

**Курсовой проект**

ПМ 01 «Разработка модулей программного обеспечения для компьютерных систем»

МДК 01.01 «Разработка программных модулей»

Специальность 09.02.07 «Информационные системы и программирование»

Тема: «Разработка мобильного приложения по продаже товаров и услуг DP Stuff Provider»

**Пояснительная записка**

Листов: 29

Руководитель \_\_\_\_\_ / А.А. Шимбирёв  
«\_\_\_» \_\_\_\_\_ 2021 г.  
Исполнитель \_\_\_\_\_ / А.М. Суслин  
«\_\_\_» \_\_\_\_\_ 2021 г.

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
«Российский экономический университет имени Г.В. Плеханова»  
**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

УТВЕРЖДАЮ

Заместитель директора по учебной  
работе

\_\_\_\_\_ Д.А. Клопов  
« \_\_\_\_ » \_\_\_\_\_ 2021 г.

**ЗАДАНИЕ**

на выполнение курсового проекта (курсовой работы)

Суслина Александра Михайловича

(фамилия, имя, отчество студента — полностью)

студенту группы П50-2-18 специальности 09.02.07 «Информационные системы и программирование»  
по МДК 01.01 «Разработка программных модулей»

1. Исходные данные к проекту (работе):
  - 1.1. Тема: «Разработка мобильного приложения подбора и информирования мероприятий в городе».
  - 1.2. Состав курсового проекта:
    - 1.2.1. Задание на выполнение курсового проекта
    - 1.2.2. Пояснительная записка
    - 1.2.3. Программный продукт (Инсталляционный пакет) на электронном носителе
    - 1.2.4. Программный продукт (Исходный проект) на электронном носителе
    - 1.2.5. Презентация на электронном носителе
2. Содержание задания по проекту (работе) — перечень вопросов, подлежащих разработке

	Разрабатываемый вопрос	Объем от всего задания, %	Срок выполнения
<b>А</b>	Описательная часть проекта (введение, общее описание и т. д.)	5%	
1.	Введение	3	29.01.21
2.	Цель разработки	1	29.01.21
3.	Средства разработки	1	29.01.21
<b>Б</b>	Анализ задачи и её постановка	10%	
1.	Определение требований к программе	2	05.02.21
2.	Спецификация программы (описание задачи, описание входных и выходных данных, метод)	3	05.02.21
3.	Тесты, контроль целостности данных	5	05.02.21
<b>В</b>	Проектирование и реализация	50%	
1.	Схемы проекта (диаграмма классов, функциональная схема, структурная схема, схема пользовательского интерфейса)	15	12.03.21
2.	Реализация в инструментальной среде	35	12.03.21
<b>Г</b>	Технологическая часть проекта	10%	
1.	Инструментальные средства разработки	3	23.04.21
2.	Отладка программа	2	23.04.21
3.	Защитное программирование	3	23.04.21
4.	Характеристика программы	2	23.04.21
<b>Д</b>	Программная документация	20%	
1.	Приложение А. Эскизный проект	5	30.04.21

	Разрабатываемый вопрос	Объем от всего задания, %	Срок выполнения
2.	Приложение Б. Скрипт базы данных	5	07.05.21
3.	Приложение В. Текст программы	5	14.05.21
4.	Приложение Г. Руководство пользователя	5	21.05.21
Е	Экспериментальная часть проекта	5%	
1.	Электронный носитель: исходный проект, эксплуатационный пакет, презентация, документация.	5	28.05.21

Руководитель курсового проекта (работы) Шимбирёв Андрей Андреевич, преподаватель

«12» января 2021 года \_\_\_\_\_ / А.А. Шимбирёв /

Дата выдачи курсового задания «12» января 2021 года

Срок сдачи законченного проекта (работы) «21» июня 2021 года

Задание принял к исполнению

«12» января 2021 года \_\_\_\_\_ / А.М. Суслин/

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1. ОБЩАЯ ЧАСТЬ .....	7
1.1. Цель разработки .....	7
1.2. Средства разработки.....	7
2. СПЕЦИАЛЬНАЯ ЧАСТЬ.....	9
2.1. Постановка задачи. ....	9
2.1.1. Входные и выходные данные. ....	9
2.1.2. Подробные требования к проекту. ....	9
2.2. Внешняя спецификация.....	11
2.2.1. Описание задачи. ....	11
2.2.2. Входные и выходные данные. ....	11
2.2.3. Методы.....	12
2.2.4. Тесты. ....	12
2.2.5. Контроль целостности данных. ....	12
2.3. Проектирование .....	14
2.3.1. Схема архитектуры программы.....	14
2.3.2. Структурная схема программы .....	14
2.3.3. Функциональная схема.....	14
2.3.4. Диаграмма классов .....	15
2.3.5. Модель базы данных .....	16
2.4. Результаты работы программы.....	17
3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ.....	26
3.1. Инструментальные средства разработки.....	26
3.2. Отладка программы .....	26
3.3. Защитное программирование .....	27
3.4. Характеристики программы .....	27
ЗАКЛЮЧЕНИЕ .....	28
СПИСОК ИСПОЛЬЗОВАННЫХ МАТЕРИАЛОВ .....	29

ПРИЛОЖЕНИЕ А Эскизный проект.

ПРИЛОЖЕНИЕ Б Скрипт базы данных.

ПРИЛОЖЕНИЕ В Текст программы.

ПРИЛОЖЕНИЕ Г Руководство Пользователя.

## ВВЕДЕНИЕ

Мобильное приложение по продаже товаров и услуг DP Stuff Provider предназначено для упрощения поиска нужного пользователю товара или услуги и оформления заказа, также благодаря мобильному приложению будет ускорен процесс оформления и доставки заказа до клиента.

Актуальностью разработки мобильного приложения по продаже товаров и услуг является постоянно растущий спрос на качественные товары и услуги, на скорость доставки товаров до рук клиента, а также на удобство оформления заказа, не выходя из дома.

Практичностью написания мобильного приложения по продаже товаров и услуг обуславливается актуальностью задачи в написании программ, связанных с облачными хранилищами, процессами получения и работы с данными из них, так и развитии в программировании на мобильных устройствах, которое будет востребовано и актуально в любое время.

Для написания курсовой работы и достижения выше обозначенной цели были изучены и использованы материалы следующих тематик:

- Создание Api на платформе ASP.NET для взаимодействия с базой данных на сервере;
- Создание и администрирование базы данных, с помощью СУБД MS SQL Server;
- Размещение базы данных и веб-приложения на облачном сервисе Azure;
- Разработка мобильного приложения с помощью IDE Android Studio на языке Kotlin;
- Использование библиотеки GoogleMaterials;
- Обращение к Api с помощью библиотеки Retrofit 2 для android.

## 1. ОБЩАЯ ЧАСТЬ

### 1.1. Цель разработки

Автоматизировать бизнес-процесс интернет-магазина по продаже товаров и услуг, а также ускорить процесс доставки заказов до клиентов и обеспечить их удобным интерфейсом для поиска и заказа нужных товаров на мобильной платформе Android.

### 1.2. Средства разработки

В качестве инструментальных средств разработки использовалось ПО, предоставленное в таблице 1.

Таблица 1 - Программные средства

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Система управления базами данных	MS SQL Server 2017	Создание и администрирование базы данных
2	Текстовый редактор	Microsoft Word 2016	Разработка документации, формирование отчетных документов по шаблонам
3	Среда разработки мобильного приложения	Android Studio 4.1.3	Разработка мобильного приложения
4	Среда разработки Api	Visual Studio 2019	Разработка и управление Api

В качестве вычислительной техники использовался стационарный компьютер. Его характеристики представлены в таблице 2.

Таблица 2 - Технические средства разработки

№	Тип оборудования	Наименование оборудования
1	2	3
1	Центральный процессор:	Intel Core i5-3450
2	Количество ядер	4
3	Видеоадаптер:	GeForce GTX 660 (2 ГБ)
4	Системная память	8 ГБ (DDR3)
5	Твердотельный накопитель:	476 ГБ SSD
6	Разрешение экрана	1920x1080
7	Операционная система	Windows 10 Pro

В качестве устройства, на котором тестировалось мобильное приложение, использовался смартфон Vivo, его характеристики представлены в таблице 3.

Таблица 3 - Технические средства разработки

№	Тип оборудования	Наименование оборудования
1	2	3
1	Центральный процессор:	MTK6762R
2	Количество ядер	8
3	Видеоадаптер:	PowerVR GE8320 (650 МГц)
4	Оперативная память	4 ГБ (DDR3)
5	Операционная система	Android 8.1.0



## 2. СПЕЦИАЛЬНАЯ ЧАСТЬ

### 2.1. Постановка задачи.

Необходимо разработать мобильное приложение на платформе Android. Приложение должно предоставлять пользователю возможность поиска нужного товара (по наименованию или категории товара), возможность добавления нужных товаров в корзину (где можно будет менять количество заказываемого товара), возможность авторизации и регистрации, возможность оформления заказа (с указанием адреса, даты, времени и комментарием к заказу), а также возможность просмотра личной информации и отслеживание статуса заказа в личном кабинете. Для взаимодействия мобильного приложения с базой данных, необходимо разработать API, в качестве ответа API должна отправлять json строку.

#### 2.1.1. Входные и выходные данные.

Входными данными программы являются логин и пароль пользователя, адрес доставки товаров и наименование товара для поиска. Выходными данными являются получаемые данные с API.

#### 2.1.2. Подробные требования к проекту.

- Сервер базы данных и API должны располагаться на облачном сервисе Azure;
- Обращение к данным из облачного хранилища должны осуществляться через программный интерфейс (API);
- Все пароли должны хешироваться алгоритмом SHA-256 и храниться в БД исключительно как хеш-строка;
- Залогиненный пользователь, также, как и корзина, должны сохраняться между сессиями работы приложения;
- Необходимо разработать меню навигации с 4-мя пунктами:
  - Главная – здесь будет отображаться список самых популярных (часто заказываемых) товаров;
  - Каталог – здесь будет отображаться список с категориями, по которому можно пройти, соблюдая всю иерархию категорий, а также поле

поиска, воспользовавшись которым можно будет найти товар по наименованию. Поиск по наименованию необходимо разработать так, чтобы при поиске возвращался список товаров, включающих в своё наименование искомую строку. Поиск по категории необходимо разработать так, чтобы при поиске возвращался список товаров, входящих в искомую категорию или если искомая категория является родительской;

- Корзина – здесь будут отображаться товары, которые перед этим в неё добавили. Здесь также можно менять количество заказываемого товара, а также перейти к оформлению заказа, где клиенту нужно указать адрес, дату, удобное время для принятия доставки и комментарий, после чего заказ занесётся в БД, и пользователь сможет посмотреть статус и код заказа в своём профиле;

- Профиль – здесь будет отображаться личная информация клиента (если клиент авторизован), в случае если клиент не авторизован ему нужно предоставить возможность авторизации или регистрации. Также здесь должна быть возможность просмотра списка оформленных заказов, где можно будет посмотреть общую информацию по заказу и код для получения заказа;

- Все пользовательские формы должны проверяться на корректность, и следует показывать всплывающие подсказки, в случаях некорректного ввода;

- В случае ошибок (не отвечает сервер) – пользователь должен получить соответствующее предупреждение.

## 2.2. Внешняя спецификация.

### 2.2.1. Описание задачи.

При запуске программы пользователя встречает главное окно со списком самых популярных заказов, где пользователь может добавить нужные товары в корзину, или воспользоваться меню и перейти в нужный ему пункт. После того как пользователь будет авторизирован, он может оформить новый заказ, после того как наберёт корзину с нужными ему товарами.

### 2.2.2. Входные и выходные данные.

Таблица 3 - входные данные

№	Поле	Тип данных	Ограничение	Формат ввода	Описание
	1	2	3	4	5
1	Почта	Строка	Не меньше 6 символов. Строка должна заканчиваться @mail.ru и ей подобными. Не должна быть пустой	Поле для ввода текста	Пользователь вводит данные своей почты
2	Пароль	Строка	Должен содержать буквы, и цифры. Не должен быть пустым	Поле для ввода текста	Пользователь вводит свой пароль

Таблица 4 - выходные данные

№	Поле	Тип данных	Ограничение	Формат вывода	Описание
	1	2	3	4	5
1	Наименование товара	Строка	Не должно быть пустым	Поле для вывода текста	Пользователь видит данные о наименовании почты
2	Описание товара	Строка	Не должно быть пустым. Должно содержать информацию о конкретном месте	Поле для вывода текста	Пользователь видит данные о местоположении места
3	Адрес главного изображения товара	Строка	Не должно быть пустым	Поле для вывода текста	Пользователь видит описание местоположения

### 2.2.3. Методы.

Для разработки использовалась методология ООП.

Объектно-ориентированное программирование (ООП) — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определённого класса, а классы образуют иерархию наследования.

[\[https://ru.wikipedia.org/wiki/Объектно-ориентированное\\_программирование\]](https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование)

### 2.2.4. Тесты.

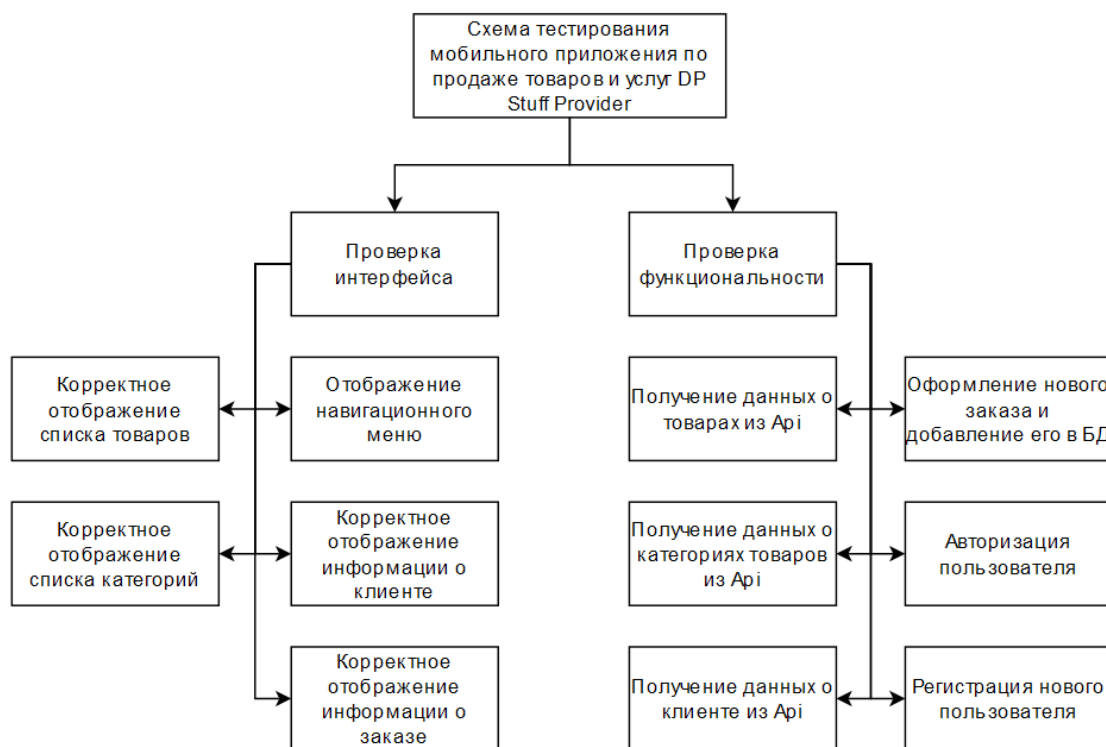


Рисунок 1 – Тесты

### 2.2.5. Контроль целостности данных.

База данных приведена к третьей нормальной форме.

Таблица 5 – контроль целостности данных

№	Ситуация	Аномалия	Реакция программы	Примечание
	1	2	3	4
1	Пользователь нажимает на кнопку для регистрации	Переход не осуществляется	Открывается окно регистрацией	

2	Пользователь нажимает на кнопку «Добавить в корзину»	Товар не добавился в корзину	Товар добавился в корзину	
3	Нажатие на кнопку «Оформить»	Заказ не оформился и не добавился в БД	Вывод всплывающего сообщения об успешном оформлении заказа	

## 2.3. Проектирование

### 2.3.1. Схема архитектуры программы

Схема архитектуры программы: локальная

### 2.3.2. Структурная схема программы

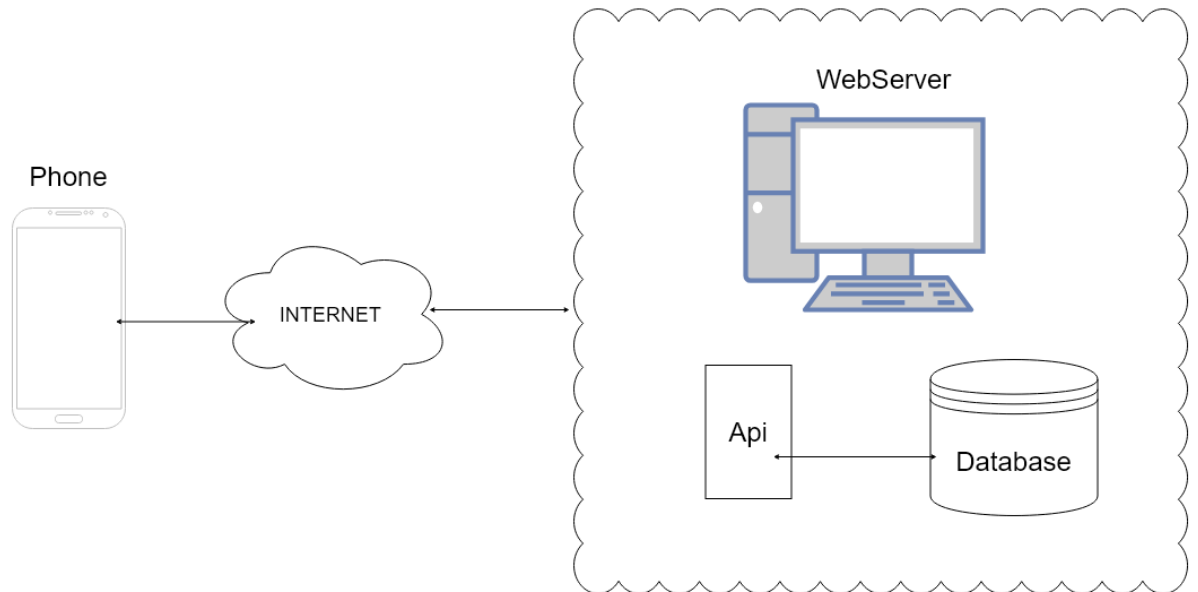


Рисунок 2 - Структурная схема

### 2.3.3. Функциональная схема

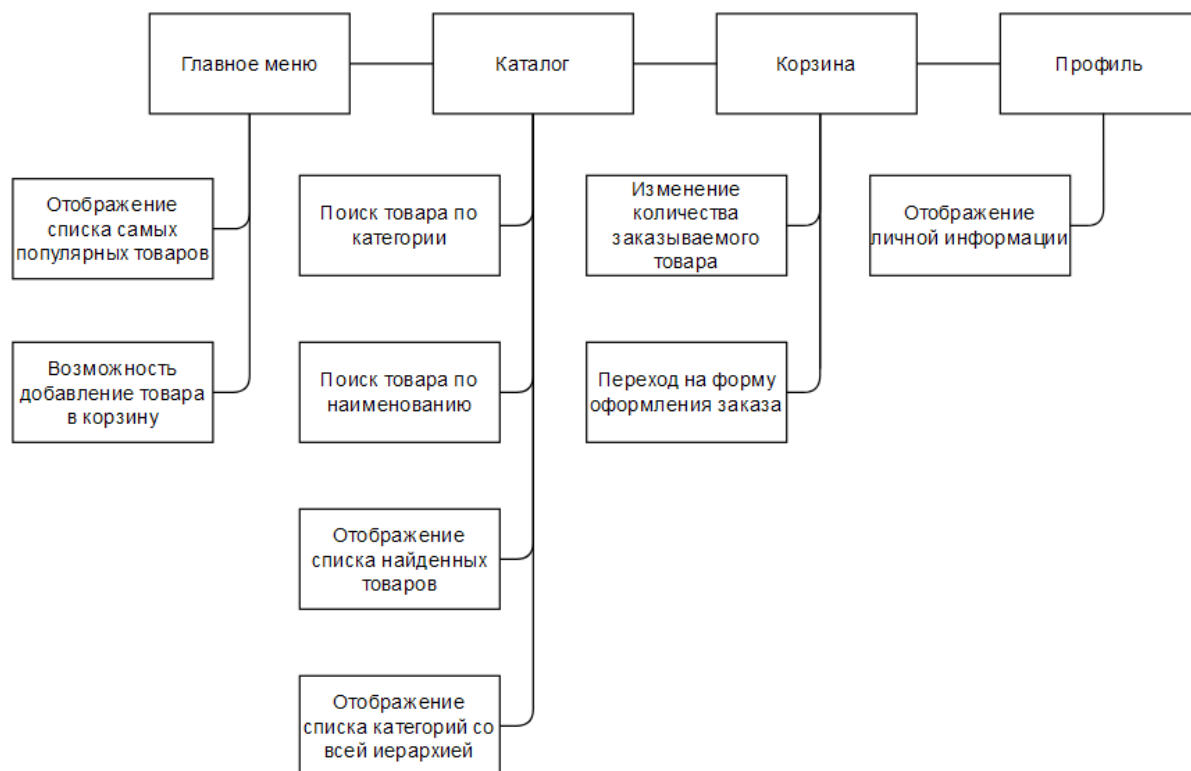


Рисунок 3 - Функциональная схема

## 2.3.4. Диаграмма классов

Таблица 6 - Диаграмма классов

№	Название	Назначение
1	CartAdapter.kt	Адаптер для заполнения корзины
2	CategoryAdapter.kt	Адаптер для заполнения категорий каталога
3	OrderAdapter.kt	Адаптер для заполнения заказов клиента
4	ProductAdapter.kt	Адаптер для заполнения информации о товарах
5	AccountFragment.kt	Фрагмент аккаунта залогиненного пользователя
6	AccountLoginFragment.kt	Фрагмент формы авторизации пользователя
7	AccountNotLoginFragment.kt	Фрагмент не залогиненного пользователя
8	CartEmptyFragment.kt	Фрагмент пустой корзины
9	CartFragment.kt	Фрагмент корзины, корзина заполняется по списку товаров cartList из MainActivity
10	CatalogFragment.kt	Фрагмент каталога с поиском товаров по наименованию и категориям
11	CheckoutFragment.kt	Фрагмент оформления нового заказа
12	ErrorFragment.kt	Фрагмент заглушки на случай ошибок (не отвечает api)
13	HomeFragment.kt	Фрагмент домашней страницы со списком популярных товаров
14	OrdersFragment.kt	Фрагмент со списком заказов пользователя





## 2.4. Результаты работы программы

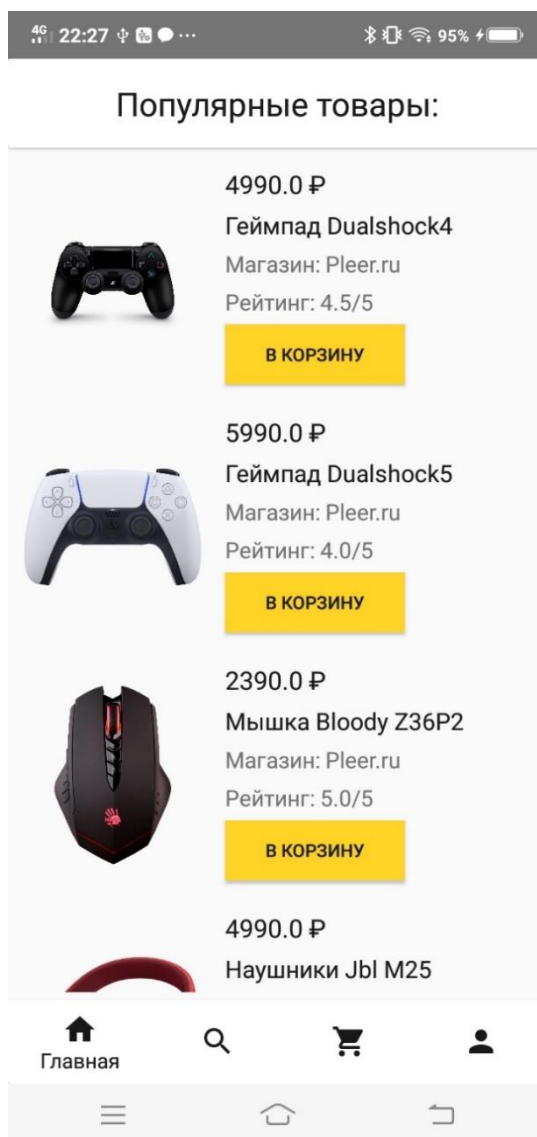


Рисунок 5 – Главная

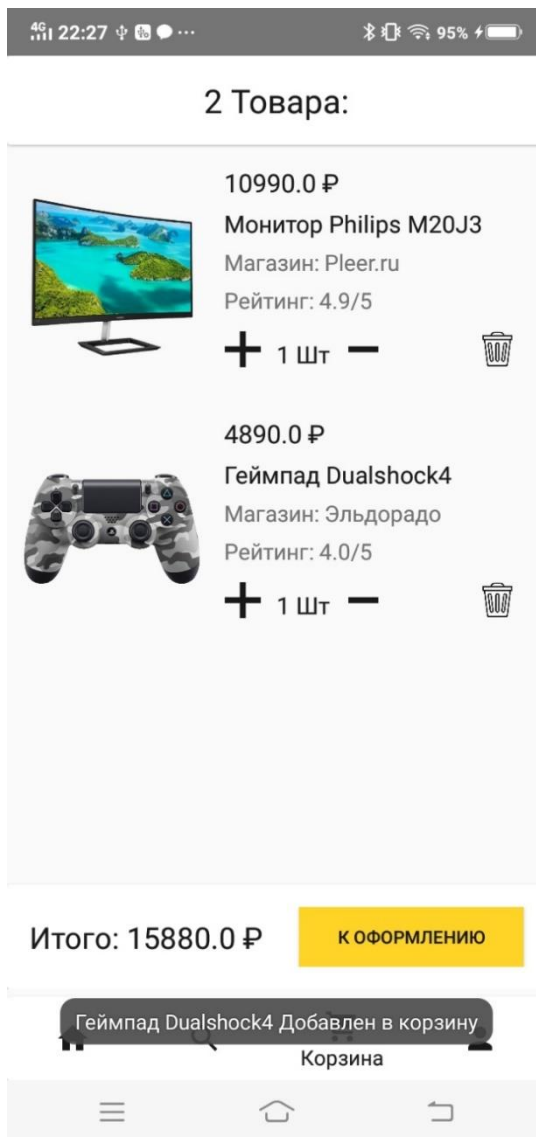


Рисунок 6 – Корзина



Рисунок 7 – Заглушка на случай ошибок

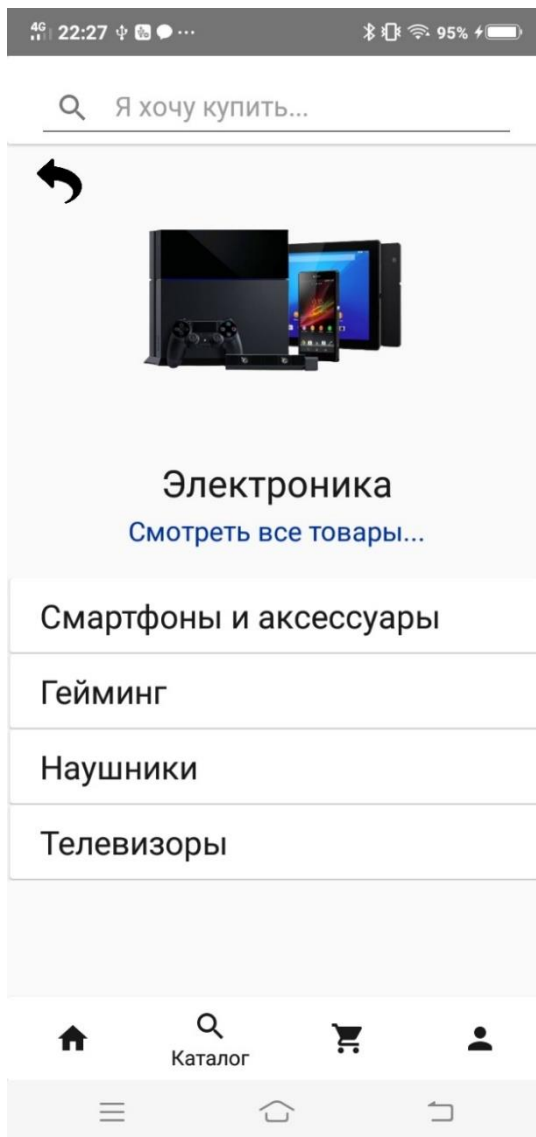


Рисунок 8 - Каталог

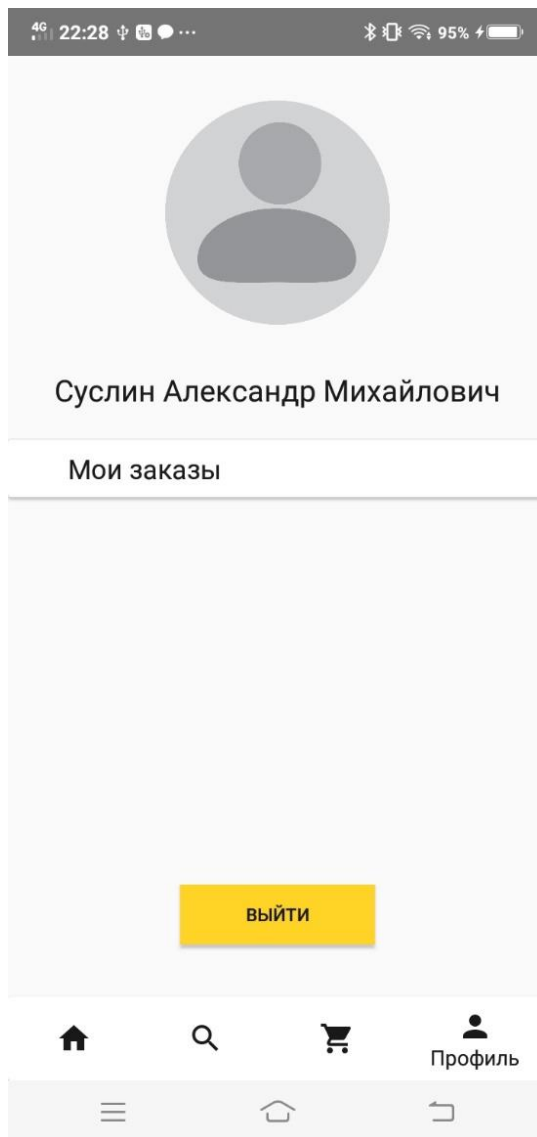


Рисунок 9 – Профиль пользователя

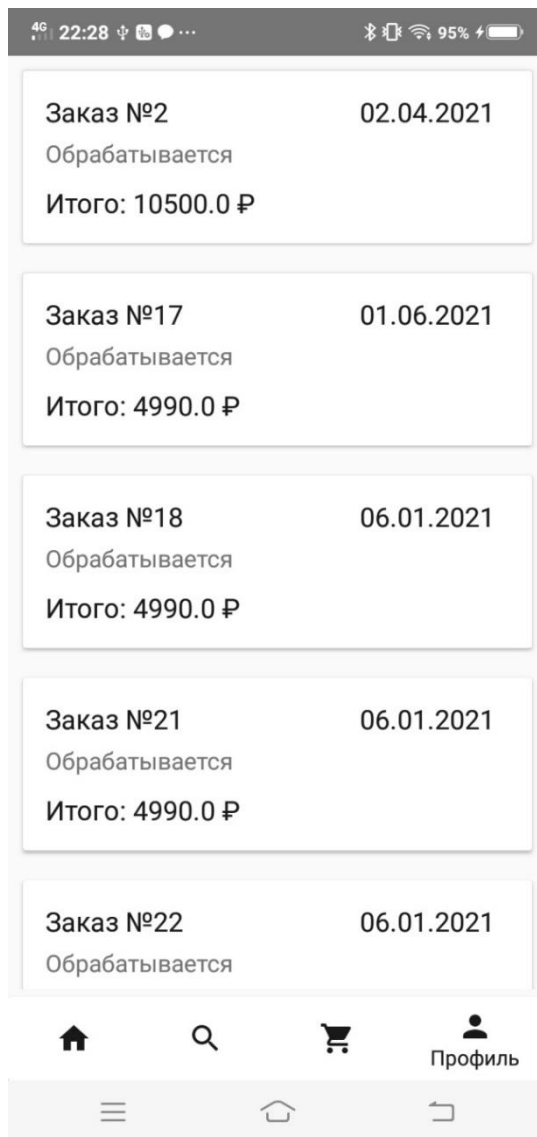


Рисунок 10 – Список заказов пользователя

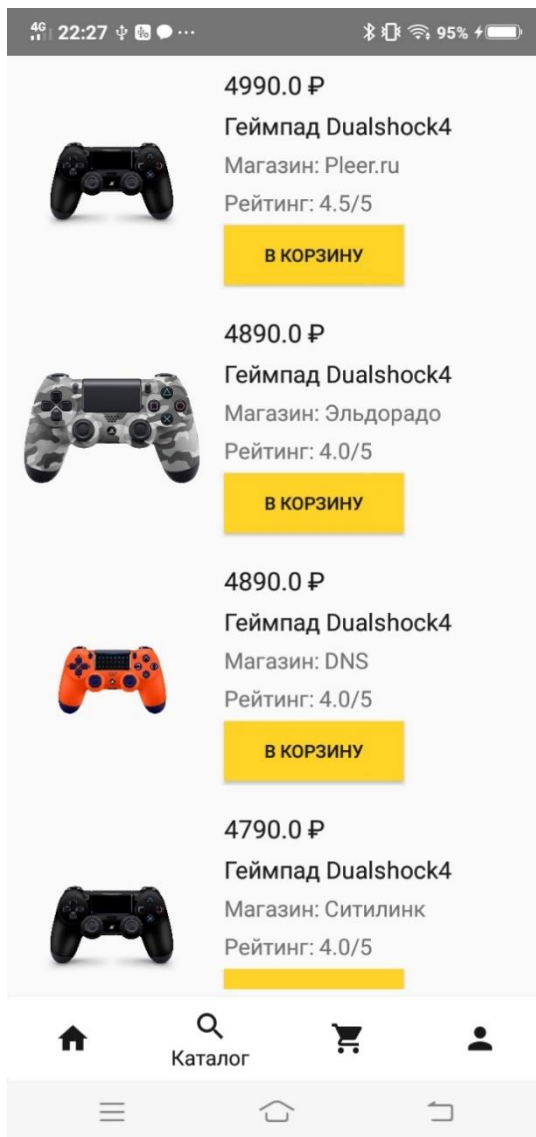


Рисунок 11 – Окно со списком товаров

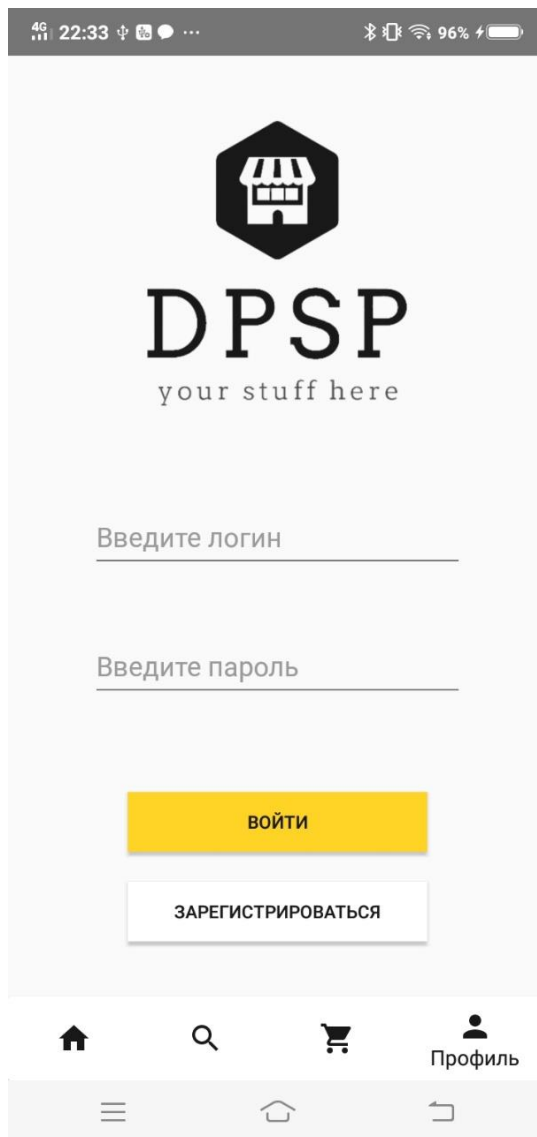


Рисунок 12 - Авторизация



4G 22:33 96%  
Регистрация

Фамилия

Имя

Отчество

Телефон

Почта

Логин

Пароль

Повторите пароль

ЗАРЕГИСТРИРОВАТЬСЯ

🏠 🔍 🛒 👤 Профиль

☰ 🏠 ↩

Рисунок 13 - Регистрация

### 3. ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

#### 3.1. Инструментальные средства разработки

В процессе разработке было решено использовать среду Android Studio, по причине удобства в создании мобильных приложений. Для разработки мобильного приложения был выбран язык Kotlin так, как он удобен и легок в освоении.

Плюсы:

- Универсальность
- Instant Run. Это функция, которой на протяжении всего времени развития Android Studio было уделено довольно много внимания, благодаря чему к выходу версии 3.0 она уже работала в полноценном режиме. Instant Run включена для того, чтобы разработчик приложений для Android после изменения кода мог сразу оценить, как это изменение повлияет на результат — и без дополнительных временных затрат на перекомпиляцию.
- Процесс разработки, который подстраивается под разработчика.
- редактор кода, с которым удобно работать.
- позволяет разрабатывать приложения не только для смартфонов/планшетов, а и для портативных ПК, приставок для телевизоров Android TV, устройств Android Wear, новомодных мобильных устройств с необычным соотношением сторон экрана.
- тестирование корректности работы новых игр, утилит, их производительности на той или иной системе, происходит непосредственно в эмуляторе.

Минусы:

- скудные возможности персонализации проявляются в редакторе кода и общих настройках. Для написания программы, которая будет формировать API запросы и связывать конечного пользователя с базой данных была выбрана среда разработки VisualStudio, язык C#, платформа разработки ASP.NET.

#### 3.2. Отладка программы

В процессе отладки была обнаружена ошибка, из-за которой время на путь не отображалось, это было связано с тем, что заполнение полей было быстрее, чем получение данных с API.

### 3.3. Защитное программирование

Решением проблемы чтения данных с API было использование асинхронного программирования (Coroutine)

```
GlobalScope.launch { this: CoroutineScope  
    response(cr1, cr2)  
}
```

Рисунок 14 - решение проблемы

### 3.4. Характеристики программы

Таблица 7 - Характеристики программы

№	Название	Кол-во строк	Размер в КБ
1	CartAdapter.kt	156	6.61
2	CategoryAdapter.kt	131	6.02
3	OrderAdapter.kt	49	2.18
4	ProductAdapter.kt	101	4.36
5	AccountFragment.kt	44	1.51
6	AccountLoginFragment.kt	53	2.08
7	AccountNotLoginFragment.kt	32	0.953
8	CartEmptyFragment.kt	34	0.993
9	CartFragment.kt	56	2.09
10	CatalogFragment.kt	75	2.65
11	CheckoutFragment.kt	118	4.86
12	ErrorFragment.kt	27	0.786
13	HomeFragment.kt	56	1.91
14	OrdersFragment.kt	44	1.7
15	ProductsFragment.kt	74	2.7
16	RegistrationFragment.kt	87	3.43
17	ClientApiService.kt	280	10.7
18	MainActivity.kt	83	3.04

## ЗАКЛЮЧЕНИЕ

Исходя из приведенных исследований, можно сделать вывод, что в итоге разработки мобильного приложения по продаже товаров и услуг, проблематика системы, в создании базы данных, была упрощена, что позволило сделать ее адаптивной. Конечный программный, продукт является реализуемым и выполняет свои запланированные функции, как приложение по продаже товаров и услуг.

На освоении новой полученной информации по разрабатываемой теме, был получен новый уровень знания в программировании и решении нестандартных проблем, которые требовали гибкого подхода. Тема актуальности в разработке мобильных приложений работы с облачными хранилищами подтвердила себя и закрепилась как основная для изучения в моем арсенале. Материалы, предназначенные для использования в написании мобильного приложения, веб приложения и облачного хранилища были изучены и раскрыты по мере их необходимости.

Основываясь на выше приведенных фактах о тематике разработки приложения и изучении проблемной области, можно сделать вывод, что полученные практический, теоретический опыт и знания, внесут свой вклад в создании будущих проектов.

## СПИСОК ИСПОЛЬЗОВАННЫХ МАТЕРИАЛОВ

1. Head First. Программирование для Android. Гриффитс Д. Питер, 2016
2. Android NDK. Руководство для начинающих. Сильвен Ретабоуил. ДМК Пресс, 2016
3. Mastering Android NDK. Sergey Kosarevsky, Viktor Latypov. Packt Publishing Ltd., 2015
4. Android. Программирование для профессионалов. Билл Филлипс, К. Стюарт, Кристин Марсикано. Питер, 2017
5. Android. Технологии асинхронной обработки данных. Андерс Ёранссон. ДМК Пресс, 2015
6. Asynchronous Android Programming. Packt Publishing Ltd., 2016
7. Android Concurrency. Addison-Wesley Professional, 2016
8. Android Security Internals: An In-Depth Guide to Android's Security Architecture. Nikolay Elenkov. No Starch Press, 2014
9. Android Security Cookbook. Keith Makan, Packt Publishing Ltd., 2013
10. Android Hacker's Handbook. Joshua J. Drake, Zach Lanier, Collin Mulliner, Pau Oliva Fora, Stephen A. Ridley, Georg Wicherski. Wiley, 2014
11. Bulletproof Android: Practical Advice for Building Secure Apps (Developer's Library). Godfrey Nolan. Addison-Wesley Professional, 2014
12. Android User Interface Design: Implementing Material Design for Developers (2nd Edition) (Usability). Ian G. Clifton. Addison-Wesley Professional, 2015
13. Android Design Patterns and Best Practice (1st Edition). Kyle Mew. Packt Publishing Ltd., 2016
14. Embedded Programming with Android: Bringing Up an Android System from Scratch (Android Deep Dive, 1st Edition). Addison-Wesley Professional, 2015
15. Android Application Testing Guide Diego Torres Milano, Packt Publishing Ltd., 2011
16. Inside the Android OS: Building, Customizing, Managing and Operating Android System Services (Android Deep Dive, 1st Edition). G. Blake Meike, Addison-Wesley Professional, 2018
17. Reactive Programming with RxJava. Creating Asynchronous, Event-Based Applications. Ben Christensen, Tomasz Nurkiewicz. -O'Reilly Media, 2016
18. Jonathon Manning, Paris Buttfield-Addison Mobile Game Development with Unity: Build Once, Deploy Anywhere, 2014
19. Mark L. Murphy Busy Coder's Guide to Android Development, 2008
20. Mastering Android NDK. Sergey Kosarevsky, Viktor Latypov. Packt Publishing Ltd., 2015

ПРИЛОЖЕНИЕ А. ЭСКИЗНЫЙ ПРОЕКТ  
АННОТАЦИЯ

В данном программном документе приведен эскизный проект для мобильного приложения подбора и информирования мероприятий в городе.

В данном программной документе, в разделе «Описание» отображены иллюстрации и описание объектов для каждой иллюстрации.

## СОДЕРЖАНИЕ

АННОТАЦИЯ.....	1
1. ОПИСАНИЕ .....	3

## 1. ОПИСАНИЕ

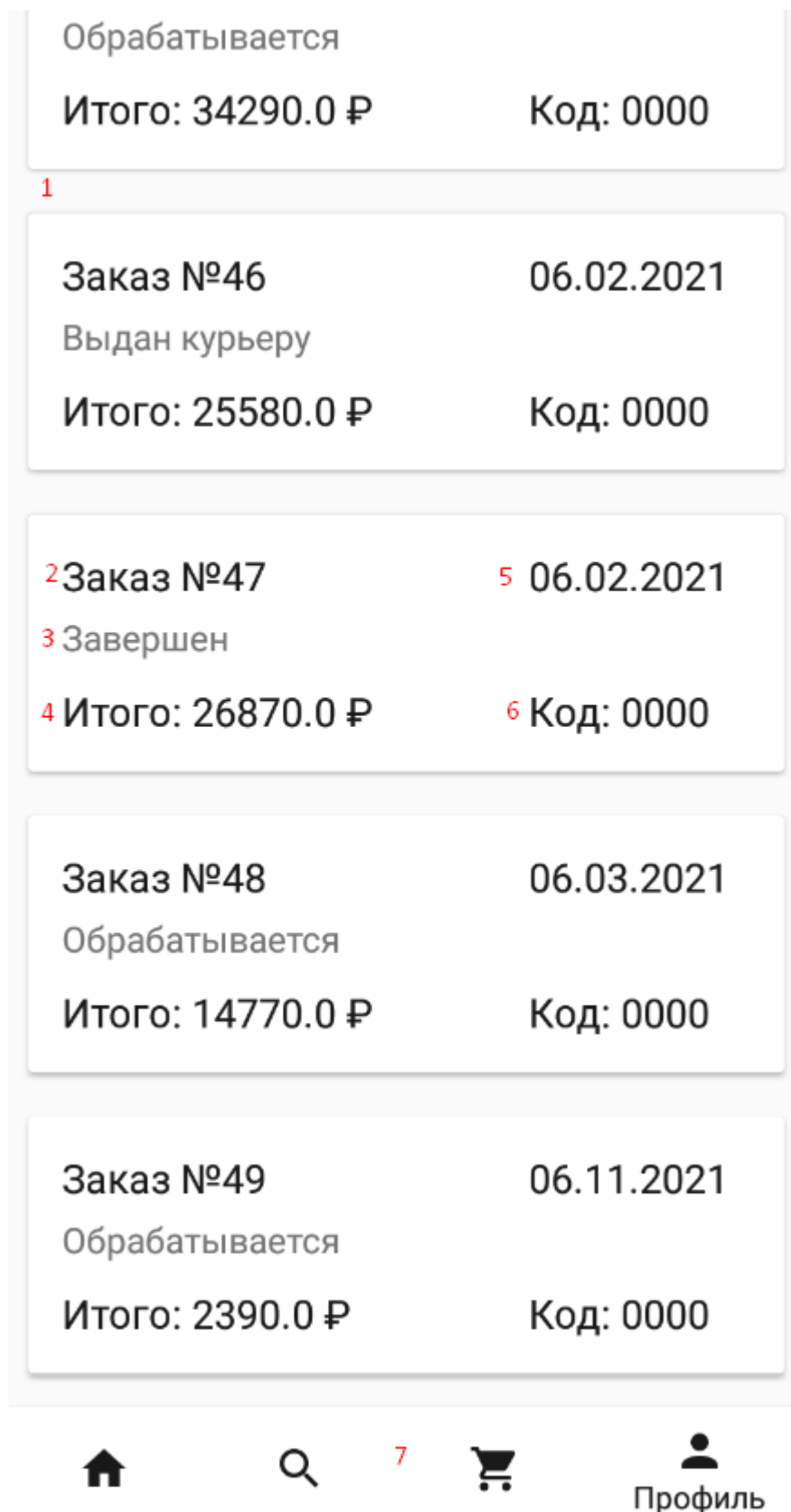


Рисунок 1 – Список заказов клиента

Таблица 1 - Список заказов клиента

№	Тип объекта	Назначение объекта
1	Список объектов	Выводит список заказов клиента
2	Текстовое поле	Подробная информация о заказе (номер)



№	Тип объекта	Назначение объекта
3	Текстовое поле	Подробная информация о заказе (статус)
4	Текстовое поле	Подробная информация о заказе (сумма)
5	Текстовое поле	Подробная информация о заказе (дата)
6	Текстовое поле	Подробная информация о заказе (код)
7	Нижнее меню навигации	Главное меню с переходами между главных окон

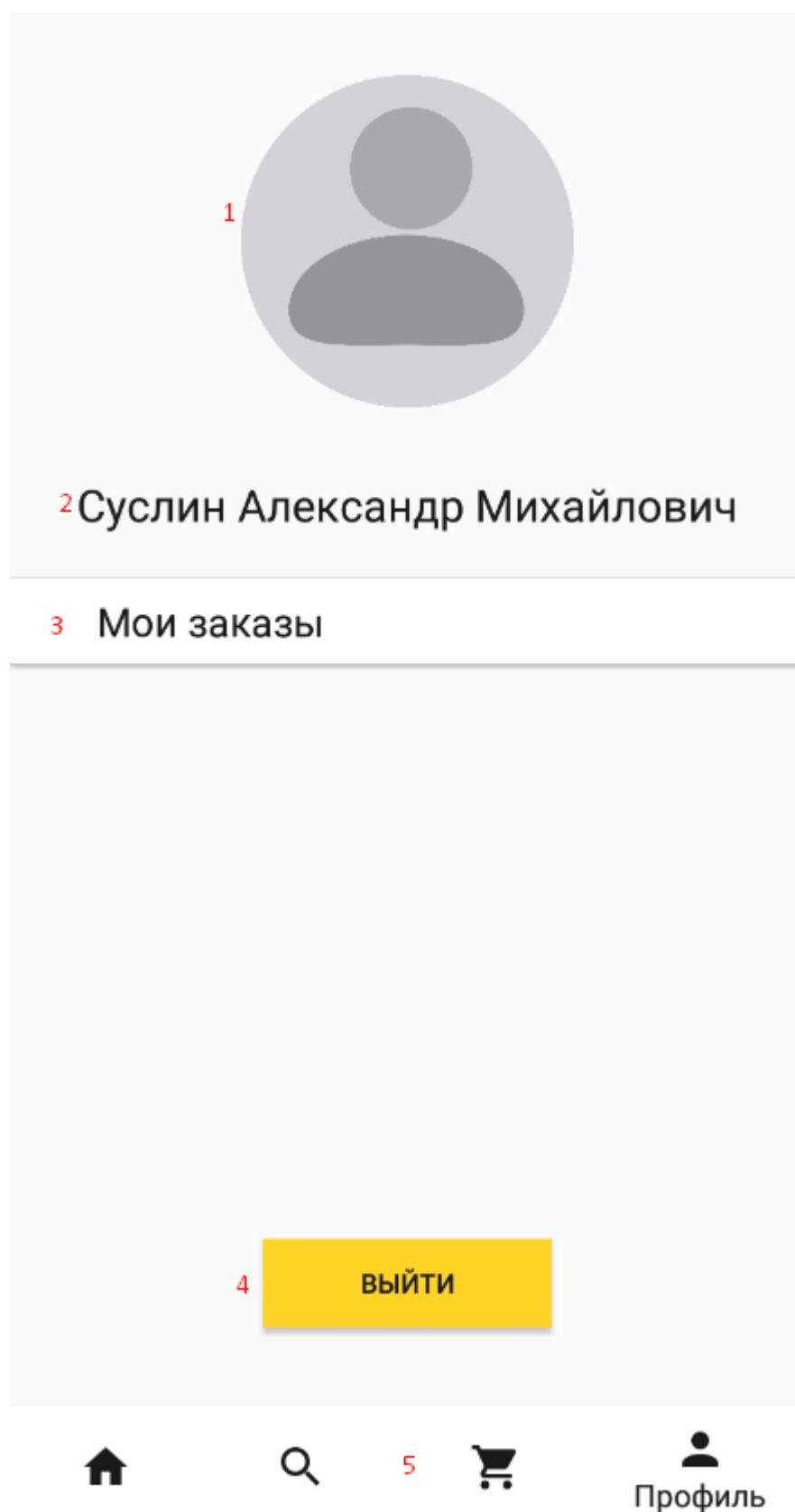


Рисунок 2 – Залогиненный пользователь


Таблица 2 - Залогиненный пользователь


№	Тип объекта	Назначение объекта
1	Картинка	Аватар пользователя
2	Текстовое поле	ФИО пользователя


№	Тип объекта	Назначение объекта
3	Кнопка	Переход ко списку всех заказов
4	Кнопка	Выход из аккаунта
5	Нижнее меню навигации	Главное меню с переходами между главных окон


## 1 2 Товара:

3



1990.0 ₽  
**Мышка Bloody Z36P2**  
4 Магазин: Эльдорадо  
Рейтинг: 3.0/5  
+ 1 шт - 5 

2


10830.0 ₽  
**Монитор Philips M20J3**  
Магазин: Эльдорадо  
Рейтинг: 3.9/5  
+ 1 шт - 

Итого: 12820.0 ₽ 6

К ОФОРМЛЕНИЮ



Корзина



Рисунок 3 – Корзина

Таблица 3 - Корзина

№	Тип объекта	Назначение объекта
1	Текстовое поле	Количество товаров в корзине
2	Список объектов	Выводит список объектов в корзине
3	Изображение	Изображение товара
4	Текстовое поле	Краткая информация о товаре
5	Кнопка	Удаляет товар из корзины
6	Нижнее меню навигации	Главное меню с переходами между главных окон

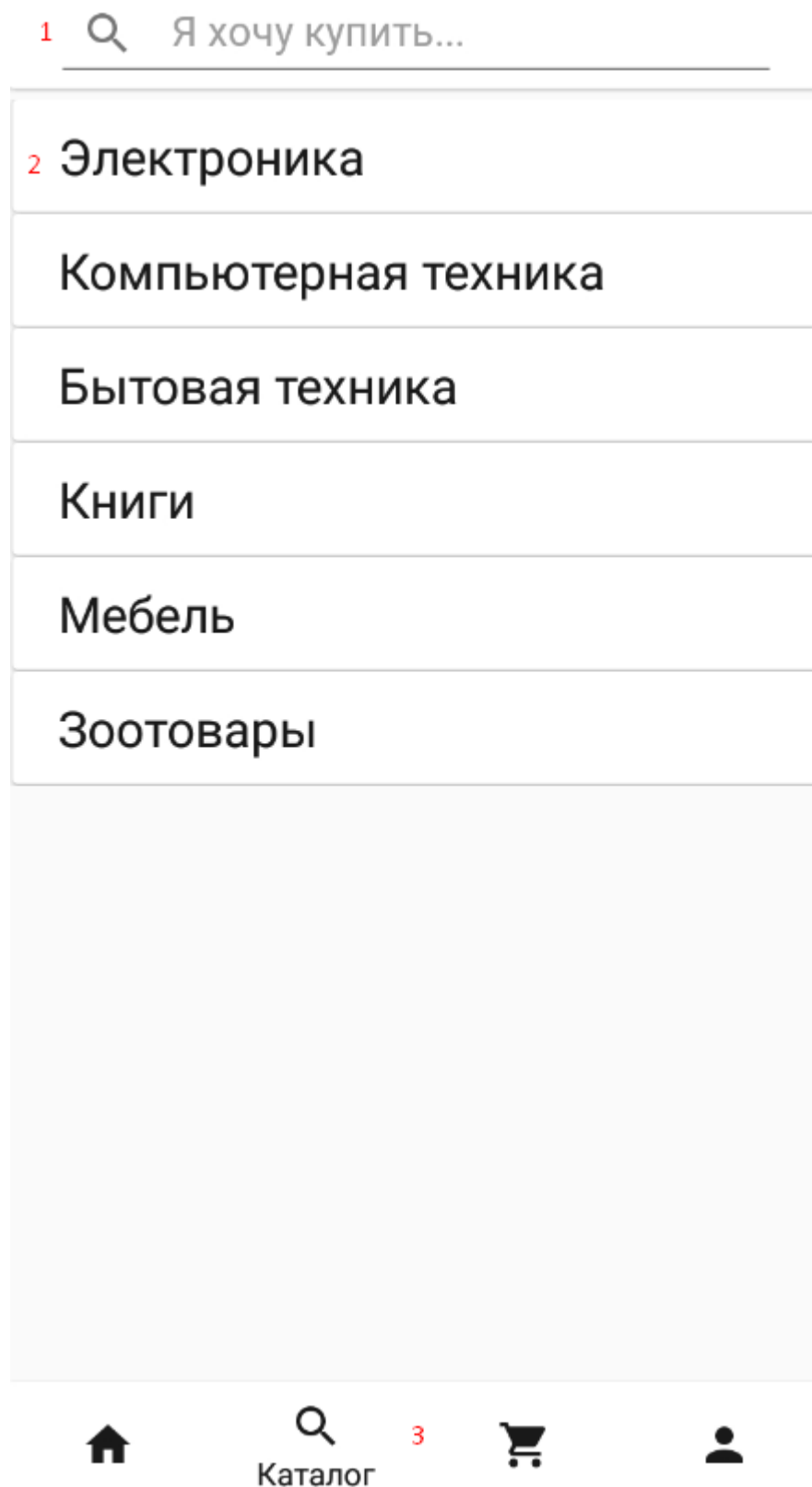


Рисунок 4 – Каталог

Таблица 4 - Каталог

№	Тип объекта	Назначение объекта
1	Поле ввода	Поле ввода для поиска товара
2	Список объектов	Выводит список категорий товаров

№	Тип объекта	Назначение объекта
3	Нижнее меню навигации	Главное меню с переходами между главных окон

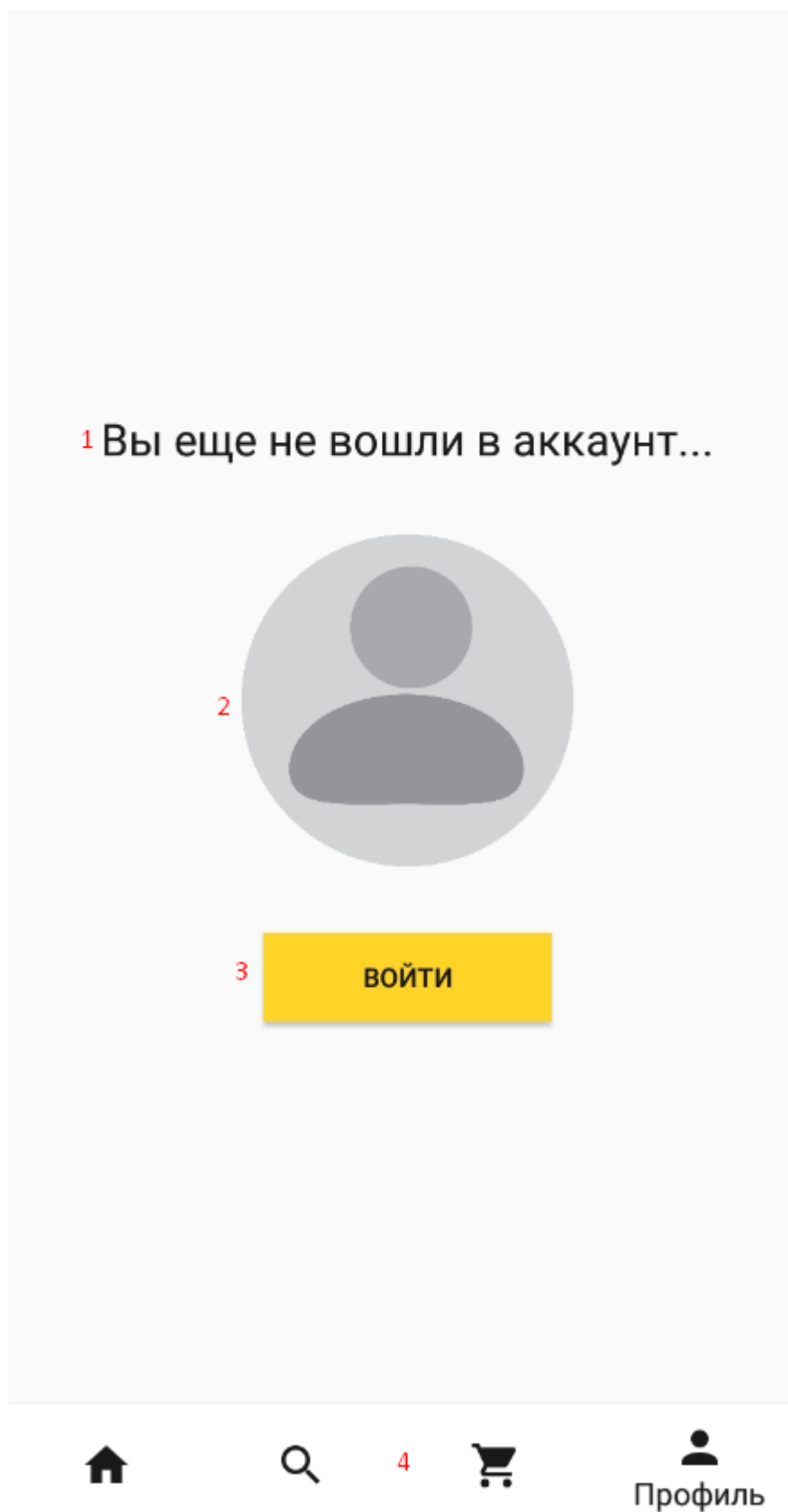


Рисунок 5 – Незалогиненный пользователь

Таблица 5 - Незалогиненный пользователь

№	Тип объекта	Назначение объекта
1	Текстовое поле	Заголовок страницы
2	Изображение	«Аватар» пользователя
3	Кнопка	Вызывает окно авторизации
4	Нижнее меню навигации	Главное меню с переходами между главных окон

1

DPSP  
your stuff here

2 Введите логин

3 Введите пароль

4 ВОЙТИ

5 ЗАРЕГИСТРИРОВАТЬСЯ

Рисунок 6 – Окно авторизации

Таблица 6 - Окно авторизации

№	Тип объекта	Назначение объекта
1	Изображение	Логотип компании
2	Поле ввода	Логин пользователя
3	Поле ввода	Пароль пользователя
4	Кнопка	Авторизует пользователя
5	Кнопка	Вызывает окно регистрации



1 Подтверждение заказа

2 Адрес доставки

Подъезд

Квартира

Этаж

3

Домофон

Дата доставки

4

11

июн.

2021

12

июл.

От

До

5 10:00

6 10:00

Итого: 12820.0 ₽ 7

ОФОРМИТЬ

8

Корзина

Рисунок 7 – Окно оформления

Таблица 7 - Окно оформления

№	Тип объекта	Назначение объекта
1	Текстовое поле	Заголовок окна

№	Тип объекта	Назначение объекта
2	Текстовое поле	Адрес доставки
3	Текстовое поле	Дополнительная информация о адресе доставки
4	Дата-пикер	Выбор даты доставки
5	Спиннер	Выбор удобного времени от
6	Спиннер	Выбор удобного времени до
7	Кнопка	Оформление заказа и занесение его в БД
8	Нижнее меню навигации	Главное меню с переходами между главных окон

## 1 Регистрация

Фамилия

Имя

Отчество

Телефон

Почта

Логин

Пароль

Повторите пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Профиль

Рисунок 8 – Окно регистрации

Таблица 8 - Окно регистрации

№	Тип объекта	Назначение объекта
1	Текстовое поле	Заголовок окна

№	Тип объекта	Назначение объекта
2	Текстовое поле	Данные, нужные для регистрации
3	Кнопка	Регистрирует и заносит в БД нового пользователя
4	Нижнее меню навигации	Главное меню с переходами между главных окон

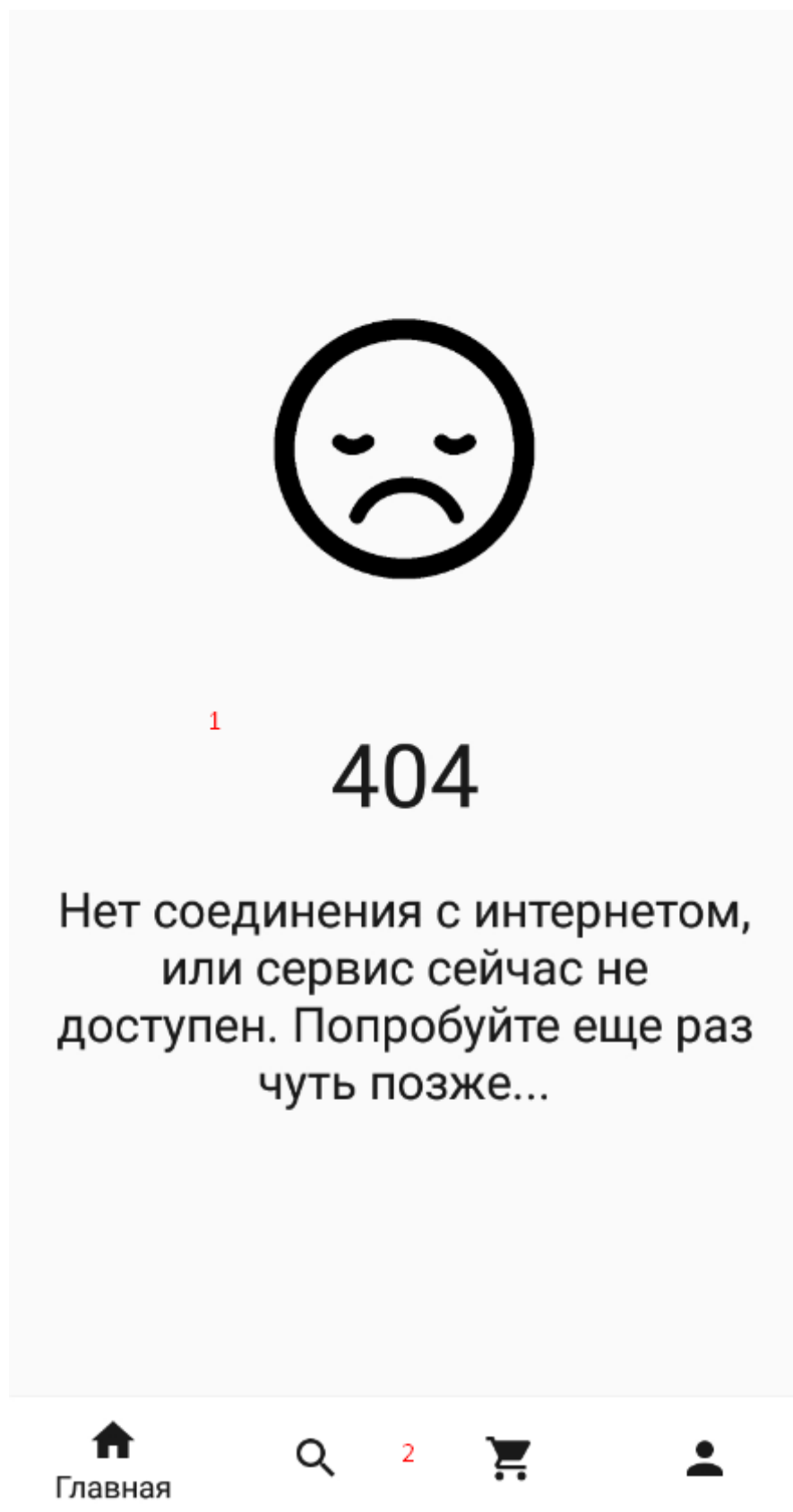


Рисунок 9 – Заглушка на случай ошибки

Таблица 9 - Заглушка на случай ошибки

№	Тип объекта	Назначение объекта
1	Изображение	Лог ошибки
2	Нижнее меню навигации	Главное меню с переходами между главных окон

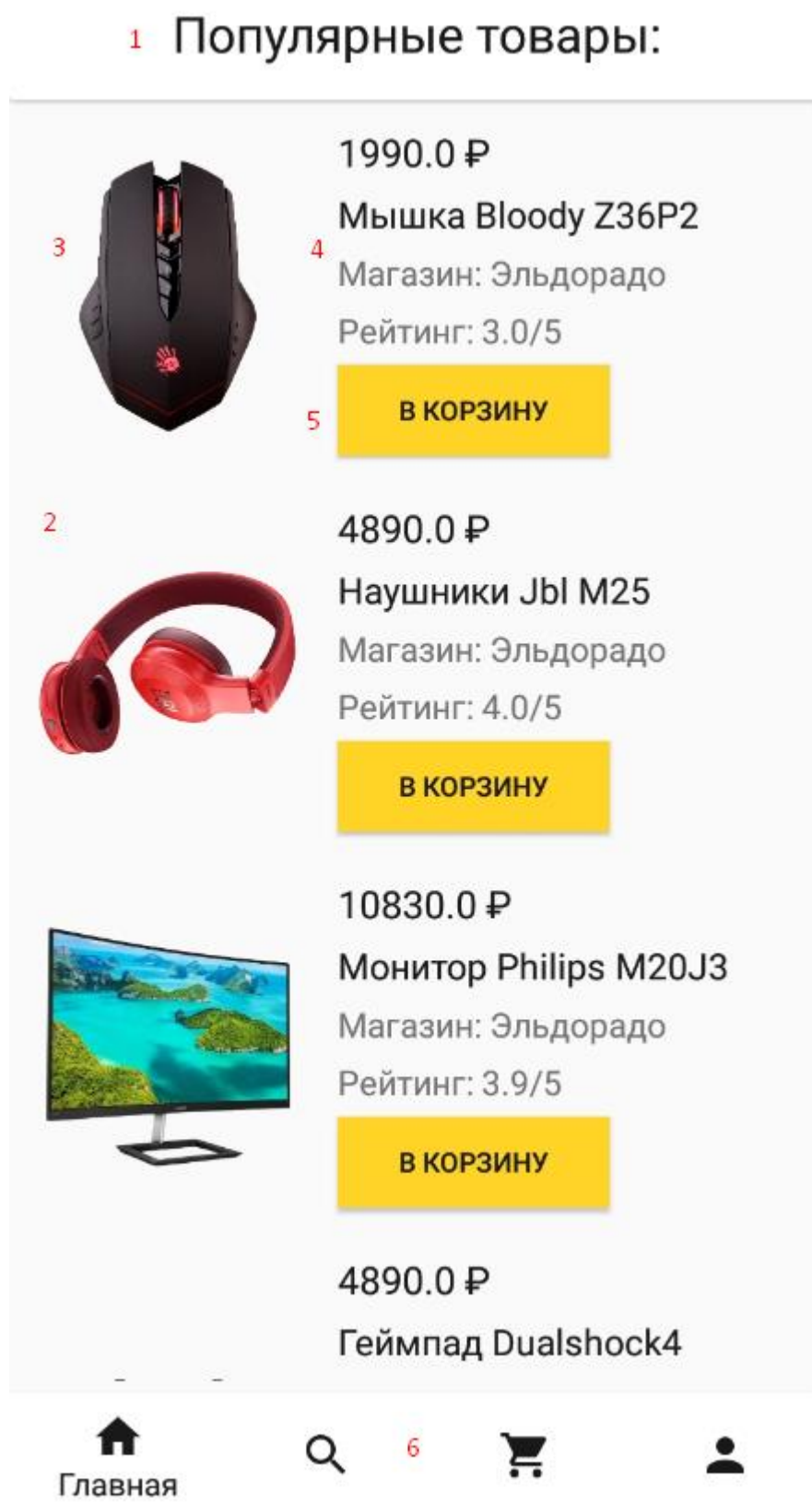


Рисунок 20 – Главная страница

Таблица 20 - Главная страница

№	Тип объекта	Назначение объекта
1	Текстовое поле	Заголовок окна
2	Список объектов	Выводит список самых популярных товаров
3	Изображение	Изображение товара
4	Текстовое поле	Информация о товаре
5	Кнопка	Добавляет товар в корзину
6	Нижнее меню навигации	Главное меню с переходами между главных окон

## ПРИЛОЖЕНИЕ Б.СКРИПТ БАЗЫ ДАННЫХ АННОТАЦИЯ

В данном программном документе приведен скрипт базы данных приложения по продаже товаров и услуг DP Stuff Provider.

В данном программном документе, в разделе «Скрипт базы данных» указан скрипт базы данных и его описание.

## СОДЕРЖАНИЕ

АННОТАЦИЯ.....	1
1. СКРИПТ БАЗЫ ДАННЫХ .....	3
1.1. Наименование скрипта.....	3
1.2. Область применения скрипта .....	3
1.3. Скрипт.....	3



## 1. СКРИПТ БАЗЫ ДАННЫХ

### 1.1. Наименование скрипта

Наименование - «DPSPDBsql».

### 1.2. Область применения скрипта

Скрипт предназначен для создания и заполнения базы данных для мобильного приложения по продаже товаров и услуг.

### 1.3. Скрипт

```
drop database [DPSP Api_db]
create database [DPSP Api_db]
use [DPSP Api_db]

create table PersonalInfo(
    id int primary key identity,
    lastName nvarchar(50) not null,
    firstName nvarchar(50) not null,
    patronymic nvarchar (50) null
);

create table Contacts(
    id int primary key identity,
    phone varchar(20) not null,
    email varchar(50) null
);

create table Client(
    id int primary key identity,
    login varchar(50) not null,
    password varbinary(32) not null,
    idPersonalInfo int not null,
    foreign key (idPersonalInfo) references PersonalInfo (id),
    idContacts int not null,
    foreign key (idContacts) references Contacts (id)
);

create table Courier(
    id int primary key identity,
    login varchar(50) not null,
    password varbinary(32) not null,
    orderQuantity int default 0,
    idPersonalInfo int not null,
    foreign key (idPersonalInfo) references PersonalInfo (id),
    idContacts int not null,
    foreign key (idContacts) references Contacts (id)
);

create table OrderStatus(
    id int primary key identity,
    name nvarchar(50)
);

create table AddressDelivery(
    id int primary key identity,
    address nvarchar(100) not null,
    frontDoor int null,
    apartmentNum int null,
    floorNum int null,
```

```

        intercom nvarchar(20) null
    );

create table Ordered(
    id int primary key identity,
    orderDateTime datetime not null,
    deliveryDate date not null,
    deliveryTimeFrom time not null,
    deliveryTimeTo time not null,
    commentary nvarchar(max) null,
    summ float default 0,
    priority int default 5,
    idAddress int not null,
    foreign key (idAddress) references AddressDelivery (id),
    idClient int not null,
    foreign key (idClient) references Client (id),
    idCourier int null,
    foreign key (idCourier) references Courier (id),
    idOrderStatus int not null,
    foreign key (idOrderStatus) references OrderStatus (id)
);

create table OrderFinished(
    id int primary key identity,
    clientScore int null,
    commentary nvarchar(max) null,
    idOrder int not null,
    foreign key (idOrder) references Ordered (id)
);

create table StoreInfo(
    id int primary key identity,
    name nvarchar(50) not null,
    fullname nvarchar(100) not null,
    tin varchar(12) not null,
    bank nvarchar(100) not null,
    bic varchar(9) not null,
    email varchar(50) not null,
    phone varchar(20) not null,
    address nvarchar(100) not null
);

create table ProductAttribute(
    id int primary key identity,
    name nvarchar(50) not null
);

create table ProductCategory(
    id int primary key identity,
    name nvarchar(50) not null,
    imageUrl nvarchar(300) default '',
    idParentCategory int null,
    foreign key (idParentCategory) references ProductCategory (id)
);
--alter table ProductCategory add imageUrl nvarchar(300) default ''

create table Product(
    id int primary key identity,
    name nvarchar(50) not null,
    cost float not null,
    rating float default 0.0,
    avail bit default 1,
    idCategory int not null,
    foreign key (idCategory) references ProductCategory (id),

```

```

        idStoreInfo int not null,
        foreign key (idStoreInfo) references StoreInfo (id)
    );

create table ProductImages(
    id int primary key identity,
    imageUrl nvarchar(max) not null,
    idProduct int not null,
    foreign key (idProduct) references Product (id)
);

create table ProductDescription(
    id int primary key identity,
    attrValue nvarchar(1000) not null,
    idProduct int not null,
    foreign key (idProduct) references Product (id),
    idProductAttribute int not null,
    foreign key (idProductAttribute) references ProductAttribute(id)
);

create table ProductCompos(
    id int primary key identity,
    quantity int not null,
    summ float not null,
    idProduct int not null,
    foreign key (idProduct) references Product (id),
    idOrder int not null,
    foreign key (idOrder) references Ordered (id)
);

create table ProductReview(
    id int primary key identity,
    clientScore int not null,
    commentary nvarchar(max) null,
    idProduct int not null,
    foreign key (idProduct) references Product (id),
    idClient int not null,
    foreign key (idClient) references Client (id)
);

create table ClientAddress(
    id int primary key identity,
    idClient int not null,
    foreign key (idClient) references Client (id),
    idAddress int not null,
    foreign key (idAddress) references AddressDelivery (id)
);

insert into PersonalInfo(lastName, firstName, patronymic) values
(N'Иванов', N'Иван', N'Иванович'),
(N'Суслин', N'Александр', N'Михайлович'),
(N'Комаров', N'Алексей', N'Олегович'),
(N'Железов', N'Андрей', null)

insert into Contacts(phone, email) values
('88005553535', 'example@mail.com'),
('89992222556', 'mail@example.ru'),
('89526545654', 'gmail@mppt.ru'),
('89255233535', 'examp13@gmail.com')

insert into Courier(login, password, idPersonalInfo, idContacts) values
('log1', hashbytes('SHA2_256', 'pass1'), 1, 1),
('log2', hashbytes('SHA2_256', 'pass2'), 2, 2),

```

```

('log3', hashbytes('SHA2_256', 'pass3'), 3, 3),
('log4', hashbytes('SHA2_256', 'pass4'), 4, 4)

insert into OrderStatus(name) values
(N'Обрабатывается'),
(N'Выдан курьеру'),
(N'Завершен'),
(N'Отменён')

insert into Client(login, password, idPersonalInfo, idContacts) values
('log1', hashbytes('SHA2_256', 'pass1'), 1, 1),
('log2', hashbytes('SHA2_256', 'pass2'), 2, 2),
('log3', hashbytes('SHA2_256', 'pass3'), 3, 3)

insert into AddressDelivery(address, frontDoor, apartmentNum, floorNum, intercom)
values
(N'г. Москва, ул. Центральная, д. 3', 2, 77, 8, N'77к2353'),
(N'г. Москва, ул. Пушкинская, д. 12', 3, 43, 3, N'53к4253'),
(N'г. Подольск, ул. Климовская, д. 36а', 1, 13, 1, N'66к1483')

insert into Ordered(idAddress, orderDateTime, deliveryDate, deliveryTimeFrom,
deliveryTimeTo, summ, idClient, idOrderStatus) values
(1, getdate(), '2021-03-30', '09:00', '22:00', 9500.0, 1, 1),
(2, getdate(), '2021-04-02', '09:00', '21:00', 10500.0, 2, 1),
(3, getdate(), '2021-04-01', '10:00', '20:00', 3200.0, 3, 1)

insert into StoreInfo(name, fullname, tin, bank, bic, email, phone, address) values
(N'Pleer.ru', N'ООО Плеер точка ру', '8003745819', N'Сбербанк', '567435440',
'mail@pleer.ru', '89526545654', N'г. Москва, ул. Серпуховская, д. 19'),
(N'Эльдорадо', N'ООО Эльдорадо', '1287003819', N'Альфа-банк', '900435440',
'mail@el.ru', '89251228456', N'г. Москва, ул. Красных фонарей, д. 123'),
(N'DNS', N'ООО ДНС груп', '8999745819', N'Сбербанк', '567923440', 'mail@dns.ru',
'84300545654', N'г. Москва, ул. Камугная, д. 27'),
(N'Ситилинк', N'ООО Citylink', '8926745819', N'Chease', '110025440', 'mail@cit.ru',
'83423545654', N'г. Москва, ул. Зелёная, д. 3'),
(N'Мвидео', N'ООО Мвидео груп', '8666745819', N'Сбербанк', '137435440',
'mail@mv.ru', '82286545654', N'г. Москва, ул. Меруэмная, д. 6')

--delete from ProductCategory
insert into ProductCategory(name, idParentCategory) values
(N'Электроника', null),
(N'Компьютерная техника', null),
(N'Бытовая техника', null),
(N'Книги', null),
(N'Мебель', null),
(N'Зоотовары', null)

update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/2852074/ac8b053e-59d4-40cb-8c23-6ce9b8cd5bc4/s1200' where name = N'Электроника'
update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/4516377/799f20f8-23ac-4c79-ba98-605ef500a007/s1200' where name = N'Компьютерная
техника'
update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/4571085/553bef76-e3f8-45d0-934a-57cc401502d0/s1200' where name = N'Бытовая техника'
update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/2434617/150b9b90-f1c3-4a07-8d86-4f5e91969560/s1200' where name = N'Книги'
update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/4265498/f6fb9e7d-b7b8-4ad4-9a06-6b4decbe9125/s1200' where name = N'Мебель'
update ProductCategory set imageUrl = N'https://avatars.mds.yandex.net/get-
pdb/4263207/0c82c267-27d8-4325-8b96-ed3d85ed8d6c/s1200' where name = N'Зоотовары'

insert into ProductCategory(name, idParentCategory) values
(N'Смартфоны и аксессуары', (select id from ProductCategory where
name=N'Электроника')),

```

```

(N'Гейминг', (select id from ProductCategory where name=N'Электроника')),
(N'Наушники', (select id from ProductCategory where name=N'Электроника')),
(N'Телевизоры', (select id from ProductCategory where name=N'Электроника')),
(N'Переферийные устройства', (select id from ProductCategory where
name=N'Компьютерная техника')),
(N'Мониторы и аксессуары', (select id from ProductCategory where name=N'Компьютерная
техника')),
(N'Техника для кухни', (select id from ProductCategory where name=N'Бытовая
техника')),
(N'Техника для дома', (select id from ProductCategory where name=N'Бытовая
техника')),
(N'Умный дом', (select id from ProductCategory where name=N'Бытовая техника')),
(N'Комиксы и манга', (select id from ProductCategory where name=N'Книги')),
(N'Художественная литература', (select id from ProductCategory where
name=N'Книги')),
(N'Психология и саморазвитие', (select id from ProductCategory where
name=N'Книги')),
(N'Столы и стулья', (select id from ProductCategory where name=N'Мебель')),
(N'Мебель для спальни', (select id from ProductCategory where name=N'Мебель')),
(N'Мебель для кухни', (select id from ProductCategory where name=N'Мебель')),
(N'Для кошек', (select id from ProductCategory where name=N'Зоотовары')),
(N'Для собак', (select id from ProductCategory where name=N'Зоотовары')),
(N'Для птиц', (select id from ProductCategory where name=N'Зоотовары'))
insert into ProductCategory(name, idParentCategory) values
(N'Playstation 4', (select id from ProductCategory where name=N'Гейминг')),
(N'Playstation 5', (select id from ProductCategory where name=N'Гейминг')),
(N'Мышки', (select id from ProductCategory where name=N'Переферийные устройства')),
(N'Мониторы', (select id from ProductCategory where name=N'Мониторы и аксессуары'))

--delete from Product
insert into Product(name, cost, idStoreInfo, idCategory) values
(N'Геймпад Dualshock4', 4990.0, 1, (select id from ProductCategory where
name=N'Playstation 4')),
(N'Геймпад Dualshock5', 5990.0, 1, (select id from ProductCategory where
name=N'Playstation 5')),
(N'Мышка Bloody Z36P2', 2390.0, 1, (select id from ProductCategory where
name=N'Мышки')),
(N'Наушники Jbl M25', 4990.0, 1, (select id from ProductCategory where
name=N'Наушники')),
(N'Монитор Philips M20J3', 10990.0, 1, (select id from ProductCategory where
name=N'Мониторы')),
(N'Геймпад Dualshock4', 4890.0, 2, (select id from ProductCategory where
name=N'Playstation 4')),
(N'Геймпад Dualshock5', 6390.0, 2, (select id from ProductCategory where
name=N'Playstation 5')),
(N'Мышка Bloody Z36P2', 1990.0, 2, (select id from ProductCategory where
name=N'Мышки')),
(N'Наушники Jbl M25', 4890.0, 2, (select id from ProductCategory where
name=N'Наушники')),
(N'Монитор Philips M20J3', 10830.0, 2, (select id from ProductCategory where
name=N'Мониторы')),
(N'Геймпад Dualshock4', 4890.0, 3, (select id from ProductCategory where
name=N'Playstation 4')),
(N'Геймпад Dualshock5', 5890.0, 3, (select id from ProductCategory where
name=N'Playstation 5')),
(N'Мышка Bloody Z36P2', 2290.0, 3, (select id from ProductCategory where
name=N'Мышки')),
(N'Наушники Jbl M25', 4990.0, 3, (select id from ProductCategory where
name=N'Наушники')),
(N'Монитор Philips M20J3', 11390.0, 3, (select id from ProductCategory where
name=N'Мониторы')),
(N'Геймпад Dualshock4', 4790.0, 4, (select id from ProductCategory where
name=N'Playstation 4')),

```

```

        (N'Геймпад Dualshock5', 5900.0, 4, (select id from ProductCategory where
name=N'Playstation 5')),
        (N'Мышка Bloody Z36P2', 2300.0, 4, (select id from ProductCategory where
name=N'Мышки')),
        (N'Наушники Jbl M25', 4900.0, 4, (select id from ProductCategory where
name=N'Наушники')),
        (N'Монитор Philips M20J3', 10900.0, 4, (select id from ProductCategory where
name=N'Мониторы')),
        (N'Геймпад Dualshock4', 4900.0, 5, (select id from ProductCategory where
name=N'Playstation 4')),
        (N'Геймпад Dualshock5', 5900.0, 5, (select id from ProductCategory where
name=N'Playstation 5')),
        (N'Мышка Bloody Z36P2', 2380.0, 5, (select id from ProductCategory where
name=N'Мышки')),
        (N'Наушники Jbl M25', 4900.0, 5, (select id from ProductCategory where
name=N'Наушники')),
        (N'Монитор Philips M20J3', 10900.0, 5, (select id from ProductCategory where
name=N'Мониторы'))

```

```

insert into ProductCompos(idOrder, idProduct, quantity, summ) values
(1, (select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=1), 1,
(select cost from Product where name=N'Геймпад Dualshock4' and idStoreInfo=1)*1),
(1, (select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=2), 2,
(select cost from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=2)*2),
(2, (select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=3), 1,
(select cost from Product where name=N'Наушники Jbl M25' and idStoreInfo=3)*1),
(2, (select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=3),
1, (select cost from Product where name=N'Монитор Philips M20J3' and idStoreInfo=3)*1),
(3, (select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=1), 1,
(select cost from Product where name=N'Геймпад Dualshock4' and idStoreInfo=1)*1),
(3, (select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=4), 1,
(select cost from Product where name=N'Геймпад Dualshock5' and idStoreInfo=4)*2)

```

```

insert into ProductImages(idProduct, imageUrl) values
((select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=1),
N'https://www.ucustom.nl/wp-content/uploads/2019/01/PS4-Controller-Skin-Zwart-1.png'),
((select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=2),
N'https://images.wbstatic.net/big/new/20820000/20822789-1.jpg'),
((select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=3),
N'https://images-eu.ssl-images-amazon.com/images/I/41GPLTxGE6L.01_SL120_.jpg'),
((select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=4),
N'https://www.ucustom.nl/wp-content/uploads/2019/01/PS4-Controller-Skin-Zwart-1.png'),
((select id from Product where name=N'Геймпад Dualshock4' and idStoreInfo=5),
N'https://images-eu.ssl-images-amazon.com/images/I/41GPLTxGE6L.01_SL120_.jpg')

```

```

insert into ProductImages(idProduct, imageUrl) values
((select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=1),
N'https://e7.pngegg.com/pngimages/465/617/png-clipart-computer-mouse-a4tech-bloody-
gaming-a4-tech-bloody-v7m-a4tech-bloody-v7-computer-mouse-electronics-computer-
keyboard.png'),
((select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=2),
N'https://e7.pngegg.com/pngimages/465/617/png-clipart-computer-mouse-a4tech-bloody-
gaming-a4-tech-bloody-v7m-a4tech-bloody-v7-computer-mouse-electronics-computer-
keyboard.png'),
((select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=3),
N'https://e7.pngegg.com/pngimages/465/617/png-clipart-computer-mouse-a4tech-bloody-
gaming-a4-tech-bloody-v7m-a4tech-bloody-v7-computer-mouse-electronics-computer-
keyboard.png'),
((select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=4),
N'https://e7.pngegg.com/pngimages/465/617/png-clipart-computer-mouse-a4tech-bloody-
gaming-a4-tech-bloody-v7m-a4tech-bloody-v7-computer-mouse-electronics-computer-
keyboard.png'),
((select id from Product where name=N'Мышка Bloody Z36P2' and idStoreInfo=5),
N'https://e7.pngegg.com/pngimages/465/617/png-clipart-computer-mouse-a4tech-bloody-

```

```

gaming-a4-tech-bloody-v7m-a4tech-bloody-v7-computer-mouse-electronics-computer-
keyboard.png')
    insert into ProductImages(idProduct, imageUrl) values
    ((select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=1),
N'https://static.onlinetrade.ru/img/items/b/besprovodnie_naushniki_jbl_e45bt_krasniy_2.jp
g'),
    ((select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=2),
N'https://24.lv/images/detailed/484/JBL_LIVE500BT_Product_Image_Fold_White_17273_x1.png')
,
    ((select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=3),
N'https://img.mvideo.ru/Pdb/50125455b2.jpg'),
    ((select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=4),
N'https://static.onlinetrade.ru/img/items/b/besprovodnie_naushniki_jbl_e45bt_krasniy_2.jp
g'),
    ((select id from Product where name=N'Наушники Jbl M25' and idStoreInfo=5),
N'https://img.mvideo.ru/Pdb/50125455b2.jpg')
    insert into ProductImages(idProduct, imageUrl) values
    ((select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=1),
N'https://www.1sm.ru/upload/iblock/274/00000112446.JPG'),
    ((select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=2),
N'https://www.1sm.ru/upload/iblock/274/00000112446.JPG'),
    ((select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=3),
N'https://www.1sm.ru/upload/iblock/274/00000112446.JPG'),
    ((select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=4),
N'https://www.1sm.ru/upload/iblock/274/00000112446.JPG'),
    ((select id from Product where name=N'Монитор Philips M20J3' and idStoreInfo=5),
N'https://www.1sm.ru/upload/iblock/274/00000112446.JPG')
    insert into ProductImages(idProduct, imageUrl) values
    ((select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=1),
N'https://static-sl.insales.ru/images/products/1/1394/416662898/1.png'),
    ((select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=2),
N'https://static-sl.insales.ru/images/products/1/1394/416662898/1.png'),
    ((select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=3),
N'https://static-sl.insales.ru/images/products/1/1394/416662898/1.png'),
    ((select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=4),
N'https://static-sl.insales.ru/images/products/1/1394/416662898/1.png'),
    ((select id from Product where name=N'Геймпад Dualshock5' and idStoreInfo=5),
N'https://static-sl.insales.ru/images/products/1/1394/416662898/1.png')

select * from ProductImages
select * from Product where name=N'Геймпад Dualshock4'
select * from Courier
select * from Ordered
select
    ProductCategory.name as [category],
    parentCategory.name as [parent],
    ProductCategory.imageUrl as [img]
from ProductCategory
    left join ProductCategory as parentCategory on
ProductCategory.idParentCategory = parentCategory.id

```

ПРИЛОЖЕНИЕ В.ТЕКСТ ПРОГРАММЫ  
АННОТАЦИЯ

В данном программном документе, в разделе «Текст программы» указан текст программы и его описание.



## СОДЕРЖАНИЕ

Приложение В.Текст программы .....	1
АННОТАЦИЯ.....	1
1. ТЕКСТ ПРОГРАММЫ.....	3
1.1. Наименование программы .....	3
1.2. Область применения программы .....	3
1.3. Модули.....	3
1.4. Код программы .....	4

## 1. ТЕКСТ ПРОГРАММЫ

### 1.1. Наименование программы

Наименование - «Мобильное приложения по продаже товаров и услуг DP Stuff Provider».

### 1.2. Область применения программы

Данная программа предназначена для рядового пользователя, с целью поиска и заказа нужных товаров.

### 1.3. Модули

Таблица 1 - Модули программы

№	Модуль	Описание	Кол-во строк	Размер в КБ
1	CartAdapter.kt	Адаптер для заполнения корзины	156	6.61
2	CategoryAdapter.kt	Адаптер для заполнения категорий каталога	131	6.02
3	OrderAdapter.kt	Адаптер для заполнения заказов клиента	49	2.18
4	ProductAdapter.kt	Адаптер для заполнения информации о товарах	101	4.36
5	AccountFragment.kt	Фрагмент аккаунта залогиненного пользователя	44	1.51
6	AccountLoginFragment.kt	Фрагмент формы авторизации пользователя	53	2.08
7	AccountNotLoginFragment.kt	Фрагмент не залогиненного пользователя	32	0.953
8	CartEmptyFragment.kt	Фрагмент пустой корзины	34	0.993
9	CartFragment.kt	Фрагмент корзины, корзина заполняется по списку товаров cartList из MainActivity	56	2.09
10	CatalogFragment.kt	Фрагмент каталога с поиском товаров по наименованию и категориям	75	2.65
11	CheckoutFragment.kt	Фрагмент оформления нового заказа	118	4.86
12	ErrorFragment.kt	Фрагмент заглушки на случай ошибок (не отвечает api)	27	0.786
13	HomeFragment.kt	Фрагмент домашней страницы со списком популярных товаров	56	1.91
14	OrdersFragment.kt	Фрагмент со списком заказов пользователя	44	1.7
15	ProductsFragment.kt	Фрагмент со списков товаров (по наименованию или категории)	74	2.7
16	RegistrationFragment.kt	Фрагмент регистрации нового пользователя	87	3.43
17	ClientApiService.kt	Класс, реализующий все методы Api-интерфейсов	280	10.7

№	Модуль	Описание	Кол-во строк	Размер в КБ
18	MainActivity.kt	Основное активити, включающая в себя меню навигации и все фрагменты	83	3.04

#### 1.4. Код программы

##### 1. CartAdapter.kt

```
package com.example.dpstuffproviderstore.adapter

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.annotation.NonNull
import androidx.cardview.widget.CardView
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.RecyclerView
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.`interface`.ICategory
import com.example.dpstuffproviderstore.`interface`.IProduct
import com.example.dpstuffproviderstore.fragment.CartEmptyFragment
import com.example.dpstuffproviderstore.fragment.CartFragment
import com.example.dpstuffproviderstore.fragment.ErrorFragment
import com.example.dpstuffproviderstore.fragment.ProductsFragment
import com.example.dpstuffproviderstore.models.CategoryData
import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.models.ProductImageData
import com.example.dpstuffproviderstore.other.ClientApiService
import com.google.gson.Gson
import com.squareup.picasso.Picasso
import kotlinx.android.synthetic.main.fragment_cart.view.*
```

```

import kotlinx.android.synthetic.main.fragment_catalog.view.*
import org.w3c.dom.Text
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

/**
 * Адаптер для заполнения корзины
 */

internal class CartAdapter(private var productsList: List<ProductData>, private var fragment:
CartFragment) : RecyclerView.Adapter<CartAdapter.MyViewHolder>() {
    internal class MyViewHolder(view: View) : RecyclerView.ViewHolder(view){
        val productImage: ImageView = view.findViewById(R.id.imageProductCart)
        val productPrice: TextView = view.findViewById(R.id.tvPriceCart)
        val productTitle: TextView = view.findViewById(R.id.tvTitleProductCart)
        val productStore: TextView = view.findViewById(R.id.tvStoreProductCart)
        val productRating: TextView = view.findViewById(R.id.tvRatingProductCart)
        val productQuantity: TextView = view.findViewById(R.id.tvQuantityCart)

        val btnPlus: Button = view.findViewById(R.id.btnPlusCart)
        val btnMinus: Button = view.findViewById(R.id.btnMinusCart)
        val btnTrash: Button = view.findViewById(R.id.btnDeleteCart)
    }

    @NonNull
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_cart, parent, false)
        return MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: CartAdapter.MyViewHolder, position: Int) {
        val mainActivity = fragment.activity as MainActivity

        holder.productPrice.text = "${productsList[position].price} ₸"
        holder.productTitle.text = productsList[position].name
    }
}

```

```

holder.productStore.text = "Магазин: ${productsList[position].store}"
holder.productRating.text = "Рейтинг: ${productsList[position].rating}/5"
holder.productQuantity.text = "${productsList[position].quantity ?: 1} шт"

ClientApiService().getImages(productsList[position].id) {
    if(it != null) {
        Picasso.with(holder.itemView.context)
            .load(it!![0].imageUrl)
            .placeholder(R.drawable.img_placeholder)
            .error(R.drawable.img_placeholder)
            .into(holder.productImage)
    }
    else{
        val fragment = ErrorFragment()
        fragment.statusCode = "404"
        mainActivity.makeCurrentFragment(fragment)
    }
}

holder.btnPlus.setOnClickListener {
    mainActivity.cartList[position].quantity = (productsList[position].quantity ?: 1) + 1
    val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()
    editor.putString("cartList", Gson().toJson(mainActivity.cartList))
    editor.apply()

    holder.productQuantity.text = "${mainActivity.cartList[position].quantity ?: 1} шт"

    var summ : Double = 0.0
    for(i in mainActivity.cartList){
        summ += i.price!! * (i.quantity ?: 1)
    }
    fragment.inflate!!.tvCheckout.text = "Итого: ${summ} P"
    fragment.summ = summ
}

fun removeProduct(product : ProductData){
    mainActivity.cartList.remove(product)
    val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()

```

```

if(mainActivity.cartList.size == 0){
    mainActivity.makeCurrentFragment(CartEmptyFragment())
    mainActivity.cartFragment = CartEmptyFragment()

    editor.putString("cartList", "").apply()
}
else{
    editor.putString("cartList", Gson().toJson(mainActivity.cartList)).apply()

    fragment.inflate!!.recyclerCart.adapter = CartAdapter(mainActivity.cartList, fragment)
    fragment.inflate!!.tvTitle.text = "${mainActivity.cartList.size} Товара:"

    var summ : Double = 0.0
    for(i in mainActivity.cartList){
        summ += i.price!! * (i.quantity ?: 1)
    }
    fragment.inflate!!.tvCheckout.text = "Итого: ${summ} Р"
}

holder.btnMinus.setOnClickListener {
    mainActivity.cartList[position].quantity = (productsList[position].quantity ?: 1) - 1
    if (mainActivity.cartList[position].quantity!! <= 0){
        removeProduct(productsList[position])
    }
    else{
        val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()
        editor.putString("cartList", Gson().toJson(mainActivity.cartList))
        editor.apply()

        holder.productQuantity.text = "${mainActivity.cartList[position].quantity ?: 1} шт"

        var summ : Double = 0.0
        for(i in mainActivity.cartList){
            summ += i.price!! * (i.quantity ?: 1)
        }
        fragment.inflate!!.tvCheckout.text = "Итого: ${summ} Р"
        fragment.summ = summ
    }
}

```

```

    }
}

holder.btnTrash.setOnClickListener {
    removeProduct(productsList[position])
}
}

override fun getItemCount(): Int {
    return productsList.size
}
}

```

## 2. CategoryAdapter.kt

```

package com.example.dpstuffproviderstore.adapter

import android.app.Activity
import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.text.TextUtils.replace
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import android.widget.Toast
import androidx.annotation.NonNull
import androidx.cardview.widget.CardView
import androidx.core.content.ContextCompat
import androidx.fragment.app.Fragment
import androidx.fragment.app.FragmentActivity
import androidx.fragment.app.FragmentManager
import androidx.recyclerview.widget.RecyclerView
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R

```

```

import com.example.dpstuffproviderstore.`interface`.ICategory
import com.example.dpstuffproviderstore.fragment.CatalogFragment
import com.example.dpstuffproviderstore.fragment.ErrorFragment
import com.example.dpstuffproviderstore.fragment.HomeFragment
import com.example.dpstuffproviderstore.fragment.ProductsFragment
import com.example.dpstuffproviderstore.models.CategoryData
import com.example.dpstuffproviderstore.other.ClientApiService
import com.google.android.material.internal.ContextUtils.getActivity
import com.squareup.picasso.Picasso
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.fragment_catalog.*
import kotlinx.android.synthetic.main.fragment_catalog.view.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import kotlinx.android.synthetic.main.activity_main.*

/**
 * Адаптер для заполнения категорий каталога
 */

internal class CategoryAdapter(private var categoryList: List<CategoryData>, private var
fragment: CatalogFragment, private var parentCategoryList: List<CategoryData>?) :
RecyclerView.Adapter<CategoryAdapter.MyViewHolder>(){
    internal class MyViewHolder(view: View) : RecyclerView.ViewHolder(view){
        val title : TextView = view.findViewById(R.id.tvTitleCategory)
        val cardCategory : CardView = view.findViewById(R.id.cardCategory)
    }

    @NonNull
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_category, parent, false)
        return MyViewHolder(itemView)
    }
}

```



```

override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
    holder.title.text = categoryList[position].name

    holder.cardCategory.setOnClickListener {

        ClientApiService().getChildCategories(categoryList[position].name) {

            if(it != null){
                fragment.inflate!!.recyclerCategory.adapter = CategoryAdapter(it, fragment, categoryList)
                fragment.inflate!!.titleCategory.visibility = View.VISIBLE
                fragment.inflate!!.titleCategory.text = it[0].parentName
                fragment.inflate!!.linkToCategory.visibility = View.VISIBLE
                fragment.inflate!!.categoryImg.visibility = View.VISIBLE

                if(!categoryList[position].imageUrl.isNullOrEmpty()){
                    Picasso.with(holder.itemView.context)
                        .load(categoryList[position].imageUrl)
                        .placeholder(R.drawable.img_placeholder)
                        .error(R.drawable.img_placeholder)
                        .into(fragment.inflate!!.categoryImg)
                }
            }
            else{
                fragment.inflate!!.categoryImg.visibility = View.GONE
            }

            fragment.inflate!!.btnUndo.visibility = View.VISIBLE

            fragment.inflate!!.btnUndo.setOnClickListener {
                ClientApiService().getMainCategories() {

                    if(it != null){
                        fragment.inflate!!.recyclerCategory.adapter = CategoryAdapter(it, fragment, null)

                        fragment.inflate!!.titleCategory.visibility = View.GONE
                        fragment.inflate!!.linkToCategory.visibility = View.GONE
                        fragment.inflate!!.categoryImg.visibility = View.GONE
                        fragment.inflate!!.btnUndo.visibility = View.GONE
                    }
                    else{
                        val mainActivity = fragment.activity as MainActivity

```

```

val fragment = ErrorFragment()
fragment.statusCode = "404"
mainActivity.makeCurrentFragment(fragment)
}
}
}

fragment.inflate!!.linkToCategory.setOnClickListener {
val mainActivity = fragment.activity as MainActivity
mainActivity.modeSearch = MainActivity.EnumModeSearch.CATEGORY
mainActivity.productCategory = fragment.inflate!!.titleCategory.text.toString()
mainActivity.makeCurrentFragment(ProductsFragment())
}
}
else{
val mainActivity = fragment.activity as MainActivity
mainActivity.modeSearch = MainActivity.EnumModeSearch.CATEGORY
mainActivity.productCategory = categoryList[position].name
mainActivity.makeCurrentFragment(ProductsFragment())
}
}

/*override fun onFailure(call: Call<List<CategoryData>>, t: Throwable){
val mainActivity = fragment.activity as MainActivity
val fragment = ErrorFragment()
mainActivity.makeCurrentFragment(fragment)
}*/
}
}

override fun getItemCount(): Int {
return categoryList.size
}
}

```

### 3. OrderAdapter.kt

```

package com.example.dpstuffproviderstore.adapter

import android.view.LayoutInflater

```

```

import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import androidx.annotation.NonNull
import androidx.recyclerview.widget.RecyclerView
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.fragment.CartFragment
import com.example.dpstuffproviderstore.fragment.OrdersFragment
import com.example.dpstuffproviderstore.models.ClientData
import com.example.dpstuffproviderstore.models.OrderData
import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.other.ClientApiService

/**
 * Адаптер для заполнения заказов клиента
 */

internal class OrderAdapter(private var ordersList: List<OrderData>, private var fragment:
OrdersFragment) : RecyclerView.Adapter<OrderAdapter.MyViewHolder>() {
    internal class MyViewHolder(view: View) : RecyclerView.ViewHolder(view){
        val orderInfo: TextView = view.findViewById(R.id.tvOrderInfo)
        val orderDate: TextView = view.findViewById(R.id.tvOrderDate)
        val orderStatus: TextView = view.findViewById(R.id.tvOrderStatus)
        val orderNum: TextView = view.findViewById(R.id.tvOrderNum)
        val orderCode: TextView = view.findViewById(R.id.tvOrderCode)
    }

    @NonNull

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
OrderAdapter.MyViewHolder {
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_order, parent, false)
        return OrderAdapter.MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: OrderAdapter.MyViewHolder, position: Int) {

```

```

holder.orderNum.text = "Заказ №${ordersList[position].id}"
holder.orderDate.text = "${ordersList[position].deliveryDate}"
holder.orderStatus.text = "${ordersList[position].status}"
holder.orderInfo.text = "Итого: ${ordersList[position].summ} ₸"
holder.orderCode.text = "Код: ${ordersList[position].codeToFinish}"
}

override fun getItemCount(): Int {
return ordersList.size
}
}

```

#### 4. ProductAdapter.kt

```

package com.example.dpstuffproviderstore.adapter

import android.content.Context
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.ImageView
import android.widget.TextView
import android.widget.Toast
import androidx.annotation.NonNull
import androidx.cardview.widget.CardView
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.RecyclerView
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.`interface`.ICategory
import com.example.dpstuffproviderstore.`interface`.IProduct
import com.example.dpstuffproviderstore.fragment.CartFragment
import com.example.dpstuffproviderstore.fragment.ErrorFragment
import com.example.dpstuffproviderstore.fragment.ProductsFragment
import com.example.dpstuffproviderstore.models.CategoryData
import com.example.dpstuffproviderstore.models.ProductData

```

```

import com.example.dpstuffproviderstore.models.ProductImagesData
import com.example.dpstuffproviderstore.other.ClientApiService
import com.google.gson.Gson
import com.squareup.picasso.Picasso
import kotlinx.android.synthetic.main.fragment_catalog.view.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

/**
 * Адаптер для заполнения информации о товарах
 */

internal class ProductAdapter(private var productsList: List<ProductData>, private var fragment:
Fragment) : RecyclerView.Adapter<ProductAdapter.MyViewHolder>() {
    internal class MyViewHolder(view: View) : RecyclerView.ViewHolder(view){
        val productImage: ImageView = view.findViewById(R.id.imageProduct)
        val productPrice: TextView = view.findViewById(R.id.tvPrice)
        val productTitle: TextView = view.findViewById(R.id.tvTitleProduct)
        val productStore: TextView = view.findViewById(R.id.tvStoreProduct)
        val productRating: TextView = view.findViewById(R.id.tvRatingProduct)
        val btnCart: Button = view.findViewById(R.id.btnAddCart)
    }

    @NonNull

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
        val itemView = LayoutInflater.from(parent.context)
            .inflate(R.layout.item_product, parent, false)
        return MyViewHolder(itemView)
    }

    override fun onBindViewHolder(holder: ProductAdapter.MyViewHolder, position: Int) {
        val mainActivity = fragment.activity as MainActivity

        holder.productPrice.text = "${productsList[position].price} ₺"
        holder.productTitle.text = productsList[position].name
        holder.productStore.text = "Магазин: ${productsList[position].store}"
    }

```

```

holder.productRating.text = "Рейтинг: ${productsList[position].rating}/5"

holder.btnCart.setOnClickListener {

    if(mainActivity.cartList.contains(productsList[position])){
        Toast.makeText(holder.itemView.context, "Товар уже добавлен в корзину",
            Toast.LENGTH_LONG).show()
    }
    else{
        Toast.makeText(holder.itemView.context, "${productsList[position].name} Добавлен в
            корзину", Toast.LENGTH_LONG).show()
        mainActivity.cartList.add(productsList[position])
        mainActivity.cartFragment = CartFragment()

        val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()
        editor.putString("cartList", Gson().toJson(mainActivity.cartList))
        editor.apply()
    }
    Log.i("myLog", "add to cart: ${Gson().toJson(mainActivity.cartList)}")
}

ClientApiService().getImages(productsList[position].id) {

    if(it != null){
        Picasso.with(holder.itemView.context)
            .load(it[0].imageUrl)
            .placeholder(R.drawable.img_placeholder)
            .error(R.drawable.img_placeholder)
            .into(holder.productImage)
    }
    else{
        val fragment = ErrorFragment()
        fragment.statusCode = "404"
        mainActivity.makeCurrentFragment(fragment)
    }
}

override fun getItemCount(): Int {

```

```

return productsList.size
}
}

```

## 5. AccountFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.content.Context
import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import kotlinx.android.synthetic.main.fragment_account.view.*

/**
 * Фрагмент аккаунта залогиненного пользователя
 */
class AccountFragment : Fragment() {

    var testText: String? = null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        val mainActivity = activity as MainActivity
        val inflate = inflater.inflate(R.layout.fragment_account, container, false)

        inflate.tvTestName.text = "${mainActivity.client!!.lastName} ${mainActivity.client!!.firstName} ${mainActivity.client!!.patronymic}"

        inflate.btnOut.setOnClickListener {
            val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()
            editor.putBoolean("isLogin", false)
            editor.apply()

            mainActivity.accountFragment = AccountNotLoginFragment()
            mainActivity.client = null
            mainActivity.makeCurrentFragment(AccountNotLoginFragment())
        }
    }
}

```

```

inflate.cardOrders.setOnClickListener {
    mainActivity.makeCurrentFragment(OrdersFragment())
}

return inflate
}

}

```

## 6. AccountLoginFragment.kt

```
package com.example.dpstuffproviderstore.fragment
```

```

import android.content.Context
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.other.ClientApiService
import kotlinx.android.synthetic.main.fragment_account_login.*
import kotlinx.android.synthetic.main.fragment_account_login.view.*
import kotlin.math.log

/**
 * Фрагмент формы авторизации пользователя
 */
class AccountLoginFragment : Fragment() {

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        val mainActivity = activity as MainActivity
        val inflate = inflater.inflate(R.layout.fragment_account_login, container, false)

        inflate.btnLogin.setOnClickListener {
            ClientApiService().getClient(inputTextLogin.text.toString(), inputTextPassword.text.toString()) {

```



```

if(it != null){
    mainActivity.client = it
    val editor = mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit()
    editor.putBoolean("isLogin", true)
    editor.putString("login", inputTextLogin.text.toString())
    editor.putString("password", inputTextPassword.text.toString())
    editor.apply()
    mainActivity.accountFragment = AccountFragment()
    mainActivity.makeCurrentFragment(AccountFragment())
}
else{
    Toast.makeText(requireContext(), "Неправильный логин и/или пароль",
    Toast.LENGTH_LONG).show()
}
}
}

inflate.btnReg.setOnClickListener {
    mainActivity.makeCurrentFragment(RegistrationFragment())
}

return inflate
}

}

```

## 7. AccountNotLoginFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import kotlinx.android.synthetic.main.fragment_account_notlogin.view.*

```

```

/**
 * Фрагмент не залогиненного пользователя
 */

class AccountNotLoginFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        var inflate = inflater.inflate(R.layout.fragment_account_notlogin, container, false)

        inflate.btnLogin.setOnClickListener {
            val mainActivity = activity as MainActivity
            mainActivity.makeCurrentFragment(AccountLoginFragment())
        }

        return inflate
    }

}

```

## 8. CartEmptyFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import kotlinx.android.synthetic.main.activity_main.*
import kotlinx.android.synthetic.main.fragment_cart.view.*
import kotlinx.android.synthetic.main.fragment_cart_empty.view.*

/**
 * Фрагмент пустой корзины
 */

class CartEmptyFragment : Fragment() {

```

```

override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    val inflate = inflater.inflate(R.layout.fragment_cart_empty, container, false)

    val mainActivity = activity as MainActivity

    inflate.btnGoToCatalog.setOnClickListener {
        mainActivity.bottomNavMenu.selectedItemId = R.id.catalog
    }

    return inflate
}
}

```

## 9. CartFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.content.Context
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.adapter.CartAdapter
import com.example.dpstuffproviderstore.adapter.ProductAdapter
import kotlinx.android.synthetic.main.fragment_cart.view.*
import kotlinx.android.synthetic.main.fragment_products.view.*

/**
 * Фрагмент корзины, корзина заполняется по списку товаров cartList из MainActivity
 */
class CartFragment : Fragment() {

```

```

var inflate : View? = null
var summ : Double = 0.0

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
    inflate = inflater.inflate(R.layout.fragment_cart, container, false)
    val mainActivity = activity as MainActivity

    inflate!!.recyclerCart.layoutManager = LinearLayoutManager(context!!)
    inflate!!.recyclerCart.adapter = CartAdapter(mainActivity.cartList, this@CartFragment)

    inflate!!.tvTitle.text = "${mainActivity.cartList.size} Товара:"

    summ = 0.0
    for(i in mainActivity.cartList){
        summ += i.price!! * (i.quantity ?: 1)
    }
    inflate!!.tvCheckout.text = "Итого: ${summ} P"

    inflate!!.btnCheckout.setOnClickListener {
        if(mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).getBoolean("isLogin",
false)){
            val fragment = CheckoutFragment()
            fragment.checkSumm = summ
            mainActivity.makeCurrentFragment(fragment)
        }
        else{
            Toast.makeText(requireContext(), "Вы не авторизованы", Toast.LENGTH_LONG).show()
        }
    }

    return inflate
}

```

## 10. CatalogFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.opengl.Visibility
import android.os.Bundle

```

```

import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.inputmethod.EditorInfo
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.`interface`.ICategory
import com.example.dpstuffproviderstore.adapter.CategoryAdapter
import com.example.dpstuffproviderstore.models.CategoryData
import com.example.dpstuffproviderstore.other.ClientApiService
import kotlinx.android.synthetic.main.fragment_catalog.*
import kotlinx.android.synthetic.main.fragment_catalog.view.*
import kotlinx.android.synthetic.main.fragment_error.view.*
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

/**
 * Фрагмент каталога с поиском товаров по наименованию и категориям
 */
class CatalogFragment() : Fragment() {

    var inflate : View? = null

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

```

```

inflate = inflater.inflate(R.layout.fragment_catalog, container, false)
val mainActivity = activity as MainActivity

GlobalScope.launch {
    inflate!!.recyclerView.layoutManager = LinearLayoutManager(context!!)

    ClientApiService().getMainCategories() {

        if(it != null){
            inflate!!.recyclerView.adapter = CategoryAdapter(it, this@CatalogFragment, null)
        }
        else{
            val fragment = ErrorFragment()
            fragment.statusCode = "404"
            mainActivity.makeCurrentFragment(fragment)
        }
    }
}

inflate!!.inputTextSearch.setOnEditorActionListener { v, actionId, event ->
    if(actionId == EditorInfo.IME_ACTION_SEARCH){
        Log.i("myLog", "Пошёл поиск...")
        mainActivity.modeSearch = MainActivity.EnumModeSearch.NAME
        mainActivity.productName = inflate!!.inputTextSearch.text.toString()
        mainActivity.makeCurrentFragment(ProductsFragment())
        true
    } else {
        false
    }
}

return inflate
}
}

```

## 11. CheckoutFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.content.Context
import android.os.Bundle

```

```

import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.models.OrderData
import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.other.ClientApiService
import com.google.gson.Gson
import kotlinx.android.synthetic.main.fragment_checkout.*
import kotlinx.android.synthetic.main.fragment_checkout.view.*
import java.util.*

/**
 * Фрагмент оформления нового заказа
 */
class CheckoutFragment : Fragment() {

    var inflate : View? = null
    var checkSumm : Double = 0.0

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        inflate = inflater.inflate(R.layout.fragment_checkout, container, false)
        val mainActivity = activity as MainActivity

        //inflate!!.dateDelivery.minDate = java.util.Calendar.getInstance()
        inflate!!.tvCheckoutConfirm.text = "Итого: ${checkSumm} Р"

        var c = Calendar.getInstance()
        c.add(Calendar.DAY_OF_MONTH, 1)
        inflate!!.dateDelivery.minDate = c.timeInMillis
        c.add(Calendar.DAY_OF_MONTH, 14)

```

```

inflate!!.dateDelivery.maxDate = c.timeInMillis

var frontDoor : Int? = null
var apartNum : Int? = null
var floorNum : Int? = null

if(!inflate!!.inputTextFrontDoorCheckout.text.isNullOrEmpty()){
frontDoor = inflate!!.inputTextFrontDoorCheckout.text.toString().toInt()
}

if(!inflate!!.inputTextApartCheckout.text.isNullOrEmpty()){
apartNum = inflate!!.inputTextApartCheckout.text.toString().toInt()
}

if(!inflate!!.inputTextFloorCheckout.text.isNullOrEmpty()){
floorNum = inflate!!.inputTextFloorCheckout.text.toString().toInt()
}

inflate!!.btnCheckoutConfirm.setOnClickListener {
if(!validForm()){
return@setOnClickListener
}

val newOrder = OrderData(
id = null,
address = inflate!!.inputTextAddressCheckout.text.toString(),
lastName = mainActivity.client!!.lastName,
firstName = mainActivity.client!!.firstName,
phone = mainActivity.client!!.phone,
frontDoor = frontDoor,
apartNum = apartNum,
floorNum = floorNum,
intercom = inflate!!.inputTextIntercomCheckout.text.toString(),
deliveryDate = "${inflate!!.dateDelivery.dayOfMonth}.${inflate!!.dateDelivery.month + 1}.${inflate!!.dateDelivery.year}",
timeFrom = inflate!!.spinnerTimeFrom.selectedItem.toString(),
timeTo = inflate!!.spinnerTimeTo.selectedItem.toString(),
commentary = inflate!!.inputTextCommentaryCheckout.text.toString(),
summ = checkSumm,

```



```

status = null,
idCourier = null,
priority = 5,
codeToFinish = ((1000..9999).random().toString()),
listProducts = mainActivity.cartList
)

Log.i("myLog", "checkout: ${Gson().toJson(newOrder.listProducts)}")

ClientApiService().addOrder(newOrder){
if(it == null){
Toast.makeText(requireContext(), "Ошибка, повторите позже", Toast.LENGTH_LONG).show()
mainActivity.makeCurrentFragment(CartFragment())
}
else{
Toast.makeText(requireContext(), "Заказ успешно оформлен", Toast.LENGTH_LONG).show()
mainActivity.cartList = arrayListOf()
mainActivity.cartFragment = CartEmptyFragment()
mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).edit().putString("cartList",
Gson().toJson(mainActivity.cartList)).apply()
mainActivity.makeCurrentFragment(CartEmptyFragment())
}
}
}

return inflate
}

fun validForm(): Boolean{
if(inflate!!.inputTextAddressCheckout.text.isNullOrEmpty()){
Toast.makeText(requireContext(), "Поле адреса обязательно для заполнения",
Toast.LENGTH_LONG).show()
return false
}

if(inflate!!.spinnerTimeFrom.selectedItemId >= inflate!!.spinnerTimeTo.selectedItemId){
Toast.makeText(requireContext(), "Некорректное время доставки",
Toast.LENGTH_LONG).show()
return false
}
}

```

```

    }

    return true
}
}

```

## 12. ErrorFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import com.example.dpstuffproviderstore.R
import kotlinx.android.synthetic.main.fragment_error.view.*

/**
 * Фрагмент заглушки на случай ошибок (не отвечает api)
 */
class ErrorFragment : Fragment() {

    var statusCode : String = "404"

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        val inflate = inflater.inflate(R.layout.fragment_error, container, false)

        inflate.tvStatusCode.text = statusCode

        return inflate
    }
}

```

## 13. HomeFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater

```

```

import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.R.string
import com.example.dpstuffproviderstore.`interface`.IProduct
import com.example.dpstuffproviderstore.adapter.ProductAdapter
import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.other.ClientApiService
import kotlinx.android.synthetic.main.fragment_home.view.*
import kotlinx.android.synthetic.main.fragment_products.view.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

/**
 * Фрагмент домашней страницы со списком популярных товаров (пока просто выводит все
 * товары)
 */
class HomeFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        val inflate = inflater.inflate(R.layout.fragment_home, container, false)
        val mainActivity = activity as MainActivity

        inflate.recyclerBestProducts.layoutManager = LinearLayoutManager(context!!)

        ClientApiService().getAllProducts() {

            if(it != null){
                inflate.recyclerBestProducts.adapter = ProductAdapter(it, this@HomeFragment)
            }
        }
    }
}

```

```

    }
    else{
        val fragment = ErrorFragment()
        fragment.statusCode = "404"
        mainActivity.makeCurrentFragment(fragment)
    }
}

return inflate
}

    }

```

#### 14. OrdersFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.content.Context
import android.opengl.Visibility
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.adapter.CartAdapter
import com.example.dpstuffproviderstore.adapter.OrderAdapter
import com.example.dpstuffproviderstore.other.ClientApiService
import com.example.dpstuffproviderstore.other.ServiceBuilder
import kotlinx.android.synthetic.main.fragment_cart.view.*
import kotlinx.android.synthetic.main.fragment_orders.view.*

/**
 * Фрагмент со списком заказов пользователя
 */
class OrdersFragment : Fragment() {

```

```

override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
savedInstanceState: Bundle?): View? {
    val inflate = inflater.inflate(R.layout.fragment_orders, container, false)
    val mainActivity = activity as MainActivity

    inflate!!.recyclerView.layoutManager = LinearLayoutManager(context!!)
    ClientApiService().getOrderByClient(mainActivity.client!!.login!!,
mainActivity.getSharedPreferences("sp", Context.MODE_PRIVATE).getString("password",
""))!!){
        if(it.isNullOrEmpty()){
            inflate!!.hintEmpty.visibility = View.VISIBLE
        }
        else{
            inflate!!.recyclerView.adapter = OrderAdapter(it, this@OrdersFragment)
        }
    }

    return inflate
}
}

```

## 15. ProductsFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.`interface`.ICategory
import com.example.dpstuffproviderstore.`interface`.IProduct
import com.example.dpstuffproviderstore.adapter.CategoryAdapter
import com.example.dpstuffproviderstore.adapter.ProductAdapter
import com.example.dpstuffproviderstore.models.CategoryData

```

```

import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.other.ClientApiService
import kotlinx.android.synthetic.main.fragment_catalog.view.*
import kotlinx.android.synthetic.main.fragment_products.view.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

/**
 * Фрагмент со списков товаров (по наименованию или категории)
 */
class ProductsFragment : Fragment() {

    var inflate : View? = null

    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        inflate = inflater.inflate(R.layout.fragment_products, container, false)
        val mainActivity = activity as MainActivity

        inflate!!.recyclerProducts.layoutManager = LinearLayoutManager(context!!)

        if(mainActivity.modeSearch == MainActivity.EnumModeSearch.CATEGORY)
        {
            ClientApiService().getProductsByCategory(mainActivity.productCategory) {

                if(it != null){
                    inflate!!.recyclerProducts.adapter = ProductAdapter(it, this@ProductsFragment)
                }
                else{
                    val fragment = ErrorFragment()
                    fragment.statusCode = "404"
                    mainActivity.makeCurrentFragment(fragment)
                }
            }
        }
    }
}

```

```

if(mainActivity.modeSearch == MainActivity.EnumModeSearch.NAME)
{
    ClientApiService().getProductsByName(mainActivity.productName) {
        if(it != null){
            inflate!!.recyclerProducts.adapter = ProductAdapter(it, this@ProductsFragment)
        }
        else{
            val fragment = ErrorFragment()
            fragment.statusCode = "404"
            mainActivity.makeCurrentFragment(fragment)
        }
    }
}

return inflate
}
}

```

## 16. RegistrationFragment.kt

```

package com.example.dpstuffproviderstore.fragment

import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Toast
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.R
import com.example.dpstuffproviderstore.models.ClientData
import com.example.dpstuffproviderstore.other.ClientApiService
import kotlinx.android.synthetic.main.fragment_registration.*
import kotlinx.android.synthetic.main.fragment_registration.view.*

/**
 * Фрагмент регистрации нового пользователя

```

```

*/
class RegistrationFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val inflate = inflater.inflate(R.layout.fragment_registration, container, false)
        val mainActivity = activity as MainActivity

        inflate.btnReg.setOnClickListener {
            if(!validForm()){
                return@setOnClickListener Unit
            }
            val client = ClientData(
                id = null,
                lastName = inputTextLastName.text.toString(),
                firstName = inputTextFirstName.text.toString(),
                patronymic = inputTextPatronymic.text.toString(),
                email = inputTextEmail.text.toString(),
                phone = inputTextPhone.text.toString(),
                login = inputTextLogin.text.toString(),
                password = null
            )

            ClientApiService().addClient(client, inputTextPassword.text.toString()){
                if(it != null){
                    Toast.makeText(context!!, "Регистрация успешна", Toast.LENGTH_LONG).show()
                    mainActivity.makeCurrentFragment(AccountNotLoginFragment())
                }
                else{
                    Toast.makeText(context!!, "Что-то пошло не по плану, повторите позже",
                        Toast.LENGTH_LONG).show()
                }
            }

        }

        return inflate
    }
}

```



```

    }

    fun validForm() : Boolean{
        if(inputTextFirstName.text.isNullOrEmpty() ||
            inputTextLastName.text.isNullOrEmpty() ||
            inputTextPatronymic.text.isNullOrEmpty() ||
            inputTextEmail.text.isNullOrEmpty() ||
            inputTextPhone.text.isNullOrEmpty() ||
            inputTextLogin.text.isNullOrEmpty() ||
            inputTextPassword.text.isNullOrEmpty()){
            Toast.makeText(context!!, "Заполните все поля", Toast.LENGTH_LONG).show()
            return false
        }

        if(!inputTextEmail.text!!.contains('@') || !inputTextEmail.text!!.contains('.')){
            Toast.makeText(context!!, "Некорректный адрес почты", Toast.LENGTH_LONG).show()
            return false
        }

        if(inputTextPassword.text!!.length < 8){
            Toast.makeText(context!!, "Пароль должны быть не короче 8-ми символов",
                Toast.LENGTH_LONG).show()
            return false
        }

        if(inputTextPassword.text.toString() != inputTextRepeatPassword.text.toString()){
            Toast.makeText(context!!, "Пароли не совпадают", Toast.LENGTH_LONG).show()
            return false
        }

        return true
    }
}

```

## 17. ClientApiService.kt

```

package com.example.dpstuffproviderstore.other

import android.util.Log
import com.example.dpstuffproviderstore.MainActivity
import com.example.dpstuffproviderstore.`interface`.ICategory

```

```

import com.example.dpstuffproviderstore.`interface`.IClient
import com.example.dpstuffproviderstore.`interface`.IOrder
import com.example.dpstuffproviderstore.`interface`.IProduct
import com.example.dpstuffproviderstore.adapter.ProductAdapter
import com.example.dpstuffproviderstore.fragment.ErrorFragment
import com.example.dpstuffproviderstore.models.*
import kotlinx.android.synthetic.main.fragment_home.view.*
import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

/**
 * Класс, реализующий все методы Api-интерфейсов
 */
class ClientApiService {
    /**
     * Получаем объект клиента по логину и паролю (авторизация)
     */
    fun getClient(login: String, password: String, onResult: (ClientData?) -> Unit){
        val api = ServiceBuilder.buildService(IClient::class.java)

        api.GetClient(login, password).enqueue(object : Callback<List<ClientData>> {
            override fun onResponse(call: Call<List<ClientData>>,
                response: Response<List<ClientData>>) {
                {
                    if(response.code() == 200){
                        onResult(response.body()!![0])
                    }
                    else{
                        Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
                        onResult(null)
                    }
                }
            }

            override fun onFailure(call: Call<List<ClientData>>, t: Throwable){
                Log.i("myLog", "Api Failure")
                onResult(null)
            }
        })
    }
}

```

```

    }
    })
}

/**
 * Регистрация нового пользователя
 */

fun addClient(clientData: ClientData, noHashPassword: String, onResult: (ClientData?) -> Unit){
    val api = ServiceBuilder.buildService(IClient::class.java)

    api.addUser(clientData, noHashPassword).enqueue(object : Callback<List<ClientData>> {
        override fun onResponse(call: Call<List<ClientData>>,
            response: Response<List<ClientData>>
        ) {
            if(response.code() == 200){
                onResult(response.body()!![0])
            }
            else{
                Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
                onResult(null)
            }
        }
    })

    override fun onFailure(call: Call<List<ClientData>>, t: Throwable){
        Log.i("myLog", "Api Failure")
        onResult(null)
    }
})

/**
 * Регистрация нового заказа
 */

fun addOrder(orderData: OrderData, onResult: (OrderData?) -> Unit){
    val api = ServiceBuilder.buildService(IOrder::class.java)

    api.addOrder(orderData).enqueue(object : Callback<OrderData> {
        override fun onResponse(call: Call<OrderData>,

```

```

response: Response<OrderData>
) {
if(response.code() == 200){
onResult(response.body()!!)
}
else{
Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
onResult(null)
}
}

override fun onFailure(call: Call<OrderData>, t: Throwable){
Log.i("myLog", "Api Failure")
onResult(null)
}
})
}

/**
 * Получаем список заказов конкретного клиента (по логину и паролю)
 */
fun getOrderByClient(login: String, password: String, onResult: (List<OrderData>?) -> Unit){
val api = ServiceBuilder.buildService(IOrder::class.java)

api.getOrdersByClient(login, password).enqueue(object : Callback<List<OrderData>> {
override fun onResponse(call: Call<List<OrderData>>,
response: Response<List<OrderData>>)
) {
if(response.code() == 200){
onResult(response.body()!!)
}
else{
Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
onResult(null)
}
}
}
}

```

```

override fun onFailure(call: Call<List<OrderData>>, t: Throwable){
    Log.i("myLog", "Api Failure")
    onResult(null)
}
})
}

/**
 * Получаем первый объект изображения, относящийся к товару (по id товара)
 */
fun getImages(productId: Int, onResult: (List<ProductImagesData>?) -> Unit){
    val api = ServiceBuilder.buildService(IProduct::class.java)

    api.GetImages(productId).enqueue(object : Callback<List<ProductImagesData>> {
        override fun onResponse(call: Call<List<ProductImagesData>>,
            response: Response<List<ProductImagesData>>
        ) {
            if(response.code() == 200){
                onResult(response.body()!!)
            }
            else{
                Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
                onResult(null)
            }
        }
    })

    override fun onFailure(call: Call<List<ProductImagesData>>, t: Throwable){
        Log.i("myLog", "Api Failure")
        onResult(null)
    }
})
}

/**
 * Получаем список основных категорий (у которых нет родительской категории)
 */
fun getMainCategories(onResult: (List<CategoryData>?) -> Unit){

```

```

val api = ServiceBuilder.buildService(ICategory::class.java)

api.GetMainCategories().enqueue(object : Callback<List<CategoryData>> {
    override fun onResponse(call: Call<List<CategoryData>>,
        response: Response<List<CategoryData>>) {
        if(response.code() == 200){
            onResult(response.body()!!)
        }
        else{
            Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()}")
            onResult(null)
        }
    }

    override fun onFailure(call: Call<List<CategoryData>>, t: Throwable){
        Log.i("myLog", "Api Failure")
        onResult(null)
    }
})

/**
 * Получаем список дочерних категорий (по имени родительской категории)
 */
fun getChildCategories(categoryName: String, onResult: (List<CategoryData>?) -> Unit){
    val api = ServiceBuilder.buildService(ICategory::class.java)

    api.GetChildCategories(categoryName).enqueue(object : Callback<List<CategoryData>> {
        override fun onResponse(call: Call<List<CategoryData>>,
            response: Response<List<CategoryData>>) {
            if(response.code() == 200){
                onResult(response.body()!!)
            }
            else{
                Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()}")

```

```

onResult(null)
}
}

override fun onFailure(call: Call<List<CategoryData>>, t: Throwable){
Log.i("myLog", "Api Failure")
onResult(null)
}
})
}

/**
 * Получаем список всех товаров в БД
 */
fun getAllProducts(onResult: (List<ProductData>?) -> Unit){
val api = ServiceBuilder.buildService(IProduct::class.java)

api.GetAllProducts().enqueue(object : Callback<List<ProductData>> {
override fun onResponse(call: Call<List<ProductData>>,
response: Response<List<ProductData>>
) {
if(response.code() == 200){
onResult(response.body()!!)
}
else{
Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
onResult(null)
}
}

override fun onFailure(call: Call<List<ProductData>>, t: Throwable){
Log.i("myLog", "Api Failure")
onResult(null)
}
})
}

/**

```

\* Получаем список товаров (по наименованию)

\*/

```
fun getProductsByName(productName: String, onResult: (List<ProductData>?) -> Unit){  
    val api = ServiceBuilder.buildService(IProduct::class.java)
```

```
    api.GetProductsByName(productName).enqueue(object : Callback<List<ProductData>> {  
        override fun onResponse(call: Call<List<ProductData>>,  
            response: Response<List<ProductData>>
```

```
    ) {
```

```
        if(response.code() == 200){
```

```
            onResult(response.body()!!)
```

```
        }
```

```
    } else{
```

```
        Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
```

```
        onResult(null)
```

```
    }
```

```
}
```

```
override fun onFailure(call: Call<List<ProductData>>, t: Throwable){
```

```
    Log.i("myLog", "Api Failure")
```

```
    onResult(null)
```

```
}
```

```
})
```

```
}
```

```
/**
```

\* Получаем список товаров, относящихся к категории и ко всем дочерним категориям (по имени категории)

\*/

```
fun getProductsByCategory(categoryName: String, onResult: (List<ProductData>?) -> Unit){  
    val api = ServiceBuilder.buildService(IProduct::class.java)
```

```
    api.GetProductsByCategory(categoryName).enqueue(object : Callback<List<ProductData>> {  
        override fun onResponse(call: Call<List<ProductData>>,  
            response: Response<List<ProductData>>
```

```
    ) {
```

```
        if(response.code() == 200){
```

```
            onResult(response.body()!!)
```



```

    }
    else{
        Log.i("myLog", "${response.code()}; ${response.errorBody()}; ${response.message()};")
        onResult(null)
    }
}

override fun onFailure(call: Call<List<ProductData>>, t: Throwable){
    Log.i("myLog", "Api Failure")
    onResult(null)
}
})
}
    }

```

## 18. MainActivity.kt

```

package com.example.dpstuffproviderstore

import android.content.Context
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import com.example.dpstuffproviderstore.fragment.*
import com.example.dpstuffproviderstore.models.ClientData
import com.example.dpstuffproviderstore.models.ProductData
import com.example.dpstuffproviderstore.other.ClientApiService
import com.google.gson.Gson
import com.google.gson.reflect.TypeToken
import kotlinx.android.synthetic.main.activity_main.*
import java.lang.reflect.Type
import kotlin.reflect.typeOf

class MainActivity : AppCompatActivity() {

    enum class EnumModeSearch{ CATEGORY, NAME }

    //val listCategory : List<CategoryData> = listOf(CategoryData(1, "1cat", ""), CategoryData(2,
    "2cat", ""), CategoryData(3, "3cat", ""))

```

```

var productCategory : String = ""
var productName : String = ""
var modeSearch = EnumModeSearch.CATEGORY
var client: ClientData? = null
var cartList: ArrayList<ProductData> = arrayListOf()

var accountFragment: Fragment = AccountNotLoginFragment()
var homeFragment: Fragment = HomeFragment()
var catalogFragment: Fragment = CatalogFragment()
var cartFragment: Fragment = CartEmptyFragment()

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    val sharedPreferences = getSharedPreferences("sp", Context.MODE_PRIVATE)
    if(sharedPreferences.getBoolean("isLogin", false)){
        ClientApiService().getClient(sharedPreferences.getString("login", "")!!,
            sharedPreferences.getString("password", "")!!){
            if (it != null){
                client = it
                accountFragment = AccountFragment()
            }
            else{
                accountFragment = AccountNotLoginFragment()
            }
        }
    }

    //sharedPreferences.edit().clear().apply()
    var jsonString: String? = sharedPreferences.getString("cartList", "")
    if(!jsonString.isNullOrEmpty() && jsonString != "[]"){
        Log.i("myLog", "json строка не пустая")
        val itemType = object : TypeToken<ArrayList<ProductData>>() {}.type
        cartList = Gson().fromJson<ArrayList<ProductData>>(jsonString, itemType)
        cartFragment = CartFragment()
    }

    setContentView(R.layout.activity_main)

```

```

makeCurrentFragment(homeFragment)

bottomNavMenu.setOnNavigationItemSelectedListener {
when(it.itemId){
R.id.home -> makeCurrentFragment(homeFragment)
R.id.catalog -> makeCurrentFragment(catalogFragment)
R.id.cart -> makeCurrentFragment(cartFragment)
R.id.account -> makeCurrentFragment(accountFragment!!)
}
true
}

}

public fun makeCurrentFragment(fragment: Fragment) =
supportFragmentManager.beginTransaction().apply {
replace(R.id.frameWrapper, fragment, "activeFragment")
commit()
}

}

```

## ПРИЛОЖЕНИЕ Г.РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

### АННОТАЦИЯ

В данном программном документе приведено руководство пользователя для мобильного приложения по продаже товаров и услуг DP Stuff Provider.

В данном программном документе, в разделе «Назначение программы» указаны функционал, место эксплуатации и конечный пользователь программы.

В данном программном документе, в разделе «Условия выполнения программы» указаны рекомендуемые технические средства для использования программы.

В данном программном документе, в разделе «Выполнение программы» указаны всевозможные руководства для запуска, удаления и использования программы.

## СОДЕРЖАНИЕ

АННОТАЦИЯ.....	1
1. НАЗНАЧЕНИЕ ПРОГРАММЫ.....	3
2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ.....	4
3. ВЫПОЛНЕНИЕ ПРОГРАММЫ .....	5
3.1. Действия для загрузки программы .....	5
3.2. Действия для запуска программы .....	6
3.3. Выполнение программы с описанием функций .....	7
3.4. Действия для удаления программы .....	12
4. СООБЩЕНИЯ ОПЕРАТОРУ .....	14

## 1. НАЗНАЧЕНИЕ ПРОГРАММЫ

Функциональным назначением программы является эксплуатация любым рядовым пользователем, с целью поиска и заказа нужного товара или услуги.

Программа должна эксплуатироваться на мобильном устройстве под управлением ОС Android с стабильным доступом к интернету.

Конечными пользователями программы должны быть любые люди, скачавшее данное приложение.

## 2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

В Таблице 1 представлены рекомендуемые технические средства для использования программы.

Таблица 1 - Технические средства

№	Тип оборудования	Наименование оборудования
1	2	3
Смартфон		
1	Процессор:	MTK6762R и лучше
2	Оперативная память:	2GB+
3	Место на телефоне:	64МБ
4	Разрешение экрана:	любое
5	Размер экрана:	5,0 дюйма и выше
6	Тип экрана:	любой
7	Операционная система:	Android 5.1 и выше

В Таблице 2 представлены программные средства для использования программы.

Таблица 2 - Программные средства

№	Тип средства	Название средства	Назначение
1	2	3	4
1	Система управления базами данных	MS SQL Server 2017	Создание и администрирование базы данных
2	Текстовый редактор	Microsoft Word 2016	Разработка документации, формирование отчетных документов по шаблонам
3	Среда разработки мобильного приложения	Android Studio 4.1.3	Разработка мобильного приложения
4	Среда разработки Api	Visual Studio 2019	Разработка и управление Api

### 3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

#### 3.1. Действия для загрузки программы

DP Stuff Provider устанавливается с помощью арк-файла «dpsp-store.apk» (Рисунок 1), актуальную версию которого можно скачать из GitHub-репозитория по ссылке <https://github.com/AlexGopher802/DPStuffProvider>, или можно просто перекинуть арк-файл с компьютера на мобильное устройство (далее будет рассматриваться этот вариант).



Рисунок 1 - Установщик

Затем подключаем телефон к компьютеру и включаем передачу файлов по USB (Рисунок 2).

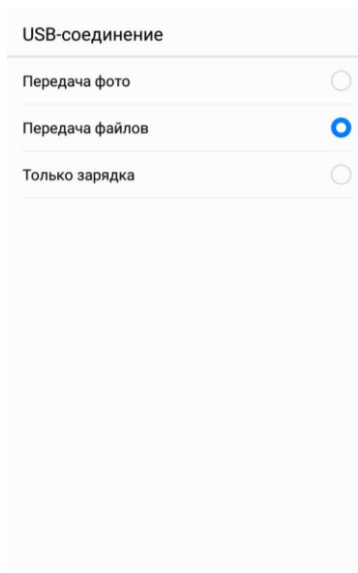


Рисунок 2 - Отладка по USB

Переходим в проводнике к мобильному устройству и ищем папку Downloads (Рисунок 3).



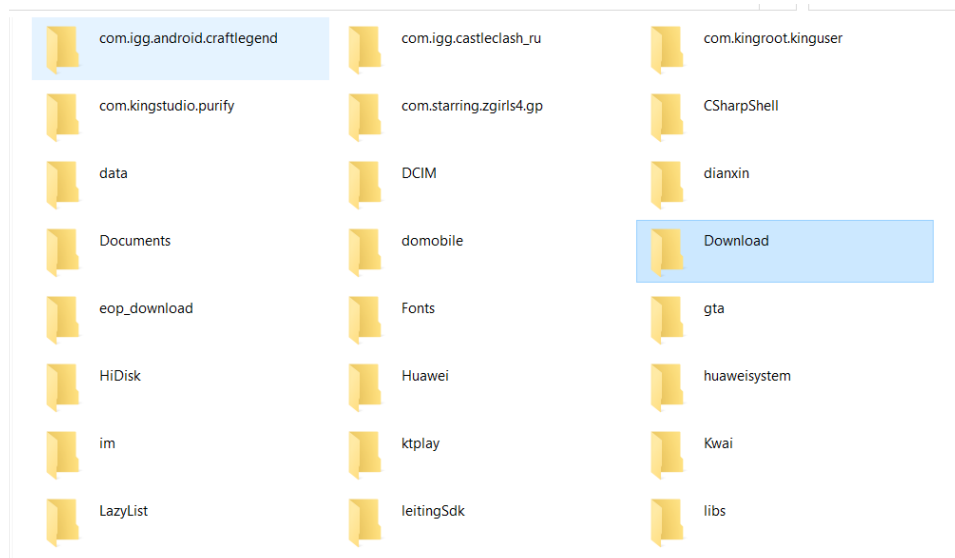


Рисунок 3 - Папка загрузок

Перекопируем установщик «dpsp-store.apk» в папку Downloads.

В телефоне переходим в проводник и ищем папку Downloads.

Нажимаем на файл «dpsp-store.apk» и затем нажимаем на кнопку «Установить» (Рисунок 4).

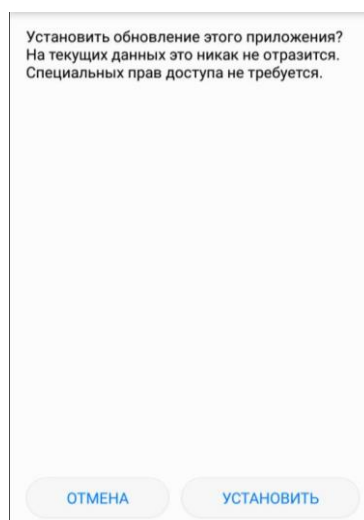


Рисунок 4 - Установка

### 3.2. Действия для запуска программы

После установки Slime Rancher wiki может быть запущено с помощью ярлыка на экране (Рисунок 5).



Рисунок 5 - Запуск программы

### 3.3. Выполнение программы с описанием функций

После запуска программы пользователя встречает главное меню и домашняя страница, где отображается список самых популярных товаров.

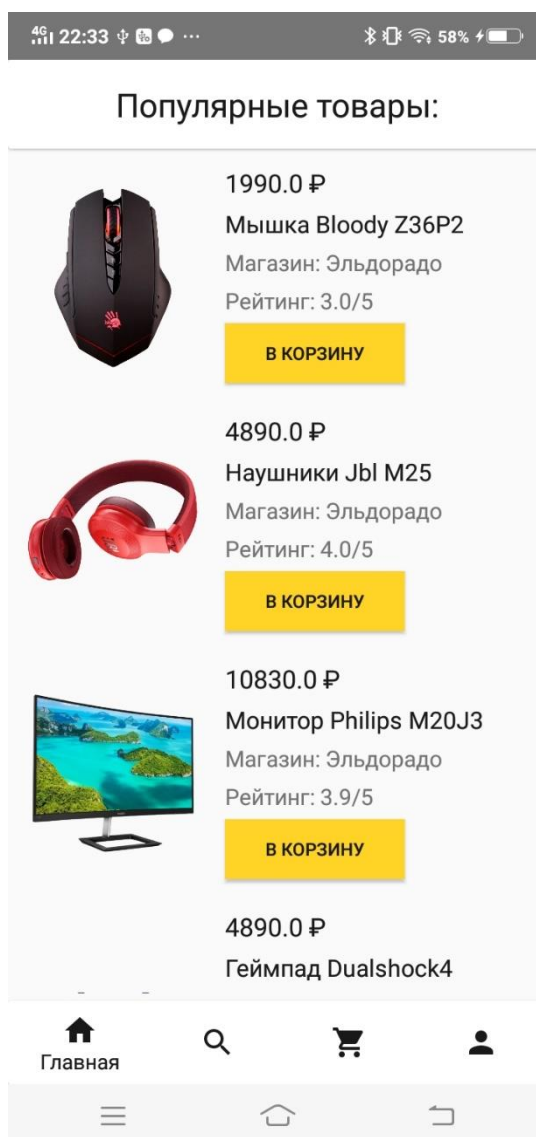


Рисунок 6 – Домашняя страница

Мы можем перейти в каталог и поискать нужный нам товар по категориям или по наименованию.

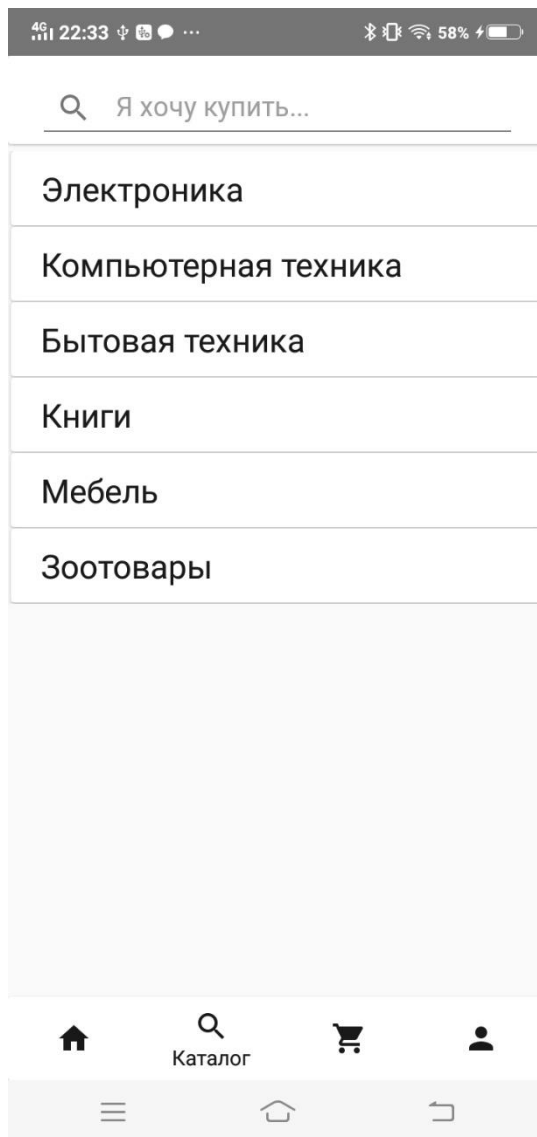


Рисунок 7 – Каталог

После нахождения и добавления нужных товаров в корзину, можем посмотреть, что находится внутри.

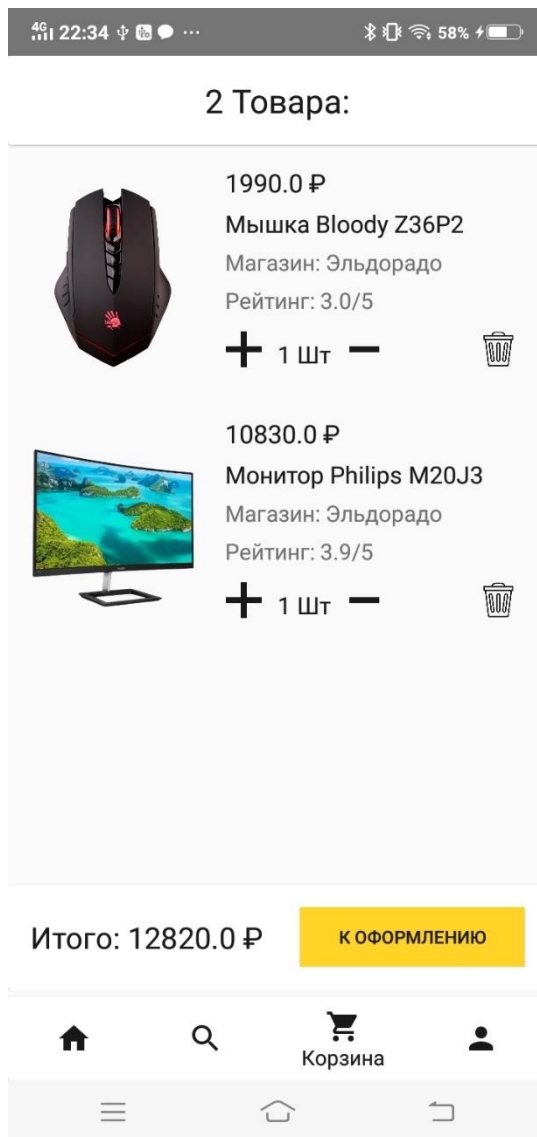


Рисунок 8 – Корзина

Если пользователь авторизован – можно перейти к оформлению заказа.

4G 22:35 58%  
Подтверждение заказа

Адрес доставки

Подъезд    Квартира    Этаж

Домофон

Дата доставки

11	июн.	2021
12	июл.	

От 10:00    До 10:00

Итого: 12820.0 ₽    **ОФОРМИТЬ**

🏠    🔍    🛒 Корзина    👤

☰    🏠    ↩

Рисунок 9 – Оформление заказа

В случае, если пользователь не авторизован – у него есть возможность авторизоваться или зарегистрироваться.

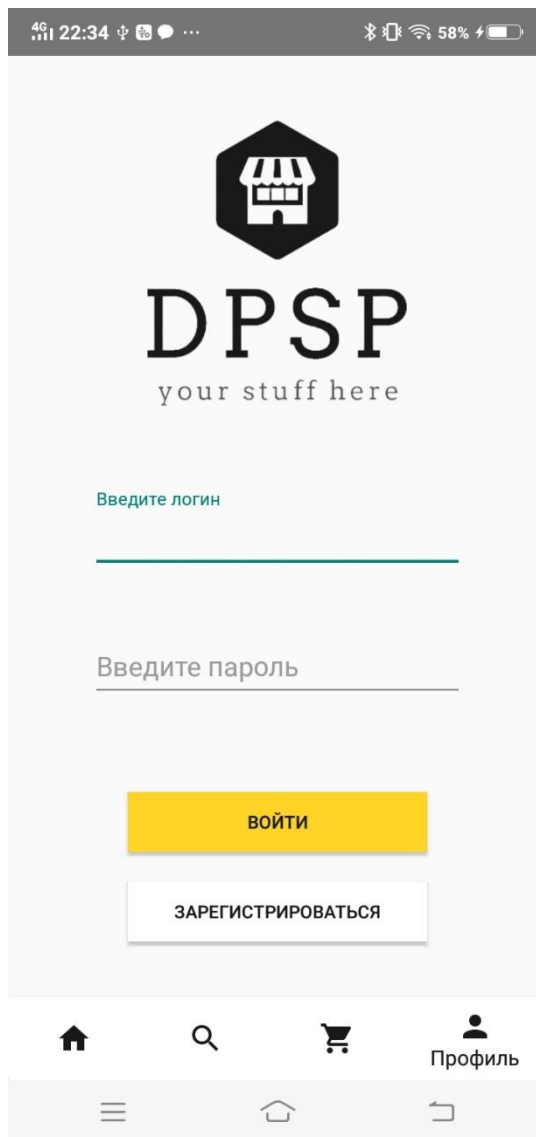


Рисунок 10 – Авторизация

Регистрация

Фамилия

Имя

Отчество

Телефон

Почта

Логин

Пароль

Повторите пароль

ЗАРЕГИСТРИРОВАТЬСЯ

Профиль

Рисунок 11 - Регистрация

### 3.4. Действия для удаления программы

Чтобы удалить приложение вы просто перетаскиваете приложение в корзину или удерживаете ярлык и нажимаете крестик (Рисунок 12) и нажимаете на кнопку удалить (Рисунок 13).

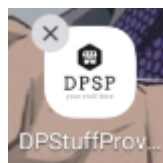


Рисунок 12 – Нажимаем крестик

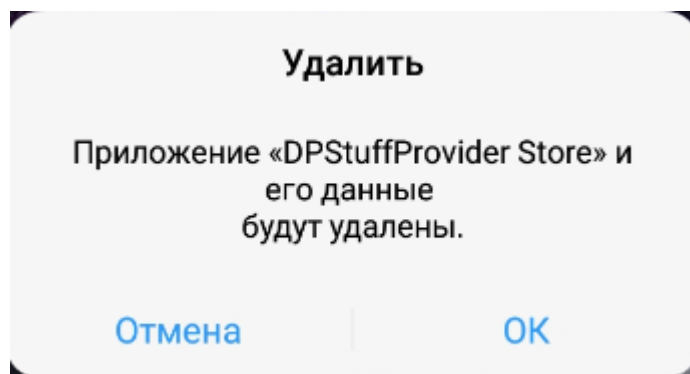


Рисунок 13 – Удаление



#### 4. СООБЩЕНИЯ ОПЕРАТОРУ

В таблице 4 представлены все сообщения, выводимые оператору во время эксплуатации программы.

Таблица 3 - Сообщения

№	Текст сообщения	Содержание сообщения	Действия оператора
1	2	3	4
1	Неизвестная ошибка, повторите позже	Указывает на неизвестную ошибку (скорее всего просто необработанное исключение или недоступность Арі в данный момент)	Повторить действие чуть позже
2	Нет соединения с интернетом или сервис сейчас не доступен. Повторите еще раз чуть позже...	Указывает на то, что нету соединения с интернетом или Арі в данный момент не доступна	Повторить действие чуть позже
3	Timeout	Указывает на то, что ответ от сервера идёт слишком долго	Повторить действие чуть позже
4	Неверный логин и/или пароль	Указывает что вводимый логин и/или пароль некорректны	Ввести корректный логин и пароль, или зарегистрировать новый аккаунт