1- لاراول یک framework برای web application ها می باشد که از ساختار MVC پیروی می کند. لاراول می تواند به عنوان یک framework full stack استفاده شود. منظور از full stack این است که route request ها برای render کردن بخش frontend از blade و Livewire یا یک Single Page Application ترکیبی مثل Inertia استفاده می کنند. لاراول همچنین می تواند به عنوان یک API backend برای یک javascript Single Page Application یا یک mobile application استفاده شود. برای مثال از لاراول می توان به عنوان API backend برای Next.js application استفاده نمود. در این مورد لاراول برای فراهم کردن Authentication و ذخیره و بازیابی داده استفاده می شود، در حالیکه همچنین می توان از مزیت های سرویس های قدرتمند لاراول مانند: queues, emails, notifications, … بهره برد.

پوشه app: این پوشه بخش مرکزی application می باشد و تقریبا بیشتر کلاس های application در این پوشه می باشد که شامل پوشه های زیر می باشد.

پوشه Broadcasting شامل تمام کلاس های broadcast channel برای برنامه است. این کلاس ها با استفاده از دستور make:channel تولید می شوند. این پوشه به طور پیش فرض وجود ندارد، اما زمانی که اولین کانال را ایجاد می کنیم ایجاد می شود.

پوشه Console حاوی تمام دستورات Artisan Custome برای برنامه است. این دستورات می توانند با استفاده از دستور make:command تولید شوند. این پوشه همچنین هسته کنسول را در خود جای داده است، جایی که دستورات Artisan Custome ثبت می شود و وظایف برنامه ریزی شده تعریف می شوند.

Translation is too long to be saved

این پوشه به طور پیش فرض وجود ندارد، اما با دستورات event:generate و make:event Artisan ایجاد می شود. پوشه events کلاس های event را در خود جای می دهد. از events برای هشدار دادن به سایر بخش‌های برنامه مبنی بر وقوع یک عمل خاص استفاده شود که انعطاف‌پذیری و جداسازی زیادی را فراهم می‌کند.

پوشه exceptions شامل exception handler برای اپلیکیشن می باشد. این پوشه جای مناسبی برای قرار دادن هر نوع exception می باشد که در اپلیکیشن رخ می دهد. برای شخصی سازی کردن اینکه چگونه exception ها نمایش داده شوند باید کلاس Handler در این پوشه را تغییر داد.

پوشه Http شامل controller، middleware و form request ها می باشد. تقریبا همه منطق ها برای handle کردن درخواست های وارد شده به اپلیکیشن باید در این پوشه قرار بگیرد. تغریبا تمام منطق handle کردن request ها فرستاده شده به اپلیکیشن در این پوشه قرار داده می شوند.

Translation is too long to be saved

دایرکتوری Models شامل تمام کلاس های مدل Eloquent می باشد. ORM Eloquent لاراول یک پیاده سازی ActiveRecord زیبا و ساده برای کار با پایگاه داده ارائه می دهد. هر جدول پایگاه داده یک "مدل" مربوطه دارد که برای تعامل با آن جدول استفاده می شود. مدل‌ها این امکان را می‌دهند که query بنویسیم، و همچنین رکوردهای جدیدی را در جدول وارد کنید.

پوشه providers شامل همه service provider های application می باشد. Service provider ها با اتصال service ها در service container, registering events و یا انجام هر کار دیگری برای آماده سازی برنامه برای درخواست های دریافتی، برنامه را bootstrap می کنند.

پوشه bootstrap حاوی فایل app.php است که framework را bootstrap می کند. این پوشه همچنین یک پوشه cache را در خود جای داده است که حاوی فایل های framework تولید شده برای بهینه سازی عملکرد فایل های cache مسیر و خدمات است. معمولاً نباید هیچ فایلی را در این دایرکتوری تغییر داد.

پوشه config، همانطور که از نامش پیداست، حاوی تمام فایل های configuration برنامه است.

پوشه پایگاه داده شامل database migration ها، model factories و seeders است. در صورت تمایل، می توان از این پوشه برای نگهداری پایگاه داده SQLite نیز استفاده کرد.

پوشه public حاوی فایل index.php است که نقطه ورود همه درخواست هایی است که وارد برنامه می شوند و بارگذاری خودکار را پیکربندی می کند. این پوشه همچنین می تواند پوشه های حاوی فایل های تصاویر، جاوا اسکریپت و CSS را نیز در خود جای دهد.

پوشه resource حاوی view ها و همچنین asset های خام و کامپایل نشده مانند CSS یا جاوا اسکریپت است.

پوشه routes شامل تمام تعاریف مسیر برای برنامه است. به طور پیش فرض، چندین فایل مسیر توسط لاراول گنجانده شده است: web.php، api.php، console.php، و channels.php.

فایل web.php حاوی مسیرهایی است که RouteServiceProvider آن ها را در گروه web middleware قرار می دهد که session state، حفاظت CSRF و رمزگذاری کوکی را فراهم می کند. اگر برنامه یک API RESTful نباشد، تمام مسیرها در فایل web.php تعریف می شوند.

فایل api.php شامل مسیرهایی است که RouteServiceProvider در گروه middleware api قرار می دهد. این مسیرها stateless در نظر گرفته شده‌اند، بنابراین درخواست‌هایی که از طریق این مسیرها وارد برنامه می‌شوند، از طریق توکن‌ها احراز هویت می‌شوند و به session state دسترسی نخواهند داشت.

فایل console.php جایی است که می توان تمام دستورات کنسول closure based را تعریف نمود. هر closure به یک نمونه فرمان محدود می شود که یک رویکرد ساده برای تعامل با روش های IO هر فرمان را فراهم می کند. اگرچه این فایل مسیرهای HTTP را تعریف نمی کند، نقاط ورودی (مسیرها) مبتنی بر کنسول را به برنامه تعریف می کند.

فایل channels.php جایی است که می‌توان همه کانال‌های event broadcastiong را که برنامه پشتیبانی می‌کند، ثبت نمود.

پوشه storage شامل لاگ ها، قالب های Blade کامپایل شده، file based sessions، کش فایل ها و سایر فایل های تولید شده توسط framework است. این پوشه به پوشه های app، framework و logs تفکیک شده است. پوشه app می تواند برای ذخیره فایل های تولید شده توسط برنامه استفاده شود. پوشه framework برای ذخیره فایل ها و حافظه های پنهان فریمورک تولید شده استفاده می شود. در نهایت، دایرکتوری logs حاوی فایل های گزارش برنامه است.

پوشه storage/app/public برای ذخیره فایل‌های تولید شده توسط کاربر، مانند آواتارهای profile، که باید برای عموم قابل دسترسی باشند، استفاده می شود. باید یک link نمادین در public/storage ایجاد نمود که به این پوشه اشاره دارد. می توان link را با استفاده از دستور php artisan storage:link Artisan ایجاد نمود.

پوشه tests شامل تست های خودکار می باشد. نمونه‌هایی از تست‌های واحد PHPUnit و تست‌های ویژگی ارائه شده‌اند. هر کلاس test باید با کلمه Test همراه شود. می توان تست ها را با استفاده از دستورات phpunit یا php vendor/bin/phpunit اجرا نمود. برای نتایج دقیق تر از آزمون، می‌توان تست‌های را با استفاده از دستور php artisan test Artisan اجرا نمود.

پوشه vendor حاوی dependencies Composer می باشد.

2- artisan CLI (Command Line Interface) برای لاراول می باشد. Artisan در root اپلیکیشن موجود می باشد. با استفاده از php artisan list لیستی از command ها به ما نشان داده می شود. برای هر دستور یک help نیز موجود می باشد و به صورت php artisan help command نوشته می شود. Laravel tinker یک REPL (Read-Eval-Print-Loop) قدرتمند برای لاراول می باشد که توسط پکیج PsySH بوجود آمده است. Tinker امکان برقراری ارتباط با application شامل eloquent models, jobs, events, … را فراهم می کند. برای وارد شدن به محیط tinker دستور php artisan tinker را در command line می نویسیم.

3- template engine ها زمانی استفاده می شوند که بخواهیم به سرعت برنامه های web بسازیم که به اجزای مختلف تقسیم می شوند. Template ها همچنین render سریع داده های سمت سرور را که باید به برنامه ارسال شوند فعال می کنند. Template engine ها به طور کلی برای برنامه های سمت سرور استفاده می شوند که فقط روی یک سرور run می شوند و به عنوان API ساخته نمی شوند. Template engine های محبوب شامل EJS، Jade، Pug، Mustache، Handlebarsjs، Jinja2 و Blade می باشند. Template engine برای laravel blade می باشد. تمام فایل های blade پسوند .blade.php دارند و به کد های php کامپایل می شوند. فایل های blade را باید در پوشه resources/views ذخیره نمود. داده ای که به صورت زیر به view پاس داده شده است را به صورت {{ $name }} در blade می توان نمایش داد

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

return view('welcome', ['name' => 'Samantha']);

});

همچنین در blade می توان به صورت زیر شرط if و حلقه foreach نوشت

@if (count($records) === 1)

I have one record!

@elseif (count($records) > 1)

I have multiple records!

@else

I don't have any records!

@endif

@foreach ($users as $user)

<p>This is user {{ $user->id }}</p>

@endforeach

4- ابتدا درخواست از طرف کاربر و در روش های get و post و put/patch و delete به کنترلر منتقل شده و کنترلر از طریق مدل با database ار تباط برقرار می کند و پاسخ را به صورت view یعنی در قالب یک فایل html کامل به کاربر نشان می دهد. اگر middleware برای آن route تعریف شده باشد برای مثال AuthMiddleware ابتدا middleware اجرا شده و دسترسی کاربر ثبت نام یا ورود نکرده را به برخی از route ها می بندد.

لاراول همچنین می تواند به عنوان API نیز استفاده شود و response را در قالب json برای بخش front که می تواند react یا vue یا ... باشد ارسال کند.

نقطه ورود همه درخواست ها به application laravel public/index.php می باشد. فایل index.php تعریف composer generated autoloader را بارگذاری می کندو سپس یک نمونه ای از laravel application را از bootstrap/app.php بازیابی می کند. اولین قدم توسط لاراول ایجاد یک نمونه ای از application / service container می باشد.

در قدم بعدی درخواست ورودی براساس نوع درخواست به http kernel یا console kernel فرستاده می شود. App/Http/Kernel.php کلاس Illuminate\Foundation\Http\Kernel را extends می کند. این کلاس یک آرایه ای از bootstrapper ها را تعریف می کند که باید قبل از request ران شوند. این bootstrapper ها error handling و logging را پیکربندی می کنند، محیط برنامه را شناسایی می کنند و بقیه وظایف را که نیاز است که قبل از اینکه request handle شود انجام می دهند. در واقع این کلاس ها پیکربندی درونی laravel را انجام می دهند.

Http Kernel همچنین یک لیستی از Http Middleware را تعریف می کند که همه request ها قبل از اینکه توسط application handle شوند باید از آن بگذرند. این Middleware خواندن و نوشتن Http Session را handle می کند و تعیین می کند که آیا application در maintenance mode است. CSRF token ها ارزیابی می کند. متد handle کار بسیار ساده ای را انجام می دهد. یک request را می گیرد و یک response بر می گرداند.

یکی از مهمترین اقدامات kernel bootstrapping بارگذاری service provider ها برای application می باشد. Service provider ها مسئول bootstrap کردن همه اجزای مختلف مثل: database, queue, validation, routing component ها هستند. همه service provider ها در config/app.php در آرایه providers پیکربندی می شوند. لاراول در provider ها iterate کرده و ابتدا متد register را در هر یک از آنها و سپس متد boot را در آنها اجرا می کند.

یکی از مهمترین provider ها در application، App\Providers\RouteServiceProvider می باشد. این provider همه route ها در پوشه routes را بارگذاری می کند. وقتی که application bootstrap می شود و همه service provider ها register می شوند، request به router تحویل داده می شود.

5- اگر هر مدل eloquent را به عنوان یک resource در نظر بگیریم، برای مثال Post model باید بتوان عملیات CRUD شامل Create, Read, Update, Delete را روی آن انجام داد. با استفاده از دستور زیر می توان یک PostController با تمام action ها شامل index, show, create, store, edit, update, destroy ایجاد نمود

php artisan make:controller PostController –resource

use App\Http\Controllers\PhotoController;

Route::resource('posts', PostController::class);

تک route بالا تمامی route ها برای handle کردن تمام action ها در PostController را ایجاد می کند.

| **Verb** | **URI** | **Action** | **Route Name** |
| --- | --- | --- | --- |
| GET | /posts | index | posts.index |
| GET | /posts/create | create | posts.create |
| POST | /posts | store | posts.store |
| GET | /posts/{photo} | show | posts.show |
| GET | /posts/{photo}/edit | edit | posts.edit |
| PUT/PATCH | /posts/{photo} | update | posts.update |
| DELETE | /posts/{photo} | destroy | posts.destroy |

6- Named route ها URL ها را ایجاد می کنند یا به route های مشخصی redirect می کنند. می توان یک name را به صورت زیر به یک route اختصاص داد

Route::get('/user/profile', function () {

// ...

})->name('profile');

وقتیکه یک name را به یک route اختصاص دادیم، می توانیم از نام route برای ساخت URL یا redirect از طریق route لاراول به صورت زیر انجام دهیم

// Generating URLs...

$url = route('profile');

// Generating Redirects...

return redirect()->route('profile');

return to\_route('profile');

پارامتر ها را به صورت زیر به عنوان دومین آرگومان می توان پاس داد

Route::get('/user/{id}/profile', function (string $id) {

// ...

})->name('profile');

$url = route('profile', ['id' => 1]);

7- به طور پیش فرض statement {{}} Blade از طریق تابع htmlspecialchars برای جلوگیری از attack های XSS استفاده می شود.

اگر داده ای با تعدادی تگ HTML را از طریق کنترلر به view پاس دهیم برای مثال به صورت زیر:

$x = <b>hello</b>

با اسفاده از {{ $x }} به صورت زیر در browser دیده می شود و بنابراین تگ b بدون تاثیر چاپ می شود

<b>hello</b>

ولی اگر با استفاده از {!! $x !!} نمایش داده شود به صورت زیر دیده شده و تگ b تاثیر می گذارد

**hello**