МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ

ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к дипломному проекту:**

Мобильное приложение для бронирования мест на маршрутное такси

Дипломник ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Макрица Ф.Н.

Руководитель проекта ­ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тимонович Г.Л., к.т.н.,доцент

Заведующий кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Блинова Е. А., к.т.н., доцент

Консультант \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Соболевский А. С., ст. преп.

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Николайчук А.Н., пред.-стажер

Дипломный проект защищен с оценкой

Председатель ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2024

Записка тут,

**Реферат**

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрциа Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

1

*Реферат*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ 74217072, 2024*

Пояснительная записка содержит \_\_ страниц, \_\_ иллюстраций, \_\_ таблиц, \_\_ использованный источник, \_\_ приложений.

Java, Node.js, HTML, CSS

Целью дипломного проекта является разработка веб-приложения для удобного и эффективного способа бронирования мест на маршрутное такси по желаемому маршруту.

Пояснительная записка дипломного проекта состоит из реферата, оглавления, \_\_\_\_\_\_ глав, заключения и списка использованных источников.

В первом разделе описаны технические требования к проекту, рассмотрены аналоги и проведено описание технологий, используемых в процессе разработки веб-приложения.

Во втором разделе представлены основные подходы к построению программной части дипломного проекта.

В третьем разделе рассмотрены этапы разработки и программной реализации дипломного проекта.

В четверном разделе проведен анализ информационной безопасности

В пятом разделе находиться руководство пользователя по эксплуатации разработанной программы.

Шестой раздел содержит экономические расчёты.

В заключении приведены результаты проделанной работы, приводятся соображения насчёт использования приложения.

**Содержание**

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л*

Ф.И.О.

Подпись

Дата

1

*Содержание*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_, 2024*

[Введение 7](#_Toc167445590)

[1 Постановка задачи и анализ аналогичных решений 8](#_Toc167445591)

[1.1 Постановка задачи 8](#_Toc167445592)

[1.2 Актуальность задачи 8](#_Toc167445593)

[1.3 Обзор аналогов 9](#_Toc167445594)

[1.3.1 Приложение «По пути!» 10](#_Toc167445595)

[1.3.2 Веб-приложение «Маршруточка» 11](#_Toc167445596)

[1.3 Сайт «7520 Молодечно-Минск» 13](#_Toc167445600)

[1.4 Анализ решений 14](#_Toc167445601)

[1.4.1 Требование к приложению 15](#_Toc167445602)

[1.4.2 Выбор библиотек и технологий для серверной стороны 15](#_Toc167445603)

[1.4.3 Выбор библиотек и технологий для клиентской стороны 17](#_Toc167445604)

[1.4.4 Выбор средств программирования 19](#_Toc167445605)

[1.5 Выводы по разделу 20](#_Toc167445607)

[2 Проектирование приложения 21](#_Toc167445608)

[2.1 Диаграмма вариантов использования 21](#_Toc167445609)

[2.1.1 Алгоритм добавления маршрута 23](#_Toc167445610)

[2.2 Архитектура приложения 25](#_Toc167445611)

[2.3 Проектирование базы данных 28](#_Toc167445612)

[2.4 Выводы по разделу 29](#_Toc167445613)

[3 Реализация веб-приложения 31](#_Toc167445614)

[3.1 Разработка серверной части приложения 31](#_Toc167445615)

[3.2 Подключение базы данных 34](#_Toc167445616)

[3.3 Реализация клиентской части приложения 35](#_Toc167445617)

[3.4 Выводы по разделу 37](#_Toc167445618)

[4 Анализ информационной безопасности приложения 39](#_Toc167445619)

[4.1 Выводы по разделу 47](#_Toc167445620)

[5 Тестирование приложения 48](#_Toc167445621)

[6 Руководство программиста 50](#_Toc167445622)

[6.1 Роль пользователя 50](#_Toc167445623)

[6.2 Роль пользователя кассира 52](#_Toc167445624)

[6.3 Роль пользователя администратор 54](#_Toc167445625)

[6.4 Выводы по разделу 55](#_Toc167445626)

[7 Технико-экономическое обоснование проекта 56](#_Toc167445627)

[7.1 Общая характеристика разрабатываемого программного средства 56](#_Toc167445628)

[7.2 Исходные данные и маркетинговый анализ 56](#_Toc167445629)

[7.3 Методика обоснования цены 58](#_Toc167445630)

[7.3.1 Определение объёма программного средства 58](#_Toc167445631)

[7.3.2 Основная заработная плата 59](#_Toc167445632)

[7.3.3 Дополнительная заработная плата 60](#_Toc167445633)

[7.3.4 Расчёт отчислений на социальные цели 60](#_Toc167445634)

[7.3.5 Расходы на материалы 61](#_Toc167445635)

[7.3.6 Расходы на специальное оборудование и платные услуги 61](#_Toc167445636)

[7.3.7 Расчет прочих прямых затрат 61](#_Toc167445637)

[7.3.8 Расчет общепроизводственных и общехозяйственных расходов 62](#_Toc167445638)

[7.3.9 Расчет суммы расходов на разработку программного средства 62](#_Toc167445639)

[7.3.10 Расходы на сопровождение и адаптацию 62](#_Toc167445640)

[7.3.11 Расчет общей суммы расходов 63](#_Toc167445641)

[7.3.12 Определение цены, оценка эффективности 63](#_Toc167445642)

[7.4 Вывод по разделу 66](#_Toc167445643)

Введение

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

1

*Введение*

Лист

1

*БГТУ 00.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_\_, 2024*

Наступил век технологий, век, когда каждый из нас стремится к оптимизации своего времени и сделать жизнь более комфортной и эффективной. В этом стремлении мобильное приложение играют ключевую роль, предлагая инновационные решения для повседневных задач. Среди множества возможностей, которые они предоставляют, особенно ценятся те, которые помогают пользователю удобно, быстро и эффективно организовать время и перемещение.

Дипломный проект представляет собой мобильное и веб–приложение, которое решает задачу бронирования мест на маршрутное такси. Цель приложения – упростить процесс бронирования мест, делая его быстрым, удобным и доступным для каждого. Проект стремится предложить пользователю интуитивно понятный интуитивно понятный интерфейс, чтобы каждый мог легко и комфортно пользоваться моим приложением. Проект разработан на программной платформе Windows 11, Node.js и Android, а также используются JavaScript, HTML и CSS для создания удобного и привлекательного для пользователя интерфейс. Для разработки мобильной и веб версии приложения использовались программы Android Studio и Visual Studio Code, которые обеспечивают стабильную работу на различных устройствах и платформах. Для реализации данного программного продукта был выбран язык Java для мобильного приложения, а для веб – JavaScript. JavaScript является основным языком программирования для Node.js, который используется для написания серверного кода, обработки HTTP-запросов, взаимодействия с базами данных, создания API и многих других задач на стороне сервера.

В качестве системы управления базой данных была использована СУБД MySQL, так как является одним из популярных приложений.

Также были представлены цели проекта, которые необходимо достичь:

– найти и проанализировать схожие по тематике проекта приложения;

– спроектировать серверную и клиентскую часть приложения;

– разработать серверную и клиентскую часть приложения;

– протестировать готовое приложение;

– провести технико-экономическое обоснование проекта.

# Постановка задачи и анализ аналогичных решений

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*1. Обзор аналогов и постановка задачи*

Лист

1

*БГТУ 01.00. ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_\_, 2024*

* 1. **Постановка задачи**

Целью дипломного проекта является разработка приложения, представляющего собой мобильное и веб-приложение, которое должно выполнять бронирование мест на маршрутное такси. Это подразумевает, что в итоге должен получится законченный программный продукт, который позволит пользователям осуществлять бронирование мест. Необходимо реализовать серверную и мобильное приложение.

Мобильная часть приложения должна:

* предоставлять пользователю список доступных мест;
* обладать простым и удобным интерфейсом.

Серверная часть приложения должна:

* быть легко расширяемой;
* обеспечивать хранение данных.

Задачи, которые должно выполнять мобильное приложение:

* регистрация и авторизация;
* просмотр списка маршрутов и свободных мест;
* добавление бронирование на место;
* помечать прибытие клиента.
  1. **Актуальность задачи**

В настоящее время междугородные пассажирские перевозки являются неотъемлемой частью жизни города, где еще не у каждого человека есть возможность прибрести личный автомобиль, а организовать в короткие сроки поездку в другой город бывает проблематично. Так же и со стороны компаний существует проблема нерационального использования своих ресурсов. Решение этих проблем может пойти на пользу как пассажирам, так и перевозчикам.

В настоящее время бронирование мест происходит в режиме онлайн, часто при помощи мобильного телефона. Общие принципы онлайн-бронирования заключаются в следующем:

* поиск доступных предложений. Пользователь на сайте выбирает удобный маршрут, даты, количество человек, расположение и другие параметры, в зависимости от предоставляемых услуг онлайн-бронирования;
* осуществляет бронирование;
* платеж за место. Обычно прохождение платежа происходит напрямую с кассиром или водителем;

У

* получение документа, подтверждающего оплату бронирования.

Маршрутное такси – микроавтобусы, осуществляющие перевозку пассажиров на остановочных пунктах и в любых, не запрещенных правилами дорожного движения местах.

У

Некоторые компании, которые занимаются организацией перевозки пассажиров, уже имеют свои собственные программные средства. Среди них есть и веб-сервисы, и мобильные приложения. Однако они работают только по своим отдельным маршрутам и имеют довольно ограниченную функциональность.

Многие организации все еще делают упор на call-центры и для бронирования необходимо выполнить ряд действий:

* найти номер телефона call-центра компании перевозчика по интересующему направлению
* дозвониться call-центру, уточнить наличие мест на интересующий рейс
* сообщение контактные данные
* ожидание подтверждения

Если по интересующему направлению мест не нашлось, придется искать номер телефона другой организации и пройти все шаги снова. Программная поддержка системы бронирования билетов на междугородние маршрутные такси позволит освободить человека от каких-либо усилий при бронировании маршрутного такси, сократить время бронирования и свести к минимуму число необходимых для этого действий. Для компаний, предоставляющих услуги перевозки, позволит снизить затраты на call-центр, так как процесс бронирования будет автоматизирован, анализировать текущий поток пассажиров в режиме реального времени, что позволит рационально использовать свои ресурсы и всегда отражать актуальную информацию. Основными задачами программного продукта являются объединить множество перевозчиков в единую сеть и снизить временные затраты при бронировании билетов.

Автоматизированная система бронирования билетов на междугородние маршрутные такси предоставит возможность для осуществления целого ряда действий: (изм)

* найти необходимый рейс и проверить наличие мест;
* сортировка и фильтрация рейсов по различным параметрам;
* забронировать себе места;
* при необходимости отменить бронь;
* зарегистрироваться в приложении и копить баллы за поездки;
* оплачивать брони баллами или со скидкой по промокоду;
* хранить все свои брони в профиле;
* поделиться информацией о своей поездке в социальных сетях.
  1. **Обзор аналогов**

На текущий момент существует множество различных приложений для бронирования мест. Для формирования формулировки требований к функционалу разрабатываемого приложения необходимо рассмотреть достоинства и недостатки аналогов. В качестве аналогов разрабатываемого приложения можно выделить список следующих:

* приложение «По пути!»;
* веб - приложение «Маршруточка»;
* сайт «7520 Молодечно-Минск».

1.3.1 Приложение «По пути!»

Приложение «По пути!» – это приложение, которое предназначено для бронирования мест на маршрутное такси. На этом сайте есть фильтрация маршрута по дате, а также можно посмотреть авторизации и регистрации пользователя, изменить профиль пользователя и посмотреть будущие, прошедшие и отмененные заказы. Главная страница сайта представлена на рисунке 1.1.

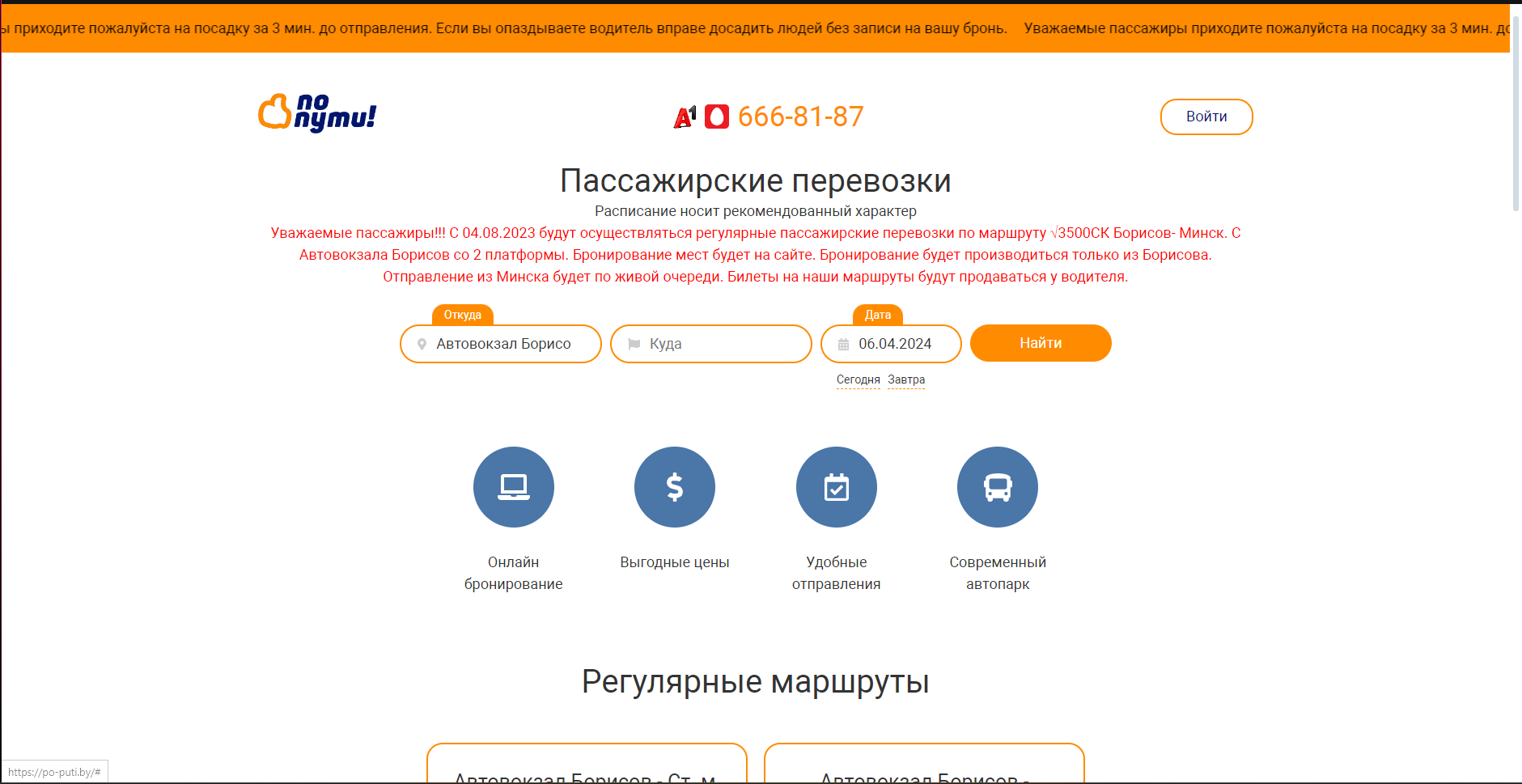


Рисунок 1.1 – Главная страница приложения «По пути!»

Также на сайте представлена возможность авторизации пользователя. Форма с авторизации представлена на рисунке 1.2.

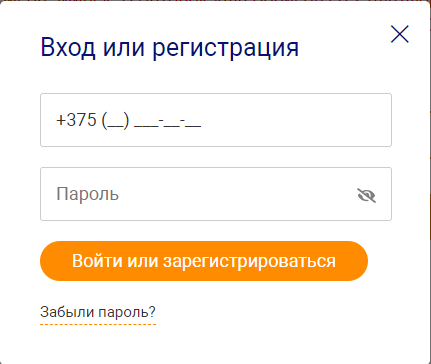


Рисунок 1.2– Окно регистрации/авторизации

Фильтр для уточнения маршрута, находиться на рисунке 1.1.

На сайте присутствует только три цвета, которые видит пользователь. На рисунке 1.1 можно наблюдать что при разработке сайта используется оранжевый, белый и синий цвет, однако следует заметить, что дизайн сайта имеет приоритет на удобство для использования с мобильного устройства, что подтверждает характерные закруглённые по бокам кнопки. Также стоит отметить, что на одной и той же странице текст отображается разными цветами.

К достоинствам сайта можно отметить:

* отсутствие рекламы;
* личный аккаунт;
* редактирование профиля;
* уведомление об изменении на всем сайте.

К недостаткам сайта можно отнести следующие:

* большое время задержки для бронирования свободных мест равных 3 – 5;
* показ отбытых маршрутов на главной странице при поиске и заказ мест на них.(изм)

1.3.2 Веб-приложение «Маршруточка»

Данный сайт представляет из себя типичный сайт со времен 90-х годов. На сайте присутствует авторизации клиента. Также присутствует быстрый поиск, обеспечивающий возможность заказать место из Минска до Трокели.

Главная страница сайта представлена на рисунке 1.3. На рисунке видно, что дизайн приложения схож со временами 90-х и не удобный для глаз, так как присутствует много объектов которые не сочетаться между собой и не позволяют человеку зацепиться за какой-то элемент сайта.

Из этого следует что дизайн на сайте играет достаточно большую роль, чтобы пользователю было комфортно использовать то приложение, с которым будет происходить взаимодействие.

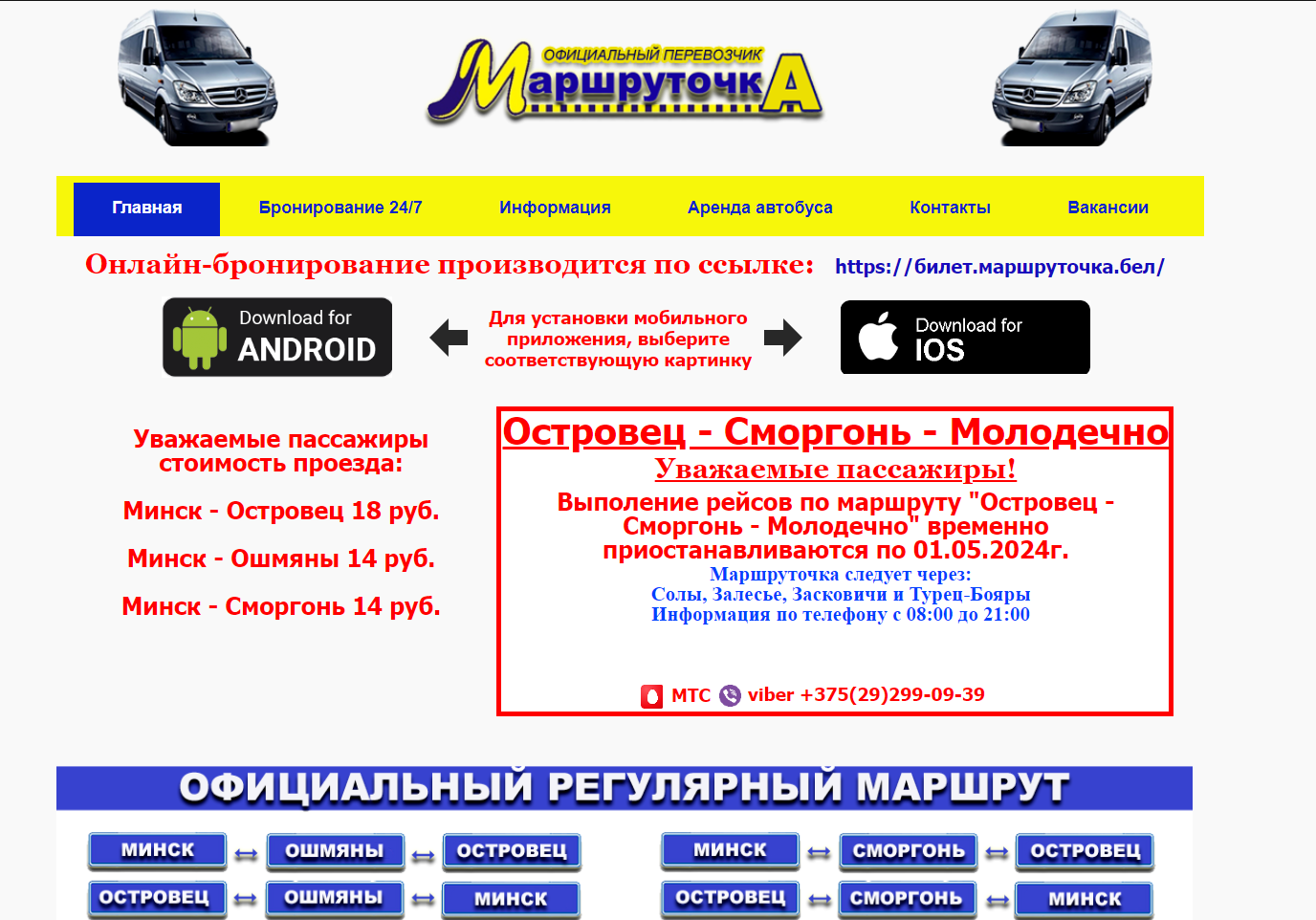


Рисунок 1.3– Главная страница веб-приложения «Маршруточка»

На странице, есть возможность заказать бронь на место нажав на соответствующую кнопку в меню выше.

При нажатии на нее мы переходим на страницу, которая имеет весьма другой дизайн в отличии от главного меню, представлено на рисунке 1.4.

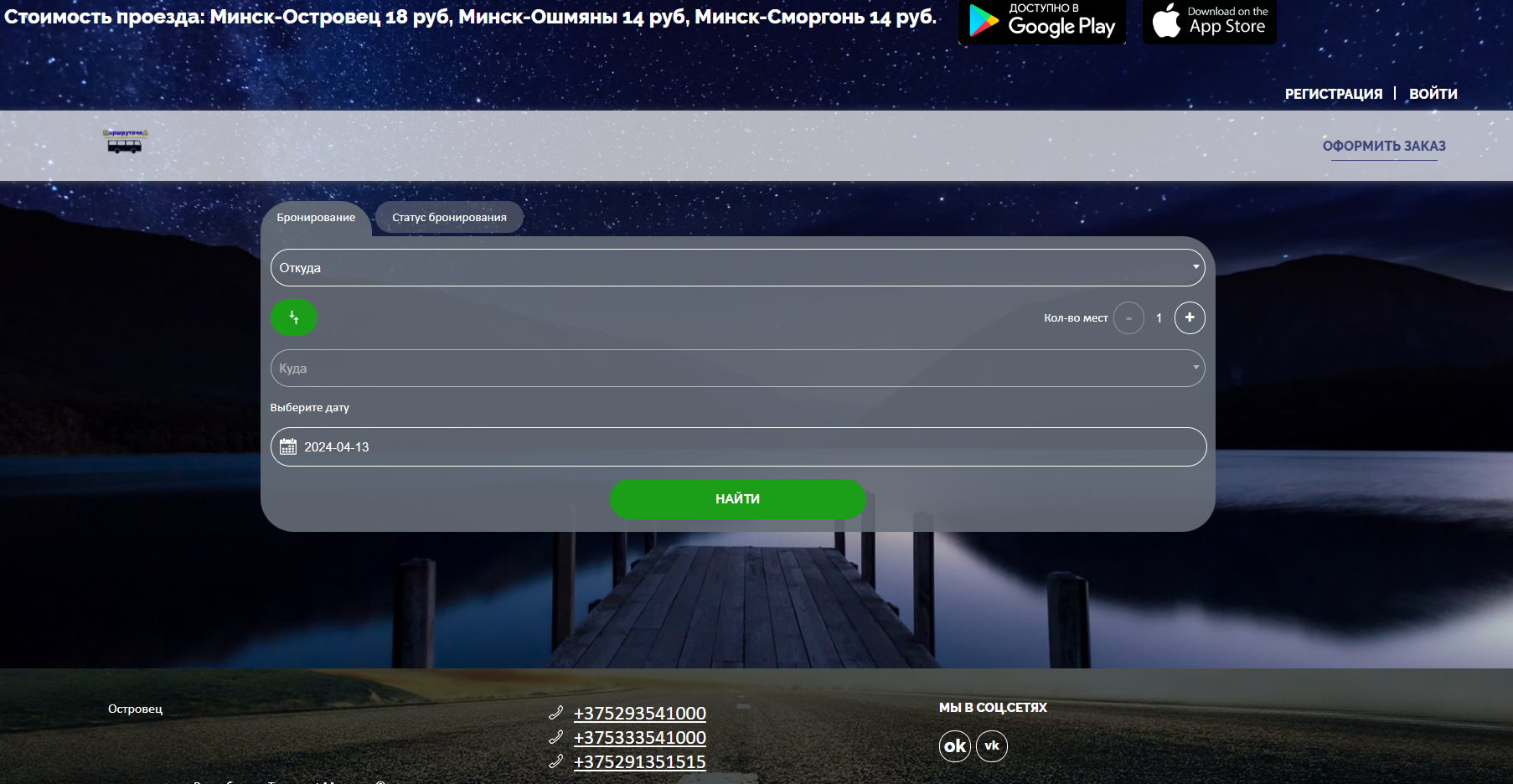


Рисунок 1.4– Страница веб-приложения «Маршруточка» для бронирования места

Как можно увидеть из рисунка выше, страница ориентирована на использования ее для мобильного устройства. Страница обладает возможностью заказа места и авторизации пользователя, следует отметить, что у страницы присутствует недочет дизайна при нажатии на кнопки войти и регистрация, который показан на рисунке 1.5.(изм)

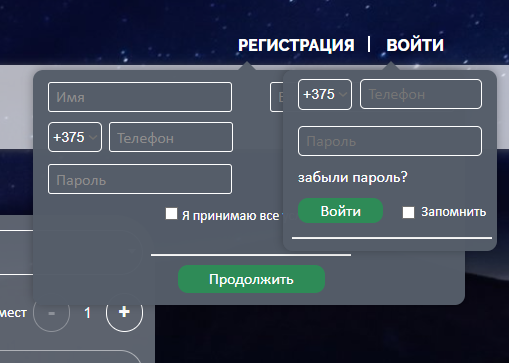


Рисунок 1.5– Окно авторизации и регистрации.

Также необходимо подметить что при заполнении обеих окно и одновременной нажатии на кнопки продолжить и войти, сайт зависает и требуется перезагрузить его.(изм)

Также на сайте при авторизации пользователя присутствует возможность управлять своими заказами.

К достоинствам сайта можно отнести:

* отсутствие рекламы на сайте;
* управление бронированием мест;
* приятный для глаза дизайн страницы для бронирования мест;
* уведомление пассажиров на сайте об изменении в расписании.

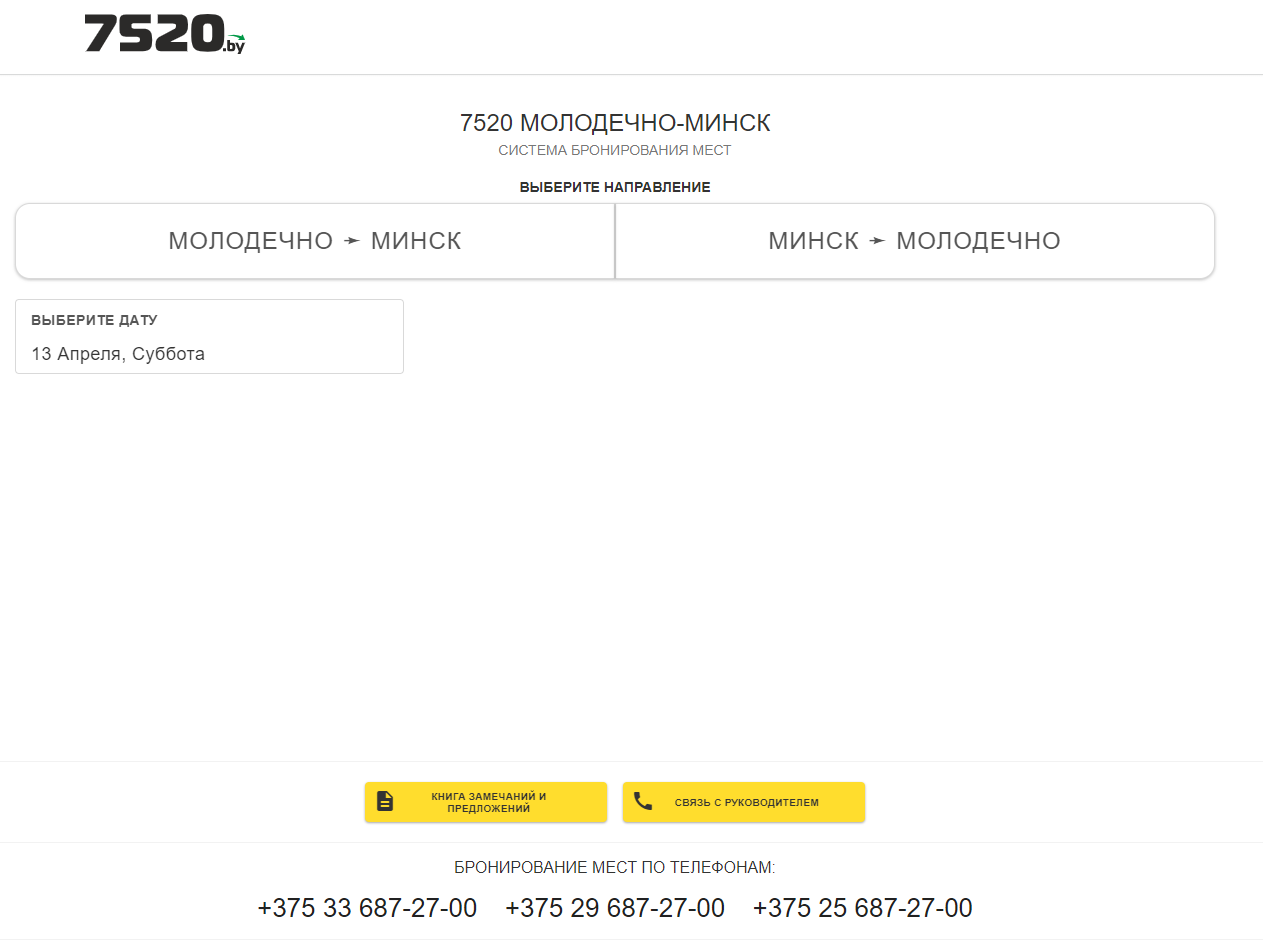
К недостаткам можно отнести:

* устаревший дизайн сайта, пока не переходишь на страницу бронирования;
* невозможность редактирования профиля;
* наличие большого числа лишних элементов, кроме страницы бронирования.(изм)

1.3 Сайт «7520 Молодечно-Минск»

Сайт «7520 Молодечно-Минск» имеет простой и не громоздкий дизайн, удобный и понятный интерфейс, быстрое бронирование.

Главная страница приложение представлена на рисунке 1.6.

 Рисунок 1.6– Главная страница сайта «7520 Молодечно-Минск»

Также хочется отметить в нерабочей логотипе и убегающей кнопке войти справа вверху.(изм)

Этот недостаток можно увидеть если уменьшить размер окна браузера или запустить сайт на телефоне (рис. 1.7) при этом логотип пропадает из виду. (изм)

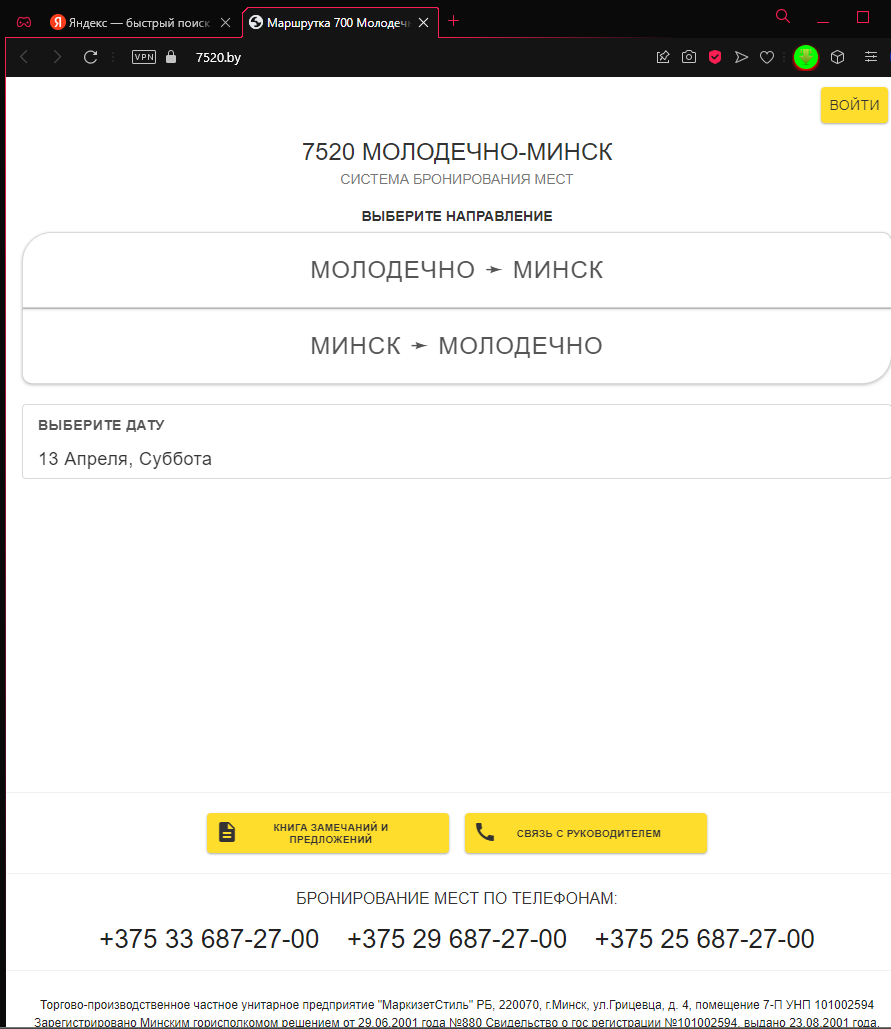


Рисунок 1.7– Главная страница сайта в маленьком окне сайта.

К достоинствам сайта можно отнести:

* отсутствие рекламы на сайте;
* управление бронированием мест;
* приятный для глаза дизайн страниц.

К недостаткам можно отнести:

* невозможность редактирования профиля;
* нету уведомление пассажиров на сайте об изменении в расписании.

1.4 Анализ решений

При создании программного средства одним из важных моментов является выбор языка программирования, так как от него зависит множество факторов, таких как скорость создания программы, скорость и удобство тестирования, возможность миграции на другие платформы, возможность быстрого внесения изменений и так далее. Также стоит понимать, что идеального языка программирования нет, так как все разные инструменты и имеют свои преимущества и недостатки при разработке. Наиболее актуальными сегодня являются Java, C#, Python, TypeScript.

1.4.1 Требование к приложению

Разработчик при создании приложении установил следующие требования:

* бронирование места для пользователя;
* корректировка количества забронированных мест пользователя;
* удаление брони для пользователя;
* создание брони на маршрутное такси для администратора;
* корректировка списка брони для администратора;
* удаления списка брони для администратора;
* изменения статуса маршрута для кассира пользователя;
* просмотр списка маршрута для кассира, пользователя и администратора.

1.4.2 Выбор библиотек и технологий для серверной стороны

В качестве языка программирования мной был выбран JavaScript c программной платформой Node.js. JavaScript – это кроссплатформенный, интерпретируемый язык программирования, который используется для создания интерактивных веб-страниц и приложений. Функционал JavaScript может улучшить удобство взаимодействия пользователя с веб-сайтом: от обновления ленты новостей в социальных сетях и до отображения анимации и интерактивных карт. Он был создан в 1995 году Бренданом Эйхом, когда тот еще был инженером в Netscape. JavaScript имеет динамическую слабую типизацию и может использоваться как для разработки на стороне клиента, так и для разработки на стороне сервера. Исторически веб-страницы были статичными, похожими на страницы в книге. Статическая страница в основном отображала информацию в фиксированном виде и не выполняла всего того, что мы сейчас ожидаем от современного сайта. Язык JavaScript возник как технология на стороне браузера, позволяющая сделать веб-сайты более динамичными. Используя его, браузеры могли реагировать на взаимодействие с пользователем и менять расположение контента на веб-странице.

Клиентская сторона. Расширения на стороне клиента позволяют приложению размещать элементы на HTML-форме и реагировать на пользовательские события, такие как клики мышкой, пользовательский ввод в форму и навигацию по странице. Среди популярных фреймворков для работы на стороне клиента стоит отметить AngularJS, ReactJS и VueJS.

Серверная сторона. Расширения на стороне сервера позволяют приложению общаться с базой данных, обеспечивать непрерывность информации от одного вызова приложения к другому или выполнять манипуляции с файлами на сервере. В настоящее время Node.js является наиболее популярным инструментом для этой цели. (изм)

Как и библиотеки JavaScript, фреймворки JavaScript представляют собой набор предварительно написанных фрагментов кода, которые выполняют различные функции и могут быть использованы повторно. Однако если библиотеки JavaScript — это специализированный инструмент для использования по запросу, то фреймворки JavaScript — это полноценный набор инструментов, который помогает сформировать и организовать любое веб-приложение. Если представить код приложения JavaScript в виде дома, то фреймворк JavaScript — это чертеж, по которому он строится [1].

Node или Node.js – программная платформа, основанная на движке [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) (компилирующем [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) в [машинный код](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%B4)), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами [ввода-вывода](https://ru.wikipedia.org/wiki/%D0%92%D0%B2%D0%BE%D0%B4-%D0%B2%D1%8B%D0%B2%D0%BE%D0%B4) через свой [API](https://ru.wikipedia.org/wiki/API), написанный на [C++](https://ru.wikipedia.org/wiki/C%2B%2B), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль [веб-сервера](https://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), но есть возможность разрабатывать на Node.js и десктопные оконные приложения (при помощи [NW.js](https://ru.wikipedia.org/wiki/NW.js), AppJS или [Electron](https://ru.wikipedia.org/wiki/Electron_(%D1%84%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA)) для [Linux](https://ru.wikipedia.org/wiki/Linux), [Windows](https://ru.wikipedia.org/wiki/Windows) и [macOS](https://ru.wikipedia.org/wiki/MacOS" \o "MacOS)) и даже программировать микроконтроллеры (например, tessel, low.js и [Espruino](https://ru.wikipedia.org/wiki/Espruino" \o "Espruino)). В основе Node.js лежит [событийно-ориентированное](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%B1%D1%8B%D1%82%D0%B8%D0%B9%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) и [асинхронное (или реактивное)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) программирование с [неблокирующим вводом/выводом](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B1%D0%BB%D0%BE%D0%BA%D0%B8%D1%80%D1%83%D1%8E%D1%89%D0%B0%D1%8F_%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F).

Node.js разработал [Райан Даль](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B9%D0%B0%D0%BD_%D0%94%D0%B0%D0%BB%D1%8C) в [2009 году](https://ru.wikipedia.org/wiki/2009_%D0%B3%D0%BE%D0%B4) после двух лет экспериментирования над созданием серверных веб-компонентов. В ходе своих исследований он пришёл к выводу, что вместо традиционной модели параллелизма на основе потоков следует обратиться к событийно-ориентированным системам. Эта модель была выбрана из-за простоты, низких накладных расходов (по сравнению с идеологией «один поток на каждое соединение») и быстродействия. Целью Node является предложить «простой способ построения масштабируемых сетевых серверов». [2]

Сервер написан на языке программирования JavaScript с использованием программной платформы Node.js, реализует архитектуру MVC (Model-View-Controller). Model-View-Controller (MVC, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо. (изм)

Модель (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Концепция MVC была описана [Трюгве Реенскаугом](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B5%D0%BD%D1%81%D0%BA%D0%B0%D1%83%D0%B3,_%D0%A2%D1%80%D1%8E%D0%B3%D0%B2%D0%B5) в 1978 году, работавшим в научно-исследовательском центре «[Xerox PARC](https://ru.wikipedia.org/wiki/Xerox_PARC)» над языком программирования «[Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk" \o "Smalltalk)». Позже, Стив Бурбек реализовал шаблон в [Smalltalk-80](https://ru.wikipedia.org/wiki/Smalltalk-80).

Окончательная версия концепции MVC была опубликована лишь в 1988 году в журнале [Technology Object](https://ru.wikipedia.org/w/index.php?title=Technology_Object&action=edit&redlink=1).

Впоследствии шаблон проектирования стал эволюционировать. Например, была представлена иерархическая версия [HMVC](https://ru.wikipedia.org/wiki/HMVC); [MVA](https://ru.wikipedia.org/w/index.php?title=MVA&action=edit&redlink=1), [MVVM](https://ru.wikipedia.org/wiki/MVVM). [3].

Схема взаимодействия Model-View-Controller представлена на рисунке 1.8 ниже.(изм вис)

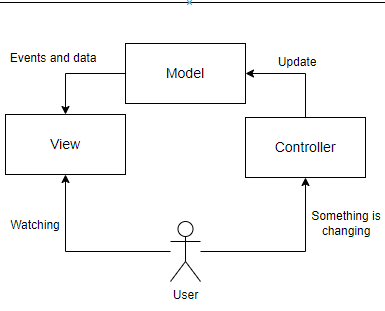


Рисунок 1.8 – Схема взаимодействия MVC (Model-View-Controller)

Java – это высокоуровневый язык программирования, разработанный компанией Sun Microsystems (сейчас Oracle) в 1995 году. Он был создан с целью обеспечить возможность создания программ, которые могут работать на различных платформах и операционных системах без изменений кода.

1.4.3 Выбор библиотек и технологий для клиентской стороны

Для создания мобильного приложения используется язык программирования Java.(изм)

Java является объектно-ориентированным языком программирования, это значит, что программы пишутся в виде набора объектов, взаимодействующих друг с другом. Этот язык широко используется для разработки различных мобильных и веб-приложений, десктопных программ и сервисов для Интернета вещей.

Создание Java является одним из самых значимых событий в истории программирования. В начале 1990-х годов, в компании Sun Microsystems, начали работу над проектом под кодовым названием "Oak".

В 1995 году был представлен новый язык программирования под названием Java. Ключевой идеей Java была возможность написания одной программы, которая могла бы работать на разных платформах и операционных системах без необходимости переписывания кода. Благодаря этому свойству Java стала популярной среди разработчиков мобильных и веб-приложений.

В 2009 году компания Sun Microsystems была приобретена компанией Oracle, что привело к дальнейшему развитию и совершенствованию языка Java.

Сегодня Java остается одним из наиболее популярных и востребованных языков программирования. Она используется для создания широкого спектра приложений, от веб-сайтов до корпоративных систем и мобильных приложений. Основные особенности языка Java представлены ниже.

Объектно-ориентированный подход (ООП). Это означает, что все в Java является объектом и позволяет разработчикам создавать модульный код, который легко поддерживать. Это повышает повторное использование и облегчает разработку сложных систем.

Платформонезависимость. Это означает, что приложение, написанное на Java, может работать на любой операционной системе без необходимости переписывания кода.

Автоматическое управление памятью, благодаря чему программисту не нужно беспокоиться о выделении и освобождении места.

Многопоточность языка позволяет одновременно выполнять несколько задач в одном приложении.

Безопасность. Java имеет встроенные механизмы безопасности, которые помогают защитить программы от вредоносного кода и предотвращают потенциальные уязвимости. Песочница (sandbox) Java ограничивает доступ приложения к ресурсам операционной системы, обеспечивая изоляцию и безопасность выполнения программ.(изм)

Большая библиотека классов позволяют создавать приложения различной сложности. Она включает в себя графический интерфейс пользователя, сетевые операции, работу с базами данных, обработку строк и многое другое. Это упрощает разработку приложений, так как многие задачи уже реализованы и готовы к использованию.

Простота и удобство использования Java делает его удобным для новичков в программировании [4].

Для создания веб-приложения использовался язык разметки HTML и язык таблиц стилей CSS.

HTML и CSS являются основными технологиями для создания веб-страниц и веб-приложений. HTML (Hypertext Markup Language) — это язык разметки, который используется для определения структуры веб-страниц. CSS (Cascading Style Sheets) — это язык таблиц стилей, который используется для определения внешнего вида веб-страницы.

HTML используется для создания разметки веб-страниц, которая определяет структуру содержимого веб-страницы, такую как заголовки, параграфы, списки, ссылки и т.д. С помощью HTML вы можете создавать элементы, определять их свойства и располагать их на странице.

CSS используется для определения внешнего вида веб-страницы, такого как цвета, шрифты, размеры и расположение элементов на странице. Он позволяет создавать стили, которые могут быть использованы на нескольких страницах, а также использовать анимации и эффекты для создания интерактивных веб-страниц.

Совместное использование HTML и CSS позволяет создавать привлекательные и функциональные веб-страницы, которые могут отображаться на различных устройствах и браузерах.

1.4.4 Выбор средств программирования

Для разработки данного проекта использовалось следующие программное обеспечение:

* Android Studio — это интегрированная среда разработки (IDE) для создания мобильных приложений под операционную систему Android. Эта IDE является официальным инструментом разработки от компании Google и основана на популярной среде разработки IntelliJ IDEA. Android Studio обладает широким набором инструментов для разработки мобильных приложений, включая редактор кода, отладчик, эмулятор устройства, инструменты для разработки пользовательского интерфейса и т.д. Эта IDE также обеспечивает интеграцию с другими инструментами разработки, такими как системы контроля версий Git и SVN, а также библиотеки и фреймворки, такие как Gradle. Android Studio также обладает большим сообществом разработчиков, которые предоставляют бесплатные плагины и инструменты, которые помогают ускорить процесс разработки. Благодаря всем этим инструментам и возможностям, Android Studio является одним из самых популярных инструментов для разработки мобильных приложений под Android.
* Visual Studio Code (VS Code) — это бесплатный и открытый исходный код редактор кода, который разработан Microsoft. Он поддерживает множество языков программирования и позволяет расширять функциональность с помощью плагинов и расширений. VS Code имеет множество функций, которые помогают упростить процесс разработки приложений. Некоторые из них включают: подсветка синтаксиса для многих языков программирования, автодополнение кода, интеграция с системой контроля версий Git, отладчик, который позволяет отслеживать выполнение кода и исправлять ошибки, интеграция с различными фреймворками и библиотеками.

MySQL Workbench — это интегрированная среда разработки (IDE) для работы с базами данных MySQL. Она предоставляет пользователю графический интерфейс для создания и управления базами данных, а также для проектирования таблиц и запросов. MySQL Workbench имеет множество полезных функций, таких как возможность создавать, изменять и удалять таблицы, индексы, процедуры и триггеры; редактировать и выполнять запросы на языке SQL; создавать модели данных с помощью инструментов ER-моделирования; создавать резервные копии и восстанавливать данные; анализировать запросы на производительность и многое другое. MySQL Workbench также имеет удобный пользовательский интерфейс, который облегчает работу с базами данных. Он позволяет легко переключаться между различными базами данных, открывать несколько вкладок с различными запросами, визуально представлять структуру таблиц и многое другое. Кроме того, MySQL Workbench предоставляет мощные средства для работы с базами данных на удаленных серверах. Он поддерживает соединение с базами данных MySQL на удаленных серверах и предоставляет возможность удаленного управления базами данных.

1.5 Выводы по разделу

В данном разделе был проведен анализ предметной области по теме «Бронирование мест на маршрутное такси». Также были рассмотрены некоторые аналоги веб-приложений для бронирования мест.

При написании проекта были учтены как положительные, так и отрицательные черты аналогов. Был изучен опыт других разработчиков. Проанализировав сильные и слабые стороны приведенных выше аналогов, была поставлена цель разработать «Мобильное приложение для бронирования мест на маршрутное такси».

# Проектирование приложения

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*2. Проектирование   
программного средства*

Лист

1

*БГТУ 02.00.ПЗ*

Лит.

Листов

*БГТУ 74202716, 2024*

2.1 Диаграмма вариантов использования

Проектирование – один из важных шагов при разработке программы, который очень часто игнорируется начинающими разработчиками. Обычно начинающие программисты пытаются удержать всё в голове или, в лучшем случае, записать некоторые важные сведения на листе бумаги. Как результат, у них нет чёткого плана дальнейших действий, и проект может быть отложен в долгий ящик.

Обычно при проектировании разработчики изображают систему графически, поскольку человеку легко разобраться в таком представлении. Именно поэтому вместо написания громоздких текстов про каждую возможность будущей программы разработчики строят различные диаграммы для описания своих систем. Это помогает им не забывать, что нужно реализовать в программе, и быстро вводить в курс дела своих коллег.

Диаграмма вариантов использования (англ. use-case diagram) – диаграмма, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов или функций, которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

В разрабатываемом приложении должна быть разработана система аутентификации пользователей и предоставление прав на определенные действия в приложении. В ходе разработки программного средства будет реализовано три типа пользователей: обычный пользователь, кассир и администратор, имеющий расширенный функционал. Администратор помимо дополнительных возможностей, которые отличают его от обычного пользователя, может использовать также и функционал, доступный обычным пользователям.

Перед началом разработки необходимо определить роли, задачи, все варианты использования программного средства. Для этого необходимо построить диаграмму вариантов использования.

У

Для обеспечения безопасности пользователя с приложением была разработана эффективная система регистрации и аутентификации. Особенности регистрации заключаются в том, что пользователь может легко создать учетную запись, заполнив простую форму с именем, телефонным номером и паролем. При этом пароль пользователя хэшируется и хранится в безопасной форме в базе данных MySQL, тем самым обеспечивая конфиденциальность информации. При входе в систему пользователь предоставляет свои данные, которые проверяются на соответствие с данными, хранящимся в базе данных. Аутентификация осуществляется с использованием bcrypt для проверс пароля на правильность. После успешной аутентификации пользователь получает доступ к функционалу приложения в зависимости от его роли. В зависимости от роли система предоставляет пользователю различные привилегии. Например, администраторы имеют доступ к определенным административным функциям, в то время как кассиры имеют доступ к функциям, связанным с обслуживанием клиентов. После успешной аутентификации пользователь сохраняется в сессии, что позволяет сохранить его состояние и обеспечивает безопасность во время сеанса использования приложения. Для обеспечения безопасности маршрутов приложения используется middleware в методе checkAuth, который проверяет аутентификацию пользователя перед выполнением запроса. Если пользователь не авторизован, ему предлагается войти в систему через форму входа. Важно отметить, что пароли хранятся в зашифрованном виде, что гарантирует безопасность пользовательских данных в случае утечки информации. Эта система обеспечивает не только безопасность, но и удобство использования моего приложения, позволяя пользователям сосредоточиться на своих задачах, не беспокоясь о безопасности своих данных.

Диаграмма вариантов использования для пользователя представлена на рисунке 2.1.

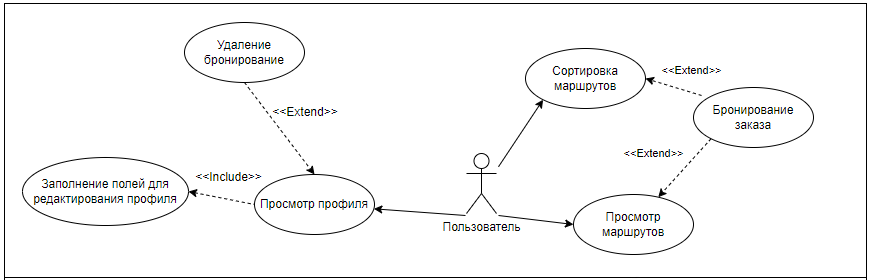


Рисунок 2.1 – Диаграмма вариантов использования для пользователя

Как видно из диаграммы вариантов использования пользователя, представленной выше, зарегистрированный пользователь. Однако незарегистрированный пользователь может взаимодействовать с системой, но только просматривать маршруты.

Зарегистрированный пользователь имеет 3 функциональных возможностей, а именно просмотра своего профиля, в котором пишется все забронированные заказы, редактировать его, удалить бронь на место. Также он может просмотреть список маршрут и забронировать место.

Диаграмма использования кассира представлена в приложении (потом нарисую). Следую из ей можно сделать вывод, что пользователь, обладающий такими правами, разрешается получить информацию об именах людей, которые забронировали место, для того чтобы отслеживать их прибытие на маршрут, также кассир имеет право заблокировать маршрут, после его отбытия от места стоянки. По сути своей кассир выполняет роль отслеживания за пассажирами и автомобилем.

2.1.1 Алгоритм добавления маршрута

Данный алгоритм разъясняет это логику для пользователя имеющего роль администратора. Для добавления администратору нового маршрута необходимо в начале пройти регистрацию на сайте, для этого необходимо заполнить три поля, а именно логин, телефон и пароль от учетной записи.

После этого у пользователя появляется возможность просмотра всех маршрутов, добавить новый или изменить уже существующий. Представлено на рисунке 2.2.

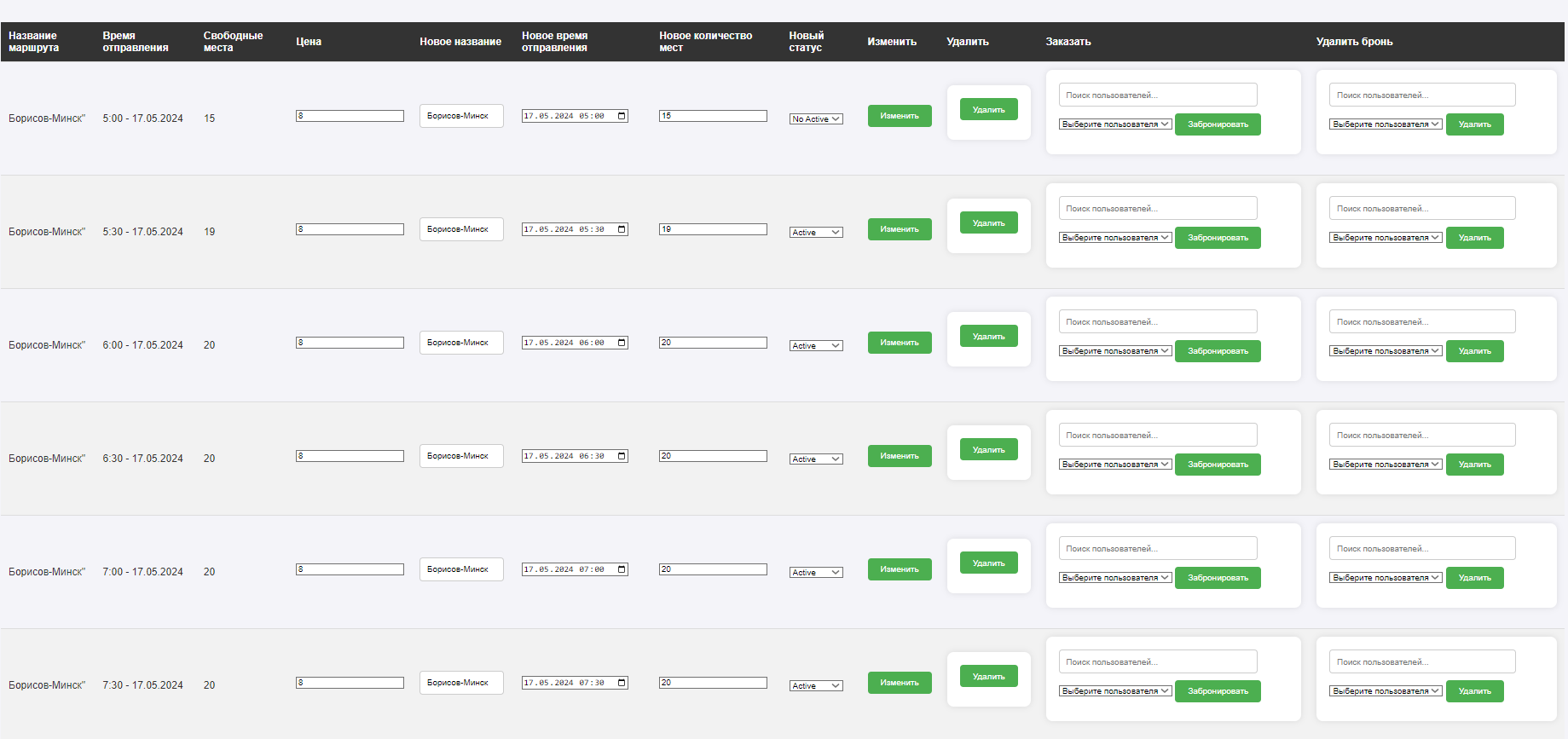


Рисунок 2.2 – Таблица всех маршрутов

Необходимо отметить, что пользователь с ролью администратор, будет использоваться исключительно на компьютере в браузере. В приведенном рисунке 2.2 используется браузером OperaGX и системой Windows 11 64–bit. И так для того, чтоб создать пользователю новый маршрут ему необходимо пройти 5 шагов:

* необходимо выбрать маршрут. Пользователь наживает Минск–Борисов или Борисов–Минск.
* необходимо выбрать дату. Пользователь наживает одну из кнопок сегодня, завтра либо послезавтра.
* необходимо выбрать время. Пользователь наживает одну из кнопок с 5:00–22:00.
* Необходимо выбрать число мест. Пользователь наживает 20 или 30 мест.
* Необходимо выбрать цену. Пользователь пишет сам.(изм и смал)

Данные действия выполнятся в рамках рисунка 2.3.

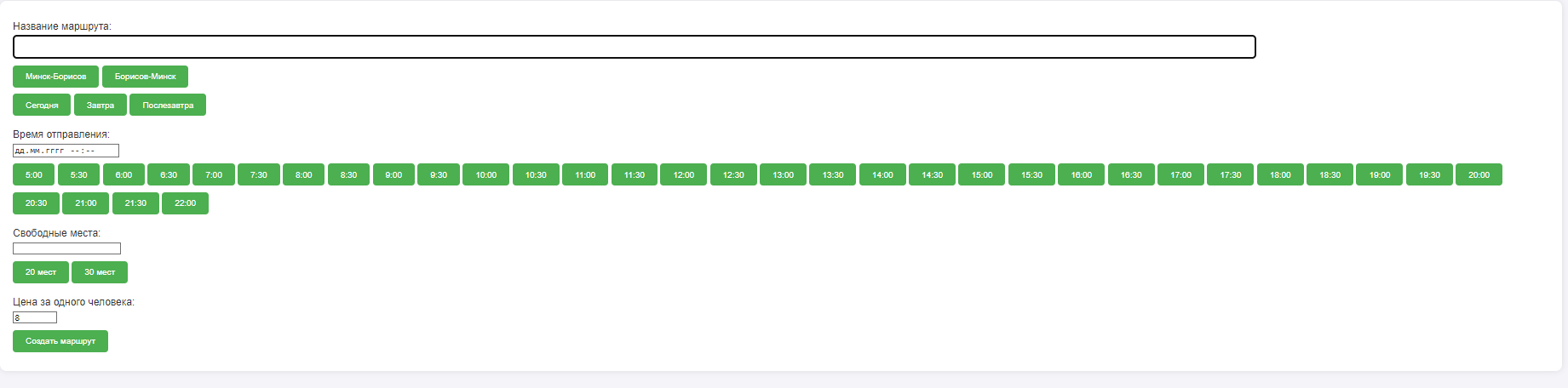


Рисунок 2.3 – Кнопки создание маршрута

Также в программе предусмотрено создания маршрута по рейсам Минск–Борисов и Борисов–Минск, с 5:00 по 22:00 на завтра. Для этого пользователь с ролью администратора нажимает кнопки Минск–Борисов и Борисов–Минск представленные на рисунке 2.4 ниже.

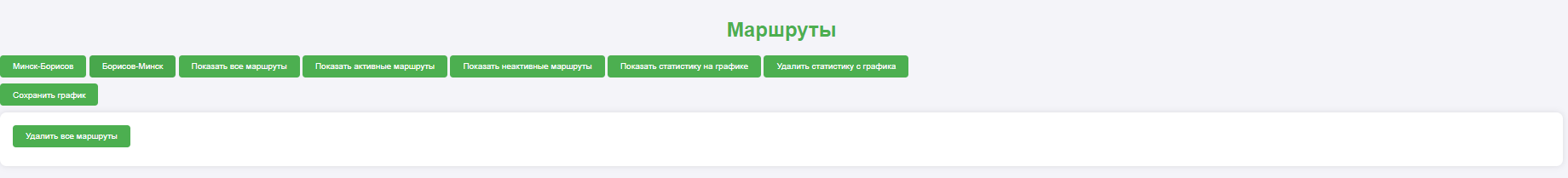


Рисунок 2.4 – Кнопки создание маршрута Минск–Борисов и Борисов–Минск

Код добавления создания нового маршрута представлен в листинге 2.1. (в прил)

|  |
| --- |
| router.post('/routes/create', checkAuth, (req, res) => {      const routeName = req.body.routeName;      const departureTime = req.body.departureTime; // Добавлено поле departureTime      const seatsAvailable = req.body.seatsAvailable;      const Price = req.body.Price;      // Проверка наличия всех необходимых данных      if (!routeName || !departureTime || !seatsAvailable) {          return res.status(400).send("Не все поля были заполнены.");      }      // Преобразование данных, если необходимо      const formattedDepartureTime = new Date(departureTime); // Преобразование времени в объект Date      // Создание нового маршрута в базе данных      const query = "INSERT INTO taxi\_route (route\_name, departure\_time, seats\_available,price) VALUES (?, ?, ?,?)";      const values = [routeName, formattedDepartureTime, seatsAvailable,Price];      database.query(query, values, (error, result) => {          if (error) {              console.error("Ошибка при создании маршрута:", error);              return res.status(500).send("Произошла ошибка при создании маршрута.");          }          console.log("Новый маршрут успешно создан.");          res.redirect('/users/isAdmin'); // Перенаправление на главную страницу      });  }); |

Листинг 2.1 – Модель создания нового маршрута

А код создания маршрута по рейсам Минск–Борисов и Борисов–Минск, с 5:00 по 22:00 на завтра представлен в приложении (потом добавлю).

**2.2 Архитектура приложения**

Архитектура веб-приложений представляет собой в первую очередь организацию системы, состоящему из компонентов, их взаимосвязи меду собой и с окружающей средой.

В случаи разрабатываемого приложения для реализации веб-приложения было выбрано архитектура клиент-сервер. Клиент - сервер - вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами. Фактически клиент и сервер — это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине. [5 ]. Эта архитектура позволяет распределить функционал и вычислительную нагрузку между клиентским и серверным приложением, обеспечивая тем самым эффективное взаимодействие между заказчиками, то есть клиентом и поставщиком услуг – организация. Сервер способен обслуживать несколько пользователей одновременно. В случаи если на сервер приходит несколько запросов одновременно, такие запросы выполняются сервером по очереди.

Клиент-серверная архитектура предлагает ряд преимуществ, включая:

* масштабируемость: Сервер может обслуживать большое количество клиентов одновременно, что позволяет масштабировать систему при увеличении обработки запросов.
* разделение обязанностей: Клиент и сервер выполняют разные задачи, что позволяет легко разделять ответственность между ними и облегчает сопровождение и модификацию системы.
* централизованное управление: Сервер является центральным узлом, который управляет данными и вычислениями, что упрощает администрирование системы.
* безопасность: Сервер может контролировать доступ к данным и ресурсам, обеспечивая безопасность системы.
* эффективная работа с данными: Клиенты могут обращаться к серверу для получения и обновления данных, что обеспечивает единообразие и целостность данных.
* использование различных технологий: Клиенты и серверы могут быть реализованы с использованием разных технологий, что позволяет выбирать наиболее подходящие средства для каждой из частей системы.

В целом, клиент-серверная архитектура обеспечивает эффективное взаимодействие между клиентскими приложениями и серверами, обеспечивая высокую производительность и удобство использования веб-приложений.

Архитектуру «клиент-сервер» принято разделять на три класса: одно-, двух- и трехуровневую. Однако, нельзя сказать, что в вопросе о таком разделении в сообществе ИТ-специалистов существует полный консенсус. Многие называют одноуровневую архитектуру двухуровневой и наоборот, то же можно сказать о соотношении двух- и трёхуровневой архитектур.[5]. Одноуровневая архитектура «клиент-сервер» (1-Tier) - такая, где все прикладные программы рассредоточены по рабочим станциям, которые обращаются к общему серверу баз данных или к общему файловому серверу. Никаких прикладных программ сервер при этом не исполняет, только предоставляет данные.

В целом, такая архитектура очень надежна, однако, ей сложно управлять, поскольку в каждой рабочей станции данные будут присутствовать в разных вариантах. Поэтому возникает проблема их синхронизации на отдельных машинах. В общем, как можно видеть из рисунка, в этой архитектуре просматривается еще один уровень - базы данных, что дает повод во многих случаях называть её двухуровневой.

К двухуровневой архитектуре «клиент-сервер» следует относить такую, в которой прикладные программы сосредоточены на сервере приложений (Application Server), например, сервере 1С или сервере CRM, а в рабочих станциях находятся программы-клиенты, которые предоставляют для пользователей интерфейс для работы с приложениями на общем сервере.

Такая архитектура представляется наиболее логичной для архитектуры «клиент-сервер». В ней, однако, можно выделить два варианта. Когда общие данные хранятся на сервере, а логика их обработки и бизнес-данные хранятся на клиентской машине, то такая архитектура носит название “fat client thin server” (толстый клиент, тонкий сервер). Когда не только данные, но и логика их обработки и бизнес-данные хранятся на сервере, то это называется “thin client fat server” (тонкий клиент, толстый сервер). Такая архитектура послужила прообразом облачных вычислений (Cloud Computing).

В трехуровневой архитектуре сервер баз данных, файловый сервер и другие представляют собой отдельный уровень, результаты работы которого использует сервер приложений. Логика данных и бизнес-логика находятся в сервере приложений. Все обращения клиентов к базе данных происходят через промежуточное программное обеспечение (middleware), которое находится на сервере приложений. Вследствие этого, повышается гибкость работы и производительность.

Трехуровневая архитектура веб-приложения представлена на рисунке 2.5

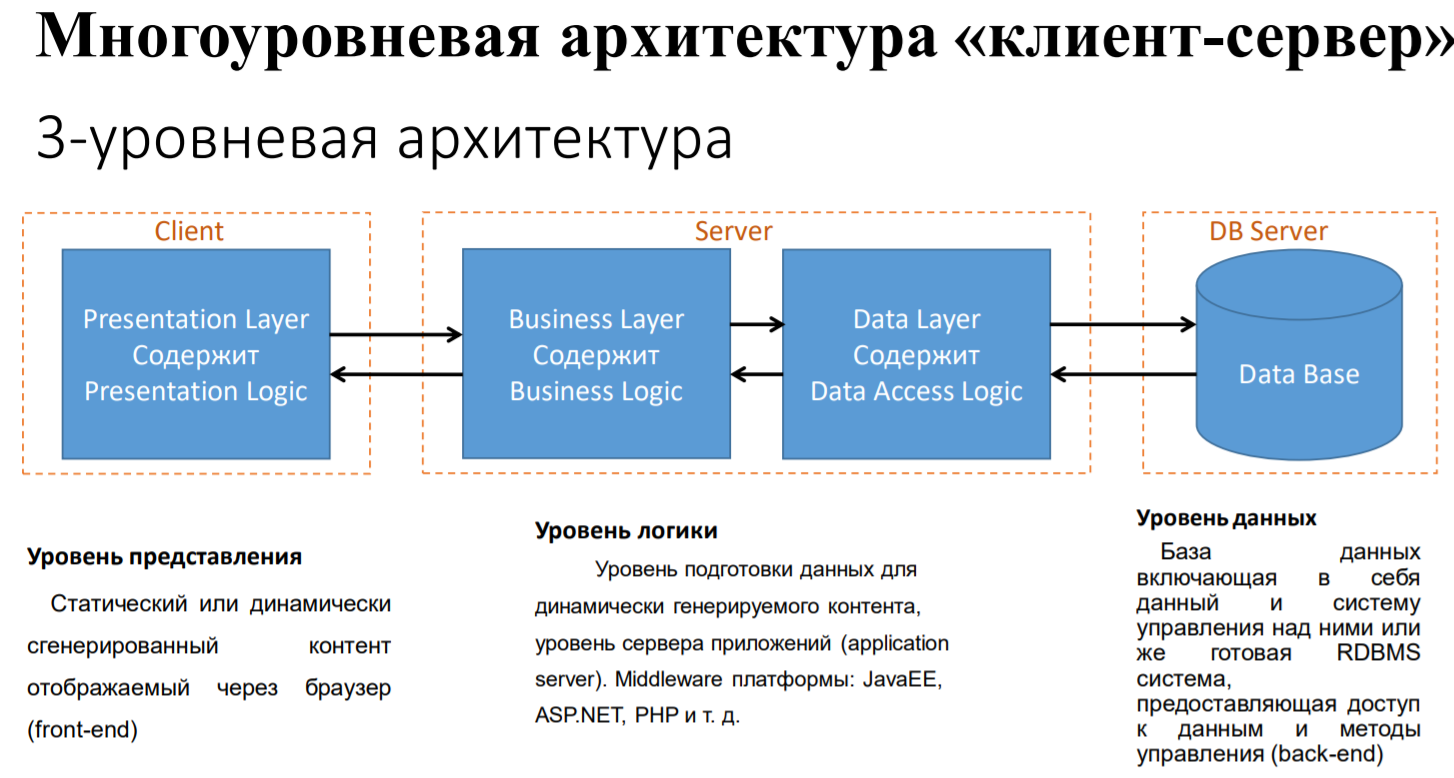


Рисунок 2.5 – Трехуровневая архитектура веб-приложения

Архитектура «клиент-сервер» - один из основных принципов работы сети Интернет. Любой веб-сайт, или приложение в Интернет работает на сервере, а его пользователи являются клиентами. Социальные сети (Фейсбук, ВК и пр.), сайты электронной коммерции (Amazon, Озон и др.) , мобильные приложения (Instagram и т.д.), устройства Интернета вещей (умные колонки или смарт-часы) работают на основе клиент-серверной архитектуры.

Хорошим примером работы системы «клиент-сервер» является автомобильный навигатор. Приложение навигации на сервере собирает данные с многих смартфонов пользователей, на которых установлены клиенты приложения. Кроме того, приложение навигации использует ещё и данные с сервера базы данных - геоинформационной системы, который предоставляет данные, например, о текущих ремонтах дорог, о появлении новых дорог и пр. Данные со многих клиентов (местоположение, скорость) обрабатывается сервером навигации и выдаётся на смартфоны пользователей в виде информации о средней скорости движения по тому или иному участку маршрута. [5]

В целом в соответствии с клиентом-серверной архитектурой веб-приложение можно разделить на три отдельных компонента:

* клиентская часть, которая состоит из пользовательского интерфейса, с которым взаимодействуют пользователи;
* серверная часть, которая обрабатывает бизнес-логику веб-приложения и отвечает на запросы пользователя;

база данных – программное обеспечение на сервере, занимающиеся хранением данных, их обработкой и выдачей по запросу. База данных располагается на стороне сервера, а северная часть приложения обращается к базе данных, извлекает данные, которые необходимы для формирования страницы, запрошенной пользователем.

Архитектура приложения представлена на рисунке 2.6.

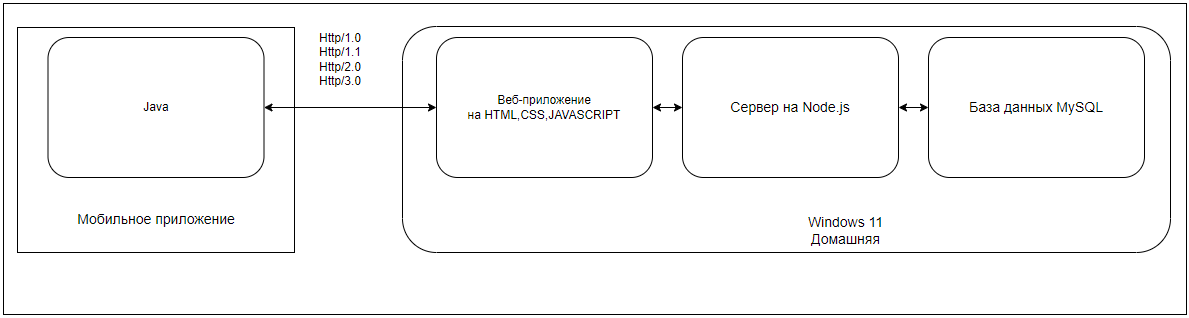


Рисунок 2.6 – Архитектура приложения

Разделение на клиент-серверную архитектуру позволяет упростить работу с приложением, поскольку настройкой каждого компонента можно заниматься отдельно и параллельно, при этом ускорив работу.

2.3 Проектирование базы данных

Перед началом разработки архитектуры приложения необходимо было спроектировать базу данных. На основе сформулированных требований к разрабатываемым приложению определятся состав информации, которая должна храниться в базе данных, а также выявляются взаимосвязи данных, размещаемых в БД. Задачей дипломного проекта создать приложение, которое будет иметь заданный функционал.

Схемы базы данных представлена в таблице 2.1 – 2.3.

Таблица 2.1 – Описание таблицы user

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | INT | Идентификатор пользователя |
| NAME | TEXT | Имя пользователя |
| PHONE | TEXT | Телефон |
| PASSWORD | TEXT | Пароль пользователя |
| ISADMIN | TINYINT (1) | Идентификатор роли пользователя на администратора |
| ISCASHIER | TINYINT (1) | Идентификатор роли пользователя на кассира |

В таблице «user» хранятся данные о пользователях, которые могут бронировать места. taxi\_route

Таблица 2.2 – Описание таблицы taxi\_route

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | INT | Идентификатор маршрута |
| ROUTE\_NAME | TEXT | Имя маршрута |
| DEPERTURE\_TIME | DATETIME | Дата отправки маршрута |

|  |  |  |
| --- | --- | --- |
| SEATS\_AVAILABLE | INT | Число мест в маршрутке |
| STATUS | VARCHAR | Статус маршрутки |

В таблице taxi\_route хранится информация о доступных маршрутах такси, и сколько свободных мест на каждом из них.

Таблица 2.3 – Описание таблицы booking

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ID | INT | Идентификатор бронирования |
| USER\_ID | INT | Идентификатор пользователя |
| ROUTE\_ID | INT | Идентификатор маршрута |
| BOOKING\_TIME | DATATIME | Время бронирования |
| BOOKING\_NAME | TEXT | Имя бронирования |
| BOOKING\_STATUS | VARCHAR | Статус бронировая |
| SEATS\_AVAILABLE | INT | Число место в маршрутке, которые забронировали |
| SEATS\_BOOKED | INT | Число место в маршрутке |

В таблице booking сохраняются данные о бронированиях пользователей для определенных маршрутов.

В базе данных данного приложения созданы для использования 3 таблицы. Скрипт создания БД и таблиц находится в Приложении (потом напишу). Также в Приложении () находиться схема базы данных. Следует отметит важность базы данных, которая играет ключевую роль в хранении, организации и обработке больших информации. Она позволяет эффективно управлять данными, тем самым обеспечивает быстрый доступ к нужным данным обеспечивает быстрый доступ к нужным данным, обеспечивает целостность и безопасность информации. Без базы данных на сервере многие приложения и веб-сайты не смогли бы работать эффективно и обработать большое количество информации. Она является основой для работы многих бизнес-процессов, а также обеспечивает надежность и стабильность работы приложений. Поэтому база данных на сервере имеет огромную важность для любой организации или компании.

2.4 Выводы по разделу

Таким образом, была спроектирована база данных, описана UML-диаграмма, некоторые основные алгоритмы, а также определен основной стек технологий, используемый в данном проекте.

Каждый выбор, осуществленный в данной главе, является осознанным и обоснованным. Продумывание всех подобных моментов в начале, исключает неопределенность в будущем. Таким образом, заранее известен список всех используемых технологий, сферы деятельности, которые покрывают и некоторые основные алгоритмы, вокруг и на основе которых потом будут появляться новые механизмы взаимодействия между модулями системы. Таким образом, создавая администратора вначале, владелец сервиса освобождается от первичной инициализации базы данный администраторской информацией, необходимой для входа, что значительно упрощает весь процесс администрации.

В разделе 2.2 построена и описана схема архитектуры приложения. Также в данном разделе было рассмотрено проектирование базы данных. Проектирование осуществлялось при помощи построения модели данных, были определены таблицы и связи между ними. В разделе также были описаны таблицы с описанием каждого из столбцов и их ограничений целостности в спроектированной базе данных.

# Реализация веб-приложения

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*3. Разработка программного средства*

Лист

1

*БГТУ 03.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_, 2024*

Одним из ключевых моментов при проектировании и создании веб-приложения является выбор архитектуры. Существует множество видов архитектур, которые наиболее часто используются для создания веб-приложений. К таким видам архитектурам относятся: клиент-серверная, трёхуровневая, Onion и другие. Разрабатываемое веб-приложение использует клиент-серверную архитектуру. К сегодняшнему дню данный вид архитектуры является наиболее распространённым.

3.1 Разработка серверной части приложения

У

Для разработки использовался фреймворк Express. Так как это популярный фреймворк для приложений Node.js, который предоставляет удобные средства для обработки HTTP-запросов, определения маршрутов и управления промежуточными обработчиками (middleware). В проекте используется middleware проверяет авторизацию пользователя с использованием JSON Web Token или коротко JWT, давайте разберем код, представленный на примере листинге 3.1 ниже.

|  |
| --- |
| const jwt = require('jsonwebtoken');  module.exports = (req, res, next) => {      try {          const token = req.headers.authorization          const decoded = jwt.verify(token, process.env.JWT\_KEY);          req.userData = decoded;          next();      } catch (error) {          return res.status(401).json({              message: 'Missing API Token'          });      }  }; |

Листинг 3.1 – Модель токена

Первая строка кода начинается с const jwt = require(‘jsonwebtoken’): этот код подключает библиотеку jsonwebtoken, которая используется для работы с JWT. module.exports = (req, res, next) => { ... }: Экспорт функции в виде middleware, которая будет принимать запрос (req), ответ (res) и функцию next, которая перейдет к следующему middleware в стеке. const token = req.headers.authorization: Получение токена авторизации из заголовков запроса. Обычно он будет передаваться в заголовке Authorization.

const decoded = jwt.verify(token, process.env.JWT\_KEY);: Расшифровка токена с использованием секретного ключа (JWT\_KEY), хранящегося в переменной среды. Если токен действителен, расшифрованные данные будут доступны в decoded.

req.userData = decoded;: Добавление расшифрованных данных в объект userData в объекте запроса. Это позволит использовать информацию о пользователе в последующих обработчиках запросов.

next();: Вызов функции next(), чтобы передать управление следующему middleware в стеке.

Обработка ошибки: если произошла ошибка при проверке токена (например, токен отсутствует, недействителен или срок его действия истек), будет отправлено сообщение об ошибке с кодом состояния 401 (Unauthorized).

По сути, этот middleware проверяет наличие и валидность JWT в заголовке Authorization. Если токен действителен, он добавляет расшифрованные данные в объект запроса и передает управление следующему middleware. Если нет, возвращается сообщение об ошибке.

Файловая структура моего приложения представлена на рисунке 3.1.

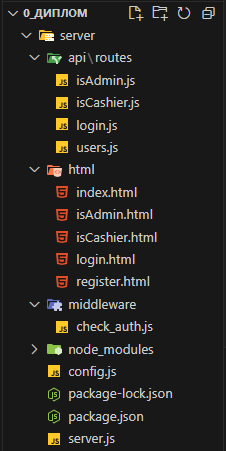


Рисунок 3.1 – Файловая архитектура приложения

В проекте существует директории с именами api\routes, html, middleware, node\_modules, а также файлы config.js, package-lock.json,package.json и server.js. Давайте их кратко разберем. Файл api\routes в Node.js выполняет важную роль в организации приложения, особенно когда я использую фреймворк Express для создания моего API. Его основная роль состоит в том, чтобы:

Определение маршрутов: Файл api/routes определяет HTTP-маршруты, которые моего приложения будет поддерживать. Эти маршруты представляют собой конечные точки, к которым клиенты могут обращаться для выполнения определенных действий.

Распределение запросов: при получении запроса от клиента, Express анализирует URL запроса и направляет его к соответствующему обработчику, определенному в маршруте. Файл api/routes указывает, какой код должен быть выполнен при обращении к определенному маршруту.

Разделение логики приложения: Размещение маршрутов в отдельном файле помогает организовать ваше приложение и разделить его на логические компоненты. Это улучшает читаемость кода и облегчает его поддержку.

Импорт контроллеров: обычно маршруты связаны с соответствующими контроллерами, которые содержат логику обработки запросов. Файл api/routes импортирует эти контроллеры и связывает маршруты с их соответствующими обработчиками.

Экспорт роутера: после определения всех маршрутов и их обработчиков, роутер экспортируется из файла, чтобы его можно было использовать в основном файле приложения или в других местах.

Таким образом, директория api/routes играет ключевую роль в организации маршрутов вашего приложения и обеспечивает правильное направление запросов от клиентов к соответствующим обработчикам.

Директория html хранит в себе страницы для отображения их в браузере или приложении пользователя.

Директория middleware – файл для авторизации пользователя.

Директория node\_modules – библиотеки, которые я использую на сервере.

Файл config.js как понятно из названия хранит в себе конфигурационные параметры.

Файл package-lock.json цель этого файла заключается в отслеживании точных версий установленных пакетов, что позволяет сделать разрабатываемый продукт стопроцентно воспроизводимым в его исходном виде даже в случае, если те, кто занимается поддержкой пакетов, их обновили [6].

Файл package.json описывает зависимости проекта. Эти зависимости используются проектом для реализации его основного функционала, они устанавливаются в ходе его развёртывания на сервере (после выхода npm 5 записи об устанавливаемых пакетах в разделе зависимостей делаются автоматически, и без использования этого флага) [6].

Файл server.js запускает мой сервер.

Распишем функционал кода файла server.js код файла смотри в листинге 2. Подключение зависимостей:

* Express.js используется для создания и настройки веб-приложения.
* Body-parser для обработки данных формы.
* Подключение к базе данных через модуль в файле config.js.
* Express-session для управления сеансами пользователей.
* dotenv для загрузки переменных среды из файла .env.

Инициализация Express и Middleware:

* Создается экземпляр Express.
* Middleware для парсинга JSON и URL-кодирования.
* Настройка сессий.
* Использование маршрута /users из userRouter.

Описание маршрута GET "/":

* Обрабатывает запросы к корневому URL.
* Извлекает параметры запроса (страница, размер страницы, откуда, куда, дата).
* Строит SQL-запрос на основе параметров и получает данные из базы данных.
* Формирует HTML-страницу с информацией о доступных маршрутах и кнопками бронирования.
* Включает обработчики событий JavaScript для динамического изменения мест отправления и назначения.

Описание маршрута POST "/booking":

* Обрабатывает запросы на бронирование мест.
* Проверяет наличие маршрута с указанными параметрами в базе данных.
* Проверяет доступность мест на выбранном маршруте.
* Создает запись о бронировании в базе данных и обновляет количество доступных мест.
* Возвращает соответствующий HTTP-код в зависимости от результата операции.

Маршрут GET "/users/logout" очищает сессию пользователя и перенаправляет на главную страницу.

Маршрут POST "/booking/:id":

* Обрабатывает запросы на бронирование мест на конкретном маршруте.
* Проверяет, авторизован ли пользователь.
* Проверяет доступность мест на выбранном маршруте.
* Создает запись о бронировании и обновляет количество доступных мест.

Запуск сервера:

* Сервер запускается на указанном порту.(изм и ;)

3.2 Подключение базы данных

В листинге 3.2 представлено подключение к базе данных. В части кода const database = require("./config"); В данной строке кода подключается модуль config, который, вероятно, содержит настройки подключения к базе данных. Предположим, что в файле config.js содержатся эти настройки. Для подключения к базе данных используется библиотека mysql. В файле config содержится следующий код представленный в листинге 3.3.

В этом коде используется модуль mysql2, который является улучшенной версией модуля mysql. Здесь также используется dotenv для загрузки переменных окружения из файла .env. Конфигурация подключения (хост, пользователь, пароль и имя базы данных) берется из переменных окружения, что обеспечивает безопасность и гибкость при работе с различными средами разработки.

Вот подробное описание подключения к базе данных и взаимодействия с ней в листинге 3.3. В начале файла мы подключаем необходимые модули, включая express, body-parser, session и dotenv. Затем мы импортируем файл с конфигурацией базы данных, который содержит код для создания пула соединений. Создаем объект приложения Express с помощью express(). Затем мы определяем порт, на котором будет работать сервер. Инициализируем сессии с помощью express-session, устанавливая секретный ключ и другие параметры. Подключаем маршрутизатор для обработки маршрутов, связанных с пользователями. Устанавливаем middleware для обработки запросов, включая парсинг JSON и URL-кодированных данных. При обработке GET-запроса на корневом маршруте (/), мы извлекаем параметры запроса (например, страницу, размер страницы, откуда, куда и дату). Затем мы формируем SQL-запрос с учетом этих параметров и выполняем его с помощью метода database.query() из нашего подключения к базе данных. После получения результатов запроса мы обрабатываем их, создавая строки таблицы с данными о маршрутах и кнопками для бронирования. После формирования страницы с данными мы отправляем ее клиенту с помощью res.send(). При обработке POST-запроса на маршруте /booking/:id, мы снова извлекаем необходимые параметры из запроса (например, идентификатор маршрута, время отправления и количество мест для бронирования). Затем мы выполняем запрос к базе данных, чтобы проверить доступность выбранного маршрута и количество доступных мест. Если бронирование возможно, мы добавляем запись в таблицу бронирований и обновляем количество доступных мест на маршруте. После успешного завершения операции мы отправляем клиенту сообщение об успешном бронировании с помощью res.status(200).send().

|  |
| --- |
| const mysql = require('mysql2');  const connectionPool = mysql.createPool({      connectionLimit: 10,      host: '192.168.1.54',//home - 192.168.1.43      user: 'root',      password: '1234567890',      database: 'bd',  })  module.exports = connectionPool; |

Листинге 3.3 – Код подключения к базе данных MySQL

Таким образом, код обеспечивает подключение к базе данных MySQL с использованием пула соединений и обработку запросов на получение данных о маршрутах и бронирование мест.

3.3 Реализация клиентской части приложения

По задумке приложение, пользователь будет осуществлять взаимодействие с серверной частью при помощи веб-страницы в браузере или при помощи данного приложения. Ниже представлено подробное описание на языке Java для приложения Android и листинг 3.4 с кодам:

|  |
| --- |
| public class MainActivity extends AppCompatActivity {  // Переменная для URL-адреса private static final String URL = "http://192.168.198.169:3000"; private WebView webView; @SuppressLint("SetJavaScriptEnabled") @Override protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);   // Скрываем верхнюю панель  if (getSupportActionBar() != null) {  getSupportActionBar().hide();  }  setContentView(R.layout.activity\_main);   webView = findViewById(R.id.webView);   // Включаем поддержку JavaScript в WebView  WebSettings webSettings = webView.getSettings();  webSettings.setJavaScriptEnabled(true);  webSettings.setLoadWithOverviewMode(true);  webSettings.setUseWideViewPort(true);    // Устанавливаем клиента WebView, чтобы перехватывать переходы между страницами  webView.setWebViewClient(new WebViewClient());   // Загружаем URL в WebView  webView.loadUrl(URL); }  @Override public void onBackPressed() {  // Проверяем, можно ли перейти назад в истории браузера  if (webView.canGoBack()) {  // Если можно, переходим назад  webView.goBack();  } else {  // В противном случае, завершаем активность  super.onBackPressed();  finishAffinity();  } }  } |

Листинг 3.4 – код приложения (в прил)

Этот код представляет собой фрагмент Android–приложения, в котором

используется компонент WebView для отображения веб–страницы. Давайте подробно разберем, что делает каждый его элемент. С начало создается константа с именем URL, которая хранит строку с URL–адресом страницы. Далее объявляется переменная WebView, которая будет использоваться для отображения страницы.@SuppressLint("SetJavaScriptEnabled") подавляет предупреждение системы об опасности включения JavaScript. Далее идет метод onCreate, в нем настраиваются основные компоненты приложения. super.onCreate(savedInstanceState) – это суперкласс для выполнения базовой инициализации активности. if (getSupportActionBar() != null) { getSupportActionBar().hide(); } скрывает верхнюю панель (ActionBar), если она существует. setContentView(R.layout.activity\_main) устанавливает макет активности, определенный в activity\_main.xml. webView = findViewById(R.id.webView); инициализирует webView, связывая его с элементом в макете с идентификатором webView. webSettings.setJavaScriptEnabled(true); включает поддержку JavaScript в WebView. webSettings.setLoadWithOverviewMode(true); подгоняет масштаб страницы к размеру экрана. webSettings.setUseWideViewPort(true); указывает, что WebView должен использовать "широкий видовой порт", что позволяет отображать веб-страницы полностью в их естественном масштабе. webView.setWebViewClient(new WebViewClient()); устанавливает WebViewClient, который перехватывает переходы между страницами, чтобы открывать ссылки внутри WebView, а не в браузере. webView.loadUrl(URL); загружает веб-страницу по указанному URL. Теперь рассмотрим метод onBackPressed, который вызывается при нажатии кнопки "Назад". if (webView.canGoBack()) { webView.goBack(); } проверяет, может ли WebView перейти назад в истории. Если да, то переходит назад. else { super.onBackPressed(); finishAffinity(); } если переход назад невозможен, вызывает стандартное поведение кнопки "Назад" и завершает все активности приложения.

В целом, этот код создает простое Android-приложение, которое загружает и отображает веб-страницу в WebView, с включенной поддержкой JavaScript и возможностью навигации назад.

По сути, WebView обладает следующими характеристиками:

* Кастомизация: Больше возможностей для кастомизации.
* Встроенность: Веб-контент отображается внутри приложения.
* Ресурсоемкость: Меньшая производительность по сравнению с Custom Tabs.
* Безопасность: требует дополнительных мер по обеспечению безопасности.(изм)

3.4 Выводы по разделу

По результатам проведенной работы в данной главе были описаны элементы, из которых состоит серверная и клиентская части разработанного веб-приложения. Между ними, как архитектурными единицами, существует одно яркое отличие, в то время как серверная часть является полностью независимым от других сервисом сервером, клиентский сервер же является полностью зависимым от серверной части приложения, так как сам по себе не несет никакой смысловой нагрузки и не способен обрабатывать никакую бизнес-логику.

Серверная часть была разработана с использованием фреймворка Express, который обеспечивает обработку HTTP-запросов и взаимодействие с данными.

Клиентская часть веб-приложения была разработана с использованием HTML, CSS, JavaScript и JAVA.

Были даны примеры и определения данным, которые составляют серверную части приложения. Описаны принципы их работы. Благодаря рассмотренным частям приложения можно представить общий процесс работы. Созданное описание упростит его понимание.

4 Анализ информационной безопасности приложения(изм)

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*4.Анализ информационной безопасности приложения*

Лист

1

*БГТУ 04.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_, 2024*

Перед анализом информационной безопасности приложений необходимо отметит преимущества и недостатки платформ, на которые разрабатывается приложение, а именно Android и IOS.

Итак, начнем с того, что выбор между платформы Android и IOS зависит от потребностей конкретного пользователя. Преимущества Android:

* Открытая платформа – Android является открытой системой, что означает большую свободу для разработчиков и пользователей в настройке и модификации устройств.
* Широкий выбор устройств – Существует множество производителей, выпускающих устройства на Android, что позволяет выбрать устройство, соответствующее бюджету и требованиям.
* Настраиваемость – Android позволяет пользователям настраивать интерфейс, устанавливать различные приложения и изменять функциональность устройства.
* Интеграция с Google – Тесная интеграция с сервисами Google, такими как Gmail, Google Calendar, Google Drive и другие.
* Многофункциональность – Android устройства часто предлагают широкий спектр функций, таких как поддержка карт памяти, сменные аккумуляторы и др.

Недостатки платформы Android:

* Фрагментация.Из-за множества производителей и версий Android, разработчики могут столкнуться с проблемой фрагментации, когда приложение работает по-разному на разных устройствах и версиях ОС.
* Безопасность. Android устройства более подвержены угрозам безопасности, чем iOS, из-за открытой природы платформы.
* Оптимизация. Некоторые приложения могут быть менее оптимизированы для Android, чем для iOS, что может привести к менее плавной работе или меньшей производительности.

Преимущества платформы iOS:

* Простота использования: iOS устройства известны своей простотой и интуитивно понятным интерфейсом, что делает их привлекательными для новичков и опытных пользователей.
* Высокая оптимизация: Приложения для iOS часто более оптимизированы для устройств Apple, что обеспечивает более плавную работу и лучшую производительность.
* Безопасность: iOS считается более безопасной платформой благодаря строгим мерам безопасности Apple и контролю над приложениями в App Store.
* Экосистема Apple: Интеграция с другими устройствами и сервисами Apple, такими как Mac, iPad, Apple Watch и iCloud, облегчает работу в экосистеме компании.

Недостатки платформы iOS:

* Ограничения: iOS имеет более строгие ограничения по сравнению с Android, что может ограничивать пользовательские возможности и настройки.
* Зависимость от экосистемы Apple: Использование iOS устройства часто означает сильную привязку к экосистеме Apple, что может быть неудобно для тех, кто предпочитает разнообразие в выборе устройств и сервисов.
* Высокая цена: Устройства Apple, особенно новые модели, обычно имеют более высокую цену по сравнению с устройствами на Android.

За 2022 год команда Positive Technologies обнаружила 216 уязвимостей в 25 парах исследованных приложений для платформ Android и iOS. Наибольшая доля уязвимостей (14%) пришлась на хранение пользовательских данных в открытом виде. Несмотря на усилия со стороны разработчиков операционных систем и сообществ по безопасной разработке приложений, этот класс уязвимостей продолжает уверенно сохранять лидерство несколько лет подряд. Описанный тренд сохранит актуальность в 2023 году, хотя использовать криптографию в мобильных приложениях сегодня очень просто: и вендорские, и опенсорсные решения облегчают разработчикам работу с криптографическими примитивами. Второе место поделили между собой уязвимости, касающиеся проверки целостности приложений и хранения конфиденциальной информации в коде (по 9%). Замыкает тройку лидеров класс уязвимостей, связанных с проверками на недоверенное окружение (8%). Наличие в приложениях вышеперечисленных уязвимостей свидетельствует о том, что разработчики недостаточно строго контролируют целостность приложений и среды их выполнения. Если добавить к этому отсутствие хорошей обфускации кода (такую комбинацию мы обнаружили в 36% приложений, исследованных в 2022 году), складывается благоприятная ситуация для злоумышленников: становится очень просто проводить качественный анализ, что, в свою очередь, упрощает создание ботов, клонов и троянов, нацеленных на конкретные продукты. [7]

Самым любопытным трендом 2022 года стало отсутствие в приложениях некоторых классов уязвимостей. Например, разработчики теперь не хранят криптографические ключи в файловой системе и не допускают ошибок, открывающих возможность обхода каталогов (Path Traversal). Уязвимость, связанная с небезопасной отправкой неявных межпроцессных сообщений, встретилась нам в исследованных приложениях в 2022 году всего лишь один раз (в предыдущем году — шесть случаев). Это связано с тем, что разработчики стали чаще применять хорошие архитектурные практики, что позволило существенно уменьшить поверхность атаки на приложения и даже полностью нивелировать некоторые типы уязвимостей. Например, в Android-приложениях, использующих подход SA, есть всего одна активность, что значительно снижает количество возможных точек входа. Разработчикам становится проще контролировать точки входа в приложение и защищать их. Мы ожидаем, что эта позитивная тенденция будет набирать обороты в 2023 году. Помогают и новые версии операционных систем: вводятся более гранулярные разрешения на выполнение системных операций, ряд разрешений можно запрашивать каждый раз. К примеру, теперь не нужно навсегда выдавать приложению разрешение на доступ к геолокации. [7]

Из текста выше можно сделать вывод о том, что уязвимости в мобильных приложениях остаются серьезной проблемой, несмотря на усилия разработчиков и сообществ по безопасности. Однако, некоторые положительные тенденции также наблюдаются. Например, разработчики стали чаще применять хорошие архитектурные практики, что привело к существенному уменьшению поверхности атаки на приложения и даже полному исчезновению некоторых классов уязвимостей. Это может быть связано с более строгим контролем целостности приложений и использованием новых функций операционных систем, таких как более гранулярные разрешения на выполнение системных операций. Однако, все еще остается много работы по улучшению безопасности приложений, особенно в контексте хранения пользовательских данных и обеспечения защиты от анализа и эксплуатации кода.

Исходя их вывода выше приоритетным целью для разработчика приложения является безопасность данных и конфиденциальность клиента. Ниже представленные ключевые меры безопасности, которые принимают, по мнению разработчика рассматриваемого приложения, для защиты личной информации и обеспечения безопасного использования приложения:

* Шифрование данных
* Аутентификация и авторизация
* Обновление безопасности
* Мониторинг безопасности
* Обучение персонала

Давайте разберем каждый пункт, что находиться выше. Шифрование данных: Все данные, передаваемые между вашим устройством и нашими серверами, защищены с использованием современных методов шифрования, чтобы предотвратить несанкционированный доступ к вашей информации.

Аутентификация и авторизация: Мы применяем механизмы аутентификации и авторизации для обеспечения доступа только к авторизованным пользователям. Это помогает предотвратить несанкционированный доступ к вашему аккаунту и личной информации.

Обновления безопасности: Мы регулярно обновляем наше приложение и серверы, чтобы исправлять обнаруженные уязвимости и улучшать безопасность системы в целом.

Мониторинг безопасности: Мы внимательно отслеживаем активность в наших системах, чтобы быстро обнаруживать и реагировать на любые подозрительные действия или попытки несанкционированного доступа.

Обучение персонала: Наш персонал проходит обучение по вопросам информационной безопасности, чтобы быть готовыми к распознаванию и предотвращению угроз безопасности данных.

Таже обходимо подметить и основные уязвимости приложений для Android и так как у разработчика нету достаточного опыта, дальнейший текст берётся из книги Positive Technologies 2023.

Существует довольно много видов уязвимостей мобильных приложений, но среди них можно выделить некоторые обобщенные типы, которые покрывают основной ландшафт. Самые популярные уязвимости связаны с небезопасным хранением пользовательских данных и данных приложения. Для их возникновения разработчику даже не нужно ничего делать — достаточно просто сохранять конфиденциальную информацию в незашифрованном виде. Некоторые разработчики, думая о безопасности, сохраняют такие данные во внутренней директории приложения — так называемой песочнице. Но в ряде случаев этого оказывается недостаточно. [7]

Например, когда на устройстве доступно выполнение команд от имени суперпользователя (root). Этой возможности, как правило, нет в стандартной поставке ОС, но продвинутые пользователи добавляют ее самостоятельно: для использования специфических приложений или для улучшения UX операционной системы. Далее возможен такой сценарий условно легитимное приложение запрашивает повышенные привилегии для выполнения своей основной функции, а получив эти привилегии, начинает делать то, чего пользователь от него не ожидает. Например, копировать данные из песочниц других приложений. [7]

Другой пример — наличие уязвимостей, позволяющих читать содержимое песочницы из другого приложения. В этом случае вредоносному приложению не нужны повышенные привилегии. Оно проэксплуатирует эту уязвимость и получит доступ к незашифрованным данным во внутренней директории целевого приложения. Именно поэтому данные должны быть зашифрованы. Благо сейчас это сделать очень просто и не нужно быть экспертом в криптографии. Достаточно использовать решения вендора и следовать практикам, описанным в официальной документации. [7]

Еще один, не менее интересный тип уязвимостей — отсутствие контроля целостности исполняемых файлов и защиты от модификации. Здесь разработчику тоже достаточно ничего не делать и, соответственно, никакой защиты не будет. Это позволит злоумышленникам модифицировать исходное приложение и распространять его под видом оригинального. Казалось бы, кому может понадобиться скачивать неоригинальное приложение? На самом деле, много кому. Помимо таких банальных потребностей, как вырезание рекламы и механизмов контроля платных функций, пользователям может понадобиться запускать приложения на устройствах с модифицированными прошивками. Такие прошивки очень часто имеют возможность выполнения команд от имени суперпользователя, и банковские приложения, содержащие соответствующие механизмы безопасности, отказываются работать на таких устройствах. Следовательно, нужно убрать из банковского приложения все эти проверки, чтобы оно работало на таких прошивках. Такой деятельностью, как правило, занимаются энтузиасты ради спортивного интереса. Но точно так же могут действовать злоумышленники, и тогда в банковском приложении не только исчезнут проверки, но и появится код, который ворует данные для входа в аккаунт. Защититься от таких модификаций довольно сложно, и, как правило, это требует дополнительного приобретения специализированных утилит упаковщиков, которые усложняют обратную разработку (reverse engineering) и позволяют «потратить» очень много времени злоумышленника на исследование механизмов безопасности. Можно попробовать написать нужные механизмы безопасности самостоятельно, но это требует высокой квалификации, выходящей за рамки компетенции обычного разработчика мобильных приложений.[7]

В мире сетевого взаимодействия особое внимание следует уделить уязвимостям, которые могут возникнуть при неудовлетворительном обеспечении защиты.

Отдельно стоит упомянуть уязвимости, связанные с сетевым взаимодействием. Многие разработчики останавливаются на использовании защищенного протокола HTTPS и не добавляют дополнительной защиты. При определенных условиях это позволяет злоумышленнику, контролирующему канал связи, выполнить MITM-атаку на приложение и получить конфиденциальную информацию. Один из базовых сценариев такой атаки выглядит следующим образом: при подключении к недоверенной сети Wi-Fi пользователю показывают поддельный captive portal и предлагают установить на устройство SSL-сертификат. После этого злоумышленник может перехватывать весь трафик со смартфона пользователя. Для защиты от этой атаки обычно применяют технику закрепления сертификата. Фактически это жестко закодированный в мобильном приложении сертификат или цепочка сертификатов легитимного сервера. Существуют и другие вариации этой защиты, но все они направлены на то, чтобы предотвратить обмен данными с другим сервером.[7]

Также необходимо отметит и уязвимости Android системы, так как приложение написано и рассчитано, что им будут пользоваться люди через эту систему, более подобнее написано в книге Positive Technologies 2023, ниже приведен текст.

Кроме того, для Android, особенно ранних версий (4.1.1 и ниже), очень характерны уязвимости, связанные с взаимодействием приложений между собой (inter-process communication) и с ненадлежащим использованием возможностей ОС и фреймворка. Долгое время документация по этим возможностям оставляла желать лучшего, а некоторые части и вовсе были не задокументированы. Наряду с отсутствием внятных руководств и описания лучших практик это заставляло разработчиков писать своеобразный код, зачастую переизобретая механизмы, которые уже были в ОС. Очень показательный пример — флаг android:exported, который управляет доступностью какого-либо компонента приложения для вызова другими приложениями. Именно в версии Android 4.1.1 и ниже этот флаг по умолчанию выставлен в true, а значит, все компоненты, где этот флаг не установлен разработчиком, явно будут доступны для вызова другими приложениями. Это может привести к обходу механизмов аутентификации, например экрана с вводом PIN-кода, или к эксплуатации других уязвимостей за счет прямого взаимодействия с теми компонентами, которые разработчик задумал как внутренние и недоступные извне. Такова концепция приложений для Android. В них не должно быть какой-то одной обязательной точки входа, таких точек может быть несколько. Поэтому очень важно уменьшать количество внешних компонентов, а в оставшихся — жестко контролировать любое общение с «внешним миром». [7] Еще одним самостоятельным типом уязвимостей можно считать хранение в коде ключей доступа к API технических сервисов. Например, систем аналитики и сбора ошибок, возникающих в приложении, облачных баз данных и прочих внешних сервисов. Часто такие сервисы предоставляют ключи с разным типом доступа, так как разработчики этих сервисов понимают, что они будут использоваться в недоверенной среде. Но разработчики приложений по разным причинам все равно оставляют в коде ключи с «лишними» привилегиями. Опасность утечки таких ключей зависит от конкретного случая, но, например, получение серверного ключа для Firebase Cloud Messaging позволит злоумышленнику отправлять произвольные push-сообщения всем зарегистрированным пользователям приложения [7]

Однако не стоит забывать о самом важной уязвимости это человеческий фактор. Также можно привести и актуальные виды уязвимости, которые также представлены ниже и взяты из книги Positive Technologies 2023 ввиду небольшого опыта разработчика рассматриваемого приложения.

Несмотря на все усилия Google и сообществ по безопасной разработке, уязвимости в приложениях все еще встречаются. Помимо уже описанных выше уязвимостей, которые условно можно назвать «простыми», потому что они существуют как бы сами по себе, сейчас все чаще встречаются «сложные». Это уже не уязвимости как таковые, а скорее полноценные атаки, которые объединяют в цепочку несколько уязвимостей и (или) особенностей работы приложения и фреймворка Android. Причин этому несколько. Помимо повышения защищенности самой платформы, растет сложность приложений, и данные, попадающиев них извне, часто проходят довольно длинную цепочку преобразований. А это, в свою очередь, приводит к тому, что на каком-то из этих этапов цепочка эксплуатации может прерваться: например, разработчик для каких-то своих нужд преобразовал данные так, что уязвимость стала неэксплуатируемой. О безопасности он при этом мог вообще не думать.[7]

Актуальными остаются уязвимости, связанные с локальной аутентификацией: PIN-кодом, биометрией, вторым фактором. Эти средства защиты можно обойти из-за ошибок в реализации или из-за недостаточного понимания разработчиками концепций, заложенных в фреймворк. В случае с локальной реализацией входа по PIN-коду разработчики порой забывают сохранять количество использованных попыток входа. В этом случае можно обнулять счетчик попыток простым перезапуском приложения. И это встречается чаще, чем может показаться на первый взгляд. В чуть более сложном варианте помогает перевод системного времени, который также может плохо детектироваться логикой приложений; он приводит к сбросу количества попыток ввода. Обход биометрии чуть более сложен, но все еще возможен, если приложение отображает биометрическое диалоговое окно для проверки предъявленных данных. При определенных условиях можно скрыть такое окно и попасть в приложение. Это возможно, потому что предъявление биометрии не связано ни с какими криптографическими операциями с данными приложения и отмена диалога не влияет ни на какие внутренние процессы аутентификации. А возможность обхода второго фактора аутентификации очень сильно зависит от логики приложения. Из недавних примеров — обход второго фактора в TikTok из-за случайного тайм-аута на сервере при нескольких неправильных попытках входа в определенной последовательности.[7]

Однако не все так плохо, ведь кроме разработчика самих приложения безопасностью озабочены и сами разработчики операционных систем. Так же ниже представлен текст взят из книги Positive Technologies 2023, который более подробно описывает куда все движется.

Android не стоит на месте, и его механизмы безопасности постоянно совершенствуются. Но не все проблемы можно решить технически, и порой их приходится решать «административно». Так, начиная с Android 14 приложения, которые таргетируются на версию Android SDK ниже 23 (Android 6.0), не могут быть установлены. Проблема в том, что злоумышленники намеренно занижают версию SDK во вредоносных приложениях, чтобы эксплуатировать хорошо известные недостатки системы благодаря механизму обратной совместимости. Приложения тоже меняются. Появляется все больше кросс-платформенных приложений, упрощается процесс разработки под несколько операционных систем сразу. Но за все нужно платить, и кроссплатформенные приложения, помимо ошибок, характерных для конкретной платформы, имеют свои особенности поведения, которые также могут быть проэксплуатированы злоумышленниками. Проблема здесь еще в том, что инструменты и библиотеки для разработки таких приложений пока далеки от совершенства, а то и отсутствуют вовсе. Поэтому некоторые функции разработчикам приходится реализовать самостоятельно, что тоже чревато ошибками, особенно при проведении криптографических операций или использовании некоторых протоколов.[7]

Разработка таких приложений всегда выполняется на определенном слое абстракции, когда механизмы конкретной платформы скрыты от программиста. При желании, конечно, он может добраться до этих механизмов и взаимодействовать с ними напрямую. Но тут появляется другая проблема: хороший разработчик приложений для Android вряд ли глубоко разбирается в механизмах безопасности платформы iOS. Верно и обратное. Все эти факты плюс отсутствие хорошо задокументированных лучших практик безопасной разработки кросс-платформенных приложений приводят к появлению довольно простых и очевидных уязвимостей. Например, в одном из таких приложений мне удалось обнаружить несколько ключей доступа к API внешних систем, которых там быть вообще не должно. Они просто не могли бы попасть в приложение в таком виде, если бы оно было разработано с использованием нативного подхода.[7]

Короче говоря, война брони и снаряда продолжается. Появляются новые возможности ОС, в них находят уязвимости, эти уязвимости закрываются, но находятся способы обхода защиты. Все это работает как постоянно эволюционирующий живой организм. Я описал лишь малую часть происходящего, чтобы показать, какой путь прошли уязвимости в приложениях для Android и какое влияние они оказали на развитие операционной системы. Разработчикам приложений я бы порекомендовал внимательно следить за новыми механизмами безопасности, которые появляются в Android, и начинать их применять как можно скорее, чтобы защитить пользователей. В свою очередь, пользователям нужно смотреть на происходящее в их устройстве критическим взглядом и помнить, что, если вам хоть на секунду показалось, что что-то не так, значит, действительно что-то не так. Все это очень многомерная история, и поэтому лучшее, что мы как специалисты по анализу защищенности мобильных приложений можем сделать, — это продолжать искать уязвимости в мобильных приложениях и ОС, совершенствовать способы защиты и давать разъяснения разработчикам, чтобы сделать эту часть мира чуточку безопаснее [7]

В разработанном сервере используется для безопасности подача секретного ключа в сеанс представлено в листинги 4.1.

|  |
| --- |
| app.use(session({      secret: '12K34L', // Секретный ключ для подписи сеансов      resave: false,      saveUninitialized: true  })); |

Листинг 4.1 – инициализация сеансов

В результате каждый пользователь, который взаимодействует с приложением, будет иметь уникальный идентификатор сеанса, который будет храниться в куках(Cookies) и отправляться на сервер с каждым запросом. Это позволяет моему серверу идентифицировать и отслеживать состояние сеанса для каждого пользователя.

Для начало необходимо разобраться что такое куки или Cookies.

Cookies – это файлы на стороне клиента, которые содержат информацию о пользователе. Cookie-файл уничтожается по истечении срока, установленного для него. Официальный (согласно стандарту RFC) максимальный размер файла cookie составляет 4 КБ. Файл cookie не зависит от сеанса.

И тут же встает вопрос что такое Сеан. Сеансы – это файлы на стороне сервера, которые содержат информацию о пользователе. Сеанс заканчивается, когда пользователь закрывает свой браузер или очищает кэш. Размер данных сессии может зависеть от конфигурации сервера и используемого механизма управления сессиями. Сеанс зависит от Cookie.

Также необходимо отметит, что сеанс идентифицирует серию запросов-ответов. В рамках сеанса сервер запоминает контекст. Сеансовый ключ (для шифрования) устанавливается в рамках одного сеанса.

Разработчик идентифицирует и отслеживает состояние сеанса для каждого пользователя в нескольких местах:

В маршруте GET /users/logout, где вы очищаете сессию пользователя, в POST /booking/:id, где пользователь бронирует места, в GET /profile, где пользователь получает информацию о себе и брони на себя. В /profile/change/:bookingId для изменения бронирования. В /profile/cancel/:bookingId - Метод для отмены бронирования. В /profile/update – метод изменения в своей информации. Также я хеширую пароль смотри в листинг 4.2.

|  |
| --- |
| bcrypt.hash(password, 10, (error, hashedPassword) =>  {          if (error)  {              console.error("Ошибка при хешировании пароля:", error);              return res.status(500).send("Ошибка при хешировании пароля");} |

Листинг 4.2 – Хешируем пароль

Также используются middleware для проверки авторизации пользователя с использованием JSON Web Token или коротко JWT, представленный на примере листинге 3.1.

4.1 Выводы по разделу

В тексте представлено описание мер безопасности, применяемых в приложении. Использование секретного ключа в сеансе обеспечивает уникальный идентификатор для каждого пользователя, что позволяет серверу идентифицировать и отслеживать состояние сеанса. Такой подход обеспечивает безопасность пользовательских данных и действий, так как каждый запрос сопровождается уникальным идентификатором сеанса. Кроме того, в тексте упоминается использование хеширования паролей с помощью bcrypt, что повышает безопасность хранения паролей пользователей. Дополнительно, middleware для проверки авторизации пользователя с использованием JSON Web Token (JWT) также улучшает безопасность, обеспечивая аутентификацию и авторизацию запросов. Таким образом, применение указанных мер безопасности способствует защите пользовательских данных и обеспечивает безопасность взаимодействия с приложением.

5 Тестирование приложения

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*5. Тестирование программного средства*

Лист

1

*БГТУ 05.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_, 2024*

Тестирование программного обеспечения – проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом. Цель тестирования – проверка соответствия ПО предъявляемым требованиям, обеспечение уверенности в качестве ПО, поиск очевидных ошибок в программном обеспечении, которые должны быть выявлены до того, как их обнаружат пользователи программы.

Существующие на сегодняшний день методы тестирования ПО не позволяют однозначно и полностью устранить все дефекты и ошибки и установить корректность функционирования анализируемой программы особенно в закрытых частных программах. Поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого ПО.

В программировании есть термин – баг. Баг – используется для обозначения ошибки, и характеризуется неожиданным поведением логики веб-приложения.

Тестировщик, планируя процесс тестирования составляет тест-кейсы. Обычно тест-кейс содержит:

* описание того, что предполагается тестировать;
* последовательность действий;
* ожидаемый результат выполнения логики.

Выполняя тест-кейсы тестировщик, получает результат и сравнивает его с ожидаемым результатом и, если результаты не совпали – тестировщик констати­рует наличие ошибки и сообщает об этом разработчику.

Для отслеживания багов в приложении, часто применяются баг-трекинговые системы. После того как разработчик исправил баг, программное средство направ­ляется на тестирование, тестировщик снова проходит шаги, требуемые в тест-кейсе.

Для отображения тестируемых элементов составлена таблица 4.1 тест-кейсов.

Таблица 5.1 – Описание функциональных тест-кейсов

|  |  |  |
| --- | --- | --- |
| Требование | Ожидаемый результа | Статус |
| 1 | 2 | 3 |
| Бронирование места для пользователя | Успешное бронироване места | Успешно |
| Корректировка количества забронированых мест | Изменить количество мест с 5 мест на 1 место | Успешно |
| Удаление брони для пользователя | Удаление место | Успешно |

У

Продолжение таблицы 5.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Создание брони на маршрутное такси для администратора | Создан новая бронь | Успешно |
| Корректировка списка брони для администратора; | Изменение даты брони в списке | Успешно |
| Удаления списка брони для администратора | Удалено | Успешно |
| Изменения статуса маршрута для кассира пользователя | Изменено | Успешно |
| Просмотр списка маршрута для кассира, пользователя и администрато-ра | Просмотр списка успешный | Успешно |
| Попытка зайти на страницу Администратора без авторизвации через url в браузаре | Выдал стараницу авторизации пользователя | Успешно |
| Попытка зайти на страницу Администратора через url в браузаре | Выдал страницу,которая уведомляет о том что пользоваеть не имеет право на вход. | Успешно |
| Попытка зайти на страницу Кассира без авторизвации через url в браузаре | Выдал стараницу авторизации пользователя | Успешно |
| Попытка зайти на страницу Администратора через url в браузаре | Выдал страницу,которая уведомляет о том что пользоваеть не имеет право на вход. | Успешно |
| Попытка закать бронь, не авторизированного пользователя | Выдал стараницу авторизации пользователя | Успешно |
| Зайти на профиль пользователя и изменить/отменить определенною бронь. | Изменено количесто с 5 мест на 2 и успешно удалено бронь | Успешно |

В таблице, приведенной выше, дано описание функциональных тестов, прове­денных на этапе тестирования.

6 Руководство программиста

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Росляков Д.П.*

*Уласевич Н.И.*

*.*

Ф.И.О.

Подпись

Дата

*6. Руководство пользователя*

Лист

1

*БГТУ 06.00.ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_, 2024*

В данном разделе описывается руководство для зарегистрированных пользователей и администратора и модератора. У каждого из описанных выше пользователей имеется свой функционал.

6.1 Роль пользователя

Перед началом работы с программой посетителю необходимо получить роль для допуска к функционалу необходимо авторизоваться.

При переходе на сайт приложения пользователь попадает на главную страницу, которая показана на рисунке 6.1, в данном примере используется устройство HONOR X8 и написанное приложение для использования рассматриваемого приложения.

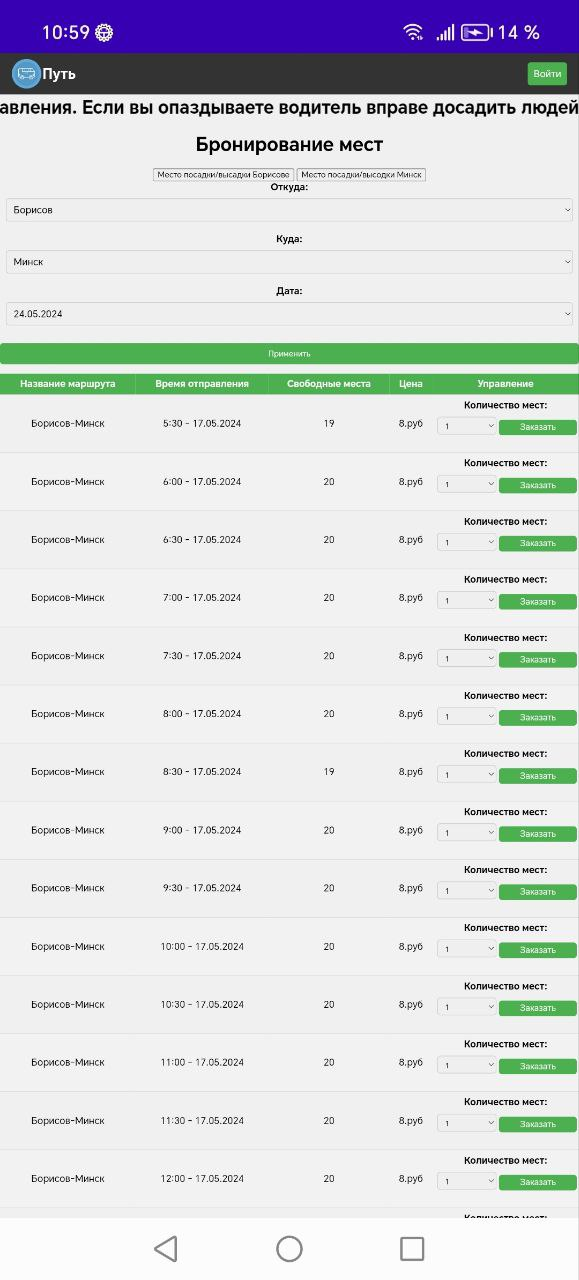


Рисунок 6.1 – Начальная страница приложения

На ней отображена меню, название проекта, две кнопки показывающие места

посадки и высадки в городах Борисов и Минск, при нажатии на них появляется карта с меткой, где будет показано место стоянки маршрутки, также есть сортировка по городам и дате, также внизу есть список всех действующих маршрутов, в данном рисунке 6.1 используется тестовая модель для того, чтоб не заполнять внимание. Для того чтоб использовать приложение для бронирования необходимо зарегистрироваться или войти в профиль для этого необходимо нажать на кнопку войти, после этого мы переходим на страницу регистрации пользователя, показано на рисунке 6.2.(изм изоб)

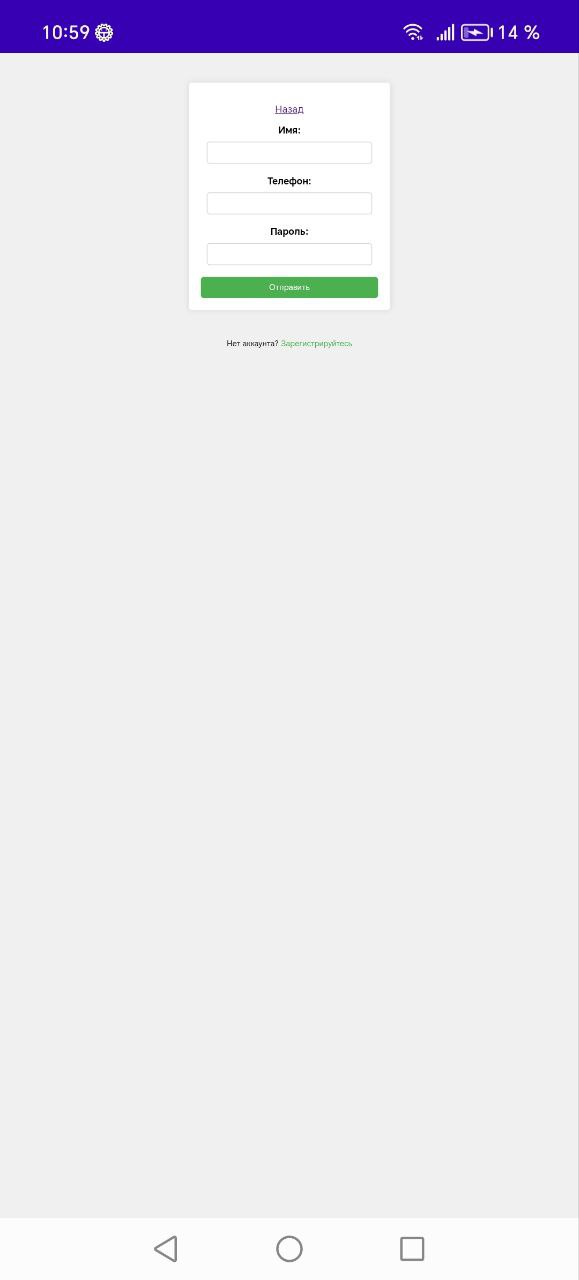


Рисунок 6.2. – Страница авторизации

На ней отображена форма для авторизации, в котором надо ввести имя, телефон и пароль. После заполнения всех необходимых полей и успешной авторизации пользователь получает доступ к полному функционалу приложения, согласно его роли. После успешной авторизации пользователь переносит на главную страницу.

На главной странице можно перейти на страницу профиля пользователя, где можно посмотреть историю бронирования и изменить свой профиль, оказано на рисунке 6.3.

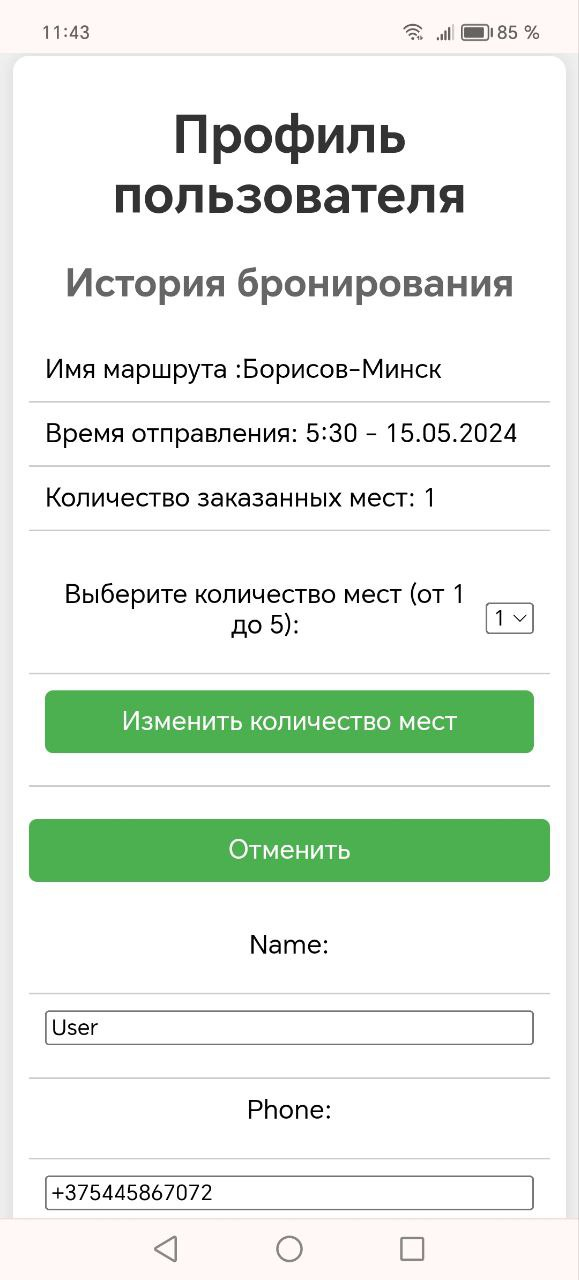


Рисунок 6.3 – Профиль пользователя

Для того что выйти из профиля можно нажать на кнопку назад или пролистать вниз и нажать на кнопку назад. После чего пользователь перейдет на главную страницу, где он может забронировать место на маршрутное такси. Для того чтоб забронировать пользователю необходимо выбрать количество мест, по умолчанию установлено 1 место, далее необходимо нажать на кнопку заказать. После заказа идет переадресация страницу, где уведомляется пользователь заказал или нет, ввиду того что у него уже есть бронь на это место.

6.2 Роль пользователя кассира

Пользователь с ролью кассира имеет доступ к элементам которых нету у администратора и пользователя, что существенно различает его.

В его обязанности входит визуальное отслеживание прибытия пользователя, расчет с ним пометка его в приложении. Для входа в приложения кассир проходит те же этапы, что и пользователь на рисунках 6.1 и 6.2. Затем он попадает на страницу для кассиров смотри рисунок 6.4. Для данного примера используется устройство HONOR X8.

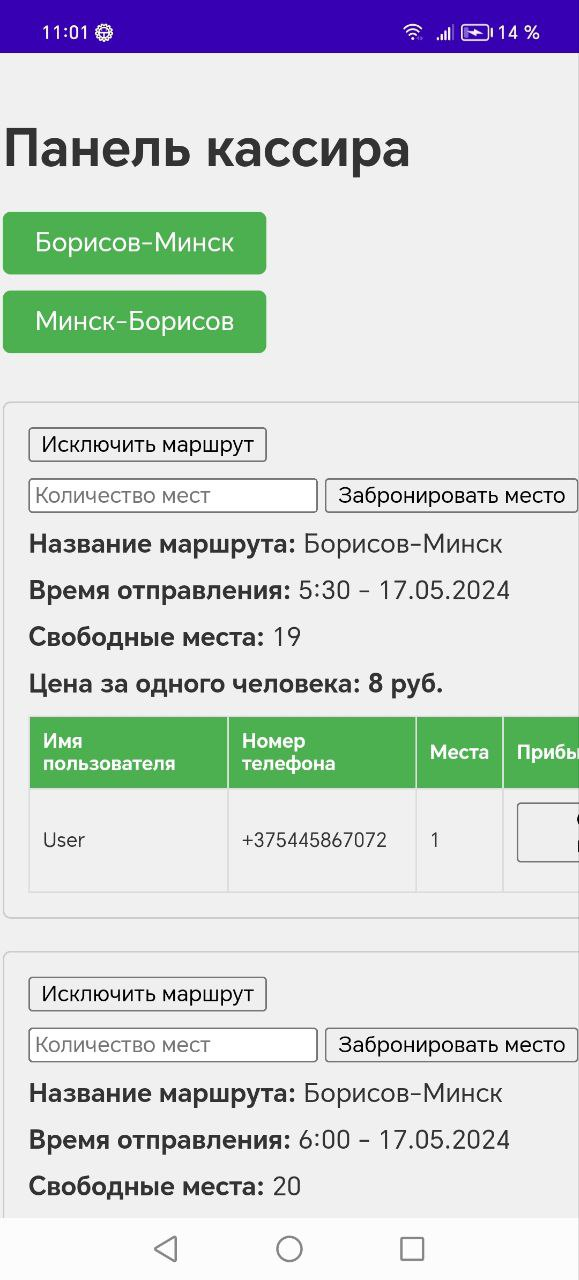


Рисунок 6.4 – Главная страница Кассира

Пользователь с правами кассира имеет возможность отмечать прибытие пассажиров, осуществлять бронирование мест и исключать маршрут. Кроме того, в системе доступна информация о пользователе, который совершил бронирование места. Также следует отметь, что пользователь кассир проводит оплату только наличными и выдает чек.

Для бронирования места на соответствующий маршрут кассир вводит количество мест и нажимает на кнопку забронировать место. При нажатии на кнопку отметить прибытия пользователь пропадает с таблицы, сделано это с целью отметить, что данный пользователь прибыл на место.

6.3 Роль пользователя администратор

Пользователь с ролью администратора имеет доступ к элементам которых нету у кассира и пользователя, что существенно различает его.

В его обязанности входит создание, редактирования и удаления маршрутов, создания графика заполнения мест маршрутов представлено на рисунке 5.5.

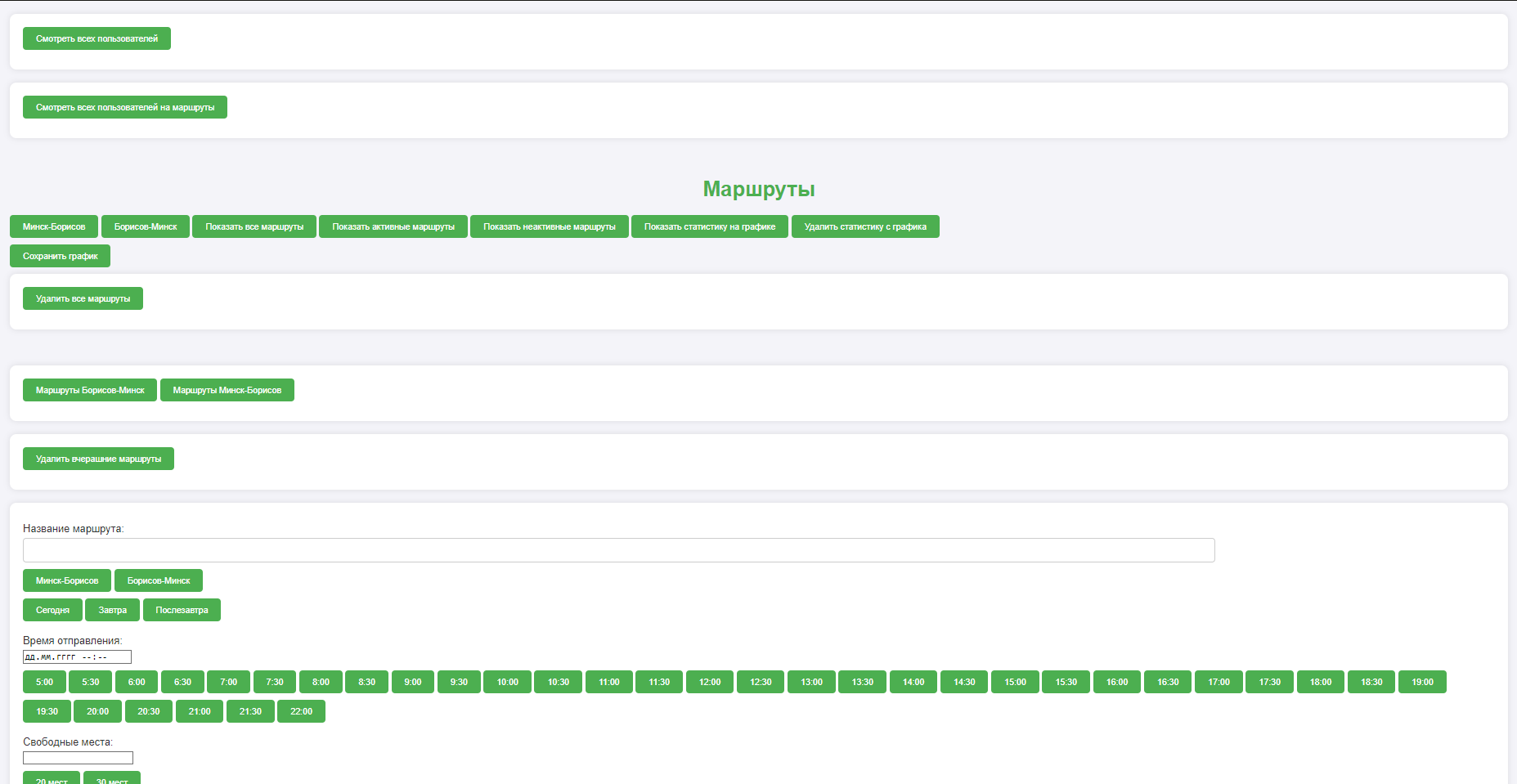


Рисунок 6.5 – Главная страница Администратора.

Для создания нового маршрута администратору нужно выполнить следующие шаги:

1. Нажать на кнопки выбора маршрута "Минск-Борисов" или "Борисов-Минск".

2. Выбрать дату, нажав на кнопку "Сегодня", "Завтра" или "Послезавтра".

3. Выбрать удобное время из диапазона 5:00-22:00.

4. Выбрать количество свободных мест - 20 или 30.

5. Указать цену за место.

6. Нажать на кнопку "Создать маршрут".

Этот процесс поможет администратору удобно и быстро создать новый маршрут с необходимой информацией.

Чтобы внести изменения в данные о маршруте, следует:

1. Из таблицы выбрать соответствующий маршрут.

2. В разделе редактирования выбрать необходимое поле для изменений.

Кроме того, у пользователя есть возможность осуществлять бронирование мест для существующих пользователей, просматривать всю базу пользователей, а также создавать, удалять и назначать роли, что показано на рисунке 6.6.

Для осуществления заказа в таблице необходимо найти столбец 'Заказать', провести поиск нужного пользователя и выбрать его, после чего нажать кнопку 'Забронировать'.

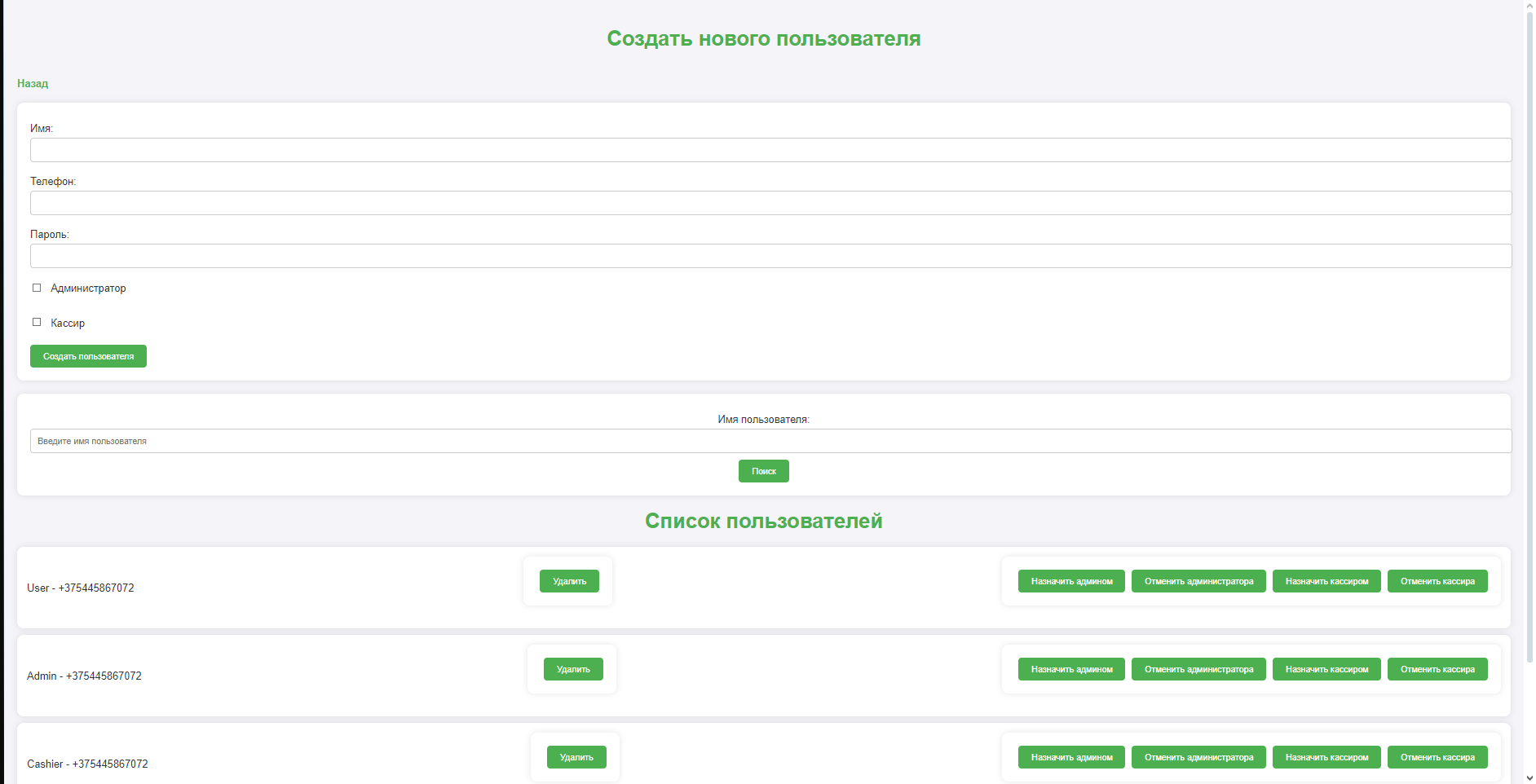


Рисунок 6.6 – Страница создания пользователя для администратора

Следует отметит, что администратор будет использовать приложение в браузере на компьютере.

6.4 Выводы по разделу

Цель данного раздела заключалась в создании руководства программиста, в соответствии с разработанным программным продуктом.

В разделе представлены основные моменты работы с приложением для пользователя, кассира и администратора. В нем содержится подробное описание того, что каждая роль может делать в пределах данного приложения. Также описан порядок действий для достижения постеленных целей при выполнении работы.

**7 Технико-экономическое обоснование проекта**

Утв.

Н. контр.

Пров.

Разраб.

Н. контр.

Провер.

*Блинова Е.А.*

Разраб.

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

*Николайчук А.Н.*

Подпись

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

*7. Технико-экономическое   
обоснование проекта*

Лист

1

*БГТУ 07.00.ПЗ*

Лит.

Листов

*БГТУ 74218034, 2023*

Листов

**7.1 Общая характеристика разрабатываемого программного средства**

Во время разработки дипломного проекта использовалась кроссплатформенная среда исполнения с открытым кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язые JavaScript, называемой Node или более формально Node.js. Разработанное программное средство представляет собой веб-приложение, позволяющее пользователям осуществлять бронирование мест на удобное время.

Данный раздел служит для определения затрат, произведенных на всех стадиях разработки программного средства.

Разработка данного приложения направлено на достижения нескольких целей: удобство и комфорт для пользователей.

Стратегия монетизации предполагает продажу приложения заказчику и имущественных прав на него.

**7.2 Исходные данные и маркетинговый анализ**

Созданная система не является новой, ввиду наличия большого количества аналогов. Она принадлежит ко второй группе программ по степени сложности. Источниками исходных данных для расчётов выступают действующие законы и нормативно-правовые акты. Исходные данные для расчёта стоимости разработки приведены в таблице 7.1.(изм)

*Соболевский А.С.*

Консульт.

У

Таблица 7.1 – Исходные данные для расчёта

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование показателя | Единица измерения | Условные обозначения | Норматив |
| Численность разработчиков | чел. | Чр | 1 |
| Норматив дополнительной заработной платы | % | Нд.з. | 15 |
| Ставка отчислений в Фонд социальной защиты населения | % | Нф.с.з.н. | 34,6 |
| Ставка отчислений по обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний | % | Нб.г.с. | 0,2 |

|  |  |  |  |
| --- | --- | --- | --- |
| Стоимость одного машино-часа | руб. | См.ч. | 0,05 |
| Норматив прочих затрат | % | Нп.з. | 20 |
| Норматив общепроизводственных и общехозяйственных расходов | % | Нобп, обх | 30 |
| Норматив расходов на сопровождение и адаптацию | % | Нр.с.а. | 10 |
| Уровень рентабельности | % | Урент. | 30 |
| Ставка НДС | % | Нндс. | 20 |
| Коэффициент изменения скорости обработки информации | ед. | Кск. | 0,6 |
| Тарифная ставка I разряда в организации за месяц | руб. | ОТI | 780 |
| Тарифный коэффициент | ед. | Кт. | 3,04 |
| Коэффициент естественных потерь рабочего времени | ед. | Кп. | 1,2 |
| Коэффициент премирования | ед. | Кпр. | 1,3 |
| Норматив расхода машинного времени за отладку 100 строк исходного кода | маш. часов | Нм.в. | 12 |

Далее эти данные понадобятся для расчета всех необходимых параметров в процессе экономического анализа приложения.

В частности, для определения рентабельности и определения себестоимости разработки мобильного приложения для бронирования мест на маршрутное такси.

Расчет рентабельности приложения включает в себя оценку всех доходов, связанных с его использованием, а также расходов, необходимых для его разработки.

# 7.3 Методика обоснования цены

В современных рыночных экономических условиях программное средство (ПС) выступает преимущественно в виде продукции организаций, представляющей собой функционально завершённые и имеющие товарный вид, реализуемые покупателям по рыночным отпускным ценам. Все завершённые разработки являются научно-технической продукцией.

Широкое применение вычислительных технологий требует постоянного обновления и совершенствования ПС. Выбор эффективных проектов ПС связан с их экономической оценкой и расчётом экономического эффекта, который может определяться как у разработчика, так и у пользователя.

У разработчика экономический эффект выступает в виде чистой прибыли от реализации ПС, остающейся в распоряжении организации, а у пользователя – в виде экономии трудовых, материальных и финансовых ресурсов, получаемой за счёт:

* снижения трудоёмкости расчётов и алгоритмизации программирования и отладки программ;
* сокращения расходов на оплату машинного времени и других ресурсов на отладку программных средств;
* снижения расходов на материалы;
* ускорение ввода в эксплуатацию новых систем;
* улучшения показателей основной деятельности в результате использования передовых программных средств.

Стоимостная оценка приложения у разработчиков предполагает определение затрат, что включает следующие статьи:

* заработная плата исполнителей – основная и дополнительная;
* отчисления в Фонд социальной защиты населения и обязательному страхованию от несчастных случаев на производстве и профессиональных заболеваний;
* расходы на материалы и комплектующие;
* расходы на спецоборудование;
* расходы на оплату машинного времени;
* общепроизводственные и общехозяйственные расходы;
* расходы, связанные с установкой и адаптацией (реализационные или коммерческие расходы).

На основании затрат рассчитывается себестоимость и отпускная цена разработанного веб-приложения.

7.3.1 Определение объёма программного средства

На основании информации о функциях, приведённой в таблице 7.3, рассчитывается общий объём программного средства *V*o по формуле 7.1.

, (7.1)

где *Vi* – объём i-ой функции ПС, условных машинных команд;

*n* – общее число функций.

Ниже приведена таблица с информацией о затратах рабочего времени на разработку программного средства.

Таблица 7.2 – Содержание и объём функций на разрабатываемое ПС

|  |  |
| --- | --- |
| Содержание работ | Затраты рабочего времени, дней |
| Инициализация клиентской и серверной частей | 6 |
| Инициализация базы данных | 3 |
| Создание ролей пользователей | 1 |
| Написание логики, связанной с созданием пользователей | 1 |
| Написание логики аутентификации | 2 |
| Написание логики, связанной с функционалом администратора | 6 |
| Написание логики, связанной с функционалом кассира | 10 |
| Написание логики, связанной с функционалом пользователя | 6 |
| Тестирование сервера | 2 |
| Написание логики, на стороне клиента | 1 |
| Тестирование мобильного приложения | 2 |
| Всего | 40 |

7.3.2 Основная заработная плата

Для определения величины основной заработной платы специалистов в сфере веб-программирования на языках программирования Java и JavaScript было проведено исследование. Для этого были использованы различные источники данных, включая открытые веб-порталы, форумы, официальную отчетность и общий средний уровень заработка в сфере информационных технологий.

Анализ этих источников позволил получить представление о том, какую заработную плату могут ожидать специалисты по программированию, на языках Java и JavaScript. В ходе исследования были учтены факторы, такие как уровень опыта, квалификация, региональные различия и специфика задач, с которыми работают программисты. Итогом изучения стала информация о том, что средняя месячная заработная плата для позиций *junior* составляет около 1700 рублей.

Основная заработная плата рассчитывается по формуле 7.3.

|  |  |
| --- | --- |
| Соз = Tраз ⋅ Краз⋅ Сзп, | (7.3) |

где Соз– основная заработная плата, руб.;

Траз – время разработки, месяцев;

Краз – количество разработчиков, человек;

Сзп – средняя месячная заработная плата.

Время разработки будет рассчитываться по формуле 7.4.

|  |  |
| --- | --- |
| Траз = Драз / 21, | (7.4) |

где Траз – время разработки, месяцев;

Драз – количество дней разработки, дней.

Проект разрабатывался одним человеком на протяжении трёх месяцев. Расчеты основной заработной платы и времени разработки представлены ниже:

Траз = 40 / 21 = 1,9 месяцев.

Соз = 1,9 ⋅ 1 ⋅ 1700 = 2890 руб.

7.3.3 Дополнительная заработная плата

Законодательство о труде предусматривает наличие выплат, определяющиеся по нормативу в процентах к основной заработной плате по формуле 7.5.

, (7.5)

где Соз – основная заработная плата, руб.;

Ндз – норматив дополнительной заработной платы, %.

Исходя из основной заработной платы, а также норматива дополнительной заработной платы, можно рассчитать сумму дополнительной заработной платы:

руб.

Дополнительная заработная плата включает выплаты, предусмотренные законодательством о труде, и определяется по нормативу в процентах по отношению к основной заработной плате.

7.3.4 Расчёт отчислений на социальные цели

Отчисления в Фонд социальной защиты населения (ФСЗН) определяются в соответствии с действующими законодательными актами по нормативу в процентном отношении к фонду основной и дополнительной зарплаты исполнителей и вычисляются по формуле 7.6.

|  |  |
| --- | --- |
|  | (7.6) |

где Соз – основная заработная плата, руб.;

Сдз – дополнительная заработная плата, руб.;

Нфсзн – норматив отчислений в Фонд социальной защиты населения, %.

Исходя из полученных данных высчитаем отчисления в Фонд социальной защиты наcеления (ФСЗН):

руб.

Полученные выше значения будут использованы при подсчёте заработной платы программиста с отчислениями.

7.3.5 Расходы на материалы

Общая сумма затрат на материалы для программного средства определяется путем анализа данных из таблицы 6.3, в которой содержится информация о фактических расходах на разработку данного программного продукта.

Таблица 7.3 – Материальные затраты на разработку ПС

|  |  |
| --- | --- |
| Вид материальных затрат | Фактические затраты, руб. |
| 1. Бумага, 2 пачки | 20,00 |
| 2. Картридж, 1 шт. | 20,00 |
| Всего | 40,00 |

Сумма расходов на материалы составляет 40,00.

7.3.6 Расходы на специальное оборудование и платные услуги

Сумма затрат на специальное оборудование и платные услуги для создания программного продукта составляет ноль рублей, так как в ходе реализации данного проекта не было понесено расходов на приобретение специального оборудования, а также не были оказаны платные услуги от других юридических лиц.

7.3.7 Расчет прочих прямых затрат

Расходы на конкретное программное средство Спз включают затраты на приобретение и подготовку специальной научно-технической информации и специальной литературы и рассчитываются по формуле 7.7.

|  |  |
| --- | --- |
|  | (7.7) |

где Нпз – норматив прочих затрат в целом по организации, %.

= ⋅ 12/100 = 346,8 руб.

Таким образом, прочие прямые затраты при разработке мобильного приложения для доставки еды из заведений общественного питания равны 346,8 рублей.

7.3.8 Расчет общепроизводственных и общехозяйственных расходов

Сумма общепроизводственных, общехозяйственных расходов Собп,обх  – произведение основной заработной платы исполнителей на конкретное программное средство Соз на норматив накладных расходов в целом по организации Нобп,обх. по формуле 7.8.

|  |  |  |
| --- | --- | --- |
|  |  | (7.8) |

Все данные необходимые для вычисления есть, поэтому можно определить сумму накладных расходов:

Собп,обх = ⋅ 30/100 =  руб.

Получим, что накладные расходы, необходимые для разработки мобильного приложения,равны 867 рублей.

7.3.9 Расчет суммы расходов на разработку программного средства

Сумма расходов на разработку программного средства Ср определяется как сумма основной и дополнительной заработных плат исполнителей на конкретное программное средство, отчислений на социальные нужды, расходов на материалы, общепроизводственных и общехозяйственных затратов и суммы прочих затрат.

Ср = Соз + Сдз + Сфсзн + См + ССОПУ + Спз + Собп,обх (7.9)

Ср = + ++ 40,00 + 0 + 346,8 +  = 5736,23 руб.

Таким образом, получим сумму всех расходов на разработку мобильного приложенияравной 5736,23 рублей.

7.3.10 Расходы на сопровождение и адаптацию

Сумма расходов на сопровождение и адаптацию программного средства Срса определяется как произведение суммы расходов на разработки на норматив расходов на сопровождение и адаптацию Нрса.

|  |  |  |
| --- | --- | --- |
|  |  | (7.10) |

где Срса – сумма расходов на сопровождение и адаптацию ПС, руб.;

Ср – общая сумма расходов на разработку ПС, руб.;

Нрса – норматив расходов на сопровождение и адаптацию, %.

Основываясь на исходные данные, расположенные в таблице 7.1 и формулу 7.10, норматив расходов на сопровождение и адаптацию Нрса равен 10%.

Сумма расходов на сопровождение и адаптацию ПС составляет:

руб.

Получим, что расходы, необходимые на сопровождение и адаптацию мобильного приложения,равны рублей.

7.3.11 Расчет общей суммы расходов

Общая сумма расходов на разработку с затратами на сопровождение и адаптацию Сп определяется по формуле 7.13.

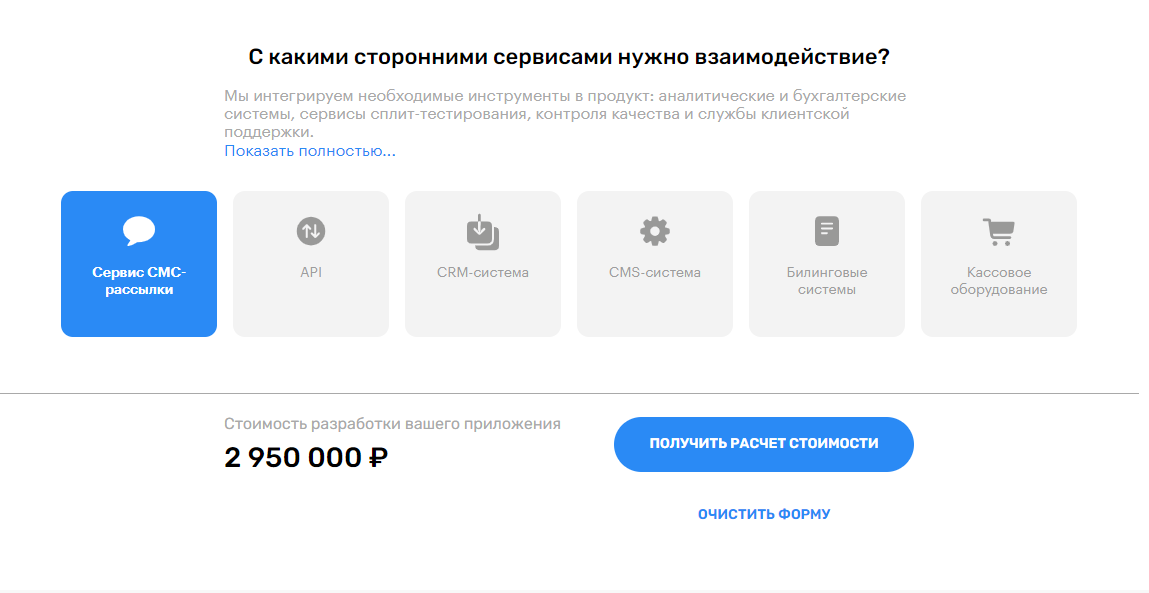
|  |  |  |
| --- | --- | --- |
|  |  | (7.13) |

Сп = + = 6309,85 руб.

Получим, что общая сумма расходов мобильного приложения, выполняющего роль заказа и доставки продуктов питания, равна 6309,85 рублей.

7.3.12 Определение цены, оценка эффективности

Для определения конечной цены разработки необходимо изучить тарифы на разработку приложений с аналогичным функционалом. Для этой цели мы воспользуемся калькуляторами, предоставляющих информацию о ценах на разработку программного обеспечения.

  
Рисунок 7.1 – Результат работы сайта thebestapp.ru

Для первого расчета воспользуемся сайтом thebestapp.ru. На сайте TheBestApp можно оценить стоимость и время разработки приложения. Указав все необходимые параметры, можно понять примерную стоимость нашего приложения. Получилось, что для схожего мобильного приложения стоимость составляет — 2 950 000 российских рублей (104 763,35 рублей).

Второй аналог для расчета — это сайт howmuchtomakeanapp.com. Данный калькулятор работает, как и остальные и также предлагает рассчитать стоимость разработки мобильного приложения. Отпускная цена тут рассчитана в долларах.

На рисунке 7.2 показан результат:

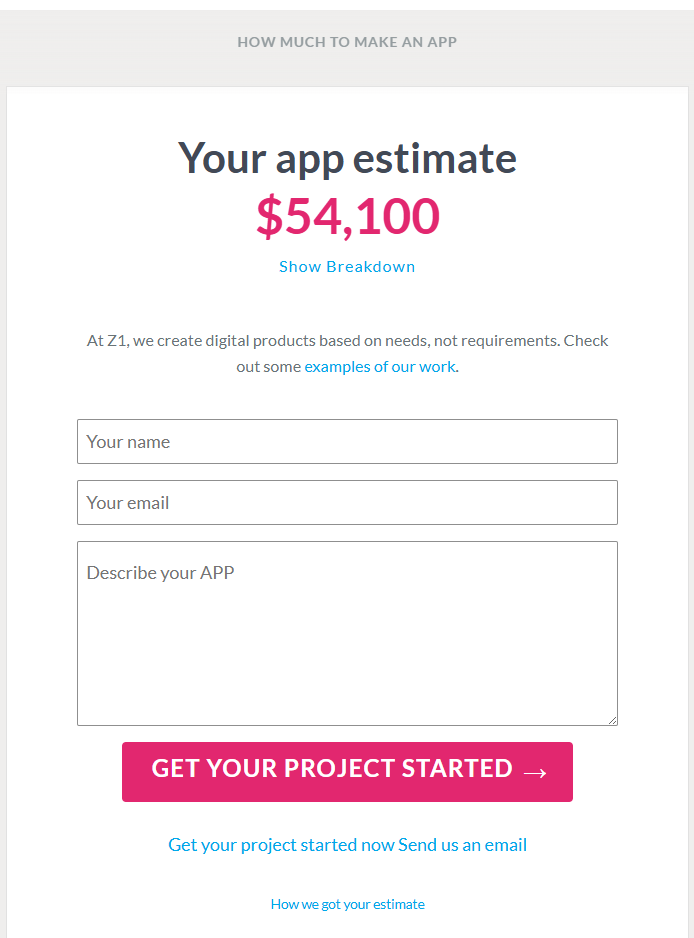
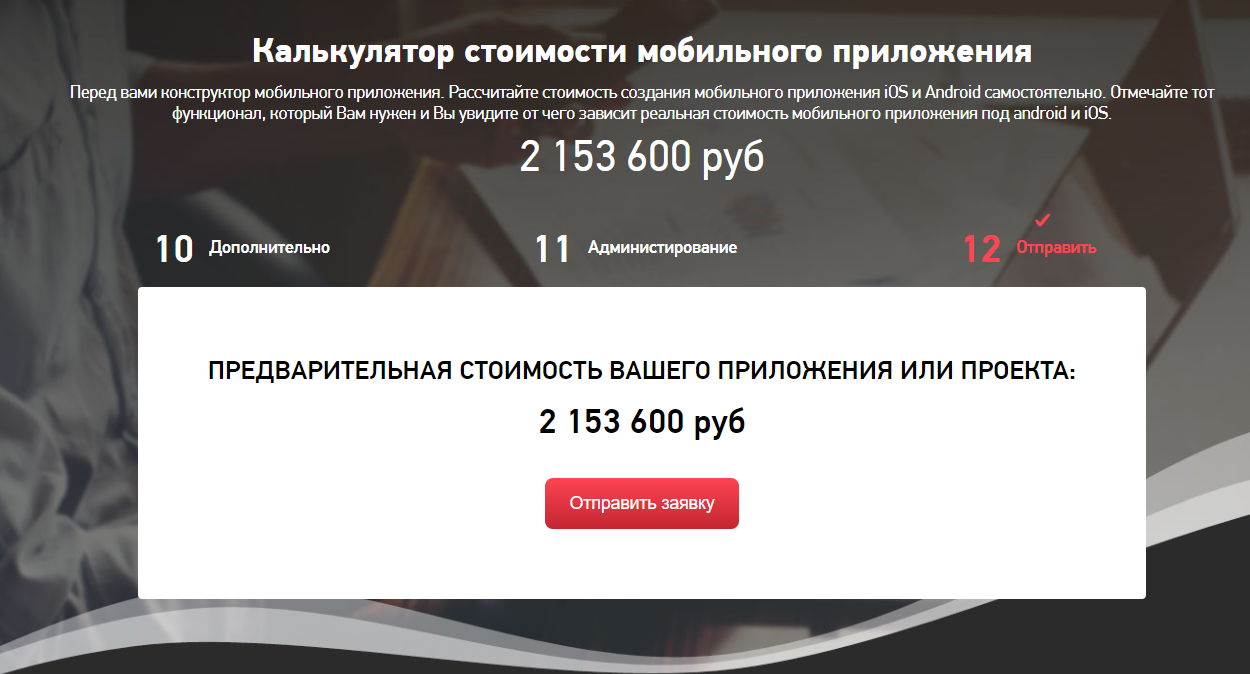


Рисунок 7.2 – Результат работы сайта howmuchtomakeanapp.com

По результатам разработка мобильного приложения выходит в 54 100$, в переводе на белорусские рубли – 177 052,53.

Третий в списке ресурс - сайт elapp.ru. Он функционирует аналогично предыдущим и предоставляет окончательные цены в российских рублях.

На рисунке 7.3 представлена стоимость мобильного приложения:

  
Рисунок 7.3 – Результат работы сайта elapp.ru

Исходя из данных, которые предоставил сайт, можно увидеть, что конечная цена для разработки мобильного приложения составит 2 153 600 российских рублей (76 480,8 бел. руб.).

Средняя цена разработки приложения с аналогичным функционалом рассчитывается как среднее арифметическое указанных выше цен:

ЦАср = (104 763,35 + 177 052,53 + 76 480,8) / 3 = 119432,22 руб.

Минимальная цена разработки приложения с аналогичным функционалом составляет 76 480,8 руб.

Полная себестоимость разработки ПС составляет 6309,85 руб., есть возможность установить цену ниже самой низкой оценочной цены.

Установим Цс НДС = 76 480,8 руб.

Цена ПС без НДС рассчитывается:

Цбез НДС = 76 480,8 (100 / 120) = 63733,3 руб.

Прибыль от реализации ПС:

Прп = 63733,3 – 5736,23 = 57997,07 руб.

Рентабельность ПС:

Рпс = (57997,07 /5736,23) 100 = 1011,06%.

Рентабельность большая, это означает что можно снизить цену в 5 раз по сравнению с рыночной и конкурировать с др.

Установим Цс НДС = 76 480,8/5 руб.

Цена ПС без НДС рассчитывается:

Цбез НДС1 = 76 480,8 (100 / 120) = 63733,3 руб.

Прибыль от реализации ПС:

Прп = 63733,3 – 5736,23 = 57997,07 руб.

Рентабельность ПС:

Рпс = (57997,07 /5736,23) 100 = 1011,06%.

# 7.4 Вывод по разделу

В таблице 7.4 представлены результаты расчетов основных экономических показателей необходимые для разработки и продажи мобильного приложения.

Таблица 7.4 – Результаты расчётов

|  |  |
| --- | --- |
| Наименование показателя | Значение |
| Время разработки, мес. | 1,9 |
| Количество программистов, чел. | 1 |
| Основная заработная плата, руб. | 2890 |
| Сумма отчислений на социальные цели, руб. |  |
| Сумма расходов на материалы, руб. | 40,00 |
| Прочие прямые затраты, руб. | 346,8 |
| Общепроизводственные и общехозяйственные расходы, руб. | 867 |
| Сумма расходов на разработку программного средства, руб. | 5736,23 |
| Расходы на сопровождение и адаптацию ПС, руб. |  |
| Общая сумма расходов, руб. | 6309,85 |
| Отпускная цена с НДС, руб. | 76 480,8 |
| Сумма налога на добавленную стоимость, руб. | 12 796 |
| Расчетная цена без НДС | 63733,3 |
| Расчетная Прибыль от реализации, руб. | 57997,07 |
| Расчет Рентабельность ПС, % | 1011,06 |

Исходя из совокупности всех вышеперечисленных факторов, можно сделать вывод, что разработка мобильного средства, для бронирования мест, несёт положительный экономический эффект.

Разработка программного средства, осуществляемая одним программистом в течении трёх месяцев, при заданных условиях обойдется компании в 5736,23 руб. Реализации данного программного средства принесет прибыль компании в сумме 57997,07 рублей. Цена за разработанный продукт соответствует низшей из рыночных цен за продукцию, обладающую аналогичным функционалом.

# Заключение

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*Заключение*

Лист

1

*БГТУ 01.00. ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_\_, 2024*

В ходе преддипломной практики было разработано мобильное приложение для бронирования мест на маршрутное такси.

Перед началом разработки был произведен анализ и сравнительный обзор аналогов приложений подобной тематики, и определение функциональных возможностей разрабатываемого приложения.

В результате написания дипломного проекта, было разработано приложение на основе программной платформы Node и интегрированной средой разработки Android Studio. Приложение написано с помощью языка программирования JavaScript, HTML, CSS и Java.

При разработке были выполнены следующие функциональные требования:

– авторизация в системе и разделение доступного функционала в соответствии с ролью пользователя;

* авторизация;
* добавление маршрута;
* изменения маршрута;
* добавление нового пользователя;
* бронирования места;
* изменения количества мест;
* отмена брони;
* сортировка маршрутов.

В ходе производственной практики были закреплены, расширены и углублены полученные теоретические знания в области веб-разработки на языке JavaScript, приобретены практические навыки самостоятельной работы, выработаны умения применять их при решении конкретных вопросов и задач. Требования технического задания были реализованы в полном объеме, поэтому цель преддипломной практики можно считать достигнутой.

# Список использованных источников

Утверд.

Н. контр.

Провер.

Разраб.

*Блинова Е.А.*

*Николайчук А.Н.*

*Макрица Ф.Н.*

*Тимонович Г.Л.*

Ф.И.О.

Подпись

Дата

*Список использованной   
литературы*

Лист

1

*БГТУ 01.00. ПЗ*

Лит.

Листов

*БГТУ \_\_\_\_\_\_\_\_\_\_\_\_\_, 2024*

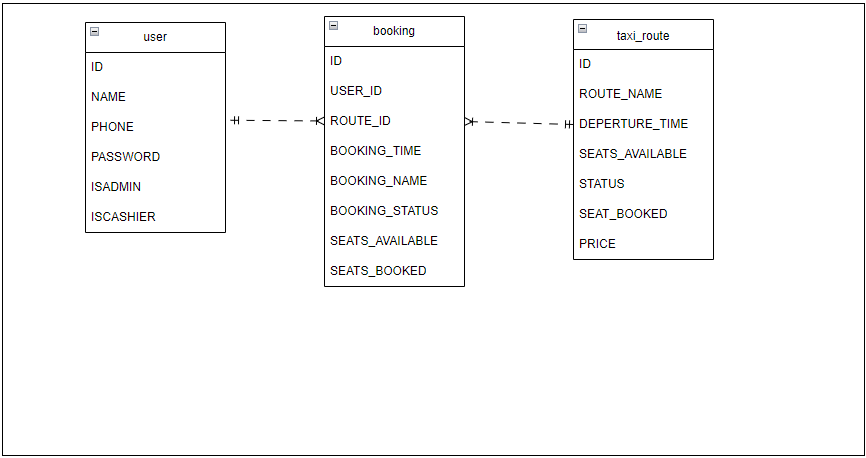
1. Документация JavaScript [Электронный ресурс]. – Режим доступа: <https://ravesli.com/intro-javascript/>. – Дата доступа: 12.03.2024.
2. Документация Node.js [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Node.js>. – Дата доступа: 12.03.2024.
3. Документация Model–View–Controller [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller>. – Дата доступа: 12.03.2022.
4. Документация Java [Электронный ресурс]. – Режим доступа: https://www.nic.ru/help/osnovy-programmirovaniya-na-yazyke-java\_11670.html. – Дата доступа: 20.03.2020.
5. Документация Клиент–Серверная архитектура [Электронный ресурс]. – Режим доступа: https://vladislaveremeev.gitbook.io/qa\_bible/seti-i-okolo-nikh/klient-servernaya-arkhitektura-client-server-architecture. – Дата доступа: 12.03.2023.
6. Документация npm [Электронный ресурс]. – Режим доступа: https://lms.kgeu.ru/mod/page/view.php?id=104573. – Дата доступа: 12.03.2023.
7. Документация Positive Research 2023 [Электронный ресурс]. – Режим доступа: https://data–express.ru/wyswyg/file/PricesforLicensesWin.pdf – Дата доступа: 12.03.2022.

Дата

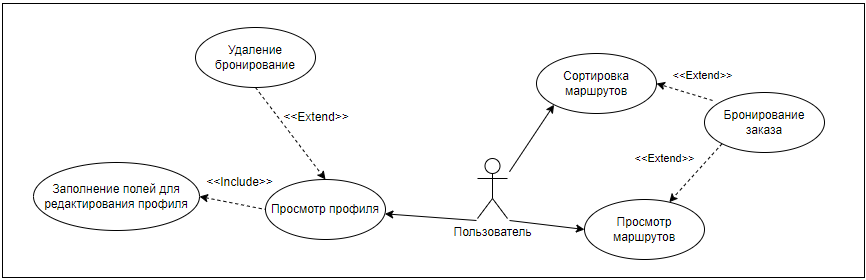
Дата

У

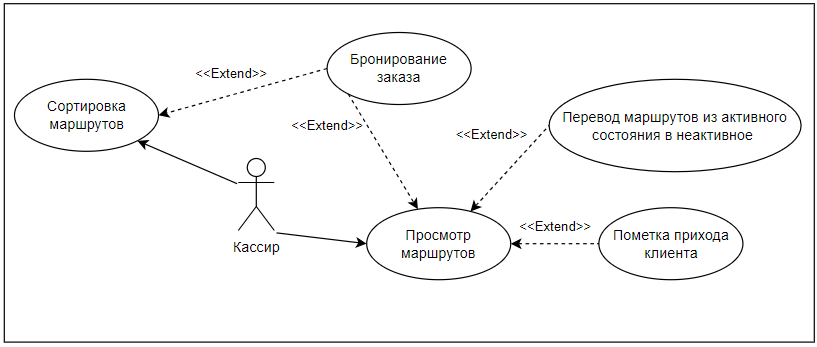
ПРИЛОЖЕНИЕ А. Схема базы данных



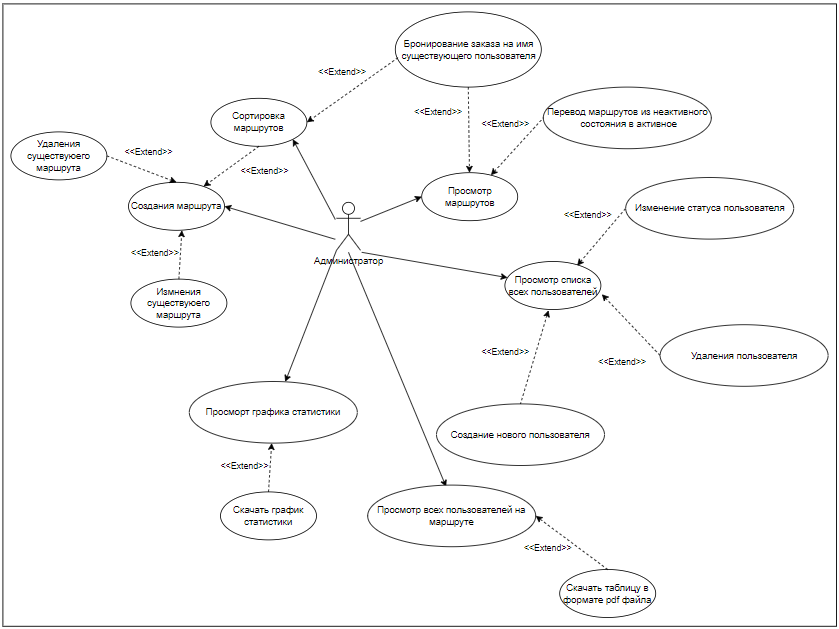
ПРИЛОЖЕНИЕ Б. Диаграмма вариантов использования пользователя



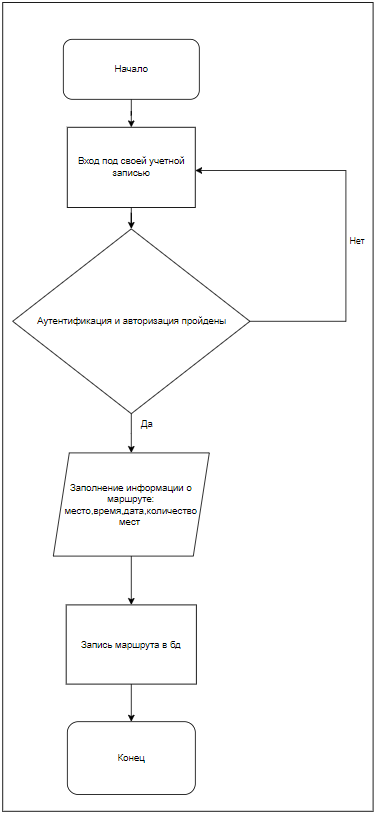
ПРИЛОЖЕНИЕ В. Диаграмма вариантов использования кассира



ПРИЛОЖЕНИЕ Г. Диаграмма вариантов использования администратора



ПРИЛОЖЕНИЕ Д. Блок-схема добавления маршрута



ПРИЛОЖЕНИЕ Е. Листинг метода добавления маршрута

|  |
| --- |
| router.post('/routes/create',checkAdmin, checkAuth, (req, res) => {      const routeName = req.body.routeName;      const departureTime = req.body.departureTime; // Добавлено поле departureTime      const seatsAvailable = req.body.seatsAvailable;      const Price = req.body.Price;      // Проверка наличия всех необходимых данных      if (!routeName || !departureTime || !seatsAvailable) {          return res.status(400).send("Не все поля были заполнены.");      }      // Преобразование данных, если необходимо      const formattedDepartureTime = new Date(departureTime); // Преобразование времени в объект Date      // Создание нового маршрута в базе данных      const query = "INSERT INTO taxi\_route (route\_name, departure\_time, seats\_available,price) VALUES (?, ?, ?,?)";      const values = [routeName, formattedDepartureTime, seatsAvailable,Price];      database.query(query, values, (error, result) => {          if (error) {              console.error("Ошибка при создании маршрута:", error);              return res.status(500).send("Произошла ошибка при создании маршрута.");          }          console.log("Новый маршрут успешно создан.");          res.redirect('/users/isAdmin'); // Перенаправление на главную страницу      });  });  router.post('/deleteRoute/:id', checkAuth,checkAdmin, (req, res) => {      const routeId = req.params.id; // Получаем ID маршрута из параметров запроса      // Здесь вы должны выполнить запрос к базе данных для удаления маршрута по его ID      const query = "DELETE FROM taxi\_route WHERE id = ?";      const values = [routeId];      database.query(query, values, (error, result) => {          if (error) {              console.error("Ошибка при удалении маршрута:", error); |

Продолжение листинга метода добавления нового маршрута

|  |
| --- |
| return res.status(500).send("Произошла ошибка при удалении маршрута.");          }          console.log("Маршрут успешно удален.");          res.redirect('/users/isAdmin'); // Перенаправляем на главную страницу после успешного удаления      });  }); |

ПРИЛОЖЕНИЕ Ж. Листинг создания пользователя

|  |
| --- |
| router.post("/login", (request, response) => {          const name = request.body.name;          const phone = request.body.phone;          const password = request.body.password;            const query = "SELECT id, name, phone, password, isAdmin, isCashier FROM user WHERE name = ? AND phone = ?";            const args = [name, phone];            database.query(query, args, (error, result) => {              if (error) {                  throw error;              }              if (result.length == 1) {                  const user = result[0];                  bcrypt.compare(password, user.password, (err, res) => {                      if (res) {                          // Сохраняем пользователя в сессии                          request.session.user = user;                          if (user.isAdmin === 1) {                              response.redirect(`/users/isAdmin`);                          } else if (user.isCashier === 1) {                              response.redirect(`/users/isCashier`);                          } else {                              response.redirect(`/?user=${user.name}`);                          }                        } else {                          response.status(401).send("Неверный пароль");                      }                  });              } else {                  response.status(214).send("Пользователь не найден");              }          });      }); |

ПРИЛОЖЕНИЕ З. Листинг аутентификации пользователя

|  |
| --- |
| // Middleware для проверки аутентификации пользователя  function checkAuth(req, res, next) {      if (req.session && req.session.user) {          const user = req.session.user; // Получаем объект пользователя из сессии          if (user.isAdmin === '1') {              // Если пользователь админ, отображаем страницу с оповещением об этом              res.send("Вы администратор");          }          else if(user.isCashier  === '1'){              res.send("Вы кассир");          }          else {              // Если пользователь не админ, пропускаем запрос дальше              next();          }        } else {          res.status(401).send("Пользователь не авторизован. <a href='/users/login'>Авторизоваться</a>");      }  } |