

**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

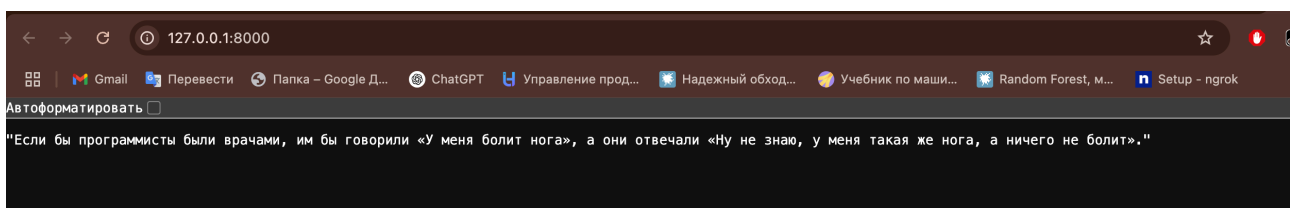
Создаем файл main.py. Импортируем fastapi и pyjokes в main.py  
Создадим объект фастапи, куда далее будут подключаться роуты. Также создадим простой роут

```
main.py × [play] [view] [tabs] [menu]

main.py > ...
1  from fastapi import FastAPI
2  import pyjokes
3
4  app = FastAPI()
5
6  @app.get("/")
7  def joke():
8      return pyjokes.get_joke(language='ru')
9
```

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

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/' \
  -H 'accept: application/json'
```

Request URL

http://127.0.0.1:8000/

Server response

Code	Details
200	<div><div>Response body</div><div>"Если вы посмотрите на код, который вы писали более полугода назад, то, скорей всего, вам покажется, что автор – кто-то другой."</div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 228 content-type: application/json date: Mon, 21 Apr 2025 10:10:26 GMT server: uvicorn</pre></div></div>

Responses

Code	Description	Links
200	Successful Response	No links

# Реализация роута с параметром:

```
@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')
```

Name	Description
friend <span>required</span> string (path)	<div>Михаил</div>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/%D0%9C%D0%B8%D1%85%D0%B8%D0%B8%D0%BB' \
  -H 'accept: application/json'
```

Request URL

http://127.0.0.1:8000/%D0%9C%D0%B8%D1%85%D0%B8%D0%B8%D0%BB

Server response

Code	Details
200	<div><div>Response body</div><div>"Михаил рассказывает анекдот: Самое главное отличие С от С++: на Си вы можете делать ошибки, а в С++ – еще и наследовать их."</div><div><div>Download</div></div></div> <div><div>Response headers</div><div><pre>content-length: 216 content-type: application/json date: Mon, 21 Apr 2025 10:14:07 GMT server: uvicorn</pre></div></div>

## Реализуем query параметр:

```
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
```

Responses

Curl

```
curl -X 'GET' \
'http://127.0.0.1:8000/multi/%D0%98%D0%BB%D1%8C%D1%8F?jokes_number=3' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/multi/%D0%98%D0%BB%D1%8C%D1%8F?jokes_number=3
```

Server response

Code	Details
200	<p>Response body</p> <p>"Илья рассказывает анекдот 1: Самое главное отличие С от C++: на Си вы можете делать ошибки, а в C++ – еще и наследовать их.\Илья рассказывает анекдот 2: Программа получилась плохой, а сроки горят, и заказчик ругается? Не волнуйтесь, смело выпускайте релиз. Просто назовите его версией 1.0.\Илья рассказывает анекдот 3: Если бы программисты были врачами, им бы говорили «у меня болит нога», а они отвечали «Ну не знаю, у меня такая же нога, а ничего не болит»."</p> <p>Response headers</p> <p>content-length: 818 content-type: application/json date: Mon, 21 Apr 2025 10:19:25 GMT server: uvicorn</p>

Responses

## Реализация пост запроса:

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

Responses

Curl

```
curl -X 'POST' \
'http://127.0.0.1:8000/' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "friend": "Максим"
}'
```

Request URL

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

Server response

Code	Details
200	<p>Response body</p> <p>"Максим рассказывает анекдот: Приходит программист к пианисту - посмотреть на новый рояль. Долго ходит вокруг, жмывает, потом заявляет: - Клава неудобная - всего 84 клавиши, п оловина функциональных, ни одна не подписана, хотя... шифт нажимать ногой - оригинально."</p> <p>Response headers</p> <p>content-length: 465 content-type: application/json date: Mon, 21 Apr 2025 10:23:32 GMT server: uvicorn</p>

Responses

## Самостоятельное задание, работа с wikipedia api

```
wiki_main.py > get_page_info
1  from fastapi import FastAPI, Query, Path, Body
2  from pydantic import BaseModel
3  from typing import List
4  import wikipedia
5
6  app = FastAPI()
7
8  class SummaryResponse(BaseModel):
9      title: str
10     summary: str
11
12     class SearchResponse(BaseModel):
13         results: List[str]
14
15     class PageInfoRequest(BaseModel):
16         title: str
17
18     class PageInfoResponse(BaseModel):
19         title: str
20         summary: str
21         url: str
22
23     @app.get("/summary/{title}", response_model=SummaryResponse)
24     def get_summary(title: str = Path(..., description="Заголовок статьи")):
25         summary = wikipedia.summary(title)
26         return {"title": title, "summary": summary}
27
28     @app.get("/search", response_model=SearchResponse)
29     def search_articles(query: str = Query(..., description="Текст для поиска")):
30         results = wikipedia.search(query)
31         return {"results": results}
32
33     @app.post("/pageinfo", response_model=PageInfoResponse)
34     def get_page_info(info: PageInfoRequest = Body(...)):
35         page = wikipedia.page(info.title)
36         return {
37             "title": page.title,
38             "summary": page.summary,
39             "url": page.url
40         }
```

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