**Uso de Middlewares**

**Video #8 - Uso de Middleware**

Veamos en este post cómo crear e implementar un middleware. La función principal es proporcionar una fácil y conveniente capa para filtrar las solicitudes HTTP. Existen diferentes maneras de hacerlo y de hecho Laravel incluye un middleware que verifica si el usuario está autenticado.

Puedes crear un middleware de registro y tener logs detallados de cada solicitud entrante, cualquier cosa que se te ocurra respecto a HTTP puedes llevarla a cabo usando esta tecnología.

**Middleware Personalizado**

$ php artisan make:middleware **Subscribed**

Este se crea en app/Http/Middleware/Subscribed.php. Con él puedes verificar si el usuario está suscrito a mi plan de pago de mi sistema web. O crear un middleware que revise si el usuario que se intenta registrar es mayor de edad.

$ php artisan make:middleware VerifyAge

En ambos casos tendremos nuestros middleware estarán creados en app\Http\Middleware\. Dentro de cada archivo debemos colocar la lógica de acceso correcto. Por ejemplo:

<?php

**namespace** **App**\**Http**\**Middleware**;

**use** **Closure**;

**class** **Subscribed**

{

//...

**public** **function** **handle**($request, Closure $next)

{

**if** ( ! $request->user()->subscribed) {

**return** abort(403, 'Sin suscripción activa');

}

**return** $next($request);

}

}

**403**: La solicitud fue legal, fue correcta, pero el servidor no la responderá porque el cliente no tiene los privilegios o permisos.

Y respecto a la edad podemos hacer lo siguiente:

<?php

**namespace** **App**\**Http**\**Middleware**;

**use** **Closure**;

**class** **VerifyAge**

{

//...

**public** **function** **handle**($request, Closure $next)

{

**if** ($request->get('age') < 18) {

**return** redirect('guidelines');

}

**return** $next($request);

}

}

Aquí dirigimos al usuario a una vista que tenga los textos apropiados para explicarle porqué no podemos seguir con el registro.

**Registro de las Clases Middleware**

<?php

**namespace** **App**\**Http**;

**use** **Illuminate**\**Foundation**\**Http**\**Kernel** **as** **HttpKernel**;

**class** **Kernel** **extends** **HttpKernel**

{

//...

**protected** $middleware = [];

//...

**protected** $middlewareGroups = [];

//...

**protected** $routeMiddleware = [

'auth' => \App\Http\Middleware\Authenticate::class,

'subscribed' => \App\Http\Middleware\Subscribed::class,

'verify-age' => \App\Http\Middleware\VerifyAge::class,

];

//...

**protected** $middlewarePriority = [];

}

Y luego podemos usarla y aplicarla donde corresponde. Veamos en una ruta varios ejemplos:

Route::get('/example', 'ExampleController@...')

->middleware('auth', 'subscribed', 'verify-age');

Acá y en el video de la clase vimos la forma correcta de proteger a nuestras rutas o métodos en controladores, lo importante es definir qué queremos proteger o interceder y crear la lógica en un archivo aparte. Una persona con poca experiencia usaría estos if pero en las vistas, en cada método de un controlador o en cada una de las rutas. Esto funcionaria pero no es la manera correcta de trabajar.