



**11<sup>a</sup>**  
Emisión

**DIPLOMADO**  
**Desarrollo de Sistemas**  
**con Tecnología Java**

**Módulo 7**  
**Persistencia con Spring Data**

*Dr. Omar Mendoza González*

*omarmendoza564@aragon.unam.mx*

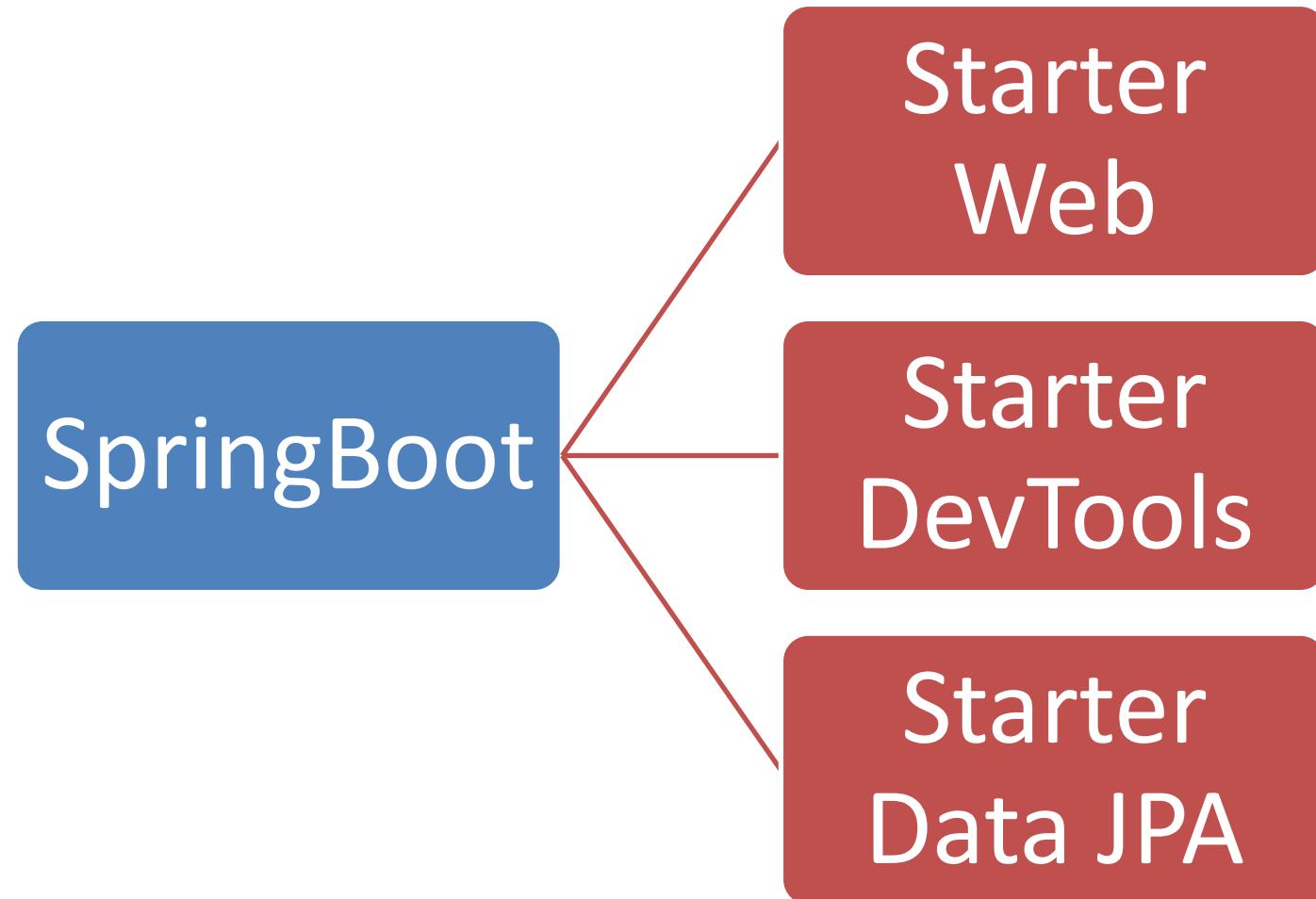


**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**  
Dirección General de Cómputo y de Tecnologías de información y Comunicación  
Dirección de Docencia en TIC



Educación  
Continua  
1971 - 2021

# Spring Data



# Spring Data Configuración

- Spring Initializr
  - Es una aplicación web que puede generar una estructura de proyecto Spring Boot
  - <https://start.spring.io/>



# Spring Data Configuración

The screenshot shows the Spring Initializr web application at [start.spring.io](https://start.spring.io). The interface is dark-themed.

**Project:**

- Maven Project
- Gradle Project

**Language:**

- Java
- Kotlin
- Groovy

**Spring Boot:**

- 2.7.0 (SNAPSHOT)
- 2.6.3 (SNAPSHOT)
- 2.6.2
- 2.5.9 (SNAPSHOT)
- 2.5.8

**Project Metadata:**

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot

**Dependencies:** No dependency selected. [ADD DEPENDENCIES...](#)

**Actions:**

- [GENERATE](#) ⌘ + ↩
- [EXPLORE](#) CTRL + SPACE
- [SHARE...](#)

# Spring Data Configuración

The screenshot shows the Spring Initializr website at [start.spring.io](https://start.spring.io). The page has a dark theme with green highlights. On the left, there's a sidebar with icons for GitHub and Twitter. The main area is titled "spring initializr". It includes sections for "Project" (Maven Project selected), "Language" (Java selected), "Spring Boot" (2.6.2 selected), "Project Metadata" (Group: mx.unam.dgtic, Artifact: datos1, Name: datos1, Description: Proyecto de Practica M7), and "Dependencies" (Spring Web, Spring Boot DevTools, Spring Data JPA, MySQL Driver). At the bottom are buttons for "GENERATE" (⌘ + ↩), "EXPLORE" (CTRL + SPACE), and "SHARE...".

start.spring.io

Aplicaciones LanguageManual... GitLab.org / GitLa... Icon component -... Nuevo Ingreso, Pr... https://zafiro.dgap... Elecciones estatal... map Generador de dia... Lista de lectura

Actualizar

spring initializr

Project

Maven Project

Gradle Project

Language

Java

Kotlin

Groovy

Spring Boot

2.7.0 (SNAPSHOT)

2.6.3 (SNAPSHOT)

2.6.2

2.5.9 (SNAPSHOT)

2.5.8

Project Metadata

Group: mx.unam.dgtic

Artifact: datos1

Name: datos1

Description: Proyecto de Practica M7

Dependencies

ADD DEPENDENCIES... ⌘ + B

Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS

Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Data JPA SQL

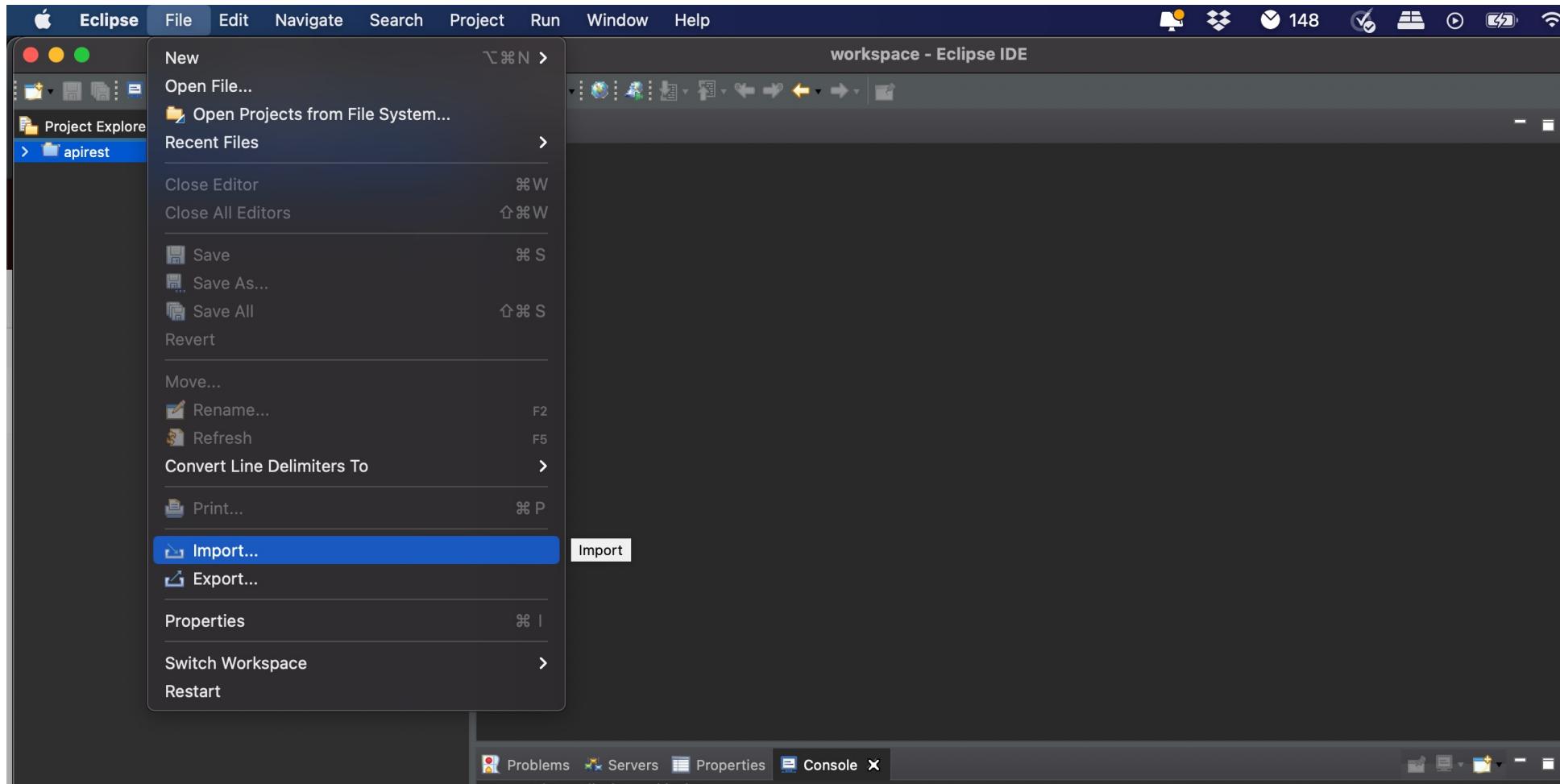
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

MySQL Driver SQL

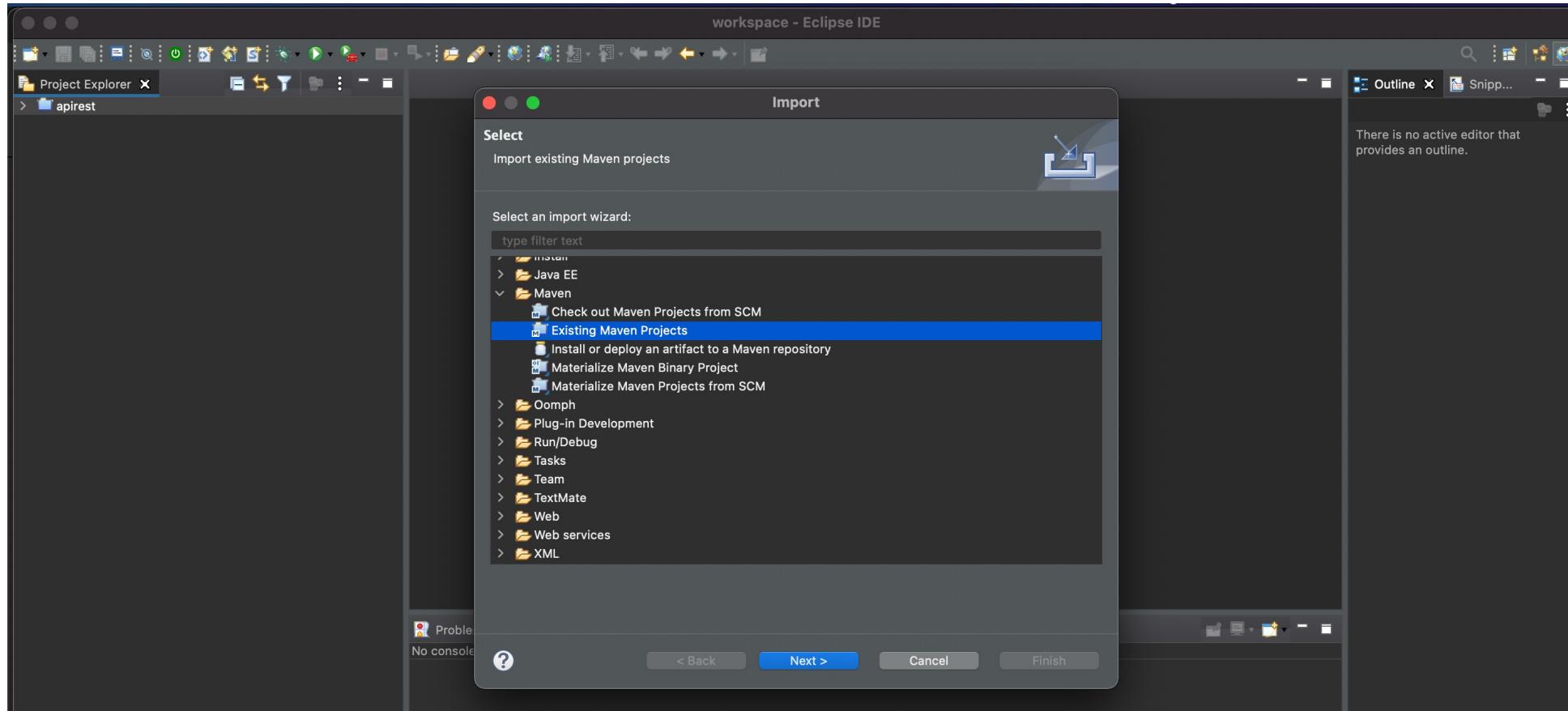
MySQL JDBC and R2DBC driver.

GENERATE ⌘ + ↩ EXPLORE CTRL + SPACE SHARE...

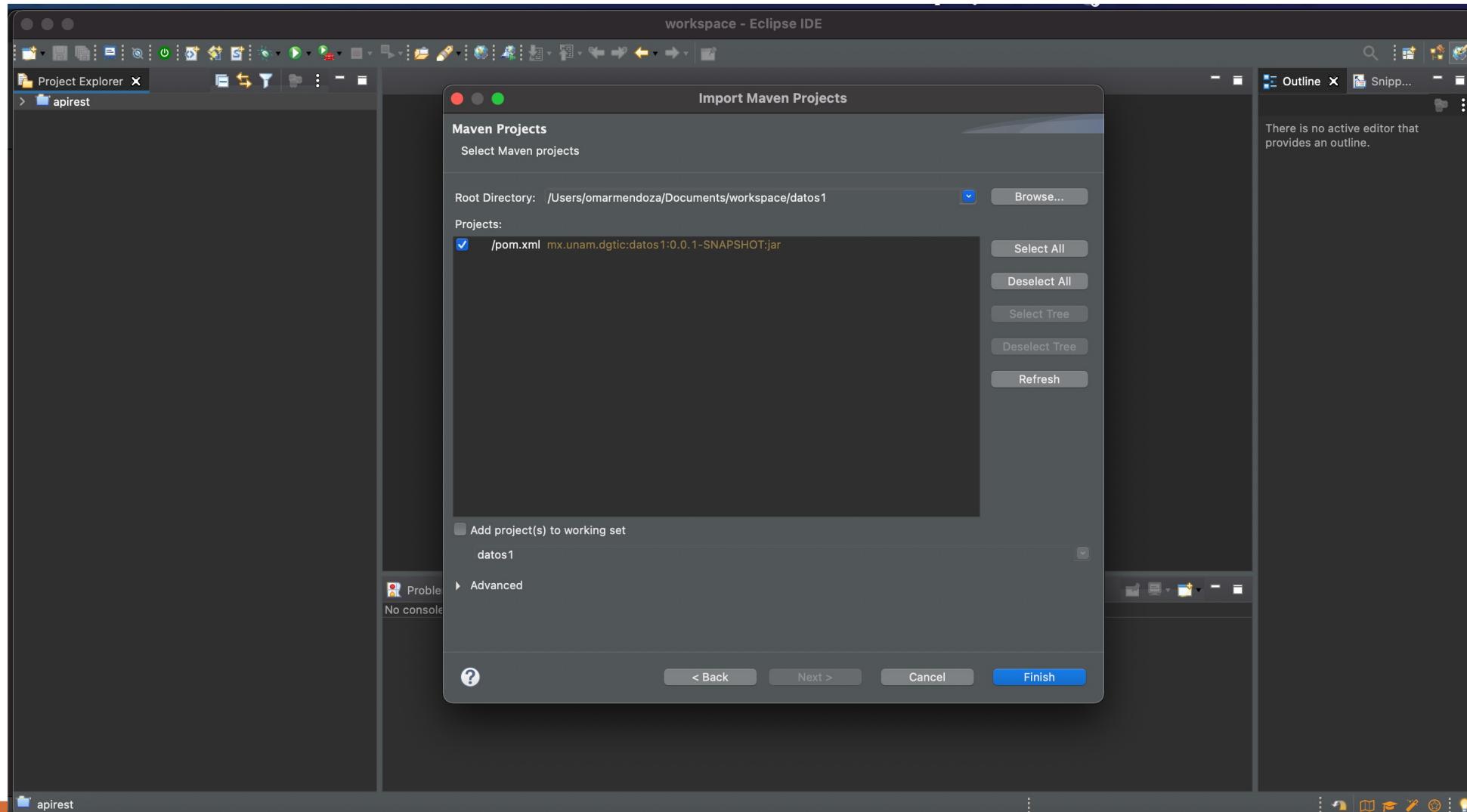
# Spring Data Configuración



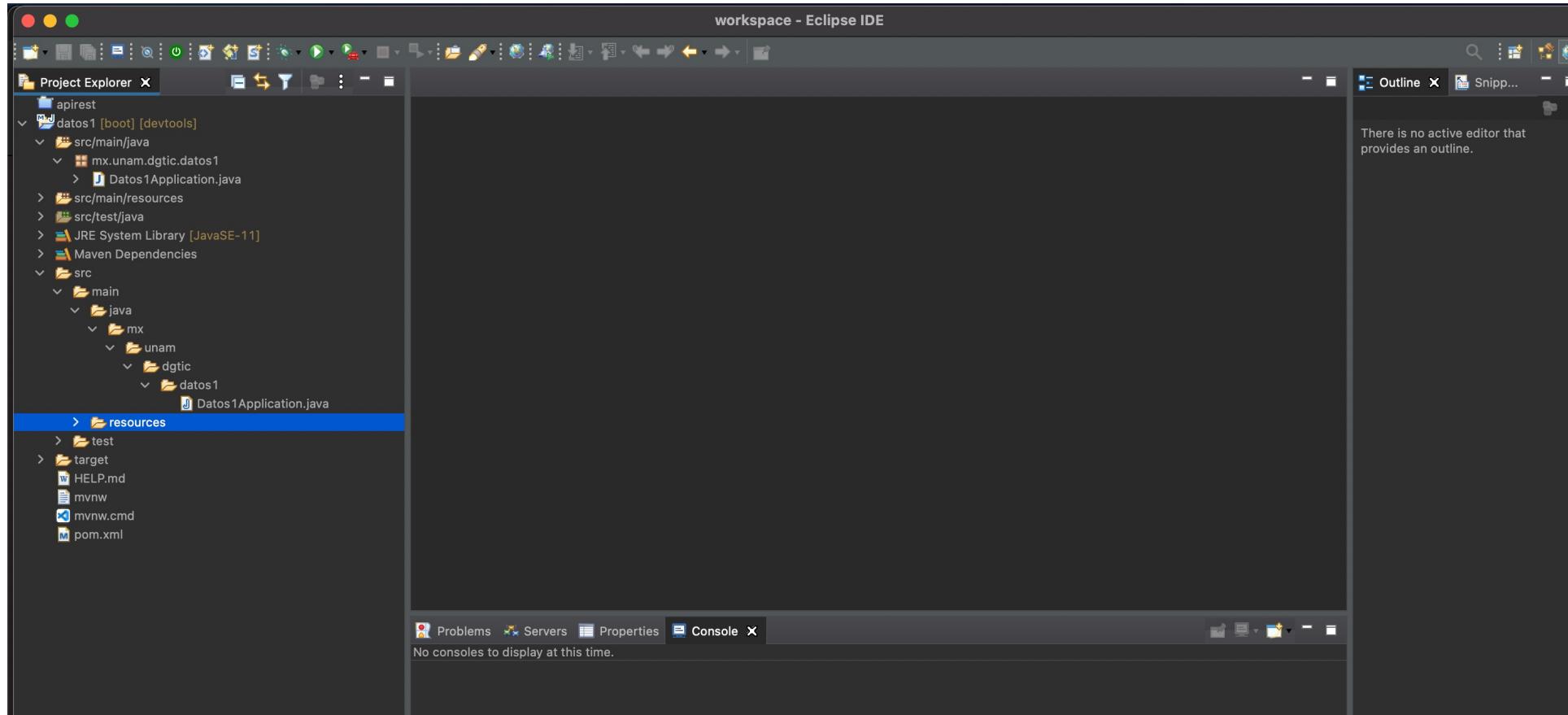
# Spring Data Configuración



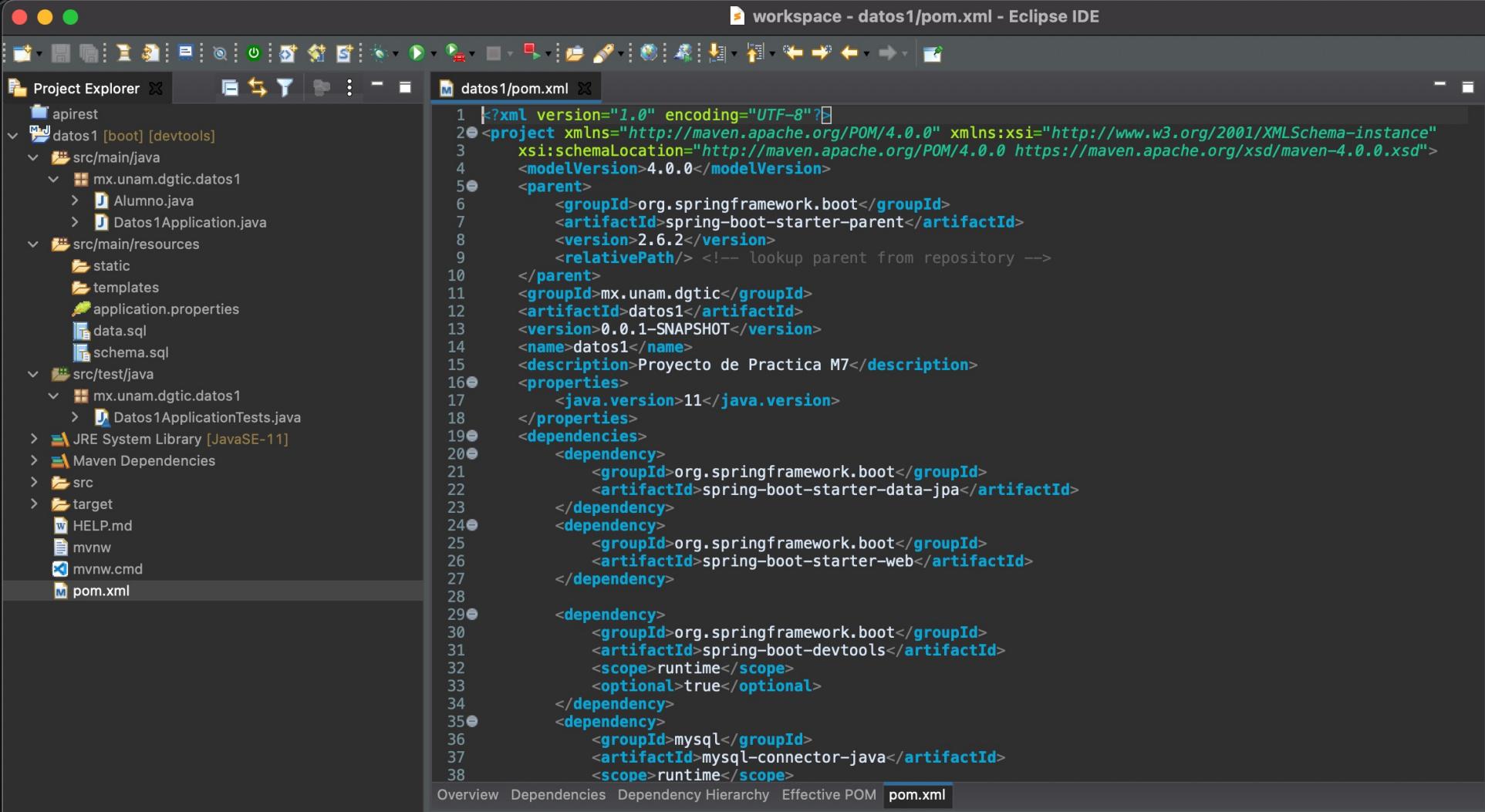
# Spring Data Configuración



# Spring Data Configuración



# pom.xml

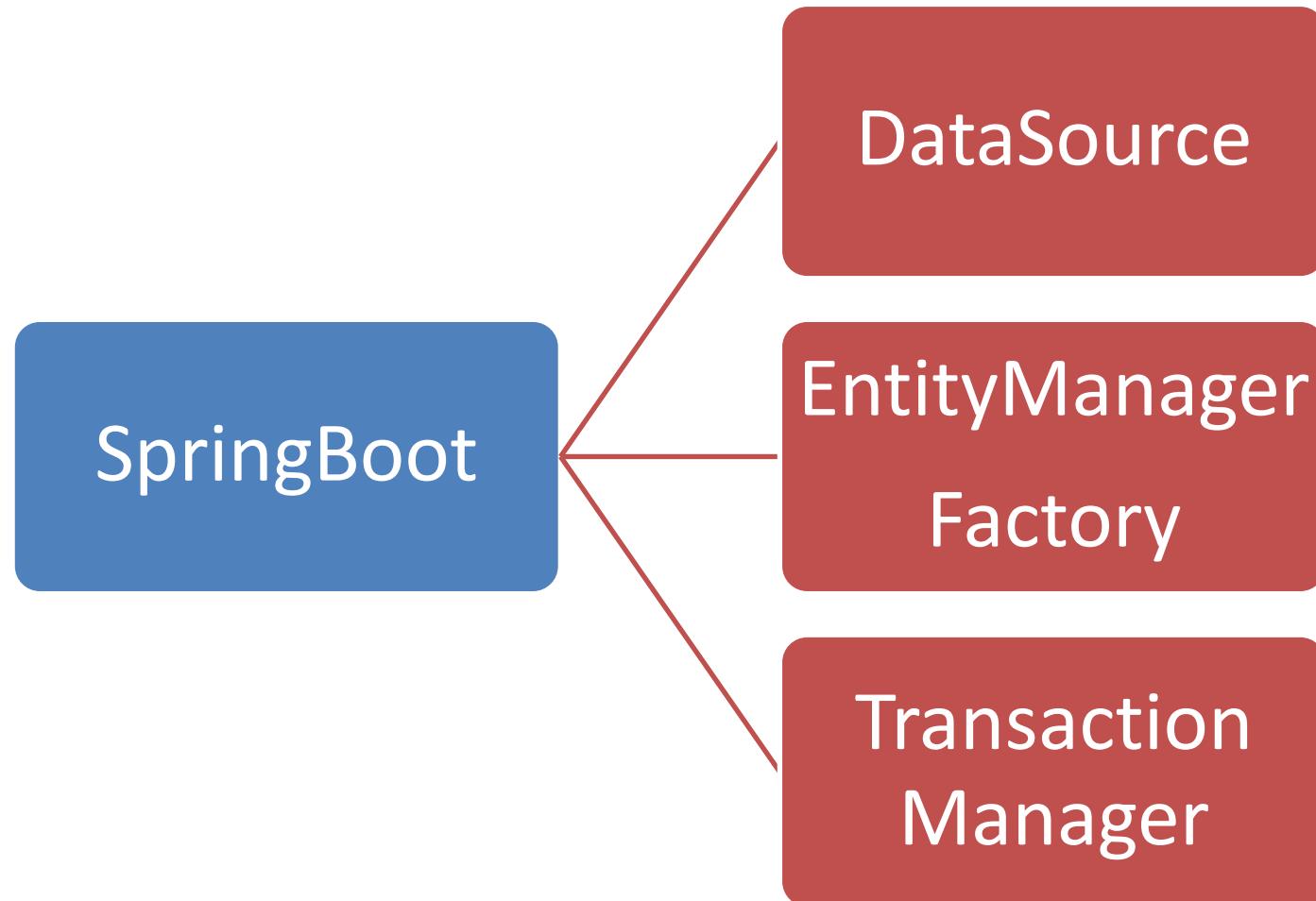


The screenshot shows the Eclipse IDE interface with the title "workspace - datos1/pom.xml - Eclipse IDE". The left side features the "Project Explorer" view, which displays the project structure for "datos1". The "src/main/java" directory contains "mx.unam.dgtic.datos1" package with "Alumno.java" and "Datos1Application.java" files. The "src/main/resources" directory contains "static", "templates", "application.properties", "data.sql", and "schema.sql" files. The "src/test/java" directory contains "mx.unam.dgtic.datos1" package with "Datos1ApplicationTests.java". The "target" directory contains "HELP.md", "mvnw", "mvnw.cmd", and "pom.xml". The right side shows the content of the "pom.xml" file, which defines a Maven project with various configurations like parent dependencies, Java version, and MySQL connector dependencies.

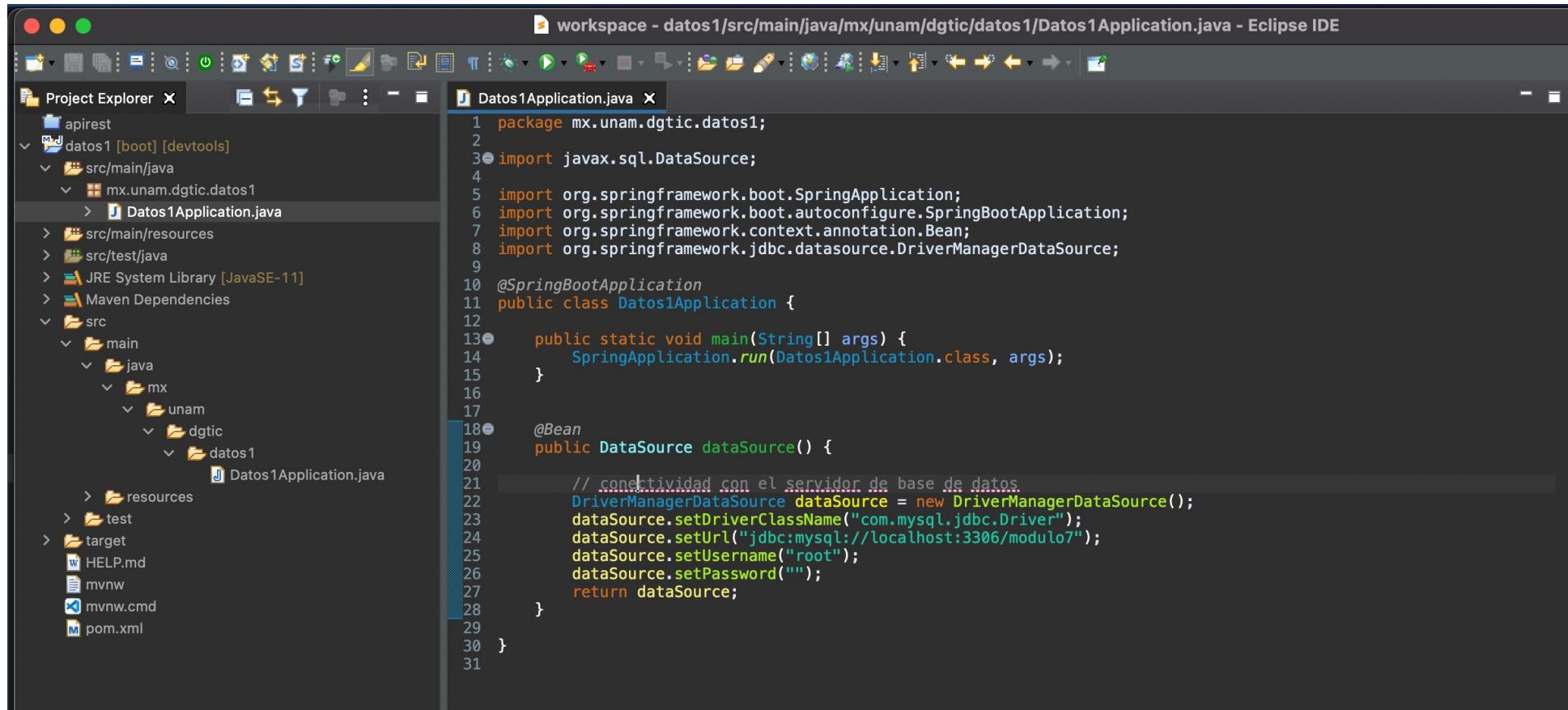
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.2</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>mx.unam.dgtic</groupId>
  <artifactId>datos1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>datos1</name>
  <description>Proyecto de Practica M7</description>
  <properties>
    <java.version>11</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <scope>runtime</scope>
    </dependency>
  </dependencies>

```

# Spring Data



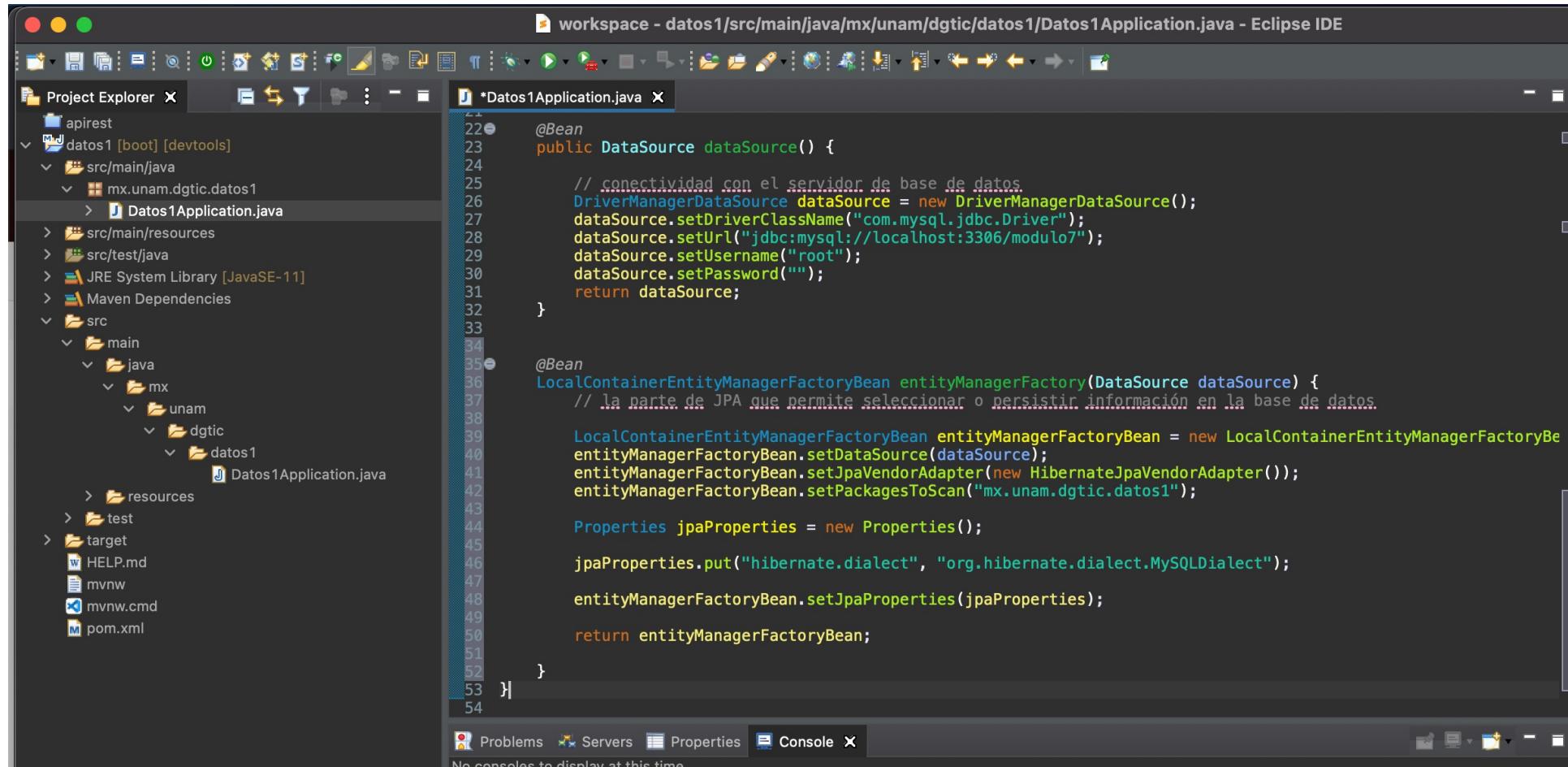
# Agregar DataSource



The screenshot shows the Eclipse IDE interface with the title bar "workspace - datos1/src/main/java/mx/unam/dgtic/datos1/Datos1Application.java - Eclipse IDE". The left side features the "Project Explorer" view, which displays the project structure:

- apirest
- datos1 [boot] [devtools]
  - src/main/java
    - mx.unam.dgtic.datos1
      - Datos1Application.java
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
    - main
      - java
        - mx
          - unam
            - dgtic
              - datos1
                - Datos1Application.java
    - target
    - HELP.md
    - mvnw
    - mvnw.cmd
    - pom.xml

# Agregar EntityManagerFactory

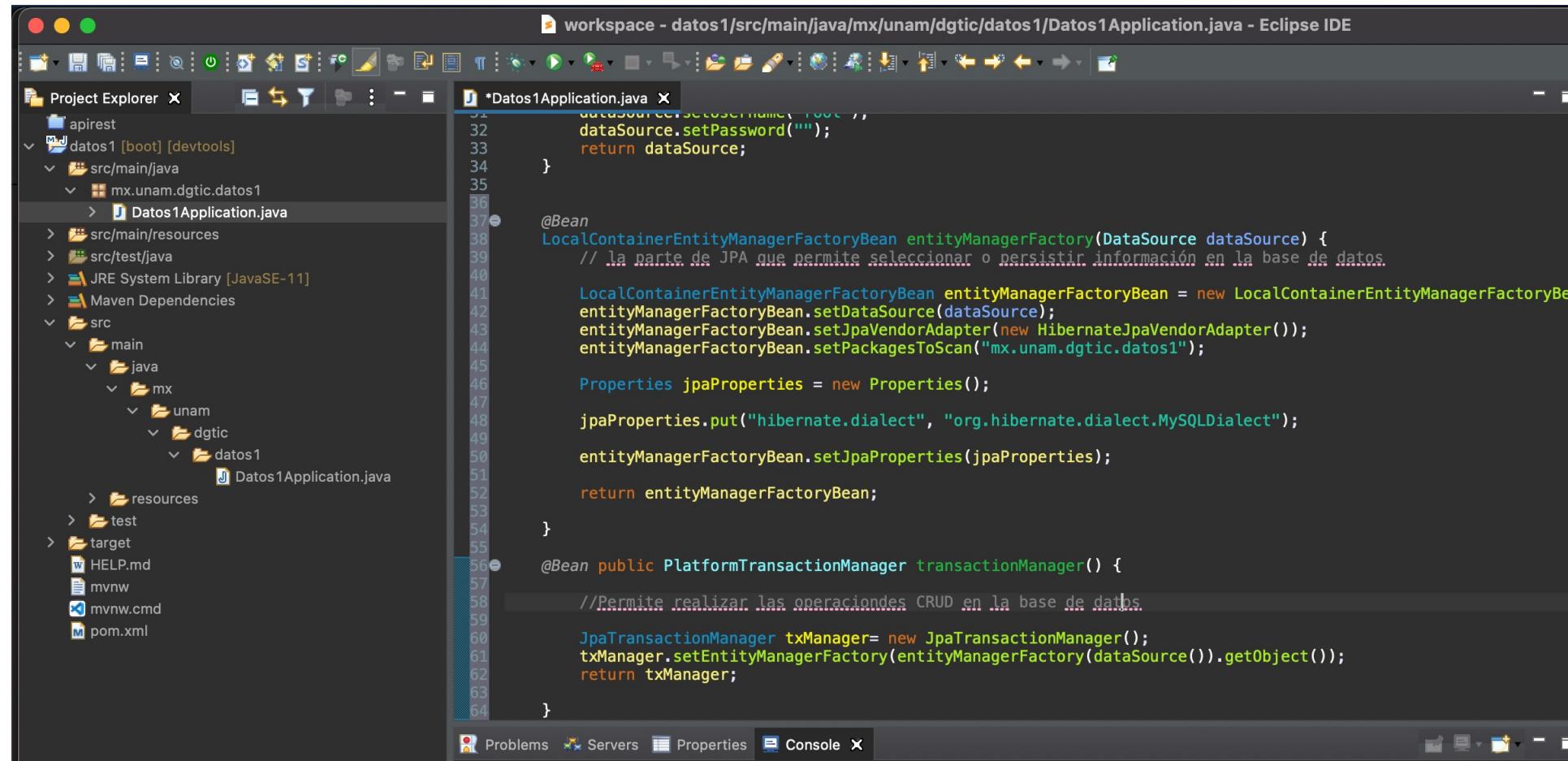


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - datos1/src/main/java/mx/unam/dgtic/datos1/Datos1Application.java - Eclipse IDE
- Project Explorer:** Shows the project structure:
  - apirest
  - datos1 [boot] [devtools]
    - src/main/java
      - mx.unam.dgtic.datos1
        - Datos1Application.java
    - src/main/resources
    - src/test/java
    - JRE System Library [JavaSE-11]
    - Maven Dependencies
    - src
      - main
        - java
          - mx
            - unam
              - dgtic
                - datos1
                  - Datos1Application.java
        - resources
        - test
      - target
      - HELP.md
      - mvnw
      - mvnw.cmd
      - pom.xml
- Code Editor:** Displays the Java code for Datos1Application.java:

```
22  * @Bean
23  public DataSource dataSource() {
24
25      // conectividad con el servidor de base de datos
26      DriverManagerDataSource dataSource = new DriverManagerDataSource();
27      dataSource.setDriverClassName("com.mysql.jdbc.Driver");
28      dataSource.setUrl("jdbc:mysql://localhost:3306/modulo7");
29      dataSource.setUsername("root");
30      dataSource.setPassword("");
31
32      return dataSource;
33  }
34
35  * @Bean
36  LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource dataSource) {
37      // la parte de JPA que permite seleccionar o persistir información en la base de datos
38
39      LocalContainerEntityManagerFactoryBean entityManagerBean = new LocalContainerEntityManagerFactoryBean();
40      entityManagerBean.setDataSource(dataSource);
41      entityManagerBean.setJpaVendorAdapter(new HibernateJpaVendorAdapter());
42      entityManagerBean.setPackagesToScan("mx.unam.dgtic.datos1");
43
44      Properties jpaProperties = new Properties();
45
46      jpaProperties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
47
48      entityManagerBean.setJpaProperties(jpaProperties);
49
50      return entityManagerBean;
51  }
52
53  }
54 }
```
- Bottom Bar:** Shows tabs for Problems, Servers, Properties, and Console. The Console tab is selected, displaying the message: "No consoles to display at this time."

# Agregar TransactionManager



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

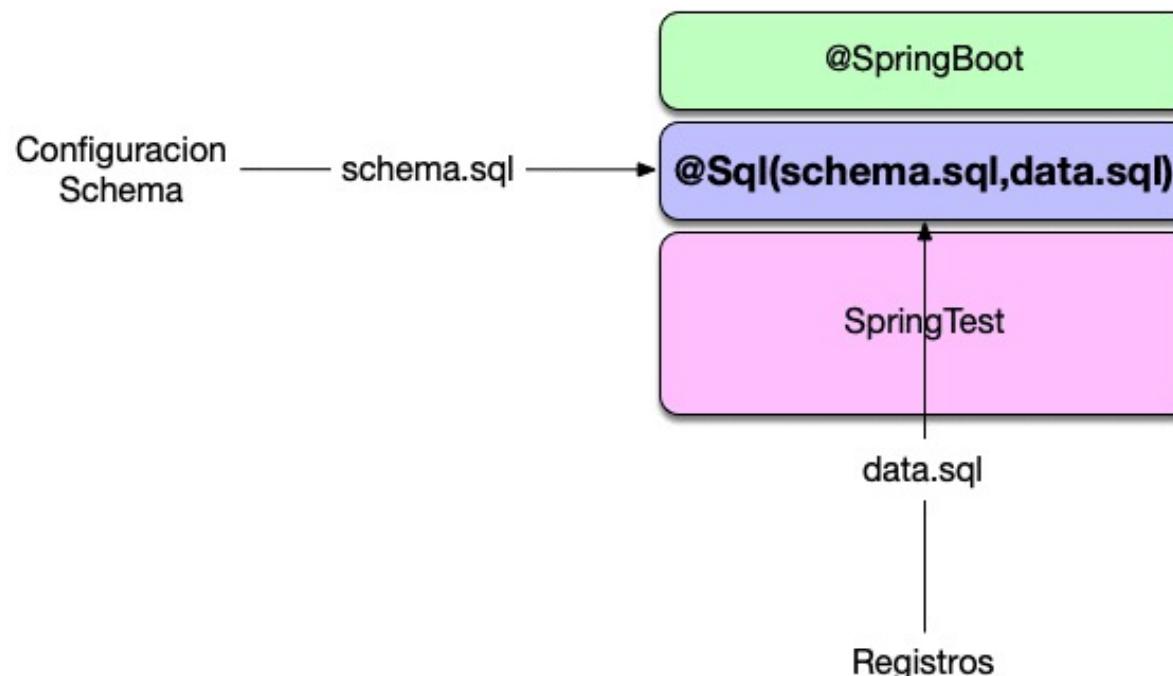
- Project Explorer:** Shows the project structure under "datos1 [boot] [devtools]". The "src/main/java" folder contains "mx.unam.dgtic.datos1" which has "Datos1Application.java". Other visible files include "src/main/resources", "src/test/java", "JRE System Library [JavaSE-11]", "Maven Dependencies", "src/main/java/mx/unam/dgtic/datos1/Datos1Application.java", "src/main/resources", "src/test/java", "target/HELP.md", "mvnw", "mvnw.cmd", and "pom.xml".
- Code Editor:** Displays the content of "Datos1Application.java". The code defines two bean methods: "entityManagerFactory" and "transactionManager".

```
31     dataSource.setUrl("jdbc:mysql://localhost:3306/tx");
32     dataSource.setPassword("");
33     return dataSource;
34 }
35
36
37 @Bean
38 LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource dataSource) {
39     // la parte de JPA que permite seleccionar o persistir información en la base de datos.
40     LocalContainerEntityManagerFactoryBean entityManagerFactory = new LocalContainerEntityManagerFactoryBean();
41     entityManagerFactory.setDataSource(dataSource);
42     entityManagerFactory.setJpaVendorAdapter(new HibernateJpaVendorAdapter());
43     entityManagerFactory.setPackagesToScan("mx.unam.dgtic.datos1");
44
45     Properties jpaProperties = new Properties();
46
47     jpaProperties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
48
49     entityManagerFactory.setJpaProperties(jpaProperties);
50
51     return entityManagerFactory;
52 }
53
54
55 @Bean public PlatformTransactionManager transactionManager() {
56     //Permite realizar las operaciones CRUD en la base de datos.
57
58     JpaTransactionManager txManager= new JpaTransactionManager();
59     txManager.setEntityManagerFactory(entityManagerFactory(dataSource()).getgetObject());
60     return txManager;
61 }
62
63
64 }
```
- Bottom Bar:** Shows tabs for "Problems", "Servers", "Properties", and "Console".



# @Sql

- A nivel de pruebas unitarias se dispone de la anotación `@Sql` que permite definir como se puede cargar de forma automática información en la base de datos.



@Sql

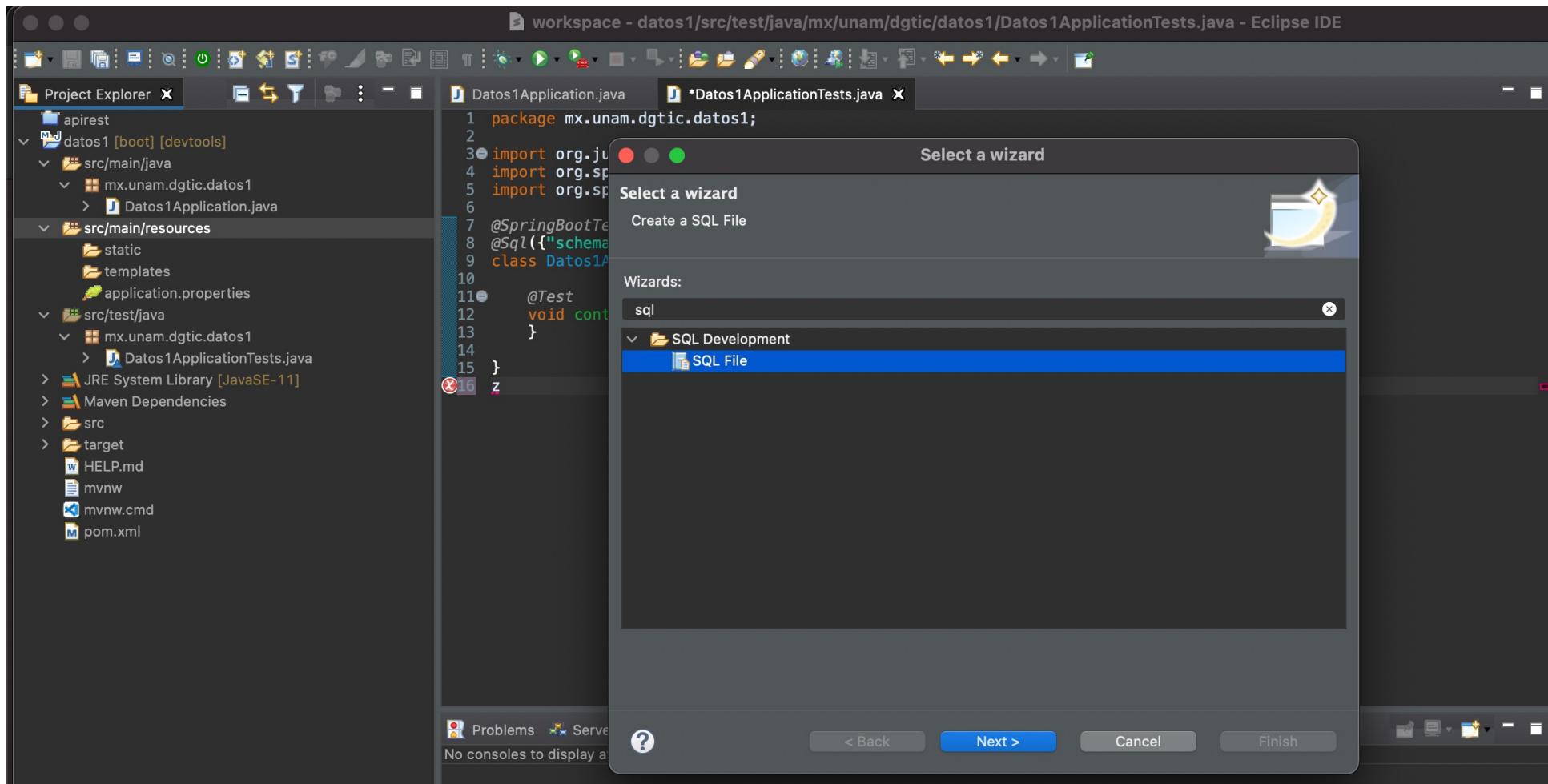
@Sql(schema.sql, data.sql)

Create

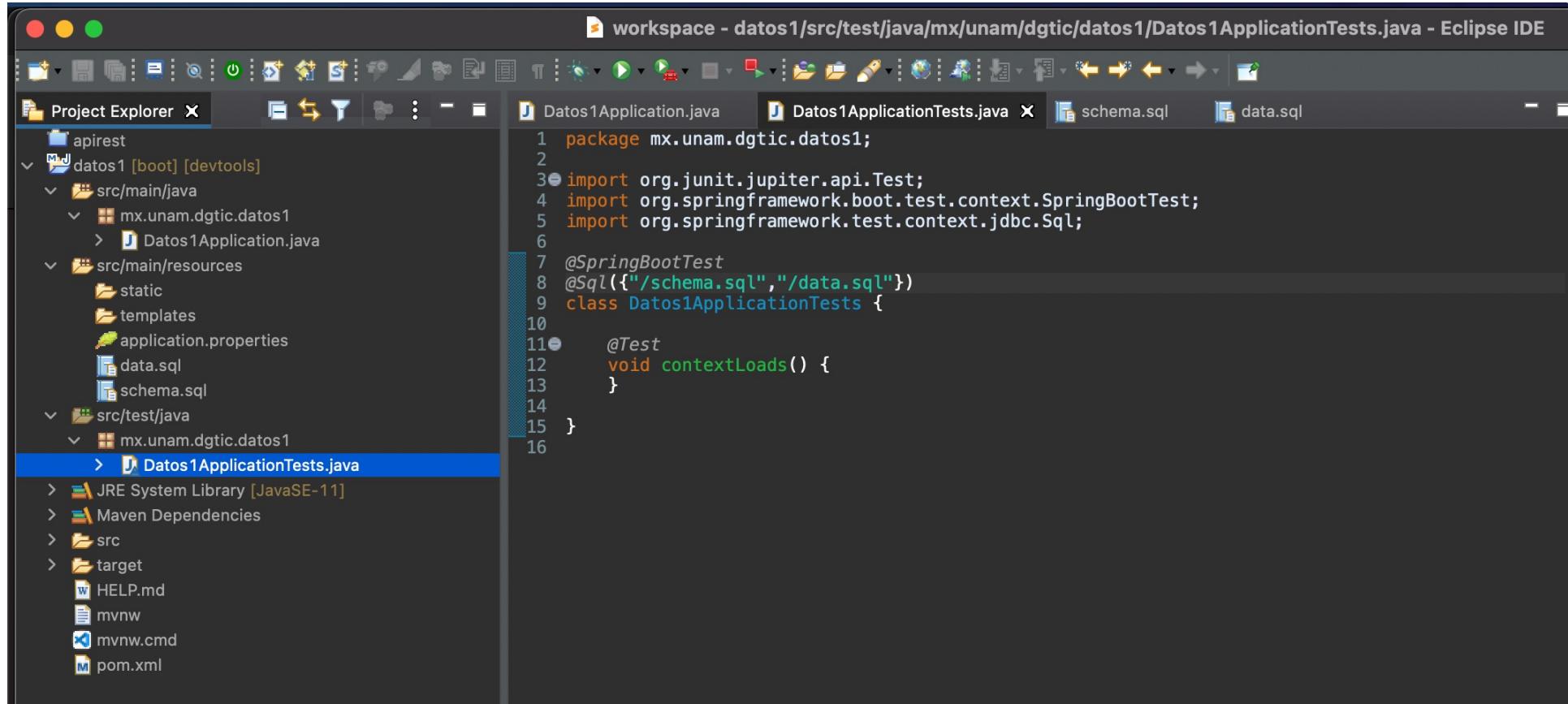
insert



# Agregar archivos sql



# Definir datos para prueba

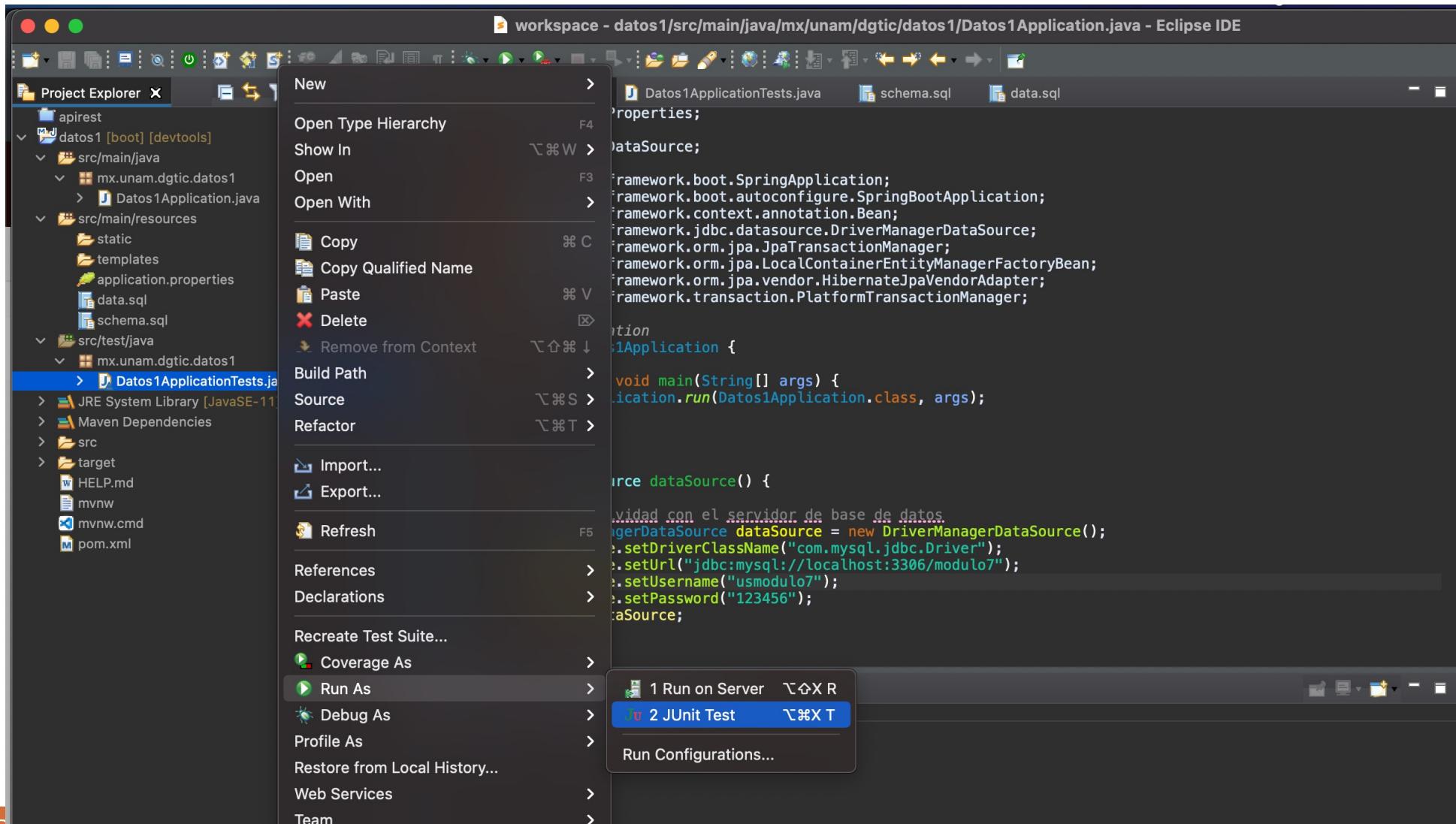


The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** workspace - datos1/src/test/java/mx/unam/dgtic/datos1/Datos1ApplicationTests.java - Eclipse IDE
- Project Explorer:** Shows the project structure:
  - apirest
  - datos1 [boot] [devtools]
    - src/main/java
      - mx.unam.dgtic.datos1
        - Datos1Application.java
    - src/main/resources
      - static
      - templates
      - application.properties
      - data.sql
      - schema.sql
    - src/test/java
      - mx.unam.dgtic.datos1
        - Datos1ApplicationTests.java
  - JRE System Library [JavaSE-11]
  - Maven Dependencies
  - src
  - target
  - HELP.md
  - mvnw
  - mvnw.cmd
  - pom.xml
- Editor Area:** Displays the code for `Datos1ApplicationTests.java`. The code is annotated with JUnit Jupiter and Spring Boot annotations to run database tests.

```
1 package mx.unam.dgtic.datos1;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.boot.test.context.SpringBootTest;
5 import org.springframework.test.context.jdbc.Sql;
6
7 @SpringBootTest
8 @Sql({"schema.sql", "data.sql"})
9 class Datos1ApplicationTests {
10
11     @Test
12     void contextLoads() {
13     }
14
15 }
16
```

# Ejecutar Test



# Ejecutar Test

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

- Project Explorer:** Displays the project structure for "datos1". It includes the "src/main/java" and "src/test/java" packages, which contain "mx.unam.dgtic.datos1" and "Datos1ApplicationTests.java" respectively.
- Editor:** Shows the code for "Datos1ApplicationTests.java". The code defines a test class named "Datos1ApplicationTests" that extends "SpringBootTest" and "Sql". It contains a single test method named "contextLoads()".

```
1 package mx.unam.dgtic.datos1;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.boot.test.context.SpringBootTest;
5 import org.springframework.test.context.jdbc.Sql;
6
7 @SpringBootTest
8 @Sql({"/schema.sql", "/data.sql"})
9 class Datos1ApplicationTests {
10
11     @Test
12     void contextLoads() {
13     }
14
15 }
```
- Outline View:** Shows the test class "Datos1ApplicationTests" and its method "contextLoads()".
- JUnit View:** Displays the test results:
  - Finished after 6.722 seconds
  - Runs: 1/1 Errors: 0 Failures: 0
  - A green progress bar indicates success.
- Failure Trace:** An empty panel.

# Contacto

Dr. Omar Mendoza González

[omarmendoza564@aragon.unam.mx](mailto:omarmendoza564@aragon.unam.mx)

