Pruebas Unitarias con JUnit

1. Introducción

Este documento explica cómo se implementaron las pruebas unitarias en el proyecto de Venta de Computadoras desarrollado con Spring Boot. Se validan conceptos como polimorfismo, inyección de dependencias y el correcto funcionamiento del controlador REST.

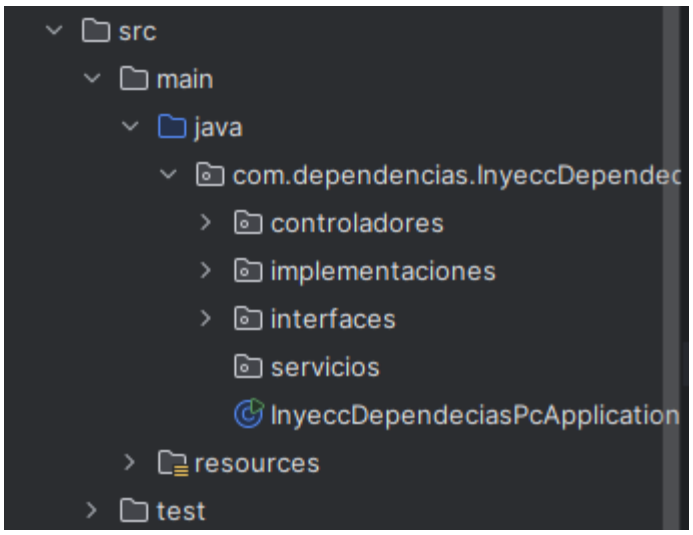
2. Dependencias necesarias

Para poder ejecutar las pruebas unitarias, se agregó al archivo pom.xml la dependencia de Spring Boot Starter Test:

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

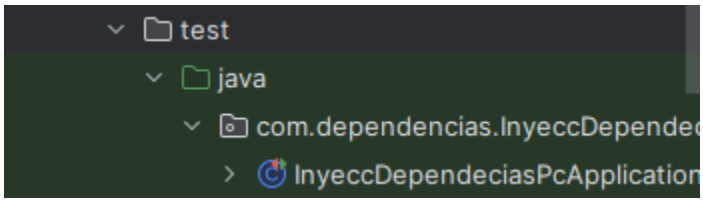
3. Estructura del proyecto con pruebas

La estructura del proyecto con la carpeta de pruebas queda de la siguiente manera:



4. Explicación de las pruebas

Se creó la clase de prueba `VentaControllerTest.java` dentro de la carpeta `src/test/java`. Esta clase utiliza la anotación `@SpringBootTest` para cargar el contexto completo de Spring Boot y `@AutoConfigureMockMvc` para simular llamadas HTTP al controlador.



4.1 Prueba de marca válida (Asus, Dell, MSI)

Se realizaron pruebas para las marcas disponibles (Asus, Dell y MSI). Cada prueba llama al endpoint `/venta/{marca}` y verifica que la respuesta contenga el nombre correcto de la marca.

```java
class VentaControllerTest {

    @Autowired  3 usages
    private MockMvc mockMvc;

    @Test
    void testVentaAsus() throws Exception {
        mockMvc.perform(get( uriTemplate: "/venta/Asus"))
                .andExpect(status().isOk())
                .andExpect(content().string(org.hamcrest.Matchers.containsString( substring: "Intel i7")));
    }

    @Test
    void testVentaDell() throws Exception {
        mockMvc.perform(get( uriTemplate: "/venta/Dell"))
                .andExpect(status().isOk())
                .andExpect(content().string(org.hamcrest.Matchers.containsString( substring: "AMD Ryzen 5, 8GB RAM")));
    }

    @Test
    void testVentaMsi() throws Exception {
        mockMvc.perform(get( uriTemplate: "/venta/MSI"))
                .andExpect(status().isOk())
                .andExpect(content().string(org.hamcrest.Matchers.containsString( substring: "Chip M1, 8GB RAM")));
    }
}
```

5. Ejemplo de código de prueba

A continuación, se muestra un ejemplo de la clase de pruebas implementada: