**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по практической работе**

по дисциплине «Введение в информационные технологии» на тему:

Основы работы с FastAPI

Выполнил: студент группы БПИ2401 Дьяченко Максим

Проверил: Мкртчян Грач Маратович

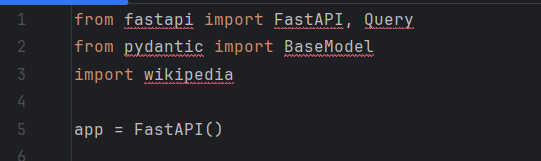
Москва

2025

Цель работы: Получить практические навыки в использовании FastAPI для тестирования и разработки API, изучить основные методы HTTP, научиться создавать и отправлять запросы, а также анализировать ответы сервера.

Выполнение:

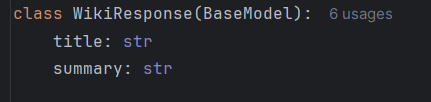
Установим зависимости(перед этим убедившись, что библиотеки загружены)



FastAPI — главный класс для создания приложения.

Query — специальная функция, позволяющая работать с query-параметрами в адресной строке,

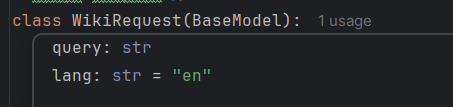
Импортируем BaseModel из pydantic — это основа для описания данных. С помощью него мы описываем, какие именно данные мы ожидаем в теле запроса или в ответе. Он автоматически проверяет правильность данных.



Создаю модель ответа, которую будет возвращать API:

title: str — заголовок статьи.

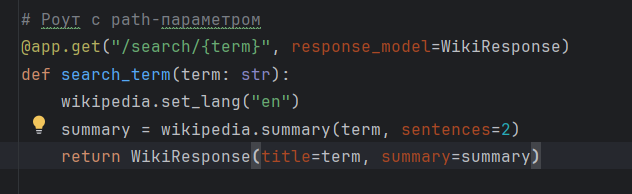
summary: str — краткое описание (резюме) статьи.



Создаю модель запроса, которая описывает, что мы ожидаем в теле запроса(при Post-запросе)

query: str — что пользователь хочет найти.

lang: str = "en" — язык запроса по умолчанию английский. Можно поменять, например, на "ru" для русского.



**@app.get("/search/{term}", response\_model=WikiResponse)**

Это декоратор, который:

делает функцию search\_term доступной по адресу /search/что\_угодно\_здесь,

автоматически вызывает её, когда пользователь делает GET-запрос,

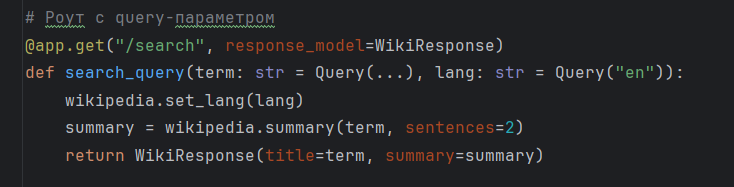
response\_model=WikiResponse — говорит FastAPI, что ответ должен соответствовать модели WikiResponse.

**def search\_term(term: str):**

Объявление функции. Она принимает параметр term, который берется из пути URL (/search/term\_тут), и автоматически интерпретируется как строка (str).

**summary = wikipedia.summary(term, sentences=2)**

Выполняем поиск статьи по term и сохраняем краткое описание ,т е саммари — первые 2 предложения.



**def search\_query(term: str = Query(...), lang: str = Query("en")):**

Объявляем функцию с двумя параметрами:

term — обязательный (... значит "обязательный").

lang — необязательный, по умолчанию "en". Эти параметры берутся из адресной строки.

**summary = wikipedia.summary(term, sentences=2)**

Получаем краткое описание по запросу term.



**@app.post("/search", response\_model=WikiResponse)**

данные передаются в теле запроса, а не в URL.

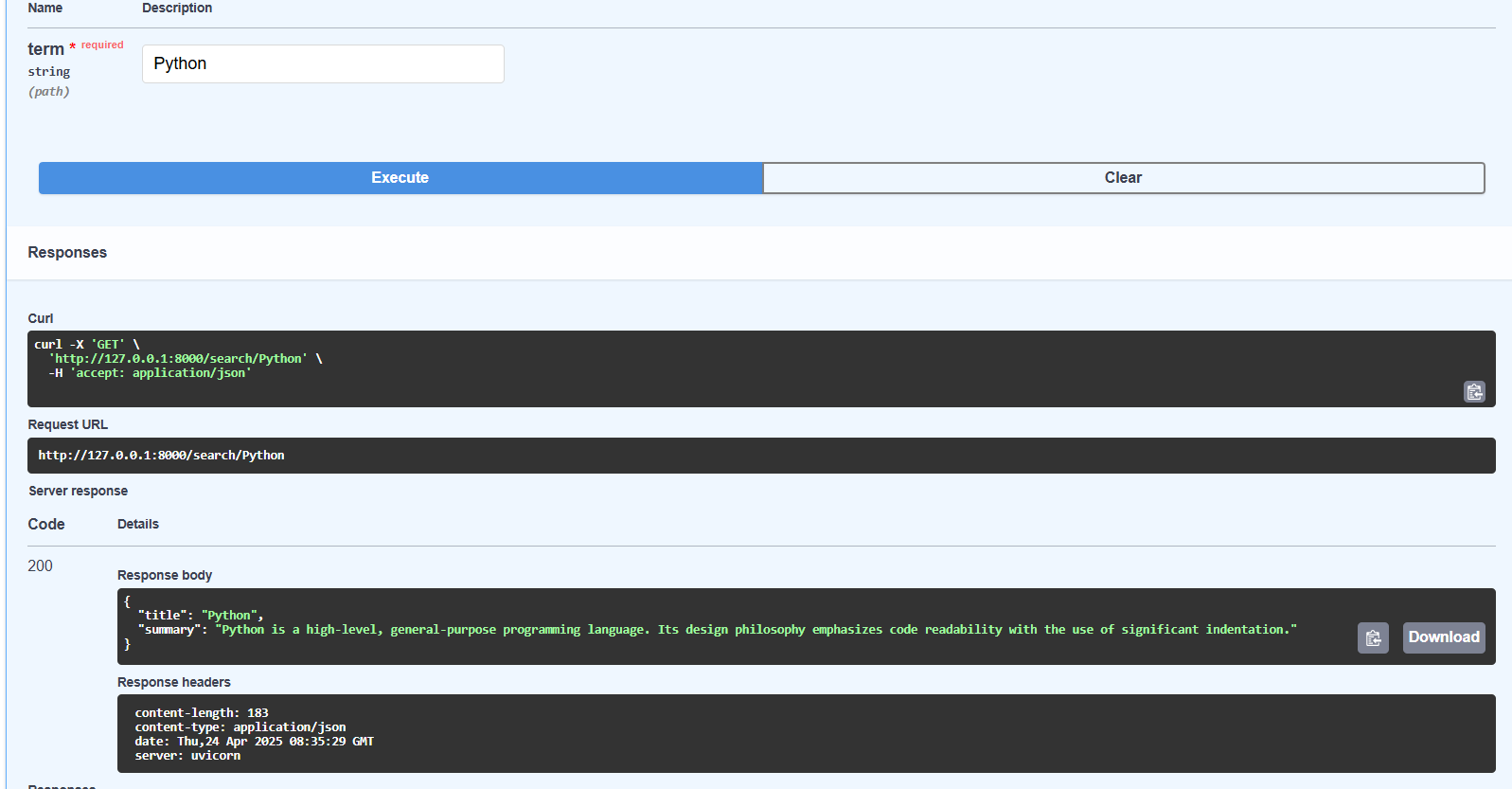
**def search\_post(request: WikiRequest):**

Принимаем объект request, который автоматически будет содержать распарсенные данные из JSON-тела запроса (например: {"query": "Python", "lang": "en"}).

**return WikiResponse(title=request.query, summary=summary)**

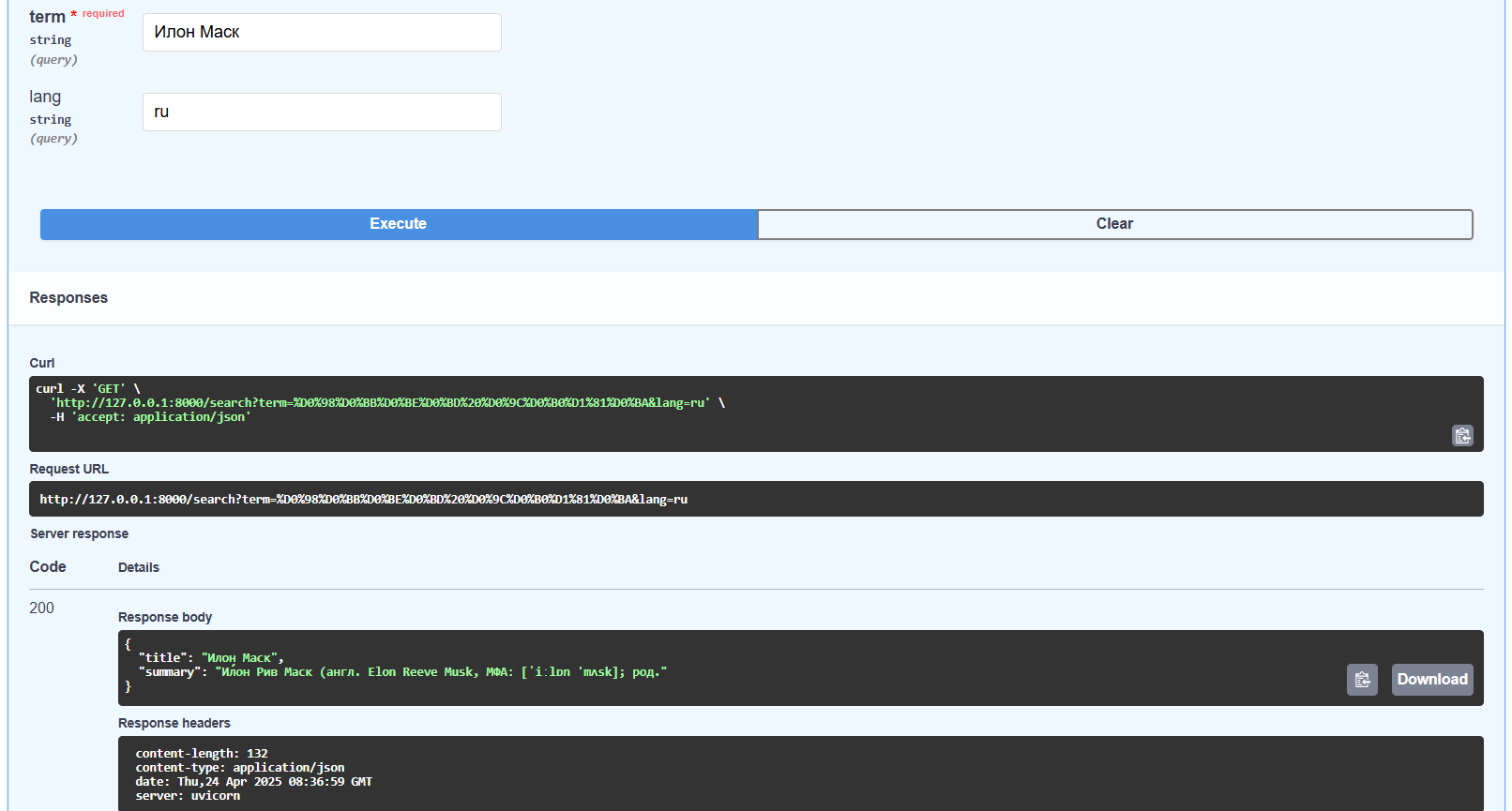
Возвращаем результат — заголовок и описание.

Проверяем 1:



Всё работает

Проверяем 2:



Проверяем 3:

{

"query": "Дмитрий Медведев",

"lang": "ru"

}



Вывод: В ходе лабораторной работы мною были получены практические навыки в использовании FastAPI для тестирования и разработки API, я научился создавать и отправлять запросы, а также анализировать ответы сервера.