

Documentação do
“Desafio Back-End Multiplier”

Por: Elisio Breno Garcia Cardoso

SUMÁRIO

- 1.** CONFIGURAÇÃO DO BANCO DE DADOS NO LARAVEL.
- 2.** TABELAS DO BANCO DE DADOS.
- 3.** POPULAÇÃO DO BANCO DE DADOS.
- 4.** END-POINTS DA API.
- 5.** EXECUÇÃO.

1. CONFIGURAÇÃO DO BANCO DE DADOS NO LARAVEL.

A API do “*Desafio Back-End Multiplier*” foi desenvolvido na linguagem PHP utilizando o Framework Laravel 8. O Banco de Dados utilizado foi o MySQL 8.0. Esta seção descreve a realização das configurações no ambiente de execução para um correto uso da API.

A execução da API requer a criação de um banco de dados chamado “multiplier_api” no MySQL (Tabela 1). No arquivo “.env”, é necessário definir o driver do banco de dados utilizado (DB_CONNECTION), além dos parâmetros *DB_HOST*, *DB_PORT*, *DB_DATABASE*, *DB_USERNAME* e *DB_PASSWORD* para setar o endereço do servidor e as credenciais do banco de dados (Tabela 2). Adicionalmente, também pode ser definido um novo usuário no servidor do MySQL (Tabela 3) para a utilização da API ou utilizar o usuário *root*.

O banco de dados da aplicação deve ser criado executando o comando **php artisan migrate**, com o terminal aberto na pasta do projeto..

```
create database multiplier_api;
```

Tabela 1: Comando para a Criação do Banco de Dados na API

```
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=multiplier_api
DB_USERNAME=api_user
DB_PASSWORD=admin123
```

Tabela 2: Parâmetros de configuração do BD no “.env”.

```
CREATE USER 'api_user'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'admin123';

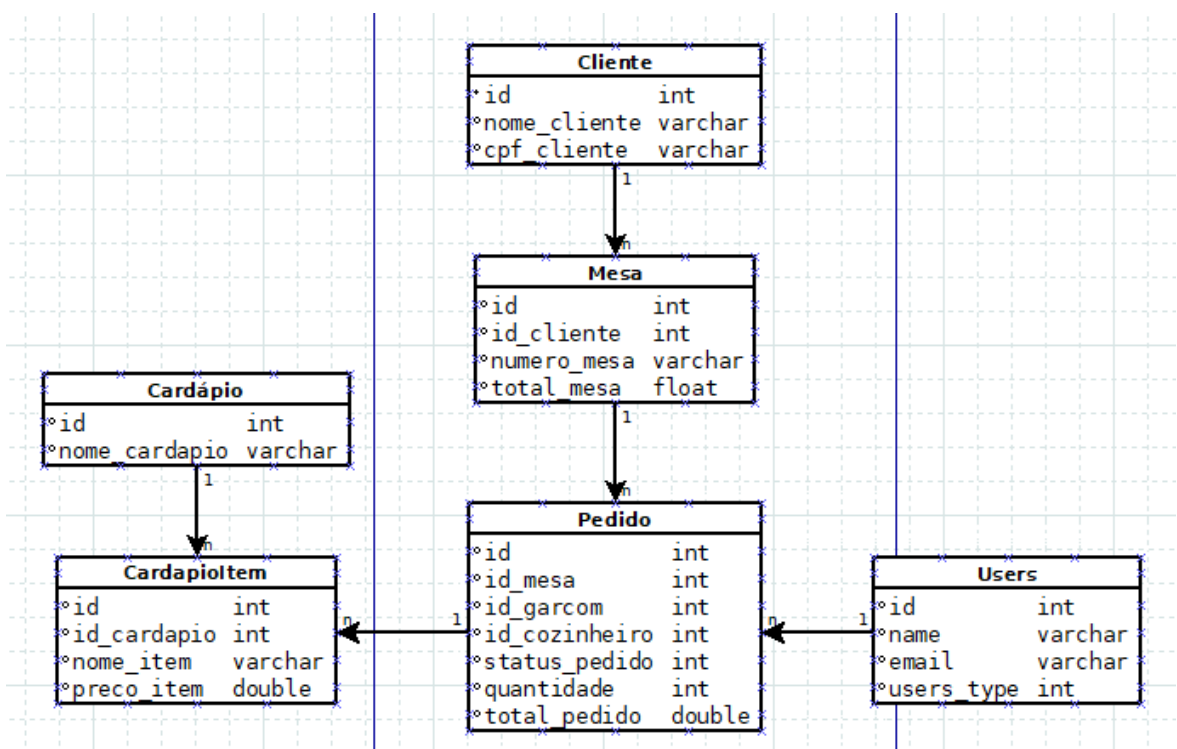
GRANT ALL PRIVILEGES ON *.* TO 'api_user'@'localhost' WITH GRANT
OPTION;
```

Tabela 3: Criação de um novo usuário no MySQL para utilização na API.

2. TABELAS DO BANCO DE DADOS

O banco de dados da imagem abaixo foi desenvolvido para o armazenamento dos dados manipulados pela API. Um total de 6 tabelas foram criadas visando à execução das funcionalidades básicas, cada uma mapeada como um *Model* e um *Migration* nas bibliotecas do Laravel:

- **Users:** Armazena os Usuários da API (Garçons e Cozinheiros).
- **Cardapio:** Armazena os cardápios do Restaurante.
- **Cardapioltem:** Armazena os produtos do restaurante. Cada item é vinculado a um único cardápio.
- **Cliente:** Armazena os clientes do restaurante.
- **Mesa:** Armazena uma mesa criada para um cliente. Uma mesa pode conter vários pedidos.
- **Pedido:** Armazena os pedidos do restaurante. Cada pedido está vinculado a uma mesa, a um item correspondente, a um garçom e a um cozinheiro.



A criação das tabelas do banco de dados é realizada com a execução no terminal do comando **php artisan migrate**.

O comando **php artisan passport:client --personal** criar a *Personal Access Client*, possibilitando a autenticação na API pelo *Passport*.

3. POPULAÇÃO DO BANCO DE DADOS

O banco de dados da API foi populado inicialmente com o auxílio da Biblioteca *Faker* e os recursos de *Seeders* disponíveis no Laravel. As classes abaixo geram os dados de teste quando o comando ***php artisan db:seed*** é executado no terminal.

- ***UserSeeder*** - Cria um conjunto de usuários
- ***CardapioSeeder*** - Cria um conjunto de 50 cardápios, inserindo itens em seguida.
- ***ClienteSeeder*** - Cria um conjunto de 10 mil clientes. Cada cliente possui 5 mesas contendo 8 pedidos. No total, 400 mil pedidos são inseridos no banco de dados.

4. END-POINTS DA API.

END-POINT (USUÁRIOS)	MÉTODO
/register - Cadastrar um Usuário na API.	POST
/login - Autenticar um Usuário na API.	POST
/user - Usuário que está logado na API.	GET

END-POINTS (CARDÁPIO)	MÉTODO
/cardapio <ul style="list-style-type: none">Retorna todos os cardápios cadastrados no BD.	GET
/cardapio/{id_cardapio} <ul style="list-style-type: none">Retorna os dados do cardápio com id = {id_cardápio}.	GET
/cardapio <ul style="list-style-type: none">Cadastra um novo cardápio com os dados passados através de um JSON.	POST
/cardapio/{id_cardapio} <ul style="list-style-type: none">Edita os dados do cardápio com o id = {id_cardápio}.	PUT
/cardapio/{id_cardapio} <ul style="list-style-type: none">Deleta o cardápio com o id = {id_cardapio}.	DELETE

END-POINTS (CARDÁPIO ITEM)	MÉTODO
/cardapio/{id_cardapio}/item <ul style="list-style-type: none">Retorna todos os itens do cardápio que possuem o id = {id_cardapio}.	GET
/cardapio/{id_cardapio}/item/{id_item} <ul style="list-style-type: none">Retorna os dados do item com id = {id_item}.	GET
/cardapio/{id_cardapio}/item <ul style="list-style-type: none">Cadastra um novo item vinculado ao cardápio que possui o id={id_Cardapio}, dados inseridos através de um JSON.	POST
/cardapio/{id_cardapio}/item/{id_item} <ul style="list-style-type: none">Edita o item de cardápio com id = {id_item}.	PUT
/cardapio/{id_cardapio}/item/{id_item} <ul style="list-style-type: none">Deleta o item de cardápio com id = {id_item}.	DELETE

END-POINTS (CLIENTE)	MÉTODO
/cliente <ul style="list-style-type: none"> Retorna todos os clientes disponíveis no BD. 	GET
/cliente/{id_cliente} <ul style="list-style-type: none"> Retorna os dados do cliente com o id = {id_cliente}. 	GET
/cliente/{id_cliente}/getallpedidoscliente <ul style="list-style-type: none"> Retorna todos os pedidos do cliente. 	GET
/cliente/{id_cliente}/getprimeiropedido <ul style="list-style-type: none"> Retorna o primeiro pedido do cliente. 	GET
/cliente/{id_cliente}/getultimopedido. <ul style="list-style-type: none"> Retorna o último pedido do cliente. 	GET
/cliente/{id_cliente}/getmaiorpedido <ul style="list-style-type: none"> Retorna o maior pedido do cliente. 	GET
/cliente/{id_cliente}/getmenorpedido <ul style="list-style-type: none"> Retorna o menor pedido do cliente. 	GET
/cliente <ul style="list-style-type: none"> Cadastra um novo cliente com os dados inseridos através de um JSON. 	POST
/cliente/{id_cliente} <ul style="list-style-type: none"> Edita os dados do cliente com o id = {id_cardápio}. 	PUT
/cliente/{id_cliente} <ul style="list-style-type: none"> Deleta o cliente com o id = {id_cardapio}. 	DELETE

END-POINTS (MESA)	MÉTODO
/cliente/{id_cliente}/mesa <ul style="list-style-type: none"> Retorna todas as mesas que já foram inseridas para o cliente que possui o id={id_cliente}. 	GET
/cliente/{id_cliente}/mesa/{id_mesa} <ul style="list-style-type: none"> Retorna os dados da mesa com o id={id_mesa}, também vinculada a cliente com o id={id_cliente}. 	GET
/cliente/{id_cliente}/mesa/{id_mesa}/getallpedidosmesa <ul style="list-style-type: none"> Retorna os pedidos da mesa com o id={id_mesa}, 	GET
/cliente/{id_cliente}/mesa <ul style="list-style-type: none"> Cadastra uma nova mesa vinculada ao cliente que possui o id={id_cliente}, dados passados através de um JSON. 	POST
/cliente/{id_cliente}/mesa/{id_mesa} <ul style="list-style-type: none"> Edita os dados do cliente com o id = {id_cardápio}. 	PUT

/cliente/{id_cliente}/mesa/{id_mesa} <ul style="list-style-type: none"> Deleta o cliente com o id = {id_cardapio}. 	DELETE
--	---------------

END-POINTS (PEDIDO)	MÉTODO
/cliente/{id_cliente}/mesa/{id_mesa}/pedido <ul style="list-style-type: none"> Retorna todos os pedidos da mesa {id_mesa} vinculada ao cliente {id_mesa}. 	GET
/cliente/{id_cliente}/mesa/{id_mesa}/pedido/{id_pedido}. <ul style="list-style-type: none"> Retorna os dados do pedido id={id_pedido}. 	GET
/pedido/getpedidospormes/{ano}/{mes} <ul style="list-style-type: none"> Retorna os pedidos de um mês. 	GET
/pedido/getpedidospordia/{ano}/{mes}/{dia} <ul style="list-style-type: none"> Retorna os pedidos de um dia. 	GET
/cliente/{id_cliente}/mesa/{id_mesa}/pedido <ul style="list-style-type: none"> Cria um novo pedido para a mesa {id_mesa}, vinculada ao cliente {id_cliente}. 	POST
/cliente/{id_cliente}/mesa/{id_mesa}/pedido/{id_pedido} <ul style="list-style-type: none"> Edita os dados de um pedido. 	PUT
/cliente/{id_cliente}/mesa/{id_mesa}/pedido/{id_pedido} <ul style="list-style-type: none"> Deleta um pedido. 	DELETE

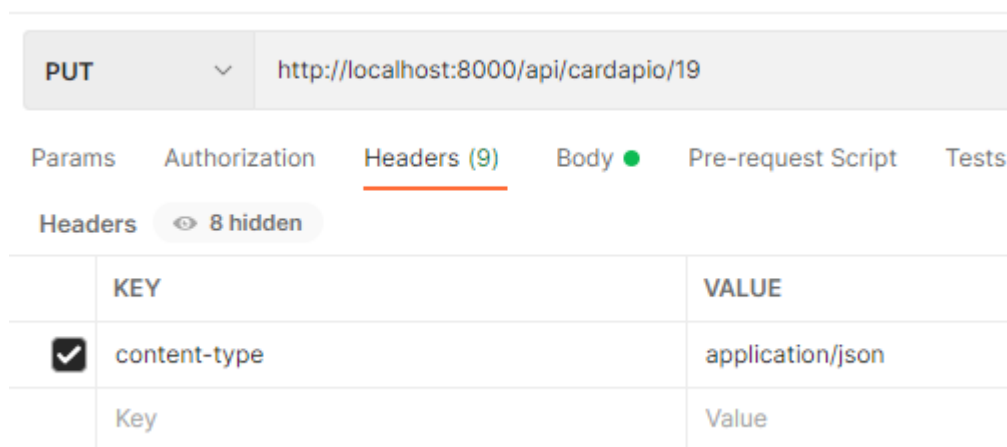
END-POINTS (PEDIDOS FUNCIONÁRIO)	MÉTODO
/user/gettodospedidosdofuncionario <ul style="list-style-type: none"> Retorna todos os pedidos de um funcionário, seja ele um garçom ou um cozinheiro. 	GET
/user/gettodospedidosgarcomemandamento <ul style="list-style-type: none"> Retorna para um garçom todos os pedidos com status “Em Andamento”. 	GET
/user/gettodospedidosgarcomafazer <ul style="list-style-type: none"> Retorna para um garçom todos os pedidos com o status “A Fazer”. 	GET
/user/gettodospedidoscozinheiroemandamento. <ul style="list-style-type: none"> Retorna para um cozinheiro todos os pedidos com status “Em Andamento”. 	GET
/user/gettodospedidoscozinheiroafazer <ul style="list-style-type: none"> Retorna para um cozinheiro todos os pedidos com status “A Fazer”. 	GET

5. EXECUÇÃO DA API

A inicialização do servidor local da API é realizada com a execução do comando **php artisan serve** no terminal (aberto na pasta do projeto). Os end-points da API podem ser consumidos através do programa **POSTMAN**.

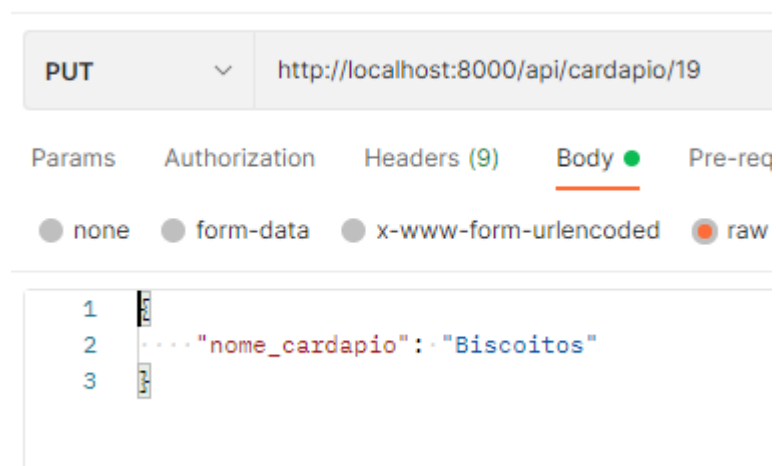
Em cada requisição, deverá ser selecionado o método HTTP correspondente (GET, POST, PUT, DELETE) bem como ser inserida a URL do end-point desejado.

Na chamada de end-points cujos métodos sejam POST ou PUT, é necessário criar o campo “content-type” preenchido com o valor “application/json” (indicando a entrada de dados através de JSON). O JSON pode ser inserido na aba “Body”



The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:8000/api/cardapio/19`. The 'Headers' tab is selected, showing a table with one header: 'content-type' with the value 'application/json'. There are 8 hidden headers.

	KEY	VALUE
<input checked="" type="checkbox"/>	content-type	application/json
	Key	Value



The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:8000/api/cardapio/19`. The 'Body' tab is selected, showing a JSON body: `{ "nome_cardapio": "Biscoitos" }`. The 'raw' radio button is selected.

```
1 {  
2   "nome_cardapio": "Biscoitos"  
3 }
```

A grande maioria dos end-points desta API necessitam de autenticação para o consumo de dados. Nestes casos, é necessário fazer login no endpoint “/login” e inserir o token de autenticação na aba “Authorization”.

