**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ**

**Кафедра веб-технологий и компьютерного моделирования**

**Разработка дизайна и Android-приложения магазина спорттоваров**

Курсовая работа

Гапановича Евгения Вадимовича

студента 3 курса, специальность 1-31 03 08-02

Математическое и программное обеспечение мобильных устройств

Научный руководитель:

Старший преподаватель,  
Вельченко Сергей Александрович

Минск, 2021

# Оглавление

[Оглавление 2](#_Toc72064846)

[1. ВВЕДЕНИЕ 3](#_Toc72064847)

[2. ОСНОВНАЯ ЧАСТЬ 4](#_Toc72064848)

[2.1. Теоретическая часть 4](#_Toc72064849)

[2.1.1. Сравнение с аналогами. Обзор существующих разработок и место данной работы среди других приложений. 4](#_Toc72064850)

[2.1.2. Обзор существующих технологий разработки приложений. 5](#_Toc72064851)

[2.1.3. Проектирование системы. Создание блок-схемы модели 11](#_Toc72064852)

[2.1.4. Системный анализ проблемы. Детализация проекта. 12](#_Toc72064853)

[2.2. ПРАКТИЧЕСКАЯ ЧАСТЬ 14](#_Toc72064854)

[2.2.1. Описание реализации разработки ключевых систем 14](#_Toc72064855)

[2.2.2. Описание функционала 25](#_Toc72064856)

[2.3. ОРГАНИЗАЦИОННАЯ ЧАСТЬ 26](#_Toc72064857)

[2.3.1. Перспективы развития проекта 26](#_Toc72064858)

[2.3.2. Вопросы маркетинга. Обоснование возможных перспектив, требуемые заемные средства, планы развития. 26](#_Toc72064859)

[3. ЗАКЛЮЧЕНИЕ. ВЫВОДЫ 28](#_Toc72064860)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 29](#_Toc72064861)

[ПРИЛОЖЕНИЕ А 30](#_Toc72064862)

[ПРИЛОЖЕНИЕ Б 31](#_Toc72064863)

[ПРИЛОЖЕНИЕ В 32](#_Toc72064864)

# ВВЕДЕНИЕ

Современный мир не стоит на месте, а тем более информационные технологии. Еще несколько десятилетий назад чтобы купить необходимый товар нужно было идти на рынок или в магазин, а теперь, когда мир продвинулся в технологическом плане, достаточно установить на свой телефон соответствующее приложение и сделать заказ прямиком в нём.

И вот за счёт этого технологического прорыва, пользователям не приходиться тратить огромное количество своего личного времени, для того чтобы куда-то добраться, потом найти необходимую вещь и купить её, когда теперь всего лишь достаточно пару нажатий смартфоне и необходимую вам вещь доставят в скором времени.

Поэтому наличие таких приложений очень помогает миру, людям, чтобы они экономили своё личное время.

Целью данной работы было создание мобильного приложения магазина спорттоваров.

# ОСНОВНАЯ ЧАСТЬ

## Теоретическая часть

### Сравнение с аналогами. Обзор существующих разработок и место данной работы среди других приложений.

Прежде чем создавать производство какого-либо продукта, нужно сперва разузнать, посмотреть, есть ли такой продукт уже на рынке, если он присутствует, то его владельцы, разработчики – наши прямые конкуренты. В таком случае нужно провести полный анализ их системы, производства:

* узнать на какой регион они ориентированы
* какая основная группа людей является потребителями
* и другое.

Будем ориентироваться на рынок в Беларуси. В интернете очень огромное количество онлайн-магазинов, в которых можно приобрести спортивные товары: Onliner, 21vek, Bigsport, Sport-Center, Wildberries, VIMPEX SPORT.

Из выше перечисленных сервисов не у всех есть мобильное приложение, поэтому анализировать будем: Onliner, 21vek, Wildberries

Каталог Onliner – функционирует как приложение для продажи товаров и услуг. Кроме как продажи спортивных товаров, он также занимается продажей различной электроникой, бытовой техникой, товарами для дома, стройки и многое другое, то есть в основном специализируется на продаже различных видов товаров.

21vek – функционирует как приложения для продажи различных товаров и услуг. Специализируется на продаже электроники и бытовой технике, товаров для дома и многое другое. То есть в основном он ориентирован на продажу различных видов товаров.

Wildberries – приложение специализировано на продажу всевозможных товаров (в том числе и на продажу спортивных товаров) как и приложения Каталог Onliner и 21vek.

Можно сделать вывод, что в основном как таковых мобильных приложений магазина спортивных товаров у нас на рынке не присутствует. Присутствуют либо же приложения, которые предоставляют товары различных видов (электроника, бытовая техника, для дома и др.), спортивные товары у них конечно же в наличии есть, но они не могут обеспечить пользователей всеми ими видами, а лишь маленькой долей, либо же присутствуют онлайн веб-сервисы для покупки спорттоваров, но мы на них не ориентируемся, так как наша цель создать мобильное приложение.

Почему именно мобильное приложение? Потому что сейчас почти у каждого есть мобильный телефон, который всегда с собой и почти каждый большую часть своего времени проводит за ним, даже можно сказать чуть ли не весь день, поэтому логичнее всего создавать мобильное приложение, и тогда у пользователя весь на протяжении всего дня будет возможность посетить наше мобильное приложение.

### Обзор существующих технологий разработки приложений.

Существует два метода разработки мобильных приложений:

* Нативная разработка
* Кроссплатформенная(гибридная) разработка

#### Нативная мобильная разработка

Нативная мобильная разработка – это такой метод разработки, который является традиционным способом создания приложений для каждой мобильной операционной системы, используя разные языки и учитывая особенности каждой платформы.

Такие приложения создаются и настраиваются исключительно для конкретных мобильных операционных систем, таких как: Android, iOS, WindowsPhone.

Нативная мобильная разработка использует инструменты и языки программирования, оригинальные для устройства и операционной системы. Apple, Google, Microsoft и другие поставщики разрабатывают специальные инструменты, SDK и элементы интерфейса для своих платформ.

Для создания приложения на iOS потребуются языки программирования Objective-C или Swift, среды, такие как XCode или AppCode, плюс iOS SDK, Cocoa Touch и другие инструменты.

Для создания приложения на Android потребуются языки программирования Java, Kotlin или С++, среды Android Studio или Eclipse, а также Android SDK, Mockplus и другие инструменты для приложений Android.

Для создания приложений для Windows Phone потребуются языки C# или C++, среда Microsoft Visual Studio, а также Windows Phone SDK, Silverlight SDK и другие инструменты.

Нативное Android-приложение не будет работать на платформе iOS, и наоборот.

Полный доступ к аппаратным функциям (камера, GPS, диктофон, аудио- и видеофайлы) и программным службам (календарь, уведомления) позволяет нативным приложениям чувствовать себя как дома на целевых устройствах.

Еще одна важная особенность заключается в том, что собственные приложения могут работать в автономном режиме, в то время как кроссплатформенные приложения требуют постоянного подключения к Интернету.

Нативные мобильные приложения скачиваются из соответствующих цифровых магазинов - App Store и Google Play.

Преимущества разработки собственных мобильных приложений:

Нативные приложения предлагают ряд существенных преимуществ, которые могут иметь решающее значение при выборе типа мобильного решения.

1. Безупречная работа.

Когда разработчики создают собственные приложения, они изначально оптимизируют их для всех конкретных функций целевой платформы и используют все доступные API. Таким образом, приложения получают полную поддержку оборудования и ОС и могут демонстрировать высокий уровень производительности.

Нативные приложения очень отзывчивы и быстры, потому что их целевые устройства загружают и хранят все визуальные элементы и элементы контента, что способствует скорости загрузки.

1. Расширенная настройка

Собственная разработка подразумевает доступ ко всем инструментам и функциям, доступным в операционной системе и на устройствах. Для клиентов это означает гибкость и возможность создавать уникальные решения в соответствии с их потребностями и желаниями.

Кроме того, тестирование мобильных приложений для нативных проектов намного проще благодаря ряду встроенных инструментов автоматического тестирования. Удаленный мониторинг ошибок позволяет разработчикам контролировать ошибки на пользовательских устройствах и устранять их причины. В результате вы получаете эффективный продукт без ошибок, соответствующий вашим требованиям.

1. Максимальное удобство для пользователя

Каждая мобильная платформа имеет уникальные стандарты UI / UX. Разработчики собственных приложений придерживаются этих стандартов, благодаря чему приложения выглядят и ощущаются как неотъемлемая часть операционной системы.

Такие интуитивно понятные и интерактивные решения улучшают пользовательский опыт. Люди могут легко взаимодействовать с новым приложением, потому что они знакомы с макетом и жестами, необходимыми для выполнения того или иного действия. Таким образом, нативные мобильные приложения помогают компаниям привлекать более лояльных клиентов.

4. Безопасность

В наши дни защита данных стоит на первом месте. В этом отношении нативные приложения обеспечивают лучшую безопасность, чем кроссплатформенные приложения. Почему? Гибридные решения наследуют уязвимости безопасности от платформ и браузеров, на которых они работают. Кроме того, универсальные языки, используемые в этих приложениях, также могут создавать риски.

Нативные приложения разработаны для единой платформы, поэтому данные шифруются в рамках одной инфраструктуры, что снижает количество рисков. Если вы гарантируете более высокий уровень защиты данных, следовательно, вы завоюете доверие своей аудитории.

5. Мгновенное обновление

Когда Android и iOS выпускают обновления, разработчики нативных приложений могут немедленно внедрять новые функции. И наоборот, кросс-платформенным инженерам приходится ждать появления дополнительных инструментов для корректировки обновлений своих решений. Это замедляет процесс и ухудшает взаимодействие с пользователем.

Мгновенный доступ к самым последним функциям делает нативный подход к разработке более выгодным для клиентов и конечных пользователей.

Проблемы нативного мобильного приложения:

В настоящее время большинство мобильных приложений создано более чем для одной платформы. Когда разработчики создают отдельное собственное приложение для каждой платформы, они должны использовать разные инструменты, API, среды и подходы к дизайну интерфейса. Следовательно, чтобы сделать три версии одного приложения совместимыми с тремя платформами, потребуется нанять три отдельные команды. Общая стоимость проекта может быть в три раза выше, не говоря уже о задержке вывода на рынок.

#### Кроссплатформенная (гибридная) мобильная разработка

Термин «кроссплатформенная разработка мобильных приложений» или «разработка гибридных мобильных приложений» говорит сам за себя: это подход, который позволяет разработчикам создавать мобильное приложение, которое одновременно совместимо с несколькими операционными системами и платформами, такими как: Android, iOS, WindowsPhone.

Гибридные приложения имеют естественный внешний вид благодаря комбинации собственного кода с независимым кодом, который подходит для нескольких платформ. Разработчики пишут код один раз, а затем повторно используют его, что позволяет быстро выпустить продукт.

Для кодирования кроссплатформенного программного обеспечения разработчики используют промежуточные языки программирования - HTML, JavaScript и CSS, которые не являются родными для устройств и ОС. Затем приложения упаковываются в собственные контейнеры и интегрируются в платформы.

Кроссплатформенная разработка направлена на создание единого приложения, которое одинаково работает на нескольких платформах. Он использует технологии, не зависящие от платформы, такие как HTML и CSS, и помогает компаниям покрывать множество конечных устройств с меньшими затратами.

Плюсы кроссплатформенной(гибридной) мобильной разработки:

Разработка мультиплатформенных мобильных приложений постоянно развивается благодаря новым технологиям, становясь все более динамичными и привлекательными для разработчиков.

1. Создание отдельных нативных приложений для каждой платформы обходится дорого, в то время как гибридное приложение использует единый общий код, который помогает вам уложиться в свой бюджет.
2. Затраты снижаются, поскольку для разработки и сопровождения приложения требуется только одна команда программистов. Более того, достаточно базовых знаний стандартных языков - остальную работу сделают инструменты разработки.
3. Кроссплатформенные приложения имеют собственный внешний вид, что очень удобно для пользователей.
4. Гибридная разработка - это, безусловно, путь для компаний, которые хотят привлечь пользователей различных мобильных устройств и быстрее выпустить продукт на рынок с меньшими затратами.

Минусы кроссплатформенной(гибридной) мобильной разработки:

1. Более сложный код гибридных решений объединяет собственные и неродные компоненты, что может повлиять на производительность.
2. Кроссплатформенные приложения не могут поддерживать все встроенные функции и возможности мобильных устройств, такие как расширенная графика и анимация или 3D-эффекты. Это приводит к ограниченной функциональности и плохому дизайну приложения.
3. Когда Google и Apple добавляют новые функции на платформы Android и iOS, нативные решения могут немедленно начать их использовать. Но гибридным приложениям придется подождать, пока эти обновления не будут адаптированы к выбранной кроссплатформенной структуре. Таким образом, всегда будет задержка обновления.

Кроссплатформенная разработка подходит для решений, которые:

1. не требуют продвинутого дизайна;
2. не нужно обрабатывать входящие данные онлайн;
3. не нужен доступ ко всем функциям устройства.

Основные фреймворки для разработки кроссплатформенных(гибридных) мобильных приложений:

В зависимости от инструмента, который вы используете, процесс и результаты разработки программного обеспечения могут отличаться. Чтобы создать приложение, похожее на нативное, крайне важно выбрать правильный фреймворк.

1. Xamarin

Xamarin - это ведущая бесплатная платформа с открытым исходным кодом для создания гибридных мобильных приложений. Приложения на основе Xamarin совместимы с любой мобильной платформой и обеспечивают такую же производительность и удобство для пользователя, как нативные приложения. Этот популярный кроссплатформенный фреймворк мобильных приложений Microsoft основан на языке программирования C#, дополненном фреймворком .NET, и доступен по лицензии MIT как часть Visual Studio. Его набор инструментов для разработки включает в себя следующие компоненты:

* Xamarin Studio / Visual Studio (собственная IDE)
* Xamarin SDK
* Xamarin Test Cloud
* Xamarin.Mac для создания собственных приложений Mac

1. React Native

React Native - одна из лучших кроссплатформенных сред разработки для мобильных устройств. Он основан на React - лучшей в своем классе библиотеке JavaScript для создания пользовательских интерфейсов - и нацелен на мобильные платформы. Самое замечательное в React Native заключается в том, что вы можете использовать универсальный язык JavaScript, а также языки Java, Swift или C, специфичные для нативных платформ, для создания кроссплатформенных приложений. Таким образом, этот фреймворк может преобразовывать исходный код в собственные элементы, улучшая взаимодействие с пользователем. Веб-разработчики могут легко использовать React Native для создания мобильных приложений, не приобретая новых навыков программирования.

1. Ionic

Ionic определенно входит в число лучших гибридных фреймворков для приложений, поскольку он объединяет все востребованные свойства: это открытый исходный код, простой в обслуживании, масштабируемый и удобный для чтения. Ionic предоставляет библиотеку компонентов HTML, CSS и JS, оптимизированных для разработки мобильных приложений, которые помогают создавать интерактивные приложения. Ionic имеет богатый набор функций и инструментов, в том числе готовую поддержку материального дизайна, мобильных элементов пользовательского интерфейса и макетов в собственном стиле. Он также предоставляет своим клиентам регулярные обновления и ресурсы от сообщества разработчиков.

Гибридные приложения на ionic-основе обладают встроенными функциями, особыми жестами и настраиваемыми инструментами, которые повышают удобство использования. Однако они полагаются на оболочки, такие как Cordova или PhoneGap, чтобы работать как нативные.

1. Adobe PhoneGap

PhoneGap, поддерживаемый Adobe, является распространением фреймворка Cordova. Эта среда разработки мобильных приложений с открытым исходным кодом обычно считается лучшим и наиболее популярным инструментом для гибридных приложений, не в последнюю очередь из-за простоты использования. Инструмент позволяет писать приложения на HTML5, CSS3 и JavaScript. Он предоставляет собственные плагины и контейнеры для каждой мобильной платформы, позволяя разработчикам создавать дополнительные функции и получать доступ к функциям устройства, таким как камера, микрофон, акселерометр, компас, файловая система и многое другое. После создания приложения запускаются в браузере WebView внутри собственного контейнера на соответствующей платформе.

1. Sencha

Sencha является лидером в разработке приложений корпоративного уровня, поскольку эта комплексная платформа помогает командам сосредоточиться на создании приложений, а не на создании инфраструктуры. Этот инструмент основан на современных веб-технологиях, включая ES6, HTML5, JavaScript и CSS, поэтому разработчики могут создавать кроссплатформенные веб-приложения и мобильные приложения с большим объемом данных. Первоначально названный Sencha Touch, сервис больше не поддерживается и был объединен с Ext JS для создания Sencha Ext JS. В нем есть набор инструментов и тем, которые повышают производительность и облегчают оптимизацию, развертывание, проектирование, создание тем и отладку.

### Проектирование системы. Создание блок-схемы модели

Так как у нас онлайн приложение магазина спорттоваров, то контент в приложение должен обновляться динамически и автоматически, то есть при добавлении нового товара в базу данных, пользователю не потребуется обновлять и скачивать новую версию приложения. На основе этого была построена блок-схема (рисунок 1.), позволяющая решать данную идею.



Рисунок 1. - Блок-схема модели

По данной блок-схеме можно видеть принцип работы такого приложения: пользователь хочет, чтобы у него на экране отобразился список всевозможных типов товаров, которые есть в приложении, для этого приложение отправляет соответствующий GET-запрос на сервер, он же в свою очередь уже подключен к базе данных и отправляет соответствующий sql-запрос для получения соответствующих данных, база данных получив запрос отправляет на сервер необходимые данные и сервер отправляет эти данные в мобильное приложение, а оно в свою очередь отображает полученные данные на экране пользователя. При оформлении заказа, пользователь делает соответствующее действие в мобильном приложении, после чего мобильное приложение отправляет POST-запрос на сервер, тот в свою очередь отправляет sql запрос в базу данных, чтобы вставить заказ в таблицу. Весь процесс данного механизма от пользователя скрыт, он получает только его результат.

В блок-схеме можно видеть такой элемент, как база данных (рисунок 2.). Ниже представлена её структура.

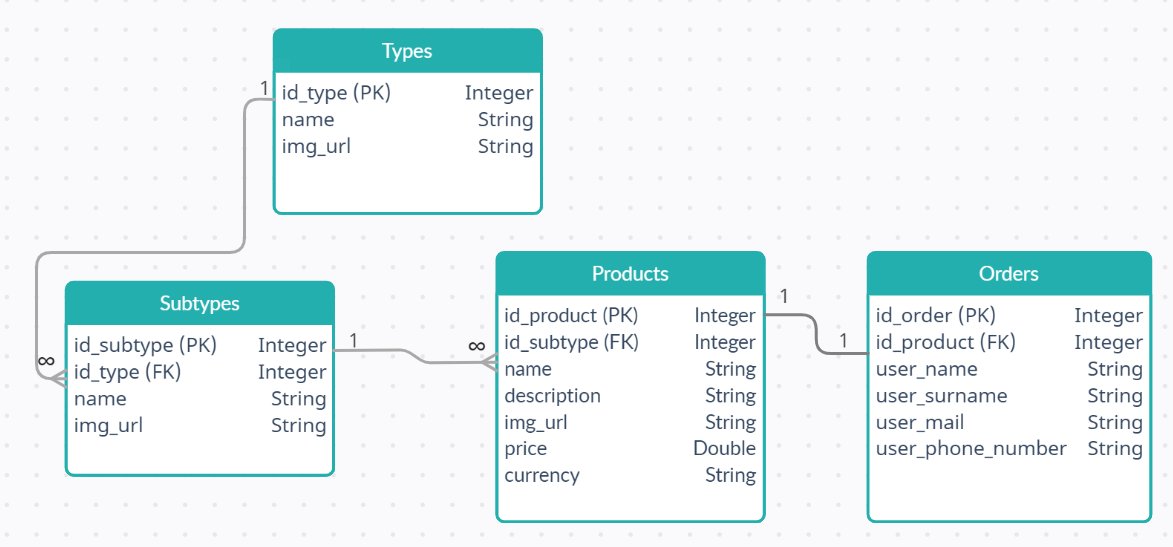


Рисунок 2. - База-данных

Как видно, в базе данных присутствуют четыре таблицы, это: таблица типов, таблица подтипов, таблица товаров и таблица заказов. Таблица подтипов связана с таблицей типов по ключу id\_type, таблица товаров связана с таблицей подтипов по ключу id\_subtype, таблица заказов связана с таблицей товаров по ключу id\_product.

### Системный анализ проблемы. Детализация проекта.

Перед началом разработки практической части, разобьём ранее описанную блок-схему на отдельные компоненты, для того чтобы было понятно с чем нам придется иметь дело, и для того, чтобы правильно выстроить план работы.

1. Создание базы данных.
2. Создание сервера.
3. Создание мобильного приложения.

Для хранения товаров, нам понадобиться база данных. С помощью её мы сможем не только хранить товары, но, а также их добавлять и удалять. Остановимся на базе данных MySQL. Взаимодействовать с ней будем с помощью инструмента MySQL WorkBeanch, он позволяет в реальном времени интерактивно управлять базой данных, а именно создавать и удалять таблицы, создавать и удалять объекты в таблицах, обновлять таблицы и многое другое.

Далее нам понадобиться помощник, который будет связующим звеном между взаимодействием мобильным приложением и базой данных. Остановимся на выборе сервера на основе Node.js с использованием фреймворка Express. Они очень просты в установке на компьютер и в самом использовании. При создании сервера, мы можем создать файл конфигурации, в котором пропишем все необходимые нам разрешения для соединения сервера с базой данных. После того как наш сервер будет подключен к созданной нами ранее базе данных, мы сможем с него отправлять sql-запросы в базу данных, а полученные ответы отправлять в мобильное приложение.

Мобильное приложение выберем на базе ОС Android, потому что в данное время Android является лидером в мире и у нас в стране приблизительно 80% населения используют ОС Android. Поэтому количество потенциальных пользователей нашего приложения огромно.

Для создания мобильного приложения будем использовать такую среду разработки как Android studio, она бесплатна и проста в установке, язык программирования будем использовать Kotlin, так как сейчас в основном большая часть мобильных приложений под Android пишется на нём и язык сам прост в изучении и использовании.

## ПРАКТИЧЕСКАЯ ЧАСТЬ

### Описание реализации разработки ключевых систем

#### Создание базы данных

Нам понадобиться база данных для того, чтобы хранить в ней наши товары. Остановимся на MySQL, основными её качествами являются: надежность, высокая скорость, гибкость и простота в установке.

Внутри MySQL создадим схему под название app\_sporting\_goods (рисунок 3.), она же будет содержать все необходимые таблицы.



Рисунок 3. - схема в базе данных

Будем использовать четыре таблицы: таблицу product\_types для хранения типов товаров, таблицу product\_subtypes для хранения подтипов товаров, таблицу products для хранения самих товаров и таблицу orders для хранения заказов.

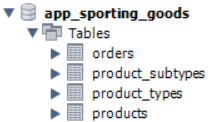


Рисунок 4. - Структура схемы

Таблица product\_types содержит такие колонки (рисунок 5.), как:

* id\_type (уникальный идентификатор типа товара)
* name (название типа товара)
* img\_url (ссылка на картинку типа товара)

|  |  |
| --- | --- |
| Рисунок 5. - Структура таблицы типов | Рисунок 6. – Таблица типов |

Таблица product\_subtypes содержит такие колонки (рисунок 7), как:

* id\_subtype (уникальный идентификатор подтипа товара)
* name (название подтипа товара)
* id\_type (идентификатор типа товара, к которому принадлежит подтип товара)
* img\_url (ссылка на картинку подтипа товара)

|  |  |
| --- | --- |
| Рисунок 7. - Структура таблицы подтипов | Рисунок 8. - Таблица подтипов |

Таблица products содержит такие колонки (рисунок 9.), как:

* id\_product (уникальный идентификатор товара)
* name (название товара)
* description (описание товара)
* price (цена товара)
* currency (валюта)
* id\_subtype (идентификатор подтипа для товара)
* img\_url (ссылка на картинку товара)

|  |  |
| --- | --- |
| Рисунок 9. - Структура таблицы товаров | Рисунок 10. - Таблица товаров |

Таблица orders содержит такие поля, как:

* id\_order (уникальный идентификатор заказа)
* user\_name (имя пользователя, сделавшего заказ)
* user\_surname (фамилия пользователя, сделавшего заказ)
* user\_mail (почта пользователя, сделавшего заказ)
* user\_phone\_number (номер пользователя, сделавшего заказ)
* id\_product (идентификатор товара)

|  |  |
| --- | --- |
| Рисунок 11. - Структура таблицы заказов | Рисунок 12. - Таблица заказов |

#### Создание локального сервера

После того как база данных была создана, необходимо теперь преступить к разработке локально сервера, с помощью которого мы сможем отправлять sql-запросы в базу данных и потом полученные ответы отправлять в мобильное приложение. Сервер будем создавать на основе Node.js с использованием фреймворка Express. Выбираем Express, потому что это один из популярных фреймворков, который поддерживает роутинг, миддлвейрс и систему шаблонов, да и сам Express предоставляется как основа для создания приложений на Node.js.

Структура проекта выглядит следующим образом (рисунок 13.).

|  |
| --- |
| Рисунок 13. - Структура проекта |

Пройдемся по имеющимся папкам. В папке config находиться файл db.config.js (рисунок 14.), в нем находиться код, с помощью которого наш сервер подключается к базе данных, для этого мы указывает необходимые поля, такие как: хост, данные пользователя, под которым мы подключаемся, и название схемы из базы данных, к которой нам необходимо подключиться (рисунок 15.).

|  |  |
| --- | --- |
| Рисунок 14. Файл конфигурации | Рисунок 15. - Данные для подключения к базе данных |

Далее идет папка controllers. В ней лежат соответствующие файлы контроллеров для каждой модели, которые присутствуют в проекте (рисунок 16.).

Далее можно видеть структуру метода для получения конкретного типа товара по определённому поле id\_type (рисунок 17.).

|  |  |
| --- | --- |
| Рисунок 16. - Структура папки controllers | Рисунок 17. - Метод получение товара по полю id\_type |

Далее следует папка modules. В ней храняться те объекты (сущности), которые используеются в проекте (рисунок 18.)

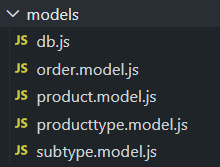


Рисунок 18. - Структура папки models

В файле db.js храниться объект который используется для соединения с базой данных (рисунок 19.). В остальных файлах используются объекты, которые возвращаются при запросе, а также прописаны методы, которые используют sql-запросы, для получения данных из таблиц (рисунок 20.).

|  |  |
| --- | --- |
| Рисунок 19 – Файл db.js | Рисунок 20. - Структура объекта типа товара |

В папке routes (рисунок 21.) хранятся файлы, в которых прописаны все возможные запросы для получения данных по соответствующей модели (рисунок 22.).

|  |  |
| --- | --- |
| Рисунок 21. - Структура паки routes | Рисунок 22. - Файл с запросами для типа товара |

В файле server.js указан порт, на котором запускается сервер, значение у него 5000, в качество ip используется IPv4-адрес той сети, к которой подключен компьютер. Сервер запускается при помощи команды node server.js (рисунок 23.).

|  |
| --- |
| Рисунок 23. - Запуск локального сервера |

#### Создание мобильного приложения

После того как реализована база данных и работает сервер, с которого можно отправлять запросы в базу данных и получить ответы, то можно преступить к реализации мобильного приложения.

Писать будем в среде разработке Android studio на языке программирования Kotlin.

Нам понадобятся библиотеки, которые помогут реализовать функционал в приложении. Воспользуемся такими библиотека, как:

* Firebase
* Retrofit
* Coroutines
* RecyclerView
* Picasso

Firebase я использовал для хранения информации о зарегистрировавшихся пользователях.

Firebase - поставщик облачных услуг, принадлежащий Google. Основной сервис - нереляционная (NoSQL) облачная база данных, дающая возможность авторам приложений хранить и синхронизировать данные между несколькими пользователями. Поддержаны особенности интеграции с приложениями под операционные системы Android и iOS, реализовано API для приложений на Java, Node.js, Objective-C, JavaScript, также возможно работать напрямую с базой данных наподобие REST из ряда JavaScript-фреймворков, включая aнгулар, реакт, Vue, эмбер и другие. Предусмотрено API для шифрования данных.

Как только пользователь заходит в приложение, то перед ним появляется страница с входом (рисунок 24.).

|  |  |
| --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-192722.jpg  Рисунок 24. - Страница входа | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-192856.jpg  Рисунок 25. - Страница регистрации |

Если он впервые в данном приложение, то ему необходимо нажать на текст «Создать аккаунт», после чего пользователя перенаправят на страницу регистрации (рисунок 25.), заполнив все поля, пользователь нажимает на кнопку «Зарегистрироваться». После того как он нажмет на кнопку, его перекинет в само приложение, а в базу данных Firebase добавится новый пользователь (рисунок 26.). В ней можно увидеть: почту, дату создания аккаунта, дату последнего посещения, уникальный идентификатор пользователя.

|  |
| --- |
| Рисунок 26. - База данных Firebase |

Если пользователь уже регистрировался в приложение, то для того чтобы войти в приложение вновь, то ему достаточно на странице входа ввести свою почту и пароль, а потом нажать на кнопку «ВОЙТИ», после чего, если в данные введены корректно, его перенаправят на страницу приложения.

Для входа использовался метод signInWithEmailAndPassword из объекта класса FirebaseAuth библиотеки Firebase, данный метод проверяет существуют ли такой пользователь в базе данных, если да, то пользователь входит в приложение, если нет, то выдается сообщение об ошибке.

Для регистрации использовался метод createUserWithEmailAndPassword из объекта класса FirebaseAuth библиотеки Firebase, данный метод проверяет, если такой пользователь не существует в базе данных, то происходит сохранение пользователя в базу данных, и он успешно входит в приложение, если же такой пользователь в базе данных существует, то регистрация не происходит и выдается сообщение об ошибке.

Как только пользователь авторизовался, он переходит на страницу «Каталог» (рисунок 27.). Здесь отображаются те типы товаров, которые находятся сейчас в базе данных. С помощью библиотеки Retrofit мобильное приложение делает соответствующий запрос на сервер, тот в свою очередь делает sql-запрос в базу данных, после чего, полученные данные сервер отправляет в приложение, далее приложение при помощи библиотеки Coroutines, которая служит, чтобы отображение данных происходило асинхронно и с помощью инструмента RecyclerView динамически отображает полученные данные, при помощи библиотеки Picasso по полученным url-картинков удаётся их загрузить на экран пользователя.

Ниже (рисунок 29.) вы можете видеть доступные товары, если же нажать на один из них, то открывается соотвествующая страница товара (рисунок 30.), здесь мы можем видеть: изображение товара, его название, цену, описание, а также две кнопки «В корзину» и «В избранное».

|  |  |  |  |
| --- | --- | --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-193200.jpg  Рисунок 27. - Страница каталог с типами товаров | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-194018.jpg  Рисунок 28. - Страница каталог с подтипами товаров | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-194113.jpg  Рисунок 29. - Страница каталог с товарами | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-195213.jpg  Рисунок 30. - Страница товара |

Если же перейти на страницы «Корзина» и «Избранное», то можем видеть, что в них ничего нету, так как мы пока ничего не добавили.

|  |  |
| --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-194331.jpg  Рисунок 31. - Страница корзины | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-194337.jpg  Рисунок 32. - Страница избранного |

Чтобы исправить эту ситуации, на странице товара достаточно нажать кнопки «В корзину» и «В избранное», убедимся что данный товар добавился на соответствующих страницах (рисунки 33., 34.), так же если мы хотим удалить ранее добавленный товар из списка, для этого достаточно нажаться кнопку «Удалить».

|  |  |
| --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210515-221651.jpg  Рисунок 33. – Страница корзины с товаров | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-195845.jpg  Рисунок 34. - Страница избранного с товаром |

При добавлении товара в корзину, видим, что у пользователя появляется кнопка «Заказать», по её нажатию открывается страница для заполнения информации для того, чтобы оформить заказ (рисунок 35.). После введённой информации и подтверждения заказа, оформленный заказ отправляется в таблицу orders нашей базы данных, а страница «Корзина» становиться пустой (рисунок 36.). Чтобы посмотреть все оформленные заказы, нужно перейти в «Профиль» и нажать на кнопку «Мои заказы».

|  |  |
| --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210515-221842.jpg  Рисунок 35. - Страница оформления заказа | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210515-222548.jpg  Рисунок 36. - Страница корзины |

Перейдя на страницу «Профиль» (рисунок 37.) можем наблюдать, что здесь есть поля виде личной информации о пользователе, которые он может заполнить или изменить, нажав на кнопку «Изменить», тогда ему откроется страница, где он сможет ввести все необходимое (рисунок 38.), также присутствует кнопка «Мои заказы», нажав на неё, пользователя перенаправит на страницу, где будут отображаться все те товары, которые он заказывал (рисунок 39). Внизу страницы «Профиль» присутствует кнопку «Выйти», нажав на которую, пользователя перенаправит на страницу входа. Для выхода из аккаунта, используется метод signOut() класса FirebaseAuth.

|  |  |  |
| --- | --- | --- |
| C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210515-222937.jpg  Рисунок 37. - Страница профиля | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210514-201010.jpg  Рисунок 38. - Страница изменения личной информации | C:\Users\User\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Screenshot_20210515-222942.jpg  Рисунок 39. - Страница заказанных товаров |

### Описание функционала

В конце разработки было получено мобильное клиент-серверное приложение, позволяющая пользователю просматривать товары и оформлять заказы.

Для начала работы пользователю достаточно войти под действительным аккаунтом или же зарегистрироваться, далее мобильное приложение отправит соответствующие запросы на сервер, тот в свою очередь соответствующие sql-запросы в базу данных, в которой находятся все необходимые нам товары, после сервер отправит полученный ответ в приложение, и то в свою очередь отобразит пользователю полученные данные. При добавлении товаров в корзину, пользователь может оформить заказ, введя необходимую информацию и подтвердив заказ, приложение отправляет post-запрос, в котором находиться информация о заказе, на сервер, тот отправляет его при помощи sql-запроса в базу данных, все свои заказанные товары пользователь может просмотреть в своём профиле.

Разработанное приложение уже содержит в себе готовые реализации получения данных из базы данных и отправление данных в базу данных, поэтому весь исходный функционал совместим с дальнейшими различными интеграциями, которые приведут к улучшению качества приложения.

## ОРГАНИЗАЦИОННАЯ ЧАСТЬ

### Перспективы развития проекта

Имеющаяся на данный момент версия приложения не является конечной, для того чтобы её отправить пользователю, предстоит не мало работы, но то что мы имеем сейчас говорит нам об многом и делает наши взгляды на будущее перспективными. Дальнейшее развитие будет зависеть от того, чтобы сделать приложение более функциональнее, оптимизирование и привлекательнее, планируется добавить такой функционал как: все возможные регистрации через различные сервисы, для того что привлечь большее количество новых пользователей, функции отмена заказа, доставки, оплаты и т.д., в качестве оптимизации планируется запуск приложений на различных версиях Android, для того чтобы у всех их владельцев был доступ к приложению, и также разработка приложения под операционную систему iOS, также в качестве оптимизации планируется рефакторинг кода, использование новейших библиотек, которые приведут к быстроте исполнения мобильного приложения, также планируется усовершенствование дизайна на всем протяжении работоспособности приложения, потому что первое на что замечает пользователь при открытии приложения – это то, как оно выглядит, то есть его дизайн, и уже на основе увиденного у пользователя может сложиться кое-какое мнение на счет приложения, и чтобы мнение не становилось негативным, будем производиться редизайн приложения, чтобы пользователям было приятно его посещать.

На основе выше сказанного имеем, что работа над приложением на данном этапе не прекращается, а будет производиться огромная работа, которая приведет приложение к наилучшему качеству.

### Вопросы маркетинга. Обоснование возможных перспектив, требуемые заемные средства, планы развития.

Дальнейшее развитие проекта заключено в разработке нового функционала, оптимизации приложения и в доработке дизайна, одному человеку выполнять такой план работы очень трудно и занять это может огромное количество времени, и по истечению которого некоторые вещи просто могут стать не актуальны, поэтому чтобы избежать такого и все выполнить так, чтобы не тратить огромное количество времени, основные затраты будут отведены на создание команд, которым будет отведена отдельная часть в развитии и разработке приложения, а именно разработка функционала, тестирование приложения и разработка дизайна.

Далее понадобятся небольшие затраты на маркетинг приложения, в основном пользователи смогут узнать о приложение из рекламы на интернет-площадках по продаже различных товаров во время поиска товаров, из рекомендаций в Google Play Market и AppStore.

Также понадобятся небольшие затраты на то, чтобы вывести приложения в Google Play Market и AppStore, для того чтобы пользователи могли его установить на свое мобильное устройство.

На начальном этапе заемные средства для создания такого приложения не планируются, так как первый капитал приложения будет базироваться на внутренних вкладах самих разработчиков, далее после вывода приложения на рынок, заработок планируется вестись от прибыли с продажи товара, плюс получение прибыли от встроенной контекстной рекламы внутри мобильного приложения.

# ЗАКЛЮЧЕНИЕ. ВЫВОДЫ

В результате проведенной работы, было получено клиент-серверное приложение на базе операционной системы Android. Во время процесса создания приложения, были разработаны такие важные компоненты работы, как:

1. Создание базы данных пользователей приложения. Использовалась база данных Firebase, предоставляющая регистрацию пользователя по введенной почте и паролю.
2. Создание базы данных для хранения товаров. Использовалась база данных MySQL, которая содержит в себе четыре таблицы для хранения типов товаров, подтипов товаров, самих товаров и заказов.
3. Создание локального сервера. Сервер создан на основе Node.js с использованием фреймворка Express. При помощи определенных параметров сервер подключается к базе данных, для того чтобы отправлять с него конкретные sql-запросы и после полученные данные отправлять в мобильное приложение.
4. Создание мобильного приложения. Мобильное приложение создано на базе операционной системы Android, с помощью его мы отправляем пользовательские запросы на сервер, а полученные ответы отображаем на экране у пользователя.

При создании приложения были исследованы различные типы и методы разработки клиент-серверных приложений, также было исследовано множество библиотек, для того чтобы решать проблемы, которые возникали во время разработки приложения, и чтобы актуальность функционала приложения оставалось на очень долгое время.

Следует отметить, что существуют перспективы развития проекта, поэтому предстоит огромная работа.

В итоге можно сказать, что получили не только базовое приложение, но, а также приложение, которое готово к дальнейшему совершенствованию.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Database Firebase [Электронный ресурс]. – Режим доступа: https://firebase.google.com/
2. Основы создания приложений [Электронный ресурс]. – Режим доступа: https://developer.android.com/guide/components/fundamentals?hl=ru
3. Создание CRUD API на Express и MySQL: часть 1 [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/otus/blog/491408/
4. Создание CRUD API на Express и MySQL: часть 2 [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/otus/blog/491786/
5. Создание GET запроса при помощи библиотеки Retrofit [Электронный ресурс]. – Режим доступа: https://www.youtube.com/watch?v=sBCE\_hOFnQU
6. Кроссплатформенная разработка мобильных приложений [Электронный ресурс]. – Режим доступа: https://www.sam-solutions.com/blog/cross-platform-mobile-development/
7. Нативная разработка мобильных приложений [Электронный ресурс]. – Режим доступа: https://www.sam-solutions.com/blog/native-mobile-app-development/
8. Документация языка программирования Kotlin [Электронный ресурс]. – Режим доступа: https://kotlinlang.org/docs/home.html

# ПРИЛОЖЕНИЕ А

**Разметка активности входа**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".activities.LoginActivity">  
  
 <EditText  
 android:id="@+id/email\_login"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:hint="Почта" />  
  
 <EditText  
 android:id="@+id/password\_login"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/email\_login"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="30dp"  
 android:hint="Пароль"  
 android:inputType="textPassword" />  
  
 <Button  
 android:id="@+id/btn\_login"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/password\_login"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="40dp"  
 android:text="Войти" />  
  
 <TextView  
 android:id="@+id/register\_txt"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/btn\_login"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="50dp"  
 android:text="Создать аккаунт"  
 android:textSize="20sp" />  
</RelativeLayout>

# ПРИЛОЖЕНИЕ Б

**Разметка активности регистрация**

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".activities.RegistrationActivity">  
  
 <EditText  
 android:id="@+id/email\_registration"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:hint="Почта" />  
  
 <EditText  
 android:id="@+id/password\_registration"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/email\_registration"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="30dp"  
 android:hint="Пароль"  
 android:inputType="textPassword" />  
  
 <Button  
 android:id="@+id/btn\_registration"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/password\_registration"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="40dp"  
 android:text="Зарегистрироваться" />  
  
</RelativeLayout>

# ПРИЛОЖЕНИЕ В

**Разметка фрагмента профиль**

<?xml version="1.0" encoding="utf-8"?>  
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
  
 tools:context=".fragments.ProfileFragment">  
  
 <RelativeLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingBottom="80dp">  
  
 <TextView  
 android:id="@+id/user\_name\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="15dp"  
 android:text="Имя:"  
 android:textColor="#000"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/show\_user\_name\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="15dp"  
 android:layout\_toRightOf="@id/user\_name\_profile\_view"  
 android:textColor="#000"  
 android:textSize="18sp" />  
  
 <TextView  
 android:id="@+id/user\_surname\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/user\_name\_profile\_view"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="30dp"  
 android:text="Фамилия:"  
 android:textColor="#000"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/show\_user\_surname\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/user\_name\_profile\_view"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_toRightOf="@id/user\_surname\_profile\_view"  
 android:textColor="#000"  
 android:textSize="18sp" />  
  
 <TextView  
 android:id="@+id/user\_mail\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/user\_surname\_profile\_view"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="30dp"  
 android:text="Почта:"  
 android:textColor="#000"  
 android:textSize="20sp" />  
  
 <TextView  
 android:id="@+id/show\_user\_mail\_profile\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/user\_surname\_profile\_view"  
 android:layout\_marginLeft="30dp"  
 android:layout\_marginTop="30dp"  
 android:layout\_toRightOf="@id/user\_mail\_profile\_view"  
 android:textColor="#000"  
 android:textSize="18sp" />  
  
 <Button  
 android:id="@+id/btn\_change\_user\_data"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/show\_user\_mail\_profile\_view"  
 android:layout\_marginLeft="10dp"  
 android:layout\_marginTop="30dp"  
 android:backgroundTint="#ffa500"  
 android:text="Изменить"  
 android:textSize="10sp" />  
  
 <Button  
 android:id="@+id/btn\_show\_orders"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@id/show\_user\_mail\_profile\_view"  
 android:layout\_toRightOf="@id/btn\_change\_user\_data"  
 android:layout\_marginLeft="20dp"  
 android:layout\_marginTop="30dp"  
 android:backgroundTint="#000"  
 android:text="Мои заказы"  
 android:textSize="10sp" />  
  
 <Button  
 android:id="@+id/btn\_log\_out"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:backgroundTint="#000"  
 android:text="Выйти"  
 android:textSize="16sp" />  
 </RelativeLayout>  
  
</FrameLayout>