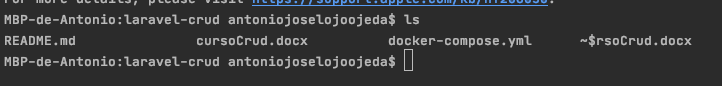
CRUD

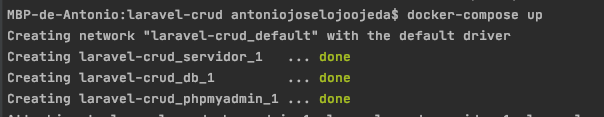
0 PREPARACIÓN

1º Creo un **directorio para**  **ubicar docker-compose.yml**



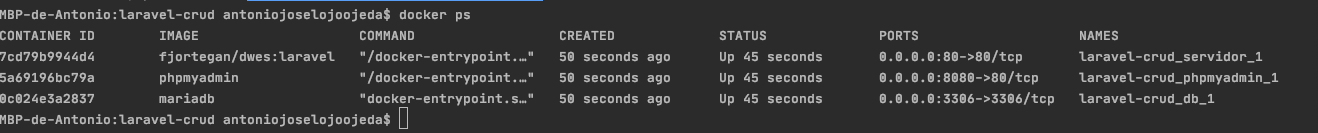
Este va ha levantar 3 contenedores. Uno con debían y nginx (**donde se creará el proyecto**)**,** un segundo con **mysql** y un tercero con **phpmyadmin** que jará más sencillo el trabajo con la BBDD.

2º **docker-compose up**



Ya tenemos en el directorio dos directorios **db-init** y **laravel**.

Además de tres contenedores funcionando. (bd, phpmyadmin, servidor)



Para crear el proyecto en el directorio del **root**

(el nombre se mira con **docker ps**)

**docker exec -it laravel-crud\_servidor\_1 /bin/bash 🡪** Ejecuta el bash del servidor

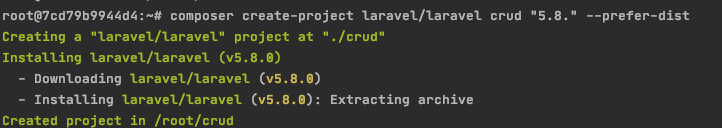
****

**3º CREAR EL PROYECTO**

Vamos a **root** y desde aquí **ejecuto el comando para crear el proyecto.**

**composer create-project laravel/laravel crud "5.8.\*" --prefer-dist**

(Crea el Proyecto laravel/laravel en /laravel con la version 5.8.lo que sea)

****

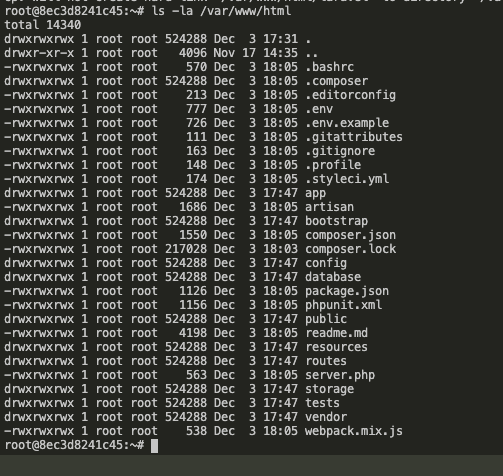
**4º DESPLEGAR EL PROYECTO**

A continuación, copiamos todo al directorio donde publica **nginx** y ya estaría laravel funcionando.

**cp -r laravel/\* /var/www/html**

**cp -R laravel/.\* /var/www/html**

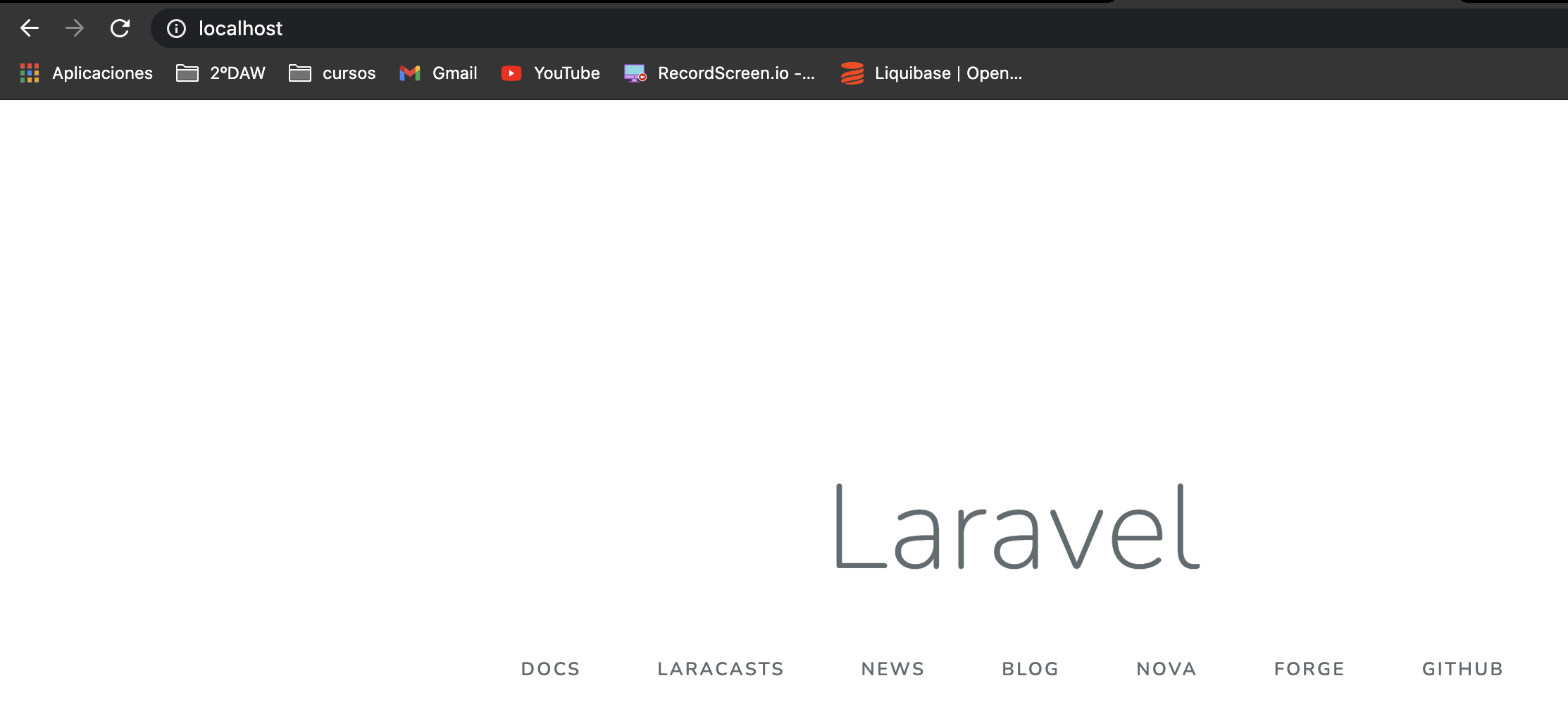
****



A plicamos permisos para poder trabajar con **nginx**.

**chown -R www-data:www-data /var/www/html**

****

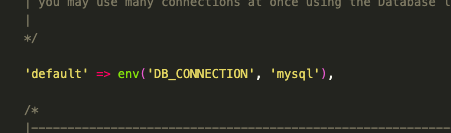
****

1 DISEÑO DE BASE DE DATOS Y CREAR PROYECTO

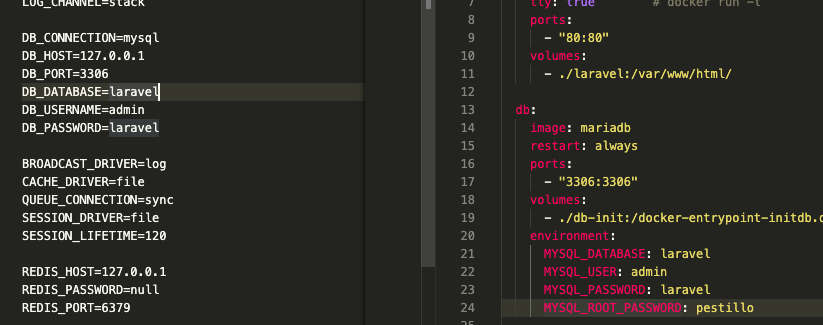
**0 CONEXIÓN CON MYSQL (MIGRATE)**

Ya podemos empezar con VSCode u otro IDE. Comenzamos comprobando que estamos usando **mysql**

**Config/Database.php**

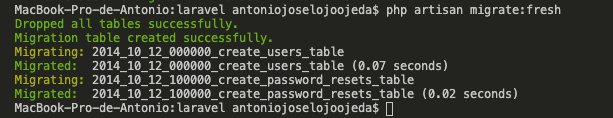
****

Comprobamos que en **.env** usamos la misma configuración que en **docker-compose.yml**

****

**Hacemos la migración.**

**php artisan migrate:fresh (limpia artisan)**

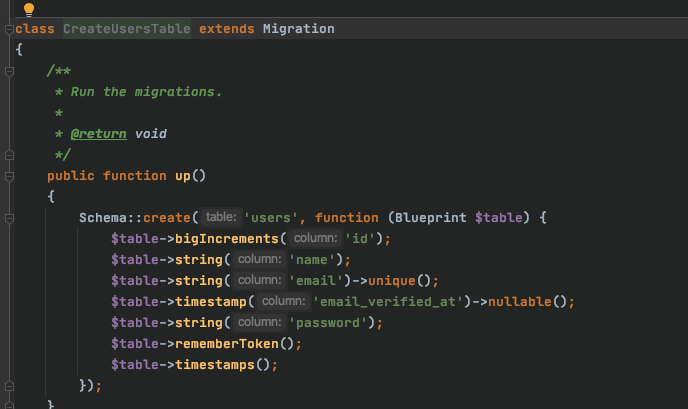
****

Estos están en **Database/migrations**

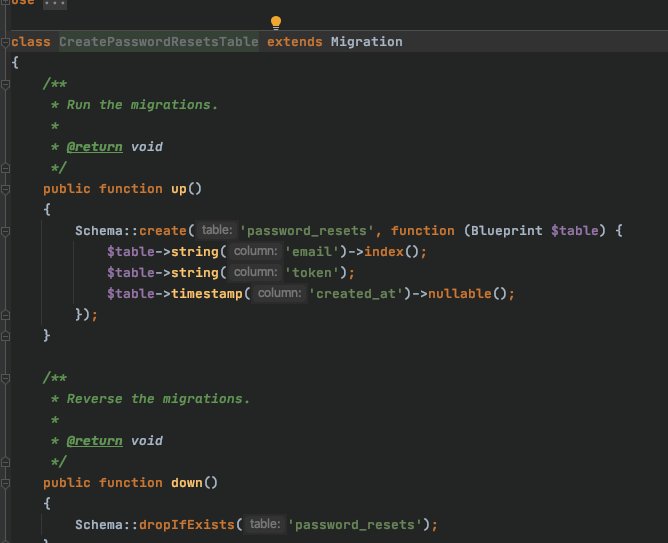
****

Ya existen dos migraciones listas **create\_users\_table, create\_password\_resets**

**createUsersTable**

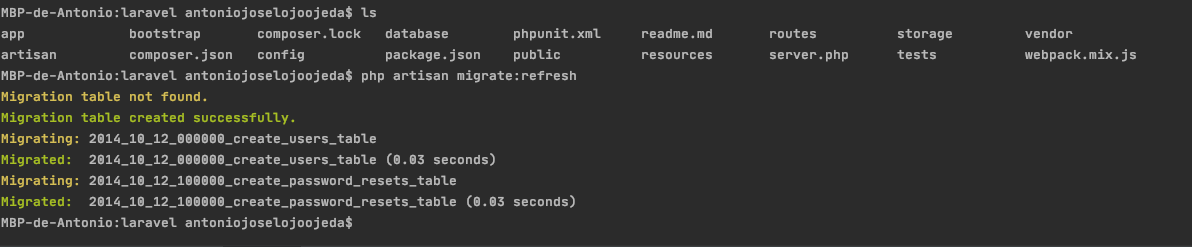
****

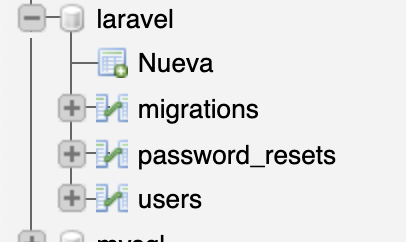
**Create\_password\_resets:table**

****

Para ejecutar todas las migraciones usamos el comando **php artisan migrate:refresh**

Este comando **laravel lee todos los datos de las tablas en el directorio migrates y los inserta en la base de datos.**



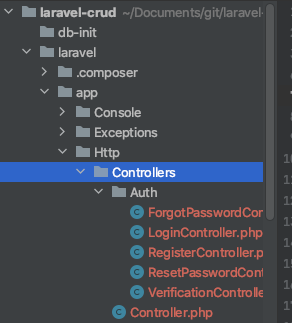


MODELO VISTA CONTROLADOR

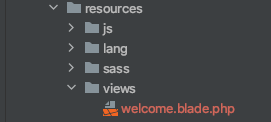
RUTA 🡪 Controlador (Puede hacer uso de un modelo que a su vez está creado a partir de una tabla de la base de datos) 🡪 vista 🡪 Renderizar un html

**¿DÓNDE ESTÁN ESTAS CARPETAS EN LARAVEL?**

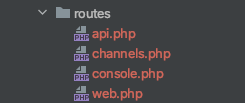
**Controladores 🡪 app/http/controllers**

****

**Vistas 🡪 resoruces/views**

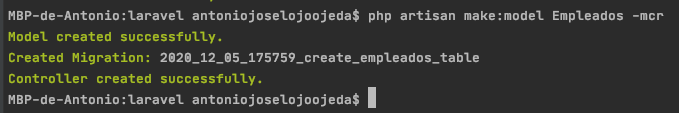
****

**Rutas routes/**

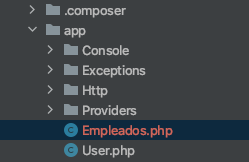
****

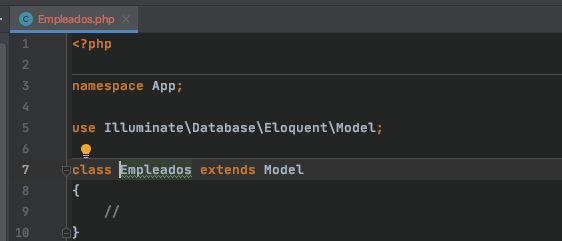
**VAMOS A CREAR ESTA ESTRUCTRA**

**php artisan make:model Empleados -mcr** 🡪 Construye el modelo Empleado junto a su controlador y recursos necesarios para empleados

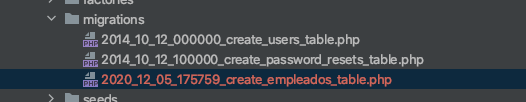


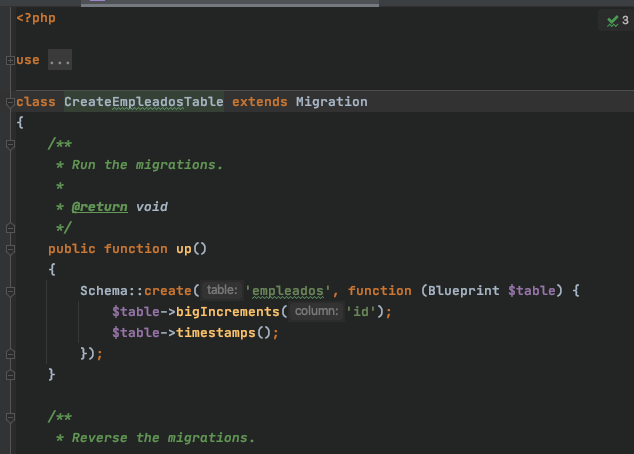
Nos indica que **se ha creado** **un modelo**, una **migración** y un **controlador**



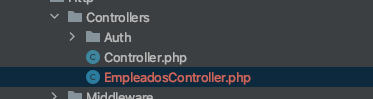


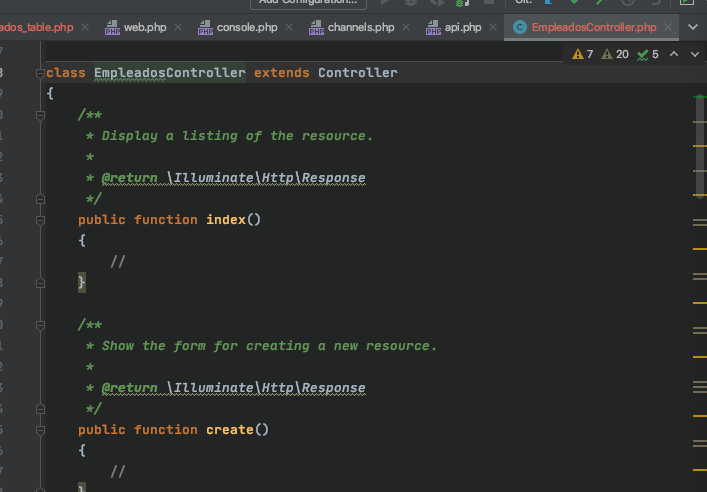
**Una migración 🡪 Esta migración es la que vamos a tener que modificar para poder generar la estructura de la tabla sobre la que vamos a trabajar**





Un controlador

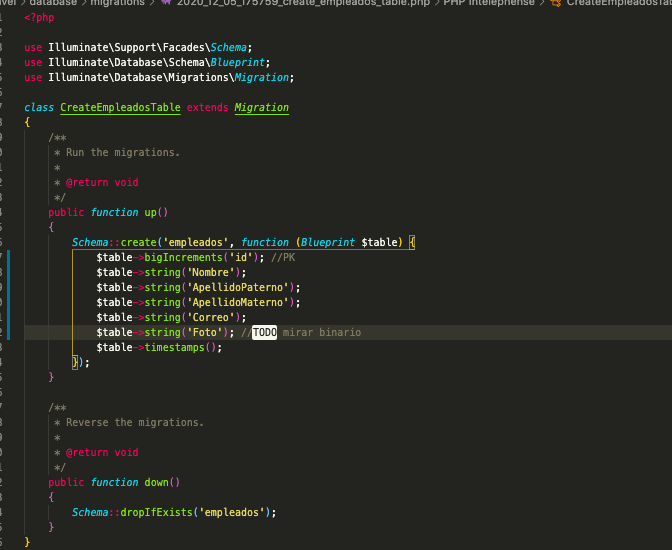




En el controlador nos crea una clase con todos los métodos para trabajar sobre una entidad **index() create() store() show() edit() update() destroy()**

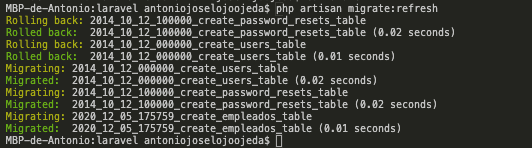
MIGRATE

Vamos a **/databasemigrations/create\_empleados\_table.php** Para modificar la tabla de empleados.

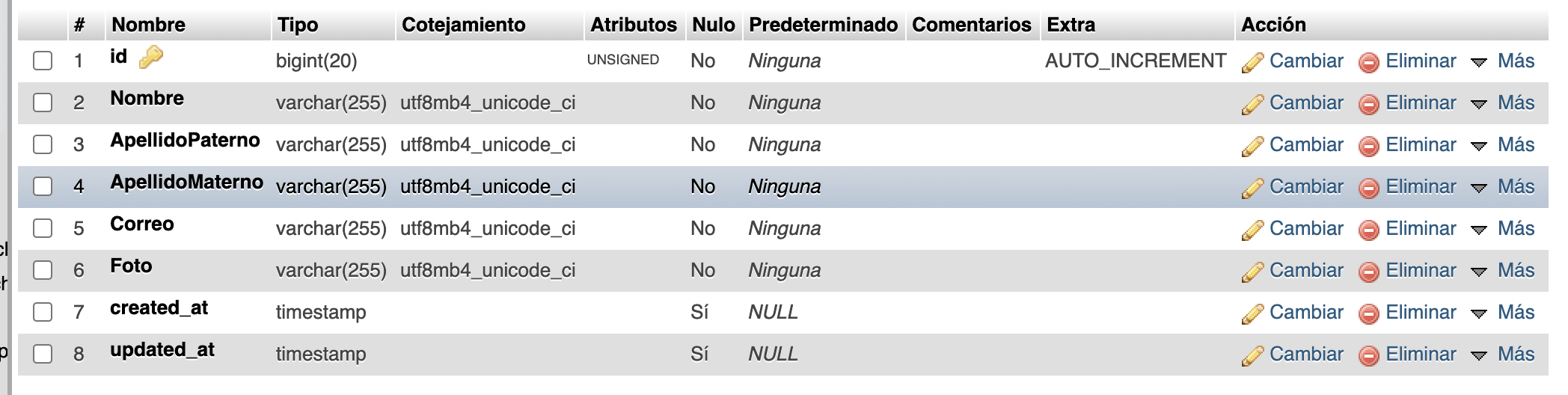


Ejecutamos la **migración para que se aplica a la base de datos**

**php artisan migrate:refresh**

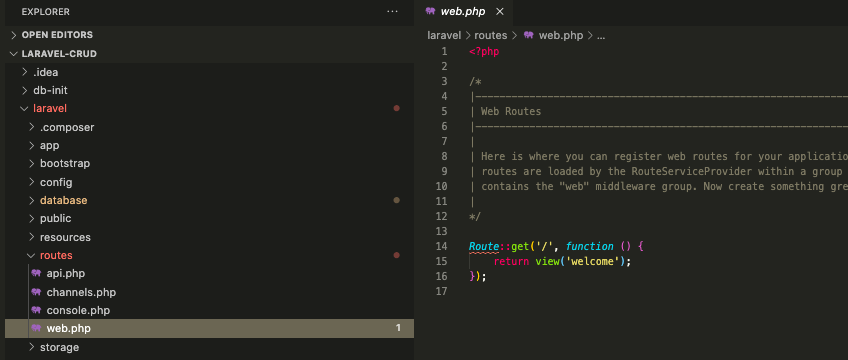
****

Comprobamos por si acaso con **phpmyadmin**

****

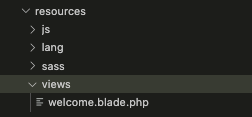
RUTA

Vamos a **routes/web.php**

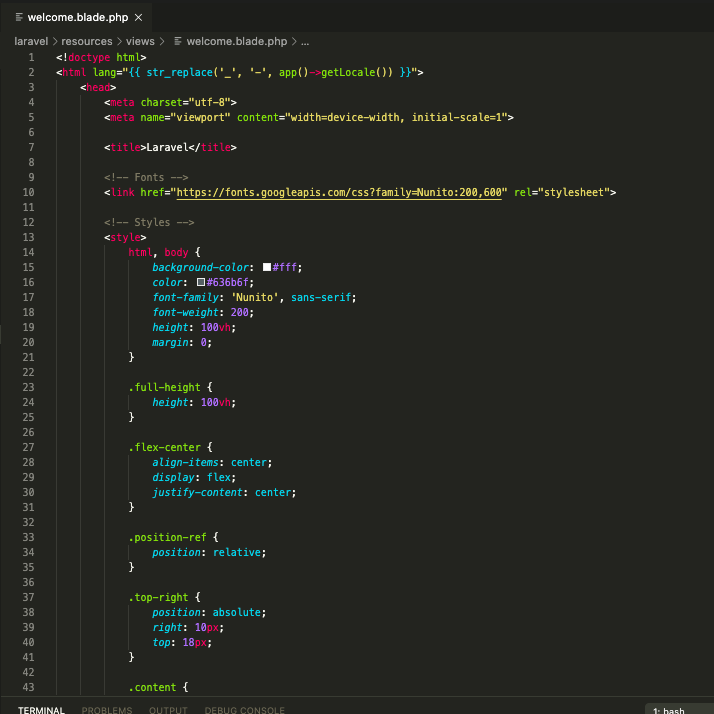
****

Aquí vemos que hay una ruta **\ (Raiz)** que nos devuelve un **vista** que se llama **welcome**

Este es el **index** por defecto **de laravel**. Este **index** está ubicado en **resources/views**

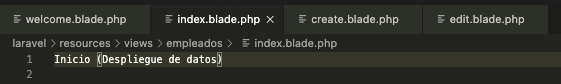
****

Corresponde a **welcome.blade.php**

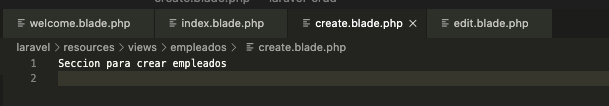
****

Para crear las nuestras vamos a iniciar creando una carpeta que se va ha llamar **empleados** dentro de esta **colocamos tres archivos (PLANTILLAS)**

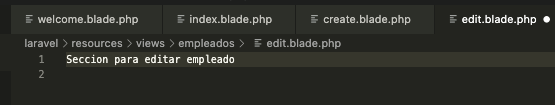
**index.blade.php** 🡪 Mostrar el contenido de index



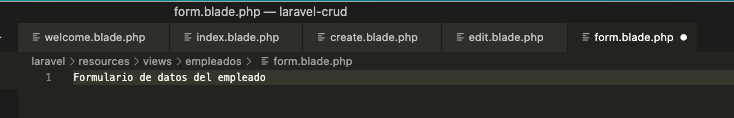
**create.blade.php**

****

**edit.blade.php**

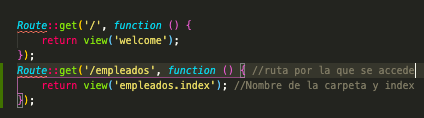
****

**CONTINUAMOS CREANDO UN FORMULARIO QUE VAMOS A USAR TANTO EN EDIT COMO EN CREATE**

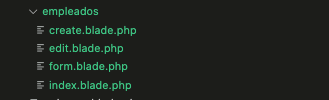
****

**PARA MOSTRAR LA INFORMACIÓN HAY QUE AÑADIR UNA RUTA IMAGINEMOS QUE QUEREMOS QUE SEA CON LA RUTA /empleados y que queremos que se vea un index**

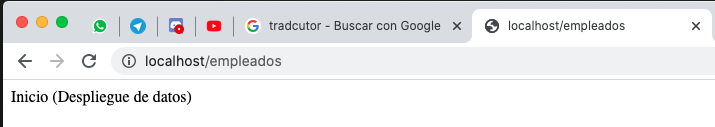
**routes/web.php**

****

**Empleados.index 🡪** Hace referencia a **index.blade.php**

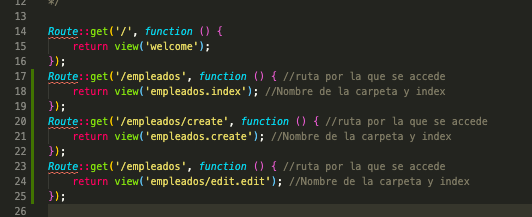
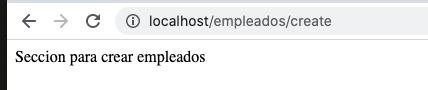
****

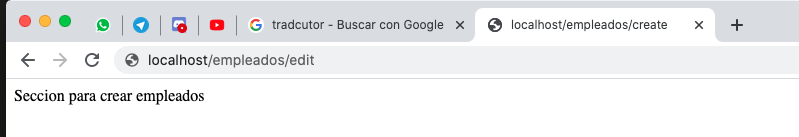
Si vamos al navegador y introducimos la ruta nos muestra lo que pusimos en la vista

****

Hacemos lo mismo con **create edit**

Teniendo en cuenta que están haciendo referencia al emnpleado se deben incluir en **/empleado**.

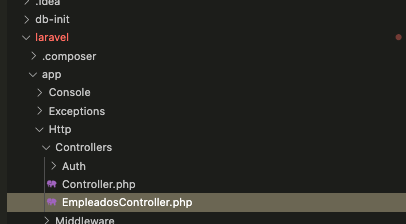
 



**¿CÓMO ACCEDER A LOS MÉTODOS QUE ESTÁN DENTRO DEL CONTROLADOR?**

Vamos al **controlador de los empleados**

/app/http/ccontrollers/EmpleadosController

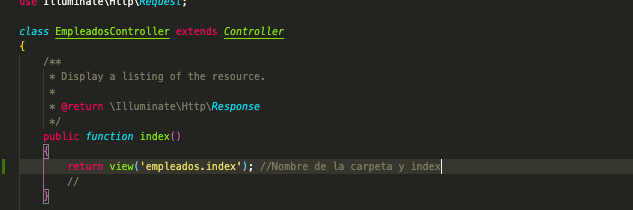


Queremos que el controlador sea el que llame a **la vista de index**.

Nos llevamos el mismo return en la función de rutas de **empleados.index**

****

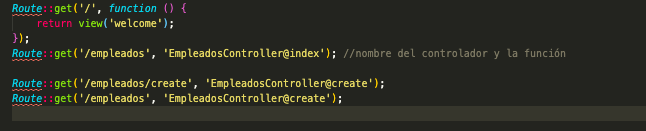
El return se lo damos a la función **index()** del controlador de los empledos.



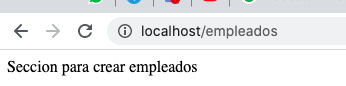
Hacemos lo mismo con **create y edit**

¿**CÓMO HACEMOS PARA QUE LA RUTA MUESTRE EL MÉTODO DEL CONTROLADOR QUE ES EL QUE VA HA LLAMAR A LA VISTA**?

Para ello usamos la siguiente sintáxis:



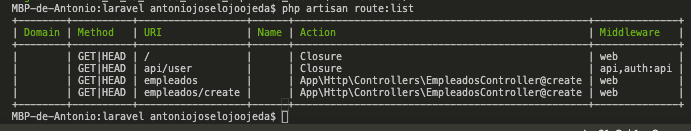
‘ruta en el navegador’, ‘controlador@función’



Sigue funcionando bien eso es **muy bien**

**PARA VER LAS RUTAS A LAS QUE PODEMOS ACCEDER TENEMOS EL COMANDO**

**php artisan route:list**



A continuación podríamos poner todas las rutas que vamos a necesitar. Pero si no las queremos poner todas podemos usar algo muy singular y que vamos a usar bastante.

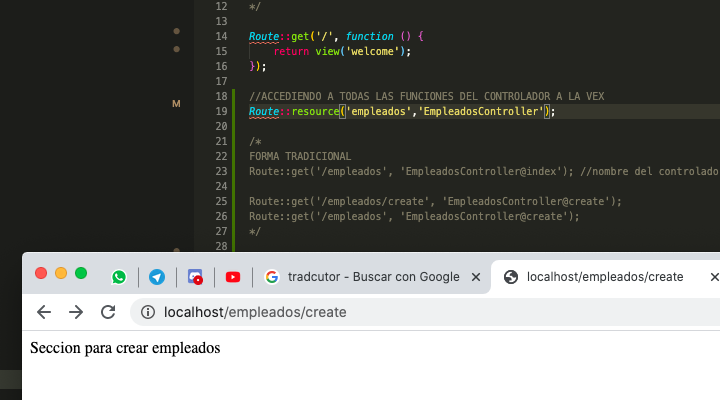
**Route::resources(‘user’, ‘UserController’)**

**‘users’ =** colocamos el nombre de la ruta principal del navegador

**‘UserController’ =** colocamos el nombre del controlador

De esta manera **laravel va ha llamar automáticamente al método del controlador siempre que este coincida con la ruta introducida**

Por ejemplo **localhost/empleado/create 🡪 EmpleadosController@create()**



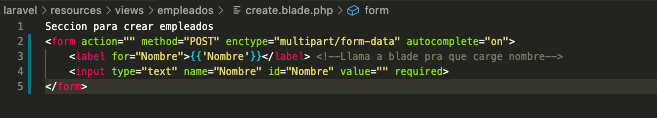
FUNCIONAS BIEEEN ESO ES BIEN

VISTA CREATE

Vamos a crear el formulario para la información del empleado que se va ha ingresar en la base de datos.

**Vamos a la vista create.blade.php**

Creamos el formulario de empleado



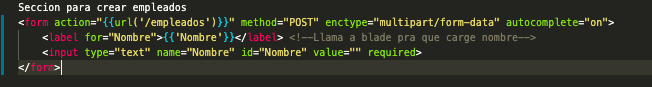
**¿HACIA DÓNDE SE HACE EL ENVÍO?**

Si hacemos **php artisan route:list**



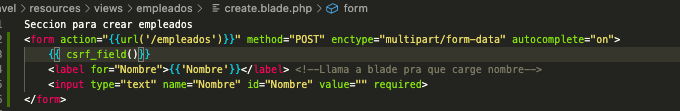
Vemos que hacemos el **método POST** al enviar a empleados y que nos lleva a **storage**

Nosotros usaremos una URL (**con el método blade**) a empleados



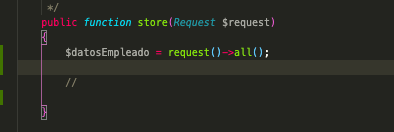
Así enviaremos a /empleados por POST de todos los empleados.

Para que esto funcione vamos a usar **una función de laravel que genera un token necesario para poder enviar POST si no es imposible**



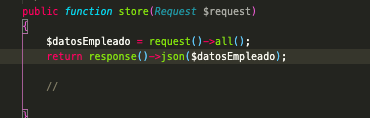
A continuación vamos a **storage()** de nuestro controlador que es donde enviamos por **POST** la información.

Si nos fijamos el parámetro es **$request** es la petición que **se genera al enviar los datos por POST**

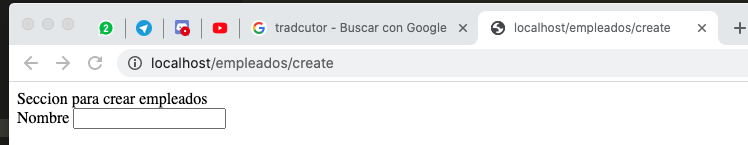


Con esto estamos accediendo a **todos los datos que hemos enviado por la petición POST al enviar el formulario.**

Para comprobar lo que se envía en lugar de una vista vamos a devolver una respuesta **json**

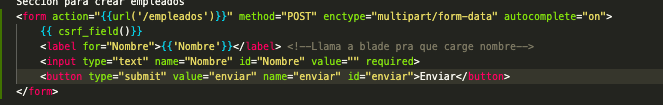


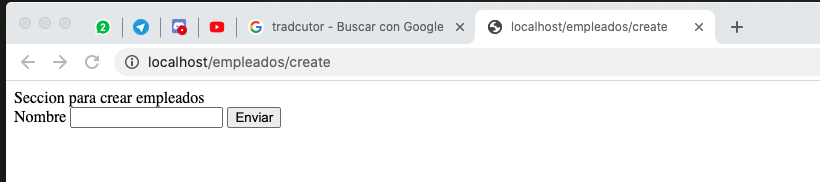
DEBERÍA DE DEVOLVERNOS UN DICCIONARIO CON LA INFORMACIÓN DEL EMPLEADO ESCRITA EN LA PANTALLA COMPROBAMOS CON EL NAVEFADOR



Carga correctamente el formulario de la vista create.blade.php que ha llamado el controlador EmpleadosCotroller@create() 🡪 VAMOS BIEN

(ME FALTA EL IMPUT ☹)

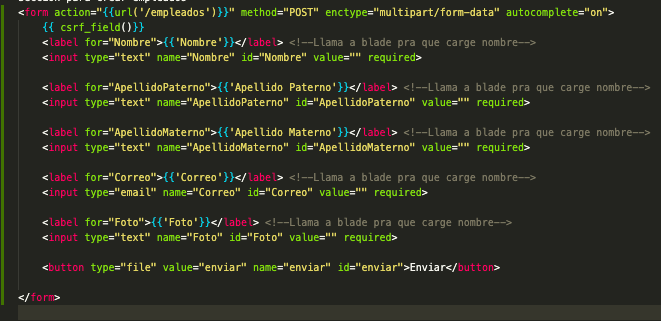




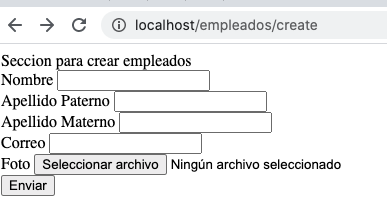


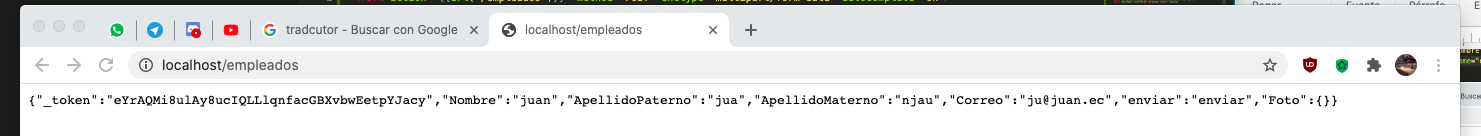
Ahora sí vemos que tenemos los valores de todos los campos genial (**Incluyendo el token**).

Una vez que ya sabemos que funciona continuamos agregando más campos.



Comprobamos si funciona





**Todo GENIAL.**

INSERTAR DATOS Y SUBIR FOTO

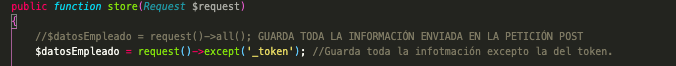
Necesitamos aislar la información y hacer que coincidan en la base de datos.

**Vamos al controller**.

En lugar de **$variable = request()->all();**

Le indico lo siguiente:

**$datosEmpleado=request()->except(‘\_token’)** Para que nos guarde toda la información de la petición menos el **token**

****

Para insertar los datos llamamos al modelo **Emleados::insert($array)**

****

[**https://www.youtube.com/watch?v=PO-DmjDAqQo**](https://www.youtube.com/watch?v=PO-DmjDAqQo)

**min 2:19**