Primeiros Passos

O arquivo FastAPI mais simples pode se parecer com:

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

Copie o conteúdo para um arquivo `main.py`.

Execute o servidor:

```
$ uvicorn main:app --reload

<span style="color: green;">INFO</span>:     Uvicorn running on
http://127.0.0.1:8000 (Press CTRL+C to quit)
<span style="color: green;">INFO</span>:     Started reloader process [28720]
<span style="color: green;">INFO</span>:     Started server process [28722]
<span style="color: green;">INFO</span>:     Waiting for application startup.
<span style="color: green;">INFO</span>:     Application startup complete.
```

!!! nota O comando `uvicorn main:app` se refere a:

```
* `main`: o arquivo `main.py` (o "módulo" Python).
* `app`: o objeto criado no arquivo `main.py` com a linha `app = FastAPI()`.
* `--reload`: faz o servidor reiniciar após mudanças de código. Use apenas para
desenvolvimento.
```

Na saída, temos:

```
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Essa linha mostra a URL onde a sua aplicação está sendo servida, que nesse caso é a sua máquina local.

Confira

Abra o seu navegador em http://127.0.0.1:8000.

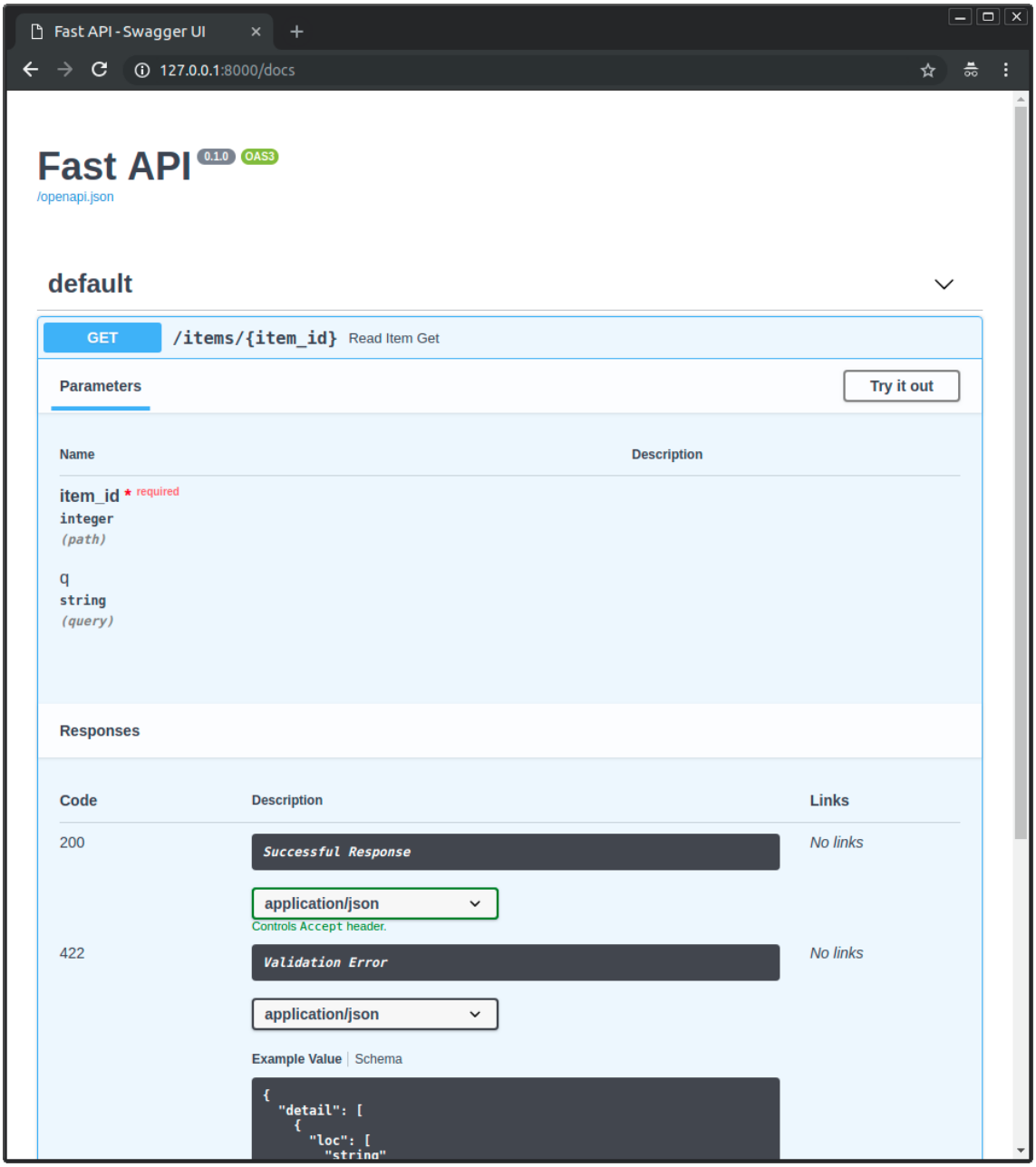
Você verá essa resposta em JSON:

```
{"message": "Hello World"}
```

Documentação Interativa de APIs

Agora vá para http://127.0.0.1:8000/docs.

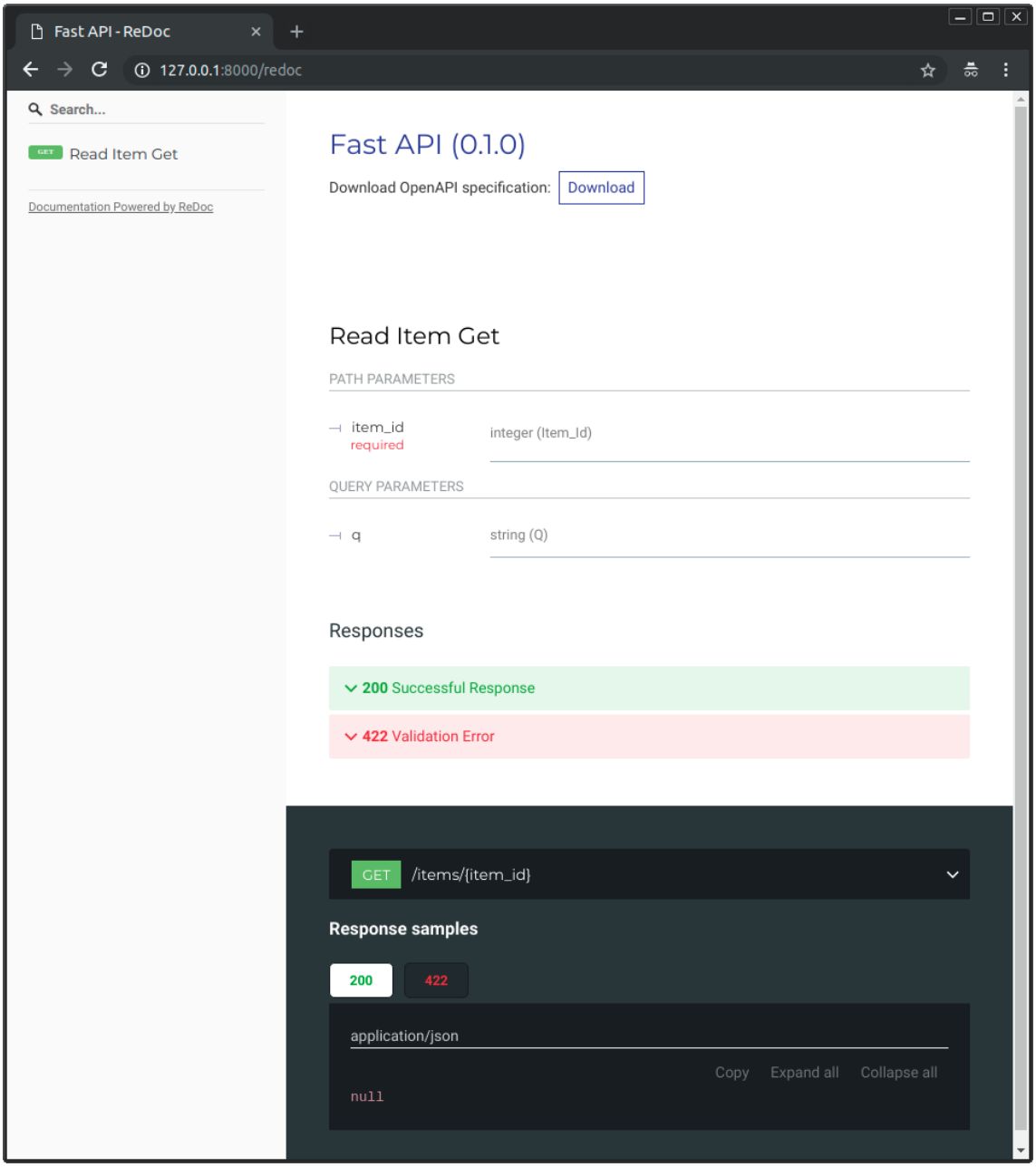
Você verá a documentação interativa automática da API (fornecida por Swagger UI):



Documentação Alternativa de APIs

E agora, vá para http://127.0.0.1:8000/redoc.

Você verá a documentação alternativa automática (fornecida por ReDoc):



OpenAPI

O **FastAPI** gera um "*schema*" com toda a sua API usando o padrão **OpenAPI** para definir APIs.

"*Schema*"

Um "*schema*" é uma definição ou descrição de algo. Não o código que o implementa, mas apenas uma descrição abstrata.

API "*schema*"

Nesse caso, OpenAPI é uma especificação que determina como definir um *schema* da sua API.

Esta definição de *schema* inclui as rotas da sua API, os parâmetros possíveis que elas usam, etc.

"*Schema*" de dados

O termo "*schema*" também pode se referir à forma de alguns dados, como um conteúdo JSON.

Nesse caso, significaria os atributos JSON e os tipos de dados que eles possuem, etc.

OpenAPI e JSON *Schema*

OpenAPI define um *schema* de API para sua API. E esse *schema* inclui definições (ou "*schemas*") dos dados enviados e recebidos por sua API usando **JSON *Schema***, o padrão para *schemas* de dados JSON.

Verifique o `openapi.json`

Se você está curioso(a) sobre a aparência do *schema* bruto OpenAPI, o FastAPI gera automaticamente um JSON (*schema*) com as descrições de toda a sua API.

Você pode ver isso diretamente em: http://127.0.0.1:8000/openapi.json.

Ele mostrará um JSON começando com algo como:

```
{
    "openapi": "3.0.2",
    "info": {
        "title": "FastAPI",
        "version": "0.1.0"
    },
    "paths": {
        "/items/": {
            "get": {
                "responses": {
                    "200": {
                        "description": "Successful Response",
                        "content": {
                            "application/json": {


...
```

Para que serve o OpenAPI

O *schema* OpenAPI é o que possibilita os dois sistemas de documentação interativos mostrados.

E existem dezenas de alternativas, todas baseadas em OpenAPI. Você pode facilmente adicionar qualquer uma dessas alternativas à sua aplicação criada com **FastAPI**.

Você também pode usá-lo para gerar código automaticamente para clientes que se comunicam com sua API. Por exemplo, aplicativos front-end, móveis ou IoT.

Recapitulando, passo a passo

Passo 1: importe `FastAPI`

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

`FastAPI` é uma classe Python que fornece todas as funcionalidades para sua API.

!!! nota "Detalhes técnicos" `FastAPI` é uma classe que herda diretamente de `Starlette`.

```
Você pode usar todas as funcionalidades do <a href="https://www.starlette.io/"
class="external-link" target="_blank">Starlette</a> com `FastAPI` também.
```

Passo 2: crie uma "instância" de `FastAPI`

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

Aqui, a variável `app` será uma "instância" da classe `FastAPI`.

Este será o principal ponto de interação para criar toda a sua API.

Este `app` é o mesmo referenciado por `uvicorn` no comando:

```
$ uvicorn main:app --reload

<span style="color: green;">INFO</span>:     Uvicorn running on
http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Se você criar a sua aplicação como:

```
{!../../../docs_src/first_steps/tutorial002.py!}
```

E colocar em um arquivo `main.py`, você iria chamar o `uvicorn` assim:

```
$ uvicorn main:my_awesome_api --reload

<span style="color: green;">INFO</span>:     Uvicorn running on
http://127.0.0.1:8000 (Press CTRL+C to quit)
```

Passo 3: crie uma *rota*

Rota

"Rota" aqui se refere à última parte da URL, começando do primeiro `/`.

Então, em uma URL como:

```
https://example.com/items/foo
```

...a rota seria:

```
/items/foo
```

!!! info "Informação" Uma "rota" também é comumente chamada de "endpoint".

Ao construir uma API, a "rota" é a principal forma de separar "preocupações" e "recursos".

Operação

"Operação" aqui se refere a um dos "métodos" HTTP.

Um dos:

- `POST`
- `GET`
- `PUT`
- `DELETE`

...e os mais exóticos:

- `OPTIONS`
- `HEAD`
- `PATCH`
- `TRACE`

No protocolo HTTP, você pode se comunicar com cada rota usando um (ou mais) desses "métodos".

Ao construir APIs, você normalmente usa esses métodos HTTP para executar uma ação específica.

Normalmente você usa:

- `POST` : para criar dados.
- `GET` : para ler dados.
- `PUT` : para atualizar dados.
- `DELETE` : para deletar dados.

Portanto, no OpenAPI, cada um dos métodos HTTP é chamado de "operação".

Vamos chamá-los de "**operações**" também.

Defina um *decorador de rota*

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

O `@app.get("/")` diz ao **FastAPI** que a função logo abaixo é responsável por tratar as requisições que vão para:

- a rota `/`
- usando o operador `get`

!!! info " `@decorador` " Essa sintaxe `@alguma_coisa` em Python é chamada de "decorador".

```
Você o coloca em cima de uma função. Como um chapéu decorativo (acho que é daí que vem o termo).

Um "decorador" pega a função abaixo e faz algo com ela.

Em nosso caso, este decorador informa ao **FastAPI** que a função abaixo corresponde a **rota** `/` com uma **operação** `get`.

É o "**decorador de rota**".
```

Você também pode usar as outras operações:

- `@app.post()`
- `@app.put()`
- `@app.delete()`

E os mais exóticos:

- `@app.options()`
- `@app.head()`
- `@app.patch()`
- `@app.trace()`

!!! tip "Dica" Você está livre para usar cada operação (método HTTP) como desejar.

```
O **FastAPI** não impõe nenhum significado específico.

As informações aqui são apresentadas como uma orientação, não uma exigência.

Por exemplo, ao usar GraphQL, você normalmente executa todas as ações usando apenas operações `POST`.
```

Passo 4: defina uma função de rota

Esta é a nossa "**função de rota**":

- **rota**: é `/`.
- **operação**: é `get`.
- **função**: é a função abaixo do "decorador" (abaixo do `@app.get("/")`).

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

Esta é uma função Python.

Ela será chamada pelo **FastAPI** sempre que receber uma requisição para a URL "`/`" usando uma operação `GET`.

Neste caso, é uma função `assíncrona`.

Você também pode defini-la como uma função normal em vez de `async def`:

```
{!../../../docs_src/first_steps/tutorial003.py!}
```

!!! nota Se você não sabe a diferença, verifique o Async: *"Com pressa?"*{.internal-link target=_blank}.

Passo 5: retorne o conteúdo

```
{!../../../docs_src/first_steps/tutorial001.py!}
```

Você pode retornar um `dict`, `list` e valores singulares como `str`, `int`, etc.

Você também pode devolver modelos Pydantic (você verá mais sobre isso mais tarde).

Existem muitos outros objetos e modelos que serão convertidos automaticamente para JSON (incluindo ORMs, etc). Tente usar seus favoritos, é altamente provável que já sejam compatíveis.

Recapitulando

- Importe `FastAPI`.
- Crie uma instância do `app`.
- Coloque o **decorador que define a operação** (como `@app.get("/")`).
- Escreva uma **função para a operação da rota** (como `def root(): ...`) abaixo.
- Execute o servidor de desenvolvimento (como `uvicorn main:app --reload`).