МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка веб-приложения «Интерактивная карта студгородка ЮУрГУ»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-542.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ В.Н. Алеева

Автор работы,
студент группы КЭ-402
_____ Н.А. Воронков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Воронкову Николаю Андреевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка веб-приложения «Интерактивная карта студгородка ЮУрГУ».
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Флэнаган Д. JavaScript. Полное руководство. Седьмое издание. // Издательство Диалектика-Вильямс 2021. – 720 с.

3.2. Документация API карт 2ГИС. [Электронный ресурс] URL: https://api.2gis.ru/doc/maps/ru/quickstart/ (дата обращения: 11.02.2024 г.).

3.3. Порселло Е, Бэнкс А. React: современные шаблоны для разработки приложений 2-е издание. // Издательство Питер, 2023. – 320 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Разработать дизайн сайта.
4.2. Спроектировать архитектуру веб-приложения.
4.3. Разработать программную реализацию веб-приложения.

5. **Дата выдачи задания:** 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. В.Н. Алеева

Задание принял к исполнению Н.А. Воронков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Студенческий городок ЮУрГУ – это отдельный микрорайон возле университета со своими больницами, достопримечательностями, закусочными, учебными корпусами, общежитиями и т.д. Для ориентации в подобном пространстве необходимо знать, где и что находится. Не известно, как быть человеку, который только начал свое обучение в университете, никогда не был в студгородке или вообще впервые приехал в незнакомый город. Людям, которые будут учиться здесь до шести лет в незнакомом городе, обязательно нужно будет изучить местность, а проще всего это сделать с помощью подручных средств, например, через новых знакомых студентов, возможно местных, которые живут здесь на протяжении многих лет или, тратя огромное количество времени на изучение студгородка самостоятельно. Однако, можно воспользоваться картой города в интернете или еще лучше картой самого студенческого городка.

Считается, что первые карты появились еще в каменном веке. Когда люди изображали часть территории на камнях в пещерах. Они это делали для того, чтобы сохранить знания о какой-либо местности. Шли годы. И вот люди различных цивилизаций открывали для себя все новые знания об окружающем мире. Люди все больше совершали путешествия и походы. И тем самым все больше информации требовалось сохранить.

Занимательно, но первыми, кто изготовил карту мира стали греки во времена античности. Но самый большой вклад в развитие картографии и географии внес великий ученый (астроном, географ, математик) Клавдий Птолемей (около 90 года – около 160 года). Именно его карты отображают мир от Канарских островов на западе до Китая на востоке. Птолемей в своих работах собрал всю информацию о мире во времена античности.

На данный момент не существует веб-приложения, в котором человек может посмотреть на карту студенческого городка ЮУрГУ. Например, карта, которая находится в приложении «ЮУрГУ-Онлайн», распространя-

ется на карту всего города. Карта приложения, созданного в данной дипломной работе, будет ориентироваться на студгородок университета ЮУрГУ. Она будет иметь отметки на карте, которые дадут возможность увидеть, где находится тот или иной учебный корпус или общежитие, какие есть достопримечательности в студенческом городке, как найти заведения для общественного питания и т.д.

Постановка задачи

Целью выпускной квалификационной работы является разработка веб-приложения с интерактивной картой студгородка ЮУрГУ. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и обзор аналогов;
2) спроектировать веб-приложение;
3) реализовать и протестировать приложение.

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения, приложения и списка литературы. Объем работы составляет 37 страниц, объем списка литературы – 15 источников.

В первой главе описывается предметная область, используемое в работе API и рассмотрены аналоги.

Вторая глава предоставляет теоретическую информацию по предметной области.

В третьей главе рассматривается макет веб-приложения, а также описываются функциональные и нефункциональные требования.

Четвертая глава демонстрирует реализацию веб-приложения, создание карты, маркеров и меню.

В пятой главе описано функциональное тестирование веб-приложения.

В приложении представлен листинг реализации некоторых компонентов карты.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Веб-приложение, использующее OpenStreetMap API, представляет собой программное приложение, разработанное для работы в онлайн-среде и использующее функциональность, предоставляемую API (интерфейсом программирования приложений) OpenStreetMap. OpenStreetMap API предоставляет доступ к геоданным, картам, информации о местах и маршрутах на карте, а также другие возможности, благодаря которым можно интегрировать данные и функциональность OpenStreetMap во внешние приложения.

При разработке веб-приложения с использованием OpenStreetMap API можно реализовать такие функции, как отображение карты с точками интереса (POI), поиск мест и учреждений, отображение информации о местах, расчет маршрутов и другие геоинформационные операции. Такое веб-приложение может быть полезным для пользователей, которым требуется доступ к актуальным геоданным и картам, а также для бизнеса, связанного с оказанием услуг на территории, представленной на картах OpenStreetMap.

Разработка подобного веб-приложения [1] требует умения работать с веб-технологиями, знания OpenStreetMap API, обработки геоданных и умения проектировать удобный пользовательский интерфейс для работы с картами и геоинформацией.

1.2. Сравнительный анализ аналогов

Существует несколько аналогов OpenStreetMap API, которые также предоставляют геоданные, карты и другие геоинформационные ресурсы для использования в различных приложениях. Некоторые из таких аналогов рассмотрены далее.

«Google Maps API»

Google Maps API [2] предоставляет разработчикам возможность интегрировать карты Google Maps и геоданные в веб-приложения, мобильные приложения и другие приложения. Это один из наиболее популярных API в мире. Он имеет следующие недостатки.

1. Ограниченные бесплатные квоты на использование.
2. Сложности лицензирования для коммерческих продуктов.
3. Зависимость от экосистемы Google.

«Yandex Maps API»

Аналогично Google Maps API, Yandex Maps API [3] предоставляет доступ к картам и геоданным, разработанным Yandex. Его недостатки включают нижеописанные пункты.

1. Ограничения на количество запросов.
2. Возможные ограничения в отношении доступности данных за пределами определенных географических регионов.

«API 2GIS»

2GIS [4] – это сервис, предоставляющий карты и геоданные с акцентом на подробную информацию о городах. Недостатки API 2GIS приведены далее.

1. Ограниченная глобальная доступность. Основной фокус на России и странах СНГ, что ограничивает его полезность для глобальных приложений.
2. Ограничения по лицензированию и использованию данных. Использование данных может требовать подписки или лицензирования, особенно для крупных проектов или коммерческого использования.
3. Менее гибкая и ограниченная настраиваемость данных по сравнению с некоторыми другими сервисами.

Каждый из аналогов имеет свои особенности, преимущества и недостатки в зависимости от конкретных потребностей разработчика и приложения. При выборе аналога API важно учитывать требования к функцио-

нальности, ограничения по использованию и доступности геоданных для конкретной территории.

«Mapbox API»

Mapbox API [5] – это мощный инструмент для интеграции интерактивных карт и геоданных в веб-приложения. Mapbox предоставляет множество возможностей для кастомизации карт, работы с данными и интеграции с другими сервисами. Недостатки Mapbox API представлены ниже.

1. Стоимость. Хотя Mapbox предлагает бесплатный уровень, его возможности ограничены. Для доступа к более продвинутым функциям и большему количеству запросов требуется платная подписка.

2. Сложность освоения. Из-за широкого спектра возможностей и высокой степени кастомизации Mapbox может быть сложным для новичков, требуя больше времени на обучение и настройку.

3. Ограниченные офлайн-возможности. В отличие от некоторых других решений, поддержка офлайн-режима в Mapbox ограничена и требует дополнительных настроек и лицензий.

Вывод по первой главе

Технология, которая была выбрана для реализации веб-приложения, в большинстве своем, не обладает теми минусами, которыми наделены аналоги. OpenStreetMap API обладает рядом преимуществ по сравнению с другими популярными и часто используемыми картографическими API, а значит, может быть наиболее предпочтительным. Например, в сравнении с такими API как Mapbox API, 2GIS API, Yandex Maps API и Google Maps API, OpenStreetMap API обладает следующими преимуществами.

1. Свободное использование. Данные OSM доступны под открытой лицензией, что поможет свободно использовать их в проектах с минимальными ограничениями.

2. Глобальное покрытие. OSM предоставляет данные для всего мира, что делает его идеальным для международных проектов.

3. Сообщество и поддержка. Большое международное сообщество, которое постоянно обновляет и улучшает данные, а также предоставляет множество ресурсов для разработчиков.

4. Гибкость и настраиваемость. Высокая степень гибкости и возможность интеграции с различными сторонними сервисами и библиотеками, что позволяет настроить данные под конкретные нужды проекта.

5. Отсутствие зависимости от коммерческих экосистем. Использование OSM не требует обязательной интеграции с коммерческими продуктами и сервисами, что может снизить затраты и увеличить независимость проекта.

Технология достаточно популярна, поэтому взаимодействие с ней может быть эффективным. Она позволит создать веб-приложение, в котором будет реализована карта с фиксированными границами студгородка.

Еще один элемент, который позволяет реализовать эта технология – меню маркеров, в котором пользователь сможет выбрать нужное место и навестись на него, отсортировать необходимые пушапы, а также, в котором будет удобно ориентироваться.

Кроме того, будет возможность проложить маршрут от местоположения пользователя до определенной точки на карте и увидеть действия, которые нужно совершить для преодоления маршрута.

2.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. OSM API

API OpenStreetMap (OSM API) [6] – это программный интерфейс приложений, предоставляемый проектом OpenStreetMap. Благодаря этому, разработчики могут интегрировать функционал OpenStreetMap в свои собственные приложения или сервисы.

С помощью OpenStreetMap API можно получать доступ к различным данным, таким как информация о дорогах, зданиях, парках, реках и других объектах, которые представлены в OpenStreetMap. Можно получить информацию о местоположениях, географических координатах, атрибутах объектов и их взаимосвязях.

Одна из основных возможностей OpenStreetMap API – это работа с картами и геоданными. Разработчик может использовать API для отображения карт OpenStreetMap в своем приложении и добавления на них функционала. Например, размещать маркеры на карте, строить маршруты, определять географические координаты конкретных точек и т.д.

OpenStreetMap API также предоставляет доступ к данным о точках интереса (POI), включая кафе, рестораны, магазины, достопримечательности и другие важные места. Пользователи могут дополнять эти данные, что позволяет держать информацию актуальной и полезной.

Чтобы начать использовать OpenStreetMap API, необходимо зарегистрироваться на официальном сайте OpenStreetMap или воспользоваться сторонними сервисами, предоставляющими доступ к данным OSM, такими как Mapbox или Leaflet. После этого будет получен доступ к документации и ключу API, который необходимо использовать для авторизации при каждом запросе к API.

Структура и возможности OpenStreetMap API могут изменяться, поэтому рекомендуется ознакомиться с актуальной документацией OpenStreetMap API, чтобы быть в курсе последних обновлений и изменений.

В целом, OpenStreetMap API предоставляет широкий спектр возможностей для работы с данными OpenStreetMap и интеграции этой информации в свои приложения или сервисы. Он может быть полезен для разработчиков, которым необходим доступ к географической информации, картографическим данным и другим объектам, представленным в OpenStreetMap.

2.2. React

React [7] – это библиотека для создания пользовательских интерфейсов, разработанная и поддерживаемая Facebook. React предоставляет разработчикам возможность создавать динамичные и интерактивные веб-приложения, благодаря компонентному подходу, позволяющему разбивать интерфейс на мелкие, переиспользуемые части.

Основные возможности React представлены ниже.

1. Создание пользовательских интерфейсов. React предоставляет разработчикам инструменты для построения высокоэффективных интерфейсов. Компоненты являются основным строительным блоком в React. Они могут быть функциональными или классовыми и используются для инкапсуляции логики и представления.

2. JSX – расширение JavaScript. React использует JSX, синтаксическое расширение JavaScript, которое позволяет писать HTML-подобный код внутри JavaScript. Это упрощает процесс создания компонентов и повышает читаемость кода.

3. Управление состоянием. React компоненты могут иметь состояние (state), которое отслеживает изменения данных внутри компонента. Состояние управляется внутри компонента и используется для обновления пользовательского интерфейса в ответ на изменения данных.

4. Реактивные обновления. React использует виртуальный DOM для повышения производительности. Когда состояние компонента изменяется,

React обновляет виртуальный DOM и затем эффективно изменяет реальный DOM, обновляя только те элементы, которые были изменены.

5. Hooks – современный подход к управлению состоянием и эффектами. С введением React Hooks в версии 16.8, разработчики могут использовать функциональные компоненты для управления состоянием и жизненным циклом компонента, что ранее было возможно только с классовыми компонентами.

6. Context API для глобального состояния. Context API передает данные через дерево компонентов без необходимости явно передавать пропсы на каждом уровне. Это упрощает управление глобальным состоянием приложения.

React предоставляет мощный и гибкий инструментарий для создания современных веб-приложений. Благодаря компонентному подходу, JSX, управлению состоянием, реактивным обновлениям, хукам и Context API, разработчики могут эффективно и удобно строить сложные интерфейсы, обеспечивая высокую производительность и легкость в поддержке.

2.3. Хуки React

Хуки [8] – это важная концепция в современных фреймворках для разработки пользовательских интерфейсов, особенно в React. Они позволяют использовать состояние и другие возможности React без написания классов.

Существуют следующие концепции хуков.

1. Хуки состояния (State Hooks). Этот хук возвращает пару значений: текущее состояние и функцию для его обновления.

2. Хуки эффекта (Effect Hooks). Этот хук принимает функцию и массив зависимостей. Функция будет выполнена после рендера, и снова при изменении одной из зависимостей.

3. Хуки контекста (Context Hooks). Он используется для доступа к контексту, созданному с помощью React.createContext. Благодаря этому

есть возможность передавать данные через дерево компонентов без необходимости пробрасывать пропсы вручную.

4. Хуки редьюсера (Reducer Hooks). Он принимает редьюсер и начальное состояние, возвращая текущее состояние и функцию диспетчера для изменения состояния.

Хуки должны вызываться только на верхнем уровне. Их нельзя вызывать внутри циклов, условий или вложенных функций. Это гарантирует, что хуки вызываются в одном и том же порядке при каждом рендере. Хуки должны вызываться только из функциональных компонентов или собственных хуков

В целом, хуки предоставляют мощный способ управления состоянием и жизненным циклом в функциональных компонентах React. Они делают код более декларативным, модульным и легким для понимания. Понимание и использование хуков является ключевым навыком для современных фронтенд-разработчиков.

2.4. Leaflet

Leaflet [9] – это легкая, открытая JavaScript-библиотека для интерактивных карт. Она является одним из самых популярных инструментов для работы с картами в веб-приложениях. Leaflet позволяет разработчикам легко добавлять карты в свои приложения и предоставляет широкий набор возможностей для работы с геоданными.

Основные возможности Leaflet приведены ниже.

1. Отображение карт. Leaflet легко интегрирует карты в веб-приложения. Поддержка различных картографических сервисов, таких как OpenStreetMap, Mapbox и других, обеспечивает гибкость и широкие возможности для кастомизации.

2. Работа с маркерами и всплывающими окнами. Leaflet добавляет маркеры на карту, а также создавать всплывающие окна для отображения

дополнительной информации. Это полезно для отображения точек интереса (POI) и другой информации на карте.

3. Полигоны и линии. Leaflet поддерживает рисование различных геометрических фигур, таких как полигоны и линии. Это полезно для визуализации маршрутов, областей и других географических данных.

4. События и интерактивность. Leaflet предоставляет широкие возможности для работы с событиями, такими как клики, зумирование и перетаскивание. Благодаря этому есть возможность создавать интерактивные карты с богатым функционалом.

5. Поддержка плагинов. Leaflet имеет богатую экосистему плагинов, которые расширяют его функциональность. Существуют плагины для кластеризации маркеров, отображения тепловых карт, рисования и редактирования геометрии и многие другие.

6. Интеграция с React. Leaflet можно легко интегрировать с React с помощью библиотек, таких как react-leaflet, что позволяет использовать всю мощь React для создания сложных и динамичных картографических приложений.

В заключение можно сказать, что Leaflet – это мощная и гибкая библиотека для работы с интерактивными картами. Благодаря простоте использования, богатому набору функций и поддержке плагинов, Leaflet является отличным выбором для разработчиков, которым необходимо интегрировать картографические данные и функционал в свои веб-приложения.

2.5. JSON

JSON (JavaScript Object Notation) [10] – простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999. JSON – текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, та-

ких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными.

JSON основан на двух структурах данных.

1. Коллекция пар ключ/значение. В разных языках, эта концепция реализована как объект, запись, структура, словарь, хэш, именованный список или ассоциативный массив.

2. Упорядоченный список значений. В большинстве языков это реализовано как массив, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме. Логично предположить, что формат данных, независимый от языка программирования, должен быть основан на этих структурах.

2.6. Диаграмма вариантов использования

Диаграммы вариантов использования являются важным инструментом в проекте по многим причинам. Ниже представлены некоторые из них.

1. Определение функциональных требований.
2. Коммуникация с заказчиком.
3. Планирование разработки.
4. Проектирование архитектуры.
5. Разработка тестовых сценариев.
6. Документация и поддержка.
7. Управление изменениями.

Была разработана диаграмма вариантов использования [11], в которой представлены возможности взаимодействия с веб-приложением. Разработанная диаграмма предоставлена на рисунке 1.

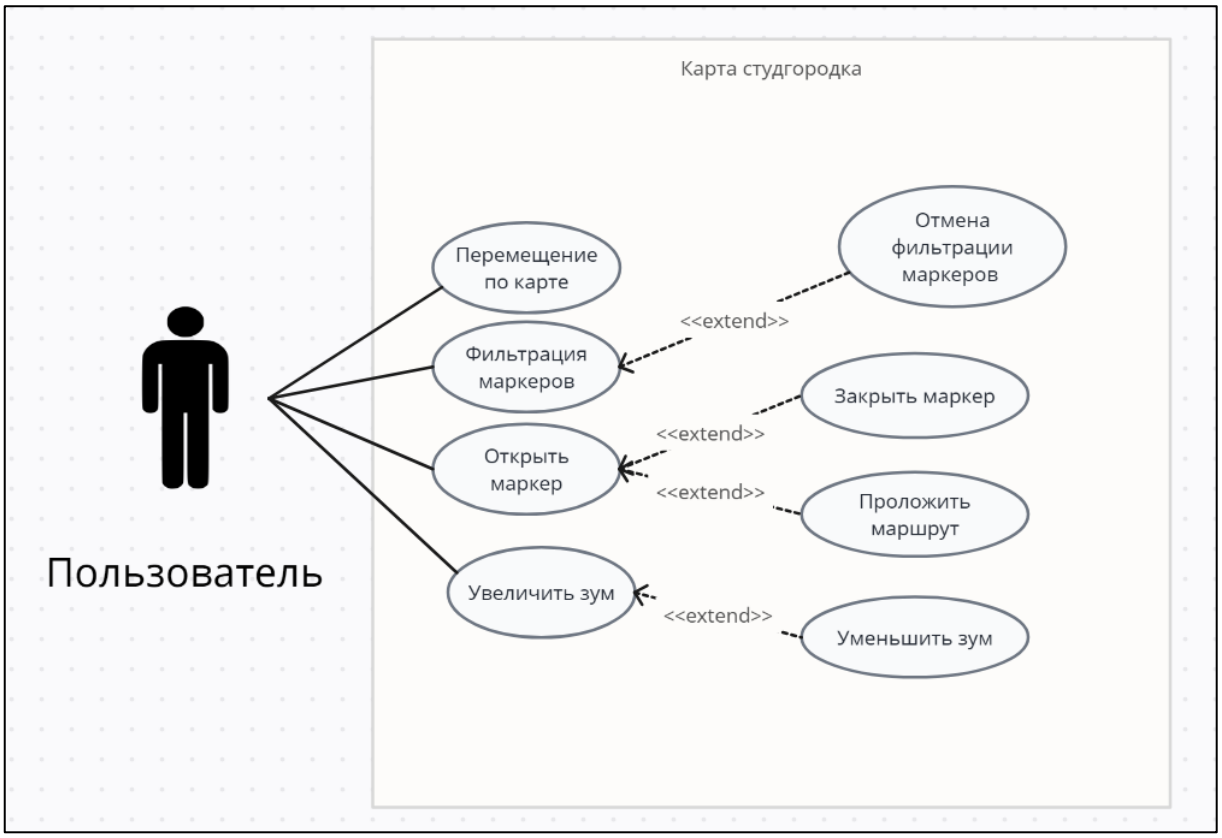

Рисунок 1 – Диаграмма вариантов использования

Краткое описание вариантов использования представлено дальше.

1. Пользователь попадает на главный экран карты со всеми маркерами, где он может перемещаться, открывать или закрывать маркеры, фильтровать маркеры с помощью меню слева, а также увеличивать и уменьшать зум.

2. При открытии маркера есть возможность прочитать содержимое в нем, простроить маршрут и закрыть его.

3. Фильтр маркера предоставляет возможность увидеть лишь те маркеры, которые необходимы пользователю, при этом он так же может перемещаться по карте, открывать и закрывать маркеры, увеличивать и уменьшать зум.

Вывод по второй главе

Рассмотрев технологию OpenStreetMap, было заключено, что она отлично подойдет для выполнения поставленных задач. Потенциально у пользователя будет возможность для перемещения по карте веб-

приложения, включая увеличение и уменьшение зума, а также взаимодействие с пушап-маркерами карты, то есть открытие и закрытие оных, получение информации, помещенной в всплывающий маркер, фильтрация и обратная фильтрация по типам маркеров, например, фильтр маркеров по критерию «Учебные корпуса» выведет на экран только соответствующие маркеры. Кроме того, у пользователя будет возможность простроить оптимальный маршрут до определенного маркера, это также позволит ему увидеть действия, которые ему необходимо сделать для преодоления данного маршрута.

OpenStreetMap предоставляет широкий спектр возможностей для разработчика, который планирует работать с технологиями ГИС для поиска, например, географической информации. Она позволяет разработчикам интегрировать данные и функциональность OpenStreetMap в свои приложения. С OpenStreetMap разработчики могут получать доступ к информации о местоположениях, картам, маршрутам и другим геопространственным данным. Это делает OpenStreetMap ценным инструментом для различных отраслей, включая логистику, недвижимость, розничную торговлю и городское планирование. На фоне аналогов OpenStreetMap выглядит наиболее предпочтительным для задействования этой технологии в выпускной квалификационной работе.

3.ПРОЕКТИРОВАНИЕ

3.1. Макет интерфейса

Интерфейс веб-приложения [12] будет представлять собой карту студгородка и меню для выбора различных фильтров для взаимодействия с картой (рисунок 2).

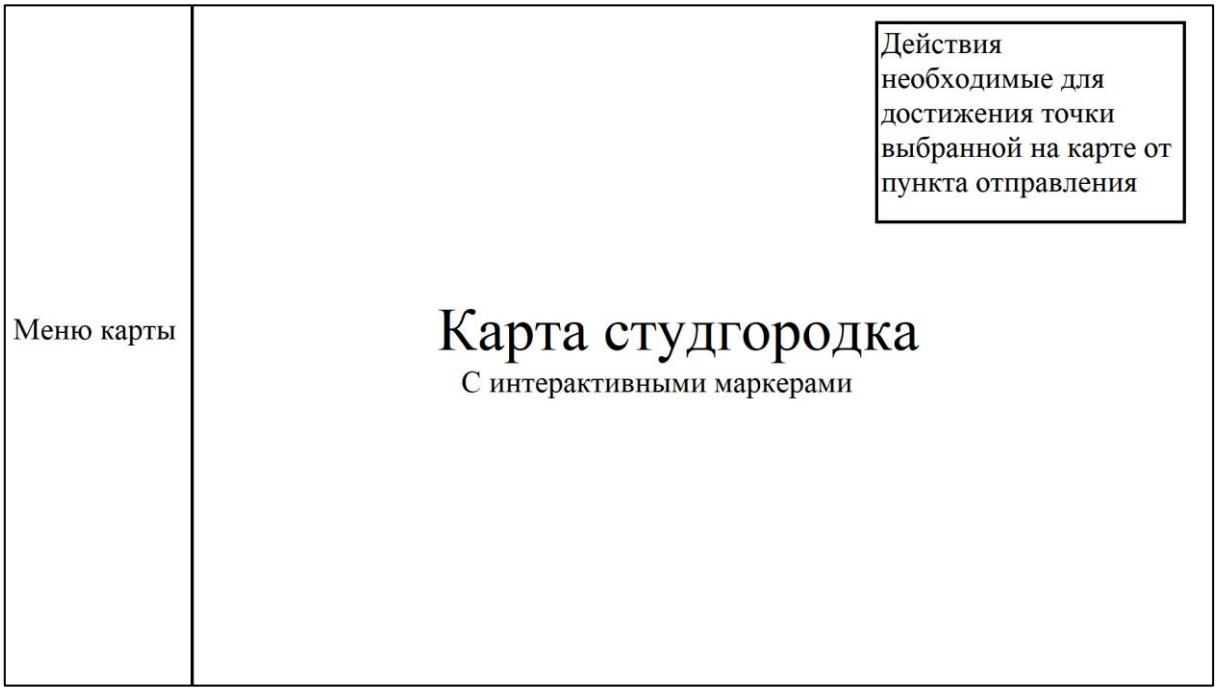

Рисунок 2 – Макет сайта

Меню будет интуитивно понятным любому пользователю, поскольку оно будет выполнено в минималистичном стиле, здесь будут отображены все маркеры, разделенные на категории с соответствующими названиями для фильтров. Фильтры представлены далее.

1. «Учебные корпуса». Делает фильтр учебных корпусов ЮУрГУ.
2. «Общежития. Делает фильтр общежитий в пределах студгородка.
3. «Достопримечательности». Делает фильтр достопримечательностей, памятных мест и т.д.
4. «Закусочные». Делает фильтр закусочных, кафе и магазинов.

3.2. Функциональные требования и нефункциональные требования

Функциональные требования и нефункциональные требования [13] играют ключевую роль в разработке программного обеспечения или со-

здании информационной системы. Обе категории требований крайне важны для успешного выполнения проекта, поскольку они помогают понять цели и ограничения разрабатываемой системы.

В ходе проектирования были определены требования к системе и возможности, которые должны предоставляться пользователю при использовании системы.

Функциональные требования – это накладываемые на поведение системы условия и действия, которые система имеет возможность выполнять.

Разрабатываемая система должна удовлетворять следующим функциональным требованиям.

1. Отображение основных зданий, учебных корпусов, мест для занятия спортом, больниц, поликлиник, общежитий, закусочных, достопримечательностей и других важных объектов на территории студенческого городка.

2. Возможность поиска конкретных зданий или объектов при помощи соответствующей строки или необходимых фильтров.

3. Предоставление подробной информации о каждом здании или объекте, включая его название и описание.

4. Возможность отфильтровать места на карте по значкам для более удобного ориентирования, а также возможность обратной фильтрации чтобы отобразить все маркеры на карте.

Нефункциональные требования – это не связанные с поведением системы условия. Они характеризуют условия окружающей среды или же качества, которыми должна обладать система.

Для выполнения проекта необходимо четко и ясно определить эти требования. Разрабатываемая система должна удовлетворять следующим нефункциональным требованиям.

1. Быструю загрузку и отзывчивость интерактивной карты, чтобы пользователи могли удобно взаимодействовать с ней без задержек.

2. Поддержку мобильных устройств для обеспечения удобства доступа к картам и информации на любых устройствах.

3. Простой и интуитивно понятный интерфейс, удобный для студентов, сотрудников университета и посетителей студенческого городка.

Вывод по третьей главе

Создав макет веб-приложения, а также определив функциональные и нефункциональные требования, можно прийти к выводу, что интерфейс будет интуитивно понятным простому пользователю, чтобы он мог свободно ориентироваться на карте студенческого городка. Функциональные требования предполагают, что веб-приложение будет отображать основные маркеры, фильтровать их, читать описание и возвращаться на «главную страницу» для отображения всех маркеров вновь. Нефункциональные требования подразумевают, что веб-приложение будет иметь быструю загрузку и поддержку мобильных устройств. Так же, предполагается, что пользователь будет иметь возможность перемещаться по карте в пределах студенческого городка, увеличивать и уменьшать зум, тем самым приближая или отдаляя карту. Изначально карта установлена на негласном центре студенческого городка – главном корпусе ЮУрГУ.

4. РЕАЛИЗАЦИЯ

4.1. Правила

В этом разделе выпускной квалификационной работы рассматриваются правила поведения программы, включающие обработку глобальных ошибок и управление жизненным циклом приложения. Представленные листинги демонстрируют, как настроено приложение для обеспечения стабильности и производительности, а также, как реализована основная логика рендеринга.

Программное обеспечение должно быть устойчивым к различным ошибкам и непредвиденным ситуациям. Одним из подходов к достижению этой цели является внедрение глобальных обработчиков ошибок, которые позволяют отлавливать и ликвидировать ошибки, возникающие в процессе выполнения программы. Кроме того, важно организовать правильный рендеринг компонентов и сбор метрик производительности, чтобы приложение функционировало эффективно и предоставляло пользователям оптимальный опыт.

В следующем листинге представлена точка входа для React-приложения. Он включает настройку глобальных обработчиков ошибок, создание корневого элемента для рендеринга, а также рендеринг основного компонента приложения и сбор метрик производительности (листинг 1).

Листинг 1 – Подключение элементов интерфейса

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

window.onerror = function(message, source, lineno, colno, error) {
    console.error("Global error caught: ", message, source, lineno, colno,
error);
    return true;
};

window.onunhandledrejection = function(event) {
    console.error("Unhandled rejection caught: ", event);
    return true;
};
const root = ReactDOM.createRoot(
    document.getElementById('root') as HTMLElement
);
```

```
root.render(
    <React.StrictMode>
        <App />
    </React.StrictMode>
);
reportWebVitals();
```

Для создания меню было решено использовать язык иерархических правил CSS [14]. В данном разделе рассмотрим основные CSS-правила, используемые для стилизации базовых элементов страницы.

Представленный листинг кода содержит стили для элемента <body> и тегов <code>, которые играют ключевую роль в визуальном восприятии веб-приложения. Настройки включают в себя следующие пункты.

1. Устранение стандартных отступов браузера для элемента <body>, что позволяет контенту страницы начинаться с самого края окна браузера.

2. Задание семейства шрифтов, включающего кроссплатформенные шрифты, обеспечивающие корректное отображение текста на различных системах.

3. Применение техник сглаживания шрифтов для улучшения их отображения в разных браузерах и операционных системах.

4. Определение моноширинных шрифтов для элементов <code>, что улучшает читаемость программного кода.

Эти стили способствуют созданию согласованного и профессионального интерфейса, соответствующего современным стандартам веб-дизайна. Рассмотрение кода поможет глубже понять, как достигнута функциональность веб-приложения (листинг 2).

Листинг 2 – CSS-правила для стилизации HTML-документа

```
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

Следующий файл представляет собой package.json – конфигурационный файл, который управляет зависимостями и скриптами проекта (листинг 3).

Листинг 3 – Зависимости и скрипты проекта

```
{
  "name": "nikolaich",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "@types/jest": "^27.5.2",
    "@types/leaflet-routing-machine": "^3.2.8",
    "@types/node": "^16.18.97",
    "@types/react": "^18.3.2",
    "@types/react-dom": "^18.3.0",
    "leaflet": "^1.9.4",
    "leaflet-routing-machine": "^3.2.12",
    "react": "^18.3.1",
    "react-dom": "^18.3.1",
    "react-leaflet": "^4.2.1",
    "react-scripts": "5.0.1",
    "typescript": "^4.9.5",
    "web-vitals": "^2.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test",
    "eject": "react-scripts eject"
  },
  "eslintConfig": {
    "extends": [
      "react-app",
      "react-app/jest"
    ]
  },
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "not op_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
    "@types/leaflet": "^1.9.12",
    "file-loader": "^6.2.0"
  }
}
```

Этот package.json файл управляет основными аспектами проекта: зависимостями, скриптами, конфигурацией линтера и поддержкой браузеров. Он играет ключевую роль в настройке среды разработки и сборки приложения, обеспечивая корректное функционирование и тестирование кода.

4.2. Подключение основных элементов интерфейса

Основой приложения будет файл `App`, который объединяет следующие элементы интерфейса: карту, боковую панель и модальное окно. Этот компонент отвечает за управление состоянием приложения и взаимодействие между различными частями интерфейса. Рассмотрение кода компонента App поможет понять, как реализована логика работы с картой и управление категориями маркеров.

Приведенный ниже листинг кода демонстрирует структуру и функциональные возможности компонента `App`. В нем используются хуки `useState` для управления состоянием, а также импортируются и используются другие компоненты приложения (листинг 4).

Листинг 4 – Подключение элементов интерфейса

```
import {useState} from 'react';
import './App.css';
import {MapComponent} from "./components/MapComponent";
import {SidebarComponent} from "./components/SidebarComponent";
import L, {LatLngExpression, LatLngBoundsExpression} from 'leaflet';
import {Modal} from "./components/Modal";
import {markersData} from "./data/markersData";

function App() {

    const [center, setCenter] = useState<LatLngExpression>([55.160119,
61.37031])
    const [visible, setVisible] = useState(false)
    const [activeCategories, setActiveCategories] = useState(
        Object.keys(markersData).reduce((acc, category) => ({...acc, [cate-
gory]: true}), {})
    );

    const toggleCategory = (category: string) => {
        setActiveCategories((prev: { [key: string]: boolean }) =>
({...prev, [category]: !prev[category]}));
    }

    return (
        <div className="App" style={{display: "flex"}}>
```

```
            <SidebarComponent setCenter={setCenter} center={center}
setVisible={setVisible}/>
            <MapComponent center={center} activeCatego-
ries={activeCategories}/>
            <Modal visible={visible} setVisible={setVisible} activeCatego-
ries={activeCategories}
                   toggleCategory={toggleCategory}/>
        </div>
    );
}

export default App;
```

Таким образом были подключены основные элементы интерфейса, которые будут управлять состоянием приложения.

4.3. Компоненты карты

Следующие листинги представляют собой компонент карты на основе библиотеки React и Leaflet, который отображает карту с маркерами и маршрутом между пользователем и выбранным пунктом назначения. Подробное рассмотрение кода позволит глубже понять подходы к разработке подобных веб-приложений и лучшие практики использования современных веб-технологий (листинг 5).

Листинг 5 – Импорт модулей и стилей

```
import React, {useState, useEffect, useRef} from 'react';
import { MapContainer, TileLayer, Marker, Popup, useMap, useMapEvent } from
'react-leaflet';
import L, { LatLngExpression, LatLngBoundsExpression } from 'leaflet';
import 'leaflet/dist/leaflet.css';
import icon from 'leaflet/dist/images/marker-icon.png';
import iconShadow from 'leaflet/dist/images/marker-shadow.png';
import 'leaflet-routing-machine';
import 'leaflet-routing-machine/dist/leaflet-routing-machine.css';
import { markersData } from "../data/markersData";
import ErrorBoundary from './Error';
const DefaultIcon = L.icon({
    iconUrl: icon,
    shadowUrl: iconShadow
});
L.Marker.prototype.options.icon = DefaultIcon;

const ChangeView = ({ center }: { center: LatLngExpression }) => {
    const map = useMap();
    useEffect(() => {
        if (map) {
            map.flyTo(center);
        }
    }, [center, map]);
    return null;
}
```

Как видно из листинга, импортируются React хуки, стили Leaflet, иконки для маркеров, Leaflet Routing Machine для построения маршрутов, данные о маркерах и компонент для обработки ошибок. После подключения, происходит настройка иконок для маркеров и компонент `ChangeView`, который изменяет центр карты при изменении значения center.

Следующий листинг показывает компонент `RoutingControl`. Он добавляет маршрутизатор на карту, который строит маршрут между пользователем и необходимым местом (листинг 6).

Листинг 6 – `RoutingControl`

```
const RoutingControl = ({ userLocation, destination }: { userLocation:
LatLngExpression, destination: LatLngExpression }) => {
    const map = useMap();
    const routingControlRef = useRef<any>(null);

    useEffect(() => {
        if (!userLocation || !destination) return;

        if (routingControlRef.current) {
            map.removeControl(routingControlRef.current);
            routingControlRef.current = null;
        }

        routingControlRef.current = L.Routing.control({
            waypoints: [
                L.latLng(userLocation),
                L.latLng(destination)
            ],
            routeWhileDragging: true
        }).addTo(map);

        return () => {
            if (map && routingControlRef.current) {
                map.removeControl(routingControlRef.current);
                routingControlRef.current = null;
            }
        };
    }, [userLocation, destination, map]);

    return null;
};
```

Переходим к главному компоненту `MapComponent`. Он обеспечивает интерактивную карту с функциональностью навигации, что поможет пользователю видеть свое местоположение и строить маршруты до выбранных пунктов назначения. Пример этого компонента представлен в приложении.

Таким образом, были созданы основные компоненты карты, которые позволят перемещаться по ней, видеть свое местоположение, строить маршрут до определенного места и взаимодействовать с маркерами.

4.4. Маркеры

Чтобы была возможность отмечать на карте точки и читать их описание необходимо добавить иконку с информацией о конкретном месте. Маркер представляет собой пушап, при нажатии на который появляется текст с описанием маркера. Кроме того, есть возможность создать свою иконку для маркера в формате «png» или «jpg».

После того как иконка была подключена, необходимо отобразить ее на карте. Для этого нужно создать объект, что позволит создать маркер с переданными географическими координатами. Примером создания маркера на карте служит листинг 7.

Листинг 7 – Создание маркеров на карте

```
import susuIcon from "../icons/susu.png";
import homeIcon from "../icons/home.png";
import memorialIcon from "../icons/memorial.png";
import treeIcon from "../icons/tree.png";
import foodIcon from "../icons/food.png";
import { Markers } from "../types/types";

export const markersData: Markers = {
  "Учебные корпуса": [
    {
      position: [55.160374, 61.3701882],
      icon: susuIcon,
      popup:
        "Главный корпус ЮУрГУ",
    },
    {
      position: [55.160493, 61.3715782],
      icon: susuIcon,
      popup: "Учебный корпус 1б",
    },
```

Аналогично создаются и маркеры общежитий, достопримечательностей и заведений общественного питания.

4.5. Меню

В следующем разделе внешний вид и поведение ключевых элементов интерфейса: основного контейнера приложения, боковой панели и модального окна (листинг 8).

Листинг 8 – Создание дизайна меню и кнопок

```
.App {height: 100%}
.sidebar {
    width: 25vw;
    background-color: black;
    overflow-x: hidden;
    padding: 20px 10px 20px 15px;
}
.sidebar h1 {
    color : #818181;
}
.sidebar ul button, .sidebar h2 {
    text-decoration: none;
    font-size: 16px;
    color: #818181;
    display: block;
    border: none;
    background: transparent;
}
.sidebar ul button:hover, .sidebar h2:hover {
    transition: 0.1s ease-in;
    color: #f1f1f1;
}
.modal_back {
    background: rgba(0, 0, 0, 0.5);
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    z-index: 999; /* увеличиваем z-index */
}
.modal {
    display: flex;
    flex-direction: column;
    background: lightgray;
    border-radius: 15px;
    padding: 10px;
    box-shadow: black 0px 0px 10px;
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    bottom: 0;
    z-index: 1000; /* увеличиваем z-index */
    width: 500px;
    max-height: 500px;
    margin: auto;
}
```

Задается цвет меню, текста в нем размеры, а также подключаются кнопки, созданные в прошлом листинге. На данный момент созданы четыре фильтра карты, а именно: учебные корпуса, общежитие, достопримечательности и закусочные.

Далее представлена реализация этого меню, где есть возможность добавить фильтр отображения определенных маркеров на карте в модальном (листинг 9).

Листинг 9 – Наполнение меню

```
import {markersData} from "../data/markersData";

export const SidebarComponent = ({setCenter, center, setVisible}:
{setCenter: any, center: any, setVisible: any}) => {

    const handlleChangePosition = (position: any) => {
        console.log(position)
        setCenter(position)

        console.log(center, 'çenter')
    }
    return (
        <div className="sidebar">
            <h1>Список мест</h1>
            {Object.keys(markersData).map(category => (
                <div key={category}>
                    <h2>{category}</h2>
                    <ul>
                        {markersData[category].map((marker, index) => (
                            <li  key={index}> <button
style={{background:"none", border: "none", padding: 0}} onClick={() =>
handlleChangePosition(marker.position)}> {marker.popup}</button></li>
                        ))}
                    </ul>
                </div>
            ))}
            <h3> Фильтр</h3>
            <button onClick={() => setVisible(true)}>Открыть модальное
окно</button>
        </div>
    );
}
```

Работа фильтрации маркеров приведена на примере листинга 10.

Листинг 10 – Правила фильтрации

```
import {markersData} from "../data/markersData";
import { useState } from 'react';


export const Modal = ({ visible, setVisible, activeCategories, toggleCate-
gory }: { visible: boolean, setVisible: any, activeCategories: { [key:
string]: boolean }, toggleCategory: (category: string) => void }) => {
    return (
```

```
        <>
            {visible &&
                <div className="modal_back">

                    <div className="modal" >
                        <div style={{display:"flex", justifyContent:"flex-
end"}} >
                            <button onClick={() => setVisible(false)}>
                            x
                        </button></div>
                        {Object.keys(markersData).map(category => (
                            <div style={{display:"flex", gap:"10px"}}
key={category}>
                                <input type="checkbox"
checked={activeCategories[category]}
                                       onChange={() => toggleCatego-
ry(category)}/>
                                <h2>{category}</h2>
                            </div>
                        ))}
                    </div>
                </div>


            }
        </>
    )
}
```

Вывод по четвертой главе

В главе «Реализация» была продемонстрирована программная реализация веб-приложения. Была подключена необходимая технология и карта с центром с координатами главного корпуса ЮУрГУ, функции которой перечислены ниже.

1. Перемещение по карте.
2. Увеличение и уменьшение зума.
3. Установление границ студгородка.
4. Определение текущего местоположения пользователя.
5. Прокладывание маршрута до определенной точки на карте.

Это необходимо для самого базового ориентирования в студенческом городке ЮУрГУ.

Также были установлены пушап-маркеры, которые будут являться основным инструментом в разработке веб-приложения. Они будут показывать местоположение того или иного здания, предоставлять возможность простроить маршрут до него и т.д.

Для удобства было реализовано меню маркеров. При взаимодействии с ним пользователь может выбрать любой маркер на карте и быстро сфокусироваться на нем. Меню содержит следующие четыре категории.

1. Учебные корпуса.
2. Общежития.
3. Достопримечательности.
4. Закусочные.

Каждую категорию можно отфильтровать в модальном окне и на карту будут выведены только определенные и необходимые пользователю маркеру.

При реализации были задействованы следующие технологии.

1. Язык иерархических правил CSS.
2. Библиотека Leaflet.
3. Библиотека React.
4. Хуки React.
5. Формат обмена данными JSON.
6. OpenStreetMap API.

Каждая технология была описана и реализована в соответствии с поставленными задачами.

Реализация отвечает всем функциональным и нефункциональным требованиям.

5. ТЕСТИРОВАНИЕ СИСТЕМЫ.

Функциональное тестирование [15] – это способ определить, работает ли программное обеспечение или приложение так, как ожидается. Функциональное тестирование интересуется не тем, как происходит обработка данных, а тем, обеспечивает ли она правильные результаты или имеет какие-либо ошибки. Итоги функционального тестирования представлены в таблице 1.

Таблица 1 – Функциональное тестирование

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
1	Открытие маркера	Маркер открылся, появилась информация о месте и был простроен маршрут	При нажатии на маркер он раскрылся и показал необходимую информацию, был простроен маршрут	Да
2	Фильтрация маркеров при помощи меню фильтров	Фильтрация прошла успешно, теперь показаны только необходимые пользователю маркеры.	При нажатии на кнопку в меню слева были показаны соответствующие кнопке маркеры.	Да
3	Отмена фильтрации маркеров при помощи модального окна	Фильтрация отменена, теперь показываются все маркеры	При нажатии на кнопки в модальном меню были показаны все маркеры.	Да
4	Построить маршрут до определенного маркера	Оптимальный маршрут построен, и пользователь видит куда ему нужно идти	При нажатии на маркер карта построила маршрут к маркеру и показала, как добраться до него	Да

Вывод по пятой главе

В ходе разработки было проведено функциональное тестирование API. Тесты были пройдены успешно, были получены ожидаемые результаты, что указывает на то, что система работает так, как было задумано.

ЗАКЛЮЧЕНИЕ

В рамках данной работы была разработана страница с картой студенческого городка с использованием различных маркеров для более удобной ориентации. При этом были решены следующие задачи.

1. Был проведен анализ предметной области и обзор аналогов.
2. Было спроектировано веб-приложение.
3. Приложение было реализовано и протестировано.

Как и предполагалось, веб-приложение представляет собой карту студенческого городка с обозначениями в виде маркеров к которым пользователь может простроить маршрут и информацию к которым он может прочитать при нажатии на пушап-маркер. При построении маршрута пользователю будет простроен наиболее рациональный маршрут, а также предоставлено меню в правом верхнем углу экрана с инструкцией, как его преодолеть. Для удобства, было реализовано меню маркеров, чтобы при нажатии на маркер в меню, камера фиксировалась на нем. Плюс ко всему, в меню маркеров есть возможность открыть модальное окно, где пользователь сможет отсортировать необходимые ему маркеры, чтобы отображались только необходимые пользователю места посещения, например, если пользователю нужно увидеть только общежития, он убирает галочку со всех остальных маркеров в модальном окне и на карте показываются только общежития.

ЛИТЕРАТУРА

1. Кириченко А., Никольский А., Дубовик Е. Web на практике. CSS, HTML, JavaScript, MySQL, PHP для fullstack-разработчиков, 2021. – 432 с.

2. Google Maps Platform Documentation. [Электронный ресурс] URL: https://developers.google.com/maps/documentation?hl=ru. (дата обращения: 26.05.2024 г.).

3. Документация для разработчиков – API Яндекс Карты. [Электронный ресурс] URL: https://yandex.ru/maps-api/docs. (дата обращения: 28.05.2024 г.).

4. 2GIS Документация. [Электронный ресурс] URL: https://docs.2gis.com/ru. (дата обращения: 29.05.2024 г.).

5. Mapbox Docs. [Электронный ресурс] URL: https://docs.mapbox.com (дата обращения: 01.06.2024 г.).

6. OpenStreetMap Documentation. [Электронный ресурс] URL: https://wiki.openstreetmap.org/wiki/RU:Руководство_новичка (дата обращения: 02.06.2024 г.).

7. React Documentation. [Электронный ресурс] URL: https://react.dev/learn (дата обращения: 03.06.2024 г.).

8. Введение в хуки. [Электронный ресурс] URL: https://ru.legacy.reactjs.org/docs/hooks-intro.html. (дата обращения: 04.06.2024 г.).

9. Documentation – Leaflet. [Электронный ресурс] URL: https://mourner.github.io/Leaflet/reference.html (дата обращения: 05.06.2024 г.).

10. Введение в JSON. [Электронный ресурс] URL: https://www.json.org/json-ru.html. (дата обращения: 05.06.2024 г.).

11. Составление диаграммы вариантов использования. [Электронный ресурс] URL: https://creately.com/ru/diagram-type/diagramma-variantov-ispolzovaniya/. (дата обращения: 07.06.2024 г.).

12. Дакетт Д. HTML и CSS Разработка и дизайн веб-сайтов. // Издательство Эксмо, 2013. – 478 с.

13. Бубнов А., Бубнов С., Майков К. Разработка и анализ требований к программному обеспечению. – КУРС, 2023. – 176 с.

14. Руководство по CSS - CSS: каскадные таблицы стилей | MDN. [Электронный ресурс] URL: https://developer.mozilla.org/ru/docs/Web/CSS/Reference. (дата обращения 07.06.2024 г.).

15. Функциональное тестирование ПО. [Электронный ресурс] URL: https://unetway.com/tutorial/funkcionalnoe-testirovanie (дата обращения: 07.06.2024 г.).

ПРИЛОЖЕНИЕ. Компонент программы «MapComponent»

Листинг 1 – MapComponent

```
export const MapComponent = ({ center, activeCategories }: { center:
LatLngExpression, activeCategories: { [key: string]: boolean } }) => {
    const [userLocation, setUserLocation] = useState<LatLngExpression |
null>(null);
    const [destination, setDestination] = useState<LatLngExpression |
null>(null);
    const zoom = 15.5;
    const maxBounds: LatLngBoundsExpression = [
        [55.169087, 61.33991],
        [55.151951, 61.390617],
    ];

    useEffect(() => {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(position => {
                setUserLocation([position.coords.latitude, posi-
tion.coords.longitude]);
            });
        }
    }, []);

    const handleMarkerClick = (position: LatLngExpression) => {
        setDestination(position);
    };

    return (
        <ErrorBoundary>
            <MapContainer center={center} maxBounds={maxBounds} zoom={zoom}
                          minZoom={17}
                          maxBoundsViscosity={1.0}
                          style={{ height: '100vh', width: '100%' }}>
                <ChangeView center={center} />
                <TileLayer

url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
                    attribution='&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contribu-
tors'
                />
                {userLocation && (
                    <Marker position={userLocation}>
                        <Popup>You are here</Popup>
                    </Marker>
                )}
                {Object.keys(markersData).filter(category => activeCatego-
ries[category]).map(category =>
                    markersData[category].map((markerData, index) => (
                        <Marker
                            key={`${category}-${index}`}
                            position={markerData.position}
                            icon={L.icon({
                                iconUrl: markerData.icon,
                                iconSize: [25, 25],
                            })}
                            eventHandlers={{
                                click: () => handleMarker-
Click(markerData.position)
                            }}
                        >
                            <Popup>{markerData.popup}</Popup>
```

Окончание листинга 1 приложения

```
                        </Marker>
                    ))
                )}
                {userLocation && destination && (
                    <RoutingControl userLocation={userLocation} destina-
tion={destination} />
                )}
            </MapContainer>
        </ErrorBoundary>
    );
}
```