которого нам будет проще реализовать проект.

1. Реализовать приложение.

В рамках курсового проекта будет разработано программное средство, обеспечивающее пользователям быстрый доступ к магазину и статусам заказов.

1. Разработать руководство пользователя.

В результате выполнения этой задачи будет составлено руководство пользователя, предназначенное для описания основных возможностей, а также их ограничений. Это руководство нацелено на пользователей данного приложения, для облегчения знакомства с интерфейсом и функционалом.

# **1 Обзор аналогов приложений «Магазин запчастей»**

# **1.1 Приложение «APZ.BY – сервис поиска автозапчастей в РБ»**

Данное приложение позиционирует себя как сервис поиска б/у и новых автомобильных запчастей по авторазборкам РБ. Регистрация производится на сайте сервиса, и уже после успешной регистрации на сайте появляется возможность войти в свою учетную запись в приложении. Приложение предоставлено скриншоте ниже (рис 1.1.1).

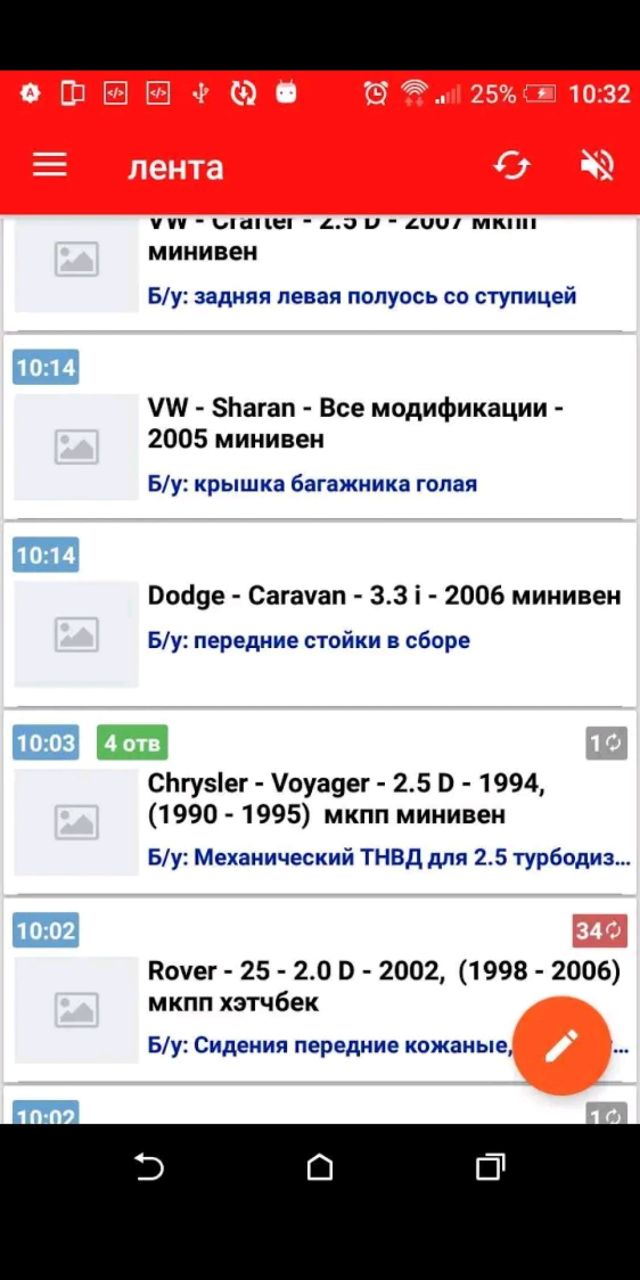


Рисунок 1.1.1 – Каталог приложения APS.BY

Здесь вы сможете как оставить заявку на необходимую деталь, так и, зарегистрировавшись как продавец, оставить свои предложения на уже созданные заявки. Все поступившие предложения закрепляются за определенной заявкой, что в дальнейшем позволяет облегчить покупателю процесс выбора наиболее подходящего продавца.

# **1.2 Приложение «ЕвроАвто: автозапчасти, сервис»**

Это приложение, имеющее приятный минималистический дизайн, содержит ряд основных функций, включающих поиск по детали, добавление детали в корзину, её удаление, возможность оформить заказ и просмотреть его содержимое. Это приложение имеет более традиционную структуру магазина, где покупатель реагирует на предложенный товар, а не продавцы оставляют свои предложения на каждую заявку. Скриншоты каталога и корзины приложения предоставлены скриншоте (рис 1.2.1).

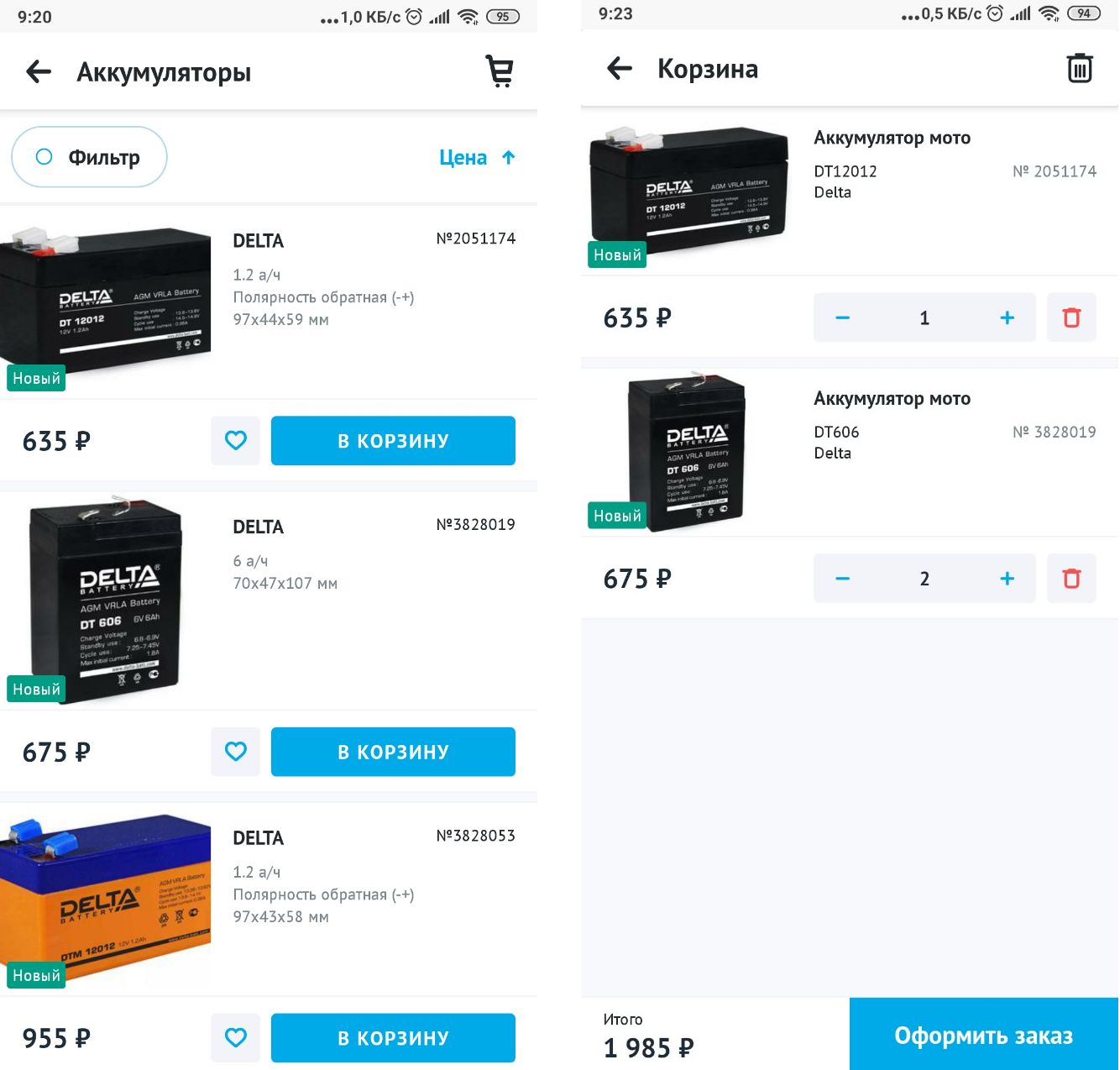


Рисунок 1.2.1 – Приложение «ЕвроАвто»

Помимо этих функций, приложение так же содержит ряд других приятных возможностей, которые, впрочем, здесь рассмотрены не будут.

# **1.3 Выводы по главе «Обзор аналогов приложений «Магазин запчастей»**

В данной главе были рассмотрены два приложения, являющихся аналогами для темы курсового проекта: «APZ.BY – сервис поиска автозапчастей в РБ» и «ЕвроАвто: автозапчасти, сервис». В результате были сделаны следующие выводы.

Дизайн у приложения «ЕвроАвто» более приятный и удобный, чем у первого аналога, интерфейс интуитивно понятный. Структура приложения является более привычной для покупателей, где продавец предоставляет доступ ко всему каталогу, а пользователь сам ищет, сравнивает и выбирает. Приложения не обладает явными минусами. К недостатку можно отнести то, что состояние заказа не отслеживается.

В случае приложения «APZ.BY», интерфейс не настолько прост и понятен, для освоения понадобилось больше времени. Несмотря на нетипичную структуру приложения, где продавцы реагируют на заявки покупателей, эта система может оказаться достаточно удобной для покупателей, учитывая редкость некоторых деталей. Так же то, что ко всем предложениям к конкретной заявке можно получить доступ с неё, является так же удобной навигационной особенностью. Таким образом, если другой покупатель будет искать тот же товар, найдя схожую заявку он получит доступ сразу ко всем уже сделанным предложениям.

К явным минусам приложения «APZ.BY» можно отнести отсутствие какого-либо поиска, все объявления выкладываются в общую ленту, что делает поиск крайне неудобным.

Для реализации приложения была выбрана более классическая структура. Более близким аналогом можно назвать приложение «ЕвроАвто: автозапчасти, сервис».

# **2 Проектирование программного продукта**

# **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать мобильное приложение позволяющего покупателям просматривать каталог, добавлять детали в корзину, оформлять заказы и отслеживать их статусы. Роль администратора имеет другие возможности: добавление деталей в каталог и их редактирование, просмотр всех заказов и изменение статусов.

При запуске приложения перед пользователем появляется окно входа и регистрации. После авторизации определяется роль пользователя. Если пользователь имеет роль администратора, то ему будет предоставлен доступ к активности с двумя функциональными вкладками: каталог и список заказов. Для покупателя после авторизации открывается активность, имеющая четыре функциональные вкладки: каталог, корзина, профиль и заказы, функционал которых был описан ранее.

Для описания функциональности можно посмотреть UML-диаграмму, а именно диаграмму вариантов использования. Она отображает функциональность программного средства с точки зрения получения значимого результата для пользователя с ролью покупателя.

UML-диаграммой вариантов использования для покупателя на рисунке 2.1.1.

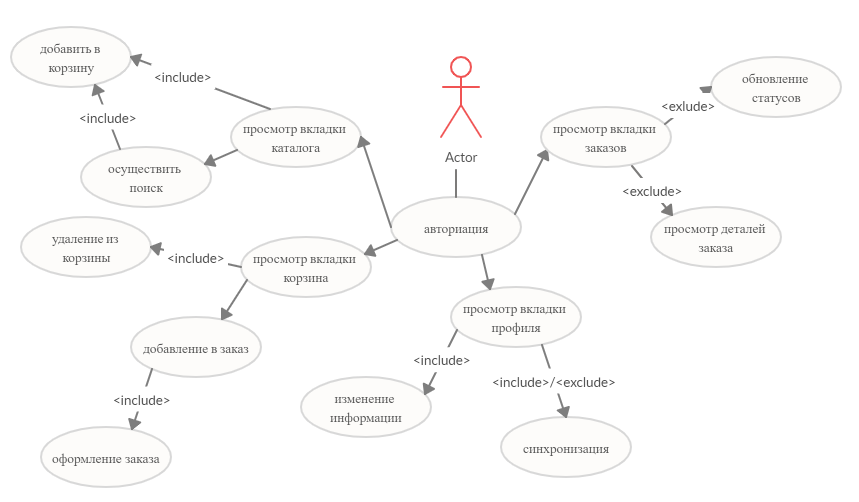


Рисунок 2.1.1 – Диаграмма вариантов использования покупателем

Таким образом, можно подытожить функции, доступные покупателю:

* добавление деталей в корзину;
* удаление детали из корзины;
* оформление заказа;
* отслеживание статуса заказа;
* изменение персональной информации.

Рассмотрим роль администратора подробнее. На вкладке каталог предоставлен весь ассортимент деталей, который администратор может пополнять, изменять и делать недоступным. Вкладка заказы содержи в себе все заказы всех пользователей. При нажатии на конкретный заказ будет предоставлен выбор: открыть заказ, чтобы увидеть входящие в него детали, или изменить статус. Вверху фрагмента находится поиск по покупателю.

Теперь рассмотрим UML-диаграмму вариантов использования для администратора. Она отображает функциональность программного средства с точки зрения получения значимого результата для пользователя с ролью администратор. (рис. 2.1.2)

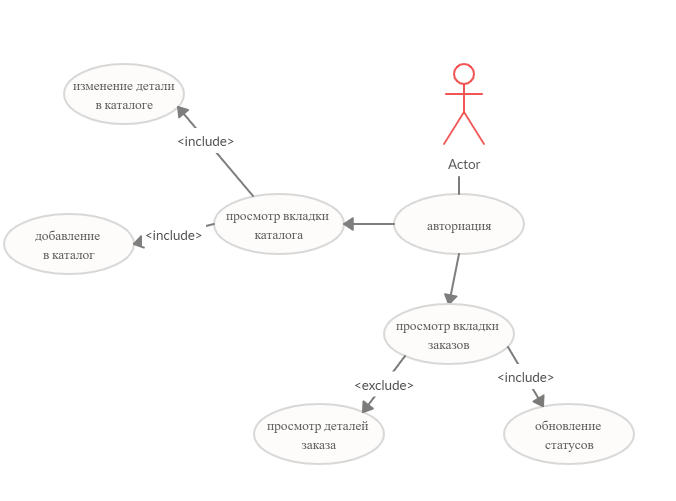


Рисунок 2.1.2 – Диаграмма вариантов использования для администратора

Таким образом приложение должно выполнять функции:

* представление каталога;
* добавление деталей;
* редактирование детали;
* поиск заказа по клиенту;
* просмотр заказов;
* изменение статуса заказов.

# **2.2 Проектирование базы данных**

В базе данных данного приложения должны быть реализованы пять таблиц. В каждой из таблиц необходимо добавить поля для синхронизации данных. Схема базы данных представлена на рисунке 2.2.1.

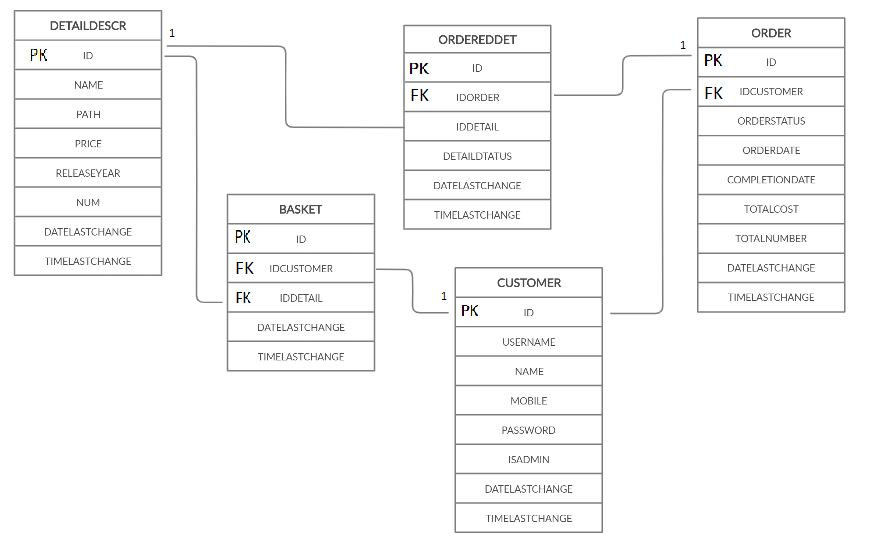


Рисунок 2.2.1 – Схема базы данных

В таблице Customer содержится информация об покупателе, а также информация, необходимая для входа в систему.

* ID – первичный ключ, представленный целочисленным значением
* USERNAME – поле, необходимое для входа в систему, представленное строковым типом данных
* NAME – имя клиента, представленное строковым типом данных
* MOBILE – номер телефона клиента, поле, представленное строковым типом данных
* PASSWORD – поле, необходимое для входа в систему, представленное строковым типом данных
* ISADMIN предназначено для определения роли пользователя в системе
* DATELASTCHANGE – целочисленный тип данных для хранения даты последнего изменения. Поле необходимое для синхронизации.
* TIMELASTCHANGE – целочисленный тип данных для хранения времени последнего изменения. Поле необходимое для синхронизации.

Таблица Detaildescr предназначена для хранения информации о детали.

* ID – первичный ключ, представленный целочисленным значением
* NAME – имя детали, представленное строковым типом данных
* PATH – поле строкового типа данных для хранения пути к изображению
* PRICE – поел типа данных, предназначенное для хранения стоимости детали
* RELEASEYEAR – год выпуска детали, поле представлено целочисленным значением
* NUM INTEGER – количество доступных деталей, поле представлено целочисленным значением
* DATELASTCHANGE – целочисленный тип данных для хранения даты последнего изменения. Поле необходимое для синхронизации.
* TIMELASTCHANGE – целочисленный тип данных для хранения времени последнего изменения. Поле необходимое для синхронизации.

Таблица Basket предназначена для хранения деталей, который пользователь добавил

* ID – первичный ключ, представленный целочисленным значением
* IDCUSTOMER – внешний ключ, представленный целочисленным значением
* IDDETAIL INTEGER – внешний ключ, представленный целочисленным значением
* DATELASTCHANGE – целочисленный тип данных для хранения даты последнего изменения. Поле необходимое для синхронизации.
* TIMELASTCHANGE – целочисленный тип данных для хранения времени последнего изменения. Поле необходимое для синхронизации.

Таблица ORDEREDDET предназначена для хранения заказанных деталей

* ID – первичный ключ, представленный целочисленным значением
* IDORDER – внешний ключ, представленный целочисленным значением
* idDetail – внешний ключ, представленный целочисленным значением
* DetailStatus – поле для обозначения текущего статуса детали, строковый тип данных
* DATELASTCHANGE – целочисленный тип данных для хранения даты последнего изменения. Поле необходимое для синхронизации.
* TIMELASTCHANGE – целочисленный тип данных для хранения времени последнего изменения. Поле необходимое для синхронизации.

Таблица ORDER хранит данные о заказе

* ID – первичный ключ, представленный целочисленным значением
* IDCUSTOMER – внешний ключ, представленный целочисленным значением
* ORDERSTATUS – поле для обозначения текущего статуса заказа, строковый тип данных
* ORDERDATE – поле для хранения даты заказа
* COMPLETIONDATE – поле для хранения даты выполнения заказа
* TOTALCOST – поле для хранения стоимости заказа
* TOTALNUMBER – поле для хранения количества деталей в заказе, целочисленный тип данных

# **2.3 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование всех компонентов приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны диаграмма вариантов использования приложения пользователями с разными ролями.

Была спроектирована база данных, где были определены необходимые таблицы и поля для каждой из них. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

# **3 Программная реализация приложений**

# **3.1 Технические средства для разработки**

Для разработки приложений курсового проекта были использованы следующие технические средства:

SQLite – компактная встраиваемая СУБД представляющая собой библиотеку.

Android Studio – интегрированная среда разработки для работы с платформой Android.

Java – строго типизированный объектно-ориентированный язык программирования.

SQLCipher – библиотека с открытым исходным кодом, созданная компанией Zetetic, для прозрачного 256-битного AES шифрования баз данных SQLite.

OkHttp – библиотека предоставляющая собой простой и удобный клиент для работы с HTTP запросами.

Newtonsoft.Json – библиотека позволяющую преобразовывать объекты в JSON формат с большим количеством настроек.

# **3.2 Разработка баз данных**

Для хранения данных на телефоне будет использоваться встраиваемая СУБД SQLite, а для обеспечения безопасности будет использована библиотека SQLCipher. Код создания структуры мобильной базы данных представлен в Приложении А.

# **3.3 Разработка мобильного приложения**

Мобильное приложение было реализовано в среде разработки – Android Studio. Для написания приложения был использован язык программирования Java.

Графическая структура проекта представлена на рисунке 3.3.1.

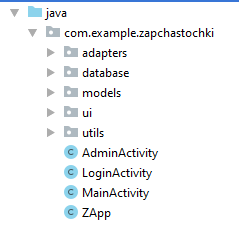


Рисунок 3.3.1 – Графическая структура проекта

Пакет adapters содержит адаптеры для заполнения RecyclerView данными.

Пакет database содержит в себе класс для работы с базой данных.

Пакет models содержит в себе модели данных.

Пакет ui содержит классы для работы с макетами.

Пакет utils содержит вспомогательные классы и сервисы.

В основном пакете классы активностей и приложения.

Теперь рассмотрим подробнее основные пакеты и классы. Содержимое пакета adapters представлено в таблице (см. таблицу 3.3.1).

**Таблица 3.3.1 – Содержание пакета adapters**

|  |  |
| --- | --- |
| Класс AdminDetailRecyclerAdapter – адаптер для заполнения макетов с RecyclerView данными. | |
| onCreateViewHolder | Метод для создания объекта ViewHolder |
| onBindViewHolder | Метод отвечает за связь Java объекта и View |
| getItemCount | Метод возвращает количество элементов в списке |
| Класс ViewHolder – предоставляет доступ ко всем view | |
| setDetail | Метод заполняет разметку данными объекта списка |
| onClick | Метод определяет реакцию на нажатие |

Остальные классы (DetailRecyclerAdapter, OrderedDetailRecyclerAdapter, OrderRecyclerAdapter) реализованы по тому же принципу и отличаются лишь java-объектами, с которыми работают.

Пакет database хранит в себе класс для работы с базой данных. Все методы описаны в таблице ниже (см. таблицу 3.3.2).

**Таблица 3.3.2 – Содержание пакета database**

|  |  |
| --- | --- |
| Класс CategorieResipeActivity – отображение списка категорий | |
| onCreate | Метод вызывается при создании базы данных |
| onUpgrade | Метод вызывается, когда увеличивается версия базы данных |
| onConfigure | Метод вызывается для конфигурации и включения внешних ключей |
| setDateTimeLastChanged | Метод для заполнения сервисных полей в базе данных |
| customerIsExist | Метод вызывается для проверки данных для авторизации |
| selectCustomer | Метод присваивает статической переменной данные авторизованного пользователя |
| insertCustomer | Метод вызывается при регистрации нового пользователя |
| updateProfile | Метод вызывается, чтобы записать измененные данные о пользователе во вкладке профиля |
| getAllCustomerInLocal | Метод нужен для синхронизации. Возвращает список пользователей с указанным id |
| getNewCustomerInLocal | Метод возвращает список новых записей о пользователях в локальную базу данных |
| insertCustomerId | Метод вызывается для вставки данных, пришедших с сервера. |
| updateCustomer | Метод вызывается для обновления данных, пришедших с сервера. |
| updateDetailStatus | Метод вызывается для обновления статуса детали |
| updateOrderStatus | Метод вызывается для обновления статуса заказа |
| updateOrderCompletionDate | Метод вызывается для обновления даты выполнения заказа |

Пакет models хранит в себе классы, которые позволяют представить данные базы данных в виде объектов (см. таблицу 3.3.3).

**Таблица 3.3.3 – Содержание пакета models**

|  |  |  |
| --- | --- | --- |
| Customer – описание объекта Customer | | |
| CURRENT\_USER | Хранит в себе данные текущего пользователя | |
| DetailDescr – описание объекта Detaildescr | | |
| DetailStatus – перечисление, содержащее в себе статусы для деталей в заказе | | |
| Order – описание объекта Order | | |
| customer | | Свойство для хранения данных объекта Customer |
| OrderedDet – описание объекта Ordereddet | | |
| OrderedDetail – наследуется от класса DetaulDescr. Используется для хранения информации их представления и заполнения адаптера. | | |
| OrderStatus – перечисление, хранящее статусы для заказов | | |

Пакет ui.admin хранит в себе классы для работы с фрагментами (см. таблицу 3.3.4).

**Таблица 3.3.4 – Содержание пакета ui.admin**

|  |  |
| --- | --- |
| Класс AddDetailFragment используется для управления фрагментом для добавления деталей в каталог | |
| onActivityResult | Метод для работы с изображениями |
| checkPermissions | Метод для проверки разрешений на записи в память устройства |
| onClick\_save | Метод для обработки и сохранения введеных в форму данных |
| Класс AdminDetailCatalogFragment – работа с датой и временем | |
| onDetailClick | Метод для обработки нажатого элемента каталога |
| onCreateOptionsMenu | Метод для наполнения опционального меню |
| onOptionsItemSelected | Метод для обработки выбранного пункта меню |
| Класс AdminOrdersDetailsFragment – для работы с фрагментом, содержащим информацию о деталях, входящих в заказ | |
| onDetailClick | Метод для обработки нажатого элемента |
| showActionsDialog | Метод для отображения диалога для смены статуса детали |
| Класс AdminOrdersFragment – для работы с фрагментом, содержащим список заказов | |
| onOrderClick | Метод обрабатывающий нажатие на заказ |
| showActionsDialog | Метод для открытия окна для смены статуса заказа |
| onCreateOptionsMenu | Для создания опионального меню |
| EditDetailFragment – класс для изменения данных детали. | |

Пакет utils хранит в себе классы для вспомогательной работы и сервисов. (см. таблицу 3.3.5).

**Таблица 3.3.5 – Содержание пакета utils**

|  |  |
| --- | --- |
| Класс DateTime – работа с датой и временем | |
| getDateSpecialFormat | Метод возвращающий дату в формате "yyMMdd" |
| getTimeSpecialFormat | Метод возвращающий дату в формате "HHmmss" |
| addMinutes | Метод прибавляющий к переданной дате в аргументе определённое количество минут |
| Класс Internet – работа с сетью | |
| checkInternetConenction | Метод проверяющий соединение с интернетом |
| serverIsReachable | Метод проверяющий доступен ли сервер |
| Settings – доступ к настройкам приложения | |
| setValue | Метод задающий значения |
| getValue | Метод получающий значения |
| SyncSomeTable – позволяет синхронизировать таблицы с сервером | |
| Update | Метод возвращающий данные с сервера |
| Upgrade | Метод для загрузки данных на сервер |
| updateSratuses | Метод возвращает данные с сервера для обнволения |
| SyncStatuses – сервис для синхронизации статусов конкретного пользователя | |
| onHandleIntent | Метод, вызываемый при старте сервиса |
| syncOrderStatus | Метод получающий данные с сервера и обновляющий их в локальной базе данных |
| SyncOnRegistration – синхронизация при регистрации нвоого пользователя | |
| onHandleIntent | Метод, вызываемый при старте сервиса |
| SyncCustomerTable | Метод, синхронизирующий таблицу с пользователями |
| Sync – аналогичен SyncOnRegistration, только синхронизирует и остальные таблицы | |

Пакет com.example.zapchastochki так же хранит в себе классы, являются основой для работы с фрагментами (см. таблицу 3.3.6). Активности AdminActivity и MainActivity так же защищены от скриншотов и записи видео в целях защиты информации, а так же в метод onResume() добавлен код, вызывающий окно с запросом отпечатка пальца для продолжения работы в профиле. Ознакомиться с кодом фрагмента с запросом отпечатка пальца можно в Приложении Б.

**Таблица 3.3.6 – Содержание пакета com.example.zapchastochki**

|  |
| --- |
| Класс AdminActivity – открывается, если в систему вошел администратор |
| Класс MainActivity – открывается, если в систему вошел не администратор |
| Класс LoginActivity является стартовой активностью и используется для авторизации |

Ознакомится с диаграммой классов разработанного приложения можно в Приложении В. Из этой диаграммы видно, что есть классы, связанные между собой отношением ассоциации. Ассоциация – структурное отношение, описывающее совокупность связей, где под связью понимается некоторая смысловая связь между объектами. Разновидностью ассоциации является агрегирование – так называется структурное отношение между целым и его частями.

# **3.4 Вывод по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне клиента.

Третий этапом стала разработка мобильного приложения, где была представлена графическая структура приложения, рассмотрен каждый класс и их методы, в частности.

# **4 Руководство пользователя**

Для того чтобы установить приложения на телефон необходимо иметь версию Android минимум 8.0 и версию API не ниже 28. Версию Android и API вашего телефона можно узнать в настройках телефона.

Если версия Android подходит, тогда можно устанавливать приложение на мобильный телефон. В меню ваших приложений или на главном экране смартфона появится новая иконка с именем «zapchastochki». Запустим приложение нажатием на иконку приложения. Откроется страница входа или регистрации. Для того, чтобы зайти в приложении необходимо зарегистрироваться и/или войти в уже созданный профиль.

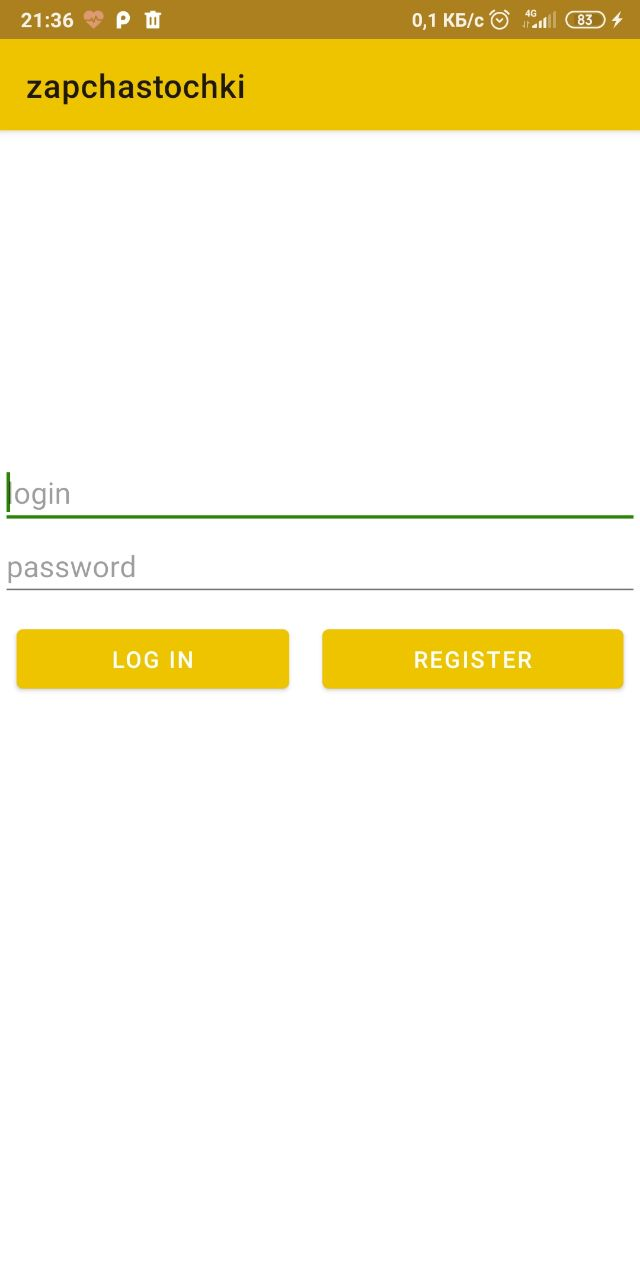


Рисунок 5.1 – Активность входа и регистрации

При нажатии кнопки Log In, а также при открытии свернутого приложения в дальнейшем, появится окно с запросом отпечатка пальца (рис. 5.2). После нажатия на кнопку Login откроется окно, отображающее данные процесса аутентификации.

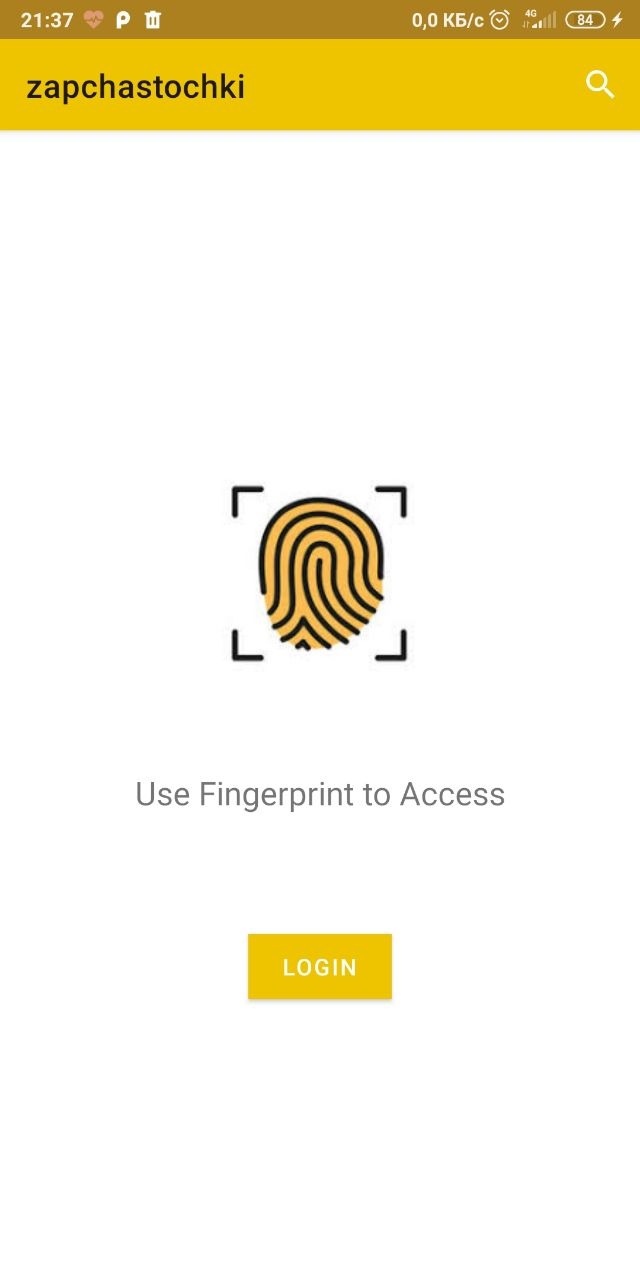


Рисунок 5.2 – Фрагмент аутентификации по отпечатку пальца

После того, как вы войдете или зарегистрируетесь как обычный пользователь и пройдете аутентификацию по отпечатку пальца, вам откроется страница с каталогом всех деталей (рис. 5.3). Стрелка №1 указывает на кнопку поиска по названию детали. Стрелка №2 указывает на элемент каталога. Если нажать на этот элемент, то деталь добавится в корзину. Стрелка №3 указывает на панель навигации, с помощью которой можно попасть в остальные разделы.

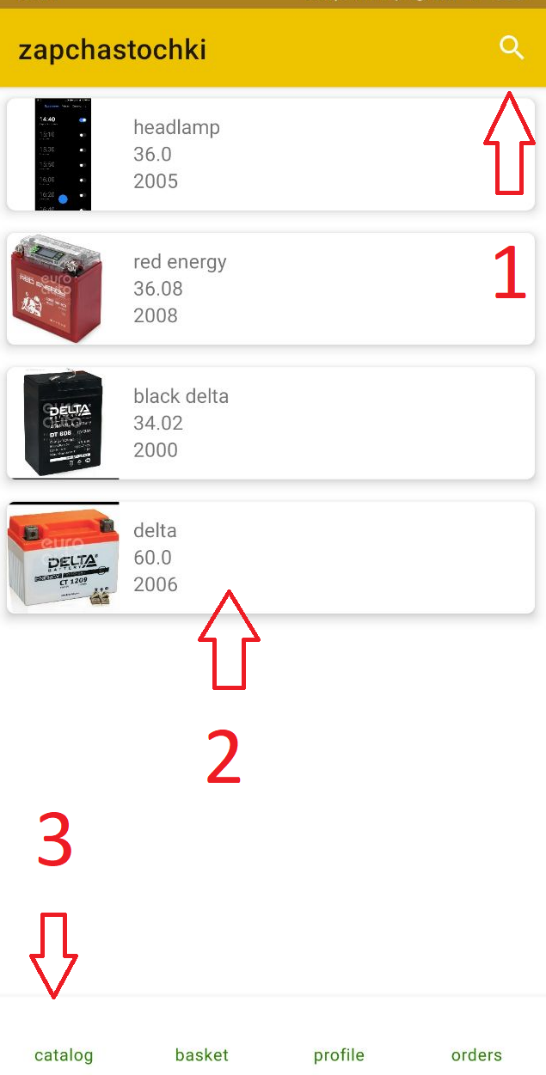


Рисунок 5.3 – Страница каталога

После нажатия на элемент каталога он добавляется в корзину, перейти к которой можно выбрав второй элемент слева на панели навигации, называемый «Basket». Перейдя в корзину, вы увидите экран, показанный на рисунке 5.4.

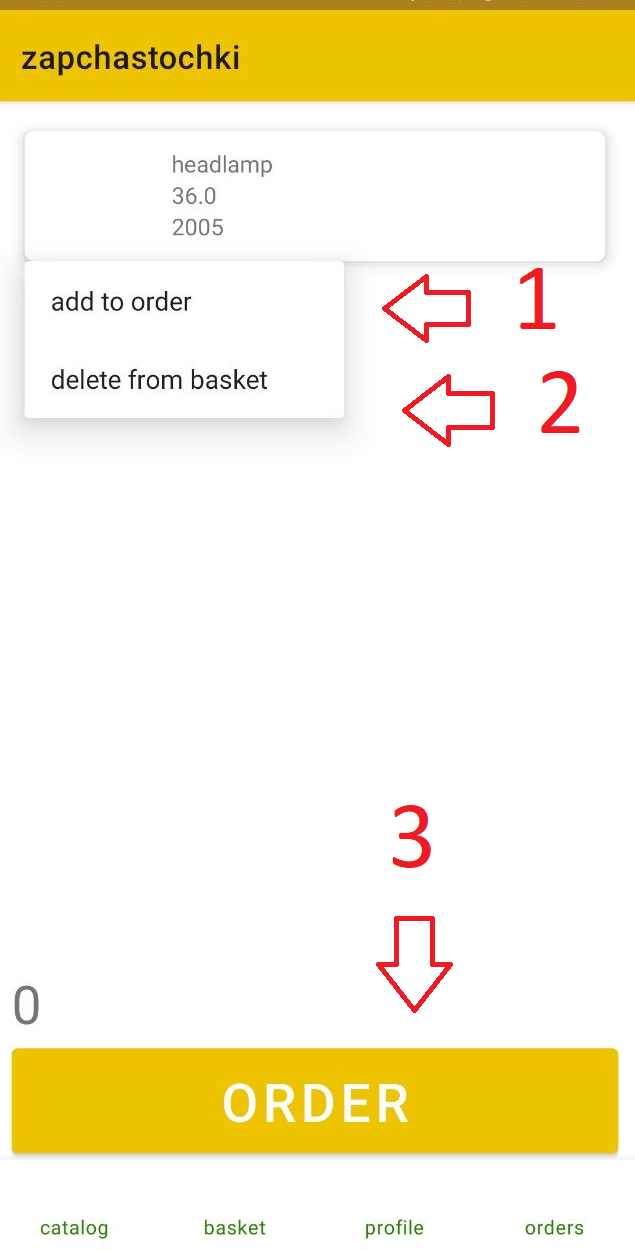


Рисунок 5.4 – Основные элементы управления со страницы корзины

Элементы в корзине также кликабельны. При нажатии на элемент в корзине вам будет предложено добавить деталь в заказ (стрелка №1) или удалить из корзины (стрелка №2). Если вы выбираете добавить деталь в заказ, то сам элемент подсветится желтым цветом. После того, как вы сформировали заказ, нажмите кнопку, указанную стрелкой №3. Вам откроется список деталей, входящих в заказ. Здесь вы можете удалить лишние детали из заказа, кликнув на деталь и выбрав соответствующий пункт меню. Если заказ соответствует вашим желаниям, нажмите снова кнопку «Order», чтобы подтвердить заказ. Если все прошло успешно, высветится окошко, сообщающее, что заказ подтвержден.

Просмотреть свои заказы вы можете нажав на кнопку, отмеченную стрелкой №1 (см. рис. 5.5).

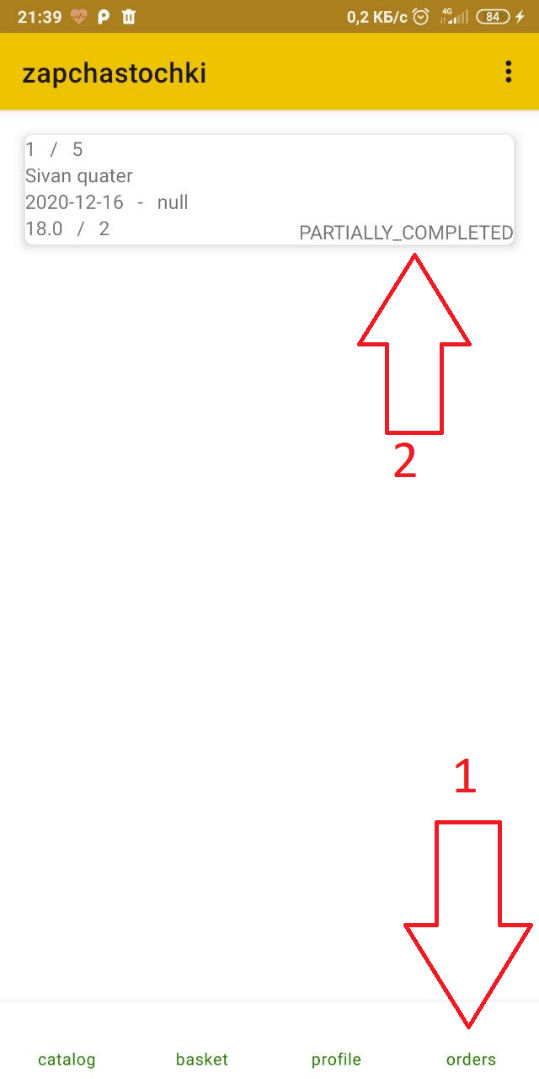


Рисунок 5.5 – Основные элементы управления для страницы заказов

Статус заказа указан стрелкой №2. При нажатии на заказ вам откроются его подробности.

Последняя вкладка, доступная пользователю – вкладка профиля (рис. 5.6). На вкладке профиля указана информация пользователя. Здесь можно и нужно изменить поля name и number. Для того, чтобы их изменить, достаточно ввести новый текст и нажать кнопку «Same changes».

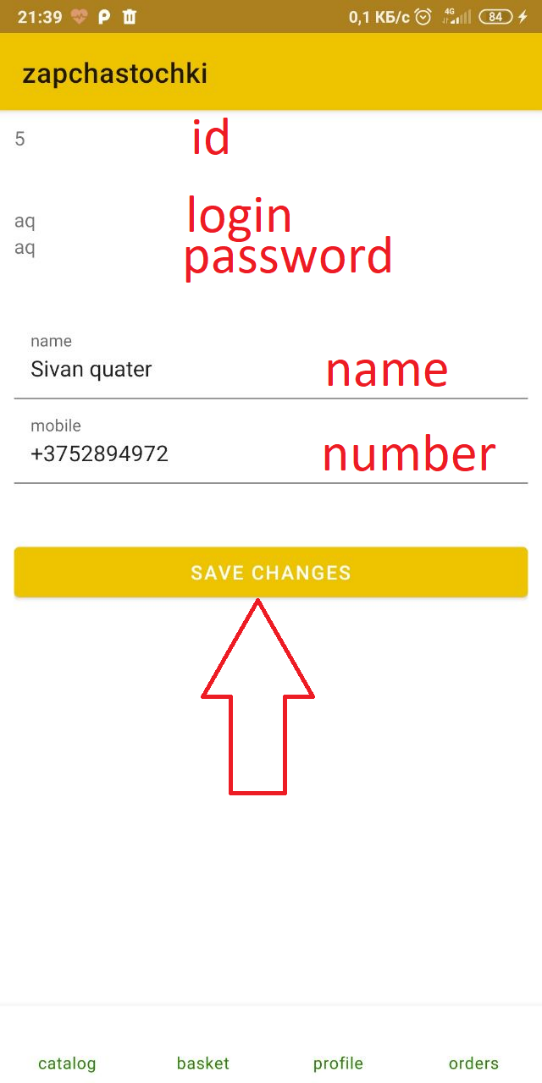


Рисунок 5.6 – Элементы управления страницы профиля

Если вы вошли в приложении под ролью администратора, вам будет доступно две вкладки в нижнем навигационном меню (рис. 5.7). Стрелка №1 указывает на кнопку, открывающую фрагмент для добавления новой детали.

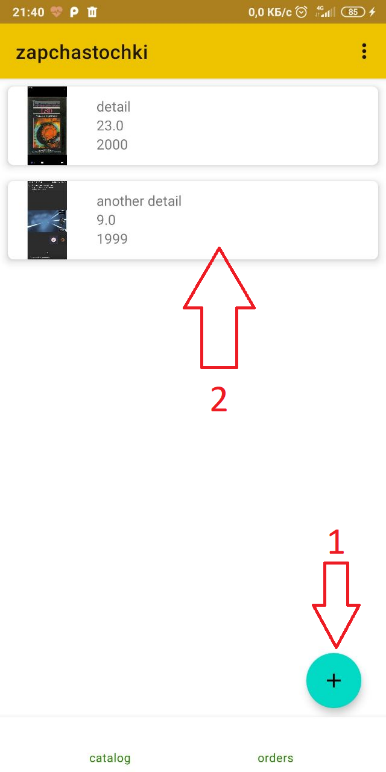


Рисунок 5.7 – Страница администратора

При нажатии на существующую деталь, указанную стрелкой №2, в каталоге будет предложен выбор: удалить или изменить деталь.

На второй вкладке администратору доступны все заказы. При нажатии на заказ, администратору будет предложено изменить статус заказа либо просмотреть его детали, у которых так же можно изменить статус.

# **4.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя.

Данное руководство позволяет пользователю узнать требования для установки приложения на телефон, а также как использовать все функции приложения. Были предоставлены скриншоты самого приложения для большей наглядности использования.

# **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта было обеспечение продаж, улучшение и укрепление позиций на рынке услуг и товаров, повышение популярности бренда, увеличение прибыли, привлечение большой целевой аудитории.

Были решены следующие поставленные задачи:

* обзор аналогов;

В результате было рассмотрено 2 приложения и были выявлены их положительные и отрицательные стороны. Эта информация была использована для улучшения функционала приложения и пользовательского интерфейса.

* проектирование приложения;

Был спроектирован каждый отдельный компонент приложения. В мобильном приложении был определен основной функционал, также были разработаны необходимые диаграммы. При проектировании базы данных, были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных и HTTP методы для работы с серверной частью.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: СУБД SQLite и MySQL, сред разработки Android Studio и Microsoft Visual Studio, языков программирования Java и C#, модульной платформы .NET Core, библиотек Entity Framework, OkHttp, Newtonsoft.Json.

Были реализованы базы данных на стороне сервера и клиента.

При решении данной задачи была достигнута поставленная цель, и был создан проект «Магазин деталей», который включает в себя android-клиент и сервер.

* разработка руководства пользователя.

Разработано руководство, позволяющее пользователю узнать требования для установки приложения на телефон, а также как использовать все функции мобильного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

* добавление новых деталей;
* просмотр всех деталей;
* изменение существующих деталей;
* добавление детали в корзину;
* поиск необходимой детали по названию;
* оформление заказа;
* отслеживание статуса;
* синхронизация с сервером.

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что задачи были выполнены в полном объеме.

# **Список использованных источников**

1 Архитектура REST [Электронный ресурс]. / habrahabr.ru – Режим доступа: https://habrahabr.ru/post/38730 – Дата доступа: 29.04.2020

1. Программирование под ОС Андроид [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/java/android. Дата доступа: 06.04.2020
2. Руководство по ASP.NET Core 3 [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sharp/aspnet5. Дата доступа: 23.01.2020
3. Руководство по MySQL [Электронный ресурс] / Metanit.com. – Режим доступа: https://metanit.com/sql/mysql. Дата доступа: 26.05.2019

**ПРИЛОЖЕНИЕ А**

Код создания структуры мобильной базы данных

db.execSQL(**"CREATE TABLE if not exists CUSTOMER(\n"** +  
 **"\tID INTEGER PRIMARY KEY,\n"** +  
 **"\tUSERNAME TEXT,\n"** +  
 **"\tNAME TEXT,\n"** +  
 **"\tMOBILE TEXT,\n"** +  
 **"\tPASSWORD TEXT,\n"** +  
 **"\tISADMIN INTEGER\n"** +  
 **"\tCHECK (ISADMIN IN (0,1)) \n"** +  
 **"\tDEFAULT 0,\n"** +  
 **"\tDateLastChange integer,\n"** +  
 **"\tTimeLastChange integer\n"** +  
 **");"**);

db.execSQL(**"CREATE TABLE if not exists DETAILDESCR(\n"** +  
 **"\tID INTEGER PRIMARY KEY,\n"** +  
 **"\tNAME TEXT NOT NULL,\n"** +  
 **"\tIMAGE TEXT NOT NULL,\n"** +  
 **"\tPRICE NUMERIC NOT NULL,\n"** +  
 **"\tRELEASEYEAR INTEGER CHECK(RELEASEYEAR BETWEEN 1800 AND strftime('%Y', CURRENT\_DATE)),\n"** +  
 **"\tNUM INTEGER, \n"** +  
 **"\tISAVAILABLE INTEGER CHECK (ISAVAILABLE IN (0,1)) DEFAULT 1,\n"** +  
 **"\tDateLastChange integer,\n"** +  
 **"\tTimeLastChange integer\n"** +  
 **");"**);  
db.execSQL(**"CREATE TABLE if not exists ORDERS(\n"** +  
 **"\tID INTEGER PRIMARY KEY,\n"** +  
 **"\tIDCUSTOMER INTEGER,\n"** +  
 **"\tSTATUS TEXT CHECK(STATUS IN (\"ACCEPTED\", \"COMPLETED\", \"PARTIALLY\_COMPLETED\", \"CANCELED\")) DEFAULT \"ACCEPTED\",\n"** +  
 **"\tORDERDATE TEXT,\n"** +  
 **"\tCOMPLITIONDATE TEXT,\n"** +  
 **"\tTOTALCOST NUM,\n"** +  
 **"\tDETAILSNUMBER INTEGER,\n"** +  
 **"\tDateLastChange integer,\n"** +  
 **"\tTimeLastChange integer,\n"** +  
 **"\tforeign key(IDCUSTOMER) references CUSTOMER(ID) on update cascade on delete cascade\n"** +  
 **");"**);  
db.execSQL(**"CREATE TABLE if not exists ORDEREDDETAIL(\n"** +  
 **"\tID INTEGER PRIMARY KEY,\n"** +  
 **"\tIDORDER INTEGER,\n"** +  
 **"\tIDDETAIL INTEGER,\n"** +  
 **"\tSTATUS TEXT CHECK(STATUS IN (\"NOT\_IN\_STACK\", \"ARRIVED\", \"ORDERED\", \"DENIED\")) DEFAULT \"ORDERED\",\n"** +  
 **"\tDateLastChange integer,\n"** +  
 **"\tTimeLastChange integer,\n"** +  
 **"\tforeign key(IDORDER) references ORDERS(ID) on update cascade on delete cascade,\n"** +  
 **"\tforeign key(IDDETAIL) references DETAILDESCR(ID) on update cascade on delete cascade\n"** +  
 **");"**);

**ПРИЛОЖЕНИЕ Б**

Код создания класса FingerprintFragment

public class FingerprintFragment extends Fragment {

BottomNavigationView bottomNavigationView;

View view;

FragmentManager fm;

Fragment fragment;

@Nullable

@Override

public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {

view = inflater.inflate(R.layout.fragment\_fingerprint, container, false);

Button btnLogin;

TextView txtMsg;

fm = getActivity().getSupportFragmentManager();

fragment = this;

Context ctx = getContext();

btnLogin = view.findViewById(R.id.btnLogin);

txtMsg = view.findViewById(R.id.txt\_msg);

final BiometricManager biometricManager = BiometricManager.from(ctx);

switch (biometricManager.canAuthenticate()){

case BiometricManager.BIOMETRIC\_SUCCESS:

Toast.makeText(ctx,"You can use your fingerprint to login",Toast.LENGTH\_LONG).show();

txtMsg.setText("Use Fingerprint to Access");

break;

case BiometricManager.BIOMETRIC\_ERROR\_NO\_HARDWARE:

Toast.makeText(ctx,"No fingerprint sensor",Toast.LENGTH\_LONG).show();

txtMsg.setText("No fingerprint sensor");

btnLogin.setVisibility(View.INVISIBLE);

break;

case BiometricManager.BIOMETRIC\_ERROR\_HW\_UNAVAILABLE:

Toast.makeText(ctx,"Biometric sensor is not available",Toast.LENGTH\_LONG).show();

txtMsg.setText("Biometric sensor is not available");

btnLogin.setVisibility(View.INVISIBLE);

break;

case BiometricManager.BIOMETRIC\_ERROR\_NONE\_ENROLLED:

Toast.makeText(ctx,"Your device don't have any fingerprint, check your security setting",Toast.LENGTH\_LONG).show();

txtMsg.setText("Your device don't have any fingerprint, check your security setting");

btnLogin.setVisibility(View.INVISIBLE);

break;

}

Executor executor = ContextCompat.getMainExecutor(ctx);

final BiometricPrompt biometricPrompt = new BiometricPrompt((FragmentActivity) ctx, executor, new BiometricPrompt.AuthenticationCallback(){

@Override

public void onAuthenticationError(int errorCode, @NonNull CharSequence errString) {

super.onAuthenticationError(errorCode, errString);

}

@Override

public void onAuthenticationSucceeded(@NonNull BiometricPrompt.AuthenticationResult result) {

super.onAuthenticationSucceeded(result);

Toast.makeText(ctx,"Login Success",Toast.LENGTH\_LONG).show();

fm.beginTransaction().remove(fragment).commit();

bottomNavigationView = (BottomNavigationView) getActivity().findViewById(R.id.bottom\_navigation);

bottomNavigationView.setVisibility(View.VISIBLE);

}

@Override

public void onAuthenticationFailed() {

super.onAuthenticationFailed();

}

});

final BiometricPrompt.PromptInfo promptInfo = new BiometricPrompt.PromptInfo.Builder()

.setTitle("Login")

.setDescription("User fingerprint to login")

.setNegativeButtonText("cancel")

.build();

btnLogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

biometricPrompt.authenticate(promptInfo);

}

});

return view;

}

}

# **ПРИЛОЖЕНИЕ В**

Диаграмма классов мобильного приложения

