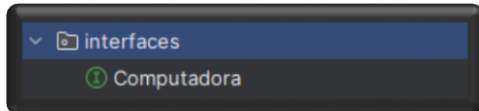


Proyecto Inyección de Dependencias

Este proyecto simula una tienda de computadoras que vende distintas marcas: Asus, Dell y MSI. Se implementan conceptos de polimorfismo e inyección de dependencias usando Spring.

1. Interfaces:

Computadora.java (Principal)

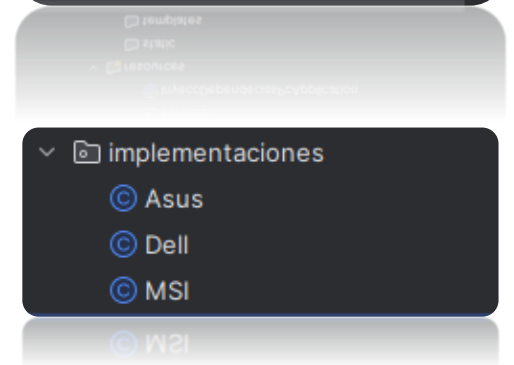
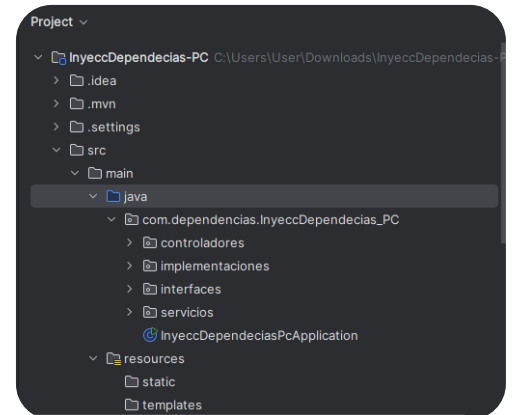


2. Interfaz Computadora

- Definimos los métodos que todas las computadoras deben implementar: **ensamblar ()** y **precio ()**.
- A su vez Permite el polimorfismo,

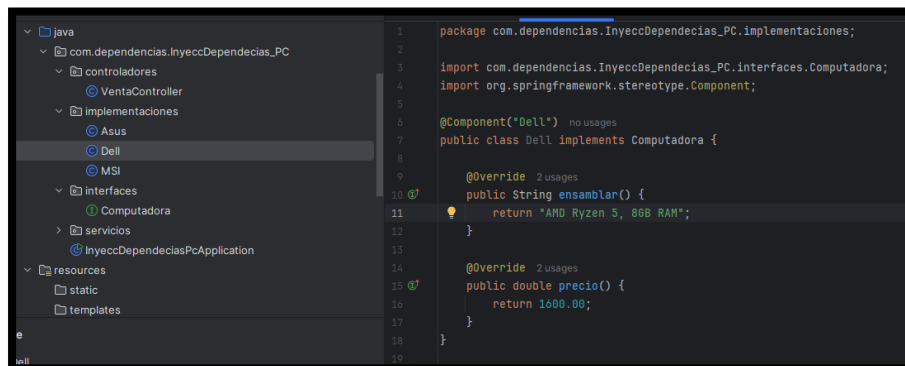
```
1 package com.dependencias.InyeccionDependencias_PC.interfaces;
2
3 public interface Computadora {
4     String ensamblar();
5     double precio();
6 }
7
```

Estructura del Proyecto

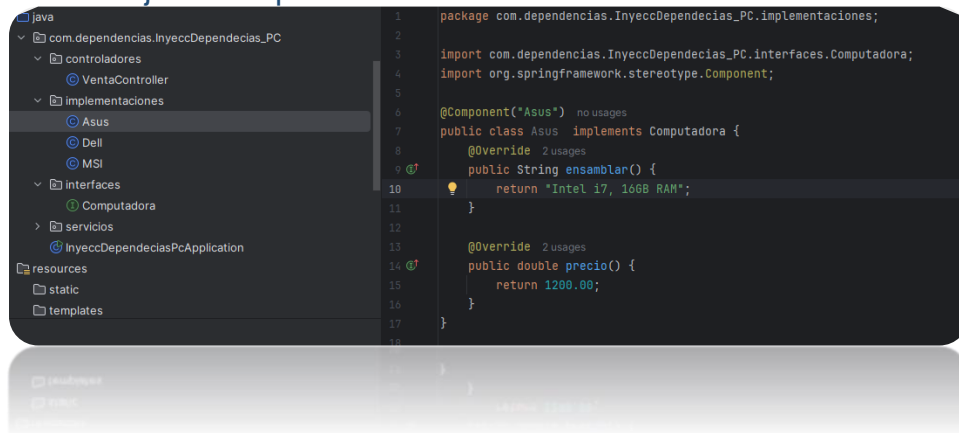


3. Implementaciones:

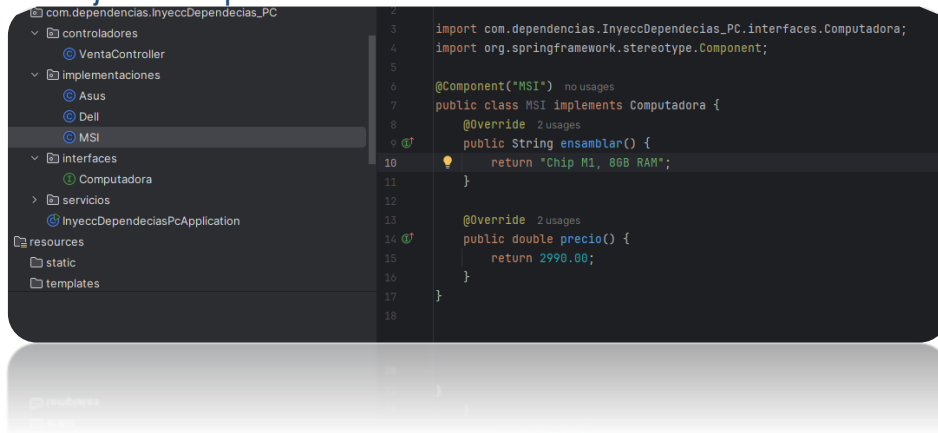
3.1 Dell.java Implementación concreta de Dell



3.2 ASUS.java = Implementación concreta de Asus



3.3 MSI.java = Implementación concreta de MSI



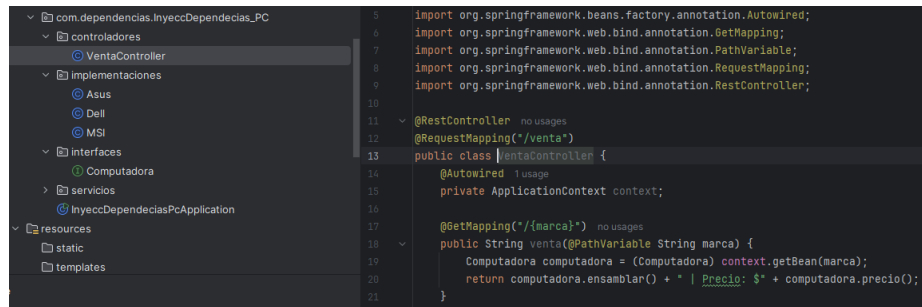
Cada clase representa una marca de computadora y define su comportamiento específico donde cada implementación sobrescribe los métodos de Computadora con su propia lógica.

- Asus → Intel i7, 16GB RAM, \$1200
- Dell → AMD Ryzen 5, 8GB RAM, \$1600
- MSI → Chip M1, 8GB RAM, \$2990

Aquí entra el Polimorfismo: El controlador puede llamar a ensamblar () o precio () y obtendrá el comportamiento correcto según la clase concreta.

4. Controladores

VentaController.java = Controlador REST para procesar ventas



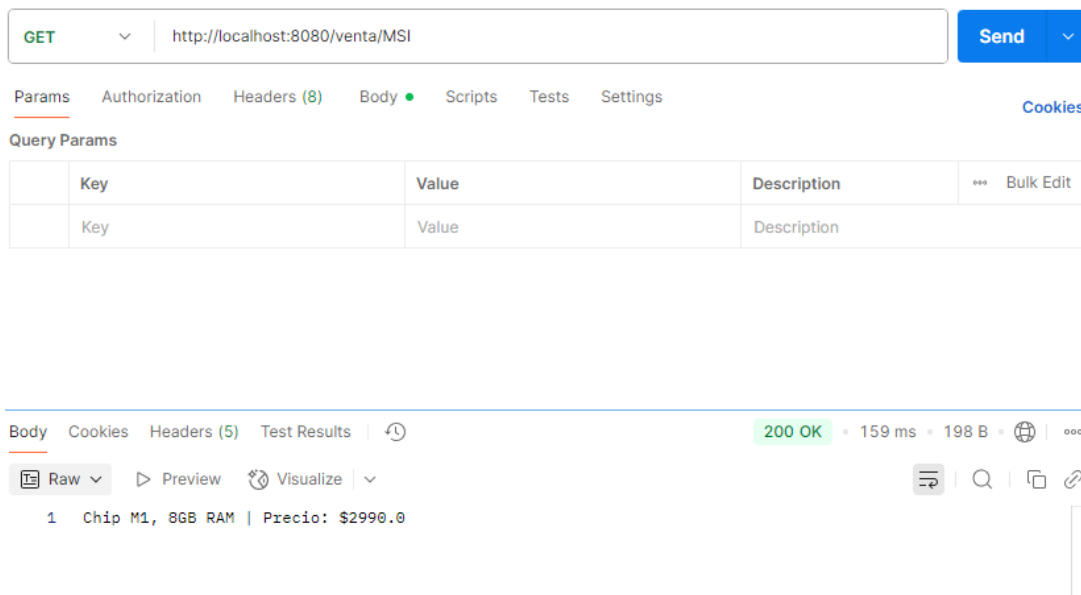
```
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.web.bind.annotation.GetMapping;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10
11 @RestController
12 @RequestMapping("/venta")
13 public class VentaController {
14     @Autowired
15     private ApplicationContext context;
16
17     @GetMapping("/{marca}")
18     public String venta(@PathVariable String marca) {
19         Computadora computadora = (Computadora) context.getBean(marca);
20         return computadora.ensamblar() + " | Precio: $" + computadora.precio();
21     }
22 }
```

En el controlador de ventas gestiono la venta de computadoras según la marca que me indiquen. Para esto, recibo un `Map<String, Computadora>` con todas las computadoras disponibles.

En cuanto a la inyección de dependencias:

- Manual: Creo el Map con las instancias y se lo paso directamente desde la clase Main Manual.
- Automática (Spring): Spring se encarga de crear los beans e inyectarlos en el constructor del controlador automáticamente.

Prueba con Postman



GET http://localhost:8080/venta/MSI Send

Params Authorization Headers (8) Body • Scripts Tests Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 159 ms • 198 B

Raw Preview Visualize

```
1 Chip M1, 8GB RAM | Precio: $2990.0
```

Diagrama de Flujo

Diagrama de Flujo: Proceso de Venta

Visualización del flujo de trabajo desde la solicitud del cliente hasta la finalización de la venta.

