

“Gestión básica de una biblioteca”

Ejercicio 1 Laravel

Objetivo: Utilización de Modelos, Migraciones y Seeders

Aprender a generar modelos Eloquent, migraciones, factories y seeders en Laravel, definiendo correctamente las relaciones entre Author, Book, Category, Loan y User, y completar la base de datos con datos de ejemplo.

Requisitos previos

Proyecto Laravel funcional.

Composer y Node instalados(<https://nodejs.org/es/download>).

.env configurado (usar database/database.sqlite para simplificar).

Apartado 1 - Crear base de datos

```
SQL
CREATE DATABASE dwes_04_biblioteca_laravel CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;
```

```
composer create-project laravel/laravel biblioteca
composer require laravel/jetstream
php artisan jetstream:install livewire
npm install
npm run build
```

Archivo: .env

```
Dotenv
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=dwes_04_biblioteca_laravel
DB_USERNAME=root
DB_PASSWORD=
```

Ejecuta:

```
php artisan migrate
```

Comprobar que existen tablas en BBDD

Recuerda:

En general el nombre de los modelos se pone en singular con la primera letra en mayúscula, mientras que el nombre de las tablas suele estar en plural.

Gracias a esto, al definir un modelo no es necesario indicar el nombre de la tabla asociada, sino que Eloquent automáticamente buscará la tabla transformando el nombre del modelo a minúsculas y buscando su plural (en inglés).

Descripción general del modelo

Author (autor) – 1 autor tiene muchos libros.

Book (libro) – pertenece a 1 autor y puede tener muchas categorías (many-to-many).

Category (categoría) – muchas categorías pueden aplicarse a muchos libros (pivot book_category).

Loan (préstamo) – representa la reserva/ préstamo de un libro: pertenece a un Book y a un User.

User – usuario del sistema (puede tener role/rol), puede tener muchos loans.

Esquema esperado (migraciones)

authors: id, name (string), bio (text, nullable), created_at, updated_at

books: id, title (string), isbn (string, nullable, unique), author_id (fk), published_at (date, nullable), pages (unsignedInteger, nullable), price (decimal 8,2, nullable), created_at, updated_at

categories: id, name (string), slug (string, unique), created_at, updated_at

book_category: book_id, category_id (ambas fks), primary key compuesta (book_id, category_id)

loans: id, book_id (fk), user_id (fk), loaned_at (datetime), returned_at (datetime, nullable), status (enum/string), created_at, updated_at

users: id, name, email (unique), password, role (string e.g. 'admin'/'user'), timestamps

Apartado 2 - Genera las siguientes migraciones

```
php artisan make:migration create_authors_table
php artisan make:migration create_categories_table
php artisan make:migration create_books_table
php artisan make:migration create_book_category_table
php artisan make:migration create_loans_table
```

Ejemplo de creación de la tabla Autores

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('authors', function (Blueprint $table) {
            $table->id();
            //Fecha de creación y actualización del registro
            $table->timestamps();
            $table->string('name');
            $table->string('country')->nullable();
            $table->date('birth_date')->nullable();
        });
    }

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

Ejemplo de libros

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('books', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            $table->string('title');
            $table->string('isbn')->unique();
            $table->unsignedSmallInteger('published_year')->nullable();

            //Clave foránea al autor
            $table->foreignId('author_id')
                ->constrained('authors')
                ->cascadeOnUpdate()
                ->restrictOnDelete();

            //Índice compuesto para búsquedas frecuentes
            $table->index(['author_id', 'title']);
        });
    }

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

Ejemplo de Categorías

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('categories', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
            //name Ciencia Ficción
            $table->string('name')->unique();
            //slug ciencia-ficcion
            //Para generar URLs amigables
            $table->string('slug')->unique();
        });
    }

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

Ejemplo de la tabla que relaciona los libros y sus categorías

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('book_category', function (Blueprint $table) {
            //Tabla Pivote Relación Muchos a Muchos
            //Un libro puede tener muchas categorías
            //Una categoría puede tener muchos libros
            $table->id();
            $table->timestamps();
            // Definición de claves foráneas
            $table->foreignId('book_id')
                ->constrained('books')
                ->cascadeOnDelete();
            $table->foreignId('category_id')
                ->constrained('categories')
                ->cascadeOnDelete();
            // Índice único para evitar duplicados
            $table->unique(['book_id', 'category_id']);
        });
    }

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

Ejemplo de Préstamos de libros

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('loans', function (Blueprint $table) {
            $table->id();
            $table->timestamps();

            // Definición de claves foráneas
            //Usuario que realiza el préstamo: users_table
            $table->foreignId('user_id')
                ->constrained('users')
                ->cascadeOnDelete();
            //Libro prestado
            $table->foreignId('book_id')
                ->constrained('books')
                ->restrictOnDelete();
            //Fechas del préstamo
            $table->date('loaned_at');
            //Fecha de vencimiento
            $table->date('due_at');
            //Fecha de devolución (puede ser nula si no se ha devuelto aún)
            $table->date('returned_at')->nullable();

            $table->string('status')->default('open'); // open | returned
            // Índice ÚNICO real: solo 1 préstamo abierto por libro
            $table->unique(['book_id', 'returned_at'], 'loans_book_open_unique');
            // Índice para optimizar consultas frecuentes
            $table->index(['user_id', 'status']);
        });
    }

    public function down(): void
    {
    }
}
```

Apartado 3 - Genera los seeders o semillas

```
php artisan make:seeder AuthorsTableSeeder CategoriesTableSeeder  
BooksTableSeeder
```

Para ejecutar un solo seeder, por ejemplo el de Autores

```
php artisan db:seed --class=AuthorsTableSeeder
```

Para ejecutar todos los seeders (Presta atención a las FK)

```
php artisan db:seed
```

Para ejecutar todas las migraciones y luego los seeders

```
php artisan migrate --seed
```

Para borrar los datos almacenados y ejecutar de nuevo

```
php artisan migrate:fresh --seed
```

Crea los Seeders para Autores y Categorías , sítete del ejemplo de libros que se muestra a continuación

Ejemplo de Seeder de Libros

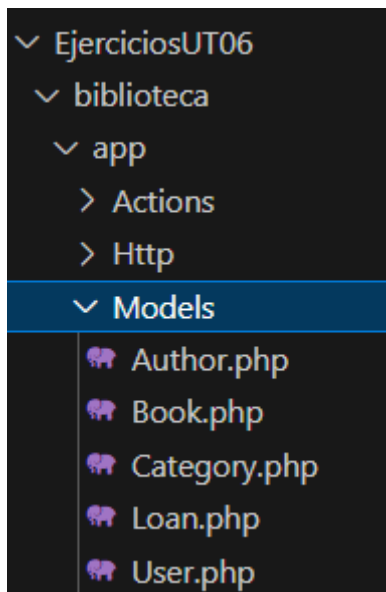
```
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class BooksTableSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        DB::table('books')->insert([
            [
                'title' => 'To Kill a Mockingbird',
                'published_year' => 1960,
                'created_at' => now(),
                'updated_at' => now(),
                'author_id' => 1,
                'isbn' => '978-0-06-112008-4',
            ],
            [
                'title' => '1984',
                'published_year' => 1949,
                'created_at' => now(),
                'updated_at' => now(),
                'author_id' => 2,
                'isbn' => '978-0-452-28423-4',
            ],
        ]);
    }
}
```

Apartado 4 - Los modelos



Ejemplo de modelo para Autores

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;
use App\Models\Book;

class Author extends Model
{
    // Campos asignables
    protected $fillable = ['name', 'country', 'birth_date'];

    // Cast para convertir birth_date a instancia de Carbon
    protected $casts = ['birth_date' => 'date'];

    // Relación uno a muchos con Book
    public function books(): HasMany
    {
        return $this->hasMany(Book::class);
    }
}
```

Ejemplo de Modelo para Libros

```
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Book extends Model
{
    // Campos asignables masivamente
    protected $fillable = ['title', 'isbn', 'published_year', 'author_id'];

    // Relaciones Eloquent

    // Un libro pertenece a un autor
    //Relación uno a muchos inversa
    public function author(): BelongsTo
    {
        return $this->belongsTo(Author::class);
    }

    // Un libro puede pertenecer a muchas categorías
    //Relación muchos a muchos
    public function categories(): BelongsToMany
    {
        return $this->belongsToMany(Category::class)->withTimestamps();
    }

    // Un libro puede tener muchos préstamos
    //Relación uno a muchos
    public function loans(): HasMany
    {
        return $this->hasMany(Loan::class);
    }

    // Método para verificar si el libro está disponible (no tiene préstamo)
    // Un libro está disponible si no tiene préstamos sin devolver
    public function isAvailable(): bool
    {
        return !$this->loans()->whereNull('returned_at')->exists();
    }
}
```

Crea los siguientes modelos

Category (Categorías) y Loan (Préstamos)

Resumen

Tareas (paso a paso)

Crear modelos + migraciones + seeder por comando.

Ejemplo comandos:

```
php artisan make:model Author -m -f
```

```
php artisan make:model Book -m -f
```

```
php artisan make:model Category -m -f
```

```
php artisan make:model Loan -m -f
```

(User ya existe) `php artisan make:seeder DatabaseSeeder` (o individual seeders)

Implementar campos en las migraciones según el esquema anterior (añadir fks y constraints).

Crear seeders:

AuthorsTableSeeder: crear N autores.

CategoriesTableSeeder: crear categorías comunes.

BooksTableSeeder: crear libros, asignar author_id y sincronizar categorías.

LoansTableSeeder: crear algunos préstamos vinculando users y books.

DatabaseSeeder: llamar a los seeders y crear un usuario admin (email: educantabria@example.com, password: password) usando Hash::make.

Ejecutar migraciones y seeders:

```
php artisan migrate --seed
```

Comandos útiles resumen

```
php artisan make:model Book -m -f
```

```
php artisan make:model Category -m -f
```

```
php artisan make:model Loan -m -f
```

```
php artisan make:seeder AuthorsTableSeeder CategoriesTableSeeder  
BooksTableSeeder LoansTableSeeder
```

```
php artisan migrate
```

```
php artisan db:seed
```

```
php artisan migrate:fresh --seed (para reset completo)
```

```
php artisan tinker (comprobar relaciones)
```