

Caso estudio 3: EcoMarket SPA

Vicente Alfaro
Ivan Morales
Gabriel Sanchez

ÍNDICE

1. Presentación del Caso
2. Diagramas
3. Base de Datos
4. Servicios Implementados
5. Plan de Pruebas
6. Uso de GitHub
7. Conclusión

Presentación del caso:

EcoMarket SPA es una empresa chilena en crecimiento que se dedica a la venta de productos ecológicos y sostenibles. Su primer punto de venta estuvo en el Barrio Lastarria, en Santiago, y debido a su éxito tanto en ventas como en su compromiso con la sostenibilidad, la empresa ha expandido su presencia con tiendas en Valdivia y Antofagasta. Con este crecimiento y el aumento de su base de clientes a nivel nacional, EcoMarket SPA necesita expandirse rápidamente. Sin embargo, su sistema monolítico actual está presentando problemas de rendimiento y disponibilidad, lo que está afectando tanto la eficiencia operativa como la experiencia del cliente. Estos inconvenientes representan un serio riesgo para la continuidad del negocio y la satisfacción de su clientela.

Diagrama casos de uso

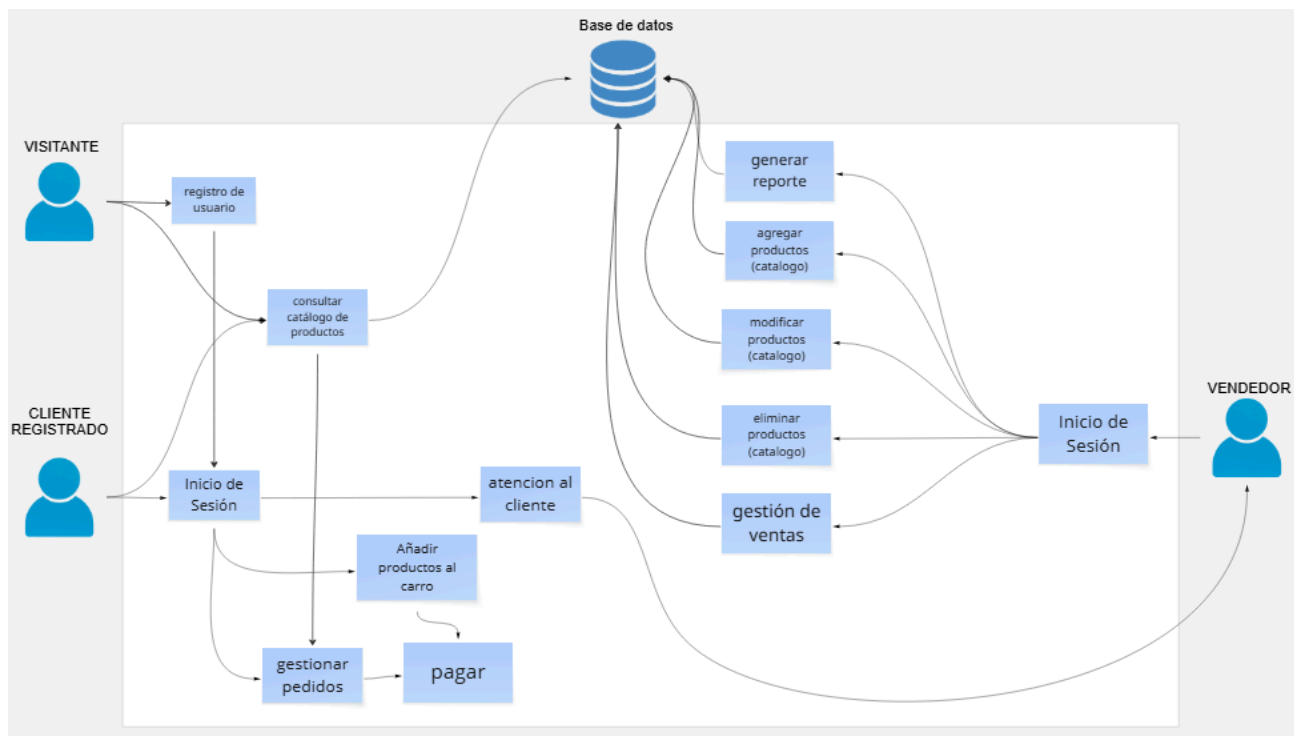


Diagrama de clases

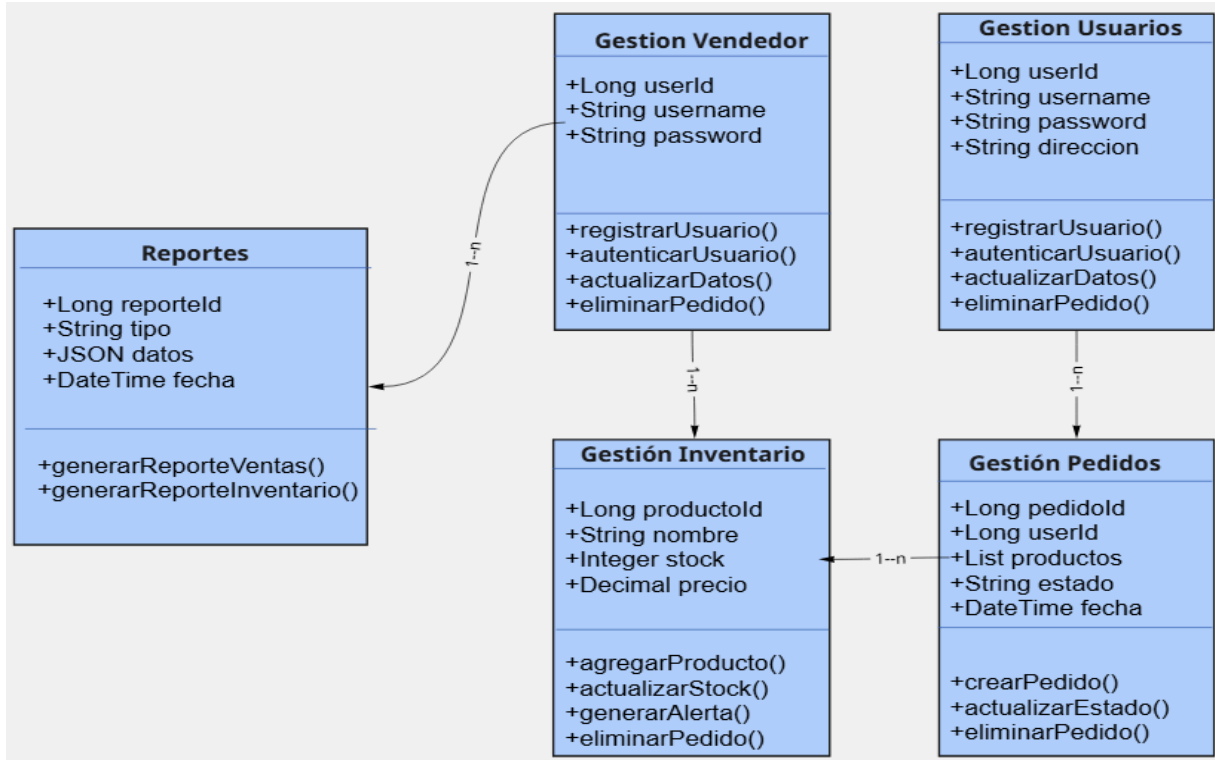
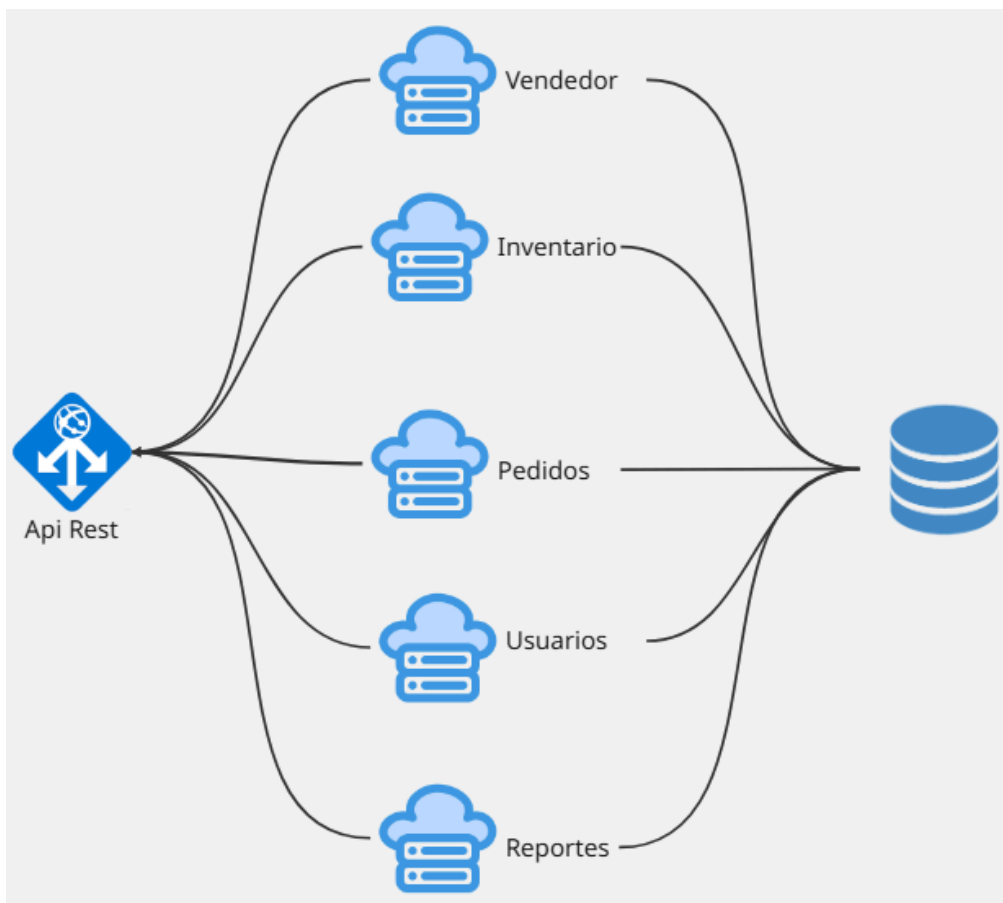


Diagrama de arquitectura de microservicios

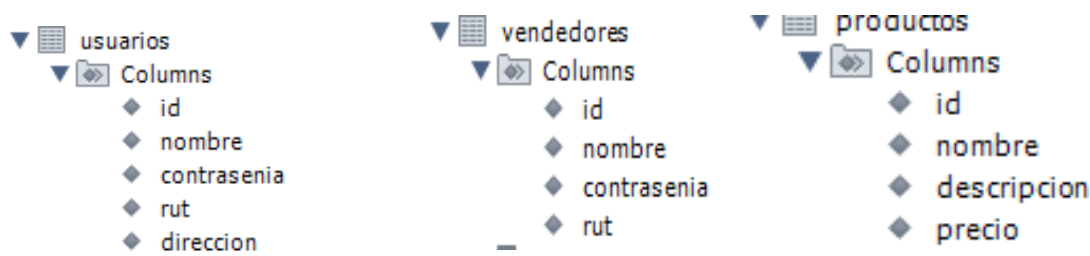


Base de Datos

Para que EcoMarket SPA haya elegido MySQL como motor de base de datos.

Estructura de la Base de Datos

La base de datos se diseñó pensando en la operatividad del negocio y en la facilidad de acceso a la información. A continuación, se presentarán las tablas:



Implementación de los Servicios

Microservicio de Gestión de Usuarios.

Crear Usuario:

POST localhost:8080/api/ecomarket/usuarios

Params Authorization Headers (8) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "nombre": "Jorge Alis",
3   "contrasenia": "1234F",
4   "direccion": "Calle siempreViva 1040",
5   "rut": "1444744-4"
6 }
7
```

Body Cookies Headers (5) Test Results 201 Created • 143 ms • 276 B

{ JSON Preview Visualize

```
1 {
2   "id": 1,
3   "rut": "1444744-4",
4   "nombre": "Jorge Alis",
5   "direccion": "Calle siempreViva 1040",
6   "contrasenia": "1234F"
7 }
```

Result Grid					
Filter Rows:					
	id	nombre	contrasenia	rut	direccion
▶	1	Jorge Alis	1234F	1444744-4	Calle siempreViva 1040
✱	NULL	NULL	NULL	NULL	NULL

Get Usuario:

GET localhost:8080/api/ecomarket/usuarios Send

Params Authorization Headers (8) Body • Scripts Settings Cookies

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK • 6 ms • 273 B

{ } JSON Preview Visualize

```

1 [
2   {
3     "id": 1,
4     "rut": "1444744-4",
5     "nombre": "Jorge Alis",
6     "direccion": "Calle siempreViva 1040",
7     "contrasenia": "1234F"
8   }
9 ]

```

Update usuarios:

PUT localhost:8080/api/ecomarket/usuarios/1 Send

Params Authorization Headers (8) Body • Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```

1 {
2   "nombre": "Jorge Curioso",
3   "contrasenia": "hola123",
4   "rut": "1444744-4",
5   "direccion": "Calle siempreviva 1040"
6 }
7
8

```

Body Cookies Headers (5) Test Results 200 OK • 34 ms • 276 B

{ } JSON Preview Visualize

```

1 {
2   "id": 1,
3   "rut": "1444744-4",
4   "nombre": "Jorge Curioso",
5   "direccion": "Calle siempreviva 1040",
6   "contrasenia": "hola123"
7 }

```

	id	nombre	contrasenia	rut	direccion
▶	1	Jorge Curioso	hola123	1444744-4	Calle siempreviva 1040
*	NULL	NULL	NULL	NULL	NULL

Delete usuarios:

DELETE

localhost:8080/apl/ecomarket/usuarios/1

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK

96 ms

233 B

{ } JSON

Preview

Visualize

1 {

2 "id": 1,

3 "rut": null,

4 "nombre": null,

5 "direccion": null,

6 "contrasenia": null

7 }

Result Grid

Filter Rows:

	id	nombre	contrasenia	rut	direccion
*	NULL	NULL	NULL	NULL	NULL

Microservicio de Gestión de Vendedores.

Agregar Vendedores:

POST localhost:8080/api/ecomarket/vendedores **Send**

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```

1 {
2   "nombre": "Mac Hernandez",
3   "contrasenia": "Macdonal123",
4   "rut": "5752916-K"
5 }

```

Body Cookies Headers (5) Test Results 201 Created • 46 ms • 248 B

{ } JSON Preview Visualize

```

1 {
2   "id": 1,
3   "nombre": "Mac Hernandez",
4   "contrasenia": "Macdonal123",
5   "rut": "5752916-K"
6 }

```

Result Grid Filter Rows: Ed

	id	nombre	contrasenia	rut
▶	1	Mac Hernandez	Macdonal123	5752916-K
*	NULL	NULL	NULL	NULL

Get Vendedores:

GET localhost:8080/api/ecomarket/vendedores **Send**

Params Authorization Headers (8) **Body** Scripts Settings Cookies

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK • 13 ms • 245 B

{ } JSON Preview Visualize

```

1 [
2   {
3     "id": 1,
4     "nombre": "Mac Hernandez",
5     "contrasenia": "Macdonal123",
6     "rut": "5752916-K"
7   }
8 ]

```


Update vendedores:

PUT localhost:8080/api/ecomarket/vendedores/1 Send

Params Authorization Headers (8) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```

1 {
2   "nombre": "Mac Hernandez",
3   "contrasenia": "BigMac1223",
4   "rut": "5752916-K"
5 }
6
7

```

Body Cookies Headers (5) Test Results 200 OK • 98 ms • 242 B • Visualize Preview Visualize

{} JSON Preview Visualize

```

1 {
2   "id": 1,
3   "nombre": "Mac Hernandez",
4   "contrasenia": "BigMac1223",
5   "rut": "5752916-K"
6 }

```

	id	nombre	contrasenia	rut
▶	1	Mac Hernandez	BigMac1223	5752916-K
✱	NULL	NULL	NULL	NULL

Delete Vendedores:

DELETE localhost:8080/api/ecomarket/vendedores/1 Send

Params Authorization Headers (8) **Body** Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 55 ms • 216 B • Visualize Preview Visualize

{} JSON Preview Visualize

```

1 {
2   "id": 1,
3   "nombre": null,
4   "contrasenia": null,
5   "rut": null
6 }

```

	id	nombre	contrasenia	rut
*	NULL	NULL	NULL	NULL

Microservicio de Gestión de Producto.

Agregar Producto:

POST

localhost:8080/api/ecomarket/productos

Send

Params

Authorization

Headers (8)

Body

Scripts

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1 {

2 "nombre": "Lasagna vegana",

3 "descripcion": "Lasagna de carne de soya",

4 "precio": 5990,

5 "stock": 25

6 }

7

8

Body

Cookies

Headers (5)

Test Results

201 Created

359 ms

269 B

{ } JSON

Preview

Visualize

1 {

2 "id": 2,

3 "nombre": "Lasagna vegana",

4 "descripcion": "Lasagna de carne de soya",

5 "precio": 5990,

6 "stock": 25

7 }

	id	nombre	descripcion	precio	strook	stock
▶	2	Lasagna vegana	Lasagna de carne de soya	5990	NULL	25
*	NULL	NULL	NULL	NULL	NULL	NULL

Get Productos:

GET localhost:8080/api/ecomarket/productos Send

Params Authorization Headers (8) Body • Scripts Settings Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK • 115 ms • 266 B

{ } JSON Preview Visualize

```

1 [
2   {
3     "id": 2,
4     "nombre": "Lasagna vegana",
5     "descripcion": "Lasagna de carne de soya",
6     "precio": 5990,
7     "stock": 25
8   }
9 ]

```

Update Productos:

PUT localhost:8080/api/ecomarket/productos/2 Send

Params Authorization Headers (8) Body • Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```

1 {
2   "nombre": "Lasagna vegana",
3   "descripcion": "Lasagna de carne de soya",
4   "precio": 6990,
5   "stock": 10
6 }
7
8

```

Body Cookies Headers (5) Test Results 200 OK • 59 ms • 264 B

{ } JSON Preview Visualize

```

1 {
2   "id": 2,
3   "nombre": "Lasagna vegana",
4   "descripcion": "Lasagna de carne de soya",
5   "precio": 6990,
6   "stock": 10
7 }

```

	id	nombre	descripcion	precio	stock	stock
▶	2	Lasagna vegana	Lasagna de carne de soya	6990	NULL	10
*	NULL	NULL	NULL	NULL	NULL	NULL

Delete Productos:



DELETE
localhost:8080/api/ecomarket/productos/2
Send

Params
Authorization
Headers (8)
Body
Scripts
Settings
Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body

200 OK • 68 ms • 226 B •  | 

{} JSON Preview Visualize

```

1  {
2    "id": 2,
3    "nombre": null,
4    "descripcion": null,
5    "precio": 0,
6    "stock": 0
7  }

```

	id	nombre	descripcion	precio	stock	stock
*	NULL	NULL	NULL	NULL	NULL	NULL

Agregamos dos nuevas Estructuras de la Base de Datos que son Reportes y Pedidos:

reportes	pedidos
Columns	Columns
id	id
descripcion	estado
fecha	fecha
id_usuario	productos_id
	user_id

Implementación de los Servicios

Microservicio de Gestión de Reportes.

Agregar Reporte:

POST localhost:8080/api/ecomarket/reportes

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON**

```

1 {
2   "descripcion": "El pedido me llego en mal estado",
3   "fecha": "21/06/2025",
4   "idUsuario": 4
5 }
6
7

```

Body Cookies Headers (5) Test Results 201 Created 39 ms

{ } JSON Preview Visualize

```

1 {
2   "id": 3,
3   "descripcion": "El pedido me llego en mal estado",
4   "fecha": "21/06/2025",
5   "idUsuario": 4
6 }

```

	id	descripcion	fecha	id_usuario
▶	1	no me llego un producto	20/06/2025	3
	3	El pedido me llego en mal estado	21/06/2025	4

Get Reportes:

GET localhost:8080/api/ecomarket/reportes

Params Authorization Headers (8) **Body** Scripts Settings

Query Params

Key	Value	Descripción
Key	Value	Descripción

Body Cookies Headers (5) Test Results 200 OK

{ } JSON Preview Visualize

```

1 [
2   {
3     "id": 1,
4     "descripcion": "no me llego un producto",
5     "fecha": "20/06/2025",
6     "idUsuario": 3
7   }
8 ]

```

Update Reportes:

PUT localhost:8080/api/ecomarket/reportes/3

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```

1  {
2    "descripcion": "El pedido me llego en mal estado y sin todos los productos",
3    "fecha": "23/06/2025",
4    "idUsuario": 4
5  }
6
7

```

Body Cookies Headers (5) Test Results **200 OK**

{ } JSON ▾ ▶ Preview Visualize ▾

```

1  {
2    "id": 3,
3    "descripcion": "El pedido me llego en mal estado y sin todos los productos",
4    "fecha": "23/06/2025",
5    "idUsuario": 4
6  }

```

	id	descripcion	fecha	id_usuario
▶	1	no me llego un producto	20/06/2025	3
	3	El pedido me llego en mal estado y sin todos los ...	23/06/2025	4

Delete Reportes:

DELETE localhost:8080/api/ecomarket/reportes/2

Params Authorization Headers (8) **Body** Scripts Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (5) Test Results ↻

{ } JSON ▾ ▶ Preview Visualize ▾

```

1  {
2    "id": 2,
3    "descripcion": null,
4    "fecha": null,
5    "idUsuario": null
6  }

```

	id	descripcion	fecha	id_usuario
▶	1	no me llego un producto	20/06/2025	3

Microservicio de Gestión de Pedidos.

Agregar Pedidos:

POST localhost:8080/api/ecomarket/pedidos

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```

1 {
2   "estado": "Preparando productos",
3   "fecha": "22/06/2025",
4   "userId": 4,
5   "productosId": "1,3,4,4"
6 }
7
8

```

Body Cookies Headers (5) Test Results **201 Created**

{ } **JSON** ▼ ▶ Preview Visualize ▼

```

1 {
2   "id": 2,
3   "userId": 4,
4   "fecha": "22/06/2025",
5   "estado": "Preparando productos",
6   "productosId": "1,3,4,4"
7 }

```

	id	estado	fecha	productosId	userId	productos_id	user_id
▶	2	Preparando productos	22/06/2025	NULL	NULL	1,3,4,4	4

Get Pedidos:

GET localhost:8080/api/ecomarket/pedidos

Params Authorization Headers (8) **Body** Scripts Settings

Query Params

	Key	Value
	Key	Value

Body Cookies Headers (5) Test Results **200 OK**

{ } **JSON** ▼ ▶ Preview Visualize ▼

```

1 [
2   {
3     "id": 2,
4     "userId": 4,
5     "fecha": "22/06/2025",
6     "estado": "Preparando productos",
7     "productosId": "1,3,4,4"
8   }
9 ]

```

Update Pedidos:

PUTlocalhost:8080/api/ecomarket/pedidos/2

ParamsAuthorizationHeaders (8)BodyScriptsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

1

2

3

4

5

6

7

8

{

"estado": "Pedido en transito",

"fecha": "22/06/2025",

"userId": 4,

"productosId": "1,3,4,4"

}

Body

Cookies

Headers (5)

Test Results

Visualize

{}

JSON

Preview

Visualize

1

2

3

4

5

6

7

{

"id": 2,

"userId": 4,

"fecha": "22/06/2025",

"estado": "Pedido en transito",

"productosId": "1,3,4,4"

}

	id	estado	fecha	productosId	userId
	2	Pedido en transito	22/06/2025	1,3,4,4	4

Delete Pedidos:

DELETElocalhost:8080/api/ecomarket/pedidos/2

ParamsAuthorizationHeaders (8)BodyScriptsSettings

Query Params

	Key	Value
	Key	Value

Body

Cookies

Headers (5)

Test Results

Visualize

{}

JSON

Preview

Visualize

1

2

3

4

5

6

7

{

"id": 2,

"userId": null,

"fecha": null,

"estado": null,

"productosId": null

}

	id	estado	fecha	productos_id	user_id
*	NULL	NULL	NULL	NULL	NULL

Plan de pruebas.

La herramienta que estamos usando es Mockito y JUnit.

Pruebas Unitarias:

```
✓ {} com.vicente.springboot.app.ecomarket.ecomarket_crud.services
> ✓ PedidoServiceImplTest 868ms
> ✓ ProductoServiceImplTest 182ms
> ✓ ReporteServiceImplTest 809ms
> ✓ UsuarioServiceImplTest 847ms
> ✓ VendedorServiceImplTest 927ms
```

Pruebas Unitarias Pedido:

```
public class PedidoServiceImplTest {

    @InjectMocks
    private PedidoServiceImpl service;

    @Mock
    private PedidoRepository repository;

    List<Pedido> list = new ArrayList<Pedido>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargePedido();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

        List<Pedido> response = service.findByAll();

        assertEquals(3, response.size());
        verify(repository, times(1)).findAll();
    }

    public void chargePedido(){
        Pedido ped1 = new Pedido(id:1L,userId:3L,fecha:"23/05/2024",estado:"Entregado",productosId:"1,5,3,2,2");
        Pedido ped2 = new Pedido(id:2L,userId:6L,fecha:"15/06/2024",estado:"En transito",productosId:"1,6,6,7");
        Pedido ped3 = new Pedido(id:3L,userId:5L,fecha:"23/06/2024",estado:"En preparacion",productosId:"1,4");

        list.add(ped1);
        list.add(ped2);
        list.add(ped3);
    }
}
```

Prueba unitaria Producto:

```
public class ProductoServiceImplTest {

    @InjectMocks
    private ProductoServiceImpl service;

    @Mock
    private ProductoRepository repository;

    List<Producto> list = new ArrayList<Producto>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargeProducto();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

        List<Producto> response = service.findByAll();

        assertEquals(3, response.size());
        verify(repository,times(1)).findAll();
    }

    public void chargeProducto(){
        Producto prod1 = new Producto(Long.valueOf(1:1),nombre:"Galletas",descripcion:"de avena",precio:1100,stock:15);
        Producto prod2 = new Producto(Long.valueOf(1:2),nombre:"Donas",descripcion:"rellenas con chocolate",precio:900,stock:10);
        Producto prod3 = new Producto(Long.valueOf(1:3),nombre:"Queque",descripcion:"de zanahoria",precio:4500,stock:8);

        list.add(prod1);
        list.add(prod2);
        list.add(prod3);
    }

}
```

Prueba unitaria Reporte:

```
public class ReporteServiceImplTest {

    @InjectMocks
    private ReporteServiceImpl service;

    @Mock
    private ReporteRepository repository;

    List<Reporte> list = new ArrayList<Reporte>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargeReportes();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

        List<Reporte> response = service.findByAll();

        assertEquals(3, response.size());
        verify(repository,times(1)).findAll();
    }

    public void chargeReportes(){
        Reporte repo1 = new Reporte(id:1L,descripcion:"Pedido no recibido",fecha:"23/05/2024",idUserario:4L);
        Reporte repo2 = new Reporte(id:2L,descripcion:"Faltaron productos del pedido",fecha:"15/06/2024",idUserario:5L);
        Reporte repo3 = new Reporte(id:3L,descripcion:"Recibi producto defectuoso",fecha:"23/06/2024",idUserario:6L);

        list.add(repo1);
        list.add(repo2);
        list.add(repo3);
    }

}
```

Prueba unitaria Usuario:

```
public class UsuarioServiceImplTest {

    @InjectMocks
    private UsuarioServiceImpl service;

    @Mock
    private UsuarioRepository repository;

    List<Usuario> list = new ArrayList<Usuario>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargeUsuario();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

        List<Usuario> response = service.findByAll();

        assertEquals(3, response.size());
        verify(repository,times(1)).findAll();
    }

    public void chargeUsuario(){
        Usuario user1 = new Usuario(Long.valueOf(1:1),rut:"21.443.332-5",nombre:"Jose Perez",direccion:"avenida hola 123",contrasenia:"PerrosyGatos12");
        Usuario user2 = new Usuario(Long.valueOf(1:2),rut:"18.334.565-9",nombre:"Lucas Ruiz",direccion:"Pajaritos 2354",contrasenia:"Temporadadepatos123");
        Usuario user3 = new Usuario(Long.valueOf(1:3),rut:"12.542.664-4",nombre:"Humberto Suazo",direccion:"planeta gol 4565",contrasenia:"Chupete2006");

        list.add(user1);
        list.add(user2);
        list.add(user3);
    }
}
```

Prueba unitaria Vendedor:

```
public class VendedorServiceImplTest {

    @InjectMocks
    private VendedorServiceImpl service;

    @Mock
    private VendedorRepository repository;

    List<Vendedor> list = new ArrayList<Vendedor>();

    @BeforeEach
    public void init(){
        MockitoAnnotations.openMocks(this);

        this.chargeVendedor();
    }

    @Test
    public void findByAllTest(){

        when(repository.findAll()).thenReturn(list);

        List<Vendedor> response = service.findByAll();

        assertEquals(3, response.size());
        verify(repository,times(1)).findAll();
    }

    public void chargeVendedor(){
        Vendedor vend1 = new Vendedor(Long.valueOf(1:1),nombre:"21.443.332-5",contrasenia:"Jose Perez",rut:"PerrosyGatos12");
        Vendedor vend2 = new Vendedor(Long.valueOf(1:2),nombre:"18.334.565-9",contrasenia:"Lucas Ruiz",rut:"Temporadadepatos123");
        Vendedor vend3 = new Vendedor(Long.valueOf(1:3),nombre:"12.542.664-4",contrasenia:"Humberto Suazo",rut:"Chupete2006");

        list.add(vend1);
        list.add(vend2);
        list.add(vend3);
    }
}
```

Prueba de Integración Usuario:

```
@SpringBootTest
@AutoConfigureMockMvc
public class UsuarioRestControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private UsuarioServiceImpl usuarioServiceImpl;

    @Mock
    private UsuarioRepository usuarioRepository;

    private List<Usuario> usuariosLista;

    @Test
    public void verUsuariosTest() throws Exception {
        when(usuarioServiceImpl.findAll()).thenReturn(usuariosLista);
        mockMvc.perform(get("/api/ecomarket/usuarios")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verUnUsuarioTest() {
        Usuario unUsuario = new Usuario(id:1L, rut:"12.546.344-6", nombre:"Humberto Suazo", direccion:"planeta gol 1243", contrasenia:"chupete2006");
        try {
            when(usuarioServiceImpl.findById(id:1L)).thenReturn(Optional.of(unUsuario));
            mockMvc.perform(get("/api/ecomarket/usuarios/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        } catch (Exception ex) {
            fail("El testing lanzo un error" + ex.getMessage());
        }
    }
}
```

```
@Test
public void crearUsuarioTest() throws Exception {
    Usuario unUsuario = new Usuario(id:null, rut:"12.546.344-6", nombre:"Humberto Suazo", direccion:"planeta gol 1243", contrasenia:"chupete2010");
    Usuario otroUsuario = new Usuario(id:5L, rut:"15.223.765-8", nombre:"Matias Fernandez", direccion:"avenida crack 3445", contrasenia:"matigol2006");
    when(usuarioServiceImpl.save(any(Usuario.class))).thenReturn(otroUsuario);
    mockMvc.perform(post("/api/ecomarket/usuarios")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(unUsuario)))
        .andExpect(status().isCreated());
}

@Test
public void eliminarUnUsuarioTest() {
    Usuario unUsuario = new Usuario(id:1L, rut:"12.546.344-6", nombre:"Humberto Suazo", direccion:"planeta gol 1243", contrasenia:"chupete2006");
    try {
        when(usuarioServiceImpl.findById(id:1L)).thenReturn(Optional.of(unUsuario));
        mockMvc.perform(delete("/api/ecomarket/usuarios/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unUsuario)))
            .andExpect(status().isNotFound());
    } catch (Exception ex) {
        fail("El testing lanzo un error" + ex.getMessage());
    }
}

@Test
void eliminarUsuarioNoExisteTest() {
    Usuario usuario = new Usuario();
    usuario.setId(id:2L);

    Mockito.when(usuarioRepository.findById(2L)).thenReturn(Optional.empty());

    Optional<Usuario> resultado = usuarioServiceImpl.delete(usuario);

    assertFalse(resultado.isPresent());
    Mockito.verify(usuarioRepository, never()).delete(any());
}
```

```
@Test
void modificarUsuarioExistenteTest() throws Exception {

    Usuario usuarioExistente = new Usuario(id:1L, rut:"12.546.344-6", nombre:"Humberto Suazo", direccion:"planeta gol 1243", contrasenia:"chupete2006");

    Usuario usuarioModificado = new Usuario(id:1L, rut:"15.223.765-8", nombre:"Matias Fernandez", direccion:"avenida crack 3445", contrasenia:"matigol2006");

    when(usuarioServiceImpl.findById(id:1L)).thenReturn(Optional.of(usuarioExistente));
    when(usuarioServiceImpl.save(any(Usuario.class))).thenReturn(usuarioModificado);

    mockMvc.perform(put("/api/ecomarket/usuarios/{id}", 1L)
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(usuarioModificado)))
        .andExpect(status().isOk());

    verify(usuarioServiceImpl).findById(id:1L);
    verify(usuarioServiceImpl).save(any(Usuario.class));
}
```

```
✓ UsuarioRestControllerTest 303ms
  ✓ verUsuariosTest() 145ms
  ✓ verUnUsuarioTest() 9.0ms
  ✓ usuarioNoExisteTest() 7.0ms
  ✓ crearUsuarioTest() 86ms
  ✓ eliminarUnUsuarioTest() 22ms
  ✓ eliminarUsuarioNoExisteTest() 12ms
  ✓ modificarUsuarioExistenteTest() 22ms
```

Prueba de Integración Vendedor:

```
@SpringBootTest
@AutoConfigureMockMvc
public class VendedorRestControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private VendedorServiceImpl vendedorServiceImpl;

    @Mock
    private VendedorRepository vendedorRepository;

    private List<Vendedor> vendedoresLista;

    @Test
    public void verVendedoresTest() throws Exception {
        when(vendedorServiceImpl.findAll()).thenReturn(vendedoresLista);
        mockMvc.perform(get("/api/ecomarket/vendedores")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verUnVendedorTest() {
        Vendedor unVendedor = new Vendedor(id:1L, nombre:"12.546.344-6", contrasenia:"Humberto Suazo", rut:"chupete2006");
        try {
            when(vendedorServiceImpl.findById(id:1L)).thenReturn(Optional.of(unVendedor));
            mockMvc.perform(get("/api/ecomarket/vendedores/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        } catch (Exception ex) {
            fail("El testing lanzo un error" + ex.getMessage());
        }
    }
}
```

```
@Test
public void vendedorNoExisteTest() throws Exception {
    when(vendedorServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
    mockMvc.perform(get("/api/ecomarket/vendedores/10")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isNotFound());
}

@Test
public void crearvendedorTest() throws Exception {
    Vendedor unVendedor = new Vendedor(id:null, nombre:"12.546.344-6", contrasenia:"Humberto Suazo", rut:"chupete2010");
    Vendedor otroVendedor = new Vendedor(id:5L, nombre:"15.223.765-8", contrasenia:"Matias Fernandez", rut:"matigol2006");
    when(vendedorServiceImpl.save(any(Vendedor.class))).thenReturn(otroVendedor);
    mockMvc.perform(post("/api/ecomarket/vendedores")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(unVendedor)))
        .andExpect(status().isCreated());
}

@Test
public void eliminarunvendedorTest() {
    Vendedor unVendedor = new Vendedor(id:1L, nombre:"12.546.344-6", contrasenia:"Humberto Suazo", rut:"chupete2006");
    try {
        when(vendedorServiceImpl.findById(id:1L)).thenReturn(Optional.of(unVendedor));
        mockMvc.perform(delete("/api/ecomarket/vendedores/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unVendedor)))
            .andExpect(status().isNotFound());
    } catch (Exception ex) {
        fail("El testing lanzo un error" + ex.getMessage());
    }
}
```

```
@Test
void eliminarvendedorNoExisteTest() {
    Vendedor vendedor = new Vendedor();
    vendedor.setId(id:2L);

    Mockito.when(vendedorRepository.findById(2L)).thenReturn(Optional.empty());

    Optional<Vendedor> resultado = vendedorServiceImpl.delete(vendedor);

    assertFalse(resultado.isPresent());
    Mockito.verify(vendedorRepository, never()).delete(any());
}

@Test
void modificarvendedorExistenteTest() throws Exception {

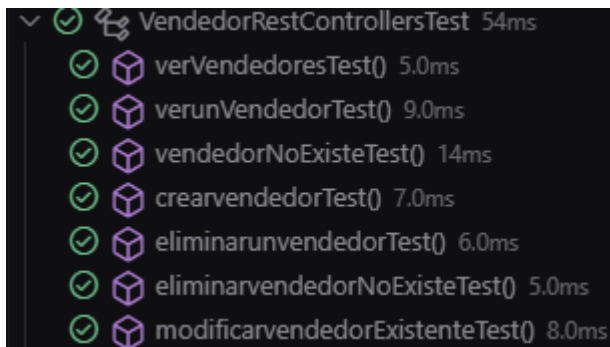
    Vendedor vendedorExistente = new Vendedor(id:1L, nombre:"12.546.344-6", contrasenia:"Humberto Suazo", rut:"chupete2006");

    Vendedor vendedorModificado = new Vendedor(id:1L, nombre:"15.223.765-8", contrasenia:"Matias Fernandez", rut:"matigol2006");

    when(vendedorServiceImpl.findById(id:1L)).thenReturn(Optional.of(vendedorExistente));
    when(vendedorServiceImpl.save(any(Vendedor.class))).thenReturn(vendedorModificado);

    mockMvc.perform(put("/api/ecomarket/vendedores/{id}", 1L)
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(vendedorModificado)))
        .andExpect(status().isOk());

    verify(vendedorServiceImpl).findById(id:1L);
    verify(vendedorServiceImpl).save(any(Vendedor.class));
}
```



A screenshot of a JUnit test runner interface showing the results for the `VendedorRestControllerTest` class. The class name is at the top with a green checkmark and a small icon. Below it, a list of test methods is shown, each with a green checkmark, a small icon, and its execution time in milliseconds. The tests are: `verVendedoresTest()` (5.0ms), `verunVendedorTest()` (9.0ms), `vendedorNoExisteTest()` (14ms), `crearvendedorTest()` (7.0ms), `eliminarunvendedorTest()` (6.0ms), `eliminarvendedorNoExisteTest()` (5.0ms), and `modificarvendedorExistenteTest()` (8.0ms).

Test Method	Execution Time
<code>verVendedoresTest()</code>	5.0ms
<code>verunVendedorTest()</code>	9.0ms
<code>vendedorNoExisteTest()</code>	14ms
<code>crearvendedorTest()</code>	7.0ms
<code>eliminarunvendedorTest()</code>	6.0ms
<code>eliminarvendedorNoExisteTest()</code>	5.0ms
<code>modificarvendedorExistenteTest()</code>	8.0ms

Prueba de Integración Productos:

```
@SpringBootTest
@AutoConfigureMockMvc
public class ProductoRestControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private ProductoServiceImpl productoServiceImpl;

    @Mock
    private ProductoRepository productoRepository;

    private List<Producto> productosLista;

    @Test
    public void verProductosTest() throws Exception{
        when(productoServiceImpl.findAll()).thenReturn(productosLista);
        mockMvc.perform(get("/api/ecomarket/productos")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verunProductoTest(){
        Producto unProducto = new Producto(id:1L,nombre:"Galletas",descripcion:"de avena",precio:1100,stock:15);
        try{
            when(productoServiceImpl.findById(id:1L)).thenReturn(Optional.of(unProducto));
            mockMvc.perform(get("/api/ecomarket/productos/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        }
        catch(Exception ex){
            fail("El testing lanzo un error" + ex.getMessage());
        }
    }
}
```

```
@Test
public void productoNoExisteTest() throws Exception{
    when(productoServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
    mockMvc.perform(get("/api/ecomarket/productos/10")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isNotFound());
}

@Test
public void crearProductoTest() throws Exception{
    Producto unProducto = new Producto(id:null,nombre:"pan amasado", descripcion:"producto chileno", precio:300,stock:10);
    Producto otroProducto = new Producto(id:4L, nombre:"pan amasado integral", descripcion:"producto chileno", precio:350,stock:8);
    when(productoServiceImpl.save(any(Producto.class))).thenReturn(otroProducto);
    mockMvc.perform(post("/api/ecomarket/productos")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(unProducto)))
        .andExpect(status().isCreated());
}

@Test
public void eliminarunProductoTest(){
    Producto unProducto = new Producto(id:1L,nombre:"pan amasado", descripcion:"producto chileno", precio:300,stock:10);
    try{
        when(productoServiceImpl.findById(id:1L)).thenReturn(Optional.of(unProducto));
        mockMvc.perform(delete("/api/ecomarket/productos/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unProducto)))
            .andExpect(status().isNotFound());
    }
    catch(Exception ex){
        fail("El testing lanzo un error" + ex.getMessage());
    }
}
```

```
@Test
void eliminarProductoNoExisteTest() {
    Producto producto = new Producto();
    producto.setId(id:2L);

    Mockito.when(productoRepository.findById(2L)).thenReturn(Optional.empty());

    Optional<Producto> resultado = productoServiceImpl.delete(producto);

    assertFalse(resultado.isPresent());
    Mockito.verify(productoRepository, never()).delete(any());
}

@Test
void modificarProductoExistenteTest() throws Exception {

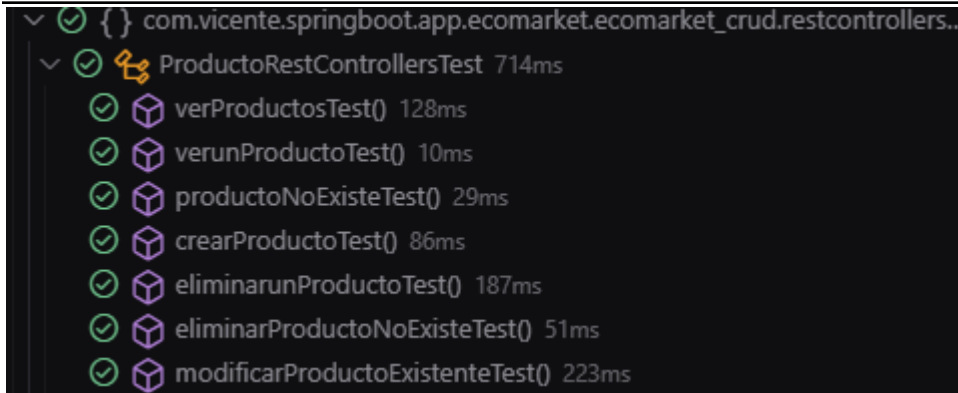
    // Producto original existente
    Producto productoExistente = new Producto(id:1L, nombre:"Platano", descripcion:"Fruta amarilla", precio:500, stock:20);

    // Producto modificado que se recibe en el body
    Producto productoModificado = new Producto(id:1L,nombre:"Manzana Verde",descripcion:"Fruta verde ácida",precio:1200,stock:30);

    // Configurar mocks
    when(productoServiceImpl.findById(id:1L)).thenReturn(Optional.of(productoExistente));
    when(productoServiceImpl.save(any(Producto.class))).thenReturn(productoModificado);

    mockMvc.perform(put("/api/ecomarket/productos/{id}",1L)
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(productoModificado)))
        .andExpect(status().isOk());

    verify(productoServiceImpl).findById(id:1L);
    verify(productoServiceImpl).save(any(Producto.class));
}
```

Prueba de Integración Reportes:

```
@SpringBootTest
@AutoConfigureMockMvc
public class ReporteRestControllerTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private ReporteServiceImpl reporteServiceImpl;

    @Mock
    private ReporteRepository reporteRepository;

    private List<Reporte> reportesLista;

    @Test
    public void verReportesTest() throws Exception{
        when(reporteServiceImpl.findAll()).thenReturn(reportesLista);
        mockMvc.perform(get("/api/ecomarket/reportes")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verunReporteTest(){
        Reporte unReporte = new Reporte(id:1L, descripcion:"Pedido nunca fue recibido", fecha:"22/06/2024", idUsuario:3L);
        try{
            when(reporteServiceImpl.findById(id:1L)).thenReturn(Optional.of(unReporte));
            mockMvc.perform(get("/api/ecomarket/reportes/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        }
        catch(Exception ex){
            fail("El testing lanzo un error" + ex.getMessage());
        }
    }
}
```

```
@Test
public void reporteNoExisteTest() throws Exception{
    when(reporteServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
    mockMvc.perform(get("/api/ecomarket/reportes/10")
        .contentType(MediaType.APPLICATION_JSON))
        .andExpect(status().isNotFound());
}

@Test
public void crearReporteTest() throws Exception{
    Reporte unReporte = new Reporte(id:null, descripcion:"Pedido nunca fue recibido", fecha:"22/06/2024", idUsuario:3L);
    Reporte otroReporte = new Reporte(id:5L, descripcion:"Pedido con productos defectuosos", fecha:"15/05/2024", idUsuario:2L);
    when(reporteServiceImpl.save(any(Reporte.class))).thenReturn(otroReporte);
    mockMvc.perform(post("/api/ecomarket/reportes")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(unReporte)))
        .andExpect(status().isCreated());
}

@Test
public void eliminarunReporteTest(){
    Reporte unReporte = new Reporte(id:1L, descripcion:"Pedido nunca fue recibido", fecha:"22/06/2024", idUsuario:3L);
    try{
        when(reporteServiceImpl.findById(id:1L)).thenReturn(Optional.of(unReporte));
        mockMvc.perform(delete("/api/ecomarket/reportes/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unReporte)))
            .andExpect(status().isNotFound());
    }
    catch(Exception ex){
        fail("El testing lanzo un error" + ex.getMessage());
    }
}
```

```
@Test
void eliminarReporteNoExisteTest() {
    Reporte reporte = new Reporte();
    reporte.setId(id:2L);

    Mockito.when(reporteRepository.findById(2L)).thenReturn(Optional.empty());

    Optional<Reporte> resultado = reporteServiceImpl.delete(reporte);

    assertFalse(resultado.isPresent());
    Mockito.verify(reporteRepository, never()).delete(any());
}

@Test
void modificarReporteExistenteTest() throws Exception {

    Reporte reporteExistente = new Reporte(id:1L, descripcion:"Pedido nunca fue recibido", fecha:"22/06/2024", idUsuario:3L);

    Reporte reporteModificado = new Reporte(id:1L, descripcion:"Pedido con producto defectuoso", fecha:"15/03/2024", idUsuario:3L);

    when(reporteServiceImpl.findById(id:1L)).thenReturn(Optional.of(reporteExistente));
    when(reporteServiceImpl.save(any(Reporte.class))).thenReturn(reporteModificado);

    mockMvc.perform(put("/api/ecomarket/reportes/{id}", 1L)
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(reporteModificado)))
        .andExpect(status().isOk());

    verify(reporteServiceImpl).findById(id:1L);
    verify(reporteServiceImpl).save(any(Reporte.class));
}
```



A screenshot of a test runner interface showing the results for the `ReporteRestControllerTest` class. The class name is at the top with a green checkmark and a link icon, followed by the total execution time of 340ms. Below it, a list of individual test methods is shown, each with a green checkmark, a cube icon, and its execution time:

- `verReportesTest()` 12ms
- `verunReporteTest()` 199ms
- `reporteNoExisteTest()` 9.0ms
- `crearReporteTest()` 9.0ms
- `eliminarunReporteTest()` 10ms
- `eliminarReporteNoExisteTest()` 5.0ms
- `modificarReporteExistenteTest()` 96ms

Prueba de Integración Pedidos:

```
@SpringBootTest
@AutoConfigureMockMvc
public class PedidoRestControllersTest {

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private ObjectMapper objectMapper;

    @MockitoBean
    private PedidoServiceImpl pedidoServiceImpl;

    @Mock
    private PedidoRepository pedidoRepository;

    private List<Pedido> pedidosLista;

    @Test
    public void verPedidosTest() throws Exception{
        when(pedidoServiceImpl.findAll()).thenReturn(pedidosLista);
        mockMvc.perform(get("/api/ecomarket/pedidos")
            .contentType(MediaType.APPLICATION_JSON))
            .andExpect(status().isOk());
    }

    @Test
    public void verunPedidoTest(){
        Pedido unPedido = new Pedido(id:1L,userId:4L,fecha:"25/04/2024",estado:"Entregado",productosId:"1,3,3,6,7");
        try{
            when(pedidoServiceImpl.findById(id:1L)).thenReturn(Optional.of(unPedido));
            mockMvc.perform(get("/api/ecomarket/pedidos/1")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
        }
        catch(Exception ex){
            fail("El testing lanzo un error" + ex.getMessage());
        }
    }
}
```

```

@Test
public void pedidoNoExisteTest() throws Exception{
    when(pedidoServiceImpl.findById(id:10L)).thenReturn(Optional.empty());
    mockMvc.perform(get("/api/ecomarket/pedidos/10")
        .contentType(MediaType.APPLICATION_JSON)
        .andExpect(status().isNotFound());
    }

@Test
public void crearPedidoTest() throws Exception{
    Pedido unPedido = new Pedido(id:null,userId:4L,fecha:"25/04/2024",estado:"Entregado",productosId:"1,3,3,6,7");
    Pedido otroPedido = new Pedido(id:5L,userId:2L,fecha:"12/02/2024",estado:"Entregado",productosId:"1,2,5");
    when(pedidoServiceImpl.save(any(Pedido.class))).thenReturn(otroPedido);
    mockMvc.perform(post("/api/ecomarket/pedidos")
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(unPedido)))
        .andExpect(status().isCreated());
    }

@Test
public void eliminarunPedidoTest(){
    Pedido unPedido = new Pedido(id:1L,userId:4L,fecha:"25/04/2024",estado:"Entregado",productosId:"1,3,3,6,7");
    try{
        when(pedidoServiceImpl.findById(id:1L)).thenReturn(Optional.of(unPedido));
        mockMvc.perform(delete("/api/ecomarket/pedidos/1")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(unPedido)))
            .andExpect(status().isNotFound());
    }
    catch(Exception ex){
        fail("El testing lanzo un error" + ex.getMessage());
    }
}

```

```

@Test
void eliminarPedidoNoExisteTest() {
    Pedido pedido = new Pedido();
    pedido.setId(id:2L);

    Mockito.when(pedidoRepository.findById(2L)).thenReturn(Optional.empty());

    Optional<Pedido> resultado = pedidoServiceImpl.delete(pedido);

    assertFalse(resultado.isPresent());
    Mockito.verify(pedidoRepository, never()).delete(any());
}

@Test
void modificarPedidoExistenteTest() throws Exception {

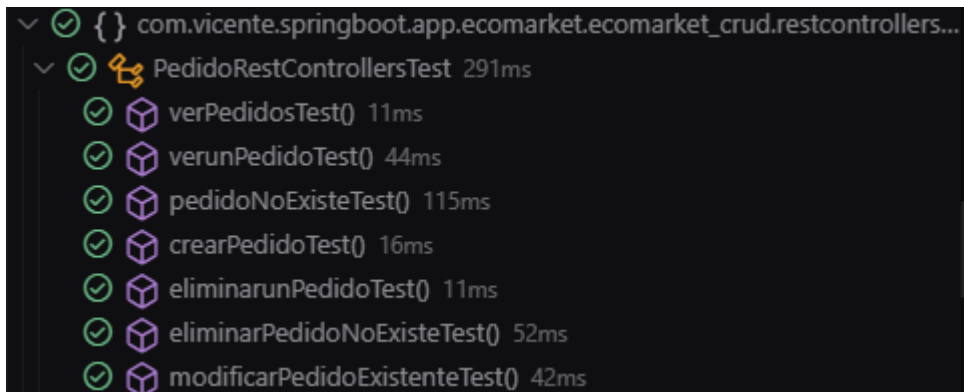
    Pedido pedidoExistente = new Pedido(id:1L,userId:4L,fecha:"25/04/2024",estado:"Entregado",productosId:"1,3,3,6,7");
    Pedido pedidoModificado = new Pedido(id:1L,userId:2L,fecha:"12/02/2024",estado:"Entregado",productosId:"1,2,5");

    when(pedidoServiceImpl.findById(id:1L)).thenReturn(Optional.of(pedidoExistente));
    when(pedidoServiceImpl.save(any(Pedido.class))).thenReturn(pedidoModificado);

    mockMvc.perform(put("/api/ecomarket/pedidos/{id}",1L)
        .contentType(MediaType.APPLICATION_JSON)
        .content(objectMapper.writeValueAsString(pedidoModificado)))
        .andExpect(status().isOk());

    verify(pedidoServiceImpl).findById(id:1L);
    verify(pedidoServiceImpl).save(any(Pedido.class));
}

```



Documentación de las pruebas ejecutadas.

Documentación Usuario:

```
@Operation(summary = "Muestra todos los usuarios creados")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Obtencion de usuarios exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudieron obtener los usuarios")
})
@GetMapping
public List<Usuario> List(){
    return service.findByAll();
}

@Operation(summary = "Muestra un usuario en especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "obtencion de usuario exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400", description = "Id de usuario no encontrada")
})
@GetMapping("/{id}")
public ResponseEntity<?> verDetalle(@PathVariable Long id){
    Optional<Usuario> usuarioOptional = service.findById(id);
    if(usuarioOptional.isPresent()){
        return ResponseEntity.ok(usuarioOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Crea un nuevo usuario")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Usuario creado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudo crear el usuario")
})
@PostMapping
public ResponseEntity<Usuario> crear(@RequestBody Usuario unUsuario){
    return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unUsuario));
}

@Operation(summary = "Actualiza los datos de un usuario creado")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "datos actualizados correctamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se encontro el usuario a modificar")
})
@PutMapping("/{id}")
public ResponseEntity<?> modificar(@PathVariable Long id, @RequestBody Usuario unUsuario){
    Optional<Usuario> usuarioOptional = service.findById(id);
    if(usuarioOptional.isPresent()){
        Usuario usuarioExistente = usuarioOptional.get();
        usuarioExistente.setNombre(unUsuario.getNombre());
        usuarioExistente.setDireccion(unUsuario.getDireccion());
        usuarioExistente.setContrasenia(unUsuario.getContrasenia());
        usuarioExistente.setRut(unUsuario.getRut());
        Usuario usuarioModificado = service.save(usuarioExistente);
        return ResponseEntity.ok(usuarioModificado);
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Elimina un usuario especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Usuario eliminado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "Usuario no encontrado, no se pudo eliminar")
})
@DeleteMapping("/{id}")
public ResponseEntity<?> eliminar(@PathVariable Long id){
    Usuario unUsuario = new Usuario();
    unUsuario.setId(id);
    Optional<Usuario> usuarioOptional = service.delete(unUsuario);
    if(usuarioOptional.isPresent()){
        return ResponseEntity.ok(usuarioOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

Documentación Vendedor:

```
@Operation(summary = "Muestra todos los vendedores creados")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Obtencion de vendedores exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudieron obtener los vendedores")
})
@GetMapping
public List<Vendedor> List(){
    return service.findByAll();
}

@Operation(summary = "Muestra un vendedor en especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200",description = "obtencion de vendedor exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400",description = "Id de vendedor no encontrada")
})
@GetMapping("/{id}")
public ResponseEntity<> verDetalle(@PathVariable Long id){
    Optional<Vendedor> vendedorOptional = service.findById(id);
    if(vendedorOptional.isPresent()){
        return ResponseEntity.ok(vendedorOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Crea un nuevo vendedor")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Vendedor creado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudo crear el vendedor")
})
@PostMapping
public ResponseEntity<Vendedor> crear(@RequestBody Vendedor unVendedor){
    return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unVendedor));
}

@Operation(summary = "Actualiza los datos de un vendedor creado")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "datos actualizados correctamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se encontro el vendedor a modificar")
})
@PutMapping("/{id}")
public ResponseEntity<?> modificar(@PathVariable Long id, @RequestBody Vendedor unVendedor){
    Optional<Vendedor> vendedorOptional = service.findById(id);
    if(vendedorOptional.isPresent()){
        Vendedor vendedorExistente = vendedorOptional.get();
        vendedorExistente.setNombre(unVendedor.getNombre());
        vendedorExistente.setContrasenia(unVendedor.getContrasenia());
        vendedorExistente.setRut(unVendedor.getRut());
        Vendedor vendedorModificado = service.save(vendedorExistente);
        return ResponseEntity.ok(vendedorModificado);
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Elimina un vendedor especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Vendedor eliminado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "Vendedor no encontrado, no se pudo eliminar")
})
@DeleteMapping("/{id}")
public ResponseEntity<?> eliminar(@PathVariable Long id){
    Vendedor unVendedor = new Vendedor();
    unVendedor.setId(id);
    Optional<Vendedor> vendedorOptional = service.delete(unVendedor);
    if(vendedorOptional.isPresent()){
        return ResponseEntity.ok(vendedorOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

Documentación Producto:


```
@Operation(summary = "Muestra todos los productos creados")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Obtencion de productos exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudieron obtener los productos")
})
@GetMapping
public List<Producto> List(){
    return service.findByAll();
}

@Operation(summary = "Muestra un producto en especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200",description = "obtencion de producto exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400",description = "Id de producto no encontrada")
})
@GetMapping("/{id}")
public ResponseEntity<?> verDetalle(@PathVariable Long id){
    Optional<Producto> productoOptional = service.findById(id);
    if(productoOptional.isPresent()){
        return ResponseEntity.ok(productoOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Crea un nuevo producto")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Producto creado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudo crear el producto")
})
@PostMapping
public ResponseEntity<Producto> crear(@RequestBody Producto unProducto){
    return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unProducto));
}

@Operation(summary = "Actualiza los datos de un producto creado")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "datos actualizados correctamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se encontro el producto a modificar")
})
@PutMapping("/{id}")
public ResponseEntity<?> modificar(@PathVariable Long id, @RequestBody Producto unProducto){
    Optional<Producto> productoOptional = service.findById(id);
    if(productoOptional.isPresent()){
        Producto productoExistente = productoOptional.get();
        productoExistente.setNombre(unProducto.getNombre());
        productoExistente.setDescripcion(unProducto.getDescripcion());
        productoExistente.setPrecio(unProducto.getPrecio());
        productoExistente.setStock(unProducto.getStock());
        Producto productoModificado = service.save(productoExistente);
        return ResponseEntity.ok(productoModificado);
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Elimina un producto especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Producto eliminado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "Producto no encontrado, no se pudo eliminar")
})
@DeleteMapping("/{id}")
public ResponseEntity<?> eliminar(@PathVariable Long id){
    Producto unProducto = new Producto();
    unProducto.setId(id);
    Optional<Producto> productoOptional = service.delete(unProducto);
    if(productoOptional.isPresent()){
        return ResponseEntity.ok(productoOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

Documentación Reporte:

```
@Operation(summary = "Muestra todos los reportes creados")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Obtencion de reportes exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudieron obtener los reportes")
})
@GetMapping
public List<Reporte> List(){
    return service.findAll();
}

@Operation(summary = "Muestra un reporte en especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "obtencion de reporte exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400", description = "Id de reporte no encontrada")
})
@GetMapping("/{id}")
public ResponseEntity<> verDetalle(@PathVariable Long id){
    Optional<Reporte> reporteOptional = service.findById(id);
    if(reporteOptional.isPresent()){
        return ResponseEntity.ok(reporteOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Crea un nuevo reporte")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Reporte creado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudo crear el reporte")
})
@PostMapping
public ResponseEntity<Reporte> crear(@RequestBody Reporte unReporte){
    return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unReporte));
}

@Operation(summary = "Actualiza los datos de un reporte creado")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "datos actualizados correctamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se encontro el reporte a modificar")
})
@PutMapping("/{id}")
public ResponseEntity<> modificar(@PathVariable Long id, @RequestBody Reporte unReporte){
    Optional<Reporte> reporteOptional = service.findById(id);
    if(reporteOptional.isPresent()){
        Reporte reporteExistente = reporteOptional.get();
        reporteExistente.setDescripcion(unReporte.getDescripcion());
        reporteExistente.setFecha(unReporte.getFecha());
        reporteExistente.setIdUsuario(unReporte.getIdUsuario());
        Reporte reporteModificado = service.save(reporteExistente);
        return ResponseEntity.ok(reporteModificado);
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Elimina un reporte especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Reporte eliminado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "Reporte no encontrado, no se pudo eliminar")
})
>DeleteMapping("/{id}")
public ResponseEntity<?> eliminar(@PathVariable Long id){
    Reporte unReporte = new Reporte();
    unReporte.setId(id);
    Optional<Reporte> reporteOptional = service.delete(unReporte);
    if(reporteOptional.isPresent()){
        return ResponseEntity.ok(reporteOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

Documentación Pedido:

```
@Operation(summary = "Muestra todos los pedidos creados")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Obtencion de pedidos exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudieron obtener los pedidos")
})
@GetMapping
public List<Pedido> List(){
    return service.findAll();
}

@Operation(summary = "Muestra un pedido en especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200",description = "obtencion de pedido exitosa",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = "20"))
    ),
    @ApiResponse(responseCode = "400",description = "Id de pedido no encontrada")
})
@GetMapping("/{id}")
public ResponseEntity<?> verDetalle(@PathVariable Long id){
    Optional<Pedido> pedidoOptional = service.findById(id);
    if(pedidoOptional.isPresent()){
        return ResponseEntity.ok(pedidoOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Crea un nuevo pedido")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Pedido creado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se pudo crear el pedido")
})
@PostMapping
public ResponseEntity<Pedido> crear(@RequestBody Pedido unPedido){
    return ResponseEntity.status(HttpStatus.CREATED).body(service.save(unPedido));
}

@Operation(summary = "Actualiza los datos de un pedido creado")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "datos actualizados correctamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "No se encontro el pedido a modificar")
})
@PutMapping("/{id}")
public ResponseEntity<> modificar(@PathVariable Long id, @RequestBody Pedido unPedido){
    Optional<Pedido> pedidoOptional = service.findById(id);
    if(pedidoOptional.isPresent()){
        Pedido pedidoExistente = pedidoOptional.get();
        pedidoExistente.setEstado(unPedido.getEstado());
        pedidoExistente.setFecha(unPedido.getFecha());
        pedidoExistente.setProductosId(unPedido.getProductosId());
        pedidoExistente.setUserId(unPedido.getUserId());
        Pedido pedidoModificado = service.save(pedidoExistente);
        return ResponseEntity.ok(pedidoModificado);
    }
    return ResponseEntity.notFound().build();
}
```

```
@Operation(summary = "Elimina un pedido especifico")
@ApiResponses(value = {
    @ApiResponse(responseCode = "200", description = "Pedido eliminado exitosamente",
        content = @Content(mediaType = "text/plain",
            examples = @ExampleObject(value = ""))
    ),
    @ApiResponse(responseCode = "400", description = "Pedido no encontrado, no se pudo eliminar")
})
>DeleteMapping("/{id}")
public ResponseEntity<?> eliminar(@PathVariable Long id){
    Pedido unPedido = new Pedido();
    unPedido.setId(id);
    Optional<Pedido> pedidoOptional = service.delete(unPedido);
    if(pedidoOptional.isPresent()){
        return ResponseEntity.ok(pedidoOptional.orElseThrow());
    }
    return ResponseEntity.notFound().build();
}
```

Git Hub:

Inicialización de git, configuración de usuario y sincronización con el repositorio en GitHub, preparar archivos para subir al repositorio.

```
$ git config user.name Vicente

vicho@DESKTOP-7SN150U MINGW64 /d/Codes/Fullstack/ecomarket-crud (main)
$ git config user.email vi.alfaros@duocuc.cl

vicho@DESKTOP-7SN150U MINGW64 /d/Codes/Fullstack/ecomarket-crud (main)
$ git config user.password ghp_jUFwze1B9wYDTpkaEYh8S4vecwQsSU4Yhg9

vicho@DESKTOP-7SN150U MINGW64 /d/Codes/Fullstack/ecomarket-crud (main)
$ git remote add origin https://github.com/VicenteAlfaroS/Experiencia-1-Caso-

vicho@DESKTOP-7SN150U MINGW64 /d/Codes/Fullstack/ecomarket-crud (main)
$ git add .
```

Push para subir los archivos a github:

```
vicho@DESKTOP-7SN150U MINGW64 /d/Codes/Fullstack/ecomarket-crud (main)
$ git push --force origin main
```

Checkout es navegar entre las ramas creadas por branch:

```
ivanj@DESKTOP-93AV4Q4 MINGW64 ~/Desktop/Uwu (master)
$ git checkout -b ivan origin/Ivan
branch 'ivan' set up to track 'origin/Ivan'.
Switched to a new branch 'ivan'
```

Add para preparar los archivos para subirlos:

```
ivanj@DESKTOP-93AV4Q4 MINGW64 ~/Desktop/Uwu (ivan)
$ git add .
```

Commit para guardar cambios:

```
ivanj@DESKTOP-93AV4Q4 MINGW64 ~/Desktop/Uwu (ivan)
$ git commit -m "Prueba Subir Reportes"
[ivan 68380bb] Prueba Subir Reportes
5 files changed, 204 insertions(+)
create mode 100644 src/main/java/com/vicente/springboot/app/ecomarket/ecomarket_crud/controllers/ReporteController.java
create mode 100644 src/main/java/com/vicente/springboot/app/ecomarket/ecomarket_crud/entities/Reporte.java
create mode 100644 src/main/java/com/vicente/springboot/app/ecomarket/ecomarket_crud/repository/ReporteRepository.java
create mode 100644 src/main/java/com/vicente/springboot/app/ecomarket/ecomarket_crud/services/ReporteService.java
create mode 100644 src/main/java/com/vicente/springboot/app/ecomarket/ecomarket_crud/services/ReporteServiceImpl.java
```

Push para subir los cambios a github:

```
ivanj@DESKTOP-93AV4Q4 MINGW64 ~/Desktop/Uwu (ivan)
$ git push -u origin ivan
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (20/20), 2.99 KiB | 765.00 KiB/s, done.
Total 20 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 4 local objects.
remote:
remote: Create a pull request for 'ivan' on GitHub by visiting:
remote:   https://github.com/VicenteAlfaroS/Exp3_Alfaro_Morales_sanchez/pull/new/ivan
remote:
remote: GitHub found 3 vulnerabilities on VicenteAlfaroS/Exp3_Alfaro_Morales_sanchez's default branch (1 high, 2 moderate). To find out more, visit:
remote:   https://github.com/VicenteAlfaroS/Exp3_Alfaro_Morales_sanchez/security/dependabot
remote:
To https://github.com/VicenteAlfaroS/Exp3_Alfaro_Morales_sanchez
 * [new branch]      ivan -> ivan
branch 'ivan' set up to track 'origin/ivan'.
```

Conclusión

EcoMarket SPA ha crecido rápidamente, pero su sistema monolítico ya no es suficiente para manejar la demanda y garantizar un buen servicio a sus clientes. La migración a microservicios permitirá mejorar el rendimiento, la escalabilidad y la estabilidad del sistema, asegurando una experiencia más fluida para los usuarios y una operación eficiente para la empresa. Aunque la transición implica desafíos, con una planificación cuidadosa y herramientas adecuadas, EcoMarket SPA podrá adaptarse al crecimiento y fortalecer su presencia en el mercado.