Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_____» _____ 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Мобильное приложение для системы складского учета автошин
тема

Руководитель	_____	доцент, канд. техн. наук	М. С. Медведев
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Т.С. Медведев
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, канд. техн. наук	М. С. Медведев
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2022

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«_____» _____ 2022 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Красноярск 2022

Студенту _____Медведеву Тимофею Сергеевичу_____

фамилия, имя, отчество

Группа __КИ18-06Б__ Направление (специальность) __09.03.01_____

номер код

_____Информатика и вычислительная техника_____

наименование

Тема выпускной квалификационной работы Мобильное приложения для системы складского учета автошин

Утверждена приказом по университету № _____ от _____

Руководитель ВКР М. С. Медведев, канд. техн. наук, доцент, кафедра ВТ_

инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР: сформулировать цели и задачи, провести анализ существующих аналогов в предметной области, реализовать мобильное приложение для сервиса «CloneService» на платформы ios и android.

Перечень разделов ВКР: анализ задания для выполнения и обзор аналогов, проектирование мобильного приложения, программная реализация.

Перечень графического материала: презентация в формате PowerPoint.

Руководитель ВКР _____ М. С. Медведев

подпись инициалы и фамилия

Задание принял к исполнению _____Т. С. Медведев

подпись, инициалы и фамилия студента

« ___ » _____ 2021 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Мобильное приложение для системы складского учёта автошин» содержит 51 страниц, исследовано 9 источников, использовано 35 рисунков, 2 приложения.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КЛИЕНТ, СЕРВЕР, ФРЕЙМВОРК, EXPO, REACT NATIVE.

Цель работы: разработка мобильного приложения для системы складского учёта автомобильных шин.

При выполнении данной работы был произведен обзор предметной области, задания на выпускную квалификационную работу, изучены существующие аналоги и сформированы требования, предъявляемые к мобильному приложению.

Объект работы – мобильное приложение, позволяющее выполнять операции по администрированию системы складского учета автомобильных шин с возможностью выгрузки данных на внешние сервисы онлайн-торговли.

Задачи:

− осуществить выбор программных средств моделирования и разработки мобильного приложения;

− выполнить моделирование разрабатываемого мобильного приложения;

− выполнить программную реализацию мобильного приложения для платформ iOS и Android;

− проанализировать полученные результаты работы.

В результате работы над ВКР было разработано и реализовано мобильное приложение для сервиса по хранению автошин, которое упрощает работу работникам склада

Содержание

ВВЕДЕНИЕ

В настоящее время часто возникает необходимость по администрированию собственного бизнеса, в частности магазина автошин. Мобильное приложение позволит с удобством управлять им и следить за сделками и товарами из любой точки мира.

Актуальностью данной темы является то, что существующие сервисы складского учета используют десктопные приложения и не предоставляют возможностей по использованию мобильных устройств. Внедрение данного функционала позволит сотрудникам склада выполнять типовые операции с системой по приему, выдаче товара с применением системы штрих-кодов.

Целью данной работы является разработка мобильного приложения для сервиса по хранению автошин с возможностью выгрузки данных для внешних сервисов онлайн-торговли.

1 Анализ задания для выполнения

Необходимо разработать кроссплатформенное мобильное приложение для сервиса по хранению автошин, реализующее следующий функционал:

- аутентификация;
- прием и учет продукции на складе (шины, диски);
- добавление контрагентов;
- создание ценовых правил;
- добавление информации о проведении покупок и продаж;
- работа с товаром через QR-коды.

Данное мобильное приложение разрабатывается для системы CloneService, модель системы – клиент-серверное взаимодействие (рисунок 1).

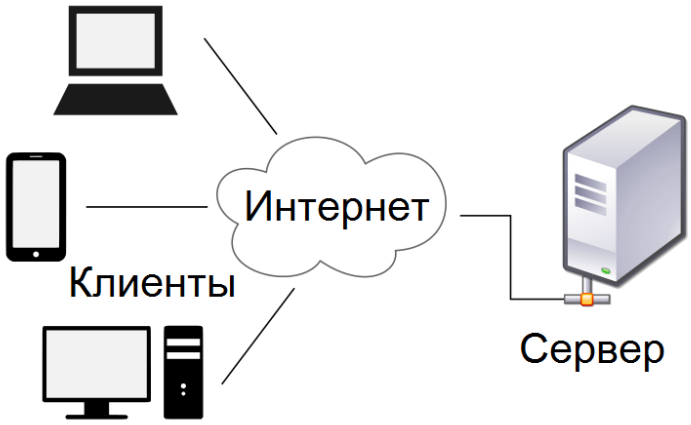

Рисунок 1 – Клиент-серверное взаимодействие

Серверная часть работает с использованием БД MySQL и Laravel для реализации общей логики сервера. В будущем API будет предоставляться сервисом CloneService.

1.1 Задание на выпускную квалификационную работу

Разработать кроссплатформенное мобильное приложение для iOS и Android для системы складского учета.

1.1.1 Назначение и цель создания мобильного приложения

1.1.1.1 Назначение мобильного приложения

Мобильное приложение – программное средство, предназначенное для работы на удобной для пользователей мобильной платформе и выполнения основных действий.

1.1.1.2 Цель создания мобильного приложения

Основной целью создания приложения является реализация программы для сервиса по хранению автошин, повышающего удобство использования сотрудниками склада. Использование мобильных устройств позволит задействовать функционал считывания QR-кодов и сократит время при проведении операции приемки и выдачи товаров.

1.1.2 Требования к структурной и функциональной части мобильного приложения

Функциональная структура мобильного приложения должна включать в себя:

- понятный и удобный интерфейс;
- загрузка, удаление, номенклатуры;
- загрузка, удаление, редактирование ценовых правил;

- сканирование qr-кодов;
- добавление информации о закупках на склад;
- добавление информации о совершении сделок.

1.2 Сервис CloneService

CloneService [1] является сервисом, который предоставляет другим компаниям готовое решение для администрирования магазином автошин, позволяет автоматизировать выгрузку товаров на площадках по продаже на вторичном рынке (avito, drom.ru) и предоставляет аналитику и отчет по продажам. главная страница изображена на рисунке 2.

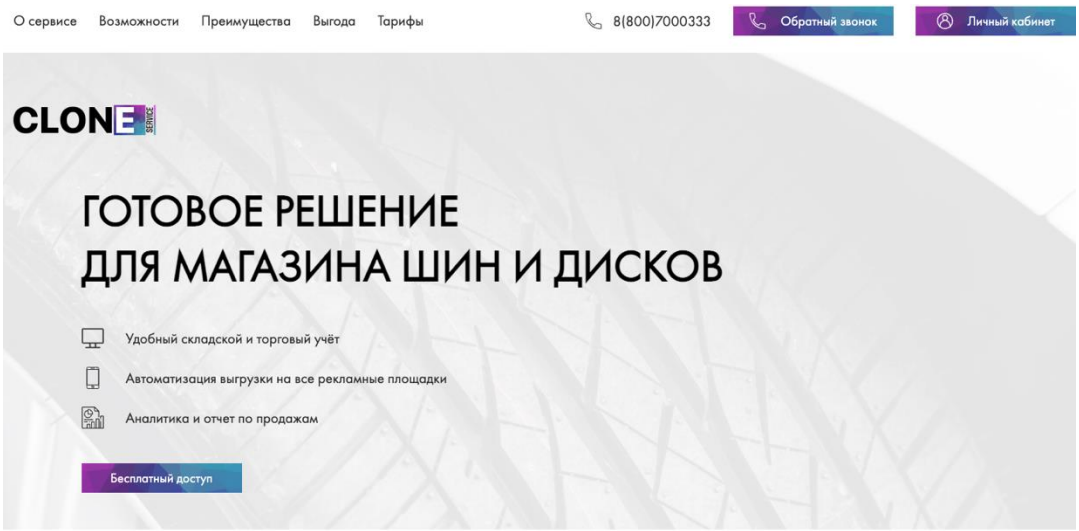

Рисунок 2 – Сайт сервиса CloneService

Сервис позволяет сравнивать цену на один товар на нескольких площадках. Возможен контроль за выручкой и чистой прибылью в реальном времени.

1.3 Анализ существующих аналогов

На данный момент существует несколько различных сервисов, предоставляющих функции по автоматизации бизнес-процессов, поэтому требуется рассмотреть существующие решения, чтобы выделить их преимущества и недостатки.

1.3.1 Бизнес.ру

Сервис «Бизнес.ру» [2] предоставляет клиенту автоматизацию бизнес-процессов. Сервис предоставляет: обработку заявок из интернет-магазина, программу учета договора, а также создание интернет-магазина. Также предоставляется управление процессами через мобильное приложение. На рисунке 3 изображена главная страница сервиса.

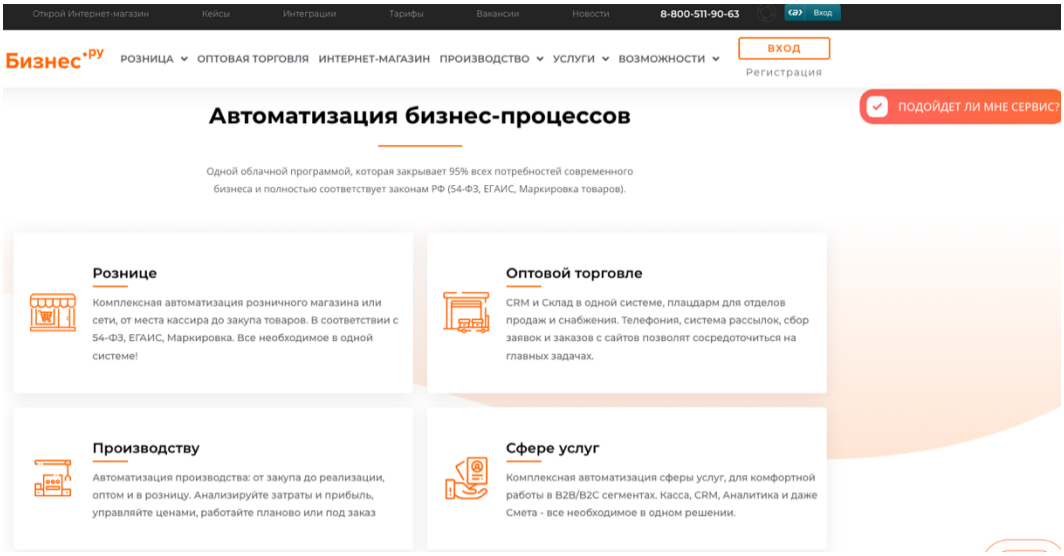

Рисунок 3 – Сайт сервиса «Бизнес продажи»

1.3.2 Система Bazon

Bazon [3] – сервис разработанный специально для продавцов б/у автозапчастей. Сервис предоставляет возможность автоматически выгружать товар на рекламные площадки, такие как Avito, drom.ru и auto.ru.

Также сервис предлагает штрихкодирование и адресное хранение, что позволит клиенту быстро и выполнить поиск товаров на складе.

Помимо этого, сервис ведет аналитику и отчетность по продажам и разграничивает права доступа для сотрудников. На рисунке 4 изображена главная страница сайта.

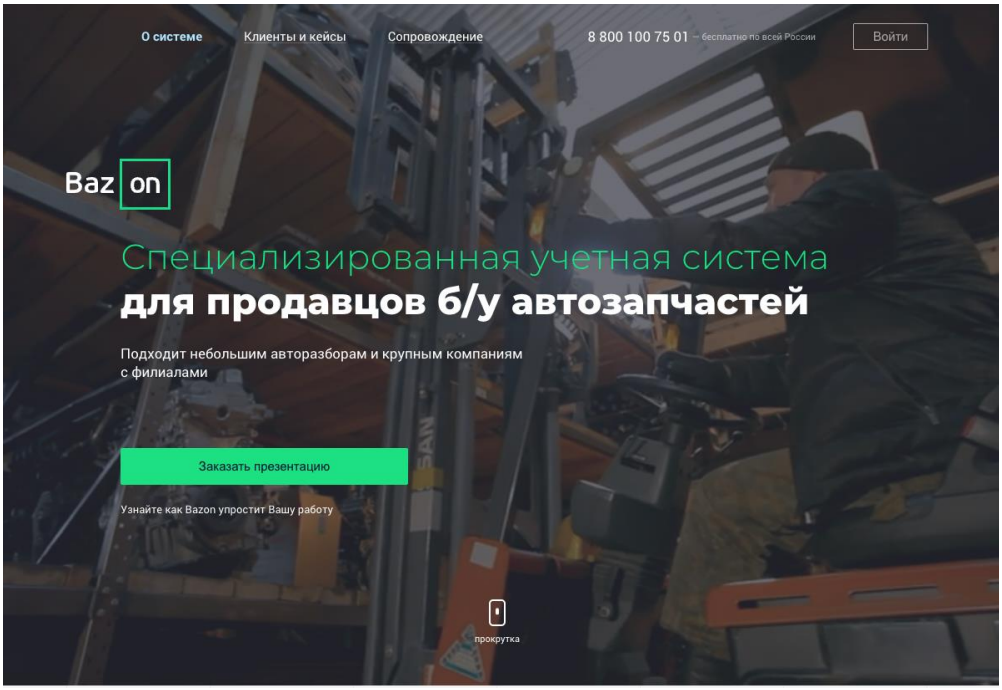

Рисунок 4 – Сайт сервиса Bazon

Мобильное приложение Bazon позволяет заполнять и просматривать информацию о товаре, использовать камеру телефона для сканирования штрих-кодов, а также отправку клиенту информации в мессенджере.

Вывод: были проанализированы сервисы по автоматизации бизнес-процессов и были выделены как достоинства и недостатки.

Бизнес-продажи – универсальный сервис, поэтому некоторые функции, которые есть у CloneService или Bazon, будут отсутствовать.

Ближайшим аналогом и конкурентом будет сервис Bazon так как он содержит схожий функционал. Поэтому в приложении необходимо будет реализовать работу с товаром, а также работу с камерой устройства.

1.4 Анализ технологий

1.4.1 Native-приложения

Native приложения – это приложения разработанные под конкретную платформу. Таким образом программа написанная под iOS будет работать только на данной операционной системе. Точно также и с программой, написанной под Android. У каждой операционной системы есть свой набор основных процедур, с помощью которого она осуществляет взаимодействие с разнообразными прикладными программами. API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

Нативные приложения разрабатываются с конкретным API, а следовательно, с конкретной операционной системой.

Преимущества нативного мобильного приложения:

- позволяет поддерживать высокую производительность;
- позволяет использовать аппаратное и программное обеспечение гаджета.
- установка приложения происходит из официальных магазинов;
- могут работать, как при наличии интернета, так и в офлайн-режиме;
- позволяет обрабатывать большое количество данных на стороне клиента.

Недостатками нативного мобильного приложения можно считать трудоемкость реализации проектов, что влияет на конечную стоимость продукта.

Также можно выделить, что охватывается только одна платформа и для разработки версии для другой ОС потребуется выделять дополнительные ресурсы.

1.4.2 Кроссплатформенные приложения

Кроссплатформенное программное обеспечение – программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе. После написания кода приложения его можно развернуть на разных устройствах и платформах, не беспокоясь о проблемах несовместимости.

Преимущества кроссплатформенной разработки:

- при разработке приложения, команда разработчиков может реализовывать новые функции на всех платформах, используя единый технологический стек;
- экономия бюджета проекта;
- охват более широкой аудитории.

Недостатки:

- не использует уникальные особенности платформ;
- медленная работа приложения.

1.5 Выбор средств разработки

1.5.1 React Native

React Native [4] – это фреймворк для разработки мобильных приложений операционные системы iOS и Android. React Native разработан корпорацией Facebook в 2015.

Фреймворк React Native позволяет разрабатывать мобильные приложения для iOS и Android с использованием языков TypeScript или JavaScript, а также использовать в двух приложениях значительную часть общего кода.

React Native имеет более низкую производительность, чем нативные решения, а его использование для сложных и нестандартных приложений затруднительно. В области бизнес-приложений React Native обеспечивает баланс между производительностью приложения и скоростью разработки.

1.5.2 Apache Cordova

Apache Cordova [5] – это среда разработки мобильных приложений с открытым исходным кодом. Она позволяет использовать стандартные веб-технологии – HTML5, CSS3 и JS для кроссплатформенной разработки. Приложения выполняются в оболочках, предназначенных для каждой платформы, и полагаются на соответствующие стандартам привязки API для доступа к возможностям каждого устройства, таким как датчики, данные и состояние сети.

Apache Cordova подходит для разработки, если:

- необходимо реализовать приложение на более чем одну платформу;
- необходимо развернуть веб-приложение, упакованное для распространения на различных порталах.

Приложение Cordova состоит из нескольких компонентов. На диаграмме (рисунок 5) представлена общая архитектура приложения Cordova. Приложение на Cordova состоит из блока, в который входят стандартные элементы для создания web приложения (HTML, css, js) и из плагинов Cordova, для работы с возможностями смартфона.

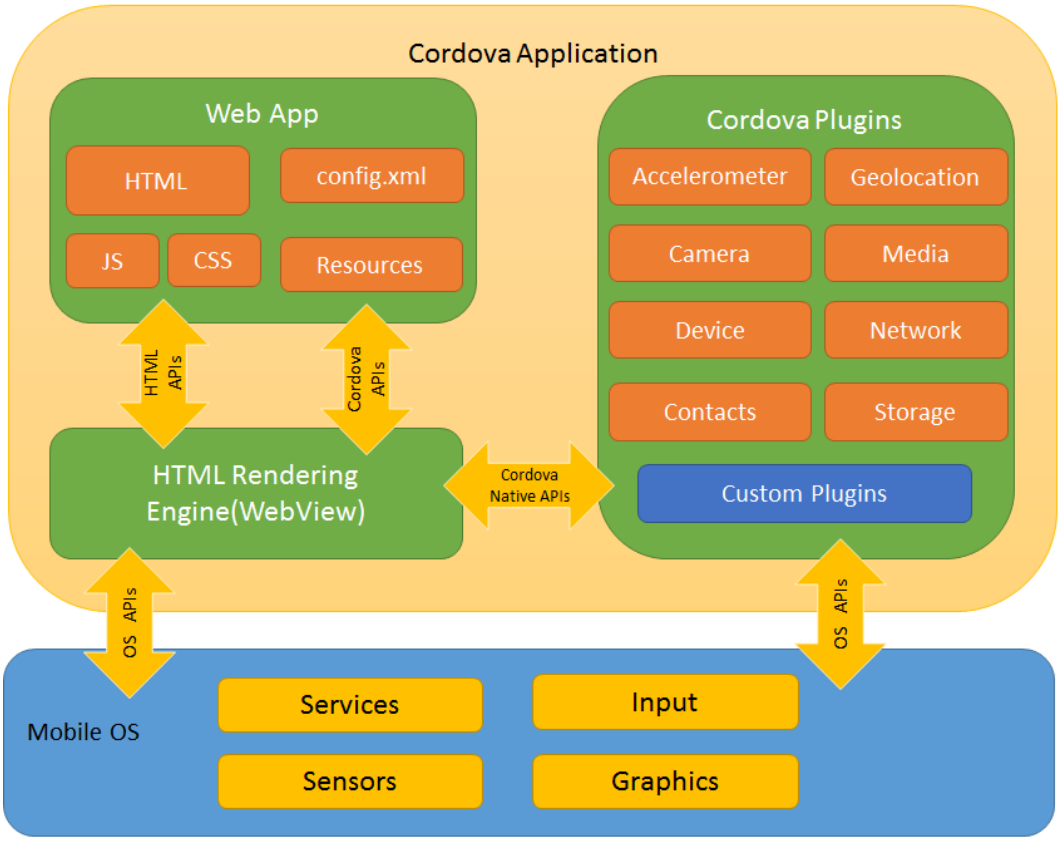

Рисунок 5 – Архитектура приложения Cordova

1.5.3 Expo

Expo [6] – это фреймворк для быстрого развертывания нативных React приложений. Expo предоставляет слой команд поверх React Native API, чтобы ими было удобнее пользоваться и управлять. Он также предоставляет инструменты, которые упрощают начальное создание и тестирование React Native приложений. Предоставляет разработчику компоненты пользовательского интерфейса и сервисы, которые обычно доступны только при установке сторонних нативных компонентов React.

1.5.4 Flutter

Технологию Flutter [7] создали инженеры компании Google. Был собран весь позитивный и негативный опыт всех существующих кроссплатформенных решений.

Приложения на Flutter пишутся на единой кодовой базе, что позволяет проще тестировать и отлаживать.

Во Flutter используется язык Dart, в качестве альтернативы js.

В отличии от Cordova, Ionic и других веб-фрейморков, Flutter не рендерит пользовательский интерфейс на WebView. За визуализацию приложения отвечает собственный графический движок и несколько пакетов кастомизируемых виджитов.

1.5.5 TypeScript

TypeScript [8] – язык программирования, представленный Microsoft в 2012 году и позиционированный как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript отличается от JavaScript возможностью явного статического назначения типов, поддержкой использования полноценных классов (как и в традиционных объектно-ориентированных языках), а также поддержкой подключения модулей, что позволяет повысить скорость разработки, облегчить читаемость, рефакторинг и повторное использование кода, помочь осуществлять поиск ошибок на этапе разработки и компиляции.

Код TypeScript преобразуется в JavaScript, который выполняется везде, где выполняется JavaScript: в браузере, на Node.js и Deno.

Язык широко поддерживается множеством IDE, удобен в использовании и является перспективным. Может быть использован в React Native для разработки под iOS и Android.

1.6 Вывод по разделу

В результате анализа были сформулированы четкие требования к разрабатываемой, системе, было решено, что разрабатываемая система, будет получать информацию от сервера для отображения ее в удобном для пользователя виде в мобильном приложении.

Так же был произведен анализ существующего решения и выявлены его недостатки. Был выделен перечень функций, отсутствующий в нем, которые необходимо будет реализовать.

Для того, чтобы упростить разработку кроссплатформенного мобильного приложения сразу под две платформы, был сформирован стек используемых технологий, в частности TypeScript как основной язык, React Native и Expo как основные средства, позволяющие разработку на две платформы.

2 Проектирование мобильного приложения

Общий алгоритм взаимодействия мобильного приложения и сервера представлен на рисунке 6.

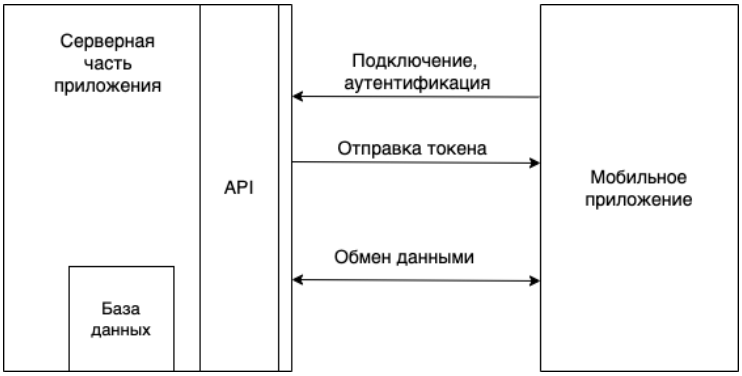

Рисунок 6 – Схема взаимодействия мобильного приложения и сервера

Клиент подключается к серверу, при этом высылая данные для аутентификации. Клиент может запросить данные, после чего сервер выдает клиенту сформированный JSON. Локально, на мобильном устройстве, клиент может производить операции с данными (добавлять, изменять, удалять). Измененные данные клиент отправляет на сервер, где принимается решение, что с ними делать: занести в БД или проигнорировать. Сервер в это время также может отправить клиенту новые данные, если такие имеются.

2.1 Формат запроса данных

Механизм формирования запроса является типовым для данного типа приложений и представлен на рисунке 7. Для создания http запроса будет использоваться axios.

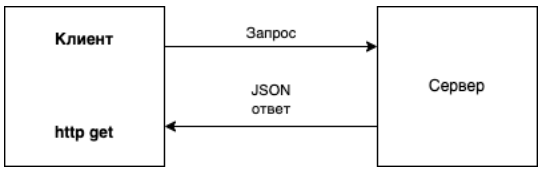

Рисунок 7 – Формат запроса

Мобильное приложение отправляет на сервер url-запрос вида «http://clone.6171.ru/price_rule/save», на что сервер отвечает в формате JSON.

2.2 Структура мобильного приложения

Приложение состоит из множества экранов, написанных на TypeScript с использованием стилей css, которые позволяет использовать фреймворк React Native с использованием обертки expo. Согласно этому любой экран должна содержать:

- файл-класса <screenname>.tsx, в котором описываются методы, используемые на этом экране, разметка на экране, стилевые изменения;
- файл-класса <interfacename>.ts, в котором описывается модель данных
- файл-класса <screenname><service>.ts, в котором описываются методы для http-запросов.

Пример файловой структуры приложения приведен на рисунке 8.

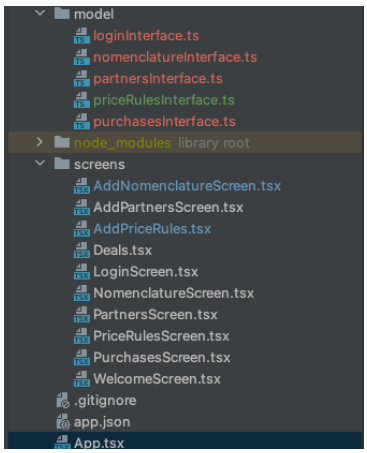

Рисунок 8 – Пример файловой структуры приложения

2.3 Маршрутизация

Навигация в приложении будет прописана в файле «navigation.tsx». Необходимо будет использовать два типа навигации. Один для перехода между экранами модуля и другой для перехода между самими модулями. На рисунке 9 приведена схема маршрутизации в мобильном приложении.

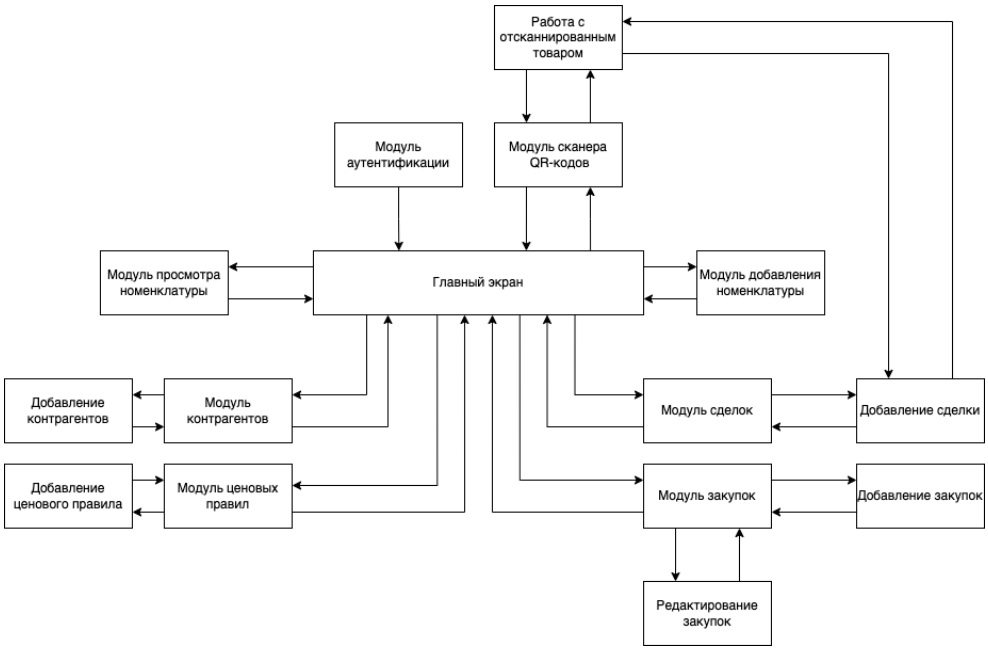

Рисунок 9 – Схема маршрутизации мобильного приложения

2.4 Модуль аутентификации

В данном модуле происходит вход пользователя в систему. После ввода данных в поля «Login» и «Password» приложение отправляет на сервер запрос на аутентификацию, на что сервер отвечает либо согласием, либо ошибкой. Если от сервера пришел положительный ответ, приложение переходит на главный экран и слева появляется меню навигации. Помимо перехода на главный экран, в приложение приходит токен, который будет использоваться в качестве id пользователя, необходимого в дальнейших запросах.

UML диаграмма последовательности изображена на рисунке 10.

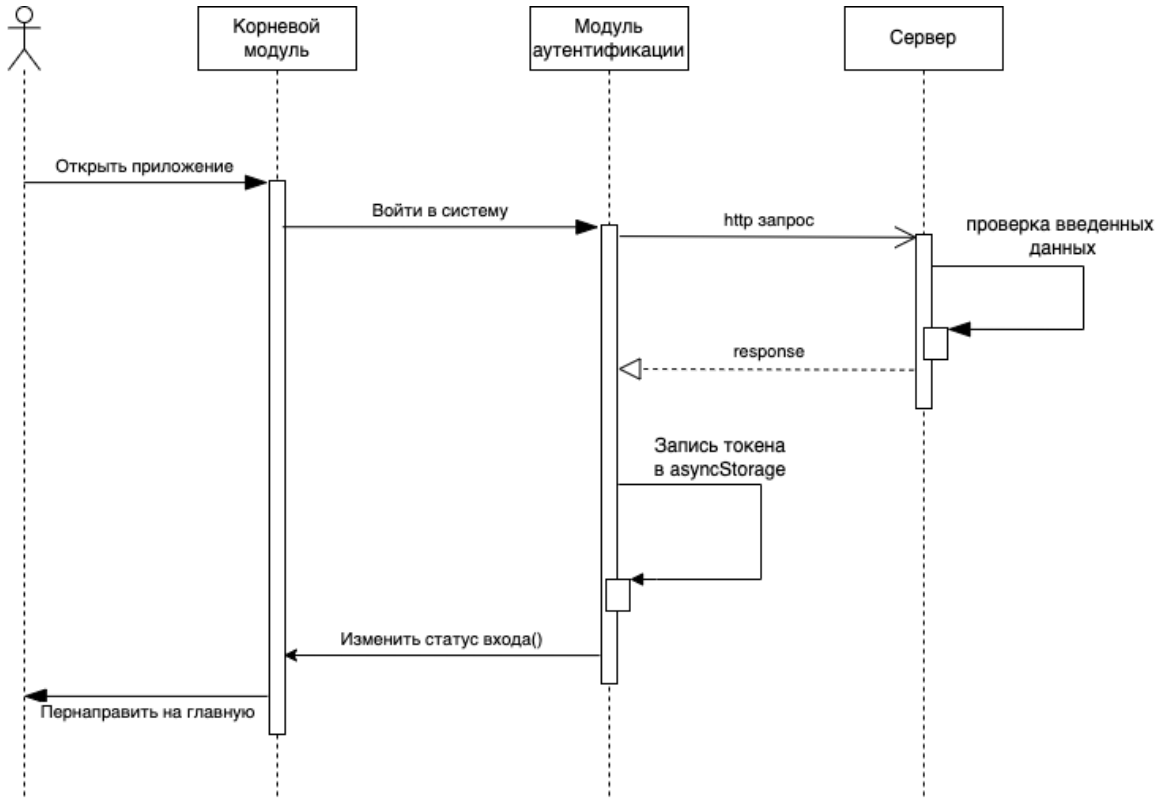

Рисунок 10 – UML диаграмма последовательности аутентификации

2.5 Главный экран

На главном экране будет изображен логотип сервиса, а также боковое меню для навигации. Оно является основным средством навигации для всего приложения после входа в система. Через него можно перейти на экраны: контрагентов, номенклатуры, сделок, создания номенклатуры, закупок, а также ценового правила.

2.6 Модуль ценовых правил

Ценовое правило – это механизм для определения и выполнения динамических условий, от которых будет зависеть регулирование цен.

В него входят: название, процент, минимальная и максимальная цена.

Объект «Ценовое правило» содержит поля:

- name: string – название правила;
- percent: number – процент;
- min: number – минимальная сумма;
- max: number – максимальная сумма.

Ценовое правило используется в собственном модуле, а также в модуле контрагентов. Оно предназначено для коррекции цен при закупке товара.

Приложение должно реализовать такие функции взаимодействия с ценовым правилом: как создание, удаление, просмотр и редактирование. Диаграмма прецедентов представлена на рисунке 11.

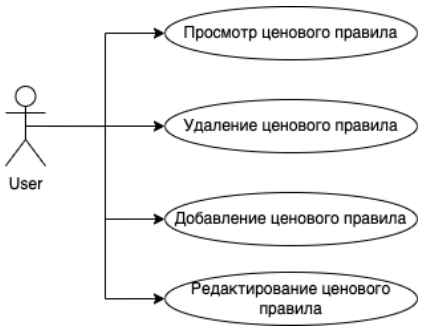

Рисунок 11 – Диаграмма прецедентов модуля ценовых правило

2.7 Модуль контрагентов

Контрагент – это партнер компании, который может быть физическим лицом, юридическим лицом и индивидуальным предпринимателем.

Физическое лицо содержит поля: имя, фамилия, отчество, номер телефона, электронная почта и ценовое правило.

Индивидуальный предприниматель содержит аналогичные поля, но к ним добавляются: адрес, ИНН, банк, БИК и номер расчетного счета.

Юридическое лицо (организация) содержит аналогичные поля, но к ним добавляется КПП.

Контрагент юр лица содержит поля:

- type: string;
- name: string;
- surname: string;
- parentname: string;
- phone: string;
- email: string;
- inn: string;
- bank: string;
- pay_number: string – номер счета;
- pricerule_id: string – ценовое правило применяемое к контрагенту;
- kpp: string – номер кпп;

- bik: string – банковский идентификационный счет;
- address: string.

Приложение должно реализовать функции добавления, просмотра и удаления контрагента. Диаграмма прецедентов представлена на рисунке 12.

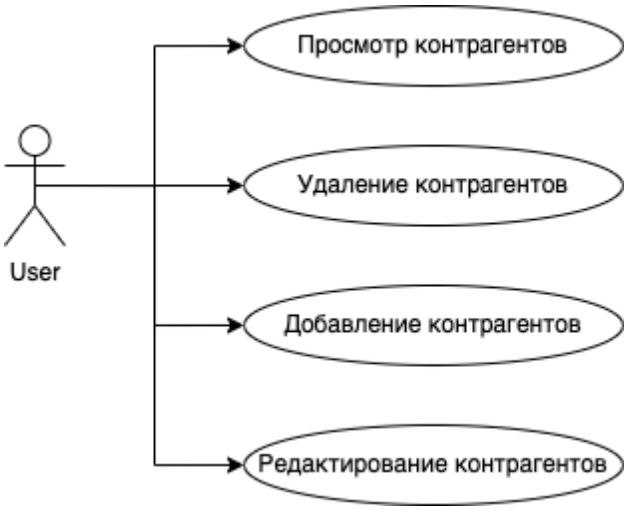

Рисунок 12 – Диаграмма прецедентов модуля контрагентов

2.8 Модуль номенклатуры

В модуль номенклатуры входит определения информации о товаре (шина, диски и т.д.).

В объект шина входит информация о производителе, модели, ширине, профиле, диаметре, индекс, год производства и описание.

Пример объекта «Шина» для номенклатуры:

- brand: string;
- model: string
- width: number;
- profile: string;
- diameter: number;
- index: string;

- year: number;
- description: string.

Приложение должно реализовать функции просмотра и добавления номенклатуры. Диаграмма прецедентов представлена на рисунке 13.

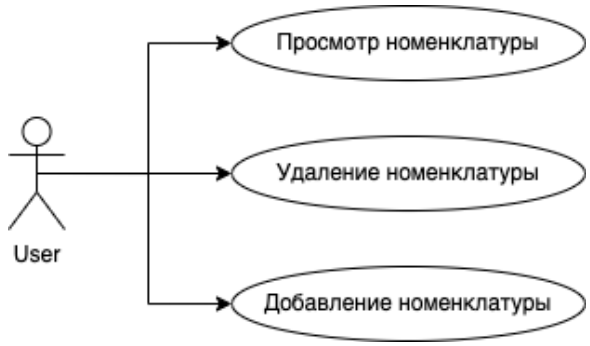

Рисунок 13 – Диаграмма прецедентов модуля номенклатуры

2.9 Модуль работы с товаром

Работа с товаром это одна из важных функций сервиса, которая позволяет выставить товар на продажу на популярных площадках.

Для выставления на продажу используем данные хранимых на складе товаров, выставляем цену, количество, указываем площадку, на которую будет выставлен товар, а также прикладываем изображение товара.

Пример объекта «шина» для выставления на продажу:

- item_id: string – id товара;
- amount: number;
- status: string;
- price: number;
- store: string.

На рисунке 14 представлена диаграмма прецедентов модуля работы с товаром

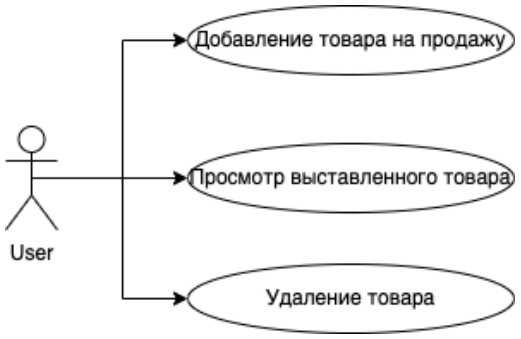

Рисунок 14 – Диаграмма прецедентов модуля работы с товаром

2.10 Модуль закупок

Данный модуль предназначен для заполнения информации о закупках товара на склад.

Для заполнения закупки необходимо ввести данные об организации, поставщике, складе, сумме, количестве товара, а также указать тип документа. Далее необходимо будет выбрать товар, который мы хотим купить и подставляем ранее заполненную номенклатуру.

Помимо этого, будет произведен расчет наценки на товар, относительно указанной стоимости и ценового правила.

Пример объекта закупка:

- nomenclature_id: string – id выбранной номенклатуры;
- partner_id: string – id выбранного контрагента;
- organization: string;
- supplier: string;
- amount: number;
- stock: string – склад на который придет товар;
- sum: number – стоимость закупки;

- add_price: number – будущая наценка на товар;
- document: string.

Приложение должно реализовывать следующие операции над закупками: добавление, просмотр, удаление и редактирование.

Диаграмма прецедентов модуля представлена на рисунке 15.

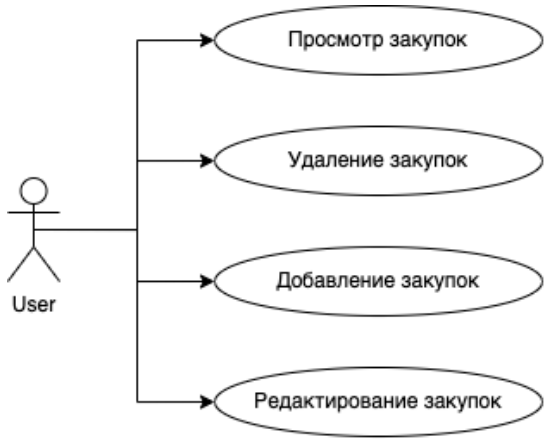

Рисунок 15 – Диаграмма прецедентов модуля закупок

2.11 Модуль сделок с клиентами

В данном модуле будет происходить просмотр и оформление информации о сделках.

Для заполнения сделки необходимо выбрать товар, лежащий на складе, указать количество, цену и покупателя.

Пример объекта сделка:

- article: string – артикул сделки;
- partner: string – покупатель;
- item_id: string – id товара;
- price: number – сумма оплаты;
- state: string – состояние товара;
- amount: number – количество продаваемого товара.

Приложение должно реализовывать возможность создания и просмотра сделок.

На рисунке 16 представлена диаграмма прецедентов модуля сделок.

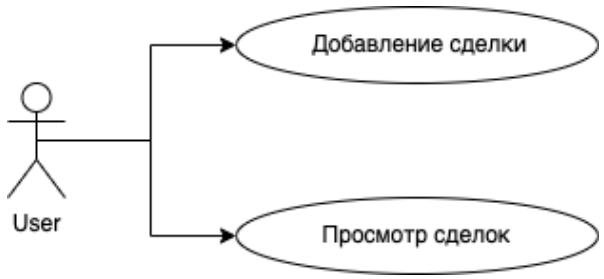

Рисунок 16 – Диаграмма прецедентов модуля сделок

2.12 Прототип серверной части приложения

Для проверки работы http-запросов будет написан прототип серверной части приложения. В качестве фреймворка был выбран NestJs, так как он удобен в создании MVP (минимально жизнеспособный продукт), в качестве базы данных была выбрана PostgreSQL, запущенная в docker-контейнере. Для работы с базой данных, а конкретно для создания таблиц, миграций и работы с сущностями, использовалась библиотека MikroORM.

На рисунке 17 представлена ER-диаграмма базы данных.

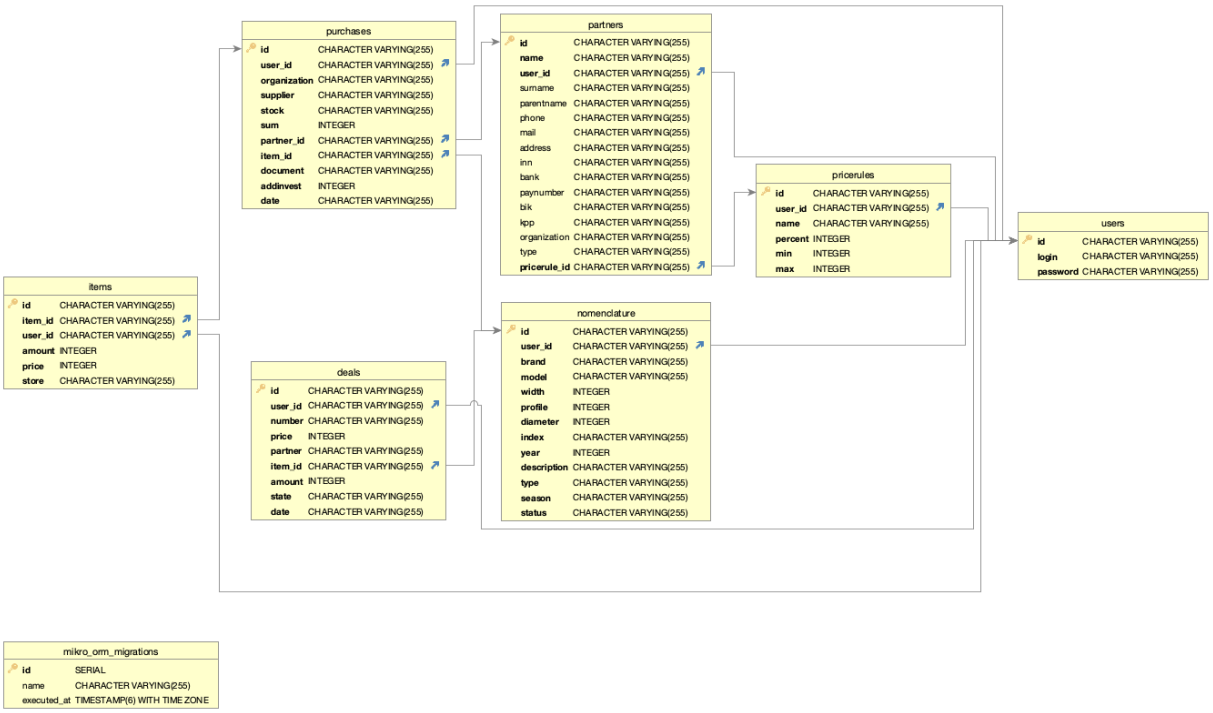

Рисунок 17 – ER-диаграмма базы данных

Для обработки запросов были созданы модули, содержащие контроллеры, которые отвечают за свой адрес запроса и в зависимости от типа запроса (get, post, patch, delete) вызывают функции из сервисов для работы с базой данных.

2.13 Вывод по второму разделу

Во втором разделе пояснительной записки был описан клиент-серверный подход и алгоритм взаимодействия сервера с клиентом.

Была определена маршрутизация в приложении.

Был реализован прототип серверной части приложения.

Также были выделены необходимые функции, которые должно реализовать приложение:

– вход в систему, возможность перехода в любой модуль;

– создание, удаление и просмотр контрагентов;

– добавление и просмотр номенклатуры по поиску;

– возможность создания, просмотра и удаление ценовых правил;
– возможность работы с товаром;
– работа с QR-кодами.

3 Программная реализация

В процессе программной реализации были решены следующие задачи:

- выполнена верстка страниц;
- реализованы основные функции для клиент-серверного взаимодействия;
- реализованы методы, подходящие как для ios так и android устройств;
- реализованы модули ценовых правил, контрагентов;
- реализован сканер QR-кодов;
- реализованы модули номенклатуры, сделок и закупок;
- реализована навигация между модулями.

3.1 Аутентификация

При открытии приложения идет сверка значения, пользователя: входил ли он раньше в систему или еще нет. В случае, если пользователь не входил в систему, откроется окно входа, для ввода логина и пароля.

При отправке формы идет проверка на заполненные поля, если какое-то поле не заполнено, выведется соответствующее предупреждение. Далее отправляется запрос на сервер, в случае успешного запроса, вернётся токен пользователя, который будет записан в хранилище. В качестве хранилища использовался AsyncStorage. После записи токена в хранилище, isAuth принимает значение true и происходит переход на главный экран.

Экран входа в систему изображен на рисунке 18.

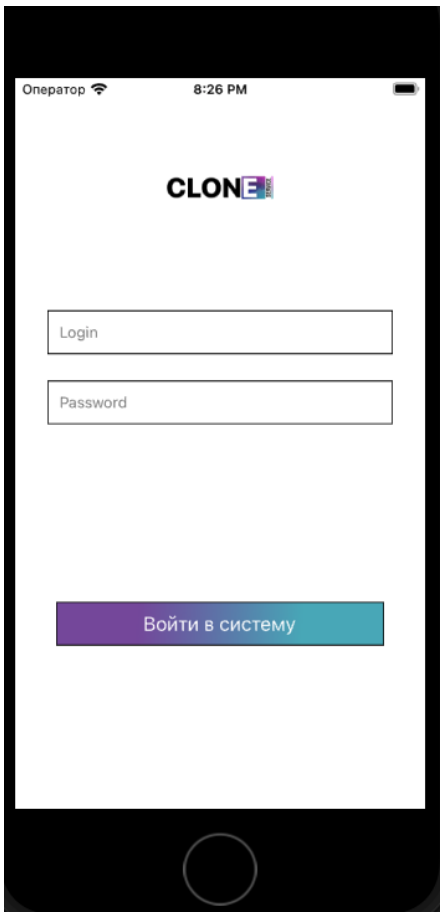

Рисунок 18 – Экран входа в систему

3.2 Главный экран приложения

После входа в систему откроется главный экран с приветствием, также откроется доступ к боковому меню навигации. Главный экран представлен на рисунках 19 и 20.

Навигация между модулями осуществляется при помощи бокового меню.

Для реализации навигации использовались две библиотеки: react-navigation/drawer и react-navigation/native.

DrawerNvigator используется для создания бокового меню, а StackNavigator для перехода между экранами внутри модуля.

Код файла navigation.tsx приведен в приложении А.

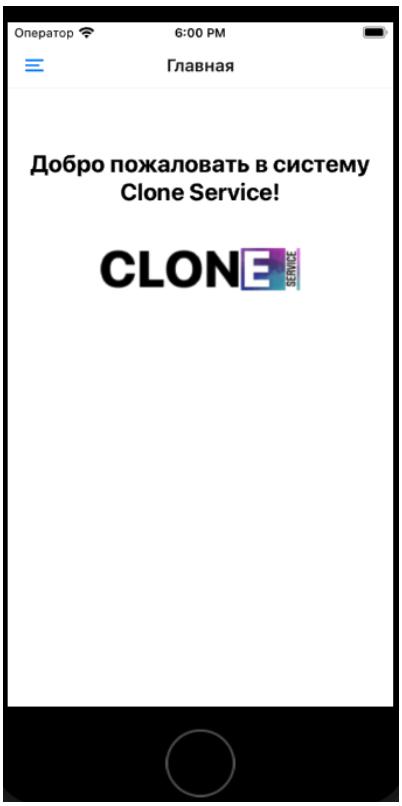

Рисунок 19 – Экран при входе в систему

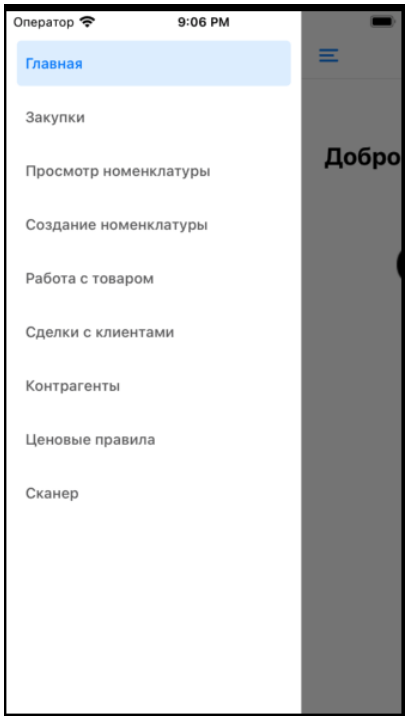

Рисунок 20 – Навигация в приложении

3.3 Модуль ценовых правил

Код модуля ценовых правил приведён в Приложении Б.

При открытии ценовых правил, загружаются уже созданные ценовые правила, они загружаются при get–запросе на сервер, функция getRules(). Полученные данные записываются в массив объектов data.

Ценовое правило можно удалить или отредактировать.

Для отображения списка был создан блок Item, которые заполняется данными из data выводится при помощи FlatList.

Также отображена кнопка «добавить», которая перенаправит на экран с созданием ценового правила (рисунок 21).

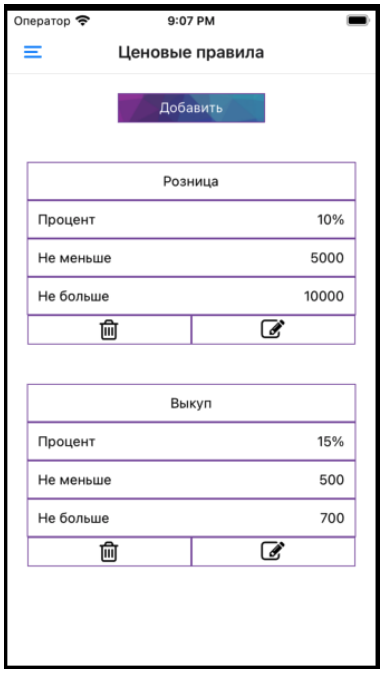

Рисунок 21 – Ценовые правила при открытии страницы

Для оформления кнопки «Сохранить» (рисунок 22) использовался градиент. В React Native он является не стилем css, а формой на странице наряду с Text или View и требует установки соответствующей библиотеки. Так как я использую expo, то для моего проекта подойдет только библиотека expo-linear-gradient.

Рисунок 22 – Экран создания ценового правила

Редактирование уже созданного правила происходит на экране добавления. Происходит передача дочернему компоненту выбранного объекта, также меняется флаг, отвечающий за последующий запрос: post/patch. Помимо этого, меняется заголовок и текст кнопки.

При удалении, на сервер отправляется delete–запрос содержащий id правила.

3.4 Модуль контрагентов

При открытии модуля контрагентов можно увидеть схожий дизайн с модулем ценовых правил. После открытия страницы происходит get–запрос на сервер на получение таблицы контрагентов. При нажатии на контрагента его можно выделить или при нажатии на кнопку удалить, будет произведен delete–запрос на сервер по id контрагента. На рисунке 23 представлен экран с выводом контрагентов, а на рисунке 24 экран с добавлением контрагента.

В зависимости от типа контрагента выводится информация только с его полями и если контрагент является ип или юр лицом, то поле ФИО будет заменено на название организации.

На экране добавления, есть выбор: добавить контрагента или добавить и вернуться на предыдущий экран. Если ответ с сервера не выдает ошибку, то запрос прошел удачно и можно продолжить добавлять контрагентов или вернуться на экран просмотра контрагентов, в зависимости от нажатой кнопки. В ином случае будет выведено предупреждение.

Для выбора ценового правила для контрагента используется Picker, в который помещаются правила после get–запроса. На экране показано название правило, а при отправки контрагента на сервер, отправиться pricerule_id.

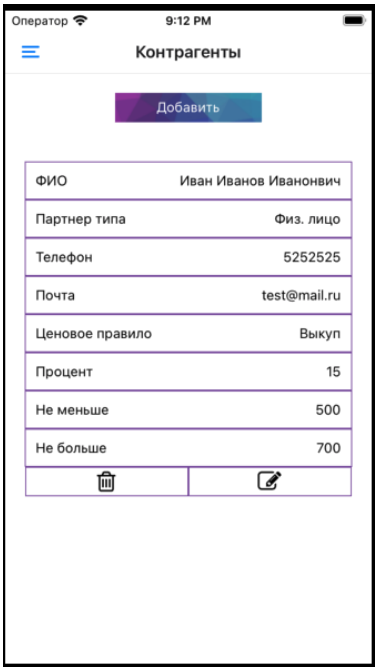

Рисунок 23 – Экран модуля «Контрагенты»

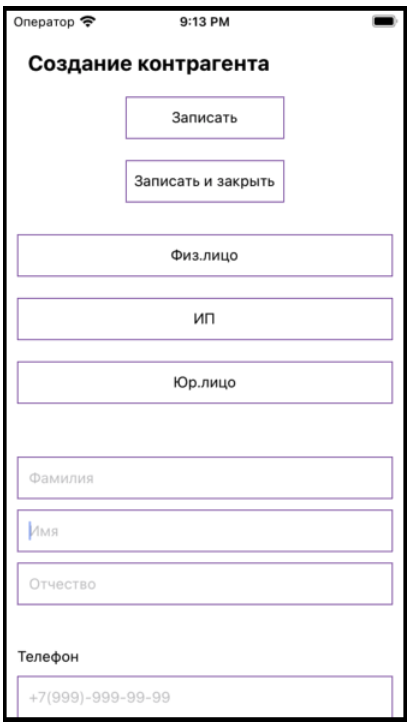

Рисунок 24 – Экран создания контрагента

3.5 Модуль номенклатуры

Этот модуль разделен на две отдельные части, в одной происходит просмотр номенклатура, когда в другой только ее добавление. Переход осуществляется исключительно через боковое меню.

При открытии модуля просмотра номенклатуры, будет произведен get-запрос. Полученные данные придут в формате json. Во время запроса происходит анимация загрузки, которая завершается после получение ответа.

Реализована возможность добавления и просмотра номенклатуры (рисунки 25-27).

Формы заполнения номенклатуры по товару, являются дочерними компонентами и меняются в зависимости от типа товара. При заполнении всех полей и нажатии кнопки отправить, будет произведен post-запрос.

Для выбора года был использован выпадающий список, для него был создан отдельный компонент. В него передается массив с датой, переменная в которую

записывается время, а также функции на запись и смены состояния выпадающего списка.

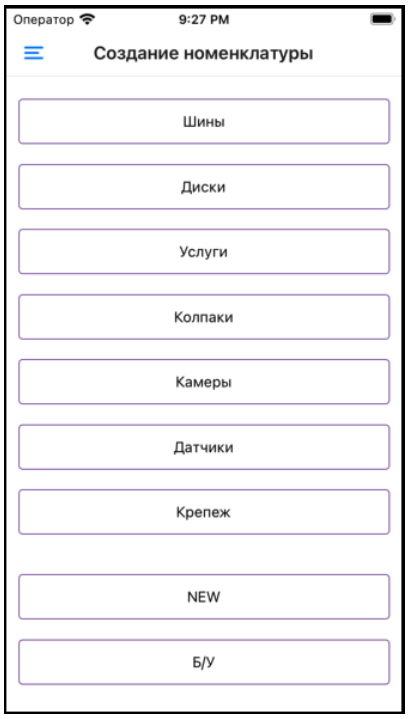

Рисунок 25 – Добавление номенклатуры (выбор типа товара)

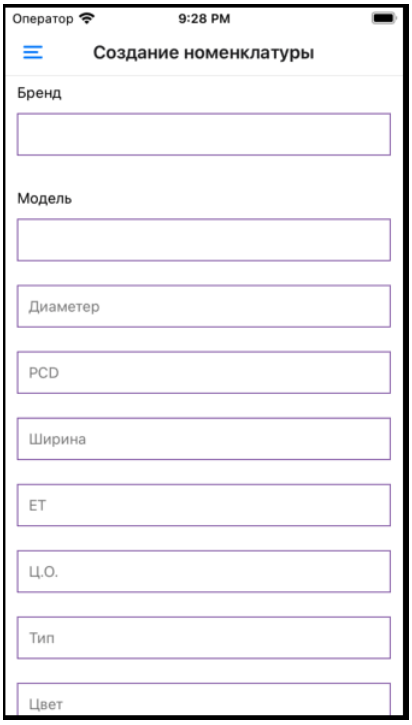

Рисунок 26 – Добавление номенклатуры (заполнение формы колпаков)

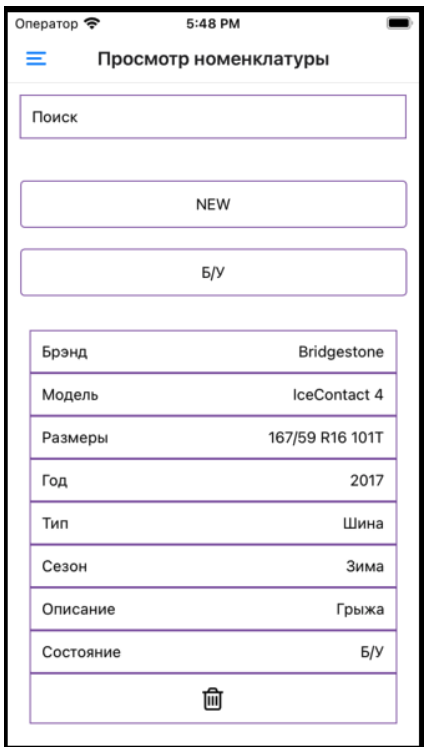

Рисунок 27 – Просмотр номенклатуры

3.6 Модуль работы с товаром

В этом модуле происходит выставление товара на продажу, необходимо выбрать доступный товар на складе, указать количество, цену за которую мы хотим продать и платформу, на которой необходимо будет выставить.

Для загрузки изображения использовался ImagePicker из библиотеки expo, которая даёт доступ к фотогалерее. Помимо установки библиотеки необходимо добавить ее в список плагинов. Экран модуля изображен на рисунках 28 и 29.

При открытии модуля происходит get-запрос на получение данных о выставленных шинах.

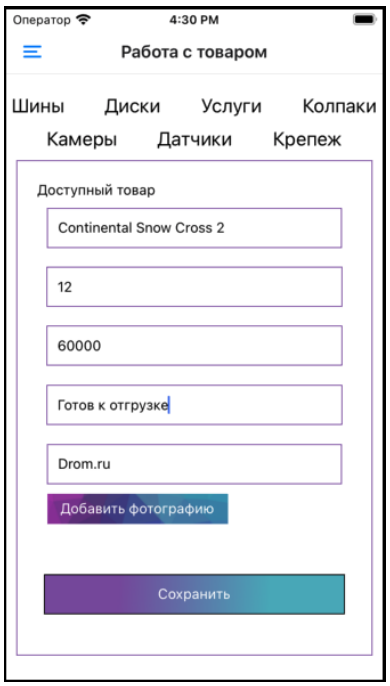

Рисунок 28 – Форма отправки товара

Рисунок 29 – Отправленный на продажу товар

3.7 Модуль считывания QR-кода

Данный модуль имеет практическую пользу для клиента сервиса и его работников на складе. Так как сервис предоставляет работу на складе, то необходимо получить информацию по товару на стеллаже, а также пометить его следующее состояние «убрать со склада» или «добавить в сделку».

Для работы с QR-кодами была выбрана библиотека expo-barcode-scanner [9], так как она совместима со всеми платформами и позволяет обрабатывать большое количество форматов штрих кодов. В таблице 1 показаны некоторые доступные форматы по платформам.

Таблица 1 – доступные форматы штрих кодов

Формат штрих кода	**IOS**	**Android**
qr	Да	Да
aztec	Да	Да
codabar	Да	Да
code39	Да	Да
code93	Да	Да
code128	Да	Да
code39mod43	Да	Нет
datamatrix	Да	Да
ean13	Да	Да
ean8	Да	Да
interleaved2of5	Да	Использовать itf14
itf14	Да	Да
maxicode	Нет	Да

На данный момент было принято решение, что QR-код содержит ссылку на товар. При считывании, получаем ссылку на товар и отправляем get–запрос. Полученная информацию в формате JSON передается на другой экран (рисунок 30).

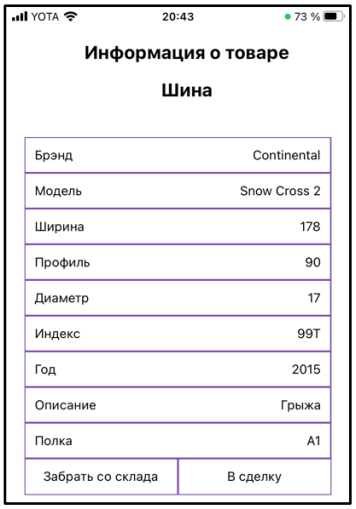

Рисунок 30 – Информация о товаре

При самом первом открытии модуля в приложении, произойдет запрос на доступ к камере. При следующих открытиях модуля запрос не будет появляться на экране.

3.8 Модуль сделок с клиентами

При открытии модуля будет выведена информация по всем сделкам (рисунок 31). Произойдет get–запрос и будут выведены такие поля как артикул, дата совершения сделки, сумма, контрагент, текущее состояние товара и его количество.

При нажатии на кнопку «добавить сделку» произойдет переход на экран с формой заполнения (рисунок 32). Также на этот экран можно будет попасть при сканировании товара. Если переход был через экран с просмотром закупок, то необходимо будет указать какой товар мы продаем, для этого после нажатия на поле «товар», произойдет get–запрос на сервер и откроется выпадающий список.

После нажатия кнопки «Добавить» произойдет post–запрос на сервер, на котором запишется дата совершения сделки.

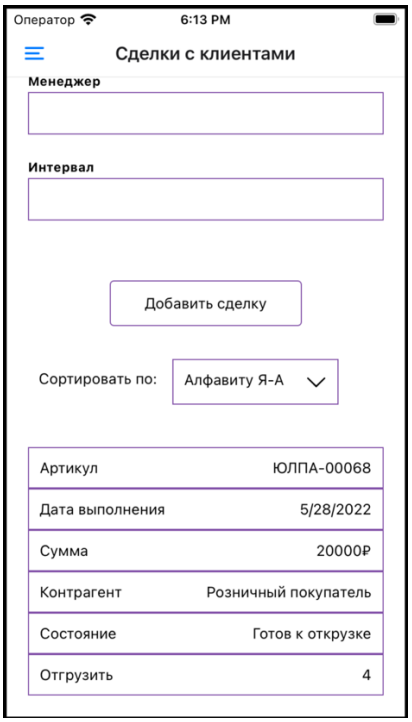

Рисунок 31 – Экран сделок с клиентами

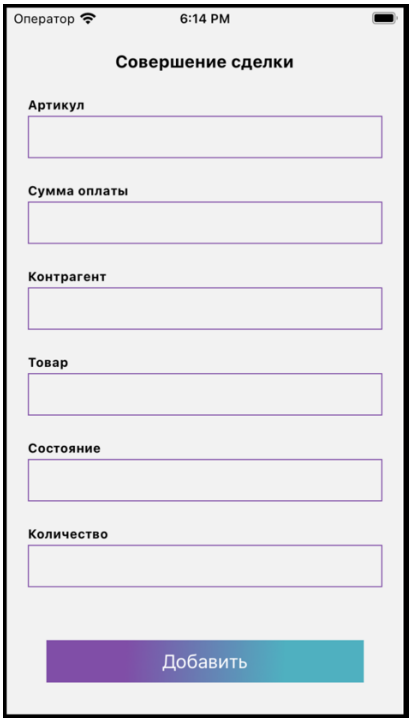

Рисунок 32 – Экран создания сделки

3.9 Модуль закупок

При открытии модуля будет выведена информация по всем закупкам (рисунок 33). Будет выведены такие поля закупок как: дата совершения закупки, поставщик, организация, склад, сумма, контрагент и данные о товаре.

При нажатии кнопки «добавить» произойдет переход на экран заполнения формы закупки (рисунок 34). При выборе товара, произойдет get–запрос на получение всей заполненной номенклатуры, при удачном запросе появиться выпадающий список с запоенной информацией о товарах. Аналогично происходит выбор контрагента. После ввода суммы закупки будет посчитана будущая применяемая наценка на товар. Для этого указанная сумма будет умножена на процент в ценовом правиле контрагента. После нажатия кнопки «отправить», будет произведен post–запрос, содержащий поля закупки в формате json.

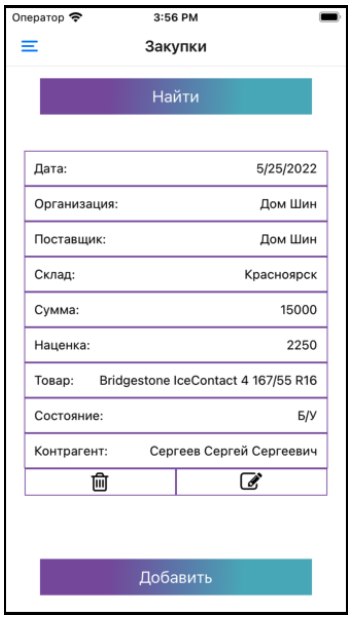

Рисунок 33 – Просмотр закупок

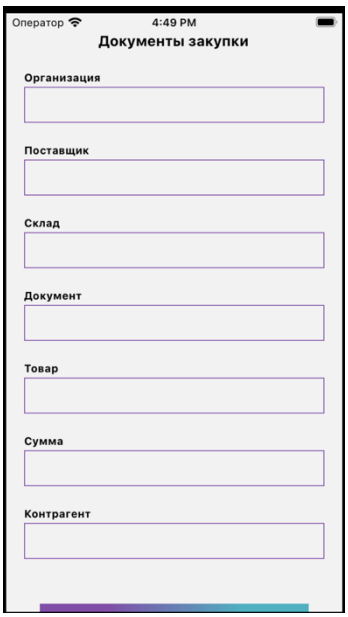

Рисунок 34 – Форма заполнения закупки

При нажатии на кнопку редактирования, мы перейдем на экран, где сможем изменить некоторые данные, а также посмотреть статус по оплате данной закупки (рисунок 35).

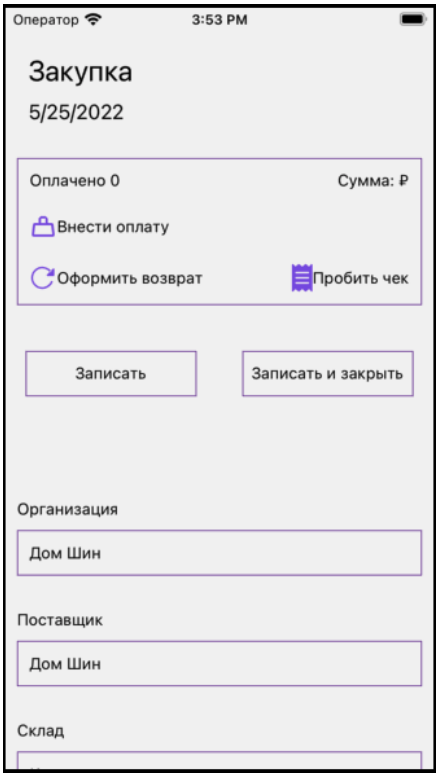

Рисунок 35 – Редактирование закупок

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была изучена предметная область и существующие на данный момент аналоги.

После изучения аналогов был сформулирован ряд требований, предъявленных к мобильному приложении. На основе сформулированных требований были определены технологии разработки. Мобильное приложение было написано с использованием фреймворков React Native и Expo.

В результате было разработано приложение под операционные системы iOS и Android, которое позволяет управлять сервисом по хранению и оборотом автошин.

Была реализована верстка и логика для необходимых модулей. А конкретно были реализованы модули: аутентификации, ценовых правил, контрагентов, закупок, сделок и сканера QR-кодов.

В дальнейшем требуется расширение функционала модулей закупок и сделок с клиентами, а конкретнее, внесении или получение оплаты. Также необходимо будет улучшить интерфейс, сделав его адаптивным.

СПИСОК СОКРАЩЕНИЙ

БД – база данных

JSON – JavaScript Object Notation

TS – TypeScript

JS – JavaScript

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. CloneService [Электронный ресурс]: – Режим доступа: http://cl.6171.ru/
2. Бизнес.ру [Электронный ресурс]: – Режим доступа: https://online.business.ru/
3. Сервис Bazon [Электронный ресурс]: – Режим доступа: https://probazon.ru/
4. React Native [Электронный ресурс]: – Режим доступа: https://reactnative.dev/
5. Apache Cordova [Электронный ресурс]: – Режим доступа: https://cordova.apache.org/
6. Expo [Электронный ресурс]: – Режим доступа: https://expo.dev/
7. Flutter [Электронный ресурс]: – Режим доступа: https://flutter.dev/
8. TypeScript [Электронный ресурс]: – Режим доступа: https://www.typescriptlang.org/
9. Expo-barcode-scanner [Электронный ресурс]: – Режим доступа: https://docs.expo.dev/versions/latest/sdk/bar-code-scanner/

ПРИЛОЖЕНИЕ А

Исходный код файла navigation.tsx

```
export function Root() {
    return(
        <Drawer.Navigator initialRouteName={navigationEnums.mainScreen}
screenOptions={{headerShown:true}}>
            <Drawer.Screen name={navigationEnums.main} component={WelcomeScreen} />
            {/*<Drawer.Screen name="LoginScreen" component={Login} />*/}
            <Drawer.Screen name={navigationEnums.purchases} component={Purchases} />
            <Drawer.Screen name={navigationEnums.openNomenclature}
component={Nomenclature}/>
            <Drawer.Screen name={navigationEnums.addNomenclature}
component={AddNomenclature}/>
            <Drawer.Screen name={navigationEnums.itemWork} component={ItemsScreen}/>
            <Drawer.Screen name={navigationEnums.deals} component={Deals}/>
            <Drawer.Screen name={navigationEnums.partners} component={PartnersScreen} />
            <Drawer.Screen name={navigationEnums.priceRule} component={PriceRules} />
            <Drawer.Screen name={navigationEnums.qrScanner} component={QRCodeScreen}/>
        </Drawer.Navigator>
    )
}

const Drawer = createDrawerNavigator();
const Stack = createStackNavigator();

export default function Navigation(){
    return(
    <NavigationContainer>
        <Stack.Navigator screenOptions={{headerShown:false}}>
            <Stack.Screen name={navigationEnums.root} component={Root}
options={{headerShown:false}} />
            <Stack.Screen name={navigationEnums.addPartner}
component={AddPartnersScreen}/>
            <Stack.Screen name={navigationEnums.addRule} component={AddPriceRules}/>
            <Stack.Screen name={navigationEnums.qrInfo} component={QRCodeInfo}/>
            <Stack.Screen name={navigationEnums.addPurchases} component={AddPurchases}/>
            <Stack.Screen name={navigationEnums.editPurchases}
component={EditPurchaseScreen}/>
            <Stack.Screen name={navigationEnums.addDeals} component={AddDeal}/>
        </Stack.Navigator>
    </NavigationContainer>
    );
}
```

ПРИЛОЖЕНИЕ Б

Исходный код файла PriceRules.tsx с просмотром и удалением ценовых правил

```
//displayed item
const Item = ({item, navigation} : any) => (
    <View style={[styles.item]}>
        <View style={styles.itemViewName}>
        <Text style={[styles.itemTextName]}>{item.name}</Text>
        </View>
        <View style={styles.itemView}>
            <Text style={[styles.itemText]}>{priceRulesConstants.percent}</Text>
            <Text style={[styles.itemText]}>{item.percent}%</Text>
        </View>
        <View style={styles.itemView}>
            <Text style={[styles.itemText]}>{priceRulesConstants.min}</Text>
            <Text style={[styles.itemText]}>{item.min}</Text>
        </View>
        <View style={styles.itemView}>
            <Text style={[styles.itemText]}>{priceRulesConstants.max}</Text>
            <Text style={[styles.itemText]}>{item.max}</Text>
        </View>
        <View style={styles.itemButtonsView}>
        <TouchableOpacity style={styles.chooseButtons} onPress={()=>deleteItem(item.id)}>
            <FontAwesome name="trash-o" size={24} color="black" />
        </TouchableOpacity>
        <TouchableOpacity style={styles.chooseButtons}
onPress={()=>navigation.navigate(navigationEnums.addRule, item)}>
            <FontAwesome name="edit" size={24} color="black" />
        </TouchableOpacity>
        </View>
    </View>
);

 function deleteItem(id: string) : void{
    console.log(id);
     PriceService.delete(id).then(r =>console.log('ok'));
}



export default function PriceRules ({navigation}:any){
    const [isLoading, setLoading] = useState(true)
    const [data, setData] = useState([]);
    const [selectedId, setSelectedId] = useState(null);


    //get request
    const getRules = async ()=>{
        await PriceService.get().then(resp=>{setData(resp.data)}).
            catch(error=>{console.log(error)}).
                finally(()=>setLoading(false))
    }
```

```
    //function to render item
    const renderItem = ({item} : any)=> {
        return (
            <Item
                item={item}
                navigation={navigation}
            />
        );
    }

    useEffect(()=>{
        setInterval(()=>getRules(), 5000);
    }, [])


    return(
        <SafeAreaView style={styles.container}>
            <View style={styles.buttonsView}>
                <ImageBackground
                    resizeMode={"cover"}
                    style={styles.img}
                    source={
                        require('../../assets/button-bg.png')
                    }
                >
                    <TouchableOpacity style={styles.button} onPress={()=>
navigation.navigate(navigationEnums.addRule)}>
                        <Text style={styles.text}>{priceRulesConstants.add}</Text>
                    </TouchableOpacity>
                </ImageBackground>
            </View>
                {isLoading ? <ActivityIndicator/> : (
                <FlatList nestedScrollEnabled={true}
                    data={data}
                    renderItem={renderItem}
                    keyExtractor={({ id }, index) => id}
                    extraData={selectedId}
                />)}
        </SafeAreaView>
    )
}
```

Исходный код модуля создания и редактирования ценового правила

```
 export default function AddPriceRules ({route, navigation} : any){

    let [NameValue, name] = React.useState("");
    let [PercentValue,percent ] = React.useState("");
    let [MinValue, min] = React.useState("");
    let [MaxValue, max] = React.useState("");
    let [idValue, id] = React.useState("");


    let Method=()=>(
        <TouchableWithoutFeedback onPress={()=>sendForm("null",false)}
onPressIn={onPressIn} onPressOut={onPressOut}>
```

```
            <LinearGradient
                start={{x:0.0, y:0.2}}
                end={{x:1.0, y:0.6}}
                locations={[0.2498,0.7503]}
                colors={['#804EA7','#4FB0C0']}
                style={styles.gradient}>
                <Animated.View style={[styles.saveButton, animatedScaleStyle]}>
                    <Text style={styles.text}>Сохранить</Text>
                </Animated.View>
            </LinearGradient>
        </TouchableWithoutFeedback>
    )


    if(route.params!=undefined){
        idValue = route.params['id'];
        [NameValue,name] = React.useState(route.params['name']);
        [PercentValue,percent ] = React.useState(route.params['percent']);
        [MinValue, min] = React.useState(route.params['min']);
        [MaxValue, max] = React.useState(route.params['max']);
        Method=()=>(
            <TouchableWithoutFeedback onPress={()=>sendForm(idValue,true)}
onPressIn={onPressIn} onPressOut={onPressOut}>
                <LinearGradient
                    start={{x:0.0, y:0.2}}
                    end={{x:1.0, y:0.6}}
                    locations={[0.2498,0.7503]}
                    colors={['#804EA7','#4FB0C0']}
                    style={styles.gradient}>
            <Animated.View style={[styles.saveButton, animatedScaleStyle]}>
                <Text style={styles.text}>Редактировать</Text>
            </Animated.View>
                </LinearGradient>
            </TouchableWithoutFeedback>
        )
    }


    //form send start
    const sendForm = async (id: string, update: boolean)=> {

        const tempMin: number = +MinValue;
        const tempMax: number = +MaxValue;
        if(tempMin>tempMax){

Alert.alert(priceRulesConstants.alertTitlePrice,priceRulesConstants.alertPriceText);
            return;
        }

        if(NameValue == '' || PercentValue=='' || MinValue=='' || MaxValue==''){

Alert.alert(priceRulesConstants.alertTitleForm,priceRulesConstants.alertFormText);
            return;
        }
        const body = {
            name: NameValue,
            percent: Number(PercentValue),
            min: Number(MinValue),
            max: Number(MaxValue),
```

```
        }
        if(update){
            let {data} = await PriceService.patch(id,
body).then(navigation.navigate(navigationEnums.priceRule));

        }else{
            let {data} = await
PriceService.post(body).then(navigation.navigate(navigationEnums.priceRule));
        }
    }

    //form send end

    return(
        <SafeAreaView style={styles.container}>
            <View style={styles.titleView}>
                <Text style={styles.title}>{priceRulesConstants.title}</Text>
            </View>
            <View style={styles.formView}>
                <TextInput
                    style={styles.formInput}
                    onChangeText={name}
                    value={NameValue}
                    placeholder={priceRulesConstants.name}
                    placeholderTextColor={'#808080'}
                />
                <TextInput
                    style={styles.formInput}
                    onChangeText={percent}
                    value={PercentValue.toString()}
                    placeholder={priceRulesConstants.percent}
                    placeholderTextColor={'#808080'}
                />
                <TextInput
                    style={styles.formInput}
                    onChangeText={min}
                    value={MinValue.toString()}
                    placeholder={priceRulesConstants.min}
                    placeholderTextColor={'#808080'}
                />
                <TextInput
                    style={styles.formInput}
                    onChangeText={max}
                    value={MaxValue.toString()}
                    placeholder={priceRulesConstants.max}
                    placeholderTextColor={'#808080'}
                />
            </View>
            <View style={styles.saveButtonView}>
                <Method/>
        </View>
        </SafeAreaView>
    )
}
```

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В. Непомнящий
подпись инициалы, фамилия
«20» 06 2022 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование специальности

Мобильное приложение для системы складского учета автошин
тема

Руководитель _____ 14.06.22 доцент, канд. техн. наук М. С. Медведев
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 14.06.22 Т.С. Медведев
подпись, дата инициалы, фамилия

Нормоконтролер _____ 14.06.22 доцент, канд. техн. наук М. С. Медведев
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2022