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»

Лабораторная работа №11

Основы работы с fastapi

Выполнил: Студент группы

БВТ2402

Юдин Владимир

Москва

2025

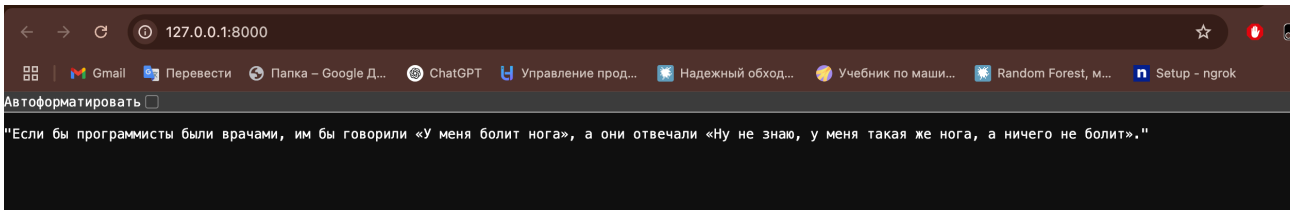
Создаем файл main.py. Импортируем fastapi и pyjokes в main.py
Создадим объект фастапи, куда далее будут подключаться роуты. Также создадим простой роут

main.py

```python
from fastapi import FastAPI
import pyjokes

app = FastAPI()

@app.get("/")
def joke():
    return pyjokes.get_joke(language='ru')

```

Запустим uvicorn

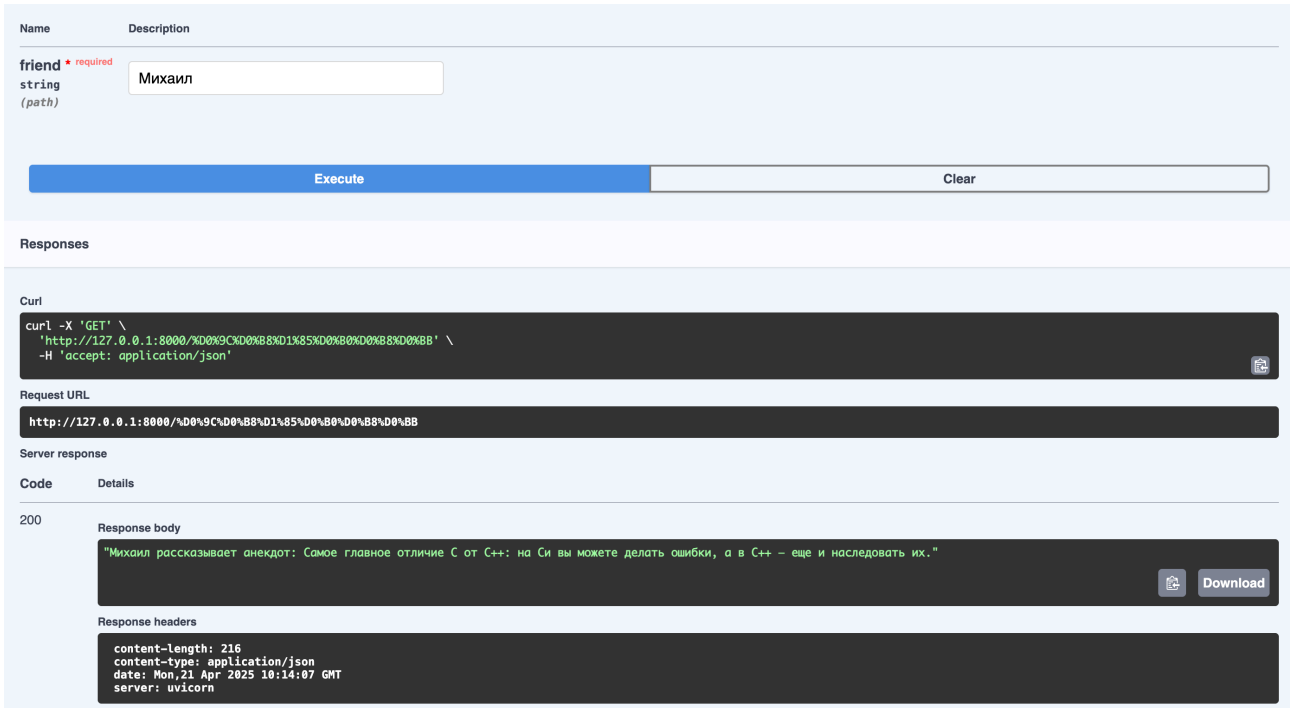
```
(base) vladimir@Noutbuk-Vladimir fast_api_lab % uvicorn main:app --reload
INFO:     Will watch for changes in these directories: ['/Users/vladimir/fast_api_lab']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [18754] using WatchFiles
INFO:     Started server process [18756]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
INFO:     127.0.0.1:60914 - "GET / HTTP/1.1" 200 OK
```



Подключение через swagger

Реализация роута с параметром:

```python
@app.get("/")
def joke():
    return pyjokes.get_joke(language='ru')

@app.get("/{friend}")
def friends_joke(friend: str):
    return friend + " рассказывает анекдот: " + pyjokes.get_joke(language='ru')
```



Реализуем query параметр:

```python
def multi_friends_joke(friend: str, jokes_number: int):
    result = ''
    for i in range(jokes_number):
        result += friend + f' рассказывает анекдот {i + 1}: ' + pyjokes.get_joke(language='ru') + '\n'
    return result
```

Реализация пост запроса:

```python
@app.post("/")
def create_joke(joke_input: JokeInput):
    return joke_input.friend + " рассказывает анекдот: " + pyjokes.get_joke(language='ru')
```

Самостоятельное задание, работа с wikipedia api

wiki_main.py > get_page_info

```python
from fastapi import FastAPI, Query, Path, Body
from pydantic import BaseModel
from typing import List
import wikipedia

app = FastAPI()

class SummaryResponse(BaseModel):
    title: str
    summary: str

class SearchResponse(BaseModel):
    results: List[str]

class PageInfoRequest(BaseModel):
    title: str

class PageInfoResponse(BaseModel):
    title: str
    summary: str
    url: str

@app.get("/summary/{title}", response_model=SummaryResponse)
def get_summary(title: str = Path(..., description="Заголовок статьи")):
    summary = wikipedia.summary(title)
    return {"title": title, "summary": summary}

@app.get("/search", response_model=SearchResponse)
def search_articles(query: str = Query(..., description="Текст для поиска")):
    results = wikipedia.search(query)
    return {"results": results}

@app.post("/pageinfo", response_model=PageInfoResponse)
def get_page_info(info: PageInfoRequest = Body(...)):
    page = wikipedia.page(info.title)
    return {
        "title": page.title,
        "summary": page.summary,
        "url": page.url
    }
```

Роутер с параметром path

Роутер с параметром query

Роутер с передачей параметров в теле запроса

```
{
  "title": "Key (cryptography)"
}
```


Execute | Clear

Responses

Curl


```
curl -X 'POST' \
  'http://127.0.0.1:8000/pageinfo' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "title": "Key (cryptography)"
}'
```


Request URL

```
http://127.0.0.1:8000/pageinfo
```

Server response

Code	Details
200	**Response body**


```
{
  "title": "Public-key cryptography",
  "summary": "Public-key cryptography, or asymmetric cryptography, is the field of cryptographic systems that use pairs of related keys. Each key pair consists of a public key and a corresponding private key. Key pairs are generated with cryptographic algorithms based on mathematical problems termed one-way functions. Security of public-key cryptography depends on keeping the private key secret; the public key can be openly distributed without compromising security. There are many kinds of public-key cryptosystems, with different security goals, including digital signature, Diffie–Hellman key exchange, public-key key encapsulation, and public-key encryption.\nPublic key algorithms are fundamental security primitives in modern cryptosystems, including applications and protocols that offer assurance of the confidentiality and authenticity of electronic communications and data storage. They underpin numerous Internet standards, such as Transport Layer Security (TLS), SSH, S/MIME, and PGP. Compared to symmetric cryptography, public-key cryptography can be too slow for many purposes, so these protocols often combine symmetric cryptography with public-key cryptography in hybrid cryptosystems.\n\n",
  "url": "https://en.wikipedia.org/wiki/Public-key_cryptography"
}
```


Download