## Configuración de laravel

En nuestro archivo .env encontraremos mucha información sobre cómo se está configurando Laravel como el nombre, el entorno de desarrollo, si muestra o no debug, la URL de nuestra app, datos de conexión a nuestra base de datos, entre otras opciones.

Toda la configuración que hacemos realmente se está llamando dentro de los archivos que están en la carpeta config

Si vienes del curso de introducción a PHP, seguramente tienes PHPMyAdmin el cual puedes seguir usando o puedes cambiar a cualquier cliente que te guste.

## Cómo funcionan las Migraciones de DB

Laravel ofrece un sistema de migraciones de bases de datos las cuales pueden ser vistas como una especie de control de código pero para bases de datos. Esto es muy útil para equipos de trabajo al poder tener los cambios en el repositorio y de esta manera cada miembro del equipo podrá ejecutar las migraciones para tener los esquemas adecuados.

Tenemos un comando llamado make:migration que nos permite generar archivos de migraciones.

Contamos con una sección entera llamada migrate que nos servirá para realizar diferentes acciones relacionadas con las migraciones. Si ejecutamos migrate sin nada más, ejecutará todas las migraciones pendientes en el equipo.

migrate:fresh va a borrar todas las tablas y las creará de nuevo utilizando todas las migraciones que tenemos.

migrate:rollback nos permite regresar un paso.

Puedes encontrar las migraciones en la carpeta database/migrations. Laravel nos ofrece dos migraciones de inicio que son para crear tablas de usuarios y para crear una tabla de resets de passwords.

### Las migraciones tienen dos partes:

up, que nos dice qué va a crear la migración.

down, que revierte lo que se hizo en la migración, activado al hacer rollback.

Al correr las migraciones se nos crearán 3 tablas, una de ella siendo migrations que nos llevará el control de cómo se va generando cada cambio.

Laravel nos ofrece Schema que nos trae diferentes cosas para trabajar sobre los sistemas de bases de datos.

Laravel 5.4: Specified key was too long error

use Illuminate\Support\Facades\Schema;

public function boot()

{

Schema::defaultStringLength(191);

}

## Modelos con Eloquent

Un ORM es un sistema que nos permite mapear registros de la base de datos a objetos dentro dentro de nuestro código. No es exclusivo de PHP ya que se usa mucho en los lenguajes de programación orientada a objetos.

make:model crea una nueva clase para representar un modelo de Eloquent.

Cuando creamos las bases de datos es estándar que las tablas tengan el nombre en plural pero los modelos como representan una clase que representa un objeto, tendrán su nombre en singular.

Todos los modelos los podremos encontrar dentro de la carpeta app. Laravel no tiene carpeta models porque los creadores creen que model puede tener muchos significados.

El comando tinker nos ofrece un entorno de pruebas para ver cómo funcionan las cosas que estamos haciendo. Tiene en cuenta variables de entorno, lo que inicializa Laravel y también sabe que estamos usando Eloquent.

php artisan make:model ExpenseReport

php artisan tinker

>>> App\expenseReport::all();

>>> $report = new App\ExpenseReport();

>>> $report->save();

## Trabajando con un Modelo

No es aconsejable modificar una migración ya que si estamos trabajando en equipo alguien puede haber ya corrido la migración con anterioridad y esto le causaría conflictos. Lo aconsejable es crear una migración adicional.

El comando migrate:fresh lo reinicia todo incluyendo la base de datos y los elementos creados.

## Controladores y recursos

CRUD significa create (crear), read (leer), update (actualizar o editar), delete (eliminar).

php artisan make:controller --resource NameController

resource genera una clase de controlador de tipo recurso. Esto significa que tendremos todo lo necesario para poder hacer CRUD.

index: Aquí se despliegan todos los elementos a mostrar.

create: Creará nuevos elementos.

store: Guardará los elementos creados en create.

show: Mostrará a detalle un solo elemento y por eso es que recibe un id.

edit: Edita los elementos que mostramos en show.

update: Almacena los cambios de edit en la base de datos.

destroy: Eliminará los elementos.

En route:list podremos ver todas las rutas que tenemos definidas

Para crear una ruta de un controlador resource no usamos get o post, sino directamente la estructura Route::resource(); ya que de esta manera se nos crearán rutas para cada uno de los métodos anteriormente mencionados que son creados por el controlador resource.

## Blade layout

Cuando se tienen muchas vistas que repiten gran parte del código HTML, una mejor práctica para evitar esta repetición es crear layouts y extender de ellos. De esta manera el layout tendrá el contenido que siempre se repite y los hijos el código específico de ellos.

@yield marca la parte en donde irá el código de los hijos que extiendan o hereden del layout.

En las vistas hijas se utiliza @section para decir que esa parte del código es la que concuerda con el @yield del layout.

## CSRF

CSRF (Cross-site request forgery) es un tipo de ataque que consiste en que un usuario puede intentar hacer muchas peticiones en nombre de otro. Para esto Laravel genera con cada sesión un token que se usará para validar que exista el usuario en el sistema y que sea él quien está haciendo la petición. Esto también implica que no se pueden hacer peticiones desde otra app hacia el post, debe manejarse de manera interna.

Si queremos que un form pueda pasar la seguridad CSRF de Laravel, debemos agregar el helper @csrf el cual nos agrega un token.

Cuando estamos guardando nuevas entradas en la base de datos podemos redireccionar adonde queramos en nuestra aplicación con una respuesta especial de Laravel llamada redirect.

Estamos trabajando con Middlewares los cuales son muy usados en aplicaciones web que consisten en capas que contienen el request. Cuando llega un request, éste deberá pasar por diferentes capas o filtros (middlewares) quienes al final regresarán una respuesta. Cada uno de los filtros puede detener las peticiones en caso de que algo no cumpla.

Si no se desea usar la protección CSRF se puede directamente quitar el middleware desde el archivo kernel.php. De la misma manera se pueden crear middleware propios y agregarlos aquí.

## Fake PUT/PATCH

En Laravel cuando usamos recursos nos pone Put y Patch como una opción para modificar nuestros recursos. El problema de esto es que en un form no se puede especificar directamente que queremos hacer un Put o un Patch y por esto Laravel nos ofrece un mecanismo para hacer ”Fake PUT/PATCH” y podamos recibir y procesar los datos.

Para que Laravel acepte el Put o Patch es necesario poner dentro del form de manera auxiliar @method(‘tipo de método usado’) y así aunque el form tenga un POST como método, realmente será traducido al que especifiquemos dentro del auxiliar.

## Borrando reportes

De la misma forma en la que pudimos crear un “Fake PUT/PATCH” también podemos crear una especie de “Fake DELETE”.

Con Javascript podemos mostrar una alerta al borrar una entrada que nos permita confirmar si la vamos a borrar o no.

Es bueno que las vistas hagan match con la acción porque permite encontrar rápidamente qué vista le pertenece a qué acción.

Laravel nos ofrece el helper @method(‘delete’) ya que no se puede especificar directamente en el form.

Tenemos con Laravel un método especial findOrFail que permitirá regresar un error 404 si un usuario está haciendo algo no válido al consultar o editar un registro en la base de datos.

## Validaciones

Es muy importante validar siempre la información que los usuarios ingresan en el sistema. En la mayoría de los casos tendrás usuarios bien intencionados que sólo busquen hacer uso del sistema, pero puede ocurrir que haya algún atacante que quiera obtener información que no le pertenece.

Cuando por ejemplo se hace submit a un form vacío, no vamos a querer que el usuario final vea los errores como son lanzados sino manejarlos de alguna manera, así que los validamos con ayuda de Laravel.

Laravel incluye todos los errores de validación que podamos encontrar dentro de un objeto especial llamado errors el cual podemos usar en nuestro template.

Se utiliza la línea vertical | para agregar más validaciones.

Si un usuario se equivoca al llenar los campos de formulario y al intentarlo de nuevo debe ingresarlos todos otra vez, eso significará una mala experiencia de usuario y creará frustración. Por esto mismo se deben poner de nuevo los valores y para esto Laravel nos ofrece un auxiliar especial llamado old que podemos usar en el valor del campo.

## Creamos la vista de reportes

El método show despliega la información específica para un recurso dado.

Existe una técnica en Laravel llamada model binding que antes de llegar a la acción realiza el findOrFail y nos envía el objeto que se requiere.

## Relaciones con eloquent

Si estamos hablando de un blogpost, es posible que éste esté relacionado con múltiples comentarios y a su vez con un solo creador. Existen muchas maneras para relacionar bases de datos y esto nos ayudará a tener una integridad y una referencia hacia lo que necesitamos.

-m o –migration son exactamente lo mismo y se pueden usar indistintamente.

Haremos una relación de muchos a uno, es decir que un reporte puede tener muchos detalles.

PHP en sus últimas versiones nos ofrece que todas las clases contienen la propiedad estática que es el nombre de la clase y la regresa directamente.

## Trabajando con relaciones

## Acción para enviar un mail

Siguiendo el proyecto que se viene trabajando en el curso, y también en muchos otros proyectos que realices puedes querer implementar un sistema de envío de mails por diferentes motivos, sea avisar a alguien de tu empresa que deba salir de viaje sobre algo que esté sucediendo o avisar a tus clientes de alguna acción que se esté llevando a cabo en tu sistema o haya sido completada.

## Enviando emails

Para enviar correos electrónicos necesitamos usar algún servicio especializado en lugar de cuentas propias tipo gmail porque estas últimas tienden a bloquearse al detectar mucha actividad de envíos. mailtrap.io ofrece un servicio especial para probar cómo funciona el envío de

Dentro de tu cuenta de mailtrap tendrás tus diferentes bandejas de entrada y al entrar a ellas encontrarás las credenciales necesarias para configurar la aplicación. Lo más común es usar credenciales SMTP.

En nuestro archivo .env encontraremos cómo configurar el envío de email. Necesitaremos el username y el password.

La configuración real se hace siempre dentro de la carpeta config en el archivo mail.php

PHP artisan nos provee el comando make:mail que se usa para crear una nueva clase de email. En Laravel los correos electrónicos son objetos que podemos trabajar con ciertas características e incluso darles un template para que sean rendereados.

Laravel nos ofrece un facade llamado Mail que nos ayuda a hacer el envío.

## Login

Para evitar que todo el contenido de la aplicación sea público, es necesario utilizar un sistema de autentificación de los usuarios. Para esto tenemos muchos mecanismos en Laravel supremamente sencillos de utilizar.

PHP artisan nos ofrece make:auth que es una estructura para hacer un login y registro básicos con las vistas y las rutas.

Al agregar la línea $this->middleware(‘auth’); tenemos el controlador protegido, lo que quiere decir que no puede ser accedido si el usuario no está logueado.