TP Integrador Final

API REST Para la gestión de ventas en un Bazar

🎯OBJETIVO

El objetivo de este proyecto integrador final es el de validar los **conocimientos prácticos y técnicos** referidos al desarrollo de APIs en el lenguaje de programación **Java** mediante **Spring Boot** para el curso *"Desarrollo de APIs en Java con Spring Boot"* de la TodoCode Academy.

ESCENARIO

Un bazar ha incrementado en gran medida sus ventas. Dado esto y que le está siendo casi imposible registrar las mismas y manejar el stock de sus productos de forma manual, necesita del desarrollo de una aplicación que le permita realizar esta tarea.
La dueña del bazar manifiesta que todas las operaciones que tenga la aplicación se deben poder realizar mediante dos tipos de clientes HTTP distintos:

- Una aplicación web, cuyo Frontend desarrollara un programador que se dedique a este ámbito.
- Una aplicación Mobile que será implementada a futuro.

Cada una de estas app representa a los dispositivos que ella y sus empleados manejan actualmente. En síntesis: una computadora y varios celulares.

Dada esta situación particular y de que necesita utilizar el **mismo backend** para ambas opciones, solicita el **desarrollo de una API**.

MODELADO

A partir del relevamiento que ha llevado a cabo un analista funcional, se detectaron que serán necesarias las siguientes clases:

- Producto
- Venta
- Cliente

En donde cada venta posee una lista de productos y uno y solo un cliente asociado. Además de eso, cada clase debe tener los siguientes atributos:

PRODUCTO

- Long codigo_producto
- String nombre
- String marca
- Double costo
- Double cantidad_disponible

VENTA

- Long codigo_venta
- LocalDate fecha_venta
- Double total
- List<Producto> listaProductos
- Cliente unCliente

CLIENTE

- Long id_cliente
- String nombre
- String apellido
- String dni

REQUERIMIENTOS

A partir del relevamiento realizado al modelado, la dueña del bazar específico que tiene los siguientes requerimientos:

1. **Poder realizar un CRUD completo de productos**
    A. **Metodos HTTP: GET, POST, DELETE, PUT**
    B. **Endpoints:**

- **Creación:** localhost:8080/productos/crear
- **Lista completa de productos:** localhost:8080/productos
- **Traer un producto en particular:** localhost:8080/productos/{codigo_producto}
- **Eliminación:** localhost:8080/productos/eliminar/{codigo_producto}
- **Edición:** localhost:8080/productos/editar/{codigo_producto}

2. Poder realizar un CRUD completo de Clientes

A. **Metodos HTTP:** GET, POST, DELETE, PUT
B. **Endpoints:**
   - **Creación:** localhost:8080/clientes/crear
   - **Lista completa de clientes:** localhost:8080/clientes
   - **Traer un cliente en particular:** localhost:8080/clientes/{id_cliente}
   - **Eliminación:** localhost:8080/clientes/eliminar/{id_cliente}
   - **Edición:** localhost:8080/clientes/editar/{id_cliente}

3. Poder realizar un CRUD completo de Ventas

A. **Metodos HTTP:** GET, POST, DELETE, PUT
B. **Endpoints:**
   - **Creación:** localhost:8080/ventas/crear
   - **Lista completa de Ventas realizadas:** localhost:8080/ventas
   - **Traer una Venta en particular:** localhost:8080/ventas/{codigo_venta}
   - **Eliminación:** localhost:8080/ventas/eliminar/{codigo_venta}
   - **Edición:** localhost:8080/ventas/editar/{codigo_venta}

NOTA: *No es necesario para este requerimiento actualizar el stock de un producto (descontar) al realizar una venta, ni tampoco controlar si cuenta con la cantidad disponible para vender: sin embargo, se considerará como "PLUS" o extra si se desea implementar esta funcionalidad.*

4. **Obtener todos los productos cuya cantidad disponible sea menor a 5**
    A. **Método HTTP:** GET
    B. **Endpoint**: localhost:8080/productos/falta-stock

5. **Obtener la lista de productos de una determinada venta**
    A. **Método HTTP:** GET
    B. **Endpoint**: localhost:8080/productos/{codigo_venta}

6. **Obtener la sumatoria del monto y también cantidad total de ventas de un determinado día**
    A. **Método HTTP:** GET
    B. **Endpoint**: localhost:8080/ventas/{fecha-venta}

7. **Obtener el codigo de venta, el total, la cantidad de productos, el nombre del cliente y el apellido del cliente de la venta con el monto mas alto de todas**
    A. **Método HTTP:** GET
    B. **Endpoint**: localhost:8080/ventas/mayor-venta
    Tener en cuenta **PATRO DTO** en este caso

8. **BONUS (OPCIONAL)**
    A. Se considera bonus cualquier propuesta de end-point, agregado de clase, etc que se proponga o implemente.
    B. Este apartado es opcional y pretende dejar volar la creatividad a la hora de proponer que otras necesidades/requerimientos podrían existir en este escenario.
    C. En caso de llevar a cabo este punto, especificar en un documento el/los nuevo/s requerimientos planteados y sus correspondientes especificaciones tecnicas (metodo HTTP, end-point, etc)

FORMATO DE ENTREGA

Se recomienda plasmar el proyecto final mediante un repositorio en GitHub simulando una entrega. Cada participante del curso creará su repositorio remoto y subirá allí su proyecto.

Es importante incluir TODOS LOS ARCHIVOS del proyecto, para asegurar la correcta ejecución del mismo.

Al mismo tiempo, incluir la colección de Postman utilizada para realizar las pruebas (esto puede incluirse en un link de descarga en el README de GitHub o en un archivo adjunto dentro del proyecto, como se prefiera).