**Introducción**

Siguiendo el vídeo no puede realizar la instalación

Lo consigo instalando Composer y después ejecutando el comando

composer create-project laravel/laravel publicaciones desde la carpeta htdocs.

Composer require laravel/breeze --dev

Php artisan breeze:install

Ahora instalamos Node.js para poder continuar con los comandos npm e instalar los elementos necesarios para el proyecto de laravel

Npm install

Npm run dev

Finaliza de forma distinta al vídeo y no crea los css en la carpeta public. Todo lo demás parece que está correcto.

Base de datos inicial

Creamos una vbase de datos vacía con el nombre de “publicaciones” con nuestro cliente de mysql que es PhpmyAdmin.

Establecemos en el fichero ,env las variables de entorno de la base de datos(nombre, usuario, password)

Php aristan migrate

Y se crean las tablas que se definen en los archivos de la carpeta migrations.

**Datos**

php artisan make:model Post

database\migrations\2024\_01\_04\_103911\_create\_posts\_table.php

Con esto estamos creando el esquema para la creación de la tabla “posts”

Añadimos las instrucciones para la creación de un campo user\_id en la tabla posts y establecemos una clave ajena en el campo user\_id que se relaciona con el campo “id” de la tabla “users”.

$table->unsignedBigInteger('user\_id');

            $table->foreign('user\_id')->references('id')->on('users');

            $table->text('body');

Ejecutamos de nuevo php artisan migrate y se crea la tabla en la base de datos.

Comprobamos que se ha creado la tabla y que no nos permite introducir datos si no hay un usuario debido a las restricciones de la clave ajena.

Php artisan migrate:refresh

Genera todas las tablas de la base de datos destruyendo las anteriores.

Añadimos un usuario mediante el formulario Register y comprobamos en la Base de Datos que se puede añadir registros en la tabla “posts” ahora que ya existe un usuario.

 $table->foreign('user\_id')->references('id')->on('users')->onDelete('cascade');

Php artisan migrate:refresh

Se crean de nuevo las tablas. Creamos un usuario mediante el formulario register y luego un post en la base de datos. Comprobamos que se puede borrar el usuario y se eliminan los posts asociados a éste.

Lo siguiente es establecer en Laravel la multiplicidad entre las clases Post y User ya que un usuario pude tener muchos posts pero cada post sólo puede pertenecer a un usuario.

Para ello modificamos la clase User de la siguiente manera:

Añadimos la función posts()

public function posts()

    {

        return $this->hasMany(Post::class);

    }

En la clase post

public function user()

    {

        return $this->belongsTo(User::class);

    }

Y añadimos en Post

protected $fillable = ['body'];

para permitir la asignación en masa en el campo body.