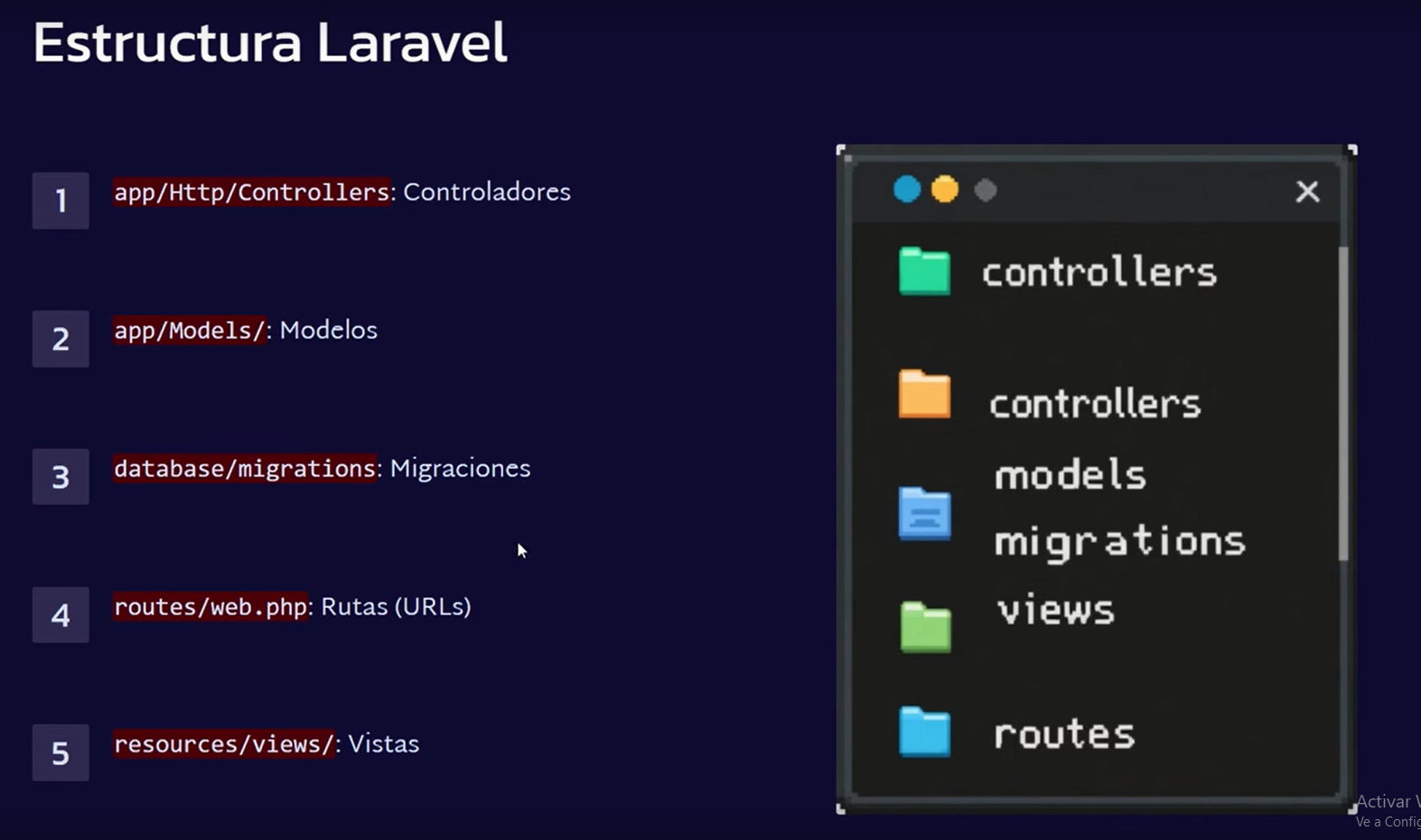
# Tutorial Laravel (https://youtu.be/PaYwBzH\_WSI?si=oDbONMitR29-n0by):

Estructura Laravel

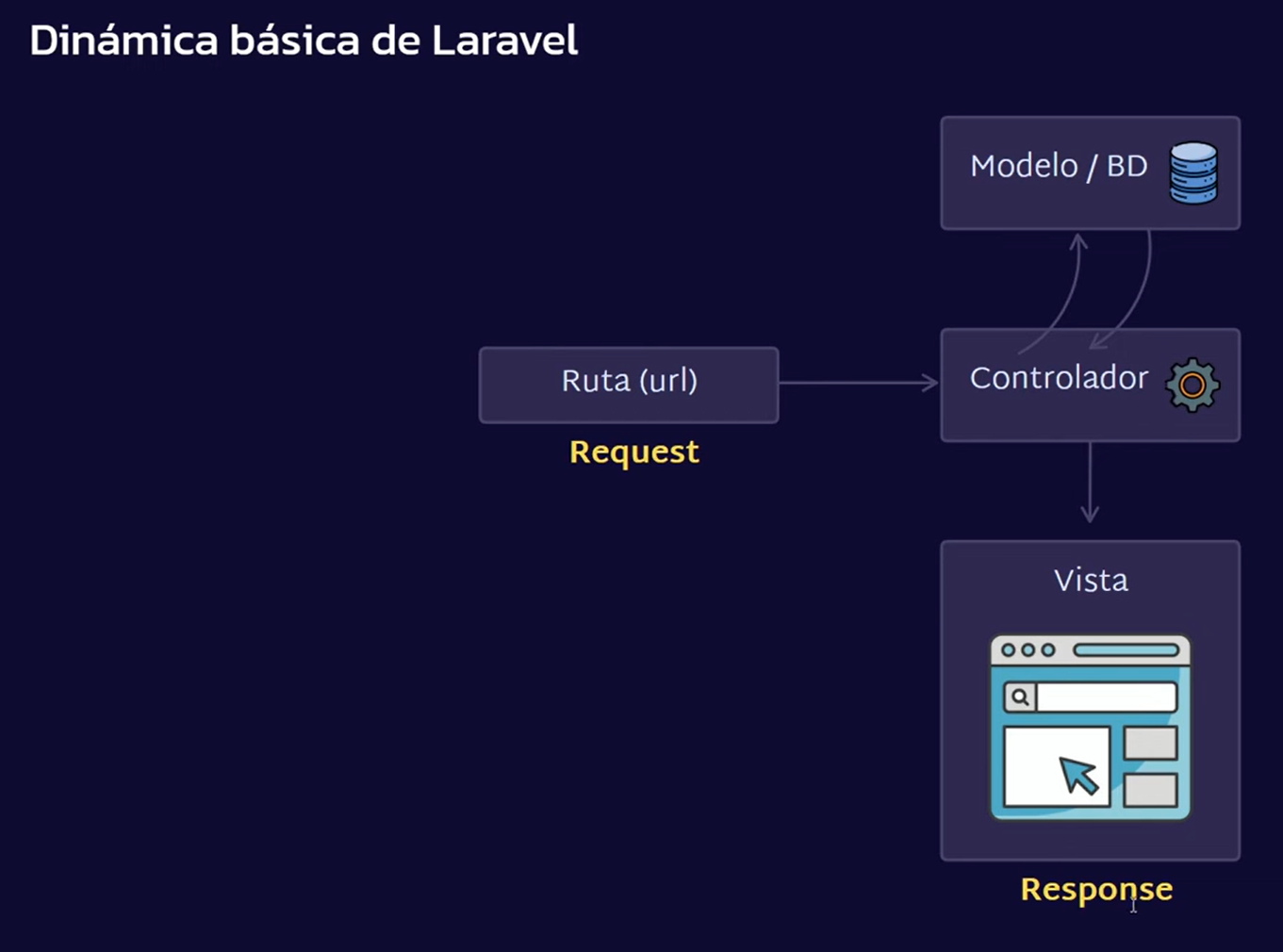


1 Rutas:

2 Migraciones

3 Modelos

4 Vistas



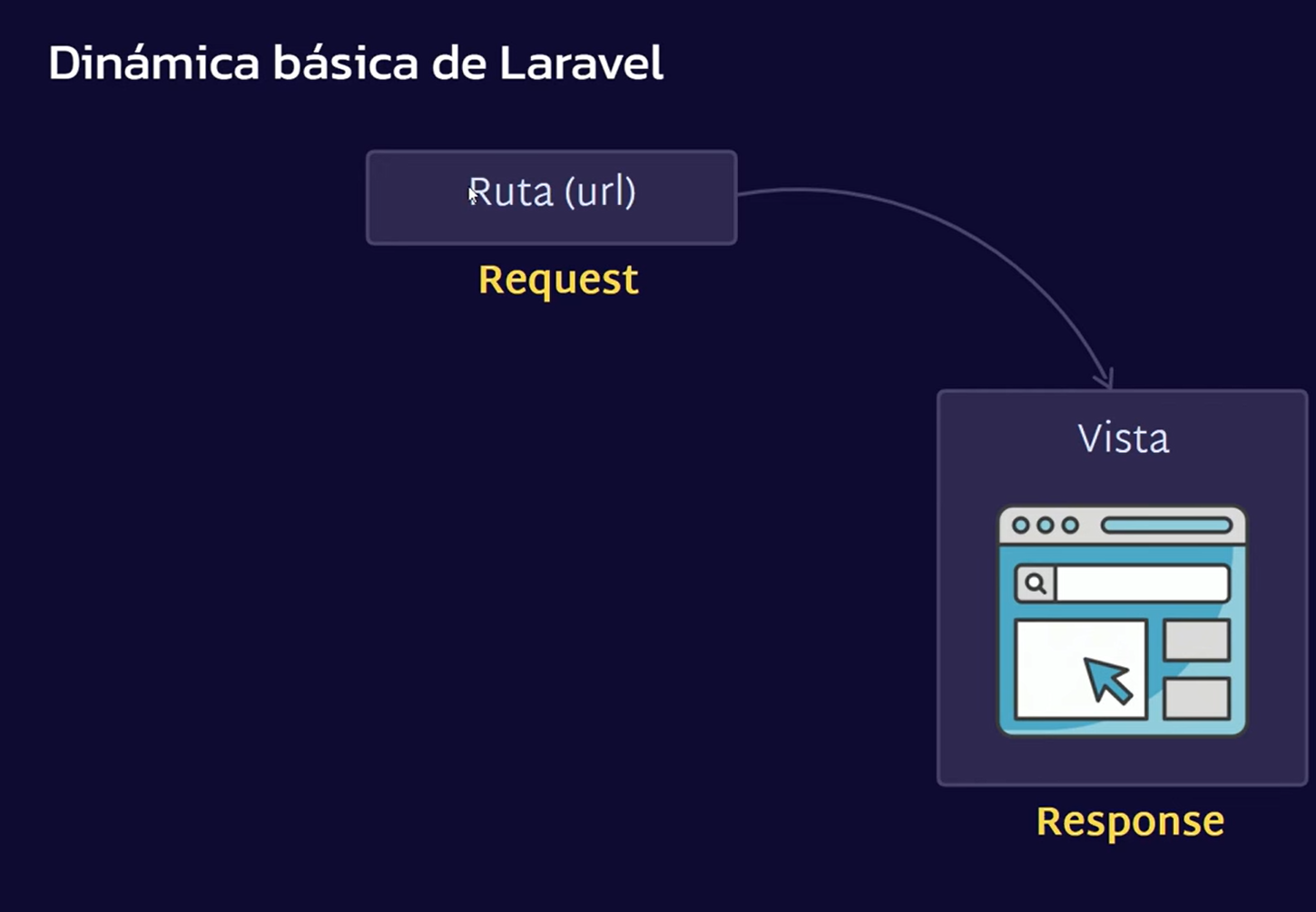
Todo en Laravel, en la web se reduce a un request, una petición y una Response, o sea una respuesta.

Como llegamos a la Response en Laravel.

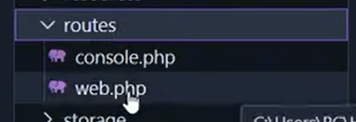
1. Usamos una ruta
2. Se conecta con el Controlador, es un intermediario que puede llegar a consultar la BD. La cual retorna una respuesta una respuesta en el controlador y en este podemos poner lógica de negocios obre la data del perfil.
3. Finalmente componemos una vista con la data modificada.

La respuesta podría ser muchas cosas, como un JSON o un XML.

La dinámica de Laravel puede reducirse a:



Las rutas se guardan en:



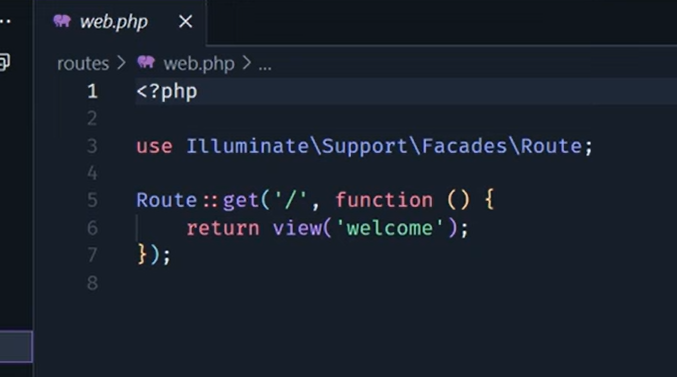
Ejemplo:

Ruta que hace una petición Get, o la que queramos.

Que se dirige a la raíz del proyecto “/”.

También hay un Callback “function”. Que retorna una vista.

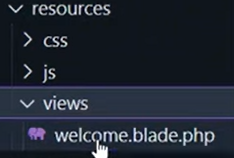
Hay una ruta y una respuesta.



Las vistas:

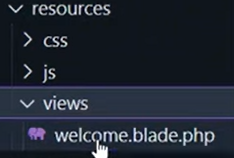
Se encuentran en resources/views.

Tienen la extensión blade.

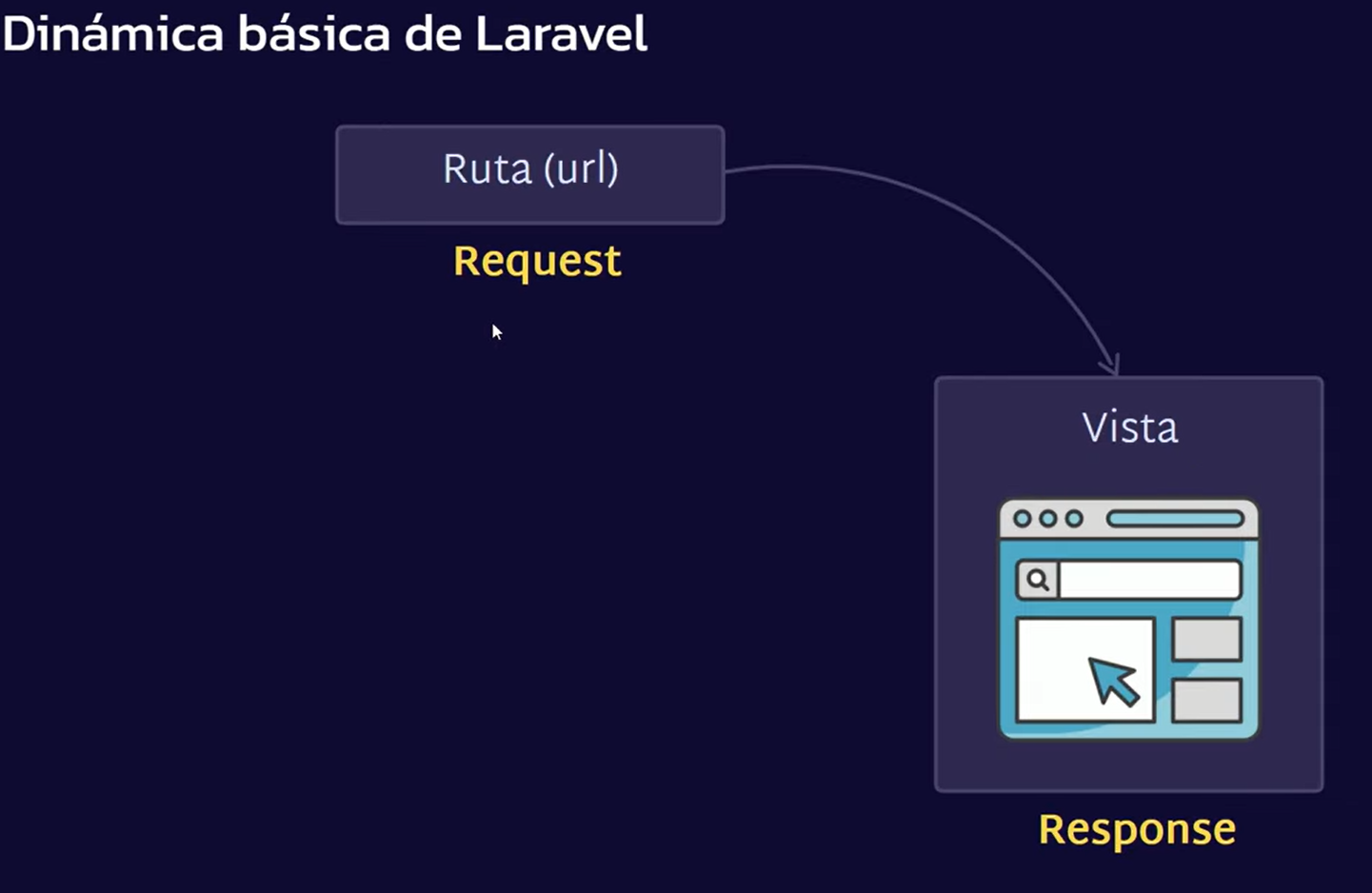


Blade es el motor de plantillas que simplifica el código. Porque se nos facilitan directivas Blades.

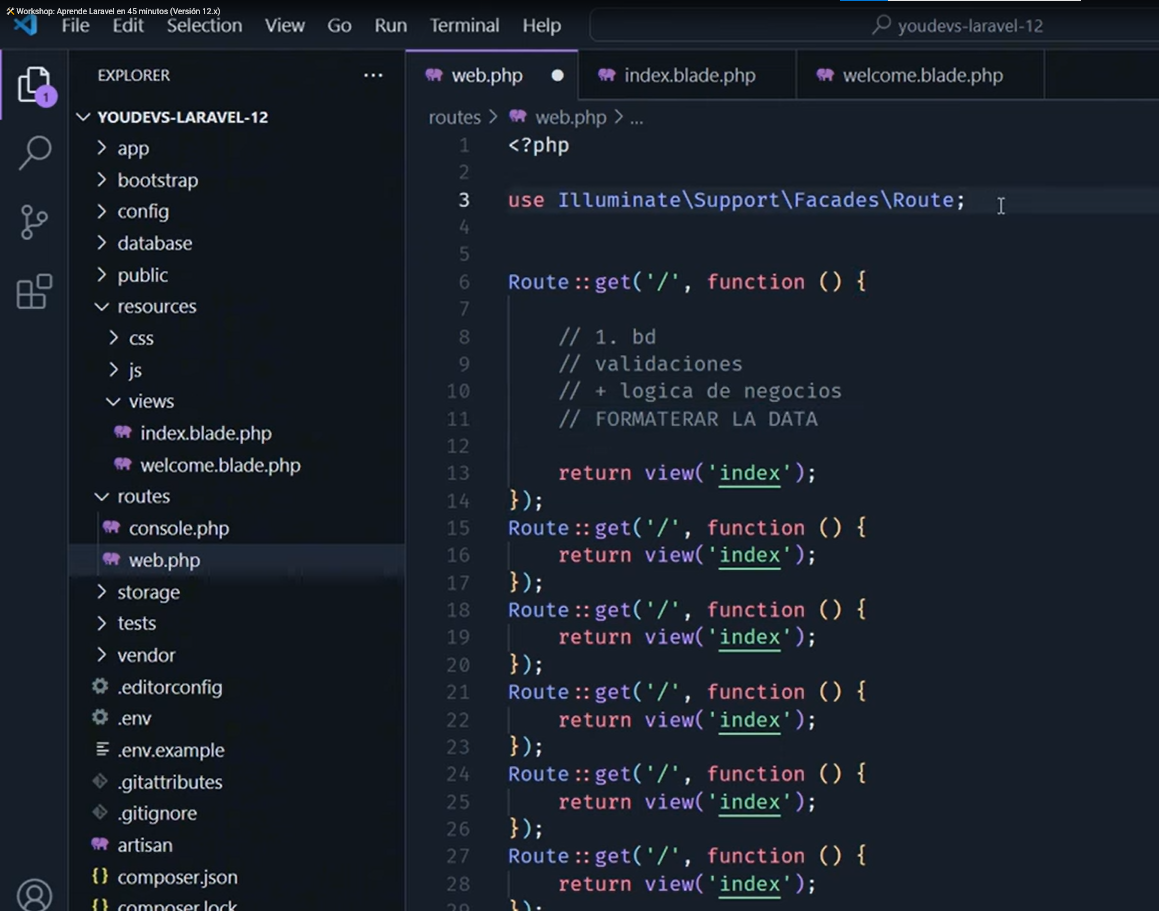
En las rutas la extensión blade no se especifica pues no es necesario, Laravel lo detecta por su cuenta.



Con el flujo básico de Laravel Ruta Vista

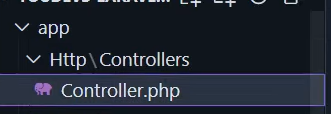


Tendríamos que escribir todas las rutas en el archivo web.php. Consultar la bd, validaciones, etc todo declarado en routes.



Para evitar eso, en lugar de conectar las rutas a una vista, las conectamos a un controlador.

Se puede crear nuevos controladores.

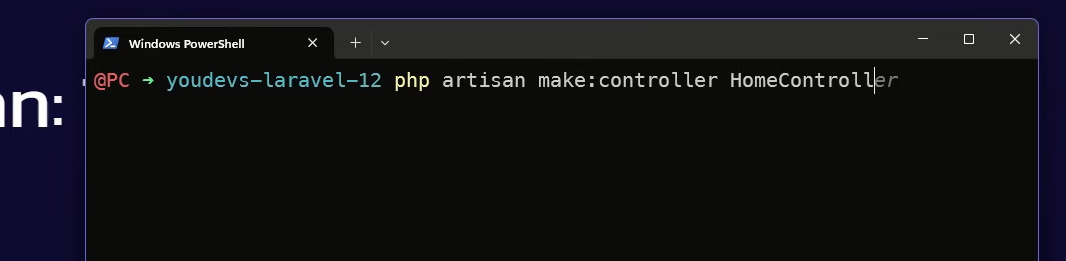


Artisan

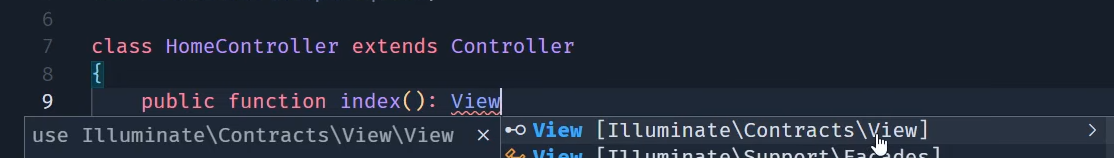


Para crear un controlador

php artisan make:controller HomeController



Para importar vistas:

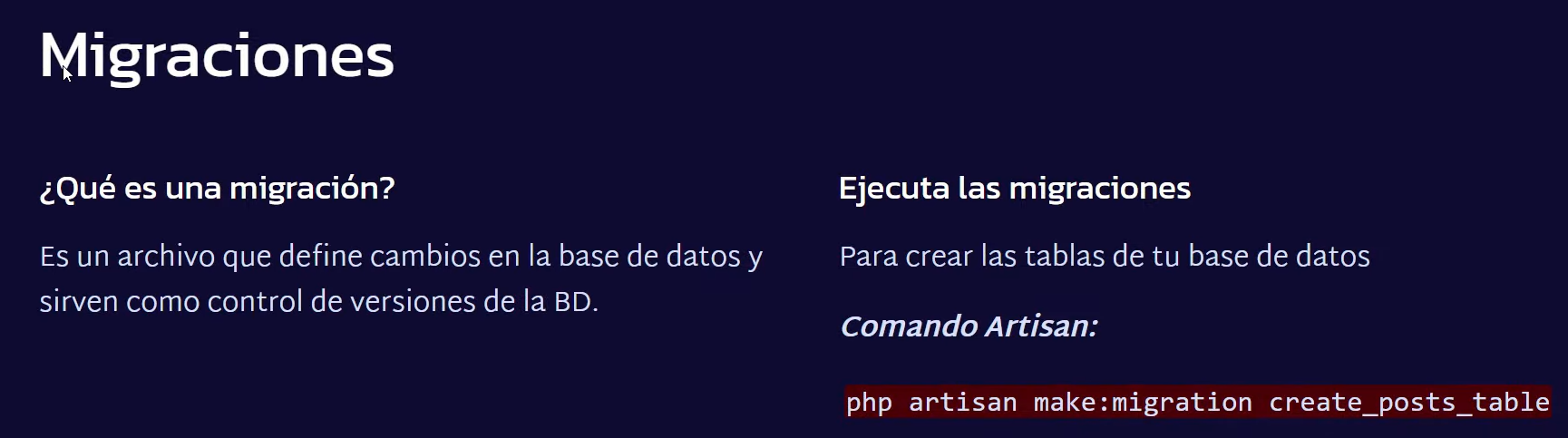


## Modelos para consultar BD



php artisan migrate: ejecuta las migraciones

## ¿Qué son las migraciones?



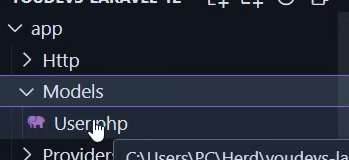
Al migrar, los archivos de las migraciones toman efecto. Solo toman efecto las migraciones que se encuentren en el archivo ../database/migrations

**¿Por qué sirve como un sistema de versiones de BD?**

Porque crea un Timestamp que se hacen y deshacen en orden especifico

Modelos

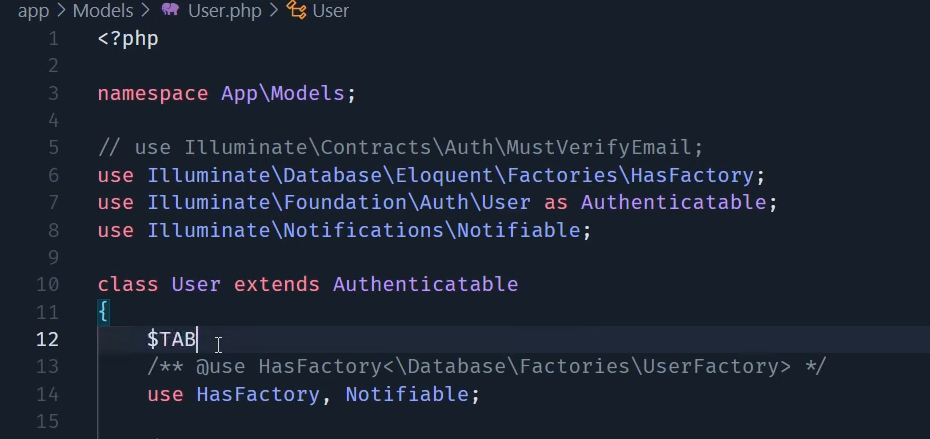
Los modelos se encuentran en la ruta:



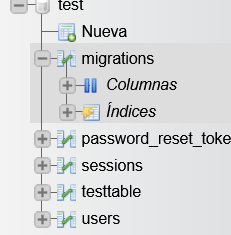


Todo esto nos permite interactuar con la BD sin la ayuda de SQL, sino con php.

Es una clase que está en singular:



Mientras que la tabla real está en plural:



<?php

namespace App\Http\Controllers;

use Illuminate\Contracts\View\View;

use Illuminate\Http\Request;

use App\Models\User;

class HomeController extends Controller

{

    public function metodo(): View {

        //conexion a la tabla con eloquent

        //find busca por id, al 1 en este caso

        $user = User::find(1);

        //vardump die

        // die(var\_dump($user));

        dd($user);

        return view("index");

    }

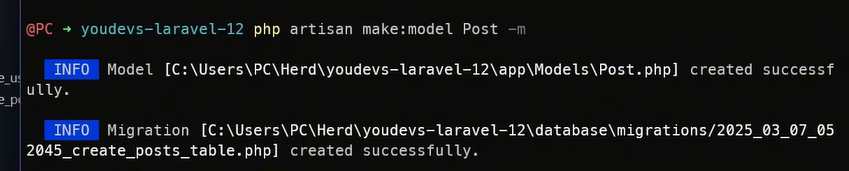
}

## Como enviar datos a la vista

Crear tabla con php artisan:



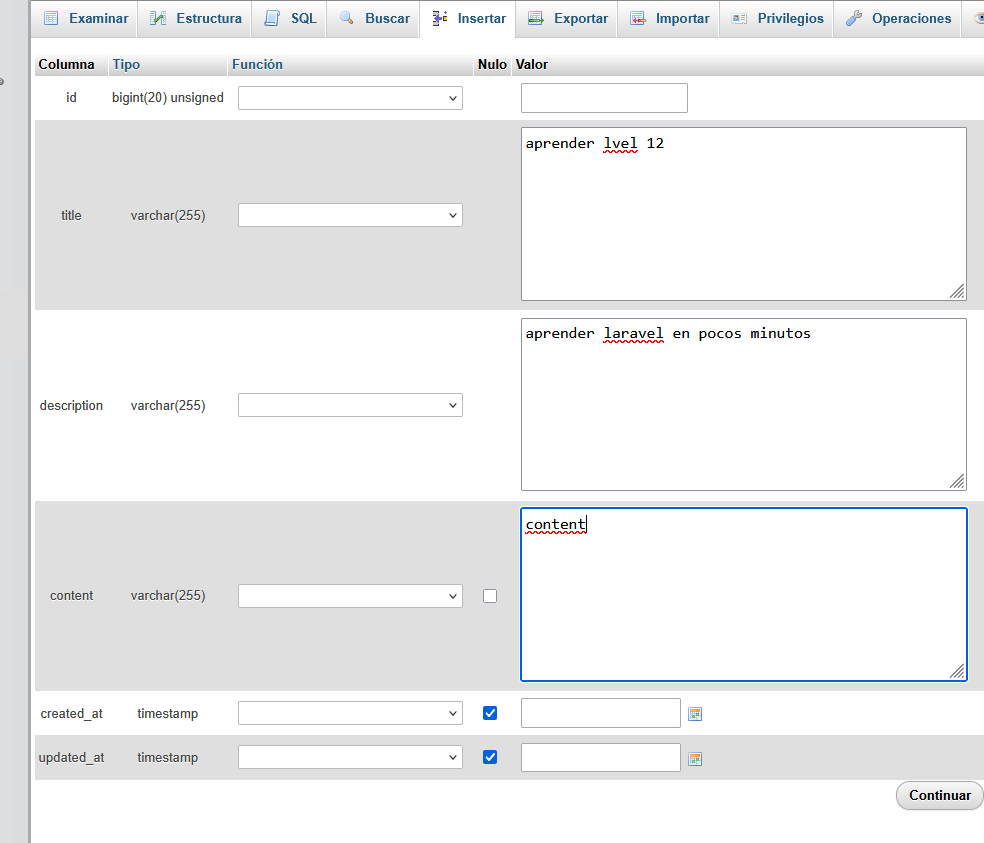
Creación de modelo y archivo de migración



Migración

Php artisan migrate

Create y updated son campos que se llenan cuando se hace con código de php, pero no cuando se hace el proceso manual.

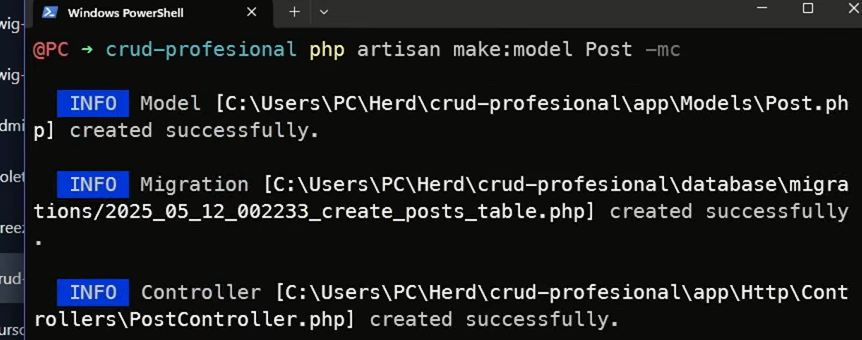


# CRUD Laravel

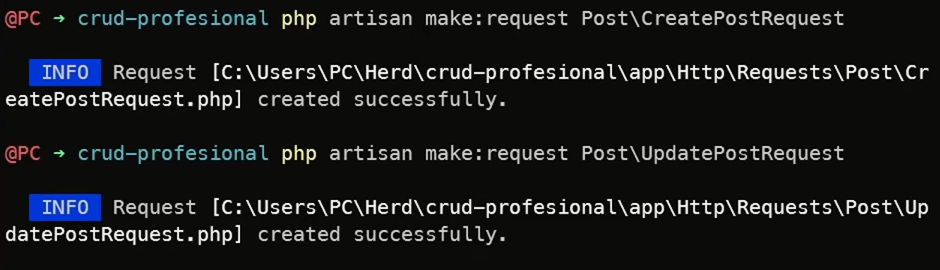
1. Mediante el modelo se maneja la información de las tablas mediante código php.
2. La migración crea la forma de la tabla.



Crear el modelo, migración y controlar de una sola vez.

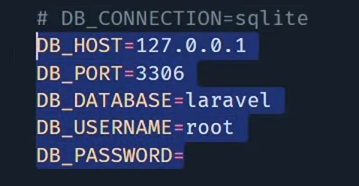


Creacion de FormResquest:



Uno para crear y otro para actualizar la request.

EN laravel 12, esta listo para trabajar en sqlite.



Necesitamos un controlador de recursos un ResourceController:

Este comando no nos sirve: php artisan make:model PostTwo -mc

Este si: php artisan make:controller PostTwoController –resource

Tenemos que separar responsabilidades idealmente:

<https://youtu.be/R94WgaWQJjo?si=rKFnL02Psc8Ddm-5&t=815>



Aquí debemos separar   
la validación de   
la lógica de creacion y   
la repuesta de redirección en un solo lugar.

La validación va en el Form Request

Para la lógica de creación crearemos una ServiceClass:  


En la service Class encapsulamos toda la lógica relacionada con una entidad, en este caso la entidad PostTwo.

Los servicios se crean en:  


La ruta app/Services/<Entidad>/<Entidad>Service.php

En este caso el controlador solo se encarga de mantener el flujo.

public function store(CreatePostTwoRequest $request)

    {

        $this->service->create($request->validated());

        return redirect()->route('postsTwo.index')->with('message', 'Post creado exitosamente!');

    }

Antes de seguir crearemos las vistas minimas para seguir con el proceso.

Escribimos el comando: php artisan make:component Layout