Создание RESTful API с использованием FastAPI

Цель работы: Получить практические навыки разработки RESTful API с использованием FastAPI, включая обработку различных типов данных, отправку ответов, работу с заголовками и куками.

Задание:

Создание проекта FastAPI:

Создайте новый проект FastAPI с использованием любого удобного вам инструмента (например, pip, poetry, pipenv).

Установите необходимые зависимости.

Создание базового API:

Создайте экземпляр класса FastAPI.

Определите маршрут / с помощью декоратора @app.get("/"), который будет возвращать простое сообщение "Hello, World!".

Запустите сервер и убедитесь, что API доступен по адресу http://127.0.0.1:8000/.

Обработка параметров:

Создайте маршрут /greet/{name}, который принимает параметр пути name и возвращает приветствие "Hello, {name}!".

Создайте маршрут /search, который принимает параметр строки запроса query и возвращает сообщение "You searched for: {query}".

Отправка различных типов данных:

Создайте маршрут /json, который возвращает JSON-ответ с данными о вас (имя, возраст, хобби).

Создайте маршрут /file, который отправляет текстовый файл с произвольным содержимым.

Создайте маршрут /redirect, который выполняет перенаправление на маршрут /.

Работа с заголовками и куками:

Создайте маршрут /headers, который возвращает все заголовки запроса в виде JSON.

Создайте маршрут /set-cookie, который устанавливает куку с именем username и значением your\_name.

Создайте маршрут /get-cookie, который возвращает значение куки username.

Обработка данных запроса:

Создайте маршрут /login, который принимает данные формы с полями username и password и возвращает сообщение "Welcome, {username}!".

Создайте маршрут /register, который принимает JSON-данные с полями username, email и password и возвращает сообщение "User {username} registered successfully!".

Работа с классами:

Создайте класс User с полями id, username, email и password.

Создайте маршрут /users, который возвращает список объектов класса User в формате JSON.

Создайте маршрут /users/{id}, который возвращает объект класса User с указанным id.

Дополнительные задания (по желанию):

Реализуйте валидацию данных на маршрутах /login и /register.

Добавьте возможность загрузки файлов на сервер.

Реализуйте аутентификацию пользователей с использованием JWT-токенов.

Цель работы:

Освоить основные принципы работы с базой данных через SQLAlchemy: подключение к базе данных, создание таблиц, выполнение запросов и интеграция с веб-приложением.

Задание:

Часть 1: Подключение к базе данных и создание таблиц

Выбор базы данных:

Выберите одну из баз данных: MSSQL, SQLite, PostgreSQL, MySQL.

Установите необходимые библиотеки для работы с выбранной базой данных и SQLAlchemy.

Создание модели данных:

Опишите модель данных, состоящую из двух таблиц: Users и Posts.

Таблица Users должна содержать следующие поля:

id (целое число, первичный ключ, автоинкремент)

username (строка, уникальное значение)

email (строка, уникальное значение)

password (строка)

Таблица Posts должна содержать следующие поля:

id (целое число, первичный ключ, автоинкремент)

title (строка)

content (текст)

user\_id (целое число, внешний ключ, ссылающийся на поле id таблицы Users)

Создание таблиц:

Напишите программу на Python, которая подключается к выбранной базе данных и создает таблицы Users и Posts на основе описанной модели данных.

Часть 2: Взаимодействие с базой данных

Добавление данных:

Напишите программу, которая добавляет в таблицу Users несколько записей с разными значениями полей username, email и password.

Напишите программу, которая добавляет в таблицу Posts несколько записей, связанных с пользователями из таблицы Users.

Извлечение данных:

Напишите программу, которая извлекает все записи из таблицы Users.

Напишите программу, которая извлекает все записи из таблицы Posts, включая информацию о пользователях, которые их создали.

Напишите программу, которая извлекает записи из таблицы Posts, созданные конкретным пользователем.

Обновление данных:

Напишите программу, которая обновляет поле email у одного из пользователей.

Напишите программу, которая обновляет поле content у одного из постов.

Удаление данных:

Напишите программу, которая удаляет один из постов.

Напишите программу, которая удаляет пользователя и все его посты.

Часть 3: Базовые операции с базой данных в веб-приложении

Создание веб-приложения:

Создайте простое веб-приложение на FastAPI.

Интегрируйте SQLAlchemy в ваше веб-приложение.

Реализация CRUD-операций:

Реализуйте веб-страницы для выполнения CRUD-операций (создание, чтение, обновление, удаление) с записями в таблицах Users и Posts.

Страницы должны включать:

Форму для создания нового пользователя/поста.

Список всех пользователей/постов с возможностью редактирования и удаления.

Страницу для редактирования информации о пользователе/посте.

Цель работы:

Получить практические навыки работы с фреймворком Django: установка и настройка, создание проекта и приложения, работа с представлениями, маршрутизацией, обработкой запросов и отправкой ответов.

Задание:

Часть 1: Настройка проекта и создание приложения

Установка Django:

Установите Django на свою локальную машину с помощью pip.

Проверьте установку, выполнив команду django-admin --version.

Создание проекта:

Создайте новый проект Django с именем my\_first\_django\_project.

Перейдите в директорию проекта и запустите сервер разработки командой python manage.py runserver.

Откройте браузер и убедитесь, что сервер работает, перейдя по адресу http://127.0.0.1:8000/.

Создание приложения:

Создайте новое приложение с именем my\_first\_app внутри проекта.

Зарегистрируйте приложение в файле settings.py проекта.

Часть 2: Работа с представлениями и маршрутизацией

Создание простого представления:

В файле views.py приложения my\_first\_app создайте функцию-представление hello\_world, которая возвращает простой HTML-ответ: <h1>Hello, World!</h1>.

Создайте маршрут для этого представления в файле urls.py приложения, используя функцию path.

Проверьте работу представления, перейдя по соответствующему URL в браузере.

Обработка параметров запроса:

Измените представление hello\_world, чтобы оно принимало параметр name из URL и возвращало ответ <h1>Hello, {name}!</h1>.

Создайте маршрут с параметром в файле urls.py, используя функцию path или re\_path.

Проверьте работу представления, перейдя по URL с разными значениями параметра name.

Получение данных из строки запроса:

Измените представление, чтобы оно также принимало параметр age из строки запроса и возвращало ответ <h1>Hello, {name}! You are {age} years old.</h1>.

Проверьте работу представления, перейдя по URL с параметрами в строке запроса.

Переадресация и отправка статусных кодов:

Создайте новое представление redirect\_example, которое перенаправляет пользователя на представление hello\_world с параметрами по умолчанию.

Используйте функцию redirect для перенаправления и HttpResponse для отправки статусного кода 302.

Проверьте работу переадресации, перейдя по соответствующему URL.

Отправка JSON-ответа:

Создайте новое представление json\_example, которое возвращает JSON-ответ с данными о пользователе (например, имя и возраст).

Используйте функцию JsonResponse для отправки JSON-ответа.

Проверьте работу представления, перейдя по соответствующему URL и просмотрев ответ в инструментах разработчика браузера.

Работа с куками:

Измените представление hello\_world, чтобы оно устанавливало куку с именем пользователя.

Создайте новое представление show\_cookies, которое отображает все куки, установленные в браузере.

Проверьте работу представлений, перейдя по соответствующим URL и просмотрев куки в инструментах разработчика браузера.

Часть 3: Вложенные маршруты и функция include

Создание вложенных маршрутов:

Создайте новое приложение nested\_app и зарегистрируйте его в проекте.

В приложении nested\_app создайте несколько представлений и маршрутов.

Используйте функцию include в файле urls.py проекта, чтобы включить маршруты из приложения nested\_app в общий маршрут проекта.

Проверьте работу вложенных маршрутов, перейдя по соответствующим URL.