

Laravel

Jaime Vendrame Filho

 /jaimevendrame

 /in/jaimevendrame

 /jaime.vendrame

 jaime.vendrame@gmail.com



+55 44 99974-8008

Agenda

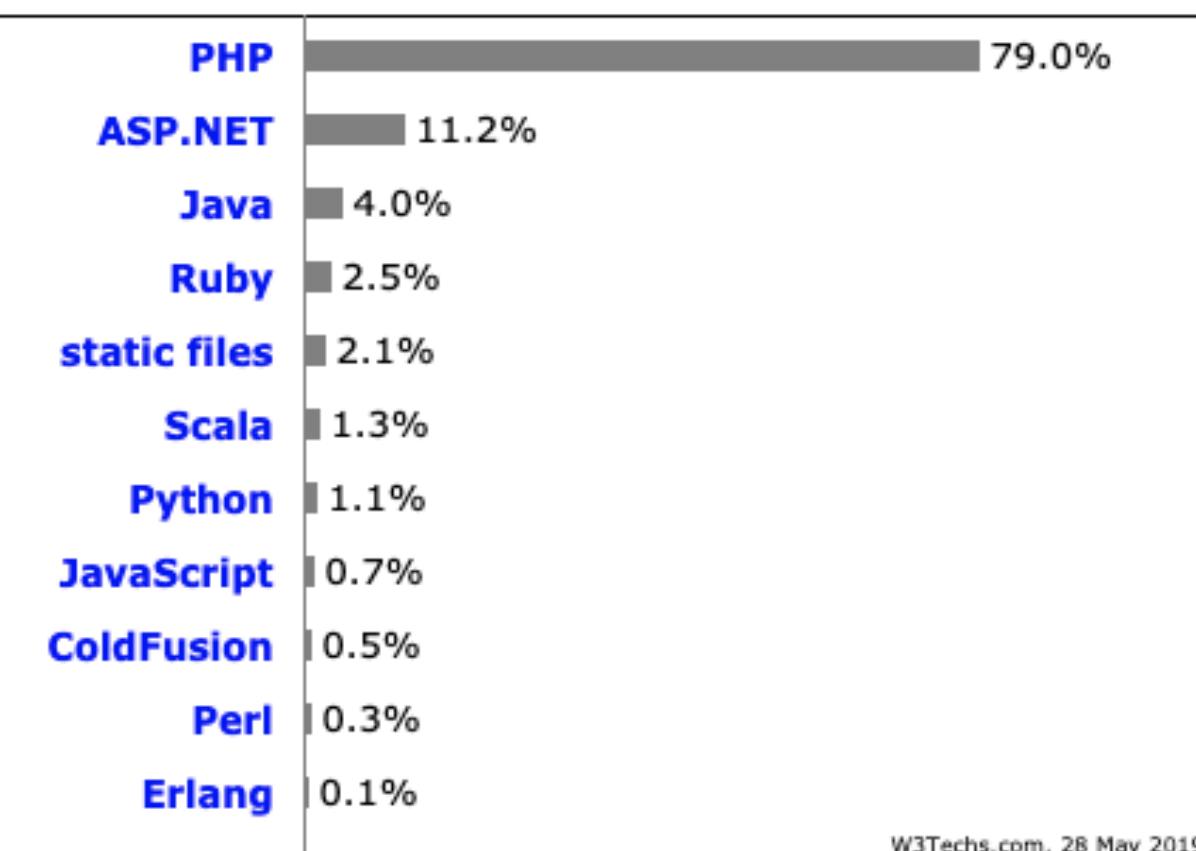
- Apresentação do Módulo
- Ambiente de Desenvolvimento
 - Requisitos
- Instalação e Configuração Laravel
 - Criando nosso Virtual Host
- Introdução ao Laravel
 - Estrutura de pastas
 - Routes
 - Controllers
 - Views & Templates
- Conhecendo projeto do Blog
- Integração layout Home & Dashboard
- Laravel avançado
 - Models & Migrations
- CRUD Usuários
- CRUD Categorias
- Auth (Autenticação personalizada)
- CRUD Post
- Site

O melhor framework PHP?



Primeiramente ...



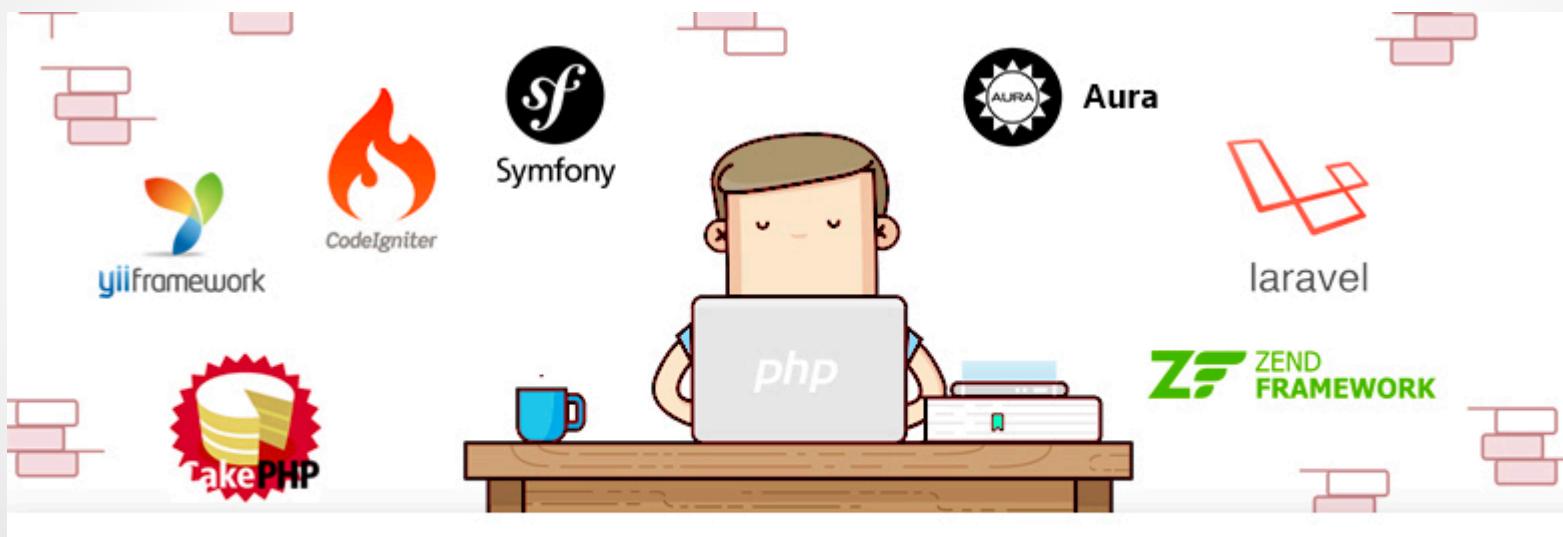


W3Techs.com, 28 May 2019

Percentages of websites using various server-side programming languages

Note: a website may use more than one server-side programming language

Segundamente ...



● laravel Termo de pesquisa	● Symfony Termo de pesquisa	● Codeigniter Software	● CakePHP Termo de pesquisa	● Zend Termo de pesquisa
--	---	---	---	---

Todo o mundo ▾

Últimos 12 meses ▾

Todas as categorias ▾

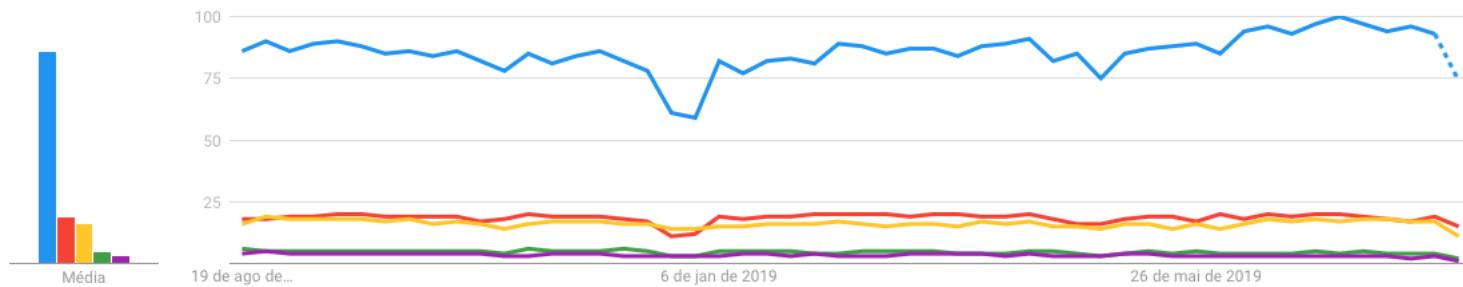
Pesquisa na Web ▾



Observação: Esta comparação contém Assuntos e Termos de pesquisa que são medidos de maneiras diferentes.

[SAIBA MAIS](#)

Interesse ao longo do tempo ?



Fonte: <https://trends.google.com/trends/explore?q=laravel,Symfony,%2Fm%2F02qgdkj,CakePHP,Zend>

[Code](#)[Pull requests 0](#)[Projects 0](#)[Security](#)[Insights](#)

A PHP framework for web artisans <https://laravel.com>

[php](#)[framework](#)[laravel](#)[6,033 commits](#)[11 branches](#)[94 releases](#)[503 contributors](#)

Branch: master ▾

[New pull request](#)[Find File](#)[Clone or download ▾](#) taylorotwell formatting

Latest commit bb43372 2 days ago

 app

Move TrustProxies to highest priority - fixes maintenance mode ip whi...

last month

 bootstrap

clean variable name

9 months ago

 config

formatting

2 days ago

 database

Apply fixes from StyleCI (#5006)

4 months ago

 public

Merge branch 'master' into develop

6 months ago

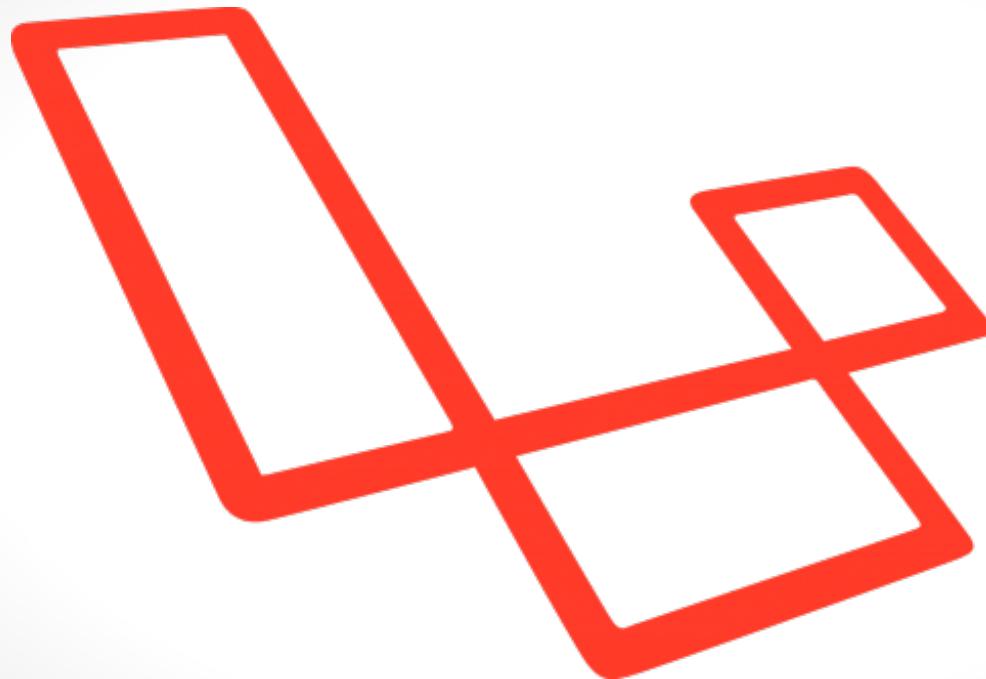
 resources

Add missing trailing semicolon

9 days ago



54.190



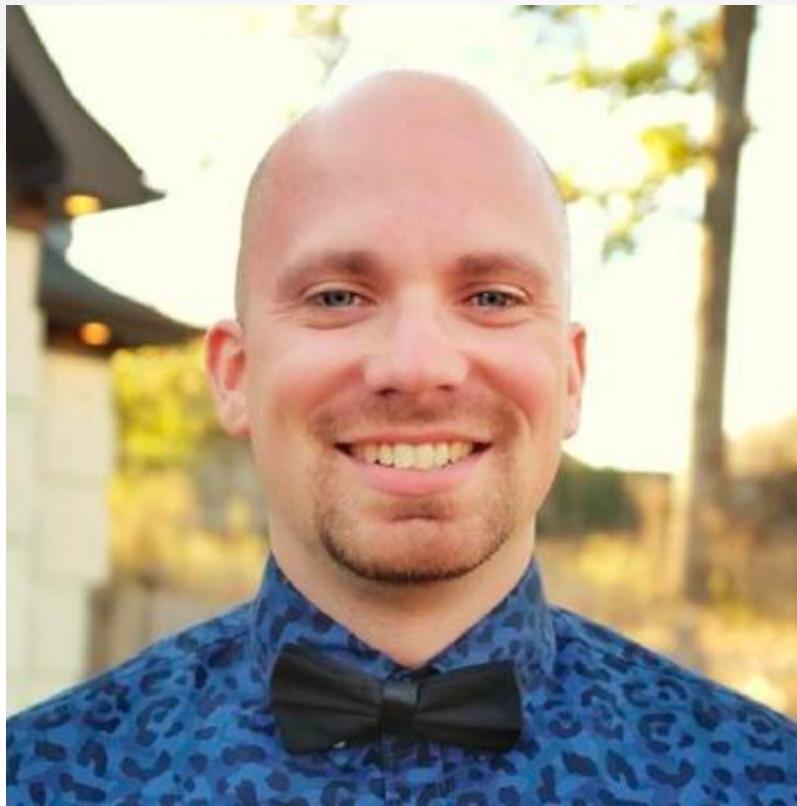
MENOS PREOCUPAÇÃO
COM A FERRAMENTA

MAIS FOCO NO PRODUTO

MENOS
ESTRESSE



MAIS DINHEIRO



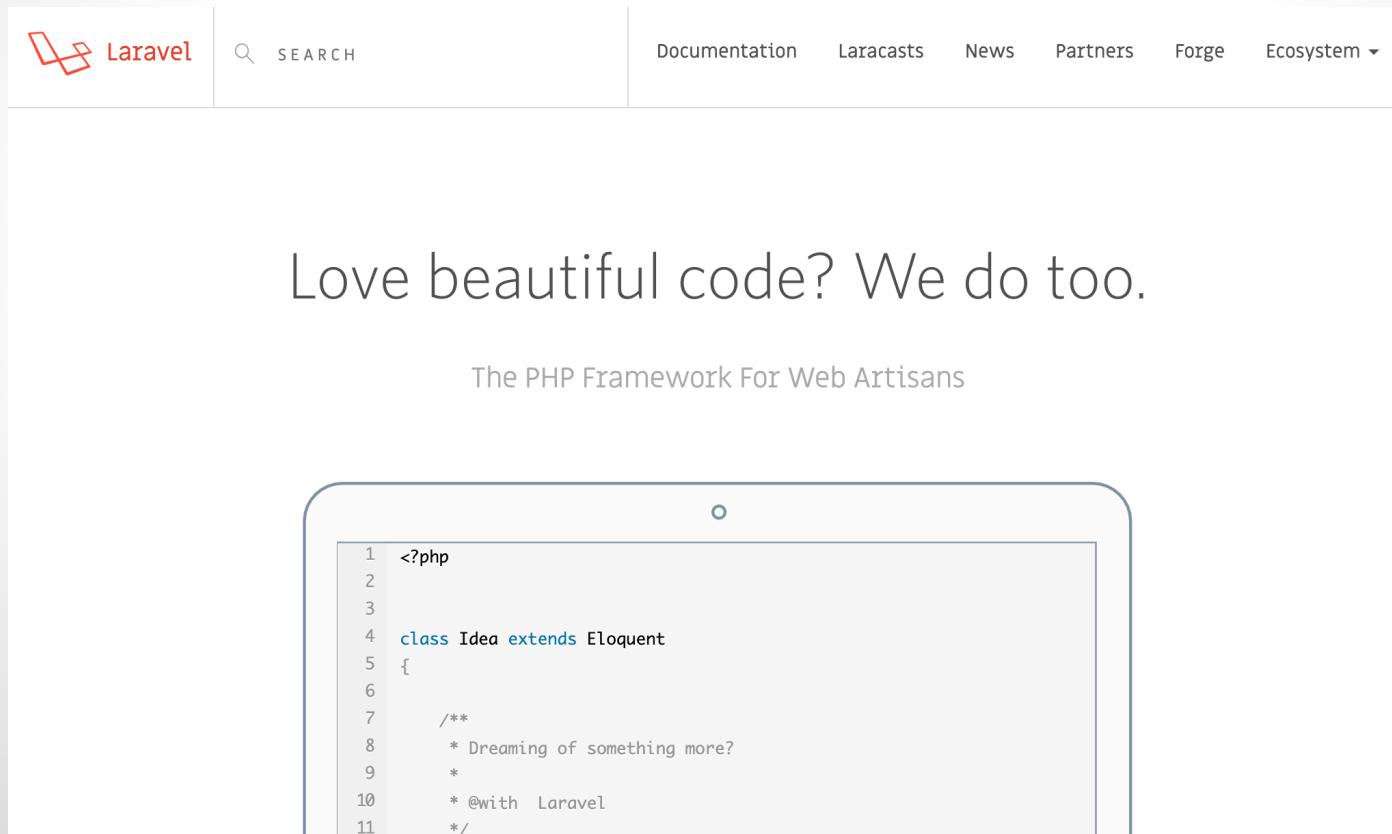
Taylor Otwell

- O PHP é uma das mais antigas e populares linguagens de desenvolvimento web, e o Laravel é seu framework mais popular. Graças Taylor Otwell, que desenvolveu este mágico framework, que até momento emplaca 52K estrelas no GitHub.
- Com objetivo principal de permitir o desenvolvimento estruturado e rápido, sendo ainda livre e de código aberto. Os desenvolvedores preferem o Laravel a outros frameworks por causa do desempenho, recursos e escalabilidade que ele oferece. Ele segue o Model View Controller (MVC), que o torna mais útil que o PHP.
- Ele tenta eliminar a dor do desenvolvimento, facilitando tarefas comuns usadas na maioria dos projetos da Web, como autenticação, roteamento, sessões e armazenamento em cache. Ele possui uma arquitetura exclusiva, onde é possível que os desenvolvedores criem sua própria infraestrutura projetada especificamente para sua aplicação. O Laravel é usado não só para grandes projetos, mas também para projetos pequenos.

O que é o Laravel?

- O Laravel é um framework em PHP baseado no conceito de RAD (Rapid Application Development).
- Desenvolvido por Taylor Otwell.
- Possui código aberto e uma comunidade ativa.
- Utiliza o composer, fornecendo todo o poder das demais bibliotecas escritas para PHP.

- **Gosta de código bonito?** Quem utiliza Laravel também gosta! Como o próprio slogan diz: “*O framework PHP para artesões da web*”.



O que é um framework?

- Framework é um “esqueleto”, um modelo que nos propicia o essencial para desenvolver alguma coisa.



A maioria das aplicações web precisa de:

- Login de usuários;
- CRUD'S;
- Integrações com redes sociais;
- Operações de banco de dados;
- Envio de e-mails;
- Notificações para os usuários.

Mas, porque devo usar Laravel? O que torna ele tão bom assim?

Ok! Já sabemos que ele é o framework mais badalado da atualidade, para muitos, isso já é um baita argumento para começar a utiliza-lo, para outros, nem tanto...

Sendo assim, vou apresentar alguns pontos:

COMPOSER

O Laravel utiliza o [Composer](#) para gerenciar suas dependências, algo que praticamente toda aplicação PHP moderna faz. O Composer é um incrível gerenciador de dependências, uma ferramenta que permite gerenciar, de forma fácil, os pacotes de terceiros da sua aplicação.

Packagist O Repositório de Pacotes PHP

Squeaky toy Enviar Criar Conta em portugues-Brasil assinar em



mail

O Packagist é o repositório principal do Composer . Agrega pacotes PHP públicos instaláveis com o Composer.

	PHP	↓ 111 815 068	Tipo de pacote
swiftmailer / swiftmailer	★ 7 987	aimeos-extensão antaresprojeto-componente antaresproject-module api aplicação asgard-module authserver-plugin backbee-bundle behat-extension pacote cakephp-plugin civicrm-extension codeigniter-library codeigniter-third-party componente	
arma de correio / mail gun-php	PHP ↓ 4 678 730	94 3 1 2 6 1 1 2 4 3 94 2 2 4 6	
Desenhado / correio chimp-api	★ 1 678	Mostre mais...	
php-imap / php-imap	PHP ↓ 910 873	Tag	
pacote hwi / oauth	★ 1 012	5 313 773	



Dependency Manager for PHP

composer require monolog/monolog

Documentação

Até na documentação, o Laravel se destaca! Ela é bastante intuitiva e você consegue encontrar de forma fácil praticamente tudo que precisa saber para começar e se aprofundar em todos os recursos disponíveis pelo framework.

<http://laravel.com/docs>

Laravel Nova is now available! [Get your copy today!](#)

 [Laravel](#)  [SEARCH](#) Documentation Laracasts News Partners Forge Ecosystem ▾ [5.7 ▾](#)



Marketing tools to help you grow your business, your way.
ADS VIA CARBON

Expand All

Prologue

Getting Started

Architecture Concepts

The Basics

Frontend

Security

Digging Deeper

Installation

Installation

- # Server Requirements
- # Installing Laravel
- # Configuration

Web Server Configuration

- # Pretty URLs

Installation

 Laracasts provides a [free, thorough introduction to Laravel](#) for newcomers to the framework. It's a great place to start your journey.

Server Requirements

The Laravel framework has a few system requirements. Of course, all of these requirements are satisfied by the [Laravel Homestead](#) virtual machine, so it's highly recommended that you use Homestead as your local Laravel development environment.

However, if you are not using Homestead, you will need to make sure your server meets the following requirements:

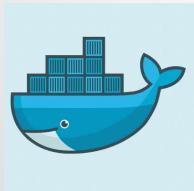
Empresas quem utilizam o Laravel

- Cielo
- Catho
- BTG Pactual
- Leroy Merlin
- CI (Companhia do Intercâmbio)
- Senac

AMBIENTE DE DESENVOLVIMENTO

Ambiente de desenvolvimento

- Existem diversas opções para criação de um ótimo ambiente de desenvolvimento Laravel, como por exemplo:
 - Containers: Docker e Laradocker
 - Boxs: Vagrant e Homestead
 - Servidores independentes: XAMPP, MAMP, LAMP ...



Laradock



VAGRANT



XAMPP



Ambiente de desenvolvimento

- Você deve levar em consideração o seu sistema operacional e as limitações de seu hardware.
- Outros aspecto bacana é que você tem a portabilidade de criar e recriar esses ambientes em qualquer lugar de forma simples e descomplicada bastando apenas ter a ‘receita mágica’ do seu ambiente.

Ambiente de Desenvolvimento

- Neste projeto utilizaremos o **LARADOCK**, por uma questão exclusivamente de padronização.
- **LARADOCK** é um importante e reconhecido projeto criado pela comunidade do PHP com configurações prontas para trabalhar com Laravel, utilizando o Docker.

Ambiente de desenvolvimento

- Com este projeto é possível facilmente trabalhar com qualquer banco de dados e ainda tem liberdade para utilizar várias ferramentas a medida que precisar.
- O laradock já conta com configurações para nginx, apache, php-fpm, mysql, mariadb, postgres, redis, mongo, e várias outras configurações.
- Em resumo, com apenas um único comando já é possível subir todos estes serviços ou subir cada um a medida que precisar.

Ambiente de desenvolvimento

Pré-requisitos:

- [Docker](#)
- [Git](#)

Ambiente de desenvolvimento

Hands-on

- Entre os vários recursos disponíveis no Laradock vamos subir os serviços: NGINX, PHP, PHPMYADMIN e MySQL.

Ambiente de desenvolvimento

Hands-on

1 – Crie um diretório em qualquer parte do seu S.O. para receber os arquivos de configuração do Docker

2 – Clone o repositório:

```
git clone https://github.com/Laradock/laradock.git laradock
```

Isso vai clonar o repositório para dentro de /laradock

Ambiente de desenvolvimento

Hands-on

3 – Acesse o diretório:

```
cd laradock
```

4 – Crie o arquivo .env (arquivo de definições) a partir do arquivo env-exemple:

```
cp env-example .env
```

Ambiente de desenvolvimento

Hands-on

5 – Subir os serviços que vamos precisar:

```
docker-compose up -d nginx mysql phpmyadmin workspace
```

6 - Ao subir os containers pode verificar o que está rodando, através deste comando:

```
docker ps
```

Ambiente de desenvolvimento

Rodando uma Aplicação PHP com Laradock

Os arquivos do laradock estão em laradock, e nossas aplicações ficarão no nível anterior.

Crie um diretório novo no mesmo nível do diretório laradock. Pode chamar este diretório (pasta) com o nome que quiser, em nosso exemplo vai se chamar app/

laradock/

app/

Dentro do diretório app/ crie um arquivo index.php com este conteúdo (app/index.php):

```
<?php  
phpinfo();
```

Ambiente de desenvolvimento

Rodando uma Aplicação PHP com Laradock

Agora precisamos alterar as configurações do NGINX para reconhecer essa nova aplicação em PHP.

Acesse o arquivo configuração do nginx e altere o Document Root para app/, o arquivo de configuração do nginx fica em laradock/nginx/sites/default.conf

Altere o Document Root assim:

```
root /var/www/app
```

Ambiente de desenvolvimento

Rodando uma Aplicação PHP com Laradock

Ao fazer isso já vai reconhecer o novo Document Root default, agora basta reiniciar os container do docker.

Dentro da pasta laradock (cd laradock/) reinicie os serviços:

```
docker-compose restart
```

Após fazer isso basta acessar a nova aplicação: <http://localhost>

Ambiente de desenvolvimento

Instalando e Rodando Aplicação Laravel com Laradock

- É possível ter diversas aplicações rodando simultaneamente com o Laradock (pode ser aplicações PHP puras, laravel ou até mesmo outros frameworks PHP).
- Primeiro passo é acessar a pasta do laradock (cd laradock/)
Acessar o container workspace para rodar comandos do PHP e Composer.

```
docker-compose exec workspace bash
```

Ambiente de desenvolvimento

Instalando e Rodando Aplicação Laravel com Laradock

- Ao rodar este comando pode observar que o terminal irá mudar para algo parecido com isso (está dentro do container workspace):
root@f59a0bd8d58b:/var/www/
- Aqui você tem acesso a um terminal para rodar comandos UNIX, como por exemplo: ls, mkdir, touch e etc.

```
root@f59a0bd8d58b:/var/www/
```

Ambiente de desenvolvimento

Instalando e Rodando Aplicação Laravel com Laradock

- Rode o comando ls para ver os arquivos deste diretório. Ao rodar vai perceber que está acessando o nível anterior ao arquivos do laradock/, é possível ver até mesmo a pasta do laradock.
- Agora que estamos acessando o terminal com container workspace podemos instalar o Laravel. Certifique-se que o comando do composer está funcionando com sucesso:

```
composer
```

Ambiente de desenvolvimento

Instalando e Rodando Aplicação Laravel com Laradock

- Para instalar o laravel, rode este comando:

```
composer create-project --prefer-dist laravel/laravel webdev-laravel
```

- Agora basta aguardar o composer baixar o framework laravel e todas as suas dependências.
- Ao terminar terá um novo diretório (pasta) chamado webdev-laravel

Ambiente de desenvolvimento

Para sair do container workspace basta pressionar as teclas CTRL + P + Q simultaneamente.

Agora que já instalou o laravel podemos criar um virtualhost (nome de endereço local) para acessar.

Vamos a configuração:

Crie um novo arquivo em laradock/nginx/sites/webdev-laravel.conf

Lembrando que este arquivo é apenas para definição do virtualhost e aplicação laravel correspondente. É recomendado que deixe o nome deste arquivo semelhante ao nome do virtualhost que vai trabalhar.

Ambiente de desenvolvimento

- Coloque este conteúdo no arquivo **webdev-laravel.conf**

```
server {  
  
    listen 80;  
    listen [::]:80;  
  
    server_name blog.webdev;  
    root /var/www/webdev-laravel/public;  
    index index.php;  
  
    location / {  
        try_files $uri $uri/ /index.php$is_args$args;  
    }  
  
    location ~ \.php$ {  
        try_files $uri /index.php =404;  
        fastcgi_pass php-upstream;  
        fastcgi_index index.php;  
        fastcgi_buffers 16 16k;  
        fastcgi_buffer_size 32k;  
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;  
        include fastcgi_params;  
    }  
}
```

Ambiente de desenvolvimento

- As configurações mais importantes são:
server_name: nome do virtualhost;
root: caminho onde está a aplicação laravel;
- Após fazer isso precisa reiniciar os containers do docker para reiniciar o nginx para ler o novo arquivo de configuração.
Com o terminal e dentro do diretório laradock/ rode:

```
docker-compose restart
```

Ambiente de desenvolvimento

- Agora falta um pequeno detalhe para funcionar, criar a configuração de virtualhost localmente no próprio S.O.
Abra o arquivo de configuração de Virtualhost
- No Linux e Mac fica em /etc/hosts
No Windows fica C:/Windows/System32/drivers/etc/hosts
- E adicione a configuração para quando acessar o endereço blog.webdev não busque fora da maquina e sim no próprio localhost:
127.0.0.1 blog.webdev
- Feito isso agora basta acessar <http://blog.webdev/> que vai acessar a nossa aplicação laravel

Ambiente de desenvolvimento

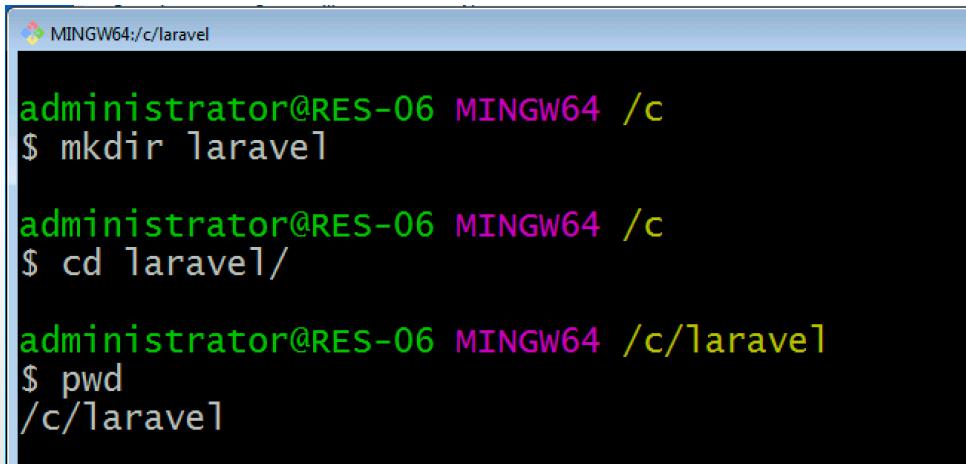
VAGRANT

Ambiente de Desenvolvimento

- **REQUISITOS:**
 - Vagrant <https://www.vagrantup.com/downloads.html>
 - VirtualBox <https://www.virtualbox.org/wiki/Downloads>
 - GIT <https://git-scm.com/download/>
 - VISUAL STUDIO CODE <https://code.visualstudio.com/>
 - PUTTY somente para usuários Windows.
<https://www.putty.org/>
- **Obs: Devem ser instaladas a ultima versão, desde que não exista outro ambiente desenvolvimento baseado em box instalando anteriormente.**

Ambiente de Desenvolvimento

- CRIANDO SUA WORKSPACE
 - Através seu terminal ou git-bash(windows) acessem a raiz de sua unidade disco e crie o diretório laravel.



```
MINGW64:/c/laravel
administrator@RES-06 MINGW64 /c
$ mkdir Laravel

administrator@RES-06 MINGW64 /c
$ cd Laravel/

administrator@RES-06 MINGW64 /c/laravel
$ pwd
/c/laravel
```

Ambiente de Desenvolvimento

- Configurações para criar o ambiente perfeito para o desenvolvimento web com PHP.
 - <https://github.com/jaimevendrame/vagrant-setup-php>

Ambiente de Desenvolvimento

- Clonando repositório:

```
administrator@RES-06 MINGW64 /c/laravel
$ git clone https://github.com/jaimevendrame/vagrant-setup-php.git webdev
```

- Rodando ambiente de desenvolvimento pela primeira vez.
 - Acesse o novo diretório webdev
 - Digite o comando: vagrant up
 - **Obs: O processo demora bastante na primeira vez.**
- Testando ...
 - Em seu browser acesse: <http://192.168.33.10> ou <http://localhost:8080>.

Instalação e Configuração Laravel

- Qual versão escolher do Laravel 5.x?
 - Quanto mais recente a versão mais recursos ela possui, e claro, melhor fica o desenvolvimento. Certo?
 - Em partes. Um ponto muito importante no momento de escolher uma versão é avaliar se ela é L.T.S., ou seja, se o tempo de suporte pra essa versão é longa. O Laravel atualmente possui 2 duas versões L.T.S, a 5.1 e a 5.5 ([veja a tabela de versões L.T.S. do Laravel 5.x](#))
 - **NOTA:** Em resumo, uma versão L.T.S. significa que a equipe responsável pelo projeto Laravel irá dā manutenção, ou seja, corrigir eventuais bugs e falhas de segurança.

Instalação e Configuração Laravel

- **Via Composer Create-Project**

Alternativamente, você também pode instalar o Laravel emitindo o comando Composer no seu terminal: **create-project**

```
composer create-project --prefer-dist laravel/laravel blog "5.5.*"
```

Instalação e Configuração Laravel

- **Criando nosso Virtual Host**

Com nosso projeto criado, agora precisamos criar e configurar nosso host virtual.

Primeiramente usando o Putty, abra o arquivo **000-default.conf**, que se encontra no diretório **/etc/apache2/sites-available**.

Instalação e Configuração Laravel

```
$> sudo nano /etc/apache2/sites-available/000-default.conf

<VirtualHost *:80>
    ServerName blog.webdev
    DocumentRoot /var/www/html/blog/public
        <Directory /var/www/html/blog/public>
            AllowOverride all
        </Directory>
</VirtualHost>
```

Ctrl X – Sair e Salvar
Y – Confirma
Enter

```
$> sudo nano /etc/hosts
127.0.0.1 blog.webdev
```

```
$> sudo service apache2 restart
```

Instalação e Configuração Laravel

- Configurar host local
- Windows ->
C:\Windows\System32\drivers\etc\hosts
- Mac/Linux -> /etc/hosts

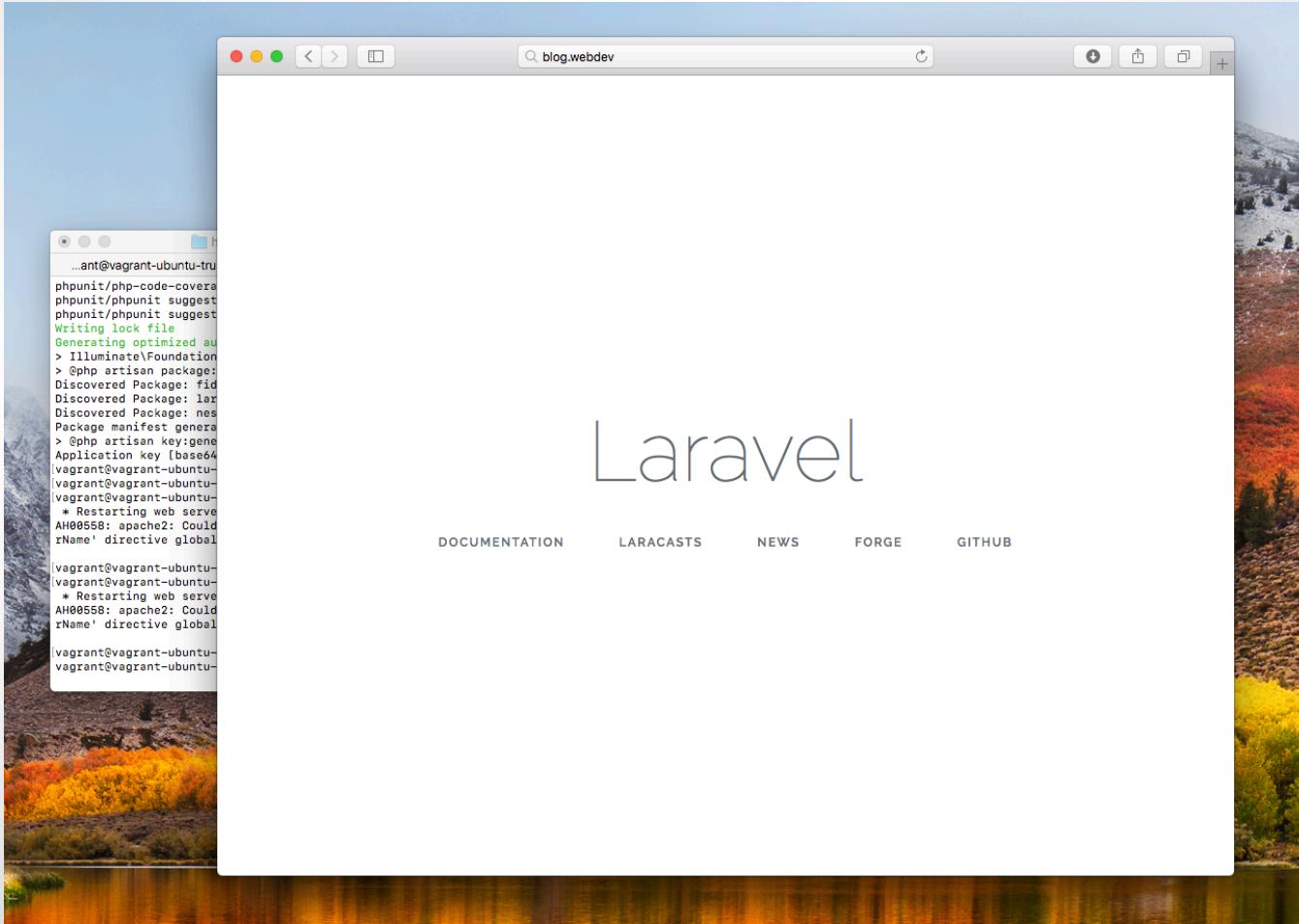
Instalação e Configuração Laravel

GNU nano 2.0.6

File: /etc/hosts

```
##  
# Host Database  
#  
# localhost is used to configure the loopback interface  
# when the system is booting. Do not change this entry.  
##  
127.0.0.1      localhost  
255.255.255.255 broadcasthost  
::1            localhost  
192.168.33.10  blog.webdev
```

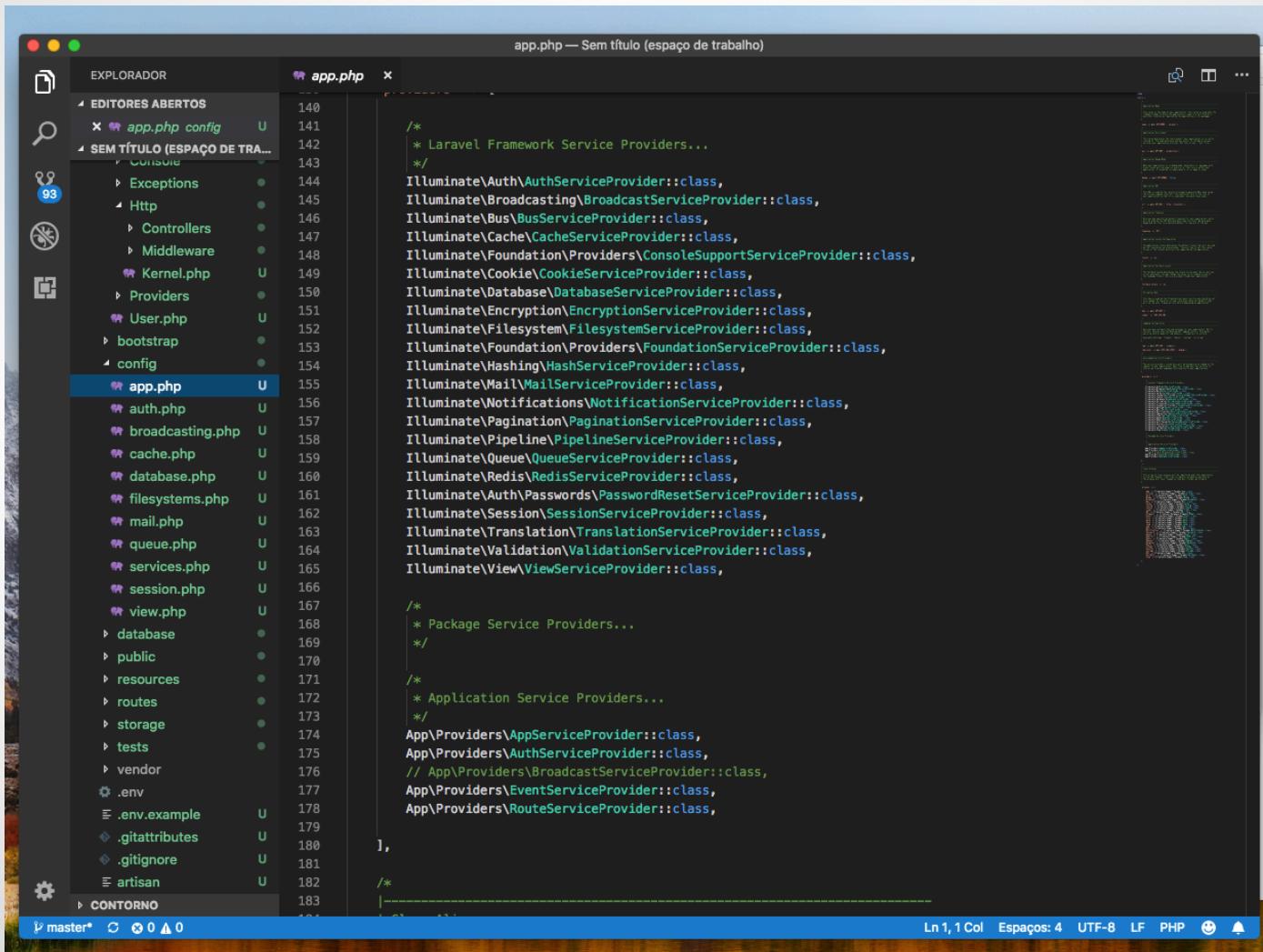
Instalação e Configuração Laravel



Introdução ao Laravel

Estrutura de pastas

Estrutura de pastas



The screenshot shows a code editor window titled "app.php — Sem título (espaço de trabalho)". The left sidebar displays a file tree with the following structure:

- EDITORES ABERTOS
- SEM TÍTULO (ESPAÇO DE TRABALHO)
- CONSOLE
- Exceptions
- Http
- Controllers
- Middleware
- Kernel.php
- Providers
- User.php
- bootstrap
- config
- app.php
- auth.php
- broadcasting.php
- cache.php
- database.php
- filesystems.php
- mail.php
- queue.php
- services.php
- session.php
- view.php
- database
- public
- resources
- routes
- storage
- tests
- vendor
- .env
- .env.example
- .gitattributes
- .gitignore
- artisan
- CONTORNO

The main pane contains the following PHP code, which is the bootstrapping code for a Laravel application:

```
/*
 * Laravel Framework Service Providers...
 */

Illuminate\Auth\ServiceProvider::class,
Illuminate\Broadcasting\BroadcastServiceProvider::class,
Illuminate\Bus\BusServiceProvider::class,
Illuminate\Cache\CacheServiceProvider::class,
Illuminate\Foundation\Providers\ConsoleServiceProvider::class,
Illuminate\Cookie\CookieServiceProvider::class,
Illuminate\Database\DatabaseServiceProvider::class,
Illuminate\Encryption\EncryptionServiceProvider::class,
Illuminate\Filesystem\FilesystemServiceProvider::class,
Illuminate\Foundation\Providers\FoundationServiceProvider::class,
Illuminate\Hashing\HashServiceProvider::class,
Illuminate\Mail\MailServiceProvider::class,
Illuminate\Notifications\NotificationServiceProvider::class,
Illuminate\Pagination\PaginationServiceProvider::class,
Illuminate\Pipeline\PipelineServiceProvider::class,
Illuminate\Queue\QueueServiceProvider::class,
Illuminate\Redis\RedisServiceProvider::class,
Illuminate\Auth\Passwords\PasswordResetServiceProvider::class,
Illuminate\Session\SessionServiceProvider::class,
Illuminate\Translation\TranslationServiceProvider::class,
Illuminate\Validation\ValidationServiceProvider::class,
Illuminate\View\ViewServiceProvider::class,

/*
 * Package Service Providers...
 */

/*
 * Application Service Providers...
 */
App\Providers\AppServiceProvider::class,
App\Providers\AuthServiceProvider::class,
// App\Providers\BroadcastServiceProvider::class,
App\Providers\EventServiceProvider::class,
App\Providers\RouteServiceProvider::class,
```

Estrutura de pastas

- No diretório raiz podemos ver que temos vários arquivos e outros diretórios;
- **Diretório “app”** - nele temos todos arquivos de nossa aplicação, podemos ver vários outros diretórios, mas iremos focar no diretório Http, pois nele temos o nosso arquivo routes.php, que é onde definimos todas as rotas de nossa aplicação. Temos também vários diretórios, entre eles, o “Controllers”, que é onde ficam todos os controllers de nossa aplicação.
- **Diretório config** - onde configuramos todo nosso projeto. Nele temos o arquivo app.php, que é onde podemos configurar várias variáveis de nossa aplicação como local, fuso-horário, os providers e definir os aliases de nossa aplicação. Ainda no diretório config temos o arquivo database.php que é onde definimos todas as configurações sobre a conexão com o Banco de Dados. Ainda temos vários outros arquivos como o mail.php para definir configurações de e-mail de nossa aplicação;
- **Diretório database** - onde definimos três tipos de arquivos importantes:
 - **Migrations**: uma das ferramentas mais poderosas do Laravel para definir, através de arquivos PHP, como nosso Banco de Dados deve ser criado. Através do Artisan, que é a interface de linha de comando do Laravel, criamos, alteramos e excluímos tabelas do nosso Banco de Dados de forma fácil, rápida e intuitiva;
 - **Seeds**: com esses arquivos podemos popular as tabelas do Banco de Dados com os dados que queremos para testes de forma fácil e rápida;
 - **Factories**: essa ferramenta foi introduzida na versão 5.1 do Framework, para popular as tabelas do [Banco de Dados](#) com dados criados de forma automática e randômica, permitindo incluir uma grande massa de dados de forma bem rápida para criação de testes;

Estrutura de pastas

- **Diretório “public”** - possui os arquivos .htaccess e index.php, que é o roteador de nossa aplicação. Ele recebe as requisições, as trata através do kernel e retorna para os usuários as respostas.
- **Diretório “resources”** - nele temos três diretórios importantes:
 - **Assets**: usado para armazenarmos todos arquivos de estilo (CSS, LESS, SASS, etc.), scripts (JavaScript, etc.), imagens e outros recursos necessários para nossa aplicação
 - **Lang**: usado para armazenarmos os arquivos de tradução para nossa aplicação;
 - **Views**: usado para armazenar os arquivos de nossa camada de visualização;
- **Diretório “tests”** - nele salvamos todos os testes de nossa aplicação;
- **Diretório “vendor”** - nele temos todos os arquivos de terceiros utilizados em nosso projeto, ou seja, bibliotecas, plugins, etc;
- **Arquivo “.env”** - nele definimos várias configurações de nossa aplicação, como os dados de configuração da conexão do banco de dados e a configuração de e-mails.
- **Arquivo gulpfile.js** - nele temos acesso a API do Elixir, que é usado para definir tarefas do Gulp para nossa aplicação.

Introdução ao Laravel

Sistema de Rotas

Ponto de Entrada: rotas

- As rotas são o ponto de entrada para uma aplicação Laravel.
- Através das rotas direcionamos as requisições dos nossos usuários para os controladores corretos.
- Podemos utilizar closures do PHP no arquivo de rotas PARA FINS DE TESTES, pois esta não é uma boa prática.

```
Route::get('/', function () {
    return view('welcome');
});
```

Sistema de Rotas

Com o Laravel, podemos trabalhar facilmente com o conceito de rotas. De forma bem simplória, as rotas fazem o mapeamento da URL digitada no navegador para alguma ação dentro da sua aplicação.

```
Route::get('hello', function ()  
{  
    return 'Hello World';  
});
```

Podemos registrar rotas que respondem a qualquer verbo HTTP:

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

Sistema de Rotas

- Além disso, o sistema de roteamento nos permite trabalhar com:
 - **Parâmetros de rotas(Opcionais ou não);**

```
//Exemplo de rota com parâmetro opcional.  
Route::get('welcome/{name?}', function ($name = 'visitante') {  
    return "Seja bem vindo $name!";  
});
```

Restrições de parâmetros com expressões regulares

```
Route::get('welcome/{name}', function ($name) {  
    //  
})->where('name', '[A-Za-z]+');
```

Sistema de Rotas

- Além disso, o sistema de roteamento nos permite trabalhar com:
 - Parâmetros de rotas(*Opcionais ou não*);

```
//Exemplo de rota com parâmetro opcional.
```

```
Route::get('welcome/{name?}', function ($name = 'visitante') {  
    return "Seja bem vindo $name!";  
});
```

Restrições de parâmetros com expressões regulares

```
Route::get('welcome/{name}', function ($name) {  
    //  
    })->where('name', '[A-Za-z]+');
```

Introdução ao Laravel

Controllers

Artisan Console

- **Artisan Console**

- O Artisan é uma interface de linha de comando que fornece vários comandos para facilitar o desenvolvimento da aplicação. Para visualizar todos os comandos disponíveis basta digitar no terminal:

```
php artisan list
```

Controller

- Controllers são responsáveis por “controlar” a nossa aplicação.
- Eles recebem a requisição e a direcionam para o local correto.
- É uma boa prática manter o controller com pouco código, deixando-o mais limpo.

Controllers

- **O Fluxo de Nossa Aplicação**

As aplicações criadas com o Framework Laravel seguem um fluxo baseado na arquitetura MVC. Primeiramente devemos definir as rotas para que nossos usuários possam fazer requisições e criar os Controllers necessários para tratar as requisições de nossos usuários.

```
Route::get('/', 'PagesController@index');
```

Assim definimos que quando nosso usuário acessar a rota /, ou seja, a rota inicial de nossa aplicação, o Controller chamado PagesController.php ficará responsável por tratar a requisição de nosso usuário e retornar uma resposta a ele.

```
php artisan make:controller PagesController
```

Controllers

Depois de executar o código, abra o diretório Controllers em app\Http e veja que um novo arquivo foi criado: o PagesController.php. Como padrão do Framework, quando criamos um novo Controller, damos um nome a ele e logo à frente colocamos a palavra “Controller”.

Abra o arquivo que foi criado e verá que o Laravel já cria, por padrão, vários métodos em nosso Controller, que você pode apagar, com exceção do método index().

Até agora já temos configurado a rota, o Controller e o método que irá cuidar da requisição de nosso usuário.

Controllers

```
public function index()
{
    return view('welcome');
}
```

Controllers

- Pronto para aplicações RESTful

O Laravel já foi feito pensado para facilitar a criação de serviços RESTful, portanto, quando devolvemos alguma variável, objeto, array ou qualquer outro dado em nossos Controllers, eles já são convertidos automaticamente para formato JSON. Para exemplificar vamos criar mais uma rota em nosso arquivo routes.php

```
Route::get('/amigos', 'PagesController@amigos');
```

Controllers

- Pronto para aplicações RESTful

Agora vamos criar mais um método em nosso arquivo PagesController.php para tratar a requisição de nossa nova rota.

```
/*
 * Retorna JSON com lista de amigos
 */
* @return Response
*/
public function amigos()
{
    $amigos = [
        ['nome' => 'José Silva', 'idade' => 22],
        ['nome' => 'Maria José', 'idade' => 20],
        ['nome' => 'João Pinheiro', 'idade' => 35]
    ];
    return $amigos;
}
```

Introdução ao Laravel

Views e Templates

Views

- Views contém a apresentação da nossa aplicação para o usuário (HTML, CSS, Javascript, etc).

Blade

- Um dos motores de template para o Laravel é o 'Blade'. Os desenvolvedores com quem conversei gostam de escrever código HTML usando o Blade.
- É muito fácil para o usuário usar ou escolher a sintaxe exata de que precisam.

Views e Templates

- Blade — Sistema de templates
 - O Blade é o compilador de templates do Laravel (template engine). A diferença dele para outros templates é a sua flexibilidade, o Blade não restringe o uso de PHP puro misturado a syntaxe do template. Os arquivos blade devem utilizar a extensão .blade.php .
 - O grande objetivo do Blade é reduzir a quantidade de código PHP inserido no meio do HTML e aumentar o reúso, para isso, ele disponibiliza uma serie de diretivas que são inseridas junto ao código HTML de acordo com a necessidade da página.
 - Os dois principais benefícios do uso do Blade são a herança e as seções, permitindo trabalhar facilmente com o conceito de master page. Vejamos o código:

Views e Templates

- <!-- master.blade.php -->

<html>
 <head>
 <title>Exemplo de arquivo Blade</title>
 </head>
 <body>
 <div class="container">
 @yield('conteudo')
 </div>
 </body>
</html>

Esse código representa a estrutura da nossa master page.

Views e Templates

Agora vamos implementar página *dashboard.blade.php* que irá herdar esse layout.

- ```
<!-- dashboard.blade.php -->

@extends('master')

@section('conteudo')
 <p>O conteúdo do nosso dashboard vem aqui!</p>
@endsection
```

Nessa página estamos herdando toda a estrutura da nossa master page utilizando a diretiva `@extends` , além de injetar o conteúdo específico da página através da diretiva `@section` .

Para renderizar esse exemplo fictício, poderíamos criar a seguinte rota:

- ```
Route::get('dashboard', function () {
    return view('dashboard');
});
```

Views e Templates

- **Passando Informações para a Camada de Visualização**
 - Além de retornar os dados como JSON para nossos usuários, podemos também retornar uma página com os dados de nossas variáveis. Vamos criar mais uma rota para nossa aplicação com o seguinte código:

```
• Route::get('/sobre', 'PagesController@sobre');
```

Agora vamos criar um método em nosso arquivo PagesController.php para tratar as requisições de nossa nova rota

```
public function sobre() {  
    $eu = [  
        'nome' => 'Wendell Adriel',  
        'idade' => 23  
    ];  
    return view('sobre', compact('eu'));  
}
```

Views e Templates

Agora vamos criar nosso arquivo sobre.blade.php no diretório resources/views

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Sobre mim</title>
  </head>
  <body>
    <h1>Meu nome é {{ $eu['nome'] }}</h1>
    <h2>Tenho {{ $eu['idade'] }} anos</h2>
  </body>
</html>
```

Conhecendo o Projeto

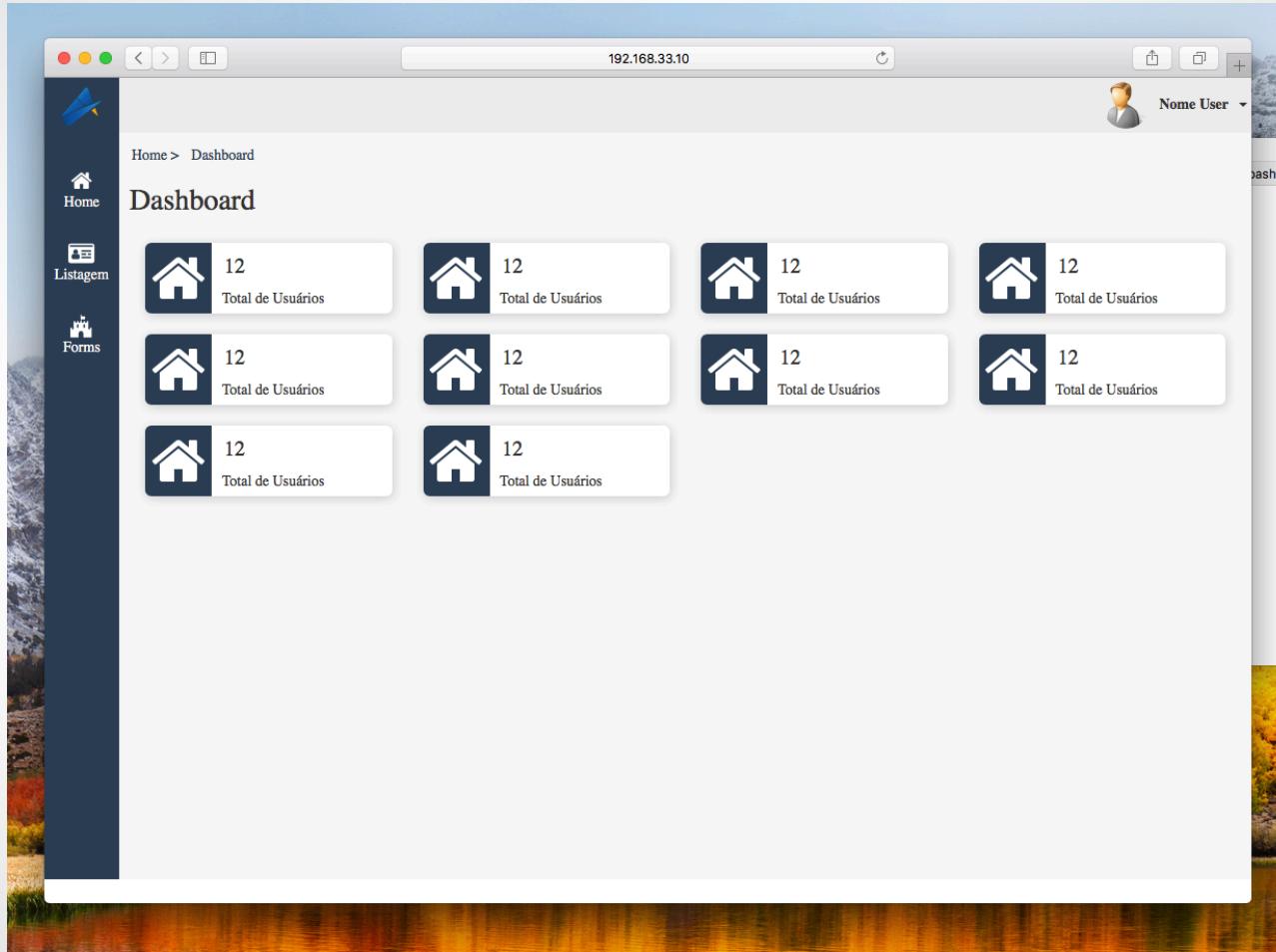
Home

Dashboard

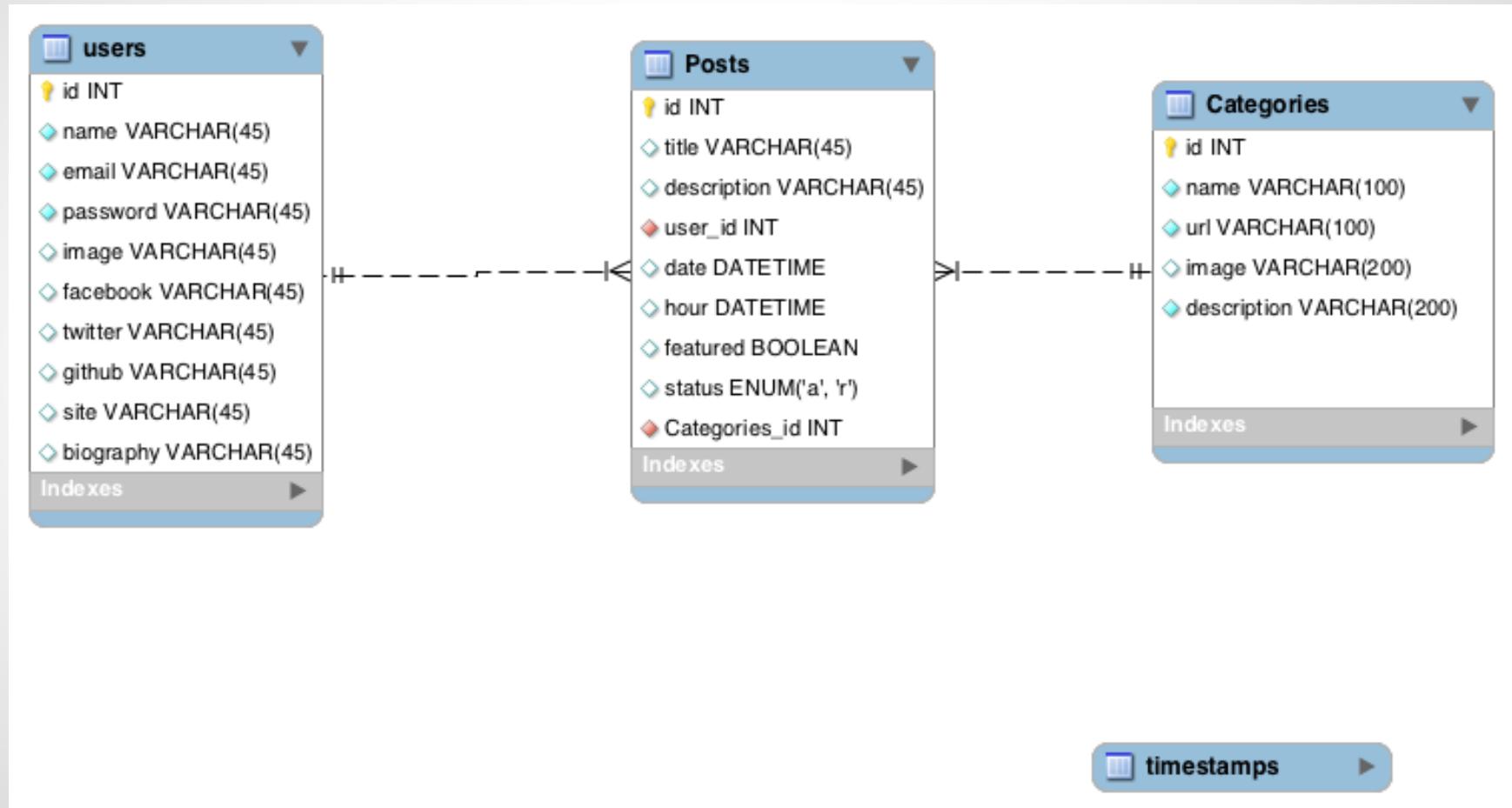
Modelo de dados

Home

Dashboard



Modelo de dados



Integração layout Home & Dashboard

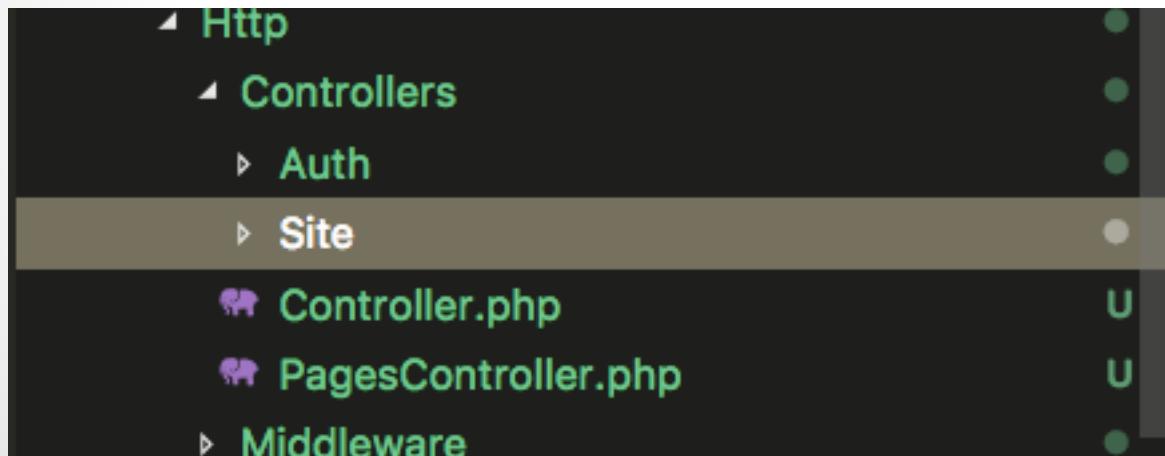
Integração layout Home

- Views – Organização das pastas do projeto.

```
▲ resources
  ▷ assets
  ▷ lang
  ▲ views
    ▷ errors
    ▲ painel
      ▲ modulos
        ≡ forms.blade.php
        ≡ list.blade.php
    ▲ templates
      ≡ dashboard.blade.php
      ≡ index.blade.php
  ▲ site
    ▲ pages
      ≡ categoria.blade.php
      ≡ contato.blade.php
      ≡ empresa.blade.php
      ≡ post.blade.php
    ▲ templates
      ≡ master.blade.php
      ≡ index.blade.php
```

SiteController

- Organizar pastas de controle do Site.
- Criar class de controller do site via terminal.



```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$  
php artisan make:controller Site\\SiteController
```

Blades - Home

- site/templates/master.blade.php
- site/index.blade.php
- site/pages/categoria.blade.php
- site/pages/contato.blade.php
- site/pages/empresa.blade.php
- site/pages/post.blade.php

<https://github.com/jaimievendrame/WebDevalfa-2019>

Router - Site

- #routes/web.php

```
/*
 * Routes Sites
 */
Route::get('/contato', 'Site\SiteController@contato');
Route::get('/empresa', 'Site\SiteController@empresa');
Route::get('/post', 'Site\SiteController@post');
Route::get('/categoria', 'Site\SiteController@categoria');
Route::get('/', 'Site\SiteController@index');
```

SiteController

```
...

class SiteController extends Controller
{
    public function index()
    {
        return view ('site.index');
    }

    public function categoria()
    {
        return view ('site.pages.categoria');
    }

    public function post()
    {
        return view ('site.pages.post');
    }

    public function empresa()
    {
        return view ('site.pages.empresa');
    }

    public function contato()
    {
        return view ('site.pages.contato');
    }
}
```

Laravel avançado

Models & Migrations

Migrations

- Migrations servem para que possamos ter um controle das alterações no banco de dados.
- Com as migrations podemos “versionar” o estado do nosso banco de dados.
- São uma maneira prática de compartilhar entre a equipe o banco de dados.
- Facilitam também o deploy da aplicação, auxiliando na automatização deste processo.

Models & Migrations

- Os bancos de dados relacionais foram criados para prover um acesso facilitado aos dados, veremos como conseguir criar essa relação e acessar esses dados. Aprenderemos a rodar o comando para criarmos a Model e a Migration.
- Na função up criamos a tabela e na função down dropamos a tabela caso ela exista, também há um método de drop, mas DropIndexIfExists é com certeza a melhor prática possível. Existe uma lista grande de tipos de colunas que podem ser usados além desses a cima, você pode ver os tipos disponíveis na documentação.

Models & Migrations

- *php artisan make:model Models\\Category -m*
- *php artisan make:model Models\\Post –m*
- *Vamos configurar os arquivos de Migrations*

Arquivos

- Users:
<https://github.com/jaimevendrame/WebDevalfa-2019/blob/master/migrations/users.php>
- Categories:
<https://github.com/jaimevendrame/WebDevalfa-2019/blob/master/migrations/categories.php>
- Posts:
<https://github.com/jaimevendrame/WebDevalfa-2019/blob/master/migrations/posts.php>

Models & Migrations

- *php artisan migrate*
- Ops! Antes temos que configurar o Banco de Dados.
- Acessar PhpMyAdmin:

Eloquent

- Eloquent é o ORM (Object Relational Mapping) do Laravel.
- É uma abstração do banco de dados, serve como uma camada intermediária pra que não tenhamos que escrever queries diretamente.
- Facilita a manipulação dos dados e também caso seja necessário mudarmos o banco de dados utilizado.

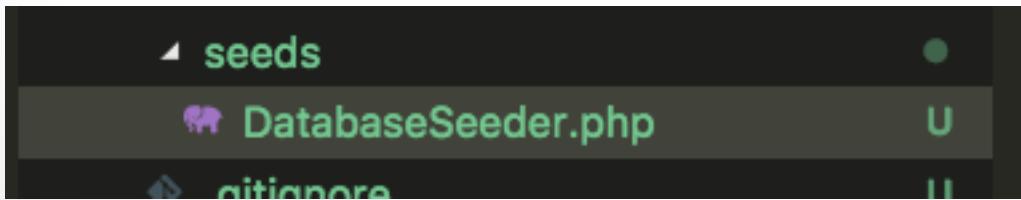
Seeds

- Seeds são uma maneira de popularmos nosso banco de dados com informações.
- Podemos utilizar informações falsas para nos auxiliar no processo de desenvolvimento.
- Temos a ajuda da biblioteca “Faker” para a criação de informações de exemplo.

Criando usuário através de Seeders

```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$  
php artisan make:seed UserTableSeeder
```

Editar arquivo DatabaseSeeder.php



Criando usuário através de Seeders

```
<?php
```

```
use Illuminate\Database\Seeder;  
use App\User;
```

```
class UsersTableSeeder extends Seeder  
{  
    /**  
     * Run the database seeds.  
     *  
     * @return void  
     */  
    public function run()  
{  
        User::created([  
            'name'      => 'Seu Nome',  
            'email'     => 'seu-email@email.com',  
            'password'  => bcrypt('secret'),  
            'biography' => 'Usuário Fulano de Tal',  
        ]);  
    }  
}
```

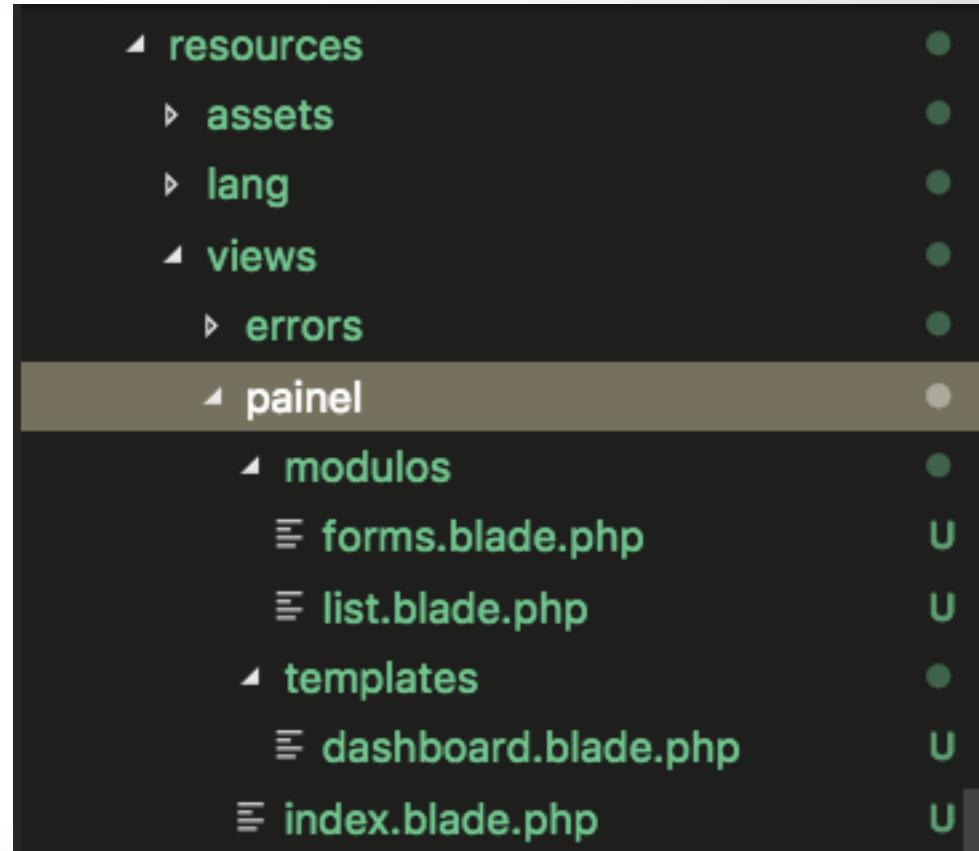
Criando usuário através de Seeders

```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$  
php artisan db:seed
```

Conferir resultado no phpmyadmin

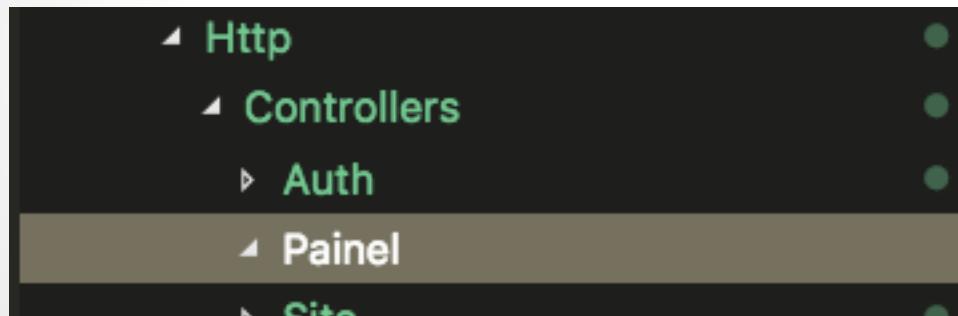
Integração layout Dashboard

- Views – Organização das pastas do projeto.



PainelController

- Organizar pastas de controle do Painel.
- Criar class de controller do painel via terminal.



```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$  
php artisan make:controller Painel\\PainelController
```

Blades - Dashboard

- painel/templates/dashboard.blade.php
- painel/index.blade.php
- painel/modulos/forms.blade.php
- painel/modulos/list.blade.php

<https://github.com/jaimevendrame/WebDevalfa-2019>

Router - Dashboard

- #routes/web.php

```
/**  
 * Routes Painel  
 */  
  
Route::get('/painel/home', 'Painel\PainelController@home');  
Route::get('/painel/list', 'Painel\PainelController@list');  
Route::get('/painel/forms', 'Painel\PainelController@forms');
```

PainelController

```
...
class PainelController extends Controller
{
public function ...()
{
...
}
...
}
...
```

CRUD – Usuários

```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$ php artisan make:controller Painel\\UserController --resource
```

```
/**  
 * Routes Usuários  
 */  
  
Route::resource('/painei/usuarios', 'Painel\UserController');
```

Vamos construir juntos agora!!!

CRUD – Usuários

- Codificando a class Painel\UserController.php
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Usuários

- Codificando a model User.php
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Usuários

- Configuração uploads de arquivos.
 - App\Config\filesystems.php

```
'local' => [  
    'driver' => 'local',  
    'root' => public_path('assets/uploads'),  
],
```

- Criar pasta upload imagem do usuário
 - Assets/uploads/users
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Usuários

- Codificar as Blades.
 - painel/modulos/usuarios/index.blade.php
 - painel/modulos/usuarios/create-edit.blade.php
 - painel/modulos/usuarios/show.blade.php
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Usuários

- Filtrar Usuários
 - Codificar rota search
 - Codificar método search
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Categoria

- Replicar conhecimento no crud de usuários para o de categorias
- <https://github.com/jaimevendrame/webdev-laravel>

CRUD – Categoria

- OTIMIZAR CONTROLLER
 - StandardController
- <https://github.com/jaimevendrame/webdev-laravel>

Authentication

- Com a mais nova instalação do Laravel, foi implementado um sistema de autenticação aprimorado para que você não precise escrever um novo código de autenticação para cada novo aplicativo desenvolvido.

php artisan make:auth

AUTH (Autenticação no Laravel)

```
vagrant@vagrant-ubuntu-trusty-64:/var/www/html/blog$  
php artisan make:auth
```

```
/**  
 * Routes  
 */  
  
Auth::routes();  
  
Route::get('/home', 'HomeController');
```

AUTH (Autenticação no Laravel)

- Personalizar Formulário de Login

```
// Authentication Routes...
$this->get('login', 'Auth\LoginController@showLoginForm'    name='login');
$this->post('login', 'Auth\LoginController@login' );
$this->post('logout', 'Auth\LoginController@logout'   name='logout');

// Registration Routes...
$this->get('register', 'Auth\RegisterController@showRegistrationForm'  name='register');
$this->post('register', 'Auth\RegisterController@register' );

// Password Reset Routes...
$this->get('password/reset', 'Auth\ForgotPasswordController@showLinkRequestForm'  name='password.request');
$this->post('password/email', 'Auth\ForgotPasswordController@sendResetLinkEmail'   name='password.email';
$this->get('password/reset/{token}', 'Auth\ResetPasswordController@showResetForm'   name='password.reset';
$this->post('password/reset', 'Auth\ResetPasswordController@reset' );
```

AUTH (Autenticação no Laravel)

- Configurar Middleware **RedirectIfAuthenticated.php**

```
if (Auth::guard($guard)->check()) {  
    return redirect('/painel');  
}
```

- Resetar senha:
 - Arquivo mail.php
 - Configurar arquivo .env

```
MAIL_DRIVER=smtp  
MAIL_HOST=smtp.zoho.com  
MAIL_PORT=465  
MAIL_USERNAME=contato@sparkcursos.com.br  
MAIL_PASSWORD=Jr130107  
MAIL_ENCRYPTION=ssl
```

AUTH (Autenticação no Laravel)

- Configurar Controllers de Autenticação

protected \$redirectTo = '/painel';

- Configurar Logout:

```
$this->get('logout', 'Auth\LoginController@logout')->name('logout');
```

AUTH (Autenticação no Laravel)

- Retornar dados do usuário logado

```
 {{ Auth::user()->name }}
```

FIM DO PROJETO

- Criar CRUD dos Post usando técnica de controller otimizado.
- Integrar a Home Page com sistema de forma que o conteúdo se atualize dinamicamente.

Ecossistema e serviços ao redor do framework

- Apesar do Laravel ser um framework open source, isso não impossibilitou que o Taylor criasse serviços ao redor do framework.

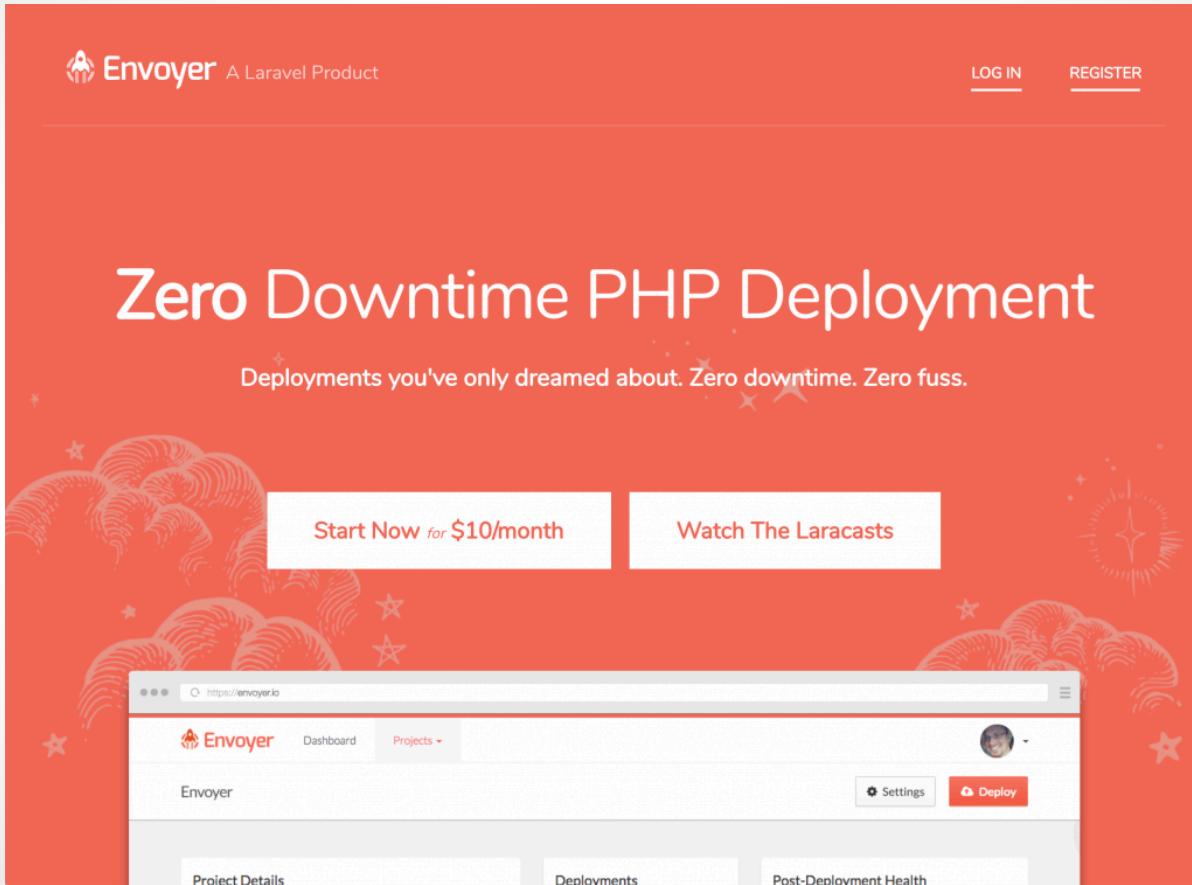
Painless PHP Servers

Provision and deploy unlimited PHP applications on DigitalOcean, Linode, AWS, and more.

Register Now

Learn More

The screenshot shows the Forge web interface. At the top, there's a navigation bar with the 'FORGE' logo, 'Servers', 'Sites', 'Recipes', and a search bar. On the right, there's a user profile for 'Adam Wathan'. Below the navigation, a large card titled 'Create a server' contains four server creation options: 'DigitalOcean (2.0)' (selected and highlighted with a green border), 'Linode', 'Amazon', and 'Custom VPS'. Each option has a corresponding icon. Below these are dropdown menus for 'Credentials' (set to 'Personal 2'), 'Name' (set to 'billowing-stream'), and 'Server Size' (set to '2GB RAM - 2 CPU Cores - 40GB SSD - \$0.03/Hour - \$20/...').





[Login](#) [Register](#) [Documentation](#)

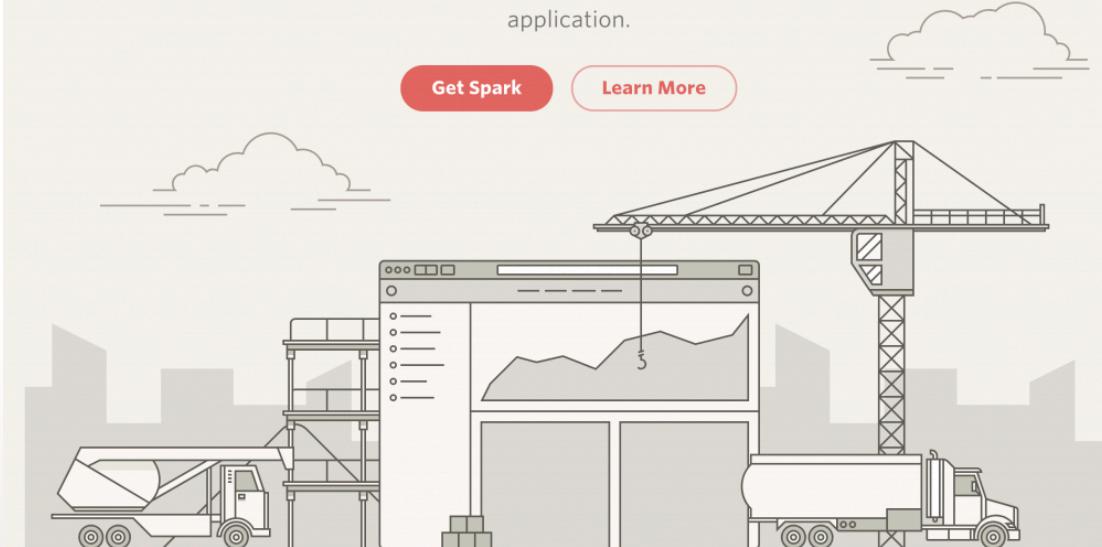
Start Here.

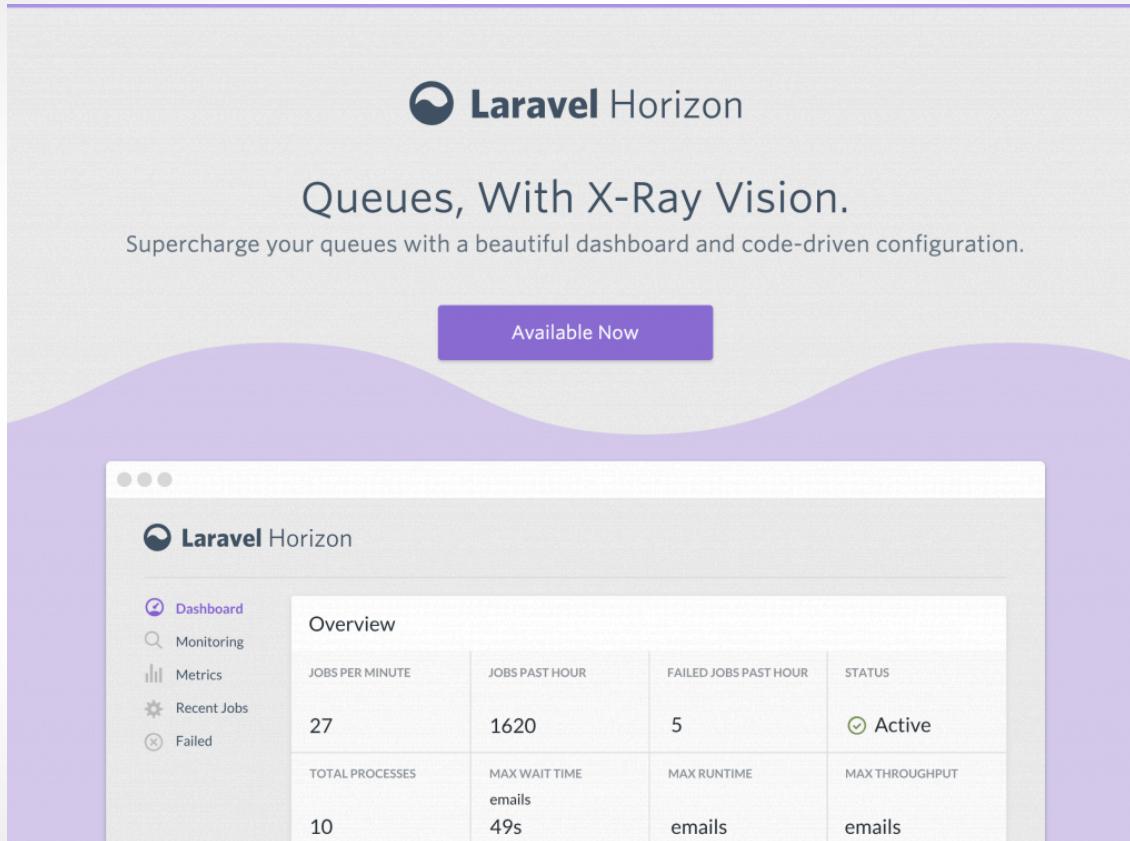
Spark provides the perfect starting point for your next big idea.

Forget all the boilerplate and focus on what matters: your application.

[Get Spark](#)

[Learn More](#)



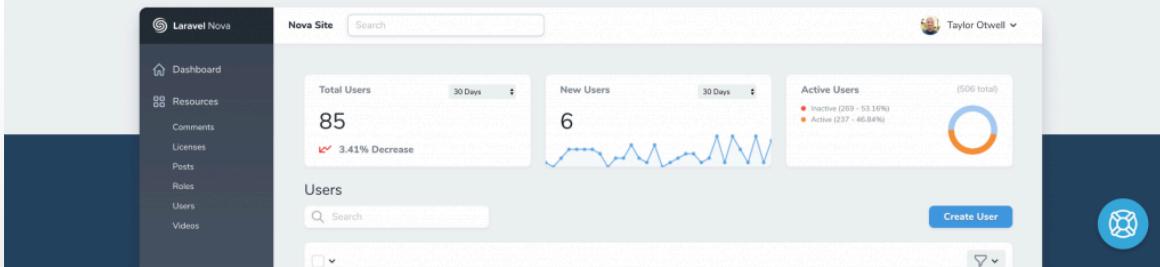
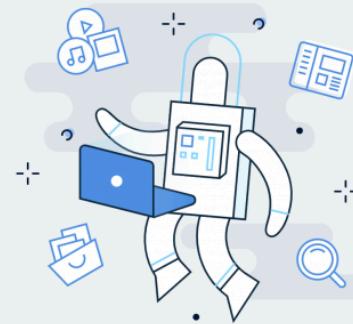


The image shows the Laravel Horizon landing page. At the top center is the Laravel logo followed by the text "Laravel Horizon". Below that is the tagline "Queues, With X-Ray Vision." and a sub-tagline "Supercharge your queues with a beautiful dashboard and code-driven configuration." A purple button labeled "Available Now" is centered. Below this is a screenshot of the Laravel Horizon dashboard showing an "Overview" table with metrics like jobs per minute, failed jobs, and total processes.

Overview			
JOBS PER MINUTE	JOBSPAST HOUR	FAILED JOBS PAST HOUR	STATUS
27	1620	5	Active
TOTAL PROCESSES	MAX WAIT TIME	MAX RUNTIME	MAX THROUGHPUT
10	emails 49s	emails	emails

Master Your Universe

Nova is a beautifully designed administration panel for Laravel. Carefully crafted by the creators of Laravel to make you the most productive developer in the galaxy.

[REGISTER NOW](#)[Read The Documentation](#)

The screenshot shows the Laravel Nova dashboard. On the left is a sidebar with links: Dashboard, Resources, Comments, Licenses, Posts, Roles, Users, and Videos. The main area has three cards: 'Total Users' (85, 3.41% Decrease), 'New Users' (6), and 'Active Users' (506 total, with 237 Active and 269 Inactive). Below these is a 'Users' section with a search bar and a 'Create User' button. A dark blue sidebar on the right contains a user icon and the name 'Taylor Otwell'.

LARAVEL MUITO PESADO,
NÃO PRECISA DE TUDO
ISSO...

LARAVI PESADI NI PRECISI DI
TUDI ISSI

WWW.GERARMEMES.COM.BR

Documentation



Lumen.

The stunningly fast micro-framework by Laravel.

```
1 <?php
2 /**
3  * Reimagine what you expect...
4  */
5 $app->get('/', function() {
6     return ['version' => '5.3'];
7 });
8 /**
9  * From your micro-framework...
10 */
11 $app->post('framework/{id}', function($framework) {
12     $this->dispatch(new Energy($framework));
13 });
14 /**
15  * API routes...
16 */
17 $app->get('api/users/{id}', function($id) {
```

Mais um pouquinho ...

- **Laracon e Laraconf**
 - Pelo fato da grande popularidade do framework, iniciativas como a Laracon foram criadas com o objetivo de unir ainda mais a comunidade, bem como trocar experiências sobre Laravel e tecnologias correlatas. Logo, na Laracon, não se fala apenas em Laravel.
 - A Laracon se espalhou por todo mundo possuindo anualmente eventos nos Estados Unidos, Europa e Austrália.
 - Para aqueles que não podem comparecer nesses ventos, a Laracon também possui uma versão online. Logo, você não tem desculpas para não participar.
- **Laraconf Brasil**

Certificação Laravel



Comunidade Laravel



Obrigado!

Bônus

- <https://github.com/jaimevendrame/laravel55-adminlte>