**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

Ганзий Владислав Викторович

Проверил: Мкртчян Грач Маратович

Москва

2025

**Цель:**

Освоение FastAPI. Создание простого REST сервиса, использующего библиотеку wikipedia. Создание 1 роута с параметром path, 1 роута с параметром query, 1 роута с передачей параметров в теле запроса. Все запросы должны возвращаться и валидирироваться по схемам.

**Ход работы:**

В ходе выполнения работы был разработан RESTful API с использованием фреймворка **FastAPI** и библиотеки **wikipedia**.  
API предоставляет доступ к данным Wikipedia через HTTP-запросы с поддержкой валидации запросов и ответов.

Для работы проекта были установлены следующие библиотеки:

pip install fastapi uvicorn wikipedia

**FastAPI** – фреймворк для создания API.

**Uvicorn** – ASGI-сервер для запуска FastAPI.

**Wikipedia** – библиотека для работы с Wikipedia API.

Сервер запускается командой:

uvicorn main:app --reload

main:app – указывает на файл main.py и объект app (экземпляр FastAPI).

--reload – автоматическая перезагрузка сервера при изменениях кода.

После запуска API доступен по адресу:  
<http://127.0.0.1:8000>  
**Swagger UI (документация):** <http://127.0.0.1:8000/docs>

**GET /page/{path} (роут с path-параметром)**

**Назначение:** Получение краткого описания статьи Wikipedia по её названию.

**GET /search/ (роут с query-параметром)**

**Назначение:** Поиск статей Wikipedia с возможностью указания количества результатов.

**POST /page/ (роут с телом запроса)**

**Назначение:** Получение полного текста статьи Wikipedia по запросу.

Для валидации запросов и ответов использовались **Pydantic-модели**:

python

from pydantic import BaseModel

class PageSummary(BaseModel):

title: str

summary: str

class SearchInput(BaseModel):

query: str

results: int = 5

class PageContent(BaseModel):

title: str

content: str

**Заключение**

Освоены основы FastAPI, создан простой REST сервис.