**Curso de Laravel 8 – Coders Free**



**Programas necesarios**

* XAMPP
* Composer 2.0
* Visual Studio Code

**Instalación**



**Correr el servidor de Laravel**



**Crear un controlador**

****

**Crear un controlador con recursos**



Por buenas practicas se debe escribir el nombre

* En singular
* Empezar con mayúsculas
* Y poner la Controller como terminación

**Crear un modelo**

****

Como buena practica el nombre del modelo debe ser singular e iniciar con Mayúsculas

**Migrar las tablas a la base de datos**

****

**Deshacer la migración anterior**

****

**Crear una migración para una tabla**

****

**Borrar todas las tablas y ejecutar de nuevo las migraciones**

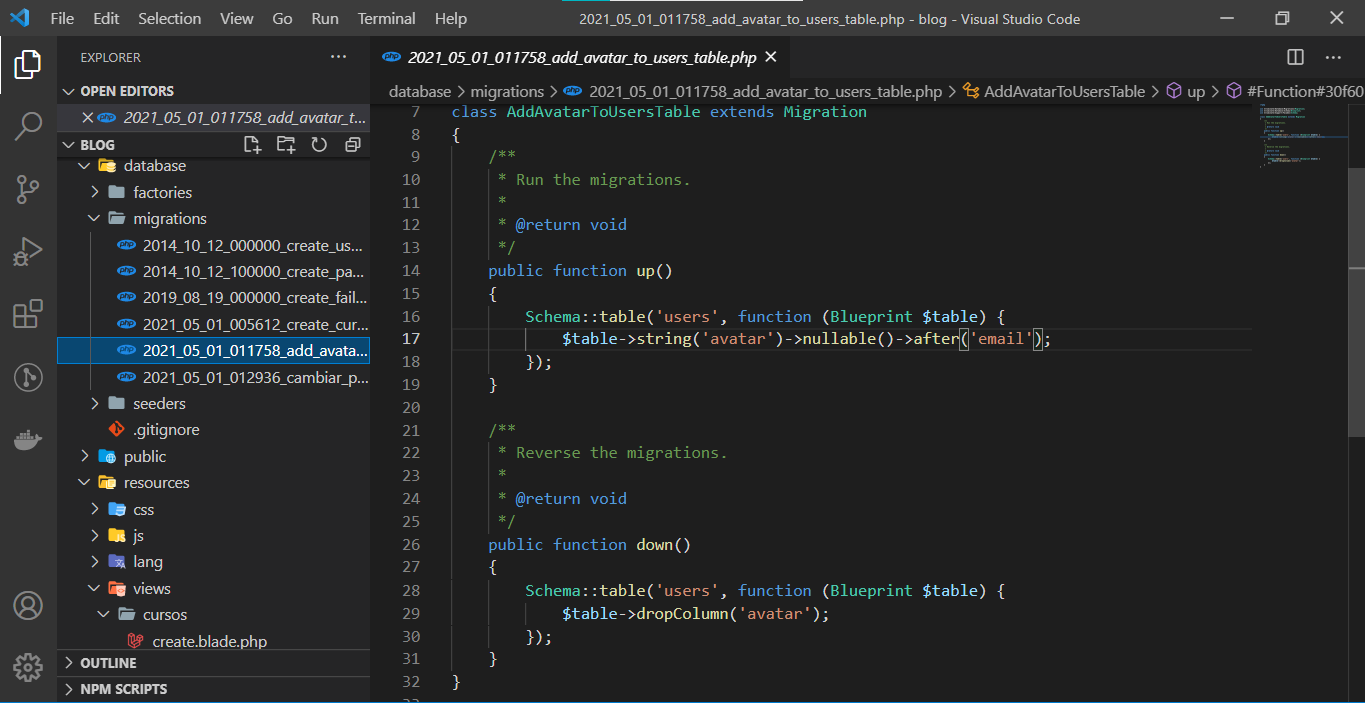
****

**Borrar uno por uno las tablas y migrarlas uno por uno**

****

**Agregar nueva o nuevas columnas a una tabla**





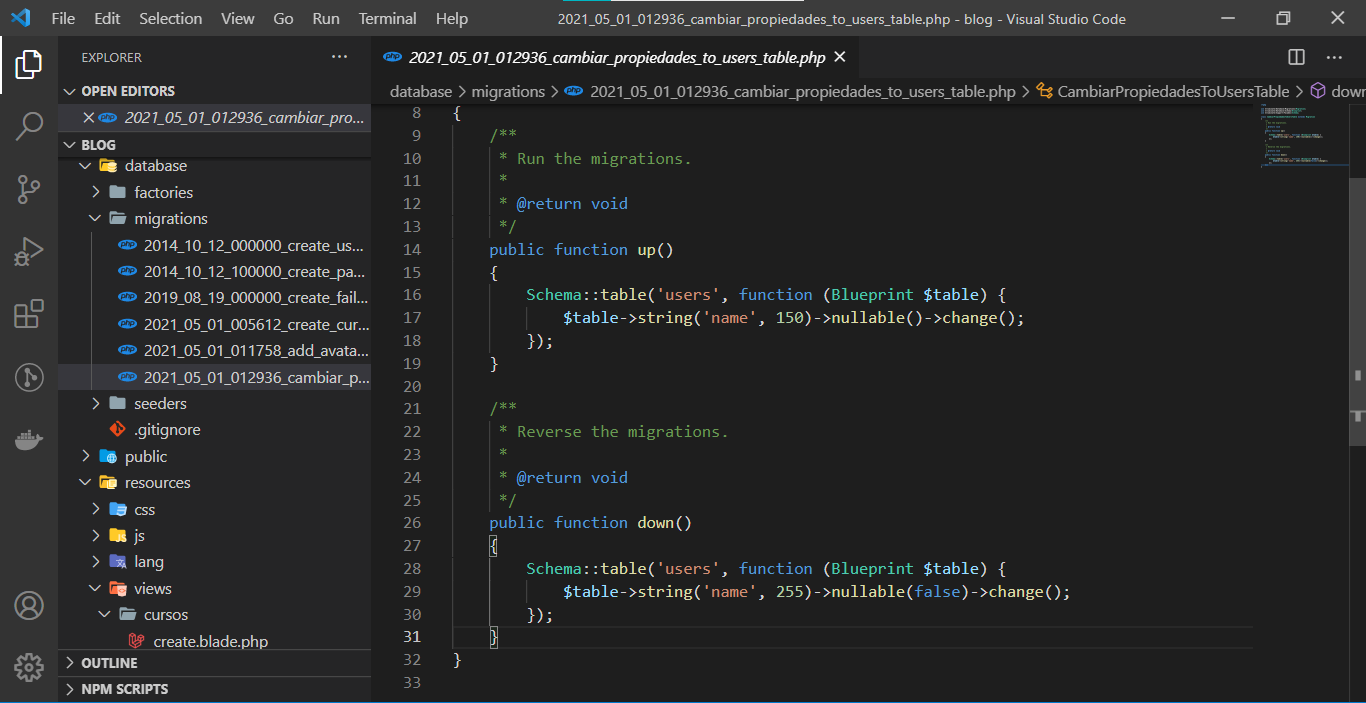
**Cambiar propiedades de una tabla**

* **Se necesita instalar doctrine**

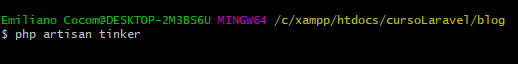
****

* **Crear una migración**

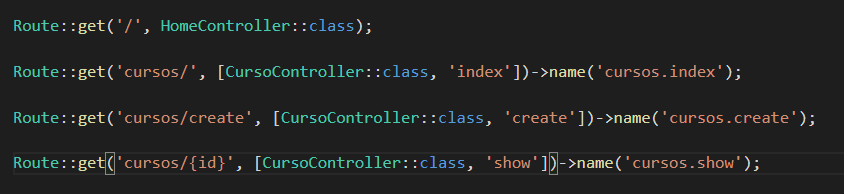
****



**Para utilizar Tinker, por defecto el framework Laravel cuando se instala tiene integrado tinker para correr se ejecuta el siguiente comando**

****

**Definir rutas**

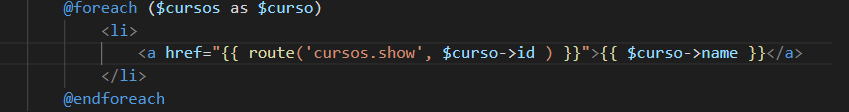
****

Como buena práctica de debe poner un nombre para identificar la ruta, de esta manera si tenemos que cambiar la ruta no necesitaremos realizar los cambios donde estemos llamado la ruta.

**Como agregar enlaces con route (recomendado)**



**Directiva foreach para recorrer los arreglos que recibimos del controlador**

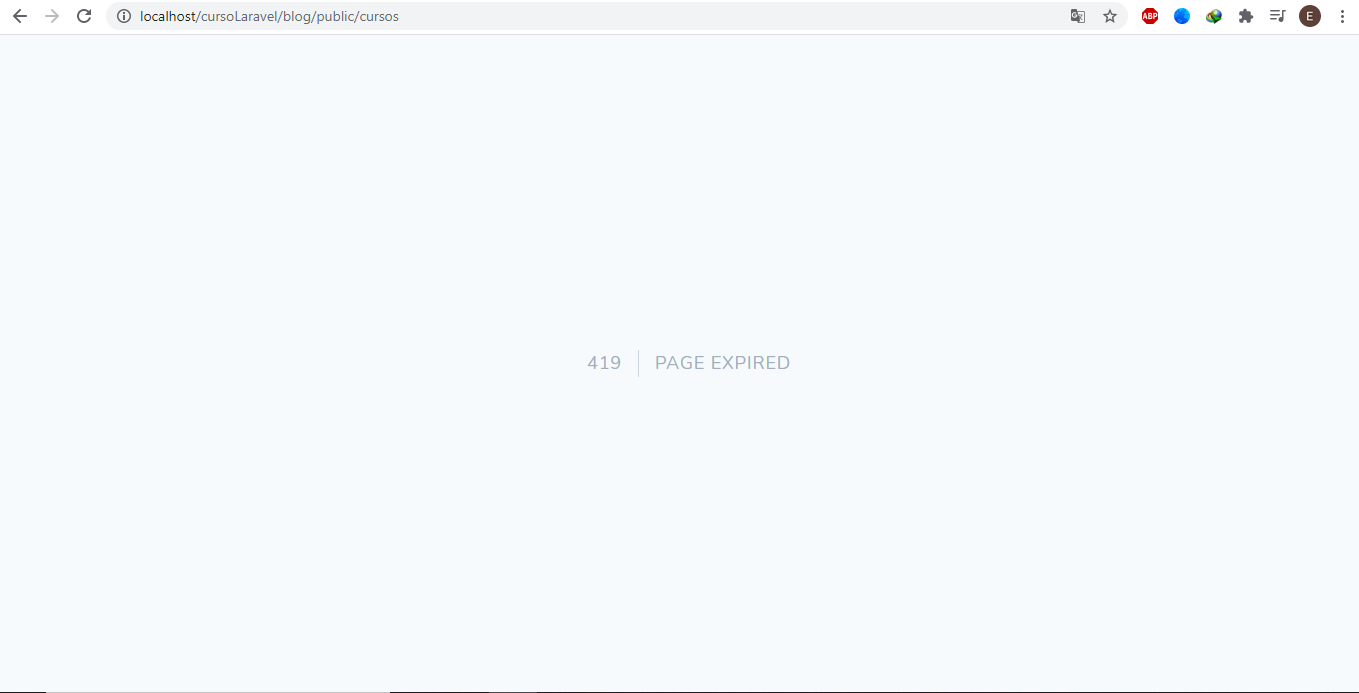
****

**Ejemplo de formulario**



Cuando creamos un formulario en Laravel es importante dentro de la etiqueta form, con action indicarle la ruta a la cual queremos que envié la información, también debemos indicarle el método, al igual que csrf para que laravel cree un toquen de seguridad para enviar la información hacia el servidor, ya que si no ponemos esta etiqueta laravel no enviara la información por motivos de seguridad y nos saldrá un error 419.

**Error 419**



**Validaciones**

**Crear un archivo para la validación del método store del controlador Curso**

****

**Ver el listado de rutas definidas**

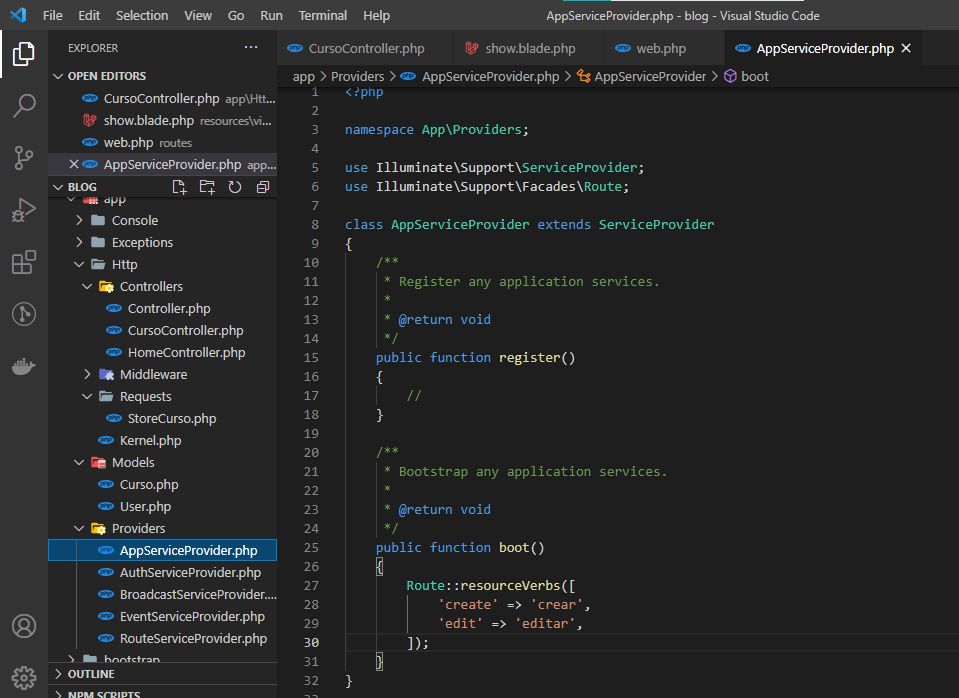
****

**Cambiar el nombre de las rutas y el parámetro que le pasamos**

****

Para cambiar el nombre de las rutas le pasamos, que se diferente al nombre de la url utilizamos el método names y para cambiar el nombre del parámetro que le estamos pasando utilizamos parameters.

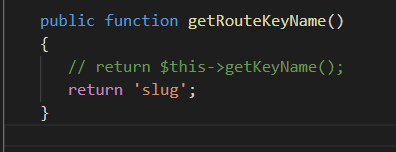
**Para traducir el create y edit en las rutas a editar y crear agregamos en siguiente código en AppServiceProvider**



**Para ejecutar borrar las migraciones, volver a ejecutarlas y ejecutar los seeders**

****

**Crear url amigables**

****

Para crear urls amigables en la tabla agregamos una columna slug y en el modelo ponemos el siguiente método indicándole que la slug va ser la clave que se va utilizar para la urls de navegación en ves de id.

**Crear archivo para enviar emails**

****

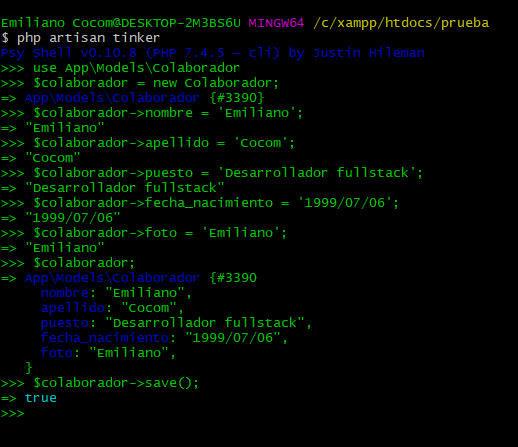
**Orden de los métodos dentro de un controlador para crear un CRUD**

* **index()**
* **create()**
* **store()**
* **show()**
* **edit()**
* **update()**
* **destroy()**

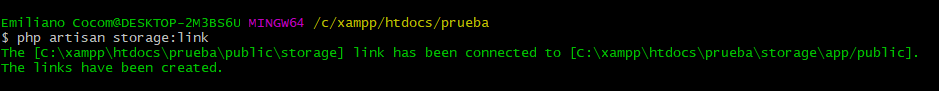
**Ruta resource que utiliza los métodos definidos por Laravel**

****

**Insertando un registro a la base de datos desde Tinker**



**Crear un acceso directo de la carpeta storage a la carpeta public**

****

**Instalar Laravel Jetstream**

1. **Primero instalar laravel**

****

1. **Instalar Jetstream**

****

1. **Seleccionar con que queremos trabajar livewire o inertia, en este caso vamos a utilizar livewire**

****

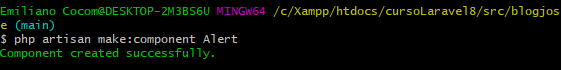
1. **Ejecutar npm install && npm run dev**

****

1. **Por ultimo ejecutar la migración**

****

**Crear componentes en Laravel**

****

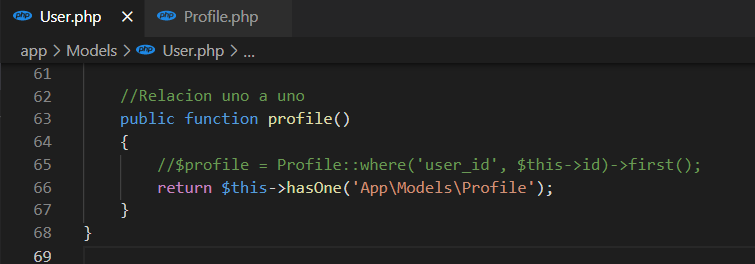
**Publicar los componentes de jetstream para modificarlos dentro del proyecto y no modificar los componentes dentro de la carpeta de vendor**

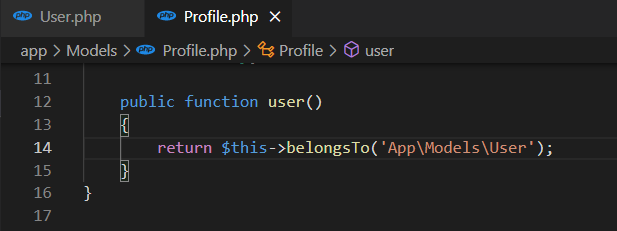
****

**Crear un middleware**

****

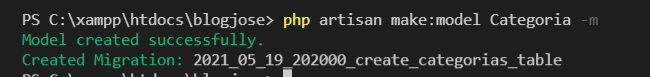
**Relación uno a uno**

****

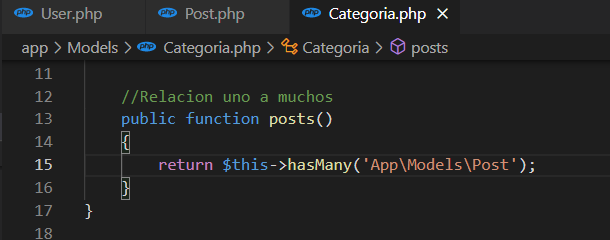
****

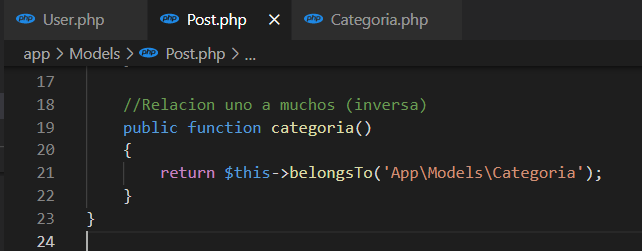
En este caso profiles es la tabla que necesita de la tabla users, siguiendo con la convención de laravel la llave foránea se debe llamar user\_id y la llave primaria de user id. Si no seguimos la convención el segundo parámetro que le debemos pasar es el nombre de la llave foránea y como tercer parámetro el nombre de la llave primaria.

**Crear modelo y migración de una juntos**

****

**Relación de uno a muchos**

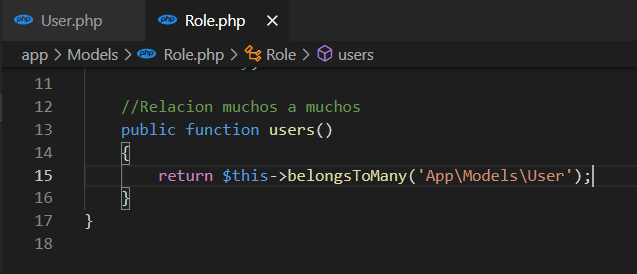
****

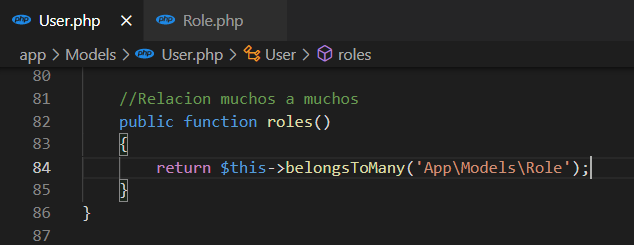
****

**Probando la relación de muchos a muchos desde tinker**

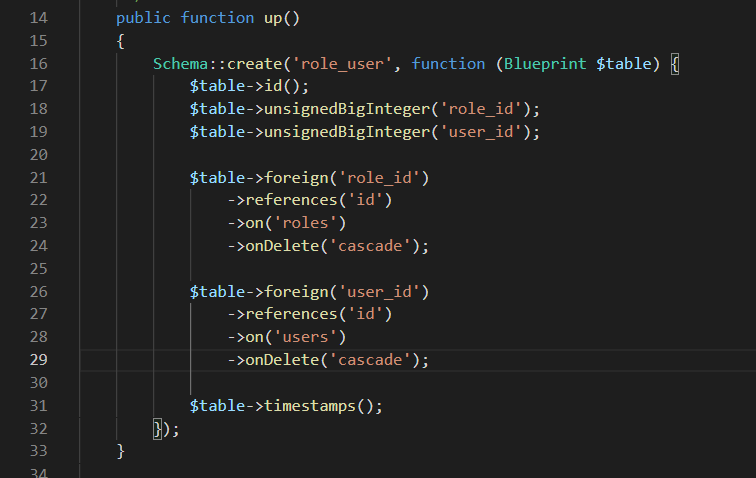
****

**Relación de muchos a muchos**

****

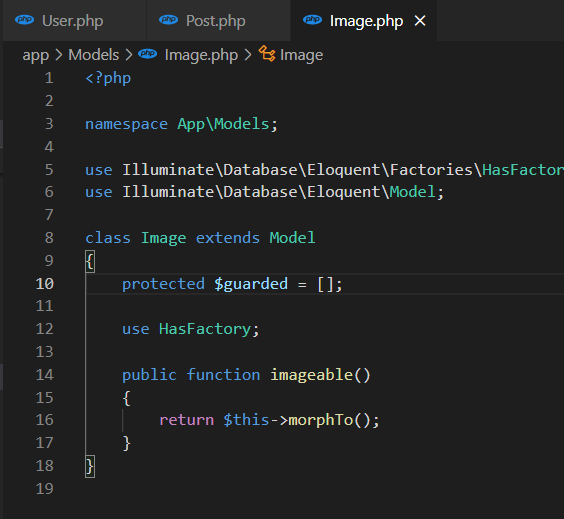
****

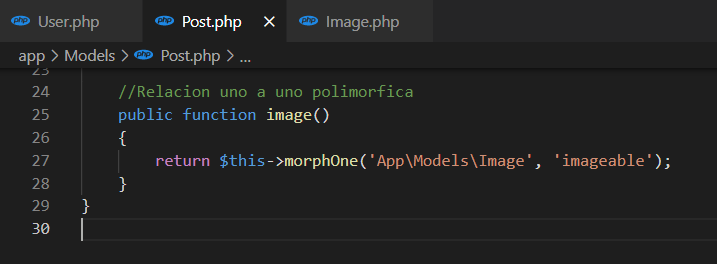
**Tabla que sale de la relación muchos a muchos**

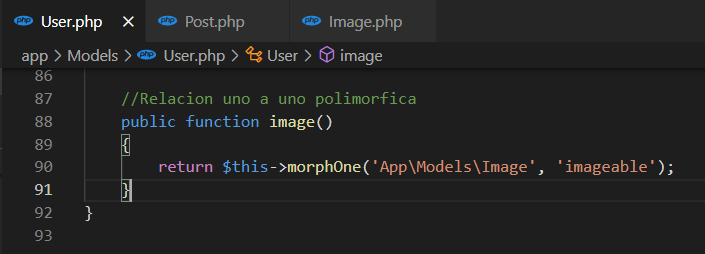
****

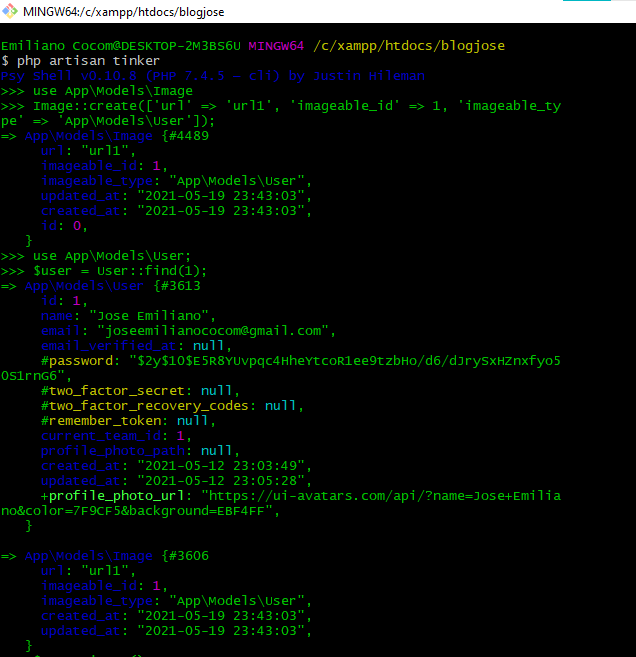
Por convención en laravel para nombrar la tabla pivot se utiliza el nombre de las tablas en singular y por orden alfabético primero va r y después u.

**Relación uno a uno polimórfica**

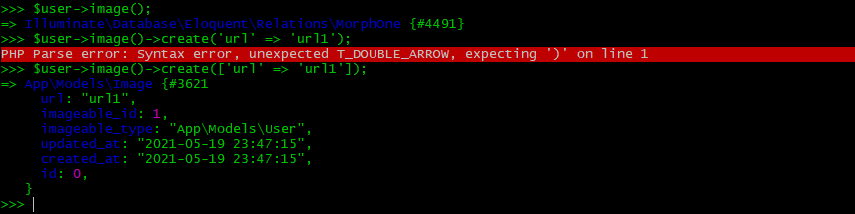
****

****

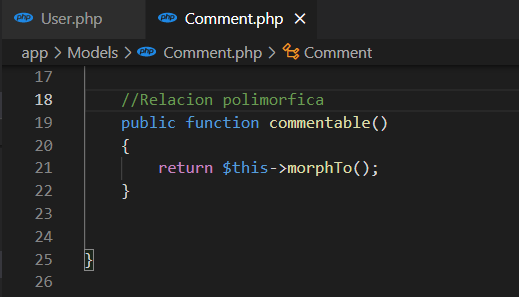
****

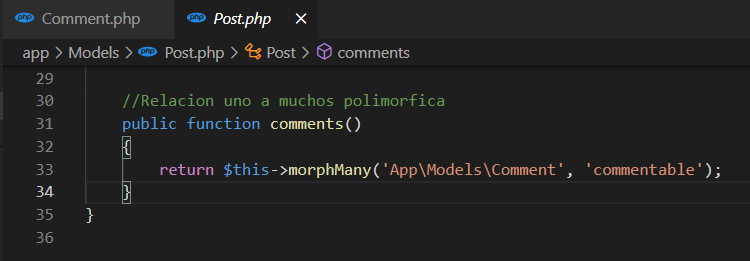
****

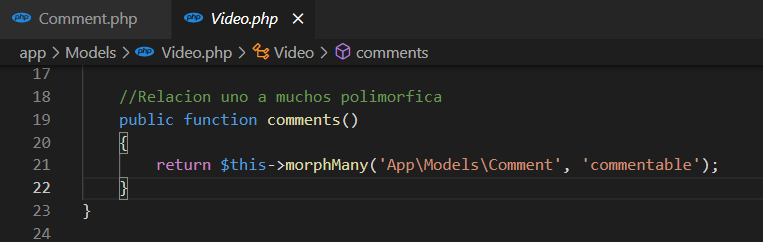
**Guardar datos desde la relación polimórfica de uno a uno**

****

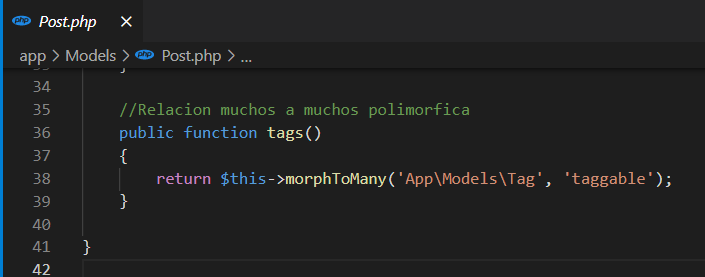
**Relación polimórfica de uno a muchos**

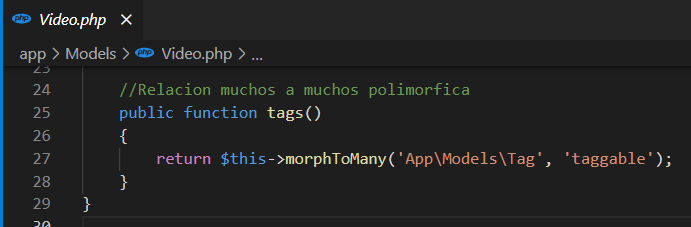
****

****

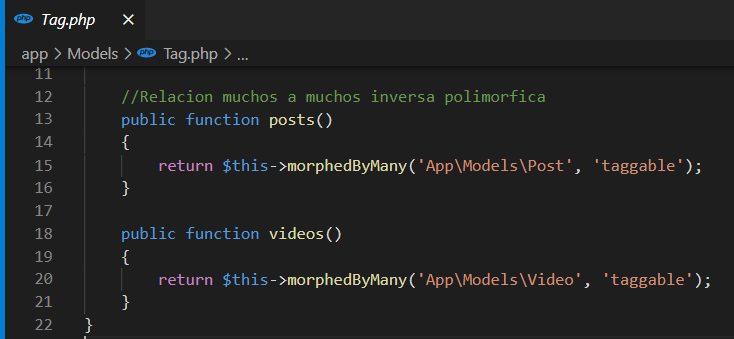
****

**Relación muchos a muchos polimórfica**

****

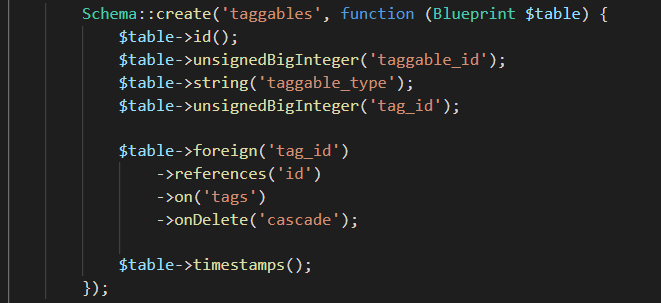
****

Se crea la función tags con donde se pasa la ruta del modelo a utilizar para la relación en este caso el modelo tags y como segundo parámetro el nombre de la tabla en singular.

****

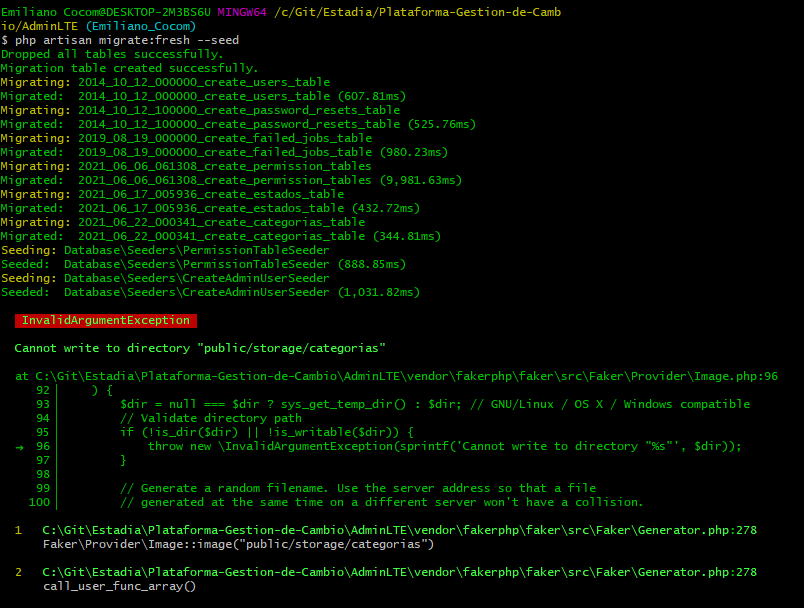
En este modelo de tags se crea dos métodos uno por cada modelo con el que tiene la relación polimórfica de muchos a muchos, en cada método se pasa como primer parámetro la ruta del modelo con el que se tiene la relación y como segundo parámetro el nombre de la tabla en singular

**Migración para la tabla taggables**

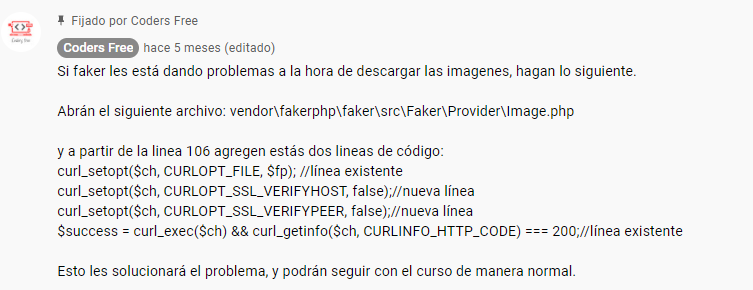
****

En las relaciones muchos a muchos polimórfica se crean el taggable\_id para guardar la clave primaria de post o video y en taggable\_type se guarda la ruta del modelo que hace referencia por ejemplo ‘App\Models\Video’.

**Errores con el acceso directo de storage a public**

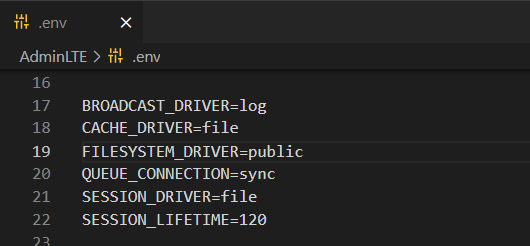
****

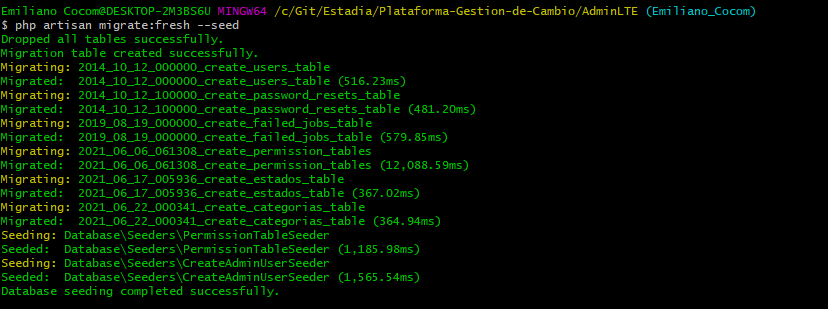
**Solución 1**

****

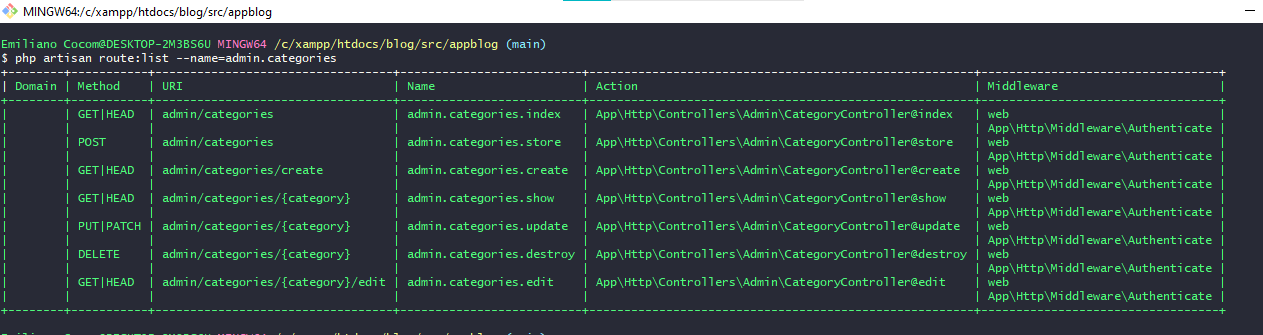
**Solución 2**

Revisar en el archivo .env, el valor por defecto de FILESYSTEM\_DRIVER, el valor de esta variable debe cambiar de local a public



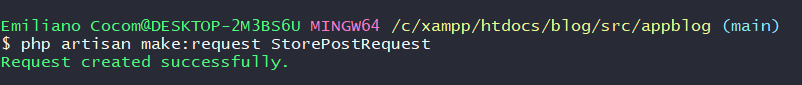


**Ver las rutas según el nombre**



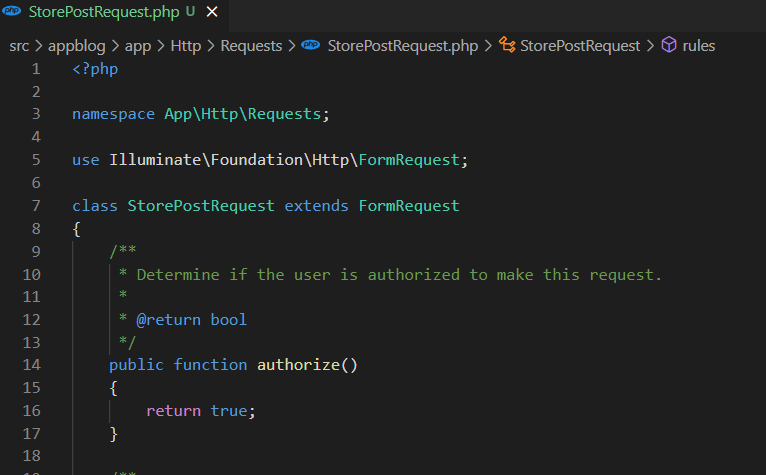
Muestra las rutas que tengan el nombre solicitado

**Crear un archivo Form-Request para validaciones complejas**

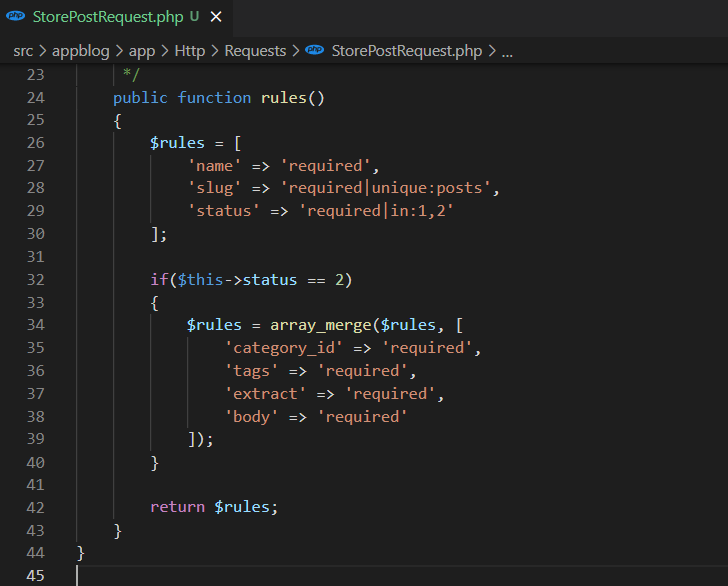
****

Un form request es una clase que incluye dos métodos, authorize y rules

Authorize permite el acceso solo a ciertas secciones, solo a usuarios que cumplan ciertas condiciones, por ejemplo, tener el rol de administrador, el rol de bloggler, ect.

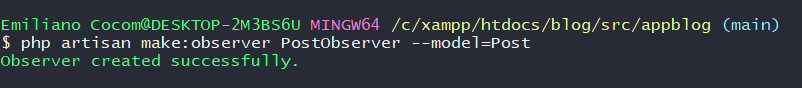


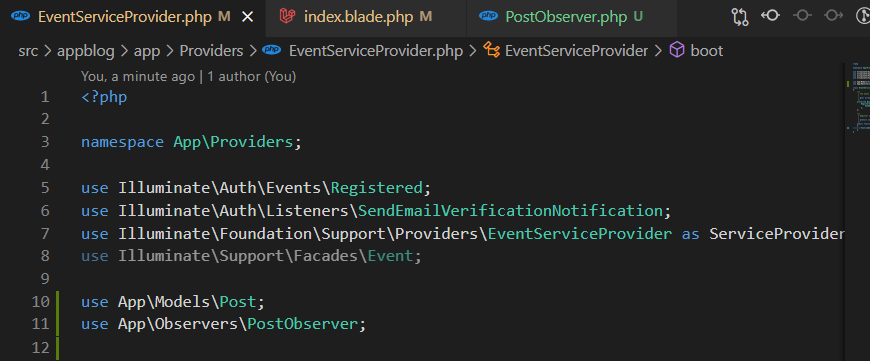
En Rules se agregan las reglas de validación



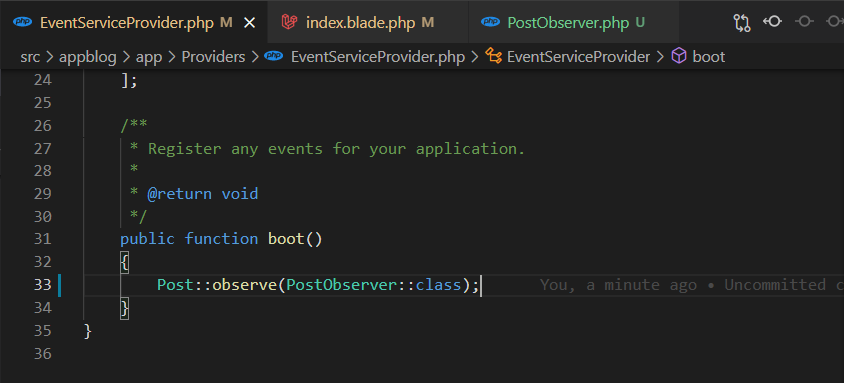
**Observers**

un observer es una clase que nos permite agrupar eventos, relacionados a un determinado modelo y estos eventos se van a ejecutar cada que se realice una acción relacionada al modelo





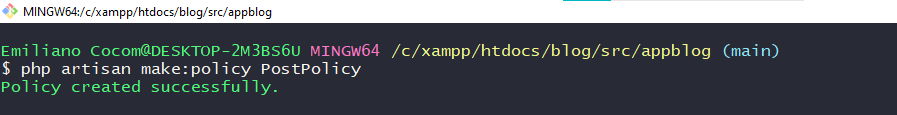
Se tiene que importar el modelo y el observer en EventServiceProvider



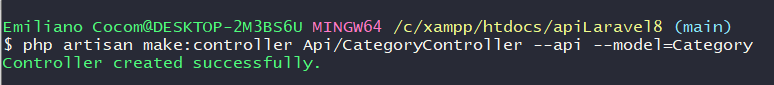
En el método boot se tiene que llamar el modelo con el observador

**Policy**

Las policies o políticas son clases que organizan la lógica de autorización entorno a un modelo o recurso en particular

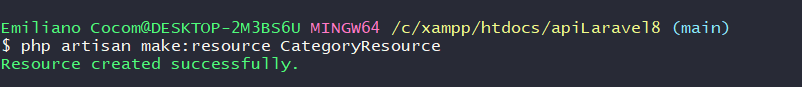


**Apis en Laravel**

****

Creación de un controller api para el modelo category

**Creación de recursos**

****

**Traits de PHP**

Los traits son clases abstractas que nos va poder permitir reutilizar código