Práctica Guiada: Sistema de Cursos con Laravel 12

1. Instalación de Laravel 12

Primero instalamos un nuevo proyecto:

```
composer create-project laravel/laravel cursos-laravel12
cd cursos-laravel12
```

Levanta el servidor de desarrollo:

```
php artisan serve
```

Verifica en tu navegador http://127.0.0.1:8000

2. Configuración de la base de datos MySQL

Edita el archivo .env y ajusta:

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

Crea la BD en MySQL:

```
CREATE DATABASE cursosdb;
```

3. Instalar Tailwind CSS

Dentro del proyecto:

```
npm install -D tailwindcss postcss autoprefixer
npx tailwindcss init -p
```

En tailwind.config.js agrega las rutas Blade:

```
content: [
  "./resources/**/*.blade.php",
  "./resources/**/*.js",
  "./resources/**/*.vue",
],
```

En resources/css/app.css agrega:

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Compila:

```
npm run dev
```

---

4. Crear Modelos y Migraciones

Curso

php artisan make:model Curso -mcrfs

En la migración create_cursos_table.php:

```
public function up(): void
{
   Schema::create('cursos', function (Blueprint $table) {
      $table->id();
      $table->string('nombre',50);
      $table->text('descripcion');
      $table->date('fecha_inicio');
      $table->date('fecha_fin');
      $table->decimal('precio', 10, 2);
      $table->timestamps();
   });
}
```

Persona

```
php artisan make:model Persona -mcrfs
public function up(): void
{
   Schema::create('personas', function (Blueprint $table) {
      $table->id();
      $table->string('numero_documento')->unique();
      $table->string('nombres');
      $table->string('apellidos');
      $table->enum('sexo',['M','F']);
      $table->date('fecha_nacimiento');
      $table->string('celular')->nullable();
      $table->timestamps();
   });
}
```

Inscripcion

```
php artisan make:model Inscripcion -mcrfs
public function up(): void
{
   Schema::create('inscripciones', function (Blueprint $table) {
      $table->id();
      $table->foreignId('curso_id')->constrained('cursos');
      $table->foreignId('persona_id')->constrained('personas');
      $table->date('fecha');
      $table->decimal('monto', 10, 2);
      $table->timestamps();
```

```
    });
}
```

Ejecuta las migraciones:

php artisan migrate

5. Definir Relaciones en los Modelos

```
app/Models/Curso.php
class Curso extends Model
{
    use HasFactory;

    protected $fillable = ['nombre','descripcion','fecha_inicio','fecha_fin','precio'];

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

```
app/Models/Persona.php
class Persona extends Model
{
    use HasFactory;

    protected $fillable =
['numero_documento','nombres','apellidos','sexo','fecha_nacimiento','celular'];

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

```
app/Models/Inscripcion.php
class Inscripcion extends Model
{
    use HasFactory;

    protected $fillable = ['curso_id','persona_id','fecha','monto'];
    protected $table='inscripciones',

    public function curso()
    {
        return $this->belongsTo(Curso::class);
    }
```

```
  public function persona()
  {
    return $this->belongsTo(Persona::class);
  }
}
```

6. Factories y Seeders

Factory Curso

```
public function definition(): array
{
  return [
    'nombre' => $this->faker->sentence(3),
    'descripcion' => $this->faker->paragraph(),
    'fecha_inicio' => $this->faker->date(),
    'fecha_fin' => $this->faker->date(),
    'precio' => $this->faker->randomFloat(2, 100, 1000),
  ];
}
```

Factory Persona

```
public function definition(): array
{
  return [
    'numero_documento' => $this->faker->unique()->numerify('########'),
    'nombres' => $this->faker->firstName(),
    'apellidos' => $this->faker->lastName(),
    'sexo' => $this->faker->randomElement(['M','F']),
    'fecha_nacimiento' => $this->faker->date(),
    'celular' => $this->faker->phoneNumber(),
  ];
}
```

Factory Inscripcion

```
public function definition(): array
  {
    return [
      'curso_id' => Curso::inRandomOrder()->first()->id,
      'persona_id' => Persona::inRandomOrder()->first()->id,
      'fecha' => $this->faker->dateTimeBetween('-1 years', 'now'),
      'monto' => $this->faker->randomFloat(2, 50, 1000),
    ];
  }
```

Seeder Curso

```
public function run(): void
  {
    Curso::factory()->count(10)->create();
  }
```

Seeder Persona

```
public function run(): void
  {
    Persona::factory()->count(50)->create();
  }
```

Seeder Inscripcion

```
public function run(): void
  {
    Inscripcion::factory()->count(150)->create();
  }
```

DatabaseSeeder.php:

```
$this::call([CursoSeeder::class,
ProfesionSeeder::class,InscripcionSeeder::class]);
```

Ejecuta:

```
php artisan migrate –seed
```

7. Controladores y Recursos

Ejemplo CursoController.php:

```
class CursoController extends Controller
{
  public function index()
  {
    $cursos = Curso::all();
    return view('cursos.index', compact('cursos'));
  }

  public function create()
  {
    return view('cursos.create');
  }
```

```
  public function store(Request $request)
  {
    $request->validate([
      'nombre'=>'required',
      'descripcion'=>'required',
      'fecha_inicio'=>'required|date',
      'fecha_fin'=>'required|date',
      'precio'=>'required|numeric'
    ]);

    Curso::create($request->all());

    return redirect()->route('cursos.index');
  }
}
```

(Los demás controladores siguen la misma lógica CRUD)

8. Rutas

En routes/web.php:

```
Route::resource('cursos', CursoController::class);
Route::resource('personas', PersonaController::class);
Route::resource('inscripciones', InscripcionController::class);
```

9. Vistas con Blade + Tailwind

Crea un layout base en resources/views/layouts/app.blade.php:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>@yield('title','Cursos')</title>
  @vite('resources/css/app.css')
</head>
<body class="bg-gray-100 text-gray-800">
  <nav class="bg-blue-600 p-4 text-white">
    <a href="{{ route('cursos.index') }}" class="mr-4">Cursos</a>
    <a href="{{ route('personas.index') }}" class="mr-4">Personas</a>
    <a href="{{ route('inscripciones.index') }}">Inscripciones</a>
  </nav>
  <div class="container mx-auto p-6">
    @yield('content')
  </div>
</body>
```

```
</html>
```

Ejemplo de vista resources/views/cursos/index.blade.php:

```
@extends('layouts.app')

@section('title','Listado de Cursos')

@section('content')
<h1 class="text-2xl font-bold mb-4">Cursos</h1>
<a href="{{ route('cursos.create') }}" class="bg-blue-500 text-white px-4 py-2
rounded">Nuevo Curso</a>

<table class="table-auto w-full mt-4 border">
  <thead>
    <tr class="bg-gray-200">
      <th class="px-4 py-2">Nombre</th>
      <th class="px-4 py-2">Descripción</th>
      <th class="px-4 py-2">Precio</th>
    </tr>
  </thead>
  <tbody>
    @foreach($cursos as $curso)
    <tr>
      <td class="border px-4 py-2">{{ $curso->nombre }}</td>
      <td class="border px-4 py-2">{{ $curso->descripcion }}</td>
      <td class="border px-4 py-2">{{ $curso->precio }}</td>
    </tr>
    @endforeach
  </tbody>
</table>
@endsection
```

10. Completar el código delos controladores,las vistas para los tres modelos utlizandoel layout

Sección de Manejo de Usuarios en Laravel 12

Laravel ya incluye el modelo User, pero debemos configurar **autenticación** y la **UI con Blade + Tailwind**.

1. Instalar Breeze (autenticación ligera con Tailwind)

Ejecuta en tu proyecto:

```
composer require laravel/breeze --dev
```

```
php artisan breeze:install blade
```
Esto genera:

- Rutas para login, register, forgot password.
- Vistas en Blade con **Tailwind**.
- Controladores y lógica de sesión.

Luego instala dependencias y compila:

```
npm install
npm run dev
```
Migra para crear la tabla users:

```
php artisan migrate
```
2. Configuración de Rutas Protegidas

Breeze ya genera un dashboard. Vamos a proteger nuestras rutas de cursos/personas/inscripciones.

En routes/web.php:

```
Route::get('/', function () {
   return view('welcome');
});

Route::middleware(['auth'])->group(function () {
   Route::resource('cursos', CursoController::class);
   Route::resource('personas', PersonaController::class);
   Route::resource('inscripciones', InscripcionController::class);
});
```

Ahora, para entrar a cursos/personas/inscripciones el usuario debe estar logueado.

3. Agregar Roles a los Usuarios (opcional pero recomendable)

Migración para roles

```
php artisan make:migration add_role_to_users_table --table=users
```
En la migración:

```
public function up(): void
{
   Schema::table('users', function (Blueprint $table) {
      $table->enum('role', ['admin','user'])->default('user');
   });
}
```

```
public function down(): void
{
   Schema::table('users', function (Blueprint $table) {
      $table->dropColumn('role');
   });
}
```

Ejecuta:

```
php artisan migrate
```

4. Middleware para Controlar Acceso por Rol

Crea un middleware:

```
php artisan make:middleware RoleMiddleware
En app/Http/Middleware/RoleMiddleware.php:
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;

class RoleMiddleware
{
   public function handle(Request $request, Closure $next, $role)
   {
      if (auth()->check() && auth()->user()->role === $role) {
         return $next($request);
      }

      abort(403, 'Acceso no autorizado.');
   }
}
Regístralo en app/Http/Kernel.php:
protected $routeMiddleware = [
   // otros middlewares
   'role' => \App\Http\Middleware\RoleMiddleware::class,
];
```

5. Usar Middleware en Rutas

Ejemplo: solo los administradores gestionan cursos:

```
Route::middleware(['auth','role:admin'])->group(function () {
   Route::resource('cursos', CursoController::class);
});
```

Los usuarios normales pueden inscribirse:

```
Route::middleware(['auth','role:user'])->group(function () {
```

```
    Route::resource('inscripciones', InscripcionController::class);
});
```

6. Ajuste de Layout para Mostrar Usuario

```
En tu layout resources/views/layouts/app.blade.php, puedes mostrar al usuario
logueado:
<nav class="bg-blue-600 p-4 text-white flex justify-between">
  <div>
    <a href="{{ route('cursos.index') }}" class="mr-4">Cursos</a>
    <a href="{{ route('personas.index') }}" class="mr-4">Personas</a>
    <a href="{{ route('inscripciones.index') }}">Inscripciones</a>
  </div>
  <div>
    @auth
      <span>{{ auth()->user()->name }} ({{ auth()->user()->role }})</span>
      <form action="{{ route('logout') }}" method="POST" class="inline">
        @csrf
        <button type="submit" class="ml-2 bg-red-500 px-3 py-1
rounded">Salir</button>
      </form>
    @endauth
  </div>
</nav>
```

7. Seeder para Crear Usuario Admin

En DatabaseSeeder.php:

```
  User::factory()->create([
   'name' => 'Administrador',
   'email' => 'admin@cursos.com',
   'password' => bcrypt('admin123'),
   'role' => 'admin'
]);
```

Así podrás ingresar con:

```
email: admin@cursos.com
pass: admin123
```