

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
```

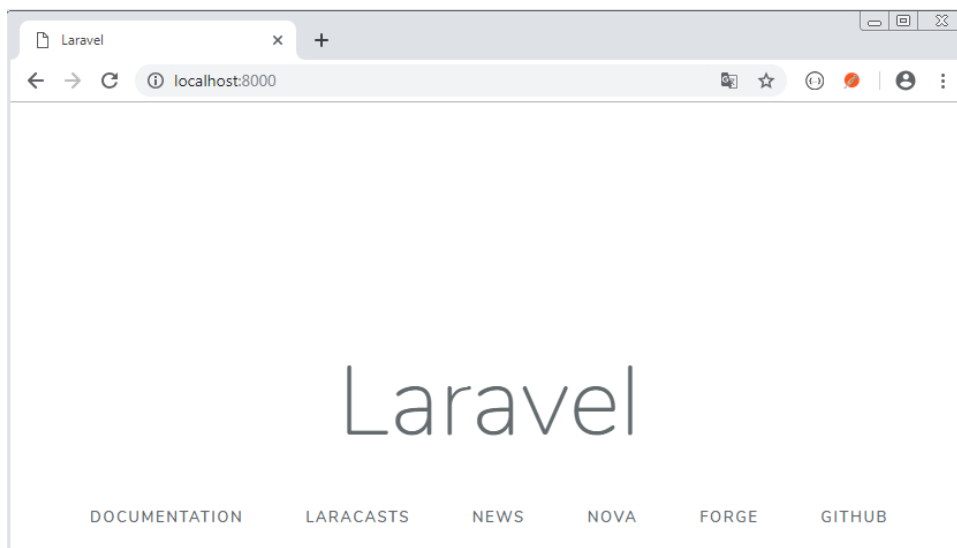
Iniciando o servidor

Dentro da pasta do projeto entre com o comando:

```
php artisan serve
```

No browser:

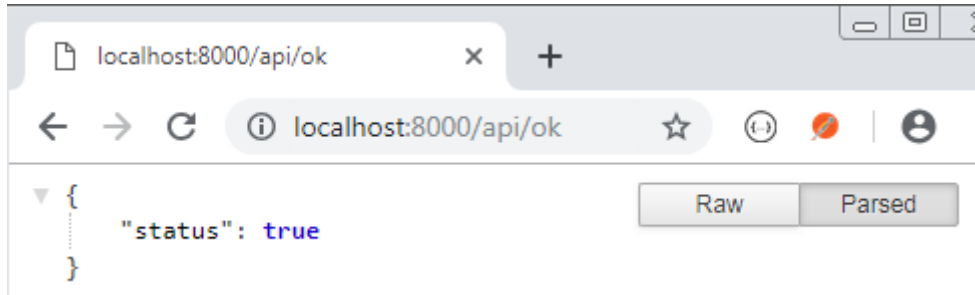
localhost:8000



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
```

```
C:\laragon\www\rest_api>php artisan make:migration create_table_products --creat  
e=products  
Created Migration: 2019_02_21_023111_create_table_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

```
C:\laragon\www>cd rest_api

C:\laragon\www\rest_api>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated: 2014_10_12_000000_create_users_table
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated: 2014_10_12_100000_create_password_resets_table
Migrating: 2019_02_21_023111_create_table_products
Migrated: 2019_02_21_023111_create_table_products

C:\laragon\www\rest_api>
```

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1 id	int(10)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
<input type="checkbox"/>	2 name	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
<input type="checkbox"/>	3 price	double(8,2)			No	None			Change Drop More
<input type="checkbox"/>	4 description	text	utf8mb4_unicode_ci		No	None			Change Drop More
<input type="checkbox"/>	5 created_at	timestamp			Yes	NULL			Change Drop More
<input type="checkbox"/>	6 updated_at	timestamp			Yes	NULL			Change Drop More

Criando uma factory

php artisan make:factory ProductFactory

```
C:\laragon\www\rest_api>php artisan make:factory ProductFactory
Factory created successfully.

C:\laragon\www\rest_api>
```

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

```
C:\laragon\www\rest_api>php artisan make:seeder ProductTableSeeder
Seeder created successfully.
C:\laragon\www\rest_api>
```

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

```
C:\laragon\www\rest_api>php artisan make:model Product
Model created successfully.
C:\laragon\www\rest_api>
```

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

```
C:\laragon\www\rest_api>php artisan db:seed
Seeding: ProductTableSeeder
Database seeding completed successfully.
C:\laragon\www\rest_api>
```

| + Options | | | | | | | | |
|--------------------------|------|------|--------|------|-----------------------------|-------------|---|---------------------|
| ← T → | | | | | | | | |
| | | | id | name | price | description | created_at | updated_at |
| <input type="checkbox"/> | Edit | Copy | Delete | 1 | Sabryna Klein | 5.57 | Asperiores ut quas non tempore alias cupiditate la... | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit | Copy | Delete | 2 | Dr. Deanna Weissnat II | 6.25 | Non optio minus ut voluptatem. Beatae libero autem... | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit | Copy | Delete | 3 | Prof. Jaycee Bartell Jr. | 3.78 | Eum molestiae debitis nulla vitae. Qui aliquid vol... | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit | Copy | Delete | 4 | Prof. Samantha Williamson I | 2.33 | Ut ea reprehenderit eos aspernatur cupiditate. Nis... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 5 | Maribel Mueller | 0.02 | Facere voluptatem et sed mollitia eos repellat vol... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 6 | Jamey Wintheiser | 5.05 | Quam in nulla rem. Sit soluta omnis et dolorem dol... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 7 | Dr. Rose Vandervort MD | 7.43 | Id ab beatae recusandae delectus illum. Enim quo d... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 8 | Jackie Halvorson | 0.47 | Ut dolorem eum consectetur ex. Deserunt nesciunt q... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 9 | Robbie Wintheiser MD | 2.78 | Unde inventore vel corporis laudantium. Voluptates... | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit | Copy | Delete | 10 | Summer Kerluke | 6.17 | Modi cupiditate sit officis odit et. Omnis molest... | 2019-02-21 04:06:55 |

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

```
C:\laragon\www\rest_api>php artisan make:controller Api/ProductController
Controller created successfully.
C:\laragon\www\rest_api>
```

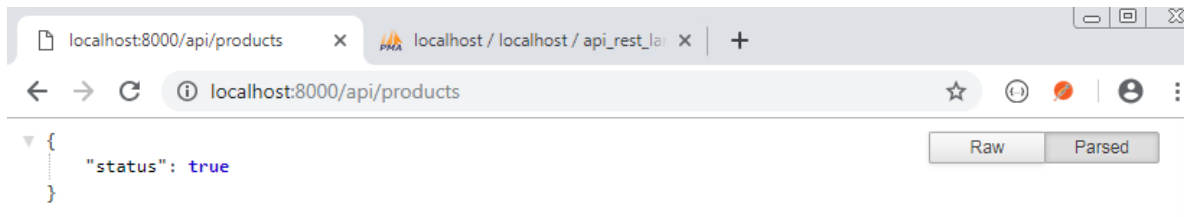
app/Http/Controllers/Api/ProductController.php

```
<?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;
```

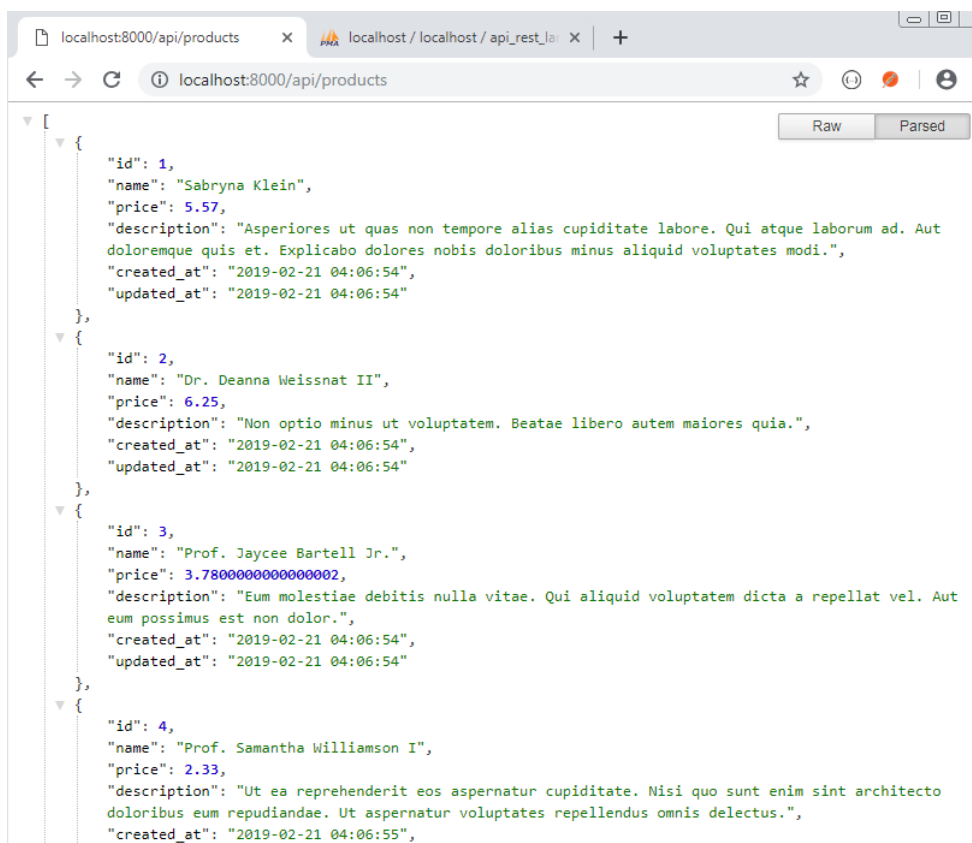
```
use App\Product;
```

```
use Illuminate\Http\Request;
```

```
use App\Http\Controllers\Controller;
```

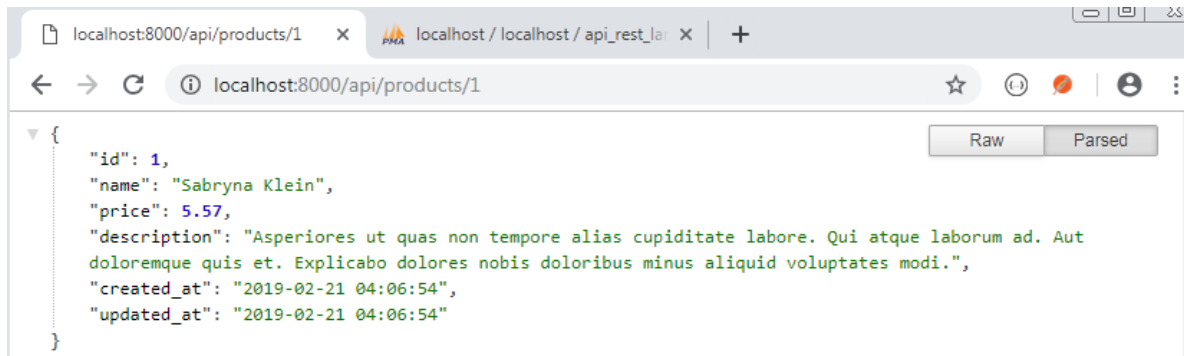
```
class ProductController extends Controller
```

```
{  
    public function index()  
    {  
        return Product::all();  
    }  
}
```



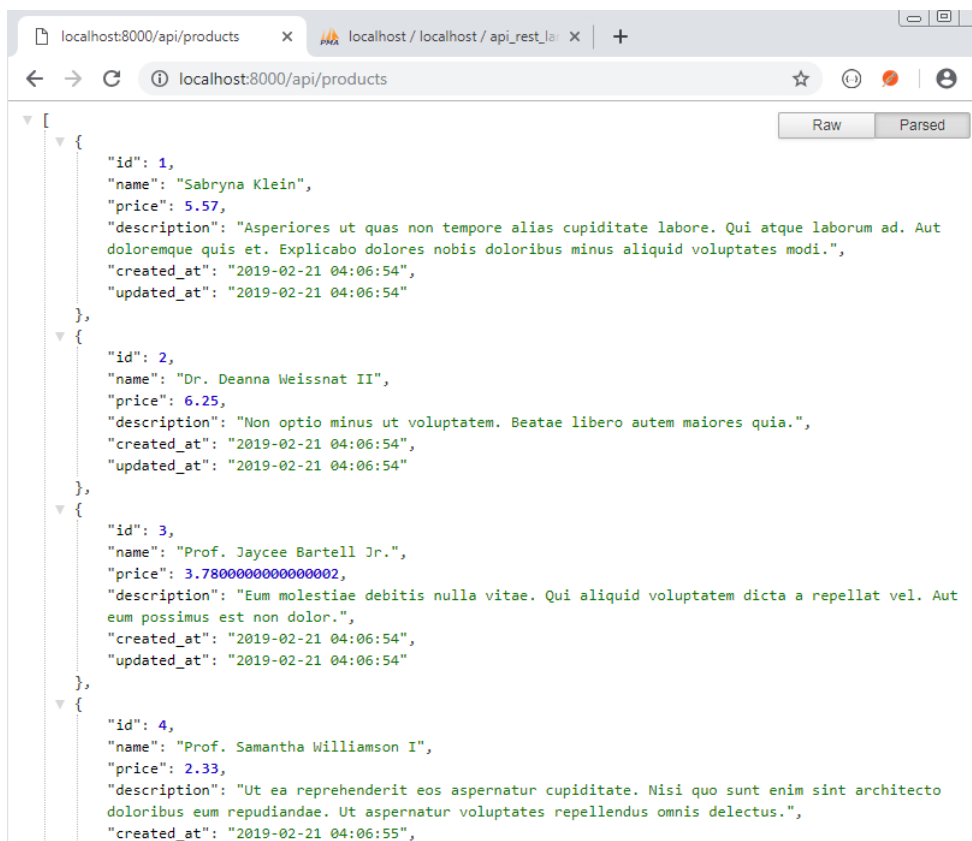
app/Http/Controllers/Api/ProductController.php

```
public function show(Product $id)
{
    return $id;
}
```



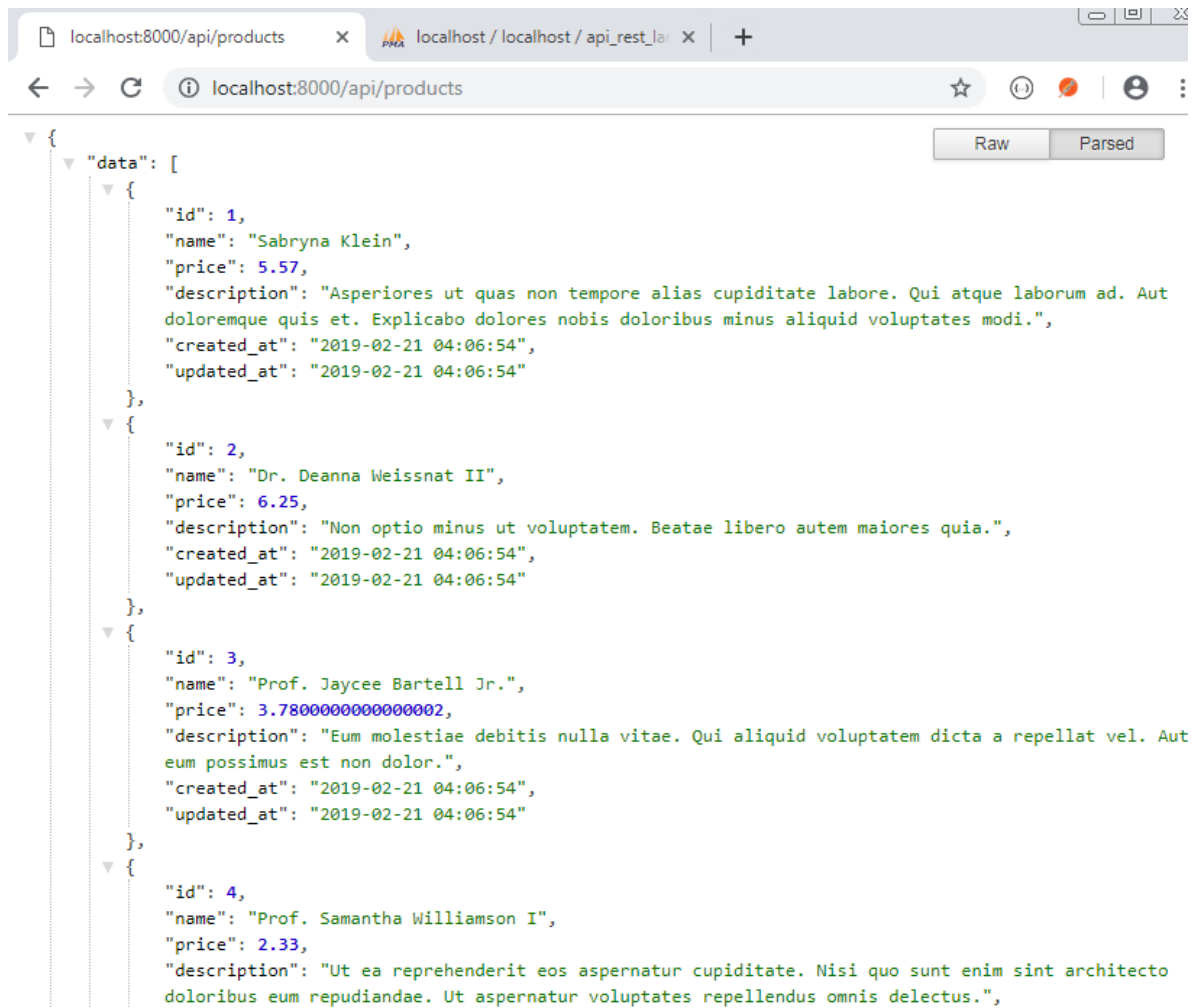
app/Http/Controllers/Api/ProductController.php

```
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);
}
```

```
{
  "data": {
    "id": 1,
    "name": "Sabryna Klein",
    "price": 5.57,
    "description": "Asperiores ut quas non tempore alias cupiditate labore. Qui atque laborum ad. Aut doloremque quis et. Explicabo dolores nobis doloribus minus aliquid voluptates modi.",
    "created_at": "2019-02-21 04:06:54",
    "updated_at": "2019-02-21 04:06:54"
  }
}
```

Paginando os produtos

app/Http/Controllers/Api/ProductController.php

```
public function index()
{
    return response()->json($this->product->paginate(3));
}
```

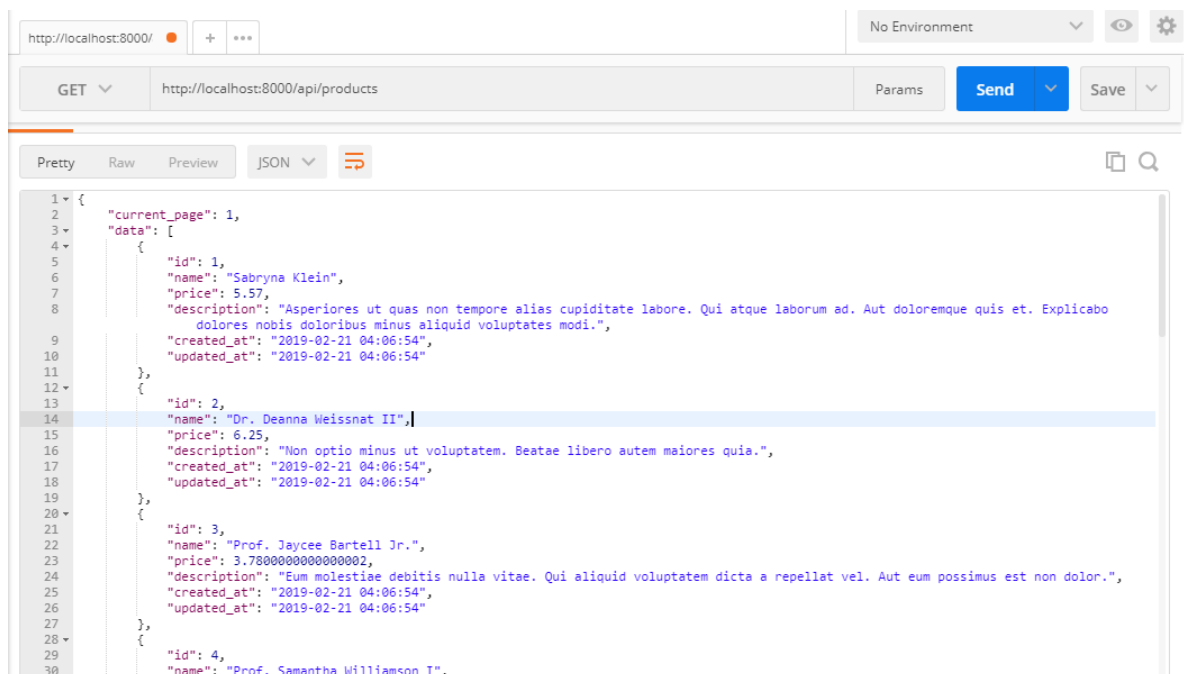
```
{
  "current_page": 1,
  "data": [
    {
      "id": 1,
      "name": "Sabryna Klein",
      "price": 5.57,
      "description": "Asperiores ut quas non tempore alias cupiditate labore. Qui atque laborum ad. Aut doloremque quis et. Explicabo dolores nobis doloribus minus aliquid voluptates modi.",
      "created_at": "2019-02-21 04:06:54",
      "updated_at": "2019-02-21 04:06:54"
    },
    {
      "id": 2,
      "name": "Dr. Deanna Weissnat II",
      "price": 6.25,
      "description": "Non optio minus ut voluptatem. Beatae libero autem maiores quia.",
      "created_at": "2019-02-21 04:06:54",
      "updated_at": "2019-02-21 04:06:54"
    },
    {
      "id": 3,
      "name": "Prof. Jaycee Bartell Jr.",
      "price": 3.7800000000000002,
      "description": "Eum molestiae debitis nulla vitae. Qui aliquid voluptatem dicta a repellat vel. Aut eum possimus est non dolor.",
      "created_at": "2019-02-21 04:06:54",
      "updated_at": "2019-02-21 04:06:54"
    }
  ],
  "first_page_url": "http://localhost:8000/api/products?page=1",
  "from": 1,
  "last_page": 4,
  "last_page_url": "http://localhost:8000/api/products?page=4",
  "next_page_url": "http://localhost:8000/api/products?page=2",
  "path": "http://localhost:8000/api/products",
  "per_page": 3,
  "prev_page_url": null,
  "to": 3,
  "total": 10
}
```

app/Http/Controllers/Api/ProductController.php

```
public function store(Request $request)
{
    dd($request->all());
}
```

Usando o Postman

Listando produtos - GET



Cadastrando produtos - POST

http://localhost:8000/ No Environment

POST http://localhost:8000/api/products Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

form-data x-www-form-urlencoded raw binary

| Key | Value | Description |
|-------------|------------------|-------------|
| name | Product 1 | |
| price | 19.99 | |
| description | Description test | |
| New key | Value | Description |

Body Cookies Headers (5) Test Results Status: 200 OK Time: 1383 ms

Pretty Raw Preview

```
array:3 [
  "name" => "Product 1"
  "price" => "19.99"
  "description" => "Description test"
]
```

app/Http/Controllers/Api/ProductController.php

```
public function store(Request $request)
{
    $productData = $request->all();
    $this->product->create($productData);
}
```

http://localhost:8000/ http://localhost:8000/ No Environment

GET http://localhost:8000/api/products?page=2 Params Send Save

Authorization Headers Body Pre-request Script Tests Code

Type No Auth

Body Cookies Headers (8) Test Results Status: 200 OK Time: 778 ms

Pretty Raw Preview JSON

```
1 {
2   "current_page": 2,
3   "data": [
4     {
5       "id": 11,
6       "name": "Product 1",
7       "price": 19.99,
8       "description": "Description test",
9       "created_at": "2019-02-21 15:08:00",
10      "updated_at": "2019-02-21 15:08:00"
11    }
12  ],
13  "first_page_url": "http://localhost:8000/api/products?page=1",
14  "from": 11,
15  "last_page": 2,
16  "last_page_url": "http://localhost:8000/api/products?page=2",
17  "next_page_url": null,
18  "path": "http://localhost:8000/api/products",
19  "per_page": 10,
20  "prev_page_url": "http://localhost:8000/api/products?page=1",
21  "to": 11,
22  "total": 11
23 }
```

Showing rows 0 - 10 (11 total, Query took 0.0180 seconds.)

`SELECT * FROM `products``

☐ Profiling [E]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

| | | id | name | price | description | created_at | updated_at |
|--------------------------|------------------|----|-----------------------------|-------|---|---------------------|---------------------|
| <input type="checkbox"/> | Edit Copy Delete | 1 | Sabryna Klein | 5.57 | Asperiores ut quas non tempore alias cupiditate la... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit Copy Delete | 2 | Dr. Deanna Weissnat II | 6.25 | Non optio minus ut voluptatem. Beatae libero autem... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit Copy Delete | 3 | Prof. Jaycee Bartell Jr. | 3.78 | Eum molestiae debitis nulla vitae. Qui aliquid vol... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | Edit Copy Delete | 4 | Prof. Samantha Williamson I | 2.33 | Ut ea reprehenderit eos aspernatur cupiditate. Nis... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 5 | Maribel Mueller | 0.02 | Facere voluptatem et sed mollitia eos repellat vol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 6 | Jamey Wintheiser | 5.05 | Quam in nulla rem. Sit soluta omnis et dolorem dol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 7 | Dr. Rose Vandervort MD | 7.43 | Id ab beatae recusandae delectus illum. Enim quo d... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 8 | Jackie Halvorson | 0.47 | Ut dolorem eum consectetur ex. Deserunt nesciunt q... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 9 | Robbie Wintheiser MD | 2.78 | Unde inventore vel corporis laudantium. Voluptates... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 10 | Summer Kerluke | 6.17 | Modi cupiditate sit officis odit et. Omnis molest... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | Edit Copy Delete | 11 | Product 1 | 19.99 | Description test | 2019-02-21 15:08:00 | 2019-02-21 15:08:00 |

☐ Check all | With selected: Edit Copy Delete Export

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

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:8000/api/products`. The request body is a JSON object with the following data:

| KEY | VALUE | DESCRIPTION |
|---|----------------------|-------------|
| <input checked="" type="checkbox"/> name | Product 2 | |
| <input checked="" type="checkbox"/> price | 19999999999999999.99 | |
| <input checked="" type="checkbox"/> description | Description Test | |
| Key | Value | Description |

The response status is 200 OK, but the response body is a JSON object indicating an error:

```
{
  "data": {
    "msg": "SQLSTATE[22003]: Numeric value out of range: 1264 Out of range value for column 'price' at row 1 (SQL: insert into `products` (`name`, `price`, `description`, `updated_at`, `created_at`) values (Product 2, 19999999999999999.99, Description Test, 2018-11-24 20:04:52, 2018-11-24 20:04:52))",
    "code": 1010
  }
}
```

Inserindo um novo produto

The screenshot shows the Swagger UI for a REST API. At the top, the URL is `http://localhost:8000/`. The selected endpoint is `POST http://localhost:8000/api/products`. The request body is a JSON object:

```
{
  "name": "Product2",
  "price": 39.99,
  "description": "Description test"
}
```

The response is displayed in the 'Body' tab, showing a status of `201 Created` and a message: `"Produto criado com sucesso!"`.

Showing rows 0 - 11 (12 total, Query took 0.0000 seconds.)

```
SELECT * FROM `products`
```

☐ Profiling [...]

☐ Show all | Number of rows: | Filter rows: | Sort by key:

+ Options

| | | | id | name | price | description | created_at | updated_at | |
|--------------------------|--|--|----|------|-----------------------------|-------------|---|---------------------|---------------------|
| <input type="checkbox"/> | | | | 1 | Sabrina Klein | 5.57 | Asperiores ut quas non tempore alias cupiditate la... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | | | | 2 | Dr. Deanna Weissnat II | 6.25 | Non optio minus ut voluptatem. Beatae libero autem... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | | | | 3 | Prof. Jaycee Bartell Jr. | 3.78 | Eum molestiae debitis nulla vitae. Qui aliquid vol... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> | | | | 4 | Prof. Samantha Williamson I | 2.33 | Ut ea reprehenderit eos aspernatur cupiditate. Nis... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 5 | Maribel Mueller | 0.02 | Facere voluptatem et sed mollitia eos repellat vol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 6 | Jamey Wintheiser | 5.05 | Quam in nulla rem. Sit soluta omnis et dolore dol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 7 | Dr. Rose Vandervort MD | 7.43 | Id ab beatae recusandae delectus illum. Enim quo d... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 8 | Jackie Halvorson | 0.47 | Ut dolore eum consectetur ex. Deserunt nesciunt q... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 9 | Robbie Wintheiser MD | 2.78 | Unde inventore vel corporis laudantium. Voluptates... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 10 | Summer Kerluke | 6.17 | Modi cupiditate sit officis odit et. Omnis molest... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> | | | | 11 | Product 1 | 19.99 | Description test | 2019-02-21 15:08:00 | 2019-02-21 15:08:00 |
| <input type="checkbox"/> | | | | 12 | Product2 | 39.99 | Description test | 2019-02-21 16:01:41 | 2019-02-21 16:01:41 |

☐ Check all | With selected: Edit Copy Delete Export

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

100 Continue

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

Esse código é enviado em resposta a um cabeçalho de solicitação **Upgrade** pelo cliente, e indica o protocolo a que o servidor está alternando.

102 Processing (**WebDAV**)

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

Respostas de sucesso Seção

200 OK

Esta 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 transmitido no corpo da mensagem.

TRACE: O corpo da mensagem contém a mensagem de requisição recebida pelo servidor

201 Created

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

202 Accepted

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

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

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

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

206 Partial Content

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

207 Multi-Status (WebDAV)

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

208 Multi-Status (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 (HTTP Delta encoding)

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

300 Multiple Choice

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

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

302 Found

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

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

304 Not Modified

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

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

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

400 Bad Request

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

401 Unauthorized

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

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

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

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

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

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

408 Request Timeout

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

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

410 Gone

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

411 Length Required

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

412 Precondition Failed

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

413 Payload Too Large

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

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

415 Unsupported Media Type

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

416 Requested Range Not Satisfiable

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

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

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

421 Misdirected Request

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 incluídas na URI da requisição.

422 Unprocessable Entity (WebDAV)

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

423 Locked (WebDAV)

O recurso sendo acessado está chaveado.

424 Failed Dependency (WebDAV)

A requisição falhou devido a falha em requisição prévia.

426 Upgrade Required

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** numa resposta 426 para indicar o(s) protocolo(s) requeridos.

428 Precondition Required

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

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

431 Request Header Fields Too Large

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

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

Respostas de erro do ServidorSeção

500 Internal Server Error

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

501 Not Implemented

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

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

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 possível, conter o tempo estimado para recuperação do serviço. O webmaster deve

também tomar cuidado com os cabeçalhos 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

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

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

506 Variant Also Negotiates

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

507 Insufficient Storage

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 (WebDAV)

O servidor detectou um looping infinito ao processar a requisição.

510 Not Extended

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

511 Network Authentication Required

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);
    }
}
```

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:8000/api/products/11
- Method:** PUT
- Status:** 201 Created
- Time:** 1053 ms
- Response Body (JSON):**

```
{  \"data\": {    \"msg\": \"Produto atualizado com sucesso!\"  }}
```

| Key | Value | Description |
|---|--------------------|-------------|
| <input checked="" type="checkbox"/> name | Product 1 - Edited | |
| <input checked="" type="checkbox"/> price | 19.99 | |
| <input checked="" type="checkbox"/> description | Description test | |
| New key | Value | Description |

PUT ▼ http://localhost:8000/api/products/1 Params Send ▼ Save ▼

| Key | Value | Description | ... | Bulk Edit |
|---|---------------------------|-------------|-----|-----------|
| <input checked="" type="checkbox"/> name | Produto 1 também alterado | | | |
| <input checked="" type="checkbox"/> price | 49.99 | | | |
| <input checked="" type="checkbox"/> description | Descrição do produto 1 | | | |
| New key | Value | Description | | |

Body Cookies Headers (8) Test Results Status: 201 Created Time: 1037 ms

Pretty Raw Preview JSON ▼ ≡

```

1  {
2    "data": {
3      "msg": "Produto atualizado com sucesso!"
4    }
5  }

```

Showing rows 0 - 11 (12 total. Query took 0.0010 seconds.)

[SELECT * FROM `products`](#) Profiling [\[Edit inline\]](#) [\[Edit\]](#) [\[Explain SQL\]](#) [\[Create PHP code\]](#) [\[Refresh\]](#)

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

| | id | name | price | description | created_at | updated_at |
|---|----|-----------------------------|-------|---|---------------------|---------------------|
| <input type="checkbox"/> Edit Copy Delete | 1 | Produto 1 também alterado | 49.99 | Descrição do produto 1 | 2019-02-21 04:06:54 | 2019-02-21 16:36:50 |
| <input type="checkbox"/> Edit Copy Delete | 2 | Dr. Deanna Weissnat II | 6.25 | Non optio minus ut voluptatem. Beatae libero autem... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> Edit Copy Delete | 3 | Prof. Jaycee Bartell Jr. | 3.78 | Eum molestiae debitis nulla vitae. Qui aliquid vol... | 2019-02-21 04:06:54 | 2019-02-21 04:06:54 |
| <input type="checkbox"/> Edit Copy Delete | 4 | Prof. Samantha Williamson I | 2.33 | Ut ea reprehenderit eos aspernatur cupiditate. Nis... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 5 | Maribel Mueller | 0.02 | Facere voluptatem et sed mollitia eos repellat vol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 6 | Jamey Wintheiser | 5.05 | Quam in nulla rem. Sit soluta omnis et dolore dol... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 7 | Dr. Rose Vandervort MD | 7.43 | Id ab beatae recusandae delectus illum. Enim quo d... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 8 | Jackie Halvorson | 0.47 | Ut dolore eum consectetur ex. Deserunt nesciunt q... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 9 | Robbie Wintheiser MD | 2.78 | Unde inventore vel corporis laudantium. Voluptates... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 10 | Summer Kerluke | 6.17 | Modi cupiditate sit officis odit et. Omnis molest... | 2019-02-21 04:06:55 | 2019-02-21 04:06:55 |
| <input type="checkbox"/> Edit Copy Delete | 11 | Product 1 - Edited | 19.99 | Description test | 2019-02-21 15:08:00 | 2019-02-21 16:33:41 |
| <input type="checkbox"/> Edit Copy Delete | 12 | Product2 | 39.99 | Description test | 2019-02-21 16:01:41 | 2019-02-21 16:01:41 |

[↑](#) ☐ Check all | With selected: [Edit](#) [Copy](#) [Delete](#) [Export](#)

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);
    }
}

```

http://localhost:8000/

http://localhost:8000/

http://localhost:8000/

+

...

No Environment

⌵

👁

⚙

DELETE

http://localhost:8000/api/products/12

Params

Send

Save

Authorization

Headers

Body

Pre-request Script

Tests

Code

Type

No Auth

Body

Cookies

Headers (8)

Test Results

Status: 200 OK

Time: 954 ms

Pretty

Raw

Preview

JSON

≡

📄

🔍

```

1 {
2   "data": {
3     "msg": "Produto: Product2 removido com sucesso!"
4   }
5 }

```

Showing rows 0 - 10 (11 total, Query took 0.0000 seconds.)

SELECT * FROM `products`

Profiling [E]

Show all

Number of rows: 25

Filter rows: Search this table


































Sort by key: None

+ Options

⬅

➡


⌵


| | id | name | price | description | created_at | updated_at |
|---|----|-----------------------------|-------|---|---------------------|---------------------|
|    | 1 | Produto 1 também alterado | 49.99 | Descrição do produto 1 | 2019-02-21 04:08:54 | 2019-02-21 16:38:50 |
|    | 2 | Dr. Deanna Weissnat II | 6.25 | Non optio minus ut voluptatem. Beatae libero autem... | 2019-02-21 04:08:54 | 2019-02-21 04:08:54 |
|    | 3 | Prof. Jaycee Bartell Jr. | 3.78 | Eum molestiae debitis nulla vitae. Qui aliquid vol... | 2019-02-21 04:08:54 | 2019-02-21 04:08:54 |
|    | 4 | Prof. Samantha Williamson I | 2.33 | Ut ea reprehenderit eos aspernatur cupiditate. Nis... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 5 | Manibel Mueller | 0.02 | Facere voluptatem et sed mollitia eos repellat vol... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 6 | Jamey Wintheiser | 5.05 | Quam in nulla rem. Sit soluta omnis et dolore dol... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 7 | Dr. Rose Vandervort MD | 7.43 | Id ab beatae recusandae delectus illum. Enim quo d... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 8 | Jackie Halvorson | 0.47 | Ut dolore eum consectetur ex. Deserunt nesciunt q... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 9 | Robbie Wintheiser MD | 2.78 | Unde inventore vel corporis laudantium. Voluptates... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 10 | Summer Kerluke | 6.17 | Modi cupiditate sit officii odit et. Omnis molest... | 2019-02-21 04:08:55 | 2019-02-21 04:08:55 |
|    | 11 | Product 1 - Edited | 19.99 | Description test | 2019-02-21 15:08:00 | 2019-02-21 16:33:41 |


⬅


Check all

With selected:









Busca de produto com mensagem de erro

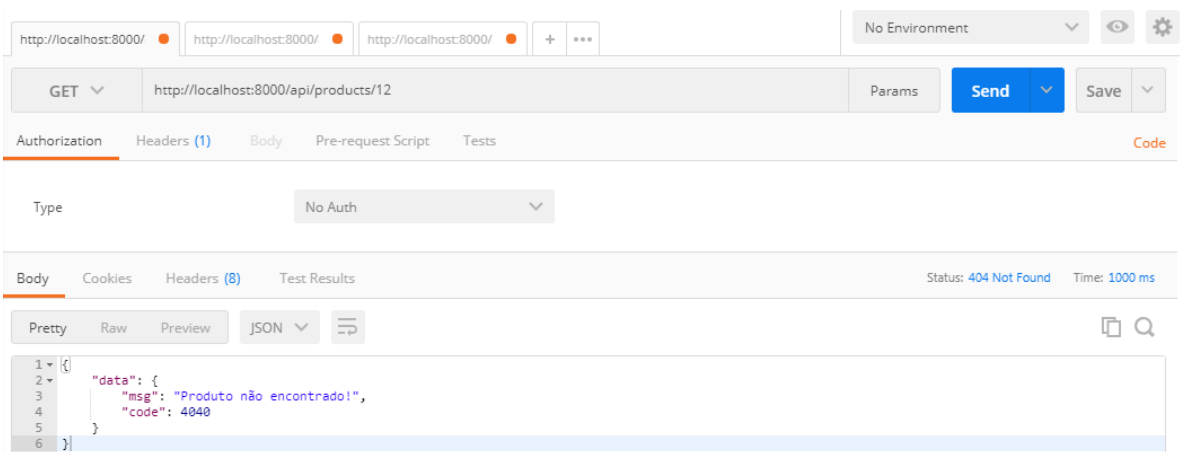
app/Http/Controllers/Api/ProductController.php

```
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



The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8000/api/products/12`
- Method: `GET`
- Authorization: `No Auth`
- Body: `JSON` (Pretty view)
- Status: `404 Not Found`
- Time: `1000 ms`
- Response Body (JSON):

```
{  "data": {    "msg": "Produto não encontrado!",    "code": 4040  }}
```

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);
    }
}
```