



DIPLOMADO

Desarrollo de sistemas con tecnología Java

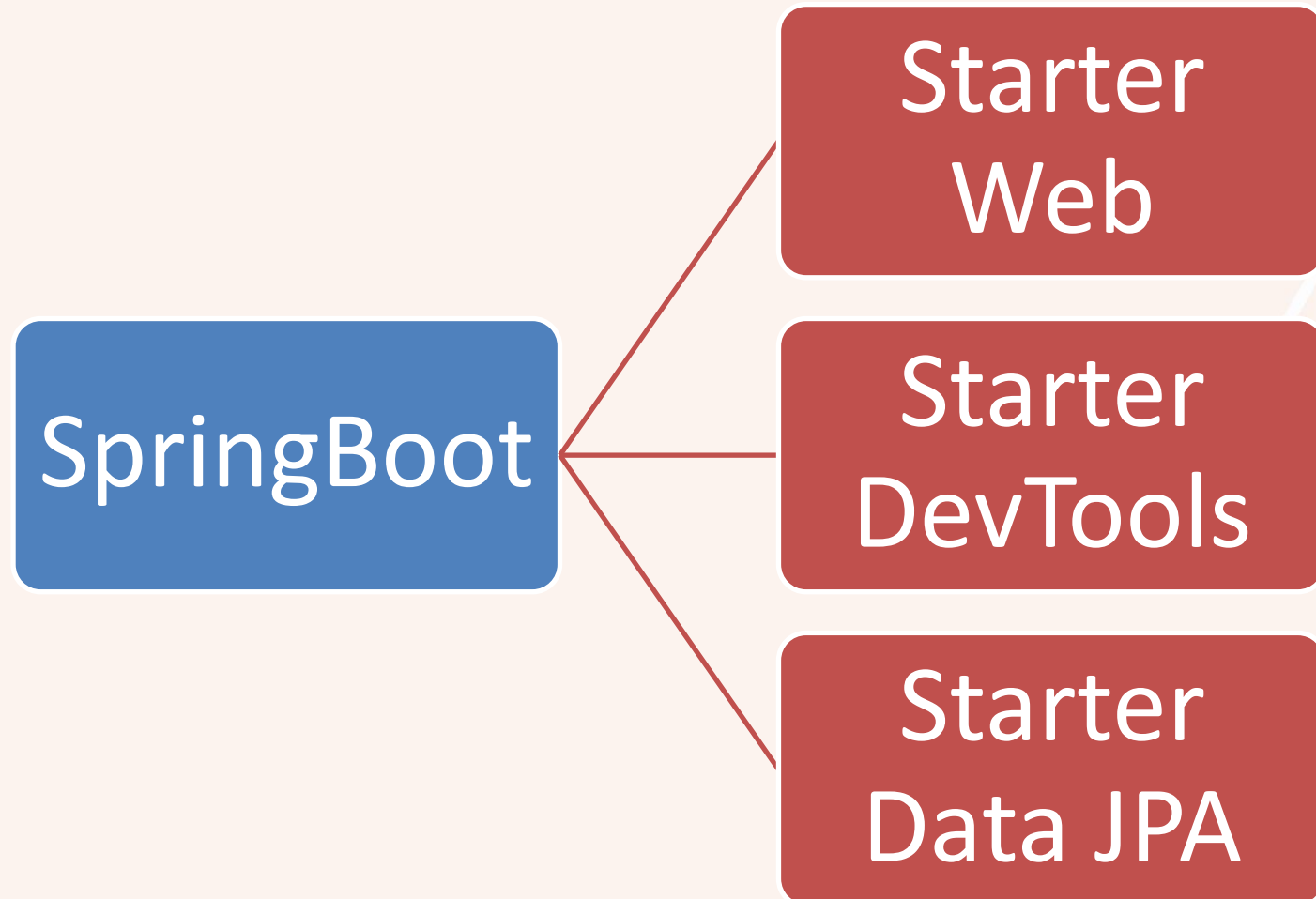
Módulo 7

Persistencia con Spring Data

Mtro. Víctor Manuel Sánchez Sánchez

victorsanchezh0@aragon.unam.mx

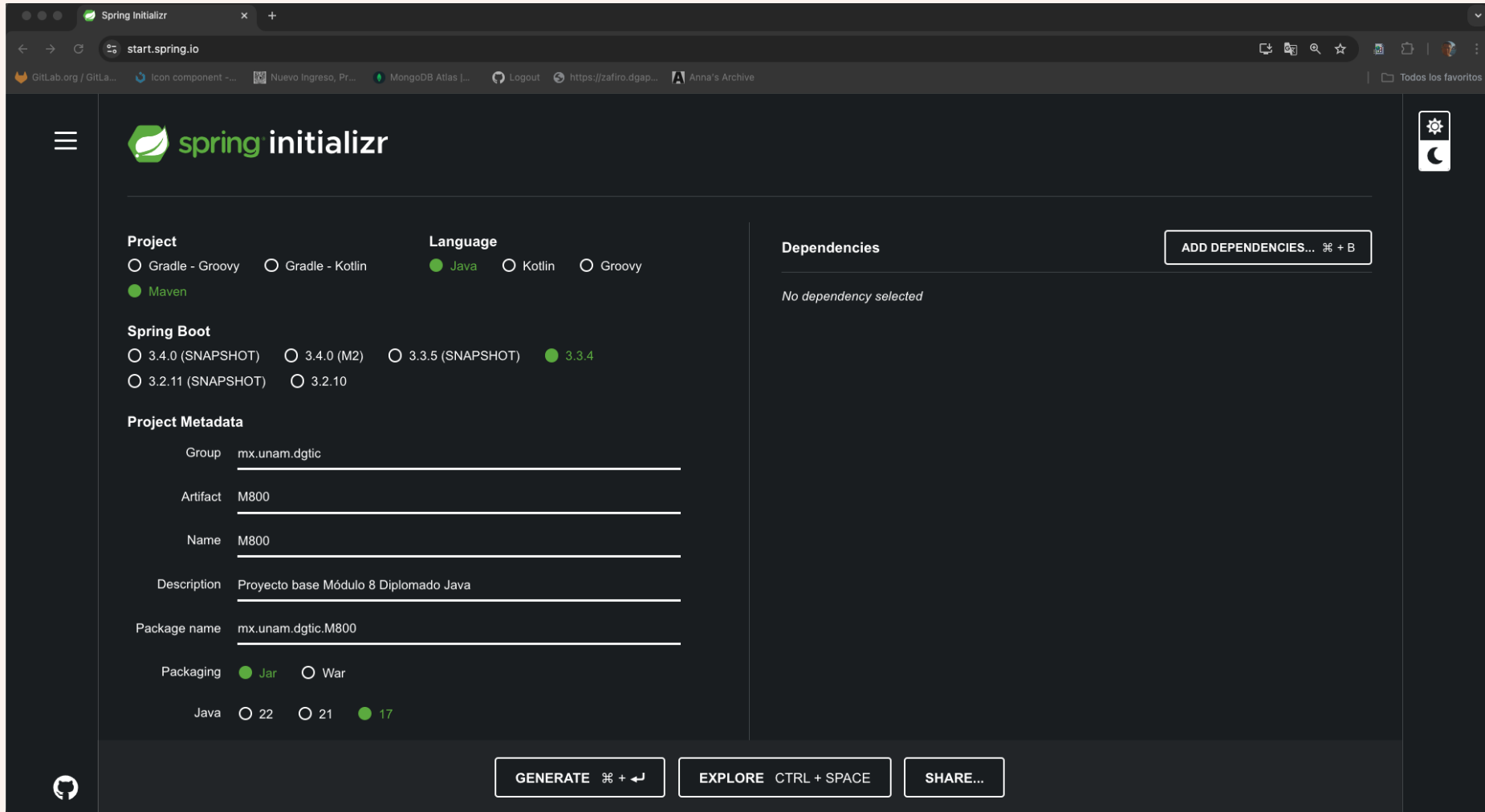
Spring Data



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

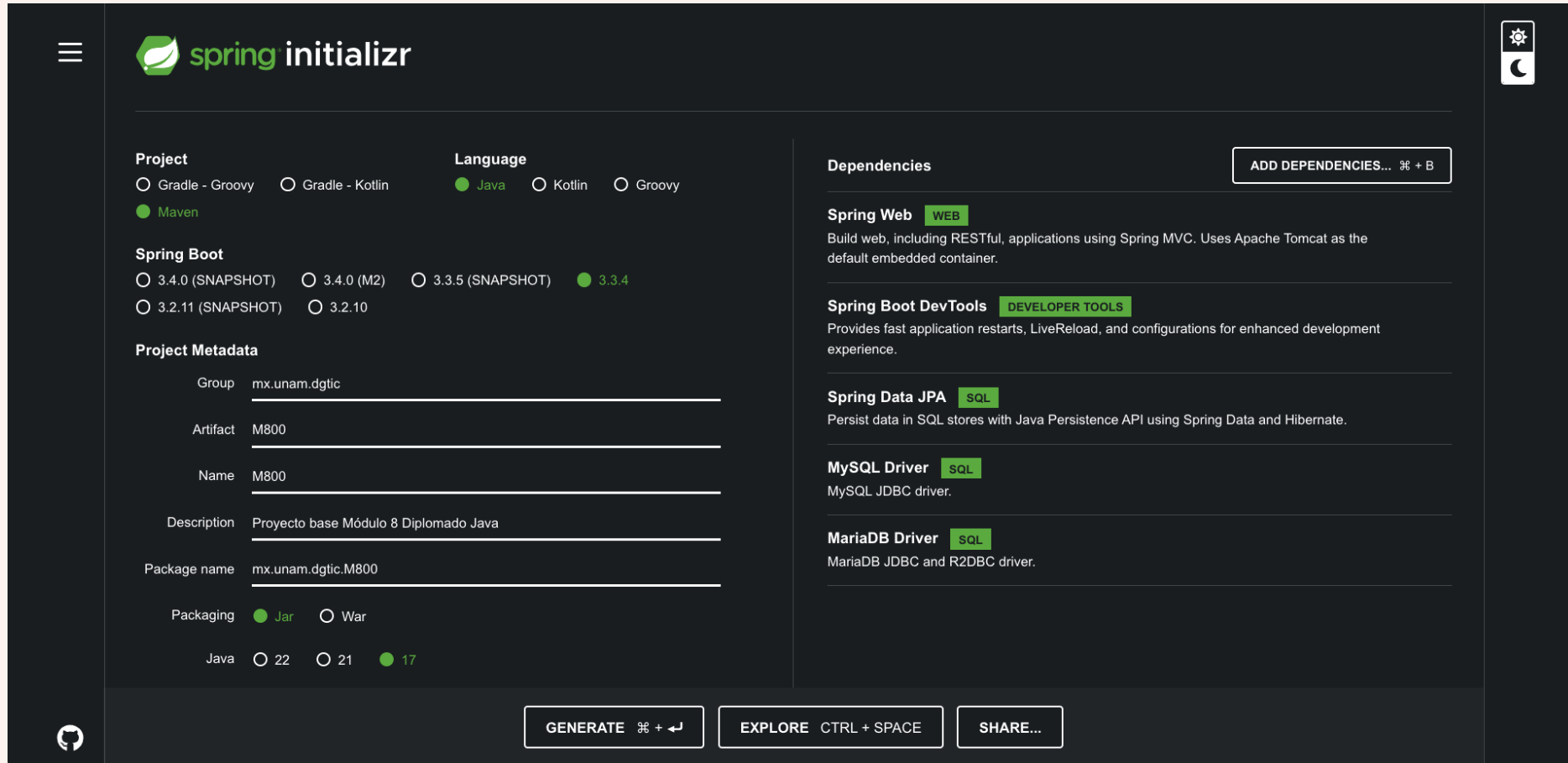


The screenshot shows the Spring Initializr web application interface. The browser address bar displays `start.spring.io`. The page features a sidebar with a hamburger menu icon and a settings icon. The main content area is divided into several sections:

- Project:** Includes radio buttons for `Gradle - Groovy`, `Gradle - Kotlin`, `Java` (selected), `Kotlin`, and `Groovy`. Below this is a radio button for `Maven` (selected).
- Spring Boot:** Includes radio buttons for `3.4.0 (SNAPSHOT)`, `3.4.0 (M2)`, `3.3.5 (SNAPSHOT)`, `3.3.4` (selected), `3.2.11 (SNAPSHOT)`, and `3.2.10`.
- Project Metadata:** Includes input fields for `Group` (filled with `mx.unam.dgtic`), `Artifact` (filled with `M800`), `Name` (filled with `M800`), `Description` (filled with `Proyecto base Módulo 8 Diplomado Java`), and `Package name` (filled with `mx.unam.dgtic.M800`).
- Packaging:** Includes radio buttons for `Jar` (selected) and `War`.
- Language:** Includes radio buttons for `Java` (selected), `22`, `21`, and `17`.
- Dependencies:** Includes a button `ADD DEPENDENCIES... ⌘ + B` and the text `No dependency selected`.

At the bottom of the page, there are three buttons: `GENERATE ⌘ + ↵`, `EXPLORE CTRL + SPACE`, and `SHARE...`.

Spring Data Configuración



The screenshot shows the Spring Initializr web application interface. It is a dark-themed form for generating a Spring project. The interface is divided into several sections: Project, Language, Spring Boot, Project Metadata, Dependencies, and a bottom bar with action buttons.

Project

☐ Gradle - Groovy ☐ Gradle - Kotlin ☒ Java ☐ Kotlin ☐ Groovy

☒ Maven

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 [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 driver.

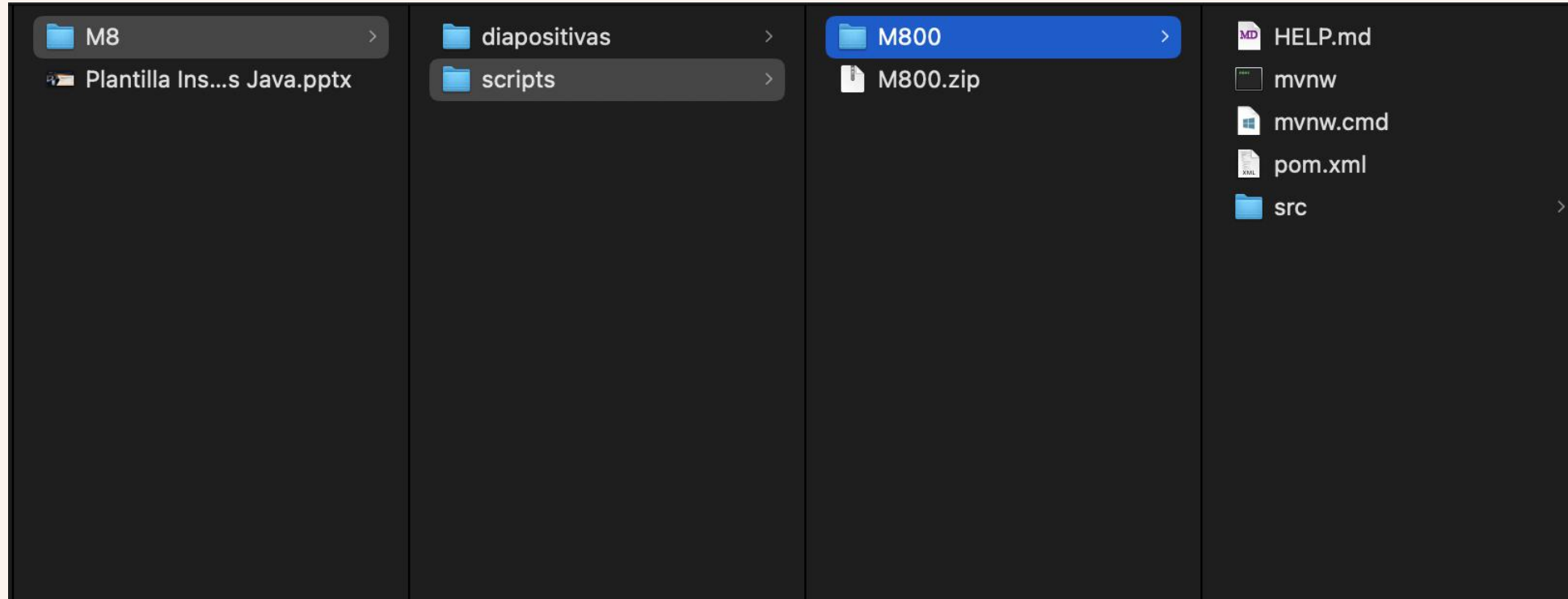
MariaDB Driver **SQL**

MariaDB JDBC and R2DBC driver.

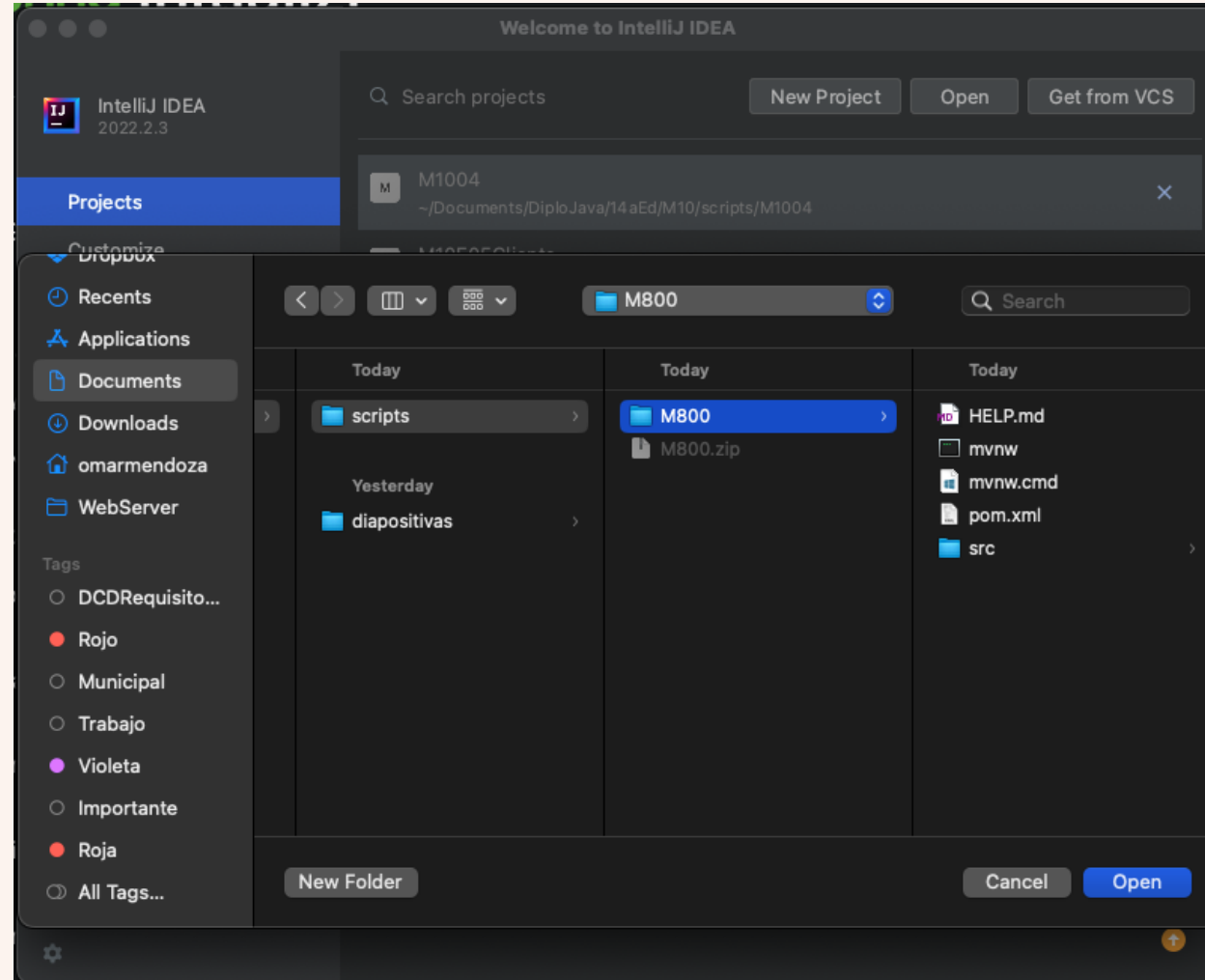
Bottom Bar:

[GENERATE % + ↵](#) [EXPLORE CTRL + SPACE](#) [SHARE...](#)

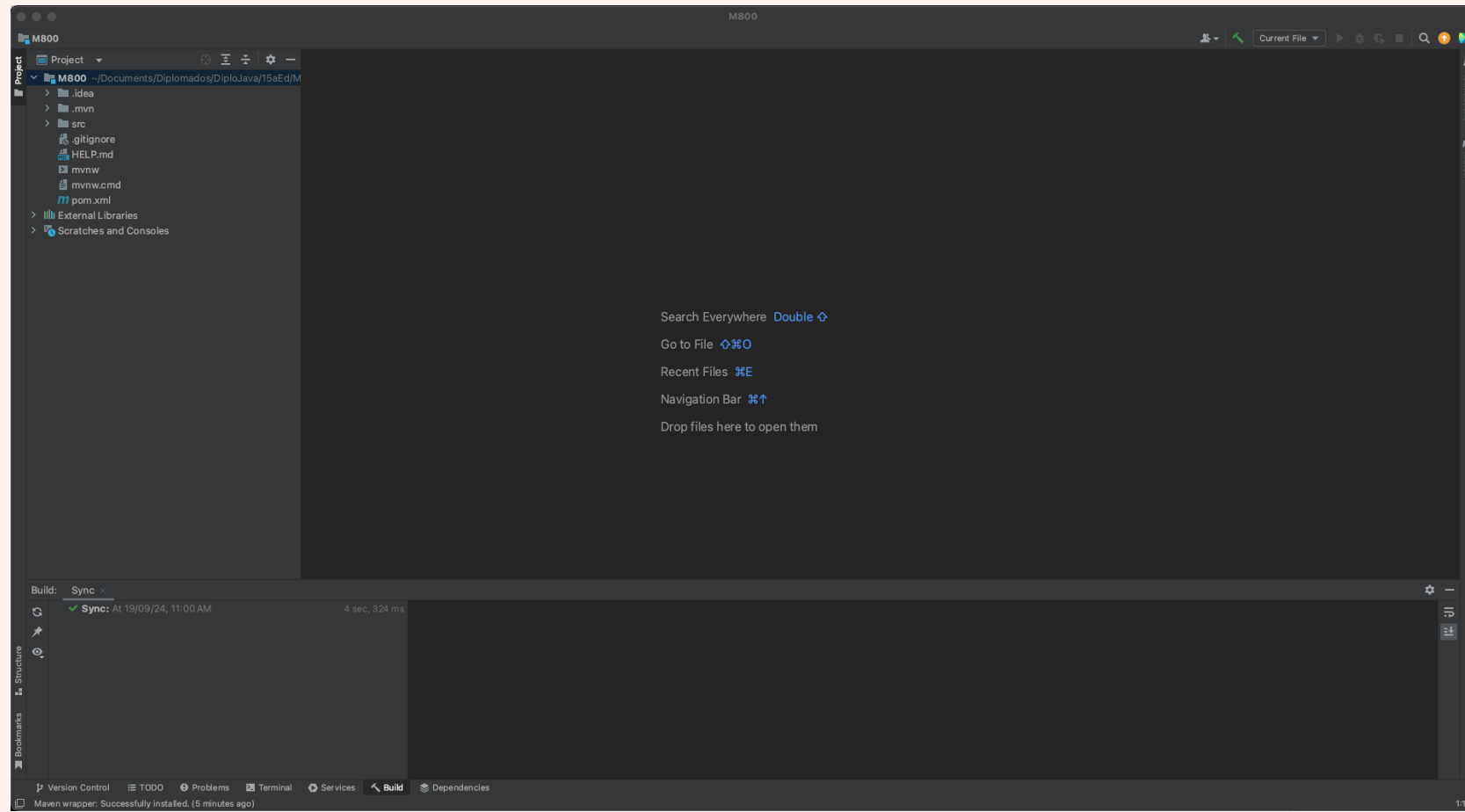
Spring Data Configuración



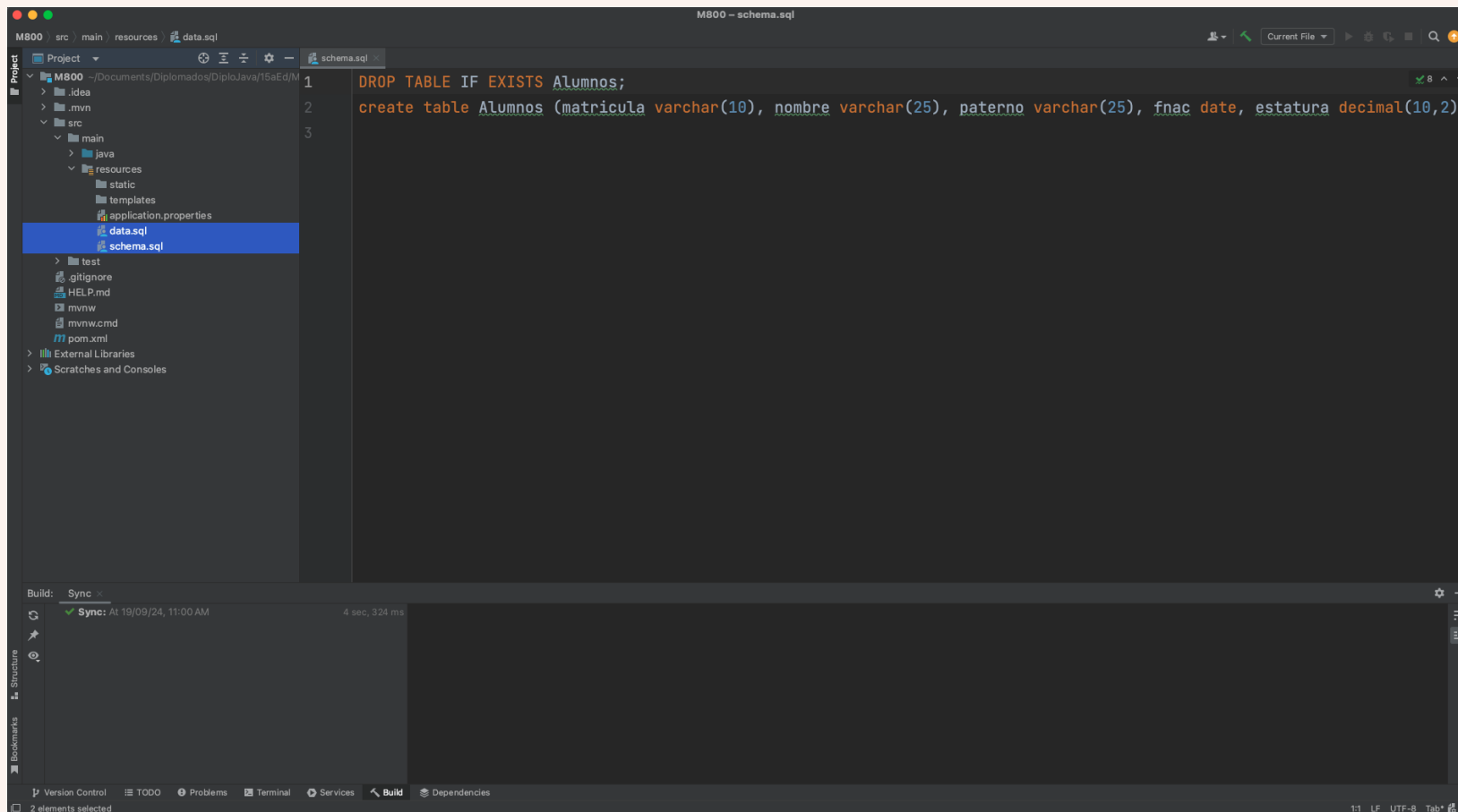
Spring Data Configuración



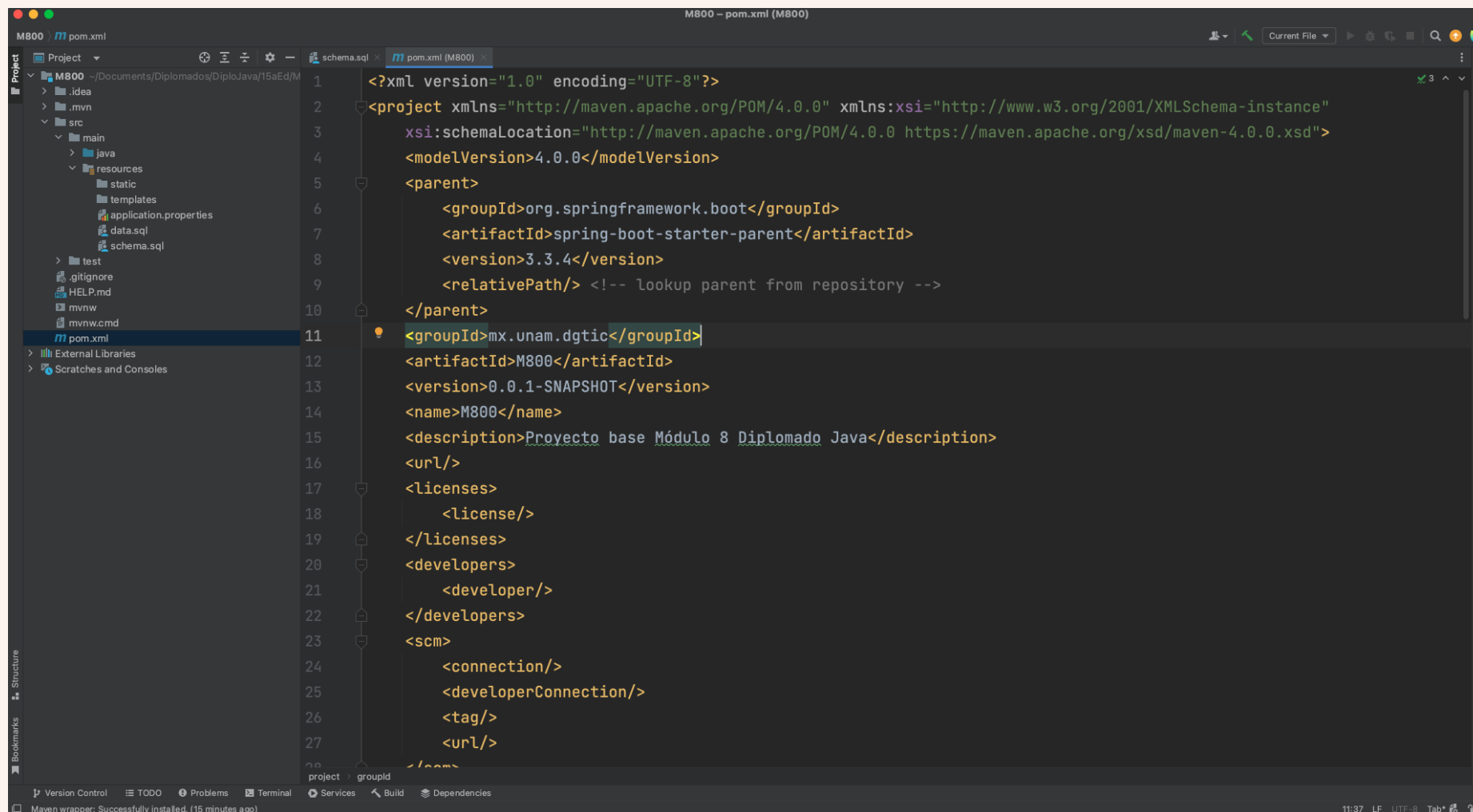
Spring Data Configuración



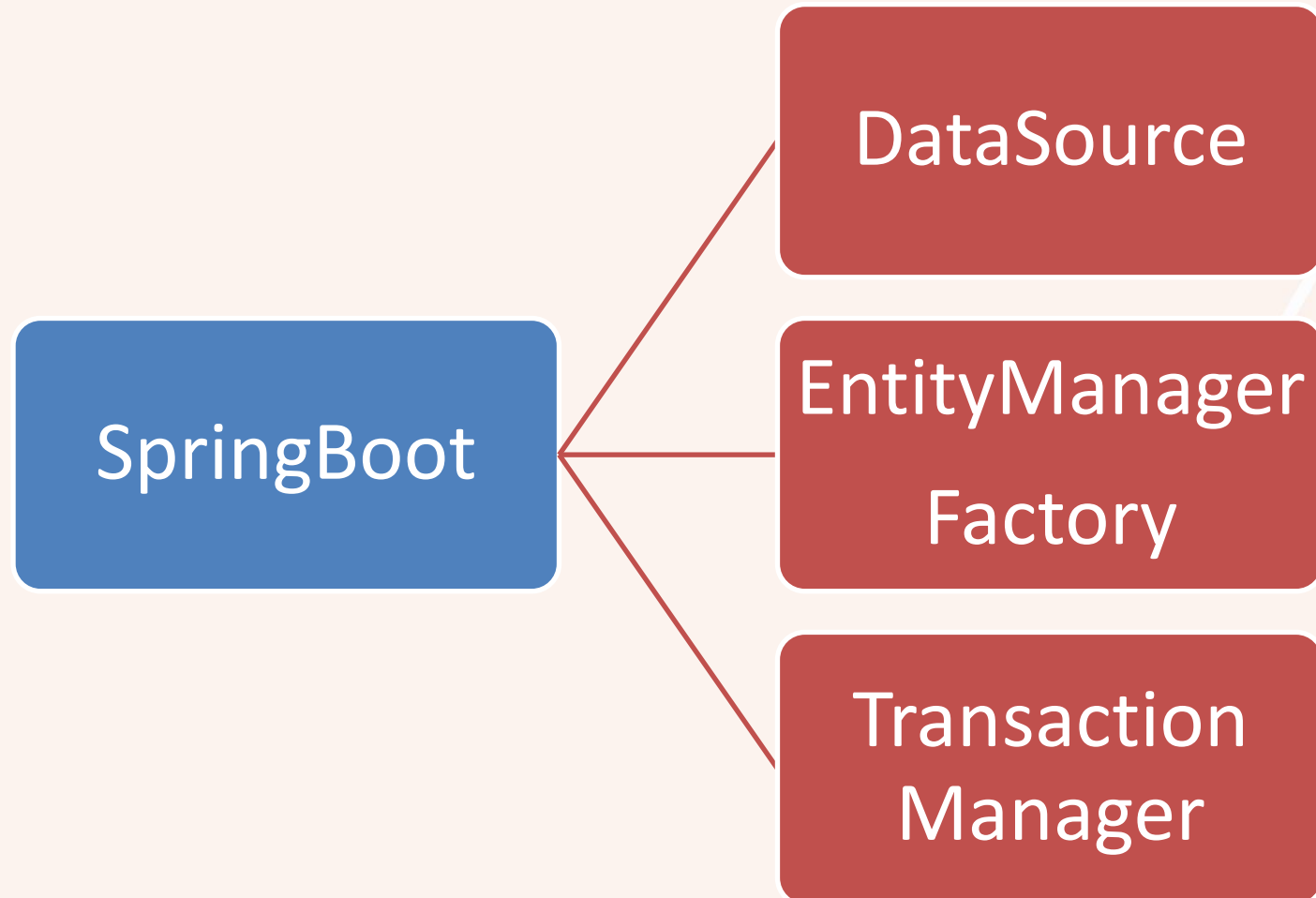
Spring Data Configuración



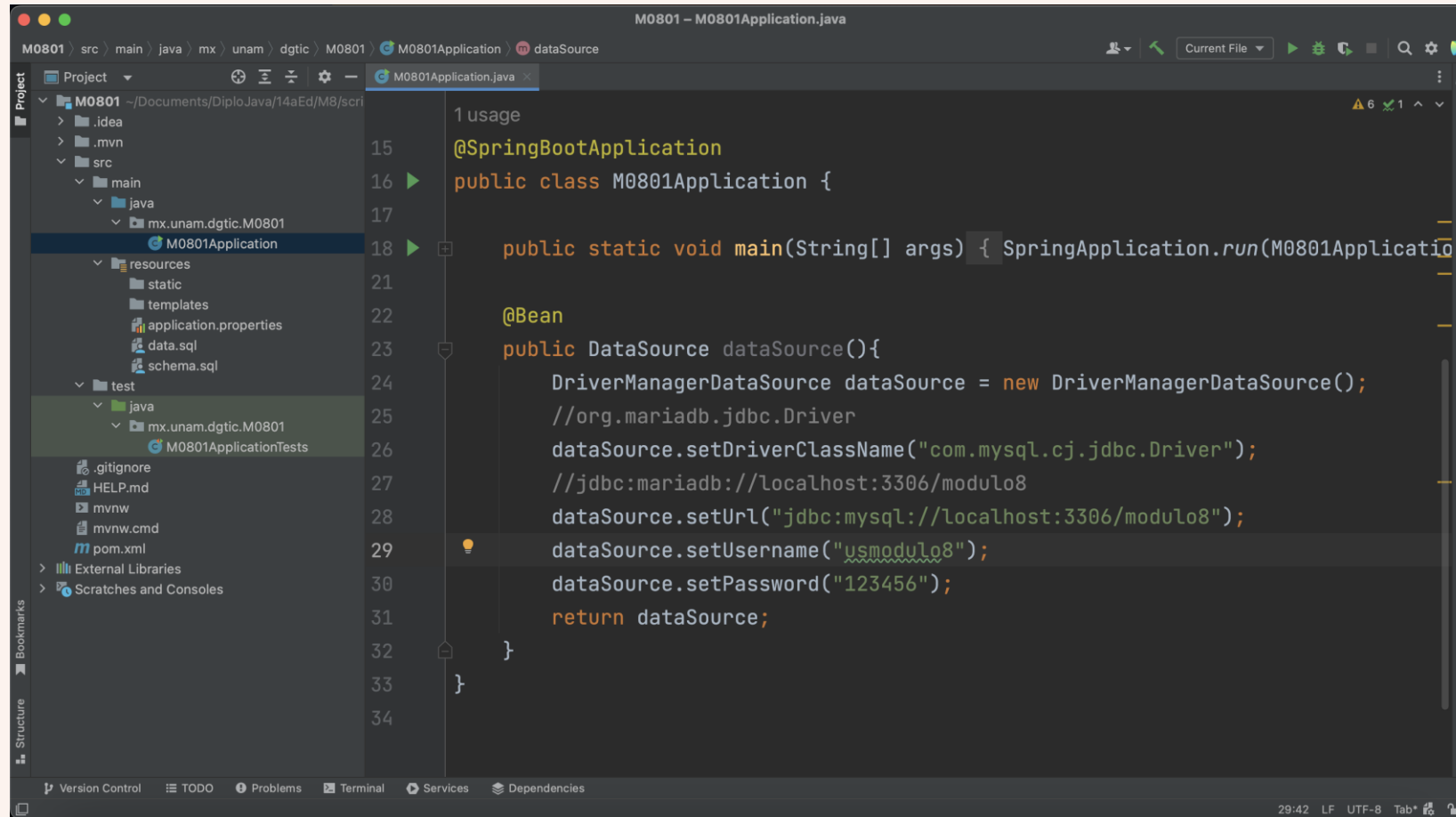
pom.xml



Spring Data



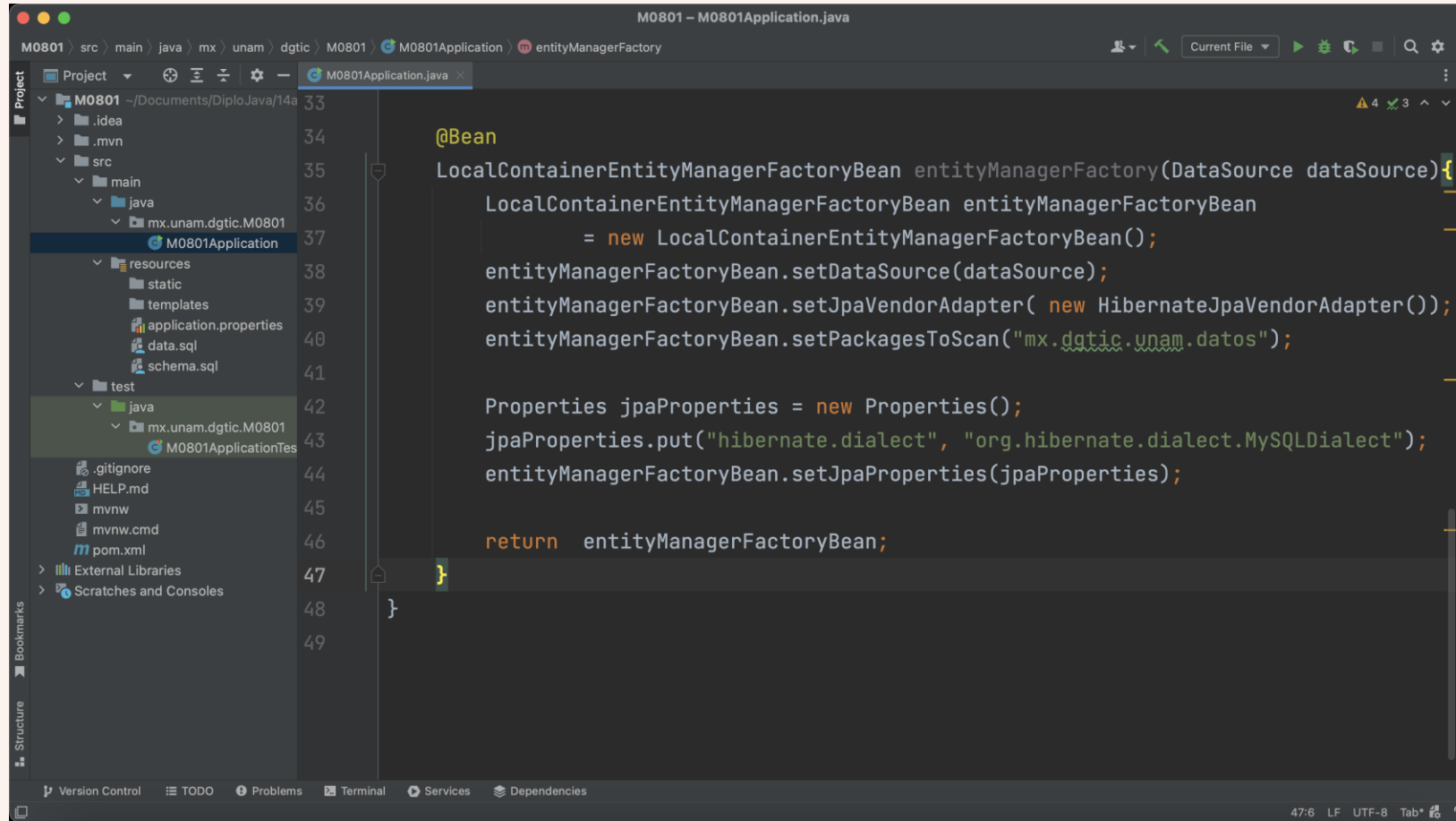
Agregar DataSource



The screenshot shows an IDE window titled "M0801 - M0801Application.java". The left sidebar displays the project structure for "M0801", including folders like "main", "resources", and "test". The main editor area shows the code for "M0801Application.java". The code includes a package declaration, imports for "SpringBootApplication" and "DataSource", and a "main" method. A new bean, "dataSource", is added with the following code:

```
1 usage
15 @SpringBootApplication
16 public class M0801Application {
17
18     public static void main(String[] args) { SpringApplication.run(M0801Application.class, args); }
19
20     @Bean
21     public DataSource dataSource() {
22         DriverManagerDataSource dataSource = new DriverManagerDataSource();
23         //org.mariadb.jdbc.Driver
24         dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");
25         //jdbc:mariadb://localhost:3306/modulo8
26         dataSource.setUrl("jdbc:mysql://localhost:3306/modulo8");
27         dataSource.setUsername("usmodulo8");
28         dataSource.setPassword("123456");
29         return dataSource;
30     }
31 }
32 }
33
34
```

Agregar EntityManagerFactory



```
M0801 – M0801Application.java
M0801 > src > main > java > mx > unam > dgtic > M0801 > M0801Application > entityManagerFactory

Project
M0801 ~/Documents/DiploJava/14a
  .idea
  .mvn
  src
    main
      java
        mx.unam.dgtic.M0801
          M0801Application
            resources
              static
              templates
              application.properties
              data.sql
              schema.sql
            test
              java
                mx.unam.dgtic.M0801
                  M0801ApplicationTes
            .gitignore
            HELP.md
            mvnw
            mvnw.cmd
            pom.xml
  External Libraries
  Scratches and Consoles

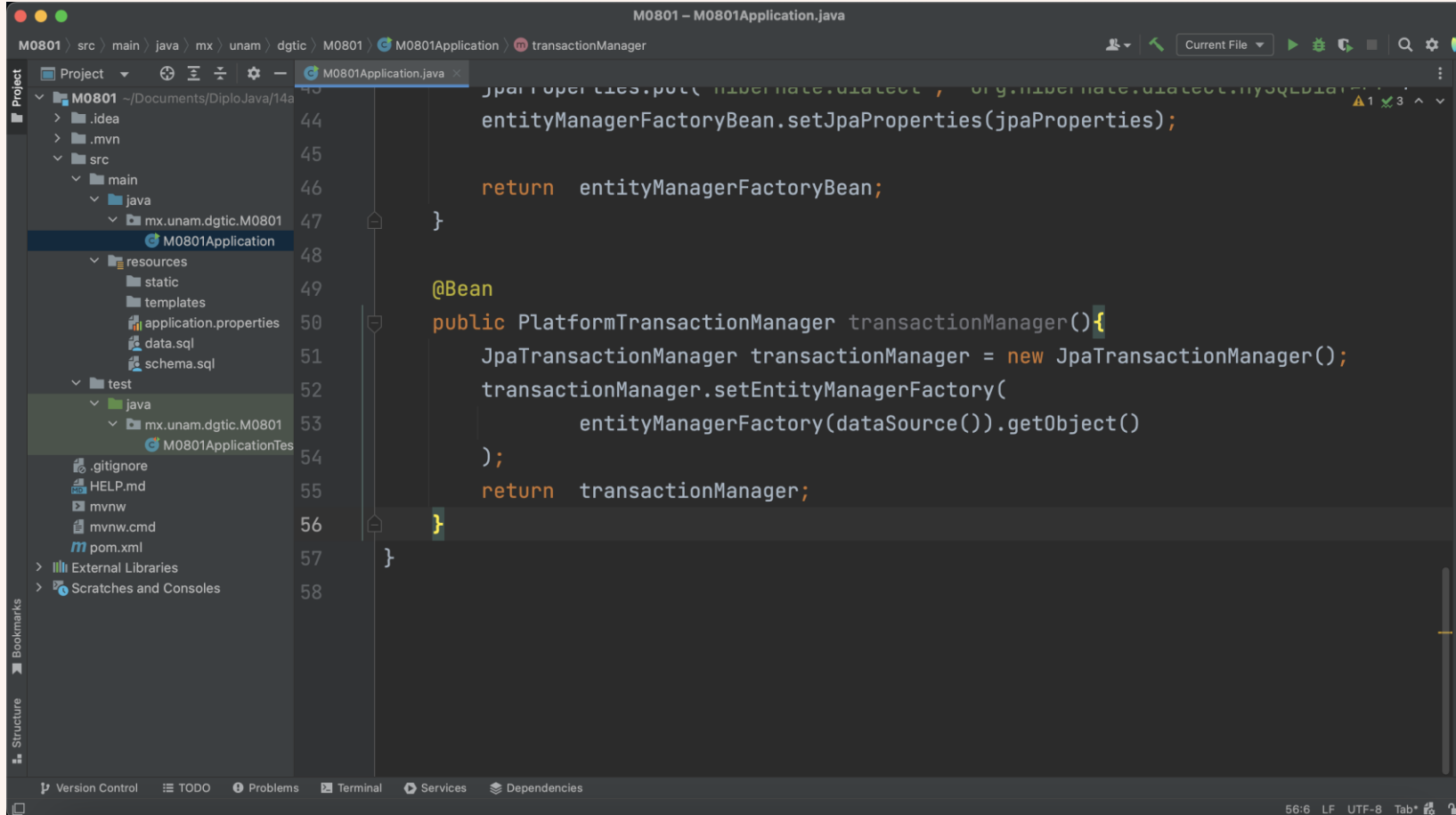
Structure
Bookmarks

M0801Application.java
33
34 @Bean
35 LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource dataSource){
36     LocalContainerEntityManagerFactoryBean entityManagerFactoryBean
37         = new LocalContainerEntityManagerFactoryBean();
38     entityManagerFactoryBean.setDataSource(dataSource);
39     entityManagerFactoryBean.setJpaVendorAdapter( new HibernateJpaVendorAdapter());
40     entityManagerFactoryBean.setPackagesToScan("mx.dgtic.unam.datos");
41
42     Properties jpaProperties = new Properties();
43     jpaProperties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
44     entityManagerFactoryBean.setJpaProperties(jpaProperties);
45
46     return entityManagerFactoryBean;
47 }
48
49

Maven
Notifications
4 3 ^ v

Version Control
TODO
Problems
Terminal
Services
Dependencies
47:6 LF UTF-8 Tab*
```

Agregar TransactionManager



```
M0801 - M0801Application.java
M0801 src > main > java > mx > unam > dgic > M0801 > M0801Application > transactionManager

Project
  M0801 -/Documents/DiploJava/14a
  > .idea
  > .mvn
  > src
  > main
  > java
  > mx.unam.dgic.M0801
  > M0801Application
  > resources
  > static
  > templates
  > application.properties
  > data.sql
  > schema.sql
  > test
  > java
  > mx.unam.dgic.M0801
  > M0801ApplicationTest
  > .gitignore
  > HELP.md
  > mvnw
  > mvnw.cmd
  > pom.xml
  > External Libraries
  > Scratches and Consoles

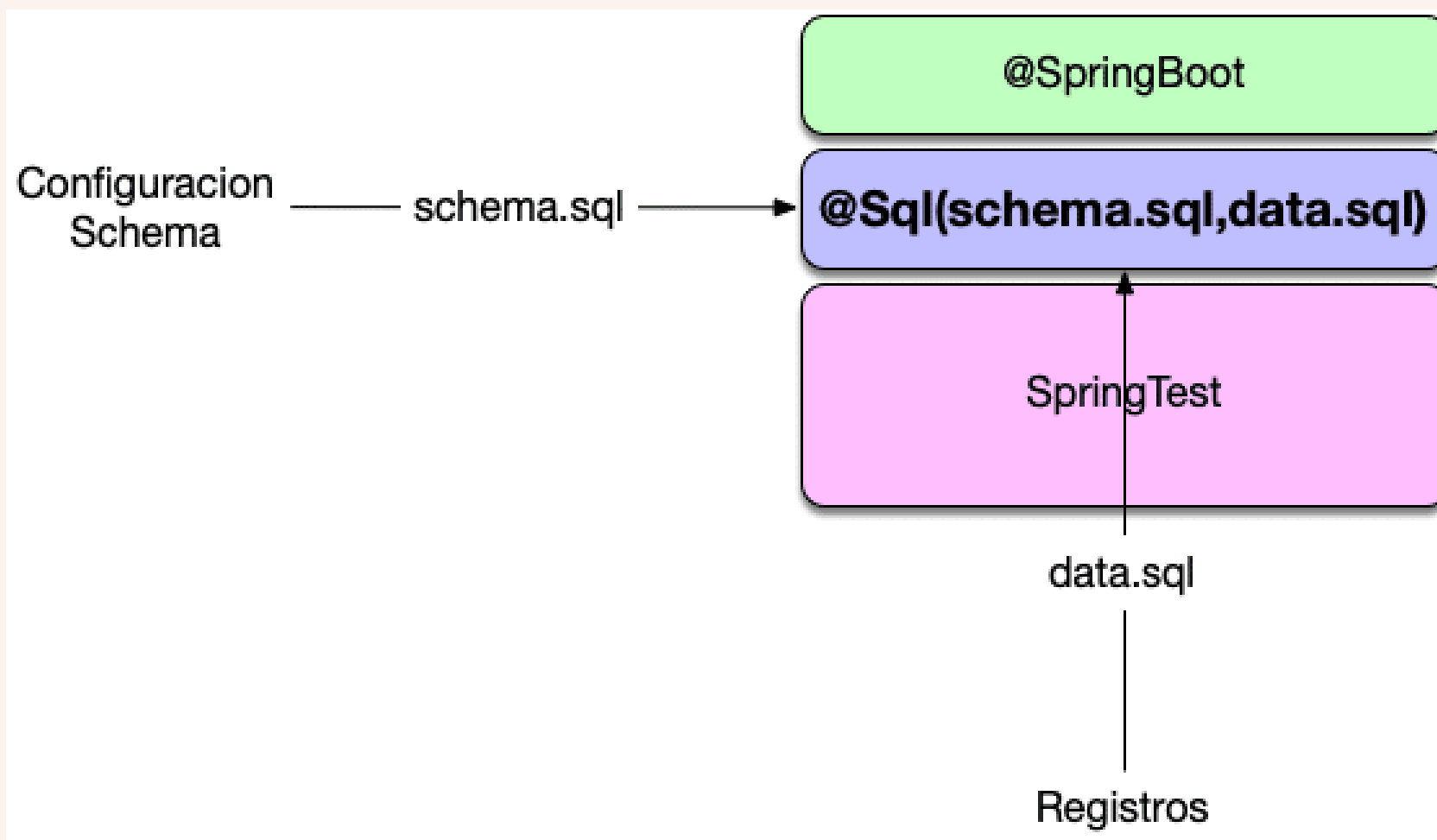
Structure
  Version Control
  TODO
  Problems
  Terminal
  Services
  Dependencies

M0801Application.java
44 jpaProperties.put("hibernate.dialect", "org.hibernate.dialect.MySQLDialect");
45 entityManagerFactoryBean.setJpaProperties(jpaProperties);
46
47 return entityManagerFactoryBean;
48 }
49
50 @Bean
51 public PlatformTransactionManager transactionManager(){
52     JpaTransactionManager transactionManager = new JpaTransactionManager();
53     transactionManager.setEntityManagerFactory(
54         entityManagerFactory(dataSource()).getObject()
55     );
56     return transactionManager;
57 }
58 }
```

@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



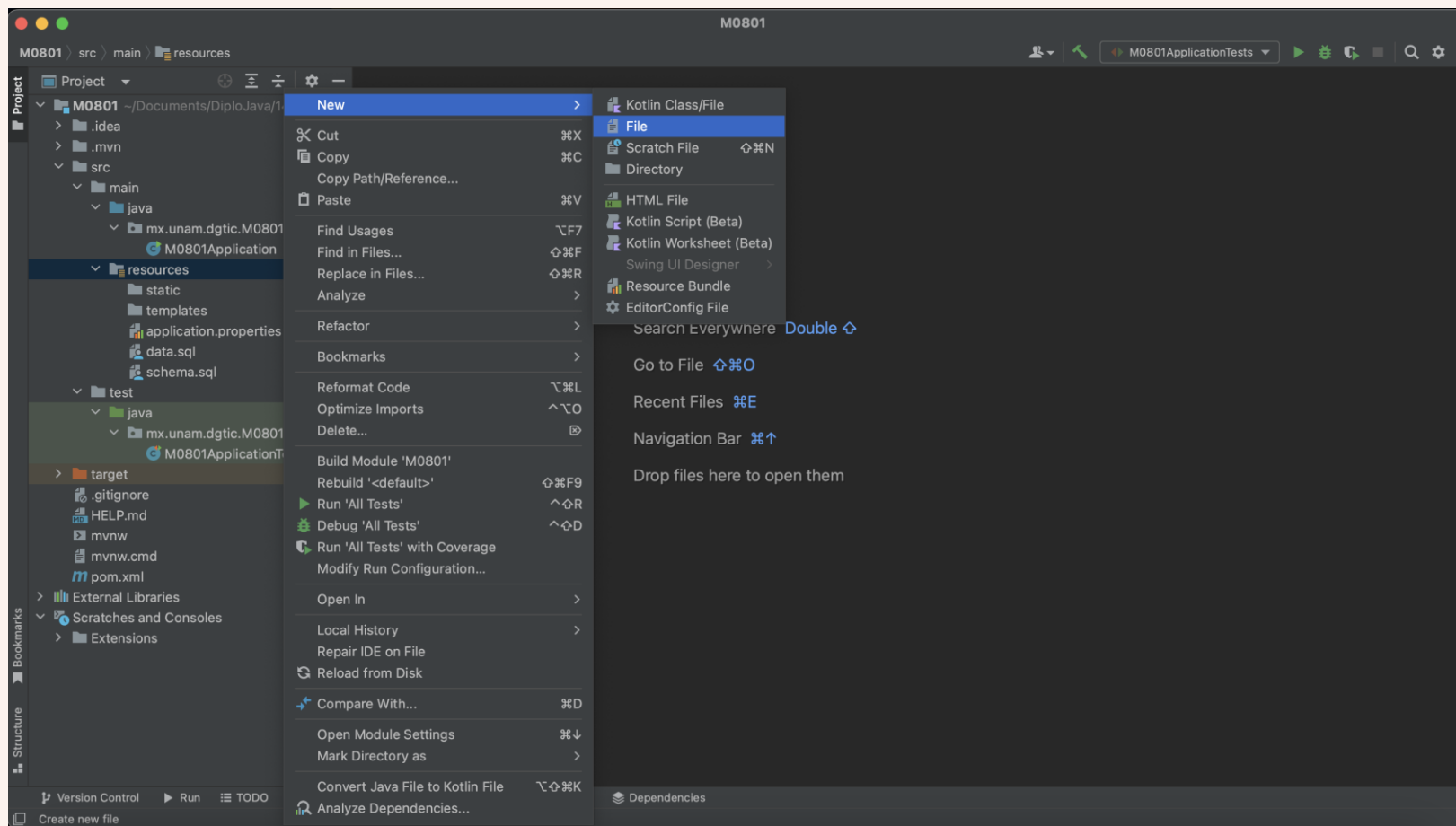
@Sql

```
@Sql(schema.sql, data.sql)
```

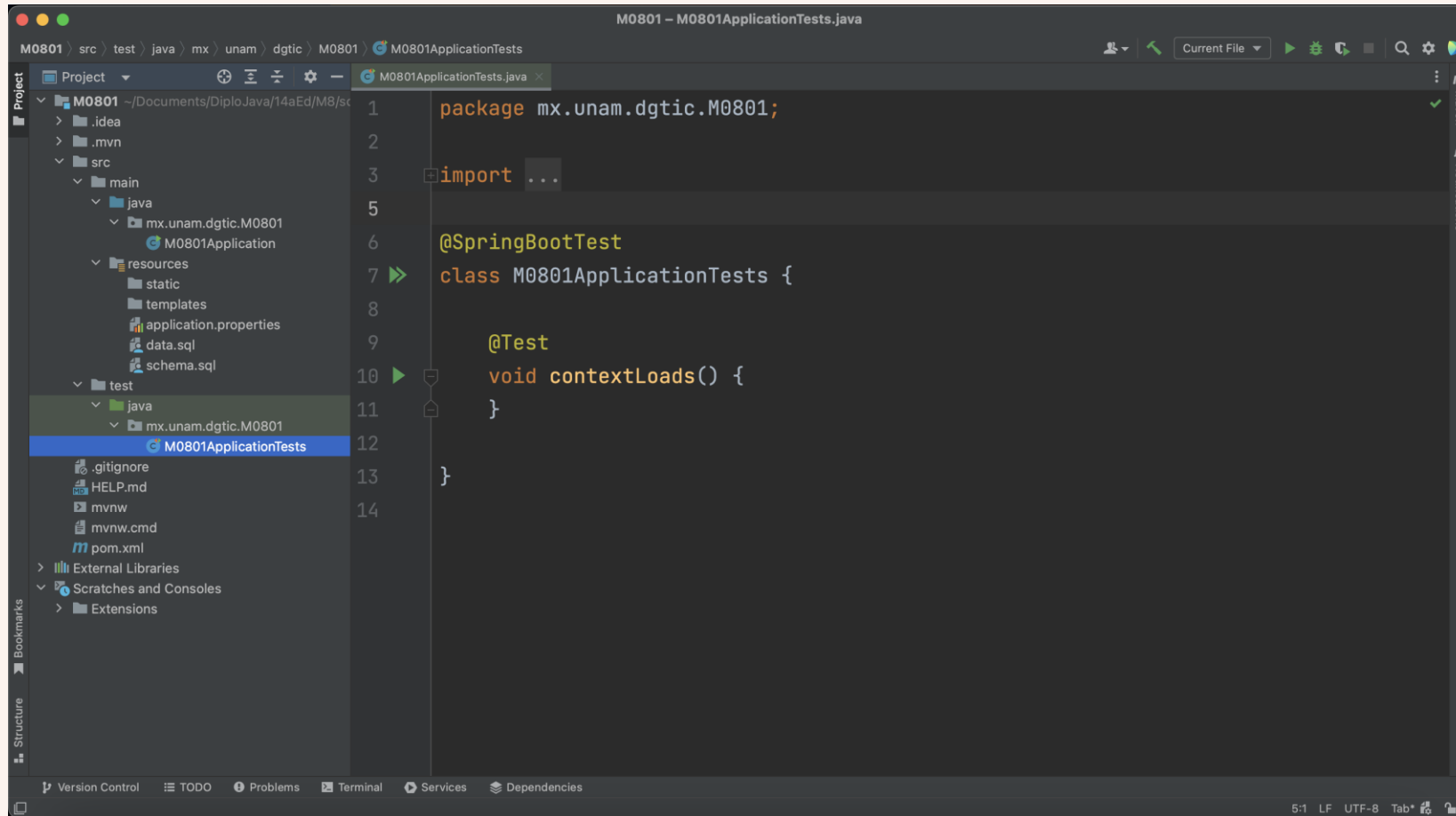
create

insert

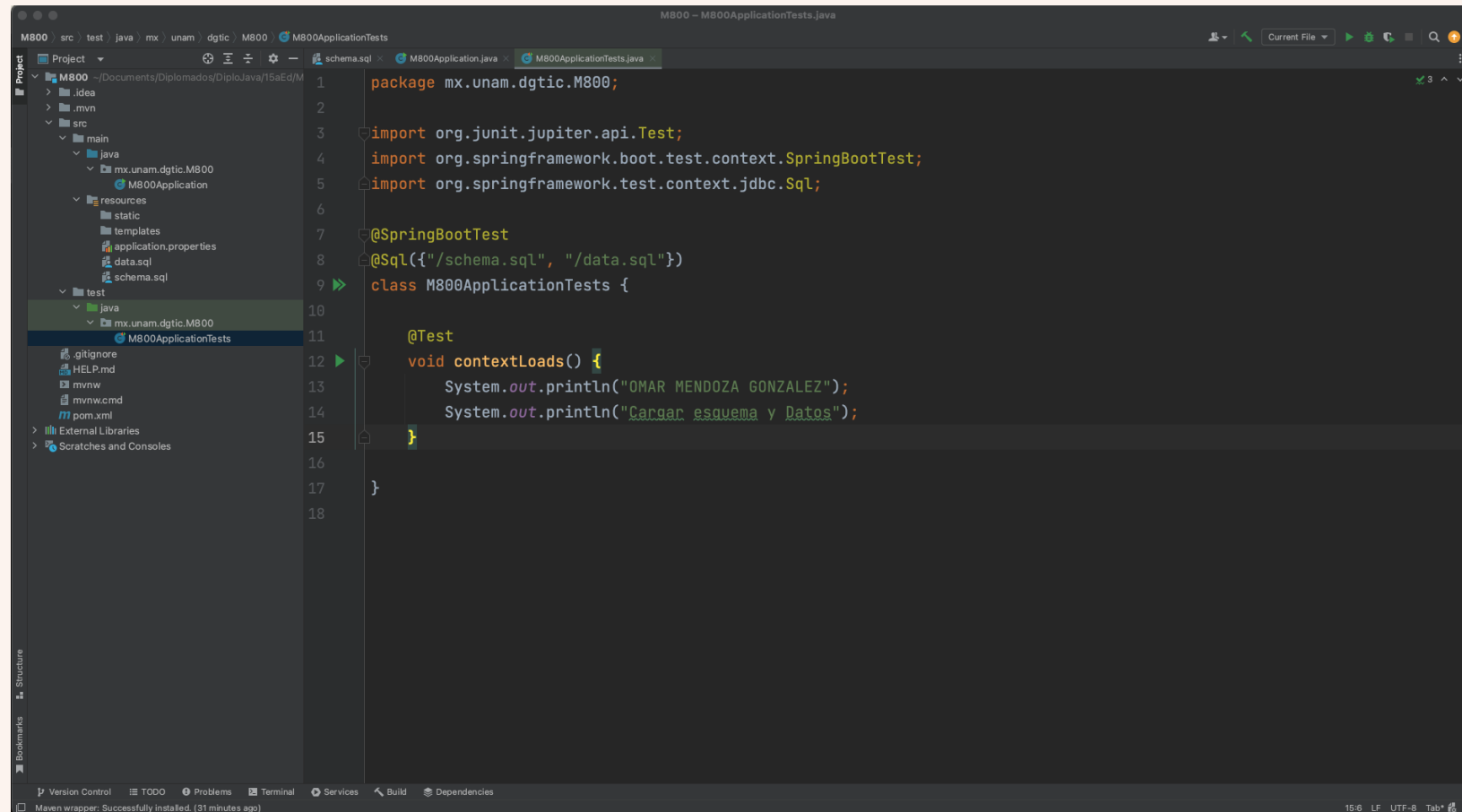
Agregar archivos sql



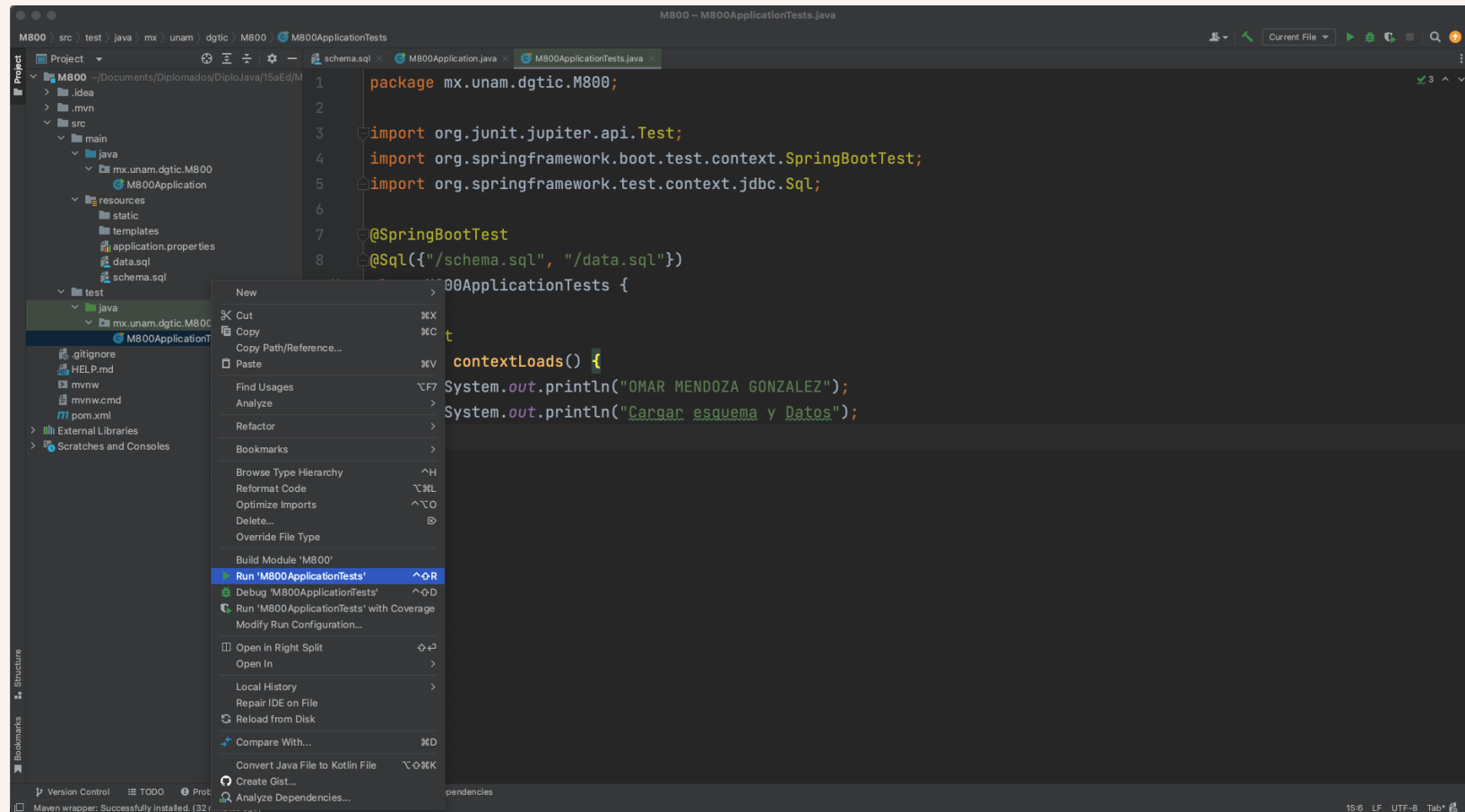
Agregar JUnitTest



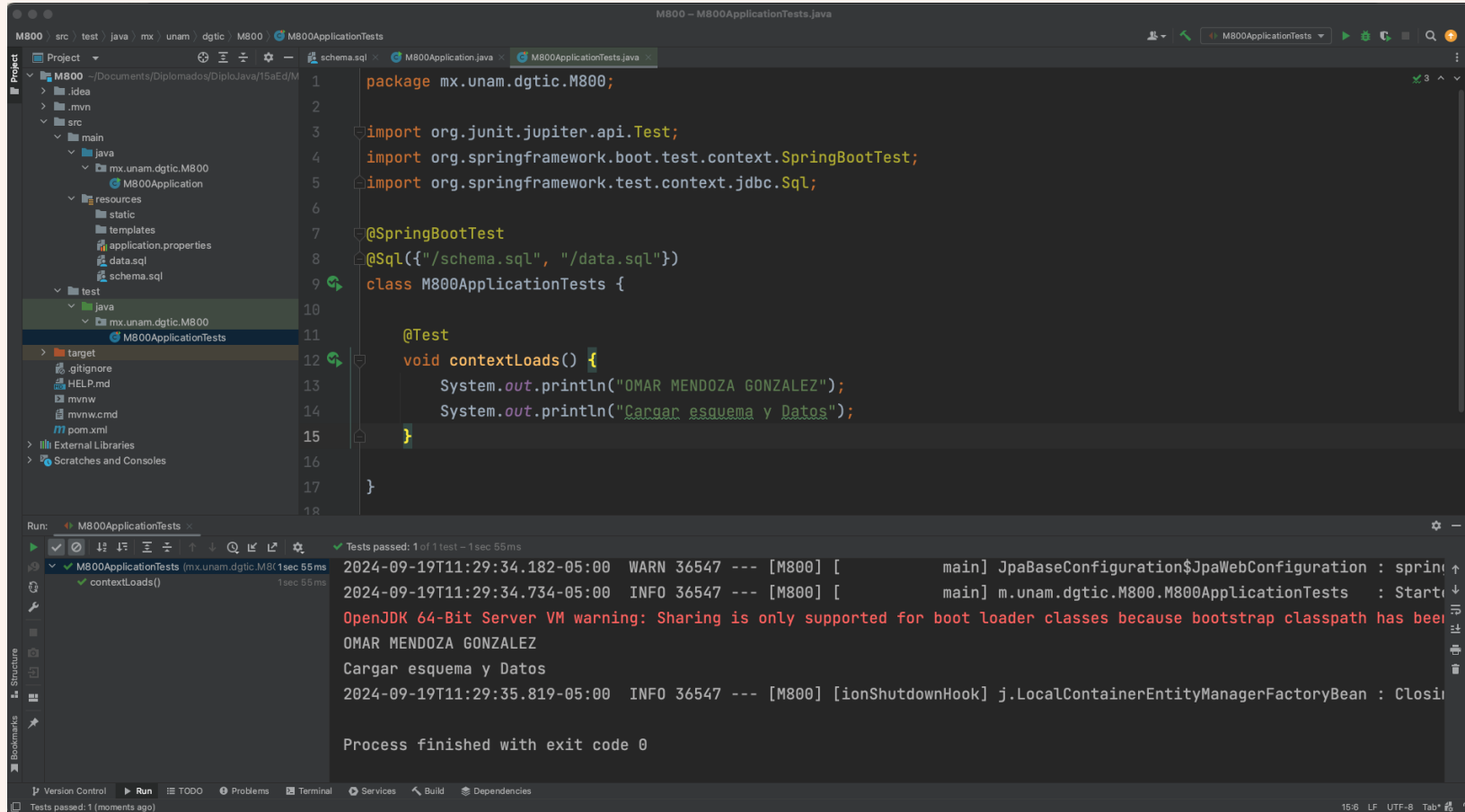
Definir datos para prueba



Ejecutar Test



Ejecutar Test



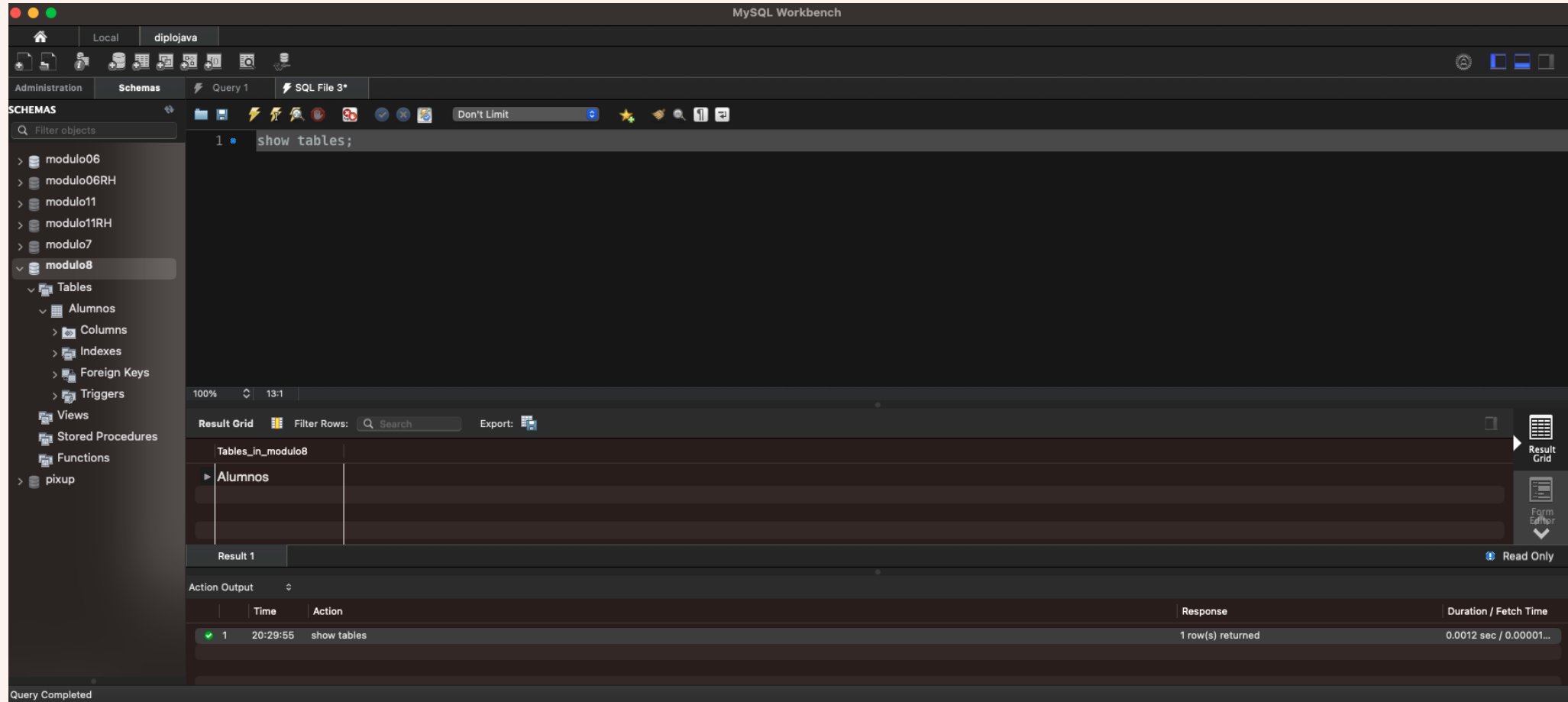
The screenshot shows an IDE window titled "M800 - M800ApplicationTests.java". The left sidebar displays a project structure with folders like "main", "resources", and "test". The main editor shows the following code:

```
1 package mx.unam.dgtic.M800;
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 M800ApplicationTests {
10
11     @Test
12     void contextLoads() {
13         System.out.println("OMAR MENDOZA GONZALEZ");
14         System.out.println("Cargar esquema y Datos");
15     }
16
17 }
```

The bottom panel shows the "Run" output for "M800ApplicationTests". The test "contextLoads()" passed in 1sec 55ms. The output log contains the following messages:

```
2024-09-19T11:29:34.182-05:00 WARN 36547 --- [M800] [main] JpaBaseConfiguration$JpaWebConfiguration : spring
2024-09-19T11:29:34.734-05:00 INFO 36547 --- [M800] [main] m.unam.dgtic.M800.M800ApplicationTests : Start
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been
OMAR MENDOZA GONZALEZ
Cargar esquema y Datos
2024-09-19T11:29:35.819-05:00 INFO 36547 --- [M800] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closi
Process finished with exit code 0
```

Validar la creación de la tabla



The screenshot shows the MySQL Workbench interface. The 'Schemas' sidebar on the left lists several schemas, with 'modulo8' selected. Under 'modulo8', the 'Tables' folder is expanded, showing a table named 'Alumnos'. The main query editor contains the SQL command 'show tables;'. The 'Result Grid' at the bottom displays the output of the query, showing a single row for the 'Alumnos' table. The 'Action Output' pane at the bottom shows the execution details of the query.

Time	Action	Response	Duration / Fetch Time
1	20:29:55 show tables	1 row(s) returned	0.0012 sec / 0.00001...

Contacto

Mtro. Víctor Manuel Sánchez Sánchez

victorsanchezh0@aragon.unam.mx