**Aula - Sua primeira API Rest com Laravel**

**Code Experts (Nanderson Castro)**

<https://www.youtube.com/watch?v=oVRWQJE5a1c>

Resumo do vídeo feito por Roberto Pinheiro

# Instalando o Laravel e criando o projeto

Inicialmente instale o Composer em sua máquina. Em seguida entre com o comando:

**composer create-project --prefer-dist laravel/laravel rest\_api**

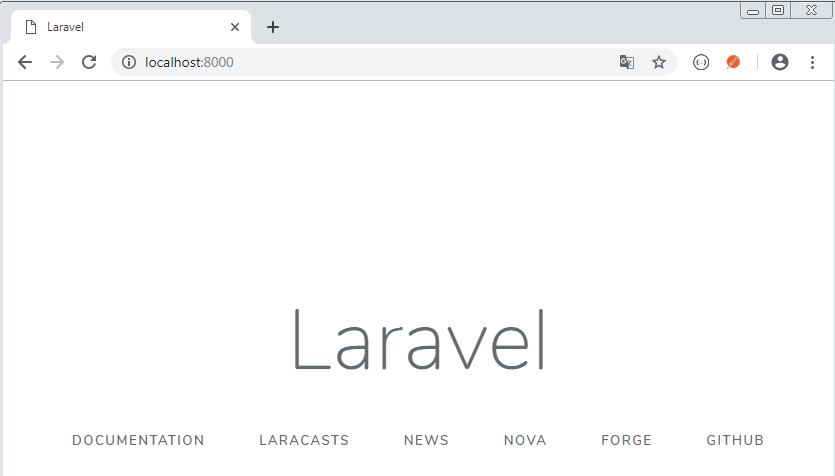
# Iniciando o servidor

Dentro da pasta do projeto entre com o comando:

**php artisan serve**

No browser:

**localhost:8000**



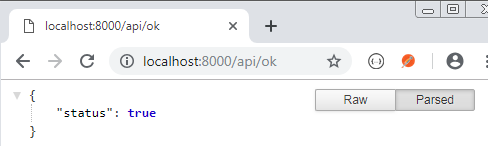
Instale a extensão para Google Chrome chamada **jsonFormatter**; ou **jsonView**.

**routes/api.php**

Route::get('/ok', function(){

return ['status' => true];

});



# Criando a migration

**php artisan make:migration create\_table\_products --create=products**



# Configurando a conexão com o banco de dados

DB\_CONNECTION=mysql

DB\_HOST=localhost

DB\_PORT=3306

DB\_DATABASE=api\_rest\_laravel

DB\_USERNAME=root

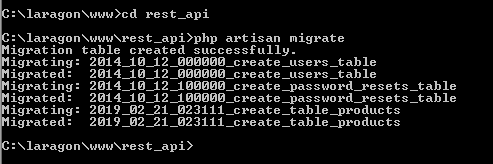
DB\_PASSWORD=

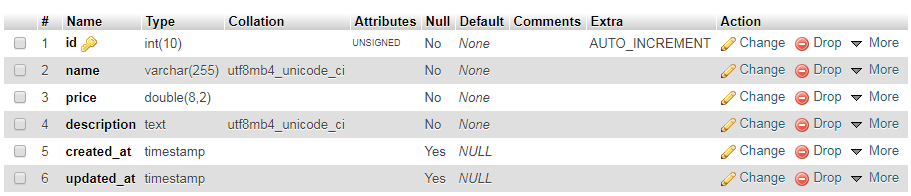
# Criando o banco de dados

Acesse o phpmyadmin e crie o banco de dados: api\_rest\_laravel

# Criando a tabela

**php artisan migrate**





# Criando uma factory

**php artisan make:factory ProductFactory**



**database/factories/ProductFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(\App\Product::class, function (Faker $faker) {

return [

'name' => $faker->name,

'price' => $faker->randomFloat(2, 0, 8),

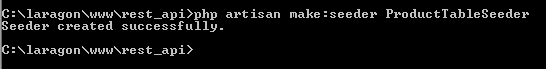
'description' => $faker->text

];

});

# Criando um seeder

**php artisan make:seeder ProductTableSeeder**



**database/seeds/ProductTableSeeder.php**

<?php

use Illuminate\Database\Seeder;

class ProductTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

factory(\App\Product::class, 10)->create();

}

}

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

// $this->call(UsersTableSeeder::class);

$this->call(ProductTableSeeder::class);

}

}

# Criando o model

**php artisan make:model Product**



**App/Product.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

protected $fillable = [

'name', 'price', 'description'

];

}

# Populando a tabela com 10 registros aleatórios

**php artisan db:seed**





# Criando as rotas

**routes/api.php**

Route::namespace('API')->name('api.')->group(function(){

Route::prefix('products')->group(function(){

Route::get('/', 'ProductController@index')->name('index\_products');

Route::get('/{id}', 'ProductController@show')->name('single\_products');

Route::post('/', 'ProductController@store')->name('store\_products');

Route::put('/{id}', 'ProductController@update')->name('update\_products');

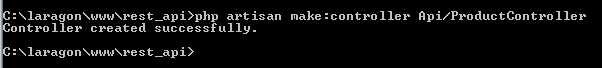
Route::delete('/{id}', 'ProductController@delete')->name('delete\_products');

});

});

# Criando o controller para a API

**php artisan make:controller Api/ProductController**



**app/Http/Controllers/Api/ProductController.php**

<?php

namespace App\Http\Controllers\Api;

use App\Product;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class ProductController extends Controller

{

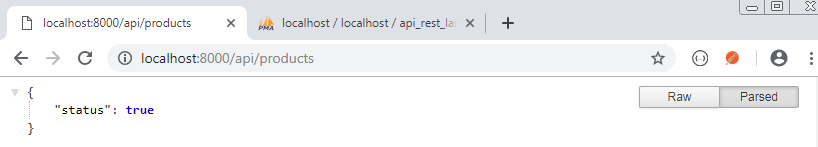
public function index()

{

return ['status' => true];

}

}



**app/Http/Controllers/Api/ProductController.php**

<?php

namespace App\Http\Controllers\Api;

use App\Product;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class ProductController extends Controller

{

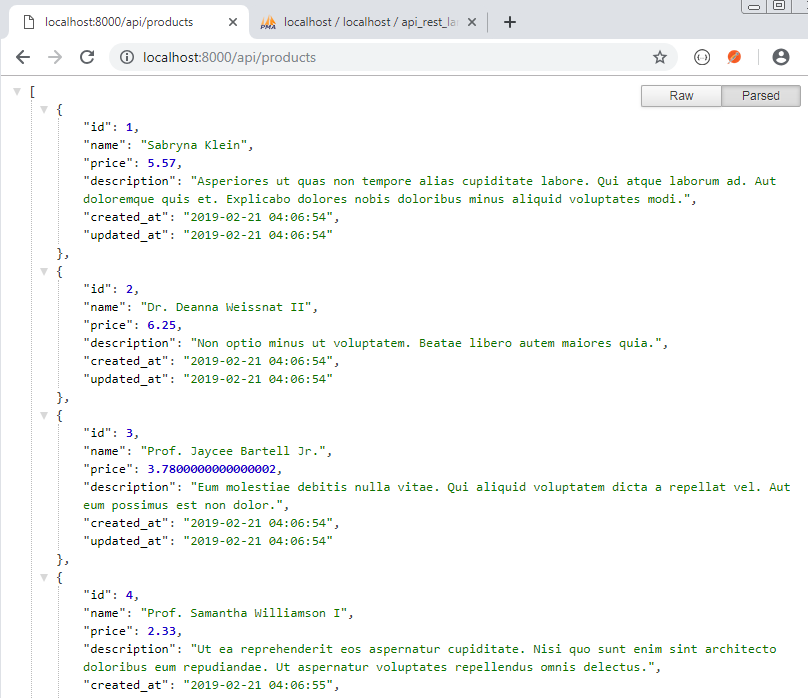
public function index()

{

return Product::all();

}

}



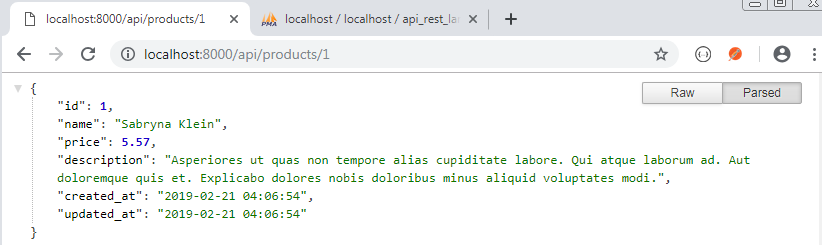
**app/Http/Controllers/Api/ProductController.php**

public function show(Product $id)

{

return $id;

}



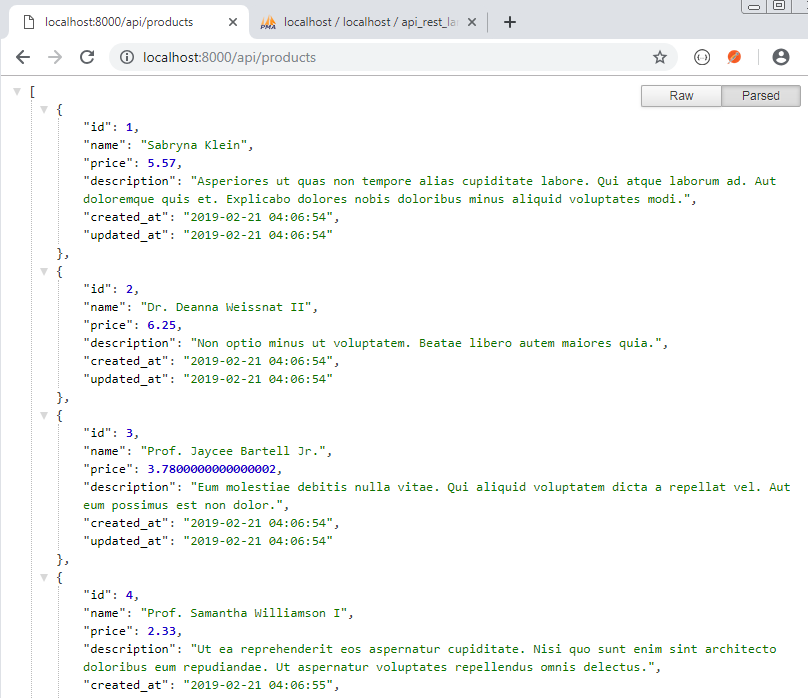
**app/Http/Controllers/Api/ProductController.php**

public function index()

{

return response()->json($this->product);

}



**app/Http/Controllers/Api/ProductController.php**

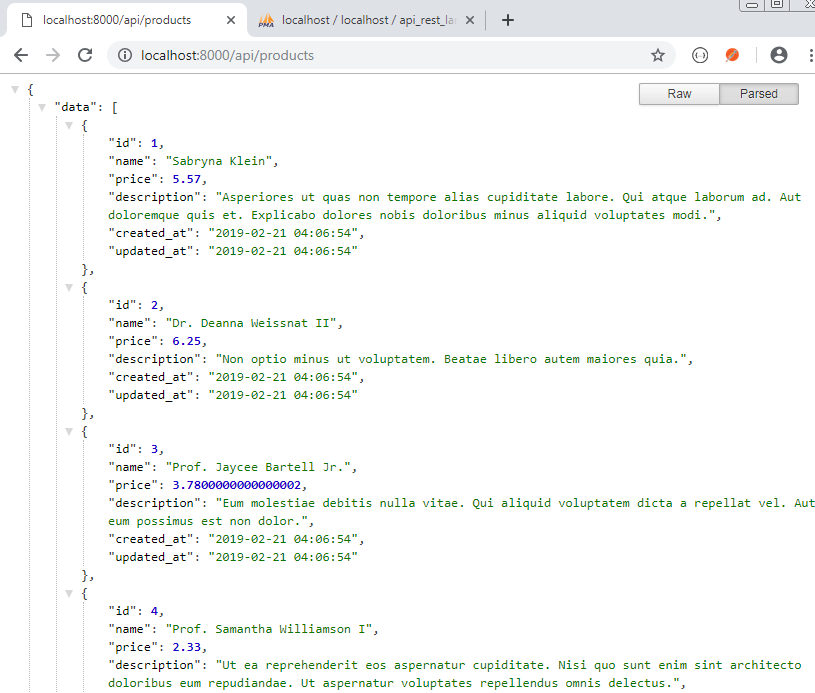
public function index()

{

$data = ['data' => $this->product->all()];

return response()->json($data);

}



**app/Http/Controllers/Api/ProductController.php**

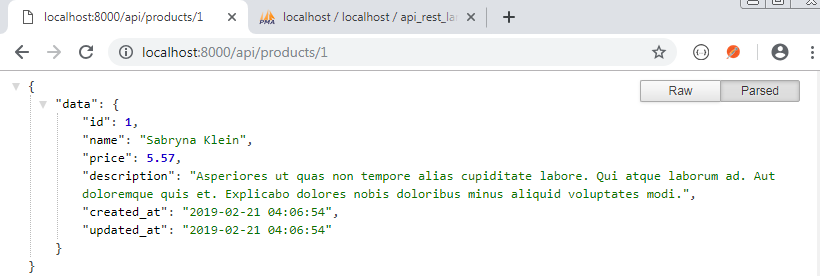
public function show(Product $id)

{

$data = ['data' => $id];

return response()->json($data);

}



# Paginando os produtos

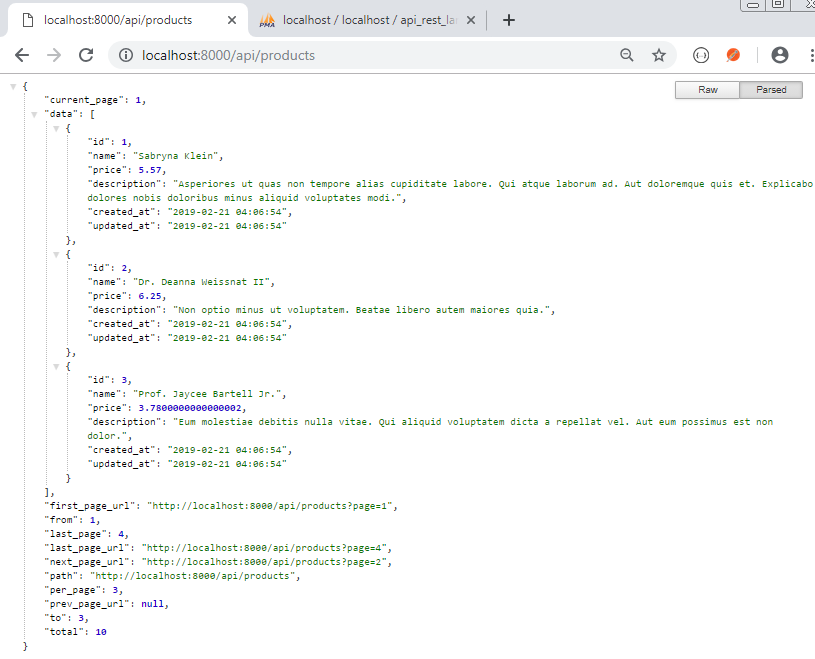
**app/Http/Controllers/Api/ProductController.php**

public function index()

{

return response()->json($this->product->paginate(3));

}



**app/Http/Controllers/Api/ProductController.php**

public function store(Request $request)

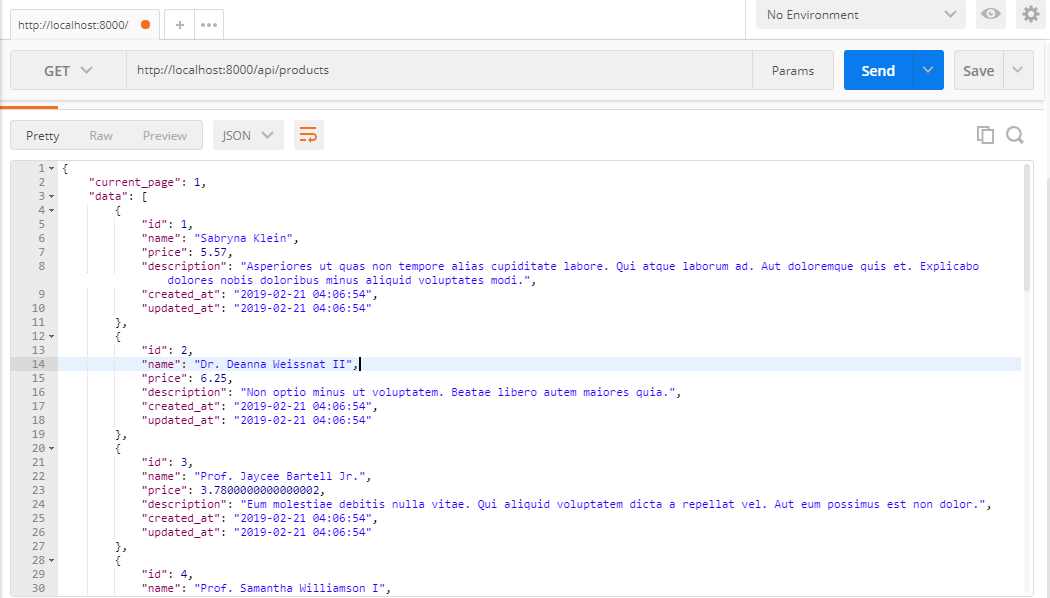
{

dd($request->all());

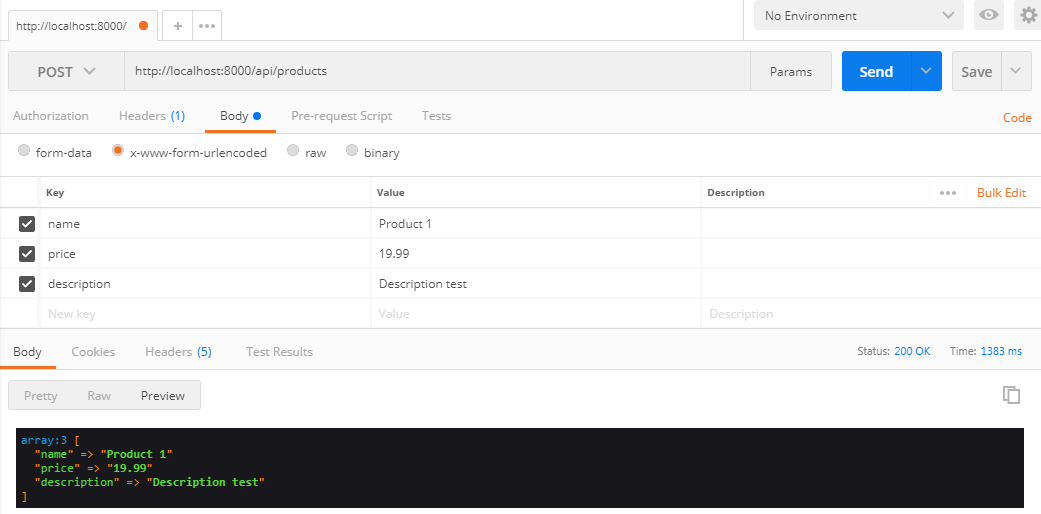
}

# Usando o Postman

## Listando produtos - GET



## Cadastrando produtos - POST



**app/Http/Controllers/Api/ProductController.php**

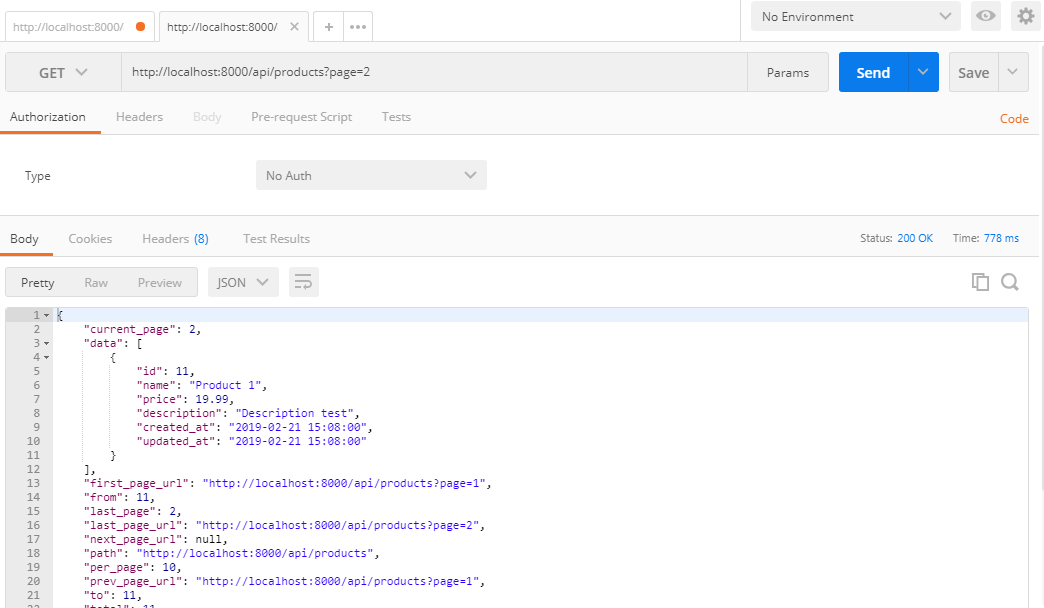
public function store(Request $request)

{

$productData = $request->all();

$this->product->create($productData);

}





**app/Http/Controllers/Api/ProductController.php**

public function store(Request $request)

{

try {

$productData = $request->all();

$this->product->create($productData);

$return = ['data' => ['msg' => 'Produto criado com sucesso!']];

return response()->json($return, 201);

} catch (\Exception $e) {

if(config('app.debug')) {

return response()->json(ApiError::errorMessage($e->getMessage(), 1010));

}

return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de salvar', 1010));

}

}

**app/API/ApiError.php**

<?php

namespace App\API;

class ApiError

{

public static function errorMessage($message, $code)

{

return [

'data' => [

'msg' => $message,

'code' => $code

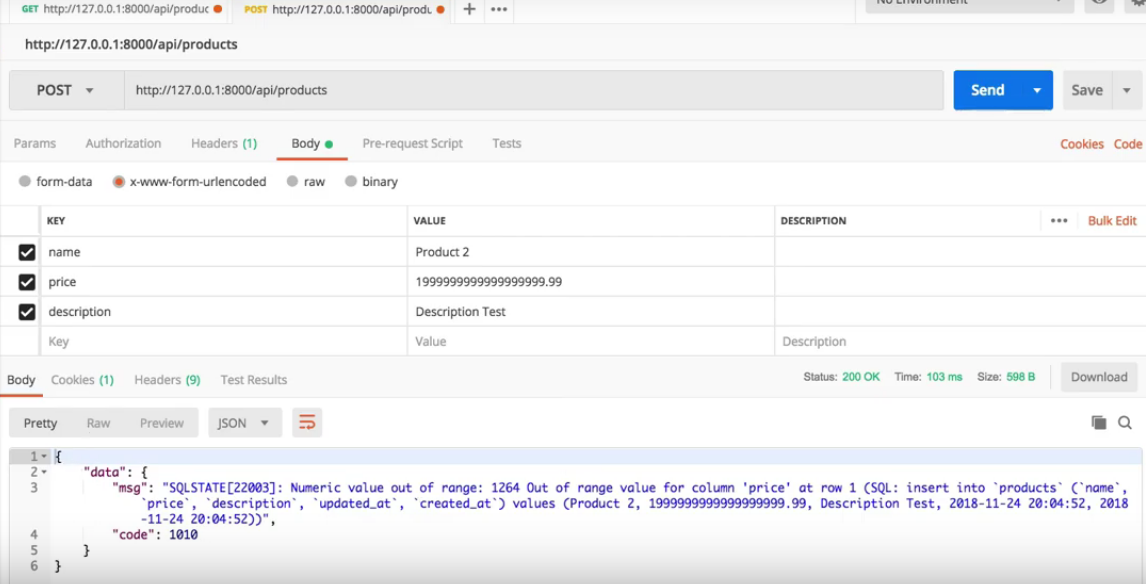
]

];

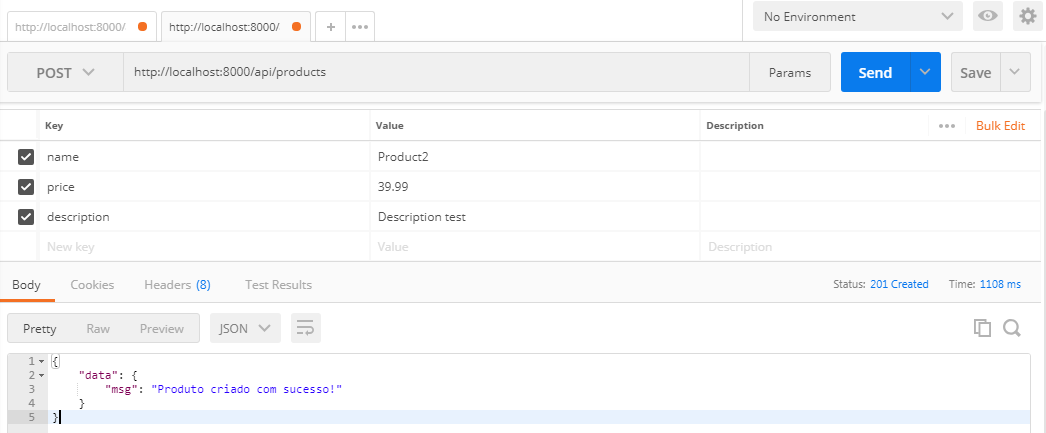
}

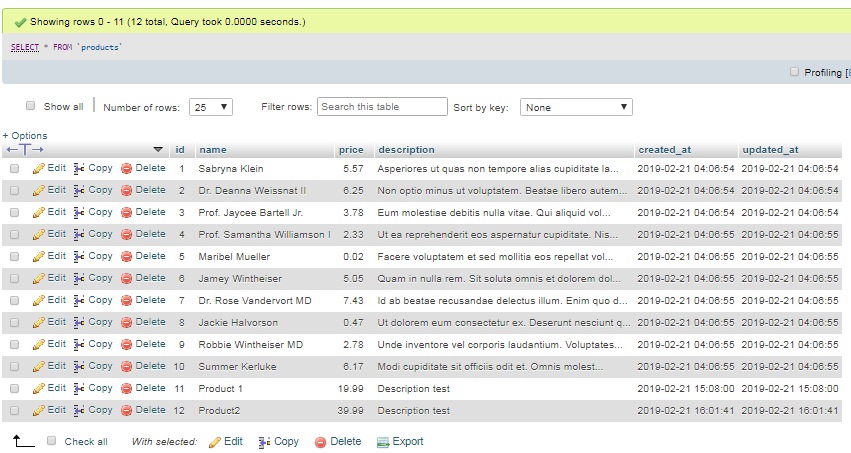
}

## Erro ao cadastrar produto



## Inserindo um novo produto





# Códigos de status de respostas HTTP

https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status

Os códigos de status das respostas HTTP indicam se uma requisição HTTP foi corretamente concluída. As respostas são agrupadas em cinco classes: respostas de informação, respostas de sucesso, redirecionamentos, erros do cliente e erros do servidor.

## Respostas informativas[Seção](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#Respostas_informativas)

[**100 Continue**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/100)

Essa resposta provisória indica que tudo ocorreu bem até agora e que o cliente deve continuar com a requisição ou ignorar se já concluiu o que gostaria.

[**101 Switching Protocol**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/101)

Esse código é enviado em resposta a um cabeçalho de solicitação [Upgrade](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Upgrade) pelo cliente, e indica o protocolo a que o servidor está alternando.

[**102 Processing**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/102)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

Este código indica que o servidor recebeu e está processando a requisição, mas nenhuma resposta está disponível ainda.

## Respostas de sucesso[Seção](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#Respostas_de_sucesso)

[**200 OK**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/200)

Estas requisição foi bem sucedida. O significado do sucesso varia de acordo com o método HTTP:  
GET: O recurso foi buscado e transmitido no corpo da mensagem.  
HEAD: Os cabeçalhos da entidade estão no corpo da mensagem.  
POST: O recurso descrevendo o resultado da ação e trasmitido no corpo da mensagem.  
TRACE: O corpo da mensagem contem a mensagem de requisição recebida pelo servidor

[**201 Created**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/201)

A requisição foi bem sucessida e um novo recurso foi criado como resultado. Esta é uma tipica resposta enviada após uma requisição PUT.

[**202 Accepted**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/202)

A requisição foi recebida mas nenhuma ação foi tomada sobre ela. Isto é uma requisição não-comprometedora, o que significa que não há nenhuma maneira no HTTP para enviar uma resposta assíncrona indicando o resultado do processamento da solicitação. Isto é indicado para casos onde outro processo ou servidor lida com a requisição, ou para processamento em lote.

[**203 Non-Authoritative Information**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/203)

Esse código de resposta significa que o conjunto de meta-informações retornadas não é o conjunto exato disponível no servidor de origem, mas coletado de uma cópia local ou de terceiros. Exceto essa condição, a resposta de 200 OK deve ser preferida em vez dessa resposta.

[**204 No Content**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/204)

Não há conteúdo para enviar para esta solicitação, mas os cabeçalhos podem ser úteis. O user-agent pode atualizar seus cabeçalhos em cache para este recurso com os novos.

[**205 Reset Content**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/205)

Esta requisição é enviada após realizanda a solicitação para informar ao user agentredefinir a visualização do documento que enviou essa solicitação.

[**206 Partial Content**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/206)

Esta resposta é usada por causa do cabeçalho de intervalo enviado pelo cliente para separar o download em vários fluxos.

[**207 Multi-Status**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/207)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

Uma resposta Multi-Status transmite informações sobre vários recursos em situações em que vários códigos de status podem ser apropriados.

[**208 Multi-Status**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/208)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

Usado dentro de um DAV: elemento de resposta propstat para evitar enumerar os membros internos de várias ligações à mesma coleção repetidamente.

[**226 IM Used**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/226)**(**[**HTTP Delta encoding**](https://tools.ietf.org/html/rfc3229)**)**

O servidor cumpriu uma solicitação GET para o recurso e a resposta é uma representação do resultado de uma ou mais manipulações de instância aplicadas à instância atual.

## Mensagens de redirecionamento[Seção](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#Mensagens_de_redirecionamento)

[**300 Multiple Choice**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/300)

A requisição tem mais de uma resposta possível. User-agent ou o user deve escolher uma delas. Não há maneira padrão para escolher uma das respostas.

[**301 Moved Permanently**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/301)

Esse código de resposta significa que a URI do recurso requerido mudou. Provavelmente, a nova URI será especificada na resposta.

[**302 Found**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/302)

Esse código de resposta significa que a URI do recurso requerido foi mudada temporariamente. Novas mudanças na URI poderão ser feitas no futuro. Portanto, a mesma URI deve ser usada pelo cliente em requisições futuras.

[**303 See Other**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/303)

O servidor manda essa resposta para instruir ao cliente buscar o recurso requisitado em outra URI com uma requisição GET.

[**304 Not Modified**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/304)

Essa resposta é usada para questões de cache. Diz ao cliente que a resposta não foi modificada. Portanto, o cliente pode usar a mesma versão em cache da resposta.

**305 Use Proxy**

Foi definida em uma versão anterior da especificação HTTP para indicar que uma resposta deve ser acessada por um proxy. Foi depreciada por questões de segurança em respeito a configuração em banda de um proxy.

**306 unused**

Esse código de resposta não é mais utilizado, encontra-se reservado. Foi usado numa versão anterior da especificação HTTP 1.1.

[**307 Temporary Redirect**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/307)

O servidor mandou essa resposta direcionando o cliente a buscar o recurso requisitado em outra URI com o mesmo método que foi utilizado na requisição original. Tem a mesma semântica do código 302 Found, com a exceção de que o user-agent não deve mudar o método HTTP utilizado: se um POST foi utilizado na primeira requisição, um POST deve ser utilizado na segunda.

[**308 Permanent Redirect**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/308)

Esse código significa que o recurso agora está permanentemente localizado em outra URI, especificada pelo cabeçalho de resposta Location. Tem a mesma semântica do código de resposta HTTP 301 Moved Permanently  com a exceção de que o user-agent não deve mudar o método HTTP utilizado: se um POST foi utilizado na primeira requisição, um POST deve ser utilizado na segunda.

## Respostas de erro do Cliente[Seção](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#Respostas_de_erro_do_Cliente)

[**400 Bad Request**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/400)

Essa resposta significa que o servidor não entendeu a requisição pois está com uma sintaxe inválida.

[**401 Unauthorized**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/401)

Embora o padrão HTTP especifique "unauthorized", semanticamente, essa resposta significa "unauthenticated". Ou seja, o cliente deve se autenticar para obter a resposta solicitada.

**402 Payment Required**

Este código de resposta está reservado para uso futuro. O objetivo inicial da criação deste código era usá-lo para sistemas digitais de pagamento porém ele não está sendo usado atualmente.

[**403 Forbidden**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/403)

O cliente não tem direitos de acesso ao conteúdo portanto o servidor está rejeitando dar a resposta. Diferente do código 401, aqui a identidade do cliente é conhecida.

[**404 Not Found**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/404)

O servidor não pode encontrar o recurso solicitado. Este código de resposta talvez seja o mais famoso devido à frequência com que acontece na web.

[**405 Method Not Allowed**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/405)

O método de solicitação é conhecido pelo servidor, mas foi desativado e não pode ser usado. Os dois métodos obrigatórios, GET e HEAD, nunca devem ser desabilitados e não devem retornar este código de erro.

[**406 Not Acceptable**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/406)

Essa resposta é enviada quando o servidor da Web após realizar a negociação de conteúdo orientada pelo servidor, não encontra nenhum conteúdo seguindo os critérios fornecidos pelo agente do usuário.

[**407 Proxy Authentication Required**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/407)

Semelhante ao **401**porem é necessário que a autenticação seja feita por um proxy.

[**408 Request Timeout**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/408)

Esta resposta é enviada por alguns servidores em uma conexão ociosa, mesmo sem qualquer requisição prévia pelo cliente. Ela significa que o servidor gostaria de derrubar esta conexão em desuso. Esta resposta é muito usada já que alguns navegadores, como Chrome, Firefox 27+, ou IE9, usam mecanismos HTTP de pré-conexão para acelerar a navegação. Note também que alguns servidores meramente derrubam a conexão sem enviar esta mensagem.

[**409 Conflict**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/409)

Esta resposta será enviada quando uma requisição conflitar com o estado corrente do servidor.

[**410 Gone**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/410)

Esta resposta será enviada quando o conteúdo requisitado foi deletado do servidor.

[**411 Length Required**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/411)

O servidor rejeitou a requisição porque o campo Content-Length do cabeçalho não está definido e o servidor o requer.

[**412 Precondition Failed**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/412)

O cliente indicou nos seus cabeçalhos pré-condições que o servidor não atende.

[**413 Payload Too Large**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/413)

A entidade requisição é maior do que os limites definidos pelo servidor; o servidor pode fechar a conexão ou retornar um campo de cabeçalho Retry-After.

[**414 URI Too Long**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/414)

A URI requisitada pelo cliente é maior do que o servidor aceita para interpretar.

[**415 Unsupported Media Type**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/415)

O formato de mídia dos dados requisitados não é suportado pelo servidor, então o servidor rejeita a requisição.

[**416 Requested Range Not Satisfiable**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/416)

O trecho especificado pelo campo Range do cabeçalho na requisição não pode ser preenchido; é possível que o trecho esteja fora do tamanho dos dados da URI alvo.

[**417 Expectation Failed**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/417)

Este código de resposta significa que a expectativa indicada pelo campo Expect do cabeçalho da requisição não pode ser satisfeita pelo servidor.

[**418 I'm a teapot**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/418)

O servidor recusa a tentativa de coar café num bule de chá.

[**421 Misdirected Request**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/421)

A requisição foi direcionada a um servidor inapto a produzir a resposta. Pode ser enviado por um servidor que não está configurado para produzir respostas para a combinação de esquema ("scheme") e autoridade inclusas na URI da requisição.

[**422 Unprocessable Entity**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/422)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

A requisição está bem formada mas inabilitada para ser seguida devido a erros semânticos.

[**423 Locked**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/423)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

O recurso sendo acessado está chaveado.

[**424 Failed Dependency**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/424)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

A requisição falhou devido a falha em requisição prévia.

[**426 Upgrade Required**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/426)

O servidor se recusa a executar a requisição usando o protocolo corrente mas estará pronto a fazê-lo após o cliente atualizar para um protocolo diferente. O servidor envia um cabeçalho [Upgrade](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Headers/Upgrade) numa resposta 426 para indicar o(s) protocolo(s) requeridos.

[**428 Precondition Required**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/428)

O servidor de origem requer que a resposta seja condicional. Feito para prevenir o problema da 'atualização perdida', onde um cliente pega o estado de um recurso (GET) , modifica-o, e o põe de volta no servidor (PUT), enquanto um terceiro modificou o estado no servidor, levando a um conflito.

[**429 Too Many Requests**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/429)

O usuário enviou muitas requisições num dado tempo ("limitação de frequência").

[**431 Request Header Fields Too Large**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/431)

O servidor não quer processar a requisição porque os campos de cabeçalho são muito grandes. A requisição PODE ser resubmetida depois de reduzir o tamanho dos campos de cabeçalho.

[**451 Unavailable For Legal Reasons**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/451)

O usuário requisitou um recurso ilegal, tal como uma página censurada por um governo.

## Respostas de erro do Servidor[Seção](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status#Respostas_de_erro_do_Servidor)

[**500 Internal Server Error**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/500)

O servidor encontrou uma situação com a qual não sabe lidar.

[**501 Not Implemented**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/501)

O método da requisição não é suportado pelo servidor e não pode ser manipulado. Os únicos métodos exigidos que servidores suportem (e portanto não devem retornar este código) são GET e HEAD.

[**502 Bad Gateway**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/502)

Esta resposta de erro significa que o servidor, ao trabalhar como um gateway a fim de obter uma resposta necessária para manipular a requisição, obteve uma resposta inválida.

[**503 Service Unavailable**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/503)

O servidor não está pronto para manipular a requisição. Causas comuns são um servidor em manutenção ou sobrecarregado. Note que junto a esta resposta, uma página amigável explicando o problema deveria ser enviada. Estas respostas devem ser usadas para condições temporárias e o cabeçalho HTTP Retry-After: deverá, se posível, conter o tempo estimado para recuperação do serviço. O webmaster deve também tomar cuidado com os cabaçalhos relacionados com o cache que são enviados com esta resposta, já que estas respostas de condições temporárias normalmente não deveriam ser postas em cache.

[**504 Gateway Timeout**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/504)

Esta resposta de erro é dada quando o servidor está atuando como um gateway e não obtém uma resposta à tempo.

[**505 HTTP Version Not Supported**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/505)

A versão HTTP usada na requisição não é suportada pelo servidor.

[**506 Variant Also Negotiates**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/506)

O servidor tem um erro de configuração interno: a negociação transparente de conteúdo para a requisição resulta em uma referência circular.

[**507 Insufficient Storage**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/507)

O servidor tem um erro interno de configuração: o recurso variante escolhido está configurado para entrar em negociação transparente de conteúdo com ele mesmo, e portanto não é uma ponta válida no processo de negociação.

[**508 Loop Detected**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/508)**(**[**WebDAV**](https://developer.mozilla.org/pt-BR/docs/Glossary/WebDAV)**)**

O servidor detectou um looping infinito ao processar a requisição.

[**510 Not Extended**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/510)

Exigem-se extenções posteriores à requisição para o servidor atendê-la.

[**511 Network Authentication Required**](https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/511)

O código de status 511 indica que o cliente precisa se autenticar para ganhar acesso à rede.

## Atualizar (alterar) um produto - PUT

**app/Http/Controllers/Api/ProductController.php**

public function update(Request $request, $id)

{

try {

$productData = $request->all();

$product = $this->product->find($id);

$product->update($productData);

$return = ['data' => ['msg' => 'Produto atualizado com sucesso!']];

return response()->json($return, 201);

} catch (\Exception $e) {

if(config('app.debug')) {

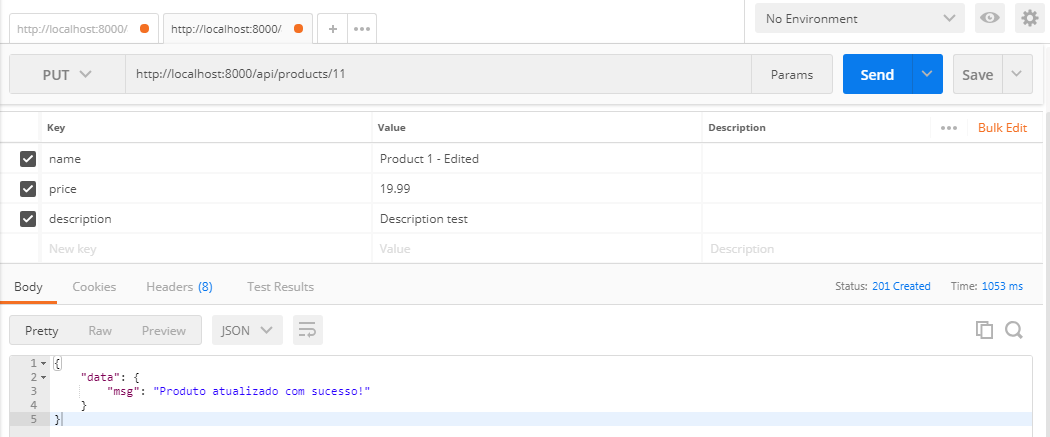
return response()->json(ApiError::errorMessage($e->getMessage(), 1011), 500);

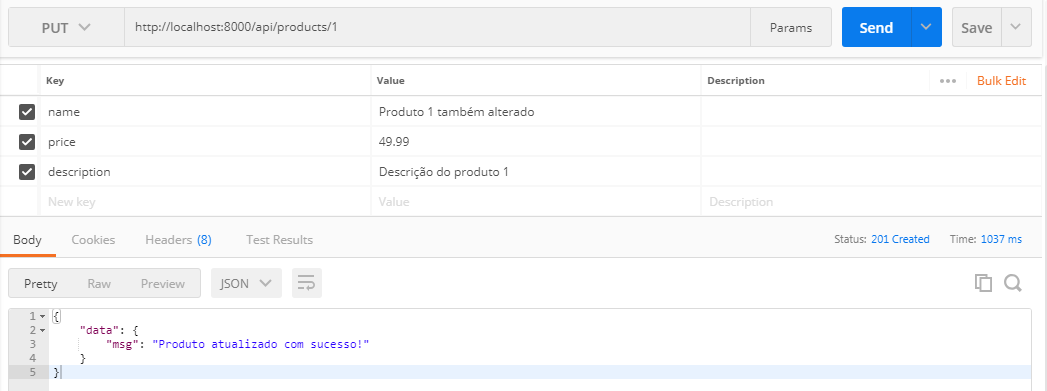
}

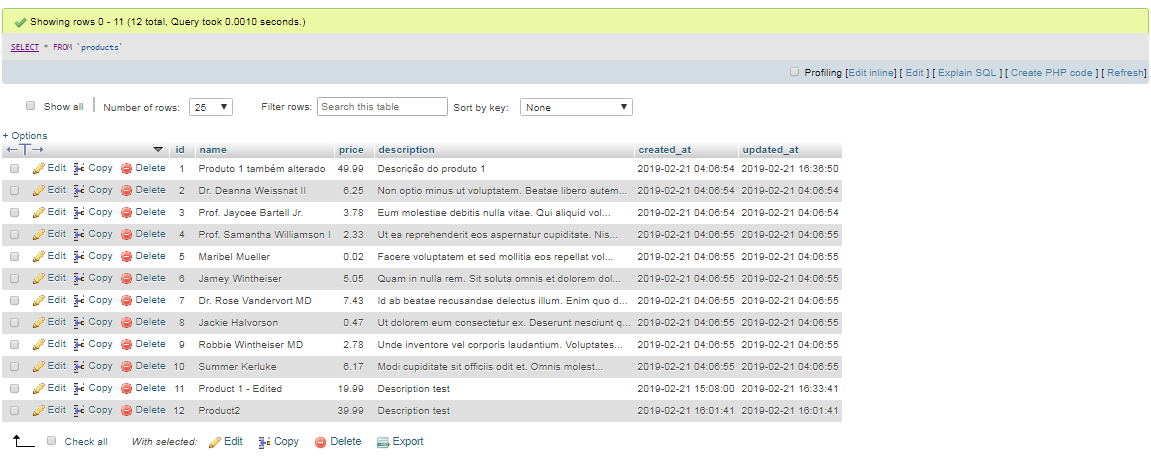
return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de atualizar', 1011), 500);

}

}







## Excluindo (deletando) um produto - DELETE

**app/Http/Controllers/Api/ProductController.php**

public function delete(Product $id)

{

try {

$id->delete();

return response()->json(['data' => ['msg' => 'Produto: ' . $id->name . ' removido com sucesso!']], 200);

}catch (\Exception $e) {

if(config('app.debug')) {

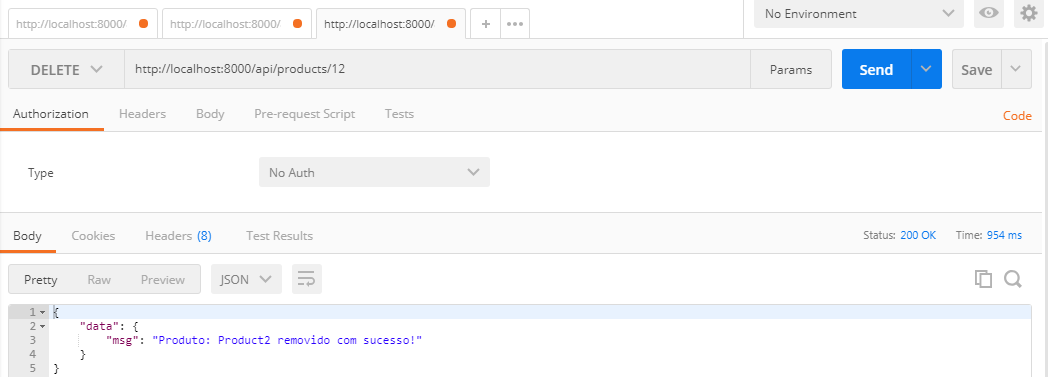
return response()->json(ApiError::errorMessage($e->getMessage(), 1012), 500);

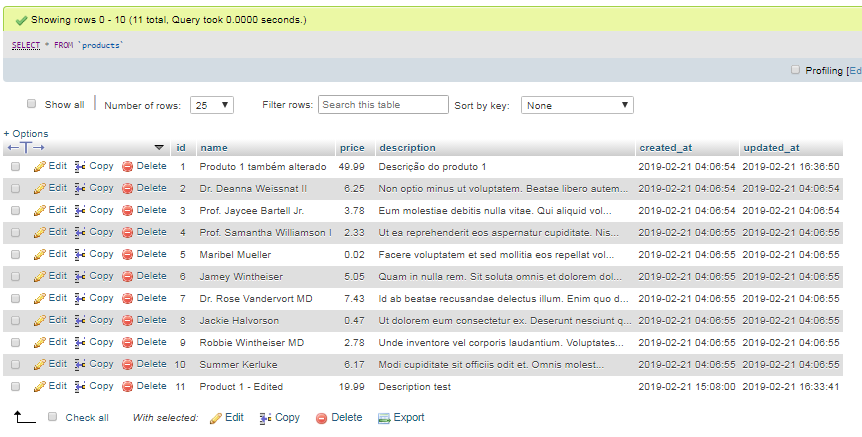
}

return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de remover', 1012), 500);

}

}





# Busca de produto com mensagem de erro

**app/Http/Controllers/Api/ProductController.php**

public function show(Product $id)

{

$product = $this->product->find($id);

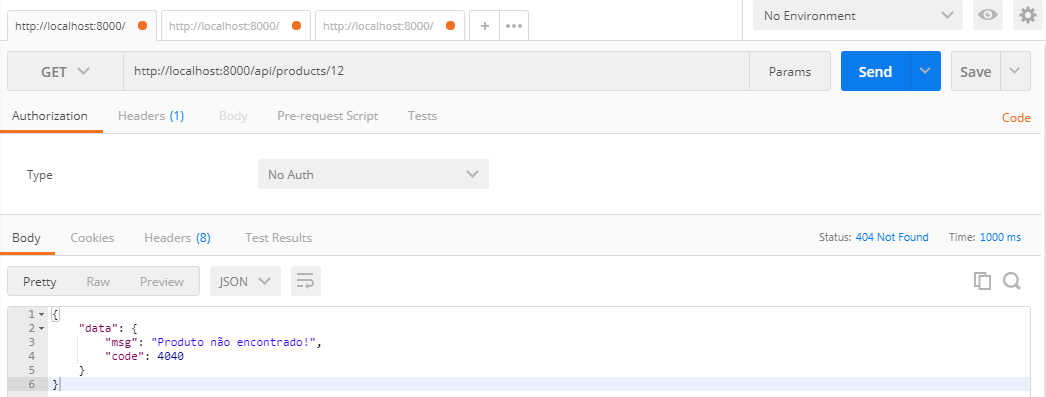
if(! $product) return response()->json(ApiError::errorMessage('Produto não encontrado!', 4040), 404);

$data = ['data' => $product];

return response()->json($data);

}

## Listar um produto não existente



# Final

**app/Http/Controllers/Api/ProductController.php**

<?php

namespace App\Http\Controllers\Api;

use App\API\ApiError;

use App\Product;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class ProductController extends Controller

{

/\*\*

\* @var Product

\*/

private $product;

public function \_\_construct(Product $product)

{

$this->product = $product;

}

public function index()

{

return response()->json($this->product->paginate(10));

}

public function show($id)

{

$product = $this->product->find($id);

if(! $product) return response()->json(ApiError::errorMessage('Produto não encontrado!', 4040), 404);

$data = ['data' => $product];

return response()->json($data);

}

public function store(Request $request)

{

try {

$productData = $request->all();

$this->product->create($productData);

$return = ['data' => ['msg' => 'Produto criado com sucesso!']];

return response()->json($return, 201);

} catch (\Exception $e) {

if(config('app.debug')) {

return response()->json(ApiError::errorMessage($e->getMessage(), 1010), 500);

}

return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de salvar', 1010), 500);

}

}

public function update(Request $request, $id)

{

try {

$productData = $request->all();

$product = $this->product->find($id);

if(! $product) {

return response()->json(ApiError::errorMessage('Produto não encontrado!', 4040), 404);

} else {

$product->update($productData);

$return = ['data' => ['msg' => 'Produto atualizado com sucesso!']];

return response()->json($return, 201);

}

} catch (\Exception $e) {

if(config('app.debug')) {

return response()->json(ApiError::errorMessage($e->getMessage(), 1011), 500);

}

return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de atualizar', 1011), 500);

}

}

public function delete(Product $id)

{

try {

$id->delete();

return response()->json(['data' => ['msg' => 'Produto: ' . $id->name . ' removido com sucesso!']], 200);

}catch (\Exception $e) {

if(config('app.debug')) {

return response()->json(ApiError::errorMessage($e->getMessage(), 1012), 500);

}

return response()->json(ApiError::errorMessage('Houve um erro ao realizar operação de remover', 1012), 500);

}

}

}