UNIVERSIDAD DEL BÍO-BÍO

Facultad de Ciencias Empresariales

Departamento de Ciencias de la Computación y Tecnologías de Información

Ingeniería Civil Informática.

Tarea Ingeniería de software 2

Integrante: Diego Alamos Vallejos

Docente: Gastón Márquez Ortega

Roberto Anabalón Suazo

Asignatura: Ingeniería de Software

Carrera: Ingeniería Civil en Informática

Fecha: 24/09/2025

Introducción sobre evaluación

Se presenta la entrega 2 de la correspondiente evaluación de la asignatura de *Ingenieria de software II .* Consiste en el contexto de una mueblería con el desarrollo de un sistema de gestión de catalogo, cotizaciones y ventas para una empresa de muebles, implementado con SpyBot y MySQL , además de testing con JUnix y Postman

Requisitos previos y cómo ejecutar el programa

- Instalación de java 17
- Instalación de Maven
- instalación de XAMPP (para usar MySQL y phpMyAdmin)
- uso de Visual Studio Code

Arquitectura del sistema

El sistema fue desarrollado bajo una arquitectura MULTICAPA , organizada en 3 niveles:

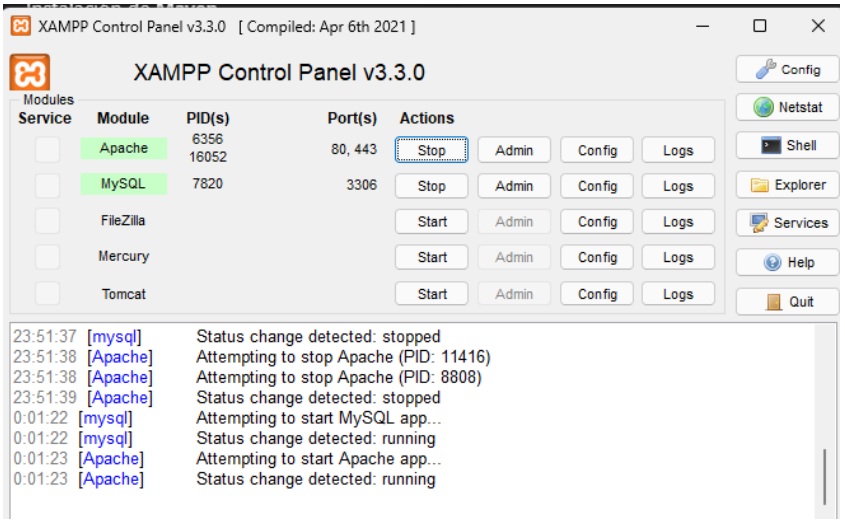
Capa	Contenido	Descripción
Modelo	Entidades JPA (Mueble, Variante, Cotizacion, ItemCotizacion)	Representa la estructura de datos y las relaciones en la base de datos.
Servicio	Clases MuebleServicio, VarianteServicio, CotizacionServicio	Contiene la lógica de negocio, validaciones y operaciones CRUD.
Controlador	Endpoints REST /api/muebles, /api/variantes, /api/cotizaciones	Expone las funcionalidades del sistema mediante HTTP.

Patrones de diseño utilizados:

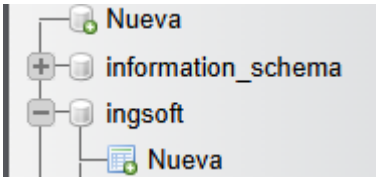
Patrón	Aplicación en el proyecto	Beneficio
MVC (Modelo-Vista-Controlador)	Separación de lógica de negocio , datos y endpoints REST.	Mejora mantenibilidad y escalabilidad
Service Layer Pattern	Clases “Servicio intermedias entre controladores y repositorios	Centraliza la lógica de negocio, facilitando el testing.
Singleton	Los servicios y repositorios son manejados como beans únicos por Spring	Control eficiente de instancias

Configuración de la base de datos

Al iniciar XAMPP



solo se deberá crear una base de datos llamada “ingsoft” en phpmyadmin para que funcione de manera local



Ya pasando con el código en el apartado de applicatio.properties vendrá con lo siguiente

src > main > resources > application.properties

```
spring.application.name=BaseDeDatos

server.port=8085

#Db
spring.datasource.url=jdbc:mysql://localhost:3306/ingsoft
spring.datasource.username=root
spring.datasource.password=

#JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true

```

Bueno pasando a la ejecución del código:

No será necesario CREAR LAS TABLAS ya que al ejecutar el código Main

```
Hibernate:
    create table cotizacion (
        id bigint not null auto_increment,
        confirmada bit not null,
        primary key (id)
    ) engine=InnoDB
Hibernate:
    create table item_cotizacion (
        id bigint not null auto_increment,
        cantidad integer not null,
        precio_total float(53) not null,
        cotizacion_id bigint,
        mueble_id bigint,
        variante_id bigint,
        primary key (id)
    ) engine=InnoDB
Hibernate:
    create table mueble (
        id bigint not null auto_increment,
        estado varchar(255),
        material varchar(255),
        nombre varchar(255),
        precio_base float(53),
        stock integer,
        tamano varchar(255),
        tipo varchar(255),
        primary key (id)
    ) engine=InnoDB
```

etc. (y muchas mas tablas)

hibernate se encargará de crear esta tabla con todos los atributos correspondientes con estos deberíamos verlos en la base de datos de phpMyAdmin

MODELO

Entidades

Mueble :Representa un producto del catálogo. Contiene atributos como nombre, tipo, material, tamaño, precio base, stock y estado.

Variante: Describe modificaciones o complementos del mueble que alteran su precio base (por ejemplo, “Barniz Premium”).

Cotización: Agrupa uno o más ítems que representan una posible venta. Contiene un estado confirmado para indicar si se ha concretado.

ItemCotización: Une un mueble con una variante y cantidad específica dentro de una cotización. Permite calcular subtotales y precios finales.

Controladores

Los controladores exponen la funcionalidad del sistema mediante endpoints REST bajo el prefijo /api/

MuebleControlador: /api/muebles Permite crear, listar, actualizar y eliminar muebles.

VarianteControlador: /api/variantes Gestiona variantes asociadas a muebles.

CotizacionControlador: /api/cotizaciones Crea y confirma cotizaciones, controlando precios y stock.

Servicios

Los servicios contienen la lógica de negocio del sistema y actúan como intermediarios entre los controladores y repositorios es un componente de Spring anotado con @Service y utiliza la inyección de dependencias (@Autowired):

MuebleServicio: Maneja el CRUD del catálogo, actualiza o desactiva muebles.

CotizacionServicio: Gestiona cotizaciones y ventas: crea cotizaciones, calcula precios totales, válida stock y confirma ventas.

Repositorios

Cada entidad cuenta con su propio repositorio que extiende JpaRepository, lo que permite ejecutar operaciones CRUD sin escribir SQL manualmente.

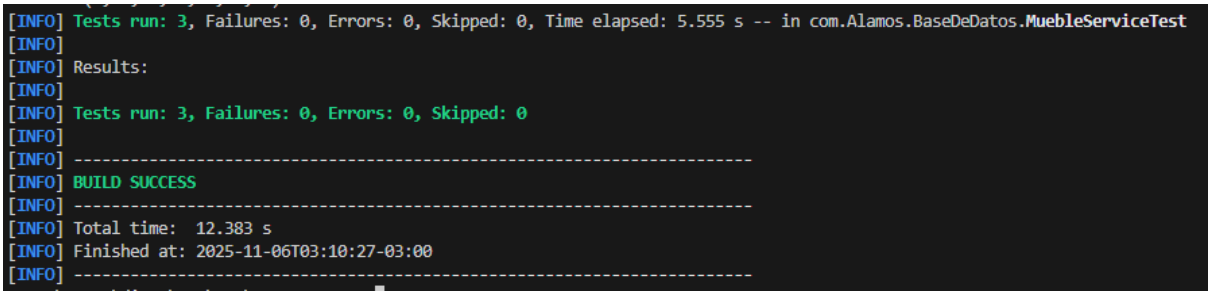
Esto permite Simplificación del acceso a datos e integración directa con Spring Data JPA

Testing

Se puede realizar pruebas mediante el Comando **.\mvnw.cmd test** o tambien dandole al boton que se encuentra al lado de los testing

Caso de prueba	Descripción	Resultado esperado
testCrearYActualizarMueble()	Verificar el correcto funcionamiento del CRUD de muebles	El mueble se crea con ID válido y us nombre actualizado coincide con el valor modificado
testVentaConStockInsuficiente()	Validar el control de stock al intentar confirmar una venta	Se lanza una excepción con el mensaje “Stock insuficiente” y la venta no se realiza
testCalcculoPrecioConVariante()	Probar el cálculo del precio total de una cotización considerando variantes o cantidades	El total acumulado coincide con el esperado, demostrando el correcto funcionamiento del servicio de precios

Salida que se debería conseguir



Apartado de POSTMAN (PRIMERO ARRANCAR PROGRAMA)

Se realizaron pruebas de integración utlizandooo la herramiento POSTMAN (ya que es un herramiento que utilizamos en el proyecto) con el objetivo de verificar el funcionamiento completo del sistema a través de sus endpoints REST. (Hacer zoom)

PRUEBAS:

1) **POST http://localhost:8085/api/muebles**

json body: {

"nombre": "Mesa de Roble",

"tipo": "Mesa",

"precioBase": 25000,

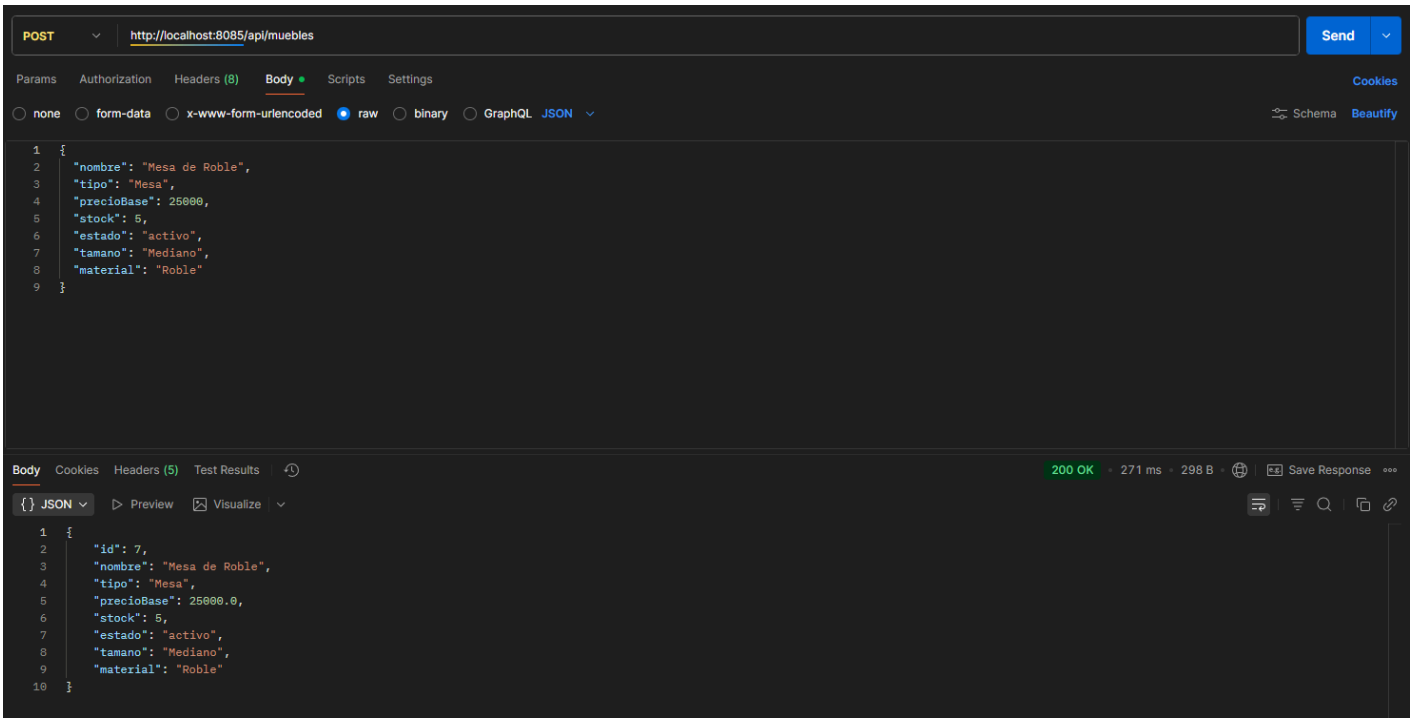
"stock": 5,

"estado": "activo",

"tamano": "Mediano",

"material": "Roble"

}



2) **POST http://localhost:8085/api/variantes**

Json body :

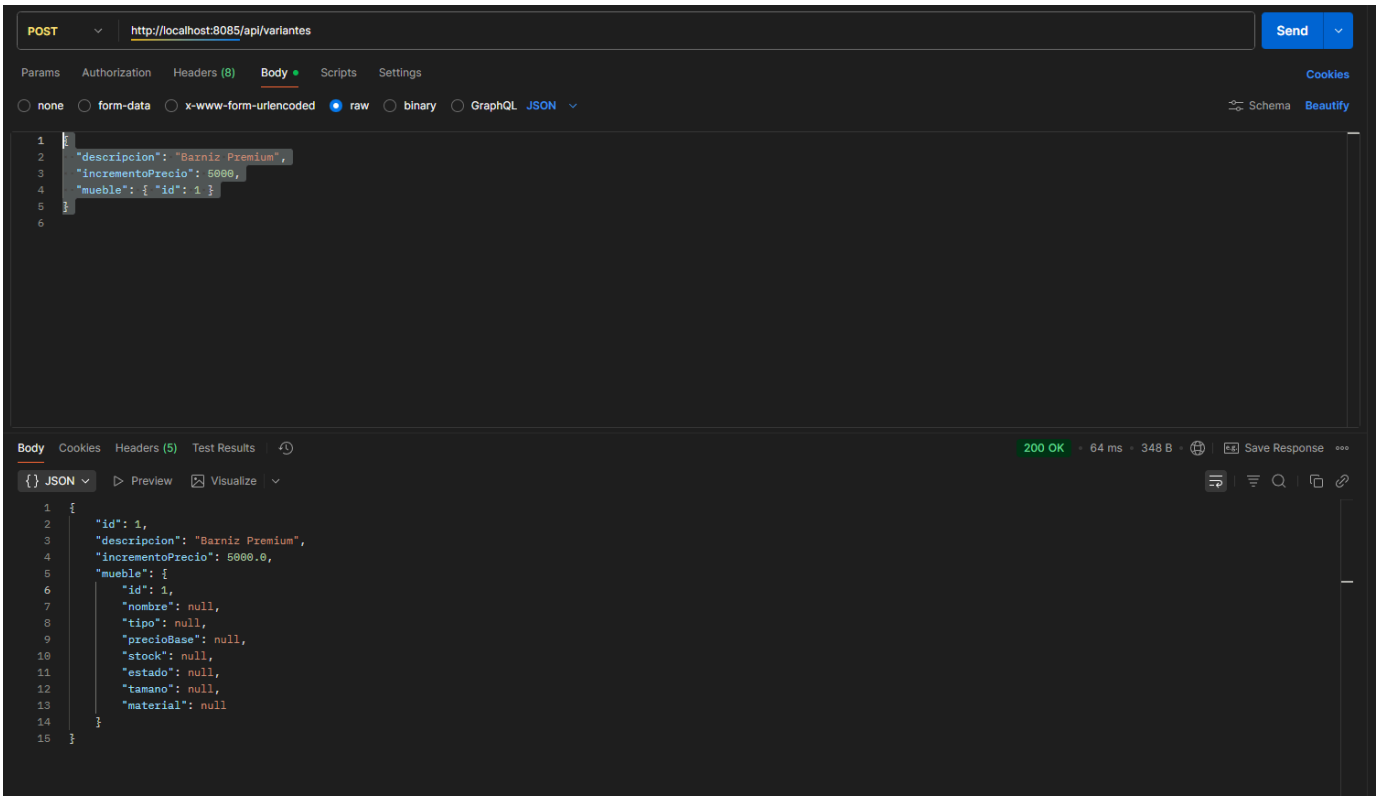
{

"descripcion": "Barniz Premium",

"incrementoPrecio": 5000,

"mueble": { "id": 1 }

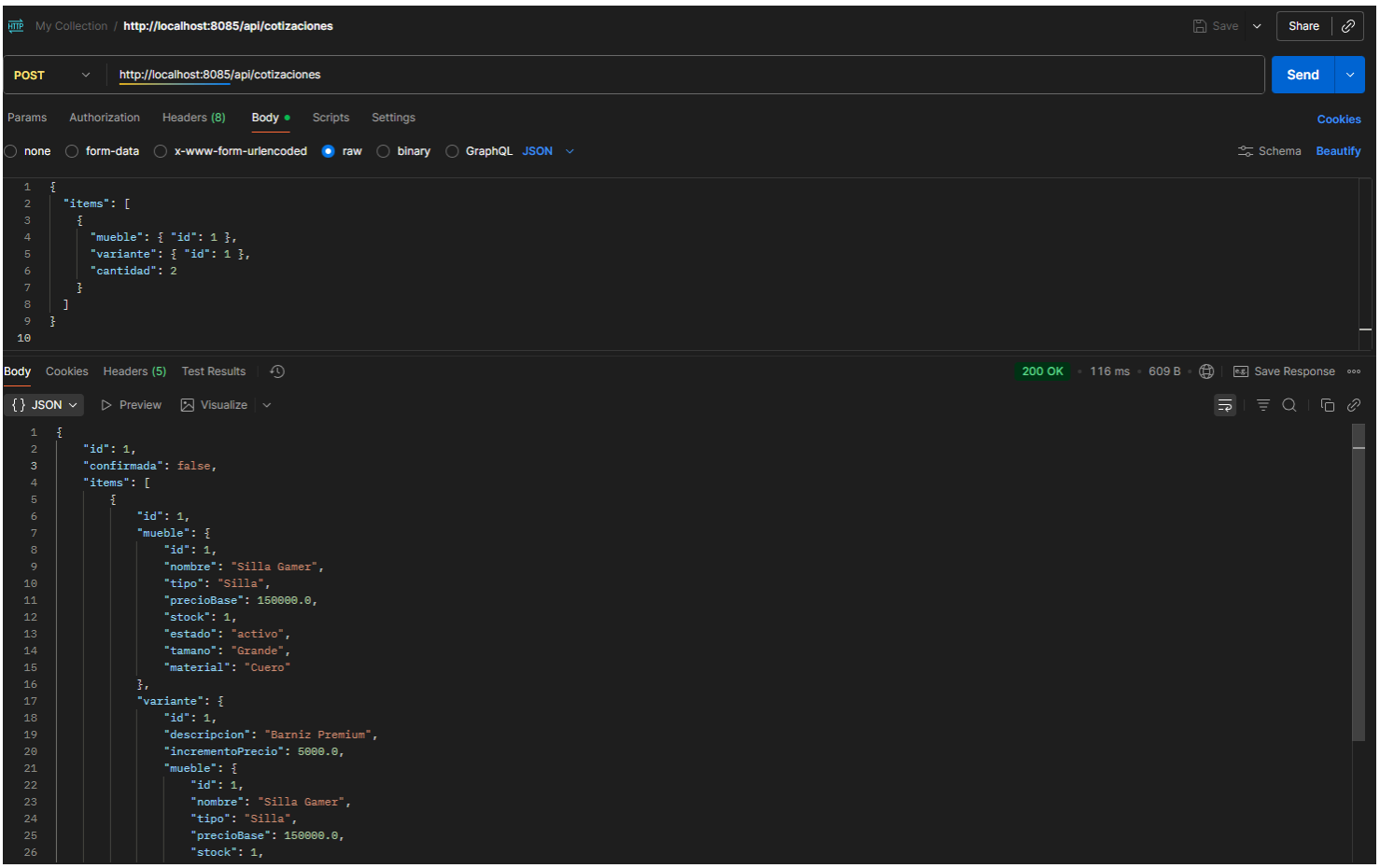
}



3) **POST http://localhost:8085/api/cotizaciones**

Json body:

```
{
  "items": [
    {
      "mueble": { "id": 1 },
      "variante": { "id": 1 },
      "cantidad": 2
    }
  ]
}
```



4) **PUT http://localhost:8085/api/cotizaciones/1/confirmar**

(Inserte este PUT para demostrar “Stock insuficiente”)

Este sale en “message”: