Содержание

[Введение 2](#_Toc9527)

[1. Создание проекта на GitHub 3](#_Toc1679)

[2. Техническое задание на разработку базы данных 4](#_Toc20769)

[3. Модуль приёма данных 5](#_Toc28236)

[4. Тестовые наборы данных для запросов GET и POST 8](#_Toc1561)

[5. Настройка Linter 10](#_Toc32505)

[Заключение 12](#_Toc26868)

[Список используемых источников 13](#_Toc11486)

## Введение

Целью производственной практики является применение теоретических знаний, полученных в процессе обучения, составление технического задания для базы данных, написание API с последующим тестированием с помощью специальных программ, а также добавление проекта на GitHub.

## Создание проекта на GitHub

Перед тем, как создать проект на GitHub необходимо зарегистрироваться или войти в уже существующий аккаунт. В правом верхнем углу экрана нажмите кнопку «+» и выберите пункт «New repository». Заполните название, краткое описание, выберете видимость проекта (публичный или приватный). На рисунке 1 представлен созданный проект.

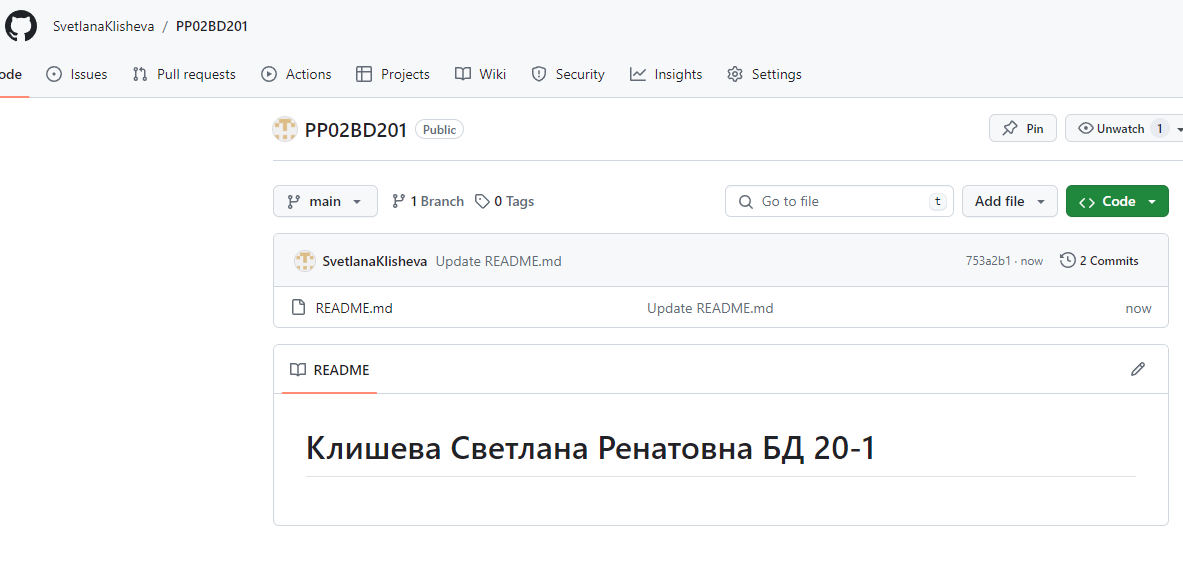


Рисунок 1 – Профиль GitHub с проектом

Ссылка на профиль GitHub:

<https://github.com/SvetlanaKlisheva/PP02BD201>

## Техническое задание на разработку базы данных

Разработка технического задания была реализована в соответствии с ГОСТ 34.602-89 на тему «Музыкальная школа».

ГОСТ 34.602-89 - это ГОСТ СССР, утверждённый в 1989 году и называется "Автоматизированные системы. Основные понятия". Он устанавливает основные термины и определения, используемые в области автоматизированных систем, включая термины, относящиеся к базам данных.

Некоторые из ключевых терминов и определений, которые могут быть включены в техническое задание по ГОСТ 34.602-89 для базы данных, включают следующее:

– База данных

– Таблица базы данных

– Поле базы данных

– Запись базы данных

– Ключ базы данных

– Связь таблиц базы данных

– Индекс базы данных

– Транзакция базы данных

– Уникальность данных

– Целостность данных

Таким образом, при разработке техническое задание для базы данных согласно ГОСТ 34.602-89, следует учитывать данные понятия и определения, чтобы обеспечить корректное понимание требований к базе данных и её функциональности.

Ссылка на техническое задание :

<https://github.com/SvetlanaKlisheva/PP02BD201/blob/main/ТЗмш.docx>

## Модуль приёма данных

Для модуля приема данных был использован FastApi.

FastAPI - это современный и быстрый веб-фреймворк для создания API используя Python, в основе которого лежит стандартная аннотация типов Python. Данным веб-фреймворком активно использует декораторы, аннотации типов и интроспекцию кода, что позволяет уменьшить количество шаблонного кода.

Использовать FastAPI стоит:

1. Когда в проекте нужна производительность и она должна быть выше, чем у других фреймворков.
2. Когда в проекте нужна асинхронность.
3. Если в проекте используются запросы к другим сервисам.
4. Если используется websockets.

Асинхронное программирование - это потоковая обработка ПО, где приложение, а не процессор, управляет потоками и переключением контекста. Когда пользователь запускает что-то асинхронно, это означает, что оно не блокируется, пользователь продолжает выполнять действие, не дожидаясь завершения и продолжает выполнять другие вещи.

Для работы с fastApi, его необходимо установить, сделать это можно с помощью команды представленной на рисунке 2. Команда устанавливает также и все нужные библиотеки.

изображение_2024_04_19_214851915_negate

Рисунок 2 – Установка FastApi

Также необходимо установить модуль psycopg2 для работы с PostgreSQL (рисунок 3).

Для работы с базой данных PostgreSQL можно использовать платформу PgAdmin 4. Это бесплатная открытая платформа управления базами данных, которая предоставляет удобный интерфейс для работы с различными типами баз данных, такими как PostgreSQL, MySQL, SQLite и другие. pgAdmin обеспечивает возможность создания, удаления и изменения баз данных, таблиц, индексов, хранимых процедур, а также выполнения SQL-запросов и мониторинга производительности баз данных. Это мощный инструмент для администрирования баз данных, который поддерживает множество функций и расширений для эффективной работы с данными.

Для того, чтобы приступить к созданию БД, необходимо установить необходимые библиотеки, название библиотек указаны на рисунке 4.

изображение_2024_04_19_220117702_negate

Рисунок 5 – Создание таблицы users

Создаем базу данных и таблицу users для приема данных (рисунок 5).

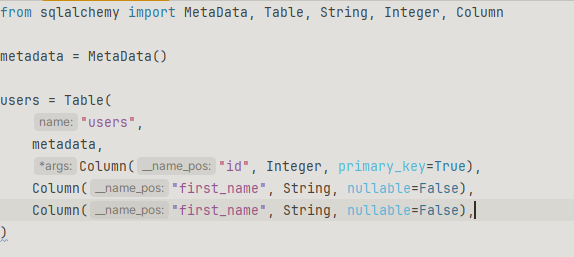


Рисунок 5 – Создание таблицы users

Далее представлен код для приема и обработки данных проверки на корректность и последующей записи в базу данных на основе PostgreSQL (рисунок 6).

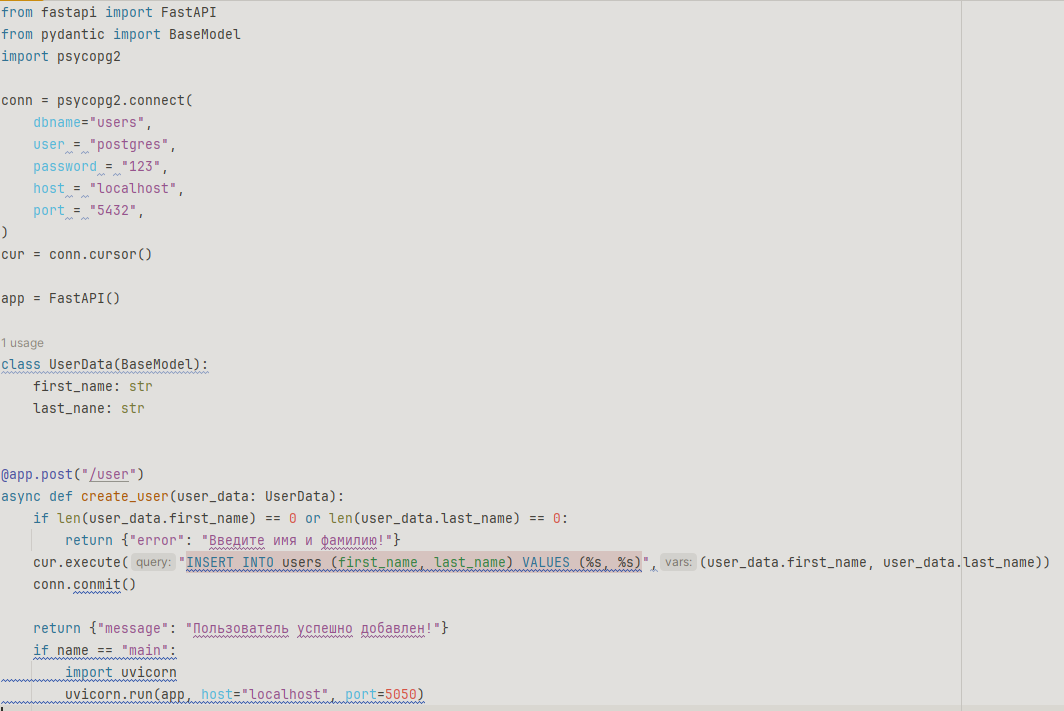


Рисунок 6 – Код для приема и обработки данных

Для реализации приема данных необходимо выполнить подключение к базе данных, далее прописать форму данных для POST–запроса.

Следующим этапом проверяется корректность введенных данных с последующим добавлением их в бд.

Для запуска сервера используется pyton main.py (рисунок 7).



Рисунок 7 – Успешный запуск сервера

Ссылка на проект GitHub:

https://github.com/SvetlanaKlisheva/PP02BD201/blob/main/main.py

## Тестовые наборы данных для запросов GET и POST

Методы GET и POST являются основными методами HTTP запросов, которые используются для отправки данных на сервер.

1. Метод GET:

– Метод GET используется для запроса данных с сервера. Он передает параметры в URL запроса.

– Он является идемпотентным, что означает, что при нескольких одинаковых запросах результаты должны оставаться неизменными.

– Данные, передаваемые через метод GET, ограничены длиной URL и могут быть просмотрены в адресной строке браузера.

– Пример использования метода GET: получение списка пользователей, фильтрация данных и т.д.

2. Метод POST:

– Метод POST используется для отправки данных на сервер для создания или обновления ресурсов.

– Данные, передаваемые через метод POST, могут быть скрыты от просмотра (например, при отправке формы).

– Он не является идемпотентным, что означает, что при нескольких одинаковых запросах могут быть получены разные результаты.

– Пример использования метода POST: отправка данных формы, создание нового пользователя и т.д.

При разработке веб-приложений, сервер должен правильно обрабатывать как метод GET, так и метод POST, чтобы эффективно обрабатывать запросы от клиентов и взаимодействовать с базой данных или другими ресурсами.

Для реализации набора тестовых данных использовался Postman.

Postman – это инструмент для тестирования и разработки API. Он позволяет создавать, отправлять и тестировать HTTP запросы к различным эндпоинтам API, а также автоматизировать процессы тестирования API. Postman также предоставляет возможность сохранять запросы, создавать коллекции запросов и обмениваться ими с другими пользователями.

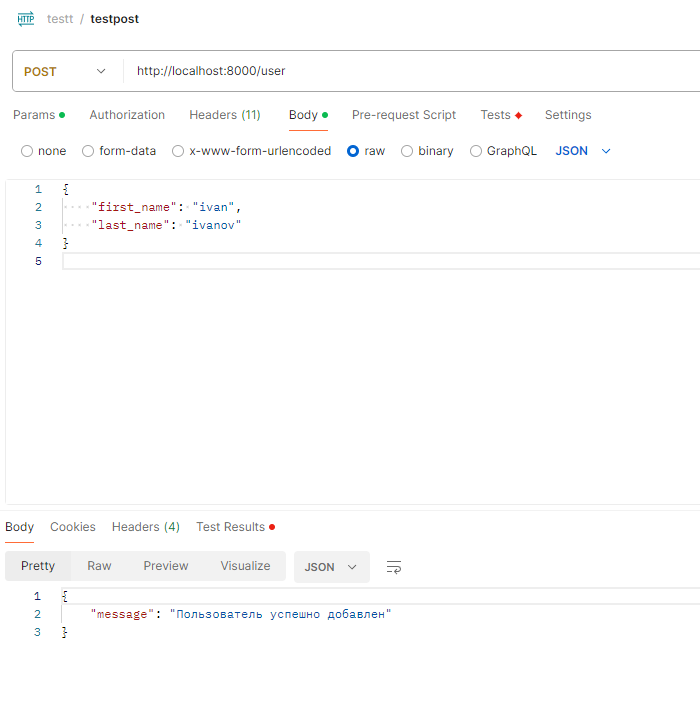


Рисунок 8 – Тестовые данные для Post



Рисунок 9 – Успешное выполнение

## Настройка Linter

Настройка Linter в Python может быть осуществлена с использованием инструментов, таких как Flake8, Pylint или Black.

Flake8 – инструмент статического анализа кода для языков программирования Python, который помогает разработчикам выявлять и исправлять ошибки, следовать стилю кодирования и соблюдать принятые стандарты написания кода. Flake8 проверяет код на соответствие PEP 8 – официальному стандарту написания кода на Python, а также на другие типичные ошибки и неоптимальные ситуации.

Pylint – это еще один инструмент статического анализа кода для языка программирования Python. В отличие от Flake8, который фокусируется преимущественно на стиле кода и соответствии стандартам, Pylint обнаруживает более широкий спектр потенциальных проблем в коде, таких как ошибки синтаксиса, неиспользуемые переменные, несоответствие PEP 8, некорректное использование функций и многое другое.

Pylint также предоставляет детальные отчеты о найденных проблемах и предлагает рекомендации по их устранению.

Black – это инструмент автоматического форматирования кода для языка программирования Python. Этот инструмент задает определенные стандарты форматирования кода, основанные на принципах PEP 8, и автоматически приводит код к соответствию этим стандартам. Black помогает улучшить читаемость и стиль кода, делая его более однородным и согласованным.

Для настройки Linter был выбран pylint. Установить его можно с помощью установщика pip (рисунок 10).

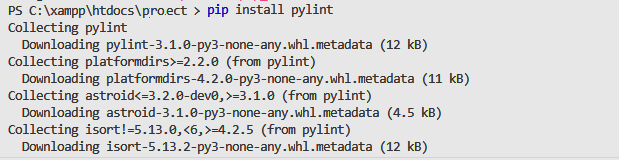


Рисунок 10 – Установка Pylint

Далее создаем файл .pylintrc в корневой директории проекта и добавляем в него конфигурацию.

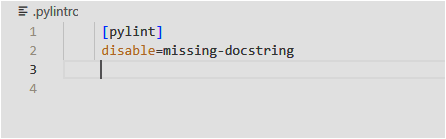


Рисунок 11 – Добавление конфигурации

Отключение предупреждения "missing-docstring" в pylint используется для игнорирования предупреждений о том, что отсутствует строка документации (docstring) для функций, классов и модулей в вашем коде.

Запускаем Pylint (рисунок 11).

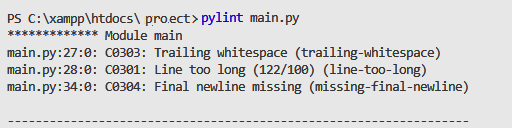


Рисунок 12 – Запуск Pylint

Заключение

Исходя из целей производственной практики, были изучены ГОСТ 34.602-89 на написание технического задания для базы данных. С помощью FastApi был написан модуль приема данных извне, с последующим записыванием их в базу данных PostgreSQL. А также были реализованы наборы тестовых данных для метода Post и Get.

В ходе прохождения практики были закреплены теоретические знания, полученные в техникуме, также закреплены знания работы с программным обеспечением.

Список используемых источников

1. studbooks.net – Техническое задание на разработку базы данных  URL:  https://studbooks.net/2176887/informatika/tehnicheskoe\_zadanie\_razrabotku\_bazy\_dannyh\_sostavlennoe\_sootvetstvii\_gost34602
2. reqres.in – API для Postman URL: <https://reqres.in/>
3. pythonru.com – Первое FastApi - приложение URL:  https://pythonru.com/biblioteki/znakomstvo-s-fastapi