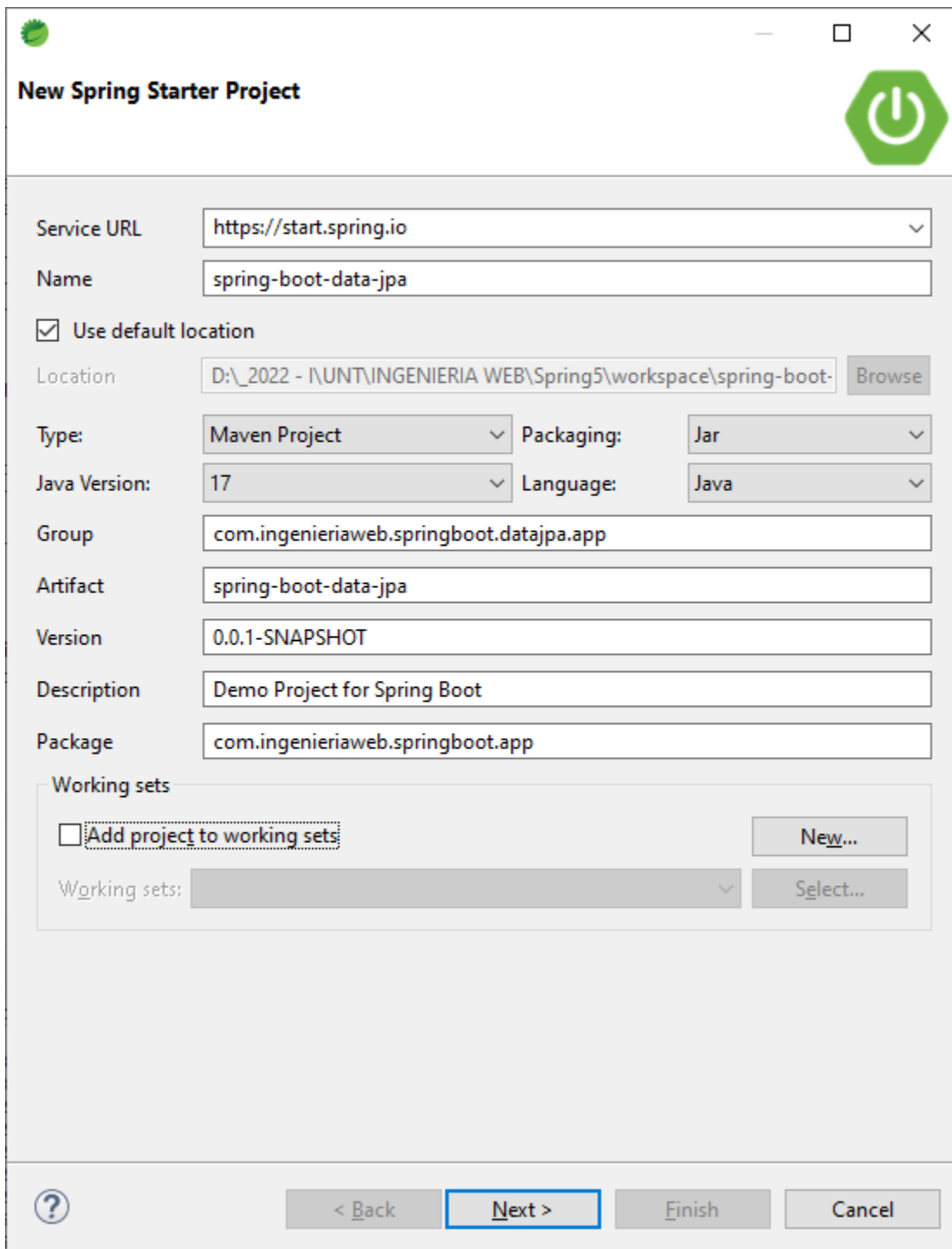


Creamos un proyecto llamado spring-boot-data-jpa



The image shows a 'New Spring Starter Project' dialog box. At the top, there is a title bar with a green icon and standard window controls. Below the title bar, the title 'New Spring Starter Project' is displayed next to a green hexagonal icon with a power symbol. The main area contains several fields and options for configuring a new project. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' field contains 'spring-boot-data-jpa'. There is a checked checkbox for 'Use default location'. The 'Location' field shows a file path 'D:_2022 - I\UNT\INGENIERIA WEB\Spring5\workspace\spring-boot-' with a 'Browse' button. Below this, there are two rows of dropdown menus: 'Type:' (Maven Project), 'Packaging:' (Jar), 'Java Version:' (17), and 'Language:' (Java). Further down are text fields for 'Group' (com.ingenieriaweb.springboot.datajpa.app), 'Artifact' (spring-boot-data-jpa), 'Version' (0.0.1-SNAPSHOT), 'Description' (Demo Project for Spring Boot), and 'Package' (com.ingenieriaweb.springboot.app). A 'Working sets' section at the bottom includes a checkbox for 'Add project to working sets', a 'New...' button, a 'Working sets:' dropdown menu, and a 'Select...' button. At the very bottom, there is a navigation bar with a help icon, and buttons for '< Back', 'Next >', 'Finish', and 'Cancel'. The 'Next >' button is highlighted with a blue border.

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

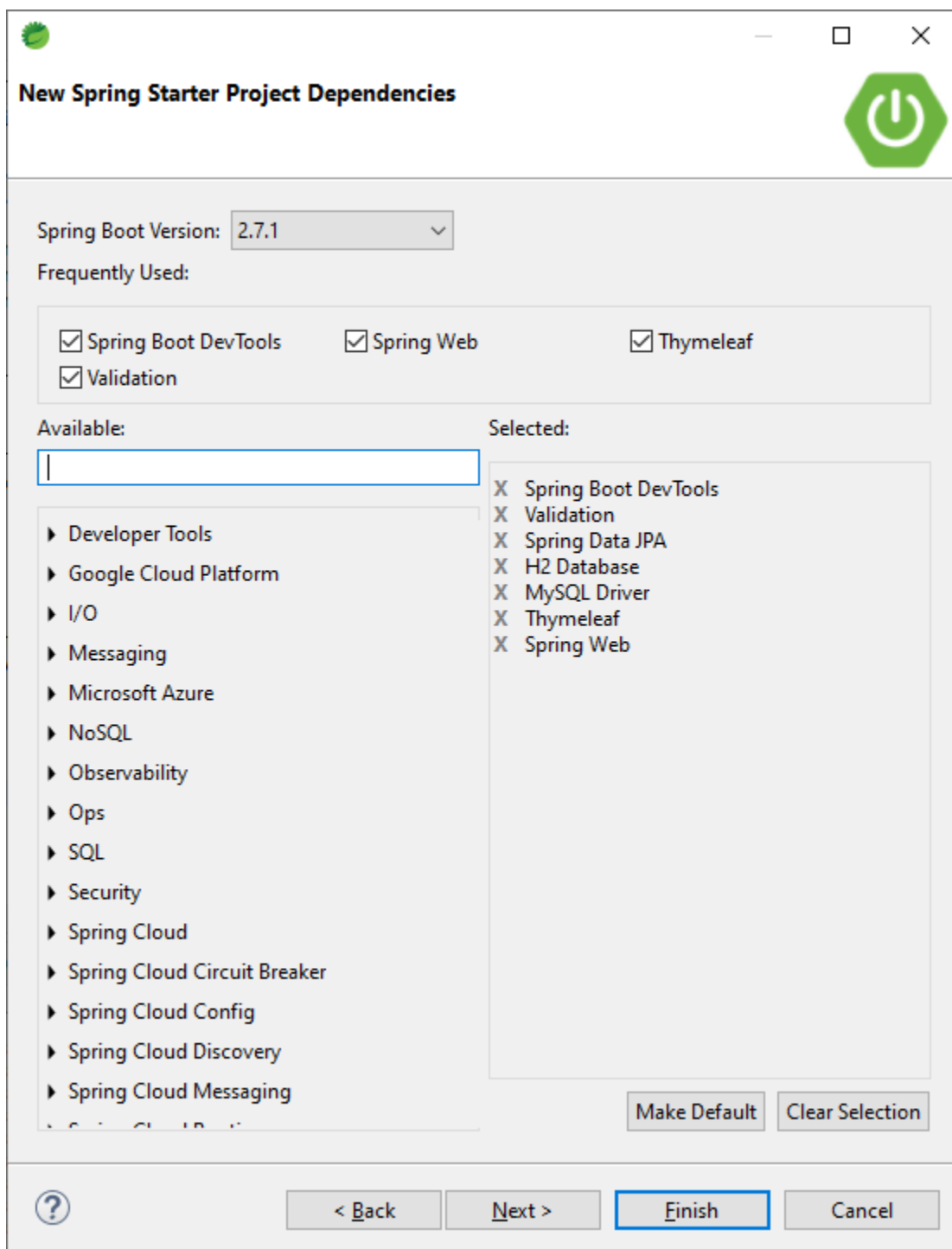
Package:

Working sets

☐ Add project to working sets

Working sets:

Marcar las dependencias:



The dialog box is titled "New Spring Starter Project Dependencies" and features a green power button icon in the top right corner. It includes a "Spring Boot Version:" dropdown menu set to "2.7.1". Below this is a "Frequently Used:" section with four checked checkboxes: "Spring Boot DevTools", "Spring Web", "Thymeleaf", and "Validation". The main area is divided into two columns: "Available:" and "Selected:". The "Available:" column contains a search bar and a list of dependency categories with expandable arrows, including Developer Tools, Google Cloud Platform, I/O, Messaging, Microsoft Azure, NoSQL, Observability, Ops, SQL, Security, Spring Cloud, Spring Cloud Circuit Breaker, Spring Cloud Config, Spring Cloud Discovery, and Spring Cloud Messaging. The "Selected:" column lists the chosen dependencies with an 'X' in front of each: Spring Boot DevTools, Validation, Spring Data JPA, H2 Database, MySQL Driver, Thymeleaf, and Spring Web. At the bottom right of the main area are "Make Default" and "Clear Selection" buttons. The footer contains a help icon, a "< Back" button, a "Next >" button, a highlighted "Finish" button, and a "Cancel" button.

New Spring Starter Project Dependencies

Spring Boot Version: 2.7.1

Frequently Used:

- ☒ Spring Boot DevTools
- ☒ Spring Web
- ☒ Thymeleaf
- ☒ Validation

Available:

- Developer Tools
- Google Cloud Platform
- I/O
- Messaging
- Microsoft Azure
- NoSQL
- Observability
- Ops
- SQL
- Security
- Spring Cloud
- Spring Cloud Circuit Breaker
- Spring Cloud Config
- Spring Cloud Discovery
- Spring Cloud Messaging

Selected:

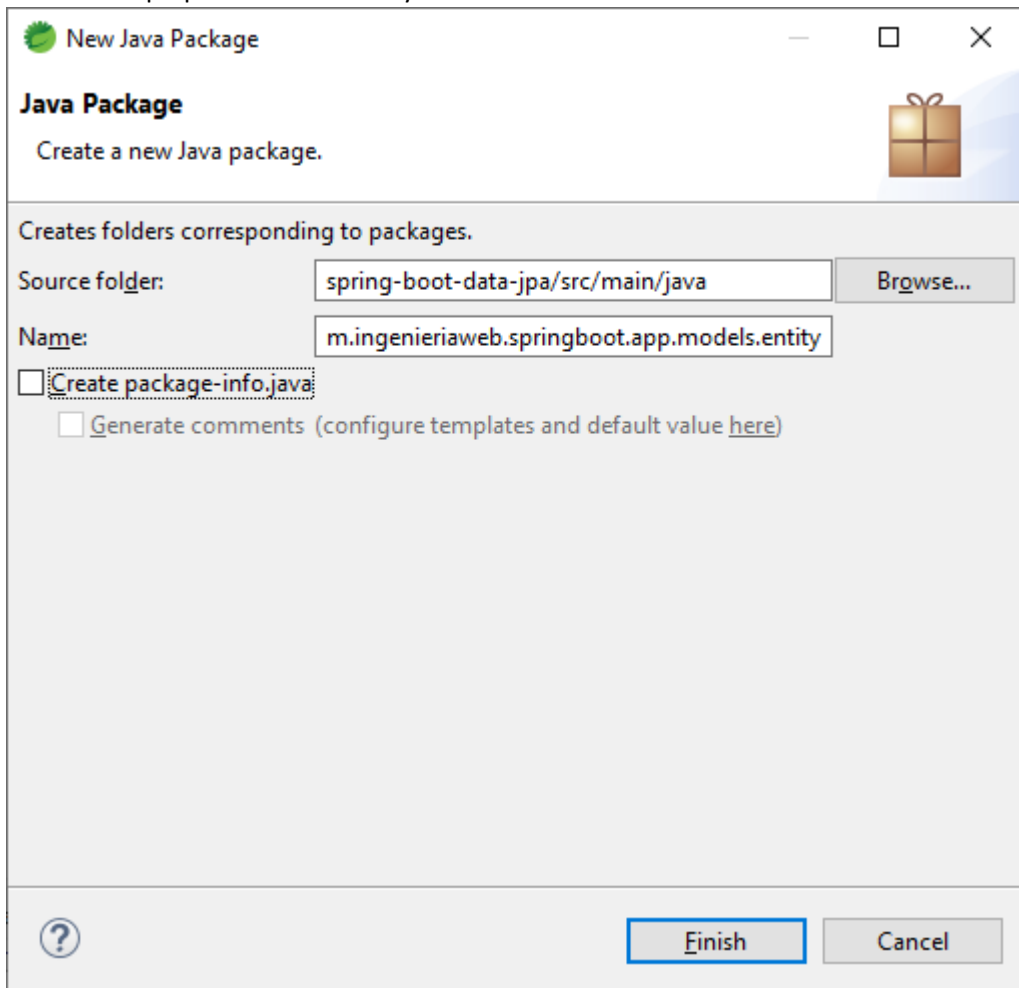
- X Spring Boot DevTools
- X Validation
- X Spring Data JPA
- X H2 Database
- X MySQL Driver
- X Thymeleaf
- X Spring Web

Make Default Clear Selection

? < Back Next > Finish Cancel

Creando la Entidad JPA anotada con @Entity

Creamos el paquete models.entity



New Java Package

Java Package
Create a new Java package.

Creates folders corresponding to packages.

Source folder:

Name:

☐ Create package-info.java

☐ Generate comments (configure templates and default value [here](#))

Dentro de este paquete creamos una clase Cliente

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

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

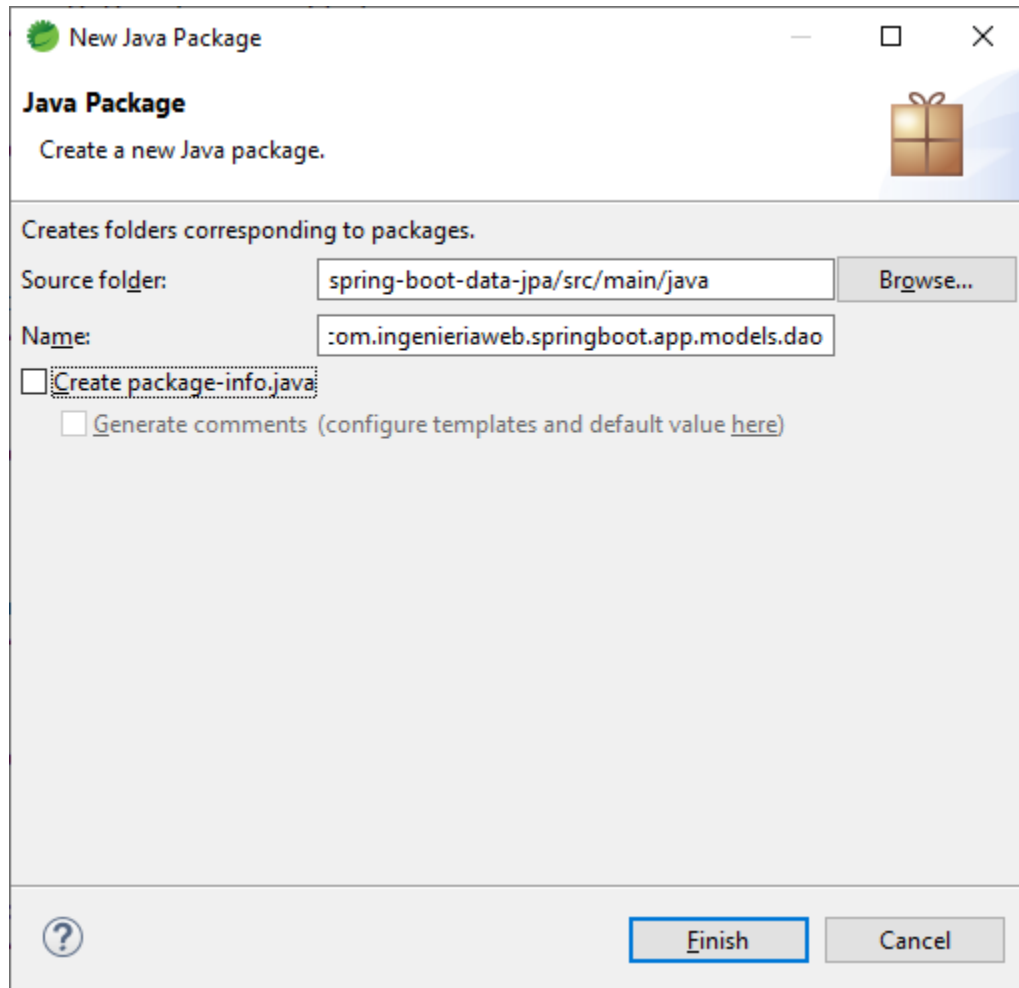
Implementamos de Serializable y le agregamos un serialVersionUID por defecto.

```
1 package com.ingenieriaweb.springboot.app.models.entity;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.persistence.Table;
7
8 @Entity
9 @Table(name="clientes")
10 public class Cliente implements Serializable{
11
12     private static final long serialVersionUID = 1L;
13
14
15 }
```

Luego colocamos sus atributos y luego colocamos sus setters y getters.

```
1 package com.ingenieriaweb.springboot.app.models.entity;
2
3 import java.io.Serializable;
4 import java.util.Date;
5 import javax.persistence.Column;
6 import javax.persistence.Entity;
7 import javax.persistence.GeneratedValue;
8 import javax.persistence.GenerationType;
9 import javax.persistence.Id;
10 import javax.persistence.Table;
11 import javax.persistence.Temporal;
12 import javax.persistence.TemporalType;
13
14 @Entity
15 @Table(name="clientes")
16 public class Cliente implements Serializable{
17     @Id
18     @GeneratedValue(strategy=GenerationType.IDENTITY)
19     private Long id;
20
21     private String nombre;
22
23     private String apellido;
24
25     private String email;
26
27     @Column(name="create_at")
28     @Temporal(TemporalType.DATE)
29     private Date createAt;
30
31     private static final long serialVersionUID = 1L;
32 }
33
```

Ahora creamos el paquete dao



The image shows a 'New Java Package' dialog box from an IDE. It has a title bar with a green icon and standard window controls. The main area is titled 'Java Package' with a gift icon and the instruction 'Create a new Java package.' Below this, it says 'Creates folders corresponding to packages.' There are two input fields: 'Source folder:' with the value 'spring-boot-data-jpa/src/main/java' and a 'Browse...' button, and 'Name:' with the value ':com.ingenieriaweb.springboot.app.models.dao'. There are two checkboxes: 'Create package-info.java' (checked) and 'Generate comments (configure templates and default value here)' (unchecked). At the bottom, there is a help icon, a 'Finish' button, and a 'Cancel' button.

New Java Package

Java Package
Create a new Java package.

Creates folders corresponding to packages.

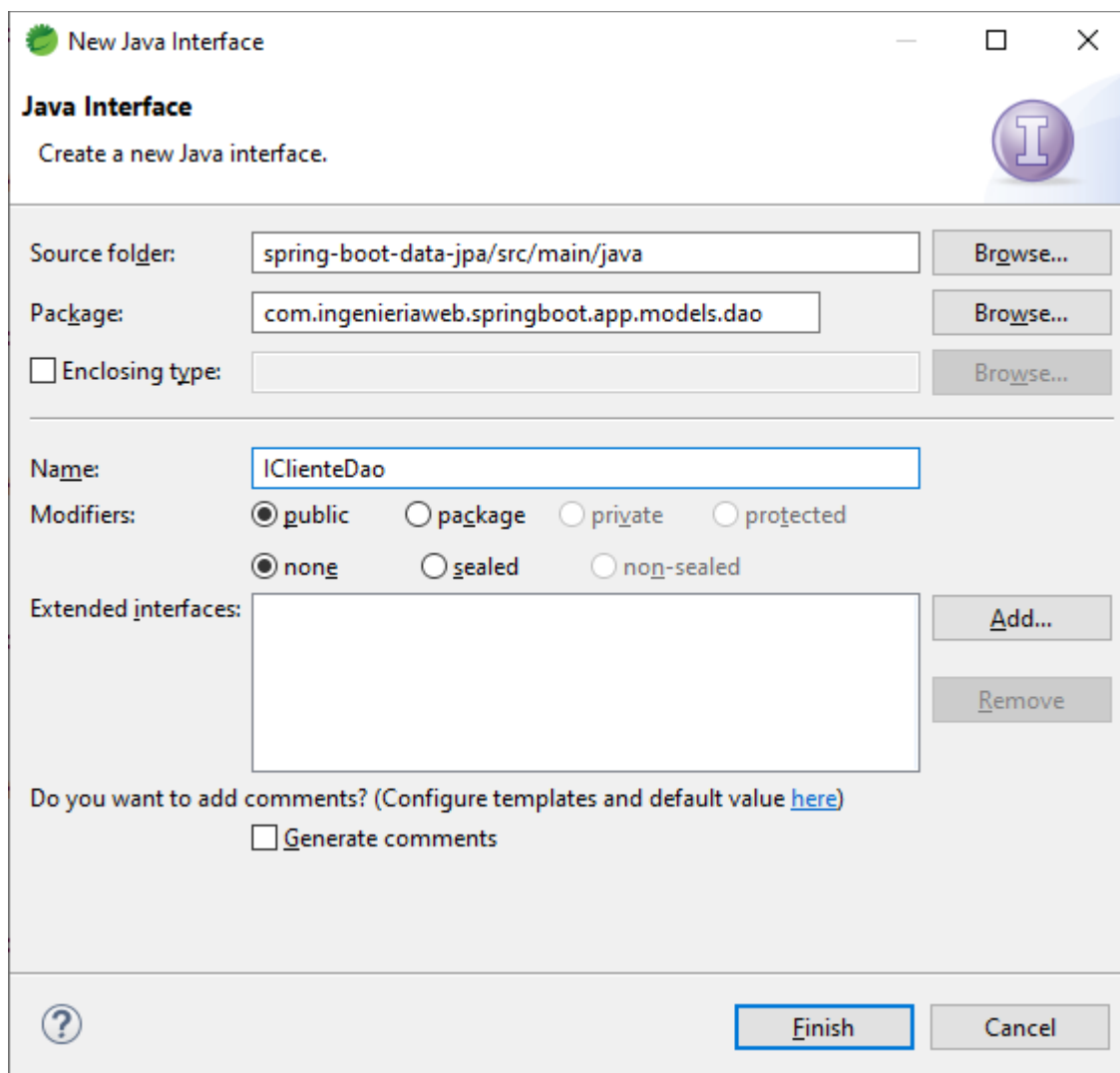
Source folder:

Name:

☒ Create package-info.java

☐ Generate comments (configure templates and default value [here](#))

Creamos la interfaz: ICienteDao



The image shows a 'New Java Interface' dialog box from an IDE. The title bar says 'New Java Interface' with a green icon. The main title is 'Java Interface' and the subtitle is 'Create a new Java interface.' There is a purple icon with a white 'I' in the top right corner. The dialog has several input fields and buttons. The 'Source folder' field contains 'spring-boot-data-jpa/src/main/java' with a 'Browse...' button. The 'Package' field contains 'com.ingenieriaweb.springboot.app.models.dao' with a 'Browse...' button. There is an unchecked checkbox for 'Enclosing type' with a 'Browse...' button. The 'Name' field contains 'ICienteDao'. The 'Modifiers' section has radio buttons for 'public' (selected), 'package', 'private', 'protected', 'none' (selected), 'sealed', and 'non-sealed'. The 'Extended interfaces' section has an empty text area with 'Add...' and 'Remove' buttons. At the bottom, there is a question 'Do you want to add comments? (Configure templates and default value [here](#))' with an unchecked checkbox for 'Generate comments'. The bottom right has 'Finish' and 'Cancel' buttons, and a help icon (?) is on the bottom left.

New Java Interface

Java Interface
Create a new Java interface.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☒ none ☐ sealed ☐ non-sealed

Extended interfaces:

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

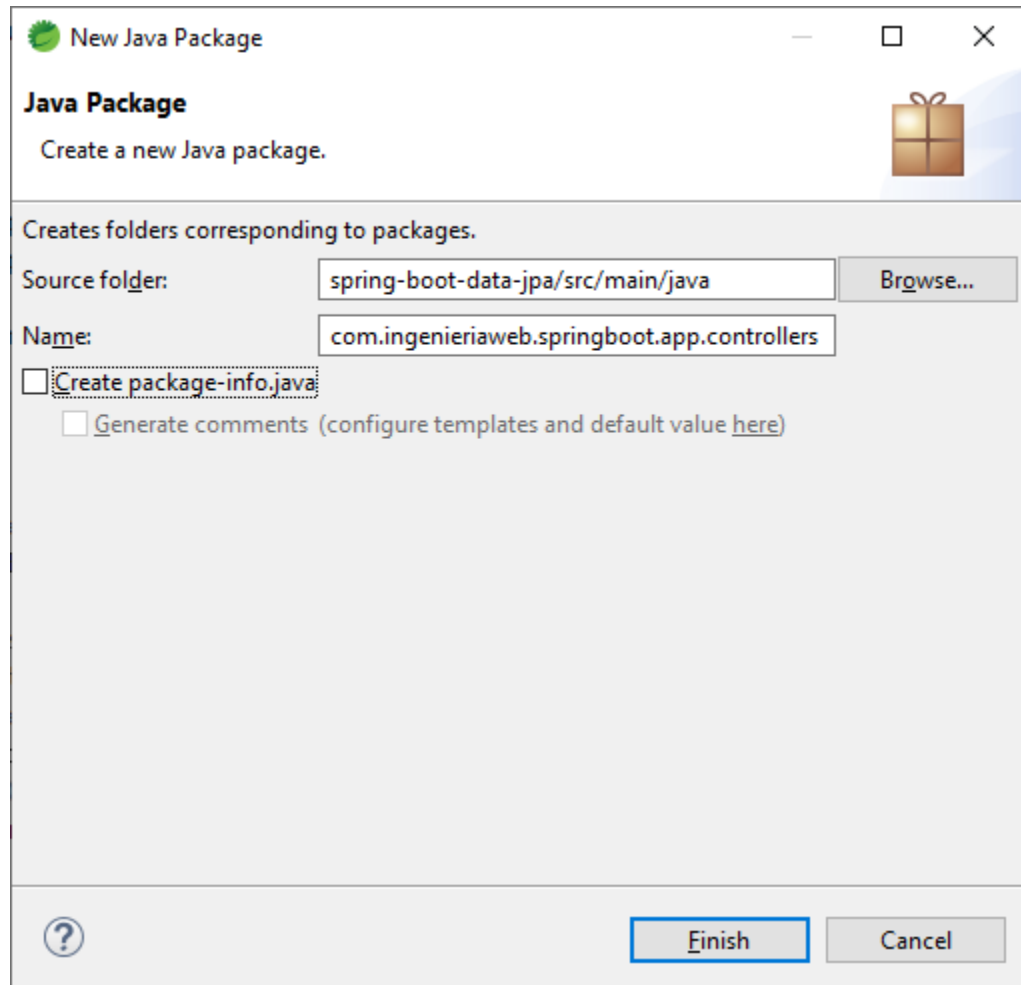
Luego creamos la clase ClienteDaoImpl que implementa de la interfaz IClienteDao

```
1 package com.ingenieriaweb.springboot.app.models.dao;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6
7 import org.springframework.stereotype.Repository;
8
9 import com.ingenieriaweb.springboot.app.models.entity.Cliente;
10
11 @Repository
12 public class ClienteDaoImpl implements IClienteDao {
13
14     private EntityManager em;
15
16     @Override
17     public List<Cliente> findAll() {
18         // TODO Auto-generated method stub
19         return null;
20     }
21 }
```

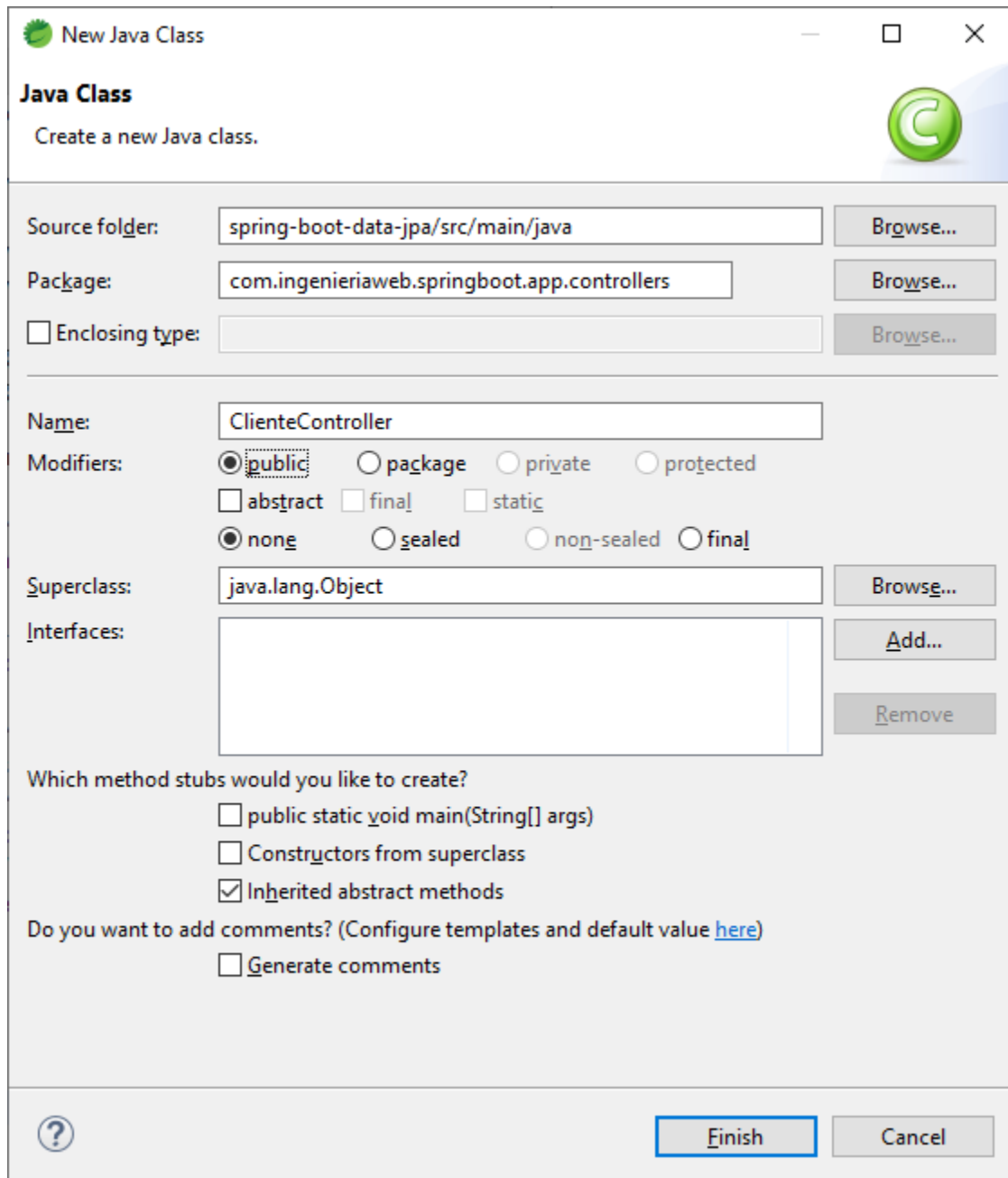
```
1 package com.ingenieriaweb.springboot.app.models.dao;
2
3 import java.util.List;
4
5 import javax.persistence.EntityManager;
6 import javax.persistence.PersistenceContext;
7
8 import org.springframework.stereotype.Repository;
9 import org.springframework.transaction.annotation.Transactional;
10
11 import com.ingenieriaweb.springboot.app.models.entity.Cliente;
12
13 @Repository
14 public class ClienteDaoImpl implements IClienteDao {
15
16     @PersistenceContext
17     private EntityManager em;
18
19     @SuppressWarnings("unchecked")
20     @Transactional(readOnly=true)
21     @Override
22     public List<Cliente> findAll() {
23         // TODO Auto-generated method stub
24         return em.createQuery("from Cliente").getResultList();
25     }
26
27 }
```


Creando Controlador con la acción Handler Listar

Creamos el paquete controllers



Creamos la clase Controladora Cliente Controller



The image shows a 'New Java Class' dialog box from an IDE. The title bar says 'New Java Class' with a green icon. Below the title bar, it says 'Java Class' and 'Create a new Java class.' with a green 'C' icon. The dialog has several sections: 'Source folder' with a text box containing 'spring-boot-data-jpa/src/main/java' and a 'Browse...' button; 'Package' with a text box containing 'com.ingenieriaweb.springboot.app.controllers' and a 'Browse...' button; 'Enclosing type' with an unchecked checkbox and a 'Browse...' button. Below these is a horizontal line. The 'Name' section has a text box containing 'ClienteController'. The 'Modifiers' section has radio buttons for 'public' (selected), 'package', 'private', and 'protected', and checkboxes for 'abstract', 'final', and 'static'. Below these are radio buttons for 'none' (selected), 'sealed', 'non-sealed', and 'final'. The 'Superclass' section has a text box containing 'java.lang.Object' and a 'Browse...' button. The 'Interfaces' section has an empty text box, an 'Add...' button, and a 'Remove' button. Below these is the question 'Which method stubs would you like to create?' with checkboxes for 'public static void main(String[] args)', 'Constructors from superclass', and 'Inherited abstract methods' (checked). Below that is the question 'Do you want to add comments? (Configure templates and default value [here](#))' with a checkbox for 'Generate comments'. At the bottom, there is a question mark icon, a 'Finish' button, and a 'Cancel' button.

New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static
☒ none ☐ sealed ☐ non-sealed ☐ final

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

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

```
1 package com.ingenieriaweb.springboot.app.controllers;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.Model;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestMethod;
8
9 import com.ingenieriaweb.springboot.app.models.dao.IClienteDao;
10
11 @Controller
12 public class ClienteController {
13
14     @Autowired
15     private IClienteDao clienteDao;
16
17     @RequestMapping(value="listar",method=RequestMethod.GET)
18     public String listar(Model model) {
19         model.addAttribute("titulo","listado de clientes");
20         model.addAttribute("clientes",clienteDao.findAll());
21         return "listar";
22     }
23
24 }
```

Creando la Vista Listar.html en resources/templates

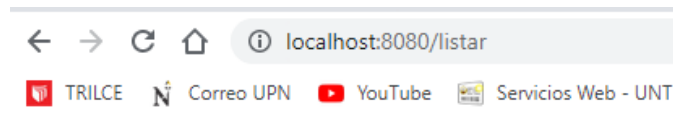
```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head>
4 <meta charset="ISO-8859-1" />
5 <title th:text="${titulo}"></title>
6 </head>
7 <body>
8 <h1 th:text="${titulo}"></h1>
9 <table>
10 <thead>
11 <tr>
12 <th>id</th>
13 <th>nombre</th>
14 <th>apellido</th>
15 <th>email</th>
16 <th>fecha</th>
17 </tr>
18 </thead>
19 <tbody>
20 <tr th:each="cliente: ${clientes}">
21 <td th:text="${cliente.id}"></td>
22 <td th:text="${cliente.nombre}"></td>
23 <td th:text="${cliente.apellido}"></td>
24 <td th:text="${cliente.email}"></td>
25 <td th:text="${cliente.createAt}"></td>
26 </tr>
27 </tbody>
28 </table>
29
30 </body>
31 </html>
```

Ahora en resources creamos el archivo import.sql con los datos de prueba.

insert into clientes(id,nombre,apellido,email,create_at) values
(1,'marcelino','torres','mtorres@unitru.edu.pe','2021-08-28');

insert into clientes(id,nombre,apellido,email,create_at) values
(2,'juan','perez','jperez@unitru.edu.pe','2022-03-20');

al Ejecutar sale:



listado de clientes

id	nombre	apellido	email	fecha
1	marcelino	torres	mtorres@unitru.edu.pe	2021-08-28
2	juan	perez	jperez@unitru.edu.pe	2022-03-20

La consola de H2

Nos vamos al archivo application.properties

```
spring.datasource.url=jdbc:h2:mem:clientesdb  
spring.datasource.username=marcelino  
spring.datasource.password=sa  
spring.datasource.driver-class-name=org.h2.Driver  
spring.h2.console.enabled=true
```

La base de datos se llama : testdb y el usuario es sa, sin clave

Levantamos el proyecto y escribimos : <http://h2-console>

