**Laís:** Já iniciamos a criação da nossa rota. Vamos abrir o terminal, onde o servidor local está sendo executado. Notamos que, ao acessarmos a rota inicial ("/"), foi retornado um erro "*404 Not Found*", pois não havia nada programado.

No entanto, ao acessarmos a rota GET /api/hello, que tínhamos criado, o retorno foi "200 OK". Ao trabalhar com isso, é sempre importante analisar o terminal para verificar se a rota está sendo encontrada ou se por algum motivo digitamos algo errado.

Agora precisamos implementar um ponto importante: separar o cardápio por restaurante e inserir essas informações no nosso servidor.

**Criando um *endpoint***

**Guilherme:** Vamos começar **criando uma nova rota**. No arquivo main.py, podemos usar @app.get() e precisaremos definir o caminho desse endpoint. Por exemplo, podemos usar "/api/restaurantes" entre aspas simples.

*main.py:*

**from** fastapi **import** FastAPI

app = FastAPI()

@app.get('/api/hello')

**def** **hello\_world**():

**return** {'Hello':'World'}

@app.get('/api/restaurantes/')

Copiar código

A seguir, vamos criar uma função para isso também. Ela pode se chamar get\_restaurantes().

# código omitido

@app.get('/api/restaurantes/')

**def** **get\_restaurantes**()

Copiar código

Um ponto importante que devemos lembrar é que não queremos exibir todos os restaurantes como fizemos inicialmente, quando usamos *requests* para trazer todas as informações. Queremos exibir informações específicas de um restaurante. Portanto, receberemos um argumento nesse endpoint.

Esse argumento será o restaurante dentro do get\_restaurantes(). O tipo desse argumento será uma *string* (str). Após o tipo do argumento, usaremos uma propriedade do FastAPI para inserir a *query* com o nome do restaurante que queremos.

Vamos então importar, no início do código, onde temos o import de FastAPI, o módulo chamado Query. É ele que vamos usar no FastAPI. Vamos abrir e fechar parênteses na chamada de Query() após str, e dizer que, quando não tiver nada, o valor padrão será None.

@app.get('/api/restaurantes/')

**def** **get\_restaurantes**(restaurante: str = Query(**None**)):

Copiar código

**Laís:** Como ficará isso? É como se colocássemos o nome de um restaurante após "/api/restaurantes/", por exemplo, "McDonald's"?

**Guilherme:** Vamos utilizar a barra, ponto de interrogação, e o que quisermos chamar, especificamente. Podemos decidir, por exemplo, chamar pelo nome do restaurante.

**Laís:** Então podemos inserir uma barra após "restaurantes" para indicar que terá outra informação naquele endpoint.

**Guilherme:** Se olharmos para o código do arquivo app.py, fizemos algo muito parecido. Temos uma url e depois fizemos a requisição; tudo isso será necessário. Então, vamos copiar até a linha que temos o else, que exibe apenas o status\_code, e jogar para a aplicação do FastAPI.

@app.get('/api/restaurantes/')

**def** **get\_restaurantes**(restaurante: str = Query(**None**)):

url = 'https://guilhermeonrails.github.io/api-restaurantes/restaurantes.json'

response = requests.get(url)

print(response)

**if** response.status\_code == 200:

dados\_json = response.json()

dados\_restaurante = {}

**for** item **in** dados\_json:

nome\_do\_restaurante = item['Company']

**if** nome\_do\_restaurante **not** **in** dados\_restaurante:

dados\_restaurante[nome\_do\_restaurante] = []

dados\_restaurante[nome\_do\_restaurante].append({

"item": item['Item'],

"price": item['price'],

"description": item['description']

})

**else**:

print(f'O erro foi {response.status\_code}')

Copiar código

Vamos precisar do requests também, então podemos fazer import requests no início do código.

**from** fastapi **import** FastAPI

**import** requests

Copiar código

**Laís:** Lembrando que, como o projeto ainda está dentro da *venv*, o requests já está instalado e não precisamos instalar para cada projeto que formos fazer.

**Guilherme:** No nosso código, temos a url, o response com o requests, podemos remover o print() da linha abaixo, e depois verificamos se o status\_code é igual a 200. Na sequência, pegamos esses dados e transformamos a resposta em JSON.

*Trecho a ser removido:*

print(response)

Copiar código

Depois, vamos fazer uma verificação diferente. Primeiro, precisamos saber se o restaurante é None. Se for None, precisamos exibir alguma coisa. Então, depois dessa verificação, podemos fazer mais uma verificação: se o restaurante não for nada, queremos exibir todas as informações.

Nesse caso, podemos colocar na linha 17 que if restaurante is None, ou seja, se nenhuma informação foi passada, precisamos adicionar determinada informação. Podemos exibir os dados com todos os restaurantes, ou uma mensagem dizendo que o restaurante não foi encontrado.

**Laís:** Podemos passar com todos os restaurantes, o cardápio completo.

**Guilherme:** Legal, então podemos colocar um return com Dados entre chaves e aspas simples, e recebendo dados\_json.

**if** response.status\_code == 200:

dados\_json = response.json()

**if** restaurante **is** **None**:

**return** {'Dados':dados\_json}

Copiar código

Na sequência, criamos dados\_restaurante, o que podemos manter. Depois, para cada item em dados\_restaurante, inserimos os valores.

Agora vamos analisar o caso em que precisaremos verificar se o argumento nome\_do\_restaurante foi passado ou não para a nossa aplicação.

Podemos apagar a linha 22, e no lugar dela, colocaremos if item['Company'] == restaurante:.

**if** item['Company'] == restaurante:

Copiar código

Feito isso, também podemos apagar a verificação das linhas 24 e 25.

*Trecho a ser removido:*

**if** nome\_do\_restaurante **not** **in** dados\_restaurante:

dados\_restaurante[nome\_do\_restaurante] = []

Copiar código

Com isso, moveremos a verificação para dentro de dados\_restaurante apenas como uma lista, em vez de ser um dicionário. Além disso, agora que mudamos para lista, não precisamos mais do parâmetro nome\_do\_restaurante na linha 23. Então, dados\_restaurante direto adicionará todas as informações.

dados\_restaurante = []

**for** item **in** dados\_json:

**if** item['Company'] == restaurante:

dados\_restaurante.append({

"item": item['Item'],

"price": item['price'],

"description": item['description']

})

Copiar código

Antes de terminar, há um último detalhe. Precisamos fazer um return no final do loop for com a mensagem, por exemplo, "'Restaurante':restaurante,'Cardapio':dados\_restaurante".

**return** {'Restaurante':restaurante,'Cardapio':dados\_restaurante}

Copiar código

Caso tenhamos um erro no else, podemos passar uma determinada informação, por exemplo, retornar o response.status\_code.

**else**:

**return** {'Erro':f'{response.status\_code} - {response.text}'}

Copiar código

**Conclusão**

**Laís:** Agora só falta testar e conferir se funciona. É isso que faremos a seguir!