Tener laravel y php descargados e instalados. En consola hacer

*composer create-project laravel/laravel toDoApp*

Esto crea la carpeta con la estructura de carpetas necesaria. Después meterme dentro de la carpeta y en esa ruta, en consola, hacer

*php artisan serve*

Inicia el servidor en la ip <http://127.0.0.1:8000/>, ya puedo acceder por allí en el navegador, sin xampp, ni nada, me envía a la página de inicio de laravel

Creo la base de datos en mysql de mi equipo, asi que en **.env**, relleno asi los datos

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=to\_do\_app

DB\_USERNAME=root

DB\_PASSWORD=

(la va a encontrar por el puerto, luego si exportamos o lo que sea, ya cambiaríamos el puerto supongo. Luego dentro del puerto además la encuentra por el nombre)

php artisan migrate

Crea las tablas y pone a la bbdd la estructura básica para el proyecto

Php artisan make:model Category –m

Crea el modelo de category

**app/Models/Category.php**

y la migración de category

**database/migrations/xxxx\_xx\_xx\_create\_categories\_table.php**

puedo entrar en la migración a modificar, y cambiar esto (hemos añadido lo que esta en amarillo que es la columna que quiero que tenga esta tabla, además de id y timestamp, que se crea solo y ya venia)



Lo mismo, para otras clases, creo el modelo, se crea el modelo y la migración, y entro en la migración, a añadir las columnas que quiero que tenga esta clase, en la base de datos. En este caso se añade todo esto

Php artisan make:model Task –m



Ver que hay dos que son fk, una con usuario que además, si se borra el usuario, se borran todas las tareas en cascada, y otra con category que es la clase que habíamos creado justo encima. Esta es nullable, y si se borra se pone en null (osea, que puede tener categoría asociada la task o no, si es que no se pone en null y no pasa nada)

Una vez tengo todo esto rellenado, otra vez en consola

Php artisan mígrate

Y me crea las dos tablas que acabo de configurar, Task y Category

Ahora que ya están creadas, vamos a modificar los tres modelos

app/Models/User.php



Lo mismo en el modelo de category



Y en el modelo de tasks



User has many tasks (relacion 1:N)

Category has many tasks (relacion 1:N)

Task belong to user y task belong to category (lado 1 de la relacion 1:N)

Ahora que ya tenemos la migración rellena y hecha, y el modelo con los métodos (creo que el modelo incluye los métodos que vienen de serie para la bbdd del crud, y aparte el que le hemos puesto personalizado que es mostrar las tareas de un usuario, de una categoría, o mostrar a que usuario o a que categoría pertenece una tarea)

Ahora por rellenar con algún dato las tablas. Hacemos los seeders

Consola php artisan make:seeder UserSeeder

Crea database/seeders/UserSeeder.php

Entro a modificarlo, y pongo lo que esta en amarillo

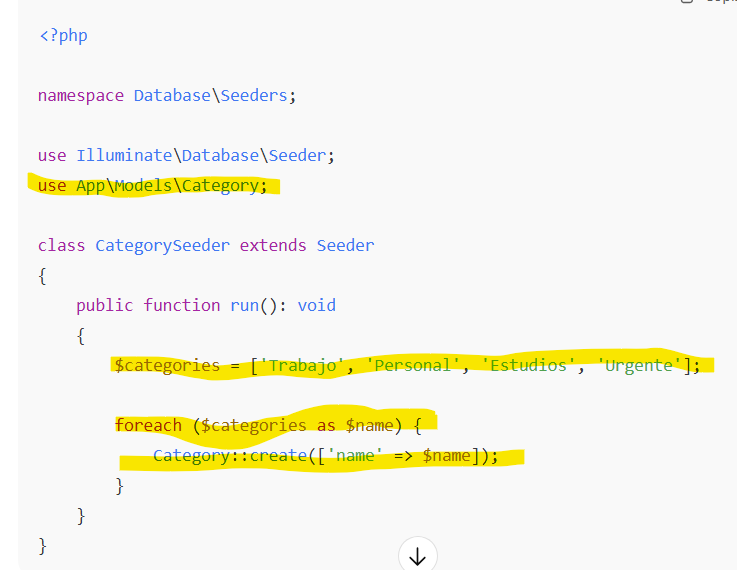


Como user es una clase que se ha creado automática con el proyecto, no tengo que poner nada, directamente ya existe un factory para crear users user::factory() asi que lo puedo usar para crear 3 usuarios

Ahora para categorías

Php artisan make:seeder CategorySeeder

Editamos el archivo CategorySeeder que se ha creado dentro de database seeders



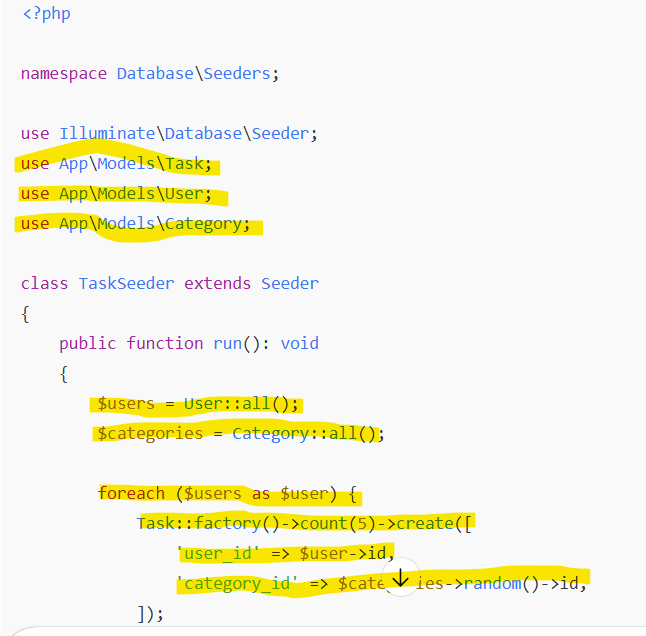
Aquí lo mismo que antes, le decimos el modelo a usar, pero además aquí al no estar creado el factory que hemos usado antes, si que hay que decirle, para cada category lo que tiene que rellenar. Recordar que category solo tenia una columa que era name, pues vamos a crear 4 categorias, que son las que hemos puesto aquí, quedando la tabla asi



Por ultimo para tareas

Php artisan make:seeder TaskSeeder

Modificamos task seeder y ponemos



Aquí, lo primero, que task va a usar a user y a category, asi que pongo el use con los dos, va a usar los dos porque en el método que he creado, le pongo que taskfactory (no existe, no como en user que si que existía, ahora lo vamos a crear) me cree una task donde el factory se va a inventar el nombre y la descripción de la task, pero que luego me ponga un user id de los user que tengo y un category id de las categorías que tengo.

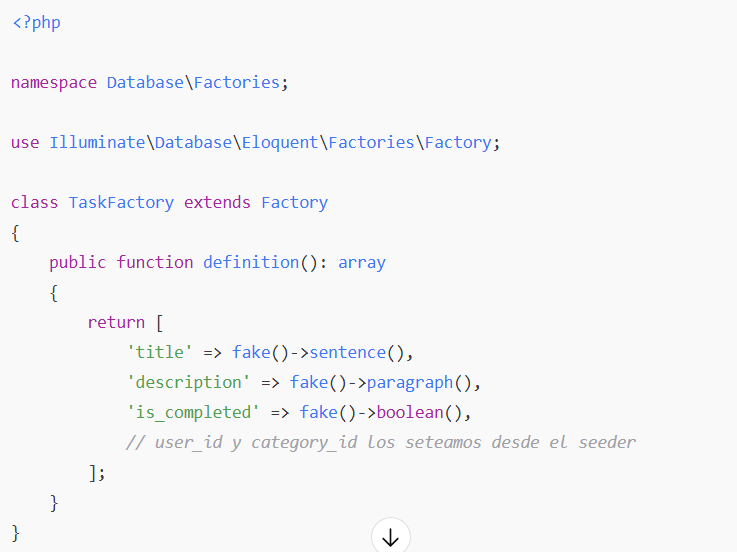
Y le digo que para cada user id me cree 5 tareas, con ese user id y con un categoría id random

Como tenemos 3 usuarios, esto me creara 5 tareas para cada uno de ellos. En total 15 tareas. 5 con cada user id, todas con un category id random, y el task factory que ahora vamos a crear les va a poner un nombre y una descripción random y is complete en false

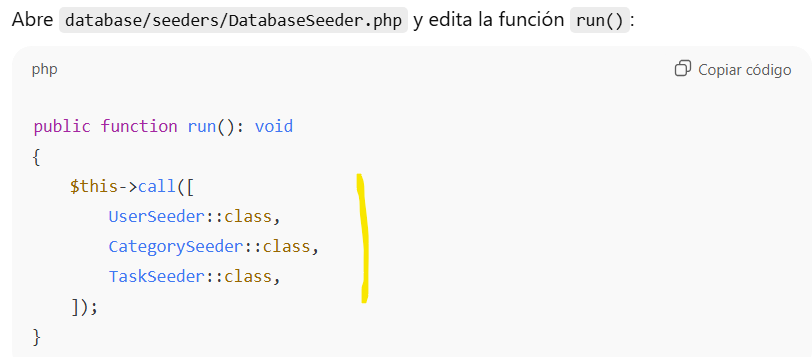
Para crear el task factory, una vez tengo esto cambiado, hago en consola.

php artisan make:factory TaskFactory --model=Task

esto me crea en **database/factories/TaskFactory.php** un archivo que voy a editar asi, para que contenga una funcion, que extiende de factory, y que lo que hace es devolver un titulo, descripción y is\_complete=false o true. Con esto junto con user\_id y category\_id que los poníamos en el seeder, ya se nos crea la task completa



Por ultimo, hay que abrir database/seeders/DatabaseSeeder.php y en la funcion run, añadir todos los seeders que acabamos de crear, asi



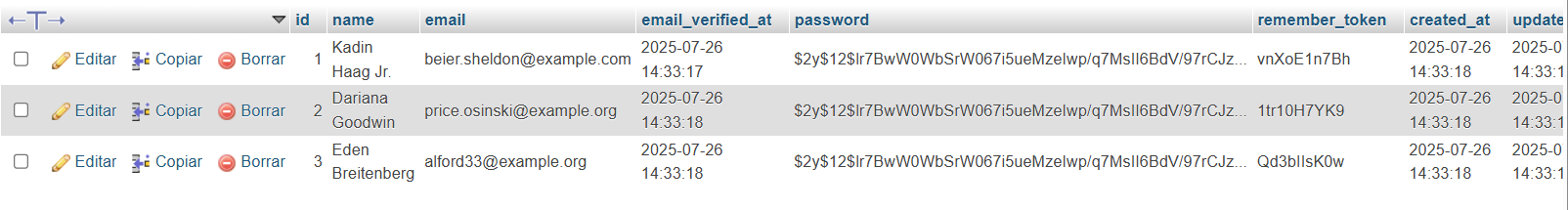
Por ultimo, una vez hecho esto hacemos en consola

php artisan migrate:fresh –seed

que borra todas las tablas, vuelve a correr todas las migraciónes y ejecuta los seeders

daba un error porque faltaba importar con use en el modelo de tasks para que pudiera usar factory, lo hemos hecho y ya esta

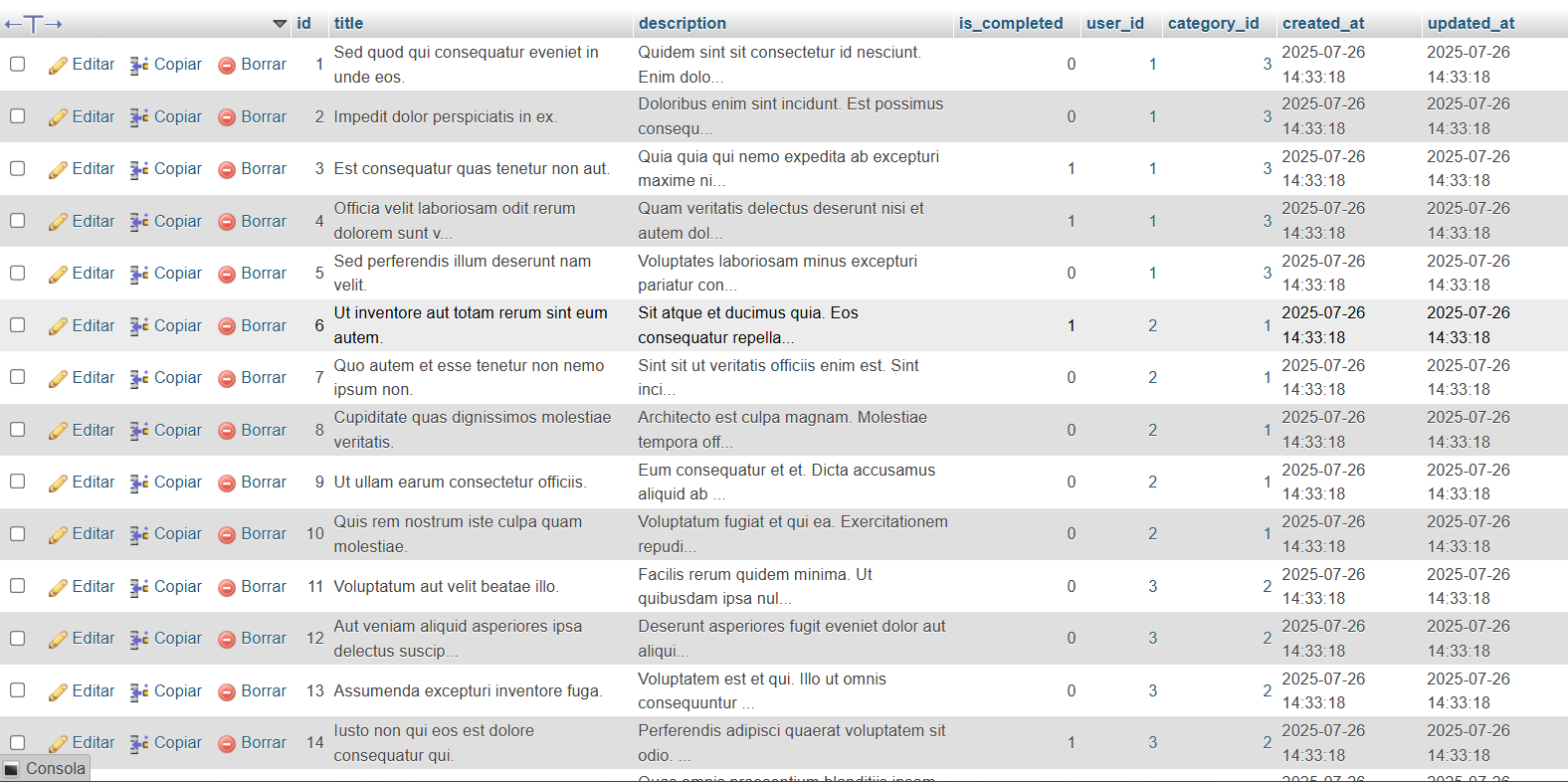
ha creado una tabla con 3 usuarios aleatorios creados por el factory



Una tabla con 4 categorias que son las que le hemos dicho



Y una tabla con 15 tasks, 5 para cada usuario, con una categoría aleatoria

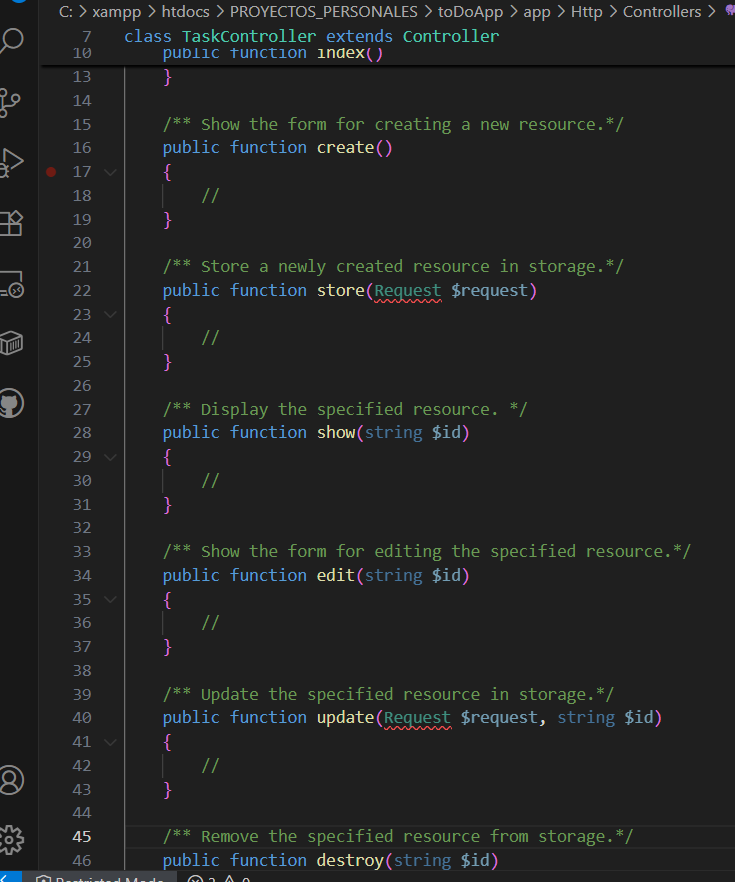


Ahora para el CRUD de tareas, creamos el **controlador de tareas**

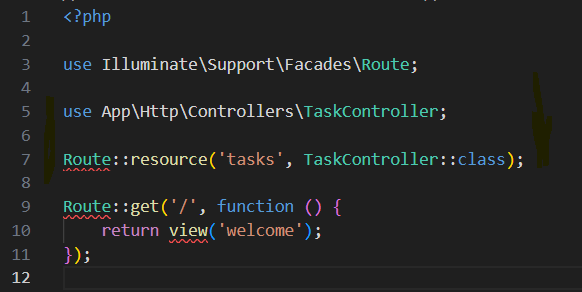
php artisan make:controller TaskController –resource

esto crea en **app/Http/Controller/taskController.php** esto

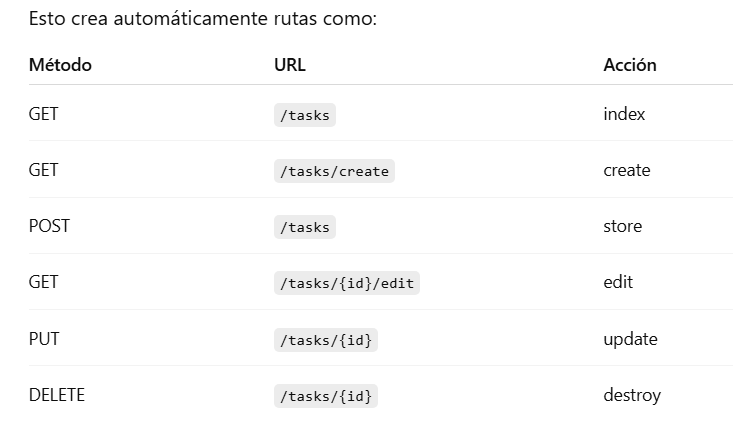
que viene con los métodos pero vacios, luego los rellenaremos



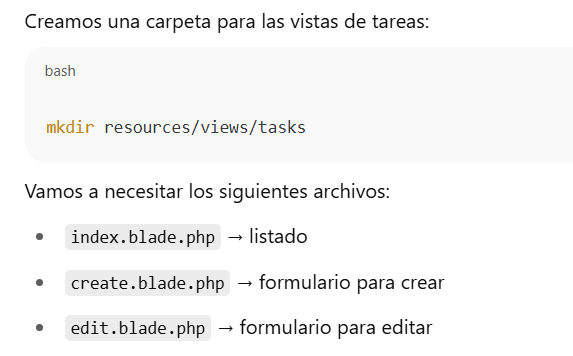
Ahora de momento, abrimos routes/web.php



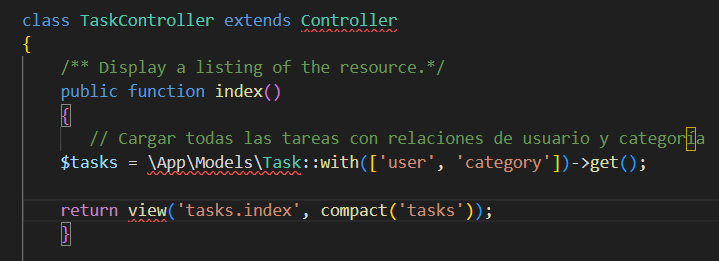
Y añadimos esto, lo cual



Y también vamos a crear las vistas.

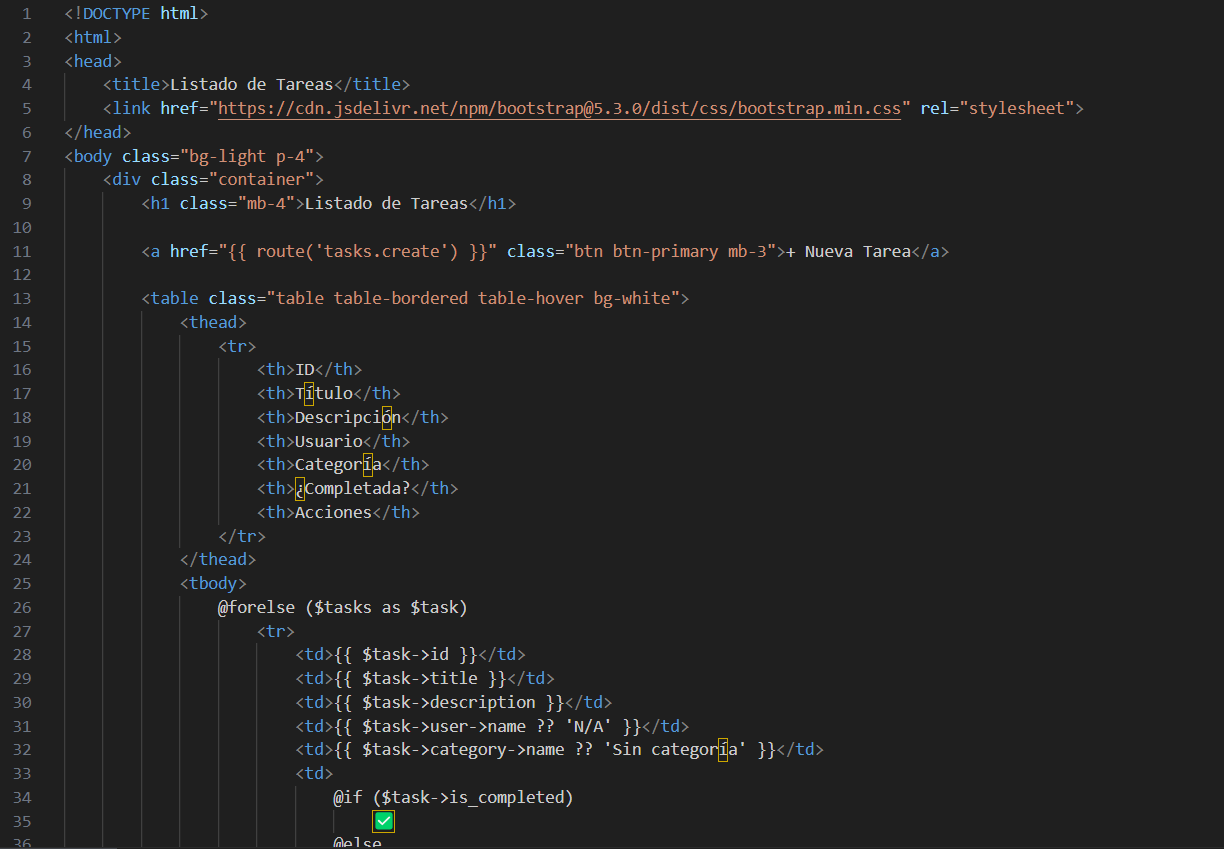


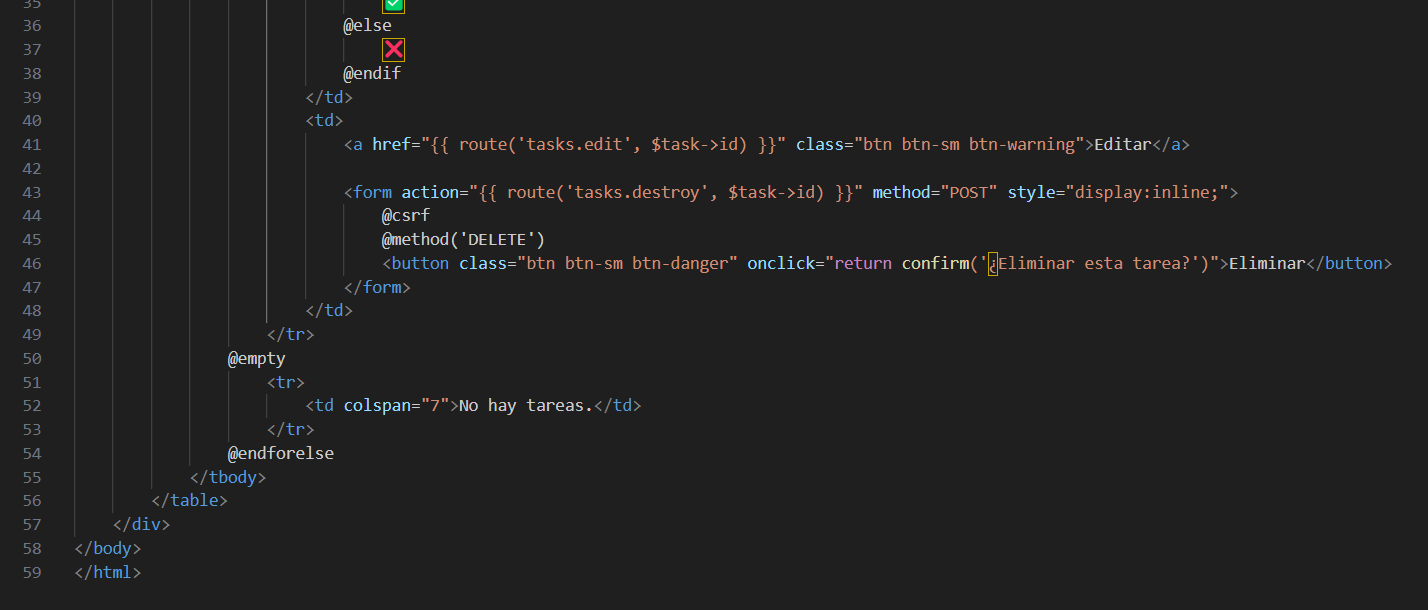
Ahora, en el método index de TaskController.php ponemos lo siguiente



Del modelo de tasks, cogemos todo lo que tenga user y category, y lo devolvemos en la vista tasks index.blade.php

Ahora en la vista tasks.index.blade.php, que habíamos creado en las vistas ponemos el siguiente código



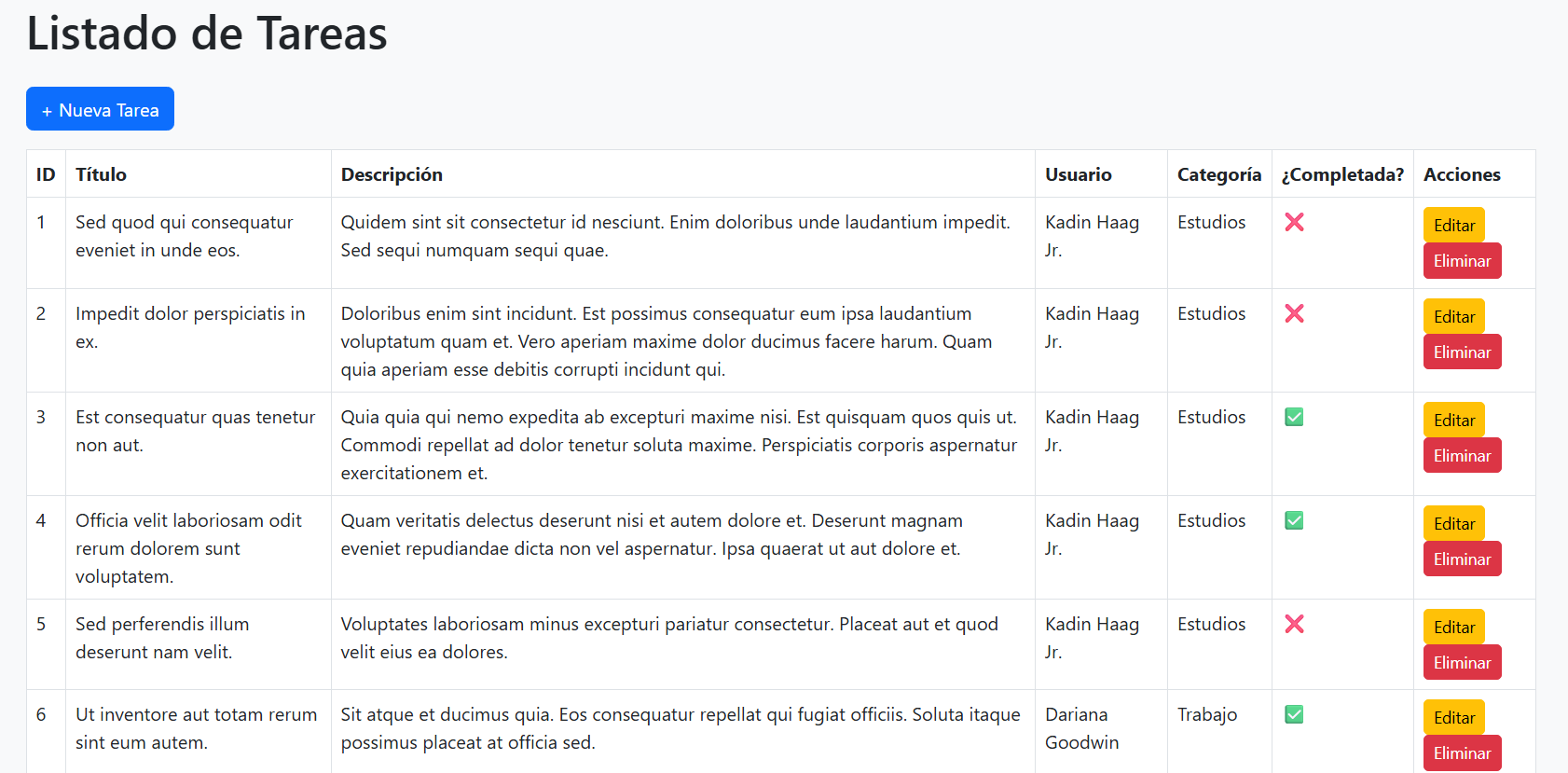


En el archivo de rutas, hemos puesto que route va a task controller, que es el que tiene los métodos, index, create, delete, etc que luego definiremos

Por lo que aquí, los botones que tienen un href a route tasks.destroy a route tasks.edit etc, tienen enlace a los métodos, del controlador de tasks, al que se llega por route porque en rutas hemos dicho que route era tasks taskcontroller y a los métodos del controller

Y luego para cada task mete una entrada en la tabla (recordar que aquí llegamos porque el controlador con el método index, coge todas las tasks y nos manda aquí con ellas) y aquí, se meten en una tabla, con los botones para llamar al resto de métodos

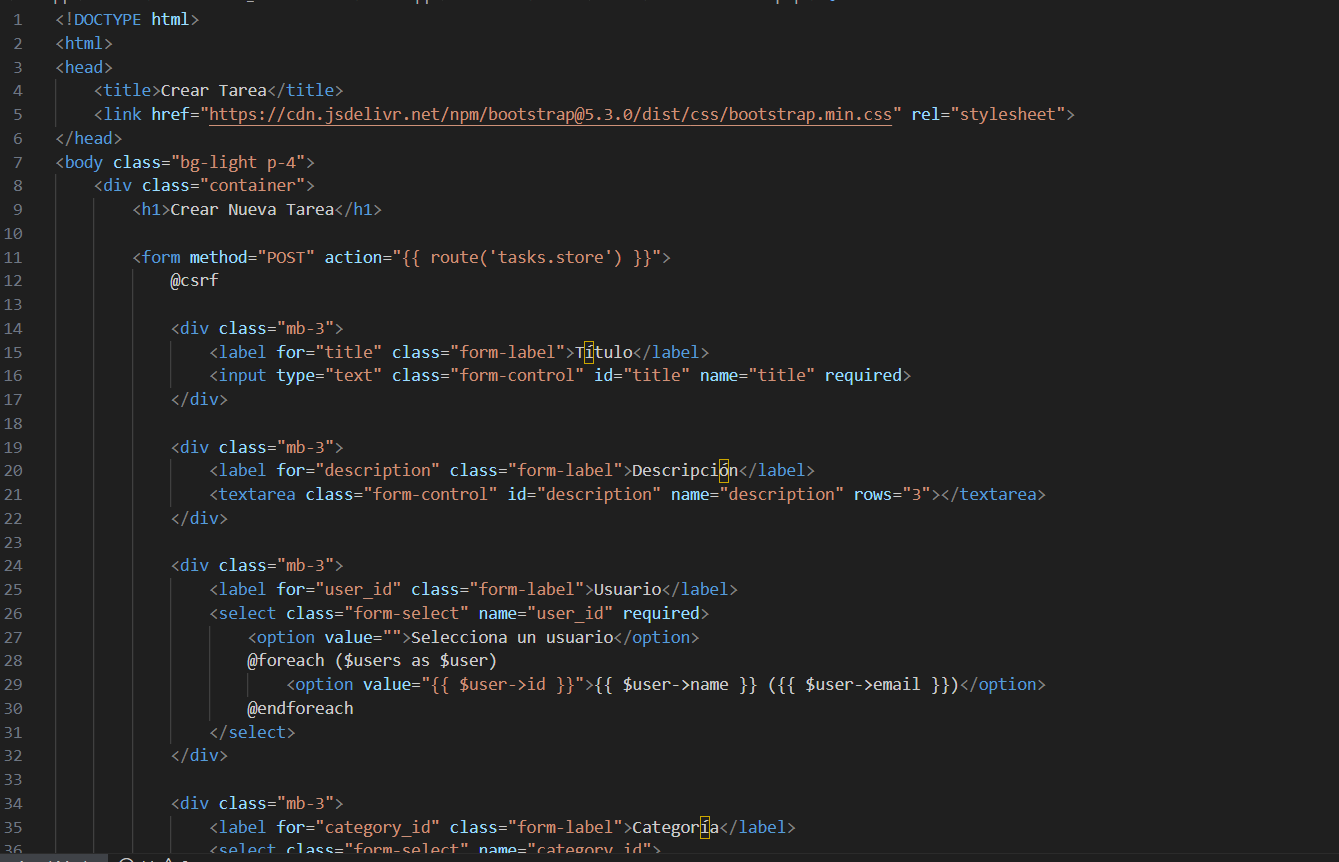
Ahora en el navegador puedo poner localhost:8000/tasks, y me saldrá esto

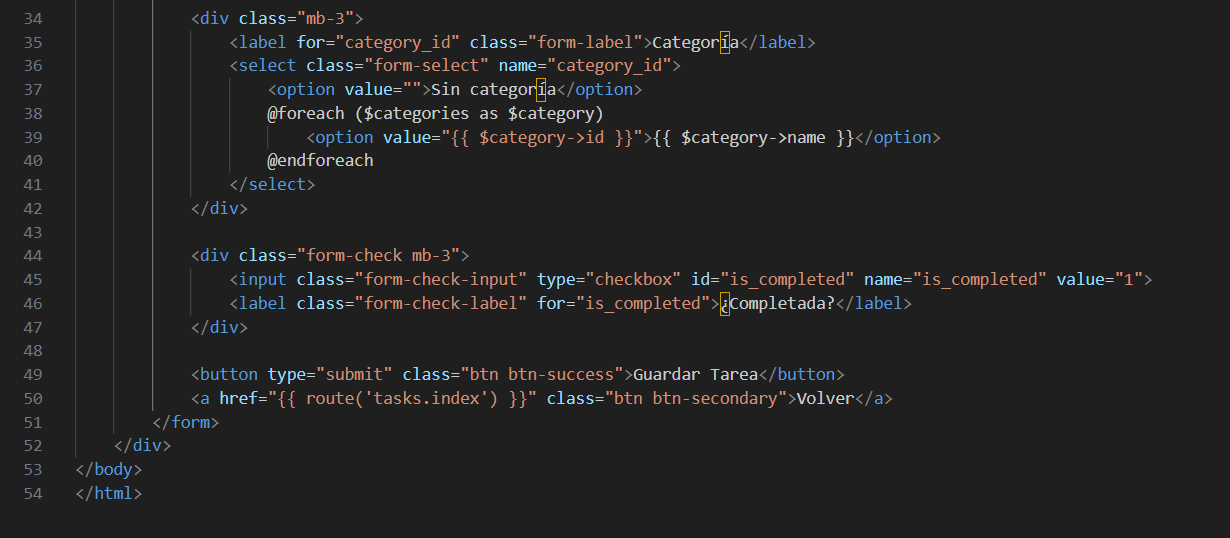


Ahora para crear una nueva tarea, tenemos que en el controlador TaskController.php rellenar el método create (que lo que hace es sacar una lista de usuarios y de categorías y cargarlas en la vista de tasks.create.blade.php)



Ahora en la vista de tasks.create.blade.php ponemos el formulario para rellenar los datos de la nueva tarea asi



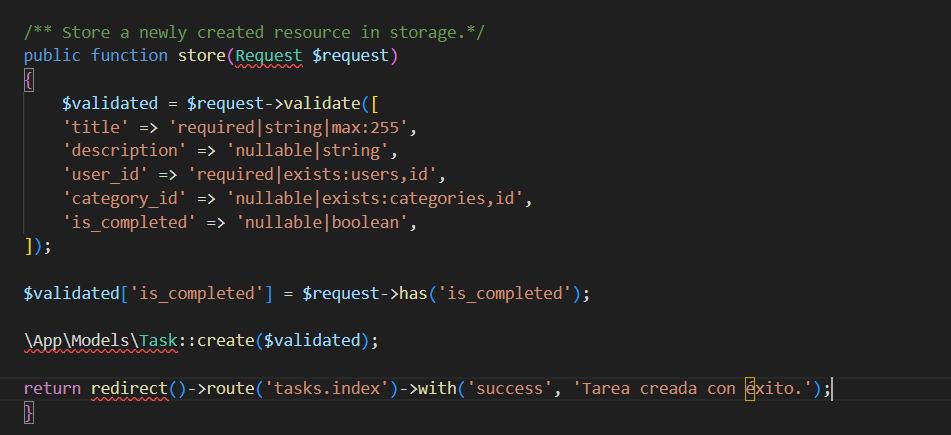


Esto tiene un formulario, que los datos que rellene aquí los pasara por el método post a TaskController pero al método store (es decir, create me recoge los usuarios y las categorías, me manda a esta vista, relleno todo el formulario y con eso vamos a store, que guarda en la base de datos)

El formulario tiene campos de texto donde poner nombre descripción y luego tienen unos select, donde importo user id user name user email y category id category name (con un foreach para cargar todos los que hay y mostrar todas las opciones en un selector) y aqui el usuario selecciona el name, y se pone el valor del id en el value

Por ultimo hay un botón que vuelve a llamar a tareas.index, para volver en caso de que el usuario no quiera guardar nada

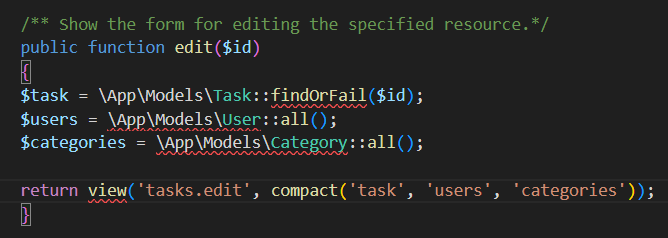
Ahora definimos en TaskController lo que hace la funcion store, con los datos que han venido del formulario



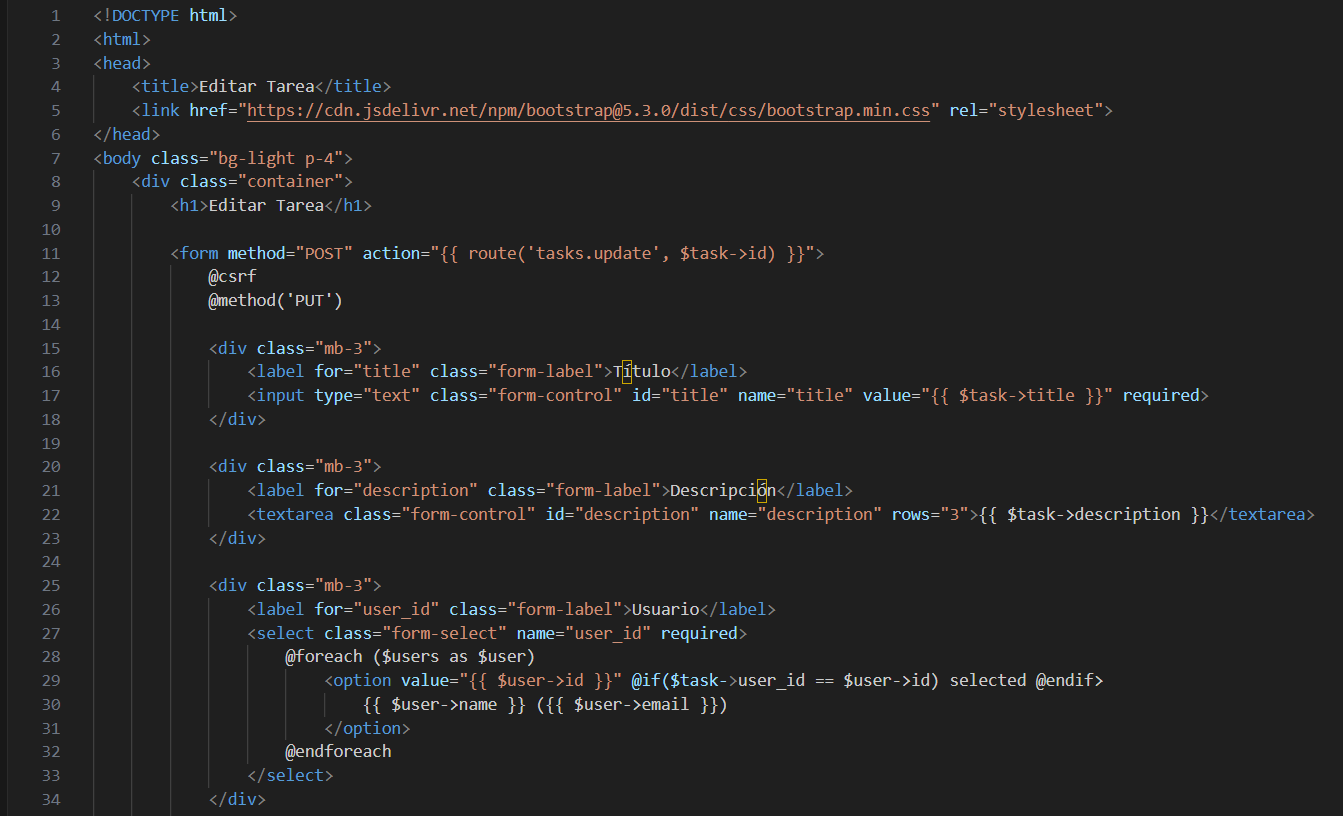
Que es validar los datos y pasarlo a model para que haga create (en model no esta definido create por nosotros, pero supongo que es la gracia de esto, que como model lo he creado junto con las migraciones, los métodos crud del model vienen ya creados)

Con esto listo, ya si le doy al botón de nueva tarea en localhost:8000/tasks, directamente me va a localhost:8000/tasks/create y aquí ya sale el formulario con el desplegable para usuario y categoría porque ya se han cargado, relleno el formulario le doy a guardar y se crea una nueva tarea que se guarda en la bbdd

Ahora que ya tenemos para crear una tarea, vamos a darle funcionalidad a editar tarea, de momento tenemos el botón que nos lleva a rutas y al controlador de tareas, por lo que lo 1º que hay que hacer es poner este código en el método edit() de task controller



Recibe un id como parámetro (lo coge del botón que ya va con el id de la tarea junto a la que sale), y lo que hace es en modelo find (otro de los métodos que tiene el modelo aunque no lo veamos porque se creo junto con la migración), con esto encuentra la tarea que quiero editar. Luego coge todos los usuarios y todas las categorías (para que se seleccionen, no se escriban por el usuario, a la hora de cambiarlos) y pasa estos 3 parametros, a la vista tasks.edit.blade.php (la tercera de las vistas que habíamos creado y la única que nos faltaba por editar, que ahora la rellanamos para que contenga el formulario de edición)





Un formulario, que va a route tasks.update (nos enviara los datos recogidos al método update del controlador, que luego rellenaremos (aquí habíamos venido con datos de la tarea desde el método edit)

De entrada tiene los values de task->name y task-> description pero si escribo cambiare el value al que ponga escrito

y luego desplegables para elegir entre todos los usuarios y todas las categorías, pero si taskid==id, ese id saldrá selected, y lo mismo con categoría, y value de primeras será category->id, pero si cambio que selecciono otro id, pues value cambiara por ese

por ultimo el botón de completada, pero si task->is completed, entonces marcar, y sino no marcar, y luego el usuario lo puede cambiar

por ultimo un botón para submit, y uno para cancelar, que nos lleva a rutas, al controlador, a mostrar tasks.index

cuando hago submit, me ire con los nuevos values que he puesto en el formulario, a task.controller, al método update, que ahora lo vamos a rellenar asi



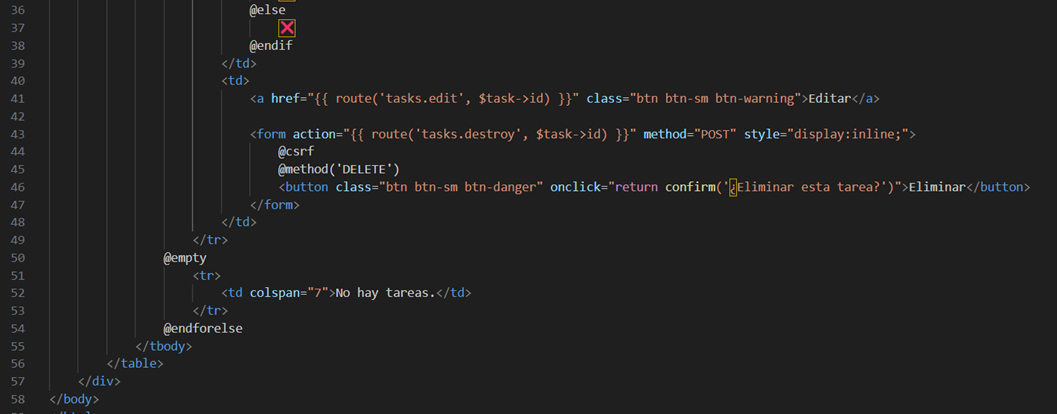
Recibe como parámetro task->id que lo hemos puesto arriba del formulario antes. Ese id lo paso como parámetro, ya dentro de la funcion, a un findOrFail que es un método de los que vienen con el modelo de tasks, esto me devolverá esa tarea, y la guardo en una variable task

Luego hemos pasado también como parámetro el request, que son los datos que se recogen del formulario, como antes, aquí hago un valídate de los datos del request, si son correctos los guardo en una variable validated.

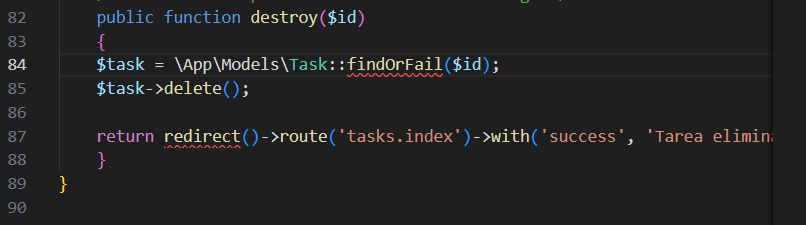
Por último llamo al método update, desde task, pasándole validated. Este método debe de existir en el modelo, y como es update, será que lo tiene que llamar la tarea que se va a editar, pasándole los parámetros que se van a poner, y esto ya me cambiara la task con los nuevos parámetros que se han recogido del formulario

Por ultimo redirigimos a tasks.index, que es hacer el método index, e ir a la vista que mostraba todas las tareas, donde debería de salir la nueva tarea ya editada.

Por ultimo nos falta el delete. En index.blade ya pusimos un formulario para eliminar esa tarea, cogiendo el task->id, y como botón enviar del formulario, el botón que pusimos en el lado



Este formulario nos lleva a tasks.destroy, de rutas, es decir al método destroy de taskController (pasando por un confirm). Por lo que nos vamos a TaskController a rellenar el método destroy



Este simplemente hace findOrFail con el id para encontrar la tarea en la bbdd y guardarla en una variable task, y cuando la tiene con task hace delete. Y volvemos el método de cargar las tareas e ir a index otra vez

Con esto ya esta el CRUD completo. Ahora vamos a mejorar la aplicación