



# Plataformas de aplica  es Web

## Aula 05 - Plataformas full stack - Parte 2



Material Did tico do Instituto Metr pole Digital - IMD

### Termo de uso

Os materiais did ticos aqui disponibilizados est o licenciados atrav s de Creative Commons **Atribui  o-SemDeriva  es-SemDerivados CC BY-NC-ND**. Voc  possui a permiss o para realizar o download e compartilhar, desde que atribua os cr ditos do autor. N o poder  alter -los e nem utiliza-los para fins comerciais.

Atribui  o-SemDeriva  es-SemDerivados

CC BY-NC-ND



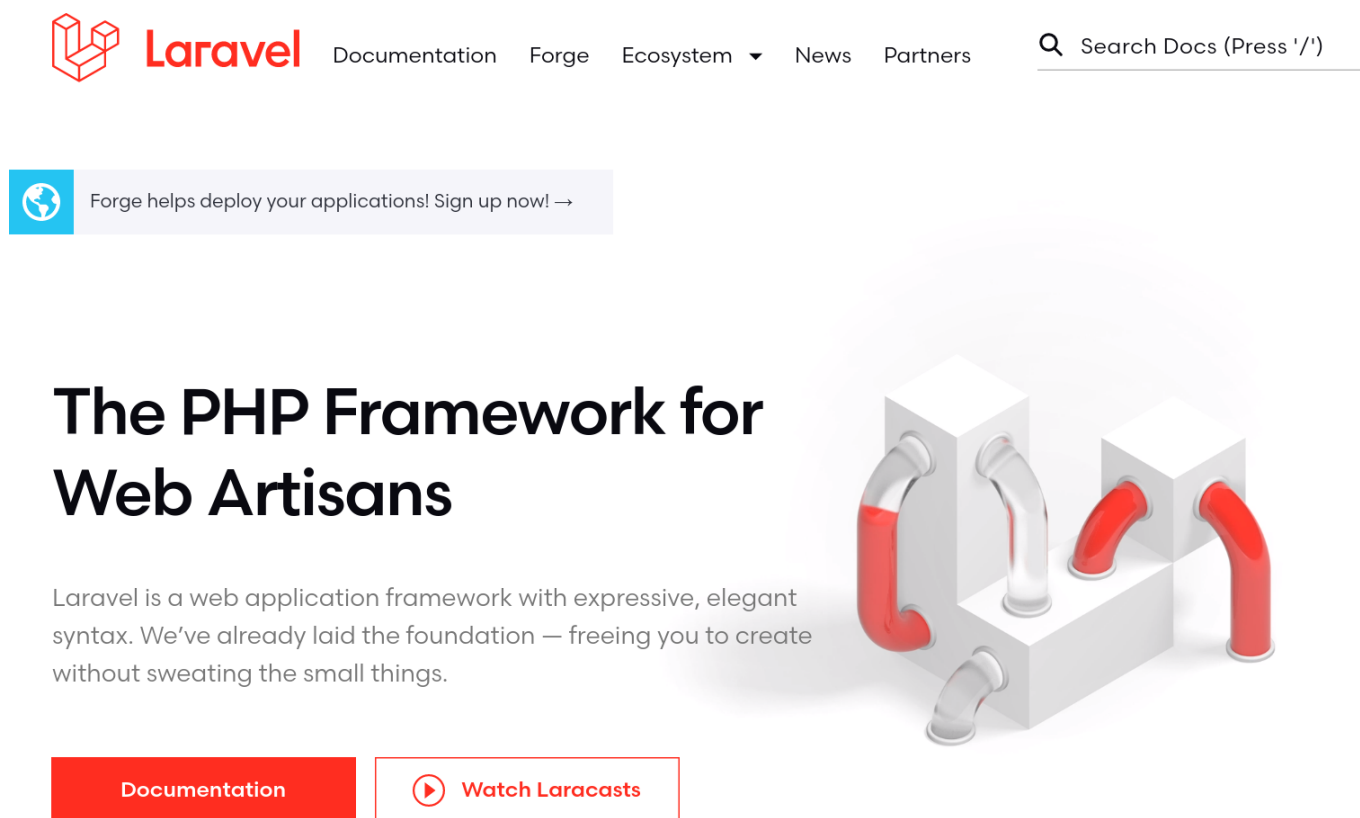
<https://creativecommons.org/licenses/by-nc-nd/4.0/>

# Apresentação

Dando continuidade à apresentação de plataformas full-stack, veremos nesta aula, as características do framework full-stack Laravel, uma plataforma bastante popular no desenvolvimento de sistemas web utilizando a linguagem de programação PHP. Iremos falar sobre suas principais funções, daremos uma visão geral do que é um ambiente de desenvolvimento com Laravel e também apresentaremos mostrar um pequeno exemplo da linguagem de programação PHP, sempre mostrando onde buscar mais informações sobre os tópicos das aulas, para o caso você desejar se aprofundar mais.

## Laravel

Um dos mais famosos frameworks baseados na linguagem de programação PHP, o Laravel surgiu em 2011, com a intenção de fornecer um framework de aplicações web full-stack para essa linguagem com uma maior quantidade de recursos que as opções da época para essa linguagem.



The screenshot shows the Laravel website homepage. At the top, there is a navigation bar with the Laravel logo (a red cube icon) and the word "Laravel" in red. To the right of the logo are links for "Documentation", "Forge", "Ecosystem", "News", and "Partners". Further right is a search bar with the text "Search Docs (Press '/')". Below the navigation bar is a blue banner with a globe icon and the text "Forge helps deploy your applications! Sign up now! →". The main content area features the heading "The PHP Framework for Web Artisans" in large, bold, black text. Below the heading is a paragraph of text: "Laravel is a web application framework with expressive, elegant syntax. We've already laid the foundation — freeing you to create without sweating the small things." To the right of the text is a 3D illustration of white cubes with red pipes connected to them. At the bottom of the main content area, there are two red buttons: "Documentation" and "Watch Laracasts".

*Website do framework full-stack Laravel. Fonte: Autor*

De acordo com o site stackshare, mais de 3000 empresas declaram usar o Laravel na como uma de suas tecnologias base, incluindo a MasterCard, Bitpanda e a rede de supermercados Tesco.

A linguagem de programação PHP, base do Laravel, segue como a mais utilizada dentre todos os websites na Internet. Em 2021, de acordo com o site W3Techs, o PHP é a linguagem de programação utilizada em 79% de todos os sites da web. Isso faz com que o Laravel seja uma opção muito interessante para uma comunidade grande de desenvolvedores familiarizados com essa linguagem de programação, impulsionando bastante sua adoção.

Os desenvolvedores do Laravel o identificam da seguinte forma: "Um framework de aplicações web com sintaxe expressiva, elegante. Já estabelecemos a base, liberando você para criar sem se preocupar com os detalhes."


Como muitos outros, o Laravel é um framework baseado na arquitetura MVC (Model-View-Controller ou Modelo-Visão-Controle, em português), que divide a aplicação em três camadas, sendo uma relacionada ao banco de dados (Model), outra à visualização de dados e páginas (View) e uma última responsável pelo fluxo de informações entre as camadas Model e View (Controller).

Laravel oferece aumento da velocidade e produtividade quando você está escrevendo seus códigos, cuidando das características de performance, segurança, modularização e flexibilidade da aplicação, pois vem com um conjunto de funcionalidades prontas como autenticação, roteamento e modelos HTML prontos.

Tem uma comunidade bastante ativa com muito conteúdo para estudo disponível de forma gratuita em vídeos e artigos. O Laracast (<https://laracasts.com/>) é uma plataforma de cursos e treinamentos oficiais.

**LARACASTS** TOPICS SERIES DISCUSSIONS PODCAST

Q SIGN IN GET STARTED



## Laravel 8 From Scratch

Version: Laravel 8

We don't learn tools for the sake of learning tools. Instead, we learn them because they help us accomplish a particular goal. With that in mind, in this series, we'll use the common desire for a blog - with categories, tags, comments, email notifications, and more - as our goal. Laravel will be the tool that helps us get there. Each lesson, geared toward newcomers to Laravel, will provide instructions and techniques that will get you to the finish line.

This version of our popular Laravel From Scratch series was recorded in 2021, and uses Laravel 8.

BEGIN ADD TO WATCHLIST

Intermediate 66 episodes 9h 4m Frameworks

f t

## YOUR TEACHER | JEFFREY WAY



VISIT WEBSITE



Hi, I'm Jeffrey. I'm the creator of Laracasts and spend most of my days building the site and thinking of new ways to teach confusing concepts. I live in Orlando, Florida with my wife and two kids.

## SECTION 1 Prerequisites and Setup

## An Animated Introduction to MVC

01

Before we get started, come along for a quick two minute overview of the MVC architecture. MVC stands for "Model, View, Controller" and is the bedrock for building Laravel applications.

EPISODE 1 2:40 minutes

Página do curso "Laravel 8 from scratch" na plataforma Laracast. Fonte: <https://laracasts.com/series/laravel-8-from-scratch>

# A linguagem de programação PHP

Web site: <https://php.net>

PHP é um acrônimo recursivo para PHP: Hypertext Preprocessor (Pré-Processador de Hipertexto), que originalmente se chamava Personal Home Page (Página Inicial Pessoal).

Trata-se de uma das linguagens de scripts existentes como JavaScript e Python e é muito usada para a criação de sistemas web do lado do servidor (back-end e full-stack),.

Por ser uma linguagem de script, o PHP precisa de um interpretador para executar os programas criados nessa linguagem. A linguagem PHP pode ser facilmente

integrada em servidores Web como o Apache e o NGINX e essa facilidade, somada a simplicidade da linguagem, deu ao PHP uma popularidade imensa como linguagem de criação de páginas dinâmicas renderizadas no lado do servidor. O PHP pode ser integrado dentro de páginas HTML comuns, misturando código HTML com código PHP de maneira muito parecida com o EJS (já visto no curso). A diferença é que o PHP não é só uma linguagem de template, e sim uma linguagem completa que permite a criação de páginas com conteúdo que pode vir de outros serviços, bancos de dados, cálculos matemáticos, etc.

Tipicamente um documento HTML que contém código PHP tem a extensão ".php" no final e é possível criar sistemas completos somente com páginas PHP puras, sem nenhum tipo de framework. O Facebook, por exemplo, foi criado dessa forma.

Entretanto, existem diversos frameworks e ferramentas Web criadas com a linguagem PHP que adicionam novas funcionalidades para a criação de páginas e sistemas web. O Laravel, por exemplo, é um framework muito popular criado em PHP, assim como o Wordpress é hoje ainda a plataforma mais popular para a criação de páginas na Internet, também escrito em PHP.

No site [https://www.php.net/manual/pt\\_BR/](https://www.php.net/manual/pt_BR/) você tem acesso a documentação oficial do PHP traduzida para o português do Brasil.

Quando o PHP interpreta um arquivo ele procura pelas tags de abertura e fechamento, `<?php` e `?>`, que dizem ao PHP para iniciar ou parar a interpretação do código entre elas. A interpretação assim permite ao PHP ser incluído em vários tipos de documentos, pois tudo que está fora destas tags é ignorado pelo interpretador do PHP. PHP inclui uma tag curta echo `<?=>` que é uma forma abreviada mais verbosa para `<?php echo`.

## Exemplo de um documento PHP

```
<html>
  <head>
    <title>Página PHP</title>
  </head>
  <body>
    <?php echo 'O comando echo do PHP imprime esse texto dentro do
documento<br/>'; ?>
    Também é possível usar a tag curta <?=> 'para mostrar esse texto'
?>
    ou para exibir o resultado de 1 + 1 que é <?=> 1 + 1 ?>.
  </body>
</html>
```

Repare no exemplo acima que as tags PHP foram abertas e fechadas várias vezes e cada uma delas "gera" um texto que a substituirá no documento final processado que será um HTML puro que irá para o navegador sem nenhum código PHP, mas sim com o resultado do seu processamento. Veja o resultado dessa página processada e enviada ao navegador:

```
<html><head>
  <title>Página PHP</title>
</head>
<body>
  O comando echo do PHP imprime esse texto dentro do
documento<br/>
  Também é possível usar a tag curta para mostrar esse texto
  ou para exibir o resultado de 1 + 1 que é 2.
</body></html>
```

É possível também abrir tags PHP em um documento que não resultam em um texto de saída, mas que podem rodar qualquer código desejado, por exemplo para criar variáveis, obter dados de bancos de dados, de APIs externas etc.

Vamos agora ver uma página PHP com algumas funcionalidades:

- Comentários
- Variáveis inicializadas com valores numéricos e texto
- Um Array criado com três strings
- Uma função de soma criada dentro do documento Essas funcionalidades serão criadas em uma tag `<?php ... ?>` sem o uso do "echo", ou seja, elas serão processadas somente com a intenção de fornecer dados e funcionalidades para durante o processamento da página que fará uso de as variáveis e funções criadas para injetar conteúdo com as tags `<?= ?>`. Veja a página abaixo:

```
<?php
/* Isso é um comentário em PHP */
$nome = "Joaquim";

/* Variáveis tem um $ no início, seguindo do seu nome */
$idade = 22;

/* Arrays são criados com o comando array() com uma lista de valores
separados pro vírgula */
$animais_de_estimacao = array("Totó", "Lulu", "Baleia");

/* funções podem ser declaradas facilmente dentro de arquivos PHP
```

```
assim: */
function somar($a, $b) {
    /* Operações matemáticas comuns em valores e variáveis seguem o
    mesmo padrão da maioria das linguagens como C e Javascript */
    $resultado = $a + $b;
    return $resultado;
}

?>

<html>
  <head>
    <title>Página do <?= $nome ?></title>
  </head>
  <body>
    <h1>Página pessoal do <?= $nome ?></h1>
    Oi eu sou o <?= $nome ?>, tenho <?= $idade ?> anos e essa é
    minha lista de animais:
    <ul>
      <?php foreach($animais_de_estimacao as $animal) { ?>
        <li><?= $animal ?></li>
      <?php } ?>
    </ul>
    Obrigado por me visitar hoje, dia: <?= date("d/m/Y") ?><br/>
    Caso tenha curiosidade, 1 + 1 = <?= somar(1, 1) ?>

  </body>
</html>
```

Repare que a variável `$nome` foi utilizada várias vezes no documento, a variável `$idade` somente uma vez, o array chamado `$animais_de_estimacao` foi iterado com o comando `foreach` para criar os itens (`<li>`) de uma lista no HTML `<ul>`. Ao fim ainda foi utilizada a função "date", interna do PHP, para exibir a data atual gerada pelo servidor no formato desejado e também a função "soma", que criamos no início do arquivo para processar e retornar o resultado de da soma de dois números (nesse caso testamos com 1 + 1).

Depois de processado o documento HTML final é esse:

```
<html>
  <head>
    <title>Página do Joaquim</title>
  </head>
  <body>
```

```
<h1>Página pessoal do Joaquim</h1>
Oi eu sou o Joaquim, tenho 22 anos e essa é minha lista de
animais:
<ul>
  <li>Totó</li>
  <li>Lulu</li>
  <li>Baleia</li>
</ul>
Obrigado por me visitar hoje, dia: 04/08/2021<br>
Caso tenha curiosidade, 1 + 1 = 2

</body>
</html>
```

## Página pessoal do Joaquim

Oi eu sou o Joaquim, tenho 22 anos e essa é minha lista de animais:

- Totó
- Lulu
- Baleia

Obrigado por me visitar hoje, dia: 04/08/2021  
Caso tenha curiosidade, 1 + 1 = 2

Na visão do navegador:

*Página PHP exibida no navegador. Fonte: Autor.*

Não é objetivo da aula ensinar completamente o PHP para você e sim somente mostrar que ela pode ser utilizada facilmente em conjunto com páginas HTML. O Laravel utiliza o PHP extensivamente, não só na geração de páginas (Views) com o motor de templates chamado "Blade" como também no processamento internos das requisições, configurações do ambiente e até para acessar bancos de dados.

Para mais informações sobre essa linguagem visite: <https://php.net>

Se desejar, para testar o PHP online você pode utilizar serviços como o REPLIT: <https://replit.com/> bastando criar uma conta e em seguida escolhendo "Create new Replit" com a opção "PHP Web Server". Nesse ambiente você pode escrever seus documentos PHP e clicar no botão de executar para ver o resultado no navegador.

## Características do Laravel

### The Progressive Framework

O Laravel se diz um framework "progressivo". Isso significa que o Laravel cresce



com você. Se você está apenas dando os primeiros passos no desenvolvimento web, a vasta biblioteca de documentação, guias e tutoriais em vídeo do Laravel o ajudará a aprender o básico sem ficar sobrecarregado.

Se você é um desenvolvedor mais experiente, o Laravel oferece ferramentas robustas para injeção de dependência, teste de unidade, filas, eventos em tempo real e muito mais. O Laravel é ajustado para construir aplicações web profissionais e pronto para lidar com cargas de trabalho corporativas.

## A Estrutura Escalável

O Laravel é incrivelmente escalonável. Graças à natureza amigável de escalonamento do PHP e ao suporte embutido do Laravel para sistemas de cache rápido e distribuído como o Redis, o escalonamento horizontal com o Laravel é uma brisa.

Existe uma plataforma como serviço oficial do Laravel chamada "Vapor" que permite que você execute seu aplicativo Laravel em escala quase ilimitada na mais recente tecnologia sem servidor da AWS (Serviços de nuvem da Amazon).

## A Estrutura da Comunidade

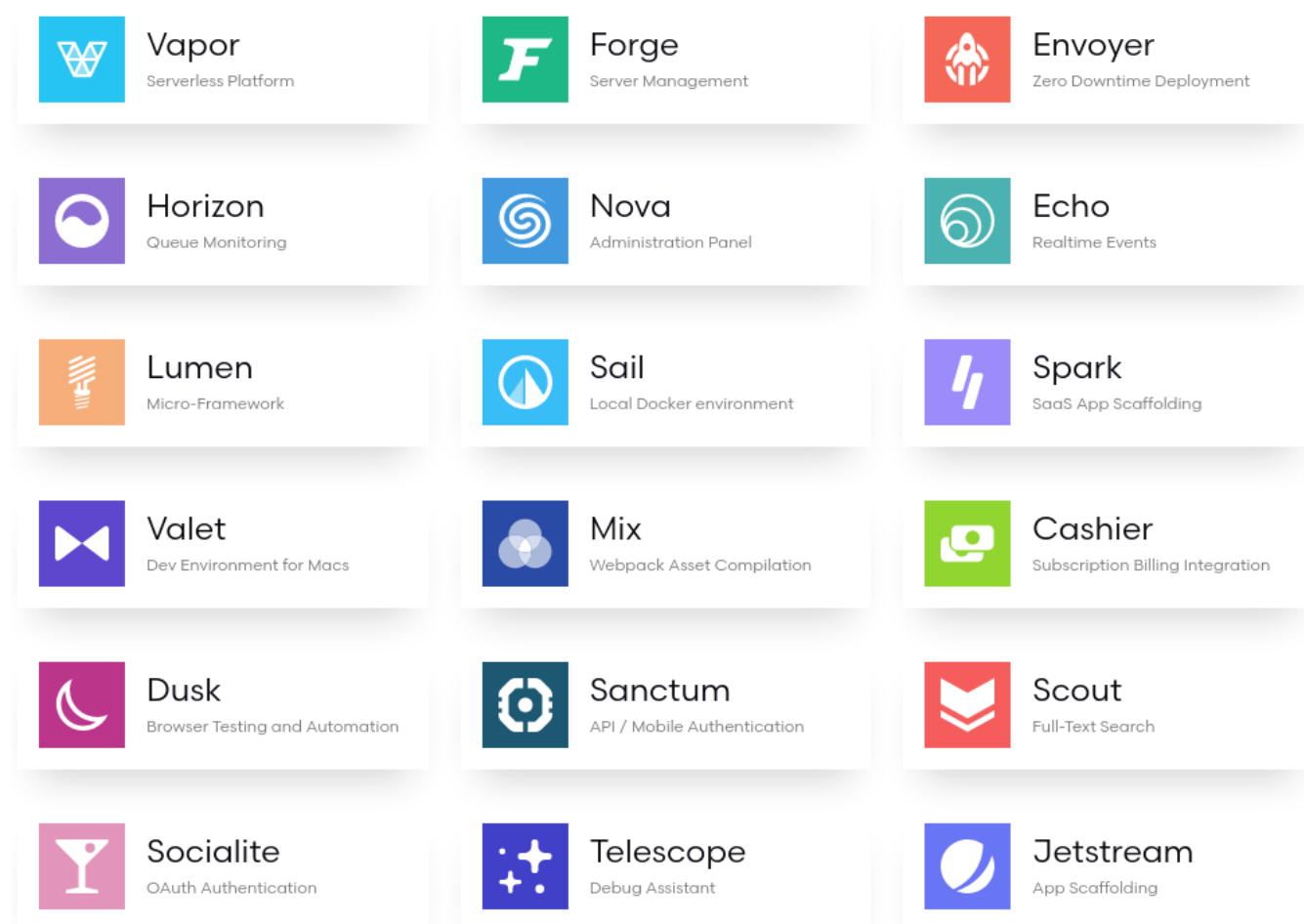
O Laravel combina um excelente pacote de ferramentas do ecossistema PHP. Além disso, milhares de desenvolvedores talentosos de todo o mundo contribuíram para a estrutura.

## Ecossistema Laravel

O Laravel conta com um ecossistema, tanto aberto como serviços comerciais que facilita a adoção não só por desenvolvedores autônomos, como também por empresas que precisam de suporte comercial e técnico na nuvem.

Revolutionize how you build the web.

# The Laravel Ecosystem



O ecossistema de serviços e produtos Laravel. Fonte: <https://laravel.com/>

## Iniciando com o Laravel

Se você se interessou pelo Laravel, a forma mais fácil de iniciar seus estudos é instalando a ferramenta na sua máquina e criando uma aplicação.

Nessa disciplina estamos somente apresentando as características do Laravel para você. Para se tornar um programador(a) Laravel é necessário um estudo aprofundado tanto na linguagem PHP como no framework em si. Não é necessário seguir o procedimento de instalação abaixo na sua máquina, mas estamos adicionando os passos necessários para que você tenha uma ideia de como eles funcionam.

Para instalar o Laravel você pode seguir a seção "Instalation" da documentação

oficial disponível em <https://laravel.com/docs/8.x/installation>. Vamos aqui realizar os passos para a instalação e criação de uma aplicação base Laravel.

Por ser uma multi-plataforma, o Laravel oferece alguns métodos de instalação para Windows, Mac e Linux.

A principal forma recomendada de instalação do Laravel é através da ferramenta oficial "Sail". O Sail é uma interface de linha de comando para interagir com um ambiente de desenvolvimento oficial baseado em Docker. e fornece uma bom ponto de partida para a construção de aplicações com os ambientes PHP, MySQL e Redis sem a necessidade de experiência anterior com o gerenciador de containers Docker. Se tiver curiosidade e deseja saber informações sobre o que é o Docker visite <https://www.docker.com/>

## Instalação no Windows

Como pré-requisito, para a instalação no Windows é necessário ter o Docker Desktop instalado no seu sistema (<https://www.docker.com/products/docker-desktop>). Além disso é necessário ter o Windows Subsystem for Linux 2 (WSL2) instalado e habilitado (<https://docs.microsoft.com/en-us/windows/wsl/install-win10>) e em seguida confirmar que o Docker Desktop está utilizando o WSL2 como Back-end (<https://docs.docker.com/docker-for-windows/wsl/>)

Esses passos iniciais criam uma base no seu sistema Windows para que o Sail possa ser utilizado para instalar os serviços necessários para se criar uma aplicação Laravel.

É necessário abrir um Windows Terminal (instale em <https://www.microsoft.com/en-us/p/windows-terminal/9n0dx20hk701?rtc=1&activetab=pivot:overviewtab>) com uma nova sessão WSL2. O WSL2 é uma camada de compatibilidade que contém um kernel Linux dentro do Windows, tornando a compatibilidade com diversas aplicações mais facilitada. Dentro de um Terminal WSL2, você pode instalar diversas ferramentas como se estivesse em uma máquina Linux (o que não deixa de ser verdade).

Depois de ter essas ferramentas instaladas e configuradas e com um terminal com o ambiente WSL2, rode o comando:

```
curl -s https://laravel.build/example-app | bash
```

Você pode trocar no comando acima o texto "example-app" pelo que desejar para sua aplicação.

Uma estrutura de diretórios será criada dentro da pasta que você executou o

comando. Basta entrar na pasta e executar o comando:

```
cd example-app  
  
./vendor/bin/sail up
```

O Comando "up" do Sail, quando executado pela primeira vez um ambiente da aplicação será criado na sua máquina, o que pode levar alguns minutos, mas as próximas execuções serão bem mais rápidas.

Quando o comando finalizar você terá uma aplicação Laravel rodando no seu computador e pode acessá-la pelo navegador em <http://localhost>

Para saber mais sobre o Sail e todas as suas funcionalidades visite sua documentação completa em: <https://laravel.com/docs/8.x/sail>

## Instalação no Linux

Da mesma forma que no Windows, no Linux é necessário ter a ferramenta Docker instalada para poder utilizar as facilidades do Sail na criação de aplicações Laravel. A instalação do Docker no Linux é diferente para cada tipo de distribuição. No Ubuntu 20.04, por exemplo você precisa executar as sequências de comandos abaixo.

Atualizando o sistema:

```
sudo apt update  
sudo apt upgrade
```

Nesse momento você estará atualizando seu Ubuntu. Realize essa operação seguindo as instruções apresentadas e se necessário reinicie a máquina.

Instalando o Docker

```
sudo apt install docker.io
```

Habilitando o Docker no boot:

```
sudo systemctl enable --now docker
```

Setando as permissões para seu usuário (troque o nome pelo seu):

```
sudo usermod -aG docker seu_usuario_no_ubuntu
```

Testando o Docker:

```
docker run hello-world
```

```
[isaac@gru ~]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
b8dfde127a29: Pull complete
Digest: sha256:df5f5184104426b65967e016ff2ac0bfcd44ad7899ca3bbcf8e44e4461491a9e
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

Container "hello-world" do Docker executando no Linux. Fonte: Autor.

Depois de instalar o Docker no Linux, basta executar os comandos abaixo para se criar uma nova aplicação Laravel utilizando o Sail e com todo o ambiente automaticamente criado:

Criando a sua primeira aplicação Laravel:

```
curl -s https://laravel.build/example-app | bash
```

```
[isaac@gru laravel]$ curl -s https://laravel.build/example-app | bash
Unable to find image 'laravelsail/php80-composer:latest' locally
latest: Pulling from laravelsail/php80-composer
852e50cd189d: Pull complete
0266fc315b01: Pull complete
4c8a5fa787a1: Pull complete
46fc127c1884: Pull complete
d9c4e3a55454: Pull complete
e145ce591f4d: Pull complete
8f8449135011: Pull complete
4a95979fde5b: Pull complete
bf7b24d2a998: Pull complete
3ea8de81da82: Pull complete
590a4814668f: Pull complete
85c633f36d40: Pull complete
d810a644e92b: Pull complete
3043238335fb: Pull complete
Digest: sha256:b387b05f2d55d32d9ab1b861b4bc8347f75b36ca2b259231a3359118682dabad
Status: Downloaded newer image for laravelsail/php80-composer:latest

-
| |
| |
| |
| | / _ ' | ' _ / _ ' \ \ / / _ \ | | | | |
| | _ _ | ( _ | | | ( _ | \ V / _ _ / |
| _ _ _ \ _ , _ | _ \ _ , _ | \ / \ _ _ | |
Warning: TTY mode requires /dev/tty to be read/writable.
Creating a "laravel/laravel" project at "./example-app"
Installing laravel/laravel (v8.5.22)
- Downloading laravel/laravel (v8.5.22)
- Installing laravel/laravel (v8.5.22): Extracting archive
Created project in /opt/example-app
```

Progresso da criação de uma aplicação Laravel no Linux. Fonte: Autor.

```
48/99 [=====>-----] 48%
57/99 [=====>-----] 57%
67/99 [=====>-----] 67%
77/99 [=====>-----] 77%
87/99 [=====>-----] 87%
97/99 [=====>-----] 97%
99/99 [=====] 100%    76 package suggestions were added by new dependencies, use
er suggest` to see details.
  Package sebastian/resource-operations is abandoned, you should avoid using it. No replacement was su
Generating optimized autoload files
  > Illuminate\Foundation\ComposerScripts::postAutoloadDump
  > @php artisan package:discover --ansi
  Discovered Package: facade/ignition
  Discovered Package: fideloper/proxy
  Discovered Package: fruitcake/laravel-cors
  Discovered Package: laravel/sail
  Discovered Package: laravel/tinker
  Discovered Package: nesbot/carbon
  Discovered Package: nunomaduro/collision
  Package manifest generated successfully.
  75 packages you are using are looking for funding.
  Use the `composer fund` command to find out more!
  > @php artisan key:generate --ansi
  Application key set successfully.

Application ready! Build something amazing.
Sail scaffolding installed successfully.

Please provide your password so we can make some final adjustments to your application's permissions.

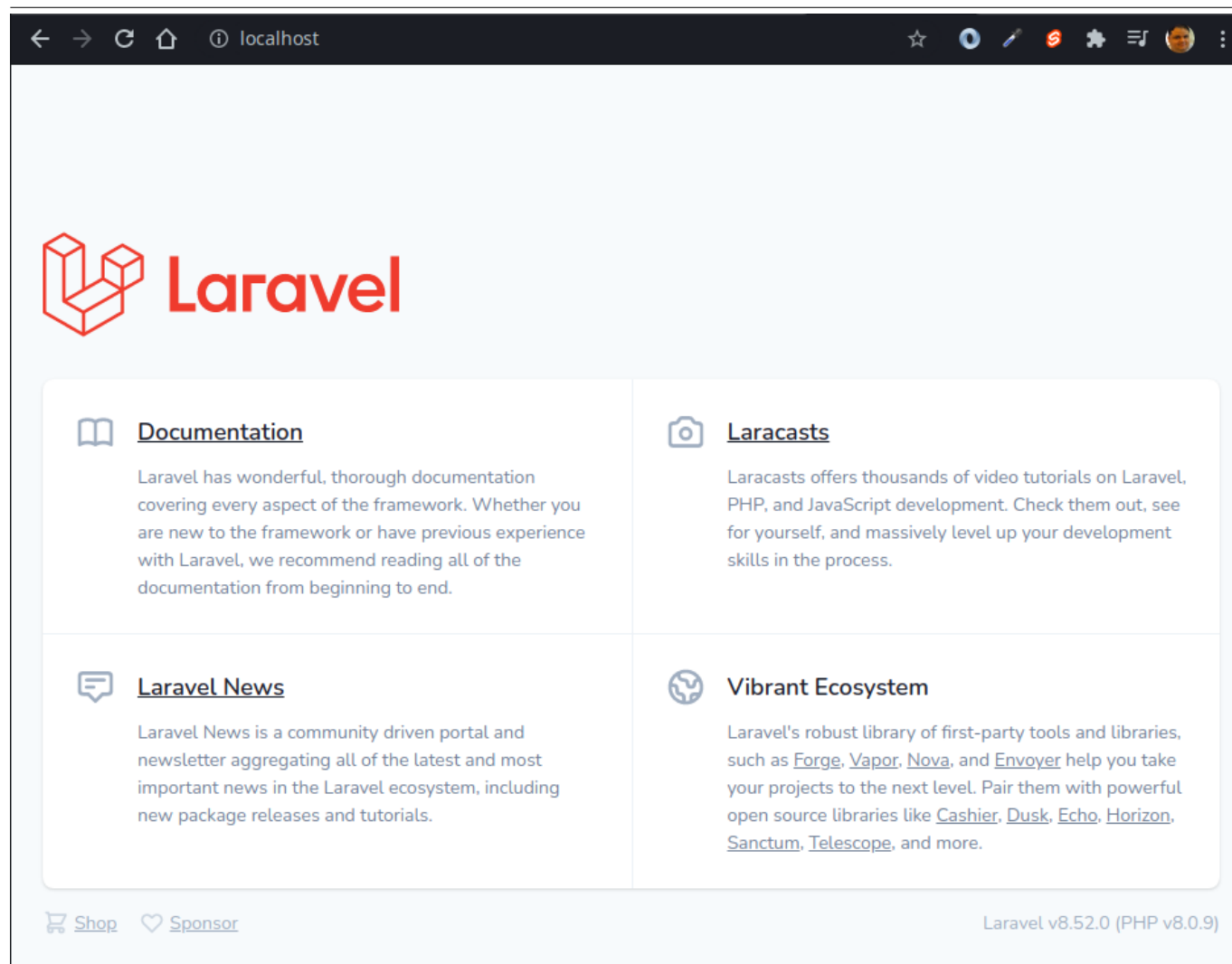
[sudo] senha para isaac:

Thank you! We hope you build something incredible. Dive in with: cd example-app && ./vendor/bin/sail up
isaac@gru laravel]$
```

*Finalização da criação de uma aplicação Laravel no Linux. Fonte: Autor.*

Para executar a aplicação criada execute os comandos:

```
cd example-app
./vendor/bin/sail up
```

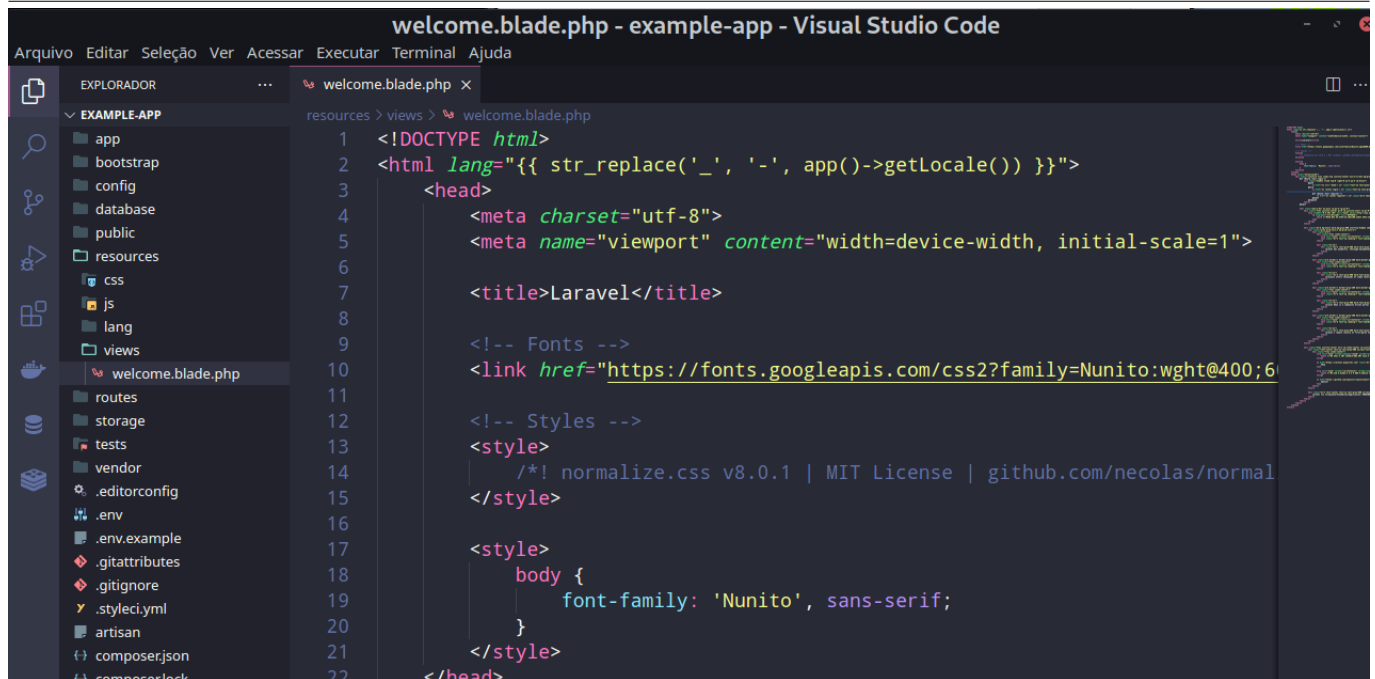


*Uma aplicação Laravel de exemplo executando em uma máquina de desenvolvimento. Fonte: Autor.*

Pronto, vemos uma aplicação Laravel com banco de dados MySQL executando em um container Docker e sendo exibida no navegador bastando acessar o endereço <http://localhost>

Note que se você por acaso já tiver o MySQL ou o MariaDB instalado e rodando no seu PC o comando acima irá falhar pois o Sail também tenta levantar um banco de dados MySQL. Para que o comando funcione você precisa antes desativar o serviço MySQL/MariaDB do seu Linux.





```
1 <!DOCTYPE html>
2 <html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
3 <head>
4     <meta charset="utf-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6
7     <title>Laravel</title>
8
9     <!-- Fonts -->
10    <link href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600" rel="stylesheet">
11
12    <!-- Styles -->
13    <style>
14        /*! normalize.css v8.0.1 | MIT License | github.com/necolas/normalize.css */
15    </style>
16
17    <style>
18        body {
19            font-family: 'Nunito', sans-serif;
20        }
21    </style>
22 </head>
```

View *welcome.blade.php* que foi exibida na página inicial da aplicação Laravel criada. Fonte: Autor

Para finalizar, você pode pressionar CTRL+C no terminal onde você executou o comando "up" do Sail para levantar sua aplicação.

Pronto. Com esses procedimentos você conseguirá criar aplicações Laravel no seu sistema Windows ou Linux e iniciar o seu aprendizado, caso decida investir nos estudos dessa ferramenta.

# Conceitos da arquitetura do Laravel

## Ciclo de vida de uma requisição

### Primeiros passos

O ponto de entrada de todas as requisições de uma aplicação Laravel é o arquivo `public/index.php`. Esse arquivo não contém muito código e é só um ponto de entrada para carregar o resto do framework.

### HTTP Kernels

Em seguida a requisição carrega o Kernel HTTP do Laravel, que está em `app/Http/Kernel.php`. Esse chamado "kernel" nada mais é que um módulo PHP que vai detectar o ambiente da aplicação e realizar outras tarefas que são necessárias antes da requisição ser efetivamente processada.

O kernel HTTP também define uma lista de "middlewares" que todas as requisições devem passar. Eles podem realizar várias tarefas como escrever e ler sessões HTTP, determinar se a aplicação está em modo de manutenção, verificar tokens, dentre

outras operações.

## Services Providers

Uma das mais importantes ações que o kernel HTTP realiza em uma requisição no Laravel é carregar os chamados "Services Providers" para sua aplicação, que estão configurados em `config/app.php`, iterando em todos eles e instanciando cada um.

Services providers são responsáveis por iniciar todos os componentes do framework como bancos de dados, filas, validações e componentes de roteamento.

## Roteamento

Um dos Services Providers mais importantes é o `RouteServiceProvider`, que carrega os arquivos de rota contidos na pasta `routes`.

Quando sua aplicação está executando e uma requisição chega nesse ponto, ela será direcionada para outro roteamento ou Controller de acordo com as configurações de roteamento.

## Finalizando a requisição

Depois de passar por todos os passos anteriores, uma rota ou controller pode retornar uma resposta, que irá realizar o envio dos dados de retorno, mas o Laravel ainda pode modificar a resposta nesse momento com novos cabeçalhos, por exemplo, se assim for configurado.

Finalmente o `index.php` chama o método "send" que envia o conteúdo final para o navegador e a jornada de uma requisição acaba.

# Características básicas do Laravel

## Roteamento

O roteamento no Laravel segue um padrão bem simples e pode ser configurado sem arquivos de roteamento complicados.

Um exemplo de uma rota no Laravel que responde na url `/ola` e retorna o conteúdo "mundo":

```
use Illuminate\Support\Facades\Route;

Route::get('/ola', function () {
    return 'Mundo';
});
```

As rotas são definidas (como visto acima) utilizando a linguagem PHP e os arquivos que contém essas rotas estão na pasta "routes" da aplicação que são automaticamente carregados pelo RouteServiceProvider. O arquivo routes/web.php definem as rotas que são para a interface web da sua aplicação (O Laravel suporta também tem outros tipos de interface, como o Console).

```
routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6  |-----
7  | Web Routes
8  |-----
9  |
10 | Here is where you can register web routes for your application.
11 | routes are loaded by the RouteServiceProvider within a group
12 | contains the "web" middleware group. Now create something great.
13 |
14 */
15
16 Route::get('/', function () {
17     return view('welcome');
18 });
19
```

*Arquivo de rotas padrão de uma aplicação exemplo Laravel. Fonte: Autor.*

Como visto na imagem acima a aplicação padrão criada pelo Laravel define no arquivo routes/web.php uma rota com o método GET (Routes::get) que responde a "/" raiz do site e recebe um segundo parâmetro uma função PHP que retorna uma view chamada "welcome". Isso faz com que o arquivo resources/views/welcome.blade.php seja processado e seu conteúdo retornado ao navegador.

Todas as rotas são definidas dessa forma, com um comando que define o método da rota, a uri (caminho na URL da requisição) e uma função PHP chamada de função callback que executa quando a rota é atingida.

Para mais informações sobre rotas no Laravel acesse:

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

## Controllers

No lugar de definir funções de callback nas suas rotas, é possível redirecionar as url

nas rotas para métodos de classes PHP especiais chamadas de controllers, tornando seu código mais organizado.

Um exemplo de um controller no Laravel:

```
<?php

namespace App\Http\Controllers;

use App\Http\Controllers\Controller;
use App\Models\User;

class UserController extends Controller
{
    /**
     * Show the profile for a given user.
     *
     * @param int $id
     * @return \Illuminate\View\View
     */
    public function show($id)
    {
        return view('user.profile', [
            'user' => User::findOrFail($id)
        ]);
    }
}
```

Veja no exemplo do controller acima que foi criado uma classe chamada `UserController` que herda (extends) de `Controller` e tem um método chamado "show" que recebe um atributo chamado `$id` (variáveis em PHP começam com \$).

Esse método "show" por sua vez retorna uma view chamada `user.profile`, que também por sua vez é um arquivo PHP dinâmico e é passado pra ele um dado chamado 'user' com o valor `User::findOrFail($id)`, que é um comando especial do laravel que busca um usuário no banco através de um determinado ID passado. Esse `User::findOrFail($id)` faz parte da ferramenta de acesso a banco de dados do Laravel chamada Eloquent ORM. A renderização da página PHP com o `user.profile` terá acesso assim aos dados do usuário para exibir o HTML apropriado.

Uma rota que redireciona para esse Controller seria assim:

```
use App\Http\Controllers\UserController;
```

```
Route::get('/user/{id}', [UserController::class, 'show']);
```

Essa rota acima (o comando `Route::get....`) recebe o primeiro parâmetro como o formato da URI que casa com a rota, nesse caso `/user/{id}` e note que o `{id}` é um parâmetro dinâmico fazendo ela casar com `/user/1`, `/user/2`, `/user/332` e assim por diante, tornando o número passado depois de `/user` o valor da variável "id" especificada. O comando recebe um segundo argumento que é uma associação chave => valor com a classe do Controller que será redirecionado o comando (`UserController::class`) e em seguida o nome do método que será executado nessa classe 'show', que receberá como parâmetro o id do usuário que está na URL.

Para estudar mais sobre Controllers no Laravel acesse:

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

## Views

As views no Laravel são por padrão arquivos PHP localizados na pasta `resources/views`. O trabalho de uma view é ser processada e retornar um HTML como resultado final que será enviada ao navegador. Abaixo, um exemplo simples de uma view no Laravel:

```
<html>
  <body>
    <h1>Hello, {{ $name }}</h1>
  </body>
</html>
```

O arquivo PHP que contém essa view de exemplo poderia estar em `resources/views/meu_perfil.blade.php`, por exemplo, e como pode ser observado é uma mistura de HTML com código de programação.

Nesse exemplo acima é possível observar que existe um código diferente do visto em HTML: `{{ $name }}`

Esse `{{ $name }}` é um comando especial na view que dá a instrução para substituir esse trecho pelo valor da variável `$name`, passada pela rota ou controller que tentou processar essa view para retornar o resultado HTML gerado por ela para o navegador. Se o valor da variável `$name` for 'Joaquim', então a palavra Joaquim ficaria no lugar de `{{ $name }}` no resultado do HTML final gerado pela view.

Um exemplo de uma rota que solicitaria o processamento e retorno dessa view mostrada acima seria:

```
Route::get('/perfil', function () {
```

```
return view('meu_perfil', ['name' => 'Joaquim']);  
});
```

Note que essa rota responde a URL /perfil e retorna o HTML retornado pelo processamento da view 'meu\_perfil' que é executada com a variável `$name = 'Joaquim'` por causa da associação chave-valor `['name' => 'Joaquim']` passada como argumento.

Para saber mais sobre views no Laravel acesse: <https://laravel.com/docs/8.x/views>

## Blade templates

As views no Laravel usam uma motor de templates chamado Blade que tem uma série de funcionalidades que permite a criação de HTMLs dinâmicos de maneira simples.

PHP é uma linguagem famosa por ser facilmente integrada com o HTML. Arquivos PHP podem ser uma mistura entre PHP e HTML e é possível usar o PHP puro para criar HTMLs dinâmicos.

O Blade trás mais funcionalidades e facilidade na criação de páginas dinâmicas no Laravel, muito similar ao que o EJS faz com relação ao Javascript e HTML no processamento do lado do servidor.

Com o Blade é possível se criar blocos condicionais (ifs), Loops, classes condicionais, usar PHP puro no HTML, etc.

Para saber mais sobre o template Blade no Laravel acesse:

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

## Bancos de dados

O Laravel suporta diversos bancos de dados e o acesso a esses dados são realizados através de uma ferramenta inclusa chamada EloquentORM.

O EloquentORM é um ORM (Object-Relational Mapper) que facilita a interação com diversos bancos de dados. Quando você está utilizando o Eloquent, cada tabela tem um "Model" correspondente, que é utilizado para interagir com essa tabela, permitindo você obter os dados da tabela, inserir, remover e atualizar.

O Laravel tem uma ferramenta de linha de comando que permite, dentre outras coisas, que você crie novos Models através de comandos do terminal.

Para saber mais sobre o EloquentORM visite: <https://laravel.com/docs/8.x/eloquent>

# Conclusão

Nessa aula somente demos uma visão geral do que é se trata o Laravel e, como você viu, ele não é uma ferramenta pequena. Trata-se de uma coleção super completa de funcionalidades que, quando dominadas, dá ao programador PHP muito poder para criar aplicações profissionais com muitas funções avançadas já prontas e testadas por milhares de empresas.

Para conhecer mais sobre o Laravel e sobre a linguagem PHP visite:

- PHP: <https://php.net>
- Laravel <https://laravel.com/>