Ejercicios – VIDEOCLUB – C3

En estos ejercicios vamos a continuar con el proyecto del videoclub que habíamos empezado en sesiones anteriores y le añadiremos todo lo referente a la gestión de la base de datos.

Ejercicio 1 - Configuración de la base de datos y migraciones

En primer lugar vamos a configurar correctamente la base de datos. Para esto tenemos que actualizar los ficheros `config/database.php` y `.env` para indicar que vamos a usar una base de datos tipo MySQL llamada "`videoclub`" junto con el nombre de usuario y contraseña de acceso.

Nota: XAMPP por defecto crea el usuario de base de datos *root* sin contraseña.

A continuación abrimos *PHPMyAdmin* y creamos la nueva base de datos llamada `videoclub`. Para comprobar que todo se ha configurado correctamente vamos a un terminal en la carpeta de nuestro proyecto y ejecutamos el comando que crea la tabla de migraciones. Si todo va bien podremos actualizar desde *PHPMyAdmin* y comprobar que se ha creado esta tabla dentro de nuestra nueva base de datos.

Nota: Si nos diese algún error tendremos que revisar los valores indicados en el fichero `.env`. En caso de ser correctos es posible que también tengamos que reiniciar el servidor o terminal que tengamos abierto.

Ahora vamos a crear la tabla que utilizaremos para almacenar el catálogo de películas. Ejecuta el comando de Artisan para crear la migración llamada `create_movies_table` para la tabla `movies`. Una vez creado edita este fichero para añadir todos los campos necesarios, estos son:

Campo	Tipo	Valor por defecto
id	Autoincremental	
title	String	
year	String de longitud 8	
director	String de longitud 64	
poster	String	
rented	Booleano	false
synopsis	Text	
timestamps	Timestamps de Eloquent	

Nota: Recuerda que en el método `down` de la migración tienes que deshacer los cambios que has hecho en el método up, en este caso sería eliminar la tabla.

Por último ejecutaremos el comando de Artisan que añade las nuevas migraciones y comprobaremos en *PHPMyAdmin* que la tabla se ha creado correctamente con los campos que le hemos indicado.

Ejercicio 2 - Modelo de datos

En este ejercicio vamos a crear el modelo de datos asociado con la tabla *movies*. Para esto usaremos el comando apropiado de Artisan para crear el modelo llamado `Movie`.

Una vez creado este fichero lo abriremos y comprobaremos que el nombre de la clase sea el correcto y que herede de la clase `Model`. Y ya está, no es necesario hacer nada más, el cuerpo de la clase puede estar vacío (`{}`), todo lo demás se hace automáticamente!

Ejercicio 3 - Semillas

Ahora vamos a proceder a rellenar la tabla de la base de datos con los datos iniciales. Para esto editamos el fichero de semillas situado en `database/seeds/DatabaseSeeder.php` y seguiremos los siguientes pasos:

- Creamos un método privado (dentro de la misma clase) llamado `seedCatalog()` que se tendrá que llamar desde el método `run` de la forma:

```
public function run()
{
  self::seedCatalog();
  $this->command->info('Tabla catálogo inicializada con
datos!');
}
```

- Movemos el array de películas que se facilitaba en los materiales y que habíamos copiado dentro del controlador `CatalogController` a la clase de semillas (`DatabaseSeeder.php`), guardándolo de la misma forma, como variable privada de la clase.
- Dentro del nuevo método `seedCatalog()` realizamos las siguientes acciones:
  En primer lugar borramos el contenido de la tabla `movies` con `DB::table('movies')->delete();`.
  Y a continuación añadimos el siguiente código:

```
foreach( $this->arrayPeliculas as $pelicula ) {
    $p = new Movie;
    $p->title = $pelicula['title'];
```

```
            $p->year = $pelicula['year'];
            $p->director = $pelicula['director'];
            $p->poster = $pelicula['poster'];
            $p->rented = $pelicula['rented'];
            $p->synopsis = $pelicula['synopsis'];
            $p->save();
        }
```

Por último tendremos que ejecutar el comando de Artisan que procesa las semillas y una vez realizado abriremos *PHPMyAdmin* para comprobar que se rellenado la tabla *movies* con el listado de películas.

Nota: Si te aparece el error "*Fatal error: Class 'Movie' not found*" revisa si has indicado el espacio de nombres del modelo que vas a utilizar (`use App\Movie;`).

Ejercicio 4 - Uso de la base de datos

En este último ejercicio vamos a actualizar los métodos del controlador `CatalogController` para que obtengan los datos desde la base de datos. Seguiremos los siguientes pasos:

- Modificar el método `getIndex` para que obtenga toda la lista de películas desde la base de datos usando el modelo `Movie` y que se la pase a la vista.

- Modificar el método `getShow` para que obtenga la película pasada por parámetro usando el método `findOrFail` y se la pase a la vista.

- Modificar el método `getEdit` para que obtenga la película pasada por parámetro usando el método `findOrFail` y se la pase a la vista.

Nota: Si al probarlo te aparece el error "*Class 'App\Http\Controllers\Movie' not found*" revisa si has indicado el espacio de nombres del modelo que vas a utilizar (`use App\Movie;`).

Ya no necesitaremos más el array de películas (`$arrayPeliculas`) que habíamos puesto en el controlador, así que lo podemos comentar o eliminar.

Ahora tendremos que actualizar las vistas para que en lugar de acceder a los datos del array los obtenga del **objeto** con la película. Para esto cambiaremos en todos los sitios donde hayamos puesto `$pelicula['campo']` por `$pelicula->campo`.

Además, en la vista `catalog/index.blade.php`, en vez de utilizar el índice del array ($key) como identificador para crear el enlace a `catalog/show/{id}`, tendremos que utilizar el campo id de la película (`$pelicula->id`). Lo mismo en la vista `catalog/show.blade.php`, para generar el enlace de editar película tendremos que añadir el identificador de la película a la ruta `catalog/edit`.