Documentación

Repositorio GitHub: https://github.com/mrValdebenito/Tarea2LABIngSoftware.git

Requisitos

Este proyecto consiste en una API REST desarrollada utilizando Spring Boot como *back-end* y MySQL como sistema de gestión de bases de datos. El objetivo es proporcionar al gerente Gustavo una herramienta para gestionar el catálogo de muebles, cotizar productos con variantes, y procesar ventas con control de stock. Para la ejecución del programa, el usuario debe tener instalados el Java Development Kit (JDK 17+), un IDE como IntelliJ IDEA, y XAMPP o un servidor MySQL local. La conexión se define en el archivo `application.properties`, especificando la URL del servidor, el nombre de la base de datos, el usuario y la contraseña.

src > main > resources > application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/evaluacion1?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Crear la base de datos

El programa se conecta a la base de datos evaluacion1. Es fundamental que la estructura de las tablas coincida con la lógica de las entidades de Spring Data JPA. A continuación, se presenta el script SQL completo para la creación de las tablas.

El script SQL para crear la base de datos y la estructura de la mueblería es el siguiente:

```
-- Creación de la base de datos
CREATE DATABASE IF NOT EXISTS evaluacion1 CHARACTER SET utf8mb4 COLLATE
utf8mb4_general_ci;
USE evaluacion1;

-- 1. Tabla MUEBLE
CREATE TABLE IF NOT EXISTS mueble (
    id_mueble INT AUTO_INCREMENT PRIMARY KEY,
    nombre_mueble VARCHAR(100) NOT NULL,
    tipo VARCHAR(50) NOT NULL,
    precio_base DECIMAL(10, 2) NOT NULL,
    stock INT NOT NULL,
    estado ENUM('activo', 'inactivo') NOT NULL,
    tamaño ENUM('Grande', 'Mediano', 'Pequeño') NOT NULL,
```

```
    material VARCHAR(50) NOT NULL
) ENGINE=InnoDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

-- 2. Tabla VARIANTE
CREATE TABLE IF NOT EXISTS variante (
    id_variante INT AUTO_INCREMENT PRIMARY KEY,
    nombre_variante VARCHAR(100) NOT NULL,
    precio_adicional DECIMAL(10, 2) NOT NULL
) ENGINE=InnoDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

-- 3. Tabla COTIZACION
CREATE TABLE IF NOT EXISTS cotizacion (
    id_cotizacion INT AUTO_INCREMENT PRIMARY KEY,
    fecha DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    estado ENUM('Cotizado', 'Vendido') NOT NULL
) ENGINE=InnoDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

-- 4. Tabla DETALLE_COTIZACION
CREATE TABLE IF NOT EXISTS detalle_cotizacion (
    id_detalle INT AUTO_INCREMENT PRIMARY KEY,
    id_cotizacion INT NOT NULL,
    id_mueble INT NOT NULL,
    cantidad INT NOT NULL,
    precio_unitario_final DECIMAL(10, 2) NOT NULL,

    FOREIGN KEY (id_cotizacion) REFERENCES cotizacion(id_cotizacion),
    FOREIGN KEY (id_mueble) REFERENCES mueble(id_mueble)
) ENGINE=InnoDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;

-- 5. Tabla DETALLE_VARIANTES_APLICADAS
CREATE TABLE IF NOT EXISTS detalle_variantes_aplicadas (
    id_detalle_variante INT AUTO_INCREMENT PRIMARY KEY,
    id_detalle INT NOT NULL,
    id_variante INT NOT NULL,

    FOREIGN KEY (id_detalle) REFERENCES detalle_cotizacion(id_detalle),
    FOREIGN KEY (id_variante) REFERENCES variante(id_variante)
) ENGINE=InnoDB CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

Funcionalidad

La API expone sus funcionalidades a través de dos controladores principales que deben ser probados con Postman.

La gestión del catálogo CRUD se realiza mediante el MuebleController. El requisito de desactivar un mueble se maneja mediante el método PATCH, cambiando su estado a inactivo en lugar de eliminar el registro.

- Crear: POST a /api/muebles
- Listar: GET a /api/muebles
- Actualizar: PUT a /api/muebles/*id*
- Desactivar: PATCH a /api/muebles/desactivar/*id*

La lógica se maneja en el CotizacionController.

- Confirmar Venta / Descuento de Stock: PATCH a /api/cotizaciones/confirmar/*id*. Este endpoint ejecuta una transacción que verifica el stock disponible y lo decrementa si la venta es exitosa.

Patrones de Diseño

Se implementaron patrones de diseño para estructurar y organizar la lógica del programa, cumpliendo con la exigencia de aplicar un mínimo de dos patrones:

1. Patrón Arquitectónico de Capas (Controller-Service-Repository): Este patrón desacopla las responsabilidades. El Controller maneja las peticiones HTTP, el Service contiene la lógica de negocio como la verificación de stock y el cálculo de precios, y el Repository se comunica directamente con la base de datos.
2. Patrón Repository: Utilizado a través de Spring Data JPA para abstraer las operaciones de la base de datos, permitiendo realizar el CRUD sin escribir consultas SQL complejas.

Testing

El requisito de testing se cumplió utilizando JUnit y Mockito para probar la lógica más sensible en el CotizacionService.

Las pruebas unitarias validaron los siguientes escenarios:

1. Cálculo de Precios (Variantes): Se verificó que el precio final es igual al precioBase más la suma de los precioAdicional de las variantes seleccionadas.
2. Stock Insuficiente: Se probó que, ante una cantidad de venta mayor al stock, la operación lanza la excepción StockInsuficienteException y no se realiza ningún descuento en el inventario.
3. Venta Exitosa: Se probó que ante stock suficiente, el método confirmarVenta cambia el estado de la cotización a "Vendido" y el stock se decrementa en la cantidad correcta.

Algunas pruebas Postman

Body Cookies Headers (5) Test Results 201 Created

{ } JSON ▷ Preview Visualize


```
{
    "idMueble": 3,
    "nombreMueble": "Mesa Auxiliar",
    "tipo": "Mesa",
    "precioBase": 75.50,
    "stock": 30,
    "estado": "activo",
    "tamaño": "Pequeño",
    "material": "Pino"
}
```


Body Cookies Headers (3) Test Results 204 No Content

Raw ▷ Preview Visualize

```
1
```

Body Cookies Headers (5) Test Results 200 OK

{ } JSON ▷ Preview Visualize


```
[
    {
        "idMueble": 1,
        "nombreMueble": "Sillón de Lujo",
        "tipo": "Sillón",
        "precioBase": 450.00,
        "stock": 8,
        "estado": "inactivo",
        "tamaño": "Grande",
        "material": "Cuero"
    },
    {
        "idMueble": 3,
        "nombreMueble": "Mesa Auxiliar",
        "tipo": "Mesa",
        "precioBase": 75.50,
        "stock": 30,
        "estado": "inactivo",
        "tamaño": "Pequeño",
        "material": "Pino"
    }
]
```


PATCH {{baseUrl}} /api/cotizaciones/confirmar/100

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results 409 Conflict

Raw ▷ Preview Debug with AI

```
stock insuficiente para el mueble: Sillón de Lujo
```

PATCH {{baseUrl}} /api/cotizaciones/confirmar/101

Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results 200 OK

{ } JSON ▷ Preview Visualize


```
{
    "idCotizacion": 101,
    "fecha": "2025-11-04T21:59:16",
    "estado": "Vendido",
    "detalles": [
        {
            "idDetalle": 201,
            "cantidad": 2,
            "precioUnitarioFinal": 500.00,
            "cotizacion": {
                "idCotizacion": 101,
                "fecha": "2025-11-04T21:59:16",
                "estado": "Vendido",
                "detalles": [
```

Luciano Valdebenito
Ingenieria de Software
Docente Roberto Anabalon
24 de septiembre del 2025