Guió bibliotecaAPI

Crear base de dades

```
CREATE USER 'bibliotecaAPI' identified BY 'bibliotecaAPI.1234';
CREATE DATABASE bibliotecaAPI;
GRANT ALL PRIVILEGES ON bibliotecaAPI.* TO 'bibliotecaAPI';
```

Crear nou projecte

```
> laravel new bibliotecaAPI
```

Configurar .env

```
DB_DATABASE=bibliotecaAPI
DB_USERNAME=bibliotecaAPI
DB_PASSWORD=bibliotecaAPI.1234
```

Crear model migració i controlador

```
> php artisan make:model Libro -mc
```

```
> php artisan make:model Autor -mc
```

Configuració de les migracions

```
public function up(): void
    {
        Schema::create('libros', function (Blueprint $table) {
            $table->id();
            $table->String('titulo');
            $table->String('editorial');
            $table->float('precio');
            $table->unsignedBigInteger('autor_id')->nullable();
            $table->timestamps();
        });
    }
```

```
public function up(): void
    {
        Schema::create('autores', function (Blueprint $table) {
            $table->id();
            $table->String('nombre');
            $table->integer('nacimiento')->nullable();
```

```
            $table->timestamps();
        });
    }
```

Configuració dels models**

```
class Autor extends Model
{
    protected $table = 'autores';

    public function libros()
    {
        return $this->hasMany(Libro::class);
    }
}
```

```
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Libro extends Model
{
    public function autor()
    {
        return $this->belongsTo(Autor::class);
    }
}
```

Afegim algunes dades des del Seeder

```
> php artisan make:seeder LibrosSeeder
> php artisan make:seeder AutresSeeder
```

LibrosSeeder

```
public function run(): void
    {
        $autor = new Autor();
        $autor->nombre = "Brandon Sanderson";
        $autor->nacimiento = 1975;
        $autor->save();
        $libro = new Libro();
        $libro->titulo = "Trenza del mar Esmeralda";
        $libro->editorial = "Editorial Nova";
        $libro->precio = 25.55;
```

```
        $libro->autor()->associate($autor);
        $libro->save();
        $libro2 = new Libro();
        $libro2->titulo = "Elantris";
        $libro2->editorial = "Editorial Nova";
        $libro2->precio = 26.50;
        $libro2->autor()->associate($autor);
        $libro2->save();
    }
```

AutoresSeeder

```
    public function run(): void
    {
        $autor = new Autor();
        $autor->nombre = "Andy Weir";
        $autor->nacimiento = 1972;
        $autor->save();
    }
```

Afegim els seeder al DatabaseSeeder

```
class DatabaseSeeder extends Seeder
{
    public function run()
    {
        ...
        $this->call(AutoresSeeder::class);
        $this->call(LibrosSeeder::class);
    }
}
```

`> php artisab db:seed`

Creem el controlador API

`> php artisan make:controller Api/LibroController --api --model=Libro`

Instal·lem API

`> php artisan install:api`

Afegim resource en routes/api.php

En **routes/api.php**

`Route::apiResource('libros', LibroController::class);`

Comprovem les rutes

`> php artisan route:list`

Configuració del controlador

```
public function index()
{
    $libros = Libro::get();
        //$libros = Libro::with('autor')->get();
        return $libros;
}
```

```
public function show(Libro $libro)
{
    return $libro;
        //return $libro->load('autor');
}
```

Protegir camps

En el model:
`protected $hidden = ['precio'];`

o en el propi mètode:

```
public function index()
    {
        $libros = Libro::with('autor')->get();
        return $libros->map(function ($libro) {
                return [
                'titulo'        => $libro->titulo,
                'editorial' => $libro->editorial,
                'autor'         => $libro->autor->nombre
                ];
        });
    }

public function show(Libro $libro)
{
    return [
        'titulo'        => $libro->titulo,
        'editorial' => $libro->editorial
    ];
}
```

Afegir codis de resposta

```
public function index()
{
    $libros = Libro::get();
    return response()->json($libros, 200);
}
...
public function show(Libro $libro)
{
    return response()->json($libro, 200);
}
```

Afegir: POST

```
public function store(Request $request)
    {
        $libro = new Libro();
        $libro->titulo = $request->titulo;
        $libro->editorial = $request->editorial;
        $libro->precio = $request->precio;
        $libro->autor()->associate(Autor::findOrFail($request->autor_id));
        $libro->save();

        return response()->json($libro, 201);
    }
```

Actualitzar: PUT

```
public function update(Request $request, Libro $libro)
{
    $libro->titulo = $request->titulo;
    $libro->editorial = $request->editorial;
    $libro->precio = $request->precio;
    $libro->autor()->associate(Autor::findOrFail($request->autor_id));
    $libro->save();

    return response()->json($libro, 200);
}
```

Eliminar: DELETE

```
 public function destroy(Libro $libro)
    {
        $libro->delete();
```

```
        return response()->json(null, 204);
    }
```

Validació de dades amb Request

`> php artisan make:request LibroRequest`

```
public function rules(): array
    {
        return [
            'titulo' => 'required|min:3',
            'editorial' => 'required',
            'precio' => 'required|numeric|min:0'
        ];
    }

    public function messages(): array
    {
        return [
            'titulo.required' => 'El título es obligatorio',
            'titulo.min' => 'Título demasiado corto',
            'editorial.required' => 'La editorial es obligatoria',
            'precio.required' => 'El precio es obligatorio',
            'precio.numeric' => 'El precio debe ser numérico',
            'precio.min' => 'El precio debe ser mayor de 0'
            ];
    }
```

En el controlador modifiquem **Request** per **LibroRequest**

Personalitzar respostes d'error

En **bootstrap/app**

```
use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;
use Symfony\Component\HttpKernel\Exception\NotFoundHttpException;
use Illuminate\Validation\ValidationException;
use Illuminate\Http\Request;
use Illuminate\Database\Eloquent\ModelNotFoundException;
    ->withExceptions(function (Exceptions $exceptions) {
        $exceptions->render(function (ModelNotFoundException $e, Request
$request) {
```

```
                if ($request->is('api*')) {
                return response()->json(['error' => 'Recurso no
encontrado'], 404);
                }
        });
        $exceptions->render(function (NotFoundHttpException $e, Request
$request) {
                if ($request->is('api*')) {
                return response()->json([
                'error' => 'Recurso no encontrado',
                'descripcion' => $e->getMessage() // Incluye la descripción
del error
            ], 404);
                }
        });
        $exceptions->render(function (ValidationException $e, Request
$request) {
                if ($request->is('api*')) {
                return response()->json(['error' => 'Datos no válidos'],
400);
                }
        });
        $exceptions->render(function (\Throwable $e, Request $request) {
                if ($request->is('api*')) {
                return response()->json(['error' => 'Error: ' . $e-
>getMessage()], 500);
                }
        });

    })->create();
```

Proves amb ThunderClient

Autenticació

Modifiquem la migració d'usuaris

```php
public function up(): void
    {
        Schema::create('usuarios', function (Blueprint $table) {
            $table->id();
            $table->string('login');
            $table->string('password');
```

```
            $table->timestamps();
        });

    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('usuarios');
    }
```

Modifiquem factory, seeder i model

En el factory **UsuarioFactory**

```
return [
                'login' => $this->faker->word,
                'password' => bcrypt('1234')
        ];
```

En el **DatabaseSeeder**

`Usuario::factory(2)->create();`

Fem la migració:
`php artisan migrate:fresh --seed`

En el model **Usuario**
`class Usuario extends Authenticatable`

Mode Sanctum

Instal·lem l'api si no ho hem fet encara:

`php artisan install:api`

En el arxiu **config/sanctum.php** establim el temps d'expiració:
`'expiration' => 5,`

En el model **Usuario** afegim **HasApiTokens**:

```
...
use Illuminate\Routing\Controller;
class Usuario extends Authenticatable
```

```
{
    use HasApiTokens, HasFactory, Notifiable;
```

Generem el controlador de login:
`php artisan make:controller Api/LoginController`

Afegim la funció **login**:

```
public function login(Request $request)
    {
        $usuario = Usuario::where('login', $request->login)->first();

        if (!$usuario ||
                !Hash::check($request->password, $usuario->password))
        {
                return response()->json(
                ['error' => 'Credenciales no válidas'], 401);
        }
        else
        {
                return response()->json(['token' =>
                $usuario->createToken($usuario->login)->plainTextToken]);
        }
    }
```

Establim la ruta en **routes/api.php**

```
Route::post('login', [LoginController::class, 'login']);
```

Modifiquem el constructor de **LibroController**:

```
public function __construct()
    {
        $this->middleware('auth:sanctum',
                ['except' => ['index', 'show']]);
    }
```

Comprovar que hem afegit

```
use Illuminate\Routing\Controller;
```