Documentação do
“Desafio Back-End Multiplier”
por Dalton Rogerio Alvarez

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 8.1.12 utilizando o Framework Laravel 9. O Banco de Dados utilizado foi o MariaDB 10.10.2.

A execução da API requer a criação de um banco de dados chamado “desafio-multiplier” e seu modelo de configuração de acesso está atualizado no arquivo .env.example do laravel. Basta renomeá-lo para .env e informar a senha do usuário root do servidor MySQL máquina que irá executar a aplicação.

Deve-se criar um banco de dados em seu gerenciador de banco de dados com o nome “desafio-multiplier”

Devemos ter o php 8.1.12 ou superior instalado na máquina e o composer.
Caso não tenha instalado segue o link do download https://getcomposer.org/download/

Após as configurações já aplicadas, composer instalado e o projeto clonado em uma pasta de trabalho, dentro dessa pasta do projeto executar o seguinte comando:

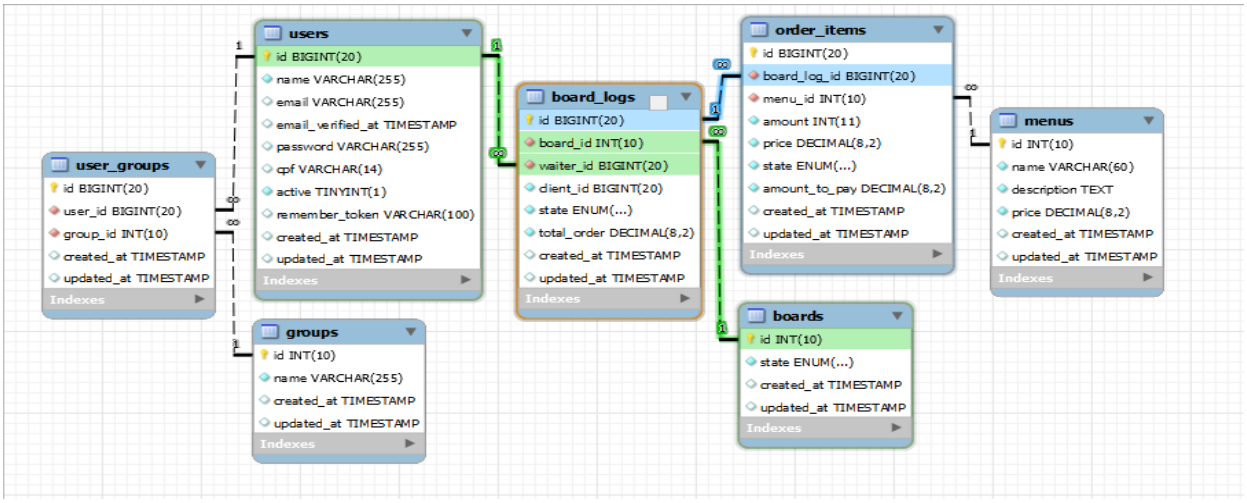
composer upgrade

Esse comando fará o restante do download dos pacotes que não estão disponíveis no repositório.

php artisan migrate:fresh --seed

Esse comando irá criar o banco de dados e popular as tabelas.
Conforme configurado em DatabaseSeeder.php ele gera 50 cardápios, 10.000 usuários e 400.000 pedidos

2 Tabelas do banco de dados



A ilustração acima apresenta o layout de relacionamento entre tabelas do banco de dados. A seguir, o sistema possui 7 tabelas principais mais as tabelas do sistema do framework.

users → Armazena todos usuários sendo Administradores, Clientes, Cozinheiros e Garçons

user_groups → Relaciona cada usuário ao seu grupo pertencente pois um usuário pode participar de um ou mais grupos diferentes, por exemplo… um usuário pode ser Administrador e Cozinheiro ao mesmo tempo ou participar dos 4 grupos simultaneamente se assim desejar.

groups → Grupos de usuários disponíveis no sistema, caso haja necessidade de expandir novos grupos basta incluir um registro em Grupos e fazer as devidas adaptações no sistema.

boards → mesas.

board_log → Registro de mesa, ou seja… A partir do momento que um cliente ocupa uma determinada mesa, seus pedidos serão identificados como registros da mesa pois o sistema terá informação sobre dados do cliente somente no fechamento do pedido.

order_items → Itens do pedido, cada item de pedido gerado no board_log será registrado em order_items.

Menus → Cardápios.

3 População do banco de dados

O banco de dados da api é iniciado com auxilio da biblioteca faker e os recursos de seeders disponíveis no laravel através do comando.

php artisan migrate:fresh –seed

4 ENDPOINTS API..
Cada endpoint tem sua documentação em um caminho /help

/api/user → usuários do sistema.

[GET]   /api/user/help   => Informações sobre o point solicitado.

[GET]   /api/user   => Apresenta todos usuários do sistema.

[POST]   /api/user -> Novos usuários para o sistema
              {name} => Nome do usuário
              {cpf} => CPF do usuário
              {email} => Email do usuário utilizado para login no sistema
              {password} => Senha de Usuário
              {password_confirmation} => Confirmação da senha informada

[PUT]   /user -> Alterar usuários do sistema
              {id} => id do usuário para localizar o registro à ser alterado
              name => Nome do usuário
              cpf => CPF do usuário
              email => Email do usuário utilizado para login no sistema
              password => Senha de Usuário.

[DELETE]   /api/user -> Desativar acesso à usuário
              {id} => id do usuário para localizar o registro à ser alterado

/api/menu → cardápios
[GET]   /api/menu => Apresenta todos os cardápios do sistema

[POST]   /api/menu -> Cadastrar novos cardápios
              {name} => Nome do cardápio
              {description} => Descrição detalhada do cardápio
              {price} => valor

[PUT]   /api/menu -> Alterar cardápios cadastrados
          {id} => id do cardápio à ser alterado
          name => Nome do cardápio
          description => 'Descrição detalhada do cardápio
          price => valor

[DELETE]   /api/menu -> Excluir cardápio cadastrados
       {id} => id do cardápio à ser excluído

/api/order → pedido

[GET]   /api/order/help   => Informações sobre o point solicitado.

[GET]   /api/order   => Apresenta todos pedidos do sistema para usuário grupo
       "administrador", usuário Grupo "garçom" somente seus pedidos e usuário grupo "cozinheiro"
       pedidos em "preparo" ou "aguardando"

[POST]   /api/order -> Novos pedidos para o sistema
          {board_id} => Número da mesa
          {waiter_id} => Id do garçom

[PUT]   /order -> Alterar pedidos do sistema
          {order_id} => id do pedido para localizar o registro à ser alterado

[DELETE] /api/order -> Deletar pedidos à usuário
          {order_id} => id do pedido para localizar o registro à ser excluído

/api/orderItems → itens do pedido
[GET] /api/orderItems/help => Informações sobre o point solicitado.

[GET] /api/orderItems => Apresenta todos pedidos do sistema para usuário grupo "administrador", usuário Grupo "garçom" somente seus pedidos e usuário grupo "cozinheiro" pedidos em "preparo" ou "aguardando".

[POST] /api/orderItems -> novos itens do pedido
{order_id} => id do pedido
{menu_id} => id do cardápio selecionado
state => Estado do pedido, caso não informado o state será "aguardando", mas pode ser informado como ["aguardando", "preparando", "pronto", "entregue"

[PUT] /api/orderItems -> Alterar usuários do sistema
{order_id} => id do pedido
{order_item_id} => id do ítem do pedido a ser alterado
menu_id => id do cardápio selecionado
state => Estado do pedido a ser definido ["aguardando", "preparando", "pronto", "entregue"]

[DELETE] /api/orderItems -> Excluir ítem do pedido
{order_id} => id do pedido
{order_item_id} => id do item do pedido

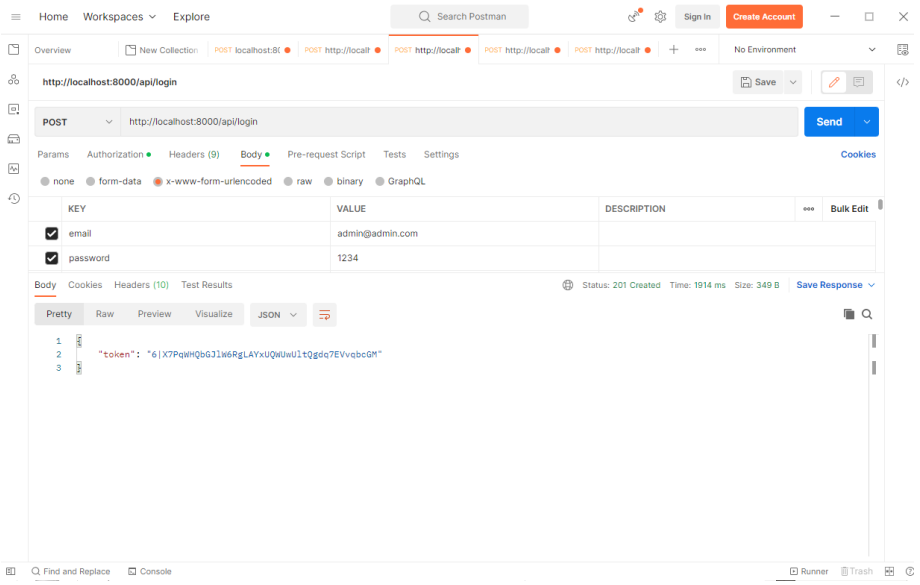
/api/orderReport – Relatórios de pedidos

[GET] /api/orderReport/help => Informações sobre o point solicitado.
[GET] /api/orderReport => Apresenta todos relatórios referentes à pedidos do sistema
[GET] /api/orderReport/client/largeOrder => Apresenta todos pedidos de cliente ordenados do maior para o menor valor de pedido.
[GET] /api/orderReport/client/firstOrder => Apresenta todos pedidos ordenados do primeiro até o último
[GET] /api/orderReport/client/lastOrder => Apresenta todos pedidos de cliente ordenados do último até o primeiro.
[GET] /api/orderReport/cooker/waiting => Todos os pedidos "aguardando" e "preparando"
[POST] /api/orderReport/cooker/waiting-client => Todos os pedidos, "aguardando" e "preparando" pelo client_id
[GET] /api/orderReport/cooker/ready => Todos os pedidos "pronto" e "entregue"
[POST] /api/orderReport/cooker/ready-client => Todos os pedidos, "pronto" e "entregue" pelo client_id
[GET] /api/orderReport/waiter/all => Todos pedidos, quando logado Garçom, pedidos do garçom, caso Administrador pedidos de todos os garçons
[GET] /api/orderReport/waiter/progress => Pedidos em andamento do garçom logado ou caso de login Administrador, todos os pedidos em andamento de todos os garçons
[GET] /api/orderReport/waiter/closed => Pedidos encerrados do garçom logado ou caso de login Administrador, todos os pedidos encerrados de todos os garçons
[GET] /api/orderReport/waiter => Visualizar pedidos referente ao garçom logado
[GET] /api/orderReport/cooker => Visualizar andamento de pedidos encaminhados pra cozinha
[GET] /api/orderReport/admin => Visualizar andamento de pedidos encaminhados pra cozinha
[POST] /api/orderReport/admin/day-week-year => Pedidos de um dia mês e ano
[POST] /api/orderReport/admin/month-year => Pedidos de um mes e ano
[POST] /api/orderReport/admin/year => Pedidos de um ano
[POST] /api/orderReport/admin/board => Pedidos gerados de uma mesa
[POST] /api/orderReport/admin/client => Pedidos gerados de um cliente de valor maior para o menor.

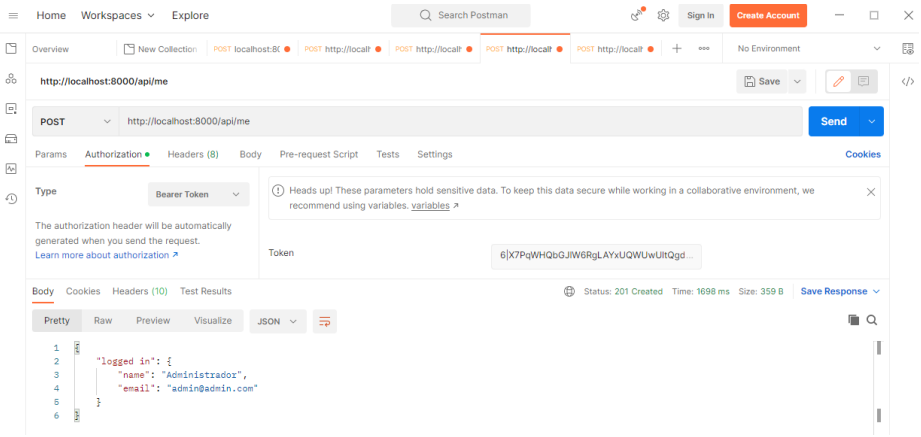
5. EXECUÇÃO DA API

A inicialização do servidor local da API é realizada com a execução do comando **php artisan serve** no terminal (dentro 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.

LOGIN



Acessando um endpoint com a autenticação:



usuários de testes do sistema:
username: admin@admin.com
password: 1234

username: cozinheiro@admin.com
password: 1234

username: garcom@admin.com
password: 1234

username: cliente@admin.com
password: 1234

demais usuários criados via DatabaseSeeder.php

password: password