

Creamos un proyecto en eclipse

- ▶ creamos un proyecto con las siguientes dependencias
- ▶ Spring Boot DevTools
- ▶ Spring Data JPA
- ▶ MySQL Driver
- ▶ Spring Web

Dependencias

- ▶ **Spring Boot DevTools** es la herramienta de Spring Boot que nos permite reiniciar de forma automática nuestras aplicaciones cada vez que se produce un cambio en nuestro código.
- ▶ **Spring Data JPA** Puede entenderse como el reencapsulado y abstracción de la especificación JPA. La capa inferior todavía se implementa utilizando la tecnología JPA de Hibernate, y se hace referencia al lenguaje de consulta JPQL (Java Persistence Query Language), que es parte de todo el ecosistema Spring
- ▶ **MySQL Driver** El concepto de Driver hace referencia al conjunto de clases necesarias que implementa de forma nativa el protocolo de comunicación con la base de datos MySQL.
- ▶ **Spring Web** proporciona integración HTTP central, incluidos algunos filtros Servlet útiles, Spring HTTP Invoker, infraestructura para integrarse con otros marcos web y tecnologías HTTP.



New Spring Starter Project Dependencies



Spring Boot Version: 2.5.7

Frequently Used:

- MySQL Driver
- Spring Web

- Spring Boot DevTools

- Spring Data JPA

Available:

Type to search dependences

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops

Selected:

- Spring Boot DevTools
- Spring Data JPA
- MySQL Driver
- Spring Web



< Back

Next >

Finish

Cancel

New Spring Starter Project

Service URL

Name

Use default location

Location

Type: Packaging:

Java Version: Language:

Group

Artifact

Version

Description

Package

Working sets

Add project to working sets

Working sets:

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer View:** On the left, it lists several projects. The project "springboot-apirest [boot] [devtools]" is selected and highlighted with a red border. Inside this project, the file "SpringbootApirestApplication.java" is open and displayed in the central editor.
- Editor View:** The central area displays the code for "SpringbootApirestApplication.java". The code is as follows:

```
1 package com.formacionjava.springboot.apirest;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class SpringbootApirestApplication {
7
8
9     public static void main(String[] args) {
10         SpringApplication.run(SpringbootApirestApplication.class, args);
11     }
12
13 }
14
```

The line "@SpringBootApplication" is highlighted with a red rectangular box. The code is written in a standard Java syntax with some specific annotations from the Spring Boot framework.

Bottom Status Bar: The status bar at the bottom shows various icons and labels: Markers, Properties, Servers, Data Source Explorer, Snippets, and Console. Below the status bar, a message states: "No consoles to display at this time."

@SpringBootApplication puede usar una sola anotación para habilitar esas tres funciones

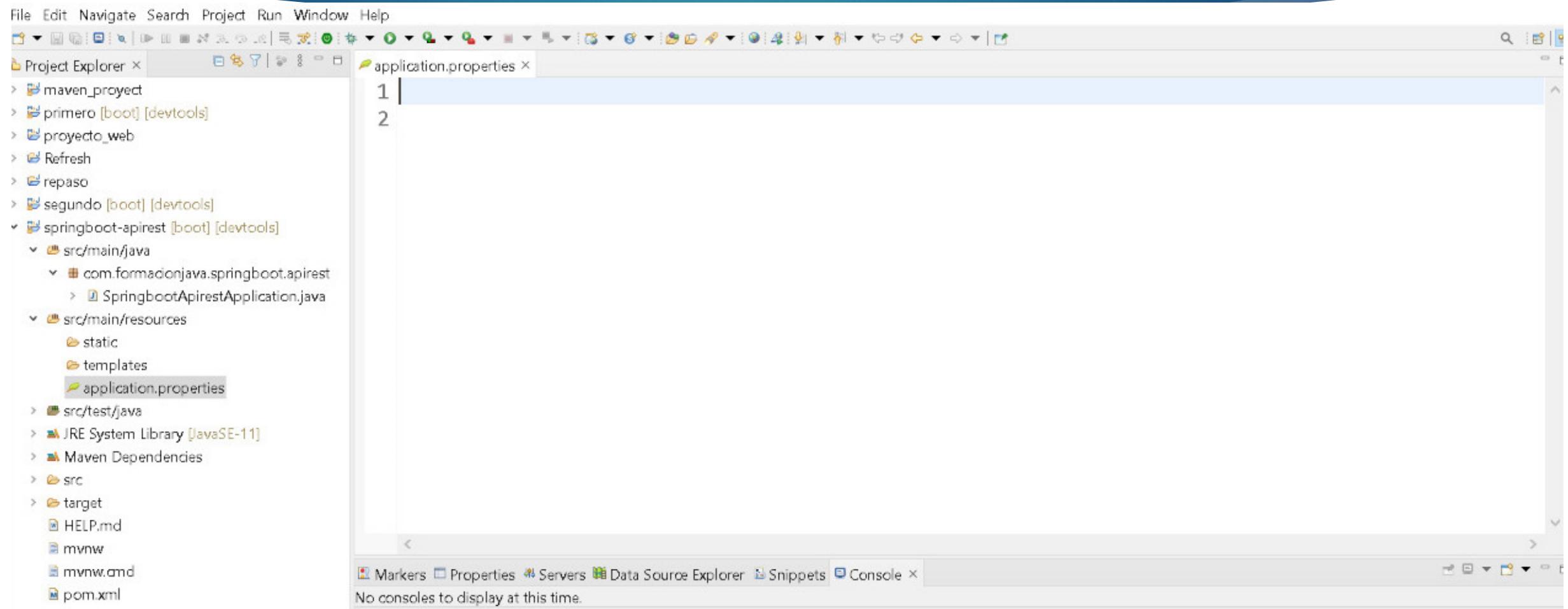
- ▶ @EnableAutoConfiguration: habilita el mecanismo de configuración automática de Spring Boot
- ▶ @ComponentScan: habilite el @Componentescaneo en el paquete donde se encuentra la aplicación (consulte las mejores prácticas)
- ▶ @Configuration: permite registrar beans adicionales en el contexto o importar clases de configuración adicionales
- ▶ La @SpringBootApplicationanotación es equivalente a usar @Configuration, @EnableAutoConfigurationy @ComponentScancon sus atributos predeterminados.

The screenshot shows a Java development environment with the following interface elements:

- Project Explorer**: On the left, it lists several Maven projects. One project, "springboot-apirest [boot] [devtools]", is expanded, revealing its structure: src/main/java, src/main/resources, src/test/java, JRE System Library [JavaSE-11], Maven Dependencies, src, target, HELP.md, mvnw, mvnw.cmd, and pom.xml. This entire section is highlighted with a red rectangular border.
- Code Editor**: The main workspace displays the file `SpringbootApirestApplication.java`. The code is annotated with various annotations:

```
44 * {@code @EnableAutoConfiguration} and {@code @ComponentScan}.
45 *
46 * @author Phillip Webb
47 * @author Stephane Nicoll
48 * @author Andy Wilkinson
49 * @since 1.2.0
50 */
51 @Target(ElementType.TYPE)
52 @Retention(RetentionPolicy.RUNTIME)
53 @Documented
54 @Inherited
55 @SpringBootConfiguration
56 @EnableAutoConfiguration
57 @ComponentScan(excludeFilters = { @Filter(type = FilterType.CUSTOM, classes = TypeExcl}
58                               @Filter(type = FilterType.CUSTOM, classes = AutoConfigurationExcludeFilter.clas
59 public @interface SpringBootApplication {
60
```
- Toolbars**: Standard Java IDE toolbars are visible at the top.
- Bottom Bar**: Includes tabs for Markers, Properties, Servers, Data Source Explorer, Snippets, and Console, along with a message: "No consoles to display at this time."

Iniciamos la configuración en application.properties se encuentra en src/main/resource

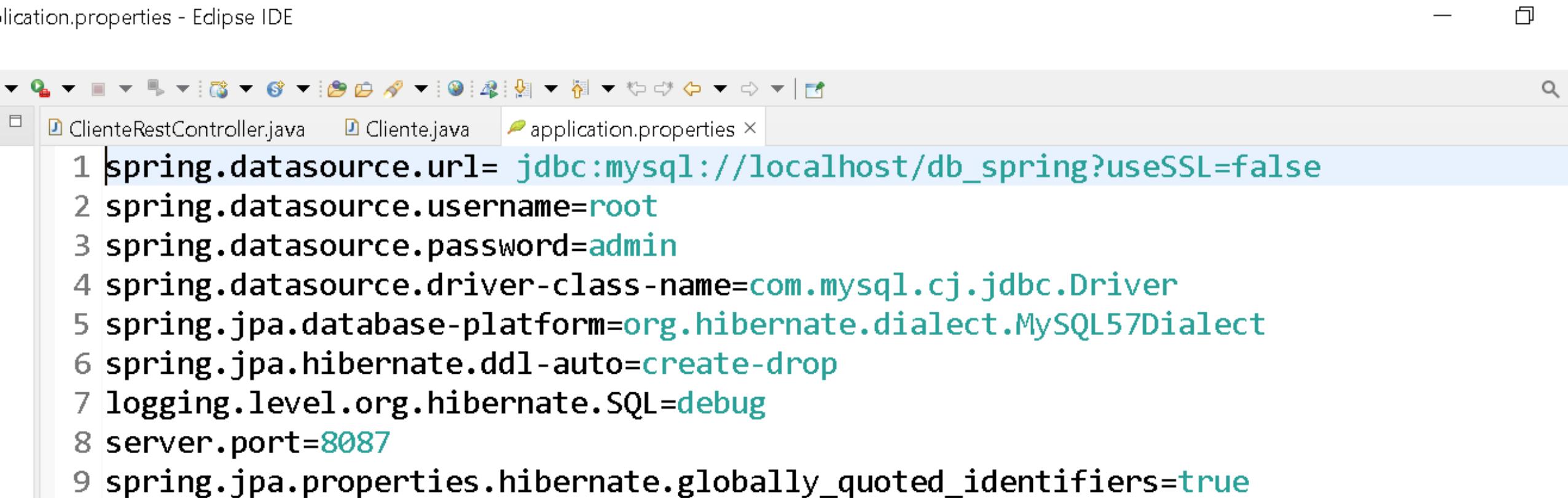


Configuramos con datos para conectarnos a la base de datos

```
application.properties
1 spring.datasource.url= jdbc:mysql://localhost/db_spring
2 spring.datasource.username=root
3 spring.datasource.password=admin
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=create-drop
7 logging.level.org.hibernate.SQL=debug
```

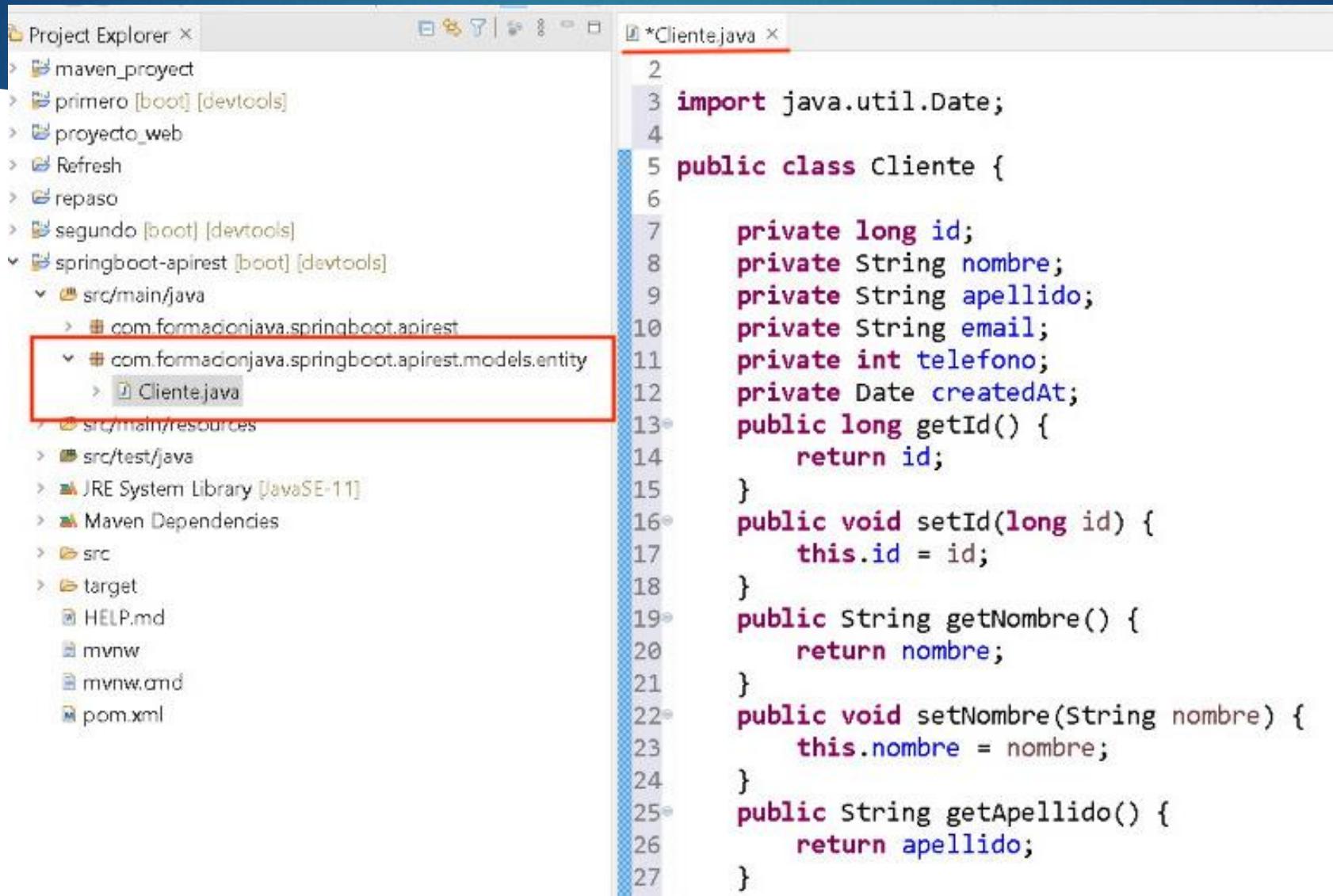
Si tenemos problemas con el puerto, podemos agregar estas dos líneas de código, server.port=8087, spring.jpa.properties.hibernate.globally_quoted_identifiers=true

application.properties - Eclipse IDE



```
1 spring.datasource.url= jdbc:mysql://localhost/db_spring?useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=admin
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=create-drop
7 logging.level.org.hibernate.SQL=debug
8 server.port=8087
9 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
```

Creamos un paquete y una clase para nuestro modelo Cliente con los siguientes atributos y sus correspondientes setters y getters



The screenshot shows a Java development environment with the following details:

- Project Explorer:** Shows the project structure. A red box highlights the package `com.formacionjava.springboot.apirest.models.entity` and the class `Cliente.java`.
- Code Editor:** Displays the `Cliente.java` file with the following code:

```
2 import java.util.Date;
3
4 public class Cliente {
5
6     private long id;
7     private String nombre;
8     private String apellido;
9     private String email;
10    private int telefono;
11    private Date createdAt;
12    public long getId() {
13        return id;
14    }
15    public void setId(long id) {
16        this.id = id;
17    }
18    public String getNombre() {
19        return nombre;
20    }
21    public void setNombre(String nombre) {
22        this.nombre = nombre;
23    }
24    public String getApellido() {
25        return apellido;
26    }
27}
```

Project Explorer

maven_proyect
primero [boot] [devtools]
proyecto_web
Refresh
repasso
segundo [boot] [devtools]
springboot-apirest [boot] [devtools]
src/main/java
 com.formacionjava.springboot.apirest
 com.formacionjava.springboot.apirest.models.entity
 Cliente.java
src/main/resources
src/test/java
JRE System Library [JavaSE-11]
Maven Dependencies
src
target
HELP.md
mvnw
mvnw.cmd
pom.xml

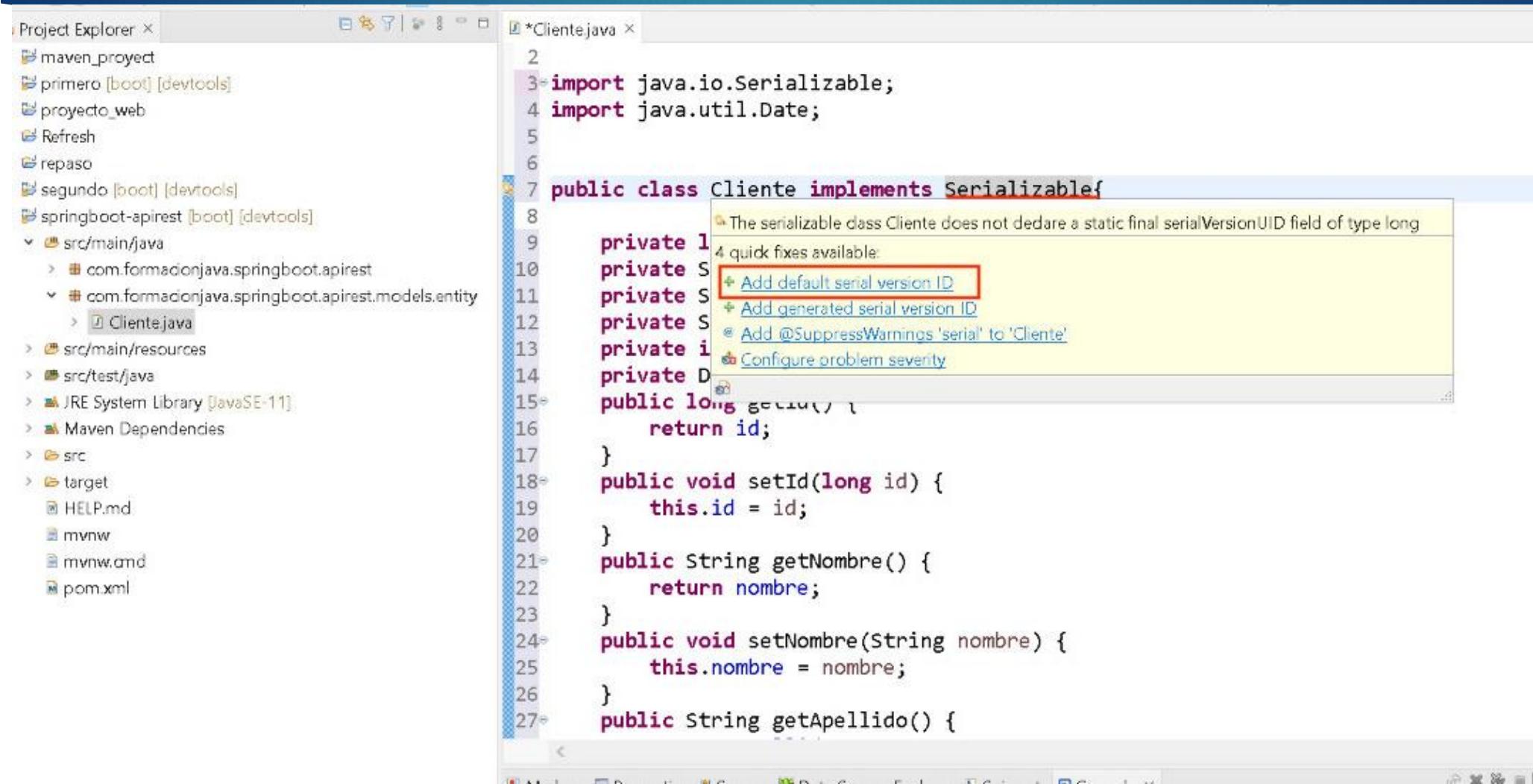
*Cliente.java

```
16°     public void setId(long id) {  
17°         this.id = id;  
18°     }  
19°     public String getNombre() {  
20°         return nombre;  
21°     }  
22°     public void setNombre(String nombre) {  
23°         this.nombre = nombre;  
24°     }  
25°     public String getApellido() {  
26°         return apellido;  
27°     }  
28°     public void setApellido(String apellido) {  
29°         this.apellido = apellido;  
30°     }  
31°     public String getEmail() {  
32°         return email;  
33°     }  
34°     public void setEmail(String email) {  
35°         this.email = email;  
36°     }  
37°     public int getTelefono() {  
38°         return telefono;  
39°     }  
40°     public void setTelefono(int telefono) {  
41°         this.telefono = telefono;
```

Serializable

- ▶ Es una clase ubicada en el paquete **Java.io.Serializable**, la cual no cuenta con ningún método, por lo que es una clase que sirve solamente para especificar que todo el estado de un objeto instanciado podrá ser escrito o enviado en la red como una trama de bytes.
- ▶ Hay que tomar en cuenta que, si un objeto tiene como atributo otro objeto, entonces se debe declarar ese objeto del tipo **Serializable**, y estos serán serializados primero. Esto se puede tomar como un árbol, donde las hojas son objetos que forman parte de otro objeto como un atributo, y todos ellos son marcados como serializables, para así entonces serializar primero las hojas del árbol y finalizar con la raíz.

Implementar Serializable



The screenshot shows an IDE interface with the Project Explorer on the left and the code editor on the right. The code editor displays the following Java code:

```
1 import java.io.Serializable;
2 import java.util.Date;
3
4 public class Cliente implements Serializable {
5     private long id;
6     private String nombre;
7     private String apellido;
8
9     public long getId() {
10         return id;
11     }
12
13     public void setId(long id) {
14         this.id = id;
15     }
16
17     public String getNombre() {
18         return nombre;
19     }
20
21     public void setNombre(String nombre) {
22         this.nombre = nombre;
23     }
24
25     public String getApellido() {
26
27 }
```

A tooltip is displayed over the line 'public class Cliente implements Serializable{'. The tooltip contains the message: 'The serializable class Cliente does not declare a static final serialVersionUID field of type long' and lists four quick fixes:

- + Add default serial version ID
- + Add generated serial version ID
- @ Add @SuppressWarnings('serial') to 'Cliente'
- Configure problem severity

The screenshot shows a Java application structure in the Project Explorer and the corresponding code in the editor.

Project Explorer:

- maven_project
- primero [boot] [devtools]
- projeto_web
- Refresh
- repaso
- segundo [boot] [devtools]
- springboot-apirest [boot] [devtools]
- src/main/java
 - com.formacionjava.springboot.apirest
 - com.formacionjava.springboot.apirest.models.entity
 - Cliente.java
- src/main/resources
- src/test/java
- JRE System Library [JavaSE-11]
- Maven Dependencies
- src
- target
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

Code Editor (Cliente.java):

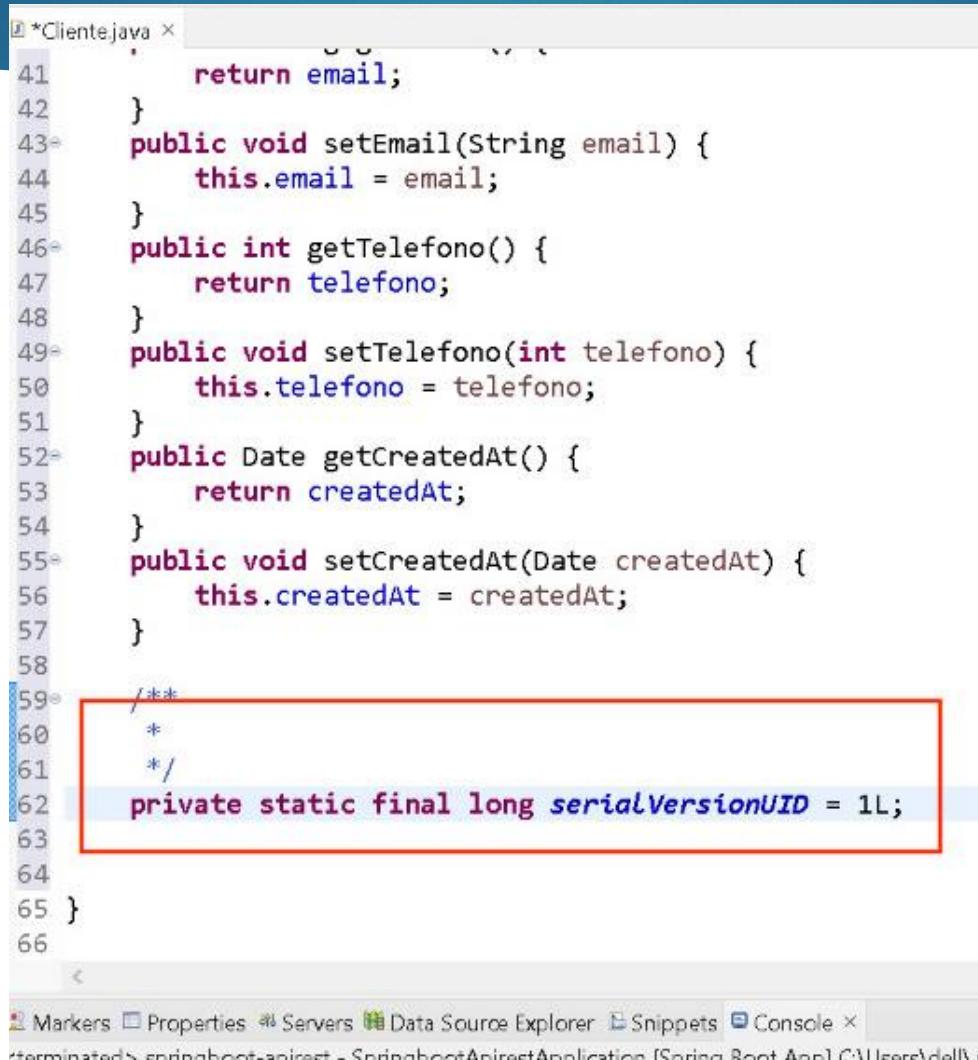
```
1 package com.formacionjava.springboot.apirest.models.entity;
2
3 import java.io.Serializable;
4 import java.util.Date;
5
6 import javax.persistence.Entity;
7 import javax.persistence.Table;
8
9
10 @Entity
11 @Table(name = "clientes")
12 public class Cliente implements Serializable{
13
14     /**
15      *
16     */
17     private static final long serialVersionUID = 1L;
18
19     private long id;
20     private String nombre;
21     private String apellido;
22     private String email;
23     private int telefono;
24     private Date createdAt;
25     public long getId() {
26         return id;
27     }
28 }
```

The line `private static final long serialVersionUID = 1L;` is highlighted with a red rectangle.

serialVersionUID

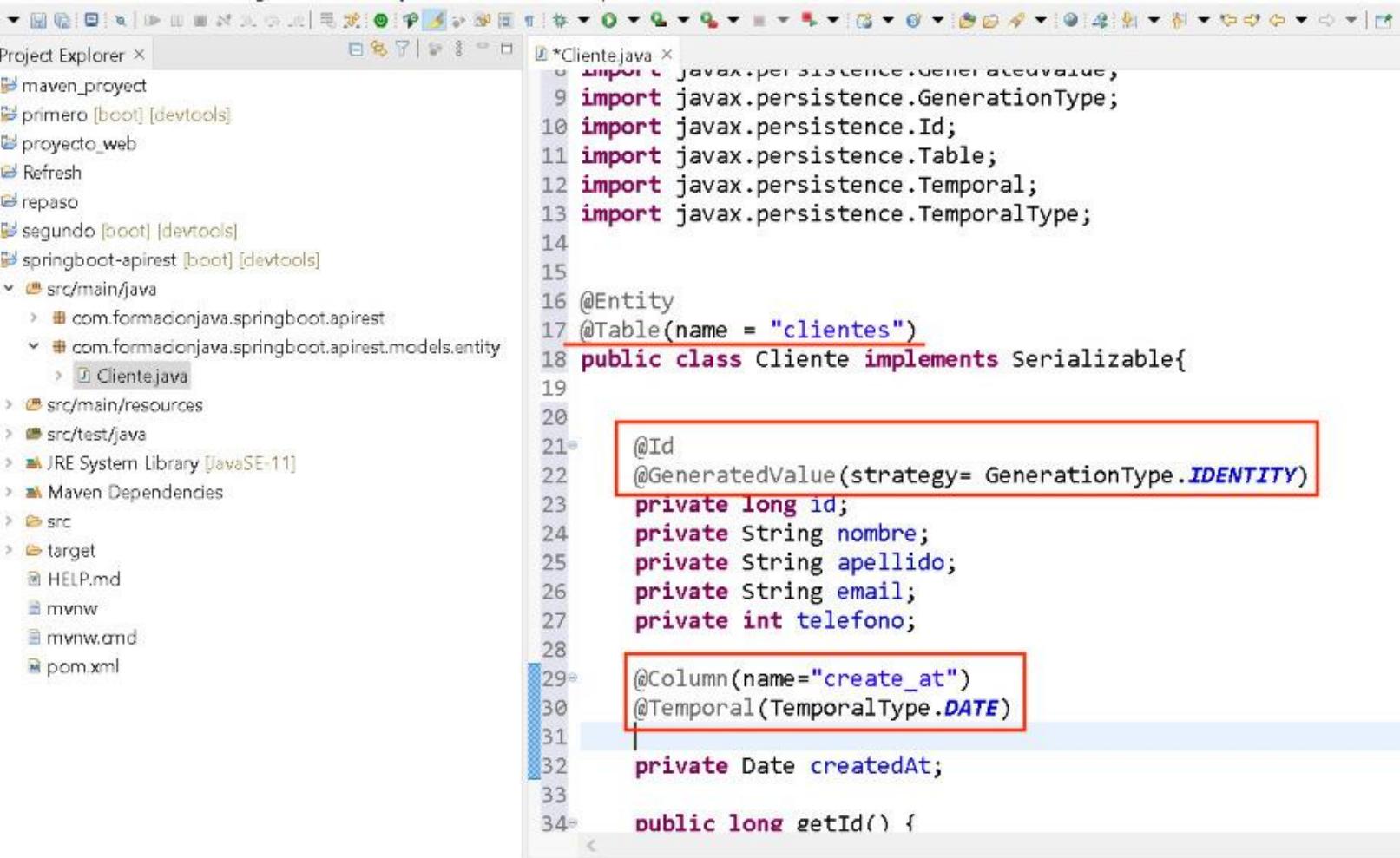
- ▶ El tiempo de ejecución de serialización asocia con cada clase serializable un número de versión, llamada serialVersionUID, que se usa durante la deserialización para verificar que el remitente el receptor de un objeto serializado hayan cargado clases para ese objeto que sean compatibles con respecto a la serialización. Si el receptor ha cargado una clase para el objeto que tiene una clase diferente serialVersionUID a la del remitente correspondiente, entonces la deserialización resultará en un InvalidClassException. Una clase serializable puede declarar la suya serialVersionUID explícitamente declarando un campo con nombre serialVersionUID que debe ser estático, final y de tipo long

Pasamos serialVersionUID a final de la clase



```
*Cliente.java x
41         return email;
42     }
43     public void setEmail(String email) {
44         this.email = email;
45     }
46     public int getTelefono() {
47         return telefono;
48     }
49     public void setTelefono(int telefono) {
50         this.telefono = telefono;
51     }
52     public Date getCreatedAt() {
53         return createdAt;
54     }
55     public void setCreatedAt(Date createdAt) {
56         this.createdAt = createdAt;
57     }
58
59 /**
60  *
61  */
62 private static final long serialVersionUID = 1L;
63
64
65 }
```

Agregamos las siguientes anotaciones



The screenshot shows an IDE interface with the Project Explorer on the left and the code editor on the right. The code editor displays a Java class named Cliente.java. Two specific annotations are highlighted with red boxes:

```
*Cliente.java x
1 import java.util.Date;
2 import javax.persistence.GeneratedValue;
3 import javax.persistence.GenerationType;
4 import javax.persistence.Id;
5 import javax.persistence.Table;
6 import javax.persistence.Temporal;
7 import javax.persistence.TemporalType;
8
9
10 @Entity
11 @Table(name = "clientes")
12 public class Cliente implements Serializable{
13
14     @Id
15     @GeneratedValue(strategy= GenerationType.IDENTITY)
16     private long id;
17     private String nombre;
18     private String apellido;
19     private String email;
20     private int telefono;
21
22     @Column(name="create_at")
23     @Temporal(TemporalType.DATE)
24
25     private Date createdAt;
26
27     public long getId() {
28
29
30
31
32
33
34 }
```

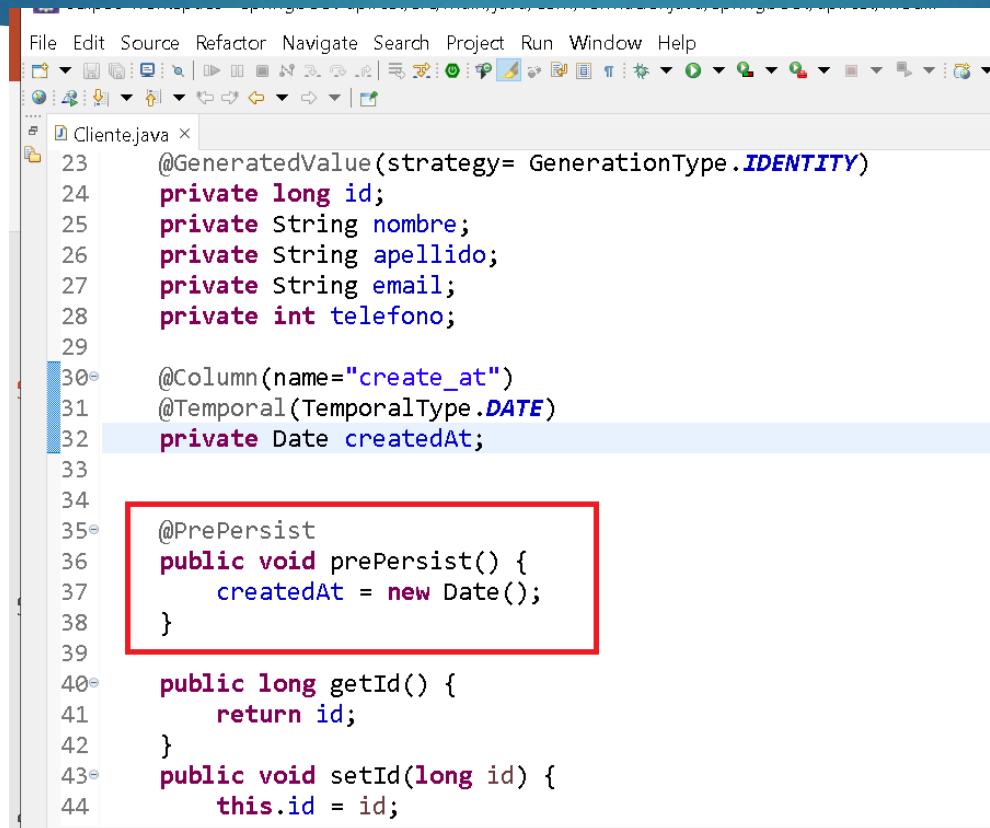
Que es una entidad

- ▶ Una entidad en Java es un objeto de persistencia.
- ▶ La persistencia es la habilidad de una aplicación para mantener(persistir) y recuperar información de sistemas de almacenamiento no volátiles.
- ▶ Una entidad representa una tabla en una base de datos, y cada instancia de entidad corresponde a una fila en la tabla.
- ▶ El estado de una entidad se representa por campos de persistencia o propiedades de persistencia.

Anotaciones

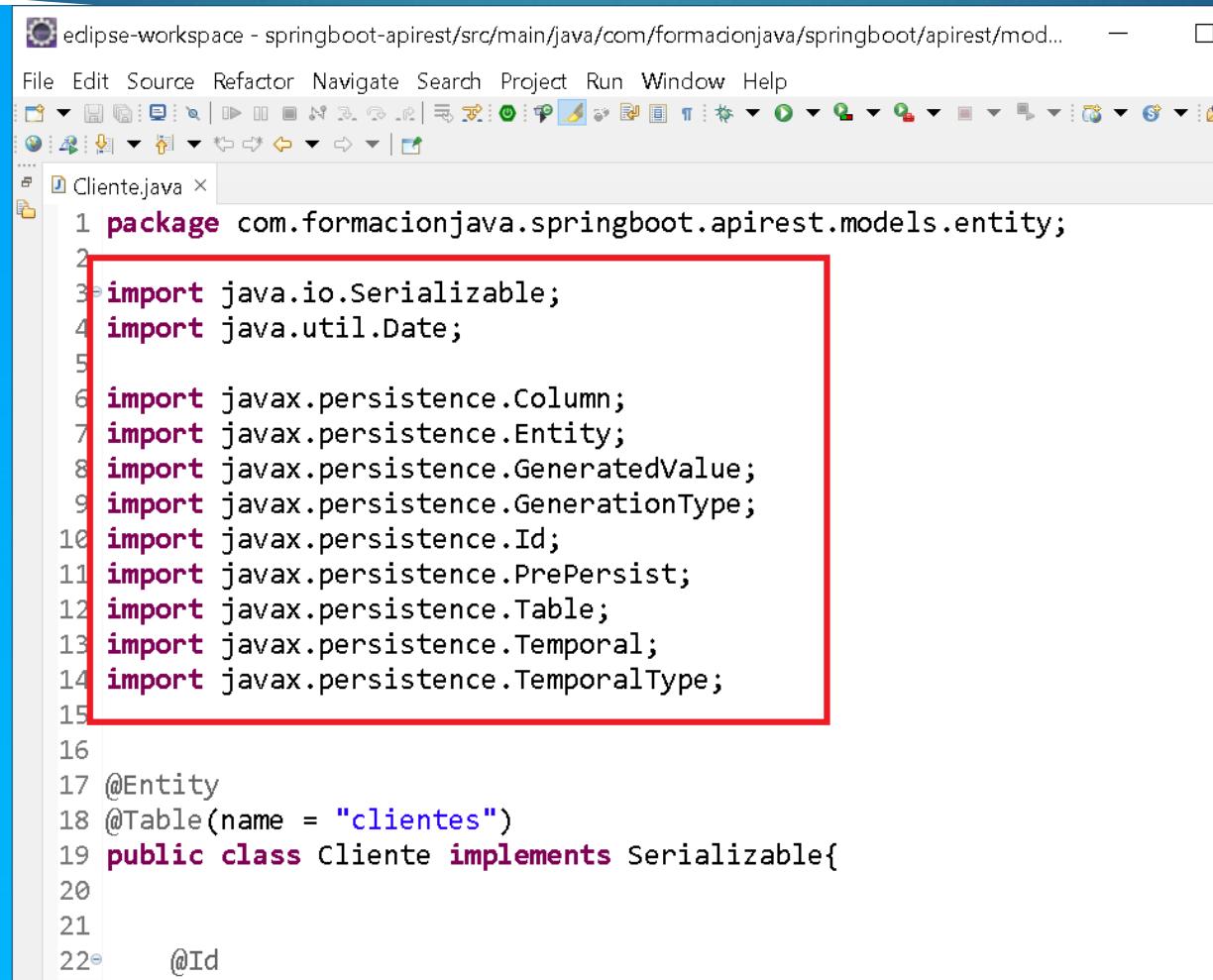
- ▶ **@Entity** Especifica que una clase es una entidad.
- ▶ **@Table** anotación especifica del nombre de la tabla de base datos que se utilizara para el mapeo.
- ▶ **@Column** se utiliza para identificar la correspondencia entre atributos en las clases de entidad y campos en las tablas de datos.
- ▶ **@Id** Especifica la llave primaria de una entidad.
- ▶ **@GeneratedValue** proporciona la especificación de estrategias de generación para los valores de las claves primarias.

Agregamos el siguiente método para que nos tome la fecha actual



```
File Edit Source Refactor Navigate Search Project Run Window Help
Client.java x
23     @GeneratedValue(strategy= GenerationType.IDENTITY)
24     private long id;
25     private String nombre;
26     private String apellido;
27     private String email;
28     private int telefono;
29
30     @Column(name="create_at")
31     @Temporal(TemporalType.DATE)
32     private Date createdAt;
33
34
35     @PrePersist
36     public void prePersist() {
37         createdAt = new Date();
38     }
39
40     public long getId() {
41         return id;
42     }
43     public void setId(long id) {
44         this.id = id;
```

A tener en cuenta estos imports en la clase



```
edipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/mod...
File Edit Source Refactor Navigate Search Project Run Window Help
Cliente.java x
1 package com.formacionjava.springboot.apirest.models.entity;
2
3 import java.io.Serializable;
4 import java.util.Date;
5
6 import javax.persistence.Column;
7 import javax.persistence.Entity;
8 import javax.persistence.GeneratedValue;
9 import javax.persistence.GenerationType;
10 import javax.persistence.Id;
11 import javax.persistence.PrePersist;
12 import javax.persistence.Table;
13 import javax.persistence.Temporal;
14 import javax.persistence.TemporalType;
15
16
17 @Entity
18 @Table(name = "clientes")
19 public class Cliente implements Serializable{
20
21
22     @Id
```

Compilamos y verificamos que nos cree la tabla solicitada en entity

The screenshot shows the Eclipse IDE interface with the following details:

- Java Code (Top Left):**

```
11 import javax.persistence.Table;
```
- Eclipse Tabs (Top):** Markers, Properties, Servers, Data Source Explorer, Snippets, Console.
- Console Output (Bottom):** The output is from a Spring Boot application named "SpringbootApirestApplication". It includes log entries from 3:18:18.347 to 3:18:22.364. Key logs related to the question include:
 - INFO 8204 --- [restartedMain] org.hibernate.SQL : drop table if exists clientes
 - INFO 8204 --- [restartedMain] org.hibernate.SQL : create table clientes (id bigint not null auto_increm
 - INFO 8204 --- [restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.transaction.CMTTransactionManagerImpl]
 - INFO 8204 --- [restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
 - WARN 8204 --- [restartedMain] JpaBaseConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefor
 - INFO 8204 --- [restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
 - INFO 8204 --- [restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8087 (http) with context p
 - INFO 8204 --- [restartedMain] c.f.s.a.SpringbootApirestApplication : Started SpringbootApirestApplication in 4.488 seconds



Navigator

SCHEMAS

Filter objects

- db_spring
 - Tables
 - clientes
 - Columns
 - id
 - apellido
 - create_at
 - email
 - nombre
 - telefono
 - Indexes
 - Foreign Keys
 - Triggers
 - Views
 - Stored Procedures
 - Functions
- prueba_db

Administration

Schemas

Information

Schema: db_spring

Query 1 Administration - Startup / Shutdown

Local instance MySQL80
Startup / Shutdown MySQL Server

The database server is started and ready for client connections. To shut the server down, use the "Stop Server" button.

The database server instance is running Stop Server Bring Offline

If you stop the server, neither you nor your applications can use the database and all current connections will be closed.

Startup Message Log

```
2021-11-18 22:30:24 - Workbench will use cmd shell commands to start/stop this instance
2021-11-18 22:30:24 - Server is running
2021-11-18 23:04:29 - Checking server status...
2021-11-18 23:04:29 - MySQL server is currently running
2021-11-18 23:04:30 - Checking server status...
2021-11-18 23:04:30 - MySQL server is currently running
2021-11-18 23:18:54 - Checking server status...
2021-11-18 23:18:54 - MySQL server is currently running
```

Refresh Status

Output

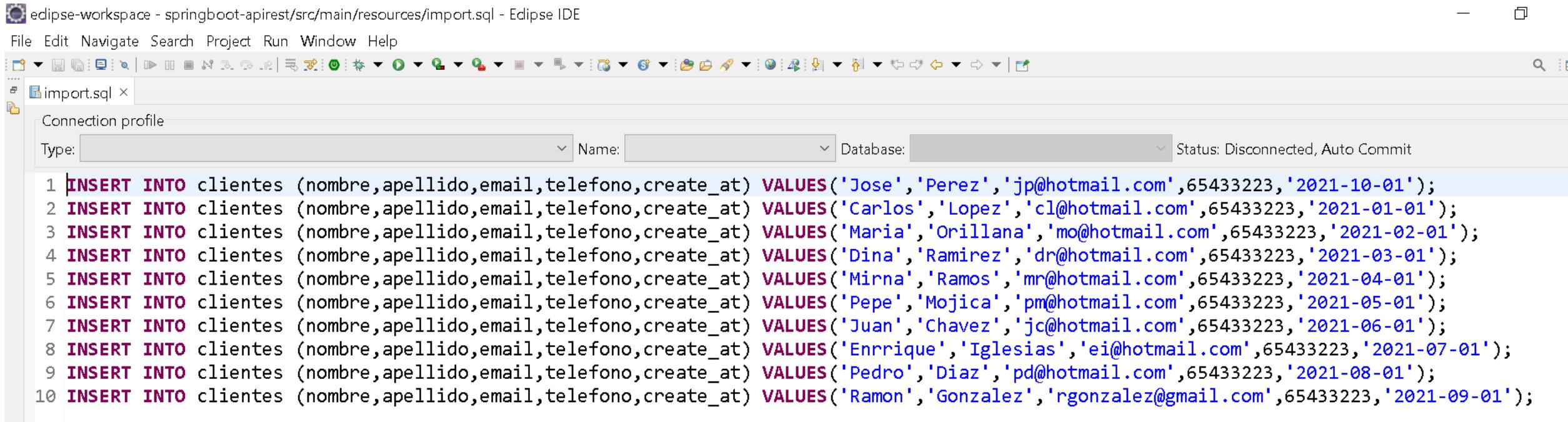
Action Output

Time Action

Message

Añadir datos de prueba

- ▶ Dentro de la carpeta src/main/resources creamos un archivo llamado import.sql y de esta forma una vez compilamos Spring de forma automática leerá este archivo y realizara el insert de los datos de prueba



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - springboot-apirest/src/main/resources/import.sql - Eclipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has various icons for file operations like Open, Save, Find, and Run. The left sidebar shows a project tree with "import.sql x" selected. The main editor area displays the following SQL code:

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

Compilamos y verificamos que se inserta los datos de prueba

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the database schema. A red box highlights the 'db_spring' schema, which contains a 'Tables' folder and the 'clientes' table.
- Query Editor:** Displays the SQL query: `SELECT * FROM db_spring.clientes;`
- Result Grid:** Shows the data from the 'clientes' table. A red box highlights the entire grid. The data is as follows:

	id	apellido	create_at	email	nombre	telefono
1	1	Perez	2021-10-01	jp@hotmail.com	Jose	65433223
2	2	Lopez	2021-01-01	cl@hotmail.com	Carlos	65433223
3	3	Orillana	2021-02-01	mo@hotmail.com	Maria	65433223
4	4	Ramirez	2021-03-01	dr@hotmail.com	Dina	65433223
5	5	Ramos	2021-04-01	mr@hotmail.com	Mirna	65433223
6	6	Mojica	2021-05-01	pm@hotmail.com	Pepe	65433223
7	7	Chavez	2021-06-01	jc@hotmail.com	Juan	65433223
8	8	Iglesias	2021-07-01	ei@hotmail.com	Enrique	65433223
9	9	Diaz	2021-08-01	pd@hotmail.com	Pedro	65433223
10	10	Gonzalez	2021-09-01	rgonzalez@gmail.com	Ramon	65433223

Creamos el paquete y la interfaz para utilizar CrudRepository

The screenshot shows a Java code editor in an IDE with the following details:

- Project Explorer:** Shows the project structure with several Maven projects and their subfolders.
- Current File:** *ClienteDao.java
- Code Content:**

```
1 package com.formacionjava.springboot.apirest.models.dao;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import com.formacionjava.springboot.apirest.models.entity.Cliente;
6
7 public interface ClienteDao extends CrudRepository<Cliente, Long>{
8
9 }
```
- Tooltips/Information:** A tooltip for the `CrudRepository` interface is displayed, providing its definition, annotations (`@NoRepositoryBean`), and author information (Oliver Gierke, Eberhard Wolff, Jens Schauder).
- Bottom Status Bar:** A message "Press 'F2' for focus" is visible.

CrudRepository

- ▶ CrudRepository implementa operaciones CRUD básicas, que incluyen contar, eliminar, deleteById, save, saveAll, findById y findAll. Lo podemos implementar gracias a SpringDataJPA

Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer x

maven_proyect

primero [boot] [devtools]

proyecto_web

Refresh

repaso

segundo [boot] [devtools]

springboot-apirest [boot] [devtools]

src/main/java

> com.formacionjava.springboot.apirest

> com.formacionjava.springboot.apirest.models.dao

> ClienteDao.java

> com.formacionjava.springboot.apirest.models.entity

src/main/resources

src/test/java

JRE System Library [JavaSE-11]

Maven Dependencies

src

target

HELP.md

mvnw

mvnw.cmd

pom.xml

* ClienteDao.java CrudRepository.class x

```
2* * Copyright 2008-2021 the original author or authors.
16 package org.springframework.data.repository;
17
18 import java.util.Optional;
19
20 /**
21  * Interface for generic CRUD operations on a repository for a specific type.
22 *
23 * @author Oliver Gierke
24 * @author Eberhard Wolff
25 * @author Jens Schauder
26 */
27 @NoRepositoryBean
28 public interface CrudRepository<T, ID> extends Repository<T, ID> {
29
30 /**
31 * Saves a given entity. Use the returned instance for further operations as the same
32 * entity instance completely.
33 *
34 * @param entity must not be {@literal null}.
35 * @return the saved entity; will never be {@literal null}.
36 * @throws IllegalArgumentException in case the given {@literal entity} is {@literal null}.
37 */
```

Markers Properties Servers Data Source Explorer Snippets Console x

```
springboot-apirest - SpringbootApirestApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
3:18:18.347 INFO 8204 --- [ restartedMain] c.t.s.a.SpringbootApirestApplication      : Star
3:18:18.348 INFO 8204 --- [ restartedMain] c.f.s.a.SpringbootApirestApplication      : No a
3:18:18.412 INFO 8204 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devt
```

Creamos los servicios en el siguiente paquete y la interfaz

The screenshot shows the Eclipse IDE interface with the following details:

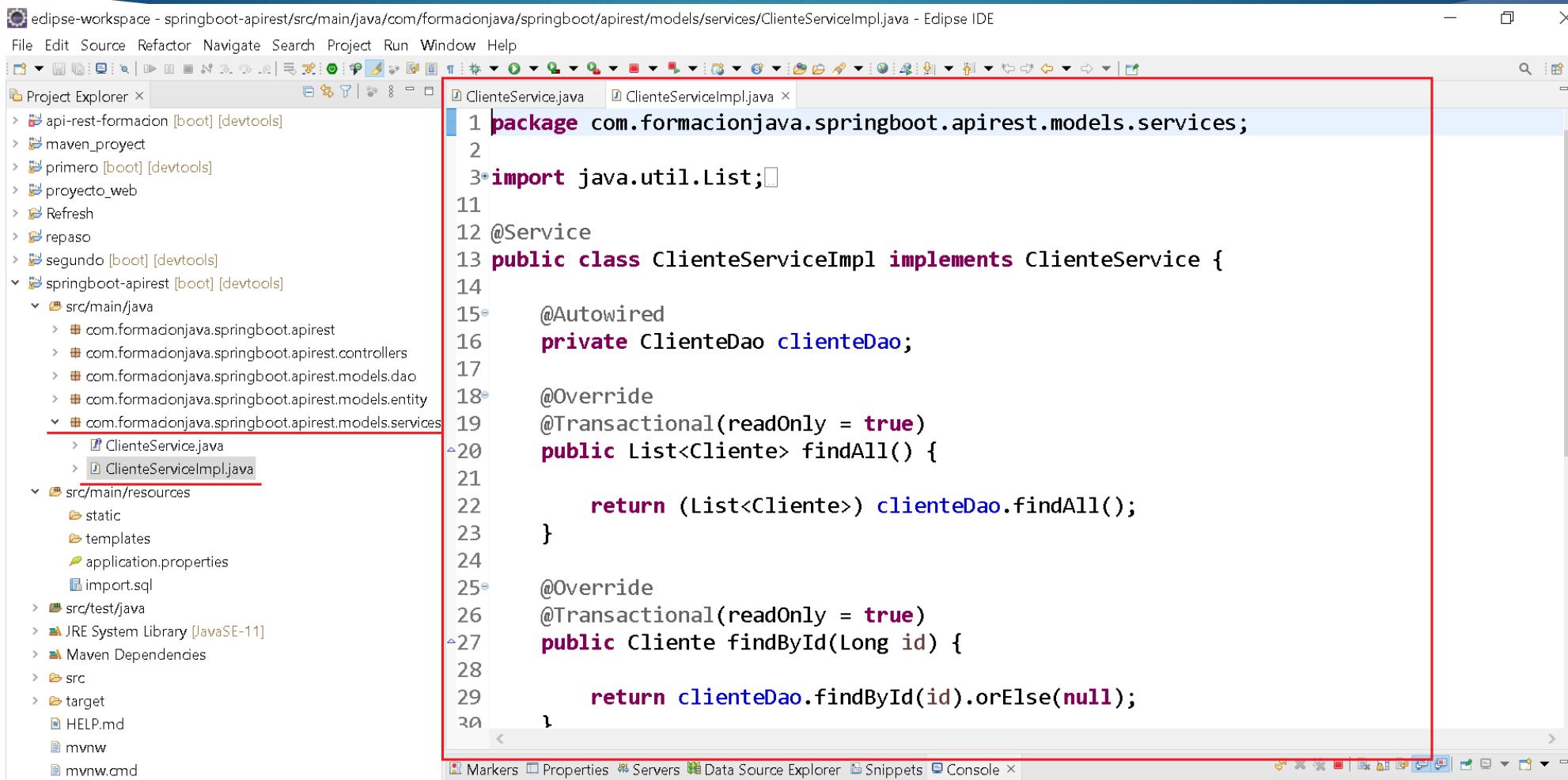
- Title Bar:** eclipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/models/services/ClienteService.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar icons.
- Project Explorer:** Shows several projects: api-rest-formacion [boot] [devtools], maven_proyect, primero [boot] [devtools], proyecto_web, Refresh, repaso, segundo [boot] [devtools], and springboot-apirest [boot] [devtools]. The springboot-apirest project is expanded, showing its structure: src/main/java, src/main/resources, and src/test/java. Under src/main/java, there is a package com.formacionjava.springboot.apirest.models.services which contains ClienteService.java and ClienteServiceImpl.java. Under src/main/resources, there are static, templates, application.properties, and import.sql files. Under src/test/java, there is JRE System Library [JavaSE-11] and Maven Dependencies.
- Code Editor:** The code editor window displays the contents of ClienteService.java, which is highlighted with a red border. The code defines a public interface ClienteService with methods for finding all clients, finding a client by ID, saving a client, and deleting a client.
- Bottom Bar:** Standard Eclipse bottom toolbar icons.

```
package com.formacionjava.springboot.apirest.models.services;
import java.util.List;
import com.formacionjava.springboot.apirest.models.entity.Cliente;
public interface ClienteService {
    public List<Cliente> findAll();
    public Cliente findById(Long id);
    public Cliente save(Cliente cliente);
    public void delete(Long id);
}
```

ClienteService

- ▶ Implementamos aquí todos los métodos que utilizaremos en nuestra api rest

Creamos la clase ClienteServiceImpl que implementara la interfaz ClienteService y definiendo los métodos



```
1 package com.formacionjava.springboot.apirest.models.services;
2
3 import java.util.List;
4
5 @Service
6 public class ClienteServiceImpl implements ClienteService {
7
8     @Autowired
9     private ClienteDao clienteDao;
10
11     @Override
12     @Transactional(readOnly = true)
13     public List<Cliente> findAll() {
14
15         return (List<Cliente>) clienteDao.findAll();
16     }
17
18     @Override
19     @Transactional(readOnly = true)
20     public Cliente findById(Long id) {
21
22         return clienteDao.findById(id).orElse(null);
23     }
24 }
```

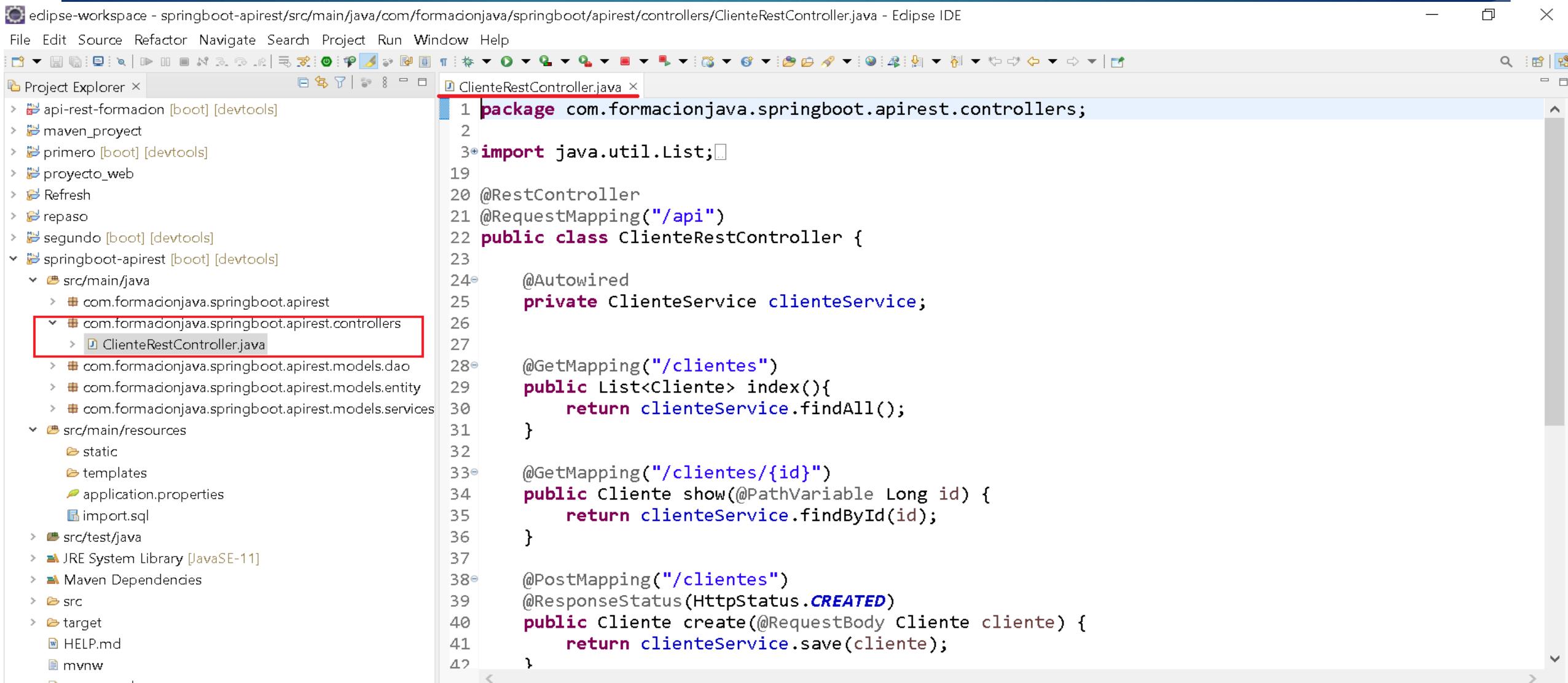
The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure. The `springboot-apirest` project is expanded, revealing its `src/main/java` directory which contains several packages and files, including `ClienteService.java` and `ClienteServiceImpl.java`.
- Code Editor:** The `ClienteServiceImpl.java` file is open. The code implements a service layer for managing Clientes. It includes three methods:
 - `public Cliente findById(Long id)`: Returns a Cliente by its ID, using the `clienteDao.findById(id).orElse(null)` pattern.
 - `public Cliente save(Cliente cliente)`: Returns the saved Cliente after calling `clienteDao.save(cliente)`.
 - `public void delete(Long id)`: Deletes a Cliente by its ID using the `clienteDao.deleteById(id)` method.
- Annotations:** The code uses `@Override`, `@Transactional`, and `@Override` annotations.
- Toolbars and Status Bar:** Standard Eclipse toolbars are visible at the top. The status bar at the bottom displays the project name (`springboot-apirest`), application type (`Spring Boot App`), and system information (`C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_16.0.2.v202107`).

Las anotaciones utilizadas

- ▶ @Autowired es una de las anotaciones más habituales cuando trabajamos con Spring Framework ya que se trata de la anotación que permite injectar unas dependencias con otras dentro de Spring
- ▶ @Transactional readonly es una de las características que en muchas ocasiones nos olvidamos al manejar Spring Framework. Spring nos provee de las capacidades de inyección de gestión transaccional distribuida y transparente para nuestros repositorios y servicios
- ▶ La pregunta que tenemos que hacernos . ¿En algún momento vamos a solicitar una modificación de la lista?. Es bastante evidente que en el 95% de las ocasiones no y por lo tanto podemos marcar con Spring @Transactional readonly el método para que no guarde el estado y tengamos un mejor rendimiento de la consulta

Por ultimo creamos el siguiente paquete para el controlador ClienteRestController



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** eclipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Toolbar:** Standard Eclipse toolbar with various icons for file operations.
- Project Explorer:** Shows the project structure:
 - api-rest-formacion [boot] [devtools]
 - maven_proyect
 - primero [boot] [devtools]
 - projeto_web
 - Refresh
 - repaso
 - segundo [boot] [devtools]
 - springboot-apirest [boot] [devtools]
 - src/main/java
 - com.formacionjava.springboot.apirest
 - com.formacionjava.springboot.apirest.controllers
 - ClienteRestController.java
 - com.formacionjava.springboot.apirest.models.dao
 - com.formacionjava.springboot.apirest.models.entity
 - com.formacionjava.springboot.apirest.models.services
 - src/main/resources
 - static
 - templates
 - application.properties
 - import.sql
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
- Editor:** The code for `ClienteRestController.java` is displayed:

```
1 package com.formacionjava.springboot.controllers;
2
3 import java.util.List;
4
5 @RestController
6 @RequestMapping("/api")
7 public class ClienteRestController {
8
9     @Autowired
10    private ClienteService clienteService;
11
12
13    @GetMapping("/clientes")
14    public List<Cliente> index(){
15        return clienteService.findAll();
16    }
17
18    @GetMapping("/clientes/{id}")
19    public Cliente show(@PathVariable Long id) {
20        return clienteService.findById(id);
21    }
22
23    @PostMapping("/clientes")
24    @ResponseStatus(HttpStatus.CREATED)
25    public Cliente create(@RequestBody Cliente cliente) {
26        return clienteService.save(cliente);
27    }
28}
```

eclipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Project Explorer x ClienteRestController.java x

```
35     return clienteService.findById(id);  
36 }  
37  
38 @PostMapping("/clientes")  
39 @ResponseStatus(HttpStatus.CREATED)  
40 public Cliente create(@RequestBody Cliente cliente) {  
41     return clienteService.save(cliente);  
42 }  
43  
44 @PutMapping("/clientes/{id}")  
45 @ResponseStatus(HttpStatus.CREATED)  
46 public Cliente update(@RequestBody Cliente cliente, @PathVariable Long id) {  
47     Cliente clienteUpdate = clienteService.findById(id);  
48  
49     clienteUpdate.setApellido(cliente.getApellido());  
50     clienteUpdate.setNombre(cliente.getNombre());  
51     clienteUpdate.setEmail(cliente.getEmail());  
52  
53     return clienteService.save(clienteUpdate);  
54 }  
55  
56 @DeleteMapping("clientes/{id}")  
57 public void delete(@PathVariable Long id) {  
58     clienteService.delete(id);  
59 }  
60 }  
61
```

Markers Properties Servers Data Source Explorer Snippets Console x

Anotaciones utilizadas

- ▶ **@Controller** . Esto es simplemente una especialización de la clase `@Component` , que nos permite detectar automáticamente las clases de implementación a través del escaneo de classpath. Por lo general, usamos `@Controller` en combinación con una anotación `@RequestMapping` para los métodos de manejo de solicitudes.
- ▶ **@RequestMapping** En pocas palabras, la anotación se utiliza para asignar solicitudes web a los métodos de Spring Controller.
- ▶ **@GetMapping** es una versión especializada de `@RequestMapping` anotación actúa como un atajo para `@RequestMapping(method = RequestMethod.GET)`
- ▶ **@PostMapping** es una versión de `@RequestMapping` anotación que actúa como un acceso directo a `@RequestMapping(method = RequestMethod.POST)`
- ▶ **@PathVariable** se puede usar para manejar variables de plantilla en la asignación de URI de solicitud
- ▶ **@RequestBody** asigna el cuerpo `HttpRequest` a un objeto de transferencia o dominio, lo que permite la deserialización automática del cuerpo `HttpRequest` entrante en un objeto Java.
- ▶ `@ResponseStatus` marca un método o clase de excepción con el código de estado y el mensaje de motivo de devolverse. El código de estado se aplica a la respuesta HTTP cuando se invoca el método controlador o siempre que se lanza la excepción específica.

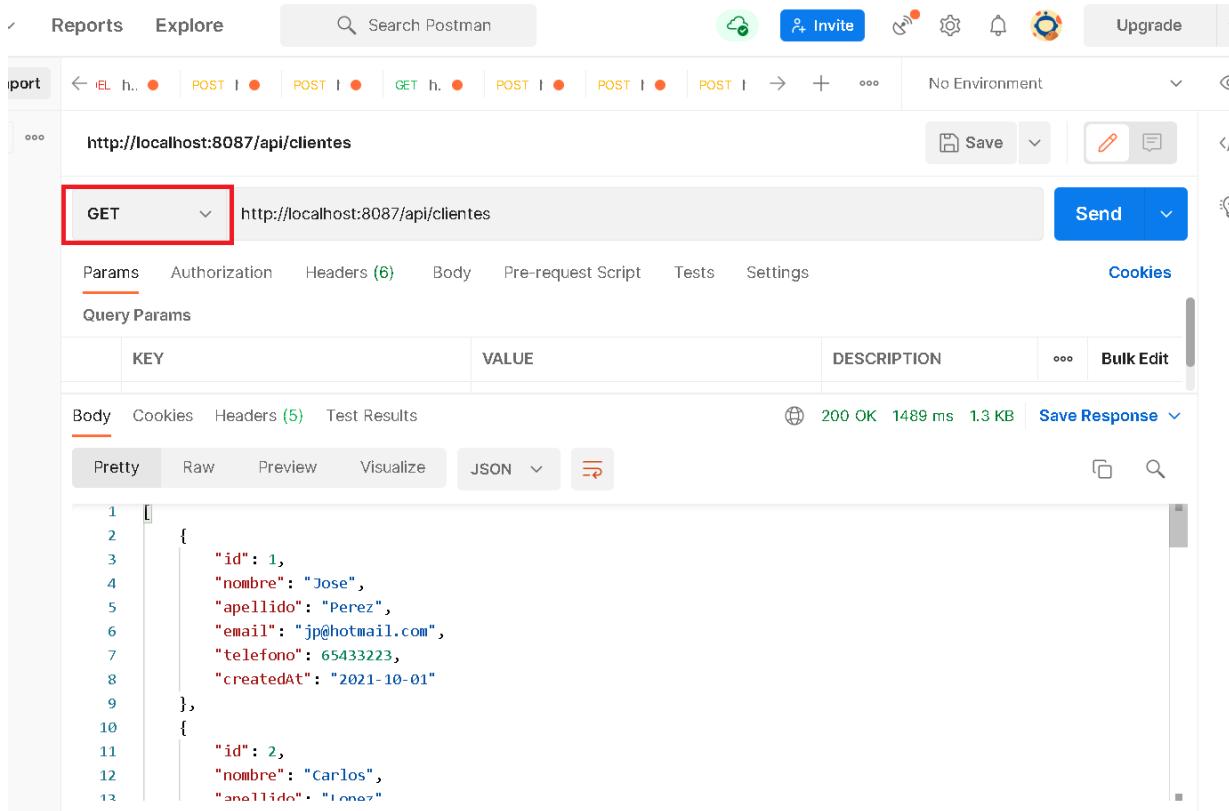
Las url de las apis que hemos creado

- ▶ GET <http://localhost:8087/api/clientes>
- ▶ POST <http://localhost:8087/api/clientes>
- ▶ GET <http://localhost:8087/api/clientes/{id}>
- ▶ DELETE <http://localhost:8087/api/clientes/{id}>

- ▶ OBSERVACION: EL PUERTO SERIA EL QUE ESTE FUNCIONANDO PARA USTEDES

Lo probamos con Postman

► PROBAR API GET



The screenshot shows the Postman application interface. At the top, there are tabs for Reports and Explore, a search bar, and various icons for account management and upgrades. Below the header, a navigation bar shows a sequence of requests: GET, POST, POST, GET, POST, POST, POST, and a plus sign. It also indicates "No Environment".

In the main workspace, a GET request is selected from a dropdown menu. The URL is set to `http://localhost:8087/api/clientes`. To the right of the URL is a "Send" button. Below the URL, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, Settings, Cookies, and Query Params. The Headers tab is currently active.

The Body tab is selected, showing the response content. The status bar at the bottom indicates a 200 OK response with 1489 ms and 1.3 KB. The response body is displayed in a JSON editor:

```
1 [ {  
2   "id": 1,  
3   "nombre": "Jose",  
4   "apellido": "Perez",  
5   "email": "jp@hotmail.com",  
6   "telefono": 65433223,  
7   "createdAt": "2021-10-01"  
8 },  
9 {  
10   "id": 2,  
11   "nombre": "Carlos",  
12   "apellido": "Lopez"  
13 }
```

Búsqueda por id

The screenshot shows the Postman application interface. At the top, there are tabs for 'Reports' and 'Explore'. A search bar contains the text 'Search Postman'. On the right side of the header are icons for cloud sync, invite, notifications, settings, and upgrade. Below the header, a list of recent requests is shown, each with a status indicator (red dot) and a preview icon.

The main workspace displays a request configuration for a GET method. The URL is set to `http://localhost:8087/api/clientes/3`. The method dropdown is highlighted with a red box. To the right of the URL is a 'Send' button with a dropdown arrow.

Below the URL, under the 'Headers' tab, there are six entries. Under the 'Body' tab, there is a table for 'Query Params' with columns: KEY, VALUE, and DESCRIPTION. The table is currently empty.

At the bottom of the request configuration, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Body' tab is selected. To the right of these tabs, response details are shown: 200 OK, 47 ms, 281 B, and a 'Save Response' button with a dropdown arrow. Below this, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON' (with a dropdown arrow).

The JSON response body is displayed in a code editor-like area:

```
1 {  
2   "id": 3,  
3   "nombre": "Maria",  
4   "apellido": "Orillana",  
5   "email": "mo@hotmail.com",  
6   "telefono": 65433223,  
7   "createdAt": "2021-02-01"  
8 }
```

At the very bottom of the screen, there are navigation icons for 'Postman', 'Profile', 'Logout', and a help icon.

Probamos el método POST

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for account management, invites, notifications, and upgrading.

The main workspace shows a sequence of requests: GET, DEL, GET, and POST. The current request is a POST to `http://localhost:8087/api/clientes/`. The method dropdown is highlighted with a red box. Below it, the URL is displayed again.

The 'Body' tab is selected, showing options for 'Params', 'Authorization', 'Headers (8)', 'Body', 'Pre-request Script', 'Tests', 'Settings', 'Cookies', and 'Beautify'. The 'Body' tab has a green dot next to it. The 'Body' section is expanded, showing a radio button for 'raw' which is selected and highlighted with a red box. Other options include 'none', 'form-data', 'x-www-form-urlencoded', 'binary', 'GraphQL', and 'JSON'. The 'JSON' option is also highlighted with a red box.

The raw JSON payload is displayed in a code editor-like area:

```
1 {  
2   "nombre": "Joaquin",  
3   "apellido": "Rosales",  
4   "email": "jrosales@hotmail.com",  
5   "telefono": 9132211  
6 }
```

Below the body, there are tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. The 'Body' tab is selected. The 'Test Results' tab shows a successful response: `201 Created`, `200 ms`, `312 B`, and a 'Save Response' button.

The response body is shown in 'Pretty' format:

```
1 {  
2   "id": 11,  
3   "nombre": "Joaquin",  
4   "apellido": "Rosales",  
5   "email": "jrosales@hotmail.com",  
6   "telefono": 9132211,  
7   "createdAt": "2021-11-21T15:55:52.129+00:00"
```

At the bottom, there are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. There are also icons for 'Bootcamp', 'Runner', 'Trash', and a help icon.

Método PUT

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for cloud sync, invite, settings, and notifications. Below the navigation is a toolbar with icons for back, forward, and search, followed by a dropdown menu for environments ('No Environment') and a save button.

The main workspace displays a request card for a PUT operation:

- Method:** PUT (highlighted with a red box)
- URL:** http://localhost:8087/api/clientes/11
- Body:** (highlighted with a red box) - Contains 8 items (headers).
- Content Type:** raw (highlighted with a red box)
- JSON:** (highlighted with a red box) - A dropdown menu for JSON beautification.

The body content is a JSON object:

```
1 {  
2   "nombre": "Rolando",  
3   "apellido": "Lopez",  
4   "email": "rlopez@hotmail.com",  
5   "telefono": 6432332  
6 }
```

Below the request card, there are tabs for Body, Cookies, Headers (5), and Test Results. The Body tab is active, showing the response in Pretty, Raw, Preview, and Visualize formats. The response body is identical to the request body:

```
1 {  
2   "id": 11,  
3   "nombre": "Rolando",  
4   "apellido": "Lopez",  
5   "email": "rlopez@hotmail.com",  
6   "telefono": 9132211,  
7   "createdAt": "2021-11-21"
```

At the bottom, there are buttons for Bootcamp, Runner, Trash, and Help.

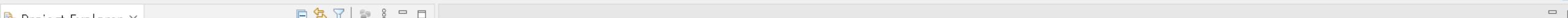
Método DELETE

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports' and 'Explore' tabs, a search bar 'Search Postman', and various icons for cloud, invite, settings, and notifications. Below the header, a toolbar shows recent requests: GET, DEL, GET, PUT, and another DEL. The main area displays a request configuration for a DELETE method:

- Method:** DELETE
- URL:** http://localhost:8087/api/clientes/1
- Params:** Headers tab is selected, showing 6 items.
- Query Params:** A table with columns KEY, VALUE, DESCRIPTION, and Bulk Edit. It contains one row with Key, Value, and Description.
- Body:** Body tab is selected, showing options Pretty, Raw, Preview, Visualize, and Text.
- Headers:** Headers tab shows 4 items.
- Test Results:** Test Results tab is present.
- Status:** Status bar at the bottom shows 200 OK, 56 ms, 123 B, and Save Response.

GIT

- ▶ Subir proyecto a repositorio remoto desde eclipse



Project Explorer ×

- > api-rest-formacion [boot] [devtools]
- > maven_proyect
- > primero [boot] [devtools]
- > proyecto_web
- > Refresh
- > repaso
- > segundo [boot] [devtools]
- > springboot-apirest [boot] [devtools]

- New >
- Go Into
- Show In Alt+Shift+W >
- Show in Local Terminal >
- Copy Ctrl+C
- Copy Qualified Name
- Paste Ctrl+V
- Delete Delete
- Remove from Context Ctrl+Alt+Shift+Down
- Build Path >
- Refactor Alt+Shift+T >
- Import... <
- Export... <
- Refresh F5
- Close Project
- Close Unrelated Projects
- Coverage As >
- Run As >
- Debug As >
- Profile As >
- Restore from Local History... >
- Maven >
- Team >
 - Apply Patch...
 - Share Project... Share Project...
- Compare With >
- Configure >
- Source >
- Spring >
- Validate >
- Properties Alt+Enter



- Project Explorer ×
- > api-rest-formacion [boot] [devtools]
 - > maven_proyect
 - > primero [boot] [devtools]
 - > proyecto_web
 - > Refresh
 - > repaso
 - > segundo [boot] [devtools]
 - > springboot-apirest [boot] [devtools]

Share Project

Configure Git Repository

Select an existing repository or create a new one

Use or create repository in parent folder of project

Repository:

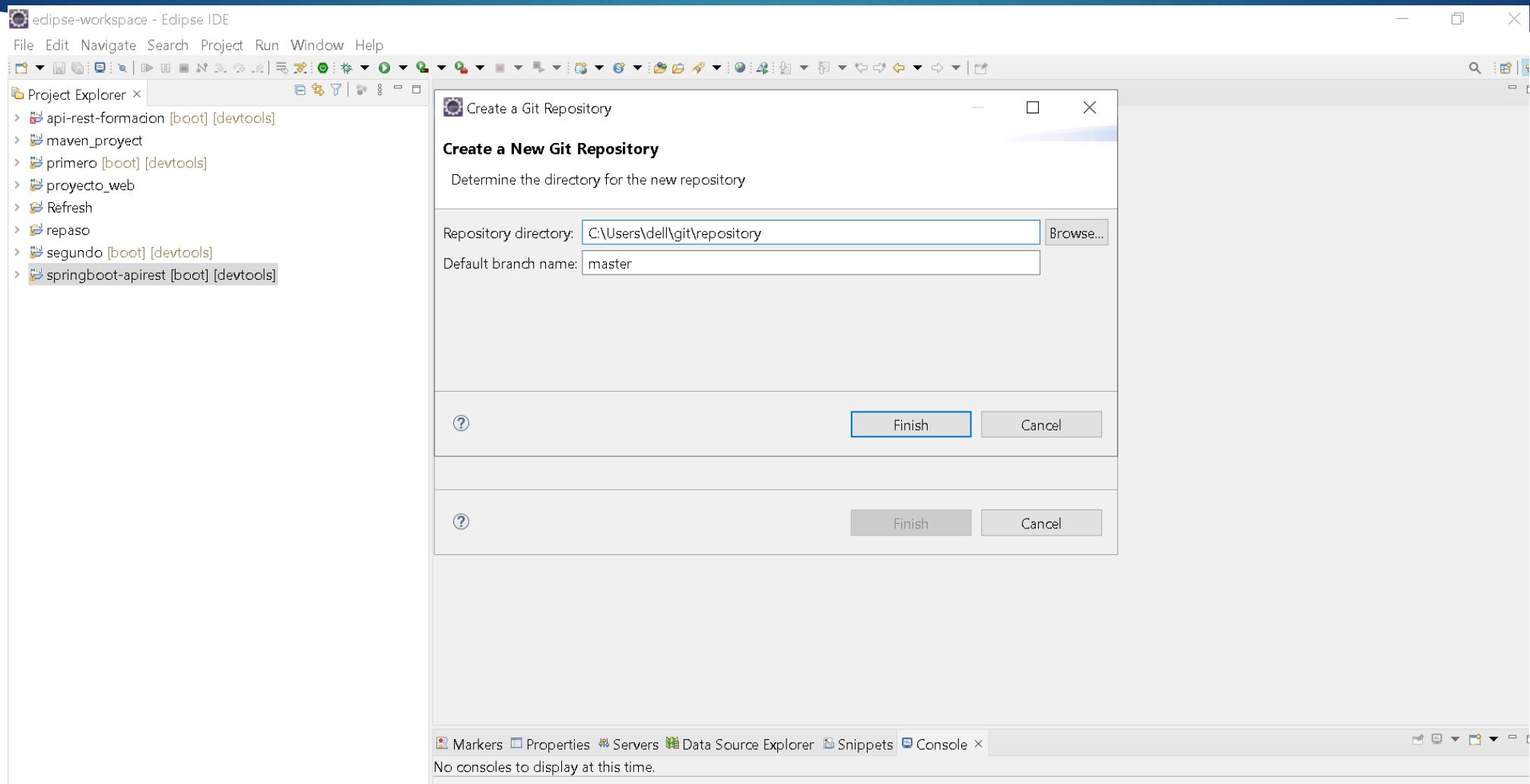
Working tree: No repository selected

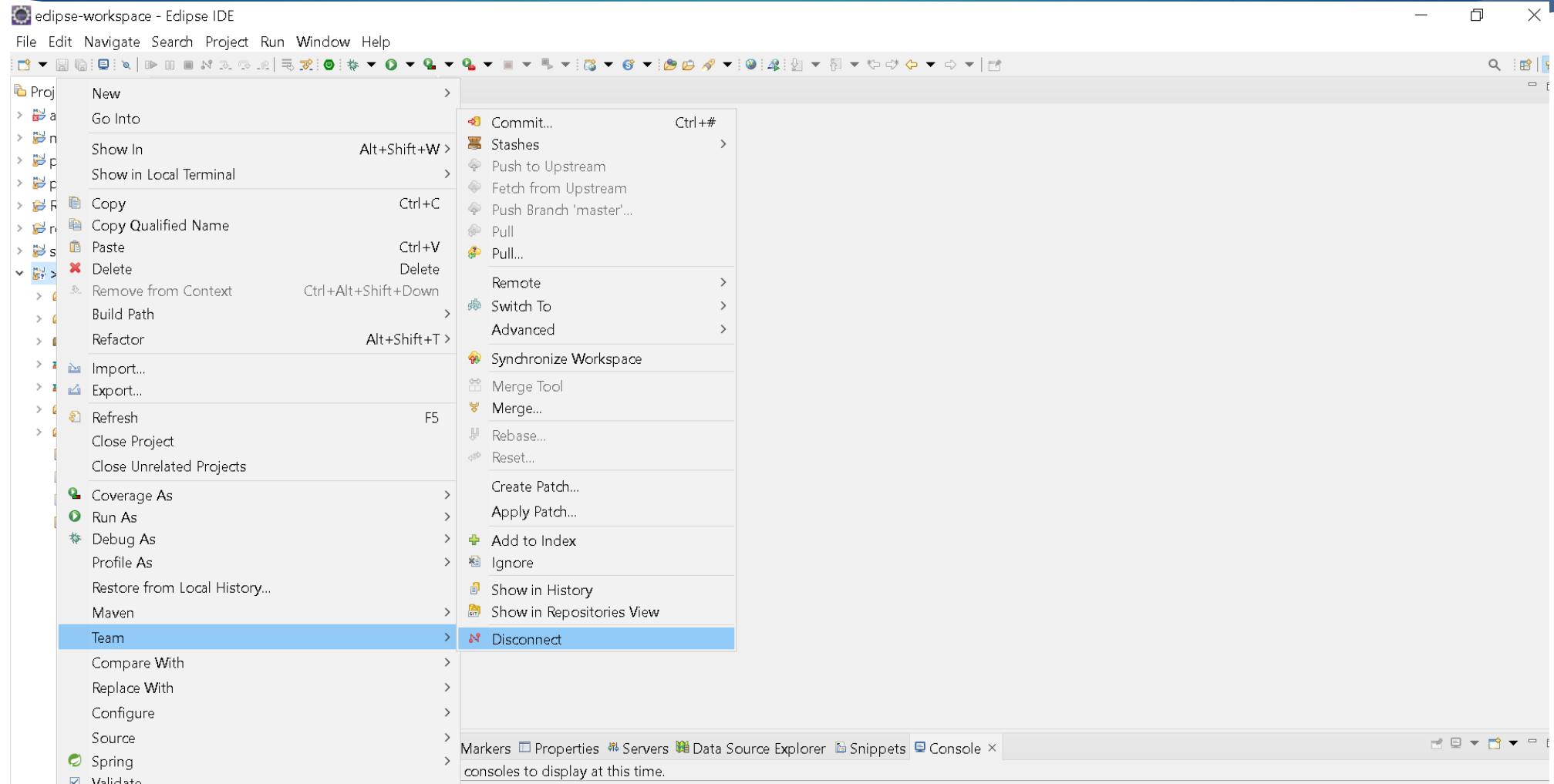
Path within repository:

Project	Current Location	Target Location
<input checked="" type="checkbox"/> spring...	C:/eclipse-workspace/spring...	

?

Finish Cancel





Project Explorer ×

- > api-rest-formacion [boot] [devtools]
- > maven_proyect
- > > primero [boot] [devtools] [repository2 master]
- > proyecto_web
- > Refresh
- > repaso
- > segundo [boot] [devtools]
- > > springboot-apirest [boot] [devtools] [repository master]
 - > src/main/java
 - > src/main/resources
 - > src/test/java
 - > JRE System Library [JavaSE-11]
 - > Maven Dependencies
 - > src
 - > target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

Markers Properties Servers Data Source Explorer Snippets Console Git Staging ×

Filter files

> repository2 [master]

Unstaged Changes (12)

- .gitignore - primero
- application.properties - primero/src/main/resources
- ControladorSaludo.java - primero/src/main/java/com/example/controller
- maven-wrapper.jar - primero/.mvn/wrapper
- maven-wrapper.properties - primero/.mvn/wrapper
- MavenWrapperDownloader.java - primero/.mvn/wrapper
- mvnw - primero
- mvnw.cmd - primero
- pom.xml - primero
- PrimeroApplication.java - primero/src/main/java/com/example/demo
- PrimeroApplicationTests.java - primero/src/test/java/com/example/demo

Staged Changes (0)

Commit Message

i Unborn branch: this commit will create the branch 'master'.

Author: rolando <isa.klopez@hotmail.com>

Committer: rolando <isa.klopez@hotmail.com>

Commit and Push... Commit

edipse-workspace - Edipse IDE

File Edit Navigate Search Project Run Window Help

Project Explorer

- api-rest-formacion [boot] [devtools]
- maven_proyect
- primero [boot] [devtools]
- projeto_web
- Refresh
- repasso
- segundo [boot] [devtools]
- springboot-apirest [boot] [devtools] [repository master]
 - src/main/java
 - src/main/resources
 - src/test/java
 - JRE System Library [JavaSE-11]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml

Push Branch master

Destination Git Repository

Enter the location of the destination repository.

Remote name: origin

Location

URL: https://github.com/isabelino/formacion-capgemini.git Local Folder...

Host: github.com

Repository path: /isabelino/formacion-capgemini.git

Connection

Protocol: https

Port:

Authentication

User: isabelino.rolando@gmail.com

Password: [REDACTED]

Store in Secure Store

Preview > Push Cancel

Author: rolando <isa.klopez@hotmail.com>

Committer: rolando <isa.klopez@hotmail.com>

Push HEAD... Commit

Filter files

Manejo de Errores

The screenshot shows the Postman application interface. On the left, the sidebar includes sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The main workspace is titled "My Workspace" and contains a collection named "PruebasApiRest". A failed API request is highlighted in red, showing a GET request to `http://localhost:8087/api/clientes/22`. The status bar at the bottom indicates a 200 OK response, which is likely a mislabel or a placeholder for a failed request. The "Headers" tab shows several errors (red dots) under the "Value" column. The "Body" tab displays a single character "1".

Postman

File Edit View Help

Home Workspaces API Network Reports Explore Search Postman

My Workspace New Import

GET h. ● | DEL h. ● | GET U | GET h. ● | PUT h. ● | DEL h. ● | → + ... | No Environment

Save | / | ⚙️ | 🔍 | ⚡ | Upgrade

+

http://localhost:8087/api/clientes/22

GET http://localhost:8087/api/clientes/22

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

1

200 OK 281 ms 123 B Save Response

Find and Replace Console Bootcamp Runner Trash ?

```
36
37  /*@GetMapping("/clientes/{id}")
38  public Cliente show(@PathVariable Long id) {
39      return clienteService.findById(id);
40  }*/|
41  @GetMapping("/clientes/{id}")
42  public ResponseEntity<?> show(@PathVariable Long id){
43      Cliente cliente = null;
44      Map<String, Object> response = new HashMap<>();
45
46      try {
47          cliente = clienteService.findById(id);
48      } catch (DataAccessException e) {
49          response.put("mensaje", "Error al realizar consulta en base de datos");
50          response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
51          return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
52      }
53
54      if(cliente == null) {
55          response.put("mensaje", "El cliente ID: ".concat(id.toString()).concat("no existe en la base de datos"));
56          return new ResponseEntity<Map<String, Object>>(response, HttpStatus.NOT_FOUND);
57      }
58
59      return new ResponseEntity<Cliente>(cliente, HttpStatus.OK);
60  }
61
```

The screenshot shows the Postman application window. In the center, there is a request card for a GET request to `http://localhost:8087/api/clientes/22`. The 'Params' tab is selected, showing a single entry: 'Key' under 'KEY' and 'Value' under 'VALUE'. Below the request card, the 'Body' tab is selected, displaying the JSON response:

```
1 [ { "mensaje": "El cliente ID: 22no existe en la base de datos" } ]
```

The 'Headers' section shows five entries. At the bottom right of the main panel, there are buttons for 'Save Response' and 'Send'.

The left sidebar contains sections for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. The 'My Workspace' section is currently active, showing collections like 'Postman Echo' and 'PruebasApiRest'. The top navigation bar includes Home, Workspaces, API Network, Reports, Explore, and a search bar.

En nuestra entidad cliente programamos que el campo nombre no sea nulo, y el campo email no sea nulo y sea único

nacionjava/springboot/apirest/models/entity/Cliente.java - Eclipse IDE

```
16
17 @Entity
18 @Table(name = "clientes")
19 public class Cliente implements Serializable{
20
21
22@  @Id
23  @GeneratedValue(strategy= GenerationType.IDENTITY)
24  private long id;
25
26@  @Column(nullable=false)
27  private String nombre;
28  private String apellido;
29
30@  @Column(nullable=false,unique=true)
31  private String email;
32  private int telefono;
33
34@  @Column(name="create_at")
35  @Temporal(TemporalType.DATE)
36  private Date createdAt;
37
38
39@  @PrePersist
40  public void prePersist() {
41      createdAt = new Date();
```

Markers Properties Servers Data Source Explorer Snippets Console × Git Staging

<terminated> springboot-apirest - SpringbootApirestApplication [Spring Boot App] C:\Users\dell\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64.

Ahora agregamos a insertar

```
est/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE
Search Project Run Window Help
*ClienteRestController.java x
61
62
63  * @PostMapping("/clientes")
64  @ResponseStatus(HttpStatus.CREATED)
65  public Cliente create(@RequestBody Cliente cliente) {
66      return clienteService.save(cliente);
67  }/*
68
69  @PostMapping("/clientes")
70  public ResponseEntity<?> create(@RequestBody Cliente cliente){
71      Cliente clienteNew = null;
72      Map<String, Object> response = new HashMap<>();
73
74      try {
75          clienteNew = clienteService.save(cliente);
76      } catch (DataAccessException e) {
77          response.put("mensaje", "Error al realizar insert en base de datos");
78          response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
79          return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
80      }
81
82      response.put("mensaje", "El cliente ha sido creado con éxito!");
83      response.put("cliente", clienteNew);
84      return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);
85  }
```

Comprobamos las excepciones

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for cloud storage, invite, notifications, settings, and OAuth. Below the bar, a header shows a sequence of requests: GET, DEL, GET, and POST. The current request is a POST to 'http://localhost:8087/api/clientes'. The main workspace displays the request details:

- Method:** POST
- URL:** http://localhost:8087/api/clientes
- Body:** (8) JSON (selected)
- Params:** none
- Headers:** (5)
- Tests:** (none)
- Settings:** (none)

The JSON body is defined as:

```
1 {  
2   "nombre": "Rolando",  
3   "apellido": "Lopez",  
4 }
```

Below the request details, the response section shows:

- Status:** 201 Created
- Time:** 362 ms
- Size:** 370 B
- Save Response:** (dropdown)

The response body is displayed in Pretty, Raw, and Preview formats, and is also JSON-formatted:

```
1 {  
2   "cliente": {  
3     "id": 11,  
4     "nombre": "Rolando",  
5     "apellido": "Lopez",  
6     "email": "rlopez@hotmail.com",  
7     "telefono": 6432332,  
8     "createdAt": "2021-11-21T23:11:53.787+00:00"  
9   },  
10  "mensaje": "El cliente ha sido creado con éxito!"  
11 }
```

No nos deja registrar un usuario con un mismo email por la propiedad unique

The screenshot shows a POST request to `http://localhost:8087/api/clientes`. The request body is a JSON object:

```
1 {  
2   "nombre": "Rolando",  
3   "apellido": "Lopez",  
4   "email": "rlopez@hotmail.com",  
5   "telefono": 6432332  
6 }
```

The response status is 500 Internal Server Error with the message:

```
1 {  
2   "mensaje": "Error al realizar insert en base de datos",  
3   "error": "could not execute statement; SQL [n/a]; constraint [clientes.UK_1c96wv36rk2hwui7qhjks3mvg]; nested  
        exception is org.hibernate.exception.ConstraintViolationException: could not execute statement: Duplicate  
        entry 'rlopez@hotmail.com' for key 'clientes.UK_1c96wv36rk2hwui7qhjks3mvg'"  
4 }
```

Si no pasamos el campo nombre también nos informa que es requerido

The screenshot shows the Postman application interface. The URL in the header is `http://localhost:8087/api/clientes`. The method is set to `POST`. In the `Body` tab, the `JSON` tab is selected, and the following JSON object is present:

```
1 {
2   ...
3   "apellido": "Lopez",
4   "email": "rlopez@hotmail.com",
5   "telefono": 6432332
6 }
```

Below the request, the response section shows a `500 Internal Server Error` with a timestamp of `12 ms` and a size of `648 B`. The response body is:

```
2   "mensaje": "Error al realizar insert en base de datos",
3   "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.
entity.cliente.nombre; nested exception is org.hibernate.PropertyValueException: not-null property
references a null or transient value : com.formacionjava.springboot.apirest.models.entity.cliente.nombre;
not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.
entity.cliente.nombre"
```

También el campo email es requerido

The screenshot shows the Postman application interface. At the top, there are tabs for 'Reports' and 'Explore'. A search bar contains 'Search Postman'. On the right side of the header, there are icons for cloud sync, invite, notifications, settings, and upgrade.

The main area shows a sequence of requests: GET, DEL, GET, and POST. The POST request is selected, indicated by an orange border. The URL is set to `http://localhost:8087/api/clientes`. Below the URL, there are buttons for 'Save' and 'Send'.

The 'Body' tab is active, showing a JSON payload:

```
1 {  
2   "nombre": "Rolando",  
3   "apellido": "Lopez",  
4   ...  
5   "telefono": 6432332  
6 }
```

Below the body, the 'Body' tab is selected, followed by 'Cookies', 'Headers (4)', and 'Test Results'. The 'Test Results' section shows a 500 Internal Server Error response with a size of 645 B. There is a 'Save Response' button.

The 'Body' section at the bottom displays the error message in JSON format:

```
1 {  
2   "mensaje": "Error al realizar insert en base de datos",  
3   "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email; nested exception is org.hibernate.PropertyValueException: not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email; not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.email"  
}
```

At the very bottom, there are buttons for 'Bootcamp', 'Runner', 'Trash', and a help icon.

Ahora para el Update

```
ClienteRestController.java x
97     }*/  
98  
99     @PutMapping("/clientes/{id}")  
100    public ResponseEntity<?> update(@RequestBody Cliente cliente, @PathVariable Long id) {  
101        Cliente clienteActual= clienteService.findById(id);  
102  
103        Cliente clienteUpdated = null;  
104        Map<String, Object> response = new HashMap<>();  
105  
106        if(clienteActual == null){  
107            response.put("mensaje", "Error: no se pudo editar, el cliente ID: ".concat(id.toString()).concat("no existe el id en la base de datos"));  
108            return new ResponseEntity<Map<String, Object>>(response, HttpStatus.NOT_FOUND);  
109        }  
110  
111        try {  
112            clienteActual.setApellido(cliente.getApellido());  
113            clienteActual.setNombre(cliente.getNombre());  
114            clienteActual.setEmail(cliente.getEmail());  
115            clienteActual.setCreatedAt(cliente.getCreatedAt());  
116  
117            clienteUpdated = clienteService.save(clienteActual);  
118        } catch (DataAccessException e) {  
119            response.put("mensaje", "Error al actualizar el cliente en base de datos");  
120            response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));  
121            return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);  
122        }  
123  
124        response.put("mensaje", "El cliente ha sido actualizado con éxito!");  
125        response.put("cliente", clienteUpdated);  
126  
127        return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);  
128    }
```

Probamos con postman

The screenshot shows the Postman application window. At the top, there's a toolbar with 'Reports', 'Explore', a search bar ('Search Postman'), and various icons for cloud sync, invite, settings, and notifications. Below the toolbar, a header bar shows a sequence of requests: GET, DEL, GET, and PUT. The current request is highlighted in red as a PUT operation to the URL `http://localhost:8087/api/clientes/1`. The main interface shows the request details for this PUT method. Under the 'Body' tab, the JSON response is displayed:

```
1 {
2   "cliente": {
3     "id": 1,
4     "nombre": "Cristiano",
5     "apellido": "Ronaldo",
6     "email": "cr7@hotmail.com",
7     "telefono": 65433223,
8     "createdAt": "2021-11-21T00:00:00.000+00:00"
9   },
10  "mensaje": "El cliente ha sido actualizado con éxito!"
11 }
```

The response status is 201 Created, with a duration of 70 ms and a size of 376 B. Below the response, there are tabs for 'Pretty', 'Raw', 'Preview', 'Visualize', and 'JSON'. The 'JSON' tab is selected, and the response body is shown in a syntax-highlighted JSON format.

No me deja actualizar un email ya registrado

The screenshot shows a Postman interface with the following details:

- Request Method:** PUT
- Request URL:** http://localhost:8087/api/clientes/1
- Body Content (JSON):**

```
1 {  
2   "nombre": "Cristiano",  
3   "apellido": "Ronaldo",  
4   "email": "rlopez@hotmail.com",  
5   "telefono": 64323322,  
6   "createdAt": "2021-11-21"  
7 }
```

- Response Status:** 500 Internal Server Error
- Response Body (Pretty JSON):**

```
1 {  
2   "mensaje": "Error al actualizar el cliente en base de datos",  
3   "error": "could not execute statement; SQL [n/a]; constraint [clientes.UK_1c96wv36rk2hwui7qhjks3mvg];  
nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement:  
Duplicate entry 'rlopez@hotmail.com' for key 'clientes.UK_1c96wv36rk2hwui7qhjks3mvg'"  
4 }
```

No me deja actualizar sin el campo email

The screenshot shows the Postman application interface. At the top, there are several tabs: Reports, Explore, and a search bar. Below the tabs, a toolbar includes a cloud icon, an Invite button, settings, and notifications. The main workspace shows a sequence of requests: GET, DEL, GET, GET, PUT, DEL, and a plus sign. The PUT request is selected, pointing to the URL `http://localhost:8087/api/clientes/1`. The method dropdown shows "PUT". The "Body" tab is active, displaying the following JSON payload:

```
1 {  
2   "nombre": "Cristiano",  
3   "apellido": "Ronaldo",  
4   ...  
5   "telefono": 64323322,  
6   "createdAt": "2021-11-21"  
7 }
```

Below the Body tab, the "Headers" tab is visible with a count of 4. The "Test Results" tab is also present. The status bar at the bottom indicates a 500 Internal Server Error with a response size of 651 B. The "Pretty" tab is selected in the results view, showing the error response:

```
1 {  
2   "mensaje": "Error al actualizar el cliente en base de datos",  
3   "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.  
models.entity.Cliente.email; nested exception is org.hibernate.PropertyValueException: not-null  
property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.  
Cliente.email: not-null property references a null or transient value : com.formacionjava.springboot.
```

No me deja actualizar sin el campo nombre

The screenshot shows a Postman window with the following details:

- Request Method:** PUT
- Request URL:** http://localhost:8087/api/clientes/1
- Body Content:**

```
1 {
2     ...
3     "apellido": "Ronaldo",
4     "email": "cr7@hotmail.com",
5     "telefono": 64323322,
6     "createdAt": "2021-11-21"
7 }
```
- Response Status:** 500 Internal Server Error
- Response Body:**

```
1 {
2     "mensaje": "Error al actualizar el cliente en base de datos",
3     "error": "not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.nombre; nested exception is org.hibernate.PropertyValueException: not-null property references a null or transient value : com.formacionjava.springboot.apirest.models.entity.Cliente.nombre: not-null property references a null or transient value : com.formacionjava.springboot.
```

Manejo de error con Delete

```
... *ClienteRestController.java x
126
127     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);
128 }
129 */
130 @DeleteMapping("clientes/{id}")
131 public void delete(@PathVariable Long id) {
132     clienteService.delete(id);
133 }
134 */
135 @DeleteMapping("clientes/{id}")
136 public ResponseEntity<?> delete(@PathVariable Long id) {
137     Map<String, Object> response = new HashMap<>();
138
139     try {
140         clienteService.delete(id);
141     } catch (DataAccessException e) {
142         response.put("mensaje", "Error al eliminar el cliente en la base de datos");
143         response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
144         return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
145     }
146     response.put("mensaje", "El cliente ha sido eliminado con éxito!");
147
148
149     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.OK);
150 }
151 }
```

Reports Explore      Upgrade

POST h. ● GET h. ● DEL h. ● GET h. ● PUT h. ● DEL h. ● → + ⚙ No Environment    

http://localhost:8087/api/clientes/1

DELETE  http://localhost:8087/api/clientes/1 

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (5) Test Results  200 OK 89 ms 218 B 

Pretty Raw Preview Visualize   

```
1 {  
2   "mensaje": "El cliente ha sido eliminado con éxito!"  
3 }
```

Si no encuentra un id valido

The screenshot shows the Postman application interface. At the top, there is a toolbar with various icons for different HTTP methods (POST, GET, PUT, DELETE) and other functions. Below the toolbar, the URL `http://localhost:8087/api/clientes/20` is entered. The main area shows a **DELETE** request being sent to the same URL. The **Params** tab is selected, showing a single entry: **Key** with a value of **Value**. In the bottom section, the **Body** tab is selected, displaying the response from the server. The response is a JSON object with the following content:

```
1 {  
2   "mensaje": "Error al eliminar el cliente en la base de datos",  
3   "error": "No class com.formacionjava.springboot.apirest.models.entity.Cliente entity with id 20 exists!: No  
        class com.formacionjava.springboot.apirest.models.entity.Cliente entity with id 20 exists!"  
4 }
```

The status bar at the bottom indicates a **500 Internal Server Error** with a duration of **30 ms** and a size of **415 B**.

Crear una Api para subida de archivo, agregamos el siguiente campo y sus respectivos setter y getter

cionjava/springboot/apirest/models/entity/Cliente.java - Eclipse IDE

File Help

*ClienteRestController.java *Cliente.java x

```
3* import java.io.Serializable;□
15
16
17 @Entity
18 @Table(name = "clientes")
19 public class Cliente implements Serializable{
20
21
22@ Id
23 @GeneratedValue(strategy= GenerationType.IDENTITY)
24 private long id;
25
26@ Column(nullable=false)
27 private String nombre;
28 private String apellido;
29
30@ Column(nullable=false,unique=true)
31 private String email;
32 private int telefono;
33
34@ Column(name="create_at")
35 @Temporal(TemporalType.DATE)
36 private Date createdAt;
37
38 private String imagen;
```

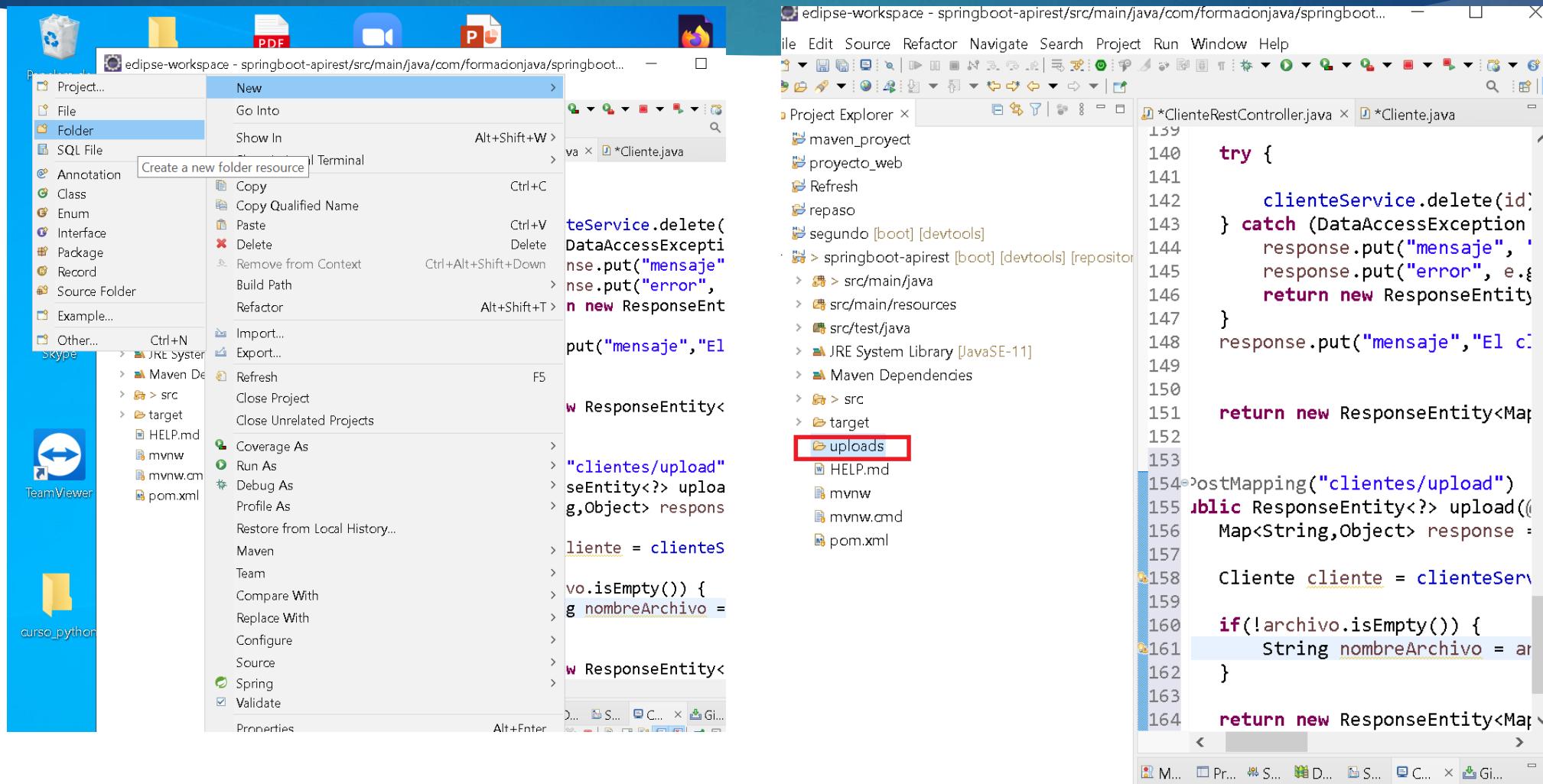
Window Help

*ClienteRestController.java *Cliente.java x

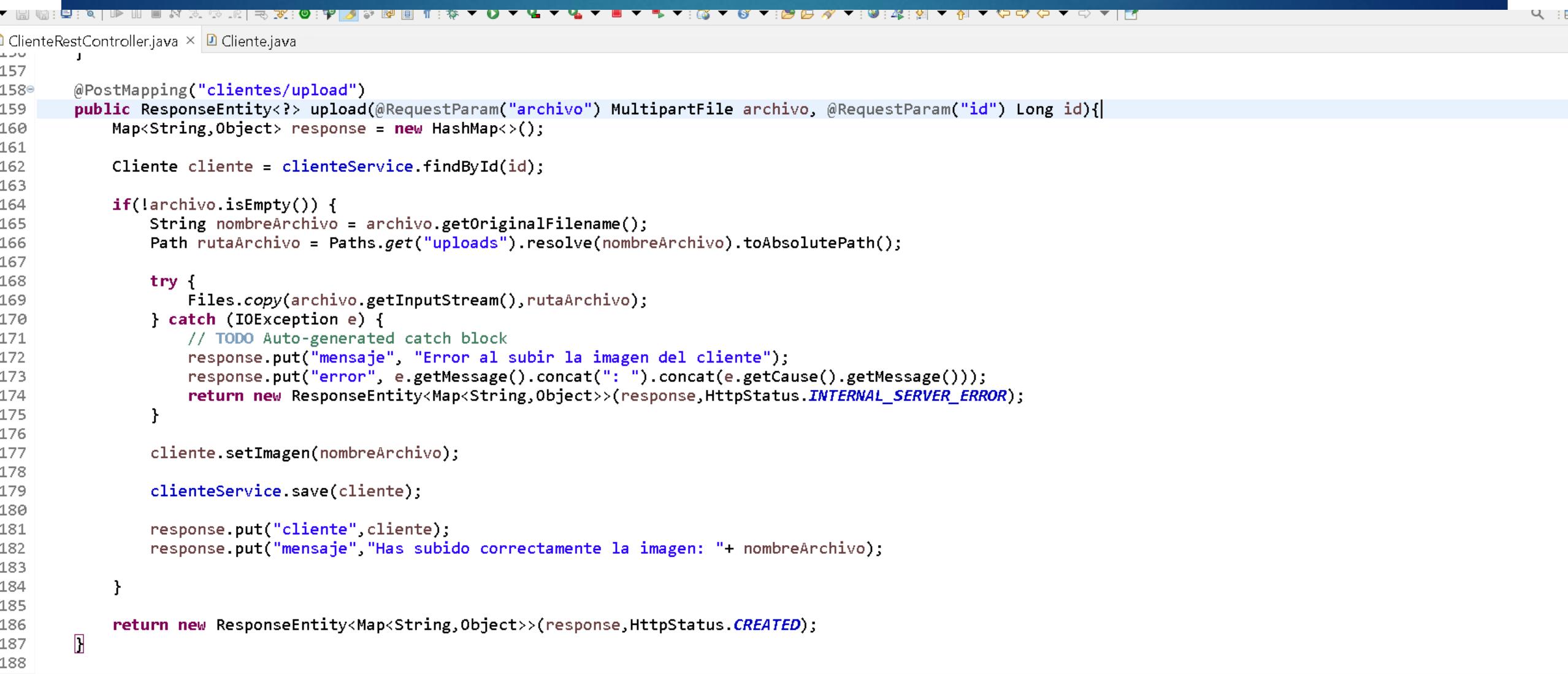
```
67  public void setEmail(String email) {
68      this.email = email;
69  }
70  public int getTelefono() {
71      return telefono;
72  }
73  public void setTelefono(int telefono) {
74      this.telefono = telefono;
75  }
76  public Date getCreatedAt() {
77      return createdAt;
78  }
79  public void setCreatedAt(Date createdAt) {
80      this.createdAt = createdAt;
81  }
82
83  public String getImagen() {
84      return imagen;
85  }
86
87  public void setImagen(String imagen) {
88      this.imagen = imagen;
89  }
90
91 /**
92 */


```

Creamos una carpeta en el directorio principal del proyecto para alojar las imágenes



Agregamos un nuevo método al controlador



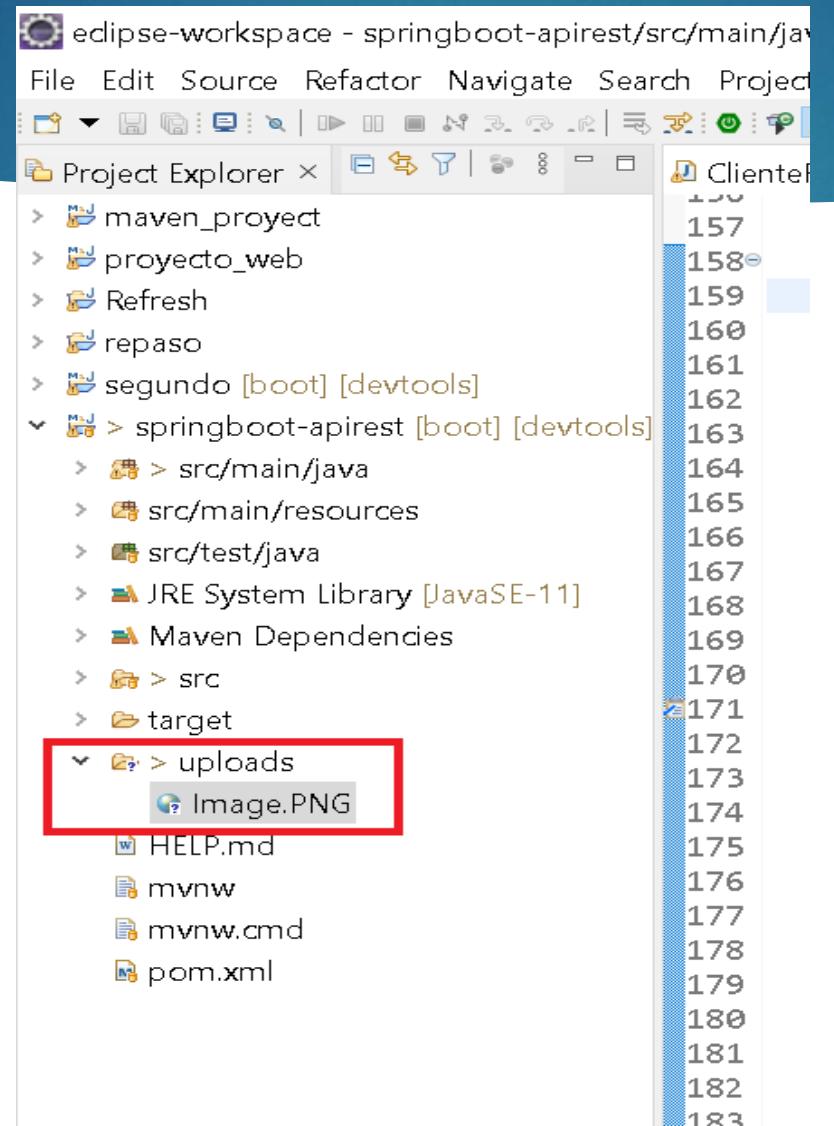
The screenshot shows a Java code editor with the following details:

- File:** ClienteRestController.java
- Line 158:** `@PostMapping("clientes/upload")`
- Line 159:** `public ResponseEntity<?> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){`
- Line 160:** `Map<String, Object> response = new HashMap<>();`
- Line 162:** `Cliente cliente = clienteService.findById(id);`
- Line 164:** `if(!archivo.isEmpty()) {`
- Line 165:** `String nombreArchivo = archivo.getOriginalFilename();`
- Line 166:** `Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();`
- Line 168:** `try {`
- Line 169:** `Files.copy(archivo.getInputStream(), rutaArchivo);`
- Line 170:** `} catch (IOException e) {`
- Line 171:** `// TODO Auto-generated catch block`
- Line 172:** `response.put("mensaje", "Error al subir la imagen del cliente");`
- Line 173:** `response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));`
- Line 174:** `return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);`
- Line 176:** `}`
- Line 177:** `cliente.setImagen(nombreArchivo);`
- Line 179:** `clienteService.save(cliente);`
- Line 181:** `response.put("cliente", cliente);`
- Line 182:** `response.put("mensaje", "Has subido correctamente la imagen: "+ nombreArchivo);`
- Line 184:** `}`
- Line 186:** `return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);`

Probamos la nueva api con postman

The screenshot shows the Postman application interface. The URL in the header is `http://localhost:8087/api/clientes/upload/`. The method is set to `POST`. The `Body` tab is selected, and the `form-data` option is chosen. Two fields are present: `archivo` with value `Image.PNG` and `id` with value `1`. The response status is `201 Created` with a time of `69 ms` and a size of `373 B`. The response body is displayed in JSON format:

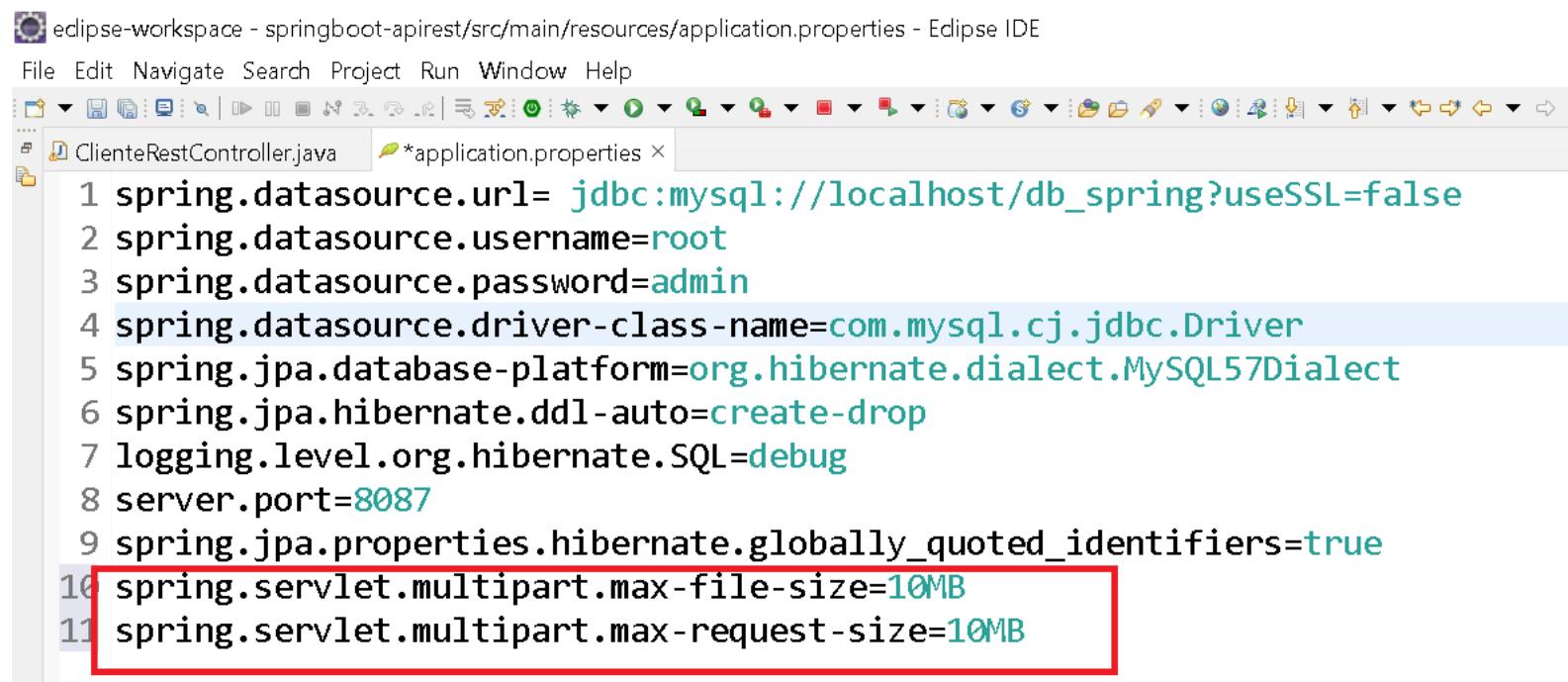
```
5 |     "apellido": "Perez",
6 |     "email": "jp@hotmail.com",
7 |     "telefono": 65433223,
8 |     "createdAt": "2021-10-01",
9 |     "imagen": "Image.PNG"
10 },
11 "mensaje": "Has subido correctamente la imagen: Image.PNG"
12 }
```



Realizamos mejoras a nuestra api

- ▶ Aumentar el tamaño permitido del archivo para subir por default esta limitado a 1MB
- ▶ Renombrar los archivos al subir para que no entren en conflicto al subir imágenes con el mismo nombre

Configuramos el tamaño permitido para subir archivos en application.properties



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - springboot-apirest/src/main/resources/application.properties - Eclipse IDE". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar below has various icons for file operations like Open, Save, Find, and Run. The editor window displays the contents of the application.properties file:

```
1 spring.datasource.url=jdbc:mysql://localhost/db_spring?useSSL=false
2 spring.datasource.username=root
3 spring.datasource.password=admin
4 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
5 spring.jpa.database-platform=org.hibernate.dialect.MySQL57Dialect
6 spring.jpa.hibernate.ddl-auto=create-drop
7 logging.level.org.hibernate.SQL=debug
8 server.port=8087
9 spring.jpa.properties.hibernate.globally_quoted_identifiers=true
10 spring.servlet.multipart.max-file-size=10MB
11 spring.servlet.multipart.max-request-size=10MB
```

Line numbers 10 and 11 are highlighted with a red rectangular box.

Agregamos la siguiente línea de código para generar nombre de archivos subidos



The screenshot shows the Eclipse IDE interface with the title bar "eclipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help. The toolbar has various icons for file operations. The left sidebar shows the project structure with "ClienteRestController.java" and "*application.properties" selected. The main editor area displays Java code for handling file uploads. Line 166 contains a highlighted line of code: "String nombreArchivo = UUID.randomUUID().toString()+"_"+ archivo.getOriginalFilename().replace(" ", "");". This line generates a unique file name by concatenating a random UUID with the original file name, replacing spaces with underscores.

```
159     @PostMapping("clientes/upload")
160     public ResponseEntity<?> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){
161         Map<String, Object> response = new HashMap<>();
162
163         Cliente cliente = clienteService.findById(id);
164
165         if(!archivo.isEmpty()) {
166             String nombreArchivo = UUID.randomUUID().toString()+"_"+ archivo.getOriginalFilename().replace(" ", "");
167             Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();
168
169             try {
170                 Files.copy(archivo.getInputStream(), rutaArchivo);
171             } catch (IOException e) {
172                 // TODO Auto-generated catch block
173                 response.put("mensaje", "Error al subir la imagen del cliente");
174                 response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));
175                 return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
176             }
177
178             cliente.setImagen(nombreArchivo);
179
180             clienteService.save(cliente);
```

Report POST GET DEL POST DEL → + ... No Environment Save Send

http://localhost:8087/api/clientes/upload/

POST http://localhost:8087/api/clientes/upload/ Send

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

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

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	archivo	WhatsApp Image 2021-04-21 at 09.5...			
<input checked="" type="checkbox"/>	id	1			

Body Cookies Headers (5) Test Results 201 Created 62 ms 511 B Save Response

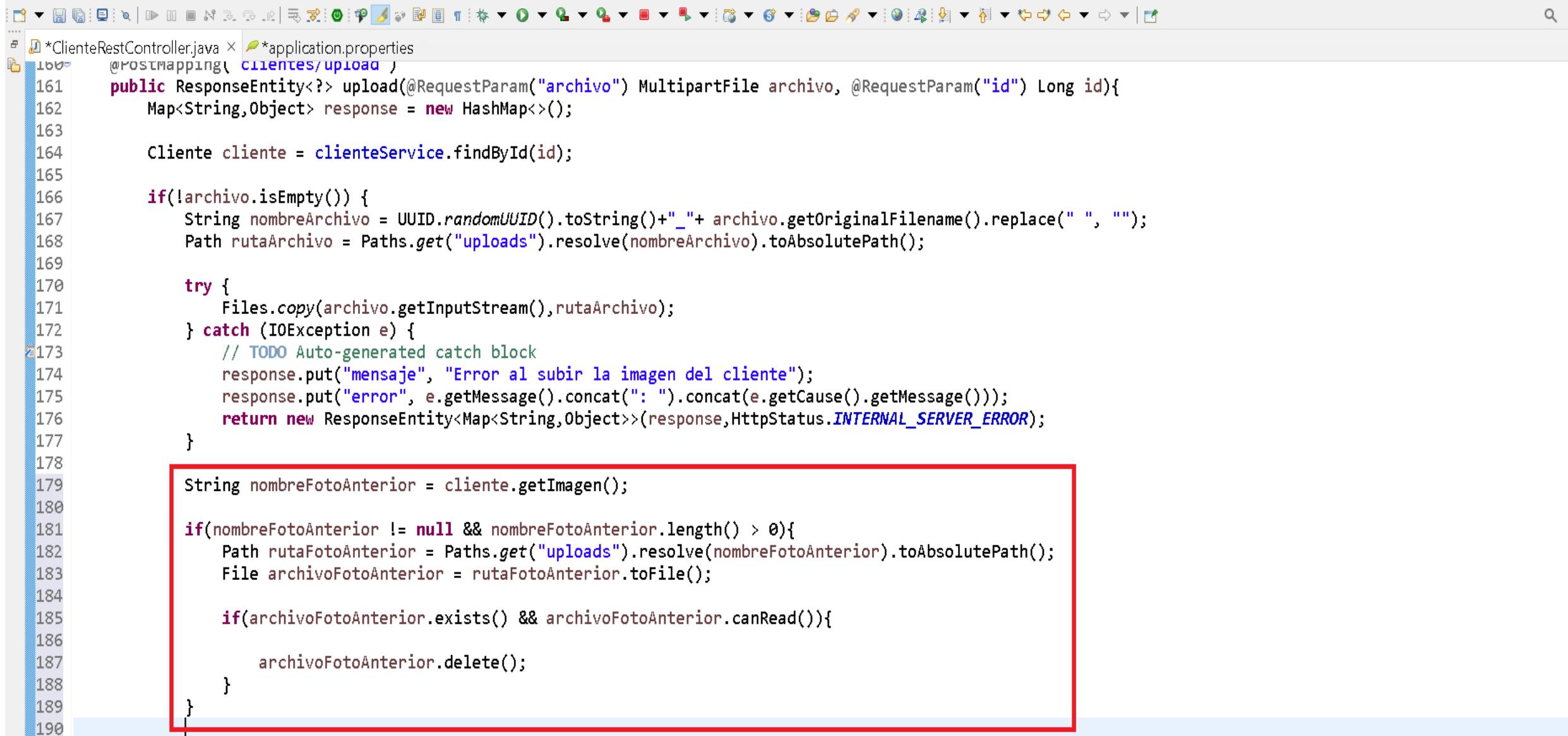
Pretty Raw Preview Visualize JSON Copy Search

```
6   "email": "jp@hotmail.com",
7   "telefono": 65433223,
8   "createdAt": "2021-10-01",
9   "imagen": "b58a2220-04ac-482b-8bef-37d1f7611976_WhatsAppImage2021-04-21at09.56.18(1).jpeg"
10 },
11   "mensaje": "Has subido correctamente la imagen:
12     b58a2220-04ac-482b-8bef-37d1f7611976_WhatsAppImage2021-04-21at09.56.18(1).jpeg"
```

Falta una ultima mejora

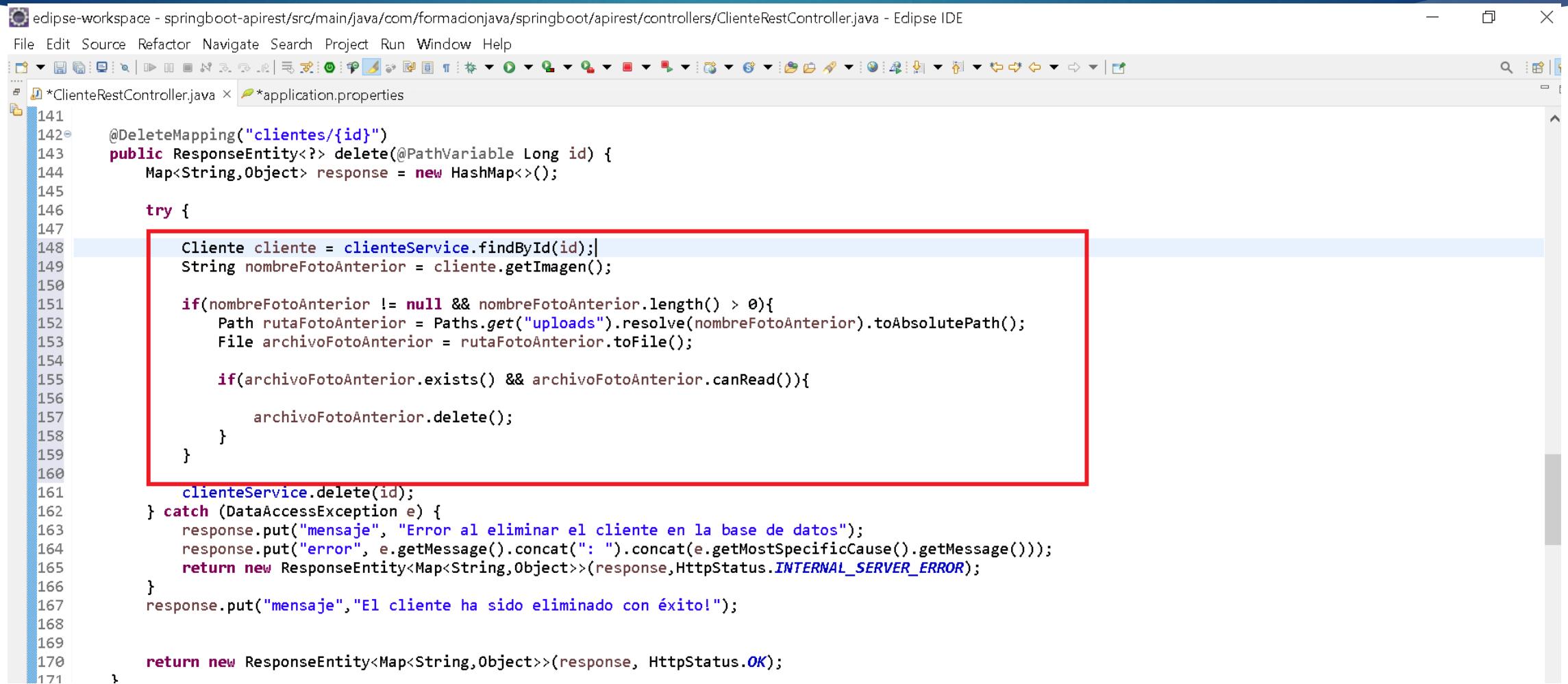
- ▶ Borrar los archivos asociados a un mismo usuario ya no se borra una vez reemplazado.

File Edit Source Refactor Navigate Search Project Run Window Help



```
*ClienteRestController.java x *application.properties
161 @PostMapping("clientes/upload")
162 public ResponseEntity<Map<String, Object>> upload(@RequestParam("archivo") MultipartFile archivo, @RequestParam("id") Long id){
163     Map<String, Object> response = new HashMap<>();
164
165     Cliente cliente = clienteService.findById(id);
166
167     if(!archivo.isEmpty()) {
168         String nombreArchivo = UUID.randomUUID().toString()+"_"+ archivo.getOriginalFilename().replace(" ", "_");
169         Path rutaArchivo = Paths.get("uploads").resolve(nombreArchivo).toAbsolutePath();
170
171         try {
172             Files.copy(archivo.getInputStream(), rutaArchivo);
173         } catch (IOException e) {
174             // TODO Auto-generated catch block
175             response.put("mensaje", "Error al subir la imagen del cliente");
176             response.put("error", e.getMessage().concat(": ").concat(e.getCause().getMessage()));
177             return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
178         }
179
180         String nombreFotoAnterior = cliente.getImagen();
181
182         if(nombreFotoAnterior != null && nombreFotoAnterior.length() > 0){
183             Path rutaFotoAnterior = Paths.get("uploads").resolve(nombreFotoAnterior).toAbsolutePath();
184             File archivoFotoAnterior = rutaFotoAnterior.toFile();
185
186             if(archivoFotoAnterior.exists() && archivoFotoAnterior.canRead()){
187
188                 archivoFotoAnterior.delete();
189             }
190         }
191     }
192 }
```

Agregamos el mismo código para eliminar imagen a la función delete



```
edipse-workspace - springboot-apirest/src/main/java/com/formacionjava/springboot/apirest/controllers/ClienteRestController.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
*ClienteRestController.java *application.properties
141
142 @DeleteMapping("clientes/{id}")
143 public ResponseEntity<?> delete(@PathVariable Long id) {
144     Map<String, Object> response = new HashMap<>();
145
146     try {
147
148         Cliente cliente = clienteService.findById(id);
149         String nombreFotoAnterior = cliente.getImagen();
150
151         if(nombreFotoAnterior != null && nombreFotoAnterior.length() > 0){
152             Path rutaFotoAnterior = Paths.get("uploads").resolve(nombreFotoAnterior).toAbsolutePath();
153             File archivoFotoAnterior = rutaFotoAnterior.toFile();
154
155             if(archivoFotoAnterior.exists() && archivoFotoAnterior.canRead()){
156
157                 archivoFotoAnterior.delete();
158             }
159         }
160
161         clienteService.delete(id);
162     } catch (DataAccessException e) {
163         response.put("mensaje", "Error al eliminar el cliente en la base de datos");
164         response.put("error", e.getMessage().concat(": ").concat(e.getMostSpecificCause().getMessage()));
165         return new ResponseEntity<Map<String, Object>>(response, HttpStatus.INTERNAL_SERVER_ERROR);
166     }
167     response.put("mensaje", "El cliente ha sido eliminado con éxito!");
168
169
170     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.OK);
171 }
```



File Edit View Help



Home Workspaces

API Network

Reports Explore

Search Postman



Invite



Upgrade



My Workspace

New Import



POST



GET



DEL



GET



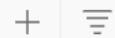
POST



DEL



No Environment



> Postman Echo

> PruebasApiRest

http://localhost:8087/api/clientes/upload/

Save



Send



POST

http://localhost:8087/api/clientes/upload/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY

archivo

VALUE

Tumble Road Multicolor.png

id

1

Body

Cookies

Headers (5)

Test Results



201 Created 41 ms 477 B

Save Response

Pretty

Raw

Preview

Visualize

JSON



4

"nombre": "Jose",
"apellido": "Perez",
"email": "jp@hotmail.com",
"telefono": 65433223,
"createdAt": "2021-10-01",
"imagen": "ab122359-53af-453d-870f-febdb865c074_TumbleRoadMulticolor.png"

5

6

7

8

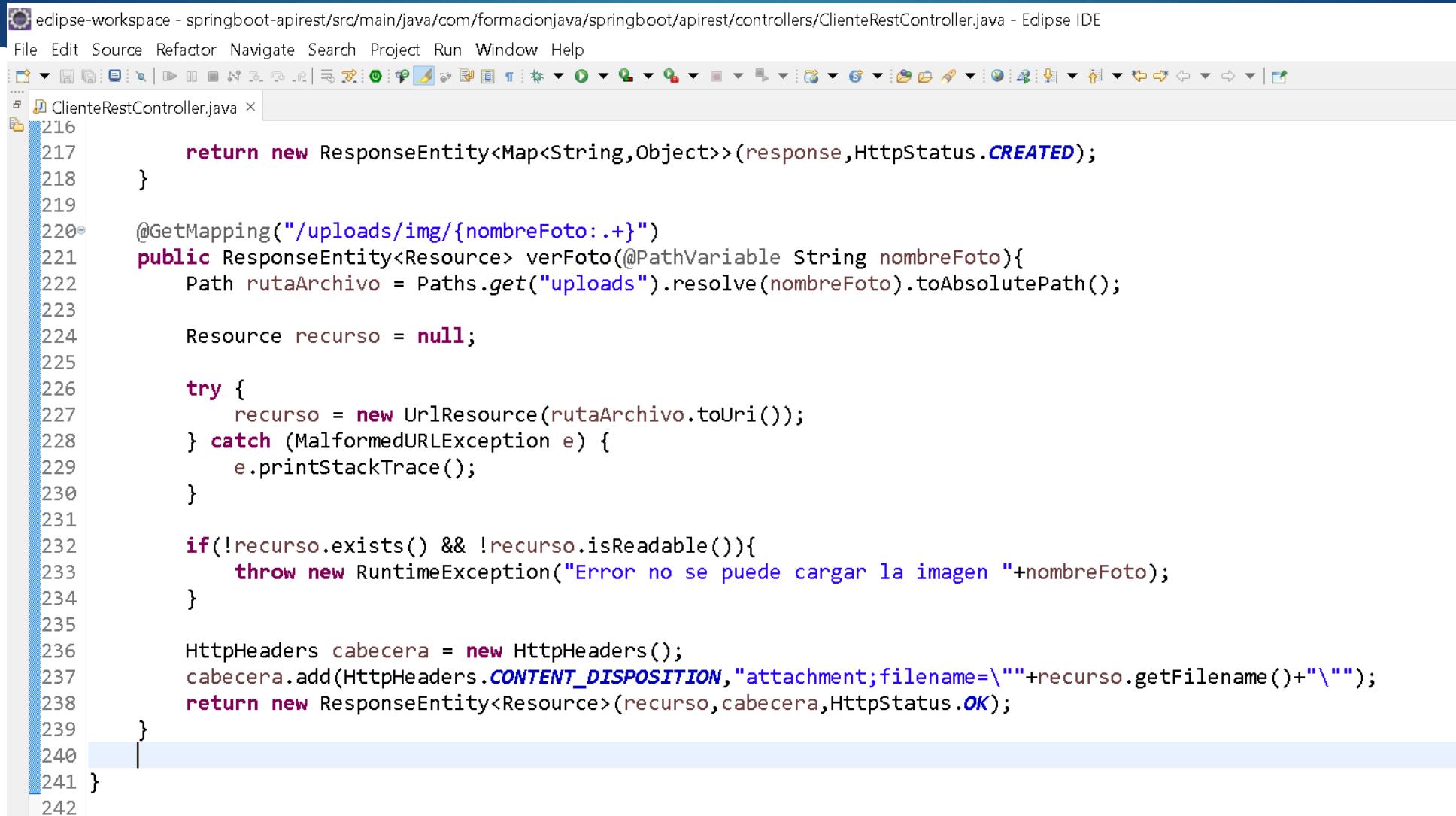
9

10

11

},
"mensaje": "Has subido correctamente la imagen: ab122359-53af-453d-870f-febdb865c074_TumbleRoadMulticolor.png"

Agregamos un API mas para mostrar imágenes



The screenshot shows the Eclipse IDE interface with the file `ClienteRestController.java` open. The code implements a REST controller for handling image files stored in the `uploads` directory.

```
216     return new ResponseEntity<Map<String, Object>>(response, HttpStatus.CREATED);
217 }
218
219
220 @GetMapping("/uploads/img/{nombreFoto:.+}")
221 public ResponseEntity<Resource> verFoto(@PathVariable String nombreFoto){
222     Path rutaArchivo = Paths.get("uploads").resolve(nombreFoto).toAbsolutePath();
223
224     Resource recurso = null;
225
226     try {
227         recurso = new UrlResource(rutaArchivo.toUri());
228     } catch (MalformedURLException e) {
229         e.printStackTrace();
230     }
231
232     if(!recurso.exists() && !recurso.isReadable()){
233         throw new RuntimeException("Error no se puede cargar la imagen "+nombreFoto);
234     }
235
236     HttpHeaders cabecera = new HttpHeaders();
237     cabecera.add(HttpHeaders.CONTENT_DISPOSITION, "attachment;filename=\""+recurso.getFilename()+"\"");
238     return new ResponseEntity<Resource>(recurso, cabecera, HttpStatus.OK);
239 }
240
241 }
```



File Edit View Help



Home

Workspaces

API Network

Reports

Explore

Search Postman



+ Invite



Upgrade



ST



I



GET

h.



POST

t



POST

t



GET

h.



DEL

h.



GET

h.



POST

t



DEL

h..



PUT

h.



→



...



No Environment



http://localhost:8087/api/clientes/upload

Save



POST http://localhost:8087/api/clientes/upload

Send

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

Cookies

none form-data x-www-form-urlencoded

raw binary GraphQL

archivo

Image (1).PNG

id

1

Key

Value

Description

Body Cookies Headers (5) Test Results

Status: 201 Created Time: 104 ms Size: 453 B

Save Response

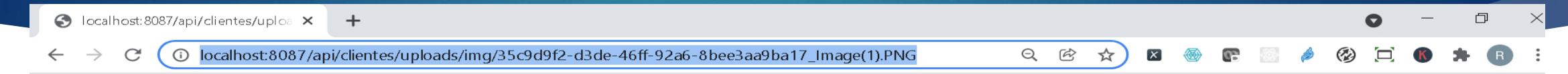
Pretty Raw Preview Visualize

JSON



```
5      "apellido": "Perez",
6      "email": "jp@hotmail.com",
7      "telefono": 65433223,
8      "createdAt": "2021-10-01",
9      "imagen": "35c9d9f2-d3de-46ff-92a6-8bee3aa9ba17_Image(1).PNG"
10     },
11      "mensaje": "Has subido correctamente la imagen: 35c9d9f2-d3de-46ff-92a6-8bee3aa9ba17_Image(1).PNG"
12 }
```

[http://localhost:8087/api/uploads/img/35c9d9f2-d3de-46ff-92a6-8bee3aa9ba17_Image\(1\).PNG](http://localhost:8087/api/uploads/img/35c9d9f2-d3de-46ff-92a6-8bee3aa9ba17_Image(1).PNG)



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Nov 23 05:31:21 CET 2021

There was an unexpected error (type=Not Found, status=404).

No message available