

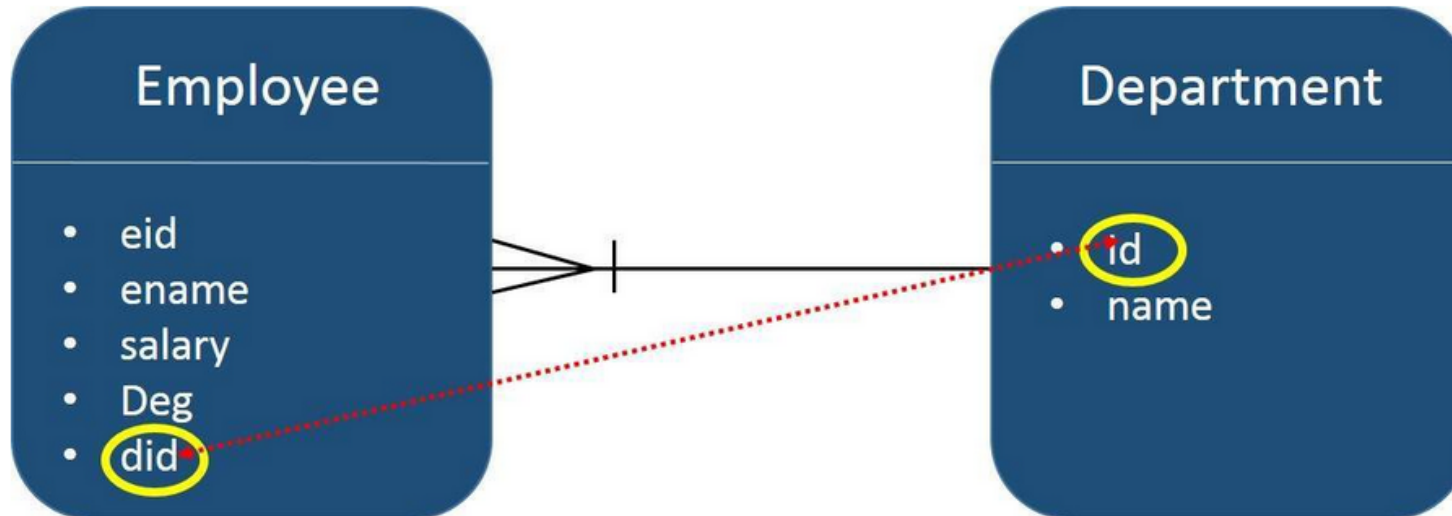
# Relaciones entre entitys

- ▶ Las clases de entidad son tratadas como tablas relacionales (concepto de JPA), por lo tanto, las relaciones entre las clases de entidad son los siguientes:
- @ManyToOne Relation: se hace referencia a una entidad (columna o conjunto de columnas) con valores únicos que contienen de otra entidad (columna o conjunto de columnas).
- @OneToMany Relation: cada fila de una entidad se hace referencia a los muchos registros secundarios en otra entidad
- @OneToOne Relation: Significa que cada fila de una entidad se refiere a una y sólo una fila de otra entidad.
- @ManyToMany Relation: Relación de varios a varios es donde una o más filas de una entidad se asocian a más de una fila en otra entidad.

# @ManyToOne- Muchos a uno

Nos permite considerar un ejemplo de una relación entre entidades empleado y departamento. De manera unidireccional, es decir, de empleado al Departamento, Many-To-One relación es aplicable. Eso significa que cada registro de empleado contiene un id de departamento, que debe ser una clave principal en la tabla Department. Aquí en la tabla Employee, Departamento id es la clave foránea.

El siguiente diagrama muestra el Many-To-One relación entre las dos tablas.



```
import javax.persistence.Id;

@Entity
public class Department
{
    @Id
    @GeneratedValue( strategy=GenerationType.AUTO )
    private int id;
    private String name;

    public int getId()
    {
        return id;
    }

    public void setId(int id)
    {
        this.id = id;
    }

    public String getName( )
    {
        return name;
    }

    public void setName( String deptName )
    {
        this.name = deptName;
    }
}
```

```
import javax.persistence.*;

@Entity
public class Employee
{
    @Id
    @GeneratedValue( strategy= GenerationType.AUTO )
    private int eid;
    private String ename;
    private double salary;
    private String deg;

    @ManyToOne
    private Department department;

    public Employee(int eid, String ename, double salary, String deg)
    {
        super( );
        this.eid = eid;
        this.ename = ename;
        this.salary = salary;
        this.deg = deg;
    }

    public Employee( )
    {
        super();
    }

    public int getEid( )
    {
        return eid;
    }

    public void setEid(int eid)
```

# @OneToMany-Uno a muchos

```
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Department
{
    @Id
    @GeneratedValue( strategy=GenerationType.AUTO )
    private int id;
    private String name;

    @OneToMany( targetEntity=Employee.class )
    private List employeeList;

    public int getId()
    {
        return id;
    }

    public void setId(int id)
    {
        this.id = id;
    }

    public String getName( )
    {
        return name;
    }

    public void setName( String deptName )
    {
        this.name = deptName;
    }

    public List getEmployeeList()
```

```
@Entity
public class Employee
{
    @Id
    @GeneratedValue( strategy= GenerationType.AUTO )
    private int eid;
    private String ename;
    private double salary;
    private String deg;

    public Employee(int eid, String ename, double salary, String deg)
    {
        super( );
        this.eid = eid;
        this.ename = ename;
        this.salary = salary;
        this.deg = deg;
    }

    public Employee( )
    {
        super();
    }

    public int getEid( )
    {
        return eid;
    }
    public void setEid(int eid)
    {
        this.eid = eid;
    }
}
```

# @OneToOne - uno a uno

```
@Entity
public class Department
{
    @Id
    @GeneratedValue( strategy=GenerationType.AUTO )
    private int id;
    private String name;

    public int getId()
    {
        return id;
    }

    public void setId(int id)
    {
        this.id = id;
    }

    public String getName( )
    {
        return name;
    }

    public void setName( String deptName )
    {
        this.name = deptName;
    }
}
```

```
import javax.persistence.OneToOne;

@Entity
public class Employee
{
    @Id
    @GeneratedValue( strategy= GenerationType.AUTO )
    private int eid;
    private String ename;
    private double salary;
    private String deg;

    @OneToOne
    private Department department;

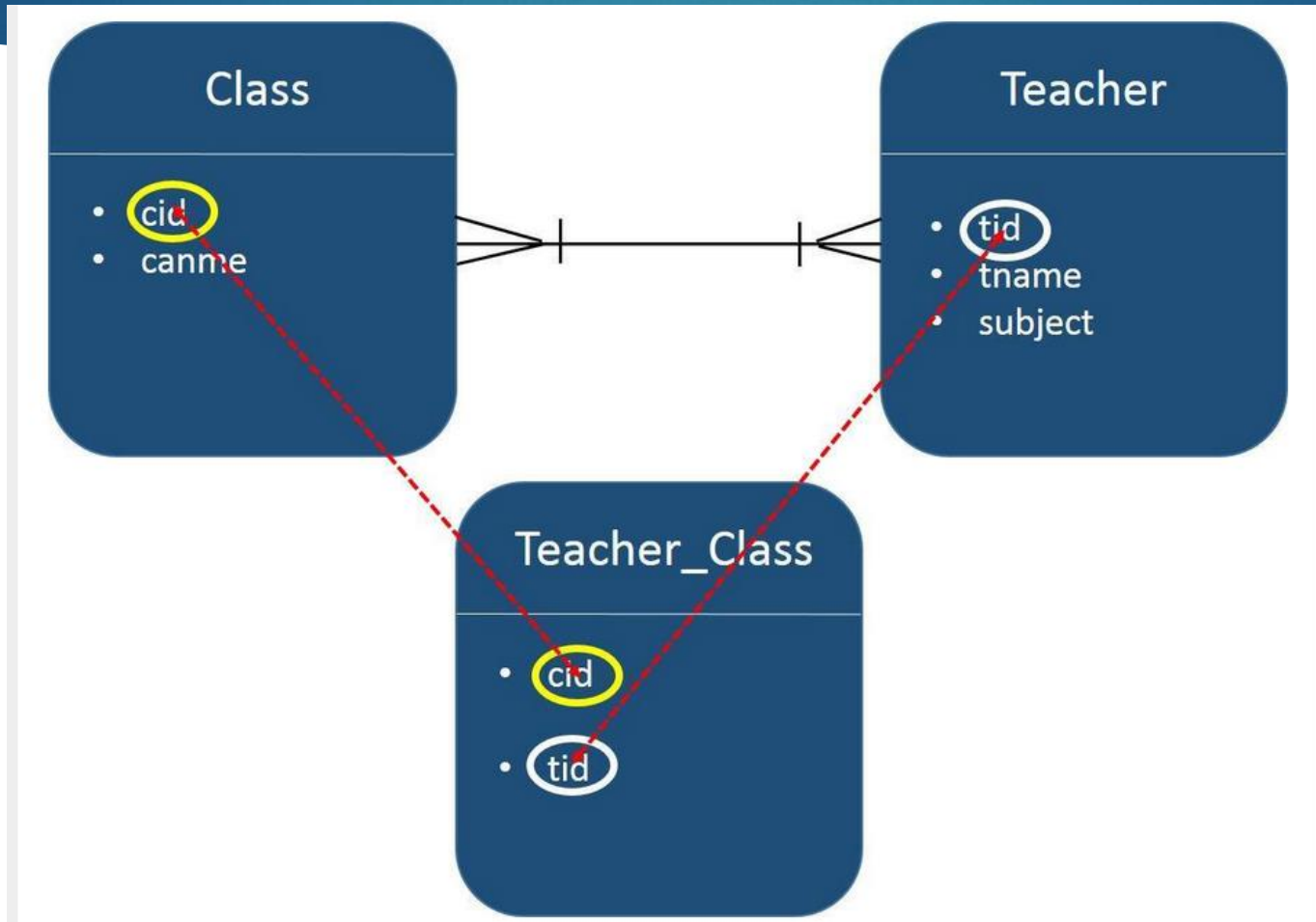
    public Employee(int eid, String ename, double salary, String deg)
    {
        super( );
        this.eid = eid;
        this.ename = ename;
        this.salary = salary;
        this.deg = deg;
    }

    public Employee( )
    {
        super();
    }

    public int getEid( )
    {
        return eid;
    }
}
```



# @ManyToMany - muchos a muchos



```
import javax.persistence.ManyToMany;

@Entity
public class Clas
{
    @Id
    @GeneratedValue( strategy = GenerationType.AUTO )
    private int cid;
    private String cname;

    @ManyToMany(targetEntity=Teacher.class)
    private Set teacherSet;

    public Clas()
    {
        super();
    }

    public Clas(int cid, String cname, Set teacherSet)
    {
        super();
        this.cid = cid;
        this.cname = cname;
        this.teacherSet = teacherSet;
    }

    public int getCid()
    {
        return cid;
    }

    public void setCid(int cid)
    {
        this.cid = cid;
    }
}
```

```
@Entity
public class Teacher
{
    @Id
    @GeneratedValue( strategy = GenerationType.AUTO )
    private int tid;
    private String tname;
    private String subject;

    @ManyToMany(targetEntity=Clas.class)
    private Set clasSet;

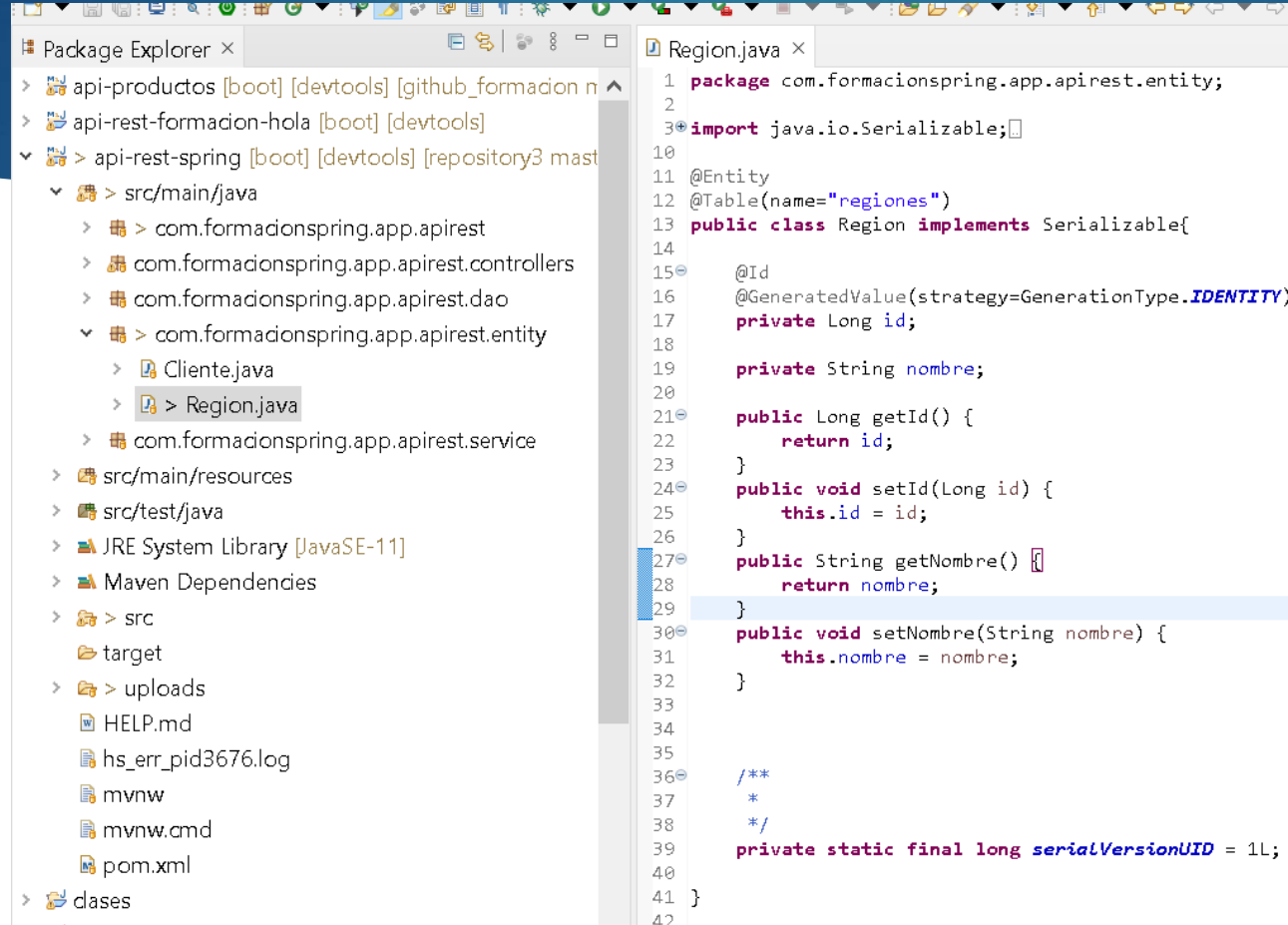
    public Teacher()
    {
        super();
    }

    public Teacher(int tid, String tname, String subject, Set clasSet)
    {
        super();
        this.tid = tid;
        this.tname = tname;
        this.subject = subject;
        this.clasSet = clasSet;
    }

    public int getTid()
    {
        return tid;
    }

    public void setTid(int tid)
    {
        this.tid = tid;
    }
}
```

# Ahora agregamos al proyecto una entity nueva Región

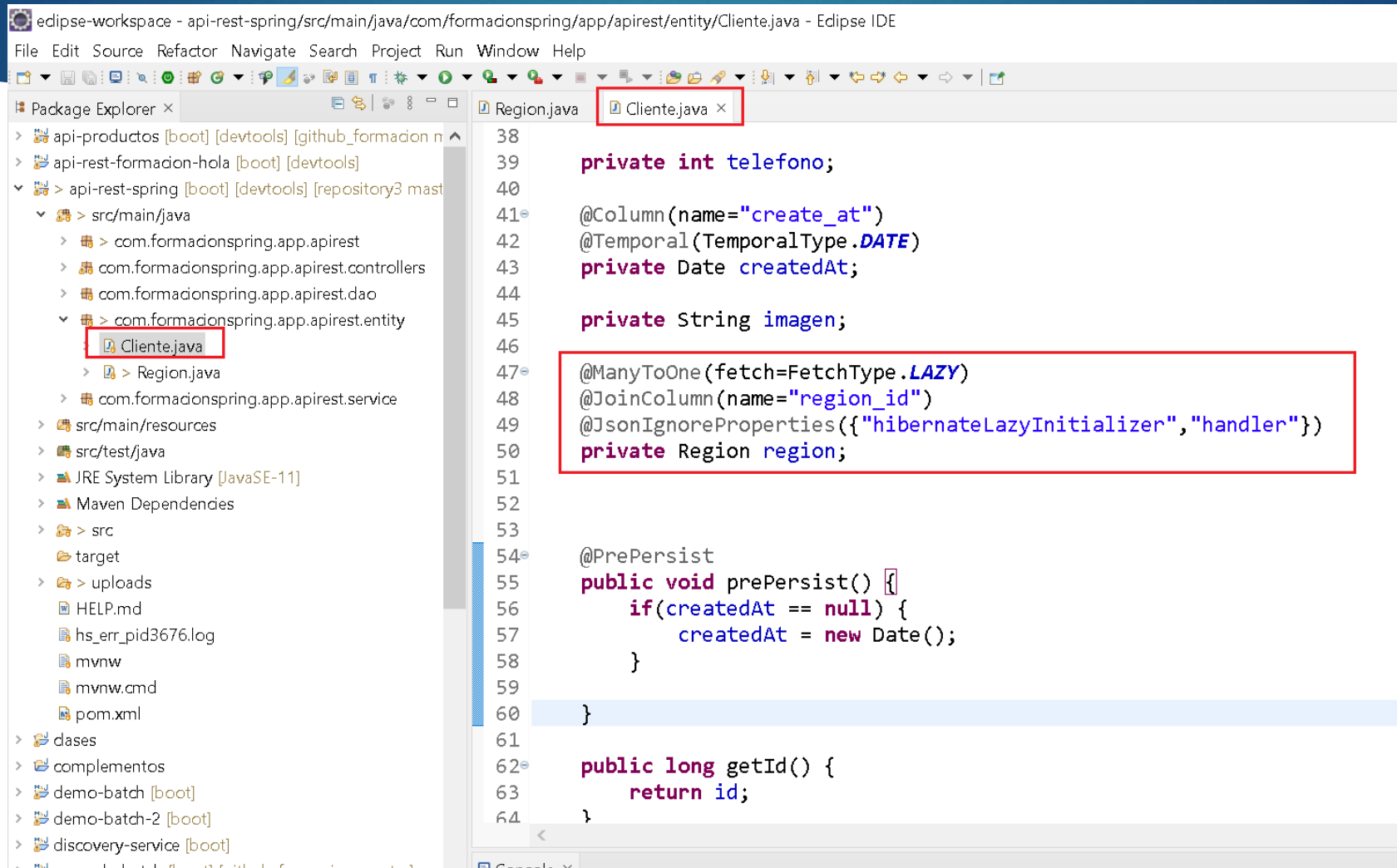


The screenshot shows an IDE with two panels. The left panel is the Package Explorer, showing a project structure with several packages. The right panel shows the code for Region.java.

```
Package Explorer ×
> api-productos [boot] [devtools] [github_formacion n
> api-rest-formacion-hola [boot] [devtools]
v > api-rest-spring [boot] [devtools] [repository3 mast
  v > src/main/java
    > com.formacionspring.app.apirest
    > com.formacionspring.app.apirest.controllers
    > com.formacionspring.app.apirest.dao
    v > com.formacionspring.app.apirest.entity
      > Cliente.java
      > Region.java
    > com.formacionspring.app.apirest.service
  > src/main/resources
  > src/test/java
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  > src
    target
  > uploads
    HELP.md
    hs_err_pid3676.log
    mvnw
    mvnw.cmd
    pom.xml
  > dases

Region.java ×
1 package com.formacionspring.app.apirest.entity;
2
3 import java.io.Serializable;
4
5
6
7
8
9
10
11 @Entity
12 @Table(name="regiones")
13 public class Region implements Serializable{
14
15     @Id
16     @GeneratedValue(strategy=GenerationType.IDENTITY)
17     private Long id;
18
19     private String nombre;
20
21     public Long getId() {
22         return id;
23     }
24     public void setId(Long id) {
25         this.id = id;
26     }
27     public String getNombre() {
28         return nombre;
29     }
30     public void setNombre(String nombre) {
31         this.nombre = nombre;
32     }
33
34
35
36 /**
37  *
38  */
39 private static final long serialVersionUID = 1L;
40
41 }
42
```

En la entity de Cliente agregamos la relación, Uso de Lazy: En la relación @ManyToOne hemos hecho uso de lazy para evitar una carga activa y traer todo, lo que podría afectar en el rendimiento de nuestra aplicación.



The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays the project structure, with the file `Cliente.java` selected under the package `com.formacionspring.app.apirest.entity`. The main editor window shows the code for `Cliente.java`. The code includes fields for `telefono`, `createAt`, and `imagen`. A `@ManyToOne` relationship is defined for the `region` field, with annotations `@FetchType.LAZY`, `@JoinColumn(name="region_id")`, and `@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})` to implement lazy loading. A `@PrePersist` method is also shown, which checks if `createdAt` is null and sets it to the current date.

```
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

private int telefono;

@Column(name="create_at")
@Temporal(TemporalType.DATE)
private Date createdAt;

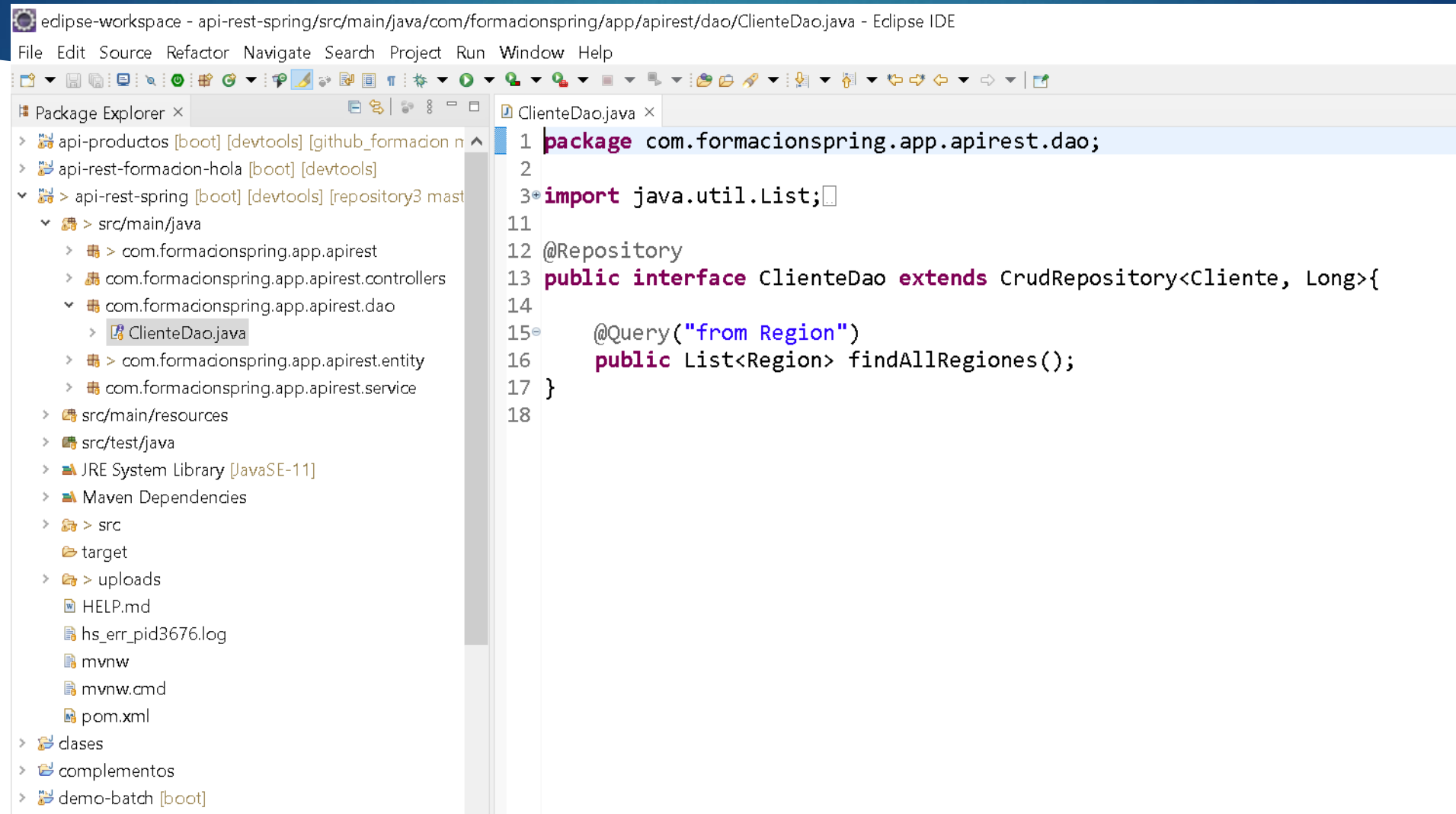
private String imagen;

@ManyToOne(fetch=FetchType.LAZY)
@JoinColumn(name="region_id")
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
private Region region;

@PrePersist
public void prePersist() {
    if(createdAt == null) {
        createdAt = new Date();
    }
}

public long getId() {
    return id;
}
```

# En nuestro repositorio dao el siguiente método

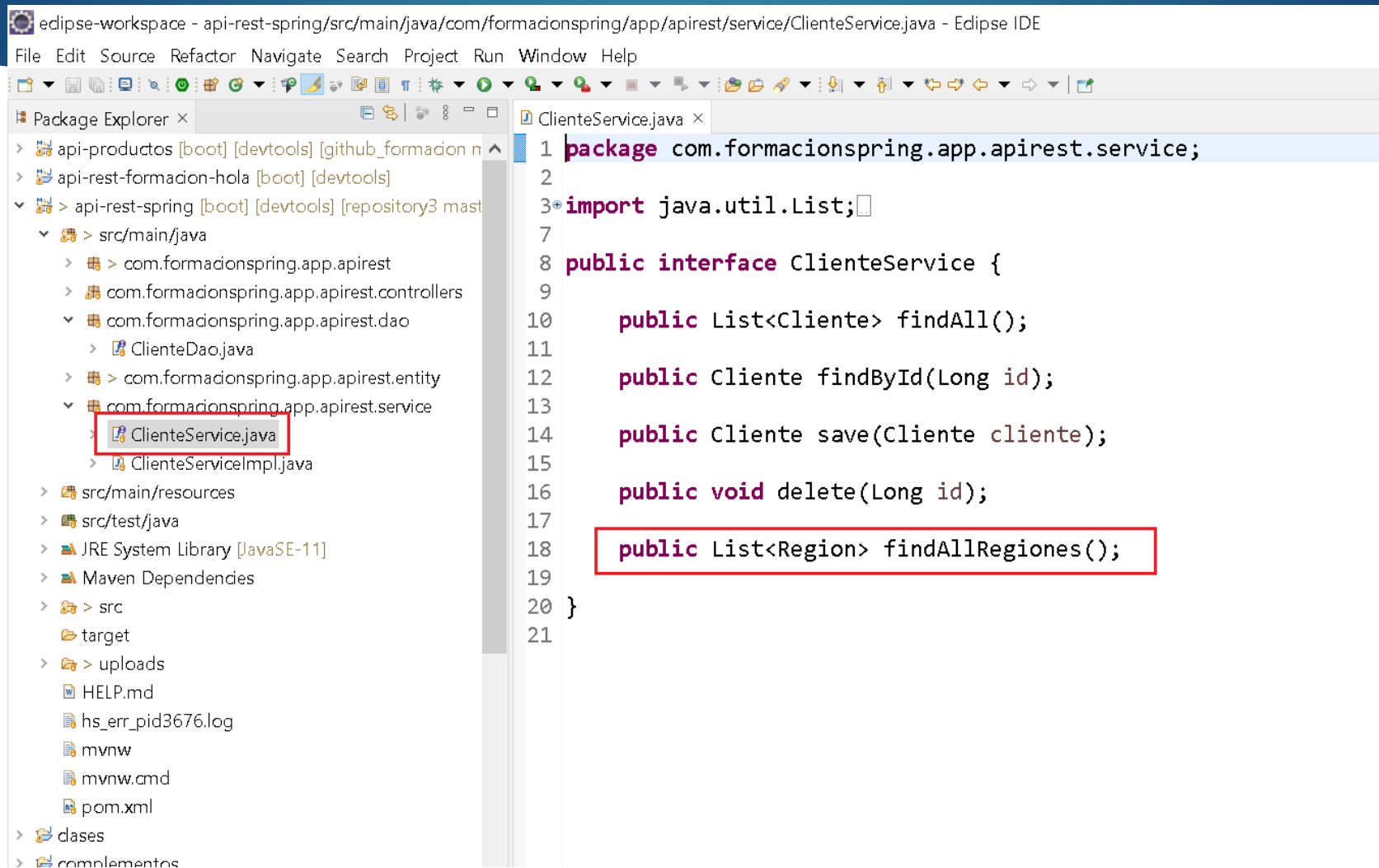


```
edipse-workspace - api-rest-spring/src/main/java/com/formacionspring/app/apirest/dao/ClienteDao.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer x
  > api-productos [boot] [devtools] [github_formacion n
  > api-rest-formacion-hola [boot] [devtools]
  > > api-rest-spring [boot] [devtools] [repository3 mast
    > > src/main/java
      > > com.formacionspring.app.apirest
      > > com.formacionspring.app.apirest.controllers
      > > com.formacionspring.app.apirest.dao
        > > ClienteDao.java
      > > com.formacionspring.app.apirest.entity
      > > com.formacionspring.app.apirest.service
    > > src/main/resources
    > > src/test/java
    > > JRE System Library [JavaSE-11]
    > > Maven Dependencies
    > > src
      > > target
    > > uploads
      > > HELP.md
      > > hs_err_pid3676.log
      > > mvnw
      > > mvnw.cmd
      > > pom.xml
  > > clases
  > > complementos
  > > demo-batch [boot]

ClienteDao.java x
1 package com.formacionspring.app.apirest.dao;
2
3 import java.util.List;
11
12 @Repository
13 public interface ClienteDao extends CrudRepository<Cliente, Long>{
14
15     @Query("from Region")
16     public List<Region> findAllRegiones();
17 }
18
```

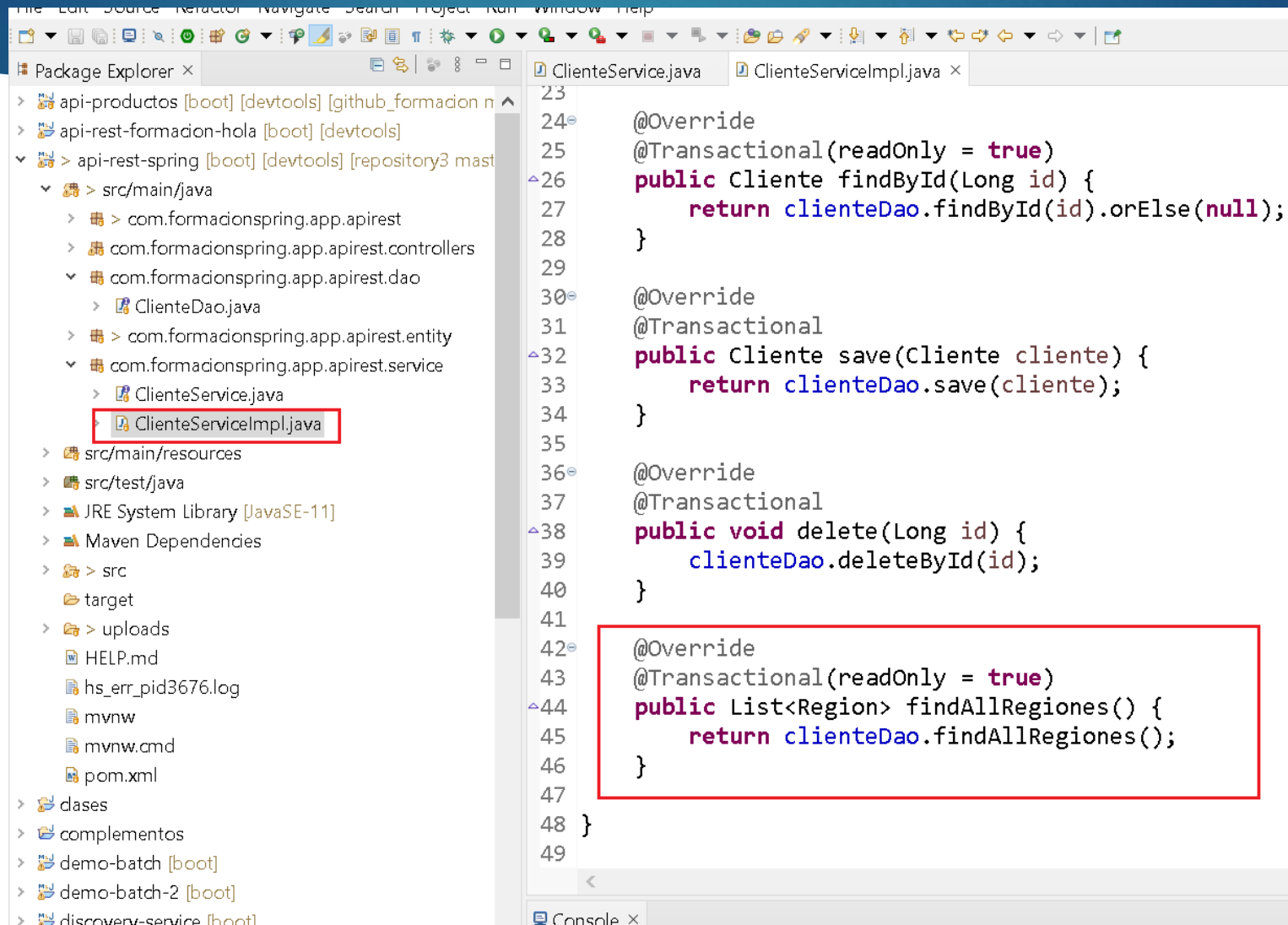
# A la interfaz de servicio agregamos el nuevo método



The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays the project structure. The package `com.formacionspring.app.apirest.service` is expanded, and the file `ClienteService.java` is selected and highlighted with a red rectangle. On the right, the source code editor shows the content of `ClienteService.java`. The code defines a package, an import, and a public interface `ClienteService` with several methods. The new method `findAllRegiones()` is highlighted with a red rectangle.

```
1 package com.formacionspring.app.apirest.service;
2
3 import java.util.List;
4
5
6
7
8 public interface ClienteService {
9
10     public List<Cliente> findAll();
11
12     public Cliente findById(Long id);
13
14     public Cliente save(Cliente cliente);
15
16     public void delete(Long id);
17
18     public List<Region> findAllRegiones();
19
20 }
21
```

# Finalmente en la implementación del servicio



The screenshot shows an IDE with the Package Explorer on the left and the code editor on the right. The Package Explorer shows the project structure with the following packages and files:

- api-productos [boot] [devtools] [github\_formation n ^
- api-rest-formation-hola [boot] [devtools]
- api-rest-spring [boot] [devtools] [repository3 mast
  - src/main/java
    - com.formacionspring.app.apirest
      - com.formacionspring.app.apirest.controllers
      - com.formacionspring.app.apirest.dao
        - ClienteDao.java
      - com.formacionspring.app.apirest.entity
      - com.formacionspring.app.apirest.service
        - ClienteService.java
        - ClienteServiceImpl.java**
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
  - target
  - uploads
  - HELP.md
  - hs\_err\_pid3676.log
  - mvnw
  - mvnw.cmd
  - pom.xml
- dases
- complementos
- demo-batch [boot]
- demo-batch-2 [boot]
- discovery-service [boot]

The code editor shows the implementation of the ClienteService interface in ClienteServiceImpl.java. The code is as follows:

```
23
24 @Override
25 @Transactional(readOnly = true)
26 public Cliente findById(Long id) {
27     return clienteDao.findById(id).orElse(null);
28 }
29
30 @Override
31 @Transactional
32 public Cliente save(Cliente cliente) {
33     return clienteDao.save(cliente);
34 }
35
36 @Override
37 @Transactional
38 public void delete(Long id) {
39     clienteDao.deleteById(id);
40 }
41
42 @Override
43 @Transactional(readOnly = true)
44 public List<Region> findAllRegiones() {
45     return clienteDao.findAllRegiones();
46 }
47
48 }
49
```

The method `findAllRegiones()` is highlighted with a red box.



# El controlador

edipse-workspace - api-rest-spring/src/main/java/com/formacionspring/app/apiest/controllers/ClienteController.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer ×

- api-productos [boot] [devtools] [github\_formacion n
- api-rest-formacion-hola [boot] [devtools]
- api-rest-spring [boot] [devtools] [repository3 mast
  - src/main/java
    - com.formacionspring.app.apiest
      - com.formacionspring.app.apiest.controllers
        - ClienteController.java
      - com.formacionspring.app.apiest.dao
      - com.formacionspring.app.apiest.entity
      - com.formacionspring.app.apiest.service
    - src/main/resources
    - src/test/java
    - JRE System Library [JavaSE-11]
    - Maven Dependencies
    - src
      - target
    - uploads
      - HELP.md
      - hs\_err\_pid3676.log
      - mvnw
      - mvnw.cmd
      - pom.xml
    - clases
    - complementos
    - demo-batch [boot]
    - demo-batch-2 [boot]
    - discovery-service [boot]

ClienteService.java ClienteServiceImpl.java ClienteController.java ×

```
275 recurso = new UriResource(rutaArchivo.toUri());
276
277 } catch (MalformedURLException e) {
278     e.printStackTrace();
279 }
280
281 if(!recurso.exists() && !recurso.isReadable()) {
282     throw new RuntimeException("Error no se puede cargar la imagen "+nombreImagen
283 }
284
285 HttpHeaders cabecera = new HttpHeaders();
286 cabecera.add(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=\""+recurso.ge
287
288 return new ResponseEntity<Resource>(recurso,cabecera,HttpStatus.OK);
289
290 }
291
292 @GetMapping("/clientes/regiones")
293 public List<Region> listaRegiones(){
294     return servicio.findAllRegiones();
295 }
296
297
298
299
300 }
301
```

# Modificamos en import.sql

```
INSERT INTO regiones (nombre) VALUES ('Sudamerica')
INSERT INTO regiones (nombre) VALUES ('Centroamerica')
INSERT INTO regiones (nombre) VALUES ('Norteamerica')
INSERT INTO regiones (nombre) VALUES ('Europa')
INSERT INTO regiones (nombre) VALUES ('Asia')
INSERT INTO regiones (nombre) VALUES ('Africa')
INSERT INTO regiones (nombre) VALUES ('Oceania')
INSERT INTO regiones (nombre) VALUES ('Antartida')
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(1,'Jose','Perez','jp@hotmail.com',65433223,'2021-10-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(2,'Carlos','Lopez','cl@hotmail.com',65433223,'2021-01-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(3,'Maria','Orillana','mo@hotmail.com',65433223,'2021-02-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(4,'Dina','Ramirez','dr@hotmail.com',65433223,'2021-03-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(5,'Mirna','Ramos','mr@hotmail.com',65433223,'2021-04-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(6,'Pepe','Mojica','pm@hotmail.com',65433223,'2021-05-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(7,'Juan','Chavez','jc@hotmail.com',65433223,'2021-06-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(8,'Enrrique','Iglesias','ei@hotmail.com',65433223,'2021-07-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(3,'Pedro','Diaz','pd@hotmail.com',65433223,'2021-08-01');
INSERT INTO clientes (region_id,nombre,apellido,email,telefono,create_at) VALUES(1,'Ramon','Gonzalez','rgonzalez@gmail.com',65433223,'2021-09-01');
```

# Swagger

- ▶ Swagger es un conjunto de herramientas de software de código abierto para diseñar, construir, documentar, y utilizar servicios web RESTful. Fue desarrollado por SmartBear Software e incluye documentación automatizada, generación de código, y generación de casos de prueba.
- ▶ Es una herramienta muy importante para la documentación de nuestra API REST.
- ▶ <https://swagger.io/tools/swagger-ui/>

# Implementar Swagger V3 a SpringBoot

edipse-workspace - apirest-dientes/pom.xml - Eclipse IDE

File Edit Source Navigate Search Project Run Window Help

Package Explorer ×

- > com.formacionspringboot.apirest
  - > com.formacionspringboot.apirest.controller
  - > com.formacionspringboot.apirest.dao
  - > com.formacionspringboot.apirest.entity
  - > com.formacionspringboot.apirest.service
- > src/main/resources
- > src/test/java
- > JRE System Library [JavaSE-11]
- > Maven Dependencies
- > src
  - target
  - uploads
  - HELP.md
  - mvnw
  - mvnw.cmd
  - > pom.xml
- > api-rest-spring [boot] [devtools] [repository3 master]
- > appwebmvc [boot] [devtools] [repository4 master]
- > example-batch [boot] [github\_formacion master]

apirest-dientes/pom.xml ×

```
10 </parent>
11 <groupId>com.formacionspringboot.apirest</groupId>
12 <artifactId>apirest-clientes</artifactId>
13 <version>0.0.1-SNAPSHOT</version>
14 <name>apirest-clientes</name>
15 <description>Spring Web MVC</description>
16 <properties>
17     <java.version>11</java.version>
18 </properties>
19 <dependencies>
20
21     <dependency>
22         <groupId>org.springdoc</groupId>
23         <artifactId>springdoc-openapi-ui</artifactId>
24         <version>1.5.9</version>
25     </dependency>
26
27     <dependency>
28         <groupId>org.springframework.boot</groupId>
29         <artifactId>spring-boot-starter-data-jpa</artifactId>
30     </dependency>
31 </dependencies>
```

Boot Dashboard ×

Overview Dependencies Dependency Hierarchy Effective POM pom.xml

# Agregamos a la clase raíz del proyecto

edipse-workspace - apirest-dientes/src/main/java/com/formacionspringboot/apirest/ApirestClientesApplication.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

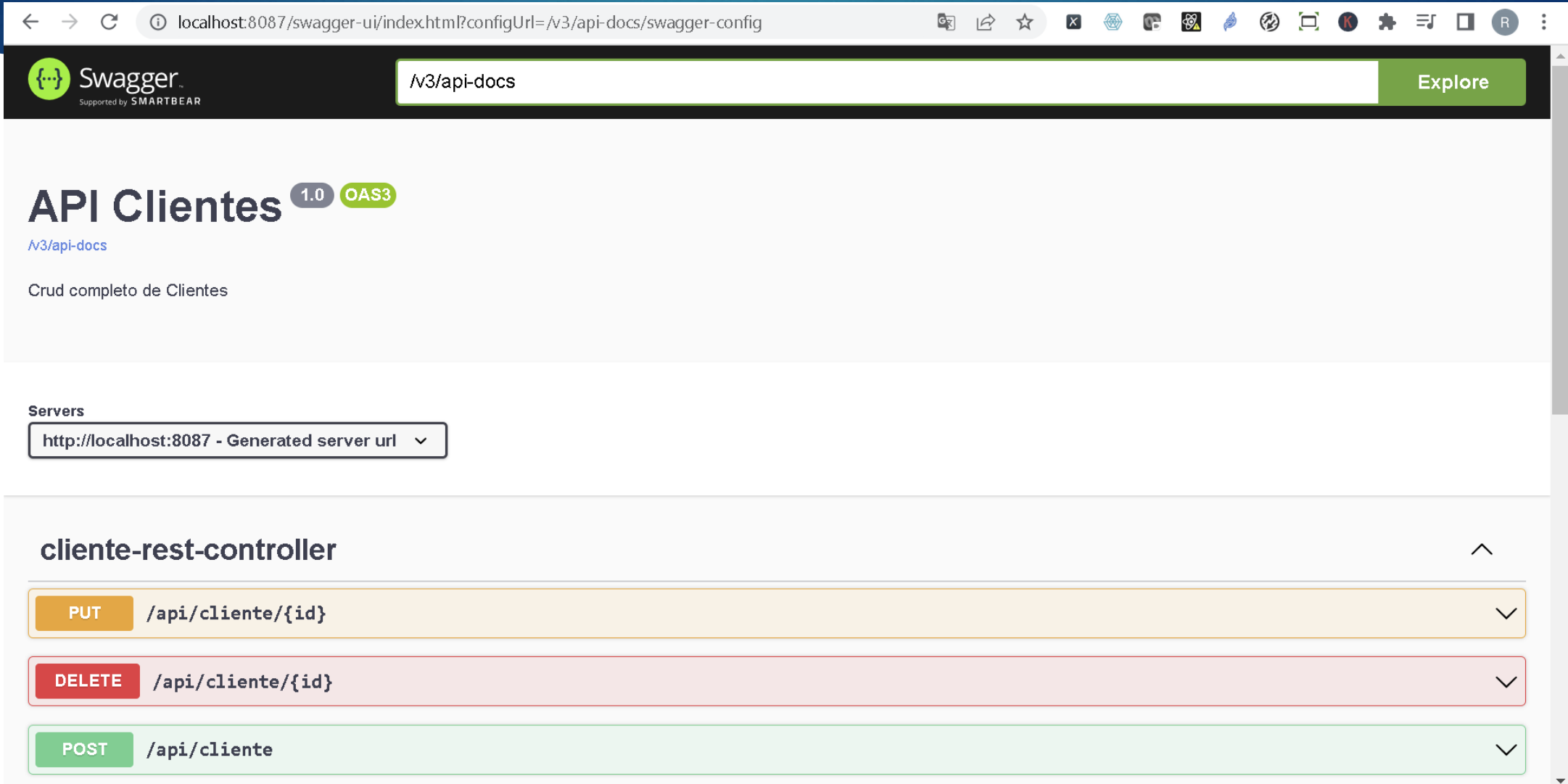
Package Explorer ×

api-productos [boot] [devtools] [github\_formacion ma...  
apirest-dientes [boot] [devtools] [repository6 maste...  
src/main/java  
com.formacionspringboot.apirest  
ApirestClientesApplication.java  
com.formacionspringboot.apirest.controller  
com.formacionspringboot.apirest.dao  
com.formacionspringboot.apirest.entity  
com.formacionspringboot.apirest.service  
src/main/resources  
src/test/java  
JRE System Library [JavaSE-11]  
Maven Dependencies  
src  
target  
uploads  
HELP.md  
mvnw  
mvnw.cmd

ApirestClientesApplication.java ×

```
1 package com.formacionspringboot.apirest;  
2  
3 import org.springframework.boot.SpringApplication;  
4 import org.springframework.boot.autoconfigure.SpringBootApplication;  
5  
6 import io.swagger.v3.oas.annotations.OpenAPIDefinition;  
7 import io.swagger.v3.oas.annotations.info.Info;  
8  
9 @SpringBootApplication  
10 @OpenAPIDefinition(info = @Info(title = "API Clientes", version = "1.0", description = "Crud completo de Clientes"))  
11 public class ApirestClientesApplication {  
12  
13     public static void main(String[] args) {  
14         SpringApplication.run(ApirestClientesApplication.class, args);  
15     }  
16  
17 }  
18
```

Ingresamos a <http://localhost:8087/swagger-ui.html>



The screenshot shows the Swagger UI interface in a web browser. The address bar displays the URL `localhost:8087/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config`. The Swagger logo is in the top left, and a search bar contains `/v3/api-docs` with an **Explore** button. The main title is **API Clientes** with version **1.0** and **OAS3** tags. Below the title, it says `/v3/api-docs` and **Crud completo de Clientes**. A **Servers** section shows a dropdown menu with `http://localhost:8087 - Generated server url`. The **cliente-rest-controller** section is expanded, showing three endpoints: **PUT** `/api/cliente/{id}`, **DELETE** `/api/cliente/{id}`, and **POST** `/api/cliente`. Each endpoint has a dropdown arrow on the right.

Swagger  
Supported by SMARTBEAR

`/v3/api-docs` Explore

# API Clientes 1.0 OAS3

`/v3/api-docs`

Crud completo de Clientes

**Servers**

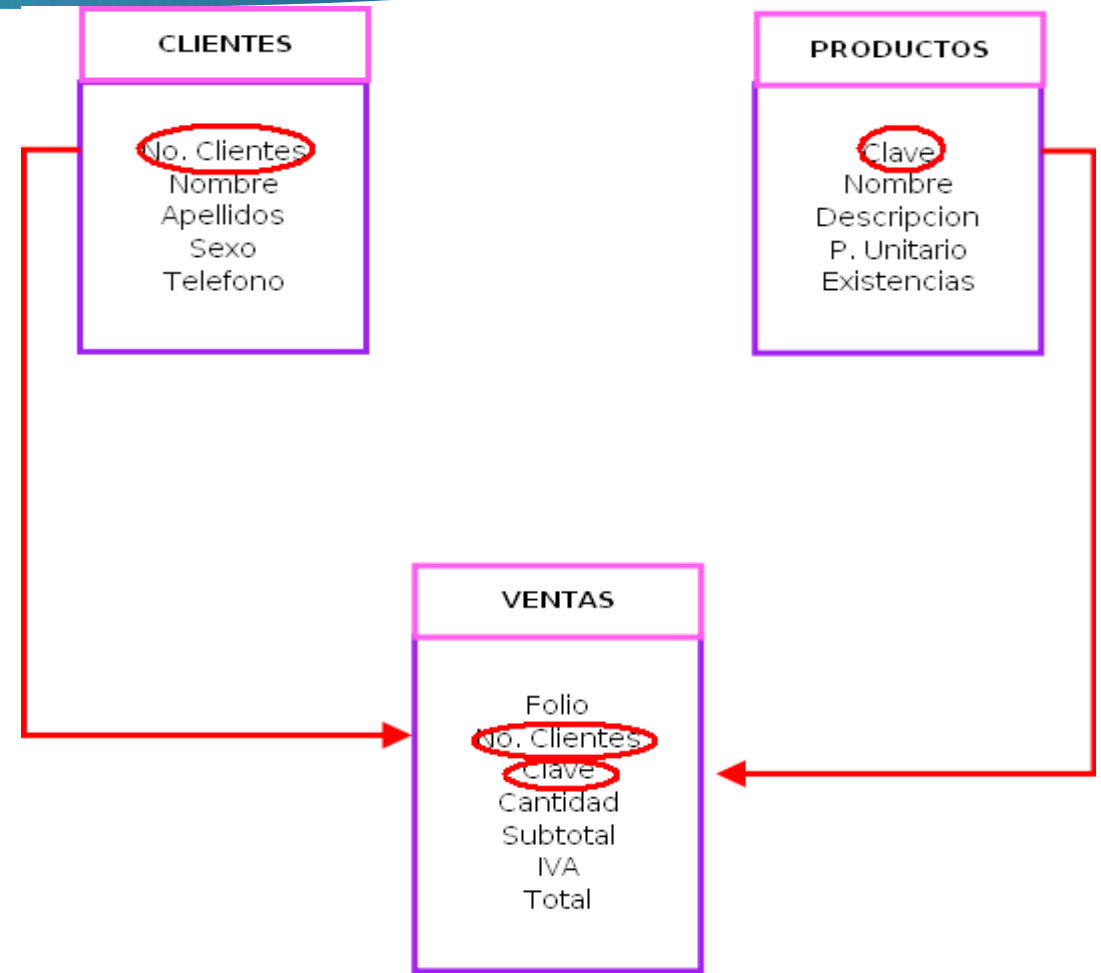
`http://localhost:8087 - Generated server url` ▾

## cliente-rest-controller ^

PUT	<code>/api/cliente/{id}</code>	▾
DELETE	<code>/api/cliente/{id}</code>	▾
POST	<code>/api/cliente</code>	▾

# Ejercicio

- Creo una api rest, teniendo en cuenta este modelo para las entitys, desarrollar los métodos CRUD de cada entidad, comprobar que todo funcione por postman



# Unit Test

- ▶ Son códigos diseñados para comprobar que el código principal este funcionando de la forma que esperamos.



# Tipos de pruebas

- ▶ Pruebas Unitarias (Unit Test)
- ▶ Pruebas de Integración
- ▶ Pruebas de funcionamiento del sistema
- ▶ Pruebas de Aceptación
- ▶ Pruebas de stress

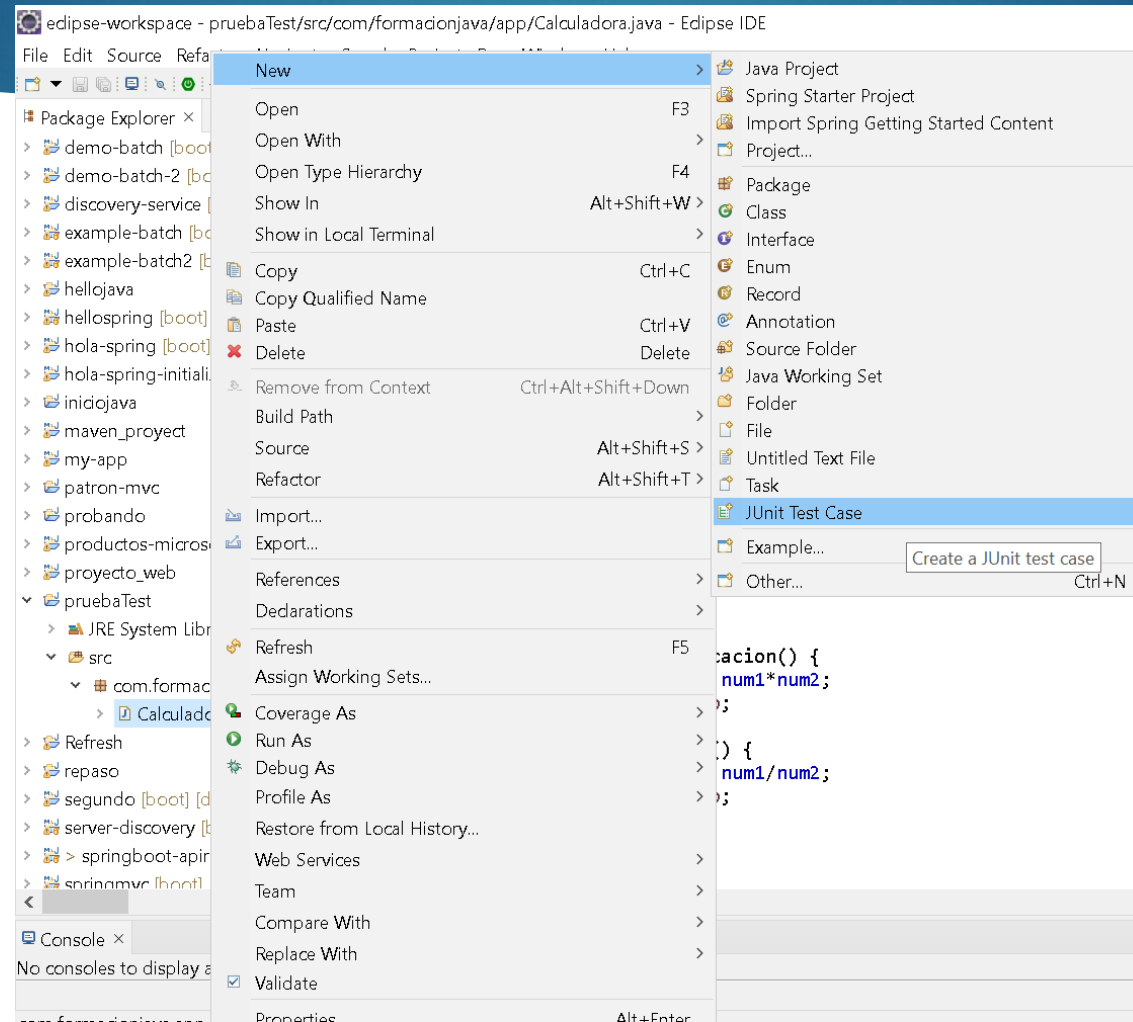
# JUnit

- ▶ Es una librería java para escribir y ejecutar pruebas unitarias

Realizamos  
una prueba

```
1 package com.fernandojavaapps;
2
3 public class Calculadora {
4
5     private int num1;
6     private int num2;
7
8     public Calculadora(int a, int b) {
9         this.num1 = a;
10        this.num2 = b;
11    }
12
13    public int suma() {
14        int resultado = num1+num2;
15        return resultado;
16    }
17
18    public int resta() {
19        int resultado = num1-num2;
20        return resultado;
21    }
22
23    public int multiplicacion() {
24        int resultado = num1*num2;
25        return resultado;
26    }
27    public int division() {
28        int resultado = num1/num2;
29        return resultado;
30    }
31 }
32
```

# Creamos la clase de unitTest



com.formacionjava/app/Calculadora.java

Project Run Win

1 package com.formacionjava

2

3 public class Calculadora {

4

5 private static final double PI = 3.14159;

6 private static final double E = 2.71828;

7

8 public static double areaCirculo(double radio) {

9 return PI \* radio \* radio;

10 }

11 }

12

13 public static double factorial(int n) {

14 if (n < 0) return -1;

15 if (n == 0) return 1;

16 return n \* factorial(n - 1);

17 }

18 public static double potencia(double base, int exponente) {

19 if (exponente < 0) return 1 / potencia(base, -exponente);

20 if (exponente == 0) return 1;

21 return base \* potencia(base, exponente - 1);

22 }

23 public static double sumaGeometrica(double a, int n) {

24 if (a == 0) return 0;

25 if (n == 0) return a;

26 return a + sumaGeometrica(a, n - 1);

27 }

28 }

29 }

30 }

31 }

32 }

## New JUnit Test Case

### JUnit Test Case

Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit Jupiter test

Source folder: pruebaTest/src Browse...

Package: com.formacionjava.app Browse...

Name: CalculadoraTest

Superclass: java.lang.Object Browse...

Which method stubs would you like to create?

- ☐ @BeforeAll setUpBeforeClass() ☐ @AfterAll tearDownAfterClass()  
☐ @BeforeEach setUp() ☐ @AfterEach tearDown()  
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

Class under test: com.formacionjava.app.Calculadora Browse...



< Back

Next >

Finish

Cancel



## New JUnit Test Case

### Test Methods

Select methods for which test method stubs should be created.

Available methods:

- ☒ **Calculadora**
  - ☐ **Calculadora(int, int)**
  - ☐ **suma()**
  - ☐ **resta()**
  - ☐ **multiplicacion()**
  - ☐ **division()**
- ☒ **Object**
  - ☐ **Object()**
  - ☐ **getClass()**
  - ☐ **hashCode()**
  - ☐ **equals(Object)**
  - ☐ **clone()**
  - ☐ **toString()**

Select All

Deselect All

0 methods selected.

☐ Create final method stubs

☐ Create tasks for generated test methods



< Back

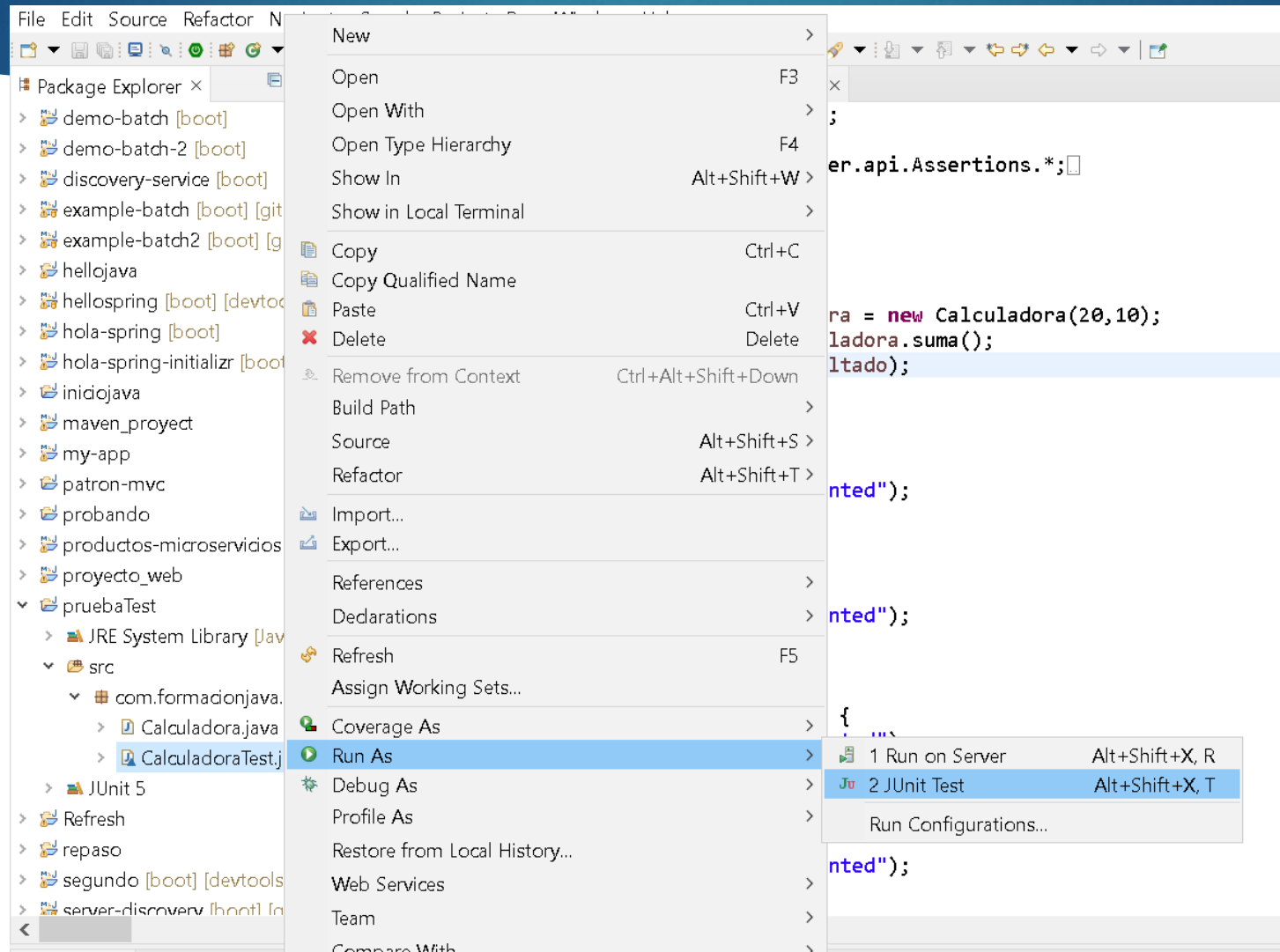
Next >

Finish

Cancel

```
Calculadora.java  CalculadoraTest.java x
1 package com.formacionjava.app;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5
6 class CalculadoraTest {
7
8     @Test
9     void testCalculadora() {
10         fail("Not yet implemented");
11     }
12
13     @Test
14     void testSuma() {
15         fail("Not yet implemented");
16     }
17
18     @Test
19     void testResta() {
20         fail("Not yet implemented");
21     }
22
23     @Test
24     void testMultiplicacion() {
25         fail("Not yet implemented");
26     }
27
28     @Test
29     void testDivision() {
30         fail("Not yet implemented");
31     }
32 }
33
34 }
```

# Ejecutamos el UnitTest





eclipse-workspace - pruebaTest/src/com/formacionjava/a

File Edit Source Refactor Navigate Search Project Run

Package Explorer JUnit x

Finished after 0,203 seconds

Runs: 5/5    x Errors: 0    x Failures: 4

▼ CalculadoraTest [Runner: JUnit 5] (0,006 s)

- testCalculadora() (0,001 s)
- testResta() (0,002 s)
- testSuma() (0,001 s)
- testMultiplicacion() (0,001 s)
- testDivision() (0,001 s)

Failure Trace

edipse-workspace - pruebaTest/src/com/formacionjava/app/CalculadoraTest.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



Package Explorer JUnit x

Finished after 0,224 seconds

Runs: 4/4 Errors: 0 Failures: 0

CalculadoraTest [Runner: JUnit 5] (0,003 s)

- testResta() (0,000 s)
- testSuma() (0,000 s)
- testMultiplicacion() (0,000 s)
- testDivision() (0,001 s)

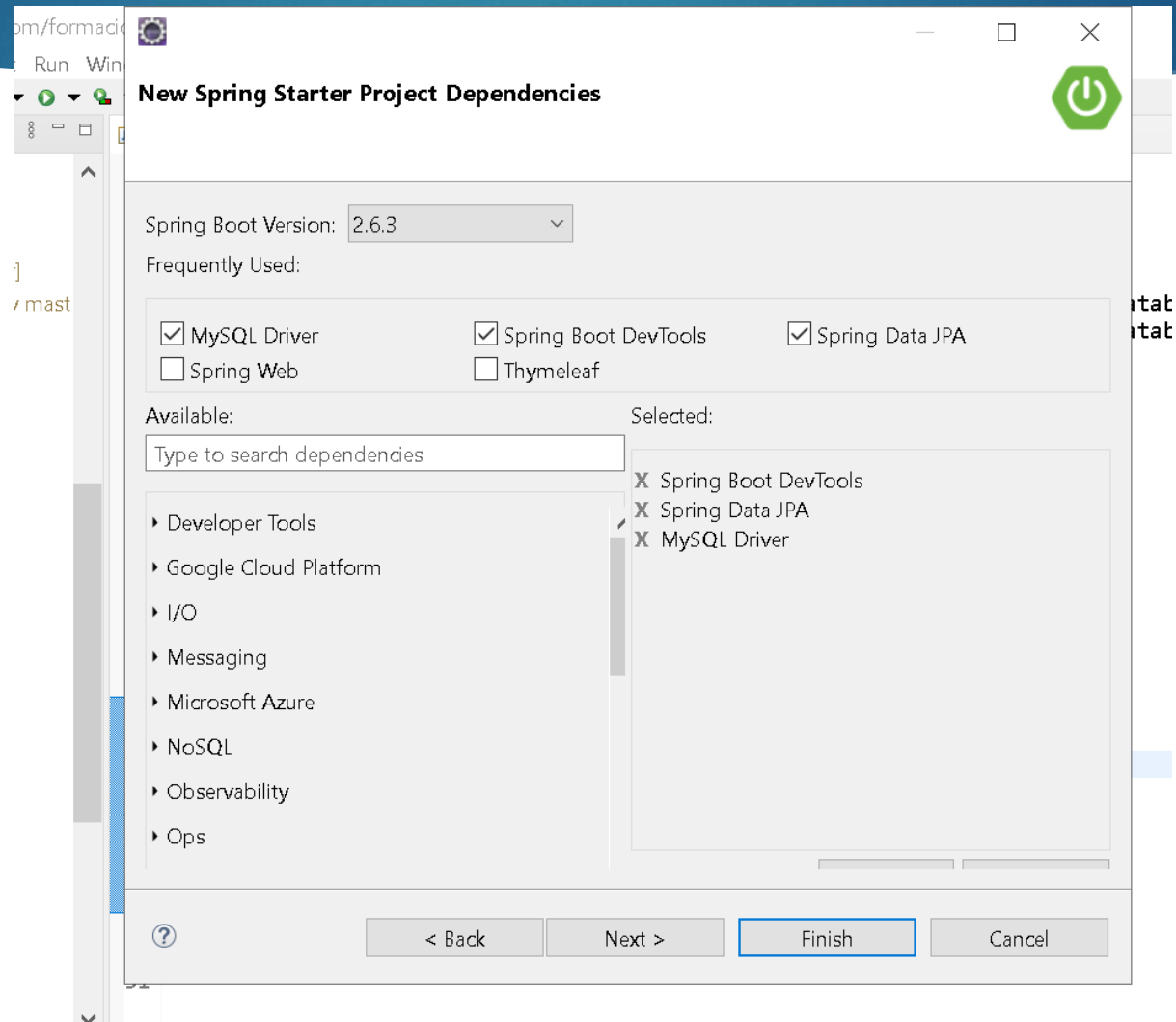
Failure Trace

Calculadora.java

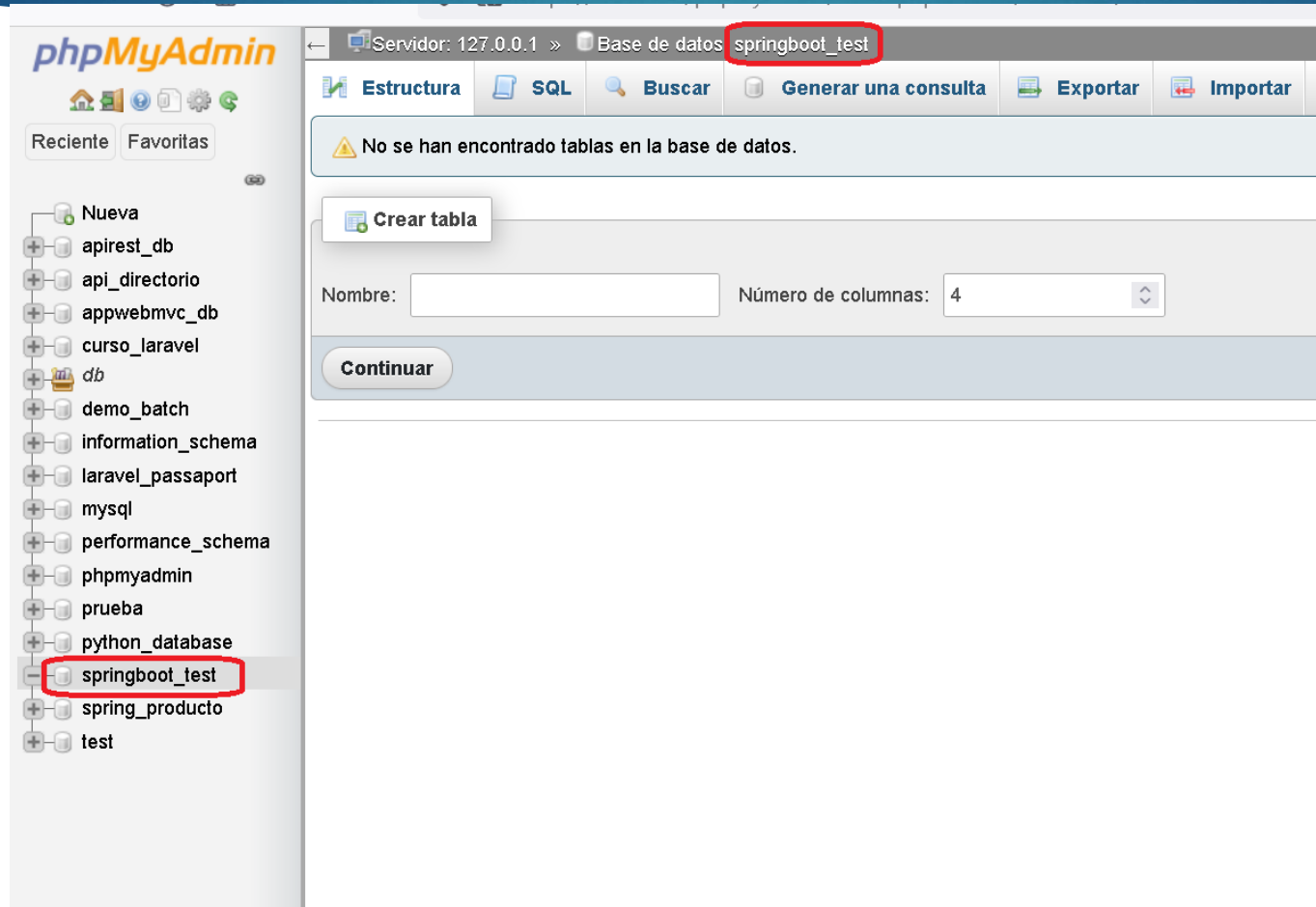
CalculadoraTest.java x

```
3 import static org.junit.jupiter.api.Assertions.*;
6
7 class CalculadoraTest {
8
9     /*@Test
10     void testCalculadora() {
11
12     }*/
13
14     @Test
15     void testSuma() {
16         Calculadora calculadora = new Calculadora(20,10);
17         int resultado = calculadora.suma();
18         assertEquals(30, resultado);
19     }
20
21     @Test
22     void testResta() {
23         Calculadora calculadora = new Calculadora(20,10);
24         int resultado = calculadora.resta();
25         assertEquals(10, resultado);
26     }
27
28     @Test
29     void testMultiplicacion() {
30         Calculadora calculadora = new Calculadora(3,2);
31         int resultado = calculadora.multiplicacion();
32         assertEquals(6, resultado);
33     }
34
35     @Test
36     void testDivision() {
37         Calculadora calculadora = new Calculadora(20,10);
38         int resultado = calculadora.division();
39         assertEquals(2, resultado);
40     }
41 }
42
43
```

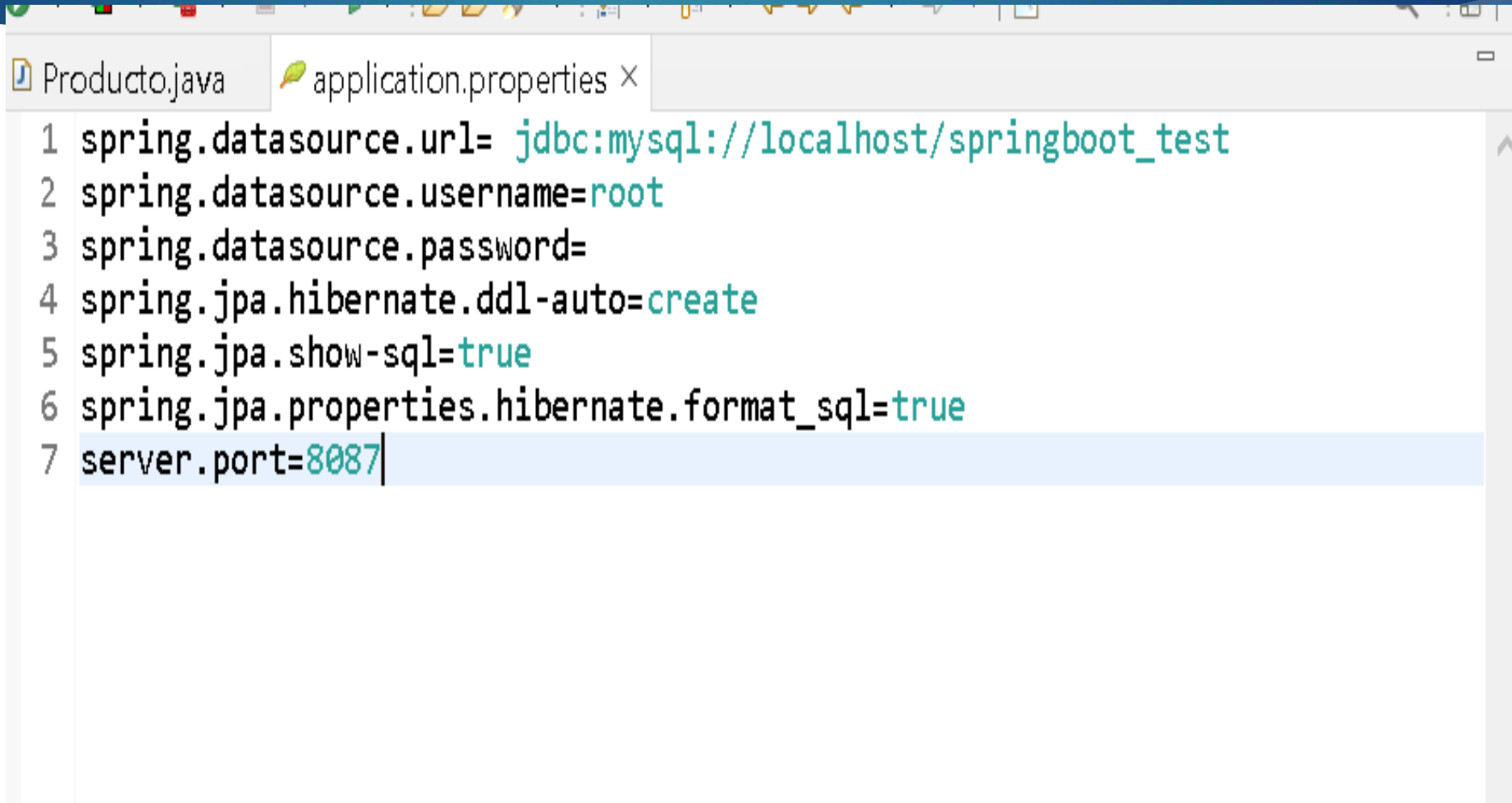
# Utilizando pruebas unitarias en SpringBoot, creamos un proyecto con las siguientes dependencias



# Creamos una base de datos



# En nuestro application.properties

A screenshot of an IDE window showing the configuration of application.properties. The window has two tabs: 'Producto.java' and 'application.properties'. The 'application.properties' tab is active, displaying a list of seven configuration properties. The properties are: 1. spring.datasource.url= jdbc:mysql://localhost/springboot\_test, 2. spring.datasource.username=root, 3. spring.datasource.password=, 4. spring.jpa.hibernate.ddl-auto=create, 5. spring.jpa.show-sql=true, 6. spring.jpa.properties.hibernate.format\_sql=true, and 7. server.port=8087. The seventh line is highlighted with a light blue background. The IDE interface includes a toolbar at the top with various icons for file operations and a vertical scrollbar on the right side of the code editor.

```
1 spring.datasource.url= jdbc:mysql://localhost/springboot_test
2 spring.datasource.username=root
3 spring.datasource.password=
4 spring.jpa.hibernate.ddl-auto=create
5 spring.jpa.show-sql=true
6 spring.jpa.properties.hibernate.format_sql=true
7 server.port=8087
```

# Creamos la entidad Producto

```
1 package com.formacionspringboot.app.entity;
2
3 import javax.persistence.Entity;
4
5 @Entity
6 @Table(name="productos")
7 public class Producto {
8
9     @Id
10    @GeneratedValue(strategy = GenerationType.IDENTITY)
11    private Long id;
12    private String nombre;
13    private float precio;
14
15    public Producto() {
16    }
17
18    public Producto(String nombre, float precio) {
19        this.nombre = nombre;
20        this.precio = precio;
21    }
22
23    public Long getId() {
24        return id;
25    }
26
27    public void setId(Long id) {
28        this.id = id;
29    }
30
31    public String getNombre() {
32        return nombre;
33    }
34
35    public void setNombre(String nombre) {
36        this.nombre = nombre;
37    }
38 }
```

# Creamos un repositorio ProductoDao

edipse-workspace - springboot-test/src/main/java/com/formacionspringboot/app/dao/ProductoDao.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



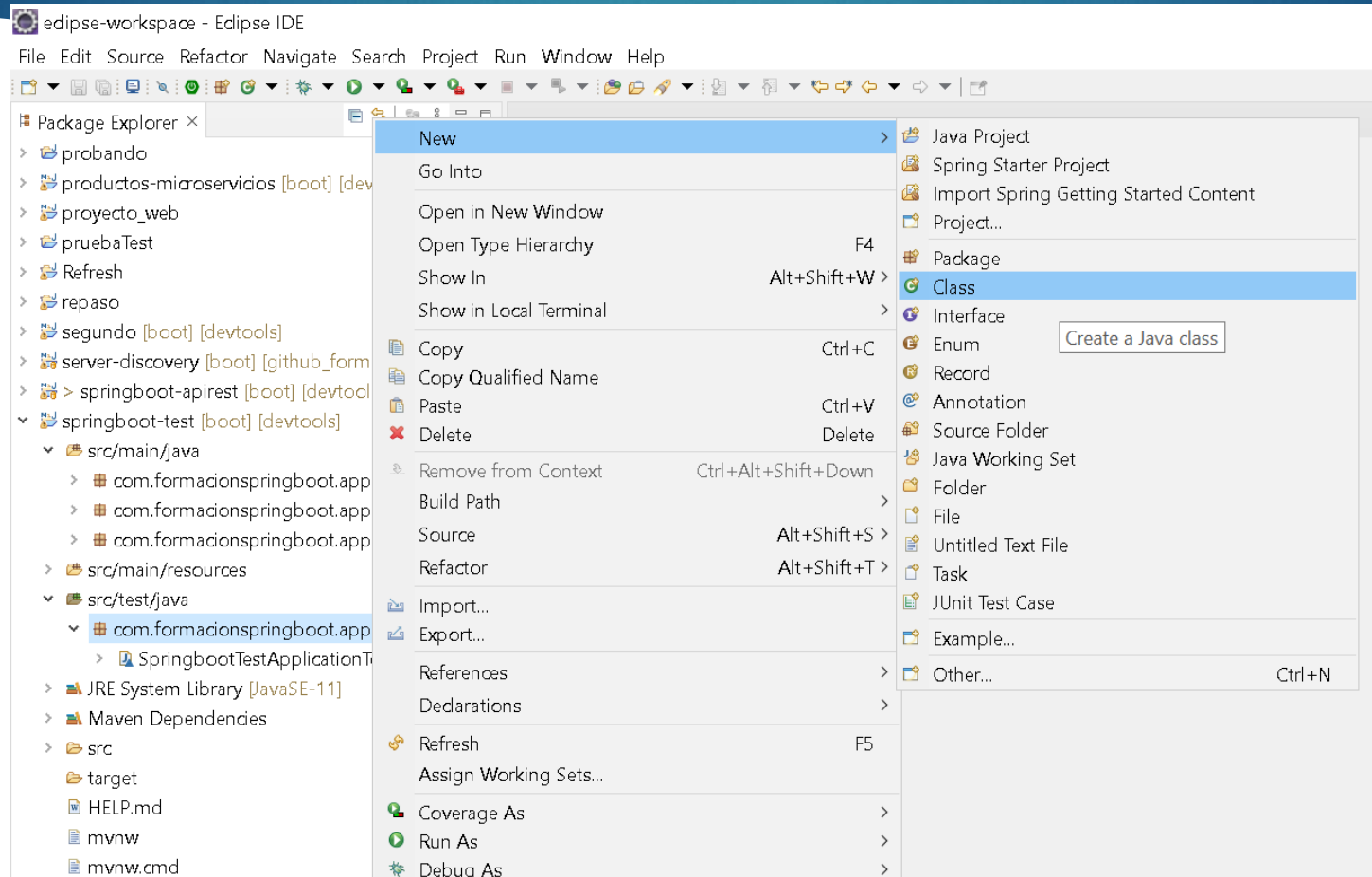
Package Explorer ×

- > probando
- > productos-microservicios [boot] [devtools]
- > proyecto\_web
- > pruebaTest
- > Refresh
- > repaso
- > segundo [boot] [devtools]
- > server-discovery [boot] [github\_formation master]
- > > springboot-apirest [boot] [devtools] [repository]
- ▼ springboot-test [boot] [devtools]
  - ▼ src/main/java
    - > com.formacionspringboot.app
    - ▼ com.formacionspringboot.app.dao
      - > ProductoDao.java
    - > com.formacionspringboot.app.entity
  - > src/main/resources
  - > src/test/java
  - > JRE System Library [JavaSE-11]
  - > Maven Dependencies
  - > src
    - target
    - HELP.md
    - mvnw

ProductoDao.java ×

```
1 package com.formacionspringboot.app.dao;
2
3 import org.springframework.data.repository.CrudRepository;
4 import com.formacionspringboot.app.entity.Producto;
5
6 public interface ProductoDao extends CrudRepository<Producto, Long>{
7
8 }
9
```

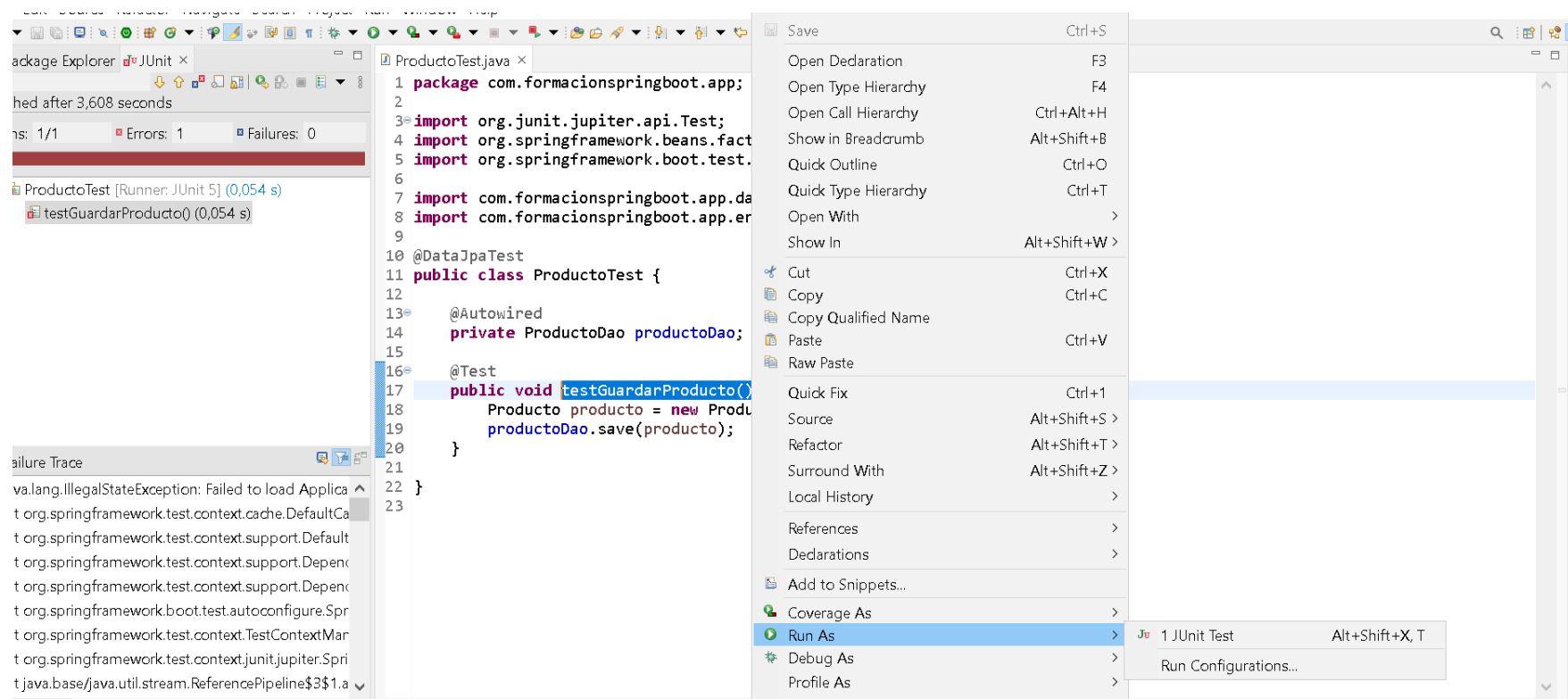
# Dentro del paquete principal de src/test/java creamos una clase ProductoTest





```
ProductoTest.java x
1 package com.formacionspringboot.app;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
6
7 import com.formacionspringboot.app.dao.ProductoDao;
8 import com.formacionspringboot.app.entity.Producto;
9
10 @DataJpaTest
11 public class ProductoTest {
12
13     @Autowired
14     private ProductoDao productoDao;
15
16     @Test
17     public void testGuardarProducto() {
18         Producto producto = new Producto("TV Samsung HD",4000);
19         productoDao.save(producto);
20     }
21
22 }
23
```

# Lo ejecutamos

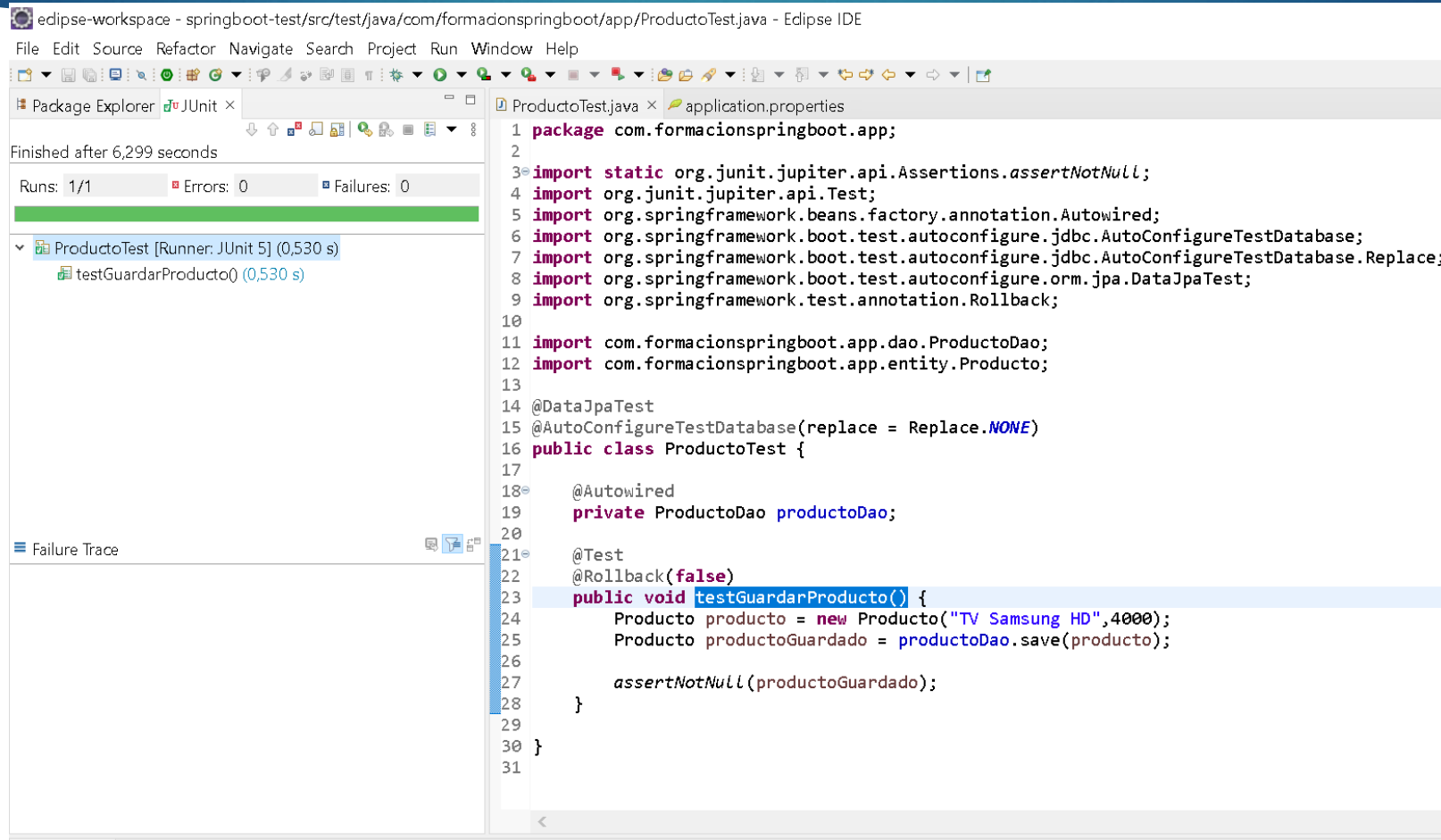


# Agregamos las notaciones y código para necesarias para lanzar los test y registrarlos en la base de datos

```
com.formacionspringboot/app/ProductoTest.java - Eclipse IDE
Window Help
ProductoTest.java x application.properties

1 package com.formacionspringboot.app;
2
3 import static org.junit.jupiter.api.Assertions.assertNotNull;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
7 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
8 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
9 import org.springframework.test.annotation.Rollback;
10
11 import com.formacionspringboot.app.dao.ProductoDao;
12 import com.formacionspringboot.app.entity.Producto;
13
14 @DataJpaTest
15 @AutoConfigureTestDatabase(replace = Replace.NONE)
16 public class ProductoTest {
17
18     @Autowired
19     private ProductoDao productoDao;
20
21     @Test
22     @Rollback(false)
23     public void testGuardarProducto() {
24         Producto producto = new Producto("TV Samsung HD", 4000);
25         Producto productoGuardado = productoDao.save(producto);
26
27         assertNotNull(productoGuardado);
28     }
29
30 }
```

# Ejecutamos el test



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure, with 'ProductoTest' selected under 'JUnit'. The Run console shows 'Finished after 6,299 seconds' and 'Runs: 1/1', 'Errors: 0', 'Failures: 0'. The Failure Trace is empty. The main editor displays the source code of 'ProductoTest.java'.

```
1 package com.formacionspringboot.app;
2
3 import static org.junit.jupiter.api.Assertions.assertNotNull;
4 import org.junit.jupiter.api.Test;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;
7 import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;
8 import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;
9 import org.springframework.test.annotation.Rollback;
10
11 import com.formacionspringboot.app.dao.ProductoDao;
12 import com.formacionspringboot.app.entity.Producto;
13
14 @DataJpaTest
15 @AutoConfigureTestDatabase(replace = Replace.NONE)
16 public class ProductoTest {
17
18     @Autowired
19     private ProductoDao productoDao;
20
21     @Test
22     @Rollback(false)
23     public void testGuardarProducto() {
24         Producto producto = new Producto("TV Samsung HD", 4000);
25         Producto productoGuardado = productoDao.save(producto);
26
27         assertNotNull(productoGuardado);
28     }
29 }
30
31
```

# Verificamos el registro en la base de datos

The screenshot shows the phpMyAdmin interface for a local server (127.0.0.1) connected to a database named 'springboot\_test'. The selected table is 'productos'. The interface displays a single record in the table:

id	nombre	precio
1	TV Samsung HD	4000

The interface also shows the SQL query used to retrieve the data: `SELECT * FROM `productos``. Below the table, there are options to perform actions on the results, such as 'Imprimir', 'Copiar al portapapeles', 'Exportar', 'Mostrar gráfico', and 'Crear vista'. At the bottom, there is a section to 'Guardar esta consulta en favoritos' with a label field and a checkbox to allow access to the favorite.

1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

n Window Help

 ProductTest.java

ProductoDao.java ×

```
1 package com.formationspringboot.app.dao;
2
3 import org.springframework.data.repository.CrudRepository;
4
5
6 public interface ProductoDao extends CrudRepository<Producto, Long>{
7
8     public Producto findByNombre(String nombre);
9
10 }
11
```

# Agregamos el siguiente método para el test

```
onspringboot/app/ProductoTest.java - Eclipse IDE
File Edit Window Help
[Icons]
ProductoTest.java x ProductoDao.java
17 public class ProductoTest {
18
19     @Autowired
20     private ProductoDao productoDao;
21
22     @Test
23     @Rollback(false)
24     public void testGuardarProducto() {
25         Producto producto = new Producto("TV Samsung HD",4000);
26         Producto productoGuardado = productoDao.save(producto);
27
28         assertNotNull(productoGuardado);
29     }
30     @Test
31     public void testBuscarProductoPorNombre() {
32         String nombre= "TV Samsung HD";
33         Producto producto = productoDao.findByNombre(nombre);
34
35         assertThat(producto.getNombre()).isEqualTo(nombre);
36     }
37
38 }
```

# Ejecutamos

The screenshot shows the Eclipse IDE interface. The title bar indicates the workspace is 'edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Edipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure, with 'ProductoTest.java' selected under 'src/test/java/com/formacionspringboot.app'. The main editor displays the code for 'ProductoTest.java', which is highlighted with a red box in the tab bar. The code includes annotations for '@Autowired', '@Test', and '@Rollback(false)', and methods for 'testGuardarProducto()' and 'testBuscarProductoPorNombre()'. The 'testBuscarProductoPorNombre()' method is highlighted with a blue box. The 'Run' button (a green play icon) is visible in the toolbar above the editor.

```
17 public class ProductoTest {
18
19     @Autowired
20     private ProductoDao productoDao;
21
22     @Test
23     @Rollback(false)
24     public void testGuardarProducto() {
25         Producto producto = new Producto("TV Samsung HD",4000);
26         Producto productoGuardado = productoDao.save(producto);
27
28         assertNotNull(productoGuardado);
29     }
30
31     @Test
32     public void testBuscarProductoPorNombre() {
33         String nombre= "TV Samsung HD";
34         Producto producto = productoDao.findByNombre(nombre);
35
36         assertThat(producto.getNombre()).isEqualTo(nombre);
37     }
38 }
```



eclipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help



Package Explorer JUnit x



Finished after 6,38 seconds

Runs: 1/1 Errors: 0 Failures: 0



ProductoTest [Runner: JUnit 5] (0,689 s)  
testBuscarProductoPorNombre() (0,689 s)

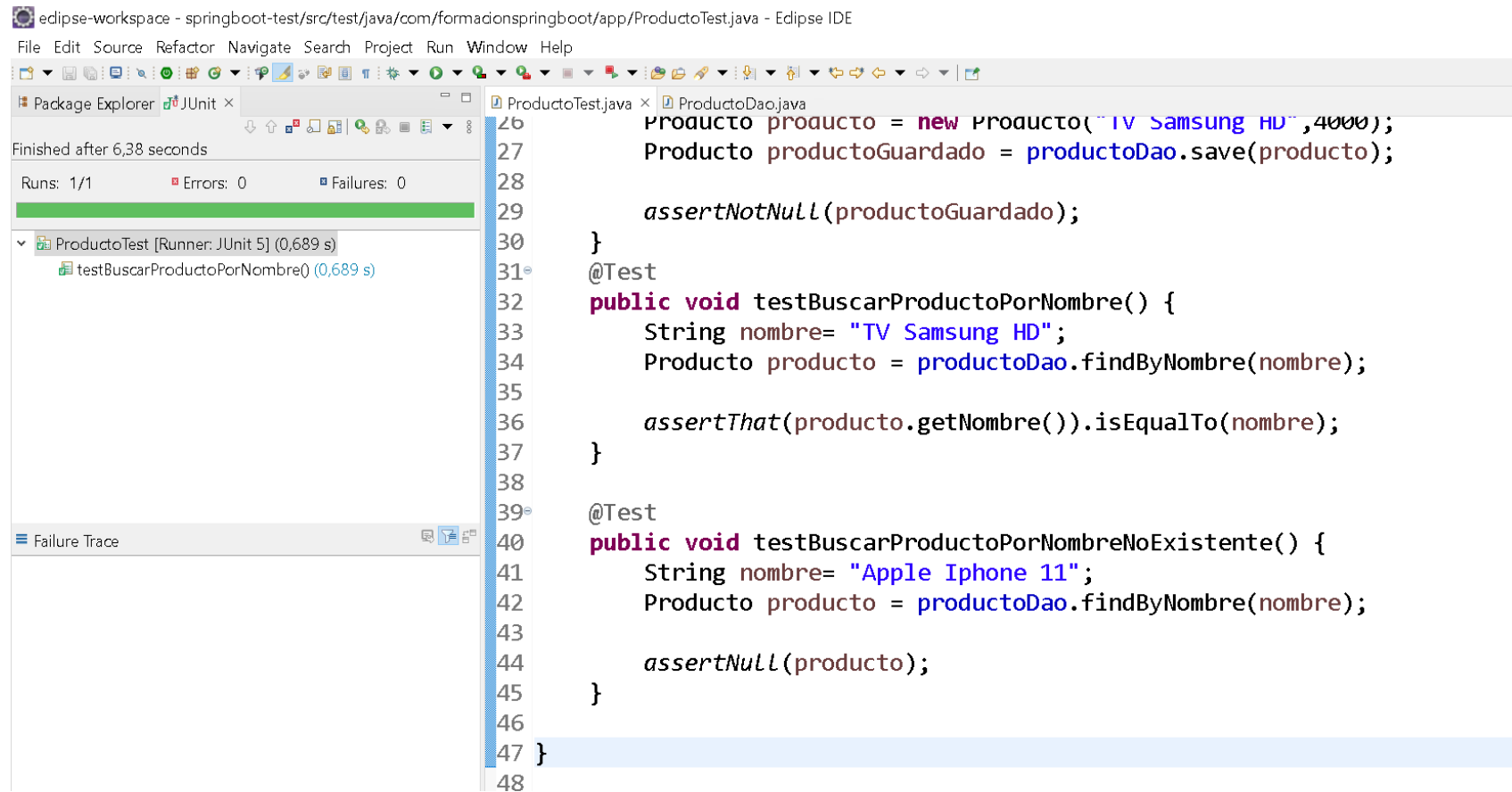
Failure Trace

ProductoTest.java x ProductoDao.java

```
17 public class ProductoTest {
18
19     @Autowired
20     private ProductoDao productoDao;
21
22     @Test
23     @Rollback(false)
24     public void testGuardarProducto() {
25         Producto producto = new Producto("TV Samsung HD", 4000);
26         Producto productoGuardado = productoDao.save(producto);
27
28         assertNotNull(productoGuardado);
29     }
30     @Test
31     public void testBuscarProductoPorNombre() {
32         String nombre= "TV Samsung HD";
33         Producto producto = productoDao.findByNombre(nombre);
34
35         assertThat(producto.getNombre()).isEqualTo(nombre);
36     }
37
38 }
```

```
26     Producto producto = new Producto("TV Samsung HD", 4000);
27     Producto productoGuardado = productoDao.save(producto);
28
29     assertNotNull(productoGuardado);
30 }
31 @Test
32 public void testBuscarProductoPorNombre() {
33     String nombre= "TV Samsung HD";
34     Producto producto = productoDao.findByNombre(nombre);
35
36     assertThat(producto.getNombre()).isEqualTo(nombre);
37 }
38
39 @Test
40 public void testBuscarProductoPorNombreNoExistente() {
41     String nombre= "Apple Iphone 11";
42     Producto producto = productoDao.findByNombre(nombre);
43
44     assertNull(producto);
45 }
46
47 }
48
```

# ejecutamos



The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure, including the test package. The JUnit runner window displays the test results: "Finished after 6,38 seconds", "Runs: 1/1", "Errors: 0", and "Failures: 0". The test "testBuscarProductoPorNombre()" is listed as passed. The main editor shows the code for ProductoTest.java, which includes two test methods: testBuscarProductoPorNombre() and testBuscarProductoPorNombreNoExistente().

```
edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Edipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit x
Finished after 6,38 seconds
Runs: 1/1 Errors: 0 Failures: 0
ProductoTest [Runner: JUnit 5] (0,689 s)
  testBuscarProductoPorNombre() (0,689 s)

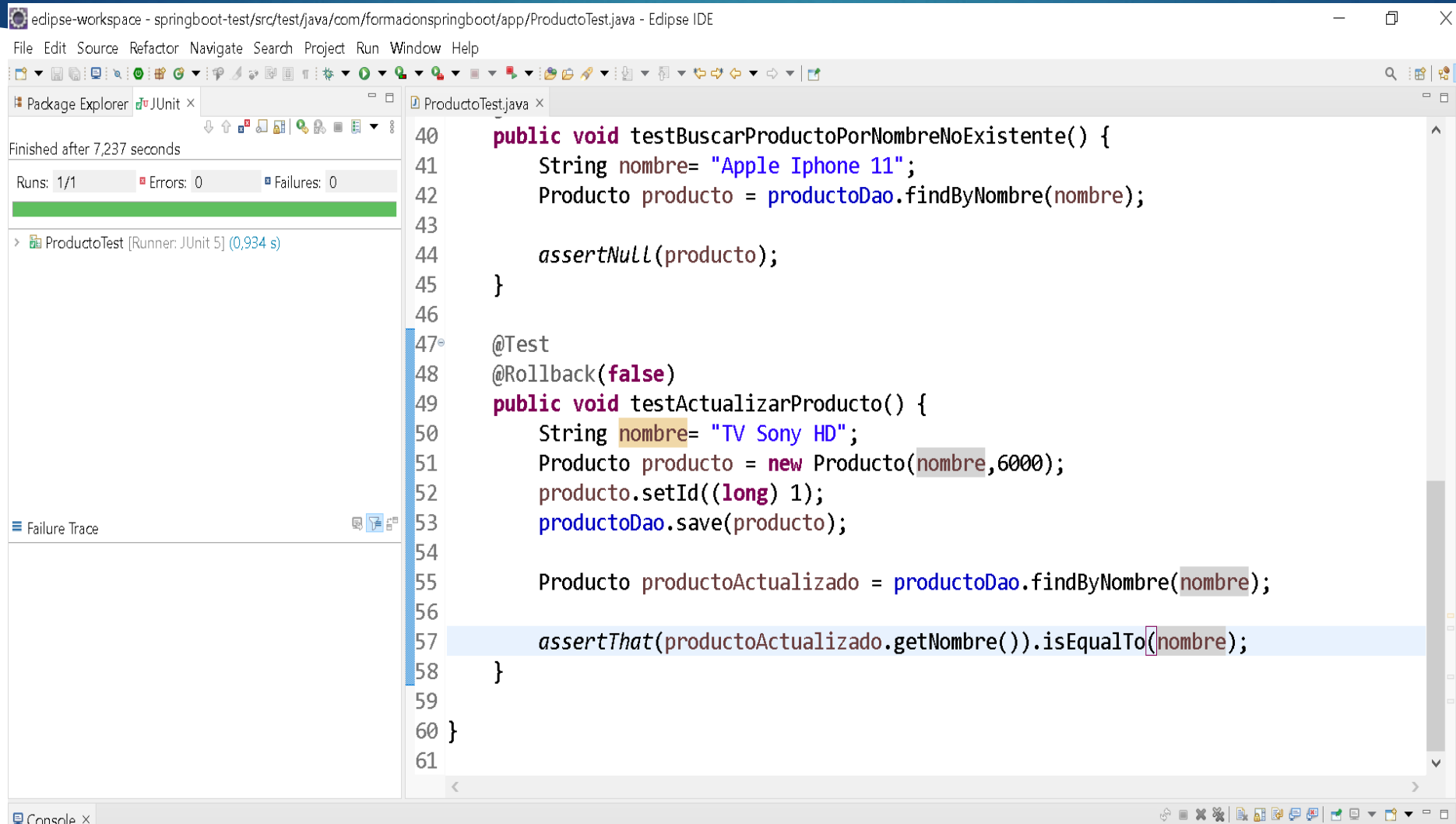
Failure Trace

ProductoTest.java x ProductoDao.java
26 Producto producto = new Producto("TV Samsung HD", 4000);
27 Producto productoGuardado = productoDao.save(producto);
28
29 assertNotNull(productoGuardado);
30 }
31 @Test
32 public void testBuscarProductoPorNombre() {
33     String nombre= "TV Samsung HD";
34     Producto producto = productoDao.findByNombre(nombre);
35
36     assertThat(producto.getNombre()).isEqualTo(nombre);
37 }
38
39 @Test
40 public void testBuscarProductoPorNombreNoExistente() {
41     String nombre= "Apple Iphone 11";
42     Producto producto = productoDao.findByNombre(nombre);
43
44     assertNull(producto);
45 }
46
47 }
48
```

# Método actualizar

```
ProductoTest.java ×
0      public void testBuscarProductoPorNombreNoExistente() {
1          String nombre= "Apple Iphone 11";
2          Producto producto = productoDao.findByNombre(nombre);
3
4          assertNull(producto);
5      }
6
7      @Test
8      @Rollback(false)
9      public void testActualizarProducto() {
10         String nombre= "TV Sony HD";
11         Producto producto = new Producto(nombre,6000);
12         producto.setId((long) 1);
13         productoDao.save(producto);
14
15         Producto productoActualizado = productoDao.findByNombre(nombre);
16
17         assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
18     }
19 }
20 }
```

# Ejecutamos

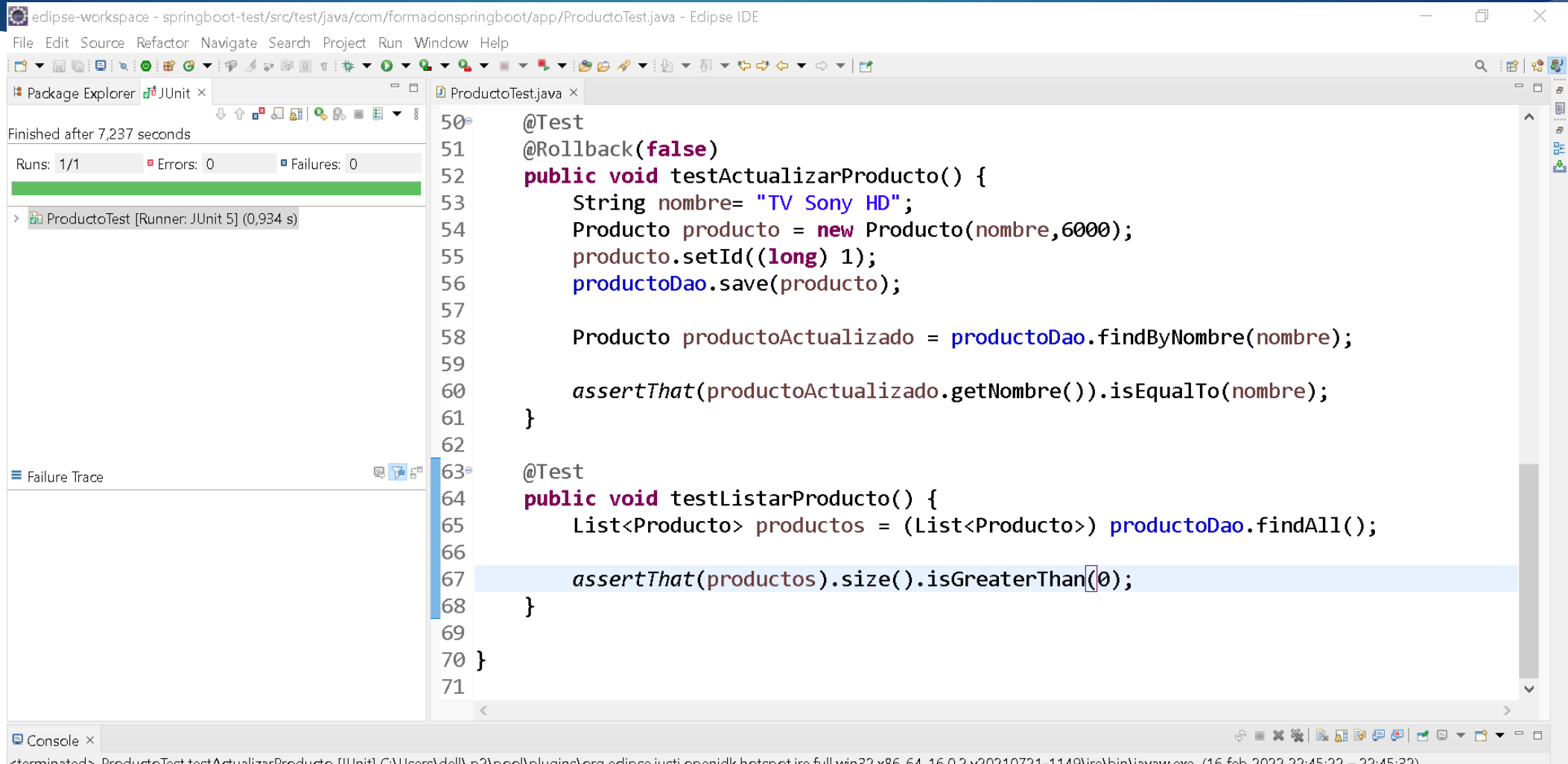


The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Package Explorer on the left shows the project structure, with a green bar indicating a successful test run. The main editor displays the code for ProductoTest.java, which includes two test methods: testBuscarProductoPorNombreNoExistente() and testActualizarProducto(). The testActualizarProducto() method is currently selected and highlighted in blue. The console at the bottom shows the output of the test run, indicating that the test passed successfully.

```
edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer x JUnit x
Finished after 7,237 seconds
Runs: 1/1 Errors: 0 Failures: 0
> ProductoTest [Runner: JUnit 5] (0,934 s)
Failure Trace
Console x
```

```
40 public void testBuscarProductoPorNombreNoExistente() {
41     String nombre= "Apple Iphone 11";
42     Producto producto = productoDao.findByNombre(nombre);
43
44     assertNull(producto);
45 }
46
47 @Test
48 @Rollback(false)
49 public void testActualizarProducto() {
50     String nombre= "TV Sony HD";
51     Producto producto = new Producto(nombre,6000);
52     producto.setId((long) 1);
53     productoDao.save(producto);
54
55     Producto productoActualizado = productoDao.findByNombre(nombre);
56
57     assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
58 }
59
60 }
61
```

# Método Listar



edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit x

Finished after 7,237 seconds

Runs: 1/1 Errors: 0 Failures: 0

> ProductoTest [Runner: JUnit 5] (0,934 s)

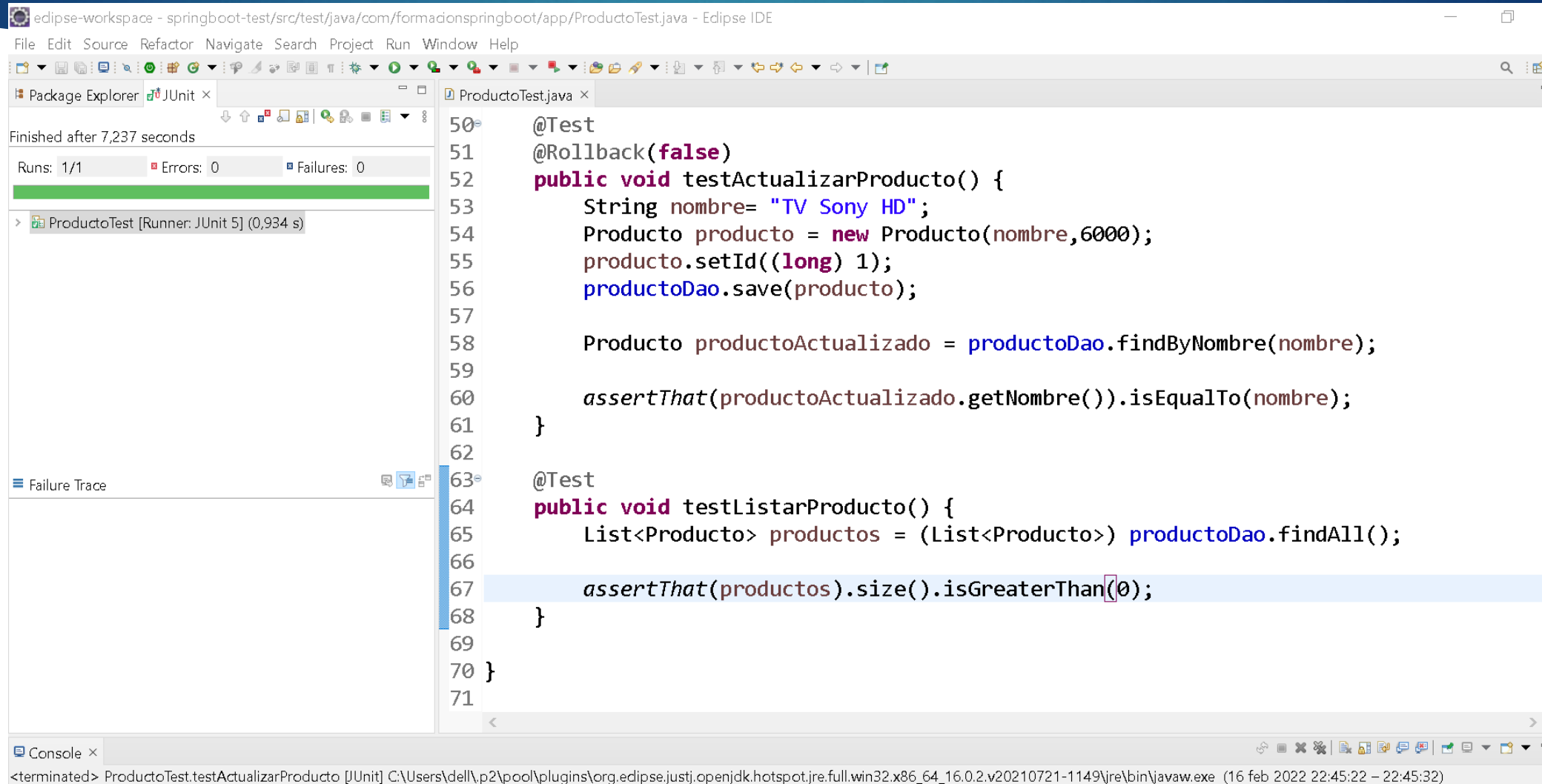
Failure Trace

```
50 @Test
51 @Rollback(false)
52 public void testActualizarProducto() {
53     String nombre= "TV Sony HD";
54     Producto producto = new Producto(nombre,6000);
55     producto.setId((long) 1);
56     productoDao.save(producto);
57
58     Producto productoActualizado = productoDao.findByNombre(nombre);
59
60     assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
61 }
62
63 @Test
64 public void testListarProducto() {
65     List<Producto> productos = (List<Producto>) productoDao.findAll();
66
67     assertThat(productos).size().isGreaterThan(0);
68 }
69
70 }
71
```

Console x

terminated> ProductoTest testActualizarProducto [JUnit] C:\Users\dell\p2\workspace\springboot-test\src\test\java\com\formacionspringboot\app\ProductoTest.java (16 feb 2022 22:45:22 - 22:45:22)

# Ejecutamos



edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit x

Finished after 7,237 seconds

Runs: 1/1 Errors: 0 Failures: 0

> ProductoTest [Runner: JUnit 5] (0,934 s)

Failure Trace

```
50 @Test
51 @Rollback(false)
52 public void testActualizarProducto() {
53     String nombre= "TV Sony HD";
54     Producto producto = new Producto(nombre,6000);
55     producto.setId((long) 1);
56     productoDao.save(producto);
57
58     Producto productoActualizado = productoDao.findByNombre(nombre);
59
60     assertThat(productoActualizado.getNombre()).isEqualTo(nombre);
61 }
62
63 @Test
64 public void testListarProducto() {
65     List<Producto> productos = (List<Producto>) productoDao.findAll();
66
67     assertThat(productos).size().isGreaterThan(0);
68 }
69
70 }
71
```

Console x

<terminated> ProductoTest.testActualizarProducto [JUnit] C:\Users\dell\p2\pool\plugins\org.edipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_16.0.2.v20210721-1149\jre\bin\javaw.exe (16 feb 2022 22:45:22 - 22:45:32)

# Método eliminar

```
ProductoTest.java x
66     public void testListarProducto() {
67         List<Producto> productos = (List<Producto>) productoDao.findAll();
68
69         assertThat(productos).size().isGreaterThan(0);
70     }
71
72     @Test
73     @Rollback(false)
74     public void testBorrarProducto() {
75         Long id=(long) 1;
76         boolean existe = productoDao.findById(id).isPresent();
77
78         productoDao.deleteById(id);
79
80         boolean existe2 = productoDao.findById(id).isPresent();
81
82         assertTrue(existe);
83         assertFalse(existe2);
84     }
85
86 }
87
```



# Ejecutamos

edipse-workspace - springboot-test/src/test/java/com/formacionspringboot/app/ProductoTest.java - Edipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer JUnit x

Finished after 6,478 seconds

Runs: 1/1 Errors: 0 Failures: 0

> ProductoTest [Runner: JUnit 5] (0,613 s)

Failure Trace

```
66 public void testListarProducto() {
67     List<Producto> productos = (List<Producto>) productoDao.findAll();
68
69     assertThat(productos).size().isGreaterThan(0);
70 }
71
72 @Test
73 @Rollback(false)
74 public void testBorrarProducto() {
75     Long id=(long) 1;
76     boolean existe = productoDao.findById(id).isPresent();
77
78     productoDao.deleteById(id);
79
80     boolean existe2 = productoDao.findById(id).isPresent();
81
82     assertTrue(existe);
83     assertFalse(existe2);
84 }
85
86 }
87
```