Министерство науки и высшего образования Российской Федерации   
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ   
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет безопасности информационных технологий

Дисциплина:

«Web программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Разработка Веб сервиса»

Выполнил:

Чан Бао Линь, студентка группы N3346



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

(подпись)

Проверил:

Менщиков Александр Алексеевич, Доцент ФБИТ

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

    (отметка о выполнении)

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

(подпись)

**Оглавление**

[Введение  3](#__RefHeading___Toc117_2474820760)

[Описание архитектуры ПО 3](#__RefHeading___Toc119_2474820760)

[Требования к сервису API 3](#__RefHeading___Toc121_2474820760)

[Требования к сервису frontend 3](#__RefHeading___Toc123_2474820760)

[Требования к базе данных 4](#__RefHeading___Toc125_2474820760)

[Описание структуры базы данных  5](#__RefHeading___Toc127_2474820760)

[Описание протокола и форматов передачи данных  5](#__RefHeading___Toc129_2474820760)

[Описание API  6](#__RefHeading___Toc131_2474820760)

[Скриншоты функционала  7](#__RefHeading___Toc133_2474820760)

[Вывод  9](#__RefHeading___Toc135_2474820760)

### **Введение**

Цель работы:

Необходимо разработать приложение-админку для управления базой данных студентов. Для этого создадим следующий архитектурный план:

### Описание архитектуры ПО

Приложение состоит из двух микросервисов.

Первый - REST API на Python FastAPI, который позволяет управлять записями данных студентов.  Для взаимодействия с БД используется библиотека sqlalchemy

Второй - собранный статистический сайт на React.

База данных выбрана PostgreSQL

***Сценарий использования***

Пользователь заходит на главную страницу сайта и видит таблицу со студентами. Таблица имеет 6 колонок: фамилия, имя, отчество, курс, факультет, группа. Пользователь может менять количество отображаемых записей на странице. Если записи не вмещаются на одну страницу, то появляются кнопки, обеспечивающие демонстрацию следующих записей.

***Техническое задание***

1.Микросервис API - функциональная REST JSON API

2. Микросервис frontend - веб-сервер (пример, на базе Ngix), который выдает собранные файлы JS, CSS, HTML на основе React;

3. База данных.

### *Требования к сервису API*

1. Должен быть разработан на языке Python с использованием фреймворка FastAPI
2. Выдача данных только в JSON.
3. Поддержка пагинации с произвольным размером страницы.
4. Хранение и извлечение данных должно быть в БД PostgreSQL или MongoDB по выбору.
5. Доступ к БД только с использованием ORM библиотеки.
6. При выполнении сложных версий лабораторной работы необходимо использовать REST подход и правильные HTTP методы: GET, POST, DELETE, PATCH.
7. Данные в БД можете заполнить тестовыми сведениями (~20 записей).

### *Требования к сервису frontend*

1. Должен быть разработан на языке JavaScript.
2. Должен использовать React.
3. Должен запрашивать данные с использованием REST API.
4. Необходимо реализовать переключатель страниц (пагинация и изменение размера страницы).
5. Итоговый бандл должен быть собран в виде статического файла. Раздавать их можно с помощью любого веб-сервера

### *Требования к базе данных*

1. Должна быть выбрана БД PostgreSQL или MongoDB.
2. Одна таблица со следующими полями: Фамилия, Имя, Отчество, Курс, Группа, Факультет.

***Архитектура***

|  |
| --- |
| Веб-сервер  (ReactJS)  Бекенд  (FastAPI)  БД  (PostgreSQL) |
| Рисунок 1. Архитектура |

Пользователь взаимодействует с фронтенд частью, которая отправляет запросы к серверной части для подгрузки записей, а бекенд в свою очередь обращается к базе данных, чтобы получить и отправить клиенту запрашиваемые данные. Также с клиента отправляются номер страницы и количество записей на страницу. Это обеспечивает пагинацию, так по сети отправляются не все данные, а лишь необходимая часть

Приложение разворачивается с помощью docker-compose файла. Ниже приведен его код

networks:

  dev:

services:

  nginx:

    image: nginx:stable-alpine

    ports:

      - "80:80"

    volumes:

      - './nginx.conf:/etc/nginx/nginx.conf'

      - '/etc/letsencrypt:/etc/letsencrypt'

    depends\_on:

      - backend

      - frontend

    networks:

      - dev

  backend:

    build:

      context: ./backend

    networks:

      - dev

  frontend:

    build:

      context: ./frontend

    networks:

      - dev

При этом frontend часть упаковывает следующий Dockerfile:

FROM node:alpine as build

COPY package.json package.json

RUN npm install

COPY . .

RUN npm run build

FROM nginx:stable-alpine

COPY --from=build /dist /usr/share/nginx/html

COPY --from=build nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 3000

CMD [ "nginx", "-g", "daemon off;" ]

backend:

FROM python:3.11.9-slim  
  
COPY requirements.txt requirements.txt  
RUN pip install -r requirements.txt  
  
COPY . .  
  
CMD [ "python", "api/main.py" ]

|  |
| --- |
|  |
| Рисунок 2: демонстрация успешного запуска с помощью docker-compose файла |

### Описание структуры базы данных

|  |
| --- |
|  |
| Рисунок 3. Структура таблицы |

База данных создается с помощью alembic. Alembic обеспечивает широкий круг возможностей в редактировании полей и добавлении новых, добавлении новых таблиц

### Описание протокола и форматов передачи данных

Для реализации взаимодействия между клиентом и сервером в системе используется протокол HTTP. Ниже описаны особенности использования и форматы данных, передаваемые между клиентом и сервером.

***Протоколы***

1. HTTP:

* Назначение: Используется для передачи данных между клиентом и сервером при запросе данных о студентах
* Формат передачи данных: JSON.
* Соединение: HTTP-соединение открывается на каждый запрос и закрывается после получения ответа.

|  |
| --- |
|  |
| Рисунок 4. Демонстрация использование http протокола на React |

***Форматы данных***

JSON:

* Основной формат данных для обмена информацией между клиентом и сервером.
* Используется в HTTP ответах.

|  |
| --- |
|  |
| Рисунок 5. Демонстрация JSON данных |

### Описание API

|  |
| --- |
|  |
| Рисунок 6. Демонстрация api в FastAPI |

API

* Описание: Возвращает список студентов
* Маршрут: GET /api/
* Входные параметры:  page — номер страницы. Позволяет выдавать желаемые записи из базы данных, используя смещение при запросе. per\_page – количество записей на странице. Оба параметра обеспечивают пагинацию
* Ответ: Список студентов в формате JSON и общее количество студентов

Следующий код извлекает из базы данных нужные записи:

|  |
| --- |
|  |
| Рисунок 7. Демонстрация запроса к БД |

Параметр number\_rows – это общее количество студентов. Данное число необходимо для меню навигации по странице, чтобы определить, сколько всего страниц существует, и отобразить нужное количество кнопок

В запросе берется срез записей в нужном количестве и с необходимым смещением, чтобы не выдавать одни и те же данные на разных страницах сайта

***Описание фронтенд части***

Есть 2 основных компонента: компонент, отображающий список студентов и делающий запрос к серверу, и компонент, обеспечивающий пагинацию на странице и использующийся внутри первого компонента

Ниже представлен код из компонента Pagination

|  |
| --- |
|  |
| Рисунок 8. Компонент Pagination |

Демонстрация основного компонента StudList

|  |
| --- |
|  |
| Рис 9.Компонент StudList |

### Скриншоты функционала

|  |
| --- |
|  |
| Рисунок 10. Таблица с записями. 1 страница при делении записей на 10 человек/страница |

|  |
| --- |
|  |
| Рисунок 11. Таблица с записями. 2 страница при делении записей на 10 человек/страница |

|  |
| --- |
|  |
| Рисунок 11. Таблица с записями. 1 страница при делении записей на 15 человек/страница |

|  |
| --- |
|  |
| Рисунок 12. Таблица с записями. 2 страница при делении записей на 15 человек/страница |

### Вывод

В ходе выполнения этой лабораторной работы были освоены технологии FastAPI, SQLAlchemy, ReactJS. Был изучен http протокол и применялись правильные методы.