Práctica UD-7 Sesión-2

|  |  |  |  |
| --- | --- | --- | --- |
| Objetivos | * Aprender a trabajar en Laravel con servicios web. | | |
|
| Instrucciones de entrega | * Subid el proyecto o ficheros con el código que resuelve los ejercicios a Edmodo (no es imprescindible haber resuelto todos para entregar). | | |

* Abre una línea de comandos desde **VS Code** y ve a la ruta donde está el directorio ***htdocs*** de Xampp (**c:\xampp\htdocs**) después crea un nuevo proyecto con nombre ***PHPPracticaUD7Sesion2***:

|  |
| --- |
| C:\xampp\htdocs>laravel new PHPPracticaUD7Sesion2 |

* O como alternativa usando *composer*:

|  |
| --- |
| C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel PHPPracticaUD7Sesion2 |

* Una vez creado carga el proyecto desde ***VS Code*** simplemente abriendo la carpeta o bien dentro del directorio con el comando ***code .*** .

## Laravel y servicios web

* Ejemplo de servicios web con modelos y relaciones basado en la página:
  + <https://manuais.iessanclemente.net/index.php/LARAVEL_Framework_-_Tutorial_01_-_Creaci%C3%B3n_de_API_RESTful>
* Vamos a crear una API REST de Fabricantes de Aviones:
  + - Un fabricante como Boeing fabricará muchos modelos de aviones
    - Un avión es fabricado por 1 fabricante.
  + Entidad Airplane (avión)
    - #Id (auto incremental)
    - Serial (clave única): cadena de caracteres única en la tabla
    - Model (modelo): cadena de caracteres
    - Length (longitud): decimal
    - Capacity (capacidad): entero
    - Speed (velocidad): decimal
    - Range (alcance): entero
  + Entidad Manufacturer (fabricante)
    - #Id (auto incremental)
    - Name (nombre): cadena de caracteres
    - Address (dirección): cadena de caracteres
    - Phone (teléfono): cadena de caracteres
* En primer lugar crearemos los modelos y sus clases asociadas para ello utilizamos los siguientes comandos:
  + *php artisan make:model Airplane -a --api*
  + *php artisan make:model Manufacturer -a --api*
* Lo siguiente será editar los ficheros de **migración** situados en ***database/migrations/*** , comenzamos con el del fabricante (Manufacturer) ***xxx\_create\_manufacturers\_table***:

Schema::create('manufacturers', function (Blueprint $table) {

$table->id();

$table->string('name');

$table->string('address');

$table->string('phone');

$table->timestamps();

});

* Seguimos con el del avión (Airplane) ***xxx\_create\_airplanes\_table***:

public function up()

{

Schema::create('airplanes', function (Blueprint $table) {

$table->id();

$table->string('serial')->unique();

$table->string('model');

$table->float('length');

$table->integer('capacity');

$table->float('speed');

$table->integer('range');

// Añadimos la clave foránea con Manufacturer: manufacturer\_id

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

// Indicamos cual es la clave foránea de esta tabla y que se borra en cascada:

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

$table->timestamps();

});

}

* Ahora editaremos los modelos para indicar qué atributos son rellenables y definir las relaciones entre los modelos, para el modelo fabricante (Manufacturer):

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'name',

'address',

'phone',

];

public function airplanes() {

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

}

* Para el modelo del avión (Airplane):

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'serial',

'model',

'length',

'capacity',

'speed',

'range',

'manufacturer\_id',

];

public function manufacturer()

{

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

}

* Continuamos editando los ficheros de la **factoría,** en primer lugar para el fabricante situado en ***database/factories/ManufacturerFactory.php***:

public function definition()

{

return [

'name' => $this->faker->company,

'address' => $this->faker->address,

'phone' => $this->faker->phoneNumber

];

}

* En segundo lugar para el avión situado en ***database/factories/AirplaneFactory.php***:

public function definition()

{

return [

'serial' => $this->faker->ean8,

'model' => $this->faker->word,

'length' => $this->faker->randomFloat(2, 10, 500),

'capacity' => $this->faker->numberBetween(1, 1000),

'speed' => $this->faker->randomFloat(2, 100, 3000),

'range' => $this->faker->numberBetween(100, 100000)

];

}

* Seguidamente editamos los ficheros de la **siembra**, primero para el modelo del avión (Airplane) situado en ***database/seeders/AirplaneSeeder*** :

public function run()

{

// Truncamos la tabla para comenzar con una vacía.

Airplane::truncate();

// No hacemos nada más ya que se crearán cuando se creen los fabricantes

}

* Después para el modelo del fabricante (Manufacturer) situado en el fichero ***database/seeders/ManufacturerSeeder*** :

public function run()

{

// Truncamos la tabla para comenzar con una vacía, necesitamos desabilitar la constrain de foreing key para poder hacerlo sin que se queje la base de datos.

Schema::disableForeignKeyConstraints();

Manufacturer::truncate();

Schema::enableForeignKeyConstraints();

// Creamos 2 fabricantes con 10 aviones asociados.

Manufacturer::factory()

->has(Airplane::factory()->count(10))

->count(2)->create();

// Creamos 5 fabricantes sin aviones asociados.

Manufacturer::factory()->count(5)->create();

}

* Por último editamos el fichero de la **siembra** general ***DatabaseSeeder.php*** para no tener que hacer la siembra de forma individual:

public function run()

{

$this->call([

AirplaneSeeder::class,

ManufacturerSeeder::class

]);

}

* En este punto creamos la tabla en base de datos (***phppracticaud7sesion2***) y ejecutamos la migración para que se creen las tablas y se rellenen con los datos de prueba:
  + *php artisan migrate --seed*
* Una vez comprobado que tenemos la base de datos correctamente pasamos a la siguiente parte que es la de crear los ficheros de recursos, comenzamos con el de aviones:
  + *php artisan make:resource AirplaneResource*
* En el caso del fichero de recurso para los fabricantes (Manufacturer), queremos que además de su información obtengamos una lista con los aviones asociados:
  + *php artisan make:resource ManufacturerResource*
* Ahora es el momento de crear las clases de tipo *form request* para ello empezamos por la del modelo del avión (Airplane):
  + *php artisan make:request AirplaneRequest*
* A continuación con la del modelo del fabricante (Manufacturer):
  + *php artisan make:request ManufacturerRequest*
* Para finalizar añadimos las rutas en ***routes/api.php***:

Route::apiResource('manufacturers', ManufacturerController::class);

Route::apiResource('airplanes', AirplaneController::class);

* Terminamos con las clases de los controles para los dos modelos, para los fabricantes (Manufacturer) en la clase ***ManufacturerController***:
* Finalmente para los aviones (Airplane) en la clase ***AirplaneController***:
* Probamos que todo funciona correctamente creando una nueva colección en Posman:

