

Министерство цифрового развития и массовых коммуникаций
Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение
высшего образования
«Московский технический университет связи и информатики»
(МТУСИ)

Кафедра Информационных технологий

Отчет по лабораторной работе № 15
по дисциплине «Введение в информационные технологии»
на тему: «FastAPI»

Выполнил:
студент группы
БВТ2402
Левая Валерия
Валерьевна

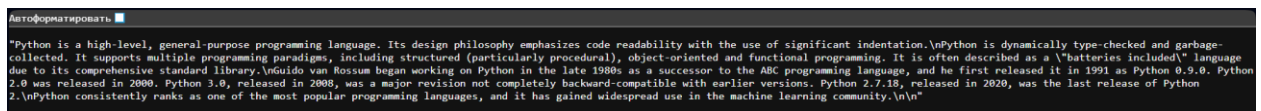
Москва 2025

Создадим простой роут:

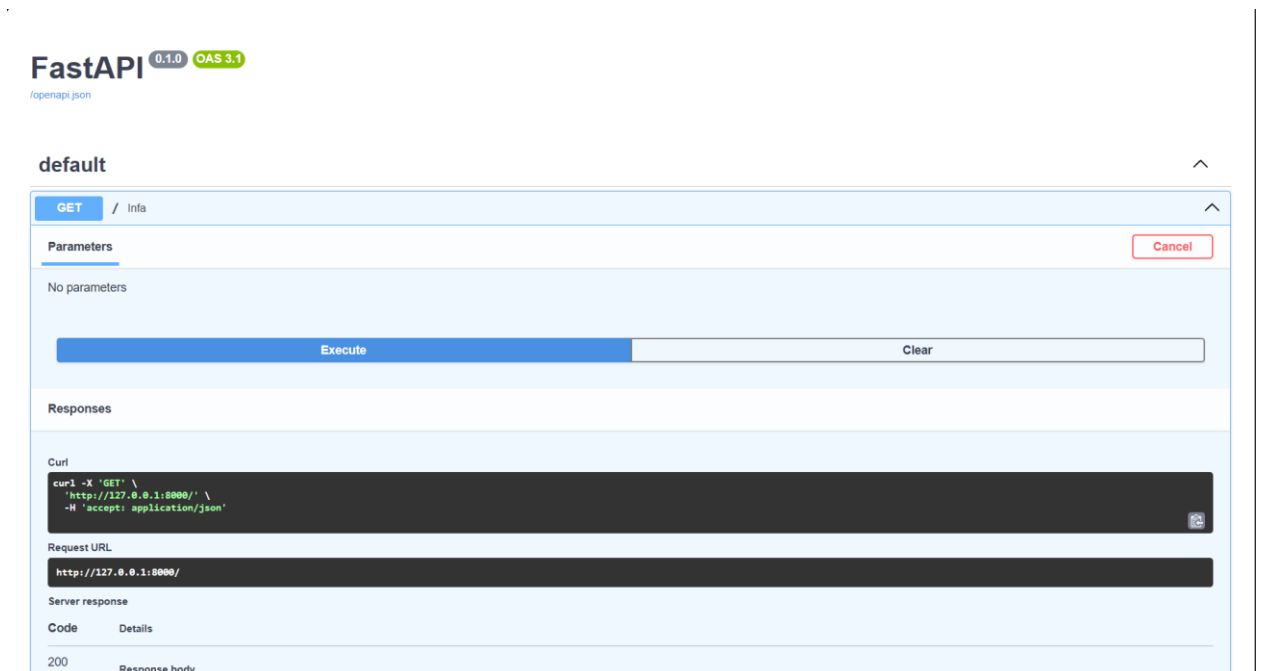
```
main.py > ...
1  from fastapi import FastAPI
2  import wikipedia
3
4  app = FastAPI()
5
6  @app.get("/")
7  def info():
8      return wikipedia.summary("Python Programming Language")
9
```

Получим резюме страницы «Python Programming Language» википедии.

Перейдем по базовому адресу, который указывается при запуске uvicorn - <http://127.0.0.1:8000/>:

A screenshot of a web browser window displaying the Wikipedia summary of Python. The text is in Russian and describes Python as a high-level, general-purpose programming language. It mentions its design philosophy, its dynamic typing, and its popularity in the machine learning community. The text is displayed in a dark-themed browser window.

Страница swagger:

A screenshot of the Swagger UI for a FastAPI application. The interface shows the 'default' API with a single endpoint 'GET / info'. The 'Parameters' section is empty. The 'Responses' section shows a 200 status code with a 'Response body' field. The 'Request URL' is 'http://127.0.0.1:8000/'. The 'Server response' section shows the response body. The 'Execute' button is highlighted.

Добавим еще один роут, где будет параметр в пути, чтобы мы могли указать, что это ответ википедии:

```
9
10 @app.get("/{wiki}")
11 def wiki_info(wiki: str):
12     return wiki + " gives the answer:" + wikipedia.summary("Python Programming Language")
13
```

Добавим к базовому пути <http://127.0.0.1:8000/Wikipedia> через слеш желаемое значение параметра name, чтобы результат был такой:

The screenshot shows a Swagger UI interface for a REST API. At the top, there's a header with 'GET' and the path '/ Info'. Below that, the endpoint is defined as 'GET /{wiki} Wiki Info'. A 'Parameters' section shows a required string parameter named 'wiki' with a text input field containing 'wiki'. Below the parameters are 'Execute' and 'Clear' buttons. The 'Responses' section shows a '200' status code with a 'Response body' containing a JSON object:

```
{"wiki gives the answer:Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.","Python is dynamically type-checked and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.","Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language, and he first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.","Python consistently ranks as one of the most popular programming languages, and it has gained widespread use in the machine learning community.",""}
```

Добавим еще один роут, где будет возможность выбрать количество строк предложения. Для этого добавим еще один параметр, который не будем указывать в пути роута. Это будет query параметр sent_number. Он не будет указан в пути, но также необходим для корректной работы роута:

```
13
14 @app.get("/multi/{wiki}")
15 def multi_wiki(wiki: str, sent_number: int):
16     result = ""
17     for i in range(sent_number):
18         result += wiki + f" gives the answer #{i + 1}: " + wikipedia.summary("Python Programming Language") + " "
19     return result
20
```

Откроем новый роут в сваггере:

GET /{wiki} Wiki Info

GET /multi/{wiki} Multi Wiki

Parameters

Name	Description
wiki * required string (path)	wiki
sent_number * required integer (query)	2

Execute Clear

Responses

Curl

```
curl -X 'GET' \
'http://127.0.0.1:8000/multi/wiki?sent_number=2' \
-H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/multi/wiki?sent_number=2
```

Server response

Code	Details
200	

Создадим новый роут, используя метод POST, и создадим схему тела запроса, которую будет принимать роут для корректной работы. Импортируем из библиотеки pydantic класс BaseModel:

```
1 from fastapi import FastAPI
2 import wikipedia
3 from pydantic import BaseModel
```

```
22 class Infa(BaseModel):
23     wiki: str
24     infa: str
25
26 class InfaInput(BaseModel):
27     wiki: str
28
29 @app.post("/")
30 def create_infa(infa_input: InfaInput):
31     return infa_input.wiki + " gives the answer:" + wikipedia.summary("Python Programming Language") + " "
32
```

Откроем сваггер и протестируем:

POST / Create Infa

Parameters
Cancel Reset

No parameters

Request body required application/json

{
 "wiki": "Wikipedia"
}

Execute Clear

Responses

Curl

Curl

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

Request URL

http://127.0.0.1:8000/

Server response

Code Details

200

Response body

"Wikipedia gives the answer:Python is a high-level, general-purpose programming language. Its design philosophy emphasises code readability with the use of significant indentation.\nPython is a dynamically type-checked and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.\nGuido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language, and he first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2.\nPython consistently ranks as one of the most popular programming languages, and it has gained widespread adoption in the machine learning community.\n"

Response headers

```
content-length: 995
content-type: application/json
date: Mon, 02 Jun 2025 17:47:25 GMT
server: uvicorn
```

Responses

Code Description Links

200 Successful Response No links

Media type
application/json

Controls Accept header.

Example Value Schema

Схемы позволяют и задавать образец ожидаемого ответа. Обновим ранее созданный POST метод, согласно примеру ниже, чтобы ответ выдавался в формате ранее описанным в схеме Infa:

```

22 class Infa(BaseModel):
23     wiki: str
24     infa: str
25
26 class InfaInput(BaseModel):
27     wiki: str
28
29 @app.post("/")
30 def create_infa(infa_input: InfaInput):
31     return Infa(wiki=infa_input.wiki, infa=wikipedia.summary("Python Programming Language"))
32

```

Посмотрим на изменения в сваггере и протестируем обновленный роут. Перед запуском обратим внимание, что схема не показывается в ожидаемом ответе:

The screenshot shows the Swagger UI interface. At the top, there's a 'Responses' section for a 200 status code, labeled 'Successful Response'. Below it, a dropdown menu for 'Media type' is set to 'application/json'. Underneath, there's a link for 'Controls Accept header.' and a section for 'Example Value' and 'Schema'. The 'Schema' section shows a single entry: `"string"`. Below this, there are 'Execute' and 'Clear' buttons. Further down, the 'Curl' section shows a cURL command for a POST request to `http://127.0.0.1:8000/` with headers for `accept: application/json` and `Content-Type: application/json`, and a body of `{ "wiki": "Wikipedia" }`. The 'Request URL' section shows `http://127.0.0.1:8000/`. The 'Server response' section shows a 200 status code. The 'Response body' section displays a JSON object: `{ "wiki": "Wikipedia", "infa": "Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically type-checked and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a 'batteries included' language due to its comprehensive standard library. Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language, and he first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000. Python 3.0, released in 2008, was a major revision not completely backward-compatible with earlier versions. Python 2.7.18, released in 2020, was the last release of Python 2. Python consistently ranks as one of the most popular programming languages, and it has gained widespread use in the machine learning community." }`. The 'Response headers' section shows `content-length: 995`, `content-type: application/json`, and `date: Mon, 02 Jun 2025 17:53:12 GMT`.

Добавим комментарий с описанием роута и поставим в параметрах `response_model`, чтобы добавить валидацию ответа функции. Обновленный эндпоинт будет выглядеть следующим образом:

```
2 class Infa(BaseModel):
3     wiki: str
4     infa: str
5
6 class InfaInput(BaseModel):
7     wiki: str
8
9 @app.post("/", response_model=Infa)
10 def create_infa(infa_input: InfaInput):
11     """Создание информации"""
12     return Infa(wiki=infa_input.wiki, infa=wikipedia.summary("Python Programming Language"))
13
```

Обновим страницу со сваггером и увидим добавленное описание эндпоинта и схему ожидаемого ответа:

POST / Create Info

^

Создание информации

Parameters

Try it out

No parameters

Request body required

application/json

Example Value | Schema

```
{
  "wiki": "string"
}
```

Responses

Code	Description	Links
200	<div>Successful Response</div> <div>Media type</div> <div>application/json</div> <div>Controls Accept header.</div> <div>Example Value Schema</div> <div><pre>{ "wiki": "string", "infa": "string" }</pre></div>	

 No links |