**SUMÁRIO**

[1. TÍTULO1 4](#_Toc51005538)

[1.1. Subtítulo1 4](#_Toc51005539)

[2. Título 2 4](#_Toc51005540)

[2.1. Subtítulo2 4](#_Toc51005541)

[2.1.1. Subtítulo do subtítulo do título2 5](#_Toc51005542)

[3. Título3 5](#_Toc51005543)

[4. BIBLIOGRAFIA 6](#_Toc51005544)

# fundamentos do workflow do laravel

Esse documento não é diferente da documentação oficial do Laravel, e especificamente da sua seção “basics”, sobre o funcionamento básico no nível do código, exceto pelo fato de que é escrito da forma mais clara possível, e com a adição de informações externas complementares.

# organização inicial

É recomendado, na própria documentação do Laravel, que os arquivos de estilização, o Javascript e imagens sejam também armazenados na pasta public.

Tela de um aparelho eletrônico

Descrição gerada automaticamente com confiança média

# O ROTEAMENTO

Todas as rotas do Laravel são definidas em seus arquivos de rota, que estão localizados no diretório routes. Cada rota é controlada por um controlador, e por uma action/método especifica dele.

Existe mais de um arquivo de roteamento, mas os principais podemos considerar como sendo o web.php, que é o roteamento para aplicações web, e o api.php, que serve para o roteamento das requisições AJAX.

## SÍNTAXE BÁSICA DO ROTEAMENTO

Para a maioria dos aplicativos, você começará definindo rotas no arquivo routes/web.php. As rotas definidas em routes/web.php podem ser acessadas inserindo a URL da rota definida em seu navegador.

A sintaxe é essa básica para a configuração de uma rota é essa:

use App\Http\Controllers\UserController;

Route::get('/user', [UserController::class, 'index']);

No exemplo acima, a rota “/user” é controlada pelo controller “UserController”, e pela sua action “index”, via método GET. Para rotas de processamento de formulários, por exemplo, usa-se o método Route::post.

## REDIRECIONAMENTO COM REDIRECT NO ARQUIVO DE ROTA

Se você está definindo uma rota que redireciona para outro URI, você pode usar o método Route::redirect. Este método fornece um atalho conveniente para que você não precise definir uma rota completa ou controlador para realizar um redirecionamento simples:

Route::redirect('/origem', '/destino');

## ROTAS NOMEADAS

As rotas nomeadas permitem a geração conveniente de URLs ou redirecionamentos para rotas específicas. Você pode especificar um nome para uma rota encadeando o namemétodo na definição de rota:

use App\Http\Controllers\UserController;

Route::get('/user', [UserController::class, 'index'])->name(“user”);

Assim, em métodos que utilizam as rotas do aplicativo, como o “redirect()”, ao invés de informada a rota com a inclusão da barra, “/”, pode ser informado apenas o seu nome – neste caso seria redirect(“user”).

## MIDDLEWARE

O middleware fornece um mecanismo conveniente para inspecionar e filtrar solicitações HTTP que entram em seu aplicativo.

Por exemplo, o Laravel inclui um middleware que verifica se o usuário do seu aplicativo está autenticado. Se o usuário não estiver autenticado, o middleware redirecionará o usuário para a tela de login do seu aplicativo.

Middleware adicional pode ser escrito para executar uma variedade de tarefas além da autenticação.

### MATRIZ DOS MIDDLEWARES

Todos os Middlewares são armazenados no arquivo App/Http/Kernel, na matriz “$routeMiddleware”. Como pode ver abaixo, eles possuem uma chave, e o seu valor é uma classe. Para utilizar um ou outro, a chave deve ser informada.

Texto

Descrição gerada automaticamente

### COMO UTILIZAR OS MIDDLEWARES PARA PROTEGER AS ROTAS

Uma vez que o middleware existe no kernel HTTP, você pode usar o **método middleware()** para ligar proteger a rota com um dos middlewares disponíveis.

use App\Http\Controllers\UserController;

Route::get('/user', [UserController::class, 'index'])->name(“user”)->middleware(“auth”);

**Múltiplos podem ser utilizados para uma mesma rota**, e basta que suas chaves sejam informadas dentro de um array.

->middleware([“auth”, “can”]);

Também existem os **grupos de middleware**, que é uma matriz de nome “$middlewareGroups”, também existente no arquivo Kernel.php. Isso serve para que múltiplos middlewares comuns sejam utilizados a partir de uma só chave. Existem dois grupos: o “web” e o “api”.

Route::middleware("web")->group(function(){

// Configuração das rotas

// Serão protegidas pelo middleware “web”

}

## PROTEÇÃO CONTRA ATAQUES CSRF

As falsificações de solicitação entre sites são um tipo de exploração maliciosa em que comandos não autorizados são executados em nome de um usuário autenticado.

O Laravel gera automaticamente um "token" CSRF para cada sessão de usuário ativa gerenciada pela aplicação. Esse token é usado para verificar se o usuário autenticado é a pessoa que está realmente fazendo as solicitações ao aplicativo.

Sempre que definir um formulário HTML "POST", "PUT", "PATCH" ou "DELETE" em seu aplicativo, você deve incluir o token com a diretiva @csfr. Na verdade, se trata de um input oculto com o atributo “value” sendo igual a “csrf\_token()” – método que retorna o token atualmente ativo.

Se esse token não for incluído no formulário, isto irá ocorrer quando seus dados forem enviados:

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

# OS CONTROLLERS

Como já foi explicado, os controladores servem para controlar as operações das rotas da aplicação. Um só controlador pode controlador uma rota, ou várias, com uma mesma action ou mais de uma.

É conveniente que seja utilizado um único controlador para uma única rota, ou métodos diferentes para diferentes rotas, devido ao nível de complexidade e densidade de um código de um controlador dedicado a resolver operações de múltiplas rotas.

## CRIAÇÃO DE UM CONTROLLER

A criação de um Controller se dá facilmente com a utilização do console Artisan. O comando é esse:

**php artisan make:controller** UserController

## CRIANDO UM CONTROLLER DEDICADO

Um controller dedicado, ou de “ação única”, é um que resolve as operações de uma única rota. Esse tipo de controller não precisa que seu método seja especificado na configuração da rota que irá tratar.

Além disso ele possui uma sintaxe diferente do controller padrão. Ao invés de ter múltiplas actions, ele possui uma só e que é criada automaticamente: \_\_invoke() é o seu nome.

**php artisan make:controller** UserController **--invokable**

## CAPTURANDO REQUESTS

A classe **Illuminate\Http\Request** do Laravel fornece uma maneira orientada a objetos para interagir com a solicitação HTTP atual sendo tratada por sua aplicação, bem como recuperar a entrada, cookies e arquivos que foram enviados com a solicitação.

**O primeiro exemplo de caso de uso** seria em controladores que possuem métodos para tratar requisições de formulário. O que ocorre é que eles devem possuir um parâmetro $request do tipo “Request”. Ele é um objeto que recebe os dados da requisição enviada para a rota.

Por exemplo, se o formulário é de login, e possui dois inputs, um de senha, com name “senha” e outro de email, com name “email”, basta utilizar $request->senha e $request->email para acessar o dado enviado por cada um dos inputs.

**O segundo exemplo de caso de uso** seria para recuperar parâmetros da URL ou query strings. Por exemplo, se a rota atual é “/user?id=123”, para recuperar o valor da query string “id” basta escrever $id = request(“id’).

Esses seriam alguns dos principais exemplos, mas existem incontáveis. O objeto $request permite recuperar o caminho da solicitação, com $request->path(), inspecionar a atual rota com $request->is(“/nome\_rota”), inspecionar a rota atual se for nomeada, com $request->routeIs(“nome\_rota”), dados dos headers, entre diversos outras coisas.

## OS RETORNOS DOS CONTROLLERS

Todas as rotas e controladores devem retornar uma resposta a ser enviada de volta ao navegador do usuário. O Laravel oferece várias maneiras diferentes de retornar respostas.

### RETORNAR UMA VIEW

Os controllers se comunicam ou não com models, para capturar dados do banco de dados e nutrir as views que retornam para o usuário.

Para retornar essas views o controller utiliza o método view(“nome\_da\_view”) antecedido do termo reservado para retornos, “return”.

return view(“home”);

Também é possível enviar dados para as views; possibilidade essa que é necessária quando devem ser nutridas com dados do banco de dados.

return view(“home”, [“personName” => $personName]);

Os dados enviados para uma view, que utiliza o template blade, são utilizados a partir de variáveis php, existentes entre dois pares de chaves, e que possuem o exato mesmo nome da chave utilizada para referenciar o dado a ser enviado para a view, no método view(). Ou seja, neste caso seria:

{{$personName}}

### RETORNAR UM OBJETO RESPONSE

Também é possível instâncias da classe **Illuminate\Http\Response**, que são objetos que organizam os retornos dos controllers quando não são views.

Retornar uma instância de Response permite que você personalize o conteúdo da resposta, o código de status HTTP e os cabeçalhos.

use Illuminate\Http\Response;

return response(“Hello World”, 200);

return response('Hello World', 200) ->header('Content-Type', 'text/plain');

### REDIRECIONAMENTO COM REDIRECT

Os redirecionamentos são instâncias da classe **Illuminate\Http\RedirectResponse.** Existem várias maneiras de gerar um redirecionamento. O método mais simples é usar o redirect global:

return redirect(“home/dashboard”); // Rota comum

return redirect(“dashboard”); // Rota nomeada

### TIPO DO DADO DA RESPONSE

O response pode ser usado para gerar outros tipos de instâncias de resposta também, como JSON, e até mesmo um caminho de arquivo para que seja baixado.

return response()->json([“name” => “Batman”]); // Retorna JSON

return response()->download($pathToFile); // Faz download

return response()->file($pathToFile); // Visualização de um arquivo

## CONTROLLERS DE RECURSOS

O roteamento de recursos do Laravel permite atribuir as rotas típicas de criação, leitura, atualização e exclusão ("CRUD") para um controlador com uma única linha de código.

Para começar, podemos usar a opção make:controller do comando Artisan –resource para criar rapidamente um controlador para lidar com estas ações:

**php artisan make:controller** UserController **–resource**

O que ocorre é que ele é criado automaticamente, como também ocorre com controladores comuns, mas com **actions e rotas nomeadas prontamente disponíveis** para realização de ações CRUD.

### CONTROLLER RESOURCE COM MODEL ESPECIFICADO

É possível que os métodos do controlador de recursos trabalhem com uma instância de um model especificado na sua criação. Para isso basta utilizar você pode usar a opção –model na geração do controlador:

**php artisan make:controller** UserController **--model=**User **–resource**

### CONTROLLER RESOURCE PARA FORMULÁRIOS

É possível fornecer a opção –requests ao gerar um controlador de recursos para instruir o Artisan uma classe dedicada a solicitação de formulários:

**php artisan make:controller** UserController **--model**=User **--resource –requests**

### ESPECIFICANDO AS ACTIONS DISPONÍVEIS

# AS VIEWS

//////

## O TEMPLADE BLADE

//////

## AS DIRETIVAS DO TEMPLATE BLADE

//////

### DIRETIVA @YIELD

//////

### DIRETIVA @EXTENDS

//////

### DIRETIVA @SECTION

//////

### OUTRAS DIRETIVAS

//////

## LAYOUTS: VIEW CONSTANTES

//////

## PÁGINAS: VIEWS VARIÁVEIS

//////

# MANIPULAÇÃO DA URL

# SESSÕES

# VALIDAÇÃO

# SISTEMA DE LOGS

# fonte

<https://laravel.com/docs/8.x>