**SUMÁRIO**

[1. FUNDAMENTOS DO LARAVEL: O KIT FULLSTACK 1](#_Toc89136308)

[2. kit laravel breeze 1](#_Toc89136309)

[3. kit laravel liveware 1](#_Toc89136310)

[4. kit laravel jetstream 1](#_Toc89136311)

# FUNDAMENTOS DO LARAVEL: O KIT FULLSTACK

Os kits ofertados pelo Laravel estruturam automaticamente o aplicativo com as rotas, controladores e visualizações para registrar e autenticar os usuários.

# kit laravel breeze

**Laravel Breeze é uma implementação mínima e simples** de todos os recursos de autenticação do Laravel , incluindo login, registro, redefinição de senha, verificação de e-mail e confirmação de senha. A camada de visualização padrão do Laravel Breeze é composta de modelos com template Blade estilizados com CSS do Tailwind .

**O Breeze oferece um excelente ponto de partida para iniciar um novo aplicativo Laravel** e é uma ótima escolha para projetos que planejam levar seus modelos Blade para o próximo nível com o **Laravel Livewire**.

# kit laravel liveware

**Laravel Livewire é uma biblioteca que simplifica a construção de interfaces modernas, reativas e dinâmicas usando o Laravel Blade** como sua linguagem de templates. Esta é uma ótima pilha para escolher se você deseja construir um aplicativo que seja dinâmico e reativo sem ter que utilizar Javascript e seus frameworks.

**O Livewire funciona de um jeito beeem peculiar**. Qualquer interação que acontecer com ele, se faz uma requisição AJAX e o HTML é renderizado do lado do servidor e depois retornado ao cliente. Esse fluxo permite que ele seja reativo, porém, um pouco mais lento.

**Ao usar o Livewire, as rotas do seu aplicativo responderão com modelos Blade** típicos. No entanto, dentro desses modelos, você pode renderizar componentes Livewire conforme necessário.

## COMO FUNCIONA O LIVEWARE NA PRÁTICA

## LIVEWARE É ORIENTADO A COMPONENTES

O Livewire funciona com a criação e manipulação de componentes – assim como faz, por exemplo, o framework React JS e também o próprio mecanismo de [componentes do Laravel](https://laravel.com/docs/7.x/blade#components).

### BREVE EXPLICAÇÃO DOS COMPONENTES BLADE DO LARAVEL

**O Laravel tem um mecanismo de componentes bastante semelhante ao do React JS**. A ideia é a mesma. Um componente, no React JS, é um arquivo criado que tem uma função que retorna um conteúdo HTML, e cuja folha de estilização é integrada a ele mesmo. Ele é utilizado com uma sintaxe de tag <NomeComponente />.

**Já no Laravel, os componentes são criados** com o console Artisan, e ao invés de funcionais são componentes de classe.

**php artisan make:component** Alert

**Quando criado o componente Laravel com esse comando**, um novo arquivo Alert.php será criado no diretório **app/views/components**. E, de forma correspondente, uma view será criada e alocada em **resources/views/components**.

Além disso, e também semelhante ao React, **para exibir um componente Laravel** é preciso escrever uma tag com o mesmo nome da classe, mas em minúsculo, e com um “x-“ como prefixo, desta forma: **<x-nomecomponente>**.

Também, assim como com os componentes React, **é possível enviar dados para os componentes Blade**.

**<x-alert** type="error" :message="$message"**/>**

Na view em que o componente será utilizado, **o valor poderá ser recuperado com a sintaxe tradicional do Blade para dados enviados para views**, isto é, com dois pares de chaves, {{ }}, e uma variável com nome igual a sua chave , que neste caso são os atributos “message” e “type”. O porquê de “message” ser antecedido de dois pontos é porque se trata de uma variável. Para expressões e variáveis isso é necessário.

No entanto, para que isso realmente seja possível, **os dados enviados** **devem ser definidos no construtor da classe** do componente. Todas as propriedades públicas do componente, configuradas na sua classe, serão automaticamente disponibilizadas para a visualização.

**Considerando esses dados**, desse exemplo, a classe do componente Alert teria dois atributos públicos, $type e $message, um construtor para atribuir valores para esses atributos, e um método render() que retornaria a view “alert”, existente em resources/views/components/.

### COMPONENTES DO LIVEWARE

Conhecendo os componentes do Laravel, agora é possível descrever os do Liveware. Pois bem: para criar um componente Liveware, “Alert” novamente, é necessário digitar esse comando:

**php artisan make:livewire** Alert

O comando também criará dois novos arquivos: **app\Http\Livewire\Alert.php** e um modelo de visualização **resources/views/livewire/alert.php**.

Para renderizar um componente Livewire, de forma semelhante a um componente Blade, basta digitar **<livewire: alert />** ou **@livewire ('alert')**.

**Como você é possível ver, o fluxo é bastante semelhante**. A única grande diferença é que com os componentes Livewire, há uma sincronização em tempo real (nenhuma atualização de página necessária) entre a classe do componente e seu modelo de visualização.

Para vincular uma propriedade de componente ao valor de um atributo definido na tag do componente, basta digitar, por exemplo:

**<livewire: alert wire:var = “teste”/>**

## ATUALIZAÇÃO DOS ESTADOS DOS COMPONENTES

Entendendo “componente” como uma estrutura HTML, o Livewire renderiza a saída do componente inicial para a página, ou Layout, assim como o típico modelo Blade renderizado pelo servidor.

A fuga da forma síncrona de operação é possível porque o Livewire faz uma solicitação AJAX ao servidor com dados atualizados sempre que ocorre uma modificação no estado do componente. Assim, o servidor renderiza novamente o componente e responde com o HTML atualizado.

## ATIVANDO LIVEWARE EM UMA PÁGINA

**Para fazer o Livewire funcionar em uma página**, você precisa incluir os estilos e scripts do Livewire em cada página que precisar deles. É interessante que isso seja feito em um layout, isto é, uma página não variável, mas que inclui outras que cujo conteúdo é variável.

**O que é necessário é a escrita de duas diretivas na página**: **@livewireStyles**, na tag <head> e **@livewireScripts** antes do fechamento da tag <body>.

## CONFIGURAÇÃO DO AMBIENTE LIVEWARE

A documentação da ferramenta pode ser encontrada [aqui](https://laravel-livewire.com/docs/2.x/installation), mas serão descritos aqui também os passos necessários.

Tendo instalado o Laravel, esse comando deve ser executado:

**composer require livewire/livewire**

# kit laravel jetstream

**Laravel Jetstream é o próximo passo para aqueles que já conhecem o kit Breeze**. O Jetstream fornece a implementação para login, registro, verificação de e-mail, autenticação de dois fatores, gerenciamento de sessão, API via Laravel Sanctum e recursos opcionais.