

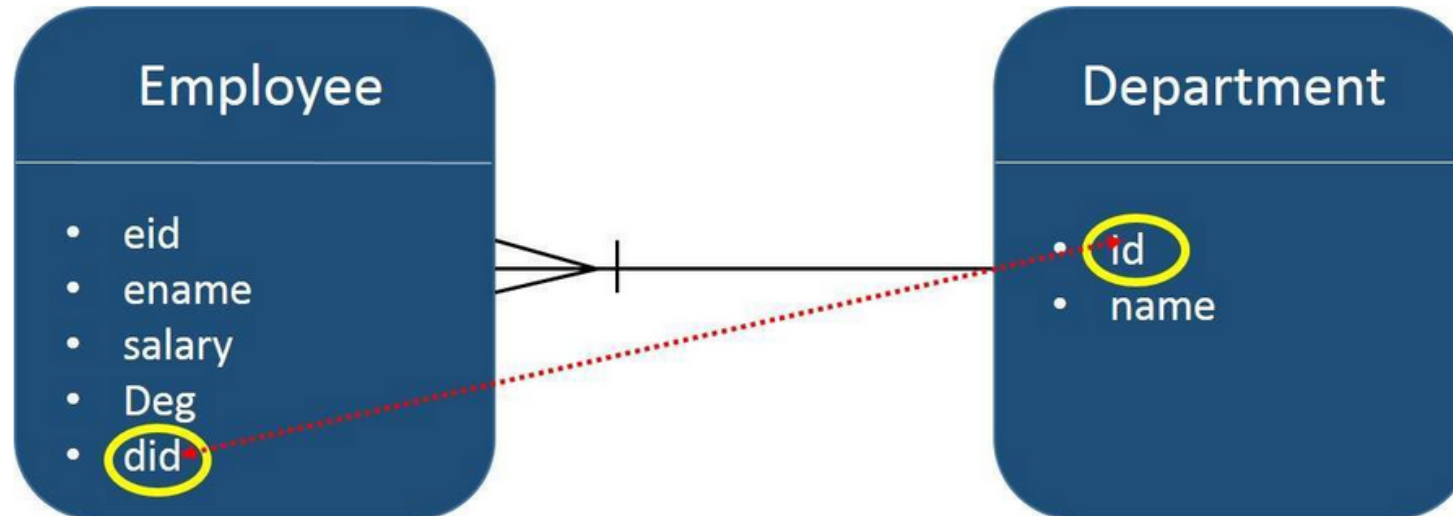
Relaciones entre entitys

- ▶ 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;

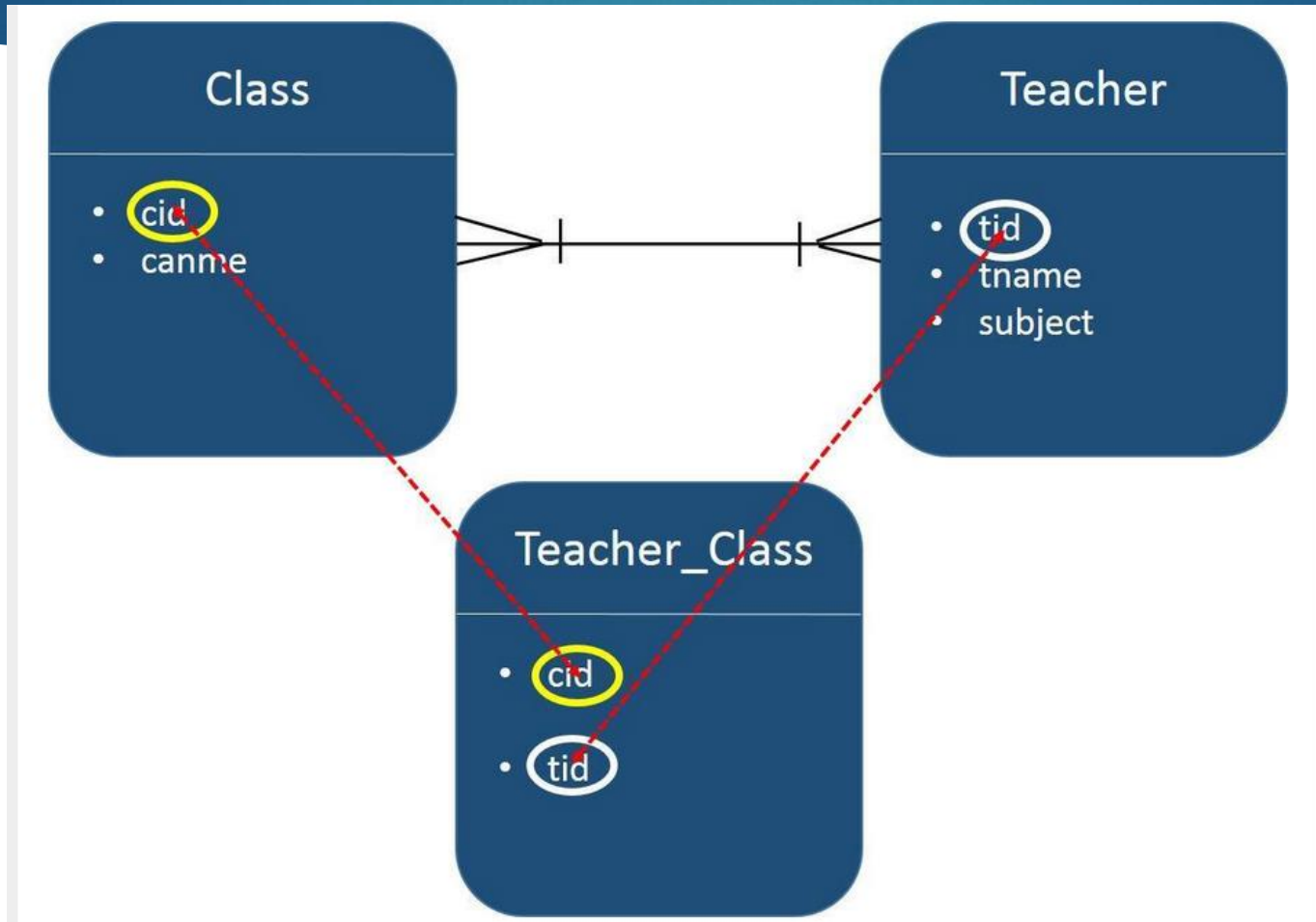
    @OneToOne
    private Department department;

    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;

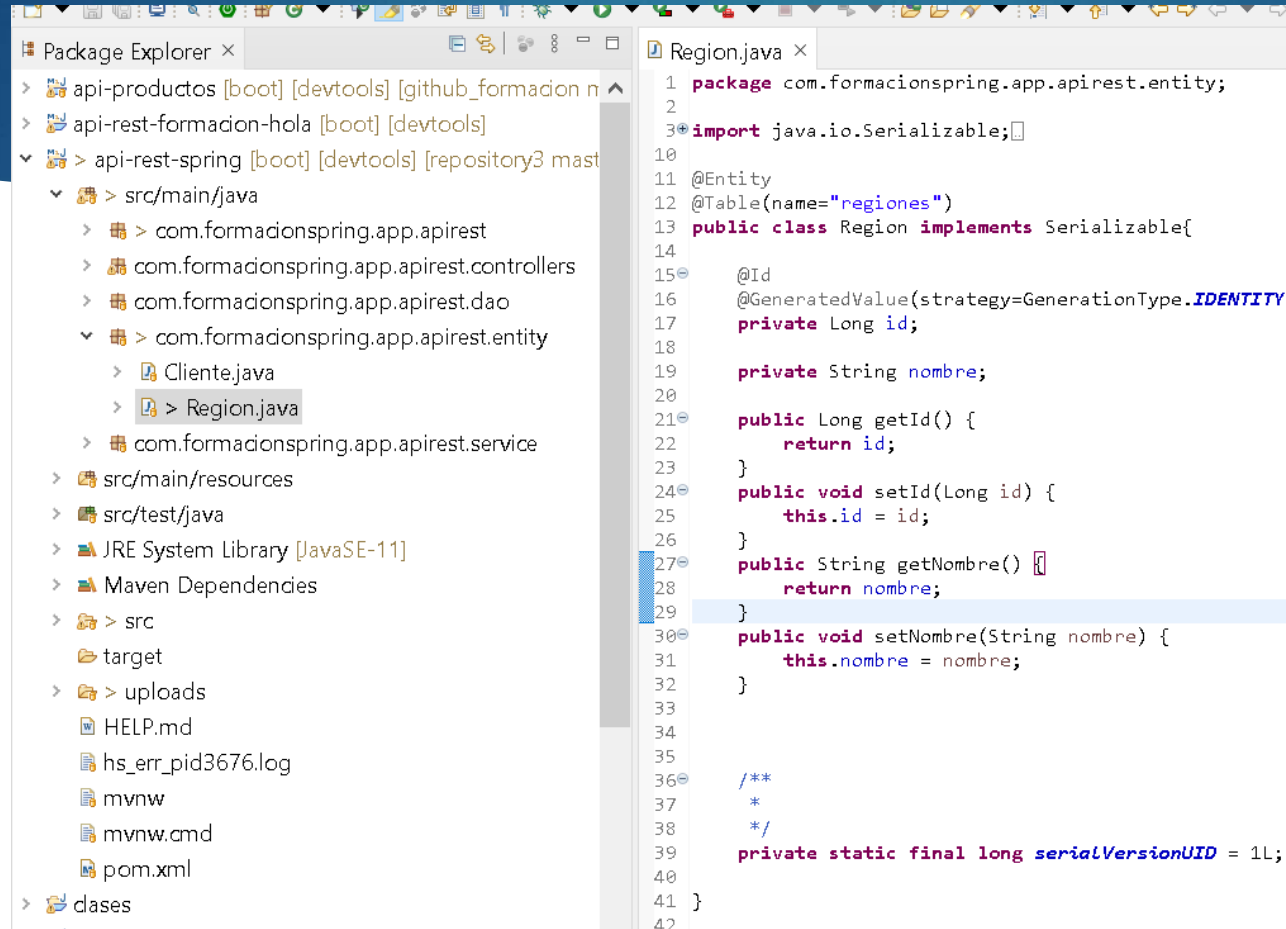
    public Teacher()
    {
        super();
    }

    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

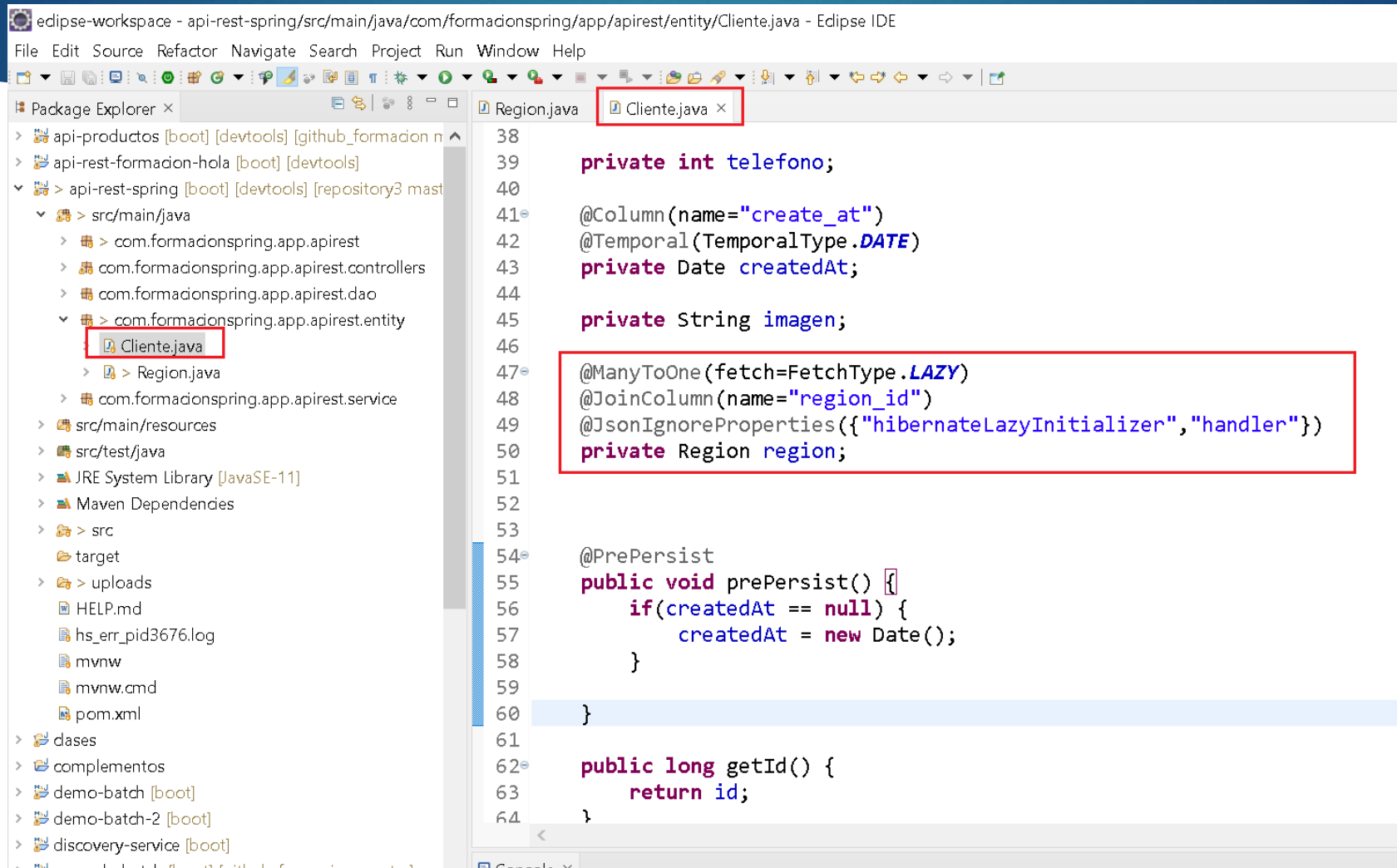


The screenshot shows an IDE with two panels. The left panel is the Package Explorer, showing a project structure with several packages. The right panel shows the code for Region.java.

```
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
```

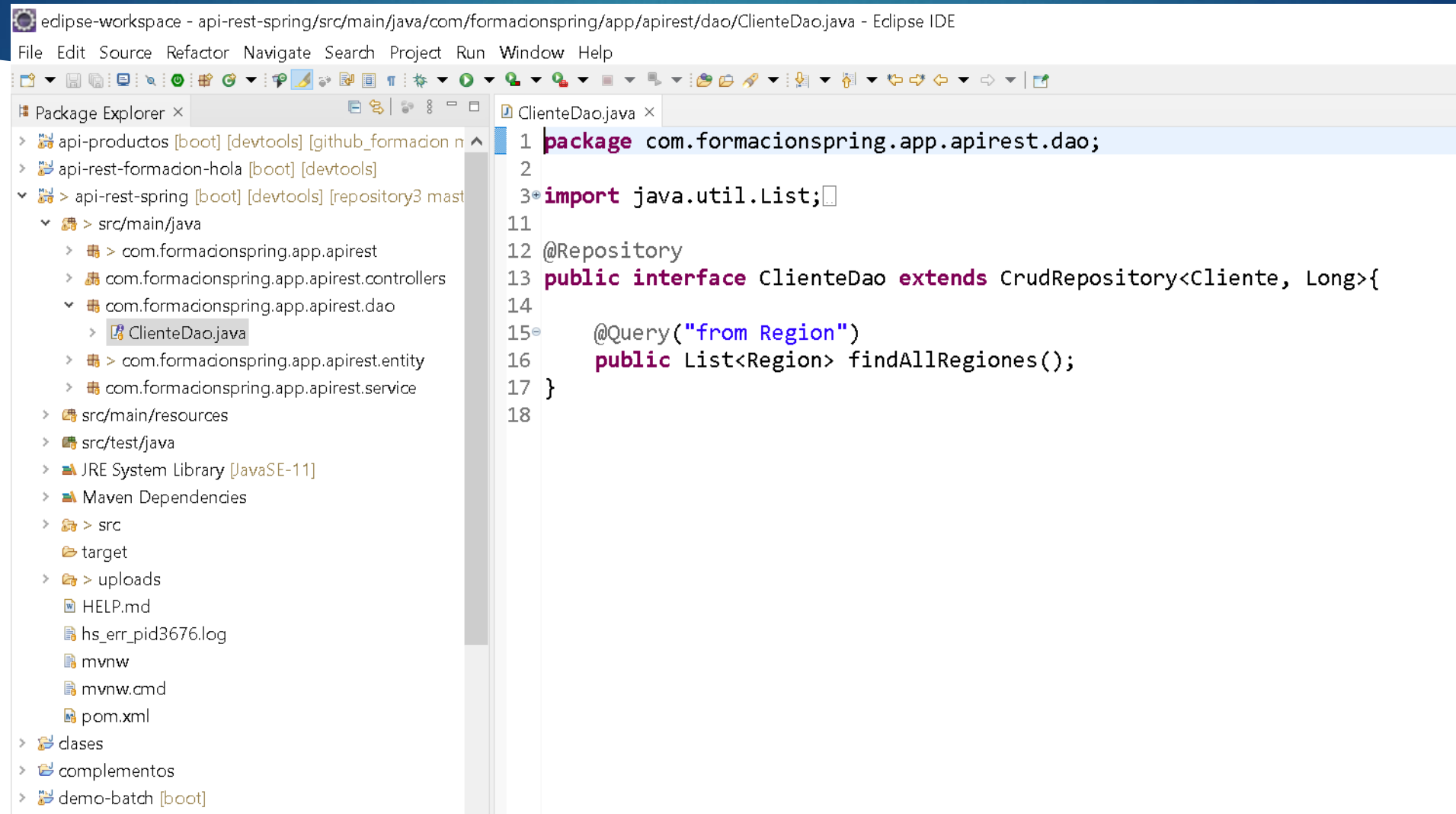
En la entity de Cliente agregamos la relación, Uso de Lazy: En la relación @ManyToOne hemos hecho uso de lazy para evitar una carga activa y traer todo, lo que podría afectar en el rendimiento de nuestra aplicación.



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`, `create_at`, `createdAt`, and `imagen`. A `@ManyToOne` relationship is defined with `fetch=FetchType.LAZY` and `@JoinColumn(name="region_id")`. The `@JsonIgnoreProperties` annotation is used to ignore `hibernateLazyInitializer` and `handler`. A `@PrePersist` method is also present, which checks if `createdAt` is null and sets it to the current date if so. The `getId` method is also shown.

```
38
39 private int telefono;
40
41 @Column(name="create_at")
42 @Temporal(TemporalType.DATE)
43 private Date createdAt;
44
45 private String imagen;
46
47 @ManyToOne(fetch=FetchType.LAZY)
48 @JoinColumn(name="region_id")
49 @JsonIgnoreProperties({"hibernateLazyInitializer","handler"})
50 private Region region;
51
52
53
54 @PrePersist
55 public void prePersist() {
56     if(createdAt == null) {
57         createdAt = new Date();
58     }
59 }
60
61
62 public long getId() {
63     return id;
64 }
```

En nuestro repositorio dao el siguiente método



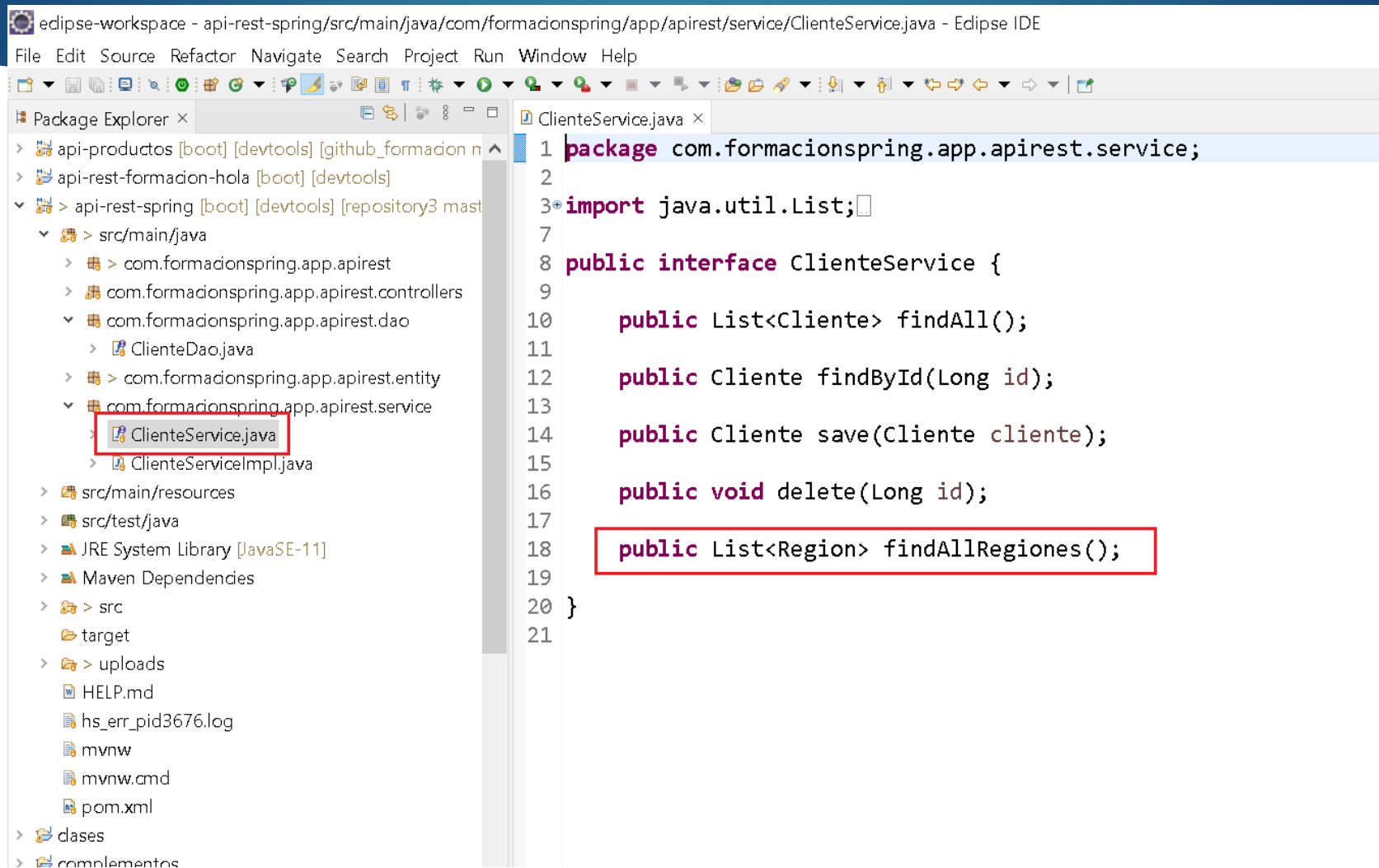
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer displays the project structure. The main editor on the right shows the source code of the `ClienteDao.java` file. The code defines a package, imports `java.util.List`, and declares a `@Repository` annotated `public interface` that extends `CrudRepository<Cliente, Long>`. It includes a `@Query("from Region")` annotation and a `findAllRegiones()` method.

```
edipse-workspace - api-rest-spring/src/main/java/com/formacionspring/app/apiest/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]
v > api-rest-spring [boot] [devtools] [repository3 mast
  v > src/main/java
    > com.formacionspring.app.apiest
    > com.formacionspring.app.apiest.controllers
    v com.formacionspring.app.apiest.dao
      > ClienteDao.java
    > 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]

ClienteDao.java x
1 package com.formacionspring.app.apiest.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
```

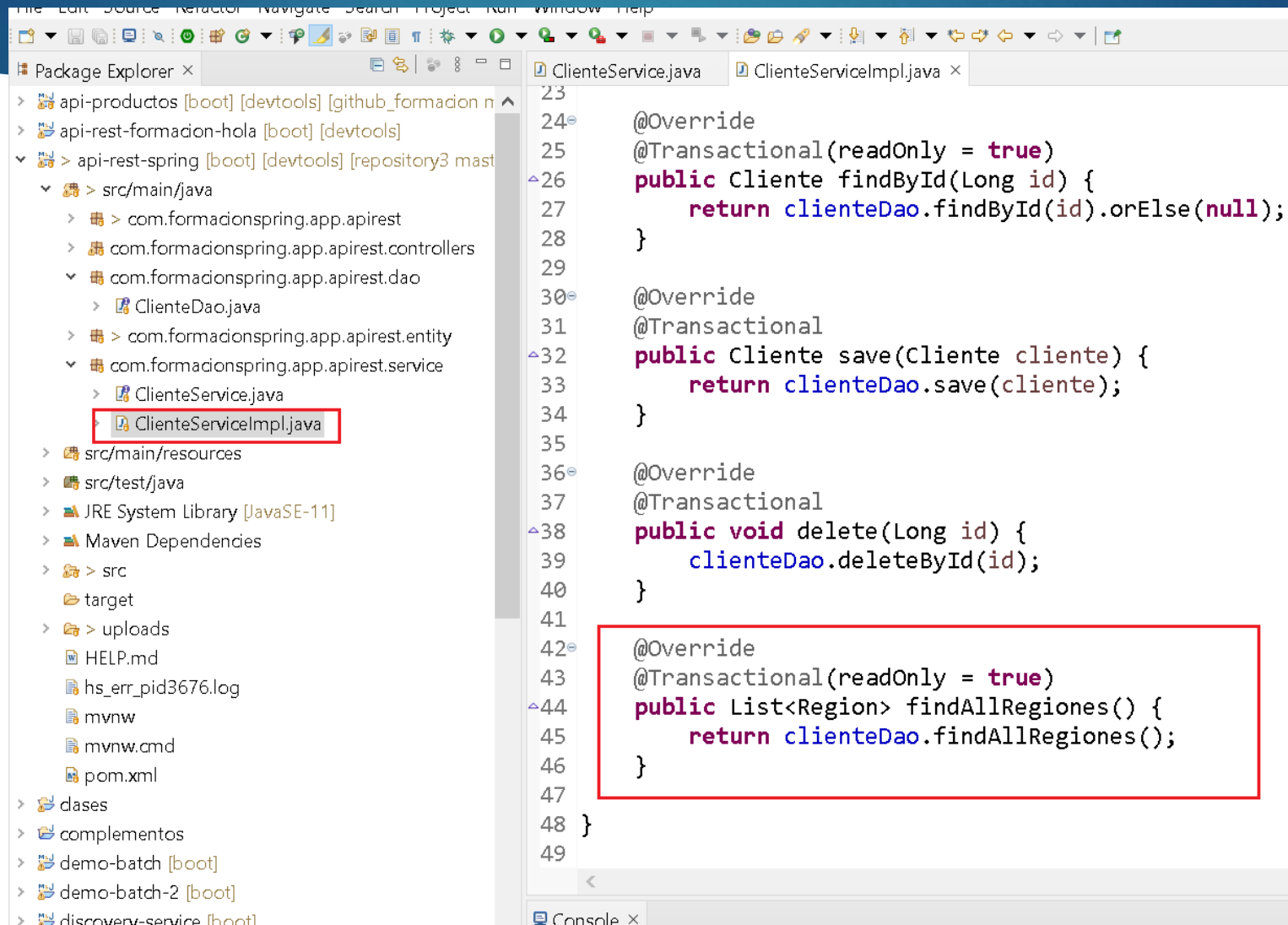
A la interfaz de servicio agregamos el nuevo método



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 (highlighted with a red box)
 - 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()` (lines 42-46) 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

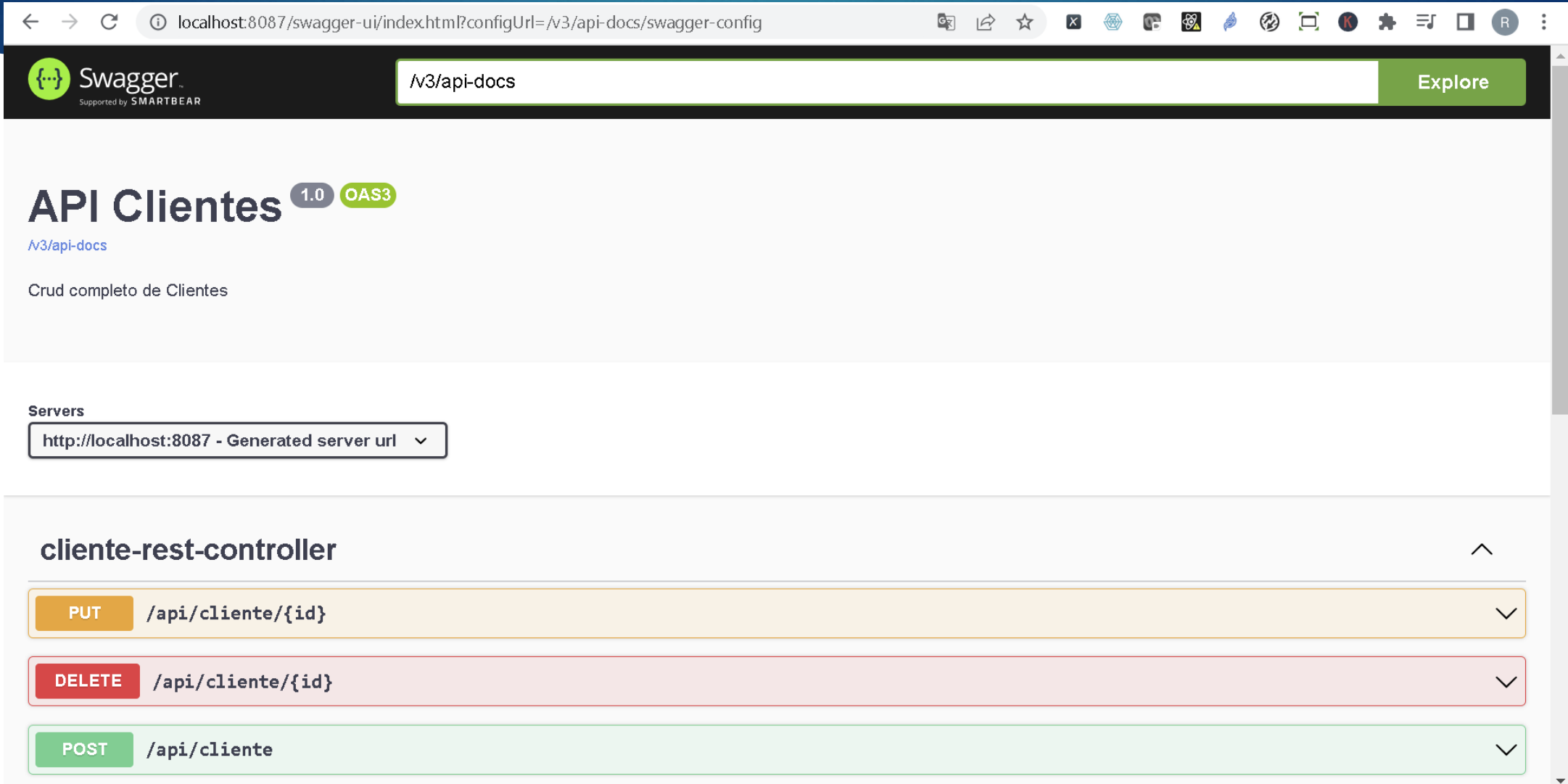
Package Explorer ×

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 and 'Supported by SMARTBEAR' are in the top left. 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 it, the URL `/v3/api-docs` and the description 'Crud completo de Clientes' are shown. A 'Servers' section has a dropdown menu set to 'http://localhost:8087 - Generated server url'. The 'cliente-rest-controller' section is expanded, showing three API endpoints: a PUT request to `/api/cliente/{id}`, a DELETE request to `/api/cliente/{id}`, and a POST request to `/api/cliente`.

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

