1- پوشه app: این پوشه بخش مرکزی application می باشد و تقریبا بیشتر کلاس های application در این پوشه می باشد. که شامل پوشه های console، exceptions، http، models، providers می شود.

پوشه های console و http یک api را به هسته application فراهم می کنند. پوشه console شامل دستورات artisan می باشد. این دستورات با استفاده از make:command ساخته می شوند. Console kernel نیز در این پوشه قرار دارد که در آن دستورات artisan نوشته می شوند.

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.

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.

Translation is too long to be saved

دایرکتوری Models شامل تمام کلاس های مدل Eloquent می باشد. ORM Eloquent همراه با 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 bootstrap حاوی فایل app.php است که framework را بوت استرپ می کند. این دایرکتوری همچنین یک دایرکتوری کش را در خود جای داده است که حاوی فایل های فریمورک تولید شده برای بهینه سازی عملکرد فایل های کش مسیر و خدمات است. معمولاً نباید هیچ فایلی را در این دایرکتوری تغییر داد.

پوشه config، همانطور که از نام آن پیداست، حاوی تمام فایل های پیکربندی برنامه شما است. خواندن تمام این فایل ها و آشنایی با تمام گزینه های موجود ایده خوبی است.

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

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 ارائه نمی دهد، 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 بدون حالت 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 مبتنی بر بسته شدن را تعریف نمود. هر بسته به یک نمونه فرمان محدود می شود که به یک رویکرد ساده برای تعامل با روش های IO هر فرمان اجازه می دهد. 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 پخش رویداد 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 فریم ورک است. این دایرکتوری به دایرکتوری های app، framework و logs تفکیک شده است. دایرکتوری app ممکن است برای ذخیره فایل های تولید شده توسط برنامه استفاده شود. دایرکتوری 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 نمایه، که باید برای عموم قابل دسترسی باشند، استفاده شود. باید یک پیوند نمادین در public/storage ایجاد نمود که به این دایرکتوری اشاره دارد. می توان پیوند 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 کنید.

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.

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.

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.

5- resource routing: مسیر های get، post، put/patch، delete در قالب یک خط کد و با استفاده از resource انجام می شود.

Route::resource('photos', PhotoController::class);

Translation is too long to be saved

این route مسیرهای متعددی را برای انجام انواع اقدامات روی resource ایجاد می کند. controller تولید شده تمام متد ها با استفاده از این دستور انجام می دهد.

6- route ها در لاراول شامل یک URL و یک Closure می باشند. همه route های لاراول در پوشه routes تعریف می شوند. این route ها با استفاده از RouteServiceProvider load می شوند.

Route::get($uri, $callback);

Route::post($uri, $callback);

Route::put($uri, $callback);

Route::patch($uri, $callback);

Route::delete($uri, $callback);

همانطور که در بالا دیده می شود، تابع get از کلاس route که request با متد get در uri مشخص را handle می کند. $callback یک آرایه می باشد که عضو اول آن نام کلاس کنترلر و عضو دوم آن نام تابعی می باشد که برای این request باید در کلاس کنترلر مشخص run شود. متد های post، put، patch و delete نیز درخواست ها با این متد ها را به صورت گفته شده در بالا انجام می دهند.

7- 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**