SUMÁRIO

1. O FRAMEWORK LARAVEL – VERSÃO 8

Laravel é um framework que utiliza a arquitetura MVC, e que foi desenvolvido para facilitar a escrita de aplicações e web services em PHP.

Com foco na simplicidade e legibilidade do código, esse framework disponibiliza interfaces de fácil utilização para os principais componentes que implementam tarefas comuns à maioria dos back-ends, abstraindo a complexidade necessária para a criação de sistemas de rotas, autenticação de usuários, cache, persistência de dados, entre outras funcionalidades.

2. INSTALAÇÃO DO LARAVEL COM COMPOSER

Se o seu computador já tem PHP e Composer instalados, você pode criar um projeto Laravel usando o Composer diretamente.

composer create-project laravel/laravel example-app

Após criado o projeto, uma arquitetura de pastas será criada, e com diversos arquivos. É o template inicial e básico do Laravel.

Como será dito posteriormente, novamente, existe uma pasta chamada “public”, com um arquivo de nome “index.php”. Esse é o entry point da aplicação. Para que a aplicação seja rodada a partir dele, existem duas opções de comandos, e ambas rodam a aplicação no servidor embutido do PHP.

cd example-app e depois **php artisan serve --port=8000**

cd example-app e depois **php -S localhost:**8000 **-t public**

O comando artisan serve é executado na pasta do projeto, e acessa automaticamente o index.php na pasta public para rodar a aplicação, e a porta por padrão é 8000, mesmo se não informada – pode ser outra, como 8080.

O comando php -S localhost é executado na pasta do projeto, e procura por um index.php. Não existe um na pasta de um projeto Laravel, mas na subpasta public, e por isso é informado, com -t public, que deve executar a aplicação a partir desta pasta, onde existe o index.

3. LARAVEL ARTISAN – A LINHA DE COMANDO DO LARAVEL

Artisan é a interface de linha de comando incluída no Laravel. Ele fornece uma série de comandos úteis que podem ajudá-lo enquanto você constrói seu aplicativo.

Para ver uma lista de todos os comandos Artisan disponíveis, você pode usar o listcomando:

php artisan list

Cada comando também inclui uma tela de "ajuda" que exibe e descreve os argumentos e opções disponíveis do comando. Para visualizar uma tela de ajuda, preceda o nome do comando com o termo “help”, dessa forma:

php artisan help migrate

4. ESTRUTURA DE DIRETÓRIOS DO APP LARAVEL

Depois de instalar o framework, e antes de começar a construir o aplicativo, é interessante se familiarizar com a estrutura de diretórios criada.

4.1. PASTA DO PROJETO

A pasta do projeto é a criada pelo comando de criação de projeto laravel, e contém todas as que a seguir serão comentadas.

4.2. PASTA IDEA

Laravel Idea é um ambiente de desenvolvimento baseado em PhpStorm, com foco em eficiência e produtividade. Basicamente, se a IDE utilizada for essa, essa pasta contém plugins de integração do Laravel com ela.

4.3. PASTA APP

O diretório “App” contém, como definido na documentação, parte do “código principal” do aplicativo Laravel. Faz sentido afirmar isso porque nessa pasta existem e são criados, entre outras coisas, os Controllers e os Models.

4.3.1. DETALHES

O arquivo “Kernel.php”, existente nessa pasta, em breves palavras, digamos que processa as funcionalidades de linha de comando.

Na pasta “Http” são encontrados os Controllers, e também os Middlewares, que servem para validar e filtrar requisição HTTP para as rotas da aplicação.

Na pasta "Mail" são encontradas as classes que configuram envios de e-mail, como seu conteúdo, e também a view utilizada – sim, podem ser criadas views para e-mails, especificamente.

A pasta "Models" contém os Models, isto é, as classes que se comunicam com o banco de dados por meio do Eloquent ORM – fato esse que será abordado em outro documento.

E, por último, a **pasta "Providers"** contém as classes que acessam e disponibilizam serviços, ou "códigos encapsulados", como uma espécie de require escalável.

4.4. PASTA BOOTSTRAP

O diretório "bootstrap", que não faz referência ao framework "bootstrap", mas quer dizer "inicialização", contém o arquivo app.php que inicializa a estrutura. Esse diretório também contém um diretório "cache" que contém arquivos gerados pela estrutura para otimização de desempenho, como a rota e os arquivos de cache de serviços.

Normalmente, você não precisa modificar nenhum arquivo nesta pasta.

4.5. PASTA CONFIG

A pasta "config", como o nome indica, contém todos os arquivos de configuração do seu aplicativo.

Às vezes, você pode precisar acessar os valores de configuração em tempo de execução. Você pode fazer isso usando a classe "Config", dessa forma:

```
Config::get('app.timezone');

Config::set('database.default', 'sqlite');
```

4.6. PASTA DATABASE

A pasta "database" contém as migrações de banco de dados, factories e os seeders, que são para fins de testes. Se necessário, também é possível usar este diretório para armazenar um banco de dados SQLite.

4.6.1. DETALHES

Os migrates, ou migrações, são importantes arquivos que gerenciam o banco de dados. A partir deles é possível criar tabelas, editar, excluir etc. São uma forma de abstrair essas ações para o código, e de permitir o seu compartilhamento entre diferentes sistemas que também usam migrações.

Os factories são arquivos que criam modelos de “fábrica” que se comunicam com os Models enviando **dados falsos**, **providos pelos seeders**, apenas para fins de testes. Ou seja, os factories atuam como uma espécie de falso controlador, que chama os métodos do Model, e que envia dados também falsos criados aleatoriamente por um Seeder.

4.7. PASTA NODE MODULES

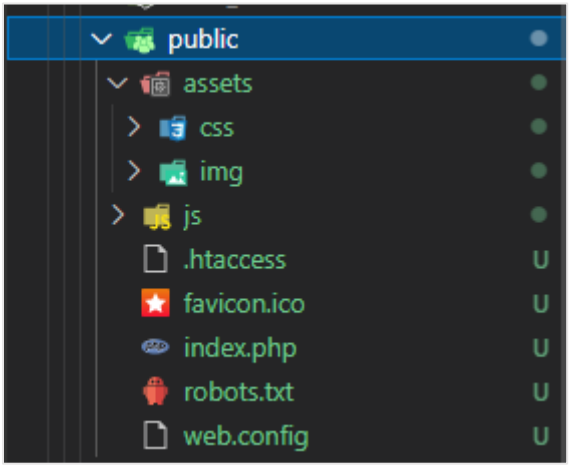
A pasta node modules armazena todos os recursos instalado por meio do gerenciador de dependências do Javascript, vulgo NPM (Node Package Manager).

4.8. PASTA PUBLIC

O diretório “public” contém o arquivo index.php, que é o ponto de entrada para todas as solicitações que entram em seu aplicativo e que, além de outras coisas, configura o carregamento automático do autoload.

4.8.1. DETALHES

Esse diretório também hospeda seus “assets”, como imagens, JavaScript e CSS. Por padrão não existem pastas neste local para esses recursos, mas elas devem, ou podem ser criadas.



Recursos como imagens, gerados pelo aplicativo, também podem ser armazenados em outro diretório, que será abordado mais para frente nesse documento, que é o storage/public.

4.9. PASTA RESOURCES

Essa pasta contém as views, bem como seus recursos brutos e não compilados, como CSS ou JavaScript.

Também contém todos os arquivos de idioma, que são mensagens configuradas por meio da forma de “chave” => “valor”. Na aplicação, cada mensagem é acessada a partir da sua chave.

4.10. PASTA ROUTES

A pasta “routes” contém todas as definições de rota para o aplicativo. Por padrão, vários arquivos de rota são incluídos. São eles: web.php, api.php, console.php, e channels.php.

4.10.1. DETALHES

O web.php contém rotas que são **RouteServiceProvider** colocadas no webgrupo de middleware, que fornece estado de sessão, proteção CSRF e criptografia de cookie. Em outras palavras, para aplicações web e suas próprias rotas, esse arquivo deve ser utilizado.

O api.php deve ser utilizado para as rotas de serviços web, isto é, para consumo de apis, como seu próprio nome diz, “api.php”. Então aqui, nesse arquivo, devem ser configuradas as rotas para consumo de APIS e requisições AJAX.

O **console.php** disponibiliza rotas que são pontos de entrada para comandos de console baseados no aplicativo Laravel.

O **channel.php** é onde são registrados os canais de transmissão de eventos, ou broadcasting, com a tecnologia de web socket. São, digamos, com o auxílio de websockets, meios para criar uma espécie de comunicação contínua de requisição e resposta à moda AJAX entre o frontend e o backend da aplicação.

Assim, e exatamente como diz na documentação, **o conceito básico por trás da transmissão de eventos é esse**: os clientes se conectam a canais nomeados no frontend, enquanto o aplicativo Laravel transmite eventos para esses canais no backend. Esses eventos podem conter quaisquer dados adicionais que você deseja disponibilizar para o front-end, e essa transmissão, se dando com o uso de websockets, é contínua, isto é, na prática, não necessita que a página seja recarregada.

4.11. PASTA STORAGE

O storage contém seus logs, modelos Blade compilados, sessões baseadas em arquivo, caches de arquivo e outros arquivos gerados pela estrutura. Este diretório é segregado em três pastas, a app, framework e logs.

4.11.1. DETALHES

O diretório app pode ser usado para armazenar quaisquer arquivos gerados por seu aplicativo, como de uploads – sim, não necessariamente deve ser na pasta /public. O diretório storage/app/public pode ser usado para armazenar arquivos gerados pelo usuário, como avatares de perfis de usuários.

O diretório framework é usado para armazenar arquivos e caches gerados pelo framework. E, finalmente, **o diretório logs** contém os arquivos de log do seu aplicativo.

4.12. PASTA TESTS

Essa pasta, “tests” contém seus testes automatizados. Os exemplos de testes de unidade e testes de recursos do **PHPUnit** são fornecidos prontos para uso.

Cada classe de teste deve ser sufixada com a palavra Test. Você pode executar seus testes usando os comandos phpunit ou php vendor/bin/phpunit.

Se desejar uma representação mais detalhada e bonita dos resultados do seu teste, você pode executar seus testes usando o comando **php artisan test**.

4.13. PASTA VENDOR

A pasta vendor armazena todos os recursos instalado por meio do gerenciador de dependências do PHP, vulgo Composer.

FONTE

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