Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине   
"Технологии разработки веб приложений"

на тему

Разработка веб-приложения для построения кратчайших маршрутов по городу

Выполнил:

студент гр. ПРИм-124

Парахин К.В.

Приняла:

преподаватель кафедры ИСПИ

Шамышев А.А.

Владимир, 2025

**Аннотация**

В данном курсовом проекте производилась разработка веб-приложения – целью которого является задача построения кратчайших маршрутов по городу.

Проект состоит из 5 основных этапов, включающих в себя основные три этапа, такие как: «Анализ предметной области», «Проектирование системы», «Разработка клиент-серверного приложения», «Тестирование системы», «Развертывание приложения».

Разработка клиент-серверного приложения подразумевает реализацию в отдельных репозиториях серверного и клиентского приложений, а также проведение их совместного взаимодействия.

Развертывание веб приложения подразумевает под собой сборку и запуск готовых приложений в одном образе.

Реализованная система может применяться для таких целей, как, создание маршрутов по городу согласно разным параметрам, навигацию пользователя по маршруту и автоматическое перестроение маршрута, хранение списка маршрутов пользователя.

Реализованная система использует разделения по ролям пользователей – и включает в себя кроме обычной роли роль администратора.

Курсовой проект представлен на страницах, основных рисунков -, использованных источников – 5, приложений – , иллюстрационный материал на 2 листах формата А1.

**ABSTRACT**

In this course project, a web application was developed, the purpose of which is to build the shortest routes around the city.

The project consists of 5 main stages, which include the main three stages, such as: "Domain analysis", "System design", "Client-server application development", "System testing", "Application deployment".

The development of a client-server application involves the implementation of server and client applications in separate repositories, as well as their joint interaction.

Deploying a web application involves building and running ready-made applications in a single image.

The implemented system can be used for such purposes as creating routes around the city according to different parameters, navigating the user along the route and automatically rebuilding the route, storing a list of user routes.

The implemented system uses separation by user roles – and includes, in addition to the usual role, the administrator role.

The course project is presented on pages of basic drawings, 5 sources used, appendices, illustrative material on 2 sheets of A1 format.

**Содержание**

[ВВЕДЕНИЕ 6](#_Toc192596668)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc192596669)

[1.1. Описание предметной области 7](#_Toc192596670)

[1.2. Сравнительный обзор аналогов 8](#_Toc192596671)

[1.3. Описание набора функций системы 11](#_Toc192596672)

[1.5. Словарь предметной области: 12](#_Toc192596673)

[2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ 13](#_Toc192596674)

[2.1. Функциональная декомпозиция системы 13](#_Toc192596675)

[2.2. Описание состава данных 16](#_Toc192596676)

[2.3. Требования к разрабатываемой системе 19](#_Toc192596677)

[3. РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ 22](#_Toc192596678)

[3.1. Выбранный вариант разработки программной системы 22](#_Toc192596679)

[3.2. Описание взаимодействия приложений 24](#_Toc192596680)

[3.3. Реализация серверного приложения 26](#_Toc192596681)

[3.5. Спецификация API 27](#_Toc192596682)

[3.6. Описание разработки клиентского приложения 28](#_Toc192596683)

[4. ТЕСТИРОВАНИЕ и РАЗВЕРТЫВАНИЕ СИСТЕМЫ 29](#_Toc192596684)

[4.1. Тестирование запросов к REST API серверного приложения в Swagger API 29](#_Toc192596685)

[4.2. Развертывание сервера 29](#_Toc192596686)

[4.3. Развертывание клиентского приложения 30](#_Toc192596687)

[5. ВЕДЕНИЕ РЕПОЗИТОРИЕВ ПРОЕКТА 31](#_Toc192596688)

[Заключение 33](#_Toc192596689)

[Список использованных источников 34](#_Toc192596690)

[ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫ 36](#_Toc192596691)

[ПРИЛОЖЕНИЕ Б. ФИЗИЧЕСКАЯ СХЕМА ДАННЫХ СИСТЕМЫ 37](#_Toc192596692)

[ПРИЛОЖЕНИЕ В. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ 38](#_Toc192596694)

# ВВЕДЕНИЕ

Для начала выполнения разработки данной программно - информационной системы требуется выполнить следующие шаги по проектированию:

- описать предметную область;

- рассмотреть основные аналоги, провести их сравнение;

- выделить основные термины предметной области, сформировать словарь предметной области;

- выделить ее основные функциональные требования, с помощью основных UML-диаграмм описать составы данных и категории рассматриваемой системы;

- провести описание основных прецедентов пользователей, сущности системы;

- провести динамическое моделирование системы

- выбрать средства разработки и основные фреймворки (библиотеки), требующиеся для реализации функционала;

- выбрать протоколы взаимодействия, систему аутентификации и ведения безопасности;

- определить основные сценарии тестирования

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. Описание предметной области

Предметная область для данной темы включает в себя логистику города, расположение и описание его дорог – и отслеживание качества покрытия участков этих дорог и условий, которые происходят в данный момент времени на этих самых участках.

Основной проблемой является отсутствие публичного реестра качества и загруженности дорог города. На фоне ежегодного роста транспортного потока в крупных городах, увеличения формирования дефектов на существующих дорожных участках – все сложнее становится организовать грамотное перемещение транспортных средств – выбрав оптимальный (менее загруженный по трафику и с нормальным качеством дорожного покрытия – самый быстрый по преодолению).

Поэтому тема является очень актуальной, особенно в современных больших городах – с населением от 250 000 человек.

Данная работа является дочерней задачей для выполнения научно-исследовательской работы по созданию системы, которая будет осуществлять построение маршрутов по дорогам согласно качеству дорожного покрытия и актуальным дорожным условиям.

Цель проекта заключается в оптимизации маршрутизации движения и логистики автомобильного транспорта, снижение кол-ва заторов и аварийных ситуаций

Основными задачами он будет ставить представление, систематизация и обобщение данных о дорогах города и дорожной ситуации для формирования графических данных и построения маршрутов для пользователей.

Разработка данной программно-информационной системы позволит повысить автоматизировать сбор данных из различных источников, представить единый регистр для их хранения, анализа и обработки и будет помогать клиентам находить оптимальные маршруты в конкретные моменты времени.

## Сравнительный обзор аналогов

Для того, чтобы погрузиться в эту тему необходимо провести сравнительный анализ нескольких сервисов, предоставляющих открытый логистический функционал.

Мною были выделены сервисы: Google Maps, Waze, Яндекс.Карты и Навигатор

Проведем краткое описание каждого из сервисов.

Google Maps: Сервис для построения маршрутов с учетом пробок, дорожных условий, предоставляет информацию о покрытиях на основе данных спутников и пользовательских отчетов.

Год выпуска: 2005

Фирма-производитель: Google LLC

Сайт производителя: <https://maps.google.com>

Waze: Приложение для навигации с акцентом на актуальные дорожные условия, построение маршрутов с учетом пробок и аварий. Уникальная функция – обновления дорожных событий в реальном времени благодаря активной пользовательской базе.

Год выпуска: 2006

Фирма-производитель: Waze Mobile, принадлежит Google LLC

Сайт производителя: <https://www.waze.com>

Яндекс.Карты и Навигатор: Сервис, ориентированный на страны СНГ, поддерживает прокладывание маршрутов с учетом качества дорог, пробок, ДТП. Данные обновляются в реальном времени.

Год выпуска: 2004

Фирма-производитель: Яндекс

Сайт производителя: <https://yandex.ru/maps>

Посмотрим на требования к аппаратному и программному обеспечению данных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Приложение | Google Maps | Waze | Яндекс Карты и Навигатор |
| ОС | Android, IOS, WEB | Android, IOS | Android, IOS, WEB |
| Внутренняя память | 100 МБ | 100 МБ | 150 МБ |
| Особенности | Веб-версия не требует установки, часто обновляется | Работает только при подключении к интернету | Содержит удобный офлайн-режим |
| Взаимодействие | Импорт контактов, интеграция с календарем и др. | Google Assistant, календарь | Экспорт данных в Яндекс.Про, API |

Сравнение интерфейсов пользователя:

Google Maps: Интуитивно понятный интерфейс, поддерживающий жесты и голосовое управление, позволяет легко находить и выбирать точки на карте, оценивать качество маршрутов.

Waze: Яркий интерфейс с акцентом на визуализацию дорожных условий, высокая интерактивность. Имеет встроенные уведомления об опасностях.

Яндекс.Карты: Ориентирован на пользователей СНГ, интерфейс интуитивный, дружелюбный и включает голосовые подсказки.

Подведем таблицу со сравнительной характеристикой программных продуктов:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Характеристика |  | Google Maps | Waze | Яндекс Карты и Навигатор |
| Тип |  | Навигация, информация о пробках | Навигация в реальном времени | Навигация, пробки |
| Данные о качестве дорог |  | Ограничено, данные Google | Данные от пользователей | Подробные, обновляются часто |
| Взаимодействие |  | Взаимодействие с Google API, интеграция с календарем | Интерактивные отчеты о событиях | Поддержка локальных сервисов Яндекс |
| Поддержка офлайн |  | Ограничено | Нет | В полной мере реализовано |

По итогу проведенного сравнения, мной было отдано предпочтение логистическому сервису «Яндекс Карты и Навигатор» – так как оно наиболее ориентировано на пользователей СНГ и поэтому для нас дает лучшие возможности в рамках пользовательского интерфейса. Также загруженных карт и материалов по дорогам РФ тут больше, чем условно в достойном конкуренте от Google Maps.

## Описание набора функций системы

Основные функции разрабатываемого веб-приложения, определяющие границы предметной области:

- зарегистрироваться в системе (пользоваться функционалом только имея учетную запись)

- войти в систему (авторизоваться в приложении)

- выйти из системы

- передать параметры запроса и построить маршрут;

- иметь возможность перестроить маршрут;

- иметь возможность сохранить маршрут в личном кабинете;

- последовать по маршруту и получать обновления в реальном времени (при изменении координат пользователя);

- закончить следование по маршруту;

- администрировать процесс и валидировать «забракованные» пользователями маршруты;

- выполнять интеграцию и платежи с реквизитами заказчика;

* 1. Описание категорий пользователей системы и прецедентов

Система включает в себя явное разделение пользователей по ролям. По умолчанию, пользователь имеет базовую роль и соответствующие возможности.

При предоставлении дополнительных (секретных) учетных данных – пользователь может получить функционал администратора – и выполнять функции по ручной пост-валидации маршрутов, сгенерированных системой – и получивших плохую оценку от пользователя.

## Словарь предметной области:

1. Маршрут – запланированный путь от начальной до конечной точки, учитывающий различные параметры (расстояние, качество участка, загруженность);
2. Дорожный трафик – это количество автомобилей и пешеходов, находящихся на дорожном участке в определенный момент времени;
3. Пробка – затор на дороге, вызванный большим количеством автомобилей, проходящим в определенный момент по дорожному участку, ремонтом какого-то участка дороги (с его сужением) или произошедшей аварийной ситуацией;
4. Граф дорожной сети – математическая модель сети дорог, состоящая из вершин (узлов – как пример, перекрестков или точек назначения участников движения) и связей между ними – ребер;
5. Коммивояжер (задача коммивояжера) - это классическая задача оптимизации исходного маршрута, заключающаяся в нахождении кратчайшего пути, проходящего через заданный набор точек и возвращающего в начальную точку;
6. Взвешенный граф – это такой граф, в котором каждому ребру присвоена весовая характеристика, отражающая затраты на его использование, например, время или расстояние. (В мат. модели данной предметной области будет выбран какой-то параметр, зависящий от различных условий.);
7. Кратчайший путь во взвешенном графе – путь между 2 вершинами, проходящий по таким ребрам, суммарный вес которых минимален.
8. ПРОЕКТИРОВАНИЕ СИСТЕМЫ
   1. Функциональная декомпозиция системы

Основные прецеденты для всех прецедентов:

- Управлять своим аккаунтом

- Аутентификация через Google

- Авторизация

- Выход из системы

- Создать запрос

- Указать начальную и конечную точки

- Выбрать параметры запроса

- Перегенерировать маршрут

- Сохранить запрос

- Использовать маршрут из запроса

- Следовать по маршруту

- Прекратить (возобновить) следование

- Получать обновления сегментов маршрута

- Отменить маршрут

- Просмотреть сохраненные запросы

- Поощрить разработчика системы

Прецеденты администратора системы:

- Просмотр маршрута по пользовательскому запросу

- Пропустить маршрут (одобрить генерацию)

- Отметить ошибку в маршруте («забраковать» генерацию)

Диаграмма прецедентов в нотации UML изображена ниже на рисунке 1:

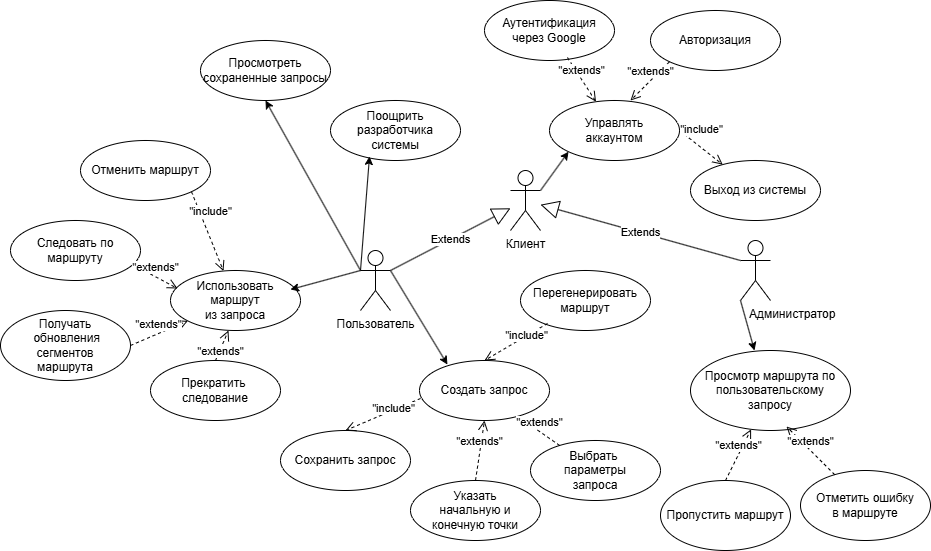


Рисунок 1. Диаграмма прецедентов

Пример описания одного из прецедентов системы:

Прецедент «Аутентифицироваться»

Предусловие: Пользователь решил воспользоваться функционалом системы при уже имеющейся учетной записи, то есть пользователь уже зарегистрирован в системе.

Действующее лицо: пользователь

Основной поток: пользователь открывает мобильное приложение, на главной странице видит раздел для авторизации, затем вводит свои данные и получает доступ к своей личной учетной записи и к личному кабинету.

Альтернативный поток: Пользователь ещё не зарегистрирован в системе. Тогда при попытке авторизоваться с некоторыми данными, он будет уведомлен об отсутствии требуемой учетной записи.

Постусловие: При отсутствии зарегистрированной учетной записи, пользователь будет перенаправлен на страницу регистрации.

* 1. Описание состава данных

Выделим следующие абстракции данных (сущности системы):

* User (пользователь, который обладает почтой и ролью)

UserRole – перечисляемый тип (User и Admin)

* Request (запрос на создание маршрута по точкам, может находиться в разных статусах, содержит список сегментов после сохранения в БД)

RequestStatus – перечисляемый тип (Created, Accepted, Followed, Unfollowed, Closed)

* Segment (сегмент дороги, минимальный линейный участок, из которых формируется общий маршрут)

SegmentType – перечисляемый тип (Initial и Active) – в зависимости от того, в каком статусе находится запрос – может быть отображаться как изначальный вариант маршрута, так и тот – который актуален в процессе следования пользователя по нему.

Как уже было сказано ранее, сущность Request может иметь разные статусы в различные моменты времени, а именно (Created, Accepted, Followed, Unfollowed, Closed) – и в зависимости от того, в каком статусе сейчас находится Request – с ним можно производить определенные действия.

Created – статус, который получает запрос после создания. Пока запрос находится в этом статусе его можно пересоздавать (при этом будет происходить перестройка маршрута – начальная и конечная точки будут использованы те, которые были указаны при создании)

Accepted – статус – который запрос получается после одобрения пользователем. Только после получения данного статуса запрос может быть использован пользователем в рамках следования по карте (следование по карте происходит после получения запросом статуса Followed)  
Также запрос будет сохранен и отображен в личном кабинете – только если получил хотя бы этот статус

Когда пользователь по какой-то причине отменяет следование по маршруту – то он получает статус Unfollowed

При закрытии запроса – он получает терминальный статус Closed – в него пользователь может перевести запрос из любого статуса, кроме статуса Followed – из статуса Followed запрос можно перевести в статус Unfollowed.

Данные сущности могут быть представлены в виде транзакционной схемы базы данных (ER-диаграммы):

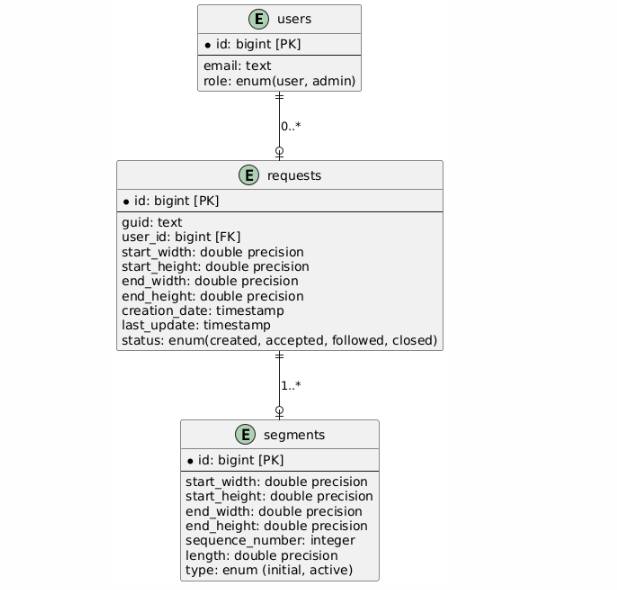


Рисунок 2. ER-диаграмма

На диаграмме состояний отобразим различные состояния запроса, которые были описаны выше в рамках описания статусов:

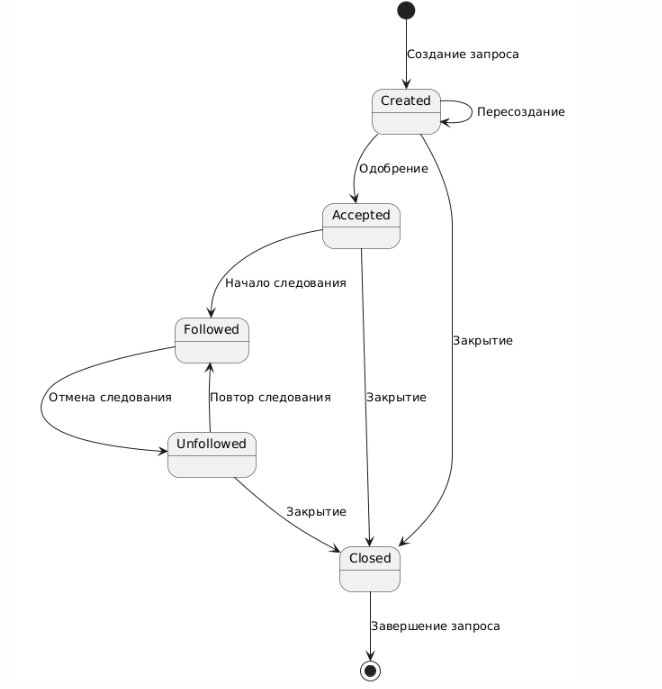


Рисунок 3. Диаграмма состояний UML

## Требования к разрабатываемой системе

Выделим основные категории требований, которые необходимо предъявить к анализируемой выше разрабатываемой системе:

1. Функциональные требования

Список функциональных требований, минимально необходимых для прототипа мобильного приложения:

1.1) Функция 1

1.1.1) Название - возможность использовать аутентификацию пользователей

1.1.2) Описание - пользователь получает доступ к функционалу на сервере с помощью access JWT токена, получаемого после аутентификации в Google сервисе (после ввода данных пользователя). Если пользователь с данным email в БД не существует – он будет автоматически создан.

1.1.3) Назначение – предозначить доступ только запросам от аутентифицированных пользователей.

1.1.4) Зависимости – фреймворк ASP.NET Core 7, Google JWT Bearer Authentification.

1.2) Функция 2.

1.2.1) Название – подтверждение создание новой учетной записи

1.2.2) Описание – возможность подтверждения регистрации новой учетной записи пользователя путем подтверждения учетной записи через ссылку, высылаемую на указанный адрес электронной почты

1.2.3) Назначение – верифицировать используемый в настройках учетной записи адрес электронной почты

3.3.3 Функция 3.

3.3.3.1 Название – получение роли администратора

3.3.3.2 Описание – возможность получения роли администратора после подтверждения полномочий в результате получения дополнительных разрешений на странице входа в систему

3.3.3.3 Назначение – создавать новые учетные записи, имеющие администраторские полномочия

3.3.4 Функция 4.

3.3.4.1 Название – получать доступ к текущему положению пользователя

3.3.4.2 Описание – возможность получения координатной метки устройства пользователя, при наличии включенного на мобильном устройстве GPS метки

3.3.4.3 Назначение – автоматическое получение и простановка текущего положения пользователя для начальной точки построения маршрута

3.3.5 Функция 5.

3.3.5.1 Название – выбирать вручную на карте начальную и конечные точки маршрута

3.3.5.2 Описание – возможность ручного выбора начальной и конечной точки маршрута путем клика на целевые точки на карте

3.3.5.3 Назначение – точечная настройка конкретных точек маршрута – настройка начальной точки (вместо выбора ее как текущее положение пользователя, по умолчанию)

3.3.6 Функция 6.

3.3.6.1 Название – настраивать параметры проезда по маршруту

3.3.6.2 Описание – возможность настроить в ручном порядке такие параметры проезда по маршруту, как: проезд по асфальтированным дорогам, скорость не ниже определенной, отсутствие светофоров, лежачих полицейских и других помех и т.д.

3.3.6.3 Назначение – точечная настройка конкретных параметров дорожных участков (может быть полезна начинающим водителям или при неисправностях в управлении транспортным средством для его доставки до места ремонта)

1. Нефункциональные требования

Список нефункциональных требований, определяющих свойства, которые система должна демонстрировать, или ограничения, которые она должна соблюдать – но не относящиеся к поведению системы:

2.1) Производительность - трафик запросов не менее 10К RPS (10 000 уникальных запросов пользователей в секунду);

2.2) Надёжность (система работает 24 на 7 и предоставляет возможность для работы с ней);

2.3) Доступность системы – доступ к приложению должен осуществляться в любой точке, в которой есть покрытие сетью Интернет.

2.4) Платформо-ориентированность

Возможность пользоваться клиентскими приложениями как в веб-браузере, так и на мобильных устройствах на операционной системе Android;

2.5) Масштабируемость системы – наличие архитектурных возможностей к возможному горизонтальному и вертикальному масштабированию.

# 3. РЕАЛИЗАЦИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

## 3.1. Выбранный вариант разработки программной системы

При разработке данного мобильного клиент-серверного приложения планируется использовать следующий список основных библиотек и фреймворков:

1. Для разработки серверной части приложения использовать библиотеку ASP.NET Core на платформе .NET 9.

Выбор данной платформы связан с ее удобством, наличием всего необходимого функционала для создания Restful WEB-API приложения, а также моими личными предпочтениями и опытом разработки.

ASP.NET Core – новейший фреймворк для разработки веб-приложений на платформе .NET, который предоставляет быстрые и продвинутые решения для создания постоянно обновляемых и улучшаемых систем, устойчивых к рефакторингу.

1. Для разработки клиентской части приложения использовать фреймворк Flutter.

Выбор данного фреймворка связан с его популярностью на рынке, удобством и простотой использования, а также наличием огромного количества плагинов и виджетов – благодаря которым возможно воплотить все основные визуальные решения с минимальными временными затратами.

1. Для работы с БД использовать ORM – фреймворк Entity Framework и СУБД Postgre SQL.

Выбор в сторону Entity Framework был сделан в результате выбора реализации серверной части приложения на платформе .NET и желания использовать доменный подход к работе с моделями. Entity Framework является одним из продуктов компании Microsoft, постоянно обновляется и поддерживается.

Выбор в сторону Postgre SQL был сделан из-за популярности данной СУБД на рынке, а также удобством использования: как с точки зрения взаимодействия со стороны кодовой базы и ее поддержкой провайдером Entity Framework, так и со стороны простоты пользовательского интерфейса приложения pgAdmin – для просмотра состояния таблиц баз данных в ручном режиме.

Также планируется использовать следующие прикладные инструменты разработки и тестирования:

1. Для разработки серверного приложения планируется использовать IDE Microsoft Visual Studio – как наиболее удобное, популярное, а также свободно распространяемое программной средство разработки приложений на .NET.
2. Для разработки клиентского веб - приложения планируется использовать IDE Visual Studio Code – как приложение, предложенное в качестве основного для проведения разработки клиентских (frontend) приложений;
3. Для тестирования REST API HTTP-запросов планируется использовать Swagger API – встроенный в ASP.NET Core библиотека, предоставляющая интерфейс для тестирования RestFul API.

3.2. Описание взаимодействия приложений

Ниже на рисунке 5 представлена диаграмма развертывания описанной выше программной системы:

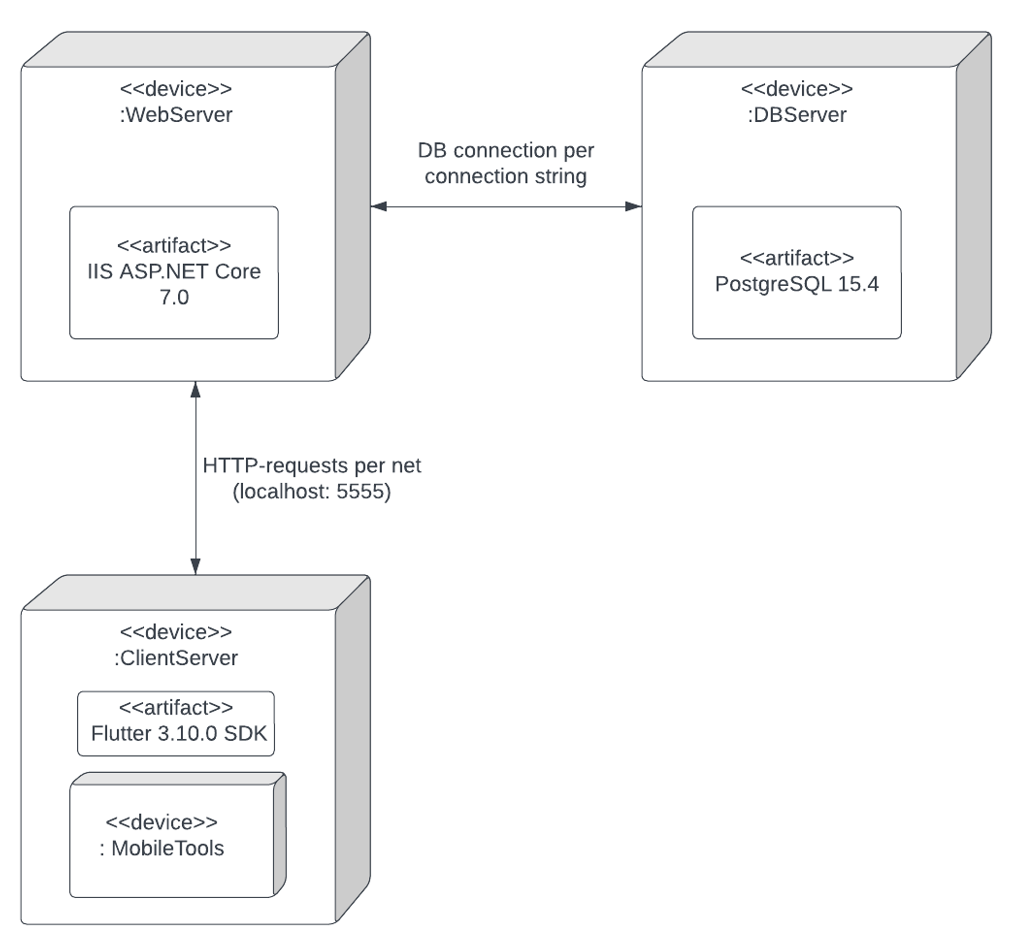


Рисунок 5. Диаграмма развертывания ПС

Построение алгоритма работы приложения

Ниже построим схему одного из основных алгоритмов приложения – схема процесса авторизации пользователя в системе.

Построение схемы работы алгоритма – алгоритм аутентификации пользователя в веб-приложении:

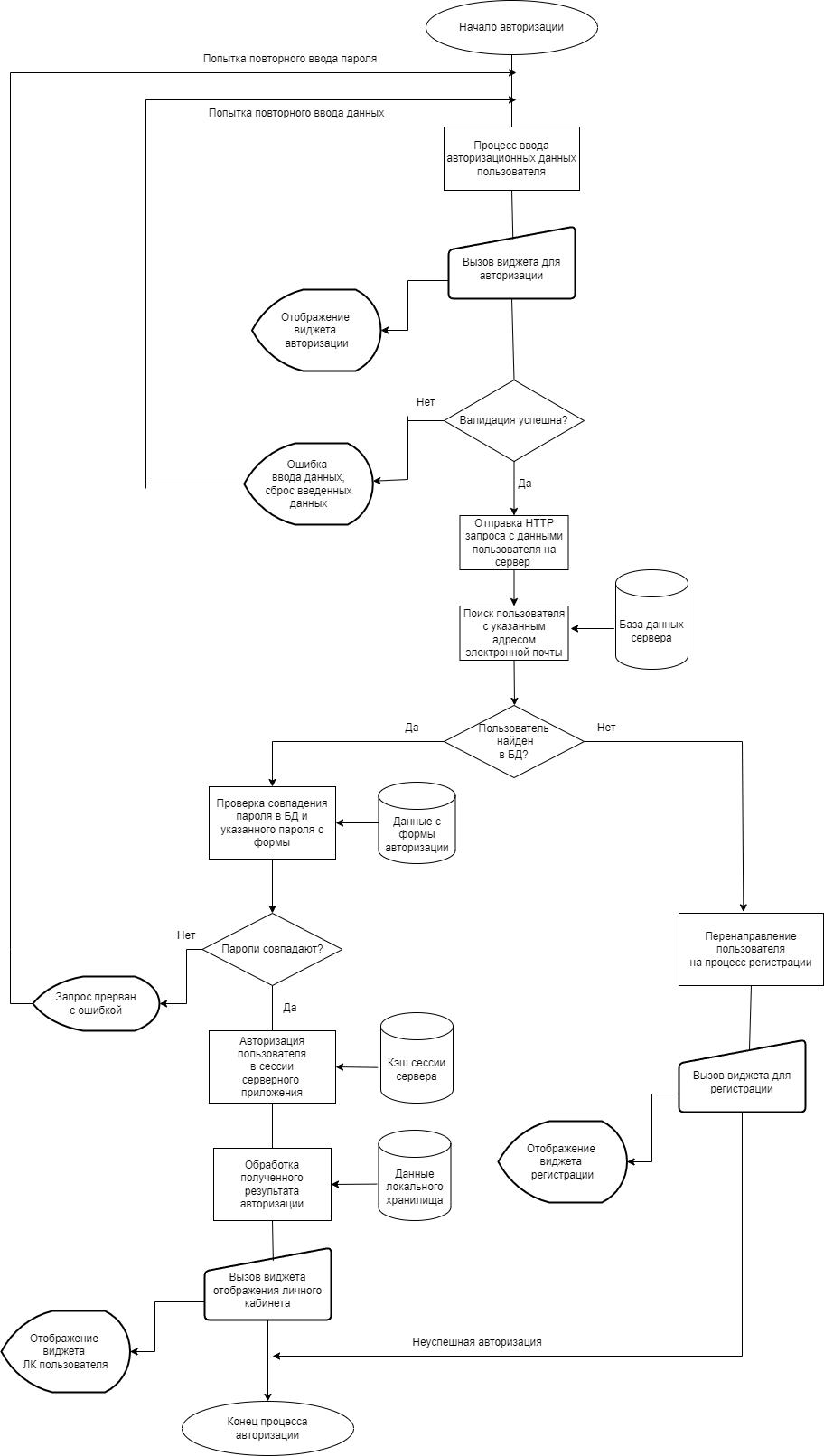


Рисунок 6.1. Схема алгоритма аутентификации пользователя

3.3. Реализация серверного приложения

Используемые технологии

В качестве платформы для разработки серверной части приложения был выбран ASP.NET Core благодоря его высокопроизводительности и кроссплатформенности.

В качестве СУБД была выбрана PostgreSQL — свободная объектно-реляционная система управления базами данных. Существует в реализациях для множества UNIX-подобных платформ, включая AIX, различные BSD-системы, HP-UX, IRIX, Linux, macOS, Solaris/OpenSolaris, Tru64, QNX, а также для Microsoft Windows.

Аутентификация была использована на основе Identity от ASP.NET Core – с моими собственными доработками (генерацией токена на основе токеном JWT и проверкой токенов на уровне middleware серверного приложения).

Используемые библиотеки

Newtonsoft.Json – библиотека для сериализации и десериализации объект в JSON.

Serilog – сторонняя библиотека для логгирования приложения (как в консоли, так и в Google Cloud)

Entity Framework Core - современный модуль сопоставления "объект — база данных" для .NET.

xUnit - инструмент тестирования (тестирование покрывает только основные интеграционные сценарии – типа сценария регистрации. И тестируется в принципе работоспособность приложения – тест на hc (health checks)).

3.5. Спецификация API

Описание HTTP протокола

Все запросы начинаются с пути к выделенному серверному порту, <http://host:5001>

Все запросы, содержащие тело – должны иметь заголовок вида:

*Content-type: Application/json*

При успешном выполнении запроса – пользователю всегда будет возвращаться код 200 – HTTP OK.

При ошибке авторизации возвращается ошибка 401 и текстом «*Unauthorized*».

При ошибке доступа возвращается ошибка со статусом 403 и текстом «*Forbidden*» или ошибка со статусом 400 и текстом «*Bad Request*»

При ошибке наполнения тела запроса возвращается ответ со статусом 400 и описанием ошибки в объекте errros.

При ошибке исполнения запроса, связанной с данными из тела, возвращается ответ со статусом 500 –«*Internal Server Error*».

Описание WebSocket протокола

Все запросы начинаются с пути к выделенному серверному порту, [ws://host:5001](http://host:5001)

## 3.6. Описание разработки клиентского приложения

Клиентская часть приложения была разработана с использованием фреймворка Flutter.

# ТЕСТИРОВАНИЕ и РАЗВЕРТЫВАНИЕ СИСТЕМЫ

## Тестирование запросов к REST API серверного приложения в Swagger API

## Развертывание сервера

Для использования серверного приложения – его удобно развернуть в отдельном контейнере, например, Docker – контейнере.

Чтобы произвести развертывание приложения, необходимо ввести в терминале проекта команду: docker compose –up build.

При этом во время развертывания с помощью Dockerfile будет создан образ серверного приложения (под названием todocalendar-app), а также образ базы данных postgres\_container (при этом при создании контейнера – к базе данных будут применены основные миграции по созданию таблиц и наполнению их некоторыми исходными данными).

По итогу, получается контейнер приложения, к которому можно получить доступ в локальной сети по адресу url: 4040:4040 (рис. 24)

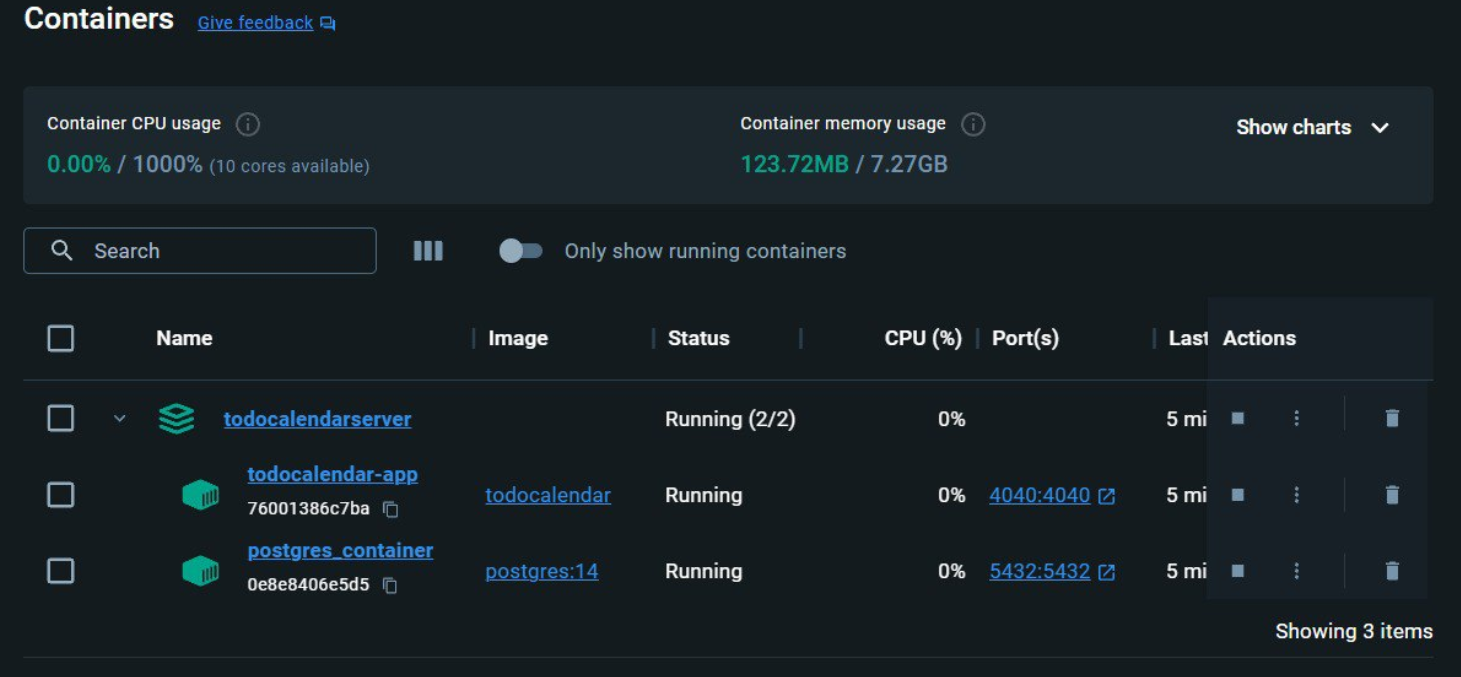


Рисунок 24. Развертывание сервера на Docker

## Развертывание клиентского приложения

## ВЕДЕНИЕ РЕПОЗИТОРИЕВ ПРОЕКТА

Вся разработка велась с использованием системы контроли версий git и сервиса для хранения удаленных репозиториев Github, на котором размещен код моей программной системы по ссылке: <https://github.com/Tigeroff2002/LogisticAPI> (рис.–– представлены скришоты репозиториев с серверной и клиентской частю разрабатываемого курсового проекта, расположенные на сайте GitHub).

Для разработки использовался базовый функционал, предлагаемый системой git. Но, несмотря на то, что разработка велась не в команде (то есть она проводилась только мной) – все равно реализация функционала велась в отдельных ветках, на каждую из задач или изменение создавались коммиты.

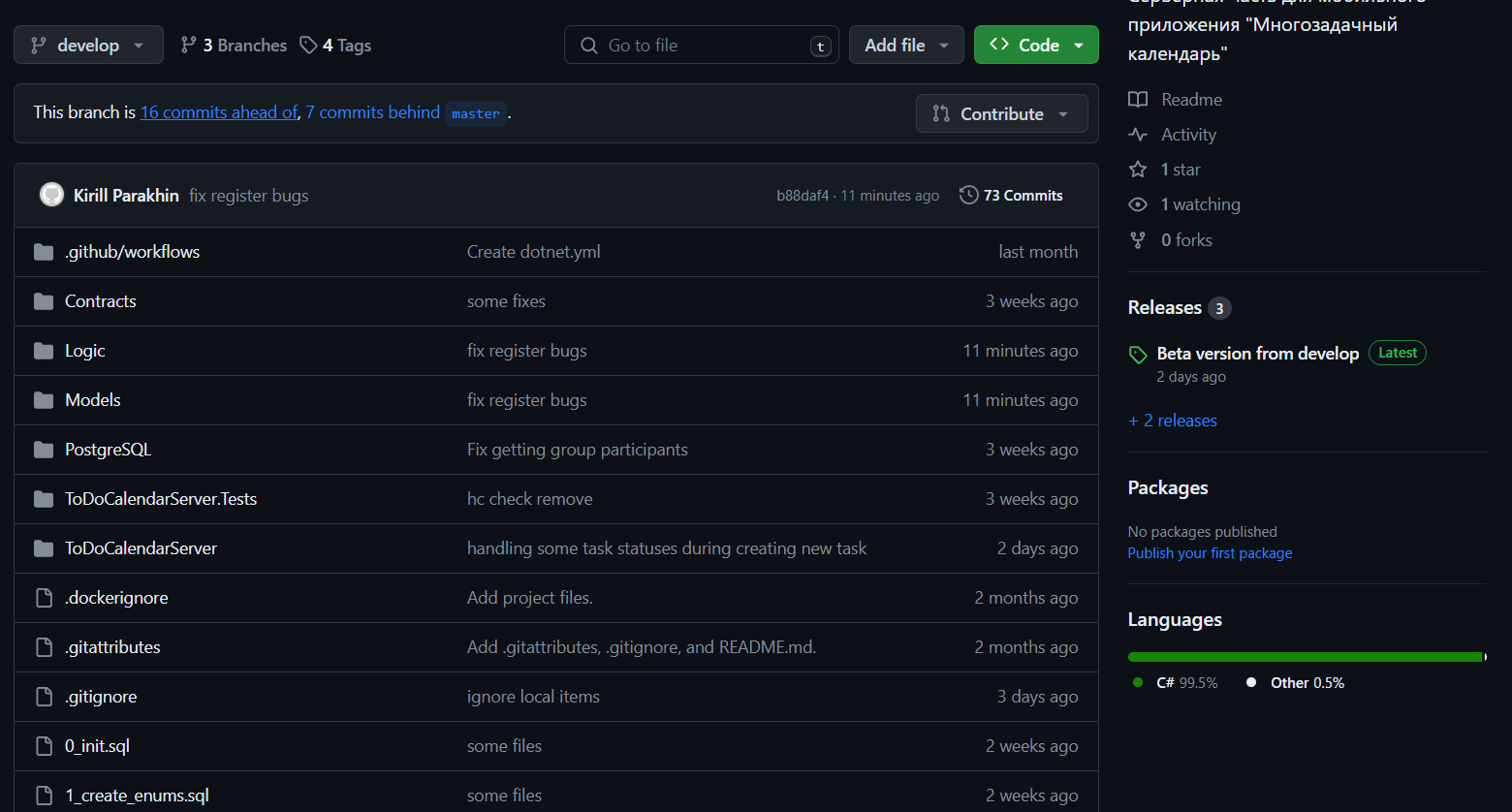


Рисунок. Репозиторий серверного приложения

Репозиторий клиентского приложения доступен на Github по ссылке: <https://github.com/Tigeroff2002/LogisticClient>

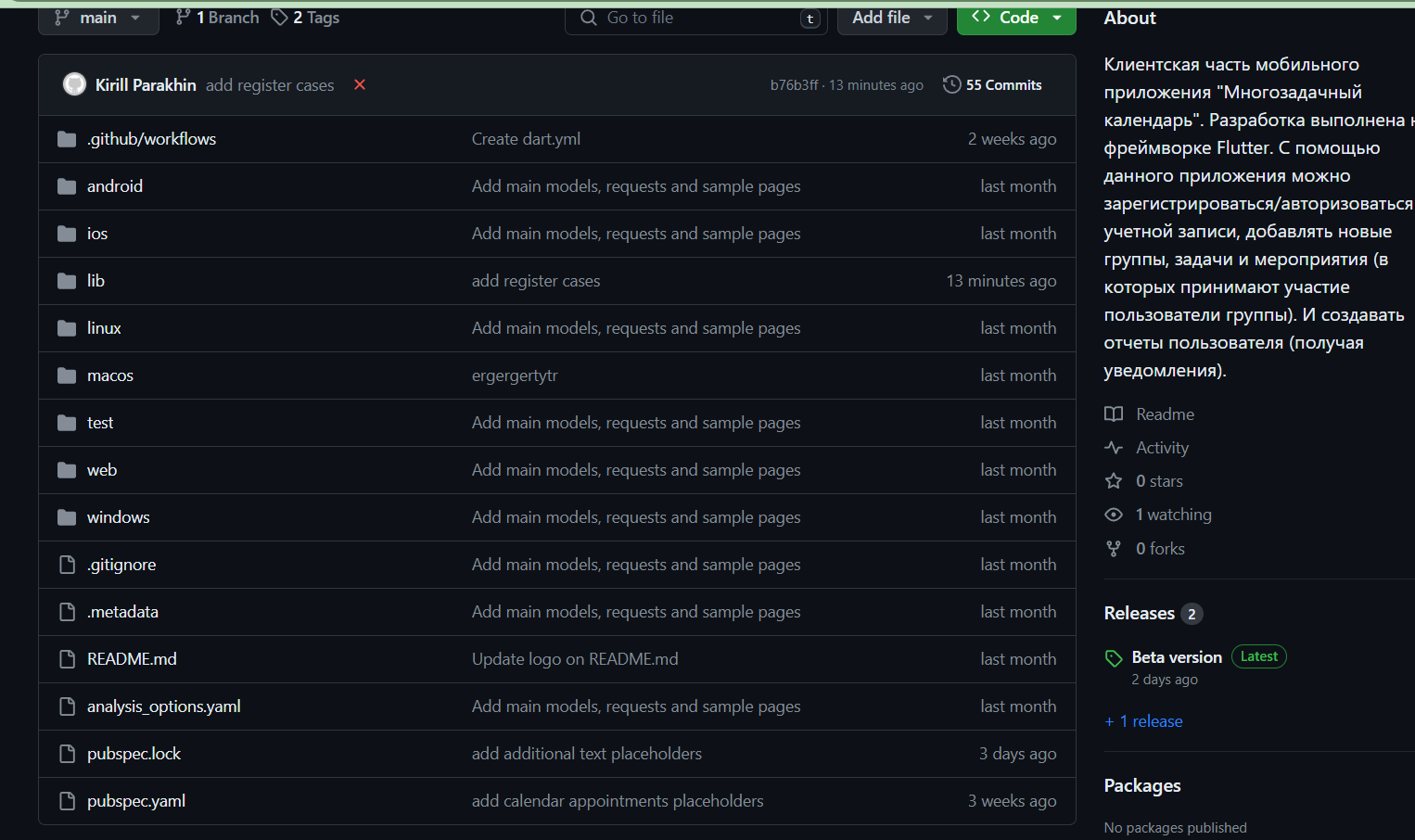


Рисунок 29.2. Репозиторий клиентского приложения

По мере внесения изменений – проводилось автоматическая удаленная сборка серверного приложения, а также его развертывание в Docker – контейнере

По окончанию разработки были созданы релизы – то есть готовые собранные приложения, готовые к скачиванию и прикладному использованию.

# Заключение

# Список использованных источников

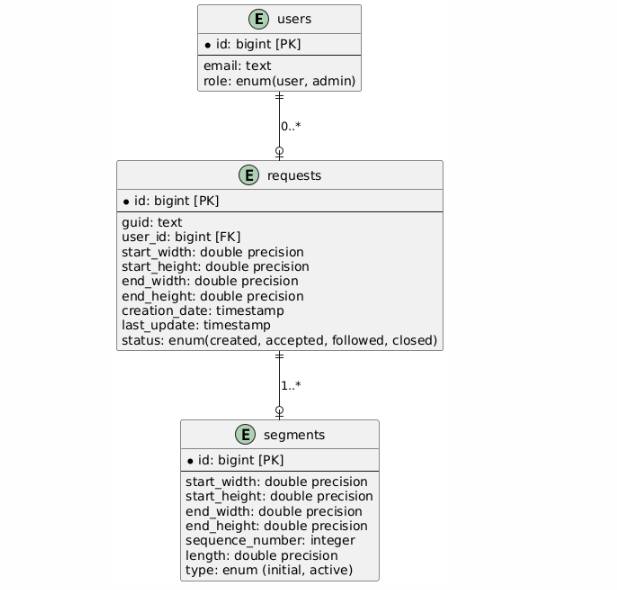
1. Черняк, И. С. Логистика для большого города / И.С. Черняк // В.Ю. Конюхов: учебное пособие, электронный научный журнал «Известия»: в 3-х т. / И. С. Черняк, В. Ю. Конюхов - М., 2018 №6. - Т. 1. - С. 101-116;
2. Лопухов, Н. В. Использование информационных технологий в управлении города / Н. В. Лопухов // Лопухов Н. В. учебно-практическое пособие / Н. В. Лопухов, О. А. Астафурова, Н. А. Сальников. - М., 2017. - С. 120-125;
3. Сагинова, О. В. Модели городской мобильности и логистика крупного города / О. В. Сагинова // Экономика, предпринимательство и право: учебносправочное пособие - М., 2020 № 2. - Т. 10. - С. 220- 225. *ISNN. 2222 – 534X;*
4. Чернышев, М. А. Приоритеты и пути повышения эффективности развития транспортной логистики крупного города / М. А. Чернышев // Т. Ю. Анопченко: Региональные проблемы преобразования экономики: учебное пособие / М. А. Чернышев, Т. Ю. Анопченко - М., 2011 - Т. 1. - С. 10-13;
5. Сапрыкин, С. С. Влияние состояния дорожного полотна и дорожных условий на безопасность дорожного движения / С. С. Сапрыкин // В. В. Пак: Вестник магистратуры: учебное пособие / С. С. Сапрыкин, В. В. Пак - М., 2022 №3 - 1. - Т. 1. - С. 125-126. *ISNN 2223-4047;*
6. Общие сведения об ASP.NET Core [Электронный ресурс]. – 2024. Режим доступа: <https://learn.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-8.0> (дата обращения: 12.01.2024 г.).
7. Авторизация с помощью определенной схемы в ASP.NET Core [Электронный ресурс]. – 2024. Режим доступа: [https://learn.microsoft.com/ru ru/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-8.0](https://learn.microsoft.com/ru-ru/aspnet/core/security/authorization/limitingidentitybyscheme?view=aspnetcore-8.0) (дата обращения: 20.02.2024 г.).
8. Создание Flutter Rest API [Электронный ресурс]. – 2023.

Режим доступа: <https://www.c-sharpcorner.com/learn/learn-flutter/flutter-rest-api> (дата обращения: 17.03.2024 г.).

1. Руководство по Docker Compose для начинающих [Электронный ресурс]. – 2022. Режим доступа: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения: 14.04.2024 г.).
2. Бёрнс Б. Распределенные системы. Паттерны проектирования. — СПб.: Питер, 2019 — 224 с.: ил. — (Серия «Бестселлеры O’Reilly»). ISBN 978-5-4461-0950-0.

# ПРИЛОЖЕНИЕ А. ДИАГРАММА ПРЕЦЕДЕНТОВ СИСТЕМЫC:\Users\parah\Desktop\Labs\Maga_1_course\Web techs\precedents.png

ПРИЛОЖЕНИЕ Б. ФИЗИЧЕСКАЯ СХЕМА ДАННЫХ СИСТЕМЫ



ПРИЛОЖЕНИЕ В. ДИАГРАММА РАЗВЕРТЫВАНИЯ ПРИЛОЖЕНИЯ

