HIBERNATE IN INTELLIJ

LOPEZ SERRANO, PABLO
IES PERE MARIA I ORTS

Contenido

Introduction

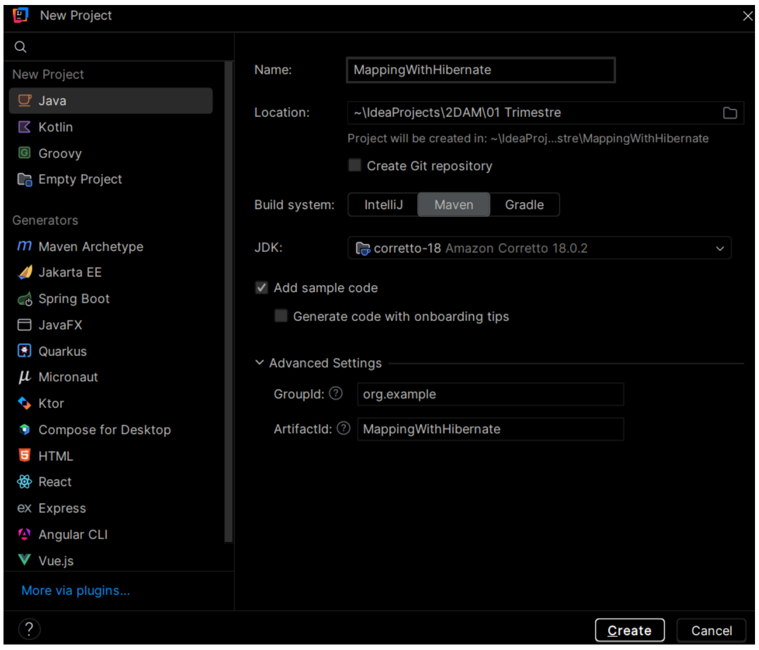
In this document, we will explain step by step how to set up and use Hibernate with IntelliJ IDEA Ultimate, specifically with MariaDB as the database. Hibernate is one of the most popular Java frameworks for object-relational mapping (ORM), enabling developers to interact with databases through Java objects. We will guide you through the necessary configuration, mapping entities to database tables, and demonstrate basic CRUD (Create, Read, Update, Delete) operations using MariaDB. Whether you are new to Hibernate or need a refresher, this guide will provide a clear and detailed path to successfully using Hibernate with MariaDB in your projects.

Create Database in MariaDB

First, you will have to create a DataBase, for instance, using MariaDB.

Creating a project in IntelliJ

Then you will create a Java project in IntelliJ with the Buidling system: **Maven.**



Name: MappingWithHibernate

Location: ~\IdeaProjects\2DAM\01 Trimestre

Project will be created in: ~\IdeaProj...stre\MappingWithHibernate

Create Git repository

Build system: IntelliJ [✓ Maven] Gradle

JDK: corretto-18 Amazon Corretto 18.0.2

[✓] Add sample code

Generate code with onboarding tips

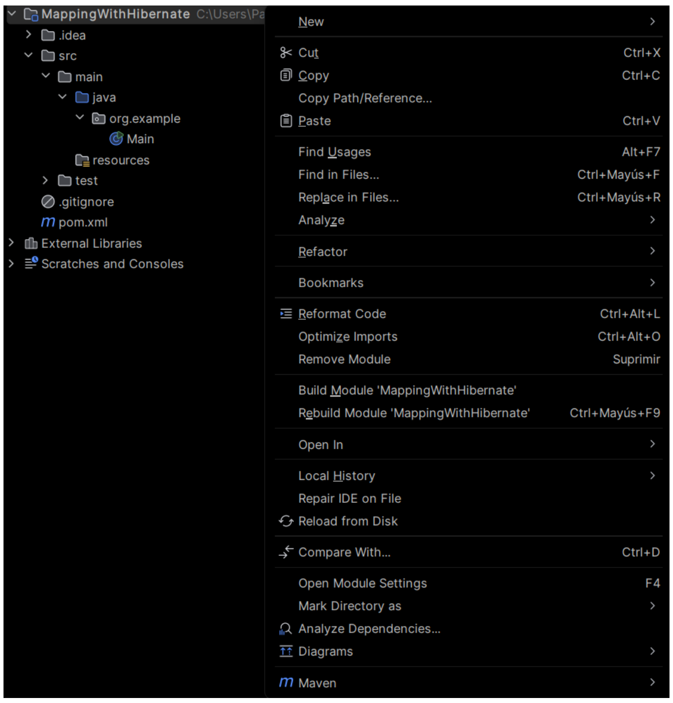
Advanced Settings

GroupId: org.example

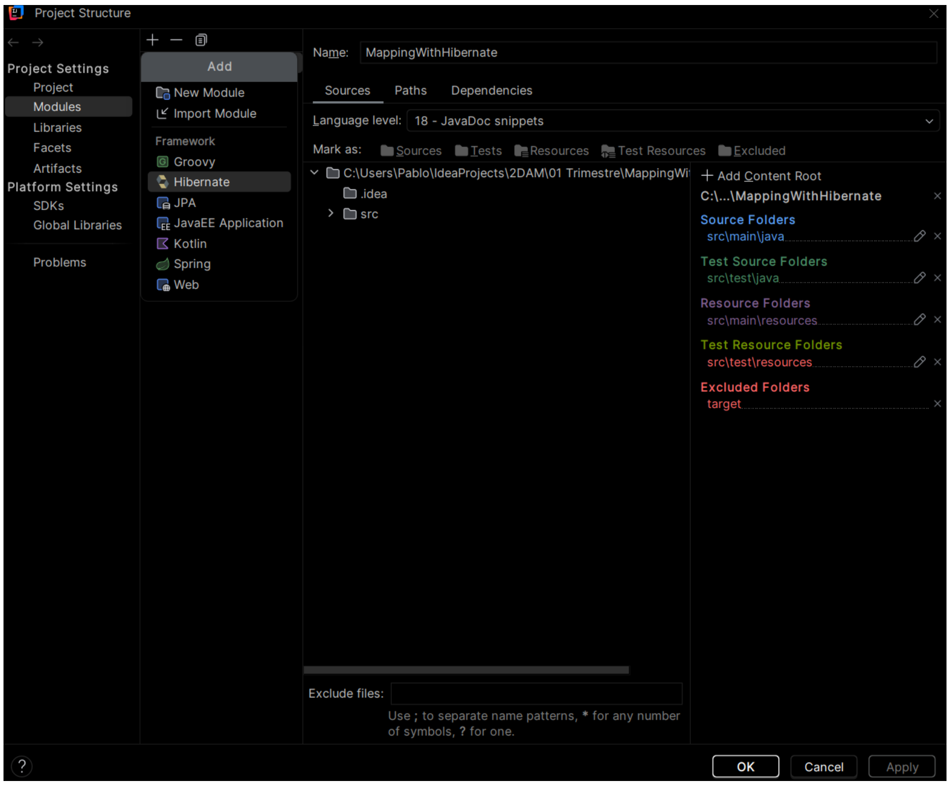
ArtifactId: MappingWithHibernate

Module Settings

Then we go into the Open Module Settings Menu (or press F4 after clicking the folders).



Add > Hibernate



Name: MappingWithHibernate

Language level: 18 - JavaDoc snippets

C:\Users\Pablo\IdeaProjects\2DAM\01 Trimestre\MappingWi
.idea
src

C:\...\MappingWithHibernate

Source Folders
src\main\java

Test Source Folders
src\test\java

Resource Folders
src\main\resources

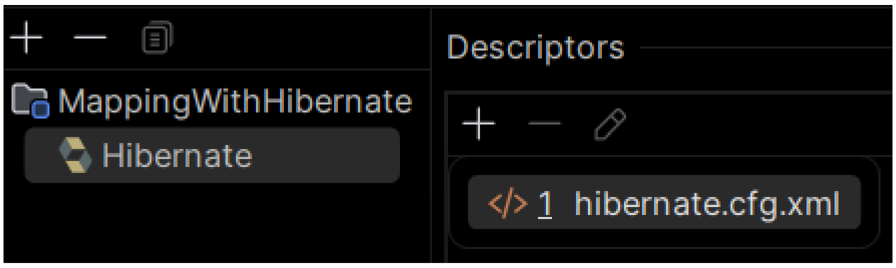
Test Resource Folders
src\test\resources

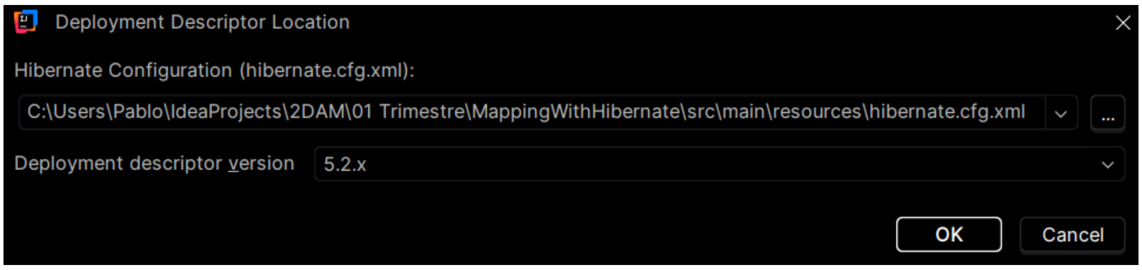
Excluded Folders
target

Exclude files:

Use ; to separate name patterns, * for any number of symbols, ? for one.

Once you have added Hibernate, you must add in the Descriptors side the hibernate.cfg.xml file in the resources folder that will be automatically created.



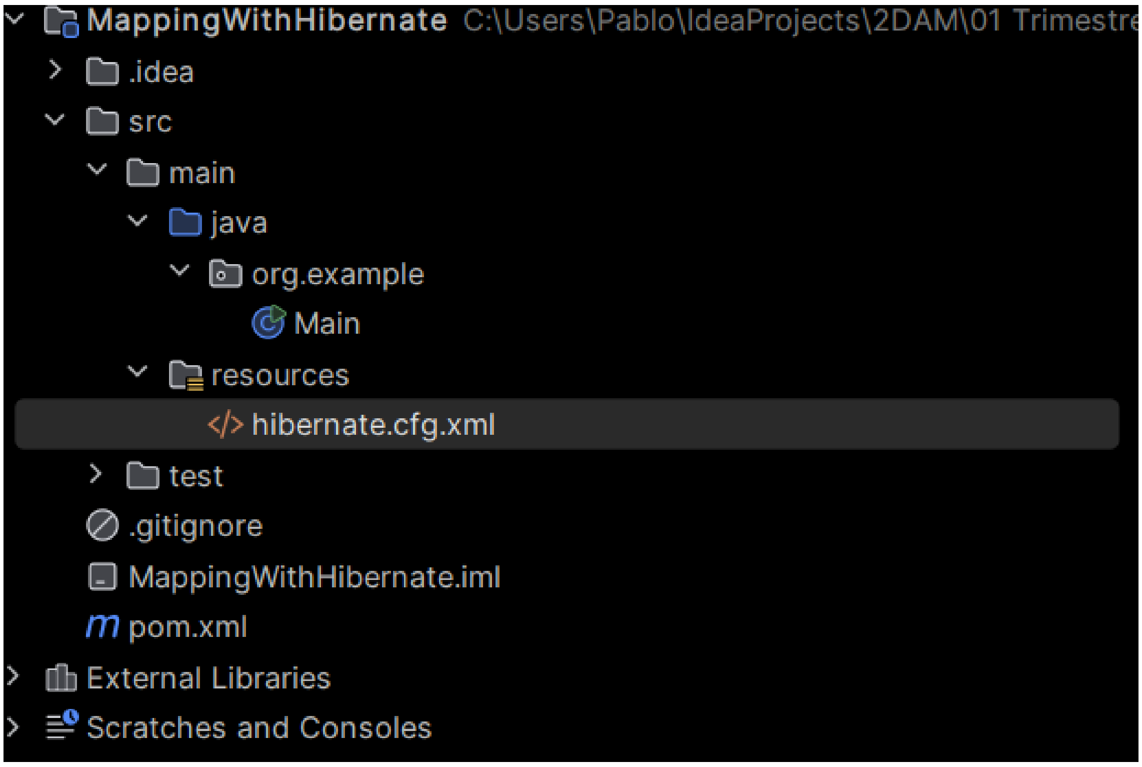


Hibernate Configuration (hibernate.cfg.xml):
C:\Users\Pablo\IdeaProjects\2DAM\01 Trimestre\MappingWithHibernate\src\main\resources\hibernate.cfg.xml

Deployment descriptor version 5.2.x

OK > Apply > OK.

Configuration of the hibernate.cfg.xml file



After following the tutorial, you will find your project looking like this. With the pom.xml and hibernate.cfg.xml files.

Your hibernate.cfg.xml file should look like this:

```
<?xml version="1.0" encoding="utf-8"?>

<!DOCTYPE hibernate-configuration PUBLIC

    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

  <session-factory>


    <!— Database connection configuration -->

    <property
name="hibernate.connection.driver_class">org.mariadb.jdbc.Driver</property>

    <property
name="hibernate.connection.url">jdbc:mariadb://localhost:3306/your_database_name</
property>

    <property name="hibernate.connection.username">your_username</property>

    <property name="hibernate.connection.password">your_password</property>
    <!-- SQL  dialect for MariaDB -->

    <property
name="hibernate.dialect">org.hibernate.dialect.MariaDBDialect</property>


    <!—Create/Update the database schemas -->

    <property name="hibernate.hbm2ddl.auto">update</property>


    <!—Show SQL queries in console -->

    <property name="hibernate.show_sql">true</property>

    <property name="hibernate.format_sql">true</property>


    <!—Entity mapped class -->

    <mapping class="com.example.YourEntityClass"/>


  </session-factory>

</hibernate-configuration>
```

With the replaced information for your DataBase.

Configuration of the pom.xml file

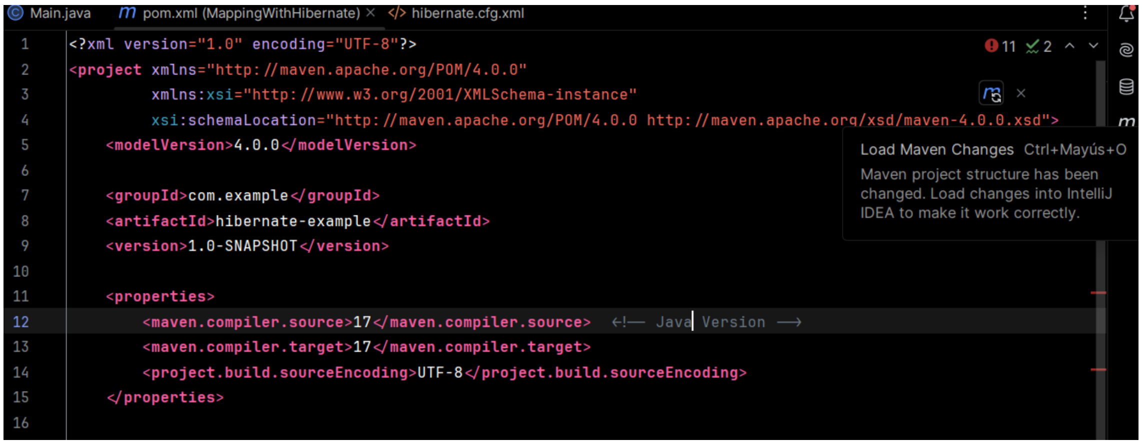
Then you replace your current file with:

```
<?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
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>
  <artifactId>hibernate-example</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>  <!-- Java Version -->
```

```xml
    <maven.compiler.target>17</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <!-- Hibernate Core -->
    <dependency>
      <groupId>org.hibernate.orm</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>6.2.12.Final</version>
    </dependency>

    <!-- MariaDB JDBC Driver -->
    <dependency>
      <groupId>org.mariadb.jdbc</groupId>
      <artifactId>mariadb-java-client</artifactId>
      <version>3.1.4</version>
    </dependency>

    <!-- JPA API -->
    <dependency>
      <groupId>jakarta.persistence</groupId>
      <artifactId>jakarta.persistence-api</artifactId>
      <version>3.1.0</version>
    </dependency>

    <!-- C3P0 Connection Pool (opcional, pero recomendado) -->
    <dependency>
      <groupId>com.mchange</groupId>
      <artifactId>c3p0</artifactId>
      <version>0.9.5.5</version>
    </dependency>

    <!-- Logging (SLF4J with Logback) -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-api</artifactId>
      <version>2.0.9</version>
    </dependency>
    <dependency>
      <groupId>ch.qos.logback</groupId>
      <artifactId>logback-classic</artifactId>
      <version>1.4.11</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
```

```
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.10.1</version>
        <configuration>
          <source>17</source>
          <target>17</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

After that you must Load the Maven Changes.



```
<?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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.example</groupId>
    <artifactId>hibernate-example</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>17</maven.compiler.source>  <!-- Java Version -->
        <maven.compiler.target>17</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

```

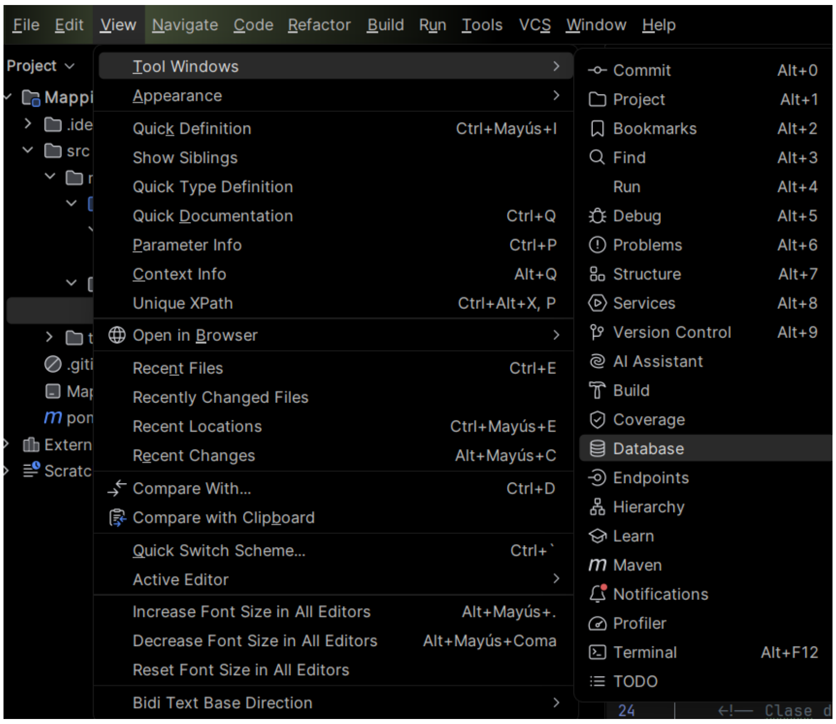
Load Maven Changes Ctrl+Mayús+O

Maven project structure has been changed. Load changes into IntelliJ IDEA to make it work correctly.

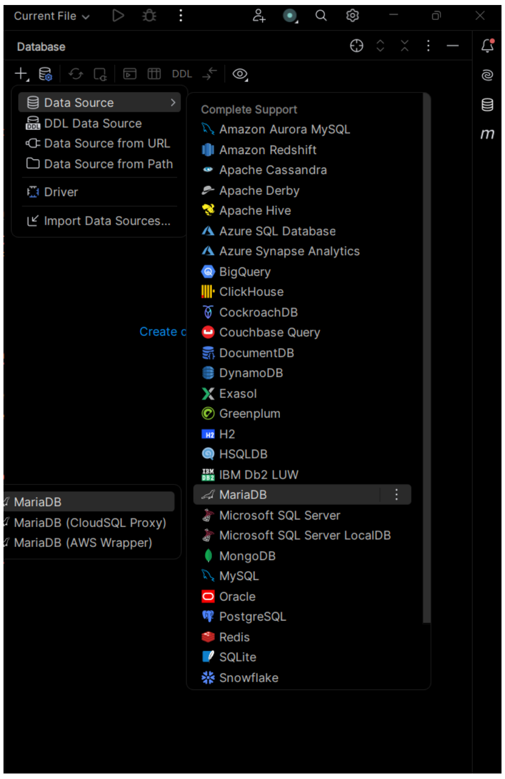
Once you have done that, the errors that appeared will disappear (it doesn’t matter if you still have errors at hibernate.cfg.xml).

Database Connection

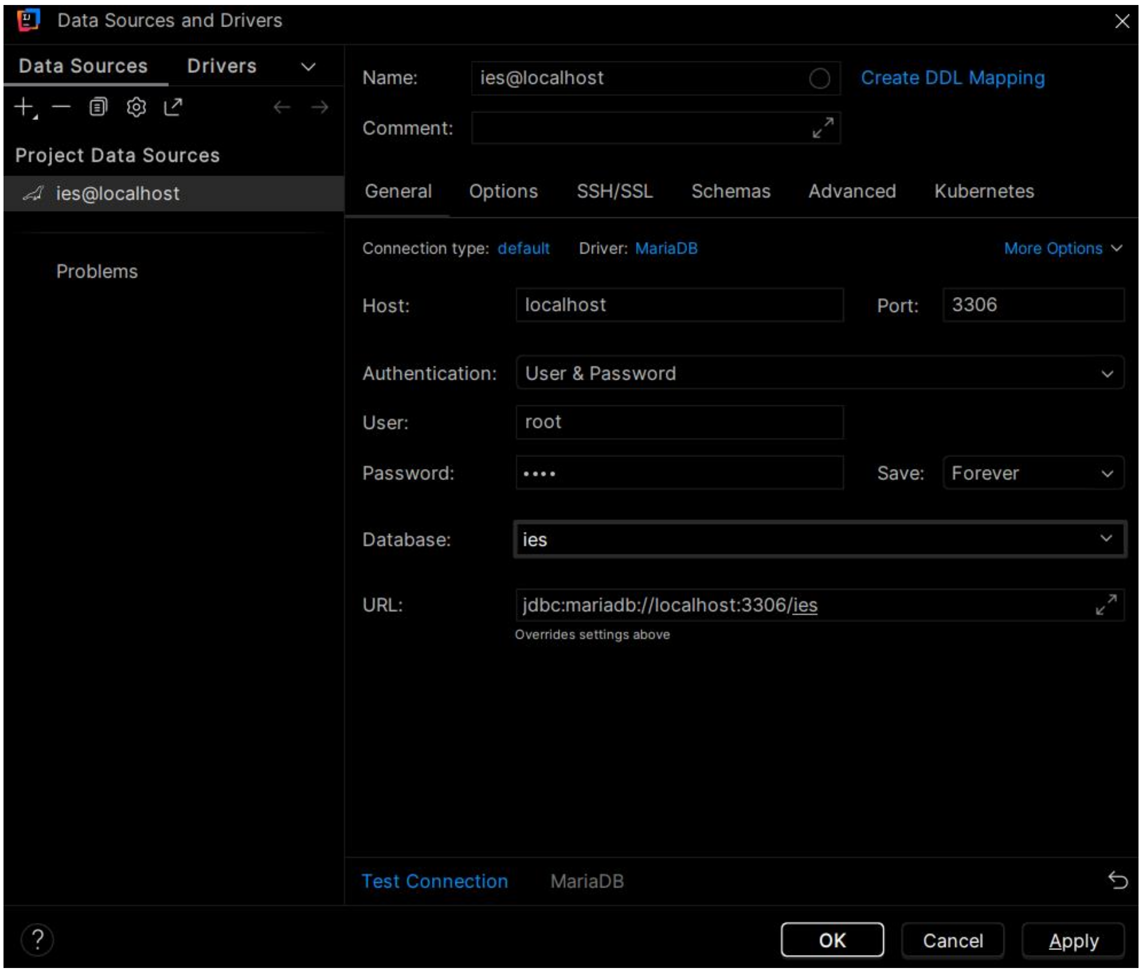
View > Tool Windows > Database



And after this you choose your Database:



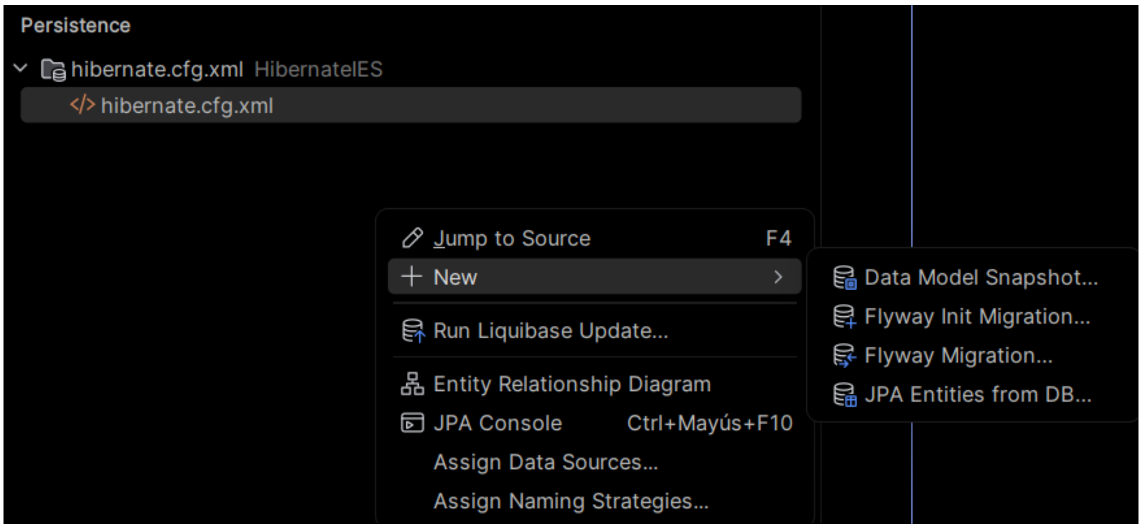
You fill in the information for accessing your database and press OK.



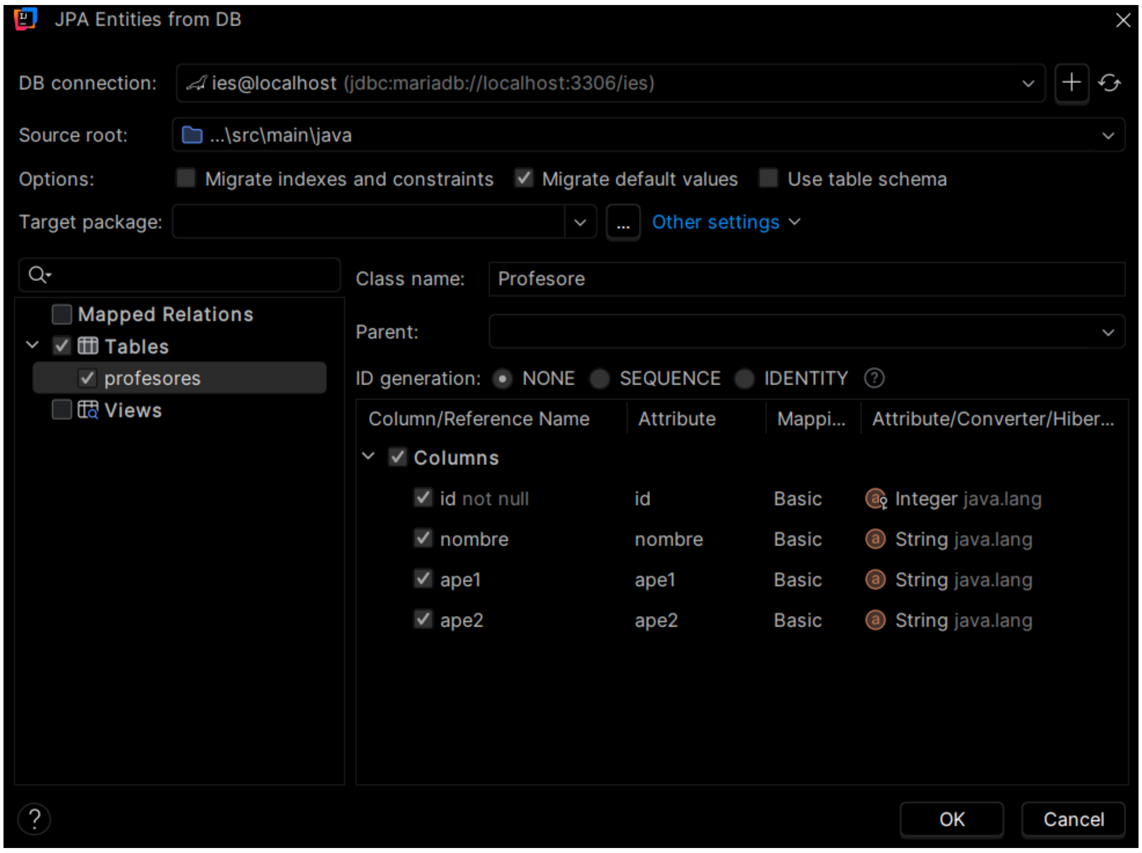
Name: ies@localhost
Comment:
Connection type: default Driver: MariaDB
Host: localhost Port: 3306
Authentication: User & Password
User: root
Password: ••••
Save: Forever
Database: ies
URL: jdbc:mariadb://localhost:3306/ies

Persistence Menu

Now you must press View > Tool Windows > Persistence



Press JPA Entities from DB



DB connection: ies@localhost (jdbc:mariadb://localhost:3306/ies)

Source root: ...\src\main\java

Options: [ ] Migrate indexes and constraints [✓] Migrate default values [ ] Use table schema

Target package:

Other settings

[ ] Mapped Relations
[✓] Tables
  [✓] profesores
[ ] Views

Class name: Profesore

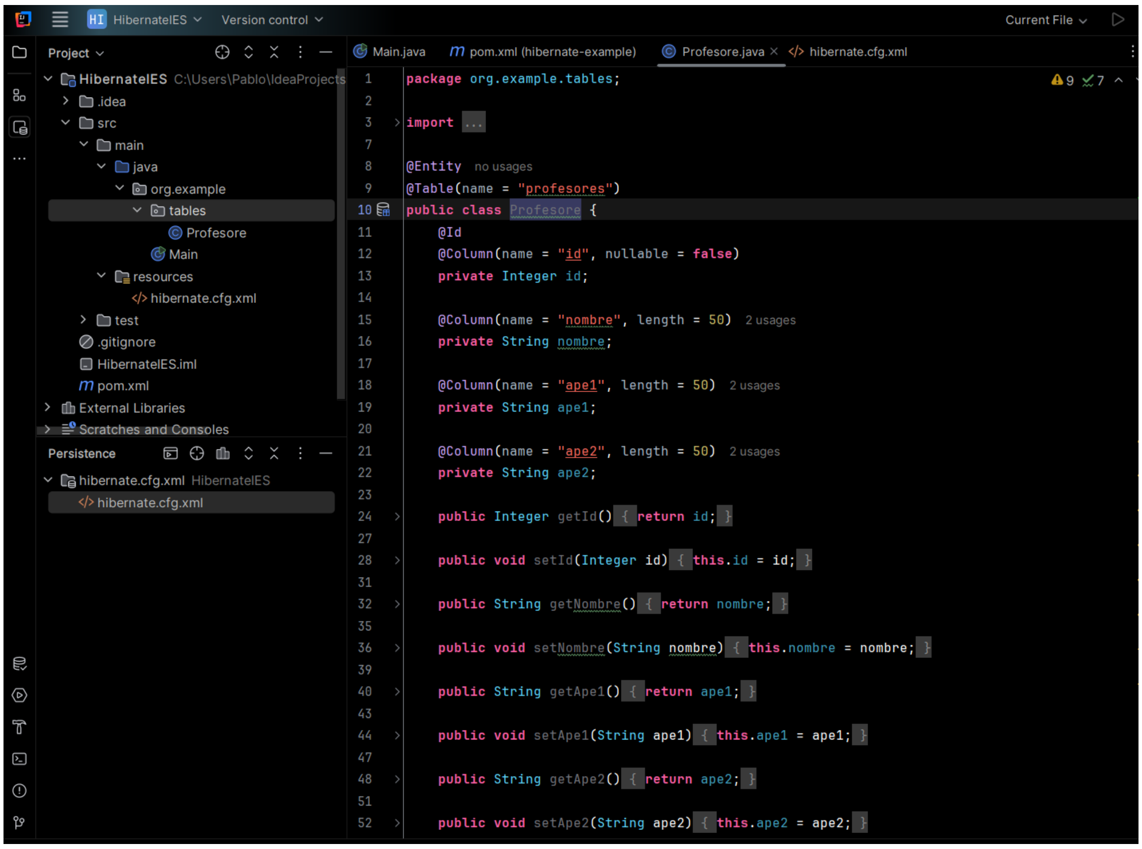
Parent:

ID generation: [✓ NONE] SEQUENCE IDENTITY

Column/Reference Name	Attribute	Mappi...	Attribute/Converter/Hiber...
[✓] Columns			
[✓] id not null	id	Basic	Integer java.lang
[✓] nombre	nombre	Basic	String java.lang
[✓] ape1	ape1	Basic	String java.lang
[✓] ape2	ape2	Basic	String java.lang

And here you can Map whichever table from your database.

And that is it, that's how you can map a DDL in Java using Hibernate.



```java
package org.example.tables;

import ...

@Entity  no usages
@Table(name = "profesores")
public class Profesore {
    @Id
    @Column(name = "id", nullable = false)
    private Integer id;

    @Column(name = "nombre", length = 50)  2 usages
    private String nombre;

    @Column(name = "ape1", length = 50)  2 usages
    private String ape1;

    @Column(name = "ape2", length = 50)  2 usages
    private String ape2;

    public Integer getId() { return id; }

    public void setId(Integer id) { this.id = id; }

    public String getNombre() { return nombre; }

    public void setNombre(String nombre) { this.nombre = nombre; }

    public String getApe1() { return ape1; }

    public void setApe1(String ape1) { this.ape1 = ape1; }

    public String getApe2() { return ape2; }

    public void setApe2(String ape2) { this.ape2 = ape2; }
```