Содержание

[Введение 7](#_Toc169687721)

[Глава 1. Постановка задачи и методы её решения 11](#_Toc169687722)

[1.1 Цель работы 11](#_Toc169687723)

[1.2 Техническое задание по выпускной работе 11](#_Toc169687724)

[Функциональные требования к приложению 11](#_Toc169687725)

[Нефункциональные требования к приложению 12](#_Toc169687726)

[1.3 Методы и способы решения разрабатываемого ПО 12](#_Toc169687727)

[Использование сторонних картографических сервисов 12](#_Toc169687728)

[Разработка с нуля с использованием веб-технологий 13](#_Toc169687729)

[Комбинированный метод 14](#_Toc169687730)

[Особенности функционирования создаваемого ПО 14](#_Toc169687731)

[Интерфейс (дизайн) ПО 15](#_Toc169687732)

[Требования по используемой аппаратуре и вспомогательного ПО 15](#_Toc169687733)

[1.4 Описание разработки (отрисовки) карт 15](#_Toc169687734)

[Глава 2. Обзор существующих программных средств 17](#_Toc169687735)

[2.1 Анализ основных технологических решений 17](#_Toc169687736)

[Выбор языка программирования 17](#_Toc169687737)

[Выбор среды разработки 17](#_Toc169687738)

[Выбор библиотек и фреймворков 18](#_Toc169687739)

[Выбор редактора SVG 20](#_Toc169687740)

[Выбор языка для серверной части 20](#_Toc169687741)

[2.2 Технологии, используемые для выполнения целей ВКР 21](#_Toc169687742)

[Инструменты дизайна карт 21](#_Toc169687743)

[Веб-разработка 21](#_Toc169687744)

[Базы данных 22](#_Toc169687745)

[Серверная логика 22](#_Toc169687746)

[2.3 Аналоги (Обзор существующих программных средств) 22](#_Toc169687747)

[Примеры аналогов: 23](#_Toc169687748)

[Глава 3. Описание разработанных программных средств 25](#_Toc169687749)

[3.1 Архитектура приложения 25](#_Toc169687750)

[Клиентская часть 25](#_Toc169687751)

[Серверная часть 26](#_Toc169687752)

[База данных 26](#_Toc169687753)

[Взаимодействие между компонентами 29](#_Toc169687754)

[3.2 Описание кода приложения 30](#_Toc169687755)

[Подключение библиотек и фреймворков 30](#_Toc169687756)

[Описание основного кода 31](#_Toc169687757)

[Глава 4. Отладка, тестирование и результаты разработанного ПО 36](#_Toc169687758)

[Отладка 36](#_Toc169687759)

[Тестирование 36](#_Toc169687760)

[Тестирование функции поиска аудитории из окна браузера 36](#_Toc169687761)

[Тестирование функции переключения этажей 37](#_Toc169687762)

[Тестирование работы мобильной версии 40](#_Toc169687763)

[Глава 5. Руководство пользователя 42](#_Toc169687764)

[5.1 Начало работы 42](#_Toc169687765)

[5.2 Основные функции 42](#_Toc169687766)

[Выбор территории 42](#_Toc169687767)

[Панорамирование и масштабирование 43](#_Toc169687768)

[Поиск объектов 43](#_Toc169687769)

[Переключение этажей 44](#_Toc169687770)

[Получение информации об объекте 44](#_Toc169687771)

[5.3 Работа с мобильными устройствами 45](#_Toc169687772)

[Глава 6. Руководство по установке и сопровождению ПО 46](#_Toc169687773)

[6.1 Требования к разработчику 46](#_Toc169687774)

[6.2 Программные требования 46](#_Toc169687775)

[6.3 Руководство по установке разработанного ПО 47](#_Toc169687776)

[6.4 Защита приложения после установки 47](#_Toc169687777)

[Аутентификация и авторизация 47](#_Toc169687778)

[Защита данных 47](#_Toc169687779)

[Защита от внешних угроз 48](#_Toc169687780)

[Мониторинг и обновления 48](#_Toc169687781)

[6.5 Перенос приложения на другой компьютер/сервер 49](#_Toc169687782)

[6.6 Редактирование карты 49](#_Toc169687783)

[Заключение 52](#_Toc169687784)

[Анализ выполненной работы 52](#_Toc169687785)

[Перспективы развития 53](#_Toc169687786)

[Добавление новых функциональностей: 53](#_Toc169687787)

[Перспективы создания новых возможностей: 53](#_Toc169687788)

[Список использованных источников 55](#_Toc169687789)

[ПРИЛОЖЕНИЕ 57](#_Toc169687790)

Введение

В условиях стремительного развития информационных технологий и глобализации современные университеты сталкиваются с рядом вызовов, связанных с обеспечением эффективного функционирования и взаимодействия всех участников образовательного процесса. Одним из важнейших аспектов, способствующих повышению качества образовательных услуг и комфорта для обучающихся, преподавателей, гостей университета и будущих студентов, является внедрение инновационных решений, облегчающих доступ к информации и навигацию по университету. В этом контексте создание интерактивной карты университета представляется особенно актуальным и перспективным направлением.

Интерактивная карта университета представляет собой современный инструмент, позволяющий пользователям легко и быстро находить нужные здания, аудитории и другие объекты инфраструктуры на территории вуза. Такая карта обеспечивает не только визуализацию пространства, но и предоставляет актуальную информацию о каждом объекте, что существенно упрощает процесс ориентации и планирования перемещений по университету. Кроме того, интерактивная карта способствует удобству составления новых карт, проведению мероприятий и обеспечивает доступ со всех устройств, включая мобильные телефоны и планшеты. Внедрение интерактивной карты позволяет университетам идти в ногу со временем, удовлетворяя растущие потребности своих обучающихся и сотрудников в доступе к информации.

Актуальность данной темы обусловлена несколькими ключевыми факторами. Прежде всего, университеты часто представляют собой сложные архитектурные комплексы с многочисленными зданиями и территориями, навигация по которым может быть затруднена, особенно для новых студентов, гостей и будущих обучающихся. Традиционные статические карты и схемы, размещенные на стендах, не всегда предоставляют достаточный уровень детализации и актуальности информации. Более того, большинство зданий не имеют доступных карт, что создаёт дополнительные сложности. Интерактивная карта предлагает динамический и гибкий подход к представлению данных, что делает процесс поиска нужных объектов быстрым и удобным. Она предоставляет инструмент для создания и обновления карт, что решает проблему отсутствия информации о многих зданиях.

Современные люди активно используют мобильные устройства и цифровые сервисы для получения информации и взаимодействия с окружающим миром. В этом контексте интерактивная карта, доступная через веб-браузер, становится необходимым инструментом

для удовлетворения информационных потребностей обучающихся. Она позволяет быстро находить нужные корпуса, аудитории и получать другую важную информацию.

Интерактивная карта также играет важную роль в повышении эффективности управления университетской инфраструктурой. Администрация вуза получает возможность оперативно обновлять информацию о зданиях и аудиториях, а в будущем — и о территориях. Это обеспечивает высокую актуальность данных и улучшает планирование использования ресурсов университета.

Необходимо отметить, что создание интерактивной карты требует интеграции различных технологий и систем. В частности, необходимо обеспечить взаимодействие между базой данных, в которой хранятся сведения об объектах ВУЗа, и пользовательским интерфейсом, через который осуществляется доступ к этим данным. Это предполагает использование современных веб-технологий, таких как HTML, CSS и JavaScript, а также специализированных библиотек и фреймворков для работы с картами, например, Leaflet. Кроме того, важную роль играет выбор реляционной базы данных, например, PostgreSQL, которая обеспечивает надежное хранение и быстрый доступ к данным.

Интерактивная карта университета представляет собой не только удобный инструмент для пользователей, но и важный элемент цифровой трансформации образовательного учреждения. Она способствует созданию единого информационного пространства, интегрируя различные сервисы и системы вуза. Это позволяет значительно повысить качество взаимодействия всех участников образовательного процесса, улучшить коммуникацию и сделать образовательную среду более прозрачной и доступной.

Следует подчеркнуть, что разработка интерактивной карты университета имеет значительный научный и практический потенциал. С одной стороны, она требует применения современных методов и технологий разработки программного обеспечения, что способствует развитию профессиональных навыков и компетенций студентов. С другой стороны, такой проект имеет прямую практическую значимость, так как его результаты могут быть использованы в реальной жизни для улучшения условий обучения и работы в университете.

Кроме того, разработка интерактивной карты университета является важным шагом в направлении цифровизации образовательной среды. Цифровизация образовательных процессов и инфраструктуры позволяет повысить прозрачность управления, улучшить доступность образовательных ресурсов и создать условия для более эффективного взаимодействия между всеми участниками образовательного процесса. Внедрение интерактивных карт является одним из ключевых элементов этой стратегии, так как они обеспечивают интеграцию различных информационных систем и сервисов, делая их использование более удобным и доступным для всех пользователей.

Также важно отметить, что интерактивные карты могут использоваться не только для навигации и предоставления информации, но и для решения других задач, связанных с управлением университетской инфраструктурой. Например, они могут применяться для анализа потоков студентов и преподавателей, планирования расписания занятий, управления занятостью аудиторий и других ресурсов. Это позволяет более эффективно использовать доступные ресурсы и улучшать качество предоставляемых образовательных услуг.

Интерактивные карты могут стать важным инструментом в организации мероприятий на территории университета. С их помощью можно планировать размещение участников и гостей, организовывать навигацию по мероприятию, а также оперативно предоставлять актуальную информацию о происходящих событиях. Это особенно актуально для крупных университетов, где регулярно проводятся научные конференции, форумы и другие значимые мероприятия.

Создание интерактивной карты также облегчает процессы обновления и внесения изменений. В отличие от статических карт, которые требуют печати и физического размещения, интерактивные карты могут быть обновлены в реальном времени. Это особенно важно в условиях частых изменений инфраструктуры, когда новые здания и аудитории вводятся в эксплуатацию, а старые — выводятся из использования.

Интерактивная карта университета может стать платформой для интеграции различных сервисов и систем. Например, она может быть связана с системой управления учебным процессом, библиотечной системой, системой безопасности и другими ключевыми компонентами информационной инфраструктуры вуза. Это создает единое информационное пространство, в котором пользователи могут получить доступ ко всем необходимым сервисам через один интерфейс.

Таким образом, разработка интерактивной карты университета является важным и актуальным проектом, который отвечает современным требованиям и потребностям образовательных учреждений. Он позволяет не только улучшить навигацию и доступ к информации для обучающихся, преподавателей и гостей, но и способствует более эффективному управлению ресурсами университета. Внедрение таких решений является важным шагом на пути к цифровой трансформации образовательной среды и повышению качества образовательных услуг в целом.

# Глава 1. Постановка задачи и методы её решения

## 1.1 Цель работы

Целью данной выпускной квалификационной работы является разработка интерактивной карты университета, которая будет доступна через веб-браузер и позволит пользователям легко находить необходимые объекты на территории ВУЗа. Это включает в себя создание системы, обеспечивающей визуализацию и управление информацией о зданиях, аудиториях и других объектах инфраструктуры университета.

## 1.2 Техническое задание по выпускной работе

Необходимо разработать приложение для предоставления доступа сотрудников и обучающихся к картам помещений СПбГМТУ.

Данное приложение должно использоваться для навигации внутри зданий университета.

Приложение должно иметь модуль поиска помещений и аудиторий.

Пользователь приложения должен иметь возможность получать доступ к картам с различных устройств.

Помимо этого, необходимо организовать создание таких карт для загрузки их в приложение.

### Функциональные требования к приложению

1. Отображение карты университета:

* Возможность выбора и отображения различных территорий университета (Ульянка, Лоцманская, Горьковская).
* Отображение зданий и инфраструктуры на карте.
* Возможность выделения и взаимодействия с объектами на карте (здания, аудитории).

1. Поиск и навигация:

* Поиск аудитории по номеру.
* Выделение и центрирование карты на выбранной аудитории.
* Отображение всплывающей информации о выбранной аудитории или здании.

1. Интерактивность:

* Возможность навигации по этажам зданий.
* Интерактивное взаимодействие с картой (наведение курсора, клики по зданиям и аудиториям)

### Нефункциональные требования к приложению

1. Производительность:

* Быстрая загрузка карты и данных при выборе территорий и зданий.
* Мгновенная реакция на взаимодействие пользователя (наведение курсора, клики).

1. Совместимость:

* Поддержка различных веб-браузеров (Google Chrome, Mozilla Firefox, Safari).
* Адаптивный дизайн для работы на различных устройствах, включая настольные компьютеры и мобильные телефоны.

1. Удобство использования:

* Интуитивно понятный интерфейс.
* Простота в использовании функций поиска и навигации.

## 1.3 Методы и способы решения разрабатываемого ПО

Для разработки интерактивной карты университета можно использовать несколько глобальных методов, каждый из которых включает в себя специфические подходы и технологии. Основные методы включают использование сторонних картографических сервисов, разработку с нуля с использованием веб-технологий, а также комбинацию обоих подходов. Далее подробно рассмотрим каждый из методов.

### Использование сторонних картографических сервисов

Этот метод подразумевает использование готовых решений для отображения карт и управления географическими данными. Основные сервисы включают Google Maps API, OpenStreetMap и Mapbox. Каждый из этих сервисов предоставляет мощные инструменты для работы с картами и данными.

Google Maps API:

Описание: Google Maps API предоставляет доступ к широкому набору картографических данных и инструментов для их отображения и взаимодействия с ними.

Преимущества: Высокое качество карт, доступ к обширной базе данных, удобные инструменты для работы с маршрутами и геокодированием.

Недостатки: Ограничения по использованию бесплатных квот, высокая стоимость при большом объеме запросов.

OpenStreetMap:

Описание: OpenStreetMap (OSM) — это открытая и бесплатная база данных картографической информации, которую можно использовать для создания собственных карт.

Преимущества: бесплатность, доступ к исходным данным, возможность кастомизации.

Недостатки: требует больше усилий для настройки и интеграции, может не обладать таким же уровнем детализации, как коммерческие сервисы.

Mapbox:

Описание: Mapbox предоставляет инструменты для создания кастомных карт с использованием данных OpenStreetMap и других источников.

Преимущества: Высокая степень кастомизации, мощные инструменты для визуализации данных, интеграция с различными платформами.

Недостатки: Платная подписка для полного доступа ко всем функциям.

### Разработка с нуля с использованием веб-технологий

Этот метод предполагает создание собственной системы отображения карт и управления данными с использованием веб-технологий. Включает использование HTML, CSS, JavaScript, а также библиотек для работы с картами, таких как Leaflet.

Leaflet:

Описание: Leaflet — это легковесная библиотека JavaScript для создания интерактивных карт.

Преимущества: Простота использования, легковесность, высокая производительность, широкие возможности для кастомизации.

Недостатки: требует наличия собственных данных и разработки логики для управления картами и взаимодействия с ними.

Основные шаги разработки с нуля:

* Создание базы данных: Использование PostgreSQL для хранения данных о зданиях и аудиториях.
* Разработка серверной части: Node.js и Express.js для обработки запросов и взаимодействия с базой данных.
* Разработка клиентской части: HTML, CSS, JavaScript для создания интерфейса и взаимодействия с картой.
* Интеграция Leaflet: Использование Leaflet для отображения карты и взаимодействия с объектами.

### Комбинированный метод

Этот метод предполагает использование сторонних сервисов для базовых картографических данных и разработку собственной логики для специфических функций и данных.

Описание метода:

* Базовые карты: Использование OpenStreetMap или Mapbox для предоставления базовых карт.
* Собственные данные: Хранение и управление собственными данными (здания, аудитории) в базе данных PostgreSQL.
* Кастомизация и интерактивность: Разработка логики для взаимодействия с картой и данными с использованием JavaScript и Leaflet.

Преимущества:

* Гибкость: Возможность использования мощных инструментов для базовой карты и полной кастомизации для специфических нужд.
* Контроль над данными: Возможность управлять и обновлять собственные данные.

Недостатки:

* Сложность разработки: требует навыков работы с несколькими технологиями и их интеграции.

Каждый из рассмотренных методов имеет свои преимущества и недостатки, и выбор конкретного подхода зависит от специфических требований проекта, доступных ресурсов и навыков команды разработчиков. Использование сторонних сервисов позволяет быстро получить доступ к качественным картографическим данным, но может ограничивать гибкость и увеличивать стоимость. Разработка с нуля обеспечивает полный контроль и кастомизацию, но требует больше времени и усилий. Комбинированный метод предлагает баланс между использованием мощных готовых инструментов и возможностью кастомизации для удовлетворения специфических нужд проекта.

### Особенности функционирования создаваемого ПО

Создаваемое ПО представляет собой веб-приложение, которое будет доступно через браузер. Основные функции приложения включают отображение интерактивной карты университета, поиск и навигацию по зданиям и аудиториям, а также интерактивное взаимодействие с элементами карты. Приложение будет поддерживать динамическое обновление данных, что позволит пользователям получать актуальную информацию в реальном времени.

### Интерфейс (дизайн) ПО

Интерфейс приложения разрабатывался с учетом современных требований к удобству и эстетике. Основные элементы интерфейса включают:

1. Верхняя панель навигации:

* Кнопки для выбора территорий университета.
* Поле ввода для поиска аудитории и кнопка "Поиск".

1. Основная область карты:

* Интерактивная карта с возможностью зуммирования и панорамирования.
* Выделение зданий и аудиторий при наведении курсора.

### Требования по используемой аппаратуре и вспомогательного ПО

1. Аппаратные требования:

* Рабочее место с доступом в интернет.
* Компьютер или мобильное устройство с современным веб-браузером.

1. Программные требования:

* Веб-браузер с поддержкой HTML5, CSS3 и JavaScript (Google Chrome, Mozilla Firefox, Safari).

1. Сервер для хостинга веб-приложения.

## 1.4 Описание разработки (отрисовки) карт

Для создания визуального представления карты университета и его территорий использовался Adobe Illustrator. Этот инструмент позволяет создавать точные и масштабируемые векторные изображения, что обеспечивает высокое качество графики при различных уровнях масштабирования карты. Процесс создания карт включал следующие этапы:

1. Сбор данных и планировка:

* Сбор данных о территориях и зданиях университета.
* Создание схем и планов территорий на основе собранных данных.

1. Создание векторных изображений:

* Импорт схем и планов в Adobe Illustrator.
* Создание векторных контуров для каждого здания и объекта на карте.
* Добавление детализированной информации и стилизация объектов (цвета, линии, текстуры).

1. Экспорт изображений:

* Экспорт готовых векторных изображений в формате SVG для дальнейшего использования в веб-приложении.

# Глава 2. Обзор существующих программных средств

## 2.1 Анализ основных технологических решений

### Выбор языка программирования



Таблица 1. Сравнение языков программирования

Для реализации интерактивной карты университета были рассмотрены различные языки программирования, такие как JavaScript, Python, Java и C#. Основным критерием выбора стал баланс между простотой использования, производительностью и возможностями интеграции с другими технологиями. В конечном итоге был выбран JavaScript по следующим причинам:

* Широкая распространенность и поддержка: JavaScript является стандартом для веб-разработки и поддерживается всеми современными браузерами.
* Мощная экосистема: наличие большого количества библиотек и фреймворков, таких как Leaflet, D3.js и других.
* Асинхронность: поддержка асинхронного программирования позволяет эффективно обрабатывать пользовательские запросы и обновлять интерфейс без задержек.

### Выбор среды разработки

Для разработки веб-приложения была выбрана среда разработки Visual Studio Code (VS Code). Это решение было обусловлено следующими факторами:

* Легкость и быстрота: VS Code является легким редактором кода, который быстро загружается и работает даже на менее мощных компьютерах.
* Расширяемость: наличие множества расширений, таких как интеграция с Git, поддержка JavaScript, HTML, CSS, и другие полезные инструменты.
* Кроссплатформенность: поддержка всех основных операционных систем (Windows, macOS, Linux), что обеспечивает гибкость в разработке.

### Выбор библиотек и фреймворков

Для реализации интерактивной карты необходимо было выбрать библиотеку. Я рассматривал четыре варианта, среди которых были Google Maps API, OpenStreetMap, MapBox, Leaflet. Мной были выбраны критерии, по которым я сравнивал эти библиотеки

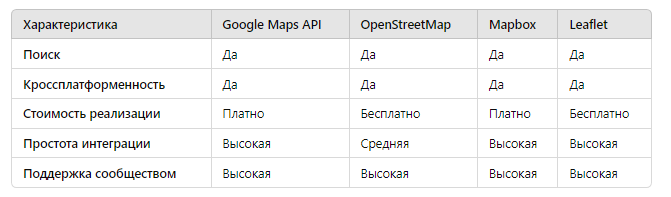


Таблица 2 Сравнение библиотек

Выбор Leaflet

Leaflet был выбран по следующим причинам:

* Бесплатность: Leaflet является бесплатной библиотекой с открытым исходным кодом.
* Легкость и производительность: Leaflet быстро загружается и работает, что делает его идеальным для веб-приложений.
* Простота интеграции: Leaflet имеет интуитивно понятный API, что облегчает разработку.
* Широкая поддержка: наличие большого сообщества и множества плагинов для расширения функциональности.

Таким образом, выбор Leaflet оказался наиболее оптимальным для реализации проекта интерактивной карты университета. Он сочетает в себе высокую производительность, простоту использования и богатый функционал, что позволяет создавать мощные и гибкие карты с минимальными затратами.

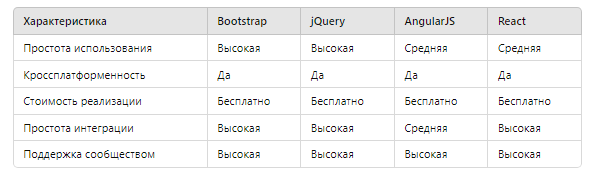


Таблица 3. Сравнение фреймворков

Bootstrap: фреймворк для создания адаптивных и стильных пользовательских интерфейсов. Bootstrap был выбран за его простоту в использовании и наличие готовых компонентов, таких как кнопки, формы и модальные окна, что ускоряет процесс разработки, что ускоряет процесс разработки. Поддержка адаптивного дизайна, что позволяет приложению корректно отображаться на различных устройствах.

jQuery: библиотека JavaScript, которая упрощает работу с DOM, обработку событий и создание анимаций. Она была выбрана для улучшения взаимодействия пользователя с интерфейсом.

* Удобство работы с DOM: простота в использовании и возможность легко манипулировать элементами на странице.
* Широкая поддержка: наличие множества плагинов и большой базы знаний, что облегчает разработку.

Почему не выбраны другие фреймворки и библиотеки:

Vue.js - отличная альтернатива для создания динамических интерфейсов, но для данного проекта может быть избыточным.

React - мощный инструмент для создания сложных интерфейсов, но для текущего проекта его использование может быть слишком сложным.

### Выбор редактора SVG



Таблица 4. Сравнение SVG редакторов

Для создания и редактирования векторных изображений карт использовался Adobe Illustrator. Этот инструмент был выбран по следующим причинам:

* Точность и детализация: возможность создания детализированных и точных векторных изображений, которые можно масштабировать без потери качества.
* Широкий набор инструментов: наличие множества функций для рисования, стилизации и редактирования векторной графики.
* Поддержка форматов: возможность экспорта изображений в различные форматы, включая SVG, что позволяет легко интегрировать их в веб-приложение.

### Выбор языка для серверной части

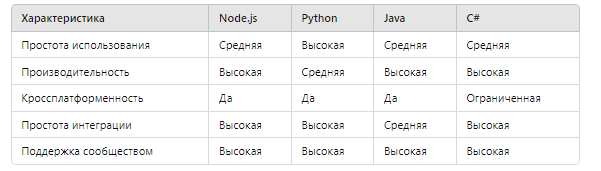


Таблица 5. Сравнение языков программирования для серверной части

Почему выбран: Node.js и Express.js обеспечивают высокую производительность и позволяют использовать JavaScript как на клиентской, так и на серверной стороне, что упрощает разработку и поддержку приложения.

## 2.2 Технологии, используемые для выполнения целей ВКР

### Инструменты дизайна карт

Adobe Illustrator – это графический редактор, является ведущим программным обеспечением для создания и редактирования векторной графики, включая SVG. Он предоставляет широкий спектр инструментов для создания высококачественных визуальных элементов, необходимых для интерактивной карты.

Официальный сайт: <https://www.adobe.com/products/illustrator.html>

### Веб-разработка

HTML - (HyperText Markup Language) является основным языком разметки, используемым для создания структуры веб-страниц. В данном проекте HTML используется для создания элементов интерфейса, таких как кнопки, поля ввода, контейнеры для карт и модальные окна.

Официальный сайт: <https://html.com/html5/>

CSS - (Cascading Style Sheets) используется для стилизации элементов веб-страницы. В данном проекте CSS используется для настройки внешнего вида карты, кнопок, полей ввода и других элементов интерфейса.

Официальный сайт: <https://www.w3.org/Style/CSS/>

JavaScript - является основным языком программирования для создания интерактивных элементов на веб-странице. В данном проекте JavaScript используется для управления картой, обработки событий, взаимодействия с пользователем и динамического обновления данных.

Официальный сайт: <https://www.javascript.com/>

Leaflet — это современная, легковесная и открытая библиотека JavaScript для создания интерактивных карт. Она поддерживает различные слои карт, интерактивные элементы и плагинную архитектуру, что делает её отличным выбором для создания карты вуза.

Официальный сайт с документацией: <https://leafletjs.com/>

Bootstrap – это популярный CSS-фреймворк для создания адаптивных и стильных пользовательских интерфейсов.

Официальный сайт: <https://getbootstrap.com/>

jQuery — это библиотека JavaScript, упрощающая работу с DOM, обработку событий и создание анимаций.

Официальный сайт: <https://jquery.com/>

Visual Studio Code (VS Code) — это мощная, бесплатная и настраиваемая среда разработки, поддерживающая множество расширений для различных языков программирования и инструментов. Она предоставляет инструменты для отладки, интеграции с системами контроля версий и редакторами кода.

Официальный сайт: <https://code.visualstudio.com/>

### Базы данных

PostgreSQL — это мощная, открытая и масштабируемая реляционная СУБД, которая поддерживает сложные запросы и расширенные функции, такие как пространственные данные через расширение PostGIS. Она идеально подходит для хранения и обработки данных, связанных с интерактивной картой вуза.

Официальный сайт: <https://www.postgresql.org/>

### Серверная логика

Node.js — это платформа для выполнения JavaScript на серверной стороне, обеспечивающая высокую производительность и масштабируемость благодаря неблокирующей модели ввода-вывода. Она идеально подходит для создания серверной части веб-приложений.

Официальный сайт: <https://nodejs.org/en>

Express.js — это минималистичный и гибкий фреймворк для Node.js, который упрощает создание серверных приложений и API. Он предоставляет мощные инструменты для маршрутизации и обработки запросов, что делает его идеальным выбором для реализации серверной логики.

Официальный сайт: <https://expressjs.com/>

## 2.3 Аналоги (Обзор существующих программных средств)

В данной под главе рассматриваются существующие программные средства, аналогичные разработанной интерактивной карте университета. Анализируются их возможности, преимущества и недостатки с целью выявления лучших практик и потенциальных областей для улучшения.

В современном образовательном процессе эффективная навигация и доступ к информации играют ключевую роль. Университеты, стремясь обеспечить комфорт и удобство для своих студентов, преподавателей и гостей, внедряют различные технологические решения, такие как интерактивные карты университета. Эти карты позволяют пользователям легко находить необходимые объекты, такие как здания, аудитории и другие важные места на территории университета. Для более глубокого понимания существующих решений и их сравнительного анализа, были рассмотрены интерактивные карты ведущих российских вузов. В этом разделе представлены анализ и сравнение интерактивных карт СПбГУ, НИУ ВШЭ и СПбГАСУ. Изучение этих аналогов позволяет выявить сильные и слабые стороны каждого решения, а также определить области для возможных улучшений в разработанном проекте.

### Примеры аналогов:

Интерактивная карта СПбГУ

Описание: Санкт-Петербургский государственный университет (СПбГУ) предоставляет интерактивную карту, которая отображает здания и аудитории университета. Карта позволяет пользователям легко находить нужные объекты и получать информацию о них.

Преимущества:

* Удобный интерфейс и навигация.
* Высокая детализация карт.
* Возможность поиска и фильтрации объектов.

Недостатки:

* Ограниченная функциональность по сравнению с коммерческими решениями.
* Возможные задержки в обновлении данных.

Интерактивная карта НИУ ВШЭ

Описание: Национальный исследовательский университет "Высшая школа экономики" (НИУ ВШЭ) предоставляет интерактивную карту ВУЗа, позволяющую пользователям находить здания, аудитории и другие объекты. Карта доступна через веб-браузер и обеспечивает высокую детализацию и актуальность данных.

Преимущества:

* Высокая детализация и актуальность данных.
* Удобный и интуитивно понятный интерфейс.
* Поддержка поиска и фильтрации объектов.

Недостатки:

* Ограниченная функциональность по сравнению с коммерческими решениями.
* Зависимость от внутренней инфраструктуры университета для обновления данных.

Интерактивная карта СПбГАСУ

Описание: Санкт-Петербургский государственный архитектурно-строительный университет (СПбГАСУ) предоставляет интерактивную карту, которая помогает пользователям находить здания и аудитории университета. Карта доступна через веб-браузер и обеспечивает удобную навигацию для студентов и преподавателей.

Преимущества:

* Высокая детализация карт.
* Удобный и понятный интерфейс.
* Поддержка поиска объектов.

Недостатки:

* Ограниченная интерактивность по сравнению с коммерческими решениями.
* Возможные задержки в обновлении данных.



Таблица 6. Таблица сравнения аналогов

Анализ существующих программных средств показал, что университетские интерактивные карты, такие как карты СПбГУ, НИУ ВШЭ и СПбГАСУ, предоставляют высокую детализацию и удобство использования. Однако они имеют некоторые ограничения. Университетские карты обеспечивают высокую актуальность данных и удобный интерфейс, что делает их полезными для студентов, преподавателей и гостей. Тем не менее, они зависят от внутренней инфраструктуры университета для обновления данных и имеют ограниченную функциональность в плане интерактивности и кастомизации.

# Глава 3. Описание разработанных программных средств

## 3.1 Архитектура приложения

Архитектура приложения интерактивной карты университета представляет собой структурированное взаимодействие различных компонентов, обеспечивающих функциональность и производительность системы. В данной главе подробно рассмотрим ключевые аспекты архитектуры, включая клиентскую часть, серверную часть и базу данных.

Архитектура приложения состоит из следующих основных компонентов:

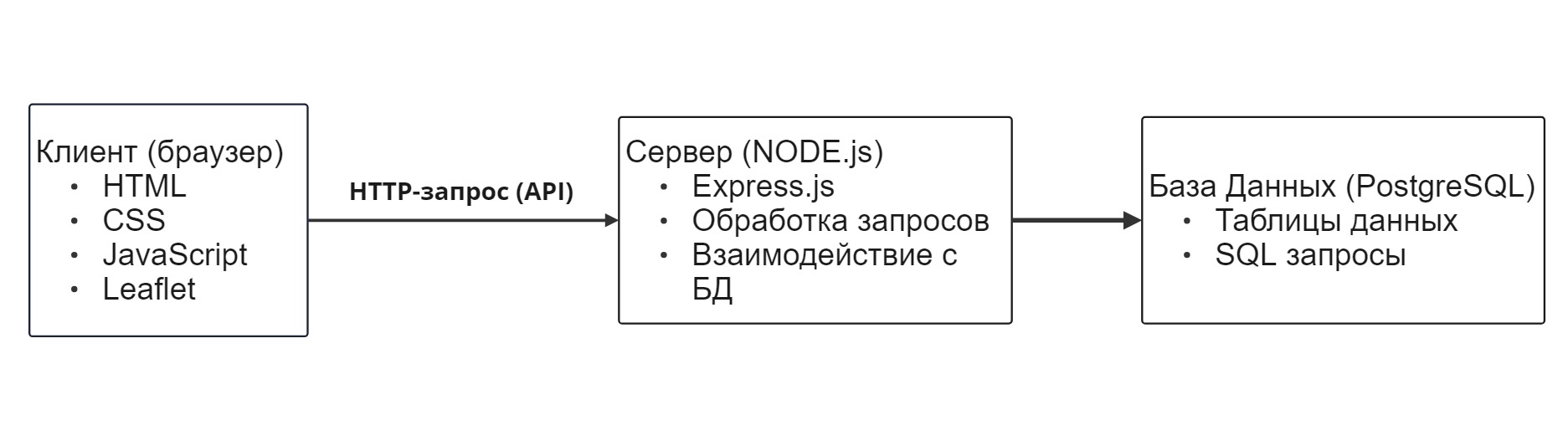


Рисунок 1. Общая схема взаимодействия компонентов

### Клиентская часть

Клиентская часть приложения отвечает за взаимодействие с пользователем и визуализацию данных. Основные компоненты клиентской части:

HTML:

Структура веб-страниц, включая элементы интерфейса, такие как поля поиска, кнопки переключения этажей и контейнер для карты.

CSS:

Оформление веб-страниц, обеспечивающее адаптивный дизайн и удобство использования на различных устройствах.

JavaScript:

* Логика взаимодействия с пользователем и обработка событий.
* Использование библиотеки Leaflet для работы с картами.

Скрипты для реализации поиска аудитории, переключения этажей и других интерактивных функций.

### Серверная часть

Серверная часть приложения обеспечивает обработку запросов от клиентской части и взаимодействие с базой данных. Основные компоненты серверной части:

Node.js:

* Среда выполнения JavaScript на сервере.
* Обеспечение асинхронной обработки запросов для высокой производительности.

Express.js:

* Минималистичный фреймворк для создания веб-приложений на Node.js.
* Маршрутизация запросов и обработка HTTP-запросов от клиента.

API:

* Реализация REST API для взаимодействия между клиентской и серверной частями.
* Эндпоинты для получения информации об аудиториях, зданиях и других объектах.

### База данных

База данных в разработанном приложении интерактивной карты университета используется исключительно для хранения данных, необходимых для поиска аудиторий и зданий. В этом разделе будет подробно описана структура базы данных, включая используемые таблицы и их поля, а также методы взаимодействия с базой данных для выполнения поисковых запросов.

Структура базы данных:

База данных PostgreSQL используется для хранения информации о зданиях и аудиториях университета. Основные таблицы в базе данных включают buildings и auditoriums.

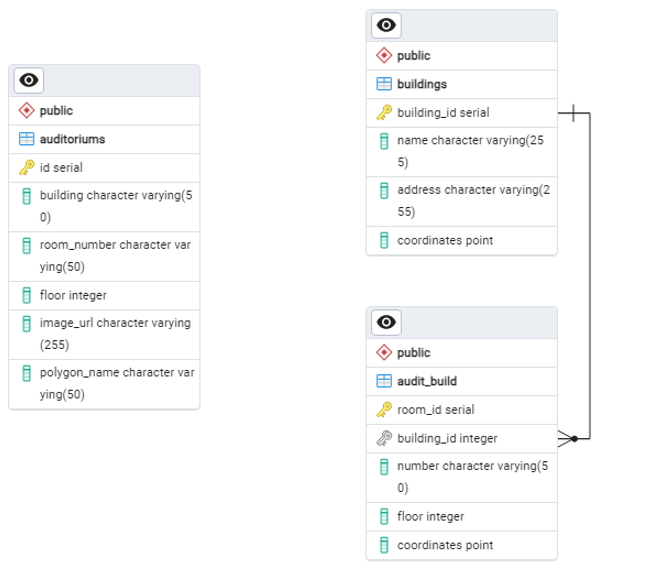


Рисунок 2. Схема базы данных

Таблица buildings:

Таблица buildings хранит информацию о зданиях университета.

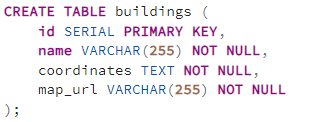


Рисунок 3. SQL запрос для создания таблицы buildings

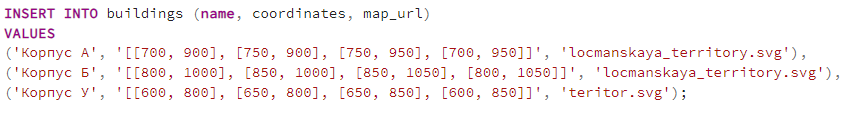


Рисунок 4. SQL запрос для заполнения таблицы buildings

Таблица audit\_build

Таблица audit\_build хранит информацию об аудиториях в зданиях университета.

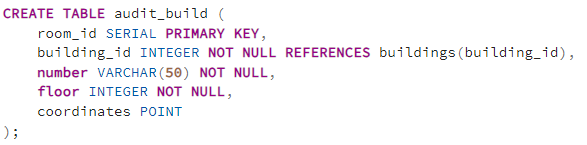


Рисунок 5. SQL запрос для создания таблицы audit\_build

Таблица auditoriums

Таблица auditoriums хранит дополнительную информацию о аудиториях.

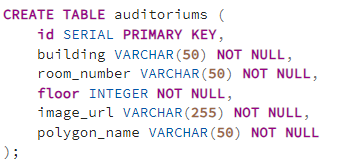


Рисунок 6. SQL запрос для создания таблицы auditoriums

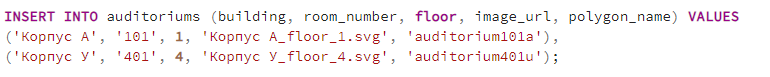


Рисунок 7.. SQL запрос для заполнения таблицы auditoriums

Связи между таблицами:

Таблица audit\_build имеет внешний ключ building\_id, который ссылается на первичный ключ building\_id в таблице buildings.

Таблица auditoriums имеет внешний ключ room\_id, который ссылается на первичный ключ room\_id в таблице audit\_build.

Взаимодействие с базой данных

Для выполнения поисковых запросов и получения информации из базы данных, включающей все три таблицы, используются SQL-запросы. Ниже приведены примеры таких запросов.

Поиск аудитории по номеру с дополнительной информацией:

Для поиска аудитории по номеру и получения дополнительной информации используется следующий SQL-запрос:

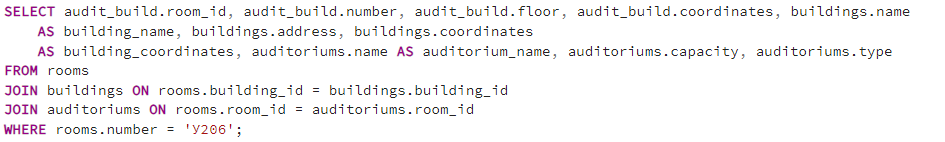


Рисунок 8. SQL запрос для поиска объекта

Этот запрос объединяет таблицы audit\_build, buildings и auditoriums по соответствующим ключам и возвращает информацию об аудитории с указанным номером, включая название и адрес здания, а также дополнительную информацию об аудитории.

База данных PostgreSQL в данном приложении используется для эффективного хранения и поиска информации о зданиях и аудиториях университета. Добавление таблицы auditoriums позволяет хранить дополнительную информацию об аудиториях, что делает поиск более детализированным и полезным для пользователей. Структура базы данных и используемые SQL-запросы обеспечивают высокую производительность и точность поиска, предоставляя пользователям возможность быстро находить необходимые объекты на территории университета.

### Взаимодействие между компонентами

Архитектура приложения обеспечивает эффективное взаимодействие между компонентами:

Клиент-серверное взаимодействие:

Клиентская часть отправляет HTTP-запросы к серверной части для получения данных о зданиях и аудиториях.

Серверная часть обрабатывает запросы, взаимодействует с базой данных и возвращает данные клиентской части в формате JSON.

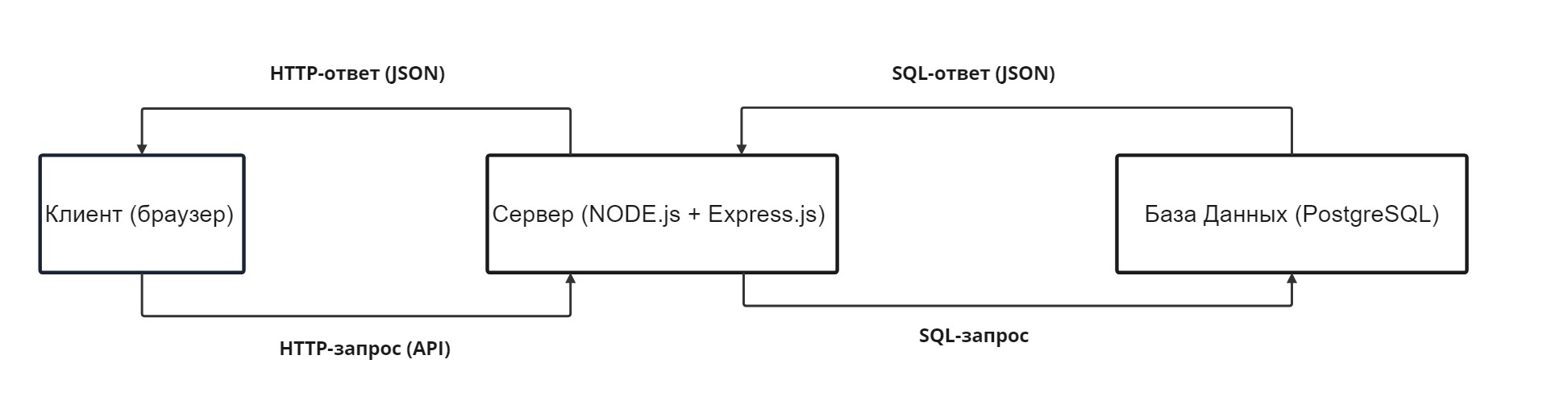


Рисунок 9. Схема взаимодействия

## 3.2 Описание кода приложения

### Подключение библиотек и фреймворков

В начале HTML-файла происходит подключение необходимых библиотек и фреймворков:

<link rel="stylesheet" href="leaflet/leaflet.css" />

<linkrel="stylesheet"href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

<link rel="stylesheet" href="css/map.css"/>

<link rel="stylesheet" href="leaflet-search-master/dist/leaflet-search.min.css">

<script src="leaflet/leaflet.js"></script>

<script src="leaflet-search-master/dist/leaflet-search.min.js"></script>

<script src="JS/script.js"></script> <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<scriptsrc="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>

<scriptsrc="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

leaflet.css и leaflet.js: подключают библиотеку Leaflet для работы с интерактивной картой.

bootstrap.min.css и bootstrap.min.js: подключают фреймворк Bootstrap для стилизации и создания адаптивного интерфейса.

jquery-3.5.1.slim.min.js: подключает библиотеку jQuery для работы с DOM и создания интерактивных элементов.

### Описание основного кода

1. Инициализация карты и переменные

document.addEventListener('DOMContentLoaded', function() {

    var map = L.map('map', {

        crs: L.CRS.Simple,

        minZoom: -1,

        maxZoom: 1,

        center: [730, 960],

        zoom: -1,

    });

const bounds = L.latLngBounds([[0, 0], [1452, 1920]]);

    const territories = {

        'Ульянка': {

            svg: 'teritorU.svg',

            buildings: ['Корпус У', 'МедЦентр', 'Конгресс - центр', 'Спорткомплекс', 'Общежитие1', 'Общежитие2', 'Общежитие3']

        },

        'Лоцманская': {

            svg: 'locmanskaya\_territory.svg',

            buildings: ['Корпус А', 'Корпус Б']

        },

        'Горьковская': {

            svg: 'figa.svg',

            buildings: []

        }

    };

let currentTerritoryLayer = null;

    let currentFloorMap = null;

    let allPolygons = L.layerGroup().addTo(map);

Описание: Этот код инициализирует карту с использованием библиотеки Leaflet. Задаются параметры карты, включая систему координат, границы и центр. Также определяется объект territories, содержащий SVG-файлы и здания для каждой территории.

1. Функции загрузки карт

function loadInitialMap(svgUrl) {

        clearMap();

        currentTerritoryLayer = L.imageOverlay(svgUrl, bounds).addTo(map);

    }

    window.goBack = function() {

        restoreState();

    };

    window.selectTerritory = function(territory) {

        saveState();

        loadTerritorySvg(territories[territory].svg, territories[territory].buildings);

    };

    function loadTerritorySvg(svgUrl, buildings) {

        clearMap();

        currentTerritoryLayer = L.imageOverlay(svgUrl, bounds).addTo(map);

        if (buildings) {

            buildings.forEach(building => {

                addBuildingPolygon(building);

            });

        }

Описание: Эти функции управляют загрузкой начальной карты и изменением карты при выборе территории. Функция loadInitialMap загружает начальную карту, selectTerritory выбирает территорию и загружает соответствующий SVG-файл.

1. Добавление и взаимодействие с полигонами зданий

function addBuildingPolygon(building) {

let polygonCoords;

// Пример для здания 'Корпус У'

if (building === 'Корпус У') {

polygonCoords = [

[-349.77356 + 1080.5, 687.91819 - 109.4],

// остальные координаты

];

}

const polygon = L.polygon(polygonCoords, {

color: 'blue',

fillOpacity: 1,

stroke: false

});

polygon.options.name = building;

addInteractivePolygon(polygon);

allPolygons.addLayer(polygon);

}

function addInteractivePolygon(polygon) {

polygon.on('mouseover', function() {

this.setStyle({ fillOpacity: 0.1, fillColor: 'blue' });

});

polygon.on('mouseout', function() {

this.setStyle({ fillOpacity: 0, fillColor: 'blue' });

});

polygon.on('click', function() {

showFloorModal(this.options.name);

});

Описание: Эти функции добавляют полигоны зданий на карту и определяют взаимодействие с ними. addBuildingPolygon добавляет полигон для каждого здания, а addInteractivePolygon добавляет интерактивное поведение, такое как изменение цвета при наведении и открытие модального окна при клике.

1. Поиск аудитории и управление элементами карты

function highlightAuditorium(auditorium) {

        const foundPolygon = allPolygons.getLayers().find(polygon => polygon.options.name === auditorium);

        if (foundPolygon) {

            map.fitBounds(foundPolygon.getBounds(), {

                padding: [50, 50],

                animate: true,

                duration: 100.0

            });

            foundPolygon.setStyle({ color: 'red' });

        }

    }

    window.searchAuditorium = function() {

        const searchBox = document.getElementById('searchBox');

        const query = searchBox.value.trim();

        if (query) {

            highlightAuditorium(query);

        }

    };

    const searchControl = new L.Control.Search({

        layer: allPolygons,

        propertyName: 'name',

        sourceData: function(text, callResponse) {

            const results = {};

            auditoriums.forEach(function(auditorium) {

                if (auditorium.title.toLowerCase().includes(text.toLowerCase())) {

                    results[auditorium.title] = auditorium.loc;

                }

            });

            callResponse(results);

Описание: Эти функции обеспечивают функциональность поиска аудитории. highlightAuditorium выделяет найденную аудиторию на карте, а searchAuditorium обрабатывает ввод пользователя для поиска аудитории. searchControl добавляет элемент управления поиском на карту и обрабатывает результаты поиска.

# Глава 4. Отладка, тестирование и результаты разработанного ПО

В ходе разработки интерактивной карты университета особое внимание уделяется качеству и функциональности продукта. Одним из ключевых этапов является ручное тестирование, которое позволяет выявить возможные ошибки и недочеты, а также убедиться в удобстве и надежности использования карты. Рассмотрим процесс тестирования основных функций интерактивной карты: поиска аудитории, переключения этажей и работы мобильной версии.

## Отладка

Отладка является неотъемлемой частью разработки программного обеспечения. В ходе отладки были выявлены и исправлены ошибки в коде, а также улучшена производительность и стабильность приложения. Для отладки использовались встроенные инструменты разработчика в браузерах (например, Chrome DevTools), а также специализированные программы и библиотеки для мониторинга и анализа.

Основные этапы отладки:

* Выявление ошибок:
* Анализ логов и сообщений об ошибках.
* Использование отладочных инструментов для пошагового выполнения кода.

Исправление ошибок:

* Коррекция неправильных алгоритмов и логики.
* Оптимизация кода для повышения производительности.
* Проверка исправлений:
* Повторное выполнение сценариев, которые вызывали ошибки, для подтверждения их устранения.

## Тестирование

Тестирование разработанного программного обеспечения проводилось в несколько этапов с целью обеспечения его корректной работы и соответствия требованиям. Основное внимание уделялось тестированию ключевых функций: поиска аудитории, переключения этажей и работы мобильной версии.

### Тестирование функции поиска аудитории из окна браузера

Подготовка:

* Запустить веб-браузер (Google Chrome, Mozilla Firefox, Safari и т.д.).
* Открыть интерактивную карту университета.

Проверка поиска существующей аудитории:

* Ввести номер аудитории (например, "У433") в поле поиска.
* Нажать кнопку "Поиск" или нажать клавишу Enter.
* Проверить, что карта центрируется на выбранной аудитории, и появляется всплывающее окно с информацией об этой аудитории.

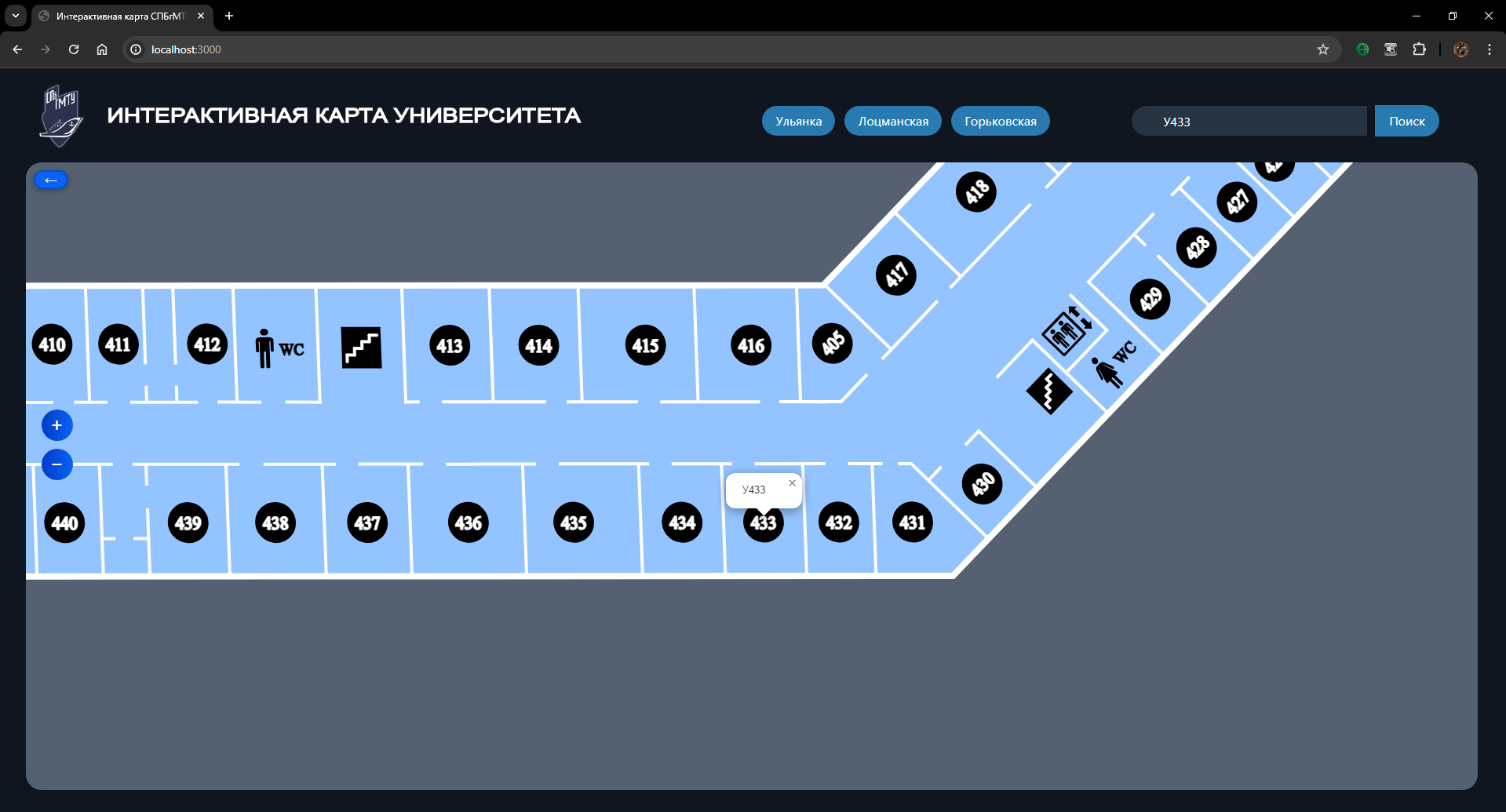


Рисунок 10. Тестирование функции поиска

Результаты тестирования функции поиска аудитории:

Функция поиска успешно находит аудитории по введенному номеру.

Отображение информации о найденной аудитории происходит корректно.

### Тестирование функции переключения этажей

Подготовка:

* Запустить веб-браузер.
* Открыть интерактивную карту университета.
* Выбрать здание с несколькими этажами.

Проверка переключения этажей:

* Найти элементы управления переключением этажей (например, выпадающее меню или кнопки "Этаж 2", "Этаж 3").
* Переключиться на другой этаж (например, с "Этаж 2" на "Этаж 3").
* Проверить, что отображается карта другого этажа выбранного здания, и все объекты (аудитории, помещения) на новом этаже видны.

Проверка сохранения состояния карты:

* Выбрать аудиторию на одном этаже и переключиться на другой этаж.
* Вернуться на предыдущий этаж и проверить, что выбранная ранее аудитория остается выделенной.

Проверка функциональности элементов управления:

* Переключаться между этажами несколько раз.
* Проверить, что карта корректно обновляется каждый раз без задержек и ошибок.

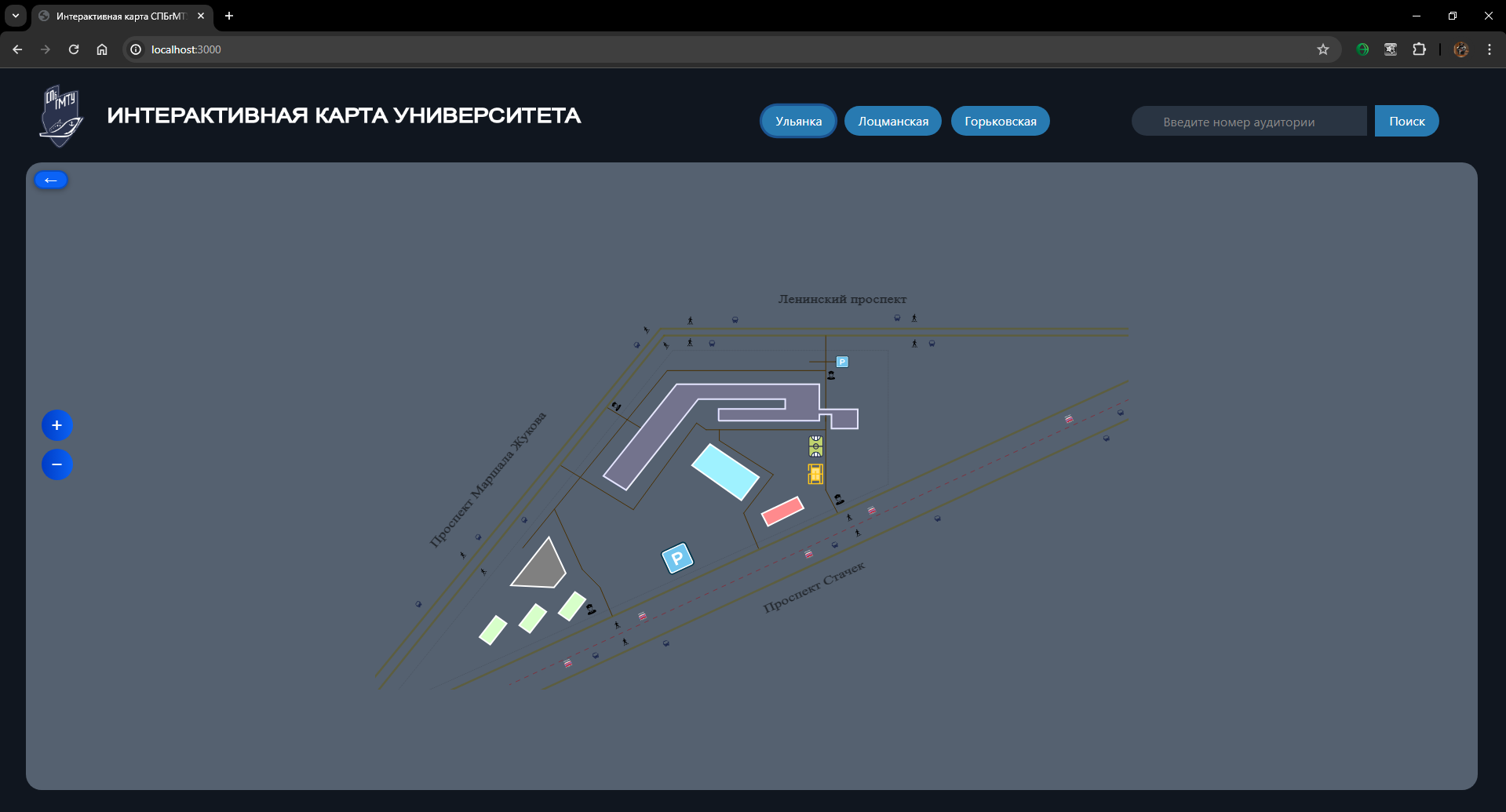


Рисунок 11. Тестирование функции переключения этажей

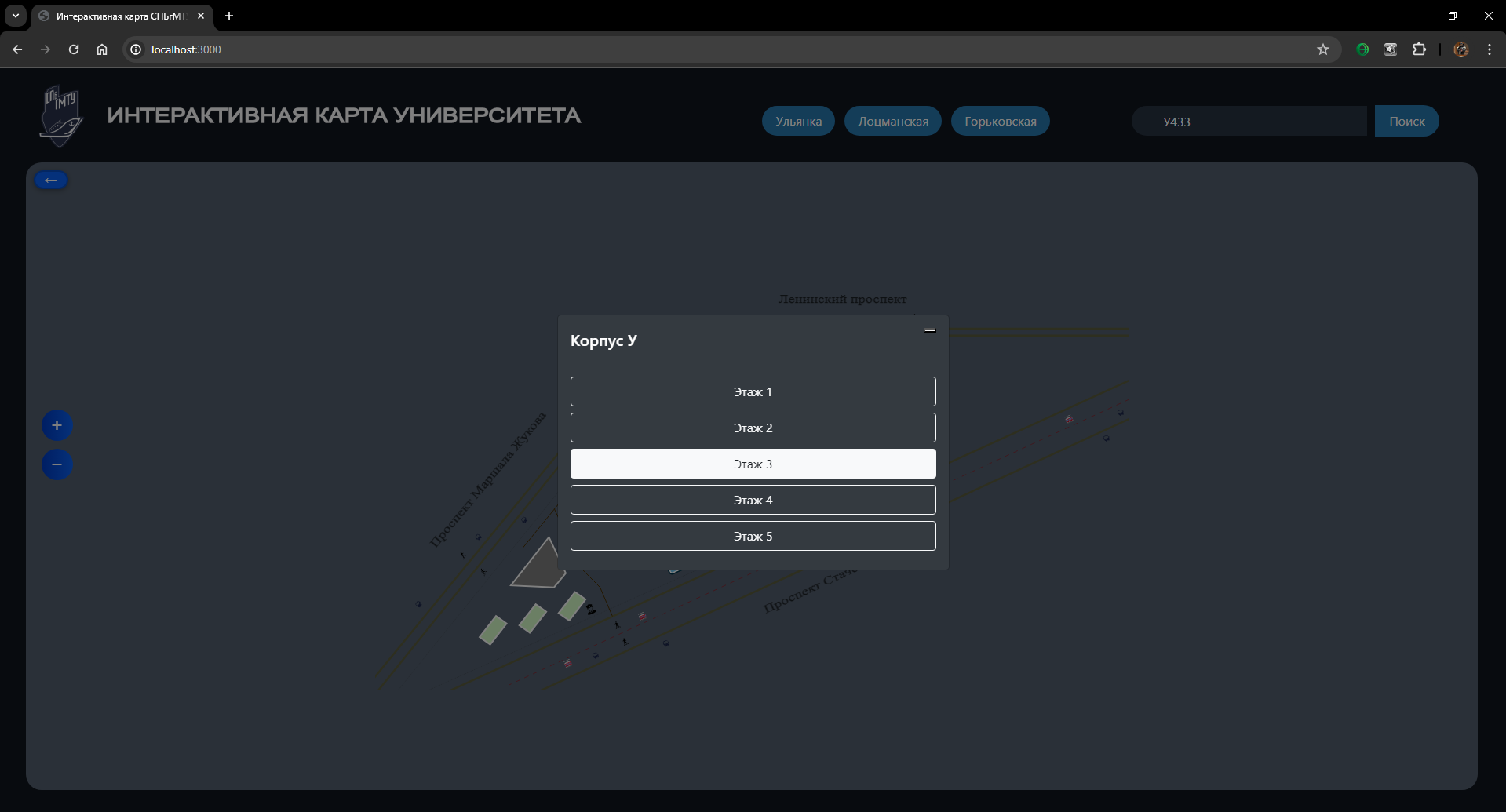


Рисунок 12. Тестирование функции переключения этажей (2)

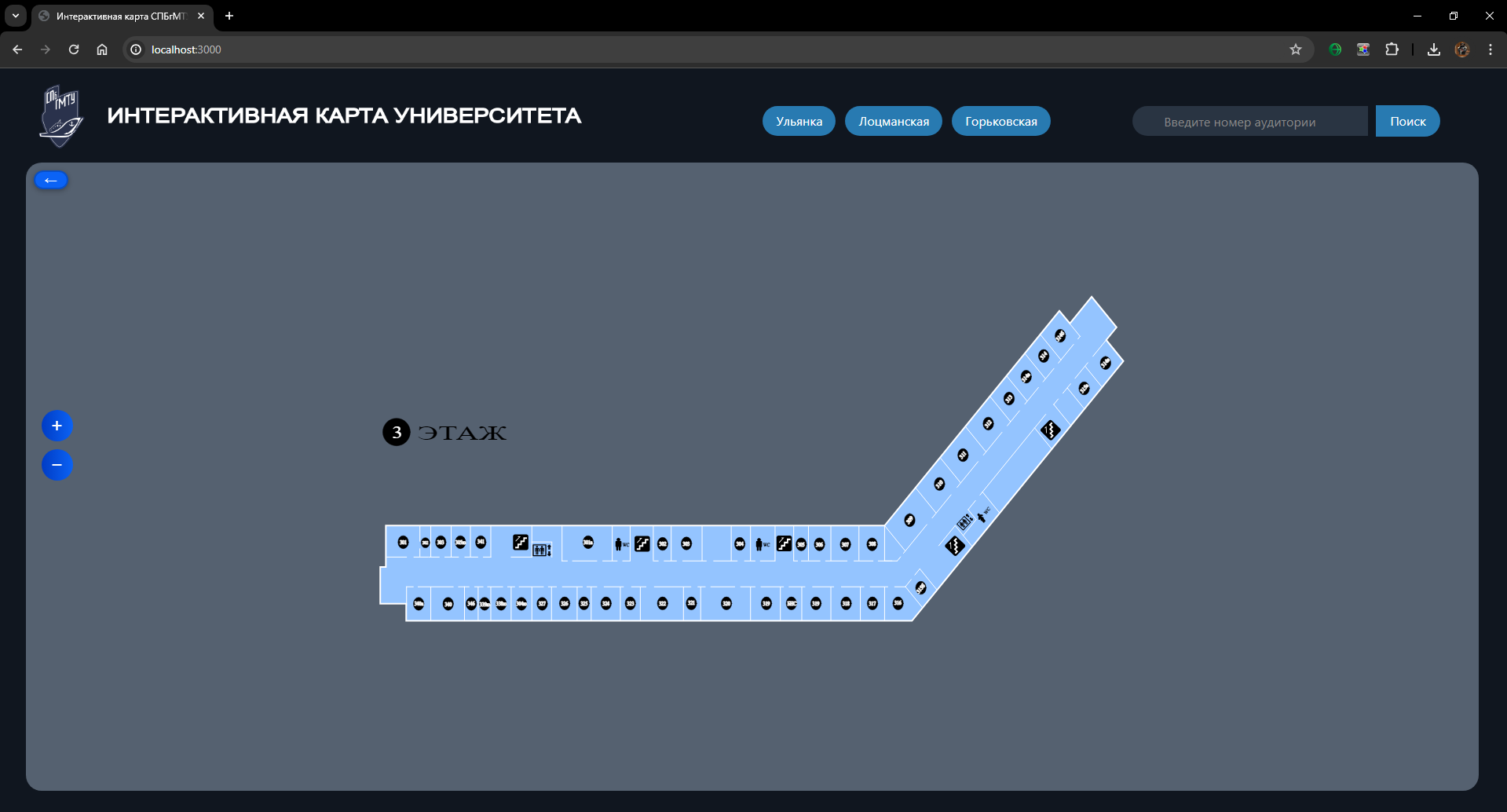


Рисунок 13. Тестирование функции переключения этажей (3)

Результаты тестирования функции переключения этажей:

* Переключение этажей происходит корректно, без задержек и ошибок.
* Все объекты на этажах отображаются правильно и соответствуют реальным данным.
* Система сохраняет состояние карты при переключении между этажами.

### Тестирование работы мобильной версии

Подготовка:

* Взять мобильное устройство (смартфон, планшет).
* Открыть веб-браузер на мобильном устройстве.
* Перейти на интерактивную карту университета.

Проверка адаптивного дизайна:

Проверить, что все элементы интерфейса (поиск, переключение этажей, информация об аудиториях) корректно отображаются и масштабируются на экране мобильного устройства.

* Проверить, что нет горизонтальной прокрутки.
* Проверка поиска аудитории:
* Ввести номер аудитории в поле поиска и выполнить поиск.
* Проверить, что карта центрируется на выбранной аудитории, и информация об аудитории отображается корректно.

Проверка переключения этажей:

* Переключиться между этажами здания и проверить, что карта обновляется корректно.

Проверка сенсорного управления:

* Проверить, что можно увеличивать и уменьшать масштаб карты с помощью жестов (щипок, разведение пальцев).
* Проверить, что можно перемещать карту путем свайпов.
* Проверить, что клики по аудиториям и объектам на карте работают корректно.

Проверка производительности:

* Проверить, что приложение работает плавно без задержек и зависаний.
* Проверить время загрузки карты и данных при использовании мобильного интернета.

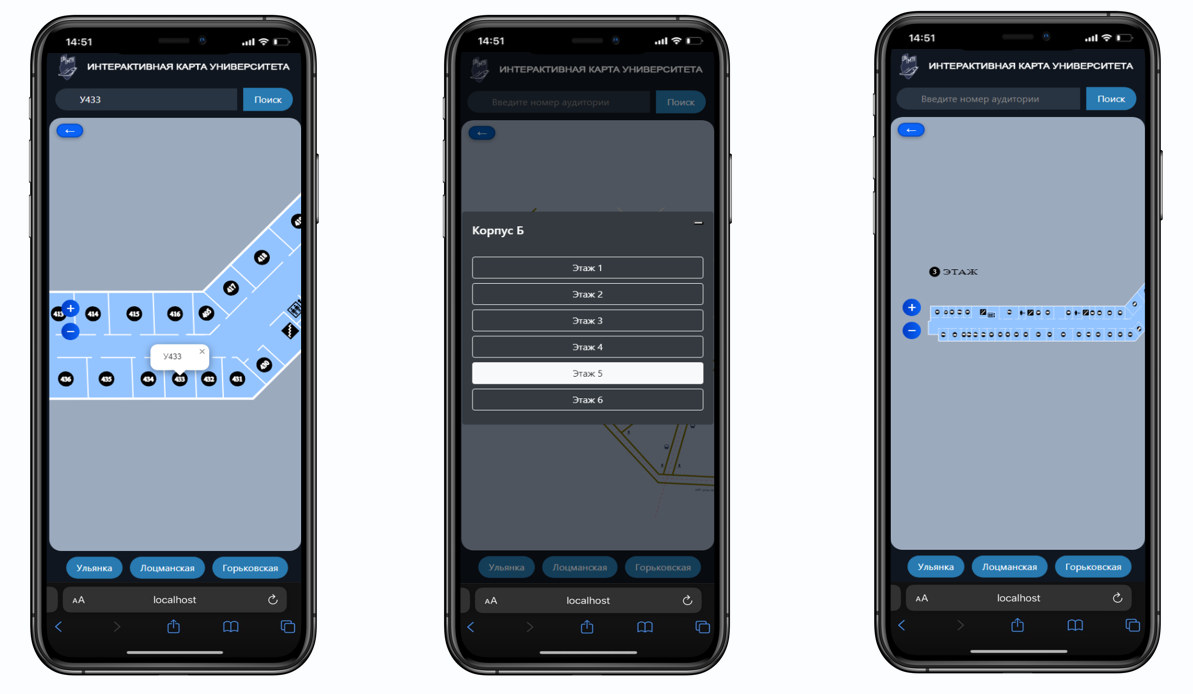


Рисунок 14. Тестирование мобильной версии

Результаты тестирования работы мобильной версии:

* Все элементы интерфейса корректно отображаются на экранах мобильных устройств.
* Функции поиска и переключения этажей работают корректно на мобильных устройствах.
* Сенсорное управление (увеличение/уменьшение масштаба, перемещение карты, клики по объектам) работает плавно и без задержек.
* Приложение работает стабильно и быстро загружает данные при использовании мобильного интернета.

Проведение ручного тестирования интерактивной карты университета позволило убедиться в корректности и надежности работы основных функций. Функция поиска аудитории, переключение этажей и мобильная версия прошли тестирование без выявления критических ошибок. Это подтверждает высокое качество разработанного продукта и его готовность к использованию пользователями. Благодаря проведенному тестированию, мы уверены, что интерактивная карта университета будет полезным и удобным инструментом для обучающихся, преподавателей и гостей, обеспечивая легкий доступ к информации и навигацию по университету.

# Глава 5. Руководство пользователя

Интерактивная карта университета предназначена для упрощения навигации по университету. Она позволяет пользователям находить здания, аудитории и другие объекты инфраструктуры, а также предоставляет дополнительную информацию об этих объектах. В этом разделе описаны основные функции и возможности интерактивной карты, а также инструкции по её использованию.

## 5.1 Начало работы

Открытие карты

Для доступа к интерактивной карте университета выполните следующие действия:

* Откройте веб-браузер (Google Chrome, Mozilla Firefox, Safari или любой другой современный браузер).
* Перейдите по ссылке на интерактивную карту университета
* Дождитесь полной загрузки страницы, чтобы все элементы интерфейса карты были доступны.

## 5.2 Основные функции

Отображение карты

После загрузки страницы вы увидите. интерактивную карту университета. Далее нужно будет выбрать территорию или перейти к поиску. Карта отображает территорию университета, включая здания и другие объекты инфраструктуры. Вы можете перемещаться по карте, увеличивать и уменьшать масштаб для более детального просмотра.

### Выбор территории

* В верхней части страницы расположены кнопки выбора территории: "Ульянка", "Лоцманская", "Горьковская".
* Для выбора территории нажмите на соответствующую кнопку. Карта автоматически обновится и покажет выбранную территорию.

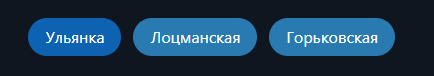


Рисунок 15. Кнопки выбора территории

### Панорамирование и масштабирование

* Панорамирование: для перемещения по карте нажмите и удерживайте левую кнопку мыши, затем перемещайте курсор в нужном направлении.
* Масштабирование: для увеличения или уменьшения масштаба используйте колесико мыши или кнопки "+" и "-" на карте.



Рисунок 16. Кнопки масштабирования

### Поиск объектов

Карта предоставляет возможность поиска зданий и аудиторий по их названию или номеру.

Поиск аудитории

* Введите номер аудитории в поле поиска, расположенное в верхней части карты.
* Нажмите кнопку "Поиск" или клавишу Enter.

Карта автоматически центрируется на выбранной аудитории, и появится всплывающее окно с информацией об аудитории.

Поиск здания

* Введите название здания в поле поиска.
* Нажмите кнопку "Поиск" или клавишу Enter.

Карта центрируется на выбранном здании, и появится всплывающее окно с информацией о здании.

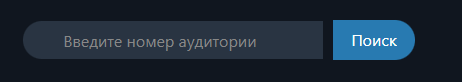


Рисунок 17. Форма для ввода и дальнейшего поиска объекта

### Переключение этажей

В многоэтажных зданиях карта позволяет переключаться между этажами для отображения соответствующих аудиторий и помещений.

* Выберите здание на карте.
* В появившемся всплывающем окне выберите вкладку с номером этажа.
* Карта обновится и отобразит план выбранного этажа.

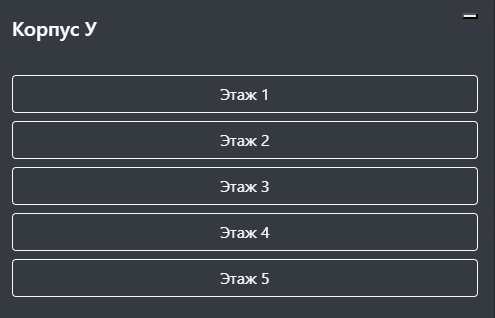


Рисунок 18. Выбор этажа

### Получение информации об объекте

Карта предоставляет возможность получения дополнительной информации о зданиях и аудиториях.

* Нажмите на здание или аудиторию на карте.

Появится всплывающее окно с детальной информацией об объекте, включая его название, номер, этаж, назначение и другие данные.

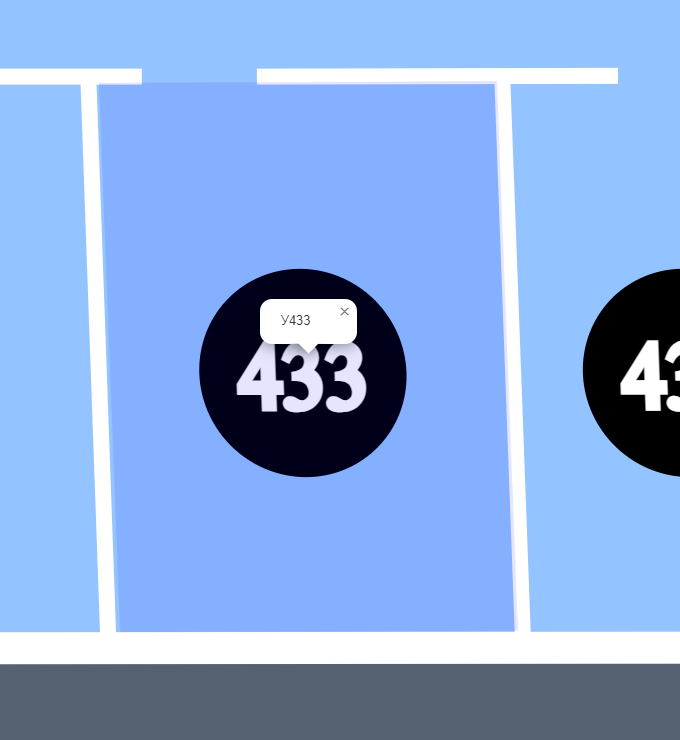


Рисунок 19. Отображение информации

## 5.3 Работа с мобильными устройствами

Интерактивная карта университета адаптирована для работы на мобильных устройствах (смартфоны и планшеты).

Использование карты на мобильных устройствах

Откройте веб-браузер на мобильном устройстве.

Перейдите по ссылке на интерактивную карту университета.

Используйте сенсорные жесты для панорамирования (перемещение двумя пальцами) и масштабирования (щипок для увеличения или уменьшения масштаба).

Функции поиска и получения информации об объектах работают аналогично версии для настольных компьютеров.

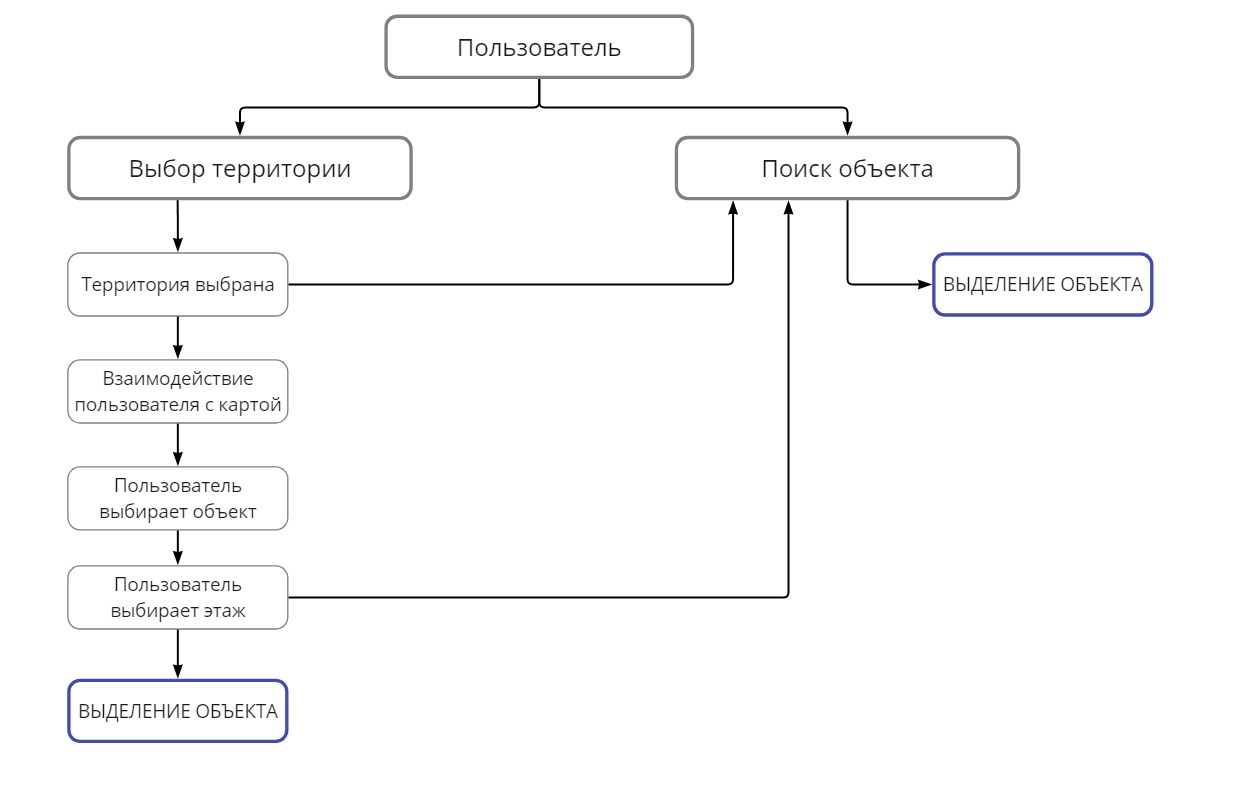


Рисунок 20. Пользовательский сценарий

# Глава 6. Руководство по установке и сопровождению ПО

## 6.1 Требования к разработчику

Компетенции и навыки

Для успешной установки, сопровождения и расширения функциональности разработанного ПО, разработчику необходимы следующие компетенции и навыки:

* Знание HTML, CSS и JavaScript: для работы с клиентской частью приложения.
* Опыт работы с Node.js: для управления серверной частью приложения.
* Опыт работы с для работы с базой данных.
* Знание основ работы с графическими редакторами: для редактирования и обновления SVG-файлов карт (например, Adobe Illustrator, Inkscape).
* Навыки работы с системами контроля версий: для управления исходным кодом (например, Git).

## 6.2 Программные требования

Для установки и сопровождения ПО необходимы следующие программные компоненты:

Клиентская часть

* Веб-браузер: поддержка современных стандартов HTML5, CSS3 и JavaScript.
* Библиотеки и фреймворки:
* Leaflet (для работы с картами)
* Bootstrap (для стилизации)
* jQuery (для взаимодействия с DOM)

Серверная часть

* Node.js: среда выполнения для серверной части.
* для хранения данных приложения.
* Git: для управления исходным кодом.

Разработка и поддержка

* Visual Studio Code: рекомендованная среда разработки (или любой другой текстовый редактор).
* Adobe Illustrator или любой другой графический редактор: для работы с векторными изображениями карт.

## 6.3 Руководство по установке разработанного ПО

Шаг 1: Клонирование репозитория

Сначала клонируйте репозиторий с исходным кодом приложения:

git clone https://github.com/LeoAW2/intmaps

cd

Шаг 2: Установка зависимостей

Установите необходимые зависимости для клиентской и серверной частей приложения:

npm install

Шаг 3: Настройка базы данных

Шаг 4: Запуск сервера

Запустите серверное приложение на Node.js:

## 6.4 Защита приложения после установки

Защита интерактивной карты университета после её установки включает несколько аспектов, направленных на обеспечение безопасности данных и устойчивости к потенциальным угрозам. В этом разделе рассматриваются основные меры по защите приложения и рекомендации по их внедрению.

### Аутентификация и авторизация

Внедрение аутентификации

* Использование системы логинов и паролей: Для доступа к административной части приложения следует использовать систему аутентификации на основе логинов и паролей. Это позволит ограничить доступ к функциям управления картой и данным.

Управление доступом

* Роли и права доступа: Настройка ролей и прав доступа для различных категорий пользователей (администраторы, редакторы, пользователи) позволит контролировать, какие действия могут выполнять разные пользователи.
* Регулярные проверки прав доступа: Периодические проверки и обновления прав доступа помогут поддерживать актуальность настроек безопасности.

### Защита данных

Шифрование данных

* SSL/TLS для передачи данных: Использование протоколов SSL/TLS для шифрования данных при передаче между клиентом и сервером защитит информацию от перехвата и подмены.
* Шифрование данных в базе данных: Шифрование данных, хранящихся в базе данных PostgreSQL, поможет защитить их от несанкционированного доступа в случае компрометации сервера.

Защита базы данных

* Настройка прав доступа к базе данных: Ограничение доступа к базе данных PostgreSQL на уровне пользователей и ролей позволит контролировать, кто и какие операции может выполнять с данными.
* Регулярные резервные копии: Настройка автоматического создания резервных копий базы данных позволит быстро восстановить данные в случае их потери или повреждения.

### Защита от внешних угроз

Защита от DDoS-атак

* Использование CDN (Content Delivery Network): Внедрение CDN для распределения нагрузки и фильтрации вредоносного трафика поможет защитить приложение от DDoS-атак.
* Мониторинг и ограничение запросов: Настройка лимитов на количество запросов от одного пользователя и использование инструментов мониторинга трафика помогут обнаружить и предотвратить DDoS-атаки.

### Мониторинг и обновления

Мониторинг системы

* Настройка систем мониторинга: Внедрение инструментов мониторинга (например, Prometheus, Grafana) для отслеживания состояния сервера, базы данных и веб-приложения поможет своевременно обнаруживать и реагировать на инциденты безопасности.
* Анализ логов: Регулярный анализ логов сервера и приложения позволит выявлять подозрительную активность и принимать меры по её устранению.

Обновления

Регулярное обновление ПО: Обеспечение своевременного обновления всех компонентов системы (серверного ПО, библиотек, фреймворков) позволит защитить приложение от известных уязвимостей.

## 6.5 Перенос приложения на другой компьютер/сервер

Шаги переноса

1. Копирование исходного кода: Склонируйте репозиторий с исходным кодом на новый компьютер/сервер.
2. Установка зависимостей: Установите все необходимые зависимости с помощью команды npm install.
3. Перенос базы данных: Скопируйте файл базы данных на новый сервер.
4. Настройка окружения: Обновите настройки окружения (например, переменные окружения) на новом сервере.
5. Запуск приложения: Запустите серверное приложение на новом сервере.

## 6.6 Редактирование карты

Шаг 1: Открытие SVG файла в редакторе

Откройте SVG файл карты в любом редакторе, поддерживающем работу с SVG, например, Adobe Illustrator или Inkscape. Эти инструменты позволяют редактировать векторные изображения и добавлять новые элементы.

Шаг 2: Добавление изменений

Добавьте необходимые изменения в SVG файл, такие как добавление новых зданий,

аудиторий или других объектов. Убедитесь, что все изменения корректно сохранены.

Шаг 3: Получение координат из XML

Выделите нужный объект в SVG редакторе и скопируйте его координаты из XML. Это можно сделать, открыв исходный код SVG файла и найдя соответствующий элемент.

Шаг 4: Конвертация координат через скрипт

Для конвертации координат в формат, совместимый с Leaflet был написан скрипт, используйте предоставленный скрипт на Python. Этот скрипт преобразует координаты из формата SVG в формат, используемый в библиотеке Leaflet.

Шаг 5: Добавление координат в код

После конвертации координат добавьте их в код вашего приложения. Например, если вы добавляете новый полигон для здания, обновите соответствующую функцию в вашем JavaScript коде.

Пример обновления JavaScript кода:

 const auditorium101aa = [

            [482.00238, 430.02731],

            [482.00238, 580.00547],

            [710.97888, 580.00547],

            [710.97888, 430.02731],

            [482.00238, 430.02731]

        ];

        if (building === 'Корпус А' && floor === 1) {

            const polygon = L.polygon(auditorium101aa, { fillOpacity: 0, stroke: false });

            polygon.options.name = 'Аудитория 101';

            addInteractivePolygon(polygon);

            allPolygons.addLayer(polygon);

            polygon.on('mouseover', function() {

                this.setStyle({ fillOpacity: 0.1, fillColor: 'blue' });

            });

            polygon.on('mouseout', function() {

                this.setStyle({ fillOpacity: 0, fillColor: 'blue' });

            });

            polygon.on('click', function() {

                map.fitBounds(polygon.getBounds(), {

                    padding: [50, 50],

                    animate: true,

                    duration: 100.0

                });

                this.bindPopup('Аудитория 101').openPopup();

            });

            map.on('overlayadd', function(e) {

                if (e.layer === currentFloorMap) {

                    polygon.addTo(map);

                }

            });

            map.on('popupclose', function() {

                map.setView([730, 960], -1, {

                    animate: true,

                    duration: 100.0

                });

            });

        }

Заключение

В ходе выполнения данной выпускной квалификационной работы была разработана интерактивная карта университета, доступная через веб-браузер. Основной целью проекта было улучшение навигации по территориям ВУЗа и обеспечение пользователей актуальной информацией о зданиях и аудиториях. В процессе работы были достигнуты следующие результаты:

* Создание и интеграция карты: Разработана интерактивная карта с использованием библиотеки Leaflet. Карта обеспечивает отображение всех значимых объектов на территории университета и позволяет пользователям взаимодействовать с ними.
* Поиск и навигация: Реализованы функции поиска аудитории по номеру и центрирования карты на выбранной аудитории. Эти функции значительно упрощают процесс поиска и ориентирования на территории университета.
* Интерактивность: Карта поддерживает интерактивные элементы, такие как выделение зданий и аудиторий при наведении курсора и клики по объектам. Это повышает удобство использования и делает взаимодействие с картой более интуитивным.
* Адаптивный дизайн: Приложение имеет адаптивный дизайн, что позволяет пользователям комфортно работать с картой как на настольных компьютерах, так и на мобильных устройствах.
* Интеграция с базой данных: Для хранения данных о зданиях и аудиториях использована реляционная база данных PostgreSQL. Это обеспечивает надежное хранение и быстрый доступ к данным, а также возможность их обновления в реальном времени.

### Анализ выполненной работы

Разработка интерактивной карты университета показала высокую актуальность и значимость данного проекта. Внедрение такого инструмента решает несколько важных задач:

* Улучшение навигации: Интерактивная карта позволяет студентам, преподавателям и посетителям быстро находить нужные объекты на территории университета, что особенно важно для новых студентов и гостей.
* Актуальность информации: Администрация университета получает возможность оперативно обновлять информацию о зданиях и аудиториях, что гарантирует пользователям доступ к актуальным данным.
* Удобство и доступность: Интерактивная карта доступна через веб-браузер, что делает её использование максимально удобным для пользователей. Адаптивный дизайн обеспечивает корректное отображение карты на различных устройствах.

### Перспективы развития

Разработка интерактивной карты университета имеет значительный потенциал для дальнейшего развития. Перспективы можно разделить на два направления: добавление новых функциональностей и перспективы создания новых возможностей.

### Добавление новых функциональностей:

1. Маршрутизация: Реализовать возможность прокладывать маршруты между зданиями и аудиториями, что упростит перемещение по кампусу и сделает навигацию еще более удобной.
2. Административный интерфейс: Создать административный интерфейс для управления картой, позволяющий легко добавлять, удалять и обновлять информацию о зданиях и аудиториях.
3. Динамический и нечеткий поиск: Внедрить функции динамического и нечеткого поиска, которые улучшат точность и скорость нахождения нужных объектов на карте.
4. Интеграция с расписанием занятий и мероприятий: Добавить возможность отображения расписания занятий и мероприятий, что позволит пользователям планировать свое время и маршруты более эффективно.
5. Поддержка инклюзивности и служб: Включить информацию о доступных маршрутах для людей с ограниченными возможностями и интегрировать данные о различных службах и сервисах университета.

### Перспективы создания новых возможностей:

* Аналитические инструменты: Разработать инструменты для анализа потока студентов и преподавателей, что поможет в оптимизации использования помещений и улучшении планирования расписания.
* Интеграция с другими системами: Обеспечить интеграцию карты с другими информационными системами университета, такими как система управления учебным процессом, библиотечная система и т.д.
* Мобильное приложение: Разработать мобильное приложение, которое будет использовать интерактивную карту и обеспечивать доступ ко всем функциям и данным на мобильных устройствах.
* Реализация VR и AR: Исследовать возможности использования виртуальной и дополненной реальности для улучшения визуализации и взаимодействия с картой.

Разработка интерактивной карты университета является важным шагом в направлении цифровизации образовательной среды. Этот проект способствует повышению качества образовательных услуг и комфорта для студентов и преподавателей, улучшает управление университетской инфраструктурой и удовлетворяет современные требования к доступу к информации. Выполненная работа демонстрирует значительный потенциал для дальнейшего развития и может стать основой для множества инновационных решений в области образовательных технологий.

Список использованных источников

1. Роббинс, Дж. (2018). HTML и CSS. Разработка и дизайн веб-сайтов. Москва: Вильямс.
2. Макфарланд, Д. (2017). JavaScript и jQuery. Исчерпывающее руководство. 3-е издание. Санкт-Петербург: Питер.
3. Уокер, Дж. (2017). Web-разработка. Полное руководство. Москва: Эксмо.
4. Касперски, В. (2018). Построение интерактивных веб-приложений. Москва: Бином.
5. Leaflet Documentation. Retrieved from: https://leafletjs.com/reference-1.7.1.html
6. PostgreSQL Documentation. Retrieved from: <https://www.postgresql.org/docs/>
7. Node.js Documentation. Retrieved from: https://nodejs.org/en/docs/
8. Express.js Documentation. Retrieved from: https://expressjs.com/en/4x/api.html
9. Coursera. Web Design for Everybody: Basics of Web Development & Coding. Retrieved from: <https://www.coursera.org/specializations/web-design>
10. Smashing Magazine. Articles on Web Design. Retrieved from: <https://www.smashingmagazine.com/>
11. Stack Overflow. Community questions and answers. Retrieved from: <https://stackoverflow.com/>
12. Интерактивная карта СПбГУ. Retrieved from: <https://guide.spbu.ru/map>
13. Интерактивная карта НИУ ВШЭ. Retrieved from: <https://navigation.hse.ru/?building=3&campus=pokrovka&endRoom=1439&floor=3&panel=route&startRoom=422>
14. Интерактивная карта СПбГАСУ. Retrieved from: <https://map.spbgasu.ru/>
15. Bootstrap Documentation. Retrieved from: https://getbootstrap.com/docs/5.0/getting-started/introduction/
16. Adobe Illustrator User Guide. Retrieved from: <https://helpx.adobe.com/illustrator/user-guide.html>
17. Duckett, J. (2014). HTML & CSS: Design and Build Websites. Indianapolis: Wiley.
18. Eloquent JavaScript: A Modern Introduction to Programming. (2018). 3rd Edition. Marijn Haverbeke.
19. Freeman, E., Robson, E., Bates, B., & Sierra, K. (2014). Head First JavaScript Programming: A Brain-Friendly Guide. O'Reilly Media.
20. Myers, G. (2004). The Art of Software Testing. 2nd Edition. Wiley.
21. Zakas, N.C. (2018). Understanding ECMAScript 6: The Definitive Guide for JavaScript Developers. No Starch Press.
22. McFarland, D. (2015). CSS: The Missing Manual. 4th Edition. O'Reilly Media.
23. Krasner, J. (2017). Vue.js: Up and Running: Building Accessible and Performant Web Apps. O'Reilly Media.
24. Rauschmayer, A. (2014). Speaking JavaScript: An In-Depth Guide for Programmers. O'Reilly Media.
25. Jurgenson, P. (2018). Pro Express.js: Master Express.js: The Node.js Framework For Your Web Development. Apress.
26. Ellis, R. (2017). Web Development with Node and Express: Leveraging the JavaScript Stack. O'Reilly Media.
27. Stack Overflow. Retrieved from: <https://stackoverflow.com/questions/tagged/leaflet.js>
28. Habr. Статьи по веб-разработке. Retrieved from: https://habr.com/ru/hub/webdev/
29. Metanit. JavaScript Руководство. Retrieved from: https://metanit.com/web/javascript/
30. Habr. Посты по теме "Интерактивные карты". Retrieved from: https://habr.com/ru/search/?q=interactive+maps
31. Metanit. Руководство по Node.js. Retrieved from: https://metanit.com/web/nodejs/
32. Habr. Посты по теме "PostgreSQL". Retrieved from: https://habr.com/ru/search/?q=PostgreSQL

ПРИЛОЖЕНИЕ

Index.js

document.addEventListener('DOMContentLoaded', function() {

    var map = L.map('map', {

        crs: L.CRS.Simple,

        minZoom: -9,

        maxZoom: 9,

        center: [730, 960],

        zoom: -1,

    });

    const bounds = L.latLngBounds([[0, 0], [1452, 1920]]);

    const territories = {

        'Ульянка': {

            svg: 'teritorU.svg',

            buildings: ['Корпус У', 'МедЦентр', 'Конгресс - центр', 'Спорткомплекс', 'Общежитие1', 'Общежитие2', 'Общежитие3']

        },

        'Лоцманская': {

            svg: 'teritorLoc.svg',

            buildings: ['Корпус А', 'Корпус Б', 'Корпус R']

        },

        'Горьковская': {

            svg: 'figa.svg',

            buildings: []

        }

    };

    let currentTerritoryLayer = null;

    let currentFloorMap = null;

    let allPolygons = L.layerGroup().addTo(map);

    let stateHistory = [];

    function saveState() {

        stateHistory.push({

            currentTerritoryLayer,

            currentFloorMap,

            polygons: allPolygons.getLayers(),

            center: map.getCenter(),

            zoom: map.getZoom()

        });

    }

    function restoreState() {

        if (stateHistory.length > 0) {

            const lastState = stateHistory.pop();

            clearMap();

            currentTerritoryLayer = lastState.currentTerritoryLayer;

            currentFloorMap = lastState.currentFloorMap;

            if (currentTerritoryLayer) {

                map.addLayer(currentTerritoryLayer);

            }

            if (currentFloorMap) {

                map.addLayer(currentFloorMap);

            }

            allPolygons.clearLayers();

            lastState.polygons.forEach(polygon => {

                addInteractivePolygon(polygon);

                allPolygons.addLayer(polygon);

            });

            map.setView(lastState.center, lastState.zoom);

        }

    }

    function loadInitialMap(svgUrl) {

        clearMap();

        currentTerritoryLayer = L.imageOverlay(svgUrl, bounds).addTo(map);

    }

    window.goBack = function() {

        restoreState();

    };

    window.selectTerritory = function(territory) {

        saveState();

        loadTerritorySvg(territories[territory].svg, territories[territory].buildings);

    };

    function loadTerritorySvg(svgUrl, buildings) {

        clearMap();

        currentTerritoryLayer = L.imageOverlay(svgUrl, bounds).addTo(map);

        if (buildings) {

            buildings.forEach(building => {

                addBuildingPolygon(building);

            });

        }

    }

    function addBuildingPolygon(building) {

        let polygonCoords;

        if (building === 'Корпус У') {

            polygonCoords = [

                [-349.77356 + 1080.5, 687.91819 - 109.4],

                [-113.8057 + 1080.5, 877.60409 - 109.4],

                [-113.8057 + 1080.5, 1244.3535 - 109.4],

                [-177.96932 + 1080.5, 1244.0029 - 109.4],

                [-178.31994 + 1080.5, 1342.1768 - 109.4],

                [-230.56245 + 1080.5, 1342.1768 - 109.4],

                [-230.21183 + 1080.5, 1271.3514 - 109.4],

                [-194.79912 + 1080.5, 1270.6501 - 109.4],

                [-195.50037 + 1080.5, 1243.0029 - 109.4],

                [-211.62893 + 1080.5, 1243.0029 - 109.4],

                [-210.57706 + 1080.5, 980.7385 - 109.4],

                [-177.26808 + 1080.5, 981.7904 - 109.4],

                [-177.26808 + 1080.5, 1152.8934 - 109.4],

                [-156.23083 + 1080.5, 1151.8415 - 109.4],

                [-155.8802 + 1080.5, 933.4047 - 109.4],

                [-388.69248 + 1080.5, 747.9262 - 109.4]

            ];

        } else if (building === 'МедЦентр') {

            polygonCoords = [

                [648, 1095], // прав низ

                [680, 1077], // прав верх

                [635.5, 983], // лев верх

                [601, 1001] // лев низ

            ];

        } else if (building === 'Спорткомплекс') {

            polygonCoords = [

                [669.6771899999999 + 140, 853.98252],

                [618.7283199999999 + 140, 810.71936],

                [532.3259599999999 + 140, 932.82312],

                [587.9854399999999 + 140, 975.83835],

                [669.6771899999999 + 140, 853.98252],

                [669.6771899999999 + 140, 853.98252]

            ];

        } else if (building === 'Конгресс - центр') {

            polygonCoords = [

                [930.9916280000001 - 359, 442.30072],

                [842.7298280000001 - 359, 483.4565],

                [808.5159780000001 - 359, 454.44911],

                [813.4745080000001 - 359, 349.32831],

                [930.9916280000001 - 359, 442.30072],

                [930.9916280000001 - 359, 442.30072]

            ];

        } else if (building === 'Общежитие1') {

            polygonCoords = [

                [1053.6408260000003 - 682, 307.86262 + 1],

                [1036.9057860000003 - 682, 331.8495 + 1],

                [986.3287960000002 - 682, 292.4292 + 1],

                [1003.0018560000002 - 682, 268.38034 + 1],

                [1053.6408260000003 - 682, 307.86262 + 1],

                [1053.6408260000003 - 682, 307.86262 + 1]

            ];

        } else if (building === 'Общежитие2') {

            polygonCoords = [

                [1023.679328 - 622, 409.28415 + 0.5],

                [1006.959098 - 622, 433.03872 + 0.5],

                [956.3162980000001 - 622, 393.63769 + 0.5],

                [972.8612180000001 - 622, 369.79547 + 0.5],

                [1023.679328 - 622, 409.28415 + 0.5],

                [1023.679328 - 622, 409.28415 + 0.5]

            ];

        } else if (building === 'Общежитие3') {

            polygonCoords = [

                [992.4959860000002 - 559.8, 509.5179 + 0.5],

                [975.7757560000002 - 559.8, 533.25055 + 0.5],

                [925.1329460000002 - 559.8, 493.82761 + 0.5],

                [941.6778660000002 - 559.8, 469.98539 + 0.5],

                [992.4959860000002 - 559.8, 509.5179 + 0.5],

                [992.4959860000002 - 559.8, 509.5179 + 0.5]

            ];

        } else if (building === 'Корпус А') {

            polygonCoords = [

                [769.9031220000002 + 56.5, 367.84375],

                [800.1843720000002 + 56.5, 403.84375],

                [485.3328120000001 + 56.5, 672.125],

                [459.82500200000015 + 56.5, 640.24219],

                [769.9031220000002 + 56.5, 367.84375],

                [769.9031220000002 + 56.5, 367.84375]

            ];

        } else if (building === 'Корпус Б') {

            polygonCoords = [

                [551.0609479999998 + 332, 522.81482],

                [575.4739479999998 + 332, 551.7679700000001],

                [472.5159179999999 + 332, 639.1757800000001],

                [448.1018579999999 + 332, 610.2265600000002],

                [551.0609479999998 + 332, 522.81482],

                [551.0609479999998 + 332, 522.81482]

            ];

        } else if (building === 'Корпус R') {

        polygonCoords = [

                [400.25807199999997 + 518, 563.83479],

                [370.31200199999995 + 518, 589.23437],

                [465.72216199999997 + 518, 701.18359],

                [494.67892199999994 + 518, 676.75643],

                [400.25807199999997 + 518, 563.83479],

                [400.25807199999997 + 518, 563.83479]

        ];

    }

        const polygon = L.polygon(polygonCoords, {

            color: 'blue',

            fillOpacity: 0,

            stroke: false

        });

        polygon.options.name = building;

        addInteractivePolygon(polygon);

        allPolygons.addLayer(polygon);

    }

    function addInteractivePolygon(polygon) {

        polygon.on('mouseover', function() {

            this.setStyle({ fillOpacity: 0.1, fillColor: 'blue' });

        });

        polygon.on('mouseout', function() {

            this.setStyle({ fillOpacity: 0, fillColor: 'blue' });

        });

        polygon.on('click', function() {

            if (this.options.name !== 'Аудитория 101') {

                showFloorModal(this.options.name);

            }

        });

    }

    function clearMap() {

        if (currentTerritoryLayer) {

            map.removeLayer(currentTerritoryLayer);

            currentTerritoryLayer = null;

        }

        if (currentFloorMap) {

            map.removeLayer(currentFloorMap);

            currentFloorMap = null;

        }

        allPolygons.clearLayers();

    }

    window.loadSvgLayer = function(building, floor) {

        saveState();

        clearMap();

        const imageUrl = `${building}\_floor\_${floor}.svg`;

        currentFloorMap = L.imageOverlay(imageUrl, bounds).addTo(map);

        const auditorium101aa = [

            [947.0698639999999 - 468, 1292.0314],

            [878.473994 - 468, 1294.5973000000001],

            [878.645054 - 468, 1344.3764],

            [947.411984 - 468, 1341.8104],

            [947.0698639999999 - 468, 1292.0314],

            [947.0698639999999 - 468, 1292.0314]

        ];

        if (building === 'Корпус У' && floor === 4) {

            const polygon = L.polygon(auditorium101aa, { fillOpacity: 0, stroke: false });

            polygon.options.name = 'У433';

            addInteractivePolygon(polygon);

            allPolygons.addLayer(polygon);

            polygon.on('mouseover', function() {

                this.setStyle({ fillOpacity: 0.1, fillColor: 'blue' });

            });

            polygon.on('mouseout', function() {

                this.setStyle({ fillOpacity: 0, fillColor: 'blue' });

            });

            polygon.on('click', function() {

                map.fitBounds(polygon.getBounds(), {

                    padding: [50, 50],

                    animate: true,

                    duration: 100.0

                });

                this.bindPopup('У433').openPopup();

            });

            map.on('overlayadd', function(e) {

                if (e.layer === currentFloorMap) {

                    polygon.addTo(map);

                }

            });

            map.on('popupclose', function() {

                map.setView([730, 960], -1, {

                    animate: true,

                    duration: 100.0

                });

            });

        }

    }

    function showFloorModal(building) {

        const modalHtml = `

            <div class="modal fade" id="floorModal" tabindex="-1" aria-labelledby="floorModalLabel" aria-hidden="true">

                <div class="modal-dialog modal-dialog-centered">

                    <div class="modal-content bg-dark text-white">

                        <div class="modal-header border-0">

                            <h5 class="modal-title" id="floorModalLabel">${building}</h5>

                            <button type="button" class="btn-close btn-close-white" data-bs-dismiss="modal" aria-label="Close"></button>

                        </div>

                        <div class="modal-body">

                            ${getFloorOptions(building)}

                        </div>

                    </div>

                </div>

            </div>

        `;

        document.body.insertAdjacentHTML('beforeend', modalHtml);

        const modal = new bootstrap.Modal(document.getElementById('floorModal'));

        modal.show();

        document.getElementById('floorModal').addEventListener('hidden.bs.modal', function () {

            document.getElementById('floorModal').remove();

        });

    }

    function getFloorOptions(building) {

        let floors = 5;

        if (building === 'Корпус Б') {

            floors = 6;

        }

        if (building === 'Конгресс - центр') {

            floors = 2;

        }

        if (building === 'Спорткомплекс') {

            floors = 2;

        }

        if (building === 'Общежитие1') {

            floors = 12;

        }

        if (building === 'Общежитие2') {

            floors = 12;

        }

        if (building === 'Общежитие3') {

            floors = 12;

        }

        if (building === 'МедЦентр') {

            floors = 3;

        }

        if (building === 'Корпус R') {

            floors = 1;

        }

        let floorOptions = '';

        for (let i = 1; i <= floors; i++) {

            floorOptions += `<button class="btn btn-outline-light btn-block mb-2" onclick="selectFloor('${building}', ${i})">Этаж ${i}</button>`;

        }

        return floorOptions;

    }

    window.selectFloor = function(building, floor) {

        $('#floorModal').modal('hide');

        loadSvgLayer(building, floor);

    }

    function highlightAuditorium(auditorium) {

        const foundPolygon = allPolygons.getLayers().find(polygon => polygon.options.name === auditorium);

        if (foundPolygon) {

            map.fitBounds(foundPolygon.getBounds(), {

                padding: [50, 50],

                animate: true,

                duration: 100.0

            });

            foundPolygon.setStyle({ color: 'red' });

        }

    }

    window.searchAuditorium = function() {

        const searchBox = document.getElementById('searchBox');

        const query = searchBox.value.trim();

        if (query) {

            highlightAuditorium(query);

        }

    };

    const searchControl = new L.Control.Search({

        layer: allPolygons,

        propertyName: 'name',

        sourceData: function(text, callResponse) {

            const results = {};

            auditoriums.forEach(function(auditorium) {

                if (auditorium.title.toLowerCase().includes(text.toLowerCase())) {

                    results[auditorium.title] = auditorium.loc;

                }

            });

            callResponse(results);

        },

        textPlaceholder: 'Введите номер аудитории',

        moveToLocation: function(latlng, title, map) {

            const foundPolygon = allPolygons.getLayers().find(polygon => polygon.options.name === title);

            if (foundPolygon) {

                map.fitBounds(foundPolygon.getBounds(), {

                    padding: [50, 50],

                    animate: true,

                    duration: 100.0

                });

                foundPolygon.setStyle({ color: 'red' });

                if (title === 'Аудитория 101') {

                    selectTerritory('Лоцманская');

                    setTimeout(function() {

                        selectFloor('Корпус А', 1);

                    }, 500);

                }

            } else {

                map.setView(latlng, 17);

            }

        }

    });

    const searchContainer = document.getElementById('search');

    searchControl.addTo(map);

    searchContainer.appendChild(searchControl.getContainer());

    L.Control.BackButton = L.Control.extend({

        onAdd: function(map) {

            const container = L.DomUtil.create('div', 'leaflet-control-custom');

            container.innerHTML = '←'; // Текст кнопки "Назад"

            container.style.background = '#0B63F6 51%'; // Цвет фона

            container.style.color = 'white'; // Цвет текста

            container.style.width = '10px';

            container.style.height = '5px';

            container.style.borderRadius = '20px'; // Округленные углы

            container.style.cursor = 'pointer'; // Указатель при наведении

            container.style.fontSize = '16px'; // Размер текста

            container.style.transition = 'background-color 0.3s ease'; // Плавный переход цвета фона

            container.onmouseover = function() {

                container.style.backgroundColor = '#0056b3';

            };

            container.onmouseout = function() {

                container.style.backgroundColor = '#0B63F6 51%';

            };

            container.onmousedown = function() {

                container.style.backgroundColor = '#0B63F6 51%';

            };

            container.onmouseup = function() {

                container.style.backgroundColor = '#0B63F6 51%';

            };

            container.onclick = function() {

                goBack();

            };

            return container;

        }

    });

    L.control.backButton = function(opts) {

        return new L.Control.BackButton(opts);

    }

    L.control.backButton({ position: 'topleft' }).addTo(map);

})

App.js

const express = require('express');

const path = require('path');

const { Pool } = require('pg');

const app = express();

const port = process.env.PORT || 3000;

// Настройка пула соединений с базой данных PostgreSQL

const pool = new Pool({

  user: 'postgres',

  host: 'localhost',

  database: 'INTmaps',

  password: '6539',

  port: 5432,

});

// Middleware для раздачи статических файлов

app.use(express.static(path.join(\_\_dirname, 'public')));

// Основной маршрут для отправки index.html

app.get('/', (req, res) => {

  res.sendFile(path.join(\_\_dirname, 'public', 'index.html'));

});

// Маршрут для поиска аудиторий

app.get('/search', async (req, res) => {

  const { room } = req.query;

  if (!room) {

    return res.status(400).send('Room number is required');

  }

  try {

    const result = await pool.query(

      'SELECT \* FROM auditoriums WHERE room\_number = $1',

      [room]

    );

    res.json(result.rows);

  } catch (error) {

    console.error(error);

    res.status(500).send('Server error');

  }

});

// Запуск сервера

app.listen(port, () => {

  console.log(`Сервер запущен на порту ${port}`);

});

Index.html

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Интерактивная карта СПБгМТУ</title>

    <link rel="stylesheet" href="leaflet/leaflet.css" />

    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">

    <link rel="stylesheet" href="css/map.css"/>

    <link rel="stylesheet" href="leaflet-search-master/dist/leaflet-search.min.css">

</head>

<header>

    <div id="territory-buttons" class="btn-group" role="group">

        <button type="button" class="btn btn-primary" onclick="selectTerritory('Ульянка')">Ульянка</button>

        <button type="button" class="btn btn-primary" onclick="selectTerritory('Лоцманская')">Лоцманская</button>

        <button type="button" class="btn btn-primary" onclick="selectTerritory('Горьковская')">Горьковская</button>

    </div>

    <div id="glav">

        <img src="glav.png" alt="llll">

    </div>

    <div id="search">

        <input type="text" id="searchBox" class="form-control" placeholder="Введите номер аудитории">

        <button class="btn btn-primary" onclick="searchAuditorium()">Поиск</button>

    </div>

</header>

<body >

    <div id="map"></div>

<script src="leaflet/leaflet.js"></script>

<script src="leaflet-search-master/dist/leaflet-search.min.js"></script>

<script src="JS/script.js"></script>

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.3/dist/umd/popper.min.js"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>