**Sumário**

[**Criando uma aplicação Laravel Passo a Passo** 1](#_Toc22928401)

[**Rotas** 2](#_Toc22928402)

[**Criar Model, Controller e Migration** 3](#_Toc22928403)

[**Criar tabela através da Migration** 4](#_Toc22928404)

[**Relacionamento entre tabelas** 5](#_Toc22928405)

[**Controller Resource** 6](#_Toc22928406)

[**Views : parte visual do sistema** 9](#_Toc22928407)

[**Configurar banco de dados** 12](#_Toc22928408)

## **Criando uma aplicação Laravel Passo a Passo**

Baixe e instale o composer se não tiver ele instalado:

• Acesse: <http://getcomposer.org/download>

Baixe o laravel através do composer com o seguinte comando no terminal ou CMD.

**composer create-project laravel/laravel --prefer-dist**

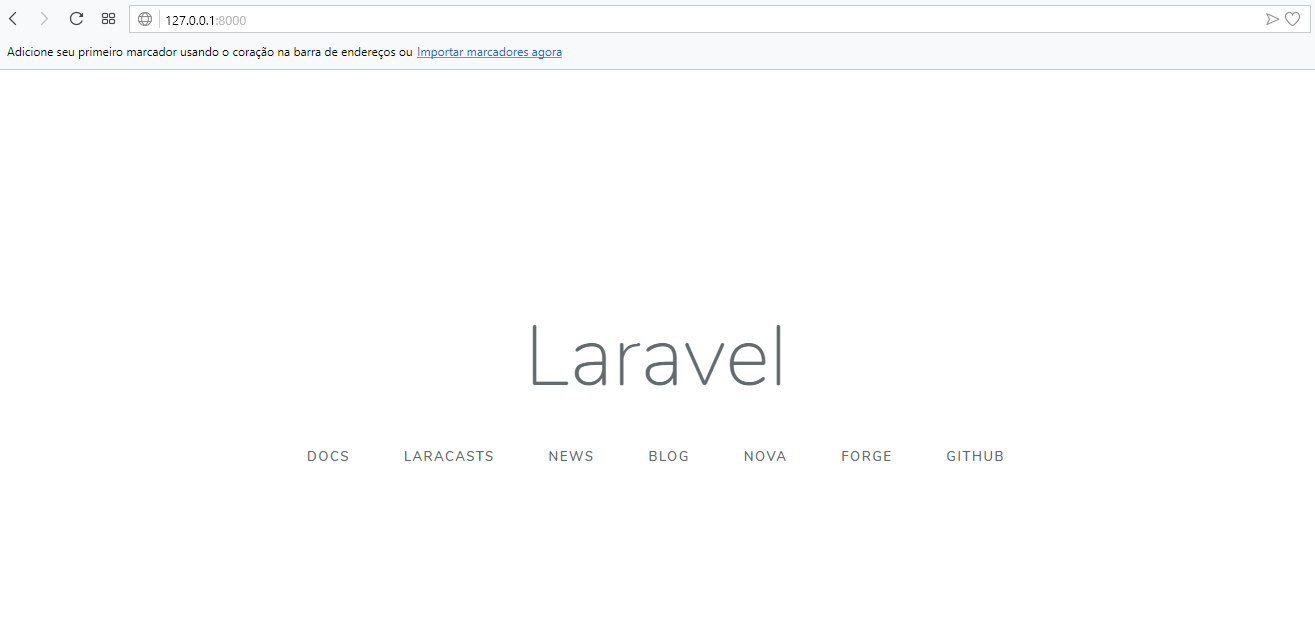
para iniciar a aplicação use:

**cd laravel**

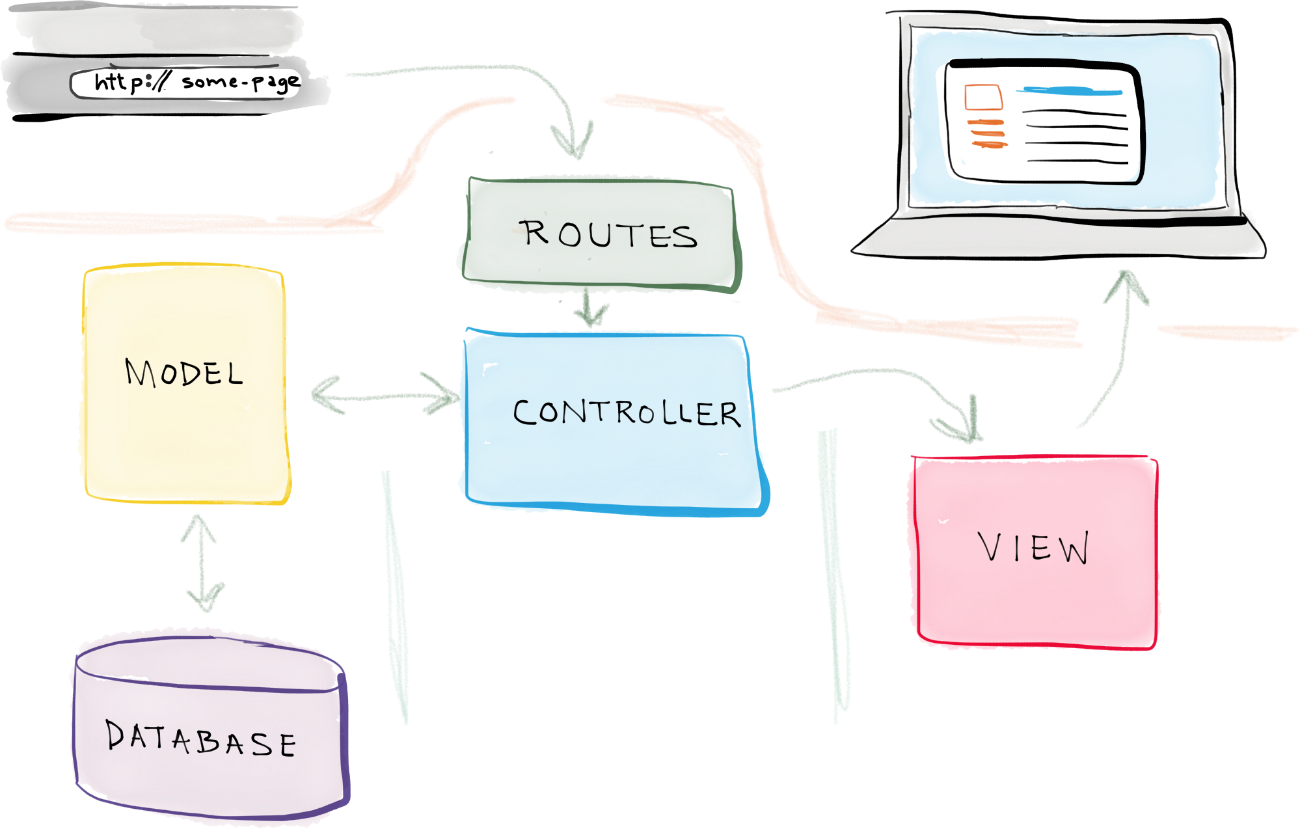
dê enter e digite

**php artisan serve**

Abra no navegador o link abaixo e a pagina inicial será carregada:

**localhost:8000 **

**O ciclo de vida de uma aplicação Laravel funciona da seguinte forma:**

****

## **Rotas**

Para configurar as rotas do site, acesse a pasta e o arquivo:

Routes/web.php

É possível criar rotas de vários tipos

Route::get('produtos/create','ProdutosController@create');

Tipo da Rota URL amigável Nome do Controller Método do Controller

É uma rota com método GET que leva qualquer requisição no navegador para a URL: localhost:8000/produtos/create para ser tratada pelo controller chamado ProjetoController que deve possuir o método create.

Vamos criar uma rota de Resource( ou seja uma rota padrão para acessar, criar, editar, excluir, etc):

Route::resource('Projeto', 'ProjetoController');

Essa rota tem o funcionamento semelhante ao seguinte conjunto de rotas:

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

Route::get('/','ProjetoController@index');

Route::get('/create','ProjetoController@create');

Route::post('/','ProjetoController@store');

Route::get('/{id}','ProjetoController@show');

Route::get('/{id}/edit','ProjetoController@edit');

Route::put('/{id}','ProjetoController@update');

Route::delete('/{id}','ProjetoController@destroy');

});

Ambos os códigos tem o mesmo resultado, ao tentar acessar ou enviar alguma informação da forma a seguir, é acessado o seguinte método do ProjetoController:

GET Localhost:8000/Projeto Método Index

GET Localhost:8000/Projeto/create Método Create

POST Localhost:8000/Projeto Método Store

GET Localhost:8000/Projeto/1 Método Show com o id = 1

GET Localhost:8000/Projeto/1/edit Método Edit com id=1

PUT Localhost:8000/Projeto/1 Método Update com id=1

DELETE Localhost:8000/Projeto/1 Método destroy com id=1

## **Criar Model, Controller e Migration**

Após definirmos as rotas, vamos criar os controllers, as models e as migrations. Execute o seguinte comando no terminal

php artisan make:model Projeto –mcr

Esse comando irá criar uma **Model** chamada Projeto.php no diretório

APP/Projeto.php

Também vai criar um **Controller** chamado ProjetoController no diretório

APP/Controllers/ProjetoController.php

E por último mas não menos importante, vai gerar também a seguinte **Migration** para criar a tabela no banco de dados.

CreateProjetosTable no diretório DATABASE/Migrations/ 2019\_10\_01\_013152\_create\_projetos\_table.php

## **Criar tabela através da Migration**

Primeiro vamos criar a tabela através da Migration create\_projetos\_table

class CreateProjetosTable extends Migration {

public function up() {

Schema::create('projetos', function (Blueprint $table) {

$table->bigIncrements('id');

$table->timestamps

$table->string('nome');

$table->string('lider');

$table->string('ano');

});

}

public function down()

{

Schema::dropIfExists('projetos');

}

Essa migration Possui dois métodos: o método UP() e o método DOWN().

O Método UP é o que vai criar a tabela, chamada de ‘projetos’ e que terá os seguintes atributos:

Campo ID do tipo BIG INT autoIncrement (atualiza o índice automaticamente)

Campo de data de criação e data de alteração

Compo String para nome

Campo String para líder

Campo String para ano

O método DOWN da migration será responsável por Apagar a tabela, ou seja desfazer a criação da tabela.

## **Relacionamento entre tabelas**

Caso precise fazer o relacionamento entre tabelas, pode ser feito da seguinte forma na migration.

$table->foreign('user\_id')->references('id')->on('users');

Em que a chave estrangeira chamada ‘user\_id’ vai fazer relação com a chave primaria chamada ‘id’ da tabela chamada ‘users’.

E na model deve ser criada uma função para retornar os valores do relacionamento.

Quando o relacionamento for de muitos para um n:1

public function user()

{

return $this->belongsTo('App\User');

}

E dessa forma Quando o relacionamento for de um para muitos 1:n

public function Medicamentos()

{

return $this->hasMany('App\sol\_medicamento');

}

Após fazer essas funções, será possível usar na view dessa forma:

{{$solicitacao->paciente->telefone}}

{{$solicitacao->user->name}}

## **Controller Resource**

O controller chamado ProjetoController vai possuir os seguintes métodos

public function **index**() { }

public function **create**() { }

public function **store**(Request $request) { }

public function **show**(Projeto $projeto) { }

public function **edit**(Projeto $projeto) { }

public function **update**(Request $request, Projeto $projeto) { }

public function **destroy**(Projeto $projeto) { }

onde o **index** deve ser usado para exibir todos os valores do banco de dados, o **create** deve ser usado para mostrar a View de Cadastro de dados, o método **Store** deve receber os dados e gravar no banco de dados, o método **Show** deve exibir informações de um registro especifico, o método **edit** deve ser usado para mostrar a View de Editar os dados,

o método **update** deve ser usado para receber os dados e atualizar no banco e o método **destroy** deve ser usado para excluir um registro do banco de dados.

Eles podem ser implementados da seguinte forma:

public function index()

{

$projetos = Projeto::all();

return view('projeto\_index',['projetos' => $projetos]);

}

usa o método **all** da model **Projeto** para retornar todos os valores, e envia esses valores para serem exibidos na View chamada de ‘projeto\_index’

public function create()

{

return view('cadastro\_projeto');

}

Retorna uma View para cadastrar os dados, chamada de ‘cadastro\_projeto’

public function store(Request $request)

{

$projeto = new Projeto;

$projeto->nome = $request->nome;

$projeto->lider = $request->lider;

$projeto->ano= $request->ano;

$projeto->save();

}

Recebe os dados através da variável $request, e grava no banco de dados com o método **save()** da model Projeto

public function show($id)

{

$projeto = Projeto::find($id);

echo $projeto;

}

Procura o registro no banco de dados pelo id passado pela URL e mostra as informações para o usuário em formato de array, apenas para testes, deve ser retornado para uma view de exibir dados depois.

public function edit($id)

{

$projeto = Projeto::find($id);

return view('editar\_projeto', $projeto);

}

O método edit procura o registro no banco de dados pelo id, e retorna os valores para a view de editar, chamada ‘editar\_projeto’, para os campos de formulário serem preenchidos.

public function update(Request $request, $id)

{

$projeto = Projeto::find($id);

$projeto->nome = $request->nome;

$projeto->lider= $request->lider;

$projeto->ano = $request->ano;

$projeto->save();

}

O método update vai receber os dados via POST do formulario de editar, e vai receber o id pela URL, pesquisa no banco de dados para encontrar o registro com o id, atualiza os valores com os recebidos pelo $request e usa o método save() para atualizar as informações no banco de dados

public function destroy($id)

{

$projeto = Projeto::find($id);

$projeto->delete();

}

O método destroy vai procurar o registro no banco de dados, pelo id, e vai excluir a linha do banco de dados através do método **delete()** da model Projeto.

## **Views : parte visual do sistema**

Disponíveis em resource/views/

Exemplo de view de cadastro, chamada cadastro\_projeto.blade.php:

<!DOCTYPE html>

<html>

<body>

<form method="POST" action="{{url('/Projeto')}}">

@csrf

nome:<br>

<input type="text" name="nome">

<br>

lider:<br>

<input type="text" name="lider">

<br>

ano:<br>

<input type="text" name="ano" >

<input type="submit" value="Submit">

</form>

</body>

</html>

Protege o sistema contra ataques de negação de serviço através do @csrf e Envia os dados desse formulário de cadastro para a rota

POST Localhost:8000/Projeto

que será tratada Método Store do ProjetoController.

View de editar, chamada editar\_projeto.blade.php:

<!DOCTYPE html>

<html>

<body>

<form method="POST" action="{{url('/Projeto',$id)}}">

@csrf

@method("PUT")

nome:<br>

<input type="text" name="nome" value="{{$nome}}">

<br>

lider:<br>

<input type="text" name="lider" value="{{$lider}}">

<br>

ano:<br>

<input type="text" name="ano" value="{{$ano}}">

<input type="submit" value="Submit">

</form>

</body>

</html>

Essa view é exatamente igual a view de cadastro, mas vai preencher os valores com os que já estão no banco de dados, passando o valor da variável para o atributo value de cada campo do formulario.

View index, chamada projeto\_index.blade.php:

@forelse ($projetos as $projeto)

{{$projeto->nome}}

{{$projeto->lider}}

{{$projeto->ano}}

<a href="/Projeto/{{$projeto->id}}"> visualizar </a>

<a href="/Projeto/{{$projeto->id}}/edit"> editar </a>

<form method="POST" action="/Projeto/{{$projeto->id}}">

@csrf

@method("delete")

<input type="submit" value="Excluir"/>

</form>

<br>

@empty

<p>Nenhum Projeto Encontrado! </p>

@endforelse

<a href="/Projeto/create">Criar novo Projeto</a>

Essa view recebe um array com todos os valores do banco de dados e utiliza o @forelse para exibir cada um, e o @empty para exibir alguma mensagem caso não encontre nenhum registro no banco de dados.

Além disso tem links para editar, visualizar e excluir o projeto.

## **Configurar banco de dados**

Acesse o arquivo .env, disponível na raiz do projeto Laravel e configure os seguintes parametros

DB\_DATABASE=nome do banco

DB\_USERNAME=nome do usuario

DB\_PASSWORD=senha

Rode as migrations com o seguinte comando no terminal, para criar todas as tabelas no banco de dados.

PHP ARTISAN MIGRATE

E teste no navegador se todas as operações estão funcionando adequadamente.

Acessando:

Localhost:8000/Projeto/