

Curso Técnico Superior Profissional em: Tecnologias e Programação de Sistemas de Informação

2º Ano/1º Semestre

Unidade Curricular: Aplicações Centradas em Redes

Docente: Michael Silva / Hugo Perdigão

## INTRODUÇÃO ÀS LINGUAGENS PARA O DESENVOLVIMENTO DE APLICAÇÕES CENTRADAS EM REDES

# Laravel – Exemplo (continuação)

### Objetivo:

- Criar mensagens de erro customizadas
- Utilizar SASS
- Criar relações em tabelas

1) Criar um **Request** para o formulário:

```
TERMINAL  PROBLEMAS  1  SAÍDA  CONSOLE DE DEPURAÇÃO

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\xampp\htdocs\loja_info> php artisan make:request NewProductRequest
Request created successfully.
PS C:\xampp\htdocs\loja_info> |
```

2) Passar as regras de validação para o novo **Request**

- a. Alterar o tipo de dados recebido no método store para o novo **Request** e eliminar as regras de validação.

```
30 public function store(NewProductRequest $request)
31 {
32
33     $name = request('name');
```

Cofinanciado por:

### b. Colocar as regras de validação na class **NewProductRequest**

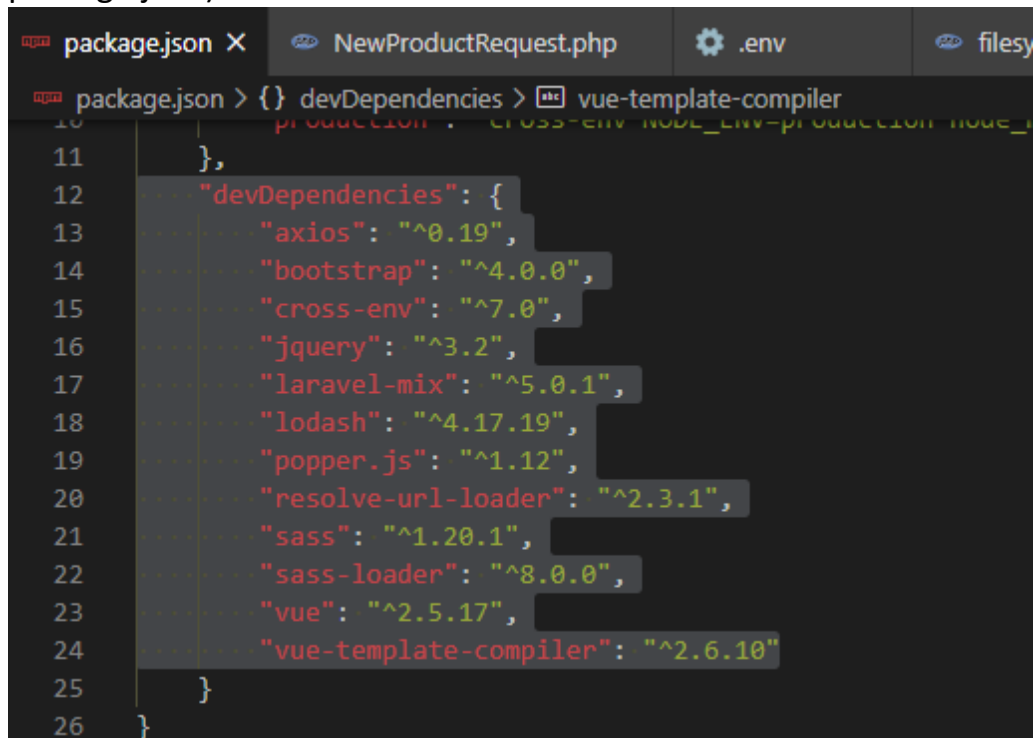
```
NewProductRequest.php X .env filesystems.php ProdutosController.php
app > Http > Requests > NewProductRequest.php > NewProductRequest > rules
1  <?php
2
3  namespace App\Http\Requests;
4
5  use Illuminate\Foundation\Http\FormRequest;
6
7  class NewProductRequest extends FormRequest
8  {
9      /**
10       * Determine if the user is authorized to make this request.
11       *
12       * @return bool
13       */
14     public function authorize()
15     {
16         return true;
17     }
18
19     /**
20      * Get the validation rules that apply to the request.
21      *
22      * @return array
23      */
24     public function rules()
25     {
26         return [
27             'name' => 'required',
28             'url' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048'
29         ];
30     }
31 }
```

- 3) Para customizar as regras de validação, na class **NewProductController** criar uma função **messages()** com as mensagens para cada validação:

```
NewProductRequest.php X .env filesystems.php ProdutosController.php
app > Http > Requests > NewProductRequest.php > NewProductRequest
19
20     * Get the validation rules that apply to the request.
21     *
22     * @return array
23     */
24     public function rules()
25     {
26         return [
27             'name' => 'required',
28             'url' => 'required|image|mimes:jpeg,png,jpg,gif|max:2048'
29         ];
30     }
31
32     public function messages()
33     {
34         return [
35             'name.required' => 'O Nome do Produto é Obrigatório.',
36             'url.required' => 'A Imagem é Obrigatória',
37             'url.image' => 'A Imagem deve ser uma imagem.',
38             'url.mimes' => 'A Imagem pode ser jpeg,png,jpg,gif.',
39             'url.max' => 'A Imagem Não pode exceder 2MB.',
40         ];
41     }
42 }
43
44
```

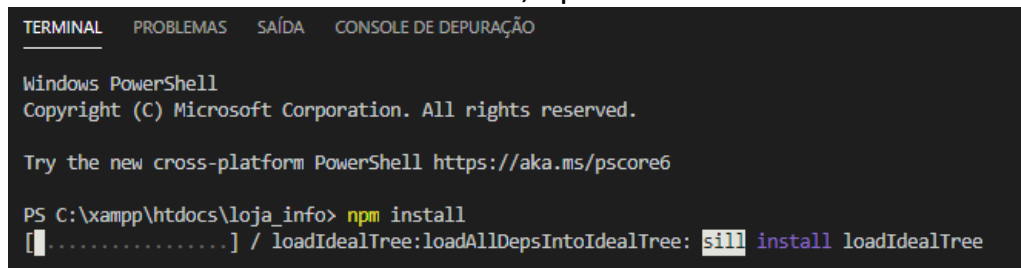
Cofinanciado por:

- 4) O Laravel já vem preparado para a utilização de SASS, sendo apenas necessário instalar alguns pacotes, que já vêm configurados (ver package.json)



```
package.json X NewProductRequest.php .env filesy
package.json > {} devDependencies > vue-template-compiler
10 production: cross-env NODE_ENV=production node _
11 },
12 "devDependencies": {
13   "axios": "^0.19",
14   "bootstrap": "^4.0.0",
15   "cross-env": "^7.0",
16   "jquery": "^3.2",
17   "laravel-mix": "^5.0.1",
18   "lodash": "^4.17.19",
19   "popper.js": "^1.12",
20   "resolve-url-loader": "^2.3.1",
21   "sass": "^1.20.1",
22   "sass-loader": "^8.0.0",
23   "vue": "^2.5.17",
24   "vue-template-compiler": "^2.6.10"
25 }
26 }
```

- 5) Para instalar executar no terminal, npm install



```
TERMINAL PROBLEMAS SAÍDA CONSOLE DE DEPURAÇÃO
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

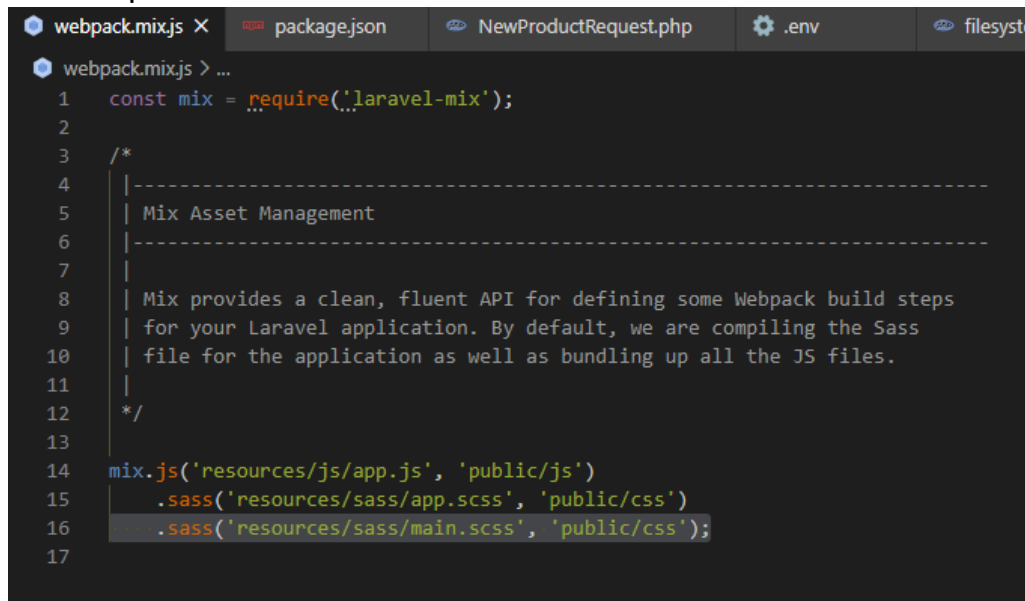
Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\xampp\htdocs\loja_info> npm install
[.....] / loadIdealTree:loadAllDepsIntoIdealTree: sill install loadIdealTree
```

Cofinanciado por:

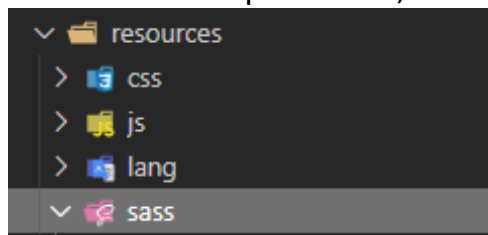


- 6) Para especificar os recursos SASS, abrir o ficheiro **webmix.js** e acrescentar a linha com o ficheiro a utilizar, indicando onde ele está e para onde deve ser compilado:

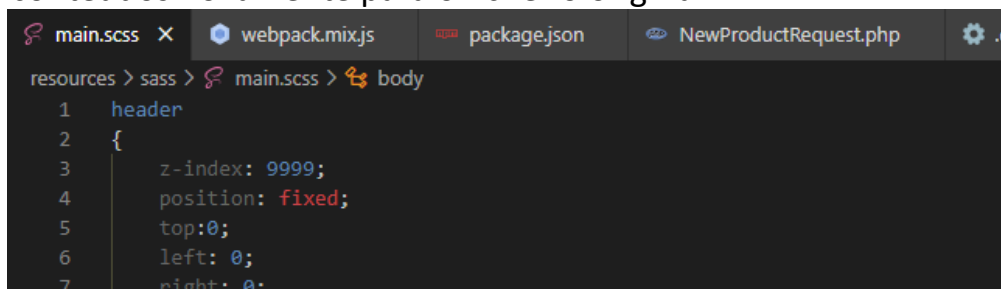


```
webpack.mix.js > ...
1  const mix = require('laravel-mix');
2
3  /*
4  |-----
5  | Mix Asset Management
6  |-----
7  |
8  | Mix provides a clean, fluent API for defining some Webpack build steps
9  | for your Laravel application. By default, we are compiling the Sass
10 | file for the application as well as bundling up all the JS files.
11 |
12 | */
13
14 mix.js('resources/js/app.js', 'public/js')
15   .sass('resources/sass/app.scss', 'public/css')
16   .sass('resources/sass/main.scss', 'public/css');
17
```

- 7) Se não existir a pasta sass, deve ser criada dentro da pasta resources.



- 8) O ficheiro main.scss tem que ser criado. Para garantir que está tudo a funcionar, cortar os conteúdos da css existente em /public/css (deixar o ficheiro vazio, e colar no novo ficheiro, ao compilar deverá passar os conteúdos novamente para o ficheiro original.



```
main.scss
resources > sass > main.scss > body
1  header
2  {
3      z-index: 9999;
4      position: fixed;
5      top: 0;
6      left: 0;
7      right: 0;
```

Cofinanciado por:

9) Executar o comando **npm run dev** para compilar os recursos.

```
TERMINAL  PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO

PS C:\xampp\htdocs\loja_info> npm run dev

> @ dev C:\xampp\htdocs\loja_info
> npm run development

> @ development C:\xampp\htdocs\loja_info
> cross-env NODE_ENV=development node_modules/webpack/bin/webpack.js --progress --config=not...

98% after emitting SizeLimitsPlugin

DONE Compiled successfully in 14649ms

Asset      Size  Chunks  Chunk Names
/css/app.css  178 KiB  /js/app  [emitted]  /js/app
/css/main.css  998 bytes  /js/app  [emitted]  /js/app
/js/app.js    1.41 MiB  /js/app  [emitted]  /js/app
PS C:\xampp\htdocs\loja_info>
```

**NOTA:** verificar de o ficheiro de destino foi criado/preenchido.

10) Para criar tabelas com relações, é necessário ter pelo menos duas tabelas.

a. Criar a tabela `tipo_produto`

i. Criar a migration correspondente

`php artisan make:migration create_tipo_produto_table`

```
TERMINAL  PROBLEMAS  SAÍDA  CONSOLE DE DEPURACÃO

DONE Compiled successfully in 14649ms

Asset      Size  Chunks  Chunk Names
/css/app.css  178 KiB  /js/app  [emitted]  /js/app
/css/main.css  998 bytes  /js/app  [emitted]  /js/app
/js/app.js    1.41 MiB  /js/app  [emitted]  /js/app
PS C:\xampp\htdocs\loja_info> php artisan make:migration create_tipo_produto_table
Created Migration: 2020_11_20_112614_create_tipo_produto_table
PS C:\xampp\htdocs\loja_info>
```

ii. Criar as colunas na migration

```
main.scss  2020_11_20_112614_create_tipo_produto_table.php  webpack.mix.js  package.json
database > migrations > 2020_11_20_112614_create_tipo_produto_table.php > CreateTipoProdutoTable > up >
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class CreateTipoProdutoTable extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::create('tipo_produto', function (Blueprint $table) {
17             $table->id();
18             $table->timestamps();
19             $table->string('nome');
20         });
21     }
22 }
```

Cofinanciado por:



b. Alterar a tabela dos produtos para incluir uma nova coluna e criar a chave estrangeira.

i. Criar uma migration para adicionar a nova coluna e chave estrangeira à tabela de

php artisan make:migration add\_fk\_products\_tipo\_produto

```
PS C:\xampp\htdocs\loja_info> php artisan make:migration create_tipo_produto_table
Created Migration: 2020_11_20_150700_create_tipo_produto_table
PS C:\xampp\htdocs\loja_info> php artisan make:migration add_fk_products_tipo_produto
Created Migration: 2020_11_20_151641_add_fk_products_tipo_produto
```

ii. Na nova migration, ir buscar a estrutura da tabela e adicionar a coluna e chave estrangeira

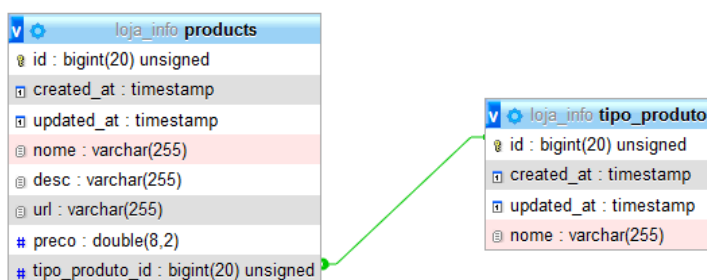
```
main.scss 2020_11_20_230455_add_fk_products_tipo_produto.php X 2020_11_20_230300_create_tipo_produto.php
database > migrations > 2020_11_20_230455_add_fk_products_tipo_produto.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  class AddFkProductsTipoProduto extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14     public function up()
15     {
16         Schema::table('products', function (Blueprint $table)
17         {
18             $table->bigInteger('tipo_produto_id')->unsigned();
19             $table->foreign('tipo_produto_id')->references('id')->on('tipo_produto');
20         });
21     }
```

iii. Executar a migração

```
PS C:\xampp\htdocs\loja_info> php artisan migrate
Migrating: 2020_11_20_150700_create_tipo_produto_table
Migrated: 2020_11_20_150700_create_tipo_produto_table (28.24ms)
Migrating: 2020_11_20_151641_add_fk_products_tipo_produto
Migrated: 2020_11_20_151641_add_fk_products_tipo_produto (0.12ms)
PS C:\xampp\htdocs\loja_info> |
```

**NOTA:** apenas as duas novas migrations devem executar

iv. Verificar no phpMyAdmin que as tabelas foram criadas corretamente com as relações especificadas



Cofinanciado por:

- c. O próximo passo é criar o modelo para a nova tabela.  
php artisan make:model tipo\_produto

```
PS C:\xampp\htdocs\loja_info> php artisan make:model tipo_produto
Model created successfully.
PS C:\xampp\htdocs\loja_info> 
```

**NOTA:** por defeito ao criar um model, o laravel assume que o nome da tabela é o **plural** do nome escolhido para o modelo e faz a ligação automaticamente, neste caso como o nome do modelo é diferente do predefinido é necessário especificar a tabela no modelo:

```
tipo_produto.php X TipoProdutoController.php index.blade.php
app > Models > tipo_produto.php > tipo_produto
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class tipo_produto extends Model
9  {
10     use HasFactory;
11
12     protected $table = 'tipo_produto';
13 }
14
```

- 11) Como um produto tem agora de ter um tipo de produto associado, acrescentar uma caixa de seleção para escolher o produto.
- a. Primeiro, no controller, além de passar os produtos é necessário também selecionar os novos tipos de produto da base de dados utilizando o model tipo\_produto.

```
26     public function create()
27     {
28         $tipos = tipo_produto::all();
29     }
```

E envia-lo para a veiw:

```
30     return view('createProduct', [ 'tipos' => $tipos]);
31 }
32
```

Cofinanciado por:

- b. Na view acrescentar um select que vai listar os tipos de produtos para seleccionar um para atribuir ao produto

```

24         <br>
25         <label for="price">preço:</label>
26         <input type="number" id="price" name="price">
27         <br>
28         <label for="tipoProduto">Tipo de Produto:</label>
29         <select name="tipoProduto" id="tipoProduto">
30             @foreach ($tipos as $tipo)
31                 <option value="{{ $tipo->id }}">{{ $tipo->nome }}</option>
32             @endforeach
33         </select>
34         <br>
35         <input type="submit" value="Criar Produto">
36     </form>
37     <a href="/produtos">voltar aos produtos</a>
38 </div>

```

- c. Agora para poder adicionar produtos, é necessário que existam primeiro tipos de produto. Para isso podem ser adicionados manualmente na base de dados alguns tipos:

✓ A mostrar registos de 0 - 2 (3 total, A consulta demorou 0,0007 segundos.)

`SELECT * FROM `tipo_produto``

☐ Mostrar tudo | Número de registos: 25 | Filtrar registos:

+ Opções

			id	created_at	updated_at	nome	
<input type="checkbox"/>	Editar	Copiar	Apagar	1	NULL	NULL	Computador
<input type="checkbox"/>	Editar	Copiar	Apagar	2	NULL	NULL	Portátil
<input type="checkbox"/>	Editar	Copiar	Apagar	3	NULL	NULL	Acessórios

- d. O último passo para poder adicionar produtos é atribuir o tipo seleccionado ao novo produto antes de este ser guardado na base de dados:

- i. Recebendo a variável do formulário

```

ProdutosController.php X createProduct.blade.php tipo_produto.php TipoProdutoController.p
app > Http > Controllers > ProdutosController.php > ProdutosController > store
30     return view('createProduct',[ 'tipos' => $tipos]);
31 }
32
33 public function store(NewProductRequest $request)
34 {
35     $name = request('name');
36     $desc = request('desc');
37     $price = request('price');
38     $tipo = request('tipoProduto');
39     //tratamento da imagem
40     $url = "";
41     if ($request->has('url'))

```



- ii. E atribuindo o tipo obtido ao novo produto antes de gravar na base de dados:

```
53
54     $produto = new Product();
55
56     $produto->nome = $name;
57     $produto->desc = $desc;
58     $produto->url = $url;
59     $produto->preco = $price;
60     $produto->tipo_produto_id = $tipo;
61
62     $produto->save();
63
```

NOTA: os valores que já estavam gravados na base de dados não têm o atributo para o tipo de produto pelo que este deve ser adicionado manualmente:

✓ A mostrar registos de 0 - 3 (4 total, A consulta demorou 0,0007 segundos.)

SELECT \* FROM `products`

☐ Mostrar tudo | Número de registos: 25 | Filtrar registos: Pesquisar esta tabela | Ordenar pela chave: Nenhum

+ Opções

				id	created_at	updated_at	nome	desc	url	preco	tipo_produto_id
<input type="checkbox"/>	✎	✂	✖	3	NULL	NULL	PC Gamer	PC Gamer Top	/img/produtos/1.jpg	2000.0	1
<input type="checkbox"/>	✎	✂	✖	4	NULL	NULL	PC Workstation	PC para trabalho	/img/produtos/2.jpg	1000.0	1
<input type="checkbox"/>	✎	✂	✖	5	NULL	NULL	PC Bonzinho	PC Bonzinho, desenrasca	/img/produtos/3.jpg	800.0	1
<input type="checkbox"/>	✎	✂	✖	6	NULL	NULL	PC Rasca	Muito fraco mesmo	/img/produtos/4.jpg	25.0	1

12) Além de criar a relação na base de dados, também é possível especificar essa relação nos modelos, o que irá facilitar as pesquisas:

- a. Indicar o tipo de relação no modelo, isto tem a ver com a necessidade de utilização, por exemplo neste caso existem vários tipos de produtos, sendo o objetivo obter os produtos de cada tipo, então deve existir uma relação de 1 para Muitos no tipo de produto, especificando que um tipo tem vários (has many) produtos:

Cofinanciado por:

i. No model tipo\_produto, criar o método products

```
tipo_produto.php X Product.php ProdutosController.php produtos.blade.php
app > Models > tipo_produto.php > tipo_produto
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class tipo_produto extends Model
9  {
10     use HasFactory;
11
12     protected $table = 'tipo_produto';
13
14     public function products()
15     {
16         return $this->hasMany('App\Models\Product', 'tipo_produto_id', 'id');
17     }
18 }
```

ii. Criar uma nova Route para aceder aos produtos por tipo de produto:

```
10
11 Route::get('/produtos', [ProdutosController::class, 'index'])->name('products.index');
12 Route::get('/produtos/tipo/{id}', [ProdutosController::class, 'produtosPorTipo'])->name('products.by.tipo');
13 Route::get('/produtos/create', [ProdutosController::class, 'create'])->name('products.create')->middleware('auth');
14 Route::post('/produtos', [ProdutosController::class, 'store'])->name('products.store')->middleware('auth');
```

iii. No controller criar o método correspondente

```
19
20 public function produtosPorTipo($id)
21 {
22     $tipos = tipo_produto::all();
23     $tipo = tipo_produto::findOrFail($id);
24     $produtos = $tipo->products;
25     //$produtos = tipo_produto::find($id)->products()->get();
26
27     return view('produtos', ['produtos' => $produtos, 'tipos' => $tipos, 'actTipo' => $id]);
28 }
```

Cofinanciado por:



- iv. Apesar de ser a mesma view, esta tem de ser alterada para interpretar os novos dados

```
produtos.blade.php X ProdutosController.php tipo_produto.php Product.php createPro...
resources > views > produtos.blade.php > ...
1  @extends('layouts.app')
2
3  @section('content')
4      <h1>Loja de Informática - Produtos</h1>
5      <div class="listaTipos">
6          @if(!isset($actTipo))
7              <b>
8              @endif
9              <a href="{{ route('products.index') }}">Todos os Produtos</a>
10             @if(isset($actTipo))
11             </b>
12             @endif
13             @foreach ($tipos as $tipo)
14                 @if(isset($actTipo) && $actTipo == $tipo->id)
15                     <b>
16                     @endif
17                     <a href="{{ route('products.by.tipo',$tipo->id) }}">{{ $tipo->nome }}</a>
18                 @if(isset($actTipo) && $actTipo == $tipo->id)
19                 </b>
20                 @endif
21             @endforeach
22         </div>
23
24         @foreach ($produtos as $produto)
25             <div class="produto">
26                 <a href="{{ route('products.show',$produto->id) }}">
27                     
28                     <h2>{{ $produto->nome }}</h2>
29                 </a>
30             </div>
31         @endforeach
32     @endsection
```

Bom Trabalho!

Cofinanciado por:

