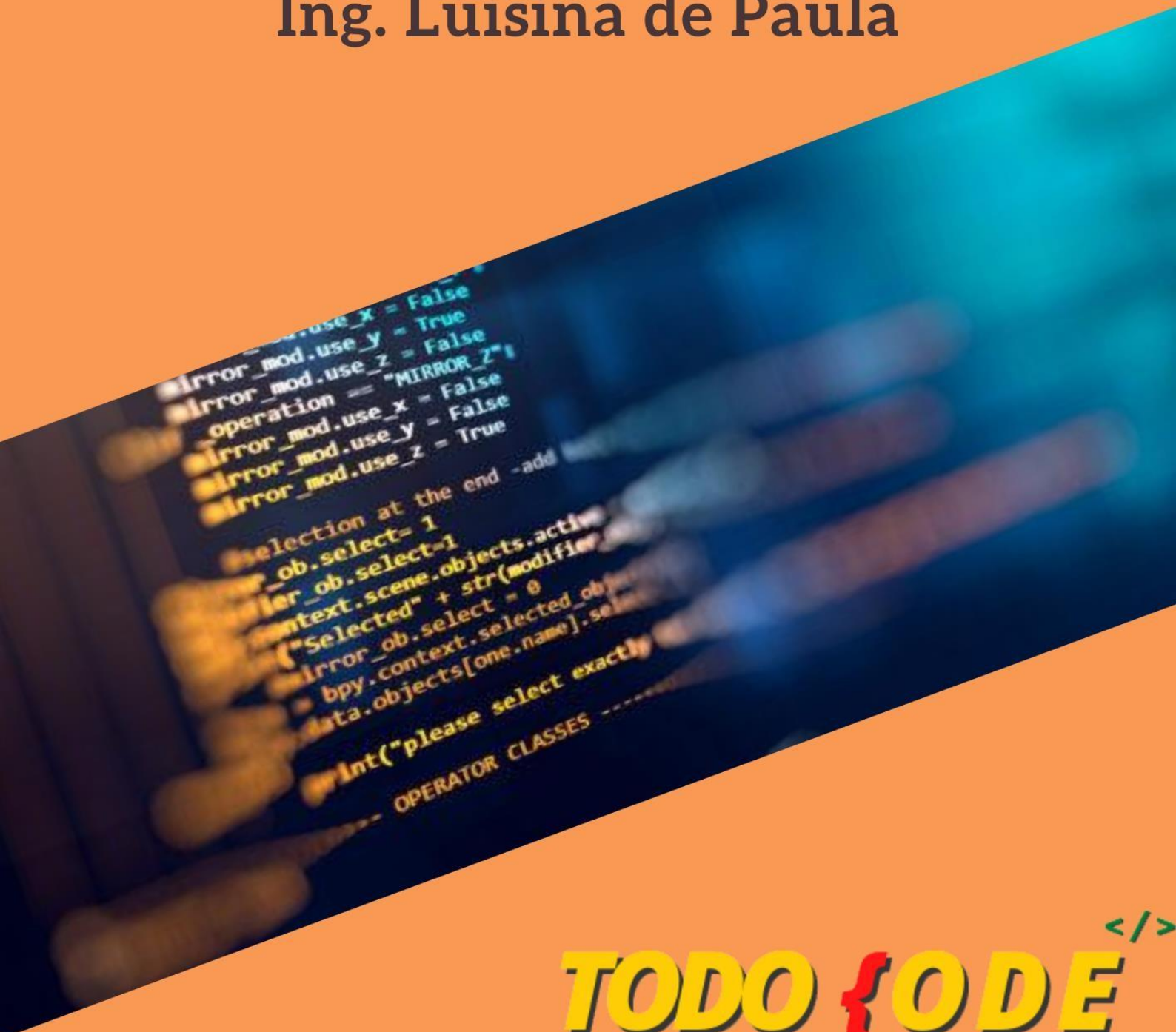
Desarrollo de APIs en Java con Spring Boot

TP Integrador Final

Ing. Luisina de Paula



TODO {ODE </>

www.todocodeacademy.com

TP Integrador Final

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 desarrollará un programador amigo (no será parte de nuestra tarea como desarrolladores backend).
- 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 respecto al modelado, la dueña del bazar especificó que tiene los siguientes requerimientos:

1. **Poder realizar un CRUD completo de productos**
   a. **Métodos 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. **Métodos 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. **Métodos 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/clientes/eliminar/{codigo_venta}

      Edición: localhost:8080/clientes/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 (para el bonus del punto 8) si se desea implementar la funcionalidad.*

4. **Obtener todos los productos cuya cantidad_disponible sea menor a 5**
   a. **Métodos HTTP:** GET
   b. **Endpoint:**

      localhost:8080/productos/falta_stock

5. **Obtener la lista de productos de una determinada venta**
   a. **Métodos HTTP:** GET
   b. **Endpoint:**

      localhost:8080/ventas/productos/{codigo_venta}

6. **Obtener la sumatoria del monto y también cantidad total de ventas de un determinado día**
   a. **Métodos HTTP:** GET
   b. **Endpoint:**
      localhost:8080/ventas/{fecha_venta}

7. **Obtener el codigo_venta, el total, la cantidad de productos, el nombre del cliente y el apellido del cliente de la venta con el monto más alto de todas.**
   a. **Métodos HTTP:** GET
   b. **Endpoint:**
      localhost:8080/ventas/mayor_venta
      *Tener en cuenta **patrón DTO** para este escenario*

8. **BONUS (OPCIONAL)**
   a. Se considera bonus cualquier propuesta de end-point, mejora, agregado de clase, etc que se proponga e implemente.
   b. Este apartado es opcional y pretende dejar volar la creatividad a la hora de proponer qué 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 técnicas (método HTTP, endpoint, etc). Tomar como ejemplo de plantilla para la especificación a las consignas de este enunciado.

Formato de Entrega

Se recomienda plasmar el proyecto Final mediante un repositorio de 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 dentro del README de Github o en un archivo adjunto dentro del proyecto, como se prefiera).

Recordar que la entrega de este TP Final es una SIMULACIÓN de cómo es posible presentar el código de un proyecto desarrollado mediante el uso de repositorios. No habrá un docente o especialista que valide la entrega. Simplemente se sugiere la misma como una forma de poner en práctica lo aprendido durante el curso.