

Asignación 2: Conexión a la Base de Datos y Eloquent



Nombre: Juan Antonio Casanova
Torres

Fecha: 24/11/2025

1. Descripción del Proyecto

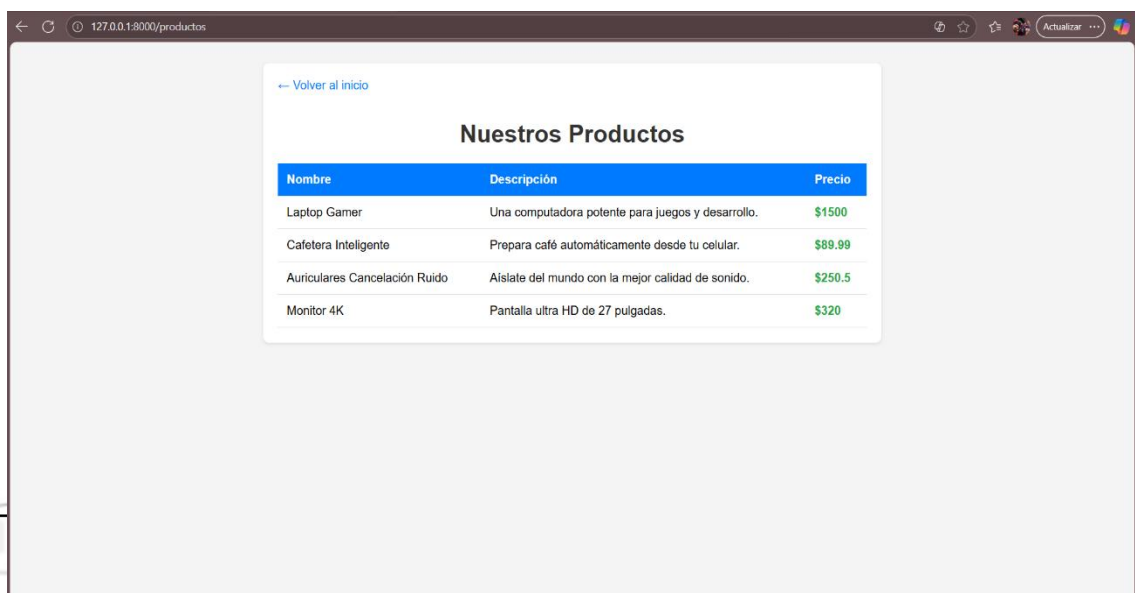
El objetivo de esta fase fue implementar la capa de persistencia de datos. Se configuró una base de datos SQLite para almacenar información de forma permanente, permitiendo que la aplicación recuerde los datos incluso después de cerrarse.

Implementación Técnica:

1. **Configuración de Base de Datos:** Se configuró el entorno (.env) para utilizar el driver sqlite y se creó el archivo físico database.sqlite como contenedor de datos.
2. **Migraciones (Estructura):** Se creó una migración para la tabla productos definiendo el esquema necesario:
 - id: Identificador único (autoincremental).
 - nombre: Cadena de texto (string).
 - descripcion: Texto largo (text).
 - precio: Número decimal con precisión (decimal).
 - timestamps: Marcas de tiempo de creación y actualización.
3. **Modelo Eloquent:** Se generó la clase app/Models/Producto.php. Este modelo actúa como intermediario, permitiendo interactuar con la tabla productos usando objetos y métodos de PHP en lugar de escribir consultas SQL manuales.
4. **Seeding (Datos de Prueba):** Se implementó la clase ProductoSeeder para poblar la base de datos automáticamente con registros iniciales, facilitando las pruebas sin necesidad de inserción manual.
5. **Visualización:** Se creó el ProductoController que utiliza el método Producto::all() para recuperar los registros y enviarlos a la vista productos.blade.php, donde se renderizan en una tabla HTML mediante un bucle @foreach.

2. Evidencia de Funcionamiento

Vista de Productos (/productos) Muestra la tabla HTML renderizando los datos recuperados de la base de datos SQLite.



Evidencia de Base de Datos Captura de la terminal mostrando la ejecución exitosa de las migraciones y el seeder.

```
Simbolo del sistema - php ar x + v
C:\Users\casan\Asignacion1>php artisan migrate
INFO Running migrations.
2025_11_23_220750_create_productos_table ..... 19.28ms DONE
C:\Users\casan\Asignacion1>php artisan db:seed --class=ProductoSeeder
INFO Seeding database.
C:\Users\casan\Asignacion1>php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
2025-11-23 16:46:57 /productos ..... ~ 1s
2025-11-23 16:46:59 /favicon.ico ..... ~ 0.55ms
^C
C:\Users\casan\Asignacion1>php artisan serve
INFO Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
2025-11-23 23:19:26 /productos ..... ~ 1s
2025-11-23 23:19:27 /favicon.ico ..... ~ 0.91ms
2025-11-23 23:19:32 /productos ..... ~ 508.01ms
2025-11-23 23:19:32 /favicon.ico ..... ~ 0.48ms
2025-11-23 23:21:38 /productos ..... ~ 3s
2025-11-23 23:21:41 /favicon.ico ..... ~ 0.55ms
2025-11-23 23:21:50 /bienvenida ..... ~ 1s
2025-11-23 23:21:51 /favicon.ico ..... ~ 0.46ms
2025-11-23 23:22:00 /productos ..... ~ 510.81ms
2025-11-23 23:22:01 /favicon.ico ..... ~ 0.65ms
2025-11-23 23:37:24 /productos ..... ~ 1s
2025-11-23 23:37:24 /favicon.ico ..... ~ 1s
```

3. Reflexión: Desafíos de Migraciones y Eloquent

El cambio de mentalidad con las Migraciones fue uno de los principales aprendizajes fue entender que las Migraciones son como un "control de versiones" para la base de datos. Al principio, es común pensar en crear las tablas manualmente en un gestor de base de datos. Sin embargo, el desafío fue adaptarse a definirlas mediante código PHP (Schema::create).

Esto permite que, si borro mi base de datos o muevo el proyecto a otra computadora, con un solo comando (php artisan migrate) toda la estructura se reconstruye idéntica.

La abstracción de Eloquent ORM para trabajar con Eloquent presentó el reto de confiar en que Laravel está haciendo el trabajo pesado "por detrás".

- En lugar de escribir `SELECT * FROM productos`, simplemente usamos `Producto::all()`.
- El desafío fue comprender cómo configurar el Modelo (`$fillable`) para permitir la inserción masiva de datos de forma segura.
- Eloquent facilita enormemente el manejo de datos, ya que convierte cada fila de la tabla en un Objeto PHP, permitiendo acceder a sus propiedades (ej. `$producto->nombre`) de forma muy intuitiva en la vista.

Conclusión: La combinación de Migraciones y Eloquent elimina la necesidad de escribir SQL repetitivo y propenso a errores, permitiendo centrarse en la lógica de la aplicación y garantizando que la estructura de datos sea consistente y fácil de mantener.