

Федеральное государственное образовательное бюджетное учреждение высшего образования

**Финансовый университет при Правительстве Российской Федерации**

Департамент анализа данных, принятия решений и финансовых технологий

Междисциплинарная курсовая работа на тему:

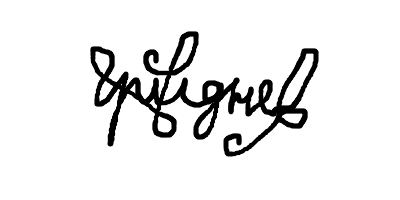
**Решение задач поиска данных с помощью графовой базы данных**

Выполнил:

Студент группы ПИ18-1

Факультета прикладной математики

и информационных технологий

**Гриднев Дмитрий Владимирович**,

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Руководитель:

к.э.н., доцент

**Макрушин Сергей Вячеславович**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Итоговая оценка)

# Содержание

[Содержание 2](#_Toc39778782)

[Введение 3](#_Toc39778783)

[Используемые технологии 3](#_Toc39778784)

[Описание работы системы 5](#_Toc39778785)

[Пример работы системы 5](#_Toc39778786)

[Визуализация здания и этажей 8](#_Toc39778787)

[Формат данных здания 8](#_Toc39778788)

[Архитектура системы 11](#_Toc39778789)

[Backend 12](#_Toc39778790)

[Документация Swagger и Route Backend-а 13](#_Toc39778791)

[Использование ArangoDB в Backend 14](#_Toc39778792)

[Frontend 15](#_Toc39778793)

[Развертывание системы 16](#_Toc39778794)

[Заключение 19](#_Toc39778795)

[Список использованных источников 20](#_Toc39778796)

[Код программной системы 21](#_Toc39778797)

[Deploy 21](#_Toc39778798)

[Frontend 24](#_Toc39778799)

[Backend 47](#_Toc39778800)

# Введение

**Почему данная задача актуальна**

В нашем мире уже привыкли к навигации в повседневной жизни. Мы ежедневно используем онлайн карты. В ней мы можем поглядеть, где сейчас едет нужный нам автобус, как лучше проехать от одного адреса к другому, построить кратчайший маршрут в метро или найти ближайший супермаркет. Это несомненно удобно, но есть сферы, которые карты не охватывают. Одной из таких сфер является план здания. Планы здания обычно мы можем встретить висящие на стене. Данные схемы малоинформативный обычному человеку, по ним долго искать информацию, если карта большая возможны сложности с пониманием как пройти из одной точки к другой. В качестве примера таких зданий можно взять главный корпус Финансового университета при Правительстве Российской Федерации, находящийся около метро Аэропорт. Данный корпус состоит из нескольких зданий, у него сложная для понимания людей планировка, сложное нумерование аудиторий. Для таких зданий можно построить интерактивный план, при помощи которого человек сможет найти необходимую аудиторию и узнать, как, и за какой время можно попасть в неё.

Данная работа является одной из возможной реализации данной системы.

# Используемые технологии

**На чем работает разработанная система**

Архитектура системы построена по принципу Separation of concerns[[1]](#footnote-1) – принцип разделения ответственностей.

На каждую из трех частей системы ложится разная ответственность:

1. Многомодельная система баз данных с открытым исходным кодом ArangoDB[[2]](#footnote-2).
2. Front-end ­ – клиентская сторона системы работающая на JavaScript. По сути, это интерфейс, который видит пользователь у себя в браузере.
   * Svelte[[3]](#footnote-3) – библиотека на которой построена вся клиентская сторона. Svelte не использует виртуальное DOM дерево, а собирает приложение в нативный JavaScript код. Благодаря этому система имеет маленький вес и максимальную производительность.
   * Rollup[[4]](#footnote-4) – библиотека ответственная за сборку Svelte в единое приложение.
3. Back-end – часть системы, которую не видит пользователь, но с ней взаимодействует Front-end.
   * Python 3.8[[5]](#footnote-5) – язык на котором работает весь Back-end.
   * FastAPI[[6]](#footnote-6) – web библиотека, которая обрабатывает HTTP запросы от front-end. Работает на Starlette.
   * Python-Arango[[7]](#footnote-7) – библиотека для взаимодействия с базой ArangoDB.
   * Uvicorn[[8]](#footnote-8) – быстрый ASGI сервер использующий uvloop. Используется для запускать копий FastAPI на разных ядрах.
   * ujson[[9]](#footnote-9) – быстрый JSON encoder/decoder.
   * pydantic[[10]](#footnote-10) – валидатор данных использующий Type Annotation из Python.

Инструменты для развертывания системы

Docker[[11]](#footnote-11) – программное обеспечение, позволяющее работать с контейнерами. Используется для развертывания сервисов, требующихся для работы разработанной системы.

NPM[[12]](#footnote-12) – менеджер пакетов, использующийся для загрузки пакетов.

PIP[[13]](#footnote-13) – система управления библиотека Python-а. Используется для установки библиотек, требующихся back-end-у.

# Описание работы системы

**Как работает разработанная система**

## Пример работы системы

Как говорилось ранее, данная система занимается отрисовкой плана здания с возможностью построения маршрутов между объектами здания. Для того, чтобы понимать, о чем идет речь, думаю стоит начать с просмотра разработанной системы.

При старте веб-сайта мы видим схему здания (Рис. 1). На ней мы можем главную, левую часть сайт, на которой расположено название здания, выбранный этаж, сам план здания по этажам и декоративная подложка под планом здания. По мимо главной части, у сайта есть Sidebar[[14]](#footnote-14), в котором есть возможность осуществить поиск по всему зданию.

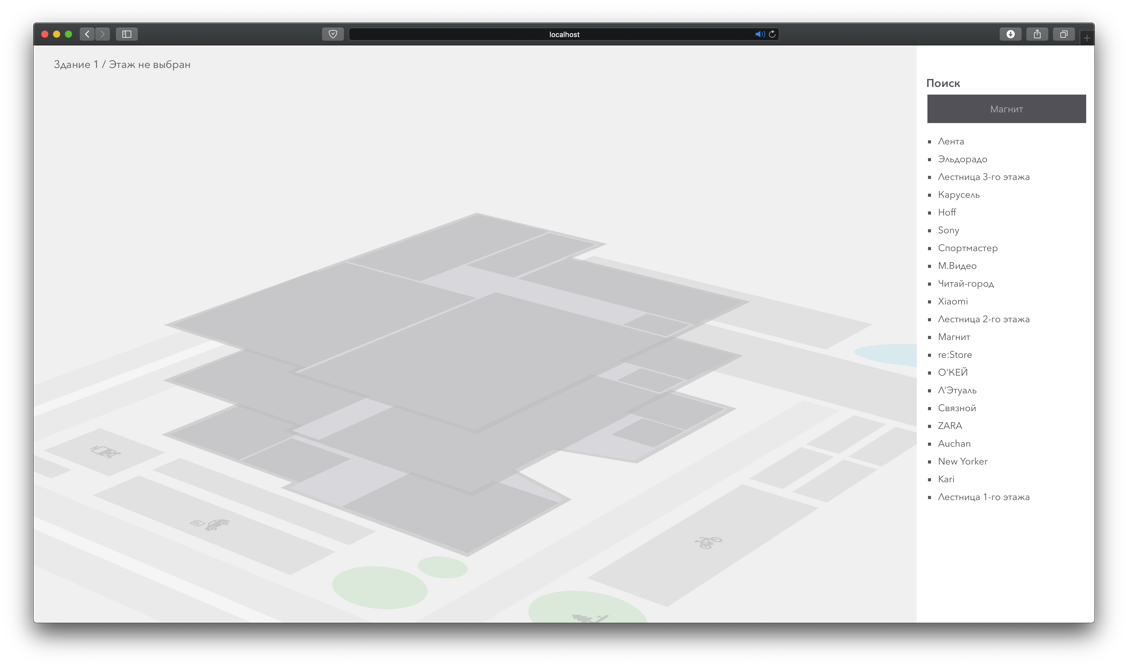
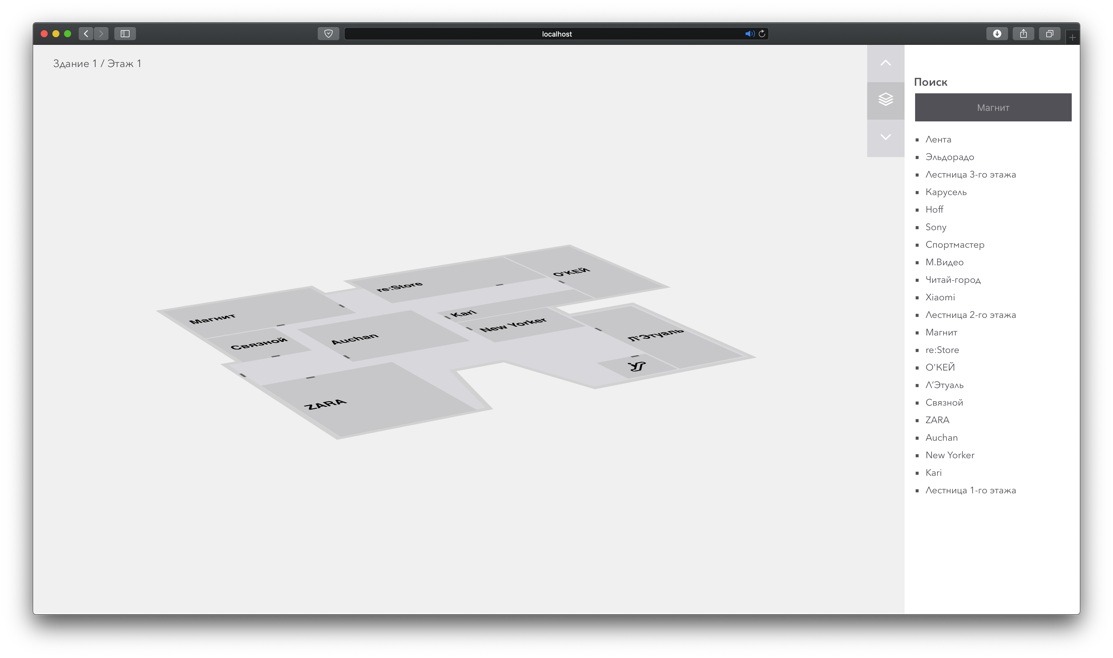


Рисунок 1. Схема здания у разработанной системы.

Если во время отображения схемы здания нажать на какой-то этаж, откроется его план (Рис. 2). На этом экране справа вверху главной части сайта добавляется новый элемент навигации, позволяющий переключаться между этажами или вернуться к схеме здания. На схеме этажа отображаются объекты, находящиеся на этаже. Каждый объект может иметь входы/выходы в него.

Рисунок 2. План этажа здания у разработанной системы.



Объект может быть как интерактивным, так и нет. Если объект интерактивен, можно на него нажать, при этом, как и в случае с нажатием на название объекта в поиске, откроется описание объекта (Рис. 3) в Sidebar. У каждого объекта есть название, этаж, на котором он располагается, краткое описание, возможность выбрать схему построения маршрута “Отсюда” или “Сюда”, и при наличии сайта, возможность перейти на него.

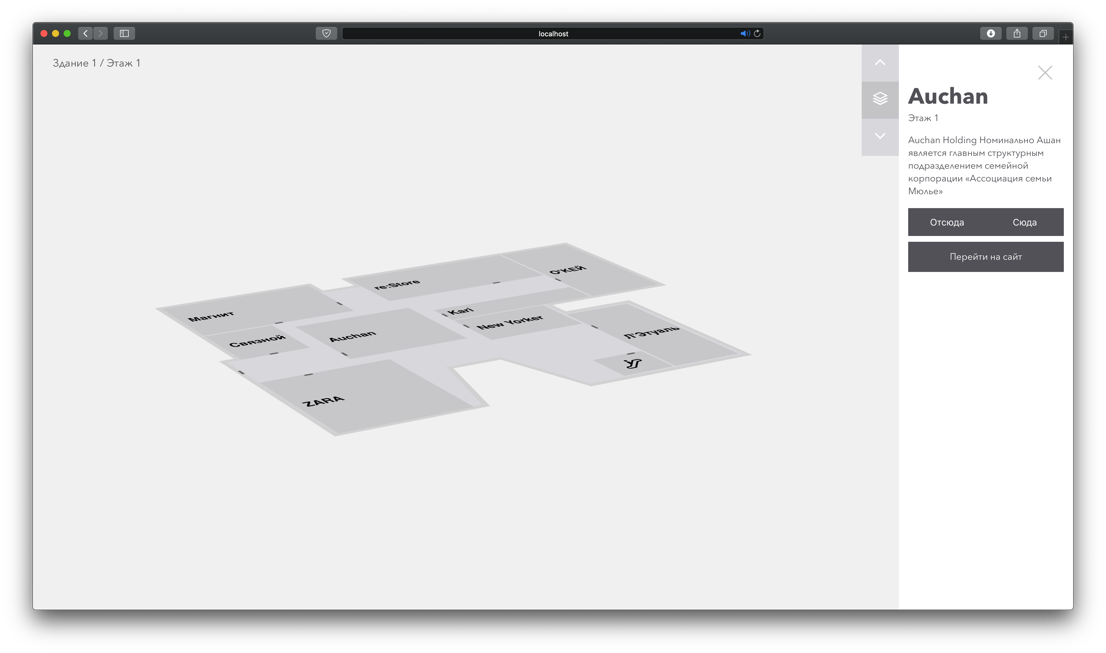


Рисунок 3. Система с выбранным объектом.

Если выбрать 2 объекта, при этом у первого нажать кнопку “Отсюда”, а у второго “Сюда”, построится маршрут от первого объекта ко второму (Рис. 4). При этом в Sidebar появятся информации о построенном объекте.

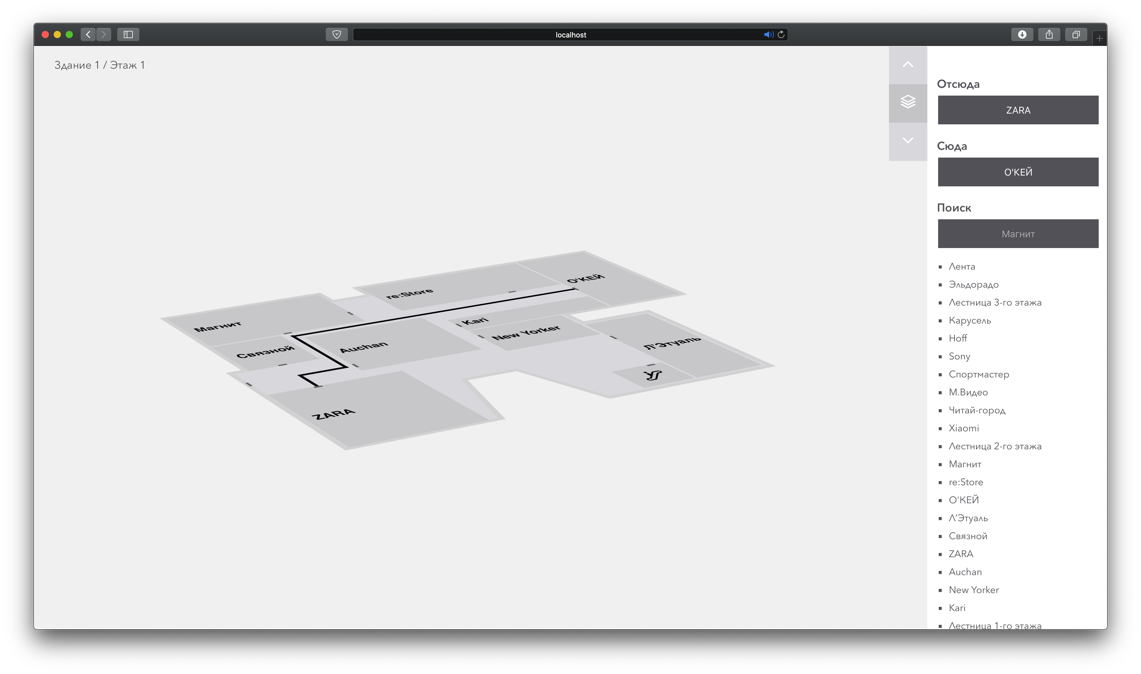


Рисунок 4. Система с построенным маршрутом.

## Визуализация здания и этажей

Когда появилась идея визуализировать данные здания, первый вопрос, который возник: “С помощью каких инструментов это сделать”. Инструментов для этого не так уж и много.

* WebGL[[15]](#footnote-15). Наверно лучший вариант визуализации сложной 3D-графики. Данный подход используется для визуализации карт 2GIS, у них даже есть в открытом доступе библиотека 2gl[[16]](#footnote-16), используемая в их картах. WebGL, хоть и является эталонным решением для 3D-графики, но для его использовании потребуется написать слишком много абстракции, которая не слишком то и нужна данной системе.
* Canvas/SVG. Сегодня браузер вместе с CSS позволяет перемещать объекты в пространстве, при этом используя стандартные CSS анимации во время перемещения. При этом не требуется разбираться с WebGL, достаточно понимать трехмерную систему координат. Выбор был сделан в пользу SVG, этому есть две причины:
  1. В статье “Новый интерфейс Яндекс.Метро и технологии, с помощью которых он работает”[[17]](#footnote-17) на Хабр приведено сравнение скорости отрисовки объектов с помощью Canvas/SVG (Табл. 1). В этом сравнении SVG сильно опережает Canvas.
  2. SVG можно редактировать в программах для векторной графики, например в Adobe Illustrator. Там удобно добавлять на схему объекты, или рисовать декоративную подложку под схему здания.

Таблица 1. Сравнение скорости отрисовки Canvas и SVG в Google Chrome.

|  |  |  |  |
| --- | --- | --- | --- |
| Technology | Time to download | Time to draw | Overall time |
| Canvas | 34.58 ms | 181.72 ms | 216.30 ms |
| SVG | 28.02 ms | 74.02 ms | 102.07 ms |

## Формат данных здания

Весь веб-сайт, включая SVG схему рисуется с помощью JavaScript библиотеки Svelte. Схема здания описана в формате JSON[[18]](#footnote-18). Структура, используемая в программе:

{

"title": "string", **Название здания**

"designation": "string", **ID здания на английском. Используется для построения путей в ArangoDB и формирования ссылок с помощью FastAPI**

"background": "string", **Ссылка на SVG декоративную подложку под схему здания**

"layers": [ **Список этажей здания**

{

"id": 1, **Идентификатор, по сути, номер этажа**

"objects": [ **Объекты находящиеся на этаже. В текущей реализации их 3 типа. У каждого объекта может быть marker – id описания объекта. При их наличии объект становится интерактивным. У кажого объекта есть дополнительные поля, требующиеся для их рисования в браузере.**

{

"type": "rect", **Прямоугольник используется для рисования**

"id": "string",

"x": 0,

"y": 0,

"width": 0,

"height": 0,

"marker": "string"

},

{

"type": "path", **Путь используется для обводки подложки под этаж**

"id": "string",

"d": "string",

"marker": "string"

},

{

"type": "polygon", **Многоугольник** **используется для рисования подложки**

"id": "string",

"points": "string",

"marker": "string"

}

],

"entrances": [ **Входы/выходы**

{

"type": "string",

"id": "string",

"x1": 0,

"y1": 0,

"x2": 0,

"y2": 0

}

],

"points": [ **Точки пересечения путей. Через них строятся пути**

{

"type": "string",

"id": "string",

"cx": 0,

"cy": 0

}

],

"routes": [ **Пути. Используются для отрисовки путей во fron-end-е и связи объектов в ArangoDB**

{

"type": "line", **Путь**

"id": "string",

"p1": "string", **Используется ArangoDB для связи объектов по id**

"p2": "string", **Используется ArangoDB для связи объектов по id**

"time": 0, **Время. Может использоваться ArangoDB для вычисления самого быстрого маршрута.**

"x1": 0,

"y1": 0,

"x2": 0,

"y2": 0

},

{

"type": "entrance\_line", **Внутренний путь между входом и объектом. Используется для возможности создания нескольких точек входа**

"id": "string",

"p1": "string", **Используется ArangoDB для связи объектов по id**

"p2": "string" **Используется ArangoDB для связи объектов по id**

}

],

"markers": { **Маркеры – описание объектов. 2 типа**

"id1": {

"title": "string", **Название объекта**

"type": "room", **Комната,**

"description": "string", **Описание объекта**

"site": "string" or “null” **Сайт**

}

"id2": {

"title": "string",

"type": "stairs", **Лестница, служит для перемещения между зданиями, отрисовывается в виде иконки.**

"description": "string",

"site": "string" or “null”

}

}

}

],

"upstairs": [ **Связи между этажами для лестниц**

{

"type": "upstairs\_line",

"id": "string",

"p1": "string", **Используется ArangoDB для связи объектов по id**

"p2": "string", **Используется ArangoDB для связи объектов по id**

"time": 0 **Время. Может использоваться ArangoDB для вычисления самого быстрого маршрута.**

}

]

}

Для удобного построения таких схем у каждого id есть свой формат, но можно использовать и другой формат id.

* **objects**:
  + **id**: “{номер этажа}\_o\_{уникальный id}”
* **entrances**:
  + **id**: “{номер этажа}\_e\_{уникальный id}”
* **points**:
  + **id**: “{номер этажа}\_p\_{уникальный id}”
* **routes**:
  + **type** “line”:
    - **id**: “{номер этажа}\_l\_{уникальный id}”
    - **p1**: id из entrance/points
    - **p2**: id из entrance/points
  + **type** “entrance\_line”:
    - **id**: “{номер этажа}\_el\_{уникальный id}”
    - **p1**: id из entrance/objects
* **markers key**:
  + “{номер этажа}\_m\_{уникальный id}”

## Архитектура системы

Система состоит из трех частей, которые связаны при помощи Docker Compose. Каждая часть системы ответственна за определенную часть работы. Схема архитектуры системы представлена ниже (Рис. 5).

ах

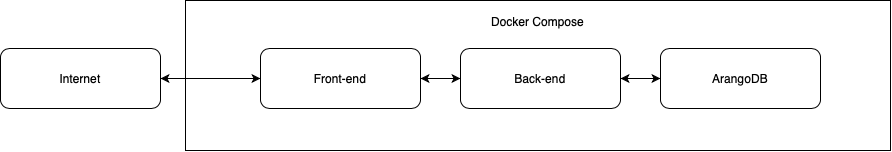


Рисунок 5. Архитектура системы.

Главной частью приложения является база ArangoDB. Она хранит данные необходимые frontend-у. По мимо хранения данных, база ответственная за поиск пути между объектами. Для связи базы ArangoDB и используется backend, построенный на асинхронной библиотеке FastAPI. Backend формирует запросы к API базы ArangoDB, получает ответ и отдает его frontend-у, который из полученных данных создает понятное, обычному человеку, визуальное представление.

## Backend

Главная задача backend-а – взаимодействие с frontend-ом. Их взаимодействие осуществляется с помощью HTTP запросов. Главной библиотекой, на которой работает backend и формируется роутинг является FastAPI.

Запуск осуществляется с помощью библиотек Gunicorn и Uvicorn. В Gunicorn указывается путь к файлу main.py, количество работников, обычно высчитывается количество ядер + 1, и в качестве работников указываем работника Uvicorn.

Команда для запуска

*gunicorn* app.main:app -w 13 -k uvicorn.workers.UvicornWorker

**Иерархия проекта**

**config.py** содержит настройки проекта. Например, адрес документации, ссылка для подключения к базе ArangoDB. Настройки получаются из окружения, в случае их отсутствия используется стандартные.

**main.py** создается инстанс[[19]](#footnote-19) объекта сервера FastAPI, в который подключаются все необходимые для работы модули и библиотеки.

**api/route.py** содержит все роутеры, которые используются в frontend-е.

**core/** содержит модули необходимые для работы backend-а.

**core/exceptions.py** содержит описание ошибок, которые могут возникать во время работы

**core/helpers/py** содержит классы/функции, требующиеся для правильной работы некоторых систем

**core/arangodb.py** содержит функции, которые используются для работой с базой ArangoDB

**database/** содержит данные для создания инстанса работающего с базой ArangoDB.

**database/\_\_init\_\_.py** описывает создание инстанса работающего с базой ArangoDB.

**database/model.py** содержит класс создающий инстанс работающий с базой ArangoDB.

**schemas/** содержит схемы, использующиеся для валидации[[20]](#footnote-20) данных. Схемы описываются с помощью библиотеки pydantic.

## Документация Swagger и Route Backend-а

Для документации роутов используется Swagger. Документация доступна в формате OpenAPI Specification 3[[21]](#footnote-21). Так же в Swagger описаны все схемы используемы в роутах.

Swagger доступен по адресу **/documentation**

Файл с выгрузкой документации в формате OAS3 доступен по адресу: **/openapi.json**

**Роуты из документации**

**GET** /api/buidings – возвращает список зданий загруженных в базу ArangoDB

**GET** /api/{building} – возвращает JSON с информации о здании, используемый в frontend.

Параметры:

**building**: поле “designation” из JSON

**DELETE** /api/{building} – удаляет данные о здании из базы ArangoDB.

Параметры:

**building**: поле “designation” из JSON

**GET** /api/{building}/background – возвращает фон здания в SVG формате.

Параметры:

**building**: поле “designation” из JSON

**PUT** /api/{building}/background – обновляет декоративный здания. Принимает файл в формате SVG.

Возвращает пути, использующиеся для заданного маршрута.

Параметры:

**building**: поле “designation” из JSON

**GET** /api/route/{building}/{of}/{to} – ищет путь по зданию.

Возвращает пути, использующиеся для заданного маршрута.

Поиск пути описан в главе “Использование ArangoDB в Backend”

Параметры:

**building**: поле “designation” из JSON

**of**: id, точки откуда искать путь

**to**: id точки, куда искать путь

**POST** /api/import/ – принимает JSON файл с описанием здания. Записывает все данные в ArangoDB и строит по ним граф. Название графа поле “designation”

Возвращает новый файл без информации, ненужной frontend-у.

## Использование ArangoDB в Backend

Backend занимается формированием ответов для frontend-а, для этого он использует данные из базы ArangoDB.

**Поиск пути**

Для поиска пути используется стандартное API ArangoDB. Указываются \_key объектов, то есть id объектов во frontend, путь между которыми необходимо найти и граф, по которому требуется производить поиск. В системе граф называется полем “designation” JSON файла. Так же запрос с помощью REGEX\_REPLACE удаляет “\_\_2”, используемый для двухсторонней связи точек.

task = *self*.async\_instance.aql.execute(  
 """  
 FOR v, e IN OUTBOUND SHORTEST\_PATH @of TO @to  
 GRAPH @graph  
 RETURN { "connection": v.\_key, "object": REGEX\_REPLACE(e.\_key, "\_\_\_2", "") }  
 """,  
 bind\_vars={"of": of, "to": to, "graph": graph},  
)

**Хранение данных**

По мимо поиска данных в ArangoDB существует таблица “building”, в которой ключ это designation, а данные содержат JSON объект, который backend отдает frontend-у во время запросы схемы здания. Во время импорта этого JSON файла, из него убираются данные, используемые только в backend-е, например связи (ребры графа). Для приведенных в качестве примера данных это экономит порядка 25 килобайт. Учитывая то, что для пользователя первая загрузка сайта, без кэширования, вместе с данными весит около 60 килобайт это экономит почти 50% трафика.

## Frontend

Frontend с помощью JavaScript загружает данные из Backend с помощью HTTP запросов и из полученной информации рисует схему здания.

**Иерархия проекта**

**package.json** содержит настройки проекта для NPM. Так же файл содержит возможные варианты запуска/сборки проекта:

* + **"build": "rollup -c"** – сборка приложения.
  + **"dev": "rollup -c -w"** – запуска в режиме разработки с livereload[[22]](#footnote-22).
  + **"start": "sirv public"** – запуск приложения в обычном режиме.
  + **"start-single":** "rollup -c; sirv public --single" – запуск приложения в Single Page Application[[23]](#footnote-23) режиме.

**rollup.config.js** содержит настройки rollup, используемого для сборки проекта в 1 JavaScript файл.

**/public** содержит статические файлы, изменение/сборка которых не требуется

**/public/build** содержит собранный файл стилей CSS3 и файл с основным кодом JavaScript приложения.

**/public/index.html** это шаблон страницы, в который загружается собранное приложение

**/public/favicon.png** картинка 16x16 px отображаемая во вкладке браузера

**/srс** содержит файлы Svelte и JavaScript, которые необходимы для сборки приложения

**/srс/App.svelte** главный файл библиотеки Svelte, который использует другие компоненты .svelte.

**/srс/main.js** хранит настройки для запуска **/srс/App.svelte**

**/srс/constants.js** содержит неизменяемые переменные приложения. В данном случае переменна одна – это ссылка к backend.

**/srс/store.js** описывает хранилище Svelte, доступное по всему приложению. Создание хранилища реализуется с помощью паттерна[[24]](#footnote-24) “Открытый модуль”[[25]](#footnote-25).

**/srс/animation.js** хранит пользовательские анимации использующие JavaScript.

**/srс/components** хранит компоненты Svelte

**/srс/components/icons** хранит иконки, используемые компонентами Svelte

# Развертывание системы

**Как запустить разработанную систему**

Для развертывания системы требуется установленный Docker.

Система разрабатывалась с использованием GitHub[[26]](#footnote-26), репозиторий[[27]](#footnote-27) [https://github.com/FlymeDllVa/BuildingPlan](https://en.wikipedia.org/wiki/Commit_(version_control)). В GitHub встроен Cl GitHub Actions, который используется для Continues integration[[28]](#footnote-28). Cпомощью GitHub Actions, после каждого коммита[[29]](#footnote-29), происходит сборка каждого компонента системы в свой пакет. Далее готовые пакеты можно использовать напрямую, без необходимости сборки.

Для сборки с использованием готовых пакетов используется файл docker-compose.yml. Для запуска требуется в папке с проектом выполнить команду “docker-compose up”. Docker сам скачает необходимые пакеты и запустит систему.

version: '3.7'  
services:  
 arangodb:  
 image: arangodb/arangodb  
 ports:  
 - 8529:8529  
 environment:  
 - ARANGO\_NO\_AUTH=1  
 - DB\_LINK\_PORT\_8529\_TCP\_ADDR=172.17.1.1  
 - DB\_LINK\_PORT\_8529\_TCP\_PORT=8529  
 volumes:  
 - ./docker/arangodb:/var/lib/arangodb3  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.1  
 backend:  
 image: docker.pkg.github.com/flymedllva/buildingplan/buildingplan-backend  
 command: gunicorn app.main:app -w 6 -k uvicorn.workers.UvicornWorker --bind=0.0.0.0  
 ports:  
 - 8000:8000  
 environment:  
 ARANGODB\_DATABASE\_NAME: "\_system"  
 ARANGODB\_DATABASE\_URL: "http://172.27.1.1:8529"  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.2  
 frontend:  
 image: docker.pkg.github.com/flymedllva/buildingplan/buildingplan-frontend  
 command: npm start  
 ports:  
 - 80:5000  
networks:  
 project\_net:  
 ipam:  
 driver: default  
 config:  
 - subnet: 172.27.0.0/16

Если же вы хотите собрать файлы проекта самостоятельно воспользуетесь файлом docker-compose.dev.yml.

version: '3.7'  
services:  
 arangodb:  
 image: arangodb/arangodb  
 ports:  
 - 8529:8529  
 environment:  
 - ARANGO\_NO\_AUTH=1  
 volumes:  
 - ./docker/arangodb:/var/lib/arangodb3  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.1  
 backend:  
 build: backend  
 command: gunicorn app.main:app -w 6 -k uvicorn.workers.UvicornWorker --bind=0.0.0.0  
 ports:  
 - 8000:8000  
 environment:  
 ARANGODB\_DATABASE\_NAME: "\_system"  
 ARANGODB\_DATABASE\_URL: "http://172.27.1.1:8529"  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.2  
 frontend:  
 build: frontend  
 command: npm start  
 ports:  
 - 80:5000  
networks:  
 project\_net:  
 ipam:  
 driver: default  
 config:  
 - subnet: 172.27.0.0/16

После запуска системы, для ее работы требуется загрузить данные о вашем здании при помощи POST HTTP запроса /api/import с designation равным building\_1, так же, если требуются у этого здания необходимо обновить фон при помощи PUT HTTP запроса /api/building\_1/background.

# Заключение

**Для каких целей может применяться данная система**

Главным преимуществом ArangoDB является возможность автоматического построения графов между объектами. API базы позволил быстро спроектировать систему, способную эффективно хранить, обрабатывать и использовать данные для построения схемы здания. От пользователя, который будет использовать разработанную систему требуется только лишь предоставить файл в необходимом формате, а база решит проблему поиска среди данных кратчайшего пути. Это лишь 1 сценарий использования базы.

Разработанная система при малых затратах с легкостью нарисует схему и позволит искать по ней кратчайший маршрут. Данную систему можно использовать в зданиях со сложным проектированием, например в: торговых центрах, учебных заведениях.

# Список использованных источников

**Информационные источники, используемые при написании программной системы**

Гриднев Д.В., Иванов М.Н., Кирилкин В.А. Разработка ботов ВКонтакте и телеграм для расписания университета // XIII Международная научно-практическая конференция «Новые информационные технологии в образовании и науке НИТО 2020»: Материалы. – Екатеринбург: ФГАОУ ВПО «Рос. гос. проф.-пед. ун-т», 2020 –С. 35-42. (819 с.) / (9820 п.з., 0,25 п.л.)

Сайт базы ArangoDB. – URL: [arangodb.org](http://arangodb.org/)

Официальная документация к базе ArangoDB. – URL: [arangodb.com/documentation/](http://arangodb.com/documentation/)

Сайт библиотеки Svelte. – URL: <svelte.dev>

Сайт библиотеки Rollup. – URL: [rollupjs.org](http://rollupjs.org/)

Сайт языка программирования Python. – URL: [python.org](https://www.python.org/)

Официальная документация к языку программирования Python. – URL: [docs.python.org/3/](http://docs.python.org/3/)

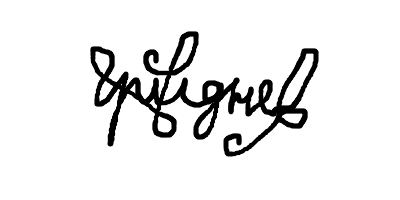
Сайт библиотеки FastAPI. – URL: [fastapi.tiangolo.com](https://fastapi.tiangolo.com/)

Сайт библиотеки Python-Arango. – URL: [python-driver-for-arangodb.readthedocs.io](http://python-driver-for-arangodb.readthedocs.io/)

Сайт ASGI сервера Uvicorn. – URL: [uvicorn.org](http://uvicorn.org/)

Сайт библиотеки ujson. – URL: [pypi.org/project/ujson/](https://pypi.org/project/ujson/)

Сайт библиотеки pydantic. – URL: [pydantic-docs.helpmanual.io/](https://pydantic-docs.helpmanual.io/)

Сайт программного обеспечения Docker. – URL: [docker.com](https://www.docker.com/)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

# Код программной системы

## Deploy

**,github/workflows/dockerimage.yml**

name: Docker Image CI  
  
on:  
 push:  
 branches: [ master ]  
 pull\_request:  
 branches: [ master ]  
  
jobs:  
  
 build:  
  
 runs-on: ubuntu-latest  
  
 steps:  
 - uses: actions/checkout@v1  
 - name: Build the Backend Docker image  
 uses: elgohr/Publish-Docker-Github-Action@master  
 with:  
 name: flymedllva/buildingplan/buildingplan-backend  
 username: ${{ secrets.DOCKER\_GITHUB\_USERNAME }}  
 password: ${{ secrets.GH\_TOKEN }}  
 registry: docker.pkg.github.com  
 workdir: backend  
 - name: Build the Frontend Docker image  
 uses: elgohr/Publish-Docker-Github-Action@master  
 with:  
 name: flymedllva/buildingplan/buildingplan-frontend  
 username: ${{ secrets.DOCKER\_GITHUB\_USERNAME }}  
 password: ${{ secrets.GH\_TOKEN }}  
 registry: docker.pkg.github.com  
 workdir: frontend  
 - uses: actions/checkout@v2  
 - name: Build the Stack  
 run: docker-compose up -d

**docker-compose.yml**

version: '3.7'  
services:  
 arangodb:  
 image: arangodb/arangodb  
 ports:  
 - 8529:8529  
 environment:  
 - ARANGO\_NO\_AUTH=1  
 - DB\_LINK\_PORT\_8529\_TCP\_ADDR=172.17.1.1  
 - DB\_LINK\_PORT\_8529\_TCP\_PORT=8529  
 volumes:  
 - ./docker/arangodb:/var/lib/arangodb3  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.1  
 backend:  
 image: docker.pkg.github.com/flymedllva/buildingplan/buildingplan-backend  
 command: gunicorn app.main:app -w 6 -k uvicorn.workers.UvicornWorker --bind=0.0.0.0  
 ports:  
 - 8000:8000  
 environment:  
 ARANGODB\_DATABASE\_NAME: "\_system"  
 ARANGODB\_DATABASE\_URL: "http://172.27.1.1:8529"  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.2  
 frontend:  
 image: docker.pkg.github.com/flymedllva/buildingplan/buildingplan-frontend  
 command: npm start  
 ports:  
 - 80:5000  
networks:  
 project\_net:  
 ipam:  
 driver: default  
 config:  
 - subnet: 172.27.0.0/16

**docker-compose.dev.yml**

version: '3.7'  
services:  
 arangodb:  
 image: arangodb/arangodb  
 ports:  
 - 8529:8529  
 environment:  
 - ARANGO\_NO\_AUTH=1  
 volumes:  
 - ./docker/arangodb:/var/lib/arangodb3  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.1  
 backend:  
 build: backend  
 command: gunicorn app.main:app -w 6 -k uvicorn.workers.UvicornWorker --bind=0.0.0.0  
 ports:  
 - 8000:8000  
 environment:  
 ARANGODB\_DATABASE\_NAME: "\_system"  
 ARANGODB\_DATABASE\_URL: "http://172.27.1.1:8529"  
 networks:  
 project\_net:  
 ipv4\_address: 172.27.1.2  
 frontend:  
 build: frontend  
 command: npm start  
 ports:  
 - 80:5000  
networks:  
 project\_net:  
 ipam:  
 driver: default  
 config:  
 - subnet: 172.27.0.0/16

**frontend/Dockerfile**

#################################################################

####################### TARGET STAGE ############################

#################################################################

FROM node:12-alpine

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 5000

**backend/Dockerfile**

##################################################################

######################## BUILD STAGE #############################

##################################################################

FROM snakepacker/python:all AS builder

RUN python3.8 -m venv /usr/share/python3/app

ADD requirements.txt /tmp/

RUN /usr/share/python3/app/bin/pip install -U pip setuptools wheel

RUN /usr/share/python3/app/bin/pip install -Ur /tmp/requirements.txt

RUN apt-get update && \

apt-get install python-dev

RUN apt-get install -y gunicorn

RUN apt-get install -y python-gevent

#################################################################

####################### TARGET STAGE ############################

#################################################################

# Use the image version used on the build stage

FROM snakepacker/python:3.8

# Copy virtualenv to the target image

COPY --from=builder /usr/share/python3/app /usr/share/python3/app

# Copy app to image

ADD app /app

RUN chmod +x ./app

# Create a symlink to pip modules

RUN ln -snf /usr/share/python3/app/bin/gunicorn /usr/bin/

RUN ln -snf /usr/share/python3/app/bin/pytest /usr/bin/

RUN ln -snf /usr/share/python3/app/bin/uvicorn /usr/bin/

EXPOSE 8000

**backend/requirements.txt**

wheel

fastapi

fastapi-utils

python-arango

gunicorn

uvicorn

ujson

pydantic

python-multipart

aiofiles

## Frontend

**frontend/package.json**

{

"name": "building-plan",

"version": "1.0.0",

"scripts": {

"build": "rollup -c",

"dev": "rollup -c -w",

"start": "rollup -c; sirv public --single --host 0.0.0.0\"",

},

"devDependencies": {

"@rollup/plugin-commonjs": "^11.0.0",

"@rollup/plugin-node-resolve": "^7.0.0",

"rollup": "^1.20.0",

"rollup-plugin-livereload": "^1.0.0",

"rollup-plugin-svelte": "^5.0.3",

"rollup-plugin-terser": "^5.1.2",

"svelte": "^3.0.0"

},

"dependencies": {

"sirv-cli": "^0.4.4"

}

}

**frontend/rollup.config.js**

import svelte from 'rollup-plugin-svelte';

import resolve from '@rollup/plugin-node-resolve';

import commonjs from '@rollup/plugin-commonjs';

import livereload from 'rollup-plugin-livereload';

import { terser } from 'rollup-plugin-terser';

const production = !process.env.ROLLUP\_WATCH;

export default {

input: 'src/main.js',

output: {

sourcemap: true,

format: 'iife',

name: 'app',

file: 'public/build/main.js'

},

plugins: [

svelte({

// enable run-time checks when not in production

dev: !production,

// we'll extract any component CSS out into

// a separate file - better for performance

css: css => {

css.write('public/build/main.css');

}

}),

// If you have external dependencies installed from

// npm, you'll most likely need these plugins. In

// some cases you'll need additional configuration -

// consult the documentation for details:

// https://github.com/rollup/plugins/tree/master/packages/commonjs

resolve({

browser: true,

dedupe: ['svelte']

}),

commonjs(),

// In dev mode, call `npm run start` once

// the bundle has been generated

!production && serve(),

// Watch the `public` directory and refresh the

// browser on changes when not in production

!production && livereload('public'),

// If we're building for production (npm run build

// instead of npm run dev), minify

production && terser()

],

watch: {

clearScreen: false

}

};

function serve() {

let started = false;

return {

writeBundle() {

if (!started) {

started = true;

require('child\_process').spawn('npm', ['run', 'start', '--', '--dev'], {

stdio: ['ignore', 'inherit', 'inherit'],

shell: true

});

}

}

};

}

**frontend/public/index.html**

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset='utf-8'>

<meta name='viewport' content='width=device-width,initial-scale=1'>

<title>Bulding plan</title>

<link rel='icon' type='image/png' href='/favicon.ico'>

<link rel='stylesheet' href='/build/main.css'>

<script defer src='/build/main.js'></script>

</head>

<body>

</body>

</html>

**frontend/src/animation.js**

import { cubicOut } from 'svelte/easing';

export function flyAnimation(node, {

delay = 200,

duration = 300,

easing = cubicOut,

z = 0,

level = 1,

}) {

const style = getComputedStyle(node);

const opacity = +style.opacity;

const transform = style.transform === 'none' ? '' : style.transform;

return {

delay: delay \* level \* 1.3 ,

duration,

easing,

css: t => `

transform: ${transform} matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, ${(1 - t) \* z}, 1);

opacity: ${t \* opacity}`

};

}

**frontend/src/constants.js**

export const serverURL = 'http://0.0.0.0:8000/api';

**frontend/src/main.js**

import App from './App.svelte';

const app = new App({

target: document.body,

});

export default app;

**frontend/src/store.js**

import { writable } from 'svelte/store';

function mapModeCreate() {

const { subscribe, set } = writable('view\_building');

return {

subscribe,

updateMode: async (newMode) => set(newMode),

};

}

function mapFloorLevelCreate() {

const { subscribe, set, update } = writable(null);

return {

subscribe,

increment: () => update(n => n + 1),

decrement: () => update(n => n - 1),

updateLevel: async (level) => set(level),

};

}

function mapChoiceFromPointCreate() {

const { subscribe, set } = writable(null);

return {

subscribe,

updatePoint: async (point) => set(point),

};

}

function mapChoiceToPointCreate() {

const { subscribe, set } = writable(null);

return {

subscribe,

updatePoint: async (point) => set(point),

};

}

function mapPathCreate() {

const { subscribe, set } = writable(new Set());

return {

subscribe,

updatePaths: async (paths) => set(new Set(paths)),

deletePaths: async () => set(new Set())

};

}

function mapOpenObjectCreate() {

const { subscribe, set } = writable(null);

return {

subscribe,

updateObject: async (newObject) => set(newObject),

deleteObject: async () => set(null)

};

}

function mapOpenSearchMenuCreate() {

const { subscribe, set } = writable(false);

return {

subscribe,

isOpen: async () => set(true),

isClose: async () => set(false)

};

}

export const mapMode = mapModeCreate();

export const mapFloorLevel = mapFloorLevelCreate();

export const mapFromPoint = mapChoiceFromPointCreate();

export const mapToPoint = mapChoiceToPointCreate();

export const mapPath = mapPathCreate();

export const mapOpenObject = mapOpenObjectCreate();

export const mapOpenSearchMenu = mapOpenSearchMenuCreate();

**frontend/src/App.svelte**

<script>

import Map from "./components/Map.svelte"

import {serverURL} from "./constants"

let mapURL = serverURL + '/building\_1'

</script>

<Map mapURL="{mapURL}"/>

<style>

:global(\*),

:global(\*::after),

:global(\*::before) {

-webkit-box-sizing: border-box;

box-sizing: border-box;

}

:global(body) {

font-family: 'Avenir Next', Avenir, 'Helvetica Neue', Helvetica, Arial, sans-serif;

color: #515158;

background: #f0f0f0;

-webkit-font-smoothing: antialiased;

-moz-osx-font-smoothing: grayscale;

margin: 0;

}

:global(a) {

text-decoration: none;

color: #aaa;

outline: none;

transition:.5s;

}

:global(a:hover),

:global(button:hover) {

color: #515158;

outline: none;

}

:global(a:focus),

:global(button:focus) {

outline: none;

}

</style>

**frontend/src/components/Map.svelte**

<script>

import {onMount} from "svelte";

import {mapMode, mapFloorLevel, mapOpenSearchMenu, mapPath, mapOpenObject} from '../store.js';

import {fade} from 'svelte/transition';

import {serverURL} from "../constants"

import MapLayer from "./MapLayer.svelte";

import MapSearch from "./MapSearch.svelte";

import SearchIcon from "./icons/SearchIcon.svelte";

import ArrowUpIcon from "./icons/ArrowUpIcon.svelte";

import ArrowDownIcon from "./icons/ArrowDownIcon.svelte";

import LayersIcon from "./icons/LayersIcon.svelte";

export let mapURL

let data = [];

let background = ""

let layers = [];

onMount(async () => {

await loadMapData()

});

async function loadMapData() {

fetch(mapURL)

.then(async (res) => res.json())

.then(async (res) => {

data = res

layers = data.layers

background = serverURL + '/' + data.designation + '/background'

})

.catch(async () => {

await new Promise(resolve => setTimeout(resolve, 500));

await loadMapData()

});

}

async function changeMapMode() {

if ($mapMode === 'view\_building' & $mapFloorLevel !== null) {

await mapMode.updateMode('view\_floor')

} else {

await mapPath.deletePaths();

await mapOpenObject.deleteObject()

await mapMode.updateMode('view\_building')

await mapFloorLevel.updateLevel(null);

}

}

async function changeConditionSearchMenu() {

$mapOpenSearchMenu ? await mapOpenSearchMenu.isClose() : await mapOpenSearchMenu.isOpen()

}

async function checkIncrementOrDecrementLayerAdd(direction) {

if (direction === 'up' && $mapFloorLevel + 1 <= layers.length)

return true

else if (direction === 'down' && $mapFloorLevel - 1 > 0)

return true

return false

}

async function changeActiveLayerDown() {

if (await checkIncrementOrDecrementLayerAdd('down')) {

await mapOpenObject.deleteObject()

await mapFloorLevel.decrement()

}

}

async function changeActiveLayerUp() {

if (await checkIncrementOrDecrementLayerAdd('up')) {

await mapOpenObject.deleteObject()

await mapFloorLevel.increment()

}

}

</script>

<div class="main">

<header class="map-header">

{#if data.title}

<h1 transition:fade="{{ duration: 400 }}">{data.title} / {$mapFloorLevel ? 'Этаж ' + $mapFloorLevel : 'Этаж не выбран'}</h1>

{/if}

</header>

<div class="space">

<div class="background" class:background-hidden="{$mapMode === 'view\_floor'}">

<img class="background-map" src="{background}" alt="background"/>

</div>

<div class="levels">

{#each layers as layer}

<MapLayer level="{layer.id}" layer="{layer}" />

{:else}

<p>Загрузка</p>

{/each}

<slot/>

</div>

</div>

<button on:click={changeConditionSearchMenu} class="boxbutton boxbutton-darker open-search search-mode-buttons-icon">

<SearchIcon/>

</button>

{#if $mapMode === 'view\_floor'}

<nav class="space-nav" transition:fade="{{ duration: 400 }}">

<button on:click={changeActiveLayerUp} class="boxbutton search-mode-buttons-icon" >

<ArrowUpIcon/>

</button>

<button on:click={changeMapMode} class="boxbutton boxbutton-dark space-nav-button-all-levels search-mode-buttons-icon">

<LayersIcon/>

</button>

<button on:click={changeActiveLayerDown} class="boxbutton search-mode-buttons-icon">

<ArrowDownIcon/>

</button>

</nav>

{/if}

</div>

<aside class="spaces-list" class:spaces-list-open="{$mapOpenSearchMenu}">

<MapSearch layers="{layers}"

filteredLayers="{JSON.parse(JSON.stringify(layers)).map(i => i.markers).reduce((a, b) => Object.assign(b, a), {})}"/>

<div class="search">

<button on:click={changeConditionSearchMenu}

class="boxbutton boxbutton-darker close-search search-mode-buttons-icon">

<SearchIcon/>

</button>

</div>

</aside>

<style>

/\* Header \*/

.map-header {

position: absolute;

z-index: 100;

top: 0;

left: 0;

display: -webkit-flex;

display: flex;

-webkit-align-items: center;

align-items: center;

padding: 1.5em 1em;

text-align: center;

}

.map-header h1 {

font-size: 1.15em;

font-weight: normal;

line-height: 1;

margin: 0 0 0 1em;

}

/\* Levels main \*/

.main {

position: fixed;

top: 0;

left: 0;

overflow: hidden;

width: calc(100vw - 300px);

height: 100vh;

}

.background,

.levels {

position: absolute;

top: 50%;

left: 50%;

}

.background {

width: 192vmin;

/\* double of space map \*/

height: 128vmin;

margin: -64vmin 0 0 -96vmin;

pointer-events: none;

-webkit-transition: opacity 0.8s;

transition: opacity 0.8s;

-webkit-transform-style: preserve-3d;

transform-style: preserve-3d;

}

.background-map {

opacity: 0.3;

max-width: 100%;

display: block;

}

.background-hidden {

opacity: 0;

}

.levels {

width: 96vmin;

height: 64vmin;

margin: -32vmin 0 0 -48vmin;

-webkit-transition: -webkit-transform 0.3s;

transition: transform 0.3s;

-webkit-transform-style: preserve-3d;

transform-style: preserve-3d;

}

.background,

.levels {

-webkit-transform: rotateX(70deg) rotateZ(-45deg) translateZ(-15vmin);

transform: rotateX(70deg) rotateZ(-45deg) translateZ(-15vmin);

}

/\* Level nav \*/

.space {

position: relative;

width: 100%;

height: 100%;

pointer-events: none;

-webkit-perspective: 3500px;

perspective: 3500px;

-webkit-perspective-origin: 0% 50%;

perspective-origin: 0% 50%;

-webkit-transition: -webkit-transform 0.8s;

transition: transform 0.8s;

-webkit-transition-timing-function: cubic-bezier(0.2, 1, 0.3, 1);

transition-timing-function: cubic-bezier(0.2, 1, 0.3, 1);

}

.space-nav {

position: absolute;

top: 0;

right: 0;

text-align: center;

-webkit-transition: opacity 0.8s;

transition: opacity 0.8s;

}

/\* Spaces list (sidebar) \*/

.spaces-list {

position: absolute;

top: 0;

right: 0;

width: 300px;

min-height: 100vh;

padding: 2em 0 1em;

background: #fff;

}

/\* Search list (sidebar) \*/

.search {

position: fixed;

z-index: 100;

top: 0;

right: 0;

left: calc(100vw - 300px);

}

:global(.search-mode-buttons-icon svg) {

height: 32px;

width: 32px;

fill: white;

}

:global(.search-mode-buttons-icon) {

cursor: pointer;

}

/\* Box button \*/

.boxbutton {

font-size: 2em;

display: block;

width: 2em;

height: 2em;

margin: 0;

padding: 0;

color: #fff;

border: 0;

background: #d7d7dc;

}

.boxbutton-dark {

background: #c4c4c7;

}

.boxbutton-darker {

background: #2c2c2f;

}

.boxbutton-alt {

background: #3d1975;

}

.boxbutton-disabled,

.boxbutton-disabled:focus,

.boxbutton-disabled:hover {

cursor: default;

pointer-events: none;

opacity: 0.2;

}

/\* Mobile compatability \*/

.open-search,

.close-search {

display: none;

}

@media screen and (max-width: 65.625em), screen and (max-height: 40.625em) {

.main {

width: 100vw;

}

.space-nav {

top: 4em;

}

.spaces-list,

.search {

width: 100vw;

right: 100%;

left: auto;

}

.spaces-list-open,

.spaces-list-open .search {

right: 0;

}

.open-search,

.close-search {

position: absolute;

display: block;

top: 0;

right: 0;

z-index: 1000;

}

}

</style>

**frontend/src/components/MapLayer.svelte**

<script>

import {mapMode, mapFloorLevel, mapPath, mapOpenObject, mapOpenSearchMenu} from '../store.js';

import {fade} from 'svelte/transition';

import {flyAnimation} from '../animation'

import MapText from "./MapText.svelte";

import MapStairs from "./MapStairs.svelte"

export let level;

export let layer;

async function selectLayer() {

if ($mapMode === 'view\_building') {

await mapFloorLevel.updateLevel(level);

await mapMode.updateMode('view\_floor');

}

}

async function selectObject(object) {

if ($mapMode === 'view\_floor') {

await mapOpenObject.updateObject(

Object.assign({}, object, layer.markers[object.marker], {"level": level})

);

await mapOpenSearchMenu.isOpen()

}

}

</script>

<div on:click={selectLayer} in:flyAnimation={{ z: 1000, delay: 100, duration: 600, level: level }}

class="level"

style="{'transform: translateZ(' + (level \* 10 - 10) +'vmin);'}

{$mapMode === 'view\_floor' ? $mapFloorLevel === level ? 'transform: translateZ(15vmin) rotate3d(0,0,1,20deg)'

: level > $mapFloorLevel ? 'transform: translateZ(' + (level \* 10 \* 2.5) + 'vmin); opacity: 0;'

: level < $mapFloorLevel ? 'transform: translateZ(' + -(level \* 10 \* 2.5) + 'vmin); opacity: 0;'

: '' : ''}">

<svg viewBox="0 0 1200 800" width="100%" height="100%" preserveAspectRatio="xMidYMid meet">

{#each layer.objects as item}

{#if item.type === 'polygon'}

<polygon points="{item.points}" class="map\_\_ground {item.id}" />

{:else if item.type === 'rect'}

<rect on:click={selectObject(item)} x={item.x} y={item.y} width={item.width} height={item.height} class="map\_\_space {item.id}"/>

{#if $mapMode === 'view\_floor' && item.marker && item.marker in layer.markers}

{#if layer.markers[item.marker].type === 'stairs'}

<MapStairs item="{item}" marker="{layer.markers[item.marker]}"/>

{:else}

<MapText item="{item}" marker="{layer.markers[item.marker]}"/>

{/if}

{/if}

{:else if item.type === 'ellipse'}

<ellipse cx={item.cx} cy={item.cy} rx={item.rx} ry={item.ry} class="map\_\_tree {item.id}" />

{:else if item.type === 'path'}

<path d="{item.d}" class="map\_\_space {item.id}" />

{:else if item.type === 'line'}

<line class="line--active" x1={item.x1} y1={item.y1} x2={item.x2} y2={item.y2} stroke-width="8" />

{/if}

{/each}

{#each layer.routes as item}

{#if $mapMode === 'view\_floor' && item.type === 'line' && $mapPath.has(item.id)}

<line transition:fade="{{ delay: 0, duration: 600 }}" class="line {item.id}" x1={item.x1} x2={item.x2} y1={item.y1} y2={item.y2} stroke-width="8" />

{/if}

{/each}

{#each layer.entrances as item}

{#if $mapMode === 'view\_floor' && item.type === 'entrance'}

<line class="map\_entrance {item.id}" x1={item.x1} y1={item.y1} x2={item.x2} y2={item.y2} stroke-width="6" />

{/if}

{/each}

<!-- {#each layer.points as item}-->

<!-- {#if $mapMode === 'view\_floor' && item.type === 'point'}-->

<!-- <circle class="{item.id}" cx={item.cx} cy={item.cy} r=6 fill="red"/>-->

<!-- {/if}-->

<!-- {/each}-->

</svg>

</div>

<style>

/\* Levels map \*/

.level {

position: relative;

width: 100%;

height: 100%;

cursor: pointer;

pointer-events: auto;

-webkit-transition: opacity 1s, -webkit-transform 1s;

transition: opacity 1s, transform 1s;

-webkit-transition-timing-function: cubic-bezier(0.7, 0, 0.3, 1);

transition-timing-function: cubic-bezier(0.7, 0, 0.3, 1);

-webkit-transform-style: preserve-3d;

transform-style: preserve-3d;

}

.level::after {

font-size: 2.5vmin;

line-height: 0;

position: absolute;

z-index: 100;

top: -2em;

left: 3.5em;

white-space: nowrap;

color: #7d7d86;

-webkit-transform: rotateZ(45deg) rotateX(-70deg) translateZ(5vmin);

transform: rotateZ(45deg) rotateX(-70deg) translateZ(5vmin);

-webkit-transition: -webkit-transform 1s, color 0.3s;

transition: transform 1s, color 0.3s;

-webkit-transition-timing-function: cubic-bezier(0.7, 0, 0.3, 1);

transition-timing-function: cubic-bezier(0.7, 0, 0.3, 1);

}

.level:not(:first-child) {

position: absolute;

top: 0;

left: 0;

}

/\* Level map \*/

.map\_entrance {

stroke: #626262;

}

.map\_\_outline {

-webkit-transition: fill 0.3s;

transition: fill 0.3s;

fill: #bbb;

}

.map\_\_space {

-webkit-transition: fill-opacity 0.8s;

transition: fill-opacity 0.8s;

fill: #bdbdbd;

fill-opacity: 0.6;

}

.map\_\_ground {

fill: #d7d7dc;

}

/\* Lines \*/

.line {

stroke: black;

}

</style>

**frontend/src/components/MapSearch.svelte**

<script>

import {slide, fade} from 'svelte/transition';

import {mapFromPoint, mapToPoint, mapPath, mapOpenObject, mapOpenSearchMenu, mapFloorLevel, mapMode} from '../store.js';

import {serverURL} from '../constants.js'

export let filteredLayers = [];

export let layers = [];

let searchTerm = "";

async function findPath() {

if ($mapFromPoint && $mapToPoint) {

const res = await fetch(serverURL + '/route' + '/building\_1' + '/' + $mapFromPoint.id + '/' + $mapToPoint.id);

await mapPath.updatePaths(await res.json());

}

}

async function fromHereSelectObject() {

await mapFromPoint.updatePoint($mapOpenObject)

await mapOpenObject.deleteObject()

await findPath()

}

async function hereSelectObject() {

await mapToPoint.updatePoint($mapOpenObject)

await mapOpenObject.deleteObject()

await findPath()

}

async function closeSelectObject() {

await mapOpenObject.deleteObject()

}

async function findSelectedObject(selectedObject) {

searchTerm = "";

layers.some(function(item) {

let found = false;

item.objects.some(function (object) {

if (selectedObject.id === object.marker) {

found = true

mapOpenObject.updateObject(Object.assign({}, selectedObject, object, {"level": item.id}));

mapOpenSearchMenu.isOpen()

if ($mapMode === 'view\_building' || $mapMode ==='view\_floor') {

mapFloorLevel.updateLevel(item.id);

mapMode.updateMode('view\_floor');

}

}

})

if (found) return true

});

}

$: filteredList = Object.keys(filteredLayers).reduce(function(r, e) {

if (!filteredLayers[e].title.toLowerCase().indexOf(searchTerm.toLowerCase()))

r.push(Object.assign(filteredLayers[e], {"id": e}))

return r;

}, [])

</script>

<div class="search\_container">

{#if $mapOpenObject}

<div class="search\_object\_box">

<div on:click={closeSelectObject} class="search\_close\_button"></div>

<h1>

{$mapOpenObject.title}</h1>

<span>

{"Этаж " + $mapOpenObject.level}

</span>

<p>

{$mapOpenObject.description}

</p>

<div class="search\_select\_button\_box">

<button on:click={fromHereSelectObject} class="search\_select\_button">Отсюда</button>

<button on:click={hereSelectObject} class="search\_select\_button">Сюда</button>

</div>

{#if $mapOpenObject.site}

<a class="search\_select\_link\_button" href="{'http://' + $mapOpenObject.site}">Перейти на сайт</a>

{/if}

</div>

{:else}

<div class="search\_route\_box">

{#if $mapFromPoint || $mapToPoint}

<div>

<p class="search\_route\_p">Отсюда</p>

{#if $mapFromPoint}

<button class="search\_select\_route\_button">{$mapFromPoint.title}</button>

{:else}

<button class="search\_select\_route\_button\_inactive">Выберите место</button>

{/if}

</div>

<div>

<p class="search\_route\_p">Сюда</p>

{#if $mapToPoint}

<button class="search\_select\_route\_button">{$mapToPoint.title}</button>

{:else}

<button class="search\_select\_route\_button\_inactive">Выберите место</button>

{/if}

</div>

{/if}

<div>

<p class="search\_route\_p">Поиск</p>

<input class="search\_button" placeholder="Магнит" bind:value={searchTerm} />

<ul class="filter\_item">

{#each filteredList as item}

<li on:click={findSelectedObject(item)}>{item.title}</li>

{/each}

</ul>

</div>

</div>

{/if}

</div>

<style>

.filter\_item {

list-style-type: square;

padding-left: 20px;

}

.filter\_item li {

margin-bottom: 8px;

cursor: pointer;

}

.search\_button {

background-color: #515158;

border: none;

color: rgb(185, 185, 185);

padding: 15px 0;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

width: 100%;

outline:none;

}

.search\_container {

margin-left: 1em;

margin-right: 1em;

}

.search\_container h1 {

font-size: 2.4em;

margin-bottom: 0;

}

.search\_route\_p {

margin-bottom: 2px;

font-weight: 600;

font-size: 1.2em;

}

.search\_select\_button {

background-color: #515158;

border: none;

color: white;

padding: 15px 0;

font-size: 16px;

cursor: pointer;

width: 100%;

transition: color .2s ease-out, background .2s ease-in-out;

}

.search\_select\_button:hover {

background-color: #717171;

}

.search\_select\_link\_button {

background-color: #515158;

border: none;

color: white;

padding: 15px 0;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin-top: 8px;

cursor: pointer;

width: 100%;

transition: color .2s ease-out, background .2s ease-in-out;

}

.search\_select\_link\_button:hover {

background-color: #717171;

}

.search\_select\_button\_box {

display: grid;

grid-template-columns: 50% 50%;

}

.search\_select\_route\_button {

background-color: #515158;

border: none;

color: white;

padding: 15px 0;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

width: 100%;

}

.search\_select\_route\_button\_inactive {

background-color: #515158;

border: none;

color: rgb(185, 185, 185);

padding: 15px 0;

text-align: center;

text-decoration: none;

display: inline-block;

font-size: 16px;

margin: 4px 2px;

cursor: pointer;

width: 100%;

}

.search\_close\_button {

position: absolute;

right: 32px;

top: 32px;

width: 32px;

height: 32px;

opacity: 0.3;

cursor: pointer;

}

.search\_close\_button:hover {

opacity: 1;

}

.search\_close\_button:before, .search\_close\_button:after {

position: absolute;

left: 15px;

content: ' ';

height: 33px;

width: 2px;

background-color: #333;

}

.search\_close\_button:before {

transform: rotate(45deg);

}

.search\_close\_button:after {

transform: rotate(-45deg);

}

</style>

**frontend/src/components/MapStairs.svelte**

<script>

import { fade } from 'svelte/transition';

export let item;

export let marker;

</script>

<defs>

<symbol id="s-stairs">

<circle class="st0" cx="24.2" cy="9.8" r="3.8"></circle>

<path class="st0" d="M52.4,11.7H39.6v0c-0.6,0-1.3,0.3-1.7,0.7l-9.2,9.2c-0.4,0.4-0.7,0.3-0.7-0.3v-2.9c0-1.8-1.5-3.3-3.3-3.3

h-0.8c-1.8,0-3.3,1.5-3.3,3.3v10.4c0,0.6-0.3,1.3-0.7,1.7L13,37.2c-0.4,0.4-1.2,0.7-1.7,0.7H5.9c-3.3,0-5.9,2.6-5.9,5.9v1.1

c0,3.2,2.6,5.9,5.9,5.9h12.8c0.6,0,1.3-0.3,1.7-0.7l24.9-24.8c0.4-0.4,1.2-0.7,1.7-0.7h5.4c3.3,0,5.9-2.6,5.9-5.9v-1.1

C58.3,14.3,55.6,11.7,52.4,11.7z M54.5,18.7c0,1.2-0.9,2.1-2.1,2.1h-7c-0.6,0-1.3,0.3-1.7,0.7L18.9,46.3c-0.4,0.4-1.2,0.7-1.7,0.7

H5.9c-1.2,0-2.1-0.9-2.1-2.1v-1.1c0-1.2,0.9-2.1,2.1-2.1h7c0.6,0,1.3-0.3,1.7-0.7l24.9-24.8c0.4-0.4,1.2-0.7,1.7-0.7h11.3

c1.2,0,2.1,0.9,2.1,2.1V18.7z"></path>

</symbol>

</defs>

<use xlink:href="#s-stairs"

x={item.x + ((item.width - 58) / 2)} y={item.y + ((item.height - 58) / 2) + 2}

width="64px"

height="64px">

</use>

**frontend/src/components/MapText.svelte**

<script>

import { fade } from 'svelte/transition';

export let item;

export let marker;

</script>

<text in:fade="{{ delay: 920, duration: 400 }}" out:fade="{{ delay: 0, duration: 150 }}"

x={item.x + 20}

y={item.y + (item.height / 2) + 6}

class="text">

{marker.title}

</text>

<style>

.text {

font: bold 2em sans-serif;

user-select: none;

}

</style>

**frontend/src/components/icons/ArrowDownIcon.svelte**

<svg viewBox="0 0 21 32" width="100%" height="100%">

<path d="M19.196 13.143q0 0.232-0.179 0.411l-8.321 8.321q-0.179 0.179-0.411 0.179t-0.411-0.179l-8.321-8.321q-0.179-0.179-0.179-0.411t0.179-0.411l0.893-0.893q0.179-0.179 0.411-0.179t0.411 0.179l7.018 7.018 7.018-7.018q0.179-0.179 0.411-0.179t0.411 0.179l0.893 0.893q0.179 0.179 0.179 0.411z"></path>

</svg>

**frontend/src/components/icons/ArrowUpIcon.svelte**

<svg viewBox="0 0 21 32" width="100%" height="100%">

<path d="M19.196 21.143q0 0.232-0.179 0.411l-0.893 0.893q-0.179 0.179-0.411 0.179t-0.411-0.179l-7.018-7.018-7.018 7.018q-0.179 0.179-0.411 0.179t-0.411-0.179l-0.893-0.893q-0.179-0.179-0.179-0.411t0.179-0.411l8.321-8.321q0.179-0.179 0.411-0.179t0.411 0.179l8.321 8.321q0.179 0.179 0.179 0.411z"></path>

</svg>

**frontend/src/components/icons/LayersIcon.svelte**

<svg class="search-mode-buttons-icon" viewBox="0 0 32 32" width="100%" height="100%">

<path d="M29.143 11.071l-13.143-6.571-13.143 6.571 13.143 6.571 13.143-6.571zM16 6.681l8.781 4.39-8.781 4.39-8.781-4.39 8.781-4.39zM26.51 14.684l2.633 1.316-13.143 6.571-13.143-6.571 2.633-1.316 10.51 5.255zM26.51 19.612l2.633 1.316-13.143 6.571-13.143-6.571 2.633-1.316 10.51 5.255z"></path>

</svg>

**frontend/src/components/icons/SearchIcon.svelte**

<svg viewBox="0 0 32 32" width="100%" height="100%">

<title>Search</title>

<path class="path1" d="M26.819 24.917c0.262 0.262 0.262 0.688 0 0.951l-0.951 0.951c-0.263 0.262-0.688 0.262-0.951 0l-6.656-6.656c-0.072-0.072-0.12-0.158-0.153-0.248-1.365 0.998-3.041 1.593-4.862 1.593-4.562 0-8.261-3.699-8.261-8.261s3.699-8.261 8.261-8.261c4.562 0 8.261 3.699 8.261 8.261 0 1.821-0.595 3.497-1.593 4.862 0.091 0.032 0.176 0.080 0.248 0.153l6.656 6.656zM13.246 7.739c-3.041 0-5.508 2.466-5.508 5.508s2.466 5.508 5.508 5.508 5.508-2.466 5.508-5.508c0-3.041-2.466-5.508-5.508-5.508z"></path>

</svg>

## Backend

**backend/app/main.py**

import uvicorn

from fastapi import FastAPI

from starlette.middleware.cors import CORSMiddleware

from app import config

from app.api import route

app = FastAPI(

title=config.PROJECT\_NAME,

description=config.PROJECT\_DESCRIPTION,

version=config.PROJECT\_VERSION,

docs\_url=config.PROJECT\_DOCS\_URL,

redoc\_url=None,

)

app.add\_middleware(

CORSMiddleware,

allow\_origins=["\*"],

allow\_credentials=True,

allow\_methods=["\*"],

allow\_headers=["\*"],

)

app.include\_router(route.router, prefix=config.PROJECT\_API\_V1\_URL)

if \_\_name\_\_ == "\_\_main\_\_":

uvicorn.run(app, host="0.0.0.0", port=8000)

**backend/app/config.py**

from os import getenv, path

DIRECTORY = path.abspath(\_\_file\_\_).rstrip("config.py")

SERVER\_NAME = getenv("SERVER\_NAME")

SERVER\_HOST = getenv("SERVER\_HOST")

PROJECT\_NAME = getenv("PROJECT\_NAME") or "BuildingPlan"

PROJECT\_DESCRIPTION = getenv("PROJECT\_DESCRIPTION") or "API"

PROJECT\_VERSION = getenv("PROJECT\_VERSION") or "1.0"

PROJECT\_DOCS\_URL = "/documentation"

PROJECT\_API\_V1\_URL = "/api"

ARANGODB\_DATABASE\_NAME = getenv("ARANGODB\_DATABASE\_NAME") or "\_system"

ARANGODB\_DATABASE\_URL = getenv("ARANGODB\_DATABASE\_URL") or "http://localhost:8529"

**backend/app/api/route.py**

from typing import List

import ujson

from fastapi import APIRouter, File, UploadFile

from fastapi.responses import FileResponse

from arango.exceptions import CollectionDeleteError

from app.core import arangodb

from app.core.exceptions import DELETE\_EXCEPTION

from app.config import DIRECTORY

from app.database import database

from app.core.helpers import check\_file\_in\_background\_folder

from app.schemas.building import BuildingSchema

from app.schemas.building\_import import ImportData

router = APIRouter()

@router.get("/buildings", tags=["collections"], response\_model=List[str])

async def get\_buildings():

"""

Получает все названия зданий

"""

return [

item["\_key"] for item in await database.find\_or\_create\_collection("buildings")

]

@router.get("/{building}", tags=["collections"], response\_model=BuildingSchema)

async def get\_building(building: str):

"""

Получает JSON здания

"""

if database.instance.has\_collection("buildings"):

building\_collection = database.instance.collection("buildings")

for data in building\_collection:

if data["\_key"] == building:

return BuildingSchema(\*\*ujson.loads(data["data"]))

@router.get("/{building}/background", tags=["collections"])

async def get\_building\_background(building):

"""

Получает фон здания

"""

return FileResponse(f"{DIRECTORY}backgrounds/{building}.svg")

@router.put("/{building}/background", tags=["collections"], response\_model=bool)

async def update\_building\_background(building, file: UploadFile = File(...)):

"""

Загружает новый фон для здания

"""

if ".svg" in file.filename and database.instance.has\_collection(

building + "\_objects"

):

return await check\_file\_in\_background\_folder(building + ".svg", file)

return False

@router.delete(

"/{building}", tags=["collections"], response\_model=bool,

)

async def delete\_building(building: str):

"""

Удаляет здание

"""

try:

database.instance.delete\_collection(f"{building}\_objects")

database.instance.delete\_collection(f"{building}\_connections")

database.instance.delete\_collection(f"{building}\_data")

except CollectionDeleteError:

raise DELETE\_EXCEPTION

database.instance.delete\_graph(building)

if database.instance.has\_collection("buildings"):

building\_collection = database.instance.collection("buildings")

if building\_collection.has(building):

building\_collection.delete(building)

return True

@router.get("/route/{building}/{of}/{to}", tags=["routes"])

async def search\_route\_in\_building(building: str, of: str, to: str):

"""

Ищет путь в здании

"""

return await database.find\_shortest\_path(

of=f"{building}\_objects/{of}", to=f"{building}\_objects/{to}", graph=building

)

@router.post("/import/", tags=["core"], response\_model=BuildingSchema)

async def import\_file(data: ImportData):

"""

Загружает файл в ArangoDB

"""

return await arangodb.parsing\_imported\_file(data)

**backend/app/core/arangodb.py**

from typing import Union, Tuple, Dict

import ujson

from arango.exceptions import GraphCreateError, DocumentInsertError

from arango.collection import StandardCollection

from app.database import database

from app.schemas.building\_import import (

ImportData,

ImportRect,

ImportPath,

ImportPolygon,

ImportEntrance,

ImportPoint,

ImportLine,

ImportEntranceLine,

ImportUpstairLine,

)

from app.schemas.building import BuildingSchema

from app.core.exceptions import CONFLICT\_GRAPH\_EXCEPTION, DOCUMENT\_INSERT\_EXCEPTION

async def objects\_to\_dict\_with\_key(

item: Union[ImportRect, ImportPath, ImportPolygon, ImportEntrance, ImportPoint]

) -> Dict:

"""

Преобразует обьектвы в словарь с \_key вместо id

:param item:

:return:

"""

dict\_item: dict = item.dict()

dict\_item["\_key"] = dict\_item.pop("id")

return dict\_item

async def connections\_to\_dict\_with\_key\_from\_to(

item: Union[ImportLine, ImportEntranceLine, ImportUpstairLine], building\_title: str

) -> Tuple[Dict, Dict]:

"""

Преобразует связи в словарь с \_key вместо id

:param item:

:param building\_title:

:return:

"""

dict\_item: dict = item.dict()

dict\_item["\_key"] = dict\_item.pop("id")

dict\_item["\_from"] = f"{building\_title}/{dict\_item.pop('p1')}"

dict\_item["\_to"] = f"{building\_title}/{dict\_item.pop('p2')}"

dict\_item\_2 = dict\_item.copy()

dict\_item["\_key"] = dict\_item["\_key"] + "\_\_\_2"

dict\_item["\_to"], dict\_item["\_from"] = dict\_item["\_from"], dict\_item["\_to"]

return dict\_item, dict\_item\_2

async def parsing\_imported\_file(data: ImportData) -> BuildingSchema:

"""

Парсинг полученого файла + загрузка в ArangoDB

:param data: файл

:return:

"""

building\_objects\_label = f"{data.designation}\_objects"

building\_connections\_label = f"{data.designation}\_connections"

building\_data\_label = f"{data.designation}\_data"

try:

graph = database.instance.create\_graph(data.designation)

except GraphCreateError:

raise CONFLICT\_GRAPH\_EXCEPTION

building\_objects = graph.create\_vertex\_collection(building\_objects\_label)

building\_connections = graph.create\_edge\_definition(

edge\_collection=building\_connections\_label,

from\_vertex\_collections=[building\_objects\_label],

to\_vertex\_collections=[building\_objects\_label],

)

buildings: StandardCollection = await database.find\_or\_create\_collection(

"buildings"

)

try:

for layer in data.layers:

for item in layer.objects:

building\_objects.insert(await objects\_to\_dict\_with\_key(item))

for item in layer.entrances:

building\_objects.insert(await objects\_to\_dict\_with\_key(item))

for item in layer.points:

building\_objects.insert(await objects\_to\_dict\_with\_key(item))

for item in layer.routes:

item\_1, item\_2 = await connections\_to\_dict\_with\_key\_from\_to(

item, building\_objects\_label

)

building\_connections.insert(item\_1)

building\_connections.insert(item\_2)

for item in data.upstairs:

item\_1, item\_2 = await connections\_to\_dict\_with\_key\_from\_to(

item, building\_objects\_label

)

building\_connections.insert(item\_1)

building\_connections.insert(item\_2)

except DocumentInsertError as e:

error = DOCUMENT\_INSERT\_EXCEPTION

error.detail = f"{error.detail} {e}"

raise error

building\_schema = BuildingSchema(\*\*data.dict())

if not buildings.has(data.designation):

buildings.insert(

{"\_key": data.designation, "data": ujson.dumps(building\_schema.dict())}

)

return building\_schema

**backend/app/core/arangodb.py**

from fastapi import HTTPException

from starlette.status import HTTP\_409\_CONFLICT

# Core

UNKNOWN\_ERROR\_EXCEPTION = HTTPException(

status\_code=HTTP\_409\_CONFLICT, detail="Unknown error."

)

# Graph

CONFLICT\_GRAPH\_EXCEPTION = HTTPException(

status\_code=HTTP\_409\_CONFLICT, detail="This graph already exists."

)

DOCUMENT\_INSERT\_EXCEPTION = HTTPException(

status\_code=HTTP\_409\_CONFLICT, detail="This document already exists."

)

DELETE\_EXCEPTION = HTTPException(

status\_code=HTTP\_409\_CONFLICT, detail="Collection or view not found."

)

**backend/app/core/helper.py**

import os

import aiofiles

from fastapi import UploadFile

from app.config import DIRECTORY

class MetaSingleton(type):

\_instances = {}

def \_\_call\_\_(cls, \*args, \*\*kwargs):

if cls not in cls.\_instances:

cls.\_instances[cls] = super(MetaSingleton, cls).\_\_call\_\_(\*args, \*\*kwargs)

return cls.\_instances[cls]

async def check\_file\_in\_background\_folder(label: str, object: UploadFile) -> bool:

"""

Обновляет файл в директории

:param label:

:param object:

:return:

"""

background\_folder = DIRECTORY + "backgrounds/"

if not os.path.exists(background\_folder):

os.makedirs(background\_folder)

file\_path = background\_folder + label

async with aiofiles.open(file\_path, "wb") as f:

await f.write(await object.read())

return True

**backend/app/database/\_\_init\_\_.py**

from app.database.model import ArangoDB

from app.config import ARANGODB\_DATABASE\_URL, ARANGODB\_DATABASE\_NAME

database: ArangoDB = ArangoDB(

hosts=ARANGODB\_DATABASE\_URL, database=ARANGODB\_DATABASE\_NAME

)

**backend/app/database/model.py**

import asyncio

import ujson

from arango import ArangoClient

from arango.database import StandardDatabase, AsyncDatabase

from arango.collection import StandardCollection

from arango.job import AsyncJob

from app.core.helpers import MetaSingleton

class ArangoDB(metaclass=MetaSingleton):

client: ArangoClient

instance: StandardDatabase

async\_instance: AsyncDatabase

tasks: dict

def \_\_init\_\_(

self,

hosts: str = "http://0.0.0.0:8529",

database: str = "\_system",

username: str = "root",

password: str = "passwd",

host\_resolver: str = "roundrobin",

):

self.client = ArangoClient(

hosts=hosts,

serializer=ujson.dumps,

deserializer=ujson.loads,

host\_resolver=host\_resolver,

)

self.instance = self.client.db(database, username=username, password=password)

self.async\_instance = self.instance.begin\_async\_execution(return\_result=True)

async def receive\_asynс\_response(self, task: AsyncJob):

while task.status() != "done":

await asyncio.sleep(0.1)

return [i for i in task.result()]

async def find\_shortest\_path(self, of: str, to: str, graph: str):

task = self.async\_instance.aql.execute(

"""

FOR v, e IN OUTBOUND SHORTEST\_PATH @of TO @to

GRAPH @graph

RETURN { "connection": v.\_key, "object": REGEX\_REPLACE(e.\_key, "\_\_\_2", "") }

""",

bind\_vars={"of": of, "to": to, "graph": graph},

)

task, paths = await self.receive\_asynс\_response(task), set()

for item in task:

if item["connection"]:

paths.add(item["connection"])

if item["object"]:

paths.add(item["object"])

return paths

async def find\_or\_create\_collection(self, label: str) -> StandardCollection:

"""

:param label:

:return:

"""

if self.instance.has\_collection(label):

return self.instance.collection(label)

else:

return self.instance.create\_collection(label)

**backend/app/schemas/base.py**

import ujson

from pydantic import BaseModel

class FastModel(BaseModel):

class Config:

json\_loads = ujson.loads

json\_dumps = ujson.dumps

**backend/app/schemas/building\_import.py**

from typing import List, Dict, Union

from pydantic import AnyHttpUrl

from app.schemas.base import FastModel

class ImportPolygon(FastModel):

type: str

id: str

points: str

marker: Union[str, None]

class ImportPath(FastModel):

type: str

id: str

d: str

marker: Union[str, None]

class ImportRect(FastModel):

type: str

id: str

x: float

y: float

width: float

height: float

marker: Union[str, None]

class ImportEntrance(FastModel):

type: str

id: str

x1: float

y1: float

x2: float

y2: float

class ImportPoint(FastModel):

type: str

id: str

cx: float

cy: float

class ImportLine(FastModel):

type: str

id: str

p1: str

p2: str

time: float

x1: float

y1: float

x2: float

y2: float

class ImportEntranceLine(FastModel):

type: str

id: str

p1: str

p2: str

class ImportUpstairLine(FastModel):

type: str

id: str

p1: str

p2: str

time: int

class ImportMarker(FastModel):

title: str

type: str

description: Union[str, None]

site: Union[str, None]

class ImportLayer(FastModel):

id: int

objects: List[Union[ImportRect, ImportPath, ImportPolygon]]

entrances: List[ImportEntrance]

points: List[ImportPoint]

routes: List[Union[ImportLine, ImportEntranceLine]]

markers: Dict[str, ImportMarker]

class ImportData(FastModel):

title: str

designation: str

layers: List[ImportLayer]

upstairs: List[ImportUpstairLine]

**backend/app/schemas/building.py**

from typing import List, Dict, Union

from pydantic import AnyHttpUrl, validator

from app.schemas.base import FastModel

class BuildingPolygon(FastModel):

type: str

id: str

points: str

marker: Union[str, None]

class BuildingPath(FastModel):

type: str

id: str

d: str

marker: Union[str, None]

class BuildingRect(FastModel):

type: str

id: str

x: float

y: float

width: float

height: float

marker: Union[str, None]

class BuildingEntrance(FastModel):

type: str

id: str

x1: float

y1: float

x2: float

y2: float

class BuildingRoute(FastModel):

type: str

id: str

x1: float

y1: float

x2: float

y2: float

class BuildingEntranceRoute(FastModel):

type: str

id: str

class BuildingMarker(FastModel):

title: str

type: str

description: Union[str, None]

site: Union[str, None]

class BuildingLayer(FastModel):

id: int

objects: List[Union[BuildingRect, BuildingPath, BuildingPolygon]]

entrances: List[BuildingEntrance]

routes: List[Union[BuildingRoute, BuildingEntranceRoute]]

markers: Dict[str, BuildingMarker]

class BuildingSchema(FastModel):

title: str

designation: str

layers: List[BuildingLayer]

1. Separation of concerns – разделения ответственности, [ru.wikipedia.org/wiki/Разделение\_ответственности](http://ru.wikipedia.org/wiki/__________________________) [↑](#footnote-ref-1)
2. ArangoDB – база данных, [arangodb.com](https://www.arangodb.com/) [↑](#footnote-ref-2)
3. Svelte – JavaScript библиотека с открытым исходным кодом, [svelte.dev](https://svelte.dev/) [↑](#footnote-ref-3)
4. Rollup – JavaScript библиотека, который собирает код в большую систему или приложение, [rollupjs.org](http://rollupjs.org/) [↑](#footnote-ref-4)
5. Python 3.8 – высокоуровневый язык программирования, [python.org](http://python.org/) [↑](#footnote-ref-5)
6. FastAPI – python web библиотека, [fastapi.tiangolo.com](http://fastapi.tiangolo.com/) [↑](#footnote-ref-6)
7. Python-Arango – библиотека для взаимодействия с ArangoDB, <python-driver-for-arangodb.readthedocs.io> [↑](#footnote-ref-7)
8. Uvicorn – ASGI сервер, [uvicorn.org](http://uvicorn.org/) [↑](#footnote-ref-8)
9. ujson – JSON encoder/decoder, [pypi.org/project/ujson/](http://pypi.org/project/ujson/) [↑](#footnote-ref-9)
10. pydantic – валидартор данных, <pydantic-docs.helpmanual.io> [↑](#footnote-ref-10)
11. Docker – ПО для работы с контейнерами, [docker.com](http://docker.com/) [↑](#footnote-ref-11)
12. NPM – менеджер JavaScript пакетов работающий на NodeJS, [npmjs.com](http://npmjs.com/) [↑](#footnote-ref-12)
13. PIP – система управления пакетами Python-а, [pypi.org/project/pip/](http://pypi.org/project/pip/) [↑](#footnote-ref-13)
14. Sidebar – боковая панель, [en.wikipedia.org/wiki/Sidebar\_(computing)](http://en.wikipedia.org/wiki/Sidebar_(computing)) [↑](#footnote-ref-14)
15. WebGL – кроссплатформенный API для 3D-графики в браузере, [ru.wikipedia.org/wiki/WebGL](https://ru.wikipedia.org/wiki/WebGL) [↑](#footnote-ref-15)
16. 2gl – библиотека компании 2GIS для отрисовки 3D-графики, [github.com/2gis/2gl](http://github.com/2gis/2gl) [↑](#footnote-ref-16)
17. Новый интерфейс Яндекс.Метро и технологии, с помощью которых он работает, [habr.com/ru/company/yandex/blog/213033/](https://habr.com/ru/company/yandex/blog/213033/) [↑](#footnote-ref-17)
18. JSON – текстовый формат обмена данными, основанный на JavaScript, [ru.wikipedia.org/wiki/JSON](https://ru.wikipedia.org/wiki/JSON) [↑](#footnote-ref-18)
19. Инсанс – экземпляр класса, [ru.wikipedia.org/wiki/Объект\_(программирование)](http://ru.wikipedia.org/wiki/_______(________________)) [↑](#footnote-ref-19)
20. Валидация – проверка данных на соответствие требованиям, <https://ru.wikipedia.org/wiki/Валидация> [↑](#footnote-ref-20)
21. OAS3 – формат документации, [openapis.org](http://openapis.org/) [↑](#footnote-ref-21)
22. livereload – автоматический перезапуск приложения, после изменения его файлов. [↑](#footnote-ref-22)
23. SPA – одностраничное приложение , [ru.wikipedia.org/wiki/Одностраничное\_приложение](http://ru.wikipedia.org/wiki/_________________________) [↑](#footnote-ref-23)
24. Шаблон проектирования – архитектурная конструкция, [ru.wikipedia.org/wiki/Шаблон\_проектирования](http://ru.wikipedia.org/wiki/_____________________) [↑](#footnote-ref-24)
25. Revealing Module – паттерн “Открытый модуль”, [medium.com/@Rahulx1/revealing-module-pattern-tips-e3442d4e352](https://medium.com/@Rahulx1/revealing-module-pattern-tips-e3442d4e352) [↑](#footnote-ref-25)
26. GitHub – крупнейший веб-сервис для хостинга IT-проектов, <https://ru.wikipedia.org/wiki/GitHub> [↑](#footnote-ref-26)
27. Репозиторий – место, где хранятся данные, [https](https://ru.wikipedia.org/wiki/Репозиторий)://ru.wikipedia.org/wiki/Репозиторий [↑](#footnote-ref-27)
28. Continues integration – непрерывная интеграция,<https://ru.wikipedia.org/wiki/Непрерывная_интеграция> [↑](#footnote-ref-28)
29. Коммит – изменение кода в хранилище, <https://en.wikipedia.org/wiki/Commit_(version_control)> [↑](#footnote-ref-29)