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:
FAST API

Выполнил: студент группы БПИ 2403

Рыбаченок Вадим Александрович

Проверил:

Москва

2025

1. **Цель работы:**

Изучить fastapi

2. Ход работы:

Практическое задание

Создать простой REST сервис, использующий библиотеку wikipedia. Создайте 1 роут с параметром path, 1 роут с параметром query, 1 роут с передачей параметров в теле запроса. Все запросы должны возвращаться и валидироваться по схемам.

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import wikipedia

# Инициализация FastAPI и настройка Wikipedia
app = FastAPI()
wikipedia.set_lang("ru")  # Устанавливаем язык (русский)

# Схемы Pydantic
class ArticleResponse(BaseModel):
    title: str
    content: str

class SearchResponse(BaseModel):
    results: list[dict]

class ArticleRequest(BaseModel):
    title: str
    content: str

# Роут с параметром пути (path parameter)
@app.get("/article/{title}", response_model=ArticleResponse)
def get_article(title: str):
    """Получить статью по её названию"""
    try:
        page = wikipedia.page(title)
        return {"title": page.title, "content": page.content}
    except wikipedia.exceptions.PageError:
        raise HTTPException(status_code=404, detail="Статья не найдена")

# Роут с параметром запроса (query parameter)
@app.get("/search", response_model=SearchResponse)
def search_articles(query: str, limit: int = 5):
    """Поиск статей по запросу"""
    results = wikipedia.search(query, results=limit)
    formatted_results = [{"title": title, "summary": wikipedia.summary(title)} for
title in results]
    return {"results": formatted_results}

# Роут с телом запроса (POST + body)
@app.post("/custom-article", response_model=ArticleRequest)
def create_custom_article(article: ArticleRequest):
    """Добавить пользовательскую статью (пример работы с телом запроса)"""
    # Здесь можно добавить логику сохранения статьи в базу данных
    return article
```

3. Вывод:

В ходе работы мы получили большое количество знаний по работе с fastapi.