МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ШКОЛА КОМПЬЮТЕРНЫХ НАУК

Кафедра программного обеспечения

ОТЧЕТ  
О РАЗРАБОТКЕ ПРОЕКТА ПО ДИСЦИПЛИНЕ  
«Интеллектуальные системы поддержки принятия решений»

ТЕМА ПРОЕКТА

«Разработка системы навигации по университетскому корпусу»

| Выполнил  обучающийся 4 курса,  МОиАИС-20.03 группы | (подпись) | Титов  Павел  Сергеевич |
| --- | --- | --- |
| Выполнил  обучающийся 4 курса,  МОиАИС-20.03 группы | (подпись) | Коняев Илья Алексеевич |
| Научный руководитель проекта  Доцент, к.т.н. | (подпись) | Воробьева Марина Сергеевна |

«Тюмень 2024»

# СОДЕРЖАНИЕ

[**СОДЕРЖАНИЕ 2**](#_s4mws2j6yoam)

[**ВВЕДЕНИЕ 3**](#_1fob9te)

[**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТА 5**](#_9x0ulp4qyja2)

[1.1. ОПИСАНИЕ СПЕЦИФИКИ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_bh5uobop3oud)

[1.2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА 6](#_qnzm587bioy)

[**ГЛАВА 2. РАЗРАБОТКА ПРОЕКТА 9**](#_mqbp9uu9si8y)

[2.1. ИДЕЯ ПРОЕКТА 9](#_ipjwc8p6wiuy)

[2.2. ТРЕБОВАНИЯ К ПРОЕКТУ И ОГРАНИЧЕНИЯ 10](#_5p82grpowyf7)

[2.3. ПРАКТИЧЕСКОЕ ОБОСНОВАНИЕ 12](#_36g3vrx66l33)

[**ГЛАВА 3. РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ 16**](#_qjtdgolvzdj3)

[3.1. СОБСТВЕННЫЕ ПРАКТИЧЕСКИЕ РЕШЕНИЯ 16](#_30hfh0qg9c7z)

[3.2. ОПИСАНИЕ ДАННЫХ 18](#_k44zj95s1tmx)

[3.3. ОСОБЕННОСТИ РАЗРАБОТКИ 20](#_diclhm98xkqm)

[**ГЛАВА 4. ФУНКЦИОНАЛЬНОСТЬ И ТЕСТИРОВАНИЕ 25**](#_b0px8qpgh0u3)

[4.1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ПРОЕКТА 25](#_ravk7cjmytew)

[4.2. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ПРОЕКТА 26](#_g7l3v5dx91z4)

[4.3. ТЕСТИРОВАНИЕ ПРОЕКТА 27](#_nlevhgp6ykq)

[**ЗАКЛЮЧЕНИЕ 30**](#_x0p5x1xc0tny)

[**СПИСОК ЛИТЕРАТУРЫ 32**](#_8m98ib4j0jpr)

[**Приложение 1 33**](#_k7z0xi9adkvf)

[**Приложение 2 34**](#_gbateu4h026t)

[**Приложение 3 35**](#_caa6zi86e70c)

[**Приложение 4 36**](#_rv66n91ozyki)

# ВВЕДЕНИЕ

Навигация в университетских корпусах является проблемной задачей для студентов первых курсов и гостей университета, которые не знакомы с расположением аудиторий, административных кабинетов и других ключевых помещений.

Современные цифровые технологии предоставляют возможности для создания интерактивных систем внутренней навигации, которые могут улучшить пользовательский опыт. Использование веб-приложений, QR-кодов и других технологических решений позволяет находить маршруты, улучшая образовательную и социальную среду университета. Анализ существующих решений, например, интерактивных карт в УрФУ[4] и ИТМО[5], показывает востребованность таких систем и их положительное влияние на удобство пользователей.

Вместе с тем в большинстве университетов отсутствуют универсальные решения для внутренней навигации, способные учитывать динамику инфраструктуры и разнообразие пользовательских групп. Это приводит к ряду проблем:

* сложности в ориентировании;
* затруднения в поиске кабинетов;
* отсутствие актуальной информации о перемещении объектов при ремонтных работах или изменении планировок.

Таким образом, актуальность разработки системы внутренней навигации обусловлена необходимостью решения этих проблем. Пользователи, включая студентов, преподавателей и сотрудников, ожидают от системы интуитивно понятного интерфейса, точного поиска маршрутов и возможности работы на разных устройствах.

Разработка интерактивной карты кампуса, включающей визуализацию помещений, поиск маршрутов и использование современных технологий, таких как QR-коды, создаст универсальный инструмент для улучшения навигации в университетских корпусах.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТА

## 1.1. ОПИСАНИЕ СПЕЦИФИКИ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область - внутренняя навигация в университетских корпусах.

Специфика этой предметной области:

* Разнообразие пользователей: Студенты разных курсов, преподаватели, сотрудники и посетители. У каждого разные уровень знакомства с инфраструктурой университета.
* Сложная архитектура зданий: Многочисленные корпуса, этажи, аудитории и переходы. Это затрудняет ориентацию для новичков.
* Интеграция с цифровыми технологиями: Возможность использования веб-приложений, QR-кодов, NFC-меток и других технологий для облегчения навигации.
* Адаптивность и масштабируемость: Система должна легко обновляться при изменениях в инфраструктуре и адаптироваться к новым требованиям.

Примеры существующих решений:

* УрФУ: Университет предоставляет интерактивную карту, доступную онлайн, которая помогает первокурсникам и гостям ориентироваться в кампусе. Также в УрФУ внедрён онлайн-навигатор для первокурсников с инструкциями и маршрутами.[4]
* ИТМО: Университет предлагает карту корпусов, где отображены все здания, общежития и учебные аудитории. Это решение позволяет быстро найти нужное место на территории университета.[5]

Создание подобной системы внутренней навигации позволит пользователям быстро находить нужное место на территории университета, повышая удобство и скорость передвижения, а также улучшая образовательную и социальную среду.

## 1.2. ТЕОРЕТИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЕКТА

Проект системы внутренней навигации в университетских корпусах основывается на ряде теоретических концепций и алгоритмов из теории графов и алгоритмов поиска кратчайших путей.[1]

Теоретические положения:

1. Теория графов:
   * Университетские помещения и пути между ними моделируются в виде графа, где вершины представляют собой комнаты, аудитории или ключевые точки (например, входы, лестницы), а рёбра — возможные пути между ними.
2. Взвешенные графы:
   * Каждому ребру присваивается вес, отражающий сложность перемещения. Это позволяет учитывать различные факторы при выборе кратчайшего маршрута.

Методы и алгоритмы:

1. Алгоритм Дейкстры[7]:
   * Используется для нахождения кратчайшего пути в графе с неотрицательными весами рёбер.
   * Обеспечивает эффективный поиск оптимального маршрута от начальной до конечной точки.
   * Применяется в функции nx.shortest\_simple\_paths из библиотеки NetworkX[2], используемой в проекте.
2. Поиск в ширину (BFS)[8]:
   * Применяется для поиска кратчайшего пути в невзвешенных графах или когда все рёбра имеют одинаковый вес.
   * Может быть полезен для упрощённых моделей навигации.

Логика работы алгоритма построения маршрута:

* Инициализация:
  + Загружается граф помещений университета с информацией о вершинах (помещениях) и рёбрах (путях между ними).
* Поиск дверей:
  + Для заданных кабинетов A и B определяются соответствующие им двери (точки входа/выхода), чтобы учесть возможные множественные входы в помещение.
* Генерация маршрутов:
  + Для каждой пары дверей (от двери кабинета A к двери кабинета B) генерируются возможные пути с использованием функции nx.shortest\_simple\_paths, которая возвращает простые пути, отсортированные по возрастанию их общей стоимости (веса).
* Оценка и отбор лучших маршрутов:
  + Для каждого пути вычисляется суммарный вес с помощью функции compute\_path\_weight.
  + Собираются все найденные пути, после чего выбираются top\_k путей с наименьшим весом, обеспечивая пользователю несколько оптимальных вариантов.
* Извлечение идентификаторов линий:
  + Для каждого выбранного пути определяется последовательность линий (коридоров, переходов) с помощью функции extract\_line\_ids, что позволяет предоставить пользователю более детальную информацию о маршруте.
* Возврат результатов:
  + Формируется список кратчайших маршрутов с указанием пути, соответствующих линий и общей стоимости, который затем представляется пользователю.

Использование этих алгоритмов и методов обеспечивает построение маршрутов внутри университетских корпусов с учётом архитектурных особенностей зданий.

Дополнительно, процесс построения маршрута представлен на диаграмме последовательности (см. приложение 4).

# ГЛАВА 2. РАЗРАБОТКА ПРОЕКТА

## 2.1. ИДЕЯ ПРОЕКТА

Основная идея: Разработка системы внутренней навигации, которая будет помогать пользователям находить необходимые аудитории и помещения с помощью интерактивной карты и предложенных маршрутов.

Как будут решены проблемы и удовлетворены требования:

1. Интерактивная карта:
   * Визуализация корпусов и этажей университета с подробной разметкой кабинетов, лекционных залов, лабораторий и административных помещений.
   * Возможность поиска кабинетов по номеру или названию (например, "Деканат" или "Аудитория 302").
2. Построение маршрутов:
   * Алгоритм определения кратчайшего маршрута между пунктами (например, от входа до нужного кабинета).[1]
3. Мобильное приложение или веб-интерфейс:
   * Доступность на смартфонах, планшетах и компьютерах.
   * QR-коды в ключевых точках корпусов для быстрого доступа к карте.
4. Гибкость и актуальность:
   * Возможность обновления данных при изменении инфраструктуры корпусов.
   * Адаптация системы для новых требований (например, добавление маршрутов для эвакуации в чрезвычайных ситуациях).

## 

## 2.2. ТРЕБОВАНИЯ К ПРОЕКТУ И ОГРАНИЧЕНИЯ

Функциональные требования:

1. Поиск кабинетов и помещений:
   * Возможность поиска аудитории по номеру, названию, назначению (например, «деканат», «лекторий»).
   * Отображение подробного описания маршрута (например, «вход с главного корпуса, поднимитесь на 3-й этаж, кабинет 312»).
2. Интерактивная карта:
   * Карта корпусов и этажей с выделением ключевых зон (кабинеты, лестницы, лифты, туалеты, зоны отдыха).
   * Визуализация маршрута от текущей точки до конечного пункта (начальная точка задаётся пользователем вручную).
3. Технологии:
   * QR-коды, размещённые в ключевых точках университета (например, на входах и этажах), для быстрого доступа к карте и маршрутам.
   * Веб-приложение для работы на большинстве устройств (смартфоны, планшеты, компьютеры).
4. Пользовательский интерфейс:
   * Интерфейс для взаимодействия.
   * Визуализации маршрутов.
   * Доступность на русском языке.
5. Актуальность данных:
   * Возможность обновления схемы помещений при изменениях (например, при переносе кабинетов или ремонте).

Нефункциональные требования:

1. Кроссплатформенность:
   * Работа через браузер и на всех распространённых операционных системах (Android, iOS, Windows, macOS).
2. Производительность:
   * Построение маршрутов и загрузка карты даже при высоком числе одновременных пользователей.
3. Надежность:
   * Стабильная работа приложения без сбоев.
   * Высокая доступность карты даже при ограниченном интернет-соединении (например, возможность загрузки упрощённых версий карт).

Учитываемые ограничения

1. Отсутствие возможности определения местоположения:
   * Технические ограничения исключают использование GPS, WiFi, Bluetooth-маяков или других методов автоматического определения местоположения пользователя.
   * Пользователь должен самостоятельно задавать начальную точку маршрута, например, выбрав её на карте или указав номер помещения.
2. Отсутствие привязки к сети университета:
   * Система не должна зависеть от внутренней IT-инфраструктуры университета, такой как WiFi-сеть, чтобы оставаться доступной для всех категорий пользователей, включая гостей.
3. Финансовые ограничения:
   * Проект должен минимизировать затраты, используя недорогие технологии, такие как QR-коды и веб-приложения, вместо дорогостоящих аппаратных решений.
4. Физические ограничения пространства:
   * Навигация должна учитывать существующую инфраструктуру университетских корпусов, включая сложную планировку, узкие коридоры и старые здания без современного оборудования (например, лифтов или широких дверей).
5. Доступность:
   * Необходимость обеспечения работы системы в режиме онлайн с минимальными требованиями к скорости интернет-соединения.

## 2.3. ПРАКТИЧЕСКОЕ ОБОСНОВАНИЕ

Проект построен с использованием современных технологий, библиотек и методик. Ниже представлены ключевые аспекты, применяемые при разработке.

Используемые технологии

1. Язык программирования:
   * Python: Основной язык разработки.
2. Фреймворк для создания API:
   * FastAPI: Высокопроизводительный веб-фреймворк, поддерживающий автоматическую генерацию документации Swagger и простую интеграцию зависимостей. Используется для создания REST API[3], обеспечивающего доступ к данным и маршрутам.
3. Графы и алгоритмы:
   * NetworkX: Библиотека для работы с графами, предоставляющая функции для построения, анализа и поиска кратчайших путей в графах.[2]
   * Применяется для моделирования университетской инфраструктуры как графа, где вершины — помещения или ключевые точки, а рёбра — пути между ними.
4. Работа с SVG:
   * XML-парсинг: Используется для обработки SVG-файлов планов этажей. Библиотека xml.etree.ElementTree обрабатывает элементы графической карты и модифицирует их, чтобы отображать маршруты или выделять текущий этаж.
5. Валидация данных:
   * Pydantic: Используется для строгой валидации входных данных, таких как JSON-файлы[6] с описанием помещений, дверей и графов, а также для типизации объектов.

Базы данных и файловая структура

* В качестве хранилища данных используется JSON-файл[6] plan\_combined.json, содержащий описание объектов (помещений, дверей) и графа (узлов и рёбер). JSON[6] формат позволяет обновлять данные и переносить их между системами.
* Данные обрабатываются в модуле GraphRepository, который обеспечивает доступ к графу и спискам объектов.

Архитектура и организация кода

1. Модульная структура:
   * Код организован в независимые модули:
     + API: Обработчики запросов (например, построение маршрутов, загрузка SVG-планов).[3]
     + Core: Конфигурационные настройки (например, пути к файлам).
     + Repositories: Работа с данными и графом.
     + Services: Логика обработки маршрутов и взаимодействия с SVG.
     + Domain: Модели данных для строгой типизации.
2. Принципы разработки:
   * Разделение ответственности: Каждая часть системы отвечает за отдельный аспект функциональности, это упрощает тестирование и поддержку.
   * Инъекция зависимостей (Dependency Injection): Используется в FastAPI для управления зависимостями, такими как доступ к репозиториям данных и сервисам маршрутизации.

Приёмы программирования

1. Алгоритм поиска маршрута:
   * Используется метод поиска простых путей в графах с минимальной стоимостью (весом) рёбер, реализованный в модуле RouteService с использованием nx.shortest\_simple\_paths.
2. Обработка SVG:
   * Модули svg\_processor.py обрабатывают графические планы, выделяя маршруты и скрывая лишние элементы.
3. Асинхронное программирование:
   * FastAPI поддерживает асинхронную обработку запросов, что повышает производительность системы при высоких нагрузках.[3]

Защита и безопасность

1. CORS:
   * Для предотвращения ограничений при вызове API из различных источников используется промежуточное ПО CORSMiddleware.
2. Обработка ошибок:
   * Обработка ошибок на каждом этапе (от загрузки данных до построения маршрутов) с возвратом соответствующих HTTP-ответов клиенту (например, 404, 400, 500).

Использование API

* API предоставляет доступ к следующему функционалу:
  + Построение маршрутов между двумя кабинетами (/floor-plan).
  + Получение плана этажа и маршрута в формате SVG.
  + Список доступных объектов и их человекочитаемых названий (/objects).

Масштабируемость и расширяемость

1. Обновление данных:
   * Простота обновления JSON-файлов с описанием помещений и графов для учёта изменений в инфраструктуре университета.
2. Добавление новых технологий:
   * Возможность интеграции дополнительных источников данных (например, базы данных) без необходимости значительных изменений в коде благодаря модульной архитектуре.
3. Интеграция с внешними системами:
   * API можно использовать для взаимодействия с мобильными приложениями, веб-порталами и другими сервисами.

Использование технологий, таких как FastAPI, NetworkX[2] и Pydantic, в сочетании с модульной архитектурой и алгоритмами работы с графами, позволяет создавать расширяемую и производительную систему внутренней навигации.

# ГЛАВА 3. РЕАЛИЗАЦИЯ И РЕЗУЛЬТАТЫ

## 3.1. СОБСТВЕННЫЕ ПРАКТИЧЕСКИЕ РЕШЕНИЯ

В проекте разработано и внедрено решение для автоматизированной обработки SVG-файлов планов этажей, обеспечивающее уникальную идентификацию объектов, таких как двери, офисы и лестницы. Это решение направлено на оптимизацию задач внутренней навигации. (см. Рисунок 1)

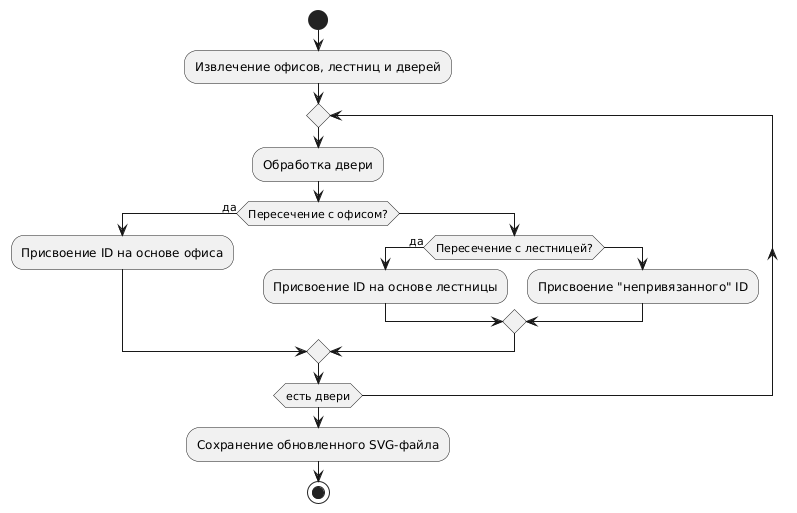


Рисунок 1. Алгоритм присвоения уникальных идентификаторов дверям на основе их положения.

### 3.1.1. УНИКАЛЬНАЯ ИДЕНТИФИКАЦИЯ ОБЪЕКТОВ С УЧЕТОМ ПОРОГОВ ПЕРЕСЕЧЕНИЯ

Разработан алгоритм, автоматически присваивающий уникальные идентификаторы дверям на основе их расположения относительно офисов и лестниц. Алгоритм учитывает пороговое значение пересечения (threshold), что позволяет корректно идентифицировать двери, даже при несовпадении координат на 100%.

Ключевые особенности:

* Пространственная обработка: Применение геометрических проверок для анализа пересечений дверей с офисами и лестницами.
* Гибкость порогов: Возможность задания порога, учитывающего незначительные отклонения в данных SVG.
* Привязка ID: Генерация логически структурированных идентификаторов (например, Floor\_Second\_Door\_Office\_201\_1), упрощающих дальнейшую работу с данными.

Решаемые проблемы:

* Автоматизация процесса идентификации дверей вместо ручного присвоения идентификаторов.
* Учет дверей, расположенных вблизи нескольких объектов.

### 3.1.2. ПРИВЯЗКА ДВЕРЕЙ К ЛЕСТНИЦАМ И ДРУГИМ ОБЪЕКТАМ

Алгоритм обрабатывает не только офисы, но и лестницы (stairs\_group), анализируя их границы. Для лестниц реализована логика определения общего прямоугольника на основе всех включенных элементов (например, нескольких прямоугольников).

Ключевые особенности:

* Генерация уникальных идентификаторов для дверей, связанных с лестницами (например, Floor\_First\_Door\_Stairs\_1\_2).
* Обработка групп объектов, таких как лестницы, даже если они состоят из нескольких геометрических фигур.

Решаемые проблемы:

* Автоматизация привязки дверей к сложным объектам, включая группы элементов.

### 3.1.3. АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ ИДЕНТИФИКАТОРОВ ДЛЯ НЕИДЕНТИФИЦИРОВАННЫХ ЭЛЕМЕНТОВ

Для объектов, не пересекающихся с офисами или лестницами, алгоритм присваивает уникальные "непривязанные" идентификаторы (например, Floor\_Second\_Door\_Unassigned\_1).

Ключевые особенности:

* Гарантия уникальности идентификаторов для объектов без прямой привязки к другим элементам.

Решаемые проблемы:

* Обеспечение целостности данных для всех элементов карты, включая нестандартные категории.

Разработанное решение автоматизирует обработку данных для внутренней навигации, ускоряет обработку SVG-файлов, генерирует уникальные идентификаторы для объектов, минимизирует ошибки и подготавливает данные для интеграции в систему маршрутизации.

## 3.2. ОПИСАНИЕ ДАННЫХ

В таблице, представленной на Рисунке 2, отражены результаты работы алгоритма автоматической идентификации объектов на основе их пространственного расположения. Каждая строка таблицы описывает отдельный объект на этаже и включает следующие ключевые параметры:

* ID объекта — уникальный идентификатор, сгенерированный алгоритмом.
* Этаж, Тип, Детали — характеристики объекта: этаж, категория (например, офис или дверь) и дополнительная информация о его назначении.
* Координаты и размеры — параметры, определяющие местоположение (X, Y) и габариты (ширина и высота) объекта на плане этажа.
* ID двери — уникальный идентификатор связанной с объектом двери (если имеется).
* Параметры двери — координаты и размеры двери, уточняющие её положение относительно основного объекта.

Эти данные обеспечивают целостность внутренней навигационной системы, упрощают взаимодействие с объектами и формирование маршрутов.

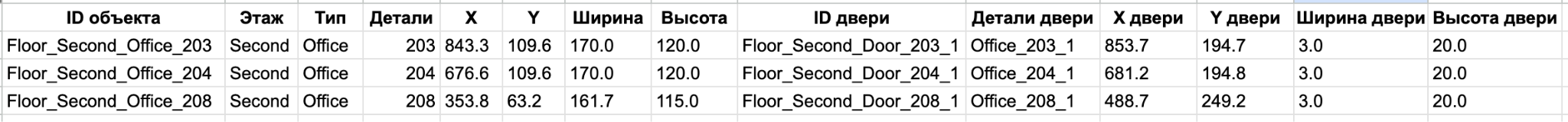


Рисунок 2. Таблица с примерами объектов и их параметрами

## 

## 3.3. ОСОБЕННОСТИ РАЗРАБОТКИ

### 3.3.1. АРХИТЕКТУРНЫЕ ПРИНЦИПЫ

Проект построен на основе модульной архитектуры с использованием принципов разделения ответственности и масштабируемости. Каждый компонент системы отвечает за конкретную часть функциональности.

Ключевые архитектурные принципы:

1. Модульность: Каждый компонент (API, обработка графов, маршрутизация, работа с данными) вынесен в отдельные модули.
2. Инъекция зависимостей (Dependency Injection): Реализация гибкого подключения компонентов для упрощения тестирования и замены модулей.
3. Расширяемость: Система легко адаптируется под новые требования, например, добавление этажей, объектов или источников данных.
4. Производительность: Использование асинхронного программирования (FastAPI) и эффективных алгоритмов работы с графами (NetworkX).
5. Кроссплатформенность: Доступность через веб-интерфейс, что делает систему независимой от операционной системы пользователей.

### 3.3.2. АРХИТЕКТУРА ПРОЕКТА

Проект имеет следующую многослойную архитектуру:

(а) Presentation Layer (Внешний слой)

* FastAPI:
  + Реализует REST API[3], предоставляющее доступ к функционалу системы.
  + Обрабатывает запросы на построение маршрутов, получение карт этажей и объектов.
  + Генерирует автоматическую документацию (Swagger/OpenAPI).
* SVG-планы: Генерация и предоставление SVG-файлов с визуализацией этажей и маршрутов.

(б) Application Layer (Логический слой)

* RouteService: Отвечает за построение маршрутов между объектами.
  + Алгоритмы поиска кратчайших путей.
  + Обработка входных данных (точек начала и конца маршрута).
  + Генерация результатов в виде последовательностей узлов и ребер.

(в) Data Layer (Слой данных)

* GraphRepository:
  + Управляет загрузкой данных (JSON)[6] и построением графа инфраструктуры университета.
  + Предоставляет методы для получения списка объектов, дверей и графовых связей.
* JSON-хранилище[6]:
  + Файл plan\_combined.json содержит описание всех узлов, рёбер и их метаданных.

(г) Utility Layer (Вспомогательный слой)

* SVG Processor: Отвечает за обработку SVG-файлов (выделение этажей, маршрутов, объектов).
* Object Processor: Упрощает работу с объектами и их отображение.

### 3.3.3 КОМПОНЕНТЫ И МОДУЛИ

1. Модуль API (app/api):
   * routes.py:
     + Обрабатывает запросы пользователей (например, получение маршрутов между кабинетами).
     + Генерирует SVG-файлы с выделением маршрутов и объектов.
   * Основные эндпоинты:
     + /floor-plan: Возвращает карту этажа с маршрутом.
     + /objects: Возвращает информацию о доступных объектах.
   * Асинхронное программирование позволяет обрабатывать запросы без блокировки.
2. Модуль Core (app/core):
   * config.py: Управляет конфигурацией приложения, включая загрузку путей и переменных окружения.
3. Модуль Repositories (app/repositories):
   * graph\_repository.py: Загружает данные, строит граф, предоставляет методы для взаимодействия с графом.
4. Модуль Services (app/services):
   * route\_service.py: Реализует алгоритмы поиска маршрутов и обработки графа.
   * svg\_processor.py: Обрабатывает планы этажей, выделяя маршруты и объекты.
   * object\_processor.py: Создает человекочитаемые названия объектов.
5. Модуль Domain (app/domain):
   * models.py: Определяет структуры данных для строгой типизации (например, узлы, рёбра, объекты).
6. CLI-инструмент для обработки SVG:
   * Автоматизирует обновление данных и подготовку SVG-файлов для визуализации.

### 

### 3.3.4. ВЗАИМОДЕЙСТВИЕ КОМПОНЕНТОВ

1. Построение маршрута:
   * Пользователь отправляет запрос на построение маршрута через /floor-plan.
   * RouteService: Получает начальный и конечный ID объектов, находит кратчайший путь в графе.
   * GraphRepository: Обеспечивает доступ к графу и связанным данным.
   * SVG Processor: Генерирует SVG-файл с визуализацией маршрута и возвращает его пользователю.
2. Получение объектов:
   * Запрос на /objects вызывает object\_processor, который формирует список доступных объектов с их описанием.
3. Обработка SVG:
   * CLI-скрипт загружает SVG-файл, анализирует элементы, добавляет уникальные ID для объектов (офисов, лестниц, дверей) и сохраняет обновленную карту.

### 

### 

### 

### 

### 

### 

### 

### 

### 3.3.5. ПОТОКИ ДАННЫХ

1. Исходные данные:
   * JSON-файл с графом и объектами.[6]
   * SVG-файл с планами этажей.
2. Обработка данных:
   * JSON[6] обрабатывается в GraphRepository, преобразуется в граф с узлами и рёбрами.
   * SVG-файлы обрабатываются с использованием svg\_processor.
3. Результат:
   * Пользователь получает интерактивный SVG-файл с маршрутом или данные о доступных объектах.

Архитектура проекта организована таким образом, чтобы поддерживать масштабируемость, гибкость и производительность.

# ГЛАВА 4. ФУНКЦИОНАЛЬНОСТЬ И ТЕСТИРОВАНИЕ

## 4.1. ФУНКЦИОНАЛЬНЫЕ ВОЗМОЖНОСТИ ПРОЕКТА

Проект реализует систему внутренней навигации для университетского корпусов. Функционал системы ориентирован на решение задач поиска объектов, построения маршрутов и предоставления информации о корпусах.

Ключевые функциональные возможности:

1. Поиск объектов:
   * Пользователь может искать помещения (кабинеты, лестницы, зоны отдыха) по их названию или уникальному идентификатору.
   * Отображение полной информации об объекте, включая его расположение (этаж, координаты на карте).
2. Расчет топ-3 кратчайших маршрутов:
   * Система предоставляет до трех маршрутов между двумя заданными точками, каждый с оценкой веса (например, длины или времени).
3. Интерактивная карта корпуса:
   * Отображение детализированных планов этажей корпуса.
   * Подсветка маршрута между точками на основе выбранного пути.
   * Возможность просмотреть информацию об объектах, щелкнув на элементы карты (например, аудиторию или дверь).
4. Получение информации об объектах:
   * API возвращает список всех объектов на этаже с их названиями и типами (например, кабинет, лестница, выход).
   * Удобная структура данных позволяет интеграцию с внешними системами (например, мобильными приложениями).

Система предоставляет набор функциональных возможностей для решения задач навигации внутри университетских корпусов.

## 4.2. ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС ПРОЕКТА

Пользовательский интерфейс проекта представляет собой систему навигации внутри здания, обеспечивающую выбор начальной и конечной точек маршрута, а также визуализацию пути на схематическом плане этажа.

### 4.2.1. ОБЩИЙ ВИД ИНТЕРФЕЙСА

На рисунке 2 (см. Приложение 1) представлен основной интерфейс системы. Левая панель содержит выпадающие списки с заголовками «Откуда?» и «Куда?», позволяющие пользователю выбрать начальный и конечный кабинеты. Центральная часть отображает схему этажа с различными зонами, обозначенными оттенками синего цвета. Правая панель включает вертикальный список кнопок для переключения между этажами, где текущий этаж выделен визуально.

### 4.2.2. ВЫБОР КАБИНЕТОВ И ОТОБРАЖЕНИЕ МАРШРУТА

При выборе пользователем обоих кабинетов, например, «Кабинет 422-1 (4-ШКН)» и «Кабинет 415-а (4-ШКН)», интерфейс обновляется, как показано на рисунке 3 (см. Приложение 2). На схеме этажа появляется зелёная линия, соединяющая выбранные кабинеты, что наглядно демонстрирует маршрут. Активный этаж по-прежнему выделен на правой панели.

### 4.2.3 ПОИСК КАБИНЕТОВ ЧЕРЕЗ ВЫПАДАЮЩИЙ СПИСОК

Функциональность поиска кабинетов реализована через выпадающий список, как показано на рисунке 4 (см. Приложение 3). Пользователь вводит номер кабинета в поле поиска, после чего отображается список соответствующих результатов с номерами кабинетов и дополнительной информацией, такой как «(4-ШКН)». Элементы списка снабжены иконками, указывающими на тип помещения. Центральная часть интерфейса при этом отображает схему соответствующего этажа.

## 4.3. ТЕСТИРОВАНИЕ ПРОЕКТА

Тестирование проводилось в корпусе УЛК-05 Тюменского государственного университета с участием первокурсников направления МОАИС. Цель тестирования заключалась в оценке эффективности разработанного решения для целевой аудитории.

Организация тестирования

Тестирование осуществлялось в реальных условиях с соблюдением следующих этапов:

1. Задачи тестирования:

• Оценить функциональность сайта проекта.

• Определить уровень удобства использования интерфейса.

• Собрать обратную связь по эмоциональному восприятию.

2. Процесс тестирования:

• Участники переходили на сайт проекта, изучали его функционал и проверяли основные возможности, включая построение маршрутов.

• После взаимодействия с проектом участники заполняли анкету, содержащую ключевые критерии оценки:

• Скорость построения маршрута.

• Точность построения маршрута.

• Удобство интерфейса.

• Эмоциональный отклик.

Результаты тестирования

На основании полученной обратной связи от 48 студентов результаты были систематизированы по следующим параметрам:

1. Скорость построения маршрута:

• Среднее время отклика системы составило 0,4 секунды, что 90% участников оценили как “отлично”.

2. Точность построения маршрута:

• Точность построенных маршрутов составила 87%.

• При этом 15% участников отметили незначительные отклонения от реальных условий, которые не мешали достижению цели.

3. Удобство интерфейса:

• 85% участников оценили интерфейс как “удобный” или “очень удобный”.

• 10% столкнулись с трудностями в навигации.

• 5% предложили улучшить визуальное оформление.

4. Эмоциональный отклик:

• 45% участников выразили положительные эмоции при работе с приложением.

• 50% сообщили о нейтральных эмоциях.

• 5% испытывали дискомфорт, связанный с непривычным функционалом.

Результаты тестирования подтверждают гипотезу об эффективности разработанного решения для целевой аудитории. Высокая скорость работы, приемлемая точность маршрутов и положительная оценка удобства интерфейса демонстрируют потенциал проекта. Обратная связь от участников будет учтена для дальнейшего улучшения функционала и визуального оформления.

# ЗАКЛЮЧЕНИЕ

В рамках проекта была разработана системы внутренней навигации для университетских корпусов, которая включает в себя функционал интерактивной карты, построения маршрутов и предоставления информации о кабинетах и других объектах.

Решенные проблемы

1. Упрощение навигации:

Система позволяет студентам, преподавателям и гостям университета быстро находить необходимые помещения, что особенно полезно для новичков, не знакомых с инфраструктурой корпусов.

2. Доступность актуальной информации:

Используемый подход с JSON-хранилищем[6] и модульной архитектурой дает возможность оперативно обновлять данные при изменении инфраструктуры (например, ремонтах, переездах кабинетов).

3. Кроссплатформенность:

Система доступна на устройствах с разными операционными системами благодаря веб-интерфейсу, что делает ее универсальной и удобной для всех категорий пользователей.

Достигнутые результаты

1. Техническая реализация:

• Построена надежная архитектура на основе современных технологий: FastAPI, NetworkX[2], SVG-обработка.

• Реализован алгоритм маршрутизации с использованием теории графов, обеспечивающий точное определение маршрутов.

2. Простой интерфейс:

Создан удобный и понятный пользовательский интерфейс с элементами интерактивной карты, визуализацией маршрутов и доступом к информации об объектах.

3. Положительная обратная связь:

Тестирование среди студентов первого курса показало высокую эффективность и удобство использования системы. Средний отклик системы составляет 0,4 секунды, а точность маршрутов — 87%.

Возможности для дальнейшего развития

1. Интеграция с мобильными приложениями:

Реализация приложений для iOS и Android, чтобы обеспечить нативную работу на мобильных устройствах.

2. Расширение функционала:

• Добавление возможности автоматического определения местоположения пользователя с использованием Wi-Fi или Bluetooth-маяков.

• Реализация маршрутов для особых случаев, например, эвакуационных путей.

3. Улучшение визуализации:

Создание 3D-моделей кампусов для более точной и наглядной навигации.

Таким образом, разработанная система отвечает актуальным потребностям университета и обладает значительным потенциалом для дальнейшего совершенствования. Проект создает предпосылки для цифровизации инфраструктуры и улучшения взаимодействия пользователей с пространством университетских корпусов.

# СПИСОК ЛИТЕРАТУРЫ

1. Теория построения кратчайшего пути // Алгоритм Йена URL: http://acm.math.spbu.ru/~sk1/mm/au-download/14f-conspect/2014-12-16-Yen-AStar.pdf (дата обращения: 19.12.2024).
2. Поиск кратчайшего маршрута // NetworkX URL: https://networkx.org (дата обращения: 12.12.2024).
3. Реализация RestAPI // FastAPI URL: https://fastapi.tiangolo.com (дата обращения: 19.12.2024).
4. Примеры успешных реализаций подобных систем // УрФУ URL: https://how-to-navigate.ru/institute/isa (дата обращения: 19.12.2024).
5. Примеры успешных реализаций подобных систем // ИТМО URL: https://en.itmo.ru/en/map/map.htm (дата обращения: 19.12.2024).
6. Формат данных // JSON URL: https://www.json.org/json-en.html (дата обращения: 19.12.2024).
7. Алгоритм на графах // Алгоритм Дейкстры URL: https://ru.algorithmica.org/cs/shortest-paths/dijkstra/ (дата обращения: 19.12.2024).
8. Поиск в ширину // BFS URL: https://ru.algorithmica.org/cs/shortest-paths/bfs/ (дата обращения: 19.12.2024).

# Приложение 1

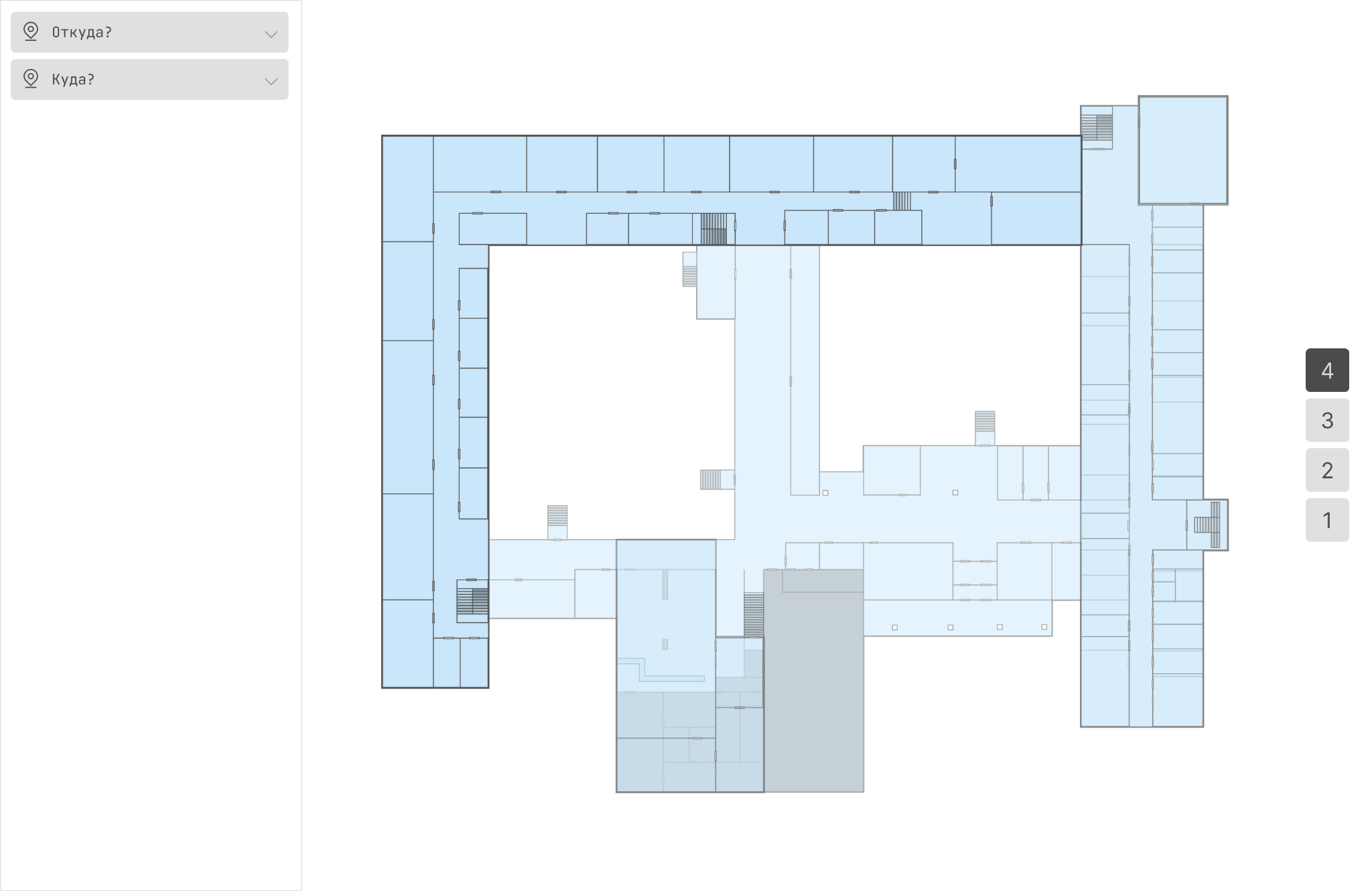


Рисунок 2. Общий вид интерфейса системы навигации.

# Приложение 2

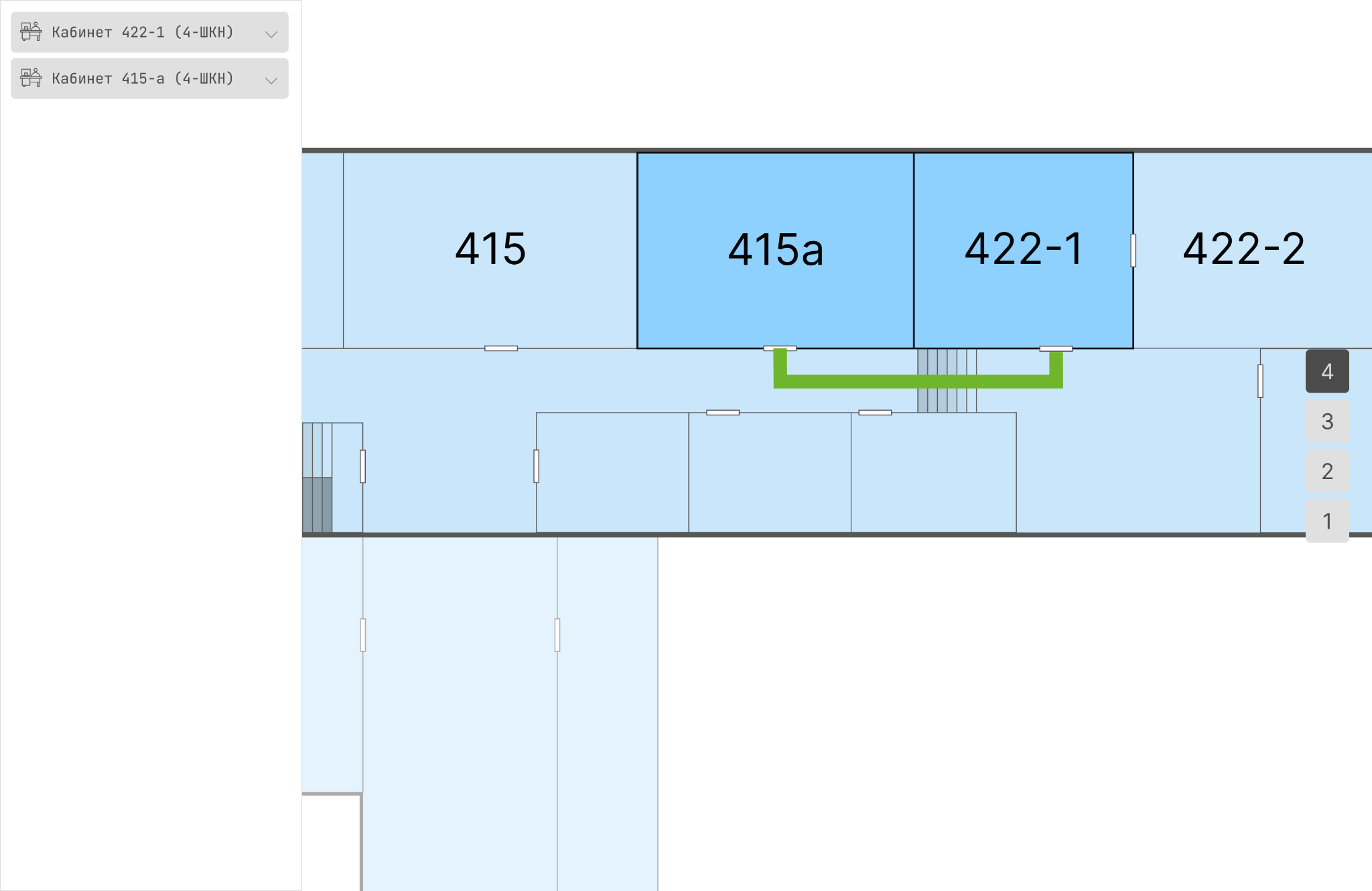


Рисунок 3. Интерфейс с отображением выбранных кабинетов и маршрута.

# Приложение 3

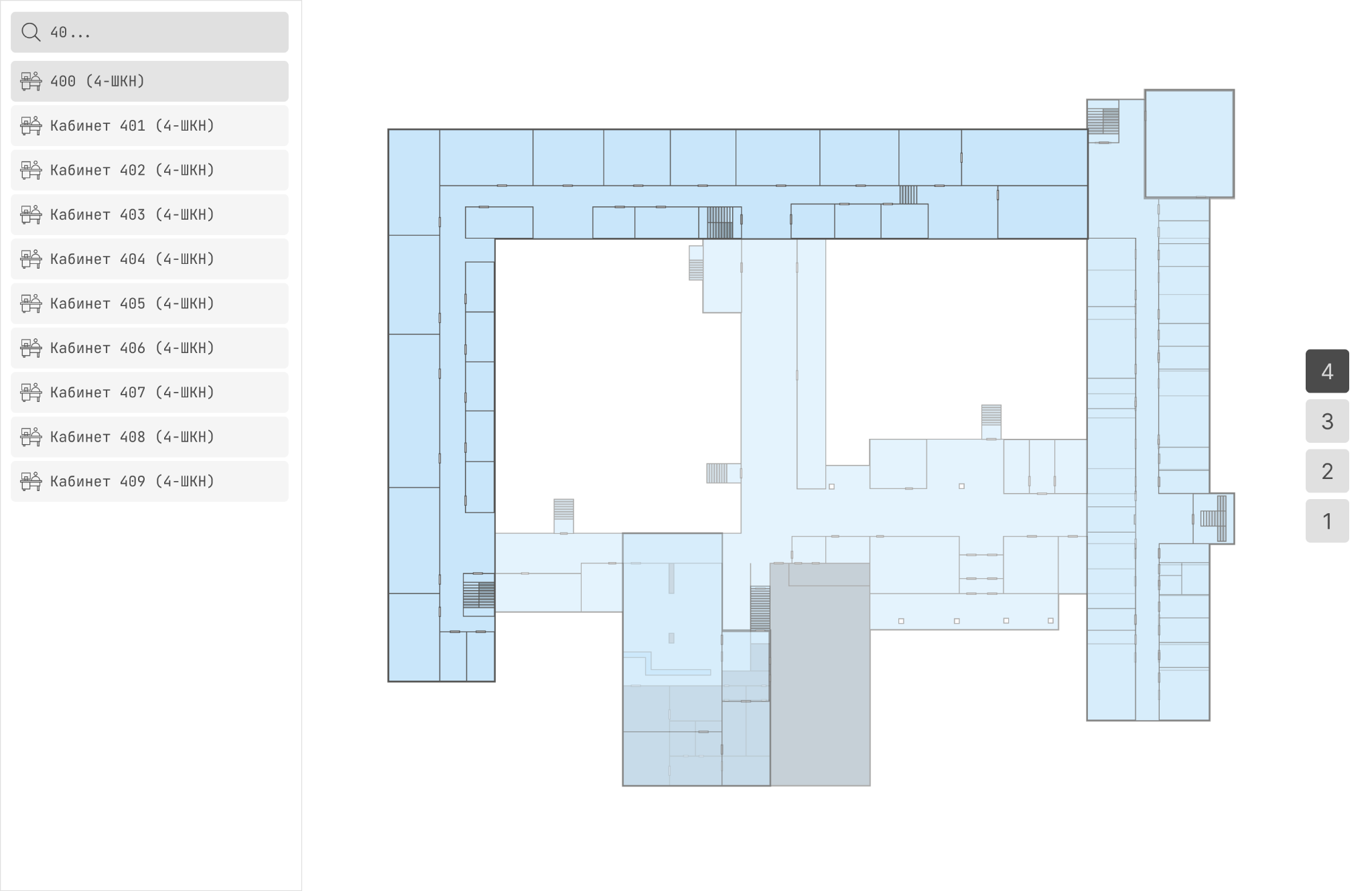


Рисунок 4. Выпадающий список поиска кабинетов.

# Приложение 4

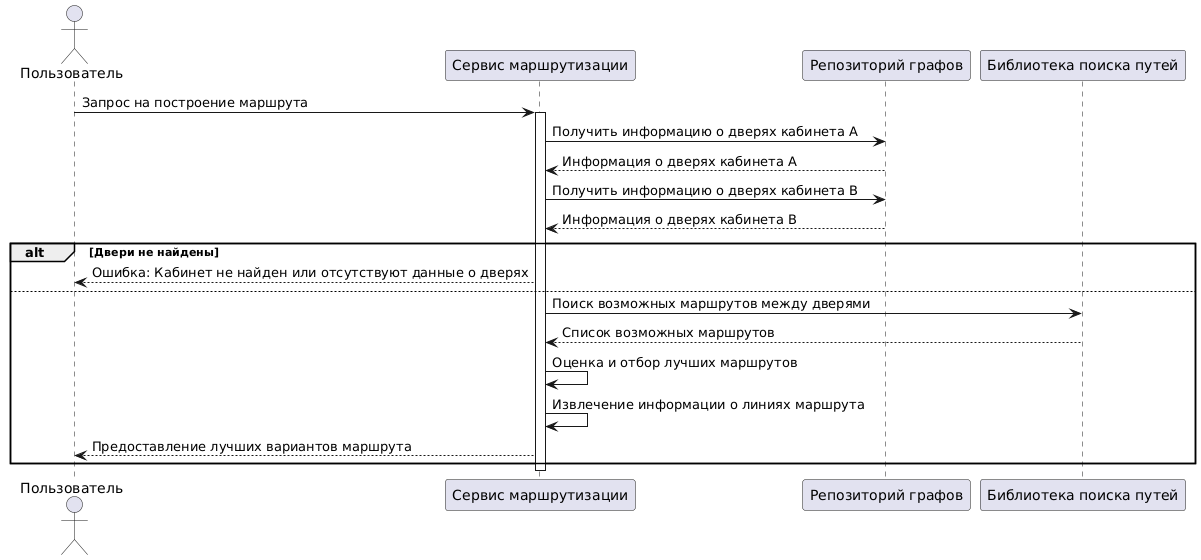


Рисунок 5. Диаграмма последовательности процесса построения маршрута