

“Informe Experiencia Examen Transversal de Desarrollo Full Stack ”

Sección:001D

Docente: Viviana Poblete

Integrantes: Javier Albornoz

Vicente Hormazabal

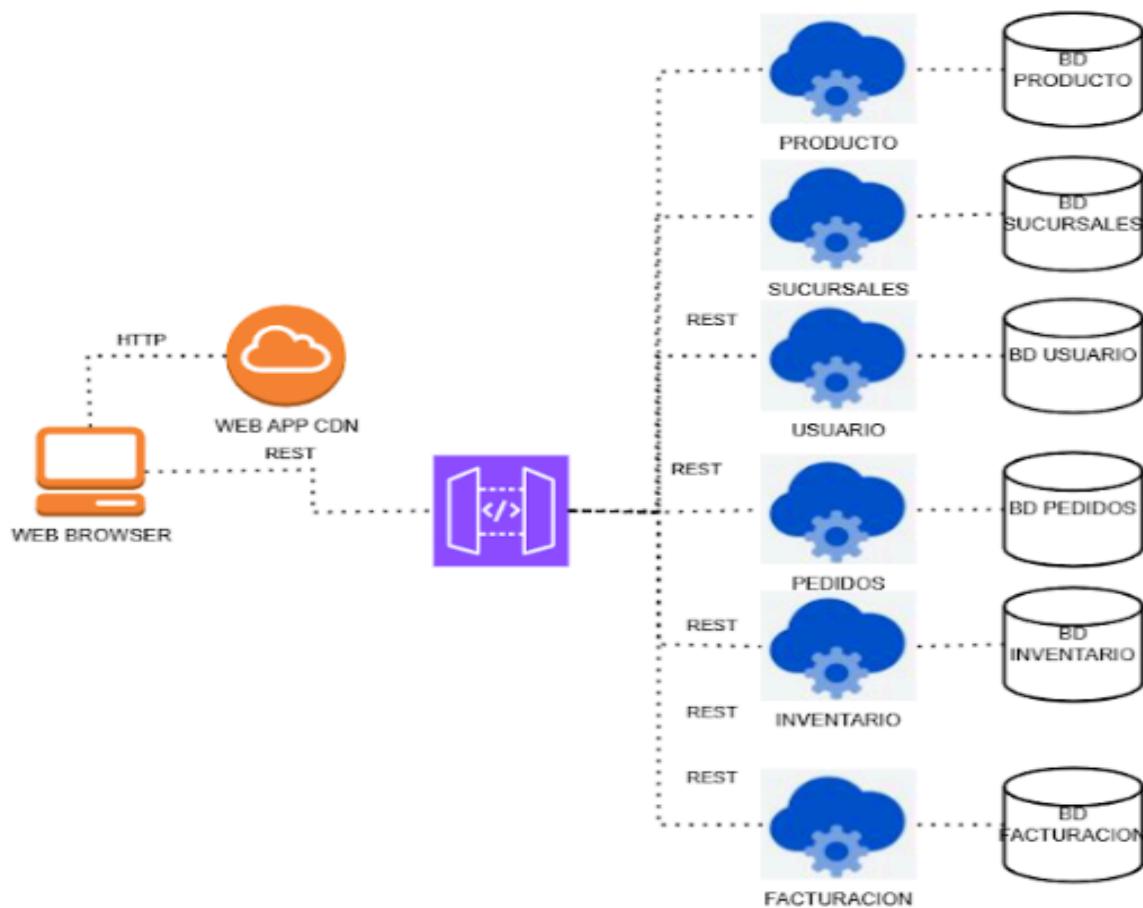
Diego Neira



Índice

“Informe Experiencia Examen Transversal de Desarrollo Full Stack ”.....	1
Índice.....	2
Diagrama arquitectura de microservicios.....	3
Plan de pruebas.....	4
1. JUnit 5 (Jupiter).....	25
2. Mockito.....	26
3. Spring Boot Test (spring-boot-starter-test).....	26
4. MockMvc.....	27
5. AssertJ / Assertions (de JUnit).....	27
❖ Pruebas Unitarias.....	27
❖ Pruebas de integración.....	31
Postman.....	37
OAS.....	40
GITHUB.....	51
Conclusión.....	52

Diagrama arquitectura de microservicios





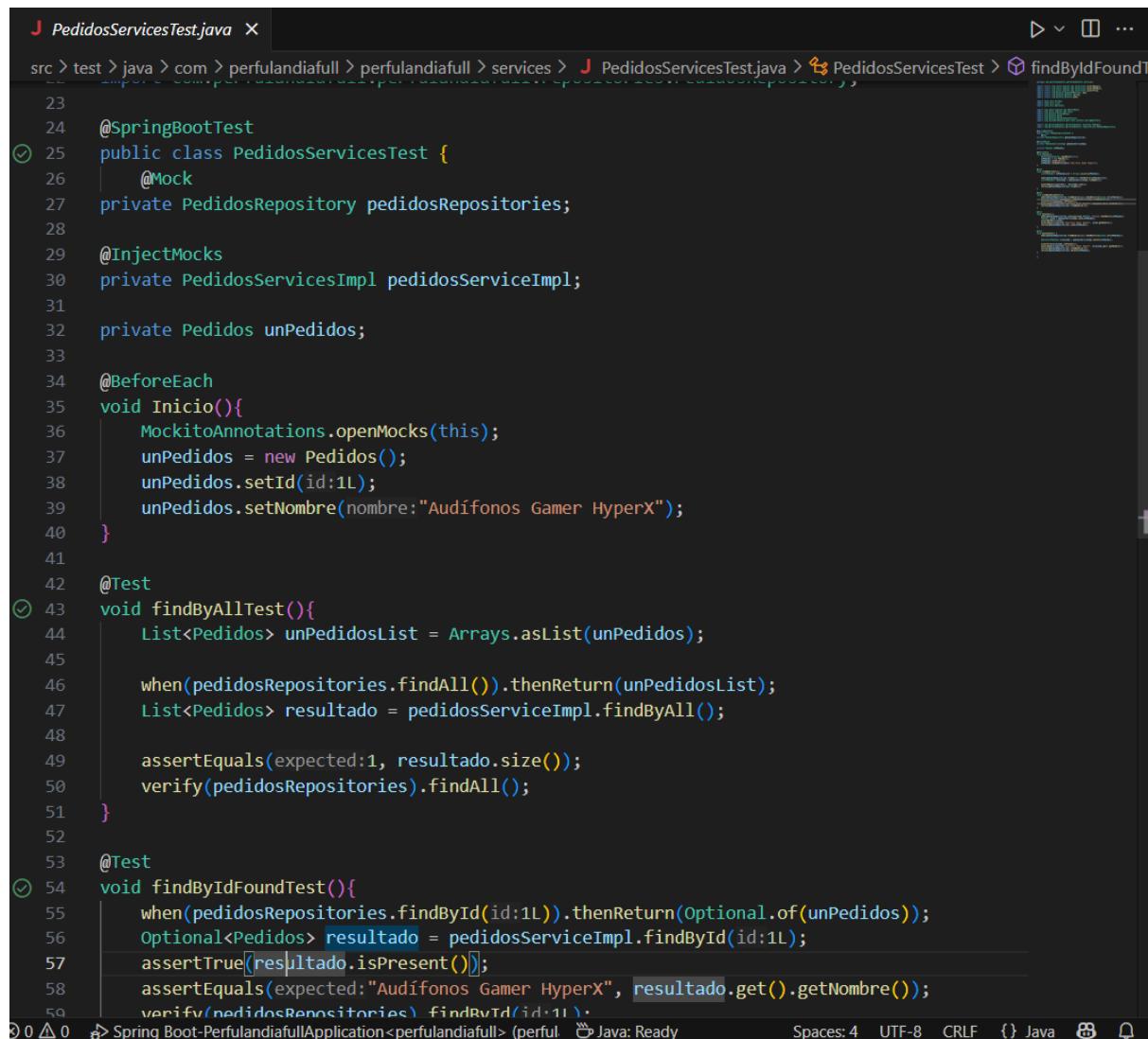
Plan de pruebas

Codificación de Prueba Unitaria a Services:

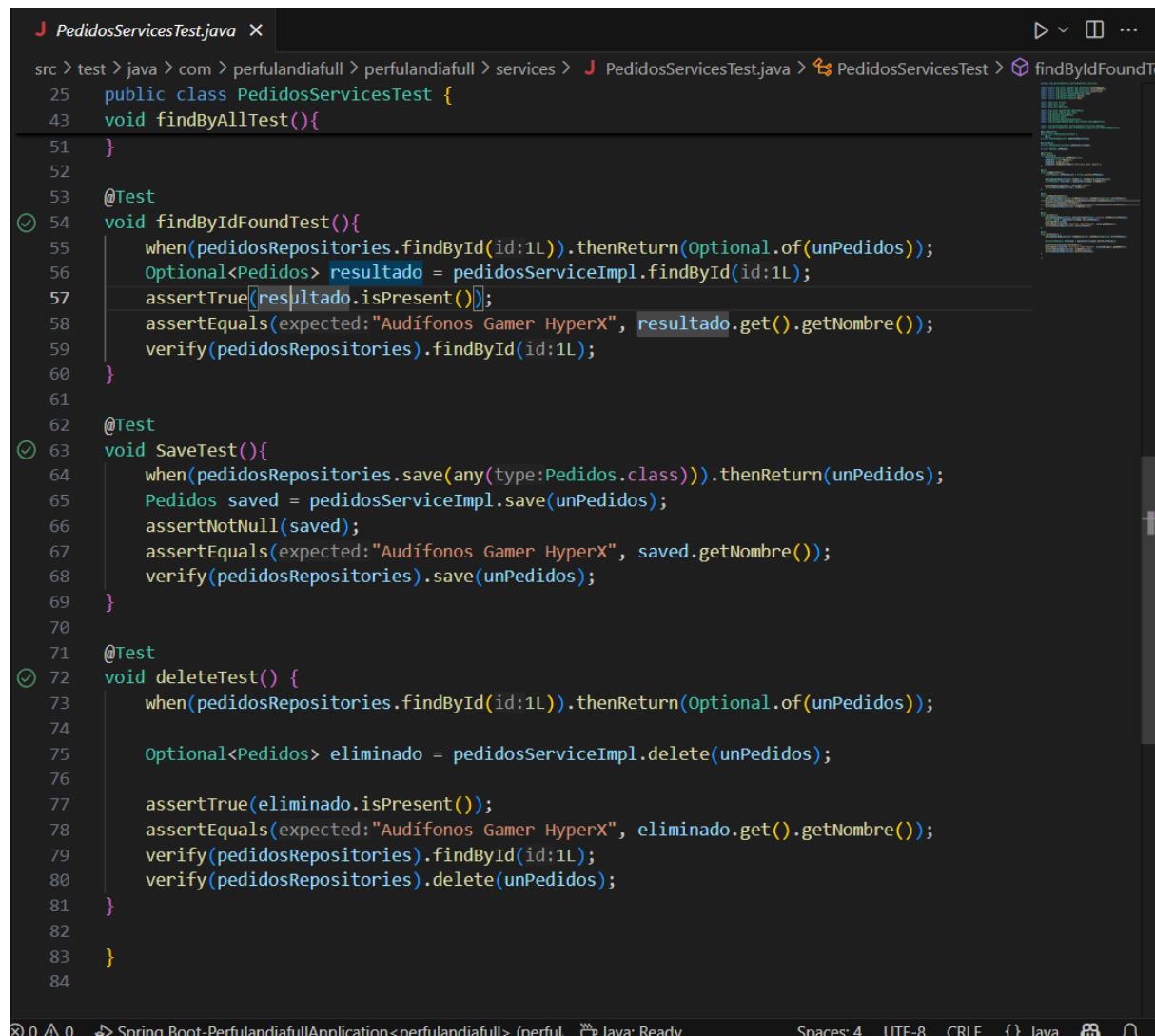
Testing a Pedidos:

The screenshot shows a Java code editor with a file named `PedidosServicesTest.java`. The code is a unit test for a service class named `PedidoServices`. It uses JUnit Jupiter and Mockito annotations. The code includes imports for assertions, lists, arrays, and specific classes from the application and repository layers. It defines a test class with a constructor that injects a `PedidosRepository` and a `PedidoServicesImpl`. A `@BeforeEach` annotation is used to initialize the repository and service objects.

```
J PedidosServicesTest.java x
src > test > java > com > perfulandiafull > perfulandiafull > services > J PedidosServicesTest.java > PedidosServicesTest > findByldFoundTo
1 package com.perfulandiafull.perfulandiafull.services;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertNotNull;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6 import static org.mockito.ArgumentMatchers.any;
7 import static org.mockito.Mockito.verify;
8 import static org.mockito.Mockito.when;
9
10 import java.util.Arrays;
11 import java.util.List;
12 import java.util.Optional;
13
14 import org.junit.jupiter.api.BeforeEach;
15 import org.junit.jupiter.api.Test;
16 import org.mockito.InjectMocks;
17 import org.mockito.Mock;
18 import org.mockito.MockitoAnnotations;
19 import org.springframework.boot.test.context.SpringBootTest;
20
21 import com.perfulandiafull.perfulandiafull.entities.Pedidos;
22 import com.perfulandiafull.perfulandiafull.repositories.PedidosRepository;
23
24 @SpringBootTest
25 public class PedidosServicesTest {
26     @Mock
27     private PedidosRepository pedidosRepositories;
28
29     @InjectMocks
30     private PedidoServicesImpl pedidoServiceImpl;
31
32     private Pedidos unPedidos;
33
34     @BeforeEach
35     void Inicio(){
36         MockitoAnnotations.openMocks(this);
37         unPedidos = new Pedidos();
38     }
39
40     @Test
41     void TestFindByldFoundTo()
42     {
43         // Given
44         Pedidos pedidos = new Pedidos();
45         pedidos.setLd("1");
46         pedidos.setCedula("123456789");
47         pedidos.setNombre("Juan");
48         pedidos.setApellido("Perez");
49         pedidos.setDireccion("Av. Ejercito");
50         pedidos.setCiudad("Santiago");
51         pedidos.setEstado("pendiente");
52         pedidos.setFecha(new Date());
53
54         Mockito.when(pedidosRepositories.findById("1"))
55             .thenReturn(Optional.of(pedidos));
56
57         // When
58         Pedidos resultado = pedidoServiceImpl.findByldFoundTo("1");
59
60         // Then
61         assertEquals("1", resultado.getLd());
62         assertEquals("123456789", resultado.getCedula());
63         assertEquals("Juan", resultado.getNombre());
64         assertEquals("Perez", resultado.getApellido());
65         assertEquals("Av. Ejercito", resultado.getDireccion());
66         assertEquals("Santiago", resultado.getCiudad());
67         assertEquals("pendiente", resultado.getEstado());
68         assertEquals(resultado.getFecha(), resultado.getFecha());
69     }
70 }
```

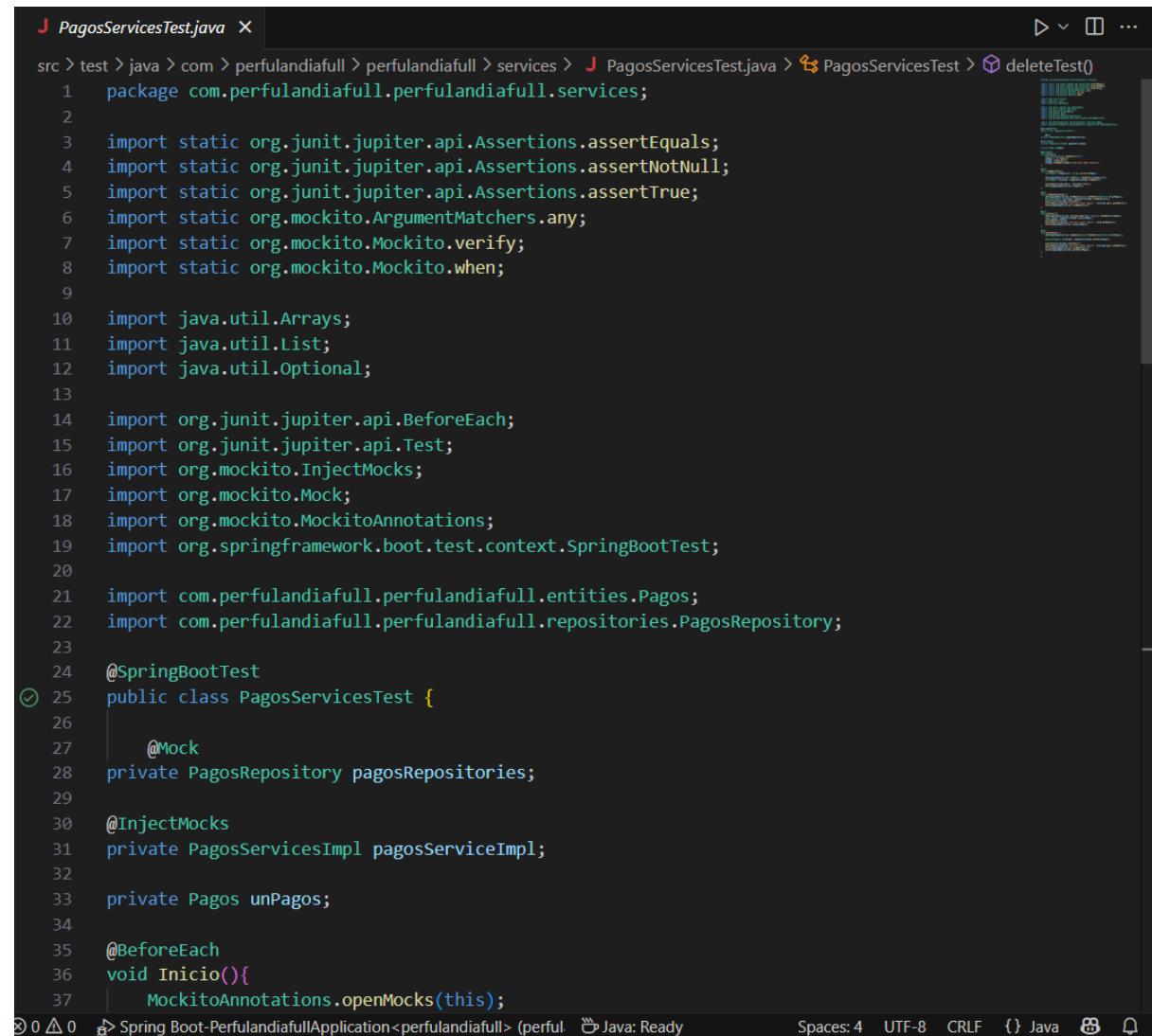


```
src > test > java > com > perfulandiafull > services > J PedidosServicesTest.java > ⚡ PedidosServicesTest > ⚡ findByIdFoundT
23
24  @SpringBootTest
25  public class PedidosServicesTest {
26      @Mock
27      private PedidosRepository pedidosRepositories;
28
29      @InjectMocks
30      private PedidosServicesImpl pedidosServiceImpl;
31
32      private Pedidos unPedidos;
33
34      @BeforeEach
35      void Inicio(){
36          MockitoAnnotations.openMocks(this);
37          unPedidos = new Pedidos();
38          unPedidos.setId(1L);
39          unPedidos.setNombre("Audífonos Gamer HyperX");
40      }
41
42      @Test
43      void findByAllTest(){
44          List<Pedidos> unPedidosList = Arrays.asList(unPedidos);
45
46          when(pedidosRepositories.findAll()).thenReturn(unPedidosList);
47          List<Pedidos> resultado = pedidosServiceImpl.findByAll();
48
49          assertEquals(expected:1, resultado.size());
50          verify(pedidosRepositories).findAll();
51      }
52
53      @Test
54      void findByIdFoundTest(){
55          when(pedidosRepositories.findById(id:1L)).thenReturn(Optional.of(unPedidos));
56          Optional<Pedidos> resultado = pedidosServiceImpl.findById(id:1L);
57          assertTrue(resultado.isPresent());
58          assertEquals(expected:"Audífonos Gamer HyperX", resultado.get().getNombre());
59          verify(pedidosRepositories).findById(id:1L);
}
0 △ 0 ⚡ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java ⚡
```



```
J PedidosServicesTest.java X
src > test > java > com > perfulandiafull > perfulandiafull > services > J PedidosServicesTest.java > PedidosServicesTest > findByIdFoundT
25 public class PedidosServicesTest {
43 void findByAllTest(){
51 }
52
53 @Test
54 void findByIdFoundTest(){
55     when(pedidosRepositories.findById(id:1L)).thenReturn(Optional.of(unPedidos));
56     Optional<Pedidos> resultado = pedidosServiceImpl.findById(id:1L);
57     assertTrue(resultado.isPresent());
58     assertEquals(expected:"Audífonos Gamer HyperX", resultado.get().getNombre());
59     verify(pedidosRepositories).findById(id:1L);
60 }
61
62 @Test
63 void SaveTest(){
64     when(pedidosRepositories.save(any(type:Pedidos.class))).thenReturn(unPedidos);
65     Pedidos saved = pedidosServiceImpl.save(unPedidos);
66     assertNotNull(saved);
67     assertEquals(expected:"Audífonos Gamer HyperX", saved.getNombre());
68     verify(pedidosRepositories).save(unPedidos);
69 }
70
71 @Test
72 void deleteTest() {
73     when(pedidosRepositories.findById(id:1L)).thenReturn(Optional.of(unPedidos));
74
75     Optional<Pedidos> eliminado = pedidosServiceImpl.delete(unPedidos);
76
77     assertTrue(eliminado.isPresent());
78     assertEquals(expected:"Audífonos Gamer HyperX", eliminado.get().getNombre());
79     verify(pedidosRepositories).findById(id:1L);
80     verify(pedidosRepositories).delete(unPedidos);
81 }
82
83 }
84 }
```

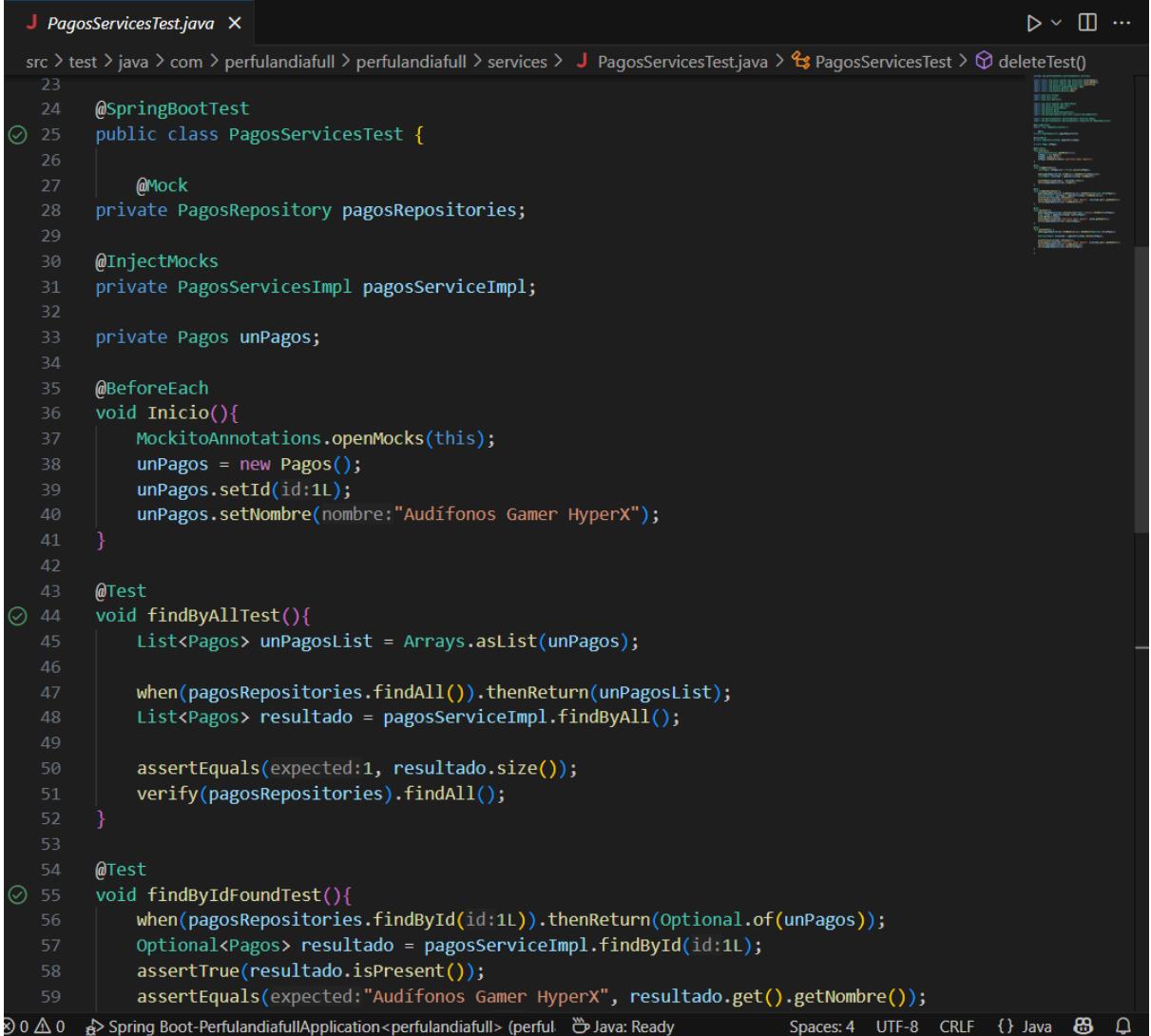
Testing a Pagos



```
J PagosServicesTest.java X
src > test > java > com > perfulandiafull > perfulandiafull > services > J PagosServicesTest.java > ⚡ PagosServicesTest > ⚡ deleteTest()

1 package com.perfulandiafull.perfulandiafull.services;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertNotNull;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6 import static org.mockito.ArgumentMatchers.any;
7 import static org.mockito.Mockito.verify;
8 import static org.mockito.Mockito.when;
9
10 import java.util.Arrays;
11 import java.util.List;
12 import java.util.Optional;
13
14 import org.junit.jupiter.api.BeforeEach;
15 import org.junit.jupiter.api.Test;
16 import org.mockito.InjectMocks;
17 import org.mockito.Mock;
18 import org.mockito.MockitoAnnotations;
19 import org.springframework.boot.test.context.SpringBootTest;
20
21 import com.perfulandiafull.perfulandiafull.entities.Pagos;
22 import com.perfulandiafull.perfulandiafull.repositories.PagosRepository;
23
24 @SpringBootTest
25 public class PagosServicesTest {
26
27     @Mock
28     private PagosRepository pagosRepositories;
29
30     @InjectMocks
31     private PagosServicesImpl pagosServiceImpl;
32
33     private Pagos unPagos;
34
35     @BeforeEach
36     void Inicio(){
37         MockitoAnnotations.openMocks(this);
38     }
39
40     @Test
41     void TestDeletePagos() {
42         when(pagosRepositories.findById(1L)).thenReturn(Optional.of(unPagos));
43         pagosServiceImpl.deletePagos(1L);
44         verify(pagosRepositories).deleteById(1L);
45     }
46
47 }
```

0 0 △ 0 ➔ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java ⌂ ⌂



The screenshot shows a Java code editor with the file `PagosServicesTest.java` open. The code is a Spring Boot test class for a payment service. It uses Mockito annotations to mock the `PagosRepository` and `PagosServiceImpl`. The class contains three test methods: `findByIdAllTest`, `findByIdFoundTest`, and `findByNameTest`. The `findByIdAllTest` method verifies that the `findAll` method returns a single item. The `findByIdFoundTest` method verifies that the `findById` method returns an optional containing the correct item. The `findByNameTest` method verifies that the `findByName` method returns the correct item based on its name.

```
src > test > java > com > perfulandiafull > perfulandiafull > services > J PagosServicesTest.java > ⚡ PagosServicesTest > ⚡ deleteTest()
23
24  @SpringBootTest
25  public class PagosServicesTest {
26
27      @Mock
28      private PagosRepository pagosRepositories;
29
30      @InjectMocks
31      private PagosServicesImpl pagosServiceImpl;
32
33      private Pagos unPagos;
34
35      @BeforeEach
36      void Inicio(){
37          MockitoAnnotations.openMocks(this);
38          unPagos = new Pagos();
39          unPagos.setId(id:1L);
40          unPagos.setNombre(nombre:"Audífonos Gamer HyperX");
41      }
42
43      @Test
44      void findByAllTest(){
45          List<Pagos> unPagosList = Arrays.asList(unPagos);
46
47          when(pagosRepositories.findAll()).thenReturn(unPagosList);
48          List<Pagos> resultado = pagosServiceImpl.findByAll();
49
50          assertEquals(expected:1, resultado.size());
51          verify(pagosRepositories).findAll();
52      }
53
54      @Test
55      void findByIdFoundTest(){
56          when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
57          Optional<Pagos> resultado = pagosServiceImpl.findById(id:1L);
58          assertTrue(resultado.isPresent());
59          assertEquals(expected:"Audífonos Gamer HyperX", resultado.get().getNombre());
50  Δ 0  ⌂ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perful  ⌂ Java: Ready   Spaces: 4   UTF-8   CRLF   {} Java   ⌂   ⌂
```



The screenshot shows a Java code editor with a dark theme. The file being edited is `PagosServicesTest.java`, located at `src > test > java > com > perfulandiafull > perfulandiafull > services >`. The code contains several test methods for a `PagosService` implementation:

```
src > test > java > com > perfulandiafull > perfulandiafull > services > J PagosServicesTest.java > ⚡ PagosServicesTest > ⚡ deleteTest()
25 public class PagosServicesTest {
26     void runByAllTest(){
27     }
28
29     @Test
30     void findByIdFoundTest(){
31         when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
32         Optional<Pagos> resultado = pagosServiceImpl.findById(id:1L);
33         assertTrue(resultado.isPresent());
34         assertEquals(expected:"Audífonos Gamer HyperX", resultado.get().getNombre());
35         verify(pagosRepositories).findById(id:1L);
36     }
37
38     @Test
39     void SaveTest(){
40         when(pagosRepositories.save(any(type:Pagos.class))).thenReturn(unPagos);
41         Pagos saved = pagosServiceImpl.save(unPagos);
42         assertNotNull(saved);
43         assertEquals(expected:"Audífonos Gamer HyperX", saved.getNombre());
44         verify(pagosRepositories).save(unPagos);
45     }
46
47     @Test
48     void deleteTest() {
49         when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
50
51         Optional<Pagos> eliminado = pagosServiceImpl.delete(unPagos);
52
53         assertTrue(eliminado.isPresent());
54         assertEquals(expected:"Audífonos Gamer HyperX", eliminado.get().getNombre());
55         verify(pagosRepositories).findById(id:1L);
56         verify(pagosRepositories).delete(unPagos);
57     }
58 }
```

The status bar at the bottom indicates the file is a Spring Boot application, Java is ready, and the code uses 4 spaces for indentation.



Testing a Inventoryo:

```
J InventarioServicesTest.java X
src > test > java > com > perfulandiafull > perfulandiafull > services > J InventarioServicesTest.java > {} com.perfulandiafull.perfulandiafull.servi
1 package com.perfulandiafull.perfulandiafull.services;
2
3 import static org.junit.jupiter.api.Assertions.assertEquals;
4 import static org.junit.jupiter.api.Assertions.assertNotNull;
5 import static org.junit.jupiter.api.Assertions.assertTrue;
6 import static org.mockito.ArgumentMatchers.any;
7 import static org.mockito.Mockito.verify;
8 import static org.mockito.Mockito.when;
9
10 import java.util.Arrays;
11 import java.util.List;
12 import java.util.Optional;
13
14 import org.junit.jupiter.api.BeforeEach;
15 import org.junit.jupiter.api.Test;
16 import org.mockito.InjectMocks;
17 import org.mockito.Mock;
18 import org.mockito.MockitoAnnotations;
19 import org.springframework.boot.test.context.SpringBootTest;
20
21 import com.perfulandiafull.perfulandiafull.entities.Inventario;
22 import com.perfulandiafull.perfulandiafull.repositories.InventarioRepository;
23
24
25 @SpringBootTest
26 public class InventarioServicesTest {
27
28     @Mock
29     private InventarioRepository inventarioRepositories;
30
31     @InjectMocks
32     private InventarioServicesImpl inventarioserviceImpl;
33
34     private Inventario unInventario;
35
36     @BeforeEach
37     void Inicio(){
```

0.0 △ 0 ➤ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java ⚙️ 🔔

The screenshot shows a code editor with the file `InventarioServicesTest.java` open. The code is a unit test for a service layer. It uses Mockito annotations to mock repository and service objects. The test cases verify the `findByIdAll` and `findByIdFound` methods.

```
src > test > java > com > perfulandiafull > perfulandiafull > services > InventarioServicesTest.java > {} com.perfulandiafull.perfulandiafull.services.InventarioServicesTest
```

```
25  @SpringBootTest
26  public class InventarioServicesTest {
27
28      @Mock
29      private InventarioRepository inventarioRepositories;
30
31      @InjectMocks
32      private InventarioServicesImpl inventarioServiceImpl;
33
34      private Inventario unInventario;
35
36      @BeforeEach
37      void Inicio(){
38          MockitoAnnotations.openMocks(this);
39          unInventario = new Inventario();
40          unInventario.setId(id:1L);
41          unInventario.setNombre(nombre:"Audífonos Gamer HyperX");
42      }
43
44      @Test
45      void findByAllTest(){
46
47
48          List <Inventario> unInventaro = Arrays.asList(unInventario);
49
50          when(inventarioRepositories.findAll()).thenReturn(unInventaro);
51          List <Inventario> resultado = inventarioServiceImpl.findByAll();
52
53          assertEquals(expected:1, resultado.size());
54          verify(inventarioRepositories).findAll();
55      }
56
57      @Test
58      void findByIdFoundTest(){
59          when(inventarioRepositories.findById(id:1L)).thenReturn(Optional.of(unInventario));
60          Optional<Inventario> resultado = inventarioServiceImpl.findById(id:1L);
61      }
62  }
```

0 0 △ 0 Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java

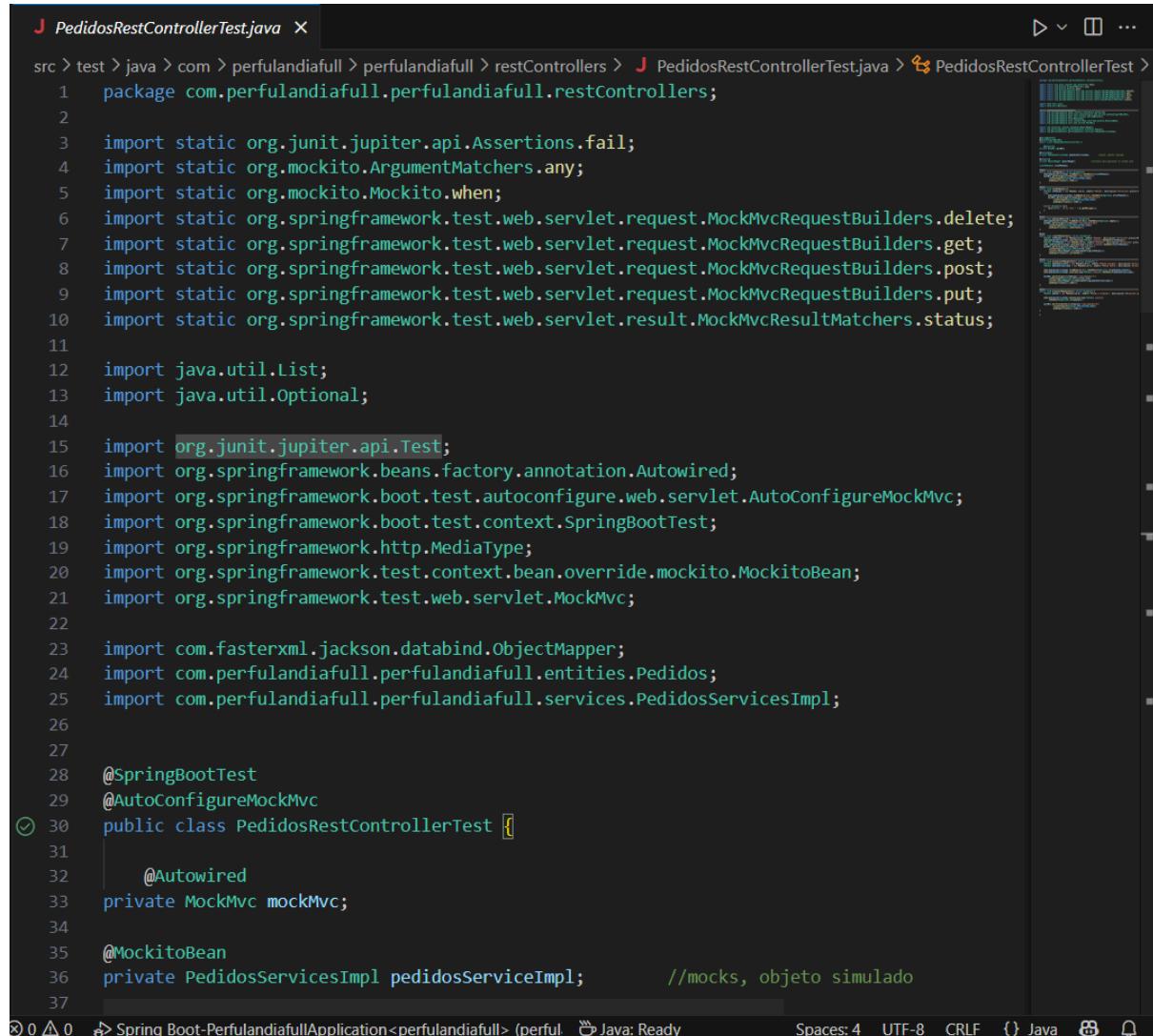
```
src > test > java > com > perfulandiafull > perfulandiafull > services > J InventarioServicesTest.java > {} com.perfulandiafull.perfulandiafull.services.InventarioServicesTest
```

```
26 public class InventarioServicesTest {  
27       
28     @Test  
29     void findByIdFoundTest(){  
30         when(inventarioRepositories.findById(id:1L)).thenReturn(Optional.of(unInventario));  
31         Optional<Inventario> resultado = inventarioServiceImpl.findById(id:1L);  
32         assertTrue(resultado.isPresent());  
33         assertEquals(expected:"Audífonos Gamer HyperX", resultado.get().getNombre());  
34         verify(inventarioRepositories).findById(id:1L);  
35     }  
36       
37     @Test  
38     void SaveTest(){  
39         when(inventarioRepositories.save(any(type:Inventario.class))).thenReturn(unInventario);  
40         Inventario saved = inventarioServiceImpl.save(unInventario);  
41         assertNotNull(saved);  
42         assertEquals(expected:"Audífonos Gamer HyperX", saved.getNombre());  
43         verify(inventarioRepositories).save(unInventario);  
44     }  
45       
46     @Test  
47     void deleteTest() {  
48         when(inventarioRepositories.findById(id:1L)).thenReturn(Optional.of(unInventario));  
49           
50         Optional<Inventario> eliminado = inventarioServiceImpl.delete(unInventario);  
51           
52         assertTrue(eliminado.isPresent());  
53         assertEquals(expected:"Audífonos Gamer HyperX", eliminado.get().getNombre());  
54         verify(inventarioRepositories).findById(id:1L);  
55         verify(inventarioRepositories).delete(unInventario);  
56     }  
57       
58 }
```

0 △ 0 Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF () Java ⚙️ 🔍

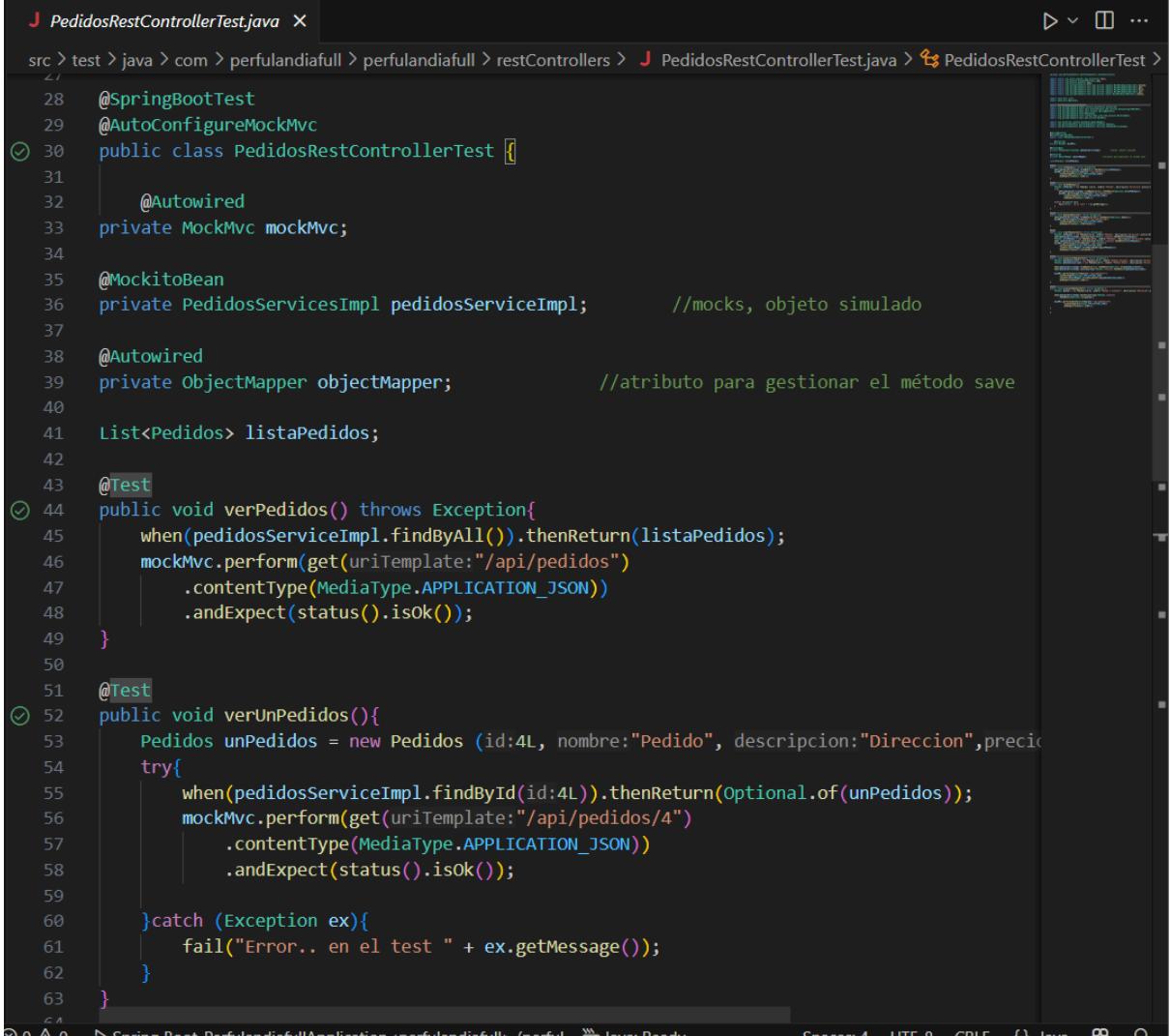
Codificación de Pruebas de integración:

Testing a Pedidos:



```
J PedidosRestControllerTest.java X
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest >
1 package com.perfulandiafull.perfulandiafull.restControllers;
2
3 import static org.junit.jupiter.api.Assertions.fail;
4 import static org.mockito.ArgumentMatchers.any;
5 import static org.mockito.Mockito.when;
6 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;
7 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
8 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
9 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.put;
10 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
11
12 import java.util.List;
13 import java.util.Optional;
14
15 import org.junit.jupiter.api.Test;
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
18 import org.springframework.boot.test.context.SpringBootTest;
19 import org.springframework.http.MediaType;
20 import org.springframework.test.context.context.OverrideMockitoMockBean;
21 import org.springframework.test.web.servlet.MockMvc;
22
23 import com.fasterxml.jackson.databind.ObjectMapper;
24 import com.perfulandiafull.perfulandiafull.entities.Pedidos;
25 import com.perfulandiafull.perfulandiafull.services.PedidosServicesImpl;
26
27
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class PedidosRestControllerTest {
31
32     @Autowired
33     private MockMvc mockMvc;
34
35     @MockitoBean
36     private PedidosServicesImpl pedidosServiceImpl; //mocks, objeto simulado
37 }
```

0 △ 0 ➔ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java



The screenshot shows a code editor with a dark theme displaying Java test code for a REST controller. The code is annotated with line numbers and highlights. It includes imports for Spring Boot, Mockito, and MockMvc, and defines two test methods: `verPedidos()` and `verUnPedidos()`. The `verPedidos()` method uses a mock service to return a list of Pedidos and performs a GET request to verify the response. The `verUnPedidos()` method creates a single Pedido object, performs a GET request for its ID, and handles any exceptions that might occur. The code is part of a project named `Spring Boot-PerfulandiafullApplication`.

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest >
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class PedidosRestControllerTest {
31
32     @Autowired
33     private MockMvc mockMvc;
34
35     @MockitoBean
36     private PedidosServicesImpl pedidosServiceImpl; //mocks, objeto simulado
37
38     @Autowired
39     private ObjectMapper objectMapper; //atributo para gestionar el método save
40
41     List<Pedidos> listaPedidos;
42
43     @Test
44     public void verPedidos() throws Exception{
45         when(pedidosServiceImpl.findAll()).thenReturn(listaPedidos);
46         mockMvc.perform(get(uriTemplate:"/api/pedidos")
47                         .contentType(MediaType.APPLICATION_JSON))
48                         .andExpect(status().isOk());
49     }
50
51     @Test
52     public void verUnPedidos(){
53         Pedidos unPedidos = new Pedidos (id:4L, nombre:"Pedido", descripcion:"Direccion", precio:10000);
54         try{
55             when(pedidosServiceImpl.findById(id:4L)).thenReturn(Optional.of(unPedidos));
56             mockMvc.perform(get(uriTemplate:"/api/pedidos/4")
57                             .contentType(MediaType.APPLICATION_JSON))
58                             .andExpect(status().isOk());
59
60         }catch (Exception ex){
61             fail("Error.. en el test " + ex.getMessage());
62         }
63     }
64 }
```

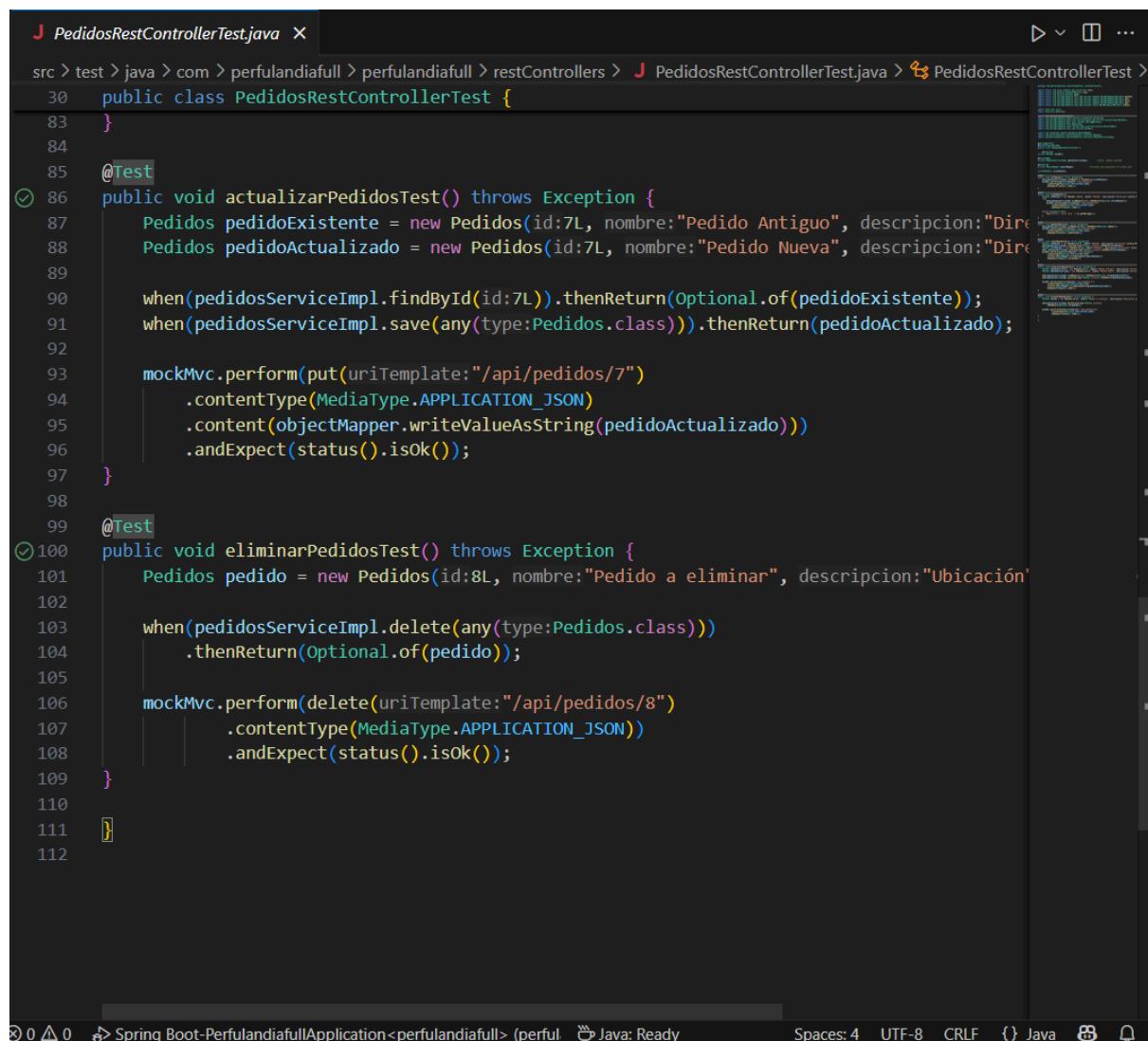


The screenshot shows an IDE interface with a code editor and a terminal window. The code editor displays a Java test class named `PedidosRestControllerTest.java`. The class contains several test methods using the `@Test` annotation. The test cases cover various operations like verifying a list of orders, testing for a non-existent order, creating a new order, and updating an existing order. The code uses Mockito for mocking services and Spring MVC for performing requests. The terminal window at the bottom shows the command to run the application and the Java version.

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest >
30 public class PedidosRestControllerTest {
31     public void verUnPedidos(){
32         try{
33             }catch (Exception ex){
34                 fail("Error.. en el test " + ex.getMessage());
35         }
36     }
37
38     @Test
39     public void pedidosNoExiste() throws Exception{
40         when(pedidosServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
41         mockMvc.perform(get(uriTemplate:"/api/pedidos/10")
42                         .contentType(MediaType.APPLICATION_JSON)
43                         .andExpect(status().isNotFound());
44     }
45
46     @Test
47     public void crearPedidosTest() throws Exception{
48         Pedidos unPedidos = new Pedidos(id:5L, nombre:"Pedido", descripcion:"Dirección", precio:1000);
49         when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(unPedidos);
50         Pedidos otroPedidos = new Pedidos(id:6L, nombre:"Pedido1", descripcion:"Dirección1", precio:1500);
51         when (pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(otroPedidos);
52         mockMvc.perform(post(uriTemplate:"/api/pedidos")
53                         .contentType(MediaType.APPLICATION_JSON)
54                         .content(objectMapper.writeValueAsString(unPedidos)))
55                         .andExpect(status().isCreated());
56     }
57
58     @Test
59     public void actualizarPedidosTest() throws Exception {
60         Pedidos pedidoExistente = new Pedidos(id:7L, nombre:"Pedido Antiguo", descripcion:"Dirección");
61         Pedidos pedidoActualizado = new Pedidos(id:7L, nombre:"Pedido Nueva", descripcion:"Dirección");
62
63         when(pedidosServiceImpl.findById(id:7L)).thenReturn(Optional.of(pedidoExistente));
64         when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(pedidoActualizado);
65
66         mockMvc.perform(put(uriTemplate:"/api/pedidos/7")
67                         .contentType(MediaType.APPLICATION_JSON)
68                         .content(objectMapper.writeValueAsString(pedidoActualizado)))
69                         .andExpect(status().isOk());
70     }
71 }
```

Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready

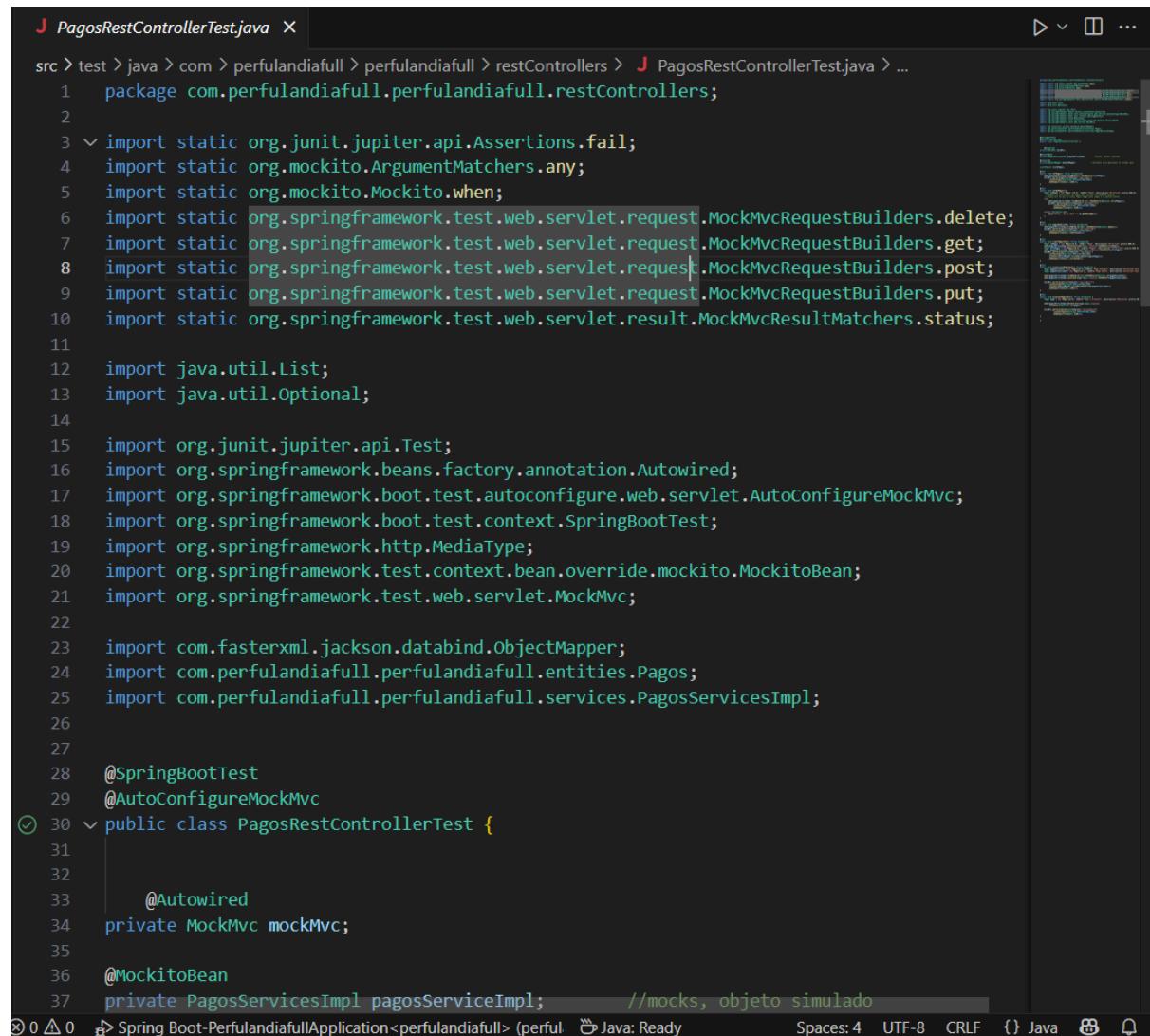
Spaces: 4 UTF-8 CRLF {} Java



```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > PedidosRestControllerTest.java > PedidosRestControllerTest >
30 public class PedidosRestControllerTest {
31
32 }
33
34
35 @Test
36 public void actualizarPedidosTest() throws Exception {
37     Pedidos pedidoExistente = new Pedidos(id:7L, nombre:"Pedido Antiguo", descripcion:"Dirección antigua");
38     Pedidos pedidoActualizado = new Pedidos(id:7L, nombre:"Pedido Nueva", descripcion:"Dirección nueva");
39
40     when(pedidosServiceImpl.findById(id:7L)).thenReturn(Optional.of(pedidoExistente));
41     when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(pedidoActualizado);
42
43     mockMvc.perform(put(uriTemplate:"/api/pedidos/{id}")
44         .contentType(MediaType.APPLICATION_JSON)
45         .content(objectMapper.writeValueAsString(pedidoActualizado)))
46         .andExpect(status().isOk());
47 }
48
49
50 @Test
51 public void eliminarPedidosTest() throws Exception {
52     Pedidos pedido = new Pedidos(id:8L, nombre:"Pedido a eliminar", descripcion:"Ubicación");
53
54     when(pedidosServiceImpl.delete(any(type:Pedidos.class)))
55         .thenReturn(Optional.of(pedido));
56
57     mockMvc.perform(delete(uriTemplate:"/api/pedidos/{id}")
58         .contentType(MediaType.APPLICATION_JSON))
59         .andExpect(status().isOk());
60 }
61
62 }
```

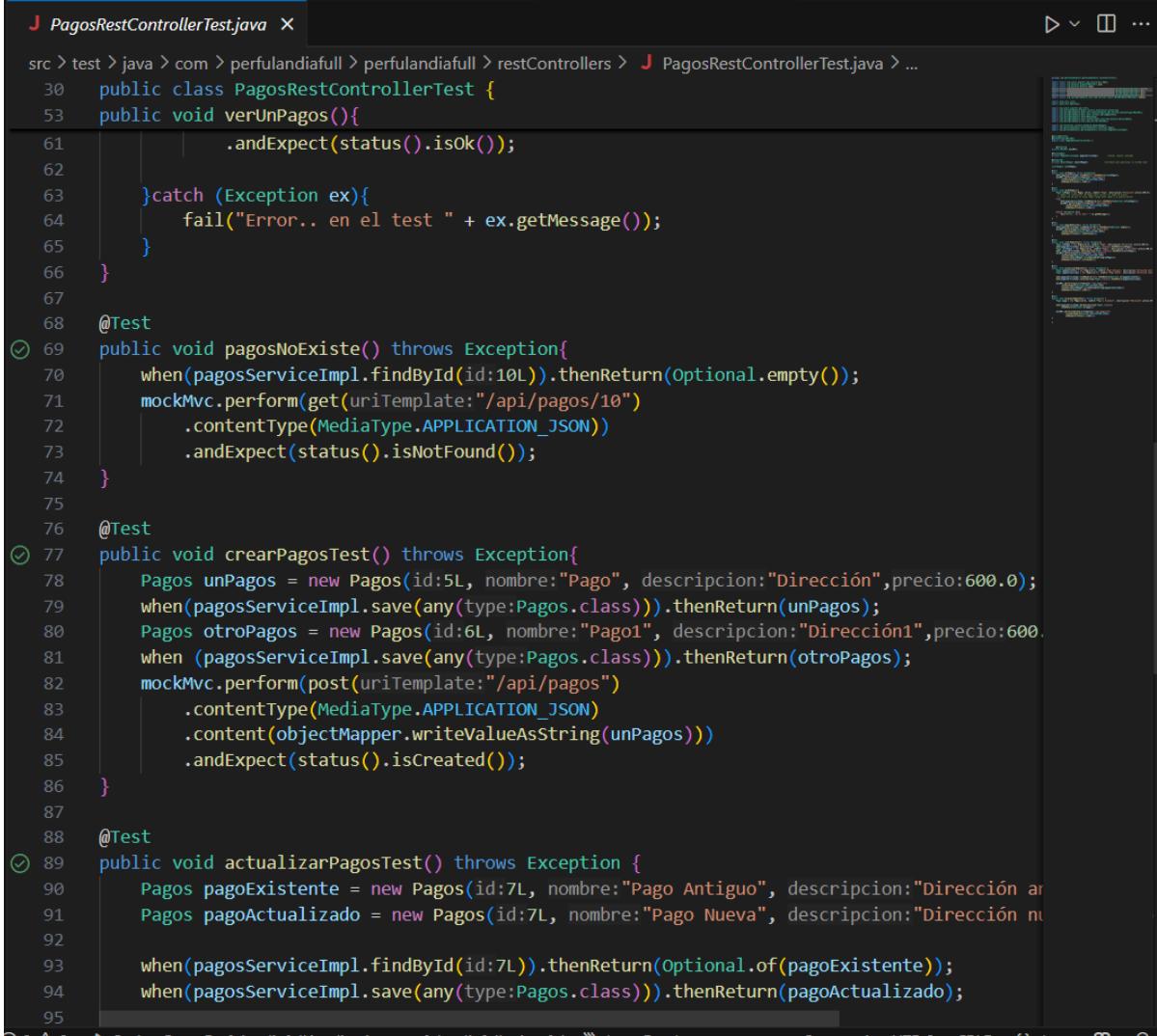
0 △ 0 Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java 🔍 🔍

Testing a Pagos:



```
J PagosRestControllerTest.java X
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PagosRestControllerTest.java > ...
1 package com.perfulandiafull.perfulandiafull.restControllers;
2
3 import static org.junit.jupiter.api.Assertions.fail;
4 import static org.mockito.ArgumentMatchers.any;
5 import static org.mockito.Mockito.when;
6 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;
7 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
8 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
9 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.put;
10 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
11
12 import java.util.List;
13 import java.util.Optional;
14
15 import org.junit.jupiter.api.Test;
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
18 import org.springframework.boot.test.context.SpringBootTest;
19 import org.springframework.http.MediaType;
20 import org.springframework.test.context.bean.override.mockito.MockitoBean;
21 import org.springframework.test.web.servlet.MockMvc;
22
23 import com.fasterxml.jackson.databind.ObjectMapper;
24 import com.perfulandiafull.perfulandiafull.entities.Pagos;
25 import com.perfulandiafull.perfulandiafull.services.PagosServiceImpl;
26
27
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class PagosRestControllerTest {
31
32
33     @Autowired
34     private MockMvc mockMvc;
35
36     @MockitoBean
37     private PagosServiceImpl pagosServiceImpl; //mocks, objeto simulado
38 }
```





```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > PagosRestControllerTest.java > ...
30 public class PagosRestControllerTest {
31     public void verUnPagos(){
32         .andExpect(status().isOk());
33     }
34     }catch (Exception ex){
35         fail("Error.. en el test " + ex.getMessage());
36     }
37 }
38
39 @Test
40 public void pagosNoExiste() throws Exception{
41     when(pagosServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
42     mockMvc.perform(get(uriTemplate:"/api/pagos/10")
43         .contentType(MediaType.APPLICATION_JSON)
44         .andExpect(status().isNotFound());
45 }
46
47 @Test
48 public void crearPagosTest() throws Exception{
49     Pagos unPagos = new Pagos(id:5L, nombre:"Pago", descripcion:"Dirección",precio:600.0);
50     when(pagosServiceImpl.save(any(type:Pagos.class))).thenReturn(unPagos);
51     Pagos otroPagos = new Pagos(id:6L, nombre:"Pago1", descripcion:"Dirección1",precio:600.0);
52     when (pagosServiceImpl.save(any(type:Pagos.class))).thenReturn(otroPagos);
53     mockMvc.perform(post(uriTemplate:"/api/pagos")
54         .contentType(MediaType.APPLICATION_JSON)
55         .content(objectMapper.writeValueAsString(unPagos)))
56         .andExpect(status().isCreated());
57 }
58
59 @Test
60 public void actualizarPagosTest() throws Exception {
61     Pagos pagoExistente = new Pagos(id:7L, nombre:"Pago Antiguo", descripcion:"Dirección antigua");
62     Pagos pagoActualizado = new Pagos(id:7L, nombre:"Pago Nueva", descripcion:"Dirección nueva");
63
64     when(pagosServiceImpl.findById(id:7L)).thenReturn(Optional.of(pagoExistente));
65     when(pagosServiceImpl.save(any(type:Pagos.class))).thenReturn(pagoActualizado);
66 }
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
```

Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java

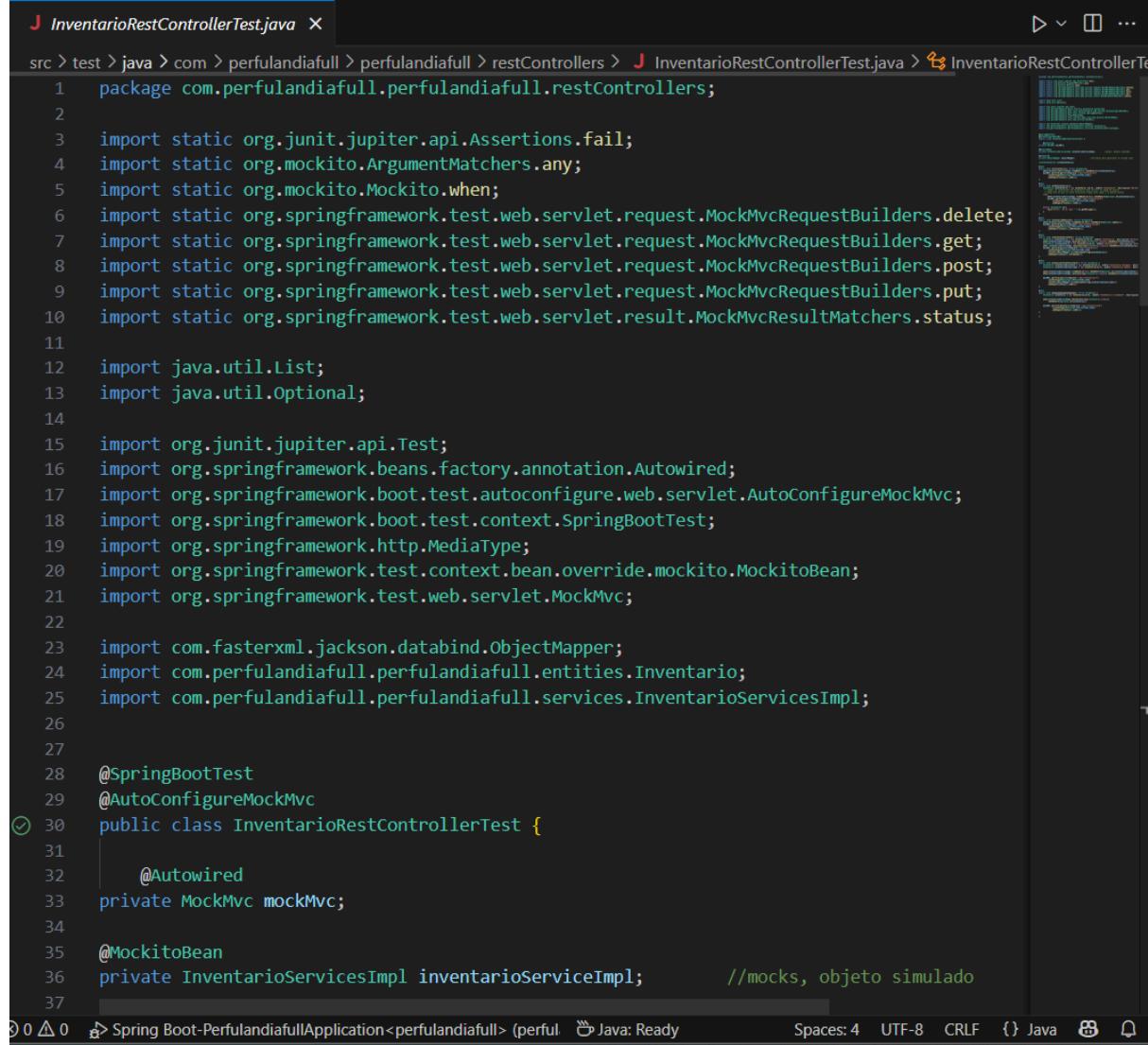
The screenshot shows a code editor with a dark theme. The file being edited is `PagosRestControllerTest.java`. The code contains two test methods: `actualizarPagosTest()` and `eliminarPagosTest()`. Both tests use the `MockMvc` and `Mockito` frameworks to interact with a `PagossServiceImpl` and verify responses. The code is well-structured with proper indentation and comments.

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PagosRestControllerTest.java > ...
30 public class PagosRestControllerTest {
31
32     @Test
33     public void actualizarPagosTest() throws Exception {
34         Pagos pagoExistente = new Pagos(id:7L, nombre:"Pago Antiguo", descripcion:"Dirección antigua", precio:10000);
35         Pagos pagoActualizado = new Pagos(id:7L, nombre:"Pago Nueva", descripcion:"Dirección nueva", precio:20000);
36
37         when(pagossServiceImpl.findById(id:7L)).thenReturn(Optional.of(pagoExistente));
38         when(pagossServiceImpl.save(any(type:Pagos.class))).thenReturn(pagoActualizado);
39
40         mockMvc.perform(put(uriTemplate:"/api/pagos/{id}")
41                         .contentType(MediaType.APPLICATION_JSON)
42                         .content(objectMapper.writeValueAsString(pagoActualizado)))
43                         .andExpect(status().isOk());
44     }
45
46     @Test
47     public void eliminarPagosTest() throws Exception {
48         Pagos pago = new Pagos(id:8L, nombre:"Pago a eliminar", descripcion:"Ubicación", precio:10000);
49
50         when(pagossServiceImpl.delete(any(type:Pagos.class)))
51             .thenReturn(Optional.of(pago));
52
53         mockMvc.perform(delete(uriTemplate:"/api/pagos/{id}")
54                         .contentType(MediaType.APPLICATION_JSON))
55                         .andExpect(status().isOk());
56     }
57
58 }
```

At the bottom of the screen, there is a status bar with the following information:

- Spring Boot-PerfulandiafullApplication<perfulandiafull> (perful...)
- Java: Ready
- Spaces: 4
- UTF-8
- CR LF
- Java
- Run

Testing a Inventory:



The screenshot shows a code editor window with a dark theme. The file being edited is `InventarioRestControllerTest.java`, located at `src > test > java > com > perfulandiafull > restControllers > InventarioRestControllerTest.java`. The code is a JUnit test for a REST controller. It imports various Spring and Mockito annotations and classes. The class is annotated with `@SpringBootTest` and `@AutoConfigureMockMvc`. It contains a constructor that injects `MockMvc` and `InventarioServiceImpl` (marked with `@MockitoBean`). The code then uses `MockMvcBuilders` to build a `MockMvc` instance and performs various HTTP requests (GET, POST, PUT, DELETE) to test the controller's behavior. The code editor has a status bar at the bottom showing the project name, Java version, and other settings.

```
J InventarioRestControllerTest.java X
src > test > java > com > perfulandiafull > restControllers > J InventarioRestControllerTest.java > InventarioRestControllerTe
1 package com.perfulandiafull.perfulandiafull.restControllers;
2
3 import static org.junit.jupiter.api.Assertions.fail;
4 import static org.mockito.ArgumentMatchers.any;
5 import static org.mockito.Mockito.when;
6 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;
7 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
8 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;
9 import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.put;
10 import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;
11
12 import java.util.List;
13 import java.util.Optional;
14
15 import org.junit.jupiter.api.Test;
16 import org.springframework.beans.factory.annotation.Autowired;
17 import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;
18 import org.springframework.boot.test.context.SpringBootTest;
19 import org.springframework.http.MediaType;
20 import org.springframework.test.context.bean.override.mockito.MockitoBean;
21 import org.springframework.test.web.servlet.MockMvc;
22
23 import com.fasterxml.jackson.databind.ObjectMapper;
24 import com.perfulandiafull.perfulandiafull.entities.Inventario;
25 import com.perfulandiafull.perfulandiafull.services.InventarioServiceImpl;
26
27
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class InventarioRestControllerTest {
31
32     @Autowired
33     private MockMvc mockMvc;
34
35     @MockitoBean
36     private InventarioServiceImpl inventarioServiceImpl; //mocks, objeto simulado
37 }
```

The screenshot shows a code editor with a dark theme. The file being edited is `InventarioRestControllerTest.java`. The code is a unit test for a `InventarioRestController`. It uses `MockMvc` and `ObjectMapper` to test the `findAll()` and `findById()` methods. The code includes annotations like `@SpringBootTest`, `@AutoConfigureMockMvc`, and `@Test`. The right side of the screen shows a preview of the JSON response for the `findById` test.

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J InventarioRestControllerTest.java > InventarioRestControllerTest.java
28  @SpringBootTest
29  @AutoConfigureMockMvc
30  public class InventarioRestControllerTest {
31
32      @Autowired
33      private MockMvc mockMvc;
34
35      @MockitoBean
36      private InventarioServicesImpl inventarioServiceImpl;           //mocks, objeto simulado
37
38      @Autowired
39      private ObjectMapper objectMapper;           //atributo para gestionar el método save
40
41      List<Inventario> listaInventario;
42
43      @Test
44      public void verInventario() throws Exception{
45          when(inventarioServiceImpl.findAll()).thenReturn(listaInventario);
46          mockMvc.perform(get(uriTemplate:"/api/inventario")
47              .contentType(MediaType.APPLICATION_JSON))
48              .andExpect(status().isOk());
49      }
50
51      @Test
52      public void verUnInventario(){
53          Inventario unInventario = new Inventario (id:4L, nombre:"Inventario", descripcion:"Dirección");
54          // Cambié el constructor de Inventario para incluir un campo de precio
55          // Asegúrate de que la clase Inventario tenga este campo y su getter/setter
56          try{
57              when(inventarioServiceImpl.findById(id:4L)).thenReturn(optional.of(unInventario));
58              mockMvc.perform(get(uriTemplate:"/api/inventario/4")
59                  .contentType(MediaType.APPLICATION_JSON))
60                  .andExpect(status().isOk());
61
62          }catch (Exception ex){
63              fail("Error.. en el test " + ex.getMessage());
64          }
65      }
66  }
```

0 0 △ 0 ➔ Spring Boot-PerfulandiafullApplication<perfulandiafull> (perfulandiafull) Java: Ready Spaces: 4 UTF-8 CRLF {} Java 🔍

The screenshot shows a code editor with a dark theme. The file being edited is `InventarioRestControllerTest.java`. The code contains several test methods using the `@Test` annotation. The tests include verifying an inventory exists, creating a new inventory, and updating an existing inventory. The code uses `MockMvc` to perform HTTP requests and `Optional.empty()` to handle empty results. The code editor has a status bar at the bottom showing "Spring Boot-PerfulandiafullApplication<perfulandiafull> (perful..." and "Java: Ready".

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J InventarioRestControllerTest.java > InventarioRestControllerTest.java
30 public class InventarioRestControllerTest {
31     @Test
32     public void verUnInventario(){
33         try {
34             inventarioService.findById(1L);
35         } catch (Exception ex){
36             fail("Error.. en el test " + ex.getMessage());
37         }
38     }
39
40     @Test
41     public void inventarioNoExiste() throws Exception{
42         when(inventarioServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
43         mockMvc.perform(get(uriTemplate:"/api/inventario/10")
44                         .contentType(MediaType.APPLICATION_JSON)
45                         .andExpect(status().isNotFound()));
46     }
47
48     @Test
49     public void crearInventarioTest() throws Exception{
50         Inventario unInventario = new Inventario(id:5L, nombre:"Inventario", descripcion:"Dirección de la casa");
51         when(inventarioServiceImpl.save(any(type:Inventario.class))).thenReturn(unInventario);
52         Inventario otroInventario = new Inventario(id:6L, nombre:"Inventario1", descripcion:"Dirección de la oficina");
53         when (inventarioServiceImpl.save(any(type:Inventario.class))).thenReturn(otroInventario);
54         mockMvc.perform(post(uriTemplate:"/api/inventario")
55                         .contentType(MediaType.APPLICATION_JSON)
56                         .content(objectMapper.writeValueAsString(unInventario)))
57                         .andExpect(status().isCreated());
58     }
59
60     @Test
61     public void actualizarInventarioTest() throws Exception {
62         Inventario inventarioExistente = new Inventario(id:7L, nombre:"Inventario Antiguo", descripción:"");
63         Inventario inventarioActualizado = new Inventario(id:7L, nombre:"Inventario Nueva", descripción:"");
64
65         when(inventarioServiceImpl.findById(id:7L)).thenReturn(Optional.of(inventarioExistente));
66         when(inventarioServiceImpl.save(any(type:Inventario.class))).thenReturn(inventarioActualizado);
67
68         mockMvc.perform(put(uriTemplate:"/api/inventario/7"))
69                         .andExpect(status().isOk());
70     }
71 }
```

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J InventarioRestControllerTest.java > InventarioRestControllerTest.java
30  public class InventarioRestControllerTest {
31      public void crearInventarioTest() throws Exception{
32          ...
33      }
34
35      @Test
36      public void actualizarInventarioTest() throws Exception {
37          Inventario inventarioExistente = new Inventario(id:7L, nombre:"Inventario Antiguo", descripcion:"Antiguo");
38          Inventario inventarioActualizado = new Inventario(id:7L, nombre:"Inventario Nueva", descripcion:"Nuevo");
39
40          when(inventarioServiceImpl.findById(id:7L)).thenReturn(Optional.of(inventarioExistente));
41          when(inventarioServiceImpl.save(any(type:Inventario.class))).thenReturn(inventarioActualizado);
42
43          mockMvc.perform(put(uriTemplate:"/api/inventario/7")
44              .contentType(MediaType.APPLICATION_JSON)
45              .content(objectMapper.writeValueAsString(inventarioActualizado)))
46              .andExpect(status().isOk());
47      }
48
49      @Test
50      public void eliminarInventarioTest() throws Exception {
51          Inventario inventario = new Inventario(id:8L, nombre:"Inventario a eliminar", descripcion:"A eliminar");
52
53          when(inventarioServiceImpl.delete(any(type:Inventario.class)))
54              .thenReturn(Optional.of(inventario));
55
56          mockMvc.perform(delete(uriTemplate:"/api/inventario/8")
57              .contentType(MediaType.APPLICATION_JSON))
58              .andExpect(status().isOk());
59      }
60
61      ...
62  }
```

Tecnologías y Frameworks utilizados:

1. JUnit 5 (Jupiter)

Tipo de pruebas: Unitarias y de integración.

Uso:

JUnit es el framework principal para la escritura y ejecución de pruebas automatizadas en Java.

Anotaciones clave:

- `@Test` Indica un método de prueba.
- `@BeforeEach`: Método que se ejecuta antes de cada test.
- `@AfterEach`: Método que se ejecuta después de cada test (opcional).

2. Mockito

Tipo de pruebas: Principalmente **unitarias**

Uso:

Mockito se utiliza para **simular (mockear)** el comportamiento de componentes externos o dependencias de las clases que se están probando. Por ejemplo, en las pruebas unitarias de tu servicio, simulas la respuesta del repositorio.

Anotaciones clave:

- `@Mock`: Crea un objeto simulado (mock).
- `@InjectMocks`: Inyecta los mocks en la clase a testear.
- `when(...).thenReturn(...)`: Define comportamientos simulados.
- `verify(...)`: Verifica que un método fue llamado.

3. Spring Boot Test (spring-boot-starter-test)

Tipo de pruebas: Integración y unitarias (con configuración Spring)

Uso:

Spring Boot Test proporciona un entorno completo para pruebas con contexto de Spring cargado. Es ideal para pruebas de controladores (`@RestController`) y servicios con dependencias reales o simuladas.

Anotaciones clave:

- `@SpringBootTest`: Carga todo el contexto de la aplicación (ideal para pruebas de integración).
- `@WebMvcTest`: Carga solo los controladores y componentes de la capa web.

- `@AutoConfigureMockMvc`: Permite usar MockMvc para simular peticiones HTTP.
- `MockMvc`: Se usa para hacer peticiones simuladas (GET, POST, etc.) a los endpoints.

4. MockMvc

Tipo de pruebas: Integración (REST)

Uso:

MockMvc permite realizar llamadas HTTP simuladas a tus endpoints REST para verificar su funcionamiento. Es ideal para probar los `@RestController` sin arrancar el servidor completo.

5. AssertJ / Assertions (de JUnit)

Tipo de pruebas: Ambas

Uso:

Utilizado para hacer afirmaciones (asserts) y verificar que los valores obtenidos en la prueba son los esperados.

❖ Pruebas Unitarias.

Prueba a Services:

Testing a Pedidos:

TESTING

Filter (e.g. text, !exclude, @tag)

101/101 20.6s

- perfulandiafull 961ms
- com.perfulandiafull.perfulandiafull 2.0ms
- PerfulandiafullApplicationTests 2.0ms
 - contextLoads() 2.0ms
- com.perfulandiafull.perfulandiafullTestControllers 81...
 - AdministradorRestControllerTest 51ms
 - ClienteRestControllerTest 52ms
 - EncargadoLogisticaRestControllerTest 53ms
 - EncargadoVentasRestControllerTest 57ms
 - GerenteRestControllerTest 53ms
 - InventarioRestControllerTest 58ms
 - PagosRestControllerTest 310ms
 - PedidosRestControllerTest 88ms
 - ProductoRestControllerTest 46ms
 - SucursalesRestControllerTest 49ms
- com.perfulandiafull.perfulandiafull.services 142ms
 - AdministradorServicesTest 12ms
 - ClientesServicesTest 9.0ms
 - EncargadoLogisticaServicesTest 13ms
 - EncargadoVentasServicesTest 12ms
 - GerenteServicesTest 20ms
 - InventarioServicesTest 14ms
 - PagosServicesTest 11ms
 - PedidosServicesTest 12ms
 - findByIdAllTest() 2.0ms
 - findByIdFoundTest() 4.0ms
 - SaveTest() 3.0ms
 - deleteTest() 3.0ms
 - ProductoServicesTest 14ms
 - SucursalesServicesTest 25ms

Java: Ready

Ln 43, Col 1 Spaces: 4 UTF-8 CRLF {)

TESTING

Filter (e.g. text, !exclude, @tag)

101/101 20.6s

- perfulandiafull 961ms
- com.perfulandiafull.perfulandiafull 2.0ms
- PerfulandiafullApplicationTests 2.0ms
 - contextLoads() 2.0ms
- com.perfulandiafull.perfulandiafullTestControllers 81...
 - AdministradorRestControllerTest 51ms
 - ClienteRestControllerTest 52ms
 - EncargadoLogisticaRestControllerTest 53ms
 - EncargadoVentasRestControllerTest 57ms
 - GerenteRestControllerTest 53ms
 - InventarioRestControllerTest 58ms
 - PagosRestControllerTest 310ms
 - PedidosRestControllerTest 88ms
 - ProductoRestControllerTest 46ms
 - SucursalesRestControllerTest 49ms
- com.perfulandiafull.perfulandiafull.services 142ms
 - AdministradorServicesTest 12ms
 - ClientesServicesTest 9.0ms
 - EncargadoLogisticaServicesTest 13ms
 - EncargadoVentasServicesTest 12ms
 - GerenteServicesTest 20ms
 - InventarioServicesTest 14ms
 - PagosServicesTest 11ms
 - PedidosServicesTest 12ms
 - findByIdAllTest() 2.0ms
 - findByIdFoundTest() 4.0ms
 - SaveTest() 3.0ms
 - deleteTest() 3.0ms
 - ProductoServicesTest 14ms
 - SucursalesServicesTest 25ms

Java: Ready

Ln 43, Col 1 Spaces: 4 UTF-8 CRLF {)

Testing a Pagos:

TESTING

```

src > test > java > com > perfulandiafull > perfulandiafull > services > J PagosServicesTest.java > ⚡ PagosServicesTest > ⚡ findByAllTest()
22 import com.perfulandiafull.perfulandiafull.repositories.PagosRepository;
23
24 @SpringBootTest
25 public class PagosServicesTest {
26
27     @Mock
28     private PagosRepository pagosRepositories;
29
30     @InjectMocks
31     private PagosServicesImpl pagosServiceImpl;
32
33     private Pagos unPagos;
34
35     @BeforeEach
36     void Inicio(){
37         MockitoAnnotations.openMocks(this);
38         unPagos = new Pagos();
39         unPagos.setId(1L);
40         unPagos.setNombre("Audifonos Gamer HyperX");
41     }
42
43     @Test
44     void findByAllTest(){
45         List<Pagos> unPagosList = Arrays.asList(unPagos);
46
47         when(pagosRepositories.findAll()).thenReturn(unPagosList);
48         List<Pagos> resultado = pagosServiceImpl.findByAll();
49
50         assertEquals(expected:1, resultado.size());
51         verify(pagosRepositories).findAll();
52     }
53
54     @Test
55     void findByIdFoundTest(){
56         when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
57         Optional<Pagos> resultado = pagosServiceImpl.findById(id:1L);
58         assertTrue(resultado.isPresent());
59         assertEquals(expected:"Audifonos Gamer HyperX", resultado.get().getNombre());
60         verify(pagosRepositories).findById(id:1L);
61     }
62
63     @Test
64     void SaveTest(){
65         when(pagosRepositories.save(any(type:Pagos.class))).thenReturn(unPagos);
66         Pagos saved = pagosServiceImpl.save(unPagos);
67         assertNotNull(saved);
68         assertEquals(expected:"Audifonos Gamer HyperX", saved.getNombre());
69         verify(pagosRepositories).save(unPagos);
70     }
71
72     @Test
73     void deleteTest() {
74         when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
75
76         Optional<Pagos> eliminado = pagosServiceImpl.delete(unPagos);
77
78         assertTrue(eliminado.isPresent());
79         assertEquals(expected:"Audifonos Gamer HyperX", eliminado.get().getNombre());
80         verify(pagosRepositories).findById(id:1L);
81         verify(pagosRepositories).delete(unPagos);
82     }
83
84 }
```

Ln 44, Col 1 Spaces:4 UTF-8 CRLF { }

TESTING

```

src > test > java > com > perfulandiafull > perfulandiafull > services > J PagosServicesTest.java > ⚡ PagosServicesTest > ⚡ findByAllTest()
25 public class PagoservicesTest {
26 }
27
28 @Test
29 void findByAllTest(){
30     when(pagosRepositories.findAll()).thenReturn(unPagosList);
31     List<Pagos> resultado = pagosServiceImpl.findByAll();
32
33     assertEquals(expected:1, resultado.size());
34     verify(pagosRepositories).findAll();
35 }
36
37 @Test
38 void findByIdFoundTest(){
39     when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
40     Optional<Pagos> resultado = pagosServiceImpl.findById(id:1L);
41     assertTrue(resultado.isPresent());
42     assertEquals(expected:"Audifonos Gamer HyperX", resultado.get().getNombre());
43     verify(pagosRepositories).findById(id:1L);
44 }
45
46 @Test
47 void SaveTest(){
48     when(pagosRepositories.save(any(type:Pagos.class))).thenReturn(unPagos);
49     Pagos saved = pagosServiceImpl.save(unPagos);
50     assertNotNull(saved);
51     assertEquals(expected:"Audifonos Gamer HyperX", saved.getNombre());
52     verify(pagosRepositories).save(unPagos);
53 }
54
55 @Test
56 void deleteTest() {
57     when(pagosRepositories.findById(id:1L)).thenReturn(Optional.of(unPagos));
58
59     Optional<Pagos> eliminado = pagosServiceImpl.delete(unPagos);
60
61     assertTrue(eliminado.isPresent());
62     assertEquals(expected:"Audifonos Gamer HyperX", eliminado.get().getNombre());
63     verify(pagosRepositories).findById(id:1L);
64     verify(pagosRepositories).delete(unPagos);
65 }
```

Testing a Inventory:

TESTING J Inventarios Spring Boot-Perfulandia

```

src > test > java > com > perfulandiafull > perfulandiafull > services > J inventarioServicesTest.java > findbyAllTest()
25 @SpringBootTest
26 public class inventarioServicesTest {
27
28     @Mock
29     private inventarioRepository inventarioRepositories;
30
31     @InjectMocks
32     private inventarioServicesImpl inventarioServiceImpl;
33
34     private inventario unInventario;
35
36     @BeforeEach
37     void Inicio(){
38         MockitoAnnotations.openMocks(this);
39         unInventario = new inventario();
40         unInventario.setId(1L);
41         unInventario.setNombre("Audifonos Gamer HyperX");
42     }
43
44     @Test
45     void findbyAllTest(){
46
47
48
49
50
51         list <inventario> unInventario = Arrays.asList(unInventario);
52
53         when(inventarioRepositories.findAll()).thenReturn(unInventario);
54         list <inventario> resultado = inventarioServiceImpl.findAll();
55
56         assertEquals(expected:1, resultado.size());
57         verify(inventarioRepositories).findAll();
58     }
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Ln 46, Col 1 Spaces:4 UTF-8 CRLF { }

TESTING J Inventarios Spring Boot-Perfulandia

```

src > test > java > com > perfulandiafull > perfulandiafull > services > J inventarioServicesTest.java > findbyAllTest()
26 public class inventarioServicesTest {
27
28     void findbyAllTest(){
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

❖ Pruebas de integración

Prueba a RestController:

Testing a inventario

TESTING

```

src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest > verPedidos()

28  @SpringBootTest
29  @AutoConfigureMockMvc
30  public class PedidosRestControllerTest {
31
32      @Autowired
33      private MockMvc mockMvc;
34
35      @MockBean
36      private PedidosServiceImpl pedidosService; //mocks, objeto simulado
37
38      @Autowired
39      private ObjectMapper objectMapper; //atributo para gestionar el método save
40
41      List<Pedidos> listaPedidos;
42
43      @Test
44      public void verPedidos() throws Exception{
45          when(pedidosService.findByIdAll()).thenReturn(listaPedidos);
46          mockMvc.perform(get(uriTemplate:"/api/pedidos")
47              .contentType(MediaType.APPLICATION_JSON))
48              .andExpect(status().isOk());
49      }
50
51      @Test
52      public void verUnPedidos(){
53          Pedidos unPedidos = new Pedidos (id:4L, nombre:"Pedido", descripcion:"Direccion",precio:500.0);
54          try{
55              when(pedidosService.findById(id:4L)).thenReturn(Optional.of(unPedidos));
56              mockMvc.perform(get(uriTemplate:"/api/pedidos/4")
57                  .contentType(MediaType.APPLICATION_JSON))
58                  .andExpect(status().isOk());
59
60          }catch (Exception ex){
61              fail("Error.. en el test " + ex.getMessage());
62          }
63      }

```

TESTING

```

src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest > verPedidos()

30  public class PedidosRestControllerTest {
31      public void verUnPedidos(){
32          andExpect(status().isOK());
33
34      }catch (Exception ex){
35          fail("Error.. en el test " + ex.getMessage());
36      }
37
38      @Test
39      public void pedidosNoExiste() throws Exception{
40          when(pedidosService.findById(id:10L)).thenReturn(Optional.empty());
41          mockMvc.perform(get(uriTemplate:"/api/pedidos/10")
42              .contentType(MediaType.APPLICATION_JSON))
43              .andExpect(status().isNotFound());
44      }
45
46      @Test
47      public void crearPedidosTest() throws Exception{
48          Pedidos unPedidos = new Pedidos(id:5L, nombre:"Pedido", descripcion:"Direccion",precio:500.0);
49          when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(unPedidos);
50          Pedidos otroPedidos = new Pedidos(id:6L, nombre:"Pedido1", descripcion:"Direccion1",precio:500.0);
51          when (pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(otroPedidos);
52          mockMvc.perform(post(uriTemplate:"/api/pedidos")
53              .contentType(MediaType.APPLICATION_JSON)
54              .content(objectMapper.writeValueAsString(unPedidos)))
55              .andExpect(status().isCreated());
56      }
57
58      @Test
59      public void actualizarPedidosTest() throws Exception {
60          Pedidos pedidoExistente = new Pedidos(id:7L, nombre:"Pedido Antiguo", descripcion:"Direccion antigua",precio:500.0);
61          Pedidos pedidoActualizado = new Pedidos(id:7L, nombre:"Pedido Nueva", descripcion:"Direccion nueva",precio:500.0);
62
63          when(pedidosServiceImpl.findById(id:7L)).thenReturn(Optional.of(pedidoExistente));
64          when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(pedidoActualizado);
65      }

```

TESTING

```

src > test > java > com > perfulandiafull > restControllers > J PedidosRestControllerTest.java > PedidosRestControllerTest > verPedidos()

30 public class PedidosRestControllerTest {
31
32
33
34
35     @Test
36     public void actualizarPedidosTest() throws Exception {
37         Pedidos pedidoExistente = new Pedidos(id:7L, nombre:"Pedido Antiguo", descripcion:"Dirección antigua", precio:500.0);
38         Pedidos pedidoActualizado = new Pedidos(id:7L, nombre:"Pedido Nueva", descripcion:"Dirección nueva", precio:500.0);
39
40         when(pedidosServiceImpl.findById(id:7L)).thenReturn(Optional.of(pedidoExistente));
41         when(pedidosServiceImpl.save(any(type:Pedidos.class))).thenReturn(pedidoActualizado);
42
43         mockMvc.perform(put(uriTemplate:"/api/pedidos/7")
44                         .contentType(MediaType.APPLICATION_JSON)
45                         .content(objectMapper.writeValueAsString(pedidoActualizado)))
46                         .andExpect(status().isOk());
47     }
48
49     @Test
50     public void eliminarPedidosTest() throws Exception {
51         Pedidos pedido = new Pedidos(id:8L, nombre:"Pedido a eliminar", descripcion:"Ubicación", precio:500.0);
52
53         when(pedidosServiceImpl.delete(any(type:Pedidos.class)))
54             .thenReturn(Optional.of(pedido));
55
56         mockMvc.perform(delete(uriTemplate:"/api/pedidos/8")
57                         .contentType(MediaType.APPLICATION_JSON))
58                         .andExpect(status().isOk());
59     }
60
61 }
```

0 o △ 0 ➔ Spring Boot-PerfulandiafullApplication-<perfudiafull> (perfudiafull) Java: Ready

Testing a Pagos:

TESTING

```

src > test > java > com > perfudiafull > restControllers > J PagosRestControllerTest.java > PagosRestControllerTest > verPagos()

27
28     @SpringBootTest
29     @AutoConfigureMockMvc
30     public class PagosRestControllerTest {
31
32         @Autowired
33         private MockMvc mockMvc;
34
35         @MockitoBean
36         private PagosServicesImpl pagosServiceImpl;           //mocks, objeto simulado
37
38         @Autowired
39         private ObjectMapper objectMapper;                   //atributo para gestionar el método save
40
41         List<Pagos> listaPagos;
42
43         @Test
44         public void verPagos() throws Exception{
45             when(pagosServiceImpl.findAll()).thenReturn(listaPagos);
46             mockMvc.perform(get(uriTemplate:"/api/pagos")
47                             .contentType(MediaType.APPLICATION_JSON))
48                             .andExpect(status().isOk());
49         }
50
51         @Test
52         public void verUnPagos(){
53             Pagos unPagos = new Pagos (id:4L, nombre:"Pago", descripción:"Dirección", precio:600.0);
54             // Cambié el constructor de Pagos para incluir un campo de precio
55             // Aseguré de que la clase Pagos tenga este campo y su getter/setter
56             try{
57                 when(pagosServiceImpl.findById(id:4L)).thenReturn(Optional.of(unPagos));
58                 mockMvc.perform(get(uriTemplate:"/api/pagos/4")
59                                 .contentType(MediaType.APPLICATION_JSON))
60                                 .andExpect(status().isOk());
61             }catch (Exception ex){
62             }
63         }
64
65 }
```

0 o △ 0 ➔ Spring Boot-PerfulandiafullApplication-<perfudiafull> (perfudiafull) Java: Ready

TESTING

Filter (e.g. text, exclude, @tag)

101/101 20.6s

- perfulandiafull 961ms
- { com.perfulandiafull.perfulandiafull 2.0ms
- PerfulandiafullApplicationTests 2.0ms
 - contextLoads() 2.0ms
- { com.perfulandiafull.perfulandiafull.restControllers 81...
 - AdministradorRestControllerTest 51ms
 - ClienteRestControllerTest 52ms
 - EncargadoLogisticaRestControllerTest 53ms
 - GerenteRestControllerTest 57ms
 - InventarioRestControllerTest 58ms
 - PagosRestControllerTest 310ms
 - verPagos() 153ms
 - verUnPago() 10ms
 - pagosNoExiste() 7.0ms
 - crearPagosTest() 125ms
 - actualizarPagosTest() 8.0ms
 - eliminarPagosTest() 7.0ms
 - PedidosRestControllerTest 88ms
 - ProductoRestControllerTest 46ms
 - SucursalesRestControllerTest 49ms
- { com.perfulandiafull.perfulandiafull.services 142ms
 - AdministradorServicesTest 12ms
 - ClienteServicesTest 9.0ms
 - EncargadoLogisticaServicesTest 13ms
 - EncargadoVentasServicesTest 12ms
 - GerenteServicesTest 20ms
 - InventarioServicesTest 14ms
 - PagosServicesTest 11ms
 - PedidosServicesTest 12ms

J PagoRestControllerTest.java x

```

src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PagoRestControllerTest.java > PagoRestControllerTest > verPagos()
30 public class PagoRestControllerTest {
31     public void verUnPago(){
32         .contentType(MediaType.APPLICATION_JSON)
33         .andExpect(status().isOk());
34     }
35
36     @Test
37     public void pagosNoExiste() throws Exception{
38         when(pagoServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
39         mockMvc.perform(get(uriTemplate:"/api/pagos/10")
40             .contentType(MediaType.APPLICATION_JSON)
41             .andExpect(status().isNotFound());
42     }
43
44     @Test
45     public void crearPagoTest() throws Exception{
46         Pago unPago = new Pago(id:5L, nombre:"pago", descripción:"Dirección", precio:600.0);
47         when(pagoServiceImpl.save(any(type:Pago.class))).thenReturn(unPago);
48         Pago otroPago = new Pago(id:6L, nombre:"pago1", descripción:"Dirección1", precio:600.0);
49         when(pagoServiceImpl.save(any(type:Pago.class))).thenReturn(otroPago);
50         mockMvc.perform(post(uriTemplate:"/api/pagos")
51             .contentType(MediaType.APPLICATION_JSON)
52             .content(objectMapper.writeValueAsString(unPago)))
53             .andExpect(status().isCreated());
54     }
55
56     @Test
57     public void actualizarPagoTest() throws Exception {
58         Pago pagoExistente = new Pago(id:1L, nombre:"Pago Antiguo", descripción:"Dirección antigua", precio:600.0);
59         Pago pagoActualizado = new Pago(id:7L, nombre:"Pago Nueva", descripción:"Dirección nueva", precio:600.0);
60
61         when(pagoServiceImpl.findById(id:7L)).thenReturn(Optional.of(pagoExistente));
62         when(pagoServiceImpl.save(any(type:Pago.class))).thenReturn(pagoActualizado);
63
64         mockMvc.perform(put(uriTemplate:"/api/pagos/7")
65             .contentType(MediaType.APPLICATION_JSON)
66             .content(objectMapper.writeValueAsString(pagoActualizado)))
67             .andExpect(status().isOk());
68     }
69
70     @Test
71     public void eliminarPagoTest() throws Exception {
72         Pago pago = new Pago(id:8L, nombre:"Pago a eliminar", descripción:"Ubicación", precio:600.0);
73
74         when(pagoServiceImpl.delete(any(type:Pago.class)))
75             .thenReturn(Optional.of(pago));
76
77         mockMvc.perform(delete(uriTemplate:"/api/pagos/8")
78             .contentType(MediaType.APPLICATION_JSON)
79             .andExpect(status().isOk()));
80     }
81
82 }
```

Java: Ready

Ln 45, Col 1 Spaces:4 UTF-8 CRLF {

TESTING

Filter (e.g. text, exclude, @tag)

101/101 20.6s

- perfulandiafull 961ms
- { com.perfulandiafull.perfulandiafull 2.0ms
- PerfulandiafullApplicationTests 2.0ms
 - contextLoads() 2.0ms
- { com.perfulandiafull.perfulandiafull.restControllers 81...
 - AdministradorRestControllerTest 51ms
 - ClienteRestControllerTest 52ms
 - EncargadoLogisticaRestControllerTest 53ms
 - EncargadoVentasRestControllerTest 57ms
 - GerenteRestControllerTest 53ms
 - InventarioRestControllerTest 58ms
 - PagosRestControllerTest 310ms
 - verPagos() 153ms
 - verUnPago() 10ms
 - pagosNoExiste() 7.0ms
 - crearPagosTest() 125ms
 - actualizarPagosTest() 8.0ms
 - eliminarPagosTest() 7.0ms
 - PedidosRestControllerTest 88ms
 - ProductoRestControllerTest 46ms
 - SucursalesRestControllerTest 49ms
- { com.perfulandiafull.perfulandiafull.services 142ms
 - AdministradorServicesTest 12ms
 - ClienteServicesTest 9.0ms
 - EncargadoLogisticaServicesTest 13ms
 - EncargadoVentasServicesTest 12ms
 - GerenteServicesTest 20ms
 - InventarioServicesTest 14ms
 - PagosServicesTest 11ms
 - PedidosServicesTest 12ms

J PagoRestControllerTest.java x

```

src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J PagoRestControllerTest.java > PagoRestControllerTest > verPagos()
30 public class PagoRestControllerTest {
31     public void crearPagoTest() throws Exception{
32         Pago unPago = new Pago(id:5L, nombre:"pago", descripción:"Dirección", precio:600.0);
33         when(pagoServiceImpl.save(any(type:Pago.class))).thenReturn(unPago);
34
35         mockMvc.perform(post(uriTemplate:"/api/pagos")
36             .contentType(MediaType.APPLICATION_JSON)
37             .content(objectMapper.writeValueAsString(unPago)))
38             .andExpect(status().isCreated());
39     }
40
41     @Test
42     public void actualizarPagoTest() throws Exception {
43         Pago pagoExistente = new Pago(id:7L, nombre:"Pago Antiguo", descripción:"Dirección antigua", precio:600.0);
44         Pago pagoActualizado = new Pago(id:7L, nombre:"Pago Nueva", descripción:"Dirección nueva", precio:600.0);
45
46         when(pagoServiceImpl.findById(id:7L)).thenReturn(Optional.of(pagoExistente));
47         when(pagoServiceImpl.save(any(type:Pago.class))).thenReturn(pagoActualizado);
48
49         mockMvc.perform(put(uriTemplate:"/api/pagos/7")
50             .contentType(MediaType.APPLICATION_JSON)
51             .content(objectMapper.writeValueAsString(pagoActualizado)))
52             .andExpect(status().isOk());
53     }
54
55     @Test
56     public void eliminarPagoTest() throws Exception {
57         Pago pago = new Pago(id:8L, nombre:"Pago a eliminar", descripción:"Ubicación", precio:600.0);
58
59         when(pagoServiceImpl.delete(any(type:Pago.class)))
60             .thenReturn(Optional.of(pago));
61
62         mockMvc.perform(delete(uriTemplate:"/api/pagos/8")
63             .contentType(MediaType.APPLICATION_JSON)
64             .andExpect(status().isOk()));
65     }
66
67 }
```

Java: Ready

Ln 45, Col 1 Spaces:4 UTF-8 CRLF {

Testing a Inventoryo:

TESTING

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J_ InventoryRestController.java > ✎ InventoryRestControllerTest > ✎ verInventario()
```

```

101/101 20.6s ⏪
26
27
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class InventoryRestControllerTest {
31
32     @Autowired
33     private MockMvc mockMvc;
34
35     @MockitoBean
36     private InventoryServicesImpl inventoryService; //mocks, objeto simulado
37
38     @Autowired
39     private ObjectMapper objectMapper; //atributo para gestionar el método save
40
41     List<Inventory> listaInventory;
42
43     @Test
44     public void verInventario() throws Exception{
45         when(inventoryService.findAll()).thenReturn(listaInventory);
46         mockMvc.perform(get(uriTemplate:"/api/inventario"))
47             .contentType(MediaType.APPLICATION_JSON)
48             .andExpect(status().isOk());
49     }
50
51     @Test
52     public void verUnInventario(){
53         Inventory unInventory = new Inventory( id:4L, nombre:"Inventoryo", descripción:"Dirección", precio:500.0 );
54         // Cambié el constructor de Inventoryo para incluir un campo de precio
55         // Asegúrate de que la clase Inventoryo tenga este campo y su getter/setter
56         try{
57             when(inventoryService.findById(id:4L)).thenReturn(Optional.of(unInventory));
58             mockMvc.perform(get(uriTemplate:"/api/inventario/4"))
59                 .contentType(MediaType.APPLICATION_JSON)
60                 .andExpect(status().isOk());
61         }catch (Exception ex){
62             fail("Error.. en el test " + ex.getMessage());
63         }
64     }
65
66     @Test
67     public void inventarioNoExiste() throws Exception{
68         when(inventoryService.findById(id:10L)).thenReturn(Optional.empty());
69         mockMvc.perform(get(uriTemplate:"/api/inventario/10"))
70             .contentType(MediaType.APPLICATION_JSON)
71             .andExpect(status().isNotFound());
72     }
73
74     @Test
75     public void crearInventarioTest() throws Exception{
76         Inventory unInventory = new Inventory(id:5L, nombre:"Inventoryo", descripción:"Dirección", precio:500.0);
77         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(unInventory);
78         Inventory otroInventory = new Inventory(id:6L, nombre:"Inventoryo1", descripción:"Dirección1", precio:500.0);
79         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(otroInventory);
80         mockMvc.perform(post(uriTemplate:"/api/inventario")
81             .contentType(MediaType.APPLICATION_JSON)
82             .content(objectMapper.writeValueAsString(unInventory)))
83             .andExpect(status().isCreated());
84     }
85
86     @Test
87     public void actualizarInventarioTest() throws Exception {
88         Inventory inventarioExistente = new Inventory(id:7L, nombre:"Inventoryo Antiguo", descripción:"Dirección antigua");
89         Inventory inventarioActualizado = new Inventory(id:7L, nombre:"Inventoryo Nueva", descripción:"Dirección nueva", p
90
91         when(inventoryService.findById(id:7L)).thenReturn(Optional.of(inventarioExistente));
92         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(inventarioActualizado);
93
94         mockMvc.perform(put(uriTemplate:"/api/inventario/7")
95             .contentType(MediaType.APPLICATION_JSON))
96     }

```

In 44 Col 1 Spaces:4 UTF-8 CR/LF { }

TESTING

```
src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J_ InventoryRestController.java > ✎ InventoryRestControllerTest > ✎ verInventario()
```

```

101/101 20.6s ⏪
26
27
28 @SpringBootTest
29 @AutoConfigureMockMvc
30 public class InventoryRestControllerTest {
31
32     @Autowired
33     private MockMvc mockMvc;
34
35     @MockitoBean
36     private InventoryServicesImpl inventoryService; //mocks, objeto simulado
37
38     @Autowired
39     private ObjectMapper objectMapper; //atributo para gestionar el método save
40
41     List<Inventory> listaInventory;
42
43     @Test
44     public void verInventario() throws Exception{
45         when(inventoryService.findAll()).thenReturn(listaInventory);
46         mockMvc.perform(get(uriTemplate:"/api/inventario"))
47             .contentType(MediaType.APPLICATION_JSON)
48             .andExpect(status().isOk());
49     }
50
51     @Test
52     public void verUnInventario(){
53         Inventory unInventory = new Inventory( id:4L, nombre:"Inventoryo", descripción:"Dirección", precio:500.0 );
54         // Cambié el constructor de Inventoryo para incluir un campo de precio
55         // Asegúrate de que la clase Inventoryo tenga este campo y su getter/setter
56         try{
57             when(inventoryService.findById(id:4L)).thenReturn(Optional.of(unInventory));
58             mockMvc.perform(get(uriTemplate:"/api/inventario/4"))
59                 .contentType(MediaType.APPLICATION_JSON)
60                 .andExpect(status().isOk());
61         }catch (Exception ex){
62             fail("Error.. en el test " + ex.getMessage());
63         }
64     }
65
66     @Test
67     public void inventarioNoExiste() throws Exception{
68         when(inventoryService.findById(id:10L)).thenReturn(Optional.empty());
69         mockMvc.perform(get(uriTemplate:"/api/inventario/10"))
70             .contentType(MediaType.APPLICATION_JSON)
71             .andExpect(status().isNotFound());
72     }
73
74     @Test
75     public void crearInventarioTest() throws Exception{
76         Inventory unInventory = new Inventory(id:5L, nombre:"Inventoryo", descripción:"Dirección", precio:500.0);
77         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(unInventory);
78         Inventory otroInventory = new Inventory(id:6L, nombre:"Inventoryo1", descripción:"Dirección1", precio:500.0);
79         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(otroInventory);
80         mockMvc.perform(post(uriTemplate:"/api/inventario")
81             .contentType(MediaType.APPLICATION_JSON)
82             .content(objectMapper.writeValueAsString(unInventory)))
83             .andExpect(status().isCreated());
84     }
85
86     @Test
87     public void actualizarInventarioTest() throws Exception {
88         Inventory inventarioExistente = new Inventory(id:7L, nombre:"Inventoryo Antiguo", descripción:"Dirección antigua");
89         Inventory inventarioActualizado = new Inventory(id:7L, nombre:"Inventoryo Nueva", descripción:"Dirección nueva", p
90
91         when(inventoryService.findById(id:7L)).thenReturn(Optional.of(inventarioExistente));
92         when(inventoryService.save(any(type:Inventoryo.class))).thenReturn(inventarioActualizado);
93
94         mockMvc.perform(put(uriTemplate:"/api/inventario/7")
95             .contentType(MediaType.APPLICATION_JSON))
96     }

```

In 44 Col 1 Spaces:4 UTF-8 CR/LF { }

TESTING

Filter (e.g. text, exclude, @tag)

101/101 20.6s

- perfulandiafull 961ms
- com.perfulandiafull.perfulandiafull 2.0ms
 - PerfulandiafullApplicationTests 2.0ms
 - contextLoads() 2.0ms
- com.perfulandiafull.perfulandiafull.restControllers 81...
 - AdministradorRestControllerTest 51ms
 - ClienteRestControllerTest 52ms
 - EncargadoLogisticaRestControllerTest 53ms
 - EncargadoVentasRestControllerTest 57ms
 - GerenteRestControllerTest 53ms
 - InventarioRestControllerTest 58ms
 - verInventario() 5.0ms
 - verUnInventario() 10ms
 - inventarioNotExist() 11ms
 - crearInventarioTest() 9.0ms
 - actualizarInventarioTest() 13ms
 - eliminarInventarioTest() 10ms
 - PagosRestControllerTest 10ms
 - PedidosRestControllerTest 88ms
 - ProductoRestControllerTest 46ms
 - SucursalesRestControllerTest 49ms
 - com.perfulandiafull.perfulandiafull.services 142ms
 - AdministradorServicesTest 12ms
 - ClienteServicesTest 9.0ms
 - EncargadoLogisticaServicesTest 13ms
 - EncargadoVentasServicesTest 12ms
 - GerenteServicesTest 20ms
 - InventarioServicesTest 14ms
 - PagosServicesTest 11ms
 - PedidosServicesTest 12ms

src > test > java > com > perfulandiafull > perfulandiafull > restControllers > J InventarioRestControllerTest.java > f InventarioRestControllerTest > M InventarioRestControllerTest > verInventario()

```

30  public class InventarioRestControllerTest {
76    public void crearInventarioTest() throws Exception{
83        .content(objectMapper.writeValueAsString(unInventario))
84        .andExpect(status().isCreated());
85    }
86
87    @Test
88    public void actualizarInventarioTest() throws Exception {
89        Inventario inventarioExistente = new Inventario(id:7L, nombre:"Inventario Antiguo", descripcion:"Dirección antigua")
90        Inventario inventarioActualizado = new Inventario(id:7L, nombre:"Inventario Nueva", descripcion:"Dirección nueva",p
91
92        when(inventarioServiceImpl.findById(id:7L)).thenReturn(Optional.of(inventarioExistente));
93        when(inventarioServiceImpl.save(any(type:Inventario.class))).thenReturn(inventarioActualizado);
94
95        mockMvc.perform(put(uriTemplate:"/api/inventario/{id}")
96            .contentType(MediaType.APPLICATION_JSON)
97            .content(objectMapper.writeValueAsString(inventarioActualizado)))
98            .andExpect(status().isOk());
99    }
100
101    @Test
102    public void eliminarInventarioTest() throws Exception {
103        Inventario inventario = new Inventario(id:8L, nombre:"Inventario a eliminar", descripcion:"Ubicación",precio:500.0)
104
105        when(inventarioServiceImpl.delete(any(type:Inventario.class)))
106            .thenReturn(Optional.of(inventario));
107
108        mockMvc.perform(delete(uriTemplate:"/api/inventario/{id}")
109            .contentType(MediaType.APPLICATION_JSON)
110            .andExpect(status().isOk()));
111    }
112
113}

```

Ln 44, Col 1 Spaces: 4 UTF-8 CRLF { }

TESTING

↻ ▶ ⏪ ⏴ ⏵ ⏷ ⏹ ⏺ ⏻ ⏸

Filter (e.g. text, !exclude, @tag)

✓ 101/101 20.6s ⏵

- ✓ () perfulandiafull 961ms
 - ✓ () { } com.perfulandiafull.perfulandiafull 2.0ms
 - ✓ () PerfulandiafullApplicationTests 2.0ms
 - ✓ () contextLoads() 2.0ms
 - ✓ () { } com.perfulandiafull.perfulandiafull.restControllers 81...
 - > () AdministradorRestController 51ms
 - > () ClienteRestController 52ms
 - > () EncargadoLogisticaRestController 53ms
 - > () EncargadoVentasRestController 57ms
 - > () GerenteRestController 53ms
 - > () InventarioRestController 58ms
 - > () PagosRestController 310ms
 - > () PedidosRestControllerTest 88ms
 - > () ProductoRestControllerTest 46ms
 - > () SucursalesRestControllerTest 49ms
- ✓ () { } com.perfulandiafull.perfulandiafull.services 142ms
 - > () AdministradorServicesTest 12ms
 - > () ClienteServicesTest 9.0ms
 - > () EncargadoLogisticaServicesTest 13ms
 - > () EncargadoVentasServicesTest 12ms
 - > () GerenteServicesTest 20ms
 - > () InventarioServicesTest 14ms
 - > () PagosServicesTest 11ms
 - > () PedidosServicesTest 12ms
 - > () ProductoServicesTest 14ms
 - > () SucursalesServicesTest 25ms

⊗ 0 △ 0 ⏵ Spring Boot-PerfulandiafullApplication<perfulandiafull> (p)

Acá se pueden apreciar todos los test de integración y los test unitarios, a los restcontrollers y servicios respectivamente. Se le aplicó testing a los 6 microservicios de perfulandia.

Postman

Get FindByAll en pedidos:

HTTP <http://localhost:8080/> | GET <http://localhost:8080/api/> | + [...](#)

HTTP <http://localhost:8080/api/pedidos>

GET <http://localhost:8080/api/pedidos>

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 214 ms Size: 921 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

```
[{"id": 2, "nombre": "Disco Duro SSD 1TB", "descripcion": "Unidad de estado sólido con velocidad de lectura hasta 500MB/s", "precio": 74990.0}, {"id": 3, "nombre": "Auriculares Inalámbricos JBL", "descripcion": "Auriculares inalámbricos con cancelación de ruido avanzada.", "precio": 129900.0}]
```

Get FindById:

HTTP <http://localhost:8080/> | GET <http://localhost:8080/api/> | + [...](#)

HTTP <http://localhost:8080/api/pedidos/2>

GET <http://localhost:8080/api/pedidos/2>

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Query Params

Key	Value
Key	Value

Body Cookies Headers (5) Test Results

Status: 200 OK Time: 32 ms Size: 299 B Save Response

Pretty Raw Preview Visualize JSON

```
1
2
3
4
5
6
```

```
{"id": 2, "nombre": "Disco Duro SSD 1TB", "descripcion": "Unidad de estado sólido con velocidad de lectura hasta 500MB/s", "precio": 74990.0}
```

**Post:**

The screenshot shows a POST request to `http://localhost:8080/api/pedidos`. The request body is a JSON object:

```
1
2
3   "nombre": "Disco",
4   "descripcion": "Bonito",
5   "precio": 20000.0
6
```

The response status is `201 Created`.

Put:

The screenshot shows a PUT request to `http://localhost:8080/api/pedidos/7`. The request body is a JSON object:

```
1
2
3   "nombre": "Disco",
4   "descripcion": "Muy Bonito",
5   "precio": 40000.0
6
```

The response status is `200 OK`.

**Delete:**

The screenshot shows a Postman interface with the following details:

- Request URL:** `http://localhost:8080/api/pedidos/7`
- Method:** `DELETE`
- Body (JSON):**

```
1
2
3     "nombre": "Disco",
4     "descripcion": "Muy Bonito",
5     "precio": 40000.0
6
```
- Response Status:** 200 OK
- Response Body (Pretty JSON):**

```
1
2     "id": 7,
3     "nombre": null,
4     "descripcion": null,
5     "precio": null
6
```

OAS

Codificación de OAS:

The screenshot shows a Java code editor with three tabs at the top: 'InventarioServicesTest.java', 'InventarioRestControllerTest.java', and 'InventarioRestController.java'. The 'InventarioRestController.java' tab is active, displaying the following code:

```
src > main > java > com > inventario > inventario > restcontrollers > InventarioRestController.java > ...
1 package com.inventario.inventario.restcontrollers;
2
3 import java.util.List;
4 import java.util.Optional;
5
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.HttpStatus;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.DeleteMapping;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.PathVariable;
12 import org.springframework.web.bind.annotation.PostMapping;
13 import org.springframework.web.bind.annotation.PutMapping;
14 import org.springframework.web.bind.annotation.RequestBody;
15 import org.springframework.web.bind.annotation.RequestMapping;
16 import org.springframework.web.bind.annotation.RestController;
17
18 import com.inventario.inventario.entities.Inventario;
19 import com.inventario.inventario.services.InventarioServices;
20 import io.swagger.v3.oas.annotations.Operation;
21 import io.swagger.v3.oas.annotations.media.Content;
22 import io.swagger.v3.oas.annotations.media.Schema;
23 import io.swagger.v3.oas.annotations.responses.ApiResponse;
24 import io.swagger.v3.oas.annotations.responses.ApiResponses;
25 import io.swagger.v3.oas.annotations.tags.Tag;
26
27 @Tag(name = "Inventario", description = "Operaciones relacionadas con el inventario")
28 @RestController
29 @RequestMapping("/api/inventario")
30 public class InventarioRestController {
31     @Autowired
32     private InventarioServices inventarioServices;
33
34     @Operation(summary = "Obtener lista de productos", description = "Devuelve todos los productos disponibles")
35     @ApiResponse(responseCode = "200", description = "Lista de productos retornada correctamente",
36                 content = @Content(mediaType = "application/json",
37                 schema = @Schema(implementation = Inventario.class)))

```

At the bottom of the editor, it says 'Application<inventario> (inventario) ⌚ Java: Ready'

añadimos los siguientes imports de swagger.

En cada método agregamos anotaciones de swagger para la documentación.



Documentación Swagger:

The screenshot shows the Swagger UI interface for the 'Inventario' endpoint. The URL in the browser is `localhost:8080/swagger-ui/index.html#/Inventario/verDetalle`. The 'Servers' dropdown is set to `http://localhost:8080 - Generated server url`.

Inventario Operaciones relacionadas con el inventario

GET /api/inventario/{id} Obtener producto por ID

Obtiene el detalle de un producto específico

Parameters

Name	Description
id * required	integer(\$int64) (path)

Responses

Curl

```
curl -X 'GET' \
'http://localhost:8080/api/inventario/4' \
-H 'accept: application/json'
```

The screenshot shows the Swagger UI interface for the 'Inventario' endpoint, similar to the previous one but with a successful curl request example.

Curl

```
curl -X 'GET' \
'http://localhost:8080/api/inventario/4' \
-H 'accept: application/json'
```

Request URL

```
http://localhost:8080/api/inventario/4
```

Server response

Code Details

200 Response body

```
{
  "id": 4,
  "nombre": "Audifonos Gamer HyperX",
  "descripcion": "Audifonos con sonido envolvente 7.1 y micrófono con cancelación de ruido",
  "cantidad": 0
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 24 Jun 2025 09:38:42 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code Description Links

200 Producto encontrado No links

Media type

application/json

Controls Accept header.

Example Value | Schema



The screenshot shows the Swagger UI interface for a RESTful API. At the top, there are two tabs: "Lista de Sucursales" and "Swagger UI". The URL in the address bar is "localhost:8080/swagger-ui/index.html#/Inventario/verDetalle".

The main content area is titled "Responses" and lists two entries:

- Code**: 200 **Description**: Producto encontrado **Links**: No links
- Code**: 404 **Description**: Producto no encontrado **Links**: No links

For the 200 response (Producto encontrado), there is a "Media type" dropdown set to "application/json". Below it, there is an "Example Value" button and a "Schema" button. The example value is shown as a JSON object:

```
{  
  "id": 0,  
  "nombre": "string",  
  "descripción": "string",  
  "cantidad": 0  
}
```

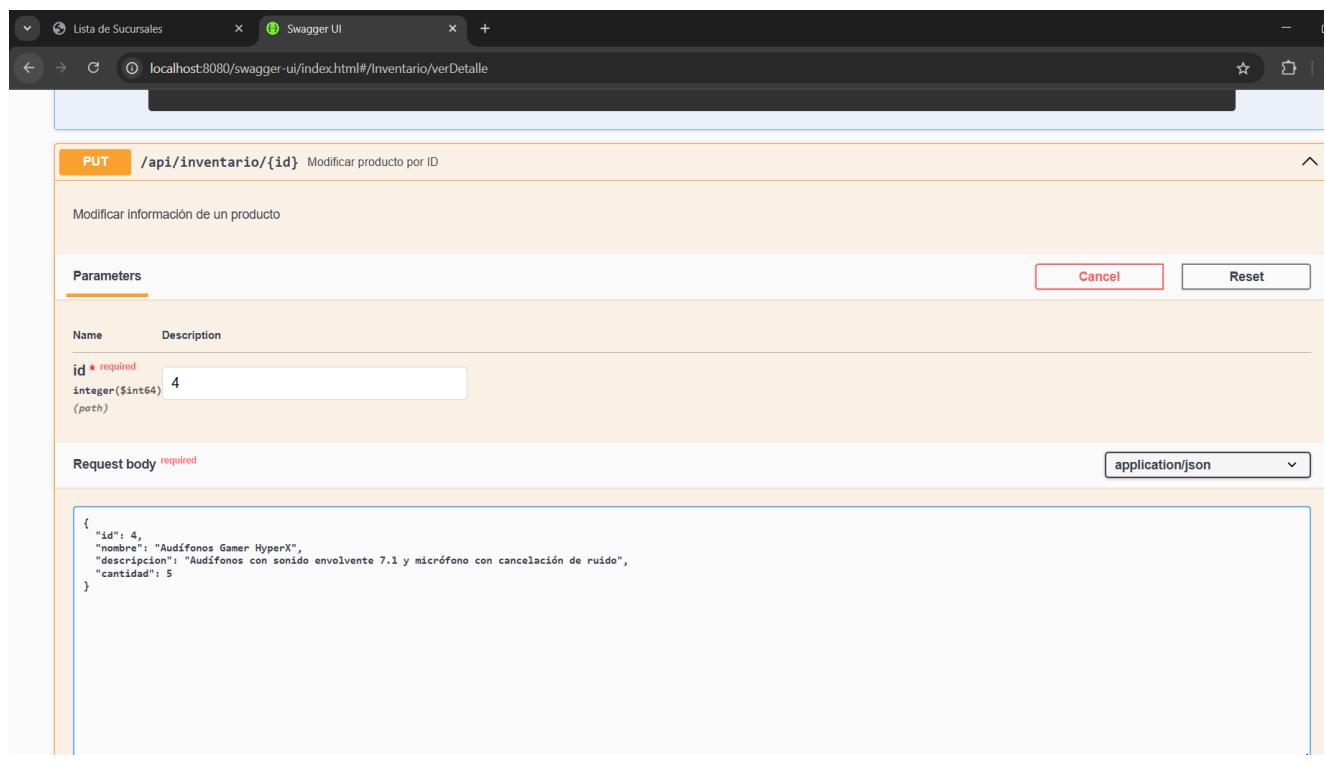
For the 404 response (Producto no encontrado), there is a "Media type" dropdown set to "text/plain". Below it, there is an "Example Value" button and a "Schema" button. The example value is shown as an empty JSON object:

```
{}
```

At the bottom, there are three API operation cards:

- PUT** /api/inventario/{id} Modificar producto por ID
- DELETE** /api/inventario/{id} Eliminar producto por ID
- GET** /api/inventario Obtener lista de productos

Aca vemos si es que está funcionando correctamente el método buscar por id.



PUT /api/inventario/{id} Modificar producto por ID

Modificar información de un producto

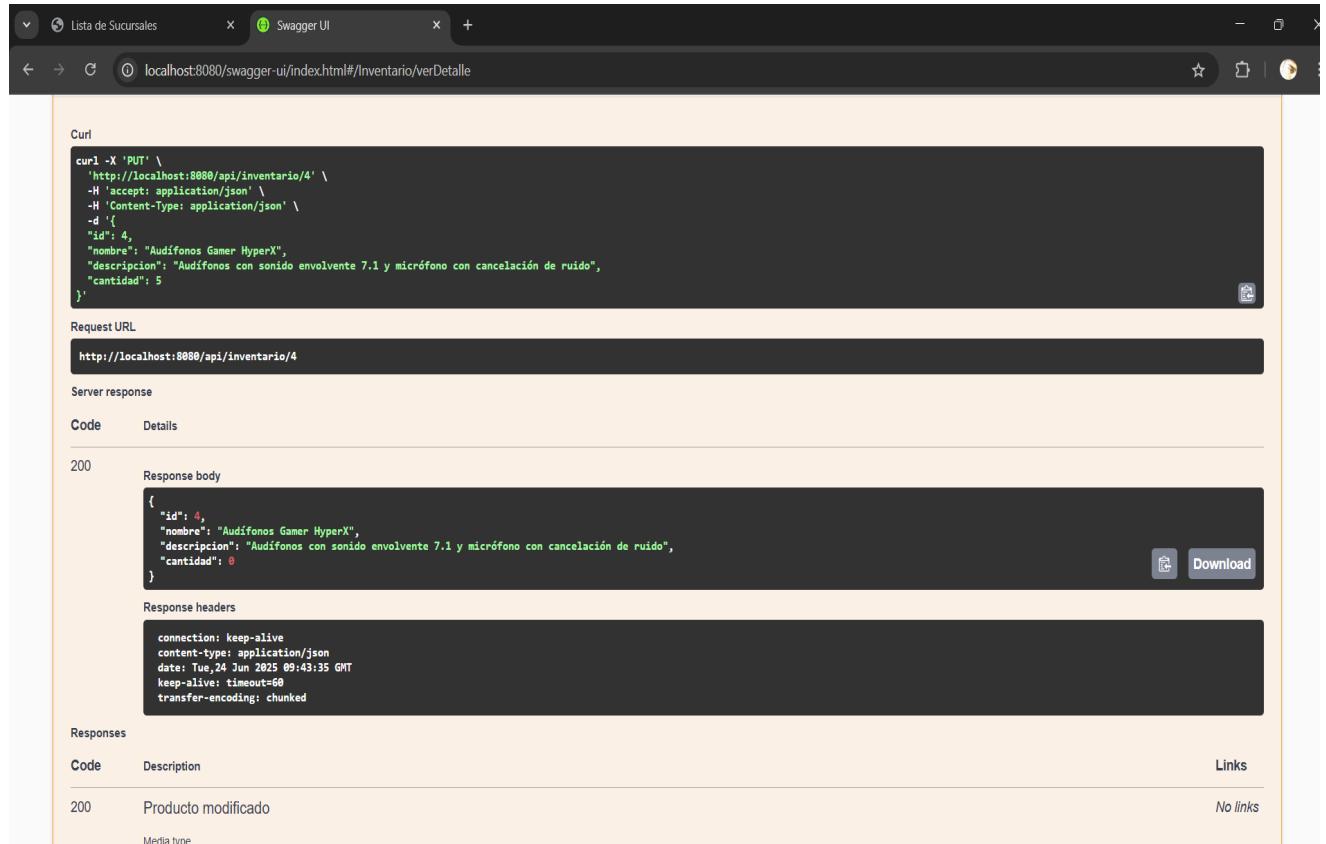
Parameters

Name	Description
id <small>required</small>	integer(\$int64) 4 (path)

Request body required

application/json

```
{
  "id": 4,
  "nombre": "Audifonos Gamer HyperX",
  "descripcion": "Audifonos con sonido envolvente 7.1 y micrófono con cancelación de ruido",
  "cantidad": 5
}
```



Curl

```
curl -X 'PUT' \
'http://localhost:8080/api/inventario/4' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "id": 4,
  "nombre": "Audifonos Gamer HyperX",
  "descripcion": "Audifonos con sonido envolvente 7.1 y micrófono con cancelación de ruido",
  "cantidad": 5
}'
```

Request URL

http://localhost:8080/api/inventario/4

Server response

Code **Details**

200

Response body

```
{
  "id": 4,
  "nombre": "Audifonos Gamer HyperX",
  "descripcion": "Audifonos con sonido envolvente 7.1 y micrófono con cancelación de ruido",
  "cantidad": 5
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 24 Jun 2025 09:43:35 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Producto modificado	No links

Aca se prueba el método Put, se cambia la cantidad de 0 a 5 y devuelve code 200.

Parameters

Name	Description
id * required integer(\$int64) (path)	4

Execute
Cancel

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:8080/api/inventario/4' \
-H 'accept: application/json'
```

Request URL

<http://localhost:8080/api/inventario/4>

Server response

Code Details

200 Response body

```
{
  "id": 4,
  "nombre": null,
  "descripcion": null,
  "cantidad": 0
}
```

Copy
Download

Lista de Sucursales Swagger UI

localhost:8080/swagger-ui/index.html#/Inventario/eliminarInventario

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost:8080/api/inventario/4' \
-H 'accept: application/json'
```

Request URL

<http://localhost:8080/api/inventario/4>

Server response

Code Details

200 Response body

```
{
  "id": 4,
  "nombre": null,
  "descripcion": null,
  "cantidad": 0
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Tue, 24 Jun 2025 09:47:05 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

Responses

Code	Description	Links
200	Producto eliminado	No links

Media type

application/json

Se prueba el método Delete, se borra el producto con id 4 y devuelvo un code 200, con valores en null.



localhost:8080/swagger-ui/index.html#/Inventario/verInventarios

Request URL
`http://localhost:8080/api/inventario`

Server response

Code Details

200 Response body

```
[  
  {  
    "id": 1,  
    "nombre": "Teclado Mecánico RGB",  
    "descripcion": "Teclado retroiluminado con switches azules y 104 teclas anti-ghosting",  
    "cantidad": 0  
  },  
  {  
    "id": 2,  
    "nombre": "Mouse Inalámbrico Logitech",  
    "descripcion": "Mouse ergonómico con conexión Bluetooth y sensor óptico de alta precisión",  
    "cantidad": 0  
  },  
  {  
    "id": 3,  
    "nombre": "Monitor LED 24\"",  
    "descripcion": "Pantalla Full HD con tecnología antirreflejo y puerto HDMI",  
    "cantidad": 0  
  }]
```

Download

Response headers

```
connection: keep-alive  
content-type: application/json  
date: Tue, 24 Jun 2025 09:49:55 GMT  
keep-alive: timeout=60  
transfer-encoding: chunked
```

Responses

Code Description Links

200 Lista de productos retornada correctamente No links

Se prueba método Get el cual retorna todos los productos.

Crea un producto con los datos proporcionados

Parameters

No parameters

Request body required

```
{
  "nombre": "Taza",
  "descripcion": "Loza de vidrio",
  "cantidad": 1
}
```

Execute Clear

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:8080/api/inventario' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "nombre": "Taza",
    "descripcion": "Loza de vidrio",
    "cantidad": 1
}'
```

Request URL

<http://localhost:8080/api/inventario>

Server response

Code	Details	Links
201	<p>Response body</p> <pre>{ "id": 5, "nombre": "Taza", "descripcion": "Loza de vidrio", "cantidad": 1 }</pre> <p>Response headers</p> <pre>connection: keep-alive content-type: application/json date: Tue, 24 Jun 2025 09:51:34 GMT keep-alive: timeout=60 transfer-encoding: chunked</pre>	 

Responses

Code	Description	Links
201	Producto creado correctamente	No links

Se prueba el método Post, creamos un producto nuevo. y devuelve un code 201.

Encargado de Logistica Operaciones relacionadas con los Encargados de Logistica

GET	/api/encargados-logistica/{id}	Obtener Encargado de Logistica por ID	▼
PUT	/api/encargados-logistica/{id}	Modificar Encargado de Logistica por ID	▼
DELETE	/api/encargados-logistica/{id}	Eliminar Encargado de Logistica por ID	▼
GET	/api/encargados-logistica	Obtener lista de Encargados de Logistica	▼
POST	/api/encargados-logistica	Crear un nuevo Encargado de Logistica	▼

Gerente Operaciones relacionadas con los gerentes

GET	/api/gerentes/{id}	Obtener Gerente por ID	▼
PUT	/api/gerentes/{id}	Modificar Gerente por ID	▼
DELETE	/api/gerentes/{id}	Eliminar Gerente por ID	▼
GET	/api/gerentes	Obtener lista de Gerentes	▼
POST	/api/gerentes	Crear un nuevo Gerente	▼

Pedidos Operaciones relacionadas con los pedidos

localhost:8080/swagger-ui/index.html#/

Pedidos Operaciones relacionadas con los pedidos

GET	/api/pedidos/{id}	Obtener Pedido por ID	▼
PUT	/api/pedidos/{id}	Modificar Pedido por ID	▼
DELETE	/api/pedidos/{id}	Eliminar Pedido por ID	▼
GET	/api/pedidos	Obtener lista de Pedido	▼
POST	/api/pedidos	Crear un nuevo Pedido	▼

Administrador Operaciones relacionadas con los Administradores

GET	/api/administradores/{id}	Obtener Administrador por ID	▼
PUT	/api/administradores/{id}	Modificar Administrador por ID	▼
DELETE	/api/administradores/{id}	Eliminar Administrador por ID	▼
GET	/api/administradores	Obtener lista de Administradores	▼
POST	/api/administradores	Crear un nuevo Administrador	▼

Productos Operaciones relacionadas con los productos



localhost:8080/swagger-ui/index.html#/

Productos

Operaciones relacionadas con los productos

GET	/api/productos/{id}	Obtener producto por ID
PUT	/api/productos/{id}	Modificar producto por ID
DELETE	/api/productos/{id}	Eliminar producto por ID
GET	/api/productos	Obtener lista de productos
POST	/api/productos	Crear un nuevo producto

Encargados de Ventas

Operaciones relacionadas con Encargados de Ventas

GET	/api/encargados-ventas/{id}	Obtener Encargado de Ventas por ID
PUT	/api/encargados-ventas/{id}	Modificar Encargado de Ventas por ID
DELETE	/api/encargados-ventas/{id}	Eliminar Encargado de Ventas por ID
GET	/api/encargados-ventas	Obtener lista de Encargados de Ventas
POST	/api/encargados-ventas	Crear un nuevo Encargado de Ventas

Pagos

Operaciones relacionadas con los pagos

GET	/api/pagos/{id}	Obtener Pago por ID
PUT	/api/pagos/{id}	Modificar Pago por ID
DELETE	/api/pagos/{id}	Eliminar Pago por ID
GET	/api/pagos	Obtener lista de pagos
POST	/api/pagos	Crear un nuevo Pago

Clientes

Operaciones relacionadas con los clientes

GET	/api/clientes/{id}	Obtener Cliente por ID
PUT	/api/clientes/{id}	Modificar Cliente por ID
DELETE	/api/clientes/{id}	Eliminar Cliente por ID
GET	/api/clientes	Obtener lista de Clientes
POST	/api/clientes	Crear un nuevo Cliente

Sucursales

Operaciones relacionadas con las sucursales

localhost:8080/swagger-ui/index.html#/

Sucursales Operaciones relacionadas con las sucursales

GET	/api/sucursales/{id}	Obtener sucursal por ID
PUT	/api/sucursales/{id}	Modificar Sucursal por ID
DELETE	/api/sucursales/{id}	Eliminar sucursal por ID
GET	/api/sucursales	Obtener lista de sucursales
POST	/api/sucursales	Crear un nuevo producto

Inventario Operaciones relacionadas con el inventario

GET	/api/inventario/{id}	Obtener producto por ID
PUT	/api/inventario/{id}	Modificar producto por ID
DELETE	/api/inventario/{id}	Eliminar producto por ID
GET	/api/inventario	Obtener lista de productos
POST	/api/inventario	Crear un nuevo producto

OAS en los 6 microservicios de perfolandia.

Tecnologías utilizadas:

Swagger/OpenAPI (springdoc-openapi)

- Es un Framework de documentación automática de APIs REST.
- ¿Cómo se usa aquí?:
 - @Tag: Agrupa los endpoints bajo una misma categoría.
 - @Operation: Describe el propósito del endpoint.
 - @ApiResponse, @ApiResponses: Detalla los posibles códigos de respuesta HTTP.
 - @Schema: Define el tipo de dato que devuelve el endpoint.



GITHUB

```
javie@GearSixth MINGW64 ~/OneDrive/Escritorio (master)
$ git init
Reinitialized existing Git repository in C:/Users/javie/OneDrive/Escritorio/.git/
javie@GearSixth MINGW64 ~/OneDrive/Escritorio (master)
$ git config user.name javier14-2
javie@GearSixth MINGW64 ~/OneDrive/Escritorio (master)
$ git config user.email javierluciano14@gmail.com
javie@GearSixth MINGW64 ~/OneDrive/Escritorio (master)
$ git config user.password ghp_tQBWt4ashaaMHhQ9R91LbNKTywQH1G3VyQTR
javie@GearSixth MINGW64 ~/OneDrive/Escritorio (master)
$ git remote add origin https://github.com/havier14-2/ET_Albornoz_Hormazabal_Neira.git|
```

Inicializamos git, configuramos nuestras credenciales. Luego vinculamos el repositorio con el local con el comando git remote add origin.

Luego se agregaran las 3 carpetas de las experiencia con git add .
Luego hacer commit con git commit -m
y finalmente git push -u origin master.

Conclusión

Durante el desarrollo de este proyecto logramos implementar de manera efectiva los distintos microservicios que componen el sistema de Perfulandia, según nuestro diagrama de microservicios. A través de las pruebas realizadas, tanto unitarias como de integración, validamos el correcto funcionamiento de las conexiones entre servicios y las operaciones CRUD.

El uso de herramientas como Spring Boot, JPA, MySQL y Git facilitó una estructura modular, ordenada y escalable. Además, nos aseguramos de seguir buenas prácticas de desarrollo, como la separación por capas, la documentación con OpenAPI y el versionamiento del código con GitHub.

Gracias a este trabajo pudimos apreciar una de las etapas más importantes de un software, el testing, para comprobar que todo esté funcionando correctamente y prevenir errores en el lanzamiento.