Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 7-8
по дисциплине «Программирование на языке Python»

Выполнил: Никулина Злата Евгеньевна,
студент группы ИУ8-13М

Проверил: Зотов Михаил Владиславович

г. Москва, 2025г.

1. ЦЕЛЬ РАБОТЫ

Цель: Реализовать API для функционала приложения по варианту.

2. ПОСТАНОВКА ЗАДАЧИ

Необходимо:

- Ваше API использует асинхронное подключение к PostgreSQL через asyncpg
- Ваши API строит ORM-модели на SQLAlchemy 2.0
- Ваш API применяет валидацию и сериализацию через Pydantic (v2)
- Ваш REST API доступен через FastAPI
- Ваш API запускается через uvicorn (например: uvicorn main:app --reload), либо поднимается в докер контейнере

Описание:

Пользователи регистрируют привычки (например, «пить воду», «делать зарядку») и отмечают их выполнение по дням.

Основные сущности:

- habit (id, user_id, name, description)
- completion (id, habit_id, date)

Эндпоинты:

- создание привычки
- отметка выполнения
- получение статистики (сколько дней подряд выполнено и т.п.).

3. ХОД РАБОТЫ

Ход работы:

Шаг 1. Осознание задачи

Разработке RESTful API на FastAPI для управления сущностями (привычками), что включает создание эндпоинтов для добавления, получения и анализа данных, а также обеспечение корректного взаимодействия с PostgreSQL через асинхронный SQLAlchemy ORM; при этом необходимо соблюдать многослойную архитектуру (роутеры, Pydantic-модели для валидации, ORM-модели для БД, операции с данными), использовать автоматическую генерацию документации, гарантировать целостность данных с помощью внешних ключей и организовать жизненный цикл приложения для автоматического создания и удаления таблиц при запуске и остановке сервиса.

Шаг 2. Описание действий

1. Подготовка окружения: создано виртуальное окружение Python и установлены все необходимые зависимости из файла requirements.txt, включая FastAPI, SQLAlchemy, asyncpg и uvicorn.

2. Проектирование архитектуры: реализована многослойная структура проекта, включающая: Pydantic-модели (store_models.py) для валидации входных и выходных данных API; ORM-модели (database_models.py) для отображения таблиц базы данных на Python-объекты; модуль операций с базой данных (database_operations.py) с асинхронными методами добавления и получения сущностей; файл миграций (database_migrations.py) для автоматического создания и удаления таблиц при запуске и остановке приложения; отдельный роутер (router_habit.py), объединяющий все эндпоинты, связанные с управлением привычками.

3. Реализация бизнес-логики:

- добавления новой привычки;
- отметки выполнения привычки в указанную дату;

- получения списка привычек пользователя;
- расчёта серии выполнений для конкретной привычки.

4. Настройка взаимодействия с PostgreSQL: использован асинхронный драйвер asyncpg, настроено подключение к базе данных через create_async_engine, реализованы сессии с помощью async_sessionmaker, а также добавлены внешние ключи (ForeignKey) для обеспечения ссылочной целостности данных.

5. Интеграция компонентов: в основном файле main.py настроен жизненный цикл приложения для автоматической инициализации БД, подключён роутер с эндпоинтами, обеспечена автоматическая генерация OpenAPI-документации.

Шаг 3. Реализация, результат

На рисунке 1 показан вид поля для добавления привычки. Пользователь должен ввести название для привычки и ее описание, а также id.

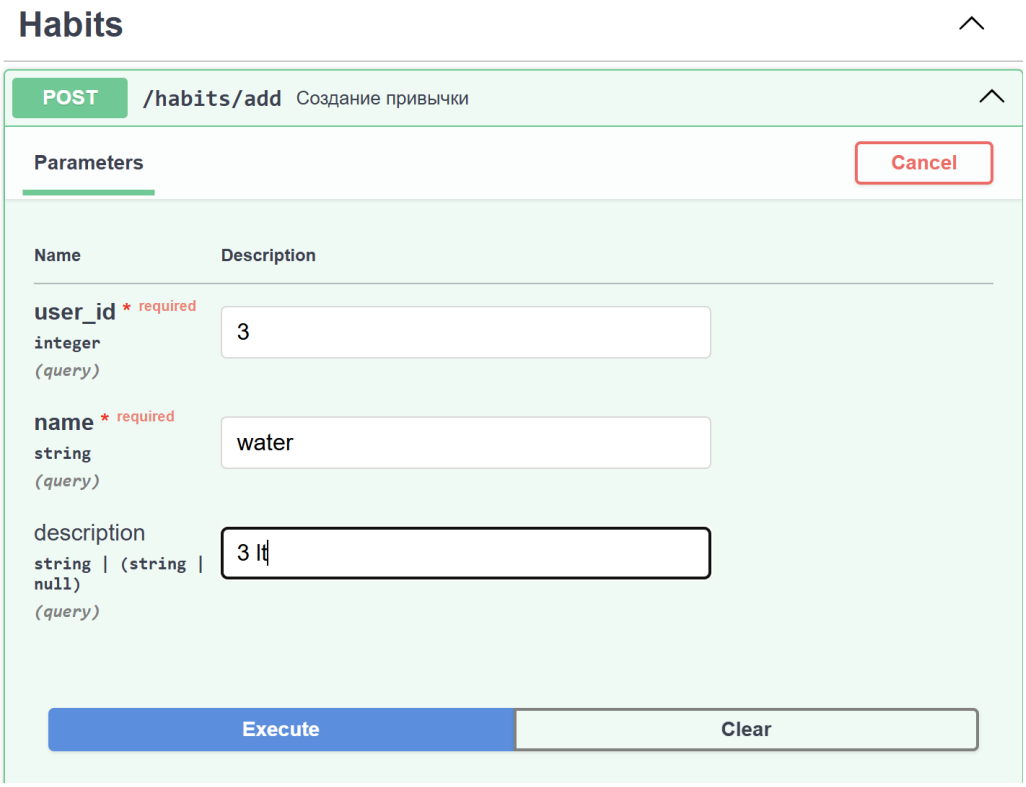

Рисунок 1 – Создание привычки

На рисунке 2 показан результат для вывода всех привычек.

Рисунок 2 – Вывод всех введенных привычек

На рисунке 3 показано, каким образом происходит отмена выполнения привычки. Пользователь вводит id и время выполнения привычки.

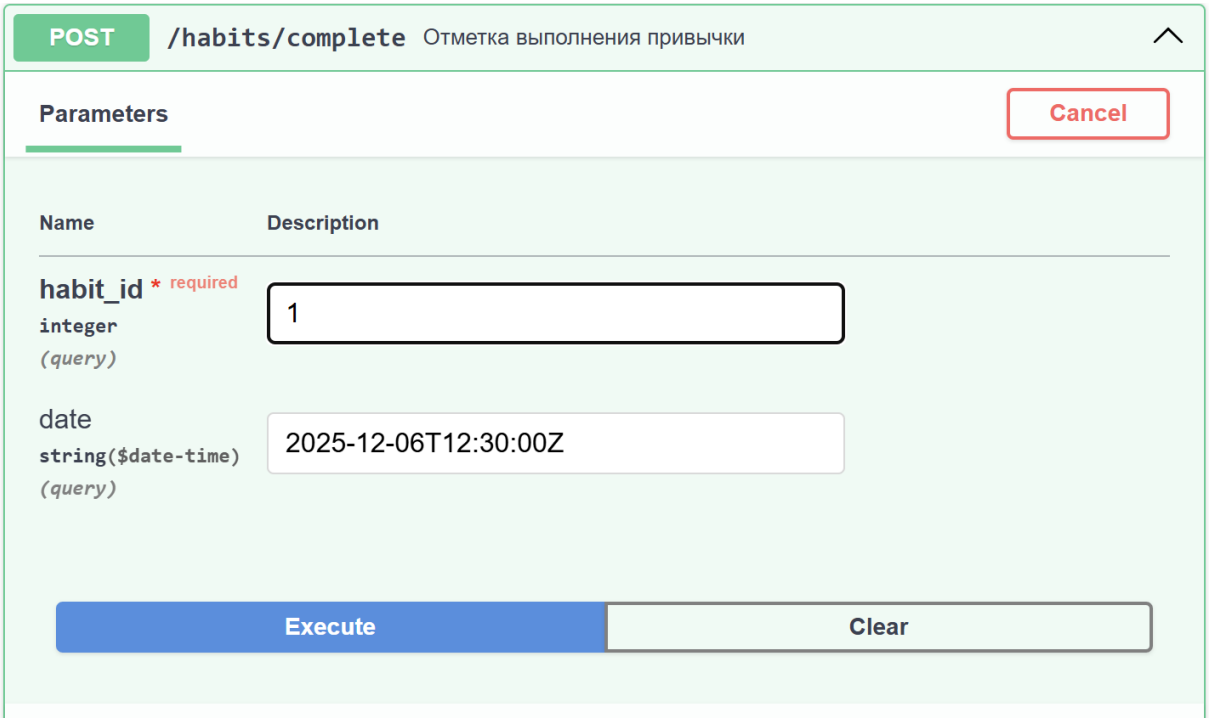

Рисунок 3 – Отметка выполнения привычки

На рисунке 4 видно, что все данные о привычках выводятся корректно. Серия подряд выполненный дней составляет 6, результат на рисунке 5 подтверждает это.

200

Response body


```
[
  {
    "habit_id": 1,
    "date": "2025-12-01T12:30:00Z",
    "id": 1
  },
  {
    "habit_id": 1,
    "date": "2025-12-02T12:30:00Z",
    "id": 2
  },
  {
    "habit_id": 1,
    "date": "2025-12-03T12:30:00Z",
    "id": 3
  },
  {
    "habit_id": 1,
    "date": "2025-12-04T12:30:00Z",
    "id": 4
  },
  {
    "habit_id": 1,
    "date": "2025-12-05T12:30:00Z",
    "id": 5
  },
  {
    "habit_id": 1,
    "date": "2025-12-06T12:30:00Z",
```


Download

Response headers

Рисунок 4 – Получение всех выполненных привычек

Code Details

200

Response body


```
{
  "streak": 6
}
```


Download

Рисунок 5 – Вывод серии подряд выполненных дней

4. ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было успешно разработано RESTful API на основе FastAPI, реализующее управление привычками пользователя: добавление новых привычек, отметка их выполнения, получение списка и расчёт серии выполнений. Архитектура приложения построена с чётким разделением слоёв (Pydantic-модели, ORM-модели, операции с БД, роутеры), использованием асинхронного взаимодействия с PostgreSQL через SQLAlchemy и автоматической генерацией документации. Были обеспечены валидация данных, целостность БД за счёт внешних ключей и жизненный цикл приложения.