

Manual programador

En este manual se explican todos los pasos necesarios para llevar a cabo un proyecto en Laravel.

En primer lugar se crea el proyecto dentro de la carpeta htdocs en xampp:

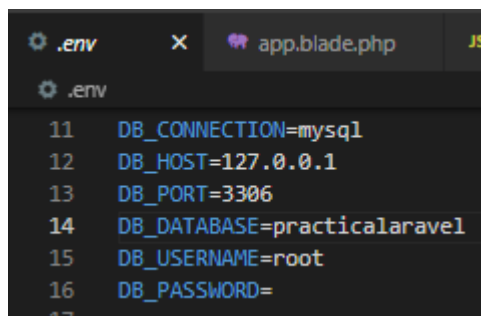
```
C:\xampp\htdocs>composer create-project laravel/laravel formula1
```

Tras ello crearemos las migraciones, que serán las tablas de nuestra futura base de datos:

```
C:\xampp\htdocs>cd formula1  
C:\xampp\htdocs\formula1>php artisan make:migration escuderias
```

Análogamente se crearán el resto de tablas (escuderías, pilotos, circuitos y resultados) prestando especial atención al orden en el que se crean, ya que es necesario que las tablas fuertes se creen de forma previa a las que son débiles.

Tras ello, se modifica el archivo .env para establecer cuál será la base de datos donde crearemos nuestras tablas:



```
.env  
11 DB_CONNECTION=mysql  
12 DB_HOST=127.0.0.1  
13 DB_PORT=3306  
14 DB_DATABASE=practicalaravel  
15 DB_USERNAME=root  
16 DB_PASSWORD=  
17
```

Creamos nuestras tablas:

```

    */
    public function up(): void
    {
        Schema::create('escuderias', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->bigIncrements('id');
            $table->string('nombre');
            $table->string('nacionalidad');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::drop('escuderias');
    }

```

```

    public function up(): void
    {
        Schema::create('pilotos', function (Blueprint $table) {
            $table->engine = 'InnoDB';
            $table->bigIncrements('id');
            $table->string('nombre_completo');
            $table->string('nacionalidad');
            $table->bigInteger('escuderia_id')->unsigned();
            $table->foreign('escuderia_id')->references('id')->on('escuderias')->onDelete('cascade');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::drop('pilotos');
    }

```

```

    */
    public function up(): void
    {
        Schema::create('circuitos', function (Blueprint $table) {
            $table->bigIncrements('id');
            $table->engine = 'InnoDB';
            $table->string('nombre');
            $table->string('pais');
            $table->bigInteger('vueltas');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        //
        Schema::drop('circuitos');
    }

```

```

public function up(): void
{
    Schema::create('resultados', function (Blueprint $table) {
        $table->engine = 'InnoDB';
        $table->bigIncrements('id');
        $table->bigInteger('circuit_id')->unsigned();
        $table->bigInteger('numero_piloto')->unsigned();
        $table->foreign('circuit_id')->references('id')->on('circuitos')->onDelete('cascade');
        $table->foreign('numero_piloto')->references('id')->on('pilotos')->onDelete('cascade');
        $table->bigInteger('posicion')->unsigned();
        $table->timestamps();
    });
}

/**
 * Reverse the migrations.
 */
public function down(): void
{
    Schema::drop('resultados');
}

```

Tras crear las estructuras de las tablas ejecutamos el siguiente comando para crearlas en la propia base de datos del proyecto:

```
\htdocs\formula1>php artisan migrate
```

Después de esto crearemos los seeders, que introducirán los datos a las tablas automáticamente mediante los siguientes comandos:

```
C:\xampp\htdocs\formula1>php artisan make:seeder EscuderiaSeeder
```

Análogamente se crearán PilotoSeeder y CircuitoSeeder. Estas clases accederán cada una a una API donde recogerán los datos del año 2022 y los introducirán en nuestra base de datos:

```

class EscuderiaSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $file = "https://ergast.com/api/f1/2022/constructors.json";
        $data = file_get_contents($file);
        $json = json_decode($data);
        $escuderias = $json->MRData->ConstructorTable->Constructors;
        foreach ($escuderias as $valor) {
            DB::table('escuderias')->insert([
                'nombre' => $valor->name,
                'nacionalidad' => $valor->nationality
            ]);
        }
    }
}

```

Recoge el nombre de la escudería, así como su nacionalidad, y lo introduce en la tabla escuderías.

```

class CircuitoSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $file = "https://ergast.com/api/f1/2022/circuits.json";
        $data = file_get_contents($file);
        $json = json_decode($data);
        $escuderias= $json->MRData->CircuitTable->Circuits;
        foreach($escuderias as $valor){
            DB::table('circuitos')->insert([
                'nombre'=>$valor->circuitName,
                'pais'=>$valor->Location->country,
                'vueltas'=>50
            ]);
        }
    }
}

```

Recoge el nombre del circuito y la ubicación de éste. Se le asigna un número de vueltas por defecto y se introduce en la tabla circuitos.

```

class PilotoSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $file = "https://ergast.com/api/f1/2022/drivers.json";
        $data = file_get_contents($file);
        $json = json_decode($data);
        $pilotos= $json->MRData->DriverTable->Drivers;
        $escuderias =Escuderia::pluck('id')->toArray();
        foreach($pilotos as $valor){
            DB::table('pilotos')->insert([
                'nombre_completo'=>$valor->givenName." ".$valor->familyName,
                'nacionalidad'=>$valor->nationality,
                'escuderia_id'=>$escuderias[array_rand($escuderias)]
            ]);
        }
    }
}

```

Consigue el nombre, apellido y nacionalidad de los pilotos de Fórmula 1 presentes en 2022 y los introduce en la tabla pilotos. Tras ello se le asigna aleatoriamente a cada uno una escudería de la tabla de escuderías.

Tras generar todos los seeders se ejecutan dentro de la clase DatabaseSeeder con el siguiente comando

```

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {
        $this->call(CircuitoSeeder::class);
        $this->call(EscuderiaSeeder::class);
        $this->call(PilotoSeeder::class);
        // \App\Models\User::factory(10)->create();
    }
}

```

```
C:\xampp\htdocs\formula1>php artisan db:seed
```

```
C:\xampp\htdocs\formula1>composer require ibex/crud-generator --dev
```

```
C:\xampp\htdocs\formula1>php artisan vendor:publish --tag=crud
```

```
C:\xampp\htdocs\formula1>php artisan make:crud escuderias
```

Ahora generaremos el CRUD para nuestras tablas. Esto nos facilitará la creación los modelos, las vistas y los controladores de nuestro proyecto. Replicaremos lo mismo para todas las tablas.

Tras ello crearemos los métodos de autenticación incorporando la iu de laravel en nuestro proyecto, donde emplearemos el bootstrap.

```
C:\xampp\htdocs\formula1>composer require laravel/ui
```

```
C:\xampp\htdocs\formula1>php artisan ui bootstrap --auth
```

Iremos al archivo web.php, que gestiona las rutas de nuestro proyecto, implementamos las rutas que usaremos y les añadimos el modificador para que solo puedan emplearlas usuarios autenticados:

```
Auth::routes();
Route::resource('escuderias',App\Http\Controllers\EscuderiaController::class)->middleware('auth');
Route::resource('pilotos',App\Http\Controllers\PilotoController::class)->middleware('auth');
Route::resource('circuitos',App\Http\Controllers\CircuitoController::class)->middleware('auth');
Route::resource('resultados',App\Http\Controllers\ResultadoController::class)->middleware('auth');
Route::get('/home', [App\Http\Controllers\HomeController::class, 'index'])->name('home');
/*
```

Tras ello, en la clase app.blade, dentro de layouts, creamos el acceso a la página principal que muestra los datos de cada tabla

```
<!-- Left Side Of Navbar -->

<ul class="navbar-nav me-auto">
  <li class="nav-item">
    <a class="nav-link" href="{{ route('escuderias.index') }}">{{ __('Escuderias') }}</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{ route('pilotos.index') }}">{{ __('Pilotos') }}</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{ route('circuitos.index') }}">{{ __('Circuitos') }}</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{ route('resultados.index') }}">{{ __('Resultados') }}</a>
  </li>
</ul>
```

Después de eso modificaremos las clases PilotoController y ResultadoController, que son los controladores de las tablas débiles de nuestra base de datos, añadiendo un pluck en los métodos de creación, modificación y muestra de datos, para que en lugar de mostrar el número identificativo de la clave foránea nos muestre el nombre asociado en su lugar:

```

    return redirect(route('home'));
}

public function create()
{
    $piloto = new Piloto();
    $escuderias = Escuderia::pluck('nombre','id');
    return view('piloto.create', compact('piloto','escuderias'));
}

```

```

public function show($id)
{
    $resultado = Resultado::find($id);
    $pilotos = Piloto::pluck('nombre_completo','id');
    $circuitos = Circuito::pluck('nombre','id');
    return view('resultado.show', compact('resultado','pilotos','circuitos'));
}

```

Y con ello también se modifican las vistas asociadas (form.blade)

```

</div>
<div class="form-group">
  {{ Form::label('escuderia') }}
  {{ Form::select('escuderia_id',$escuderias, $piloto->escuderia_id, [
    {!! $errors->first('escuderia', '<div class="invalid-feedback">:message</div>
  }}
  }}
</div>

```

```

<div class="form-group">
  {{ Form::label('circuit_id') }}
  {{ Form::select('circuit_id',$circuitos, $resultado->circuit_id, ['class' =>
    {!! $errors->first('circuit_id', '<div class="invalid-feedback">:message</div>
  }}
  }}
</div>
<div class="form-group">
  {{ Form::label('numero_piloto') }}
  {{ Form::select('numero_piloto',$pilotos, $resultado->numero_piloto, ['class' =>
    {!! $errors->first('numero_piloto', '<div class="invalid-feedback">:message</div>
  }}
  }}
</div>

```

e index.blade

```

<tr>
  <th>No</th>

  <th>Nombre Completo</th>
  <th>Nacionalidad</th>
  <th>Escuderia</th>

  <th></th>
</tr>
</thead>
<tbody>
  @foreach ($pilotos as $piloto)
    <tr>
      <td>{{ ++$i }}</td>

      <td>{{ $piloto->nombre_completo }}</td>
      <td>{{ $piloto->nacionalidad }}</td>
      <td>{{ $piloto->escuderia->nombre }}</td>

      <td>
        <form action="{{ route('pilotos.destroy',$piloto->id) }}">
          <a class="btn btn-sm btn-primary" href="{{ route('pilotos.show',$piloto->id) }}">Ver</a>
          <a class="btn btn-sm btn-success" href="{{ route('pilotos.create',$piloto->id) }}">Crear</a>
          @csrf
          @method('DELETE')
          <button type="submit" class="btn btn-danger btn-sm">Eliminar</button>
        </form>
      </td>
    </tr>
  @endforeach
</tbody>
</table>

```

```

<thead class="thead">
  <tr>
    <th>No</th>

    <th>Circuito</th>
    <th>Piloto</th>
    <th>Posicion</th>

    <th></th>
  </tr>
</thead>
<tbody>
  @foreach ($resultados as $resultado)
    <tr>
      <td>{{ ++$i }}</td>

      <td>{{ $resultado->circuito->nombre }}</td>
      <td>{{ $resultado->piloto->nombre_completo }}</td>
      <td>{{ $resultado->posicion }}</td>

      <td>
        <form action="{{ route('resultados.destroy',$resultado->id) }}">
          <a class="btn btn-sm btn-primary" href="{{ route('resultados.show',$resultado->id) }}">Ver</a>
          <a class="btn btn-sm btn-success" href="{{ route('resultados.create',$resultado->id) }}">Crear</a>
          @csrf
          @method('DELETE')
          <button type="submit" class="btn btn-danger btn-sm">Eliminar</button>
        </form>
      </td>
    </tr>
  @endforeach
</tbody>
</table>

```