PROYECTO FINAL

PROGRAMACION WEB-DINAMICO

DR. GUILLERMO MONROY RODRÍGUEZ

Presentado por:
Sebastian Robledo Torres

Sistema de Gestión de Inventario Básico

Universidad: Universidad Autónoma Metropolitana
Curso: Programación Web Dinámico
Profesor: Dr. Guillermo Monroy
Stack Tecnológico: Angular + Angular Material + Spring Boot + MySQL
Fecha: 12 Diciembre 2024

1. Introducción

El presente documento describe la implementación de un Sistema de Gestión de Inventario desarrollado como proyecto final de la materia Programación de Web Dinámico. El sistema permite administrar productos, categorías y movimientos de stock mediante una aplicación web full-stack que integra Angular en el frontend y Spring Boot en el backend.

Características principales:

- CRUD completo de productos y categorías
- Registro de movimientos de inventario (entradas/salidas)
- Dashboard con métricas en tiempo real
- Sistema de alertas para productos con bajo stock
- Búsqueda y filtrado avanzado de productos
- Documentación API con Swagger
- Validaciones en frontend y backend

2. REQUERIMIENTOS DEL PROYECTO

2.1 Descripción General

Los estudiantes desarrollan un proyecto completo con:

- **Frontend:** Angular + Angular Material
- **Backend:** Spring Boot
- **Base de datos:** MySQL
- **Documentación:** Swagger
- **Tiempo estimado:** 7 jornadas de aproximadamente 4 horas cada una
- **Autenticación:** No se requiere

2.2 Funcionalidades Requeridas

Proyecto Seleccionado: Sistema de Gestión de Inventario Básico

Módulos principales:

1. **Gestión de Productos**
    - CRUD completo (Crear, Leer, Actualizar, Eliminar)
    - Atributos: nombre, descripción, categoría, precio, stock actual
    - Registro de movimientos de stock (entrada/salida)
    - Tabla con filtros, paginación y búsqueda
2. **Gestión de Categorías**
    - CRUD completo de categorías
3. **Dashboard**
    - Métricas simples:
        - Productos con bajo stock
        - Total de categorías
    - Visualización clara de información
4. **Validaciones**
    - Frontend: Formularios reactivos
    - Backend: Bean Validation
5. **Confirmaciones**
    - Ventanas de confirmación para acciones destructivas
6. **Documentación**
    - Swagger: Documentación clara, organizada y completa

2.3 Requerimientos Técnicos

Frontend (Angular)

- Angular 15+
- Angular Material
- Uso obligatorio de:
    - Routing
    - Servicios para comunicación HTTP
    - Formularios reactivos
    - Tablas con paginación, ordenamiento y filtros
    - Componentes organizados por módulos
- Manejo básico de errores
- Interfaz consistente con Angular Material

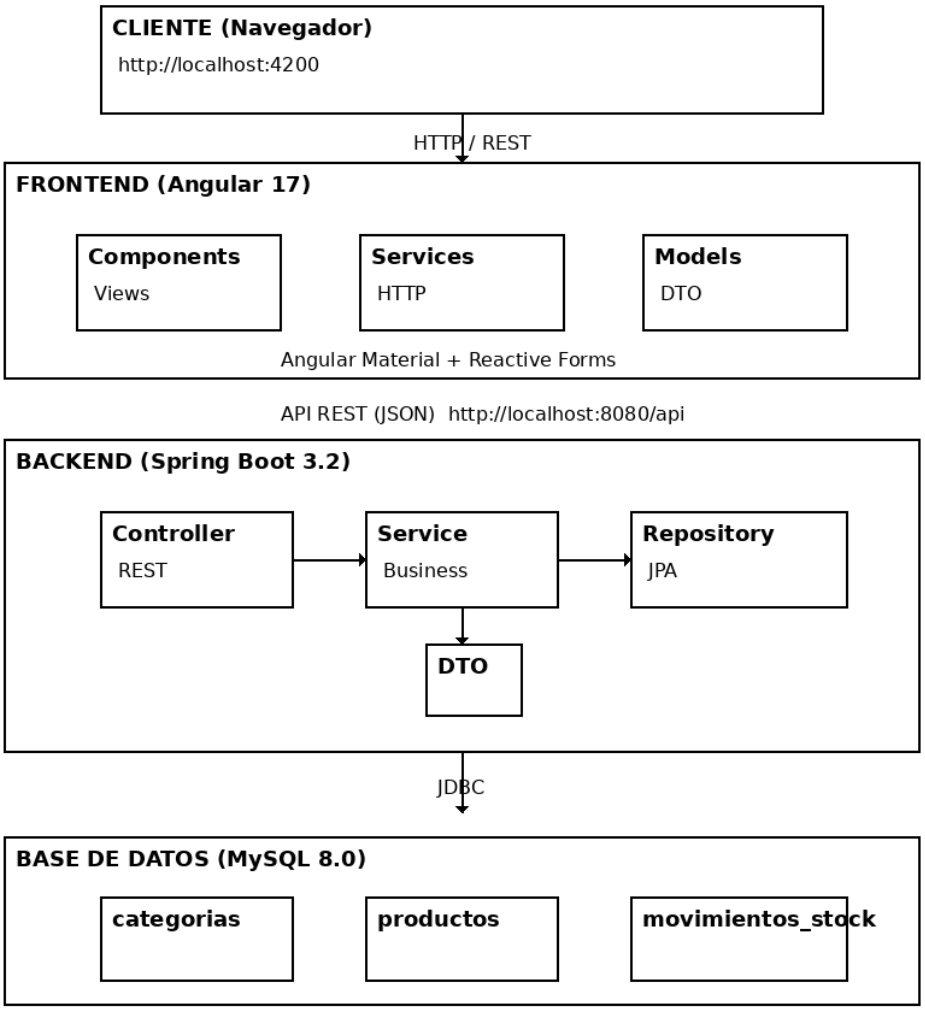
Backend (Spring Boot)

- Spring Web
- Spring Data JPA
- Spring Validation
- MySQL
- Swagger (OpenAPI)
- Arquitectura recomendada:
    - Controller
    - Service
    - Repository
    - Entity
    - DTO

- Config

3. ARQUITECTURA DEL SISTEMA

3.1 Arquitectura General



3.2 Patrón de Arquitectura Backend

Arquitectura en Capas (Layered Architecture)

1. **Controller Layer** - Manejo de peticiones HTTP
2. **Service Layer** - Lógica de negocio
3. **Repository Layer** - Acceso a datos
4. **Entity Layer** - Modelo de dominio
5. **DTO Layer** - Transferencia de datos

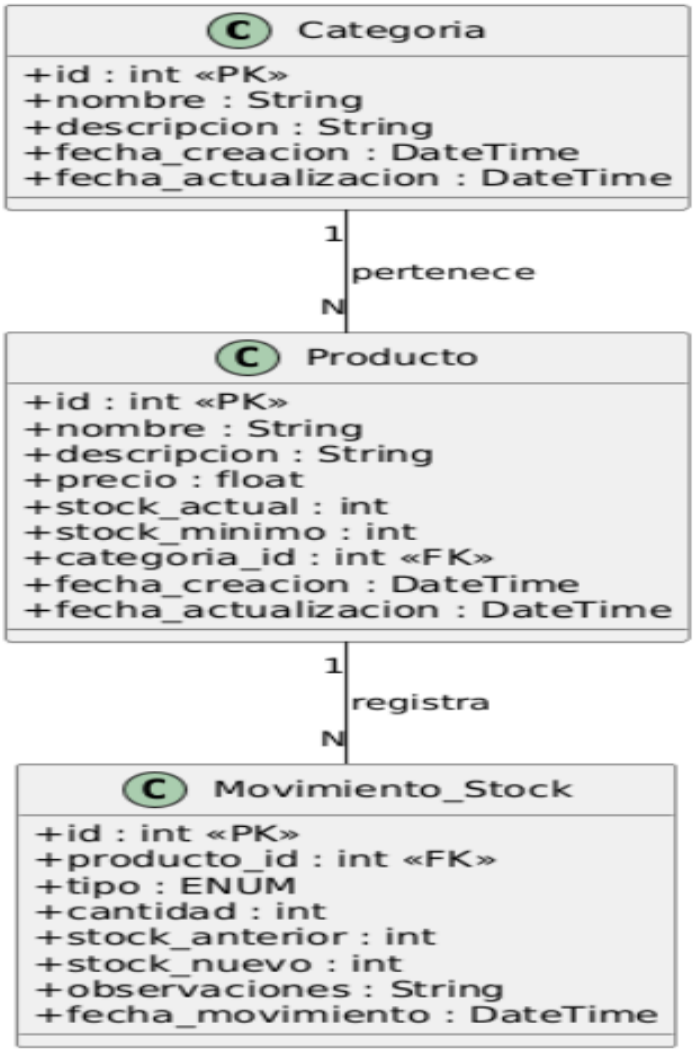
3.3 Patrón de Arquitectura Frontend

Arquitectura por Features (Feature-Based Architecture)

- Componentes standalone
- Servicios inyectables
- Routing modular
- Material Design System

4. MODELO DE DATOS

4.1 Diagrama Entidad-Relación



4.2 Descripción de Entidades

Tabla: CATEGORÍAS

Campo	Tipo	Descripción
id	BIGINT (PK)	Identificador único
nombre	VARCHAR(255)	Nombre de la categoría (único)
descripcion	VARCHAR(500)	Descripción opcional
fecha_creacion	DATETIME	Fecha de creación (auto)
fecha_actualizacion	DATETIME	Última actualización (auto)

Tabla: PRODUCTOS

Campo	Tipo	Descripción
id	BIGINT (PK)	Identificador único
nombre	VARCHAR(255)	Nombre del producto
descripcion	VARCHAR(1000)	Descripción opcional
precio	DECIMAL(10,2)	Precio del producto
stock_actual	INTEGER	Cantidad en inventario
stock_minimo	INTEGER	Stock mínimo (default: 5)
categoria_id	BIGINT (FK)	Referencia a categoría
fecha_creacion	DATETIME	Fecha de creación
fecha_actualizacion	DATETIME	Última actualización

Tabla: MOVIMIENTOS_STOCK

Campo	Tipo	Descripción
id	BIGINT (PK)	Identificador único
producto_id	BIGINT (FK)	Referencia a producto
tipo	ENUM	ENTRADA o SALIDA
cantidad	INTEGER	Cantidad del movimiento
stock_anterior	INTEGER	Stock antes del movimiento

stock_nuevo	INTEGER	Stock después del movimiento
observaciones	VARCHAR(500)	Notas opcionales
fecha_movimiento	DATETIME	Fecha del registro

5. FUNCIONALIDADES IMPLEMENTADAS

5.1 Módulo de Categorías

Operaciones CRUD:

- **Crear:** Formulario con validaciones (nombre requerido)
- **Listar:** Tabla paginada con cantidad de productos
- **Editar:** Actualización de datos existentes
- **Eliminar:** Con confirmación (verifica productos asociados)

Reglas de negocio:

- No se puede eliminar una categoría con productos asociados
- El nombre debe ser único
- Validación de campos obligatorios

5.2 Módulo de Productos

Operaciones CRUD:

- **Crear:** Formulario completo con validaciones
- **Listar:** Tabla con paginación, búsqueda y filtros
- **Editar:** Actualización de todos los campos
- **Eliminar:** Con confirmación previa

Funcionalidades avanzadas:

- Búsqueda en tiempo real por nombre (debounce)
- Filtro por categoría
- Indicador visual de productos con bajo stock
- Paginación configurable (5, 10, 25, 50)
- Formato de precio en moneda

Reglas de negocio:

- Precio debe ser mayor a 0
- Stock no puede ser negativo

- Stock mínimo por defecto: 5 unidades
- Alerta cuando stock_actual ≤ stock_minimo

5.3 Módulo de Movimientos de Stock

Operaciones:

- **Registrar Entrada:** Incrementa stock
- **Registrar Salida:** Decrementa stock
- **Listar:** Historial completo con paginación
- **Filtrar:** Por producto específico

Información registrada:

- Producto afectado
- Tipo de movimiento (ENTRADA/SALIDA)
- Cantidad
- Stock anterior y nuevo
- Fecha y hora del movimiento
- Observaciones opcionales

Reglas de negocio:

- No permite salidas mayores al stock disponible
- Actualiza automáticamente el stock del producto
- Registra auditoría completa de cambios

5.4 Dashboard

Métricas mostradas:

- Total de categorías registradas
- Total de productos en inventario
- Cantidad de productos con bajo stock
- Lista detallada de productos que requieren atención

Características:

- Actualización en tiempo real
- Diseño con tarjetas visuales
- Indicadores de color para alertas
- Tabla de productos críticos

5.5 Validaciones

Frontend (Angular):

- Formularios reactivos con validaciones en tiempo real
- Mensajes de error específicos por campo

- Deshabilitación de botones hasta validación completa
- Validación de tipos de datos (números, decimales)

Backend (Spring Boot):

- Bean Validation con anotaciones
- Validaciones personalizadas en servicios
- Manejo de excepciones HTTP apropiadas
- Mensajes de error descriptivos

5.6 Documentación API (Swagger)

Configuración:

- OpenAPI 3.0
- Interfaz Swagger UI
- Agrupación por controladores

Endpoints documentados:

- Descripción de cada operación
- Parámetros requeridos y opcionales
- Códigos de respuesta HTTP
- Modelos de datos (schemas)
- Ejemplos de peticiones y respuestas

Acceso: `http://localhost:8080/swagger-ui.html`

6. STACK TECNOLÓGICO

6.1 Backend

Tecnología	Versión	Propósito
Java	17	Lenguaje de programación
Spring Boot	3.2.x	Framework principal
Spring Web	3.2.x	API REST
Spring Data JPA	3.2.x	Persistencia de datos
Hibernate	6.x	ORM
MySQL	8.0+	Base de datos

Lombok	1.18.x	Reducción de boilerplate
Spring Validation	3.2.x	Validación de datos
SpringDoc OpenAPI	2.2.x	Documentación Swagger
Maven	3.6+	Gestión de dependencias

6.2 Frontend

Tecnología	Versión	Propósito
Angular	17+	Framework frontend
Angular Material	17+	Componentes UI
TypeScript	5.x	Lenguaje tipado
RxJS	7.x	Programación reactiva
HttpClient	17+	Comunicación HTTP
Reactive Forms	17+	Formularios validados
Angular Router	17+	Navegación SPA

7. ENDPOINTS DE LA API

7.1 Categorías

Método	Endpoint	Descripción
GET	`/api/categorias`	Obtener todas las categorías
GET	`/api/categorias/paginado`	Obtener con paginación
GET	`/api/categorias/{id}`	Obtener por ID
POST	`/api/categorias`	Crear nueva categoría

PUT	`/api/categorias/{id}`	Actualizar categoría
DELETE	`/api/categorias/{id}`	Eliminar categoría

7.2 Productos

Método	Endpoint	Descripción
GET	`/api/productos`	Listar con paginación
GET	`/api/productos/buscar`	Buscar con filtros
GET	`/api/productos/bajo-stock`	Productos críticos
GET	`/api/productos/{id}`	Obtener por ID
POST	`/api/productos`	Crear nuevo producto
PUT	`/api/productos/{id}`	Actualizar producto
DELETE	`/api/productos/{id}`	Eliminar producto

7.3 Movimientos

Método	Endpoint	Descripción
GET	`/api/movimientos`	Listar con paginación
GET	`/api/movimientos/producto/{id}`	Historial por producto
POST	`/api/movimientos`	Registrar movimiento

7.4 Dashboard

Método	Endpoint	Descripción

GET	`/api/dashboard`	Obtener métricas

8. INSTALACIÓN Y CONFIGURACIÓN

8.1 Requisitos Previos

Software necesario:

- JDK 17 o superior
- Node.js 18+ y npm
- MySQL 8.0+
- Maven 3.6+
- Angular CLI 17+
- IDE (IntelliJ IDEA / VS Code)

8.2 Configuración del Backend

Paso 1: Clonar/descargar el proyecto

Paso 2: Configurar MySQL

sql

```
CREATE DATABASE inventory_db;
```

Paso 3: Configurar `application.properties`

properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/inventory_db
spring.datasource.username=root
spring.datasource.password=tu_password
```

Paso 4: Ejecutar

bash

```
cd inventory-backend
mvn clean install
mvn spring-boot:run
```

Verificar: `http://localhost:8080/swagger-ui.html`

8.3 Configuración del Frontend

Paso 1: Instalar dependencias

bash
cd inventory-frontend

npm install

Paso 2: Configurar endpoint en `environment.ts`

typescript
export const environment = {
  production: false,
  apiUrl: 'http://localhost:8080/api'
};

Paso 3: Ejecutar

bash

ng serve

Verificar: `http://localhost:4200`

9. CUMPLIMIENTO DE REQUERIMIENTOS

9.1 Checklist de Funcionalidades

Requerimiento	Estado	Evidencia
Funcionalidades Mínimas		
CRUD completo en todos los módulos principales	*	Categorías, Productos, Movimientos
Validaciones frontend	*	Reactive Forms con validaciones
Validaciones backend	*	Bean Validation + lógica de negocio

Listados con filtros	*	Búsqueda y filtro por categoría
Paginación	*	Todas las tablas paginadas
Ordenamiento	*	MatSort en tablas
Ventanas de confirmación	*	Dialog para eliminaciones
Documentación Swagger	*	Completa y organizada
Frontend (Angular)		
Angular 15+	*	Versión 17
Angular Material	*	Todos los componentes
Routing	*	Navegación por módulos
Servicios HTTP	*	HttpClient en todos los servicios
Formularios reactivos	*	Todos los formularios
Tablas con paginación	*	MatPaginator implementado
Componentes por módulos	*	Arquitectura por features
Manejo de errores	*	SnackBar para notificaciones
Interfaz consistente	*	Material Design aplicado
Backend (Spring Boot)		
Spring Web	*	API REST implementada
Spring Data JPA	*	Persistencia con JPA
Spring Validation	*	Validaciones con anotaciones
MySQL	*	Base de datos configurada
Swagger (OpenAPI)	*	Documentación completa
Arquitectura		
Controller	*	4 controladores REST
Service	*	Interfaces + implementaciones

Repository	*	JpaRepository extendido
Entity	*	3 entidades mapeadas
DTO	*	4 DTOs para transferencia
Config	*	CORS + OpenAPI

9.2 Funcionalidades Adicionales

Mejoras implementadas más allá de los requisitos:

1. **Dashboard Interactivo**
    - Métricas en tiempo real
    - Tarjetas con iconos Material
    - Lista de productos críticos
2. **Búsqueda Avanzada**
    - Búsqueda con debounce (300ms)
    - Búsqueda mientras se escribe
    - Combinación de filtros
3. **Experiencia de Usuario**
    - Notificaciones con SnackBar
    - Loading states
    - Navegación lateral responsive
    - Indicadores visuales (chips de colores)
4. **Auditoría**
    - Timestamps automáticos
    - Historial completo de movimientos
    - Stock anterior y nuevo registrados

10. PRUEBAS Y VALIDACIÓN

10.1 Pruebas Funcionales Realizadas

Categorías:

- Crear categoría nueva
- Editar categoría existente
- Listar con paginación
- Eliminar sin productos
- Validar eliminación con productos
- Validar nombre duplicado

Productos:

- Crear producto con todos los campos
- Editar producto existente
- Búsqueda por nombre
- Filtro por categoría
- Validar precio negativo
- Validar stock negativo
- Verificar alerta de bajo stock
- Eliminar producto

Movimientos:

- Registrar entrada de stock
- Registrar salida de stock
- Validar salida mayor al disponible
- Verificar actualización de stock
- Listar historial
- Filtrar por producto

Dashboard:

- Mostrar métricas correctas
- Actualizar en tiempo real
- Listar productos bajo stock

10.2 Valoraciones Verificadas

Frontend:

- Campos requeridos
- Tipos de datos (number, text)
- Valores mínimos (precio > 0)
- Mensajes de error específicos
- Botones deshabilitados en formularios inválidos

Backend:

- @NotNull en campos requeridos
- @Min para valores mínimos
- @DecimalMin para precios
- Validaciones de lógica de negocio
- Mensajes HTTP apropiados (400, 404, 500)

11. CONCLUSIONES

11.1 Logros del Proyecto

El Sistema de Gestión de Inventario cumple exitosamente con todos los requerimientos establecidos:

1. **Arquitectura Sólida:** Implementación correcta del patrón MVC con separación de responsabilidades
2. **Funcionalidad Completa:** CRUD operativo en todos los módulos con validaciones robustas
3. **Experiencia de Usuario:** Interfaz intuitiva y responsiva con Angular Material
4. **Documentación:** API completamente documentada con Swagger
5. **Buenas Prácticas:** Código organizado, nomenclatura consistente y patrones de diseño aplicados

11.2 Competencias Desarrolladas

- Desarrollo full-stack con tecnologías modernas
- Diseño e implementación de APIs RESTful
- Manejo de bases de datos relacionales
- Programación reactiva en frontend
- Validación de datos en múltiples capas
- Documentación técnica de APIs
- Arquitectura de aplicaciones web

11.3 Posibles Mejoras Futuras

Funcionalidades:

- Sistema de autenticación y autorización
- Reportes en PDF/Excel
- Gráficos estadísticos
- Gestión de proveedores
- Sistema de alertas por email
- Gestión de usuarios y permisos

Técnicas:

- Testing unitario y de integración
- CI/CD pipeline
- Contenedorización con Docker
- Cache con Redis
- Búsqueda full-text con Elasticsearch

12. ANEXOS

12.1 Estructura de Directorios Backend

src/main/java/com/proyecto/inventory/

```
├── config/
├── controller/
├── dto/
├── entity/
├── repository/
├── service/
│   └── impl/
└── InventorySystemApplication.java
```

12.2 Estructura de Directorios Frontend

```
src/app/
├── core/
│   ├── models/
│   └── services/
├── shared/
│   ├── components/
│   └── material.module.ts
├── features/
│   ├── dashboard/
│   ├── categorias/
│   ├── productos/
│   └── movimientos/
├── app.component.ts
├── app.routes.ts
└── app.config.ts
```

12.3 Tecnologías por Capa

Capa de Presentación:

- Angular 17, Angular Material, TypeScript, RxJS

Capa de Lógica de Negocio:

- Spring Boot, Spring MVC, Bean Validation

Capa de Persistencia:

- Spring Data JPA, Hibernate, MySQL

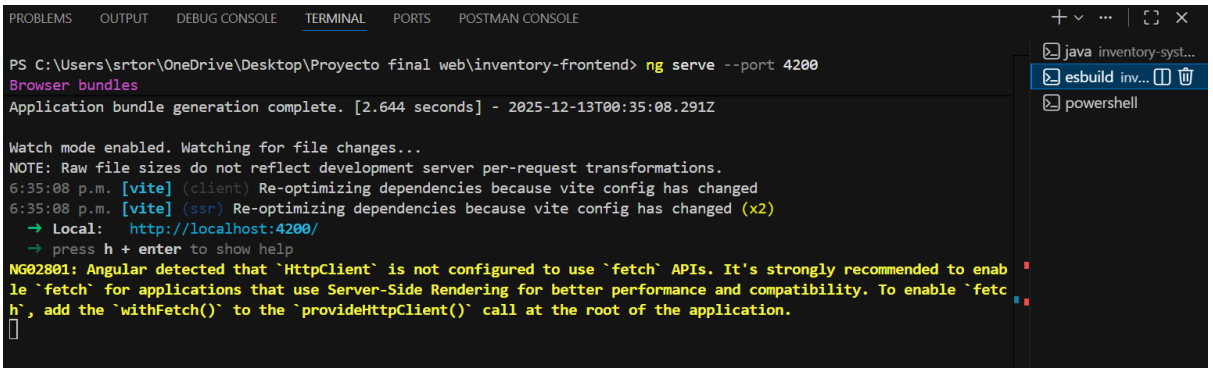
Documentación:

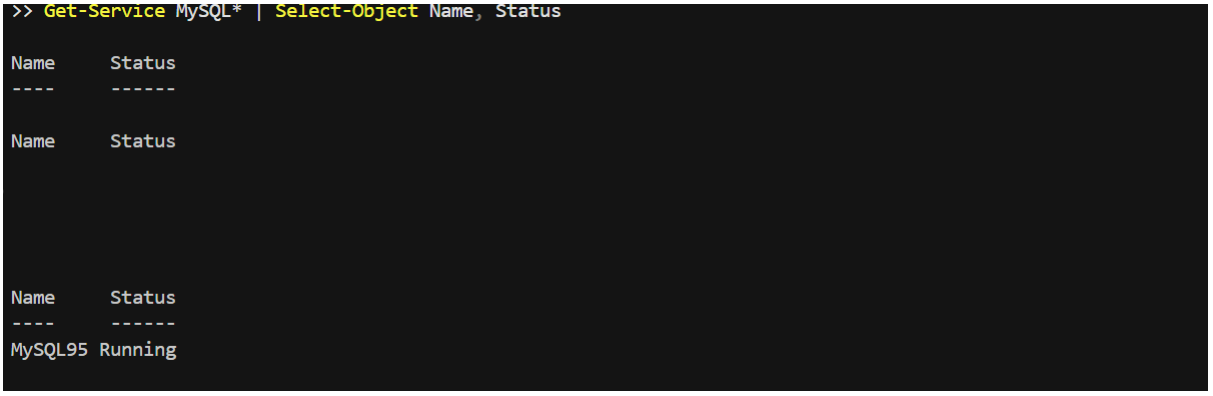
- SpringDoc OpenAPI (Swagger)

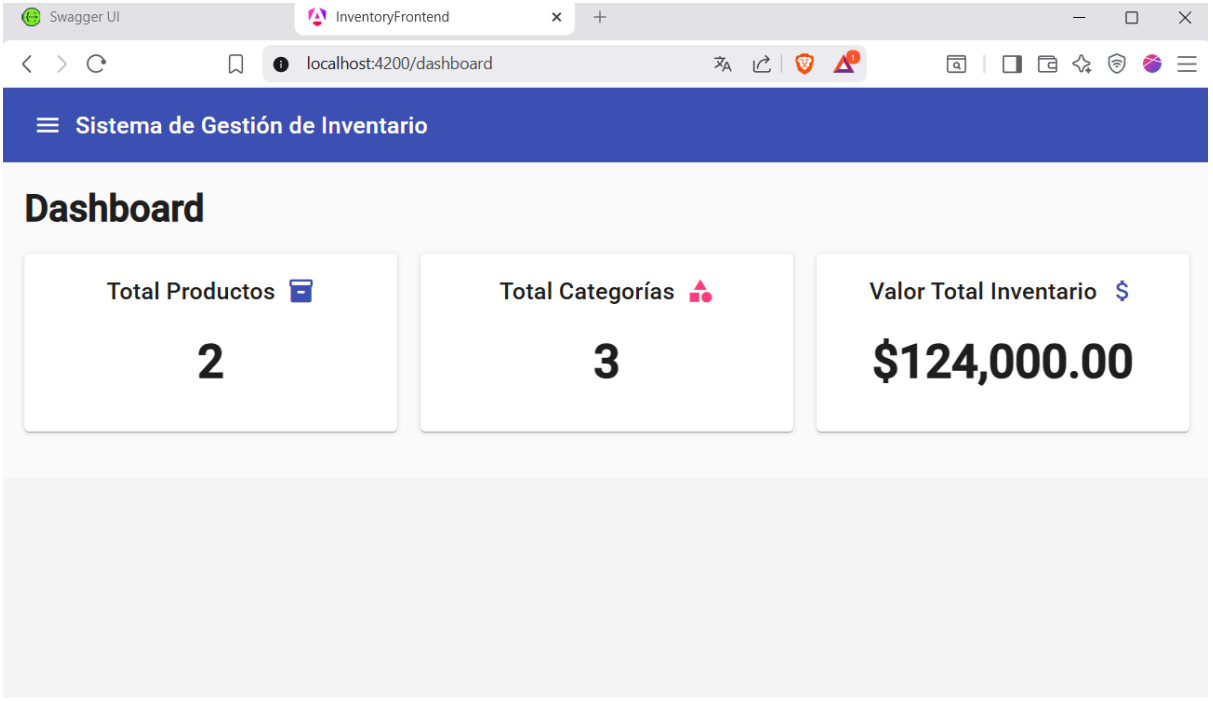
13. REFERENCIAS

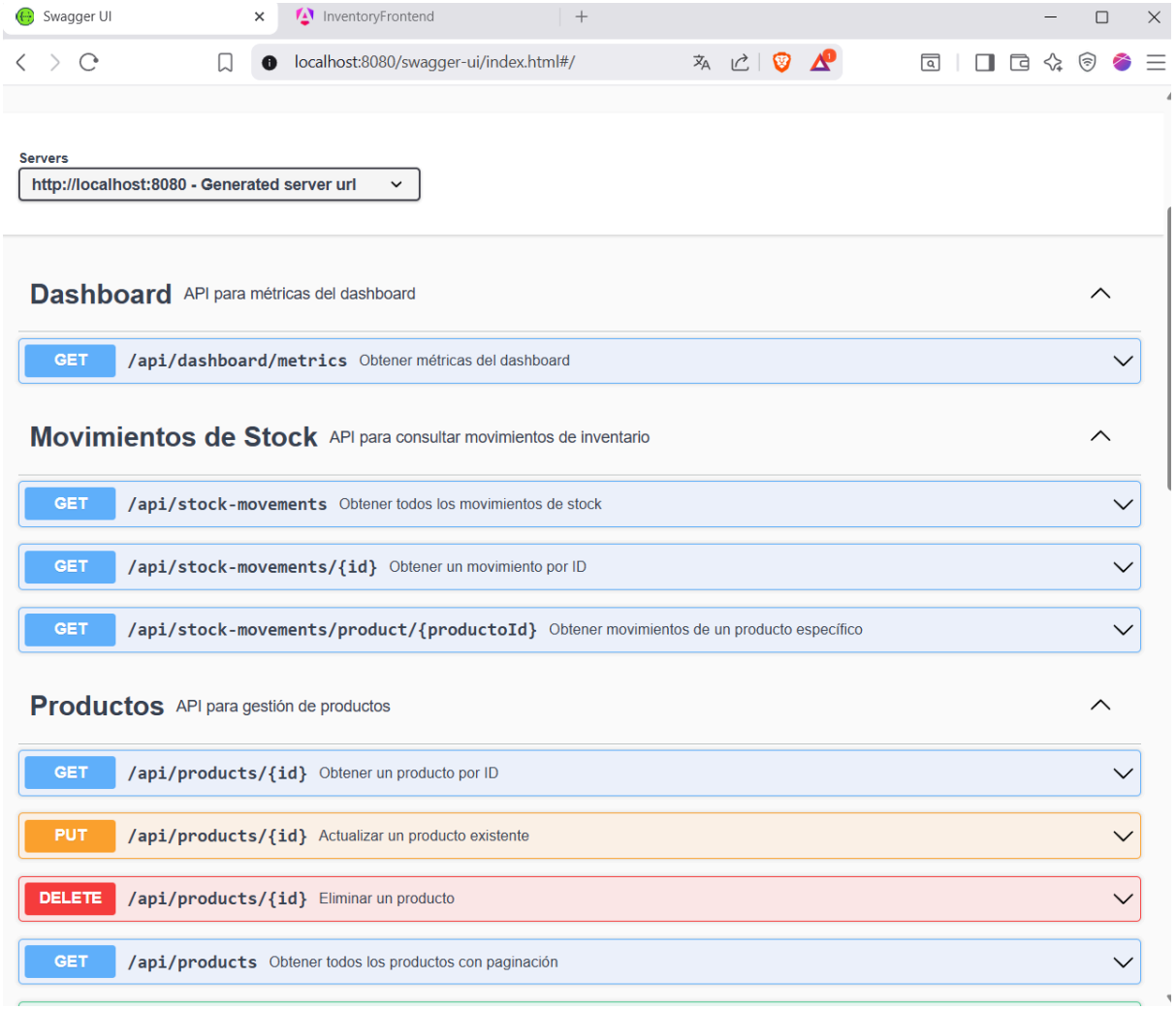
- Spring Boot Documentation
- Angular Documentation
- Angular Material
- Spring Data JPA
- Swagger OpenAPI

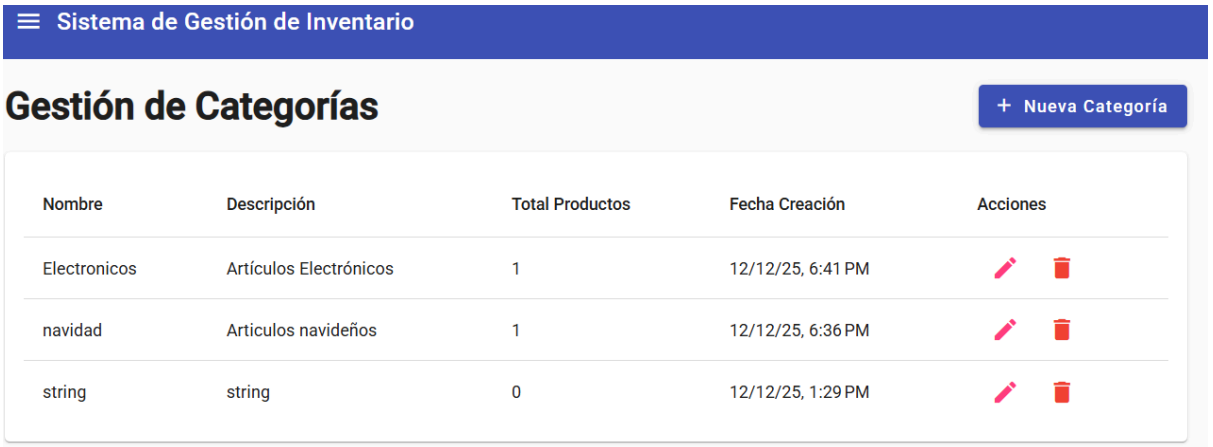
14. RESULTADOS

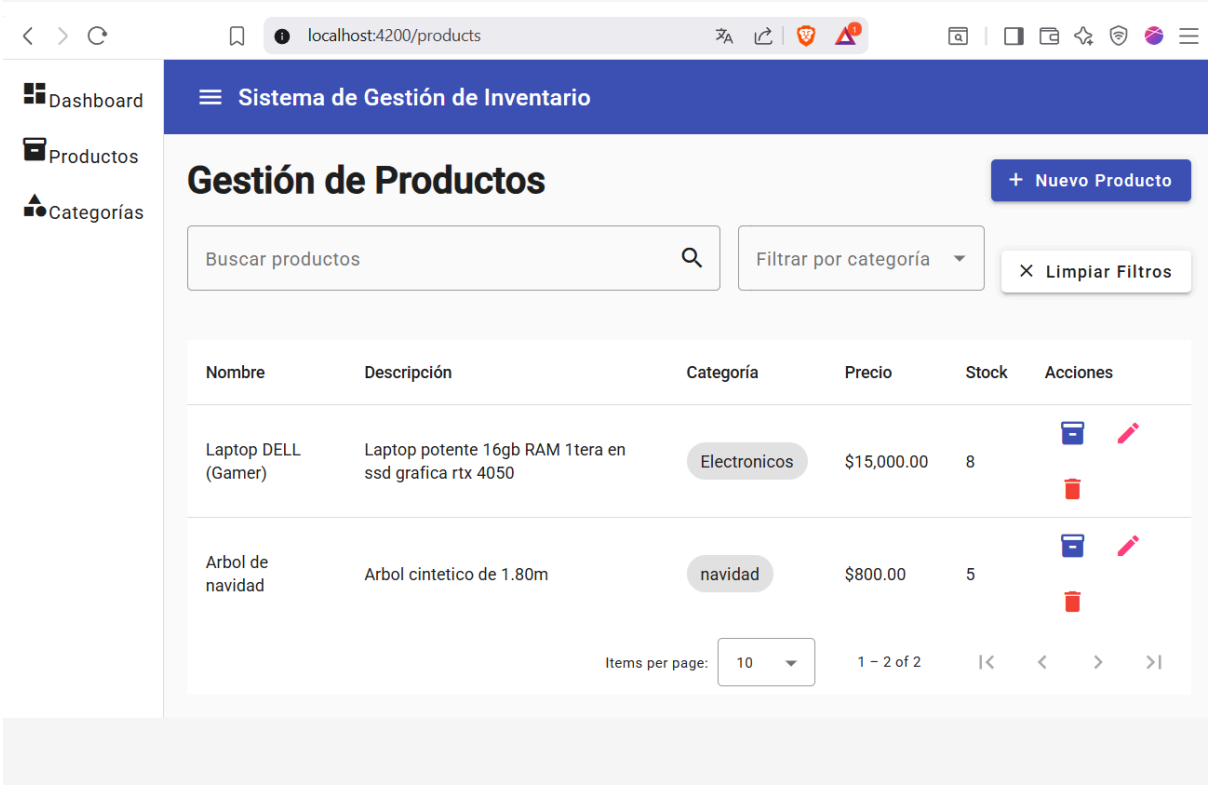





Swagger UI
InventoryFrontend
localhost:4200/dashboard
Sistema de Gestión de Inventario
Dashboard
Total Productos
2
Total Categorías
3
Valor Total Inventario
$124,000.00


Swagger UI
InventoryFrontend
localhost:8080/swagger-ui/index.html#/
Servers
http://localhost:8080 - Generated server url
Dashboard API para métricas del dashboard
GET /api/dashboard/metrics Obtener métricas del dashboard
Movimientos de Stock API para consultar movimientos de inventario
GET /api/stock-movements Obtener todos los movimientos de stock
GET /api/stock-movements/{id} Obtener un movimiento por ID
GET /api/stock-movements/product/{productoId} Obtener movimientos de un producto específico
Productos API para gestión de productos
GET /api/products/{id} Obtener un producto por ID
PUT /api/products/{id} Actualizar un producto existente
DELETE /api/products/{id} Eliminar un producto
GET /api/products Obtener todos los productos con paginación


Sistema de Gestión de Inventario
Gestión de Categorías
+ Nueva Categoría
Nombre
Descripción
Total Productos
Fecha Creación
Acciones
Electronicos
Artículos Electrónicos
1
12/12/25, 6:41 PM
navidad
Articulos navideños
1
12/12/25, 6:36 PM
string
string
0
12/12/25, 1:29 PM


localhost:4200/products
Dashboard
Productos
Categorías
Sistema de Gestión de Inventario
Gestión de Productos
+ Nuevo Producto
Buscar productos
Filtrar por categoría
Limpiar Filtros
Nombre
Descripción
Categoría
Precio
Stock
Acciones
Laptop DELL (Gamer)
Laptop potente 16gb RAM 1tera en ssd grafica rtx 4050
Electronicos
$15,000.00
8
Arbol de navidad
Arbol cintetico de 1.80m
navidad
$800.00
5
Items per page:
10
1 – 2 of 2