МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет безопасности информационных технологий**

**Направление подготовки: 10.03.01 Информационная безопасность**

**Образовательная программа: "Информационная безопасность / Information security"**

**Дисциплина:**

**«Web программирование»**

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

**«Разработка Веб сервиса»**

**Выполнил студент(ы):**

группа/поток \_\_\_N3345\_\_\_\_

\_Молитвин Илья Алексеевич\_/\_\_\_\_**Изображение выглядит как зарисовка

Контент, сгенерированный ИИ, может содержать ошибки.**\_\_\_\_\_\_

*ФИО Подпись*

**Проверил:**

\_\_Менщиков Александр Алексеевич\_\_/\_\_\_\_\_\_\_\_\_\_

*ФИО Подпись*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Отметка о выполнении (один из вариантов:*

*отлично, хорошо, удовлетворительно, зачтено)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата*

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc181108318)

[ВВЕДЕНИЕ 3](#_Toc181108319)

[1. Описание архитектуры ПО 4](#_Toc181108320)

[2. Описание структуры базы данных 5](#_Toc181108321)

[3. Описание протокола и форматов передачи данных 6](#_Toc181108322)

[4. Описание API 8](#_Toc181108323)

[5. Тестирование функционала ПО 9](#_Toc181108324)

[ЗАКЛЮЧЕНИЕ 16](#_Toc181108325)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc181108326)

[ПРИЛОЖЕНИЕ 18](#_Toc181108327)

ВВЕДЕНИЕ

Цель лабораторной работы – разработать систему для управления данными студентов, состоящую из трёх компонентов: базы данных, серверной части (API) и клиентского интерфейса.

* **База данных** будет хранить информацию о студентах в PostgreSQL.
* **Серверная часть (API)**, разработанная с использованием FastAPI, обеспечит взаимодействие с базой данных через REST API.
* **Клиентский интерфейс** на React будет предоставлять функционал для отображения и управления данными студентов.

Весь проект будет развернут с использованием Docker для организации работы каждого компонента в отдельном контейнере. Система будет поддерживать создание, редактирование, удаление и поиск данных о студентах.

1. Описание архитектуры ПО;

Архитектура программного обеспечения представляет собой многослойную структуру, состоящую из следующих компонентов:

1. Frontend (Клиентская часть):

-Клиентская часть реализована с использованием React. Она взаимодействует с Backend через HTTP запросы, используя библиотеку axios.

-Веб-страница предоставляет пользователю интерфейс для работы с информацией о студентах (просмотр, добавление, редактирование, удаление).

-Frontend контейнер запускается в Docker в связке с другими сервисами.

2. Backend (Серверная часть):

- Серверная часть построена с использованием FastAPI, что позволяет эффективно обрабатывать HTTP запросы.

- API реализует CRUD операции для работы с данными о студентах и взаимодействует с базой данных PostgreSQL.

- Backend контейнер также запускается в Docker и зависит от контейнера базы данных.

3. Database (База данных):

- Для хранения данных используется PostgreSQL, настроенный внутри Docker-контейнера.

- Схема базы данных включает таблицу студентов с полями для имени, фамилии, отчества, курса, группы и факультета.

- Контейнер с базой данных зависит от настроек переменных окружения для подключения к базе и хранит данные в отдельном томе для сохранности.

4. Docker: - Вся система (Frontend, Backend, Database) работает в изолированных контейнерах с использованием Docker и Docker Compose.

- Все контейнеры связаны друг с другом, используя сервисы, описанные в «docker-compose.yml».

1. Описание структуры базы данных;

База данных PostgreSQL содержит одну основную таблицу `students`, которая хранит информацию о студентах. Структура таблицы следующая:

Таблица «students»:

- id (integer, PRIMARY KEY): Уникальный идентификатор студента.

- surname (varchar): Фамилия студента.

- name (varchar): Имя студента.

- patronymic (varchar): Отчество студента.

- course (integer): Курс, на котором учится студент.

- group (varchar): Группа, в которой состоит студент.

- faculty (varchar): Факультет, к которому принадлежит студент.

Для обеспечения целостности данных и улучшения производительности создаются соответствующие индексы и ограничения (например, уникальность «id»).

1. Описание протокола и форматов передачи данных;

Для обмена данными между клиентской и серверной частями используется протокол HTTP. Все данные передаются в формате JSON. Запросы к серверу осуществляются с помощью метода GET, POST, PATCH и DELETE: - GET /students: Получение списка студентов с поддержкой фильтров по фамилии, лимитации и пропуска. - POST /students: Создание нового студента в базе данных. - PATCH /students/{id}: Обновление информации о студенте. - DELETE /students/{id}: Удаление студента.

1. Описание API;

API приложения включает следующие эндпоинты:

1. **GET /students**:
   * Описание: Получение списка студентов с возможностью фильтрации по фамилии, лимитации и пропуску.
   * Параметры:
     + skip (int): Сколько записей пропустить (по умолчанию 0).
     + limit(int): Ограничение на количество записей (по умолчанию 10).
     + surname (string): Фильтрация по фамилии (необязательный параметр).
2. **POST /students**:
   * Описание: Создание нового студента.
   * Тело запроса:

json

{

"surname": "Иванов",

"name": "Иван",

"patronymic": "Иванович",

"course": 3,

"group": 33502",

"faculty": "ИТ"

}

1. **PATCH /students/{id}**:
   * Описание: Обновление данных студента по ID.
   * Параметры:
     + id (int): Уникальный идентификатор студента.
   * Тело запроса: Такие же параметры, как и для POST.
2. **DELETE /students/{id}**:
   * Описание: Удаление студента по ID.
   * Параметры:
     + id (int): Уникальный идентификатор студента.
3. Скриншоты функционала;

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Запущенный docker-контейнер

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – Главная страница с таблицей студентов.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Форма добавления нового студента.

Изображение выглядит как снимок экрана, текст, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 – Форма редактирования информации о студенте.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Подтверждение удаления студента.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – Логи контейнера students-backend.

ЗАКЛЮЧЕНИЕ

В ходе лабораторной работы было разработано веб-приложение «админка» для управления базой данных студентов, состоящее из трёх микросервисов:

1. **Backend-сервис** – реализован на языке Python с использованием фреймворка FastAPI, обеспечивающий REST API для работы с базой данных. Поддерживает операции просмотра, создания, редактирования и удаления записей, а также фильтрацию и пагинацию. Взаимодействие с базой данных реализовано с помощью ORM-библиотеки SQLAlchemy.
2. **Frontend-сервис** – веб-интерфейс, разработанный на React, предоставляющий удобный пользовательский интерфейс для работы с базой студентов. Данные запрашиваются с API с помощью REST-запросов.
3. **База данных** – используется PostgreSQL для хранения информации о студентах.

Все сервисы были упакованы в Docker-контейнеры, а их развёртывание осуществляется с помощью docker-compose.yml. Настроена поддержка персистентного хранения данных. Фронтенд-сервис раздаётся через веб-сервер Nginx, а API доступен по порту 3000.

Таким образом, в результате работы была разработана и развёрнута админ-панель для управления студентами, поддерживающая пагинацию, CRUD-операции, фильтрацию данных и развертываемая в контейнерной среде.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. FastAPI Documentation. [Электронный ресурс] // URL: [FastAPI](https://fastapi.tiangolo.com/).
2. PostgreSQL Documentation. [Электронный ресурс] // URL: <https://www.postgresql.org/docs/13/index.html>.
3. Docker Documentation. [Электронный ресурс] // URL: https://docs.docker.com/get-started/overview/.
4. Лёгкий, кроссплатформенный HTTP-сервер для раздачи статических файлов. [Электронный ресурс] //  
   <https://www.npmjs.com/package/http-server>
5. PostgreSQL Documentation. [Электронный ресурс] // URL: <https://www.postgresql.org/docs/13/index.html>.