

CONFIGURACIÓN BACKEND LARAVEL 9:

1. Creación del proyecto laravel:

`composer create-project laravel/laravel api`

2. Creación de la base de datos para los personajes en mysql:

`CREATE DATABASE starwars_db CHARACTER SET utf8 COLLATE utf8_spanish_ci;`

3. Para manejar los datos de los personajes creamos el modelo Personaje:

`php artisan make:model Personaje -m`

4. Creamos controlador necesario:

`php artisan make:controller Api/PersonajeController --api`

5. Configuramos fichero de migración correspondiente a nuestra base de datos, añadiendo los campos necesarios:

```
public function up(): void
{
    Schema::create('personajes', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('height');
        $table->string('mass');
        $table->string('hair_color');
        $table->text('skin_color');
        $table->string('eye_color');
        $table->string('birth_year');
        $table->text('gender');
        $table->string('homeworld');
        $table->timestamps();
    });
}
```

6. Para habilitar la asignación masiva en nuestra tabla, modificamos el modelo de personajes, añadiendo la siguiente propiedad:

```
protected $fillable = ['name', 'height', 'mass', 'hair_color', 'skin_color', 'eye_color',
    'birth_year', 'gender', 'homeworld'];
```

7. Modificamos el código de nuestro fichero controlador PersonajeController:

```
class PersonajeController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $personajes = Personaje::all();
        return $personajes;
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $personaje = new Personaje();
        $personaje->name = $request->name;
        $personaje->height = $request->height;
        $personaje->mass = $request->mass;
        $personaje->hair_color = $request->hair_color;
        $personaje->skin_color = $request->skin_color;
        $personaje->eye_color = $request->eye_color;
        $personaje->birth_year = $request->birth_year;
        $personaje->gender = $request->gender;
        $personaje->homeworld = $request->homeworld;
        $personaje->save();
    }
}
```

```
public function show(string $id)
{
    $personaje = Personaje::find($id);
    return $personaje;
}

/**
 * Update the specified resource in storage.
 */
public function update(Request $request, string $id)
{
    $personaje = Personaje::findOrFail($request->id);
    $personaje->name = $request->name;
    $personaje->height = $request->height;
    $personaje->mass = $request->mass;
    $personaje->hair_color = $request->hair_color;
    $personaje->skin_color = $request->skin_color;
    $personaje->eye_color = $request->eye_color;
    $personaje->birth_year = $request->birth_year;
    $personaje->gender = $request->gender;
    $personaje->homeworld = $request->homeworld;

    $personaje->save();
    return $personaje;
}
```

```
public function destroy(string $id)
{
    $personaje = Personaje::destroy($id);
    return $personaje;
}
```

8. Agregación de las rutas necesarias en el fichero:

```
use App\Http\Controllers\Api\PersonajeController;

/*
|-----
| API Routes
|-----
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {
    return $request->user();
});

Route::controller(PersonajeController::class)->group(function () {
    Route::get('/personajes', 'index');
    Route::post('/personaje', 'store');
    Route::get('/personaje/{id}', 'show');
    Route::put('/personaje/{id}', 'update');
    Route::delete('/personaje/{id}', 'destroy');
});
```

- Para testear nuestra API, utilizaremos un simulador de llamadas a API, en nuestro caso hemos utilizado la extensión de Thunder en VS Code:

GET: Obtenemos dato específico de la base de datos

The screenshot shows the Thunder client interface. At the top, a GET request is configured to `http://127.0.0.1:8000/api/personajes` with a 'Send' button. Below the request bar, the 'Query' tab is active, showing 'Query Parameters' with a table for adding parameters. The 'Response' tab on the right displays the JSON response from the server, which is a single object for Luke Skywalker.

parameter	value
-----------	-------

```
[
  {
    "id": 1,
    "name": "Luke Skywalker",
    "height": "172",
    "mass": "77",
    "hair_color": "blond",
    "skin_color": "fair",
    "eye_color": "blue",
    "birth_year": "19bbby",
    "gender": "male",
    "homeworld": "Tatooine",
    "created_at": null,
    "updated_at": null
  }
]
```

GET

http://127.0.0.1:8000/api/personaje/1

Send

Status: 200 OK

Size: 215 Bytes

Time: 341 ms

Query

Headers²

Auth

Body

Tests

Pre Run

Response

Headers⁹

Cookies

Results

Query Parameters

☐

parameter

value

1 {

2 "id": 1,

3 "name": "Luke Skywalker",

4 "height": "172",

5 "mass": "77",

6 "hair_color": "blond",

7 "skin_color": "fair",

8 "eye_color": "blue",

9 "birth_year": "19bby",

10 "gender": "male",

11 "homeworld": "Tatooine",

12 "created_at": null,

13 "updated_at": null

14 }

POST: Se solicita la subida de recursos a la base de datos

POST

http://127.0.0.1:8000/api/personaje?name=Leia&height=166

Send

Status: 200 OK

Size: 0 Bytes

Time: 325 ms

Query

Headers²

Auth

Body

Tests

Pre Run

Response

Headers⁹

Cookies

Results

Query Parameters

☒

name

Leia

▼

☒

height

166

▼

☒

mass

60

▼

☒

hair_color

brown

▼

☒

skin_color

fair

▼

☒

eye_color

brown

▼

☒

birth_year

1980

▼

☒

gender

female

▼

☒

homeworld

Alderaan

▼

PUT: Actualiza recurso específico de nuestra base de datos

PUT

http://127.0.0.1:8000/api/personaje/2?name=Padme&height=1

Send

Status: 200 OK

Size: 255 Bytes

Time: 377 ms

Query

Headers²

Auth

Body

Tests

Pre Run

Response

Headers⁹

Cookies

Results

Docs

{

Query Parameters

☒

name

Padme

▼

☒

height

155

▼

☒

mass

55

▼

☒

hair_color

brown

▼

☒

skin_color

fair

▼

☒

eye_color

brown

▼

☒

birth_year

1920

▼

☒

gender

female

▼

☒

homeworld

Naboo

▼

☐

parameter

value

1 {

2 "id": 2,

3 "name": "Padme",

4 "height": "155",

5 "mass": "55",

6 "hair_color": "brown",

7 "skin_color": "fair",

8 "eye_color": "brown",

9 "birth_year": "1920",

10 "gender": "female",

11 "homeworld": "Naboo",

12 "created_at": "2023-09-18T11:34:05.000000Z",

13 "updated_at": "2023-09-18T11:44:22.000000Z"

14 }

DELETE: Borra recurso específico de nuestra base de datos

DELETE	▼	http://127.0.0.1:8000/api/personaje/1	Send	Status: 200 OK	Size: 1 Bytes	Time: 253 ms
Query	Headers ²	Auth	Body	Tests	Pre Run	Response
						Headers ⁹
						Cookies
						Results
						Docs
						1 1