

# INTRODUÇÃO AO LARAVEL



**Douglas Nassif Roma Junior**

 /douglasjunior

 /in/douglasjunior

 douglasjunior.me

 nassifrroma@gmail.com

Slides: <https://git.io/vAd6S>

# AGENDA

- Introdução ao Laravel
- Requisitos do sistema
- Instalando o Laravel
  - Diretório public
  - Arquivos de Configuração
  - Permissão de diretórios
  - Chave da Aplicação
  - Configuração Adicional
- Roteamento

# INTRODUÇÃO AO LARAVEL

*Love beautiful code? We do too.*

*The PHP Framework For Web Artisans*

*Ama código bonito? Nós fazemos também.*

*O Framework PHP Para Artesãos da Web*

# INTRODUÇÃO AO LARAVEL

- **Sintaxe expressiva e elegante**
  - Valoriza a elegância, a simplicidade e a legibilidade? O Laravel foi projetado para pessoas como você.

## Expressive, beautiful syntax.

Value elegance, simplicity, and readability? You'll fit right in. Laravel is designed for people just like you. If you need help getting started, check out [Laracasts](#) and our [great documentation](#).

```
1 class Purchase implements ShouldQueue
2 {
3
4     /**
5      * Purchase a new podcast.
6      */
7     public function handle(Repository $re
8     {
```

# INTRODUÇÃO AO LARAVEL

- Adaptado para sua equipe
  - Se você é um desenvolvedor solo ou uma equipe de 20 pessoas, Laravel atenderá às suas necessidades. Mantenha todos sincronizados usando as migrações agnósticas e o construtor de esquemas do banco de dados do Laravel.

```
~/Apps $ php artisan make:migration create_users
Migration created successfully!

~/Apps $ php artisan migrate --seed
Migrated: 2015_01_12_000000_create_users_table
Migrated: 2015_01_12_100000_create_password_resets_table
Migrated: 2015_01_13_162500_create_projects_table
Migrated: 2015_01_13_162508_create_servers_table
```

## Tailored for your team.

Whether you're a solo developer or a 20 person team, Laravel is a breath of fresh air. Keep everyone in sync using Laravel's database agnostic **migrations** and **schema builder**.

# INTRODUÇÃO AO LARAVEL

- Moderno kit de ferramentas
  - Um excelente ORM, roteamento indolor, biblioteca de fila poderosa e autenticação simples oferecem as ferramentas que você precisa para o PHP moderno e sustentável. Nós suamos as pequenas coisas para ajudá-lo a oferecer aplicativos incríveis.

## Modern toolkit. Pinch of magic.

An **amazing ORM**, painless **routing**, powerful **queue library**, and **simple authentication** give you the tools you need for modern, maintainable PHP. We sweat the small stuff to help you deliver amazing applications.

```
1 Route::resource('photos', 'PhotoController')
2
3 /**
4  * Retrieve A User ...
5  */
6 Route::get('/user/{user}', function(App\l
7 {
8     return $user;
```

# INTRODUÇÃO AO LARAVEL

- Moderno kit de ferramentas
  - Um excelente ORM, roteamento indolor, biblioteca de fila poderosa e autenticação simples oferecem as ferramentas que você precisa para o PHP moderno e sustentável. Nós suamos as pequenas coisas para ajudá-lo a oferecer aplicativos incríveis.

## Modern toolkit. Pinch of magic.

An **amazing ORM**, painless **routing**, powerful **queue library**, and **simple authentication** give you the tools you need for modern, maintainable PHP. We sweat the small stuff to help you deliver amazing applications.

```
1 Route::resource('photos', 'PhotoController')
2
3 /**
4  * Retrieve A User ...
5  */
6 Route::get('/user/{user}', function(App\l
7 {
8     return $user;
```

# REQUISITOS DO SISTEMA

- PHP  $\geq$  7.1.3
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension
- XML PHP Extension
- Ctype PHP Extension
- JSON PHP Extension



# INSTALANDO O LARAVEL

- Laravel utiliza o Composer para gerenciar suas dependências.
- Instalando via Laravel Installer
  - Primeiro faça o download do Laravel Installer utilizando o Composer:  
`composer global require "laravel/installer"`
  - Para criar o projeto, execute:  
`laravel new meu-projeto`
- Instalando via Composer:  
`composer create-project --prefer-dist laravel/laravel meu-projeto`

# DIRETÓRIO PUBLIC

- Após a criação do projeto, você deve configurar o seu web server (apache ou nginx) aponte a raiz para o diretório `public`.
- O `index.php` existente dentro da pasta `public` é controlador frontal para todas as requisições HTTP que chegam até sua aplicação.

# ARQUIVOS DE CONFIGURAÇÃO

- Todos os arquivos de configuração para o framework Laravel são armazenados no diretório de `config`.
- Cada opção está documentada, então fique à vontade para examinar os arquivos e familiarizar-se com as opções disponíveis.

# PERMISSÃO DE DIRETÓRIOS

- Depois de instalar o Laravel, talvez seja necessário configurar algumas permissões.
- Os diretórios `storage` e `bootstrap/cache` devem ser graváveis pelo seu web server (apache ou nginx).

# CHAVE DA APLICAÇÃO

- A próxima coisa que você deve fazer depois de instalar o Laravel é configurar sua chave de aplicativo para uma `string` aleatória.
- Se você instalou o Laravel via Composer ou o instalador do Laravel, essa chave já foi configurada para você pelo comando:

```
php artisan key:generate
```

# CONFIGURAÇÃO ADICIONAL

- Laravel não precisa de nenhuma outra configuração para trabalhar adequadamente.
- Porém, você pode querer visitar o arquivo `config/app.php` e sua documentação, para configurar coisas como o `timezone` e o `locale` de sua aplicação.

# ROTEAMENTO

- A forma mais básica de se declarar uma rota no Laravel é através de uma URI e uma função de *callback* (*closure*).

routes/web.php

```
Route::get('ola', function () {  
    return 'Olá Laravel';  
});
```

# ROTEAMENTO

- A forma mais básica de se declarar uma rota no Laravel é através de uma URI e uma função de *callback* (*closure*).

routes/web.php

```
Route::get('ola', function () {  
    return 'Olá Laravel';  
});
```



# ROTEAMENTO

- Todas as rotas do Laravel são definidas em seus arquivos de rota, que estão localizados no diretório `routes`. Esses arquivos são carregados automaticamente pelo framework.
- O arquivo `routes/web.php` define as rotas que são para sua interface web.
  - Essas rotas são atribuídas ao grupo de middleware `web`, que fornece recursos como o estado da sessão e a proteção [CSRF](#).
- As rotas em `routes/api.php` são *stateless* (não armazenam estado) e são atribuídos o grupo de middleware `api`.

# ROTEAMENTO

- Métodos HTTP disponíveis para as rotas:

```
Route::get($uri, $callback);  
Route::post($uri, $callback);  
Route::put($uri, $callback);  
Route::patch($uri, $callback);  
Route::delete($uri, $callback);  
Route::options($uri, $callback);
```

# ROTEAMENTO

- Você também pode registrar rotas que respondem a múltiplos métodos HTTP:

```
Route::match(['get', 'post'], '/ola', function () {  
    //  
});  
  
Route::any('/ola', function () {  
    //  
});
```

# ROTEAMENTO

- Se você estiver definindo uma rota que redireciona para outro URI, você pode usar o método `Route::redirect`.

```
Route::redirect('/daqui', '/prala', 301);
```

# ROTEAMENTO

- Se sua rota precisa somente redirecionar para uma página (view), então você pode usar o método `Route::view`.

```
Route::view('/bemvindo', 'bemvindo');
```

- O método `view` aceita um URI como seu primeiro argumento e um nome de `view` como seu segundo argumento. Além disso, você pode fornecer uma matriz de dados para passar para a exibição como um terceiro argumento opcional:

```
Route::view('/bemvindo', 'bemvindo', ['nome' => 'Douglas']);
```

# ROTEAMENTO

- Algumas vezes você pode necessitar passar parâmetros para suas rotas, você pode fazer isso da seguinte forma:

```
Route::get('usuario/{id}', function ($id) {  
    return 'Usuário selecionado: ' . $id;  
});
```

- Você também pode definir múltiplos parâmetros:

```
Route::get('venda/{venda}/item/{item}', function ($vendaId, $itemId) {  
    //  
});
```

# ROTEAMENTO

- Algumas vezes você pode necessitar passar parâmetros **obrigatórios** para suas rotas, você pode fazer isso da seguinte forma:

```
Route::get('usuario/{id}', function ($id) {  
    return 'Usuário selecionado: ' . $id;  
});
```

- Você também pode definir múltiplos parâmetros **obrigatórios**:

```
Route::get('venda/{venda}/item/{item}', function ($vendaId, $itemId) {  
    //  
});
```

# ROTEAMENTO

- Em alguns casos, você precisa que o parâmetro seja opcional, para isso basta utilizar o marcador `?` após o nome do parâmetro.

```
Route::get('usuario/{nome?}', function ($nome = null) {  
    return 'Usuário selecionado: ' . $nome;  
});
```

- Você também pode definir um valor padrão para este parâmetro:

```
Route::get('usuario/{nome?}', function ($nome = 'Douglas') {  
    return 'Usuário selecionado: ' . $nome;  
});
```



# ROTEAMENTO

- Os **grupos de rotas** permitem que você compartilhe atributos de rota, como middleware, namespaces e prefixos, em um grande número de rotas.
- Exemplo de prefixo de rota:

```
Route::prefix('admin')->group(function () {  
    Route::get('usuarios', function () {  
        // define rota para /admin/usuarios  
    });  
    Route::get('vendas', function () {  
        // define rota para /admin/vendas  
    });  
});
```

# MIDDLEWARES

- Os `middlewares` fornecem um mecanismo conveniente para filtrar pedidos HTTP que entram no seu aplicativo.
- Por exemplo, o Laravel inclui um `middleware` que verifica se o usuário do seu aplicativo está autenticado. Se o usuário não for autenticado, o `middleware` redirecionará o usuário para a tela de login.
- Existem vários `middlewares` incluídos no framework Laravel, incluindo `middleware` para autenticação e proteção CSRF. Todos esses estão localizados no diretório `app/Http/Middleware`.

# MIDDLEWARES

- Para criar um novo middleware, utilize o comando `make:middleware` do Artisan.
- `php artisan make:middleware ChecarIdade`

# MIDDLEWARES

- Em `app/Http/Middleware` **edite o arquivo criado** `ChecarIdade.php`

```
public function handle($request, Closure $next)
{
    if ($request->query('idade') > 200) {
        return response('A idade deve ser menor que 200.', 400);
    }
    return $next($request);
}
```

# MIDDLEWARES

- Registre o seu middleware no arquivo `app/Http/Kernel.php`

```
protected $routeMiddleware = [  
    ...  
    'checar.idade' => \App\Http\Middleware\ChecarIdade::class,  
];
```

# MIDDLEWARES

- Defina uma rota que utilize este middleware em `routes/web.php`

```
Route::middleware('checar.idade')->group(function () {  
    Route::view('/bemvindo', 'bemvindo');  
});
```

# MIDDLEWARES

- Verifique o seu middleware em ação acessando as URLs:

<http://seu-app/benvindo?idade=100>

E também:

<http://seu-app/benvindo?idade=300>

# MIDDLEWARES

- Você também pode definir um middleware que execute uma ação após a requisição ser tratada pela rota de destino.

```
public function handle($request, Closure $next)
{
    $response = $next($request);

    // Implemente sua operação aqui

    return $response;
}
```



# DÚVIDAS?



**Douglas Nassif Roma Junior**

 /douglasjunior

 /in/douglasjunior

 douglasjunior.me

 nassifrroma@gmail.com

Slides: <https://git.io/vAd6S>