**Guilherme:** Chegou a hora de testarmos nossa API para verificar se tudo funciona corretamente!

**Consultando *endpoints***

Não precisamos recarregar o servidor, pois ele já faz o --reload automaticamente.

**Laís:** Com o terminal aberto, vamos digitar o comando uvicorn seguido de main, que é o nome da aplicação, :app --reload.

uvicorn main:app --reload

Copiar código

**Guilherme:** Assim, podemos clicar com a tecla "Ctrl" pressionada no endereço retornado e seremos redirecionados para a API.

Existem dois *endpoints*: o /api/hello, e o /api/restaurantes. Vamos testá-los. O primeiro exibirá uma mensagem. Agora precisamos acessar os restaurantes especificamente. Vamos testar dois caminhos.

Para o primeiro, vamos inserir /api/restaurantes no endereço da barra de navegação e pressionar "Enter". O que temos quando fazemos isso?

**Laís:** Todos os dados de todos os restaurantes. Portanto, obtemos exatamente esses dados e colocando-os nesse endereço basicamente.

**Guilherme:** Criamos nossa própria API com os mesmos dados. Porém, ainda vamos aprimorar isso. Como fazemos para pesquisar, por exemplo, todos os itens de um restaurante nessa API?

**Laís:** Quando voltamos ao arquivo main.py, como definimos o que seria inserido? Definimos o restaurante e haveria uma Query(). Então, para fazer isso, podemos inserir um formato com interrogação (?) e seguida de "restaurante" e o nome do restaurante.

127.0.0.1:8000/api/restaurantes/?restaurante=McDonald's

Após pressionar "Enter", ele mostra exatamente o formato que tínhamos colocado. Ou seja, ele informa que o restaurante do momento é o *McDonald's* e mostra o cardápio apenas dos itens do McDonald's.

**Guilherme:** Vamos verificar sobre outros restaurantes. Pegaremos o nome dos nossos arquivos JSON. Nesse caso, vamos utilizar o *KFC*.

127.0.0.1:8000/api/restaurantes/?restaurante=KFC

**Laís:** Com isso, é exibido o restaurante KFC e seu cardápio. Isso é idealmente como estaríamos observando. Porém, há um detalhe: como sabemos as rotas que colocamos na aplicação?

**Guilherme:** É verdade. Tem um ponto muito importante: se digitarmos uma letra errada, por exemplo, removendo o "C" de "KFC", deixando apenas "KF", ele não vai achar. Ou se escrevermos api/restaurante no singular e o restante estiver tudo igual, também não vamos achar.

**Laís:** Ele retorna o mesmo dado que tínhamos feito com todos os itens do restaurante, porque ele não identifica o restaurante que estamos inserindo.

**Guilherme:** O grande desafio é como conseguimos passar essa mensagem para outras equipes. Será que precisaremos ligar, enviar uma mensagem no *Slack*, no *Discord*, dizendo "se você quiser acessar esse endpoint, precisa ser esse endereço"? Não faz sentido.

O software precisa se garantir na forma como deve funcionar. É como se tivéssemos um manual de como essa API funciona. E existe uma forma de visualizar todos os endpoints criados em uma API usando o FastAPI. Faremos isso agora.

**Laís:** Podemos pegar essa mesma rota, e a rota inicial dela é sem nada. Então, se colocarmos o número padrão, que é 127.0.0.1:8000, exatamente a porta que trabalhamos localmente, ele diz que nada foi encontrado.

Porém, se adicionarmos ao final o /docs, ele vai abrir a documentação do nosso projeto em específico.

127.0.0.1:8000/docs

Nessa página, são exibidas as rotas padrão, os endpoints padrão. O primeiro listado é o /api/hello, que vai retornar "Hello World". Também temos o api/restaurantes/, que vai pegar os restaurantes.

**Guilherme:** Há uma boa prática que podemos fazer. Para isso, vamos voltar no código, no arquivo main.py em /api/hello, vamos criar uma ***docstring*** dele na linha 7.

Digitamos três aspas simples, do mesmo jeito que já vimos anteriormente. Podemos escrever algo como, por exemplo, "Endpoint que exibe uma mensagem incrível do mundo da programação".

*main.py:*

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

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

'''

Endpoint que exibe uma mensagem incrível do mundo da programação!

'''

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

Copiar código

Vamos fazer isso também para o endpoint de restaurantes.

**Laís:** Após a criação da função get\_restaurantes(), vamos abrir e fechar três aspas simples. Entre elas, vamos inserir "Endpoint para ver os cardápios dos restaurantes".

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

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

'''

Endpoint para ver os cardápios dos restaurantes

'''

# código omitido

Copiar código

**Guilherme:** Acho que isso já é suficiente. Agora vamos voltar na nossa aplicação no navegador e atualizar a página para conferir como vai ficar.

Agora, ao clicar em "Hello World", temos a mensagem "Endpoint que exibe uma mensagem incrível do mundo da programação!". Conseguimos testar isso. Vamos abrir o endpoint do "Hello World" novamente e clicar no botão cinza "*Try it out*" no canto direito. Em seguida, clicamos em "*Execute*".

Repare que conseguimos até testar. Ele dá uma forma de testar somente usando o navegador. Descendo um pouco a página, temos a seção "*Response body*", o corpo da resposta.

{

"Hello": "World"

}

Copiar código

Temos a mensagem "Hello": "World", o status\_code 200, e assim por diante.

Vamos testar agora o endpoint dos restaurantes?

**Laís:** À direita, temos o botão "Try it out" que significa "tentar". No primeiro campo de descrição, vamos colocar "KFC", por exemplo. Feito isso, apertaremos o botão de executar.

Dessa forma, ele mostra toda a resposta, o endpoint correto que deveríamos utilizar para executar, e também a resposta que é bem grande, aquela resposta com o cardápio do KFC.

Voltando para o *VS Code*, vamos abrir o terminal com "Ctrl + J", teclar "Ctrl + C", e assim, encerramos o servidor local!