



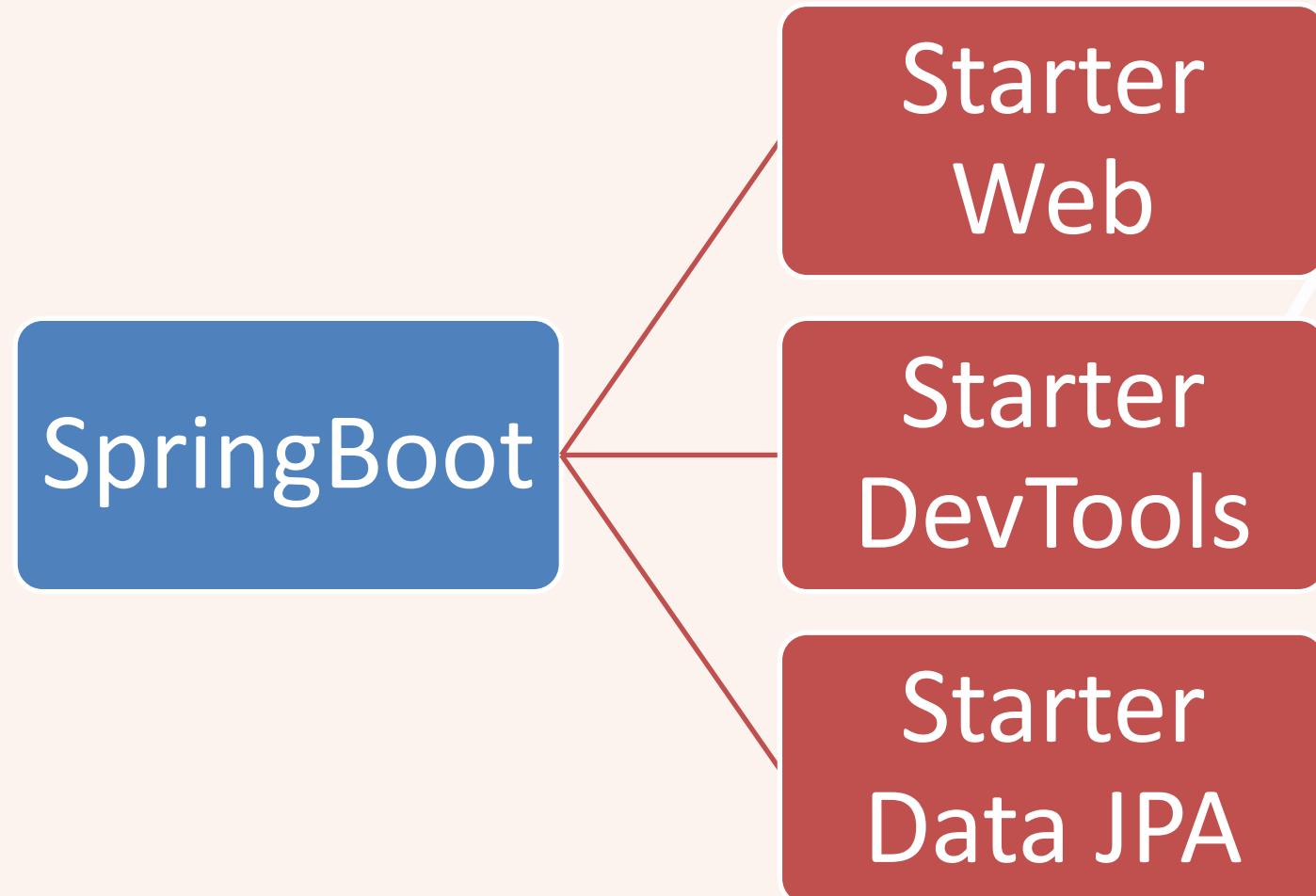
DIPLOMADO  
**Desarrollo de sistemas con  
tecnología Java**

**Módulo 8**  
Persistencia con Spring Data

*Dr. Omar Mendoza González*

*omarmendoza564@aragon.unam.mx*

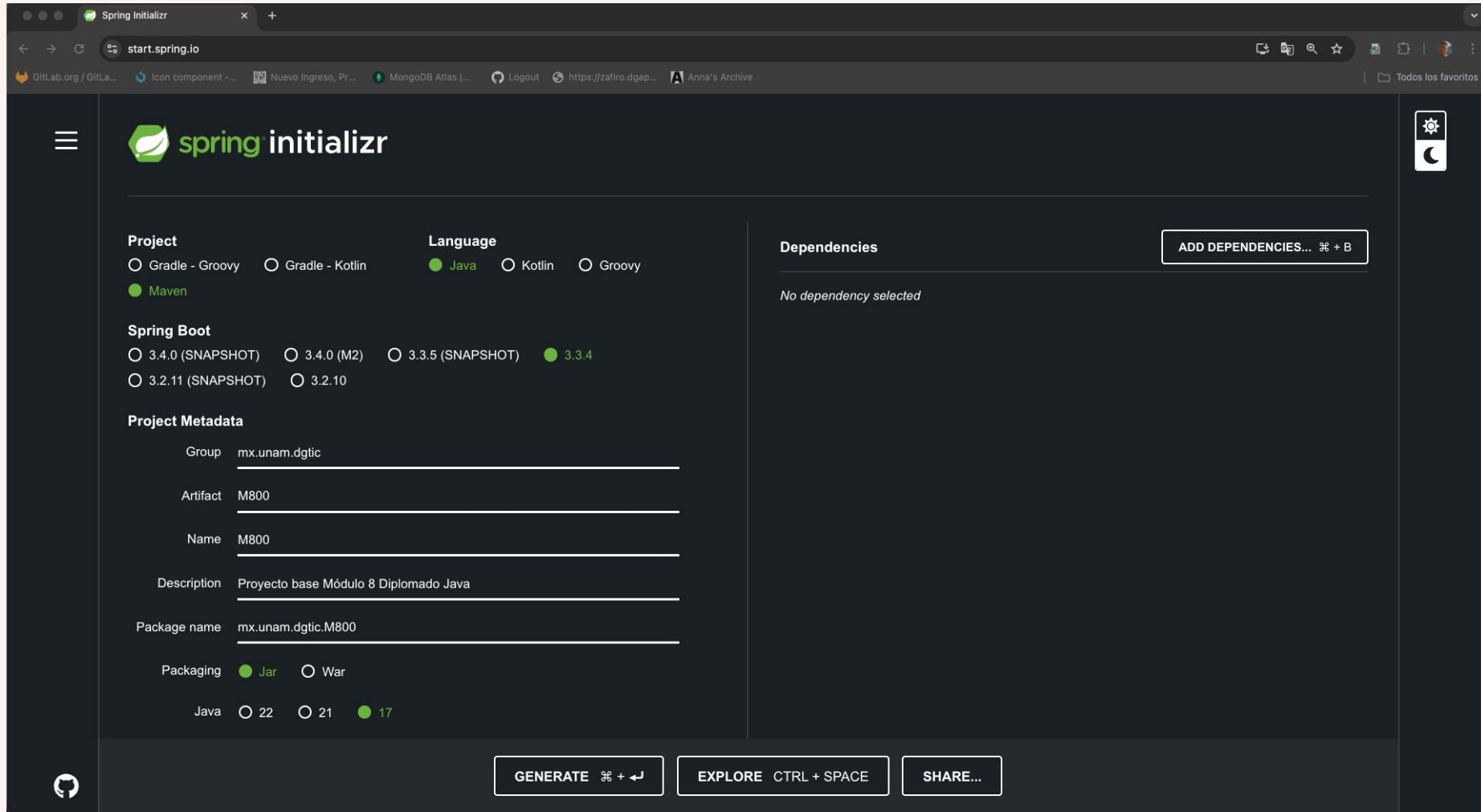
# Spring Data



# 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 interface at [start.spring.io](https://start.spring.io). The configuration is set for a Maven-based Java Spring Boot project (version 3.3.4). The project metadata includes Group: mx.unam.dgtic, Artifact: M800, Name: M800, Description: Proyecto base Módulo 8 Diplomado Java, Package name: mx.unam.dgtic.M800, and Packaging: Jar. The Java version selected is 17. There are no dependencies added yet.

**Project**

- Gradle - Groovy
- Gradle - Kotlin
- Maven

**Language**

- Java
- Kotlin
- Groovy

**Spring Boot**

- 3.4.0 (SNAPSHOT)
- 3.4.0 (M2)
- 3.3.5 (SNAPSHOT)
- 3.3.4

- 3.2.11 (SNAPSHOT)
- 3.2.10

**Project Metadata**

Group:

Artifact:

Name:

Description:

Package name:

Packaging:  Jar    War

Java:  22    21    17

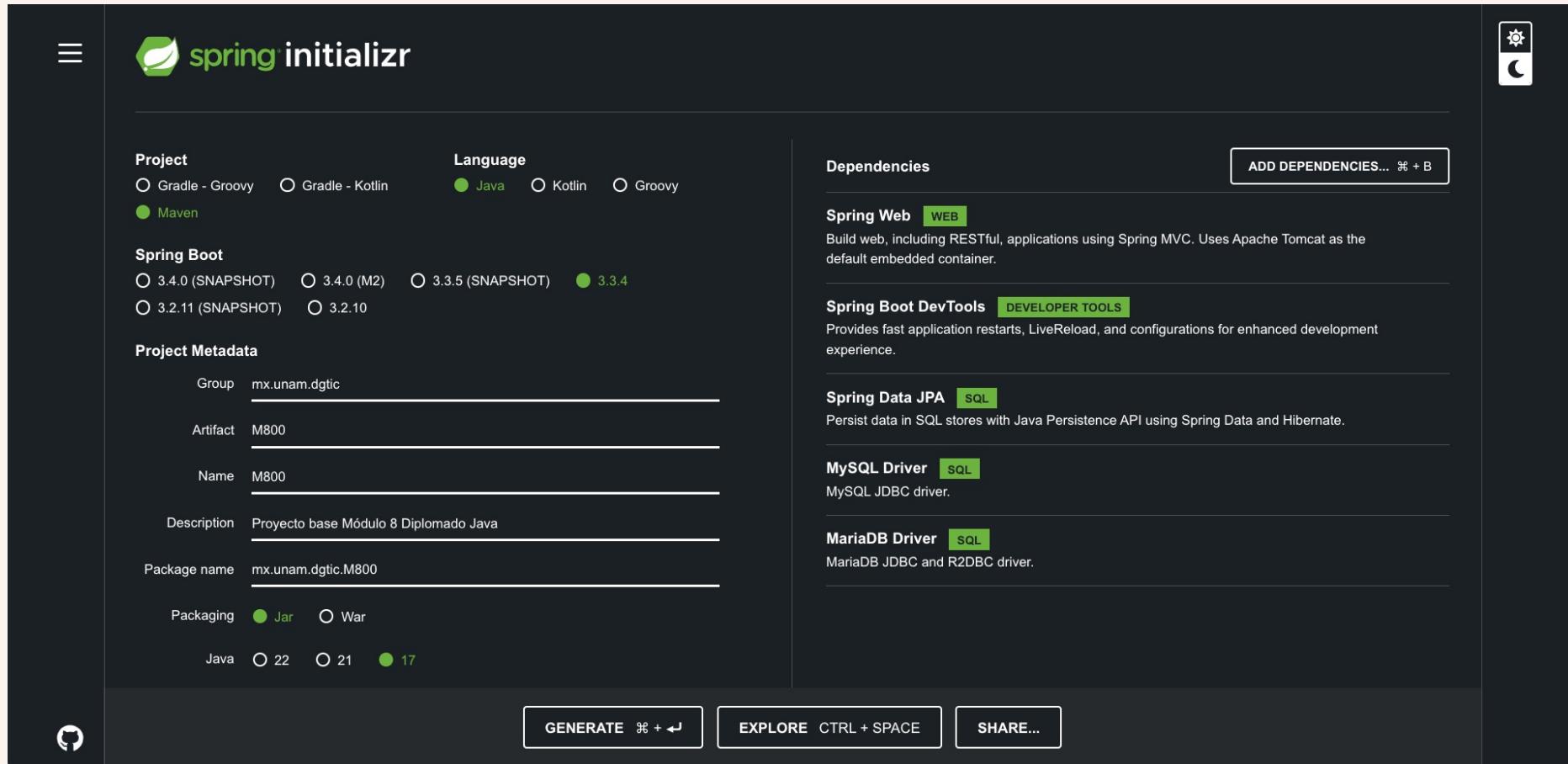
**Dependencies**

No dependency selected

**Buttons**

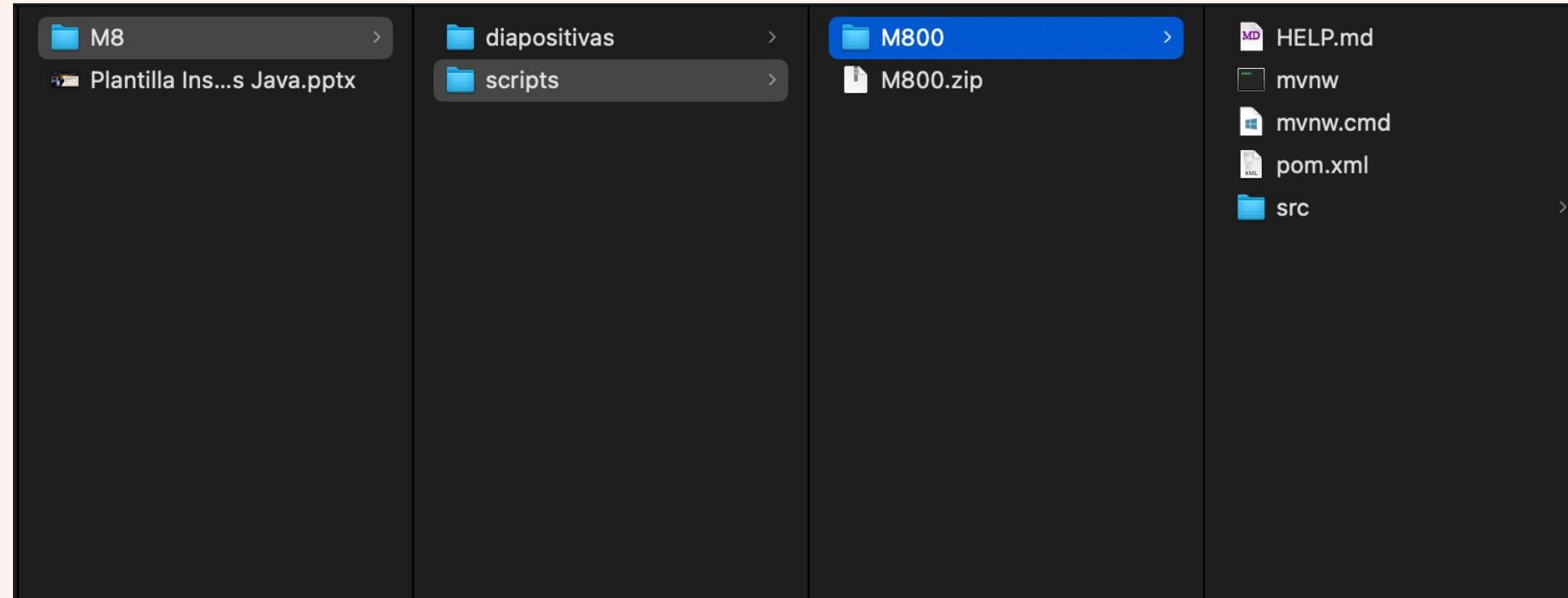
- ADD DEPENDENCIES... ⌘ + B
- GENERATE ⌘ + ↵
- EXPLORE CTRL + SPACE
- SHARE...

# Spring Data Configuración

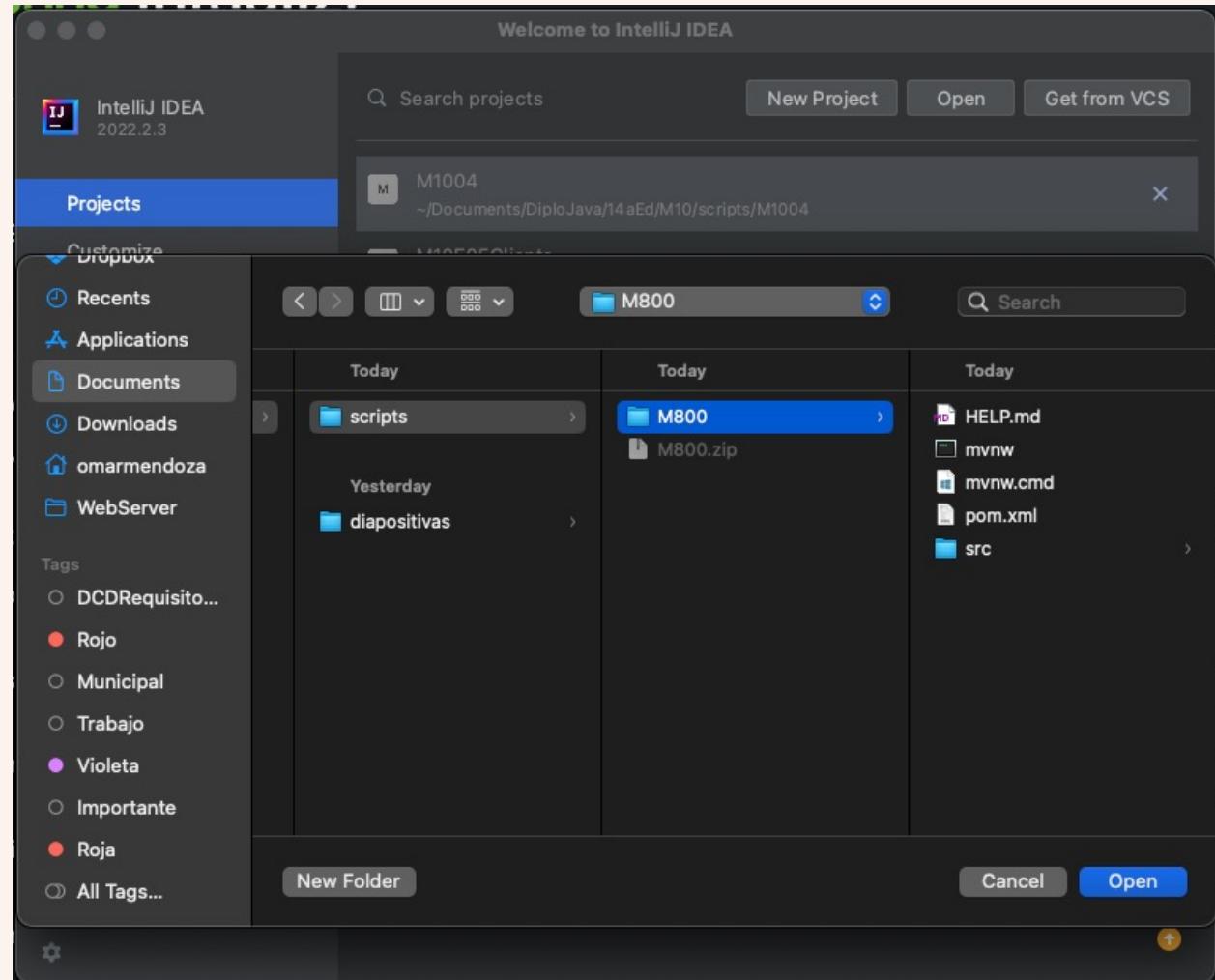


The screenshot shows the Spring Initializr web application interface. The left sidebar has a three-line menu icon. The main area starts with the "spring initializr" logo. Below it, under "Project", there are radio buttons for Gradle - Groovy, Gradle - Kotlin, and Maven (which is selected). Under "Language", there are radio buttons for Java (selected), Kotlin, and Groovy. Under "Spring Boot", there are radio buttons for 3.4.0 (SNAPSHOT), 3.4.0 (M2), 3.3.5 (SNAPSHOT) (selected), 3.3.4, 3.2.11 (SNAPSHOT), and 3.2.10. The "Project Metadata" section contains fields for Group (mx.unam.dgtic), Artifact (M800), Name (M800), Description (Proyecto base Módulo 8 Diplomado Java), Package name (mx.unam.dgtic.M800), and Packaging (Jar, selected over War). Below these, Java version options (22, 21, 17) are shown. On the right, the "Dependencies" section lists several optional add-ons: "Spring Web" (selected, WEB), "Spring Boot DevTools" (selected, DEVELOPER TOOLS), "Spring Data JPA" (SQL), "MySQL Driver" (SQL), and "MariaDB Driver" (SQL). At the bottom, there are "GENERATE" and "EXPLORE" buttons, and a "SHARE..." button.

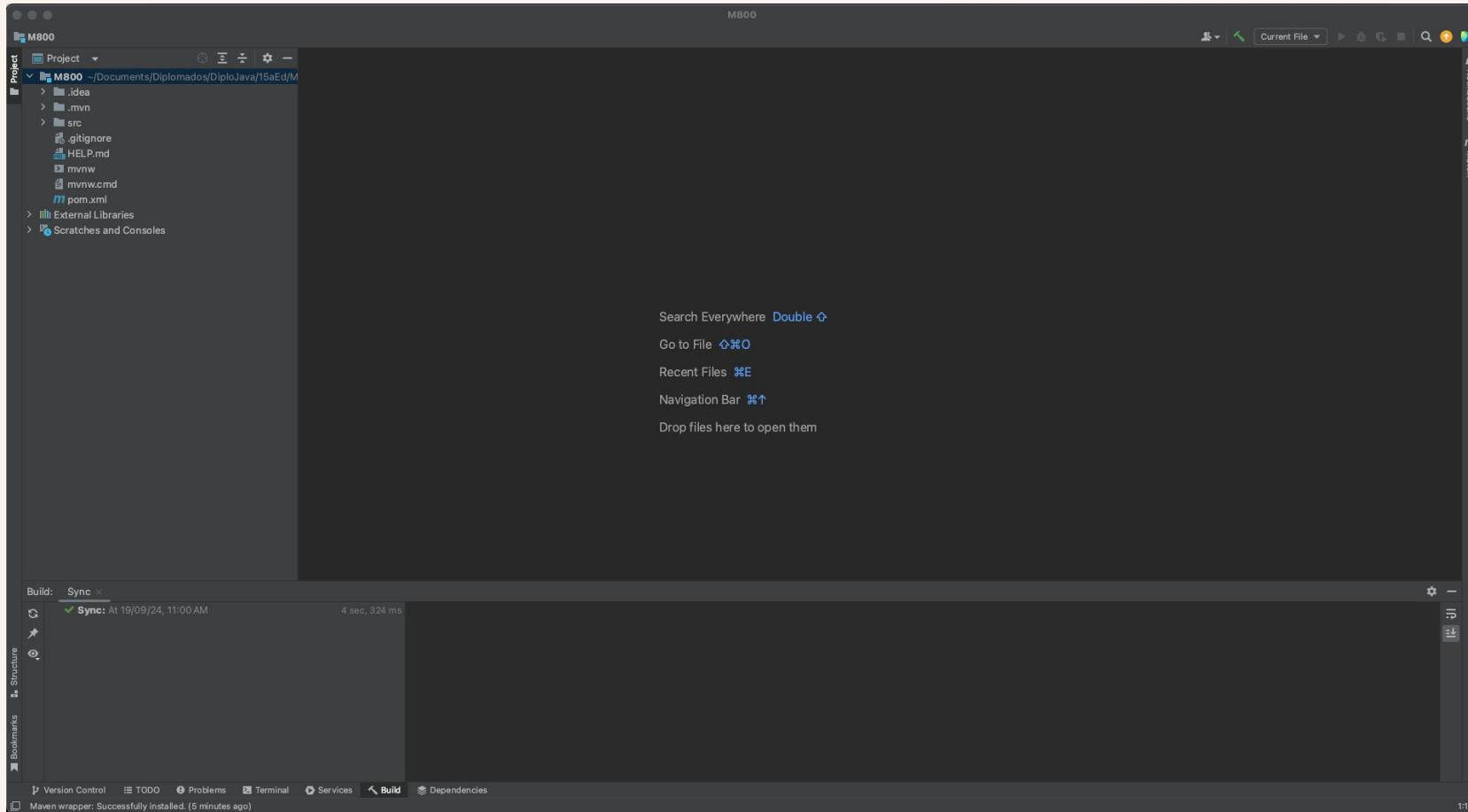
# Spring Data Configuración



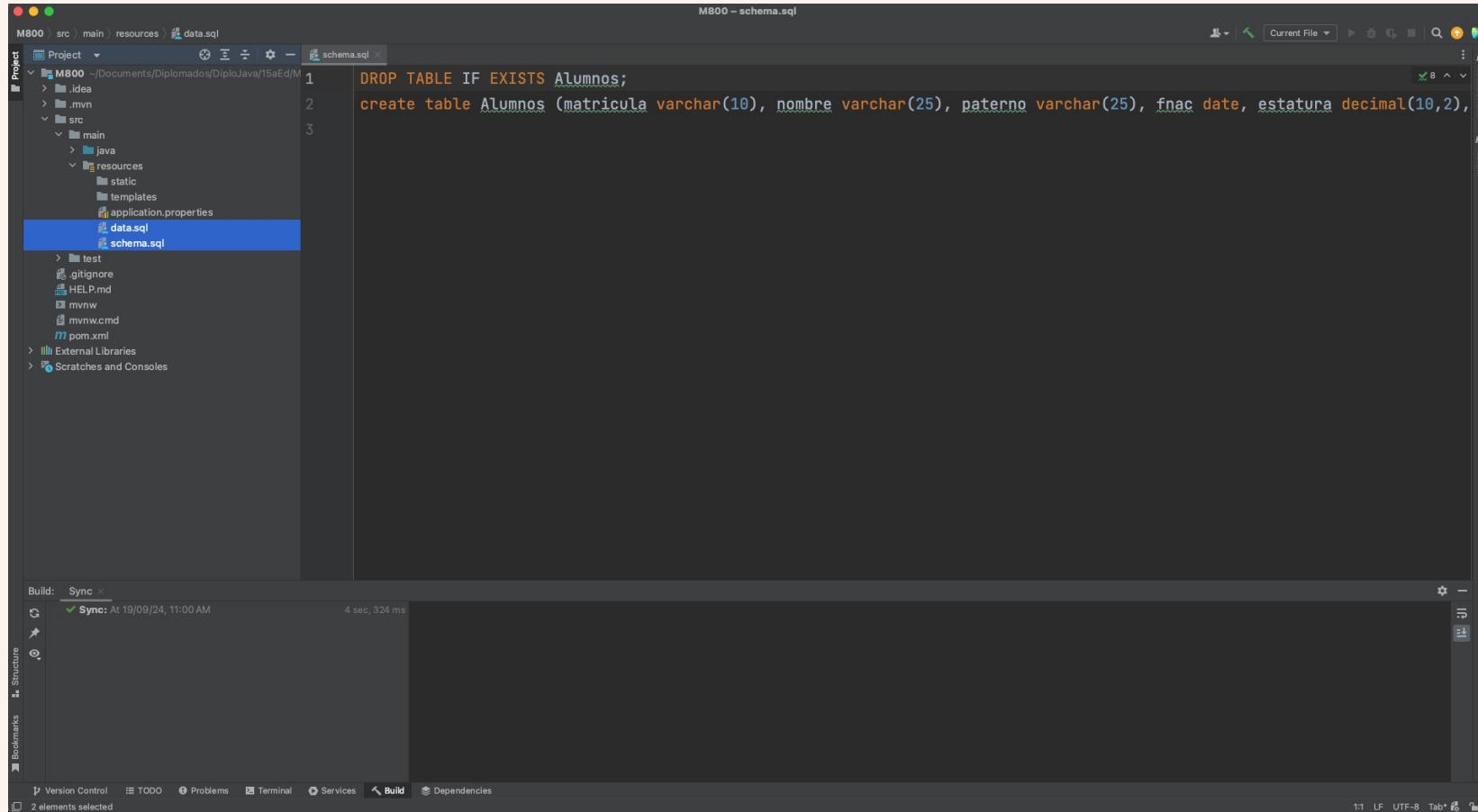
# Spring Data Configuración



# Spring Data Configuración



# Spring Data Configuración

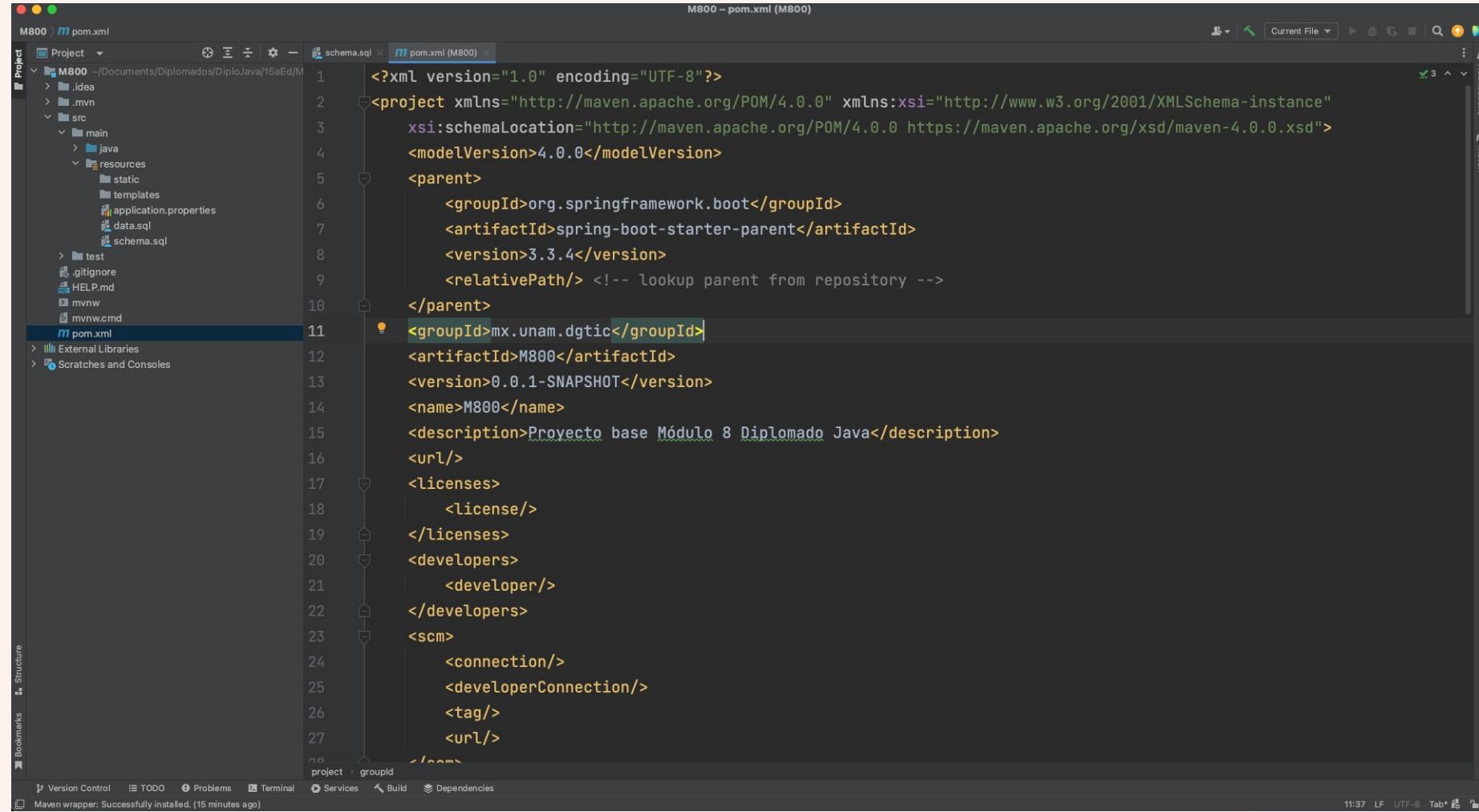


The screenshot shows an IDE interface with two tabs open: 'data.sql' and 'schema.sql'. The 'data.sql' tab contains the following SQL code:

```
DROP TABLE IF EXISTS Alumnos;
create table Alumnos (matricula varchar(10), nombre varchar(25), paterno varchar(25), fnac date, estatura decimal(10,2),
```

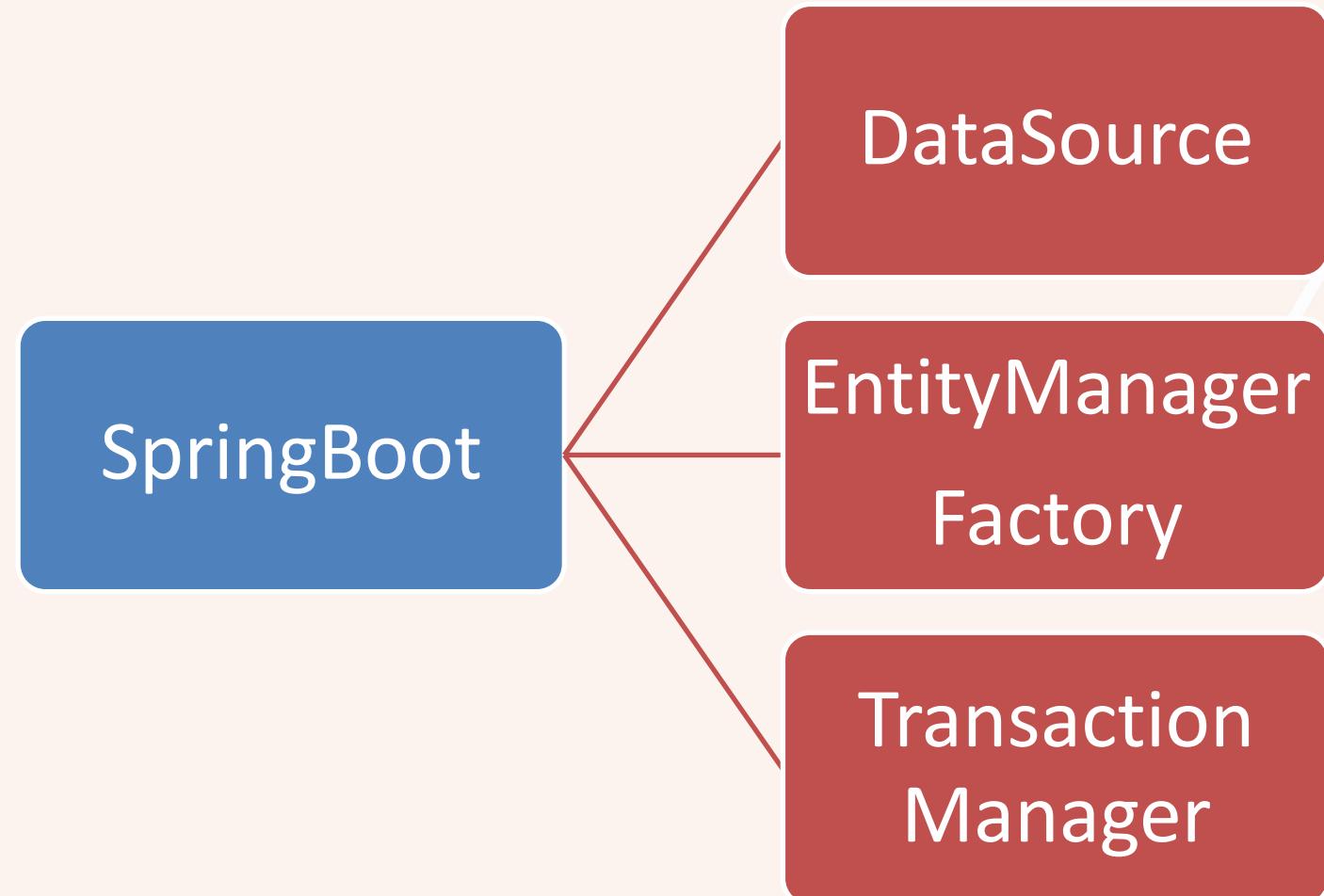
The 'schema.sql' tab contains the same SQL code. The left sidebar shows the project structure with files like 'application.properties', 'data.sql', and 'schema.sql' under 'src/main/resources'. The bottom status bar indicates a successful sync at 19/09/24, 11:00 AM.

# pom.xml

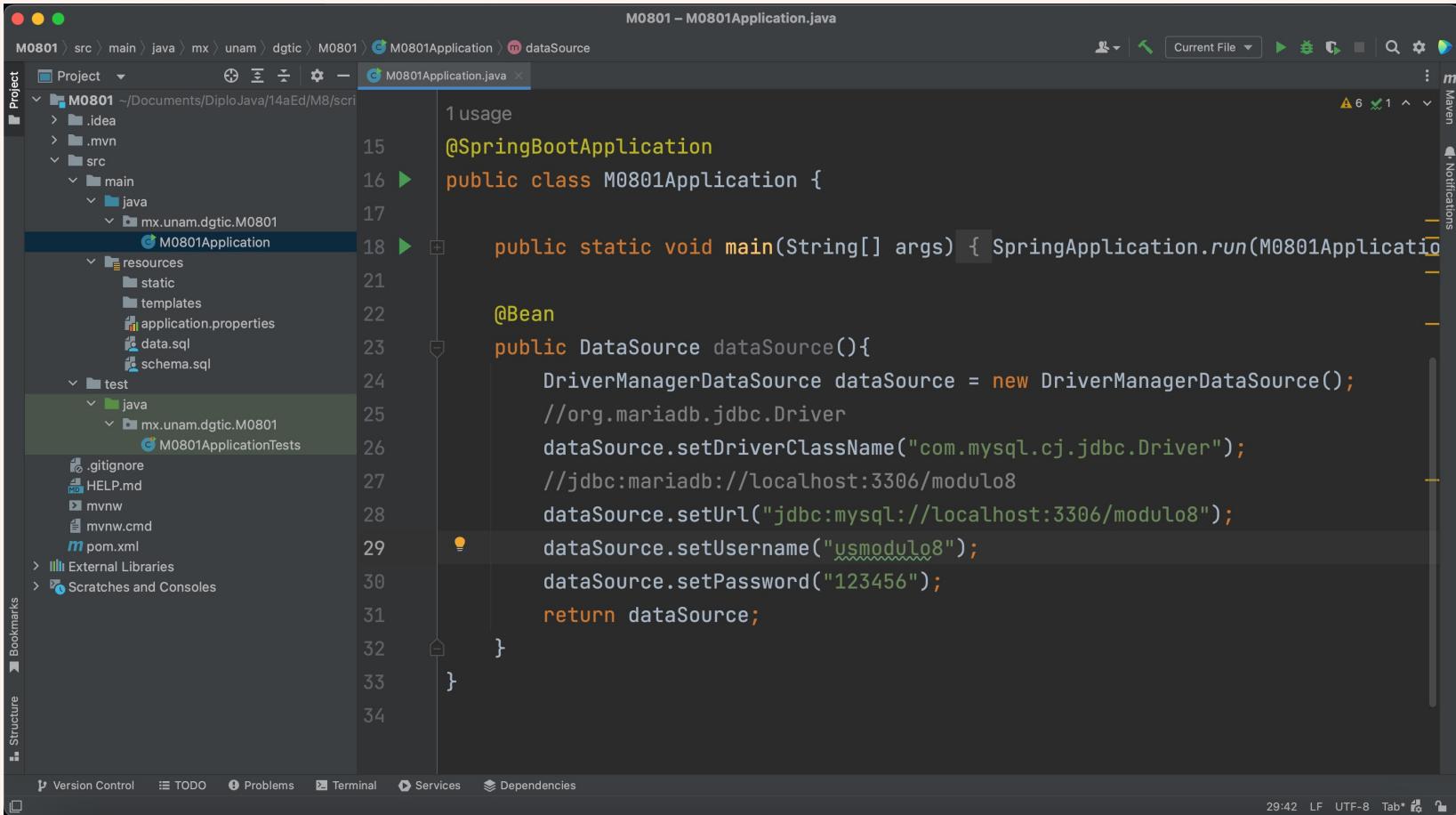


```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.3.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>mx.unam.dgtic</groupId>
  <artifactId>M800</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>M800</name>
  <description>Proyecto base Módulo 8 Diplomado Java</description>
  <url></url>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
</project>
```

# Spring Data



# Agregar DataSource



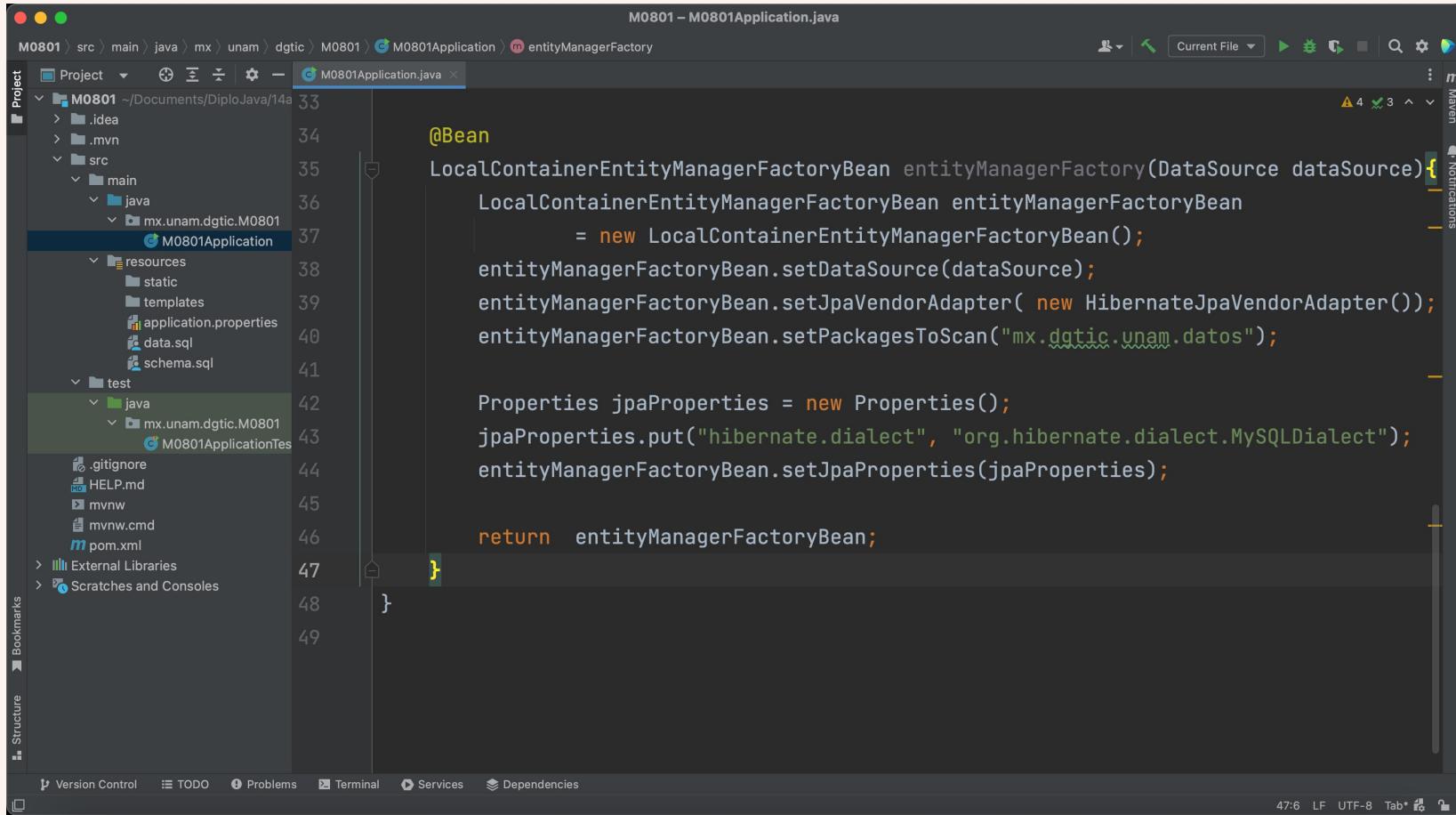
```
M0801 – M0801Application.java
M0801Application.java

1 usage
@SpringBootApplication
public class M0801Application {

    public static void main(String[] args) { SpringApplication.run(M0801Application.class, args); }

    @Bean
    public DataSource dataSource(){
        DriverManagerDataSource dataSource = new DriverManagerDataSource();
        //org.mariadb.jdbc.Driver
        dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
        //jdbc:mariadb://localhost:3306/modulo8
        dataSource.setUrl("jdbc:mysql://localhost:3306/modulo8");
        dataSource.setUsername("usmodulo8");
        dataSource.setPassword("123456");
        return dataSource;
    }
}
```

# Agregar EntityManagerFactory

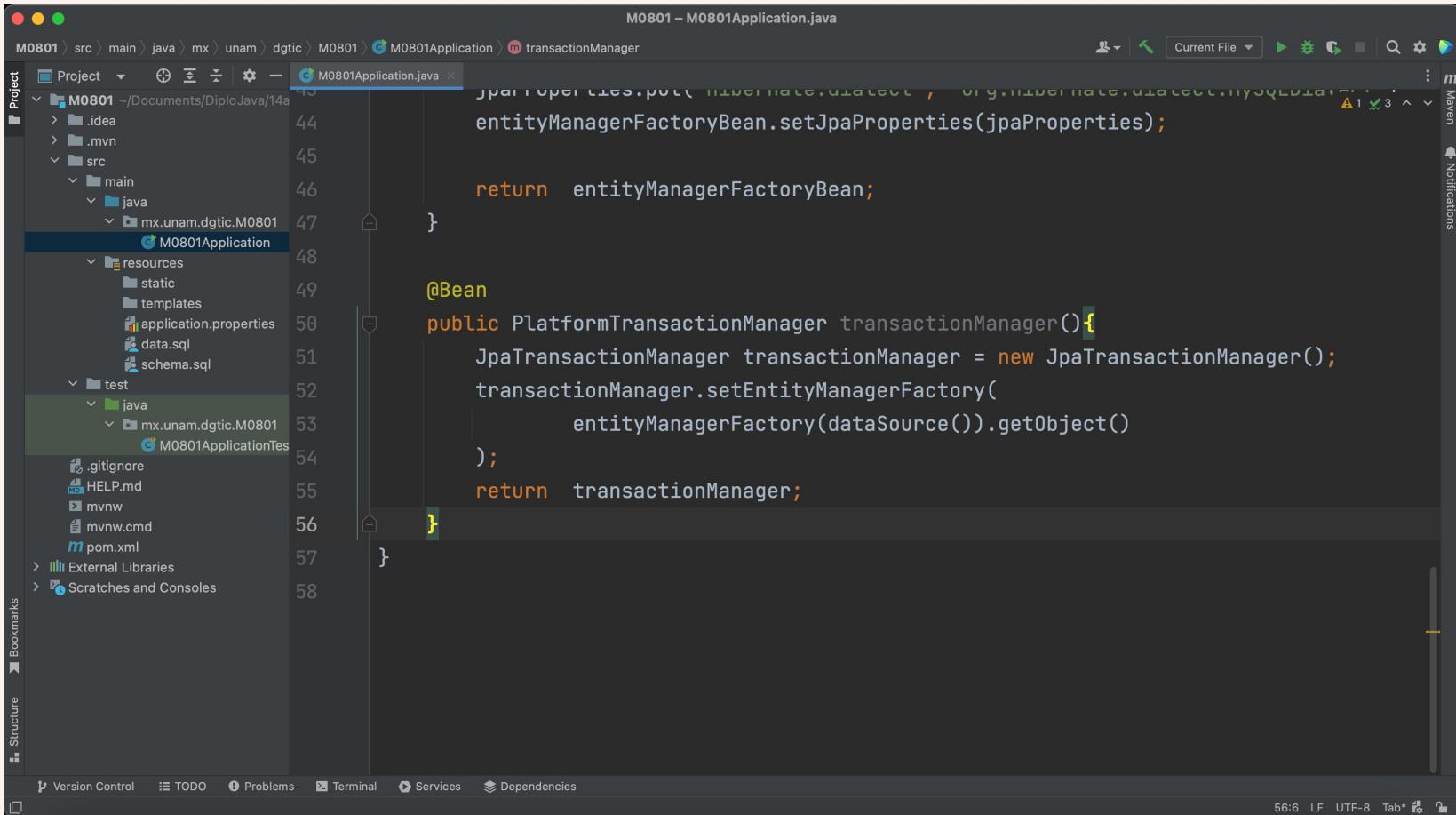


The screenshot shows the IntelliJ IDEA IDE interface with the file `M0801Application.java` open. The code implements a bean for the EntityManagerFactory:

```
33 @Bean
34 LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource dataSource) {
35     LocalContainerEntityManagerFactoryBean entityManagerFactoryBean =
36         new LocalContainerEntityManagerFactoryBean();
37     entityManagerFactoryBean.setDataSource(dataSource);
38     entityManagerFactoryBean.setJpaVendorAdapter(new HibernateJpaVendorAdapter());
39     entityManagerFactoryBean.setPackagesToScan("mx.dgtic.unam.datos");
40
41     Properties jpaProperties = new Properties();
42     jpaProperties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
43     entityManagerFactoryBean.setJpaProperties(jpaProperties);
44
45     return entityManagerFactoryBean;
46 }
47 }
```

The project structure on the left shows the directory `M0801` containing `.idea`, `.mvn`, `src` (with `main` and `test`), and `M0801Application.java`.

# Agregar TransactionManager



```
M0801 – M0801Application.java
entityManagerFactoryBean.setJpaProperties(jpaProperties);

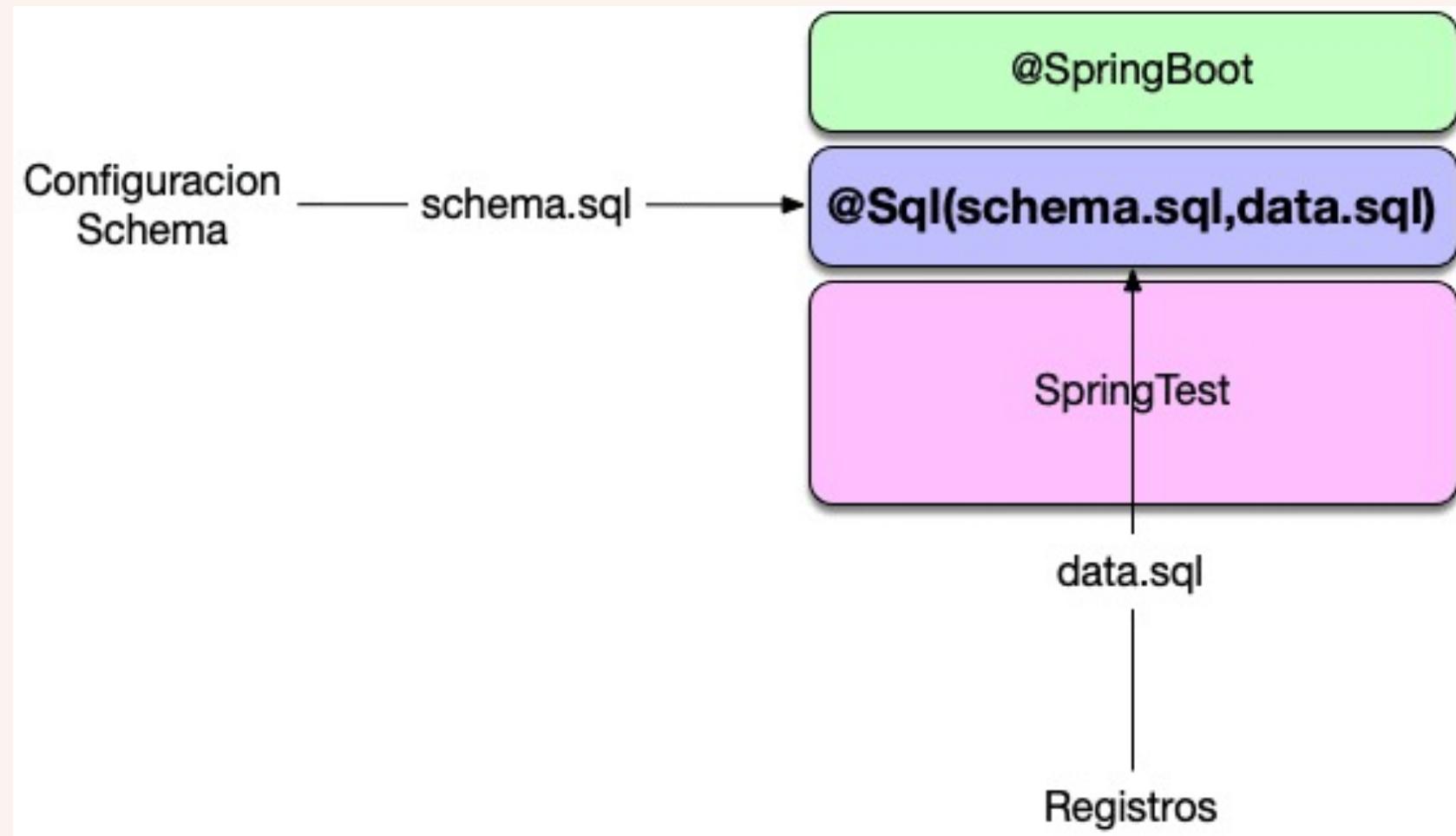
return entityManagerFactory;

@Bean
public PlatformTransactionManager transactionManager(){
    JpaTransactionManager transactionManager = new JpaTransactionManager();
    transactionManager.setEntityManagerFactory(
        entityManagerFactory(dataSource()).getObject()
    );
    return transactionManager;
}
```

## @Sql

- En el contexto de pruebas unitarias, la anotación @Sql permite cargar automáticamente información en la base de datos antes o después de la ejecución de los tests.
- Facilita la configuración de scripts SQL para inicializar el estado de la base de datos, proporcionando un entorno controlado para las pruebas.
- Se puede usar a nivel de clase o método de prueba, y admite múltiples scripts.

# @Sql



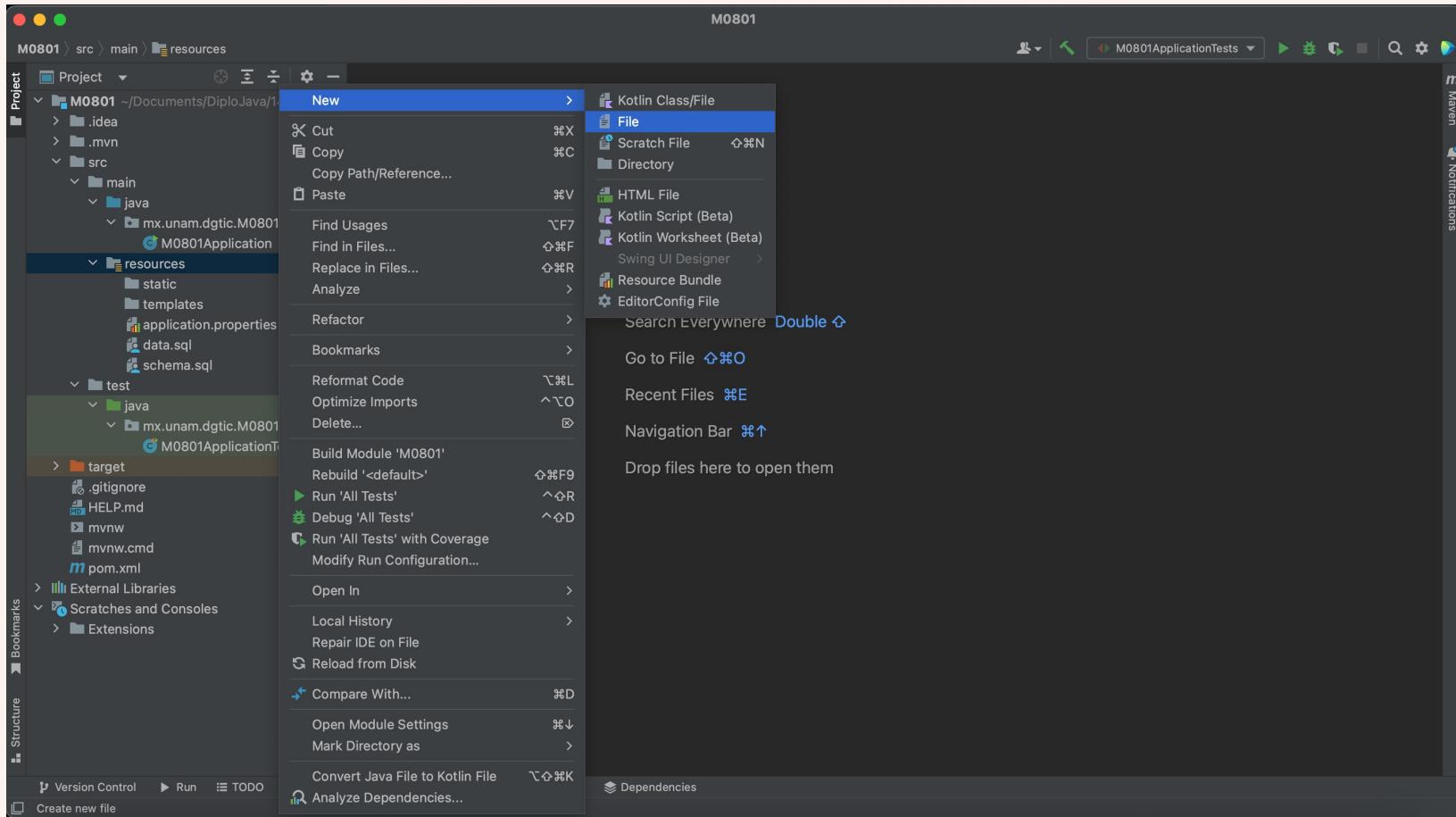
@Sql

@Sql(schema.sql, data.sql)

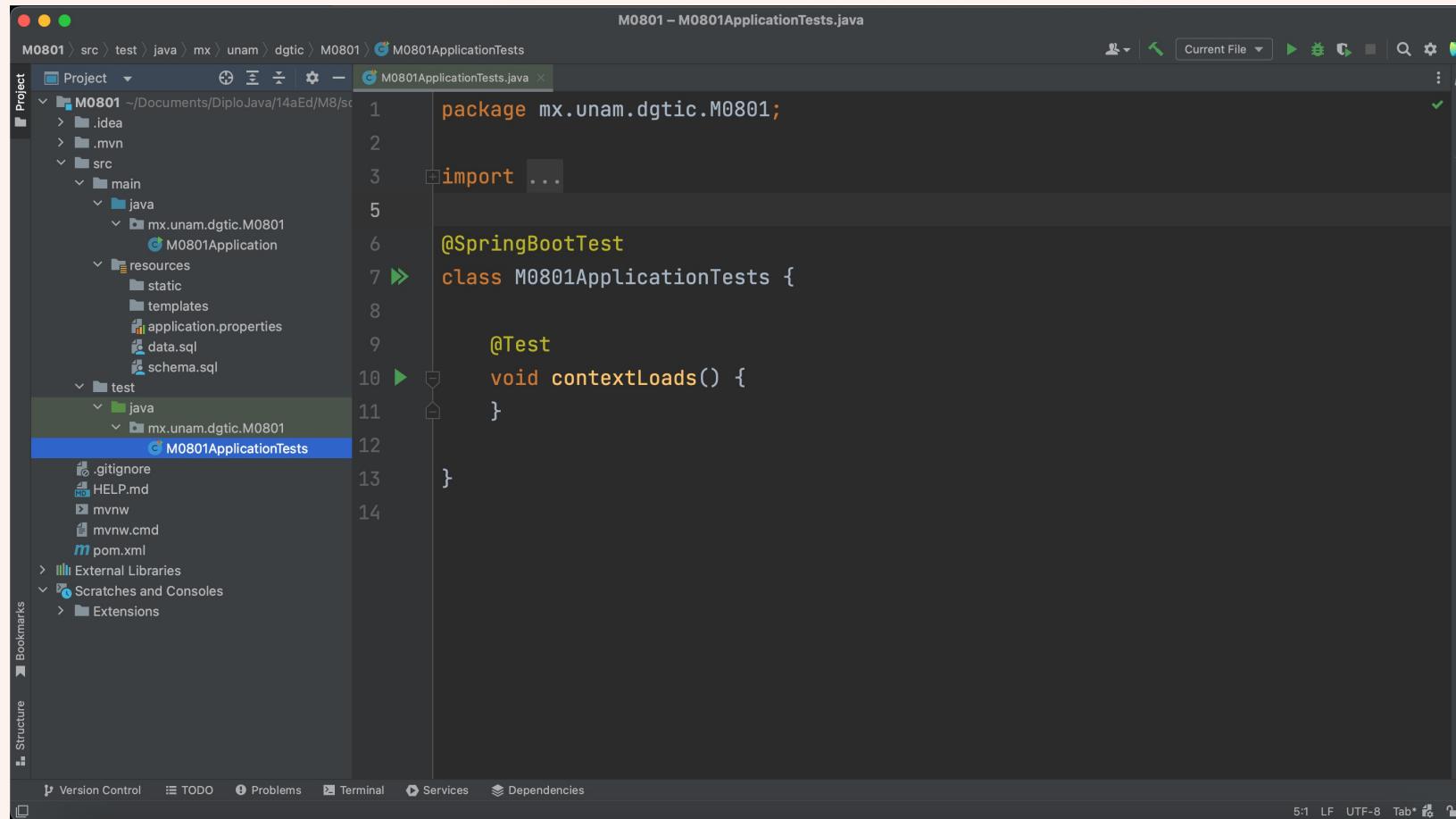
create

insert

# Agregar archivos sql



# Agregar JUnit Test



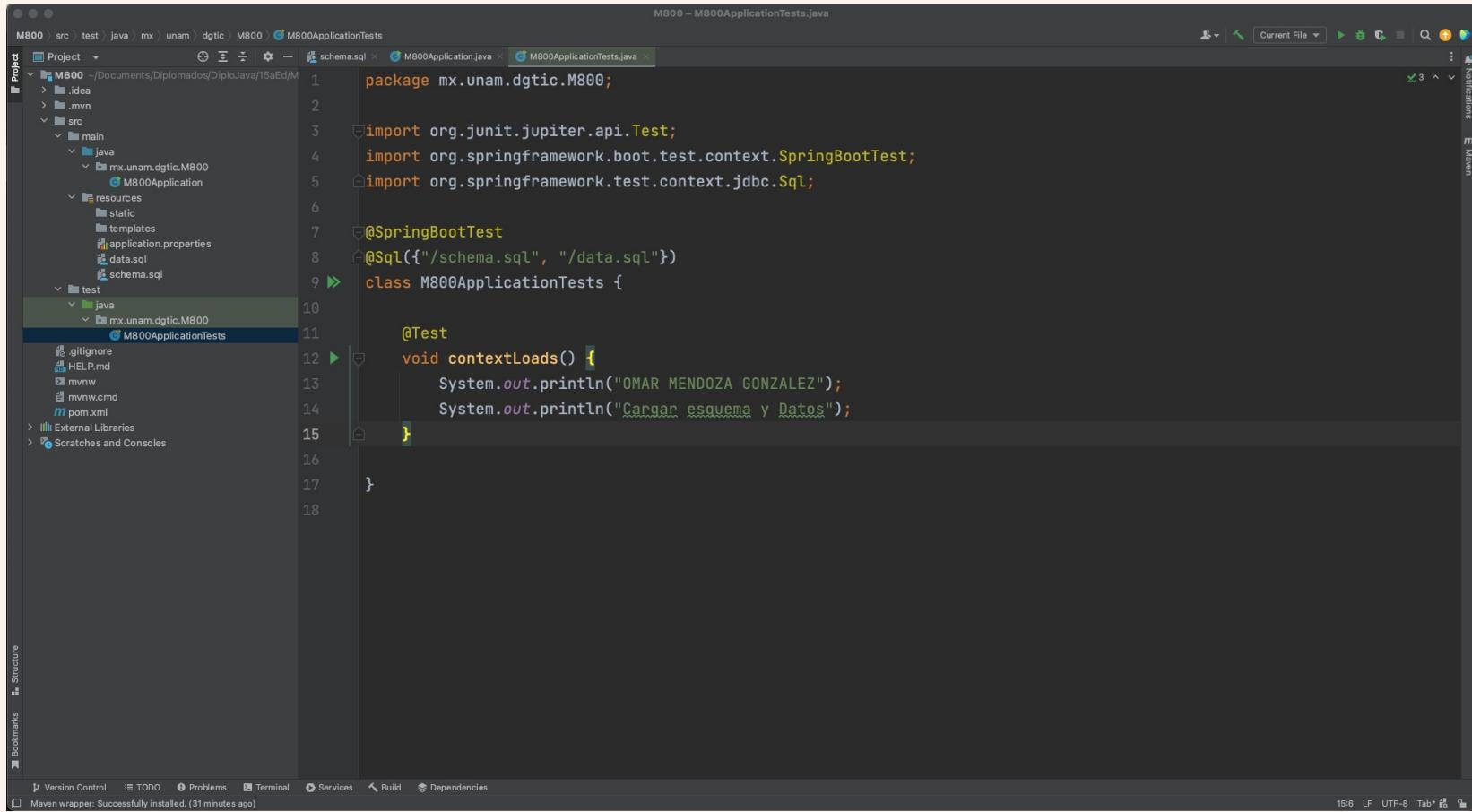
The screenshot shows the IntelliJ IDEA interface with the following details:

- Title Bar:** M0801 – M0801ApplicationTests.java
- Project Tool Window:** Shows the project structure for 'M0801'. It includes a .idea folder, a .mvn folder, a 'src' directory containing 'main' and 'test' sub-directories. 'main/java' contains 'mx.unam.dgtic.M0801' which has a 'M0801Application' class. 'test/java' contains 'mx.unam.dgtic.M0801' which has a 'M0801ApplicationTests' class selected.
- Code Editor:** Displays the code for 'M0801ApplicationTests.java'. The code is as follows:

```
package mx.unam.dgtic.M0801;  
  
import ...  
  
@SpringBootTest  
class M0801ApplicationTests {  
  
    @Test  
    void contextLoads() {  
    }  
}
```

The code editor also shows file navigation icons (back, forward, search) and a Maven tool window.

# Definir datos para prueba



The screenshot shows the IntelliJ IDEA interface with a Java test class named `M800ApplicationTests.java`. The code defines a test method `contextLoads()` that prints "OMAR MENDOZA GONZALEZ" and "Cargar esquema y Datos" to the console.

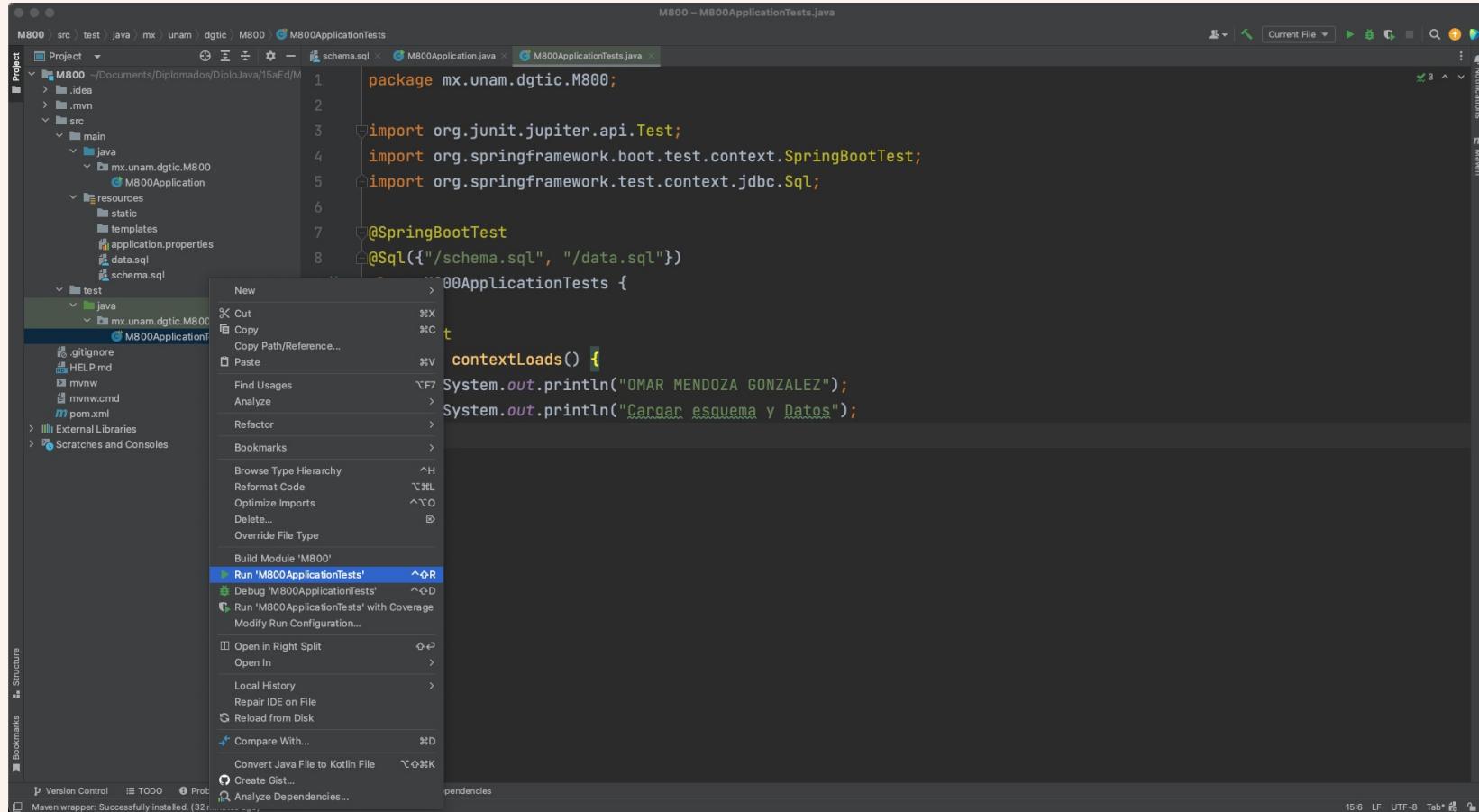
```
package mx.unam.dgtic.M800;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.jdbc.Sql;

@SpringBootTest
@Sql({"schema.sql", "data.sql"})
class M800ApplicationTests {

    @Test
    void contextLoads() {
        System.out.println("OMAR MENDOZA GONZALEZ");
        System.out.println("Cargar esquema y Datos");
    }
}
```

# Ejecutar Test

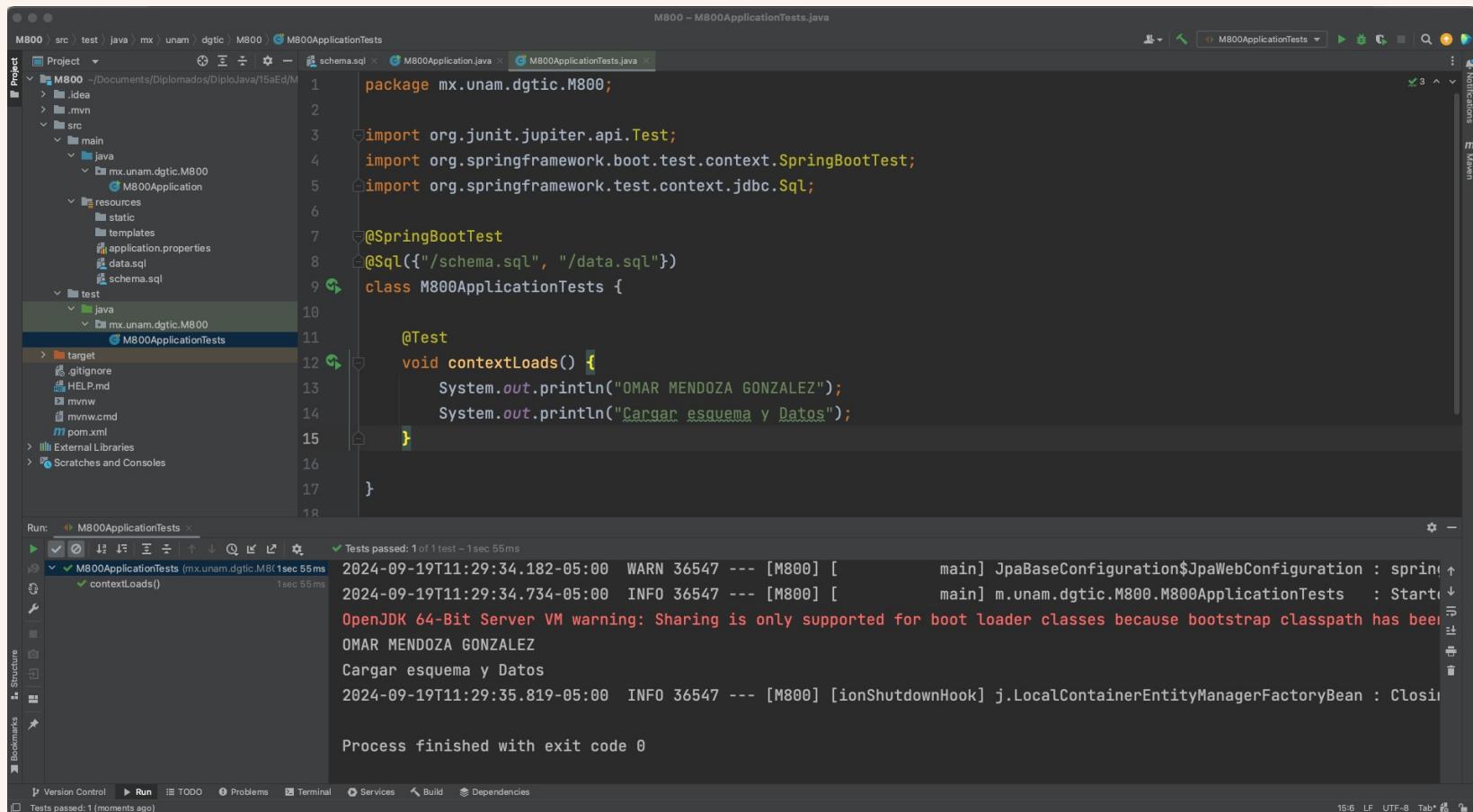


The screenshot shows the IntelliJ IDEA interface with the code editor open to `M800ApplicationTests.java`. The code is a Spring Boot test class:

```
package mx.unam.dgtic.M800;  
  
import org.junit.jupiter.api.Test;  
import org.springframework.boot.test.context.SpringBootTest;  
import org.springframework.test.context.jdbc.Sql;  
  
@SpringBootTest  
@SpringBootTest({"/schema.sql", "/data.sql"})  
public class M800ApplicationTests {  
  
    @Test  
    void contextLoads() {  
        System.out.println("OMAR MENDOZA GONZALEZ");  
        System.out.println("Cargar esquema y Datos");  
    }  
}
```

A context menu is displayed over the code, with the option `Run 'M800ApplicationTests'` highlighted in blue.

# Ejecutar Test

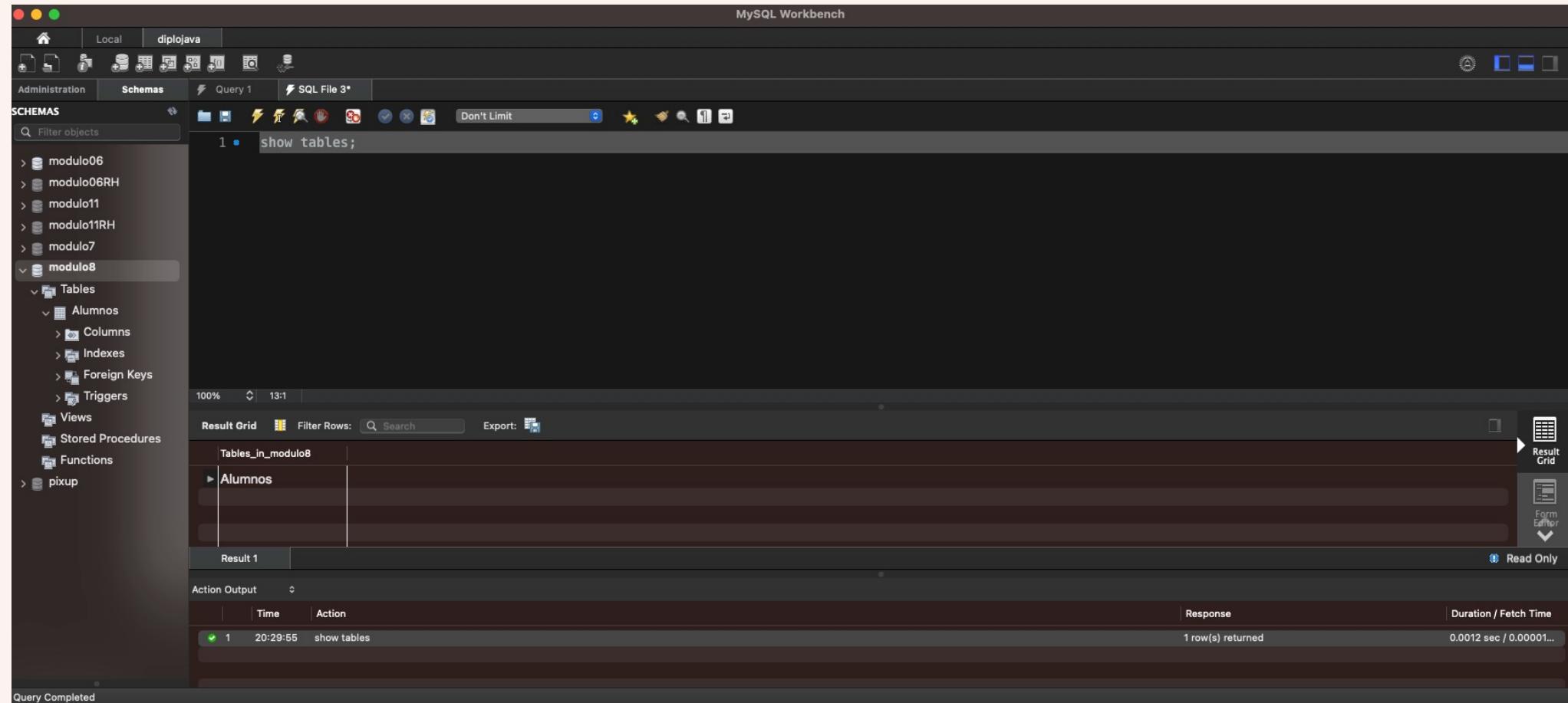


The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "M800". It contains a "src" directory with "main" and "test" sub-directories. "main" contains "java" (with "M800Application") and "resources" (with "application.properties", "data.sql", and "schema.sql"). "test" contains "java" (with "M800ApplicationTests") and "resources" (with "schema.sql").
- M800ApplicationTests.java:** The code defines a test class `M800ApplicationTests` with a single test method `contextLoads()` that prints "OMAR MENDOZA GONZALEZ" and "Cargar esquema y Datos".
- Run Tab:** The "Run" tab shows the test was run successfully: "Tests passed: 1 of 1 test – 1 sec 55ms". The log output shows the test name, duration, and the printed messages.
- Log Output:** The log shows the following entries:

```
2024-09-19T11:29:34.182-05:00  WARN 36547 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, by default, JPA queries cannot be registered when embedded database is used. To change the behavior, configure 'spring.jpa.open-in-view' to false.
2024-09-19T11:29:34.734-05:00  INFO 36547 --- [main] m.unam.dgtic.M800.M800ApplicationTests : Started M800ApplicationTests in 1.01 seconds (JVM: 1.01s)
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been explicitly specified, by -bootclasspath option and/or boot.class.path entry in the java.vm.specification_class_path property
OMAR MENDOZA GONZALEZ
Cargar esquema y Datos
2024-09-19T11:29:35.819-05:00  INFO 36547 --- [main] org.springframework.boot.SpringApplication : Application finished successfully
Process finished with exit code 0
```

# Validar la creación de la tabla



# Contacto

Dr. Omar Mendoza González

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