



AULA 02

# FASTAPI

ENDPOINTS E EXPLORANDO PATH E QUERY  
PARAMETERS NO FASTAPI

# O QUE VEREMOS HOJE

- 01 ENDPOINTS
- 02 PATH PARAMETERS
- 03 QUERY PARAMETERS
- 04 DOCUMENTAÇÃO NO FASTAPI

# ENDPOINTS

Endpoints são como portas de entrada para uma API, que permitem que outros sistemas ou aplicativos interajam com ela.

Para acessar um endpoint, você envia uma **solicitação HTTP** para um endereço específico, e a API responde com dados ou comandos específicos.

Cada endpoint é definido por um URL único, e geralmente recebe um método HTTP, como GET, POST, PUT, DELETE, etc.



# ENDPOINTS ESTÁTICOS

Endpoints estáticos são pontos de acesso fixos em sua API, como páginas em um website.

A URL do endpoint é fixa e sempre aponta para a mesma função, permitindo que você acesse dados ou execute funções específicas com um URL específico.




```
1  import fastapi
2
3  app = fastapi.FastAPI()
4
5  @app.get("/about")
6  def about():
7      return {"data": "info about the API"}
8
```

# ENDPOINTS DINÂMICOS

Endpoints dinâmicos, em contraste com os estáticos, permitem flexibilidade na interação com sua API. Eles se adaptam a diferentes entradas, como dados do usuário, valores variáveis ou parâmetros específicos, tornando sua API mais versátil e interativa.

Imagine um sistema de pesquisa de produtos: você pode criar um endpoint dinâmico que recebe o nome do produto como parâmetro e retorna a lista de produtos correspondentes. Assim, o endpoint se adapta a diferentes pesquisas, sem precisar de uma URL específica para cada produto.



```
1  import fastapi
2
3  app = fastapi.FastAPI()
4
5  @app.get("/items/{item_id}")
6  def read_item(item_id: int):
7      return {"item_id": item_id}
8
```

# PATH PARAMETERS

Path Parameters são variáveis que você insere diretamente na URL do endpoint, geralmente para identificar um recurso específico.

A cada requisição é capturado a informação específica na URL.

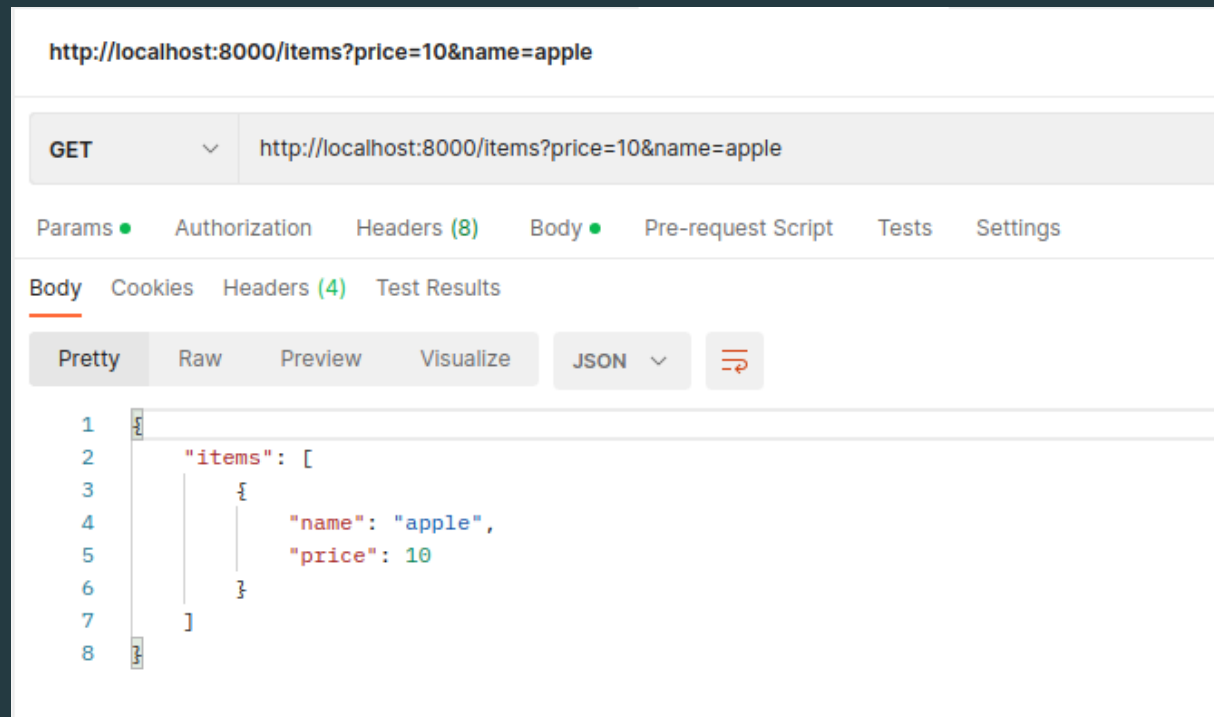
Um path parameter é definido na URL do endpoint utilizando {}:

```
1  from fastapi import FastAPI
2
3  app = FastAPI()
4
5  users = [
6      {"user_id": 1, "name": "João"},
7      {"user_id": 2, "name": "Maria"},
8      {"user_id": 3, "name": "José"},
9  ]
10
11 @app.get("/user/{user_id}")
12 def get_user(user_id: int):
13     for user in users:
14         if user["user_id"] == user_id:
15             return {
16                 "user": user
17             }
18     return {"message": "User not found"}
19
```

# QUERY PARAMETERS

Query Parameters são usados para filtrar, ordenar ou refinar os dados que você está recuperando de um endpoint.

Os dados da consulta devem ser um conjunto de pares-chave-valor que vão atrás do ? em uma URL, separado por &



# QUERY PARAMETERS

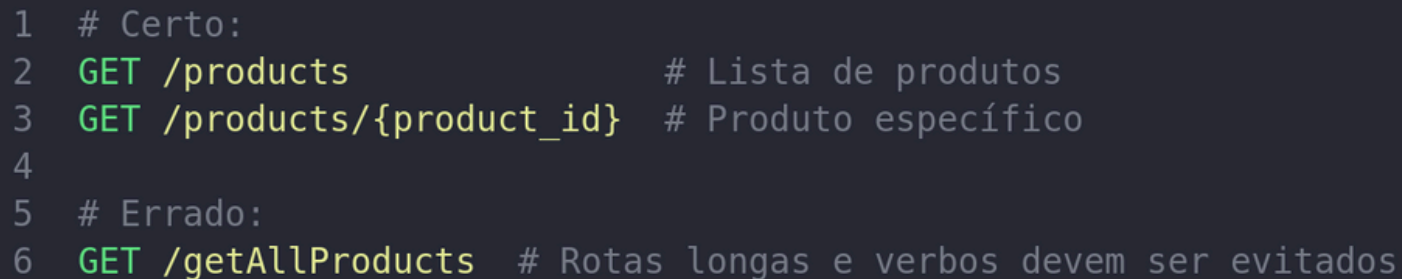
Para permitir parâmetros de consulta no FastAPI, basta adicionar o parâmetro na função e definir None como valor padrão para os casos em que os parâmetros de consulta não sejam enviados.

```
1 from fastapi import FastAPI
2
3 app = FastAPI()
4
5 users = [
6     { "user_id": 1, "name": "Alice", "age": 25, "status": "active"},
7     { "user_id": 2, "name": "Roberto", "age": 30, "status": "inactive" },
8     { "user_id": 3, "name": "Zefinha", "age": 35, "status": "active" }
9 ]
10
11 @app.get('/users')
12 def get_users(min_age: int = None, status: str = None, name: str = None):
13     filtered_users = users
14
15     if min_age:
16         filtered_users = [user for user in filtered_users if user['age'] >= min_age]
17
18     if status:
19         filtered_users = [user for user in filtered_users if user['status'] == status]
20
21     if name:
22         filtered_users = [user for user in filtered_users if user['name'] == name]
23
24     return {
25         'users': filtered_users
26     }
```



# BOAS PRÁTICAS COM ENDPOINTS

- **Uso correto de métodos HTTP:** usar os verbos de acordo com a função de cada um.
- **Path Parameters vs Query Parameters:** usar o path para recursos específicos e o query para filtros e consultas menos específicas
- **Estrutura de Rotas Simples e Consistentes:** manter as rotas curtas e simples



```
1  # Certo:
2  GET /products           # Lista de produtos
3  GET /products/{product_id} # Produto específico
4
5  # Errado:
6  GET /getAllProducts    # Rotas longas e verbos devem ser evitados
```

# ATIVIDADE PRÁTICA

Você está gerenciando um sistema de controle de funcionários de uma empresa. A empresa possui uma lista de funcionários que precisa ser exibida e consultada. Sua tarefa é criar endpoints para:

- Buscar a lista de todos os funcionários.
- Buscar um funcionário com base no ID.

Considere para o exercício uma lista de produtos como esse exemplo:

```
1 employees = [  
2   { "employee_id": 1, "name": "Thawana", "position": "Manager", "salary": 5000.00 },  
3   { "employee_id": 2, "name": "Victor", "position": "Developer", "salary": 4500.00 },  
4   { "employee_id": 3, "name": "Dhennys", "position": "Designer", "salary": 4000.00 }  
5 ]
```

# ATIVIDADE PRÁTICA

Neste cenário, os funcionários podem fazer solicitações de férias, e essas solicitações precisam ser filtradas e visualizadas pela equipe de recursos humanos. Para isso, vamos criar dois endpoints:

- Buscar uma solicitação de férias específica pelo ID.
- Listar todas as solicitações de férias, permitindo que as solicitações sejam filtradas por status e por duração mínima e máxima (em dias).

Considere para o exercício uma lista de pedidos como esse exemplo:

```
1  leave_requests = [  
2      {  
3          "request_id": 1,  
4          "employee_id": 1,  
5          "start_date": "2024-01-15",  
6          "end_date": "2024-01-20",  
7          "status": "approved", # pending, approved, rejected  
8          "duration_days": 5  
9      },  
10     {  
11         "request_id": 2,  
12         "employee_id": 2,  
13         "start_date": "2024-02-10",  
14         "end_date": "2024-02-15",  
15         "status": "pending",  
16         "duration_days": 5  
17     },  
18     {  
19         "request_id": 3,  
20         "employee_id": 3,  
21         "start_date": "2024-03-01",  
22         "end_date": "2024-03-10",  
23         "status": "rejected",  
24         "duration_days": 9  
25     }  
26 ]
```

# DOCUMENTAÇÃO NO FASTAPI

O FastAPI oferece uma documentação automática, ou seja, ao criar endpoints eles já são automaticamente colocados na documentação.

O FastAPI possui duas interfaces de documentação automática, o Swagger e o ReDoc.



# SWAGGER

O Swagger é uma ferramenta de documentação interativa que o FastAPI gera automaticamente.

Ele permite que os usuários explorem e testem os endpoints da API diretamente no navegador, facilitando o entendimento e uso da API. O FastAPI inclui o Swagger UI por padrão, que pode ser acessado no caminho **/docs**.

Principais pontos sobre Swagger no FastAPI:

- **Automático:** Basta criar os endpoints que ele documenta tudo automaticamente.
- **Interativo:** É possível testar os endpoints diretamente pela interface do Swagger, sem precisar de outra ferramenta.
- **Configurações adicionais:** Você pode customizar o título e a descrição da documentação usando o parâmetro `title` e `description` ao inicializar o FastAPI.

# REDOC

O ReDoc é uma outra interface de documentação, também integrada automaticamente no FastAPI.

Ao contrário do Swagger, o ReDoc foca mais na apresentação visual da documentação. Embora ele não seja interativo como o Swagger (não permite testar os endpoints diretamente), ele apresenta a documentação de forma bem organizada e agradável para leitura.

Você pode acessar a documentação gerada pelo ReDoc no FastAPI indo para o caminho **/redoc**.

Principais pontos sobre ReDoc no FastAPI:

- **Visualmente atraente:** Ótimo para leitura e navegação, com uma interface limpa e organizada.
- **Documentação de referência:** Ideal quando o foco é ler e entender a API, sem testar os endpoints diretamente.

# PERSONALIZAÇÃO DA DOCUMENTAÇÃO

É possível adicionar **title**, **description** e **version** a documentação usando o FastAPI.

```
1 app = FastAPI(  
2     title='Product API',  
3     description='This is a simple API to manage products',  
4     version='0.0.1',  
5 )
```

Também é possível personalizar os endpoints adicionando o **summary** e a **description** do endpoint

```
1  
2 @app.get('/products', summary='Get all products', description='Get all products available')  
3 def get_products():  
4     return {  
5         'products': products  
6     }  
7
```