Actividad 2 UT 3 - Actividad Relaciones Muchos a Muchos

Índice

Enunciado

Actividad 2 UT 3 - Actividad Relación muchos a muchos
En esta actividad aplicarás los conceptos de relaciones muchos a muchos con Hibernate en un contexto de pilotos y escuderías de Fórmula 1.
Cada piloto puede haber competido para varias escuderías a lo largo de su carrera.
A su vez, cada escudería puede haber tenido varios pilotos en diferentes temporadas.
Esto se modelará con una relación N:M entre Piloto y Escuderia.
Tablas del esquema
Tabla: pilotos
❖ id_piloto : Identificador del piloto (INT)
❖ nombre : Nombre completo del piloto (VARCHAR)
❖ nacionalidad : País del piloto (VARCHAR)
Tabla: escuderias
❖ id_escuderia : Identificador de la escudería (INT)
❖ nombre : Nombre de la escudería (VARCHAR)
❖ sede : Ciudad o país donde se encuentra la sede (VARCHAR)
Tabla intermedia: piloto_escuderia
❖ piloto_id : Referencia al piloto (INT)
❖ escuderia_id : Referencia a la escudería (INT)
Relaciones
● Un piloto puede haber competido para muchas escuderías.
● Una escudería puede tener muchos pilotos asociados.
● Hibernate generará automáticamente la tabla intermedia piloto_escuderia.
Tareas
1. Definir las entidades Piloto y Escuderia, implementando la relación N:M con @ManyToMany y @JoinTable.
2. Configurar hibernate.cfg.xml para mapear ambas clases.
3. Implementar los DAOs PilotoDAO y EscuderiaDAO con sus métodos CRUD correspondientes .
 - En PilotoDAO añadir un método específico para listar escuderías de un piloto.
 - En EscuderiaDAO añadir un método específico para listar pilotos de una escudería.
4. En el programa principal (Main):
 - Permitir crear pilotos y escuderías.
Asignar pilotos a escuderías y viceversa.
 - Listar todos los pilotos que pertenecen a una escudería.
 - Listar todas las escuderías donde ha corrido un piloto.
 - Mostrar todos los pilotos de una nacionalidad concreta que hayan corrido en una escudería específica (ambos valores se suministran por teclado)
 - Mostrar todas las escuderías donde haya corrido un piloto con un nombre suministrado por teclado.

Explicación de las secciones del código

Main.java

```
import modelo.*;
import dao.*;
import util.HibernateUtil;
import org.hibernate.Session;

import java.util.List;
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);
       PilotoDAO pilotoDAO = new PilotoDAO();
       EscuderiaDAO escuderiaDAO = new EscuderiaDAO();

       while (true) {
           System.out.println("\n--- MENÚ FÓRMULA 1 ---");
           System.out.println("1. Crear piloto");
           System.out.println("2. Crear escudería");
           System.out.println("3. Asignar piloto a escudería");
           System.out.println("4. Mostrar pilotos de una
escudería");
           System.out.println("5. Mostrar escuderías donde corrió
un piloto");
           System.out.println("6. Pilotos de una nacionalidad en
una escudería");
           System.out.println("7. Escuderías donde corrió un
piloto por nombre");
           System.out.println("0. Salir");
           System.out.print("Opción: ");

           int op = Integer.parseInt(sc.nextLine());

           switch (op) {

               case 1 -> {
                   System.out.print("Nombre piloto: ");
                   String n = sc.nextLine();
                   System.out.print("Nacionalidad: ");
                   String nac = sc.nextLine();

                   pilotoDAO.guardar(new Piloto(n, nac));
```

```
                }

                case 2 -> {
                    System.out.print("Nombre escudería: ");
                    String n = sc.nextLine();
                    System.out.print("Sede: ");
                    String s = sc.nextLine();

                    escuderiaDAO.guardar(new Escuderia(n, s));
                }

                case 3 -> {
                    System.out.print("ID Piloto: ");
                    Long idP = Long.parseLong(sc.nextLine());
                    System.out.print("ID Escudería: ");
                    Long idE = Long.parseLong(sc.nextLine());

                    pilotoDAO.asignarEscuderia(idP, idE);
                    System.out.println("Asignado correctamente.");
                }

                case 4 -> {
                    System.out.print("ID Escudería: ");
                    Long id = Long.parseLong(sc.nextLine());

                    List<Piloto> pilotos =
escuderiaDAO.getPilotosDeEscuderia(id);
                    pilotos.forEach(System.out::println);
                }

                case 5 -> {
                    System.out.print("ID Piloto: ");
                    Long id = Long.parseLong(sc.nextLine());

                    List<Escuderia> esc =
pilotoDAO.getEscuderiasDePiloto(id);
                    esc.forEach(System.out::println);
                }

                case 6 -> {
                    System.out.print("Nacionalidad: ");
                    String nac = sc.nextLine();
                    System.out.print("Nombre escudería: ");
                    String escu = sc.nextLine();

                    Session session =
HibernateUtil.getSessionFactory().openSession();
                    List<Piloto> lista = session.createQuery(
```

```
                                    "SELECT p FROM Piloto p JOIN
p.escuderias e " +
                                            "WHERE p.nacionalidad =
:n AND e.nombre = :en",
                                    Piloto.class)
                            .setParameter("n", nac)
                            .setParameter("en", escu)
                            .list();
                    session.close();

                    lista.forEach(System.out::println);
                }

                case 7 -> {
                    System.out.print("Nombre piloto: ");
                    String nombre = sc.nextLine();

                    Session session =
HibernateUtil.getSessionFactory().openSession();
                    List<Escuderia> lista = session.createQuery(
                                    "SELECT e FROM Escuderia e JOIN
e.pilotos p " +
                                            "WHERE p.nombre = :n",
                                    Escuderia.class)
                            .setParameter("n", nombre)
                            .list();
                    session.close();

                    lista.forEach(System.out::println);
                }

                case 0 -> {
                    System.out.println("Saliendo...");
                    HibernateUtil.getSessionFactory().close();
                    return;
                }
            }
        }
    }
}
```

El Main controla toda la interacción con el usuario, usando un menú que llama a los DAOs y ejecuta consultas HQL.

Imports
import modelo.*;
import dao.*;

```
import util.HibernateUtil;
import org.hibernate.Session;
import java.util.List;
import java.util.Scanner;
```

Qué hace cada uno:

modelo.* -> Importa las entidades Piloto y Escuderia.

dao.* -> Importa PilotoDAO y EscuderiaDAO para usar CRUD.

HibernateUtil -> Permite obtener sesiones a la BD.

Session -> Se usa para consultas HQL directas.

Scanner -> Lee datos del teclado.

List -> Para devolver listas de entidades.

Creación de Scanner y DAOs

```
Scanner sc = new Scanner(System.in);
PilotoDAO pilotoDAO = new PilotoDAO();
EscuderiaDAO escuderiaDAO = new EscuderiaDAO();
```

Scanner permite leer opciones por teclado.

Cada DAO contiene los métodos CRUD y consultas.

Menú principal

Se usa un bucle infinito while (true) hasta que el usuario pulse 0.
Cada opción ejecuta un case del switch.

Opción 1 - Crear piloto

```
String n = sc.nextLine();
String nac = sc.nextLine();
pilotoDAO.guardar(new Piloto(n, nac));
```

Crea un nuevo objeto Piloto.

Llama al DAO -> inserta en BD.

Opción 2 - Crear escudería

```
escuderiaDAO.guardar(new Escuderia(n, s));
```

Similar al caso anterior.

Opción 3 - Asignar piloto a escudería (relación N:M)
pilotoDAO.asignarEscuderia(idP, idE);

El DAO realiza:

Recuperar piloto y escudería.

Agregar la escudería al Set del piloto.

Agregar el piloto al Set de la escudería.

Guardar.

Esto inserta un registro en la tabla piloto_escuderia.

Opción 4 - Mostrar pilotos de una escudería
List<Piloto> pilotos = escuderiaDAO.getPilotosDeEscuderia(id);
pilotos.forEach(System.out::println);

Usa HQL:

SELECT p FROM Escuderia e JOIN e.pilotos p WHERE e.id = :id

Opción 5 - Mostrar escuderías donde corrió un piloto
List<Escuderia> esc = pilotoDAO.getEscuderiasDePiloto(id);

HQL:

SELECT e FROM Piloto p JOIN p.escuderias e WHERE p.id = :id

Opción 6 - Pilotos de una nacionalidad que hayan corrido en una escudería

Consulta HQL avanzada:

SELECT p FROM Piloto p
JOIN p.escuderias e
WHERE p.nacionalidad = :n
AND e.nombre = :en

Filtra por dos tablas relacionadas.

Opción 7 - Escuderías donde corrió un piloto por nombre
SELECT e FROM Escuderia e JOIN e.pilotos p WHERE p.nombre = :n

Opción 0 - Salir

Cierra el menú y SessionFactory

Piloto.java

```
package modelo;

import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "pilotos")
public class Piloto {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id_piloto")
   private Long id;

   private String nombre;
   private String nacionalidad;

   @ManyToMany
   @JoinTable(
           name = "piloto_escuderia",
           joinColumns = @JoinColumn(name = "piloto_id"),
           inverseJoinColumns = @JoinColumn(name = "escuderia_id")
   )
   private Set<Escuderia> escuderias = new HashSet<>();

   public Piloto() {}

   public Piloto(String nombre, String nacionalidad) {
       this.nombre = nombre;
       this.nacionalidad = nacionalidad;
   }

   public Long getId() {
       return id;
   }
```

```
    public String getNombre() {
        return nombre;
    }

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

    public String getNacionalidad() {
        return nacionalidad;
    }

    public void setNacionalidad(String nacionalidad) {
        this.nacionalidad = nacionalidad;
    }

    public Set<Escuderia> getEscuderias() {
        return escuderias;
    }

    public void setEscuderias(Set<Escuderia> escuderias) {
        this.escuderias = escuderias;
    }

    @Override
    public String toString() {
        return "Piloto{" + "id=" + id + ", nombre='" + nombre +
'\'' +
                ", nacionalidad='" + nacionalidad + '\'' + '}';
    }
}
```

Anotación de entidad
@Entity
@Table(name = "pilotos")

Hibernate la mapea con la tabla pilotos.

ID autoincremental
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
@Column(name = "id_piloto")
private Long id;

Atributos simples
private String nombre;

```
private String nacionalidad;

Relación N:M con Escuderia
@ManyToMany
@JoinTable(
   name = "piloto_escuderia",
   joinColumns = @JoinColumn(name = "piloto_id"),
   inverseJoinColumns = @JoinColumn(name = "escuderia_id")
)
private Set<Escuderia> escuderias = new HashSet<>();
```

Explicación:

@ManyToMany -> relación N:M.

@JoinTable -> Hibernate genera tabla intermedia.

joinColumns -> columna que referencia al piloto.

inverseJoinColumns -> columna que referencia a la escudería.

Se usa HashSet para evitar duplicados.

Escuderia.java

```java
package modelo;

import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "escuderias")
public class Escuderia {

   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   @Column(name = "id_escuderia")
   private Long id;

   private String nombre;
   private String sede;

   @ManyToMany(mappedBy = "escuderias")
   private Set<Piloto> pilotos = new HashSet<>();

   public Escuderia() {}
```

```
    public Escuderia(String nombre, String sede) {
        this.nombre = nombre;
        this.sede = sede;
    }

    public Long getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getSede() {
        return sede;
    }

    public void setSede(String sede) {
        this.sede = sede;
    }

    public Set<Piloto> getPilotos() {
        return pilotos;
    }

    public void setPilotos(Set<Piloto> pilotos) {
        this.pilotos = pilotos;
    }

    @Override
    public String toString() {
        return "Escuderia{" + "id=" + id + ", nombre='" + nombre +
'\'' +
                ", sede='" + sede + '\'' + '}';
    }
}
```

@ManyToMany(mappedBy = "escuderias")
private Set<Piloto> pilotos = new HashSet<>();

Significado:

mappedBy = "escuderias" → indica que LA OTRA CLASE (Piloto) tiene la tabla intermedia.

Esta es la parte inversa de la relación.

PilotoDAO.java – Operaciones CRUD + consultas

```
package dao;

import modelo.Piloto;
import modelo.Escuderia;
import org.hibernate.Session;
import org.hibernate.Transaction;
import util.HibernateUtil;

import java.util.List;

public class PilotoDAO {

   public void guardar(Piloto p) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.persist(p);
       tx.commit();
       session.close();
   }

   public Piloto obtener(Long id) {
       try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
           return session.find(Piloto.class, id);
       }
   }

   public List<Piloto> listar() {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       List<Piloto> lista = session.createQuery("FROM Piloto",
Piloto.class).list();
       session.close();
       return lista;
   }

   public void actualizar(Piloto p) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.merge(p);
```

```
        tx.commit();
        session.close();
    }

    public void eliminar(Piloto p) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.remove(p);
        tx.commit();
        session.close();
    }

    public void asignarEscuderia(Long idPiloto, Long idEscuderia) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        Piloto p = session.find(Piloto.class, idPiloto);
        Escuderia e = session.find(Escuderia.class, idEscuderia);

        if (p != null && e != null) {
            p.getEscuderias().add(e);
            e.getPilotos().add(p);
            session.merge(p);
        }

        tx.commit();
        session.close();
    }

    public List<Escuderia> getEscuderiasDePiloto(Long idPiloto) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        List<Escuderia> lista = session
                .createQuery("SELECT e FROM Piloto p JOIN
p.escuderias e WHERE p.id = :id",
                        Escuderia.class)
                .setParameter("id", idPiloto)
                .list();
        session.close();
        return lista;
    }
}
```

Métodos:
guardar()

Crea un piloto en la BD.

obtener()

Busca un piloto por ID.

listar()

Devuelve todos los pilotos.

actualizar()

Guarda cambios en la BD.

eliminar()

Borra un piloto.

asignarEscuderia()

p.getEscuderias().add(e);
e.getPilotos().add(p);
session.merge(p);

Hibernate:

Gestiona la tabla intermedia

Inserta (piloto_id, escuderia_id).

getEscuderiasDePiloto()

Consulta HQL:

SELECT e FROM Piloto p JOIN p.escuderias e WHERE p.id = :id

EscuderiaDAO.java – Igual que PilotoDAO pero al revés

```
package dao;

import modelo.Escuderia;
import modelo.Piloto;
import org.hibernate.Session;
import org.hibernate.Transaction;
import util.HibernateUtil;
```

```
import java.util.List;

public class EscuderiaDAO {

   public void guardar(Escuderia e) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.persist(e);
       tx.commit();
       session.close();
   }

   public Escuderia obtener(Long id) {
       try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
           return session.find(Escuderia.class, id);
       }
   }

   public List<Escuderia> listar() {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       List<Escuderia> lista = session.createQuery("FROM
Escuderia", Escuderia.class).list();
       session.close();
       return lista;
   }

   public void actualizar(Escuderia e) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.merge(e);
       tx.commit();
       session.close();
   }

   public void eliminar(Escuderia e) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.remove(e);
       tx.commit();
       session.close();
   }
```

```
    public List<Piloto> getPilotosDeEscuderia(Long idEscuderia) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        List<Piloto> lista = session
                .createQuery("SELECT p FROM Escuderia e JOIN
e.pilotos p WHERE e.id = :id",
                        Piloto.class)
                .setParameter("id", idEscuderia)
                .list();
        session.close();
        return lista;
    }
}
```

Incluye una consulta específica:

SELECT p FROM Escuderia e JOIN e.pilotos p WHERE e.id = :id

HibernateUtil.java – SessionFactory

```
package util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory =
buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new
Configuration().configure().buildSessionFactory();
        } catch (Exception e) {
            System.err.println("Error en SessionFactory: " + e);
            throw new RuntimeException(e);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

Crea y gestiona la SessionFactory

hibernate.cfg.xml – Configuración de Hibernate

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
   <session-factory>
       <property
name="hibernate.connection.driver_class">org.sqlite.JDBC</property
>
       <property
name="hibernate.connection.url">jdbc:sqlite:f1.db</property>
       <property
name="hibernate.dialect">org.hibernate.community.dialect.SQLiteDia
lect</property>
       <property name="hibernate.show_sql">true</property>
       <property name="hibernate.format_sql">true</property>
       <property name="hibernate.hbm2ddl.auto">update</property>

       <mapping class="modelo.Piloto"/>
       <mapping class="modelo.Escuderia"/>
   </session-factory>
</hibernate-configuration>
```

Define:

Driver SQLite

Dialecto

Activar logs SQL

BD f1.db

Clases mapeadas:

<mapping class="modelo.Piloto"/>
<mapping class="modelo.Escuderia"/>

pom.xml – Dependencias del proyecto

```
<?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>org.example</groupId>
    <artifactId>act2ut3</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>26</maven.compiler.source>
        <maven.compiler.target>26</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <hibernate.version>7.1.0.Final</hibernate.version>
        <sqlite.jdbc.version>3.50.3.0</sqlite.jdbc.version>
        <slf4j.version>2.0.17</slf4j.version>
    </properties>

    <dependencies>
        <!-- Hibernate core (incluye implementación JPA) -->
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>${hibernate.version}</version>
        </dependency>

        <!-- Community Dialects (SQLite) -->
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-community-dialects</artifactId>
            <version>${hibernate.version}</version>
        </dependency>

        <!-- SQLite JDBC -->
        <dependency>
            <groupId>org.xerial</groupId>
            <artifactId>sqlite-jdbc</artifactId>
            <version>${sqlite.jdbc.version}</version>
        </dependency>

        <!-- SLF4J Simple -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
```

```
    </dependencies>

</project>
```

Incluye:

Hibernate Core

Dialectos Community (SQLite)

SQLite JDBC

SLF4J para logs

Versión Hibernate: 7.1.0.Final
SQLite: 3.50.3.0

GenerarBD.java – Genera base de datos de ejemplo

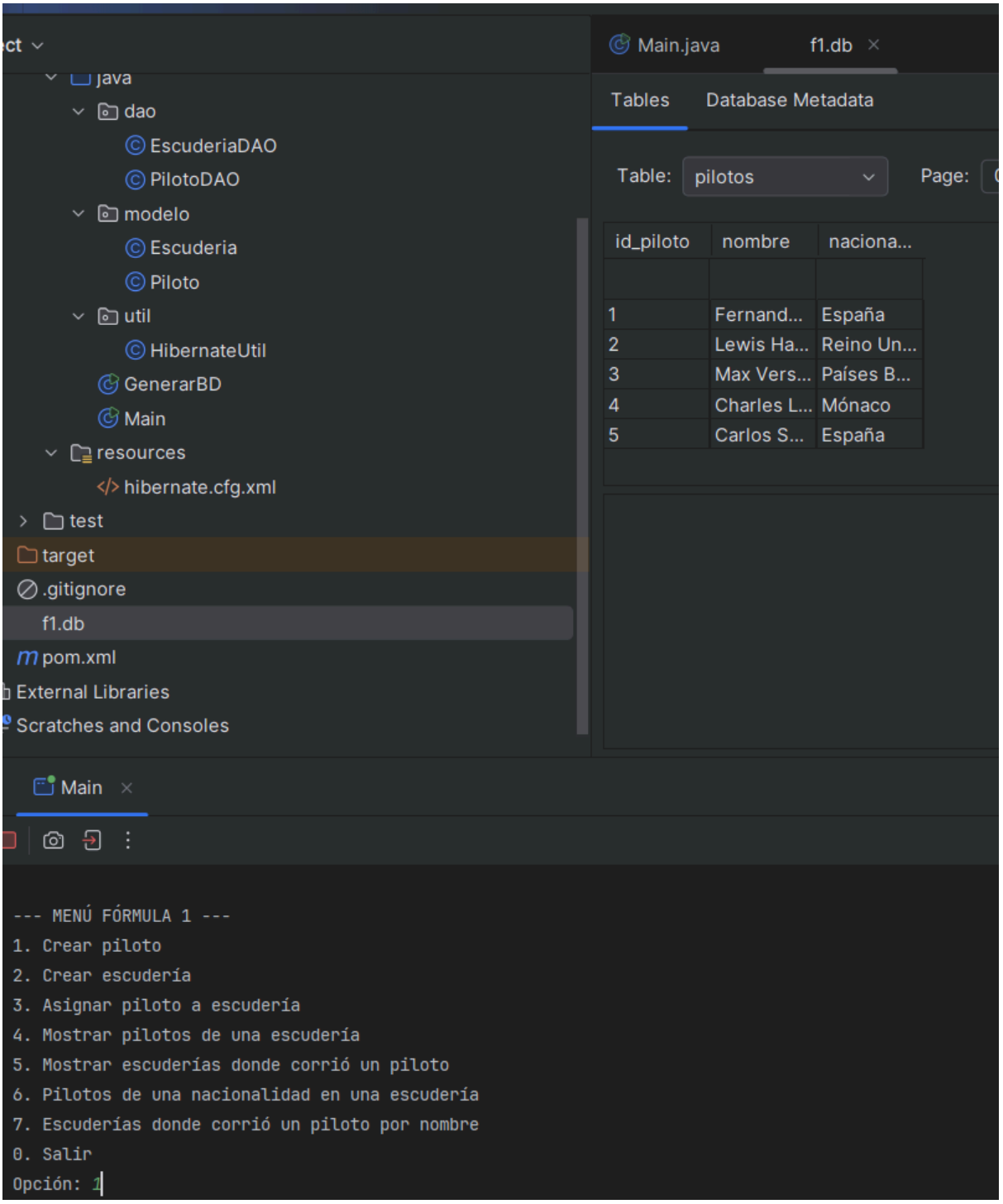
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class GenerarBD {

    public static void main(String[] args) {
        String url = "jdbc:sqlite:f1.db";

        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement()) {

            // Eliminar tablas si existen
            stmt.executeUpdate("DROP TABLE IF EXISTS
piloto_escuderia");
            stmt.executeUpdate("DROP TABLE IF EXISTS pilotos");
            stmt.executeUpdate("DROP TABLE IF EXISTS escuderias");

            // Crear tabla pilotos
            stmt.executeUpdate("""
                CREATE TABLE pilotos (
                    id_piloto INTEGER PRIMARY KEY AUTOINCREMENT,
                    nombre TEXT NOT NULL,
                    nacionalidad TEXT NOT NULL
```

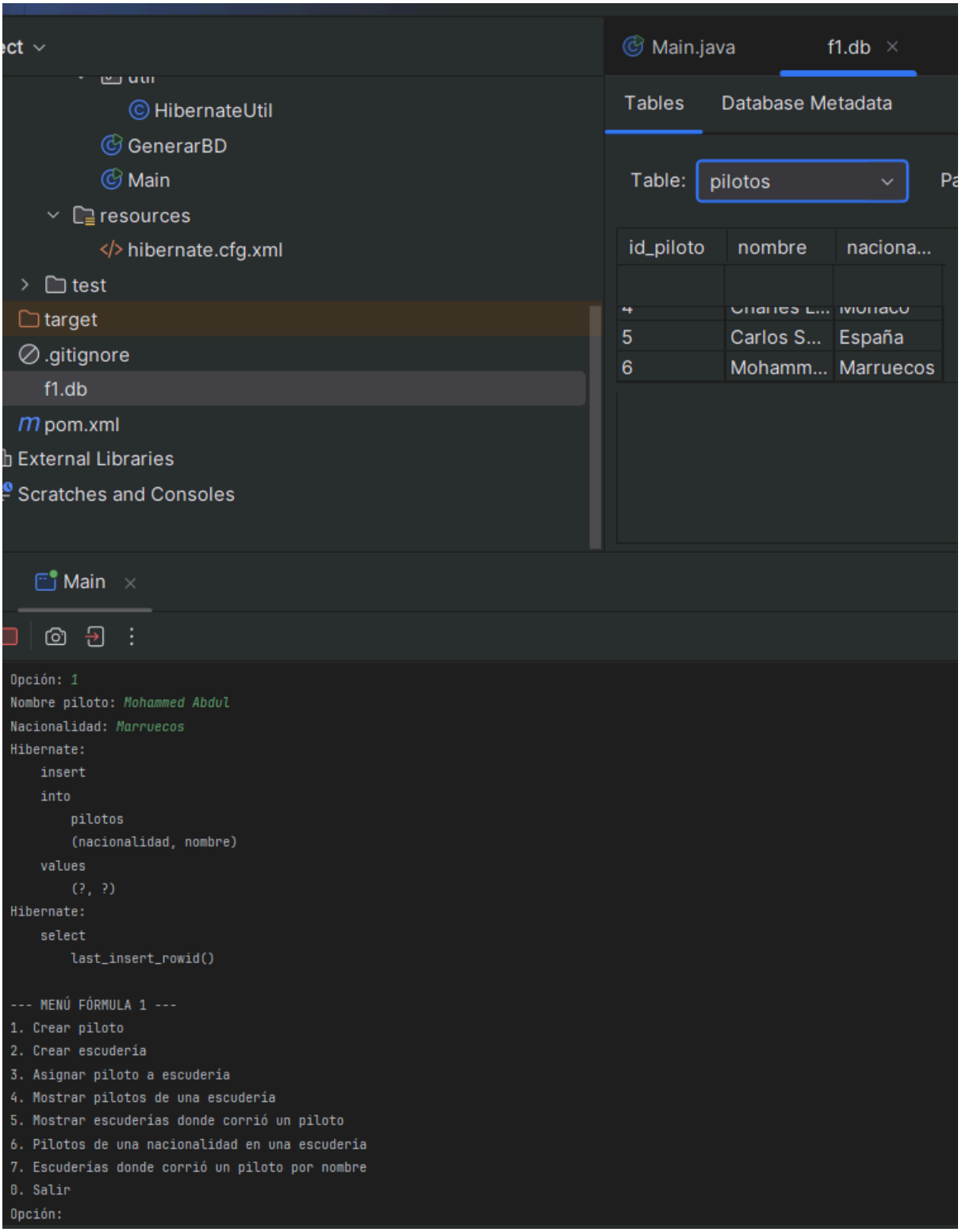
```
            );
        """);

        // Crear tabla escuderias
        stmt.executeUpdate("""
            CREATE TABLE escuderias (
                id_escuderia INTEGER PRIMARY KEY AUTOINCREMENT,
                nombre TEXT NOT NULL,
                sede TEXT NOT NULL
            );
        """);

        // Crear tabla intermedia
        stmt.executeUpdate("""
            CREATE TABLE piloto_escuderia (
                piloto_id INTEGER NOT NULL,
                escuderia_id INTEGER NOT NULL,
                PRIMARY KEY (piloto_id, escuderia_id),
                FOREIGN KEY (piloto_id) REFERENCES
pilotos(id_piloto),
                FOREIGN KEY (escuderia_id) REFERENCES
escuderias(id_escuderia)
            );
        """);

        // Insertar pilotos
        stmt.executeUpdate("""
            INSERT INTO pilotos (nombre, nacionalidad) VALUES
            ('Fernando Alonso', 'España'),
            ('Lewis Hamilton', 'Reino Unido'),
            ('Max Verstappen', 'Países Bajos'),
            ('Charles Leclerc', 'Mónaco'),
            ('Carlos Sainz', 'España');
        """);

        // Insertar escuderías
        stmt.executeUpdate("""
            INSERT INTO escuderias (nombre, sede) VALUES
            ('Ferrari', 'Maranello, Italia'),
            ('Mercedes', 'Brackley, Reino Unido'),
            ('Red Bull Racing', 'Milton Keynes, Reino Unido'),
            ('McLaren', 'Woking, Reino Unido'),
            ('Renault/Alpine', 'Enstone, Reino Unido');
        """);

        // Insertar relaciones piloto ↔ escudería
        stmt.executeUpdate("""
            INSERT INTO piloto_escuderia VALUES
```

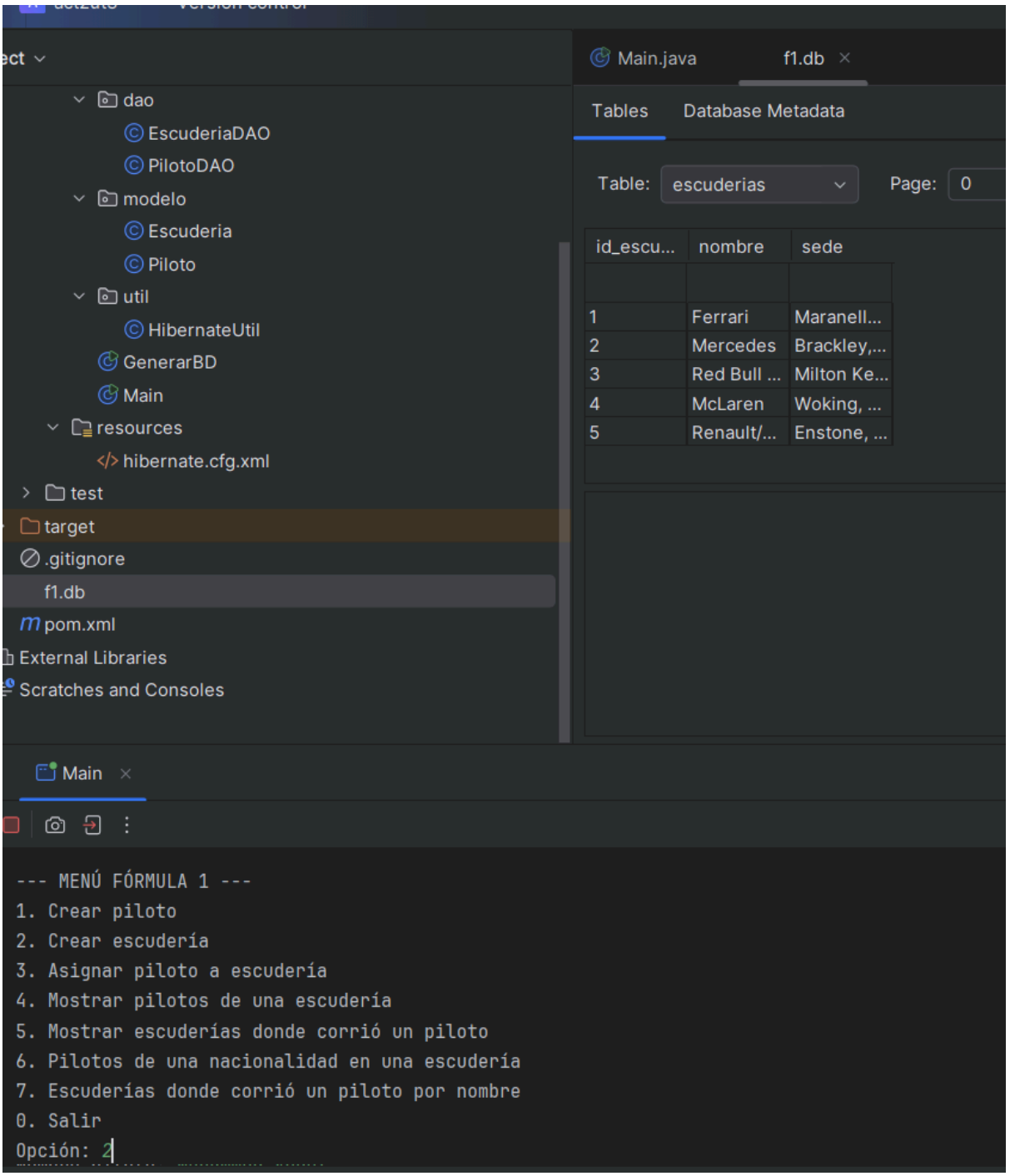
```
                (1, 4),    -- Alonso en McLaren
                (1, 5),    -- Alonso en Renault/Alpine
                (2, 2),    -- Hamilton en Mercedes
                (2, 4),    -- Hamilton en McLaren
                (3, 3),    -- Verstappen en Red Bull
                (4, 1),    -- Leclerc en Ferrari
                (5, 1),    -- Sainz en Ferrari
                (5, 5);    -- Sainz en Renault/Alpine
            """);

            System.out.println("Base de datos f1.db generada
correctamente.");

        } catch (Exception e) {
            System.err.println("Error al generar la base de datos:
" + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

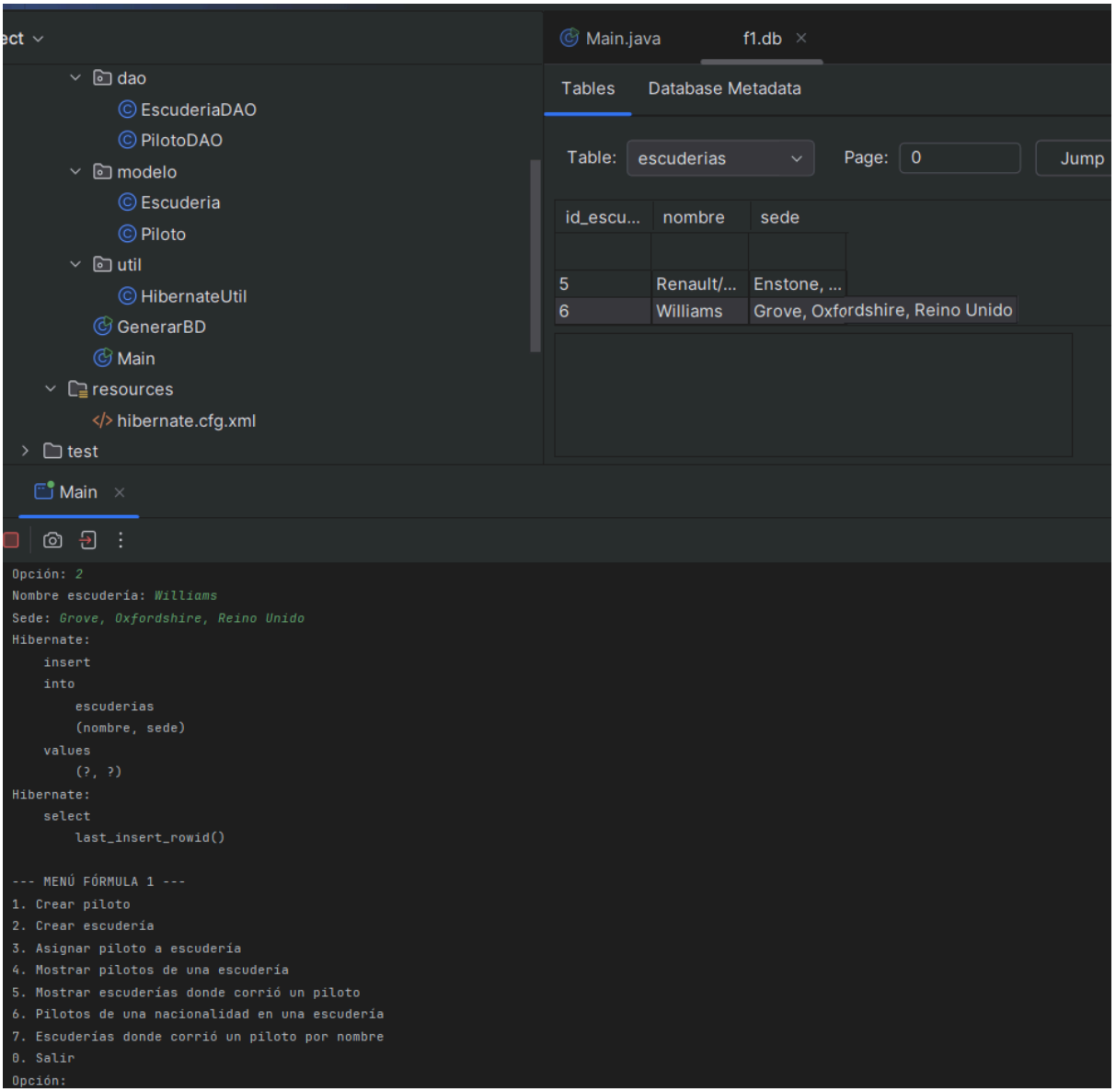
Genera un f1.db con inserciones de ejemplo.

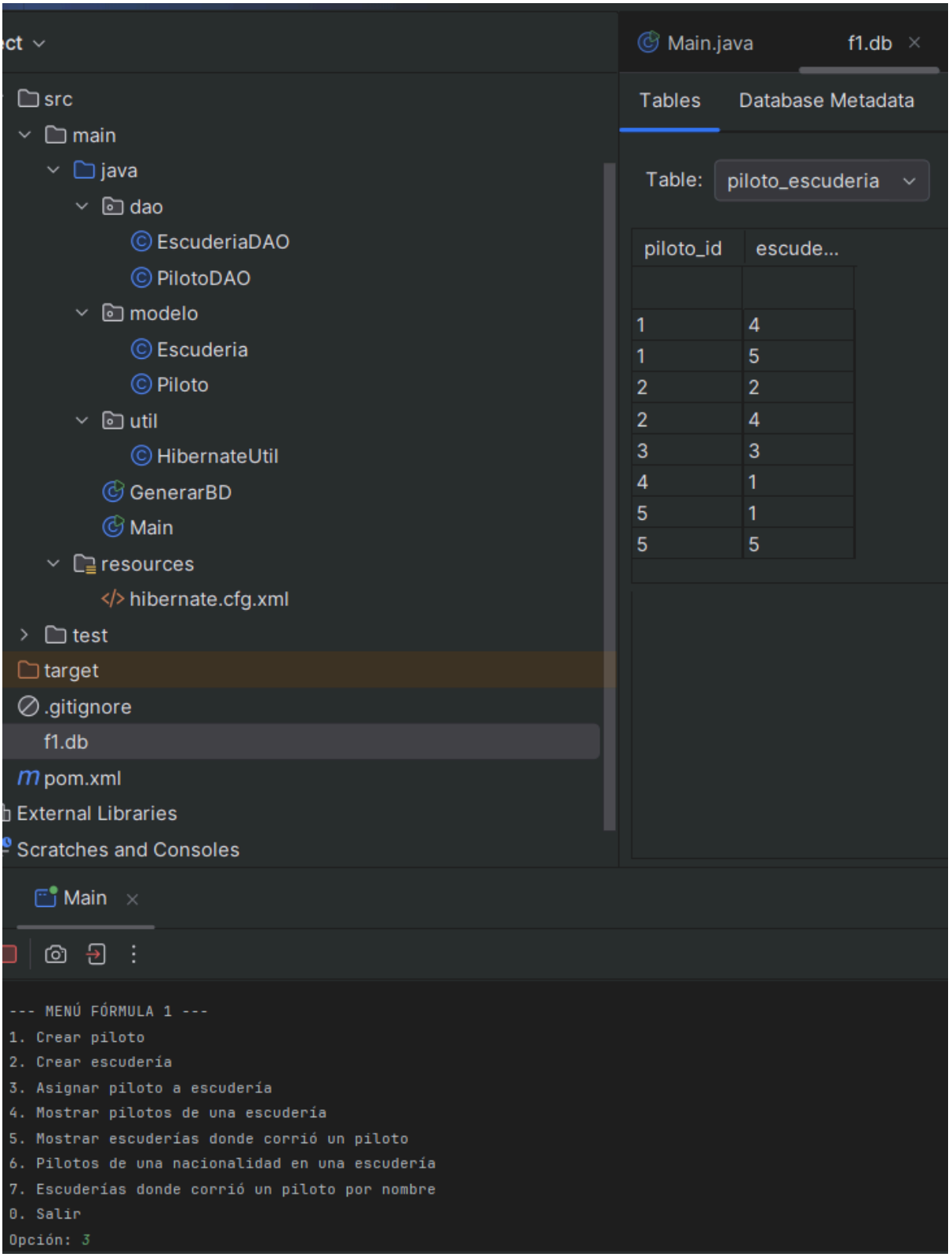
Capturas de pantalla




HibernateUtil
GenerarBD
Main
resources
hibernate.cfg.xml
test
target
.gitignore
f1.db
pom.xml
External Libraries
Scratches and Consoles
Main.java
f1.db
Tables
Database Metadata
Table: pilotos
id_piloto
nombre
naciona...
5
Carlos S...
España
6
Mohamm...
Marruecos
Main
Opción: 1
Nombre piloto: Mohammed Abdul
Nacionalidad: Marruecos
Hibernate:
insert
into
pilotos
(nacionalidad, nombre)
values
(?, ?)
Hibernate:
select
last_insert_rowid()
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:


dao
EscuderiaDAO
PilotoDAO
modelo
Escuderia
Piloto
util
HibernateUtil
GenerarBD
Main
resources
hibernate.cfg.xml
test
target
.gitignore
f1.db
pom.xml
External Libraries
Scratches and Consoles
Main.java
f1.db
Tables
Database Metadata
Table: escuderias
Page: 0
id_escu...
nombre
sede
1
Ferrari
Maranell...
2
Mercedes
Brackley,...
3
Red Bull ...
Milton Ke...
4
McLaren
Woking, ...
5
Renault/...
Enstone, ...
Main
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 4

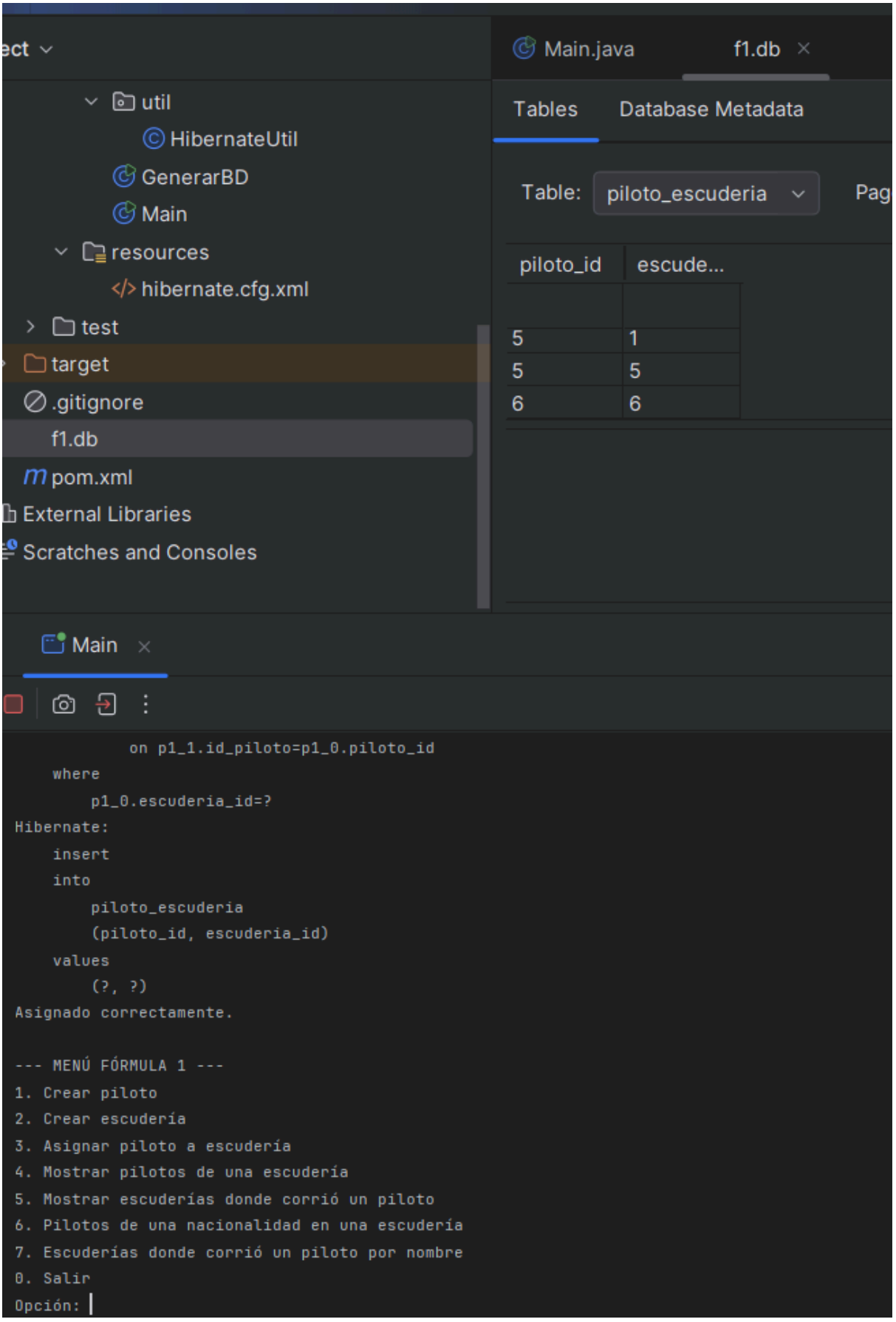

dao
EscuderiaDAO
PilotoDAO
modelo
Escuderia
Piloto
util
HibernateUtil
GenerarBD
Main
resources
hibernate.cfg.xml
test
Main.java
f1.db
Tables
Database Metadata
Table: escuderias
Page: 0
Jump
id_escu...
nombre
sede
5
Renault/...
Enstone, ...
6
Williams
Grove, Oxfordshire, Reino Unido
Main
Opción: 2
Nombre escudería: Williams
Sede: Grove, Oxfordshire, Reino Unido
Hibernate:
insert
into
escuderias
(nombre, sede)
values
(?, ?)
Hibernate:
select
last_insert_rowid()
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:


src
main
java
dao
EscuderiaDAO
PilotoDAO
modelo
Escuderia
Piloto
util
HibernateUtil
GenerarBD
Main
resources
hibernate.cfg.xml
test
target
.gitignore
f1.db
pom.xml
External Libraries
Scratches and Consoles
Main.java
f1.db
Tables
Database Metadata
Table: piloto_escuderia
piloto_id
escude...
1 4
1 5
2 2
2 4
3 3
4 1
5 1
5 5
Main
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 3

```
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 3
ID Piloto: 6
ID Escudería: 6
```

```
Hibernate:
    select
        p1_0.id_piloto,
        p1_0.nacionalidad,
        p1_0.nombre
    from
        pilotos p1_0
    where
        p1_0.id_piloto=?
Hibernate:
    select
        e1_0.id_escuderia,
        e1_0.nombre,
        e1_0.sede
    from
        escuderias e1_0
    where
        e1_0.id_escuderia=?
```

```
Hibernate:
    select
        e1_0.piloto_id,
        e1_1.id_escuderia,
        e1_1.nombre,
        e1_1.sede
    from
        piloto_escuderia e1_0
    join
        escuderias e1_1
            on e1_1.id_escuderia=e1_0.escuderia_id
    where
        e1_0.piloto_id=?
Hibernate:
    select
        p1_0.escuderia_id,
        p1_1.id_piloto,
        p1_1.nacionalidad,
        p1_1.nombre
    from
        piloto_escuderia p1_0
    join
        pilotos p1_1
            on p1_1.id_piloto=p1_0.piloto_id
    where
        p1_0.escuderia_id=?
```


util
HibernateUtil
GenerarBD
Main
resources
hibernate.cfg.xml
test
target
.gitignore
f1.db
pom.xml
External Libraries
Scratches and Consoles
Main.java
f1.db
Tables
Database Metadata
Table: piloto_escuderia
piloto_id
escude...
5 1
5 5
6 6
Main
on p1_1.id_piloto=p1_0.piloto_id
where
p1_0.escuderia_id=?
Hibernate:
insert
into
piloto_escuderia
(piloto_id, escuderia_id)
values
(?, ?)
Asignado correctamente.
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:

```
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 4
ID Escudería: 1
Hibernate:
    select
        p1_1.id_piloto,
        p1_1.nacionalidad,
        p1_1.nombre
    from
        escuderias e1_0
    join
        piloto_escuderia p1_0
            on e1_0.id_escuderia=p1_0.escuderia_id
    join
        pilotos p1_1
            on p1_1.id_piloto=p1_0.piloto_id
    where
        e1_0.id_escuderia=?
Piloto{id=4, nombre='Charles Leclerc', nacionalidad='Mónaco'}
Piloto{id=5, nombre='Carlos Sainz', nacionalidad='España'}

--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:
```

```
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 5
ID Piloto: 1
Hibernate:
    select
        e1_1.id_escuderia,
        e1_1.nombre,
        e1_1.sede
    from
        pilotos p1_0
    join
        piloto_escuderia e1_0
            on p1_0.id_piloto=e1_0.piloto_id
    join
        escuderias e1_1
            on e1_1.id_escuderia=e1_0.escuderia_id
    where
        p1_0.id_piloto=?
Escuderia{id=4, nombre='McLaren', sede='Woking, Reino Unido'}
Escuderia{id=5, nombre='Renault/Alpine', sede='Enstone, Reino Unido'}

--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:
```

```
Opción: 6
Nacionalidad: España
Nombre escudería: Ferrari
Hibernate:
    select
        p1_0.id_piloto,
        p1_0.nacionalidad,
        p1_0.nombre
    from
        pilotos p1_0
    join
        piloto_escuderia e1_0
            on p1_0.id_piloto=e1_0.piloto_id
    join
        escuderias e1_1
            on e1_1.id_escuderia=e1_0.escuderia_id
    where
        p1_0.nacionalidad=?
        and e1_1.nombre=?
Piloto{id=5, nombre='Carlos Sainz', nacionalidad='España'}

--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:
```

```
Opción: 7
Nombre piloto: Carlos Sainz
Hibernate:
    select
        e1_0.id_escuderia,
        e1_0.nombre,
        e1_0.sede
    from
        escuderias e1_0
    join
        piloto_escuderia p1_0
            on e1_0.id_escuderia=p1_0.escuderia_id
    join
        pilotos p1_1
            on p1_1.id_piloto=p1_0.piloto_id
    where
        p1_1.nombre=?
Escuderia{id=1, nombre='Ferrari', sede='Maranello, Italia'}
Escuderia{id=5, nombre='Renault/Alpine', sede='Enstone, Reino Unido'}

--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción:
```

```
--- MENÚ FÓRMULA 1 ---
1. Crear piloto
2. Crear escudería
3. Asignar piloto a escudería
4. Mostrar pilotos de una escudería
5. Mostrar escuderías donde corrió un piloto
6. Pilotos de una nacionalidad en una escudería
7. Escuderías donde corrió un piloto por nombre
0. Salir
Opción: 0
Saliendo...

Process finished with exit code 0
```

Código completo

Main.java:

```
import modelo.*;
import dao.*;
import util.HibernateUtil;
import org.hibernate.Session;

import java.util.List;
import java.util.Scanner;

public class Main {

   public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);
       PilotoDAO pilotoDAO = new PilotoDAO();
       EscuderiaDAO escuderiaDAO = new EscuderiaDAO();

       while (true) {
           System.out.println("\n--- MENÚ FÓRMULA 1 ---");
           System.out.println("1. Crear piloto");
           System.out.println("2. Crear escudería");
           System.out.println("3. Asignar piloto a escudería");
           System.out.println("4. Mostrar pilotos de una
escudería");
           System.out.println("5. Mostrar escuderías donde corrió
un piloto");
           System.out.println("6. Pilotos de una nacionalidad en
una escudería");
           System.out.println("7. Escuderías donde corrió un
piloto por nombre");
           System.out.println("0. Salir");
           System.out.print("Opción: ");

           int op = Integer.parseInt(sc.nextLine());

           switch (op) {

               case 1 -> {
                   System.out.print("Nombre piloto: ");
                   String n = sc.nextLine();
                   System.out.print("Nacionalidad: ");
                   String nac = sc.nextLine();

                   pilotoDAO.guardar(new Piloto(n, nac));
```

```
                }

                case 2 -> {
                    System.out.print("Nombre escudería: ");
                    String n = sc.nextLine();
                    System.out.print("Sede: ");
                    String s = sc.nextLine();

                    escuderiaDAO.guardar(new Escuderia(n, s));
                }

                case 3 -> {
                    System.out.print("ID Piloto: ");
                    Long idP = Long.parseLong(sc.nextLine());
                    System.out.print("ID Escudería: ");
                    Long idE = Long.parseLong(sc.nextLine());

                    pilotoDAO.asignarEscuderia(idP, idE);
                    System.out.println("Asignado correctamente.");
                }

                case 4 -> {
                    System.out.print("ID Escudería: ");
                    Long id = Long.parseLong(sc.nextLine());

                    List<Piloto> pilotos =
escuderiaDAO.getPilotosDeEscuderia(id);
                    pilotos.forEach(System.out::println);
                }

                case 5 -> {
                    System.out.print("ID Piloto: ");
                    Long id = Long.parseLong(sc.nextLine());

                    List<Escuderia> esc =
pilotoDAO.getEscuderiasDePiloto(id);
                    esc.forEach(System.out::println);
                }

                case 6 -> {
                    System.out.print("Nacionalidad: ");
                    String nac = sc.nextLine();
                    System.out.print("Nombre escudería: ");
                    String escu = sc.nextLine();

                    Session session =
HibernateUtil.getSessionFactory().openSession();
                    List<Piloto> lista = session.createQuery(
```

```
                                    "SELECT p FROM Piloto p JOIN
p.escuderias e " +
                                            "WHERE p.nacionalidad =
:n AND e.nombre = :en",
                                    Piloto.class)
                            .setParameter("n", nac)
                            .setParameter("en", escu)
                            .list();
                    session.close();

                    lista.forEach(System.out::println);
                }

                case 7 -> {
                    System.out.print("Nombre piloto: ");
                    String nombre = sc.nextLine();

                    Session session =
HibernateUtil.getSessionFactory().openSession();
                    List<Escuderia> lista = session.createQuery(
                                    "SELECT e FROM Escuderia e JOIN
e.pilotos p " +
                                            "WHERE p.nombre = :n",
                                    Escuderia.class)
                            .setParameter("n", nombre)
                            .list();
                    session.close();

                    lista.forEach(System.out::println);
                }

                case 0 -> {
                    System.out.println("Saliendo...");
                    HibernateUtil.getSessionFactory().close();
                    return;
                }
            }
        }
    }
}
```

Piloto.java:

```
package modelo;

import jakarta.persistence.*;
import java.util.HashSet;
```

```java
import java.util.Set;

@Entity
@Table(name = "pilotos")
public class Piloto {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_piloto")
    private Long id;

    private String nombre;
    private String nacionalidad;

    @ManyToMany
    @JoinTable(
            name = "piloto_escuderia",
            joinColumns = @JoinColumn(name = "piloto_id"),
            inverseJoinColumns = @JoinColumn(name = "escuderia_id")
    )
    private Set<Escuderia> escuderias = new HashSet<>();

    public Piloto() {}

    public Piloto(String nombre, String nacionalidad) {
        this.nombre = nombre;
        this.nacionalidad = nacionalidad;
    }

    public Long getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getNacionalidad() {
        return nacionalidad;
    }

    public void setNacionalidad(String nacionalidad) {
        this.nacionalidad = nacionalidad;
    }
```

```
    public Set<Escuderia> getEscuderias() {
        return escuderias;
    }

    public void setEscuderias(Set<Escuderia> escuderias) {
        this.escuderias = escuderias;
    }

    @Override
    public String toString() {
        return "Piloto{" + "id=" + id + ", nombre='" + nombre +
'\'' +
                ", nacionalidad='" + nacionalidad + '\'' + '}';
    }
}
```

Escuderia.java:

```
package modelo;

import jakarta.persistence.*;
import java.util.HashSet;
import java.util.Set;

@Entity
@Table(name = "escuderias")
public class Escuderia {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_escuderia")
    private Long id;

    private String nombre;
    private String sede;

    @ManyToMany(mappedBy = "escuderias")
    private Set<Piloto> pilotos = new HashSet<>();

    public Escuderia() {}

    public Escuderia(String nombre, String sede) {
        this.nombre = nombre;
        this.sede = sede;
    }
```

```
    public Long getId() {
        return id;
    }

    public String getNombre() {
        return nombre;
    }

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

    public String getSede() {
        return sede;
    }

    public void setSede(String sede) {
        this.sede = sede;
    }

    public Set<Piloto> getPilotos() {
        return pilotos;
    }

    public void setPilotos(Set<Piloto> pilotos) {
        this.pilotos = pilotos;
    }

    @Override
    public String toString() {
        return "Escuderia{" + "id=" + id + ", nombre='" + nombre +
'\'' +
                ", sede='" + sede + '\'' + '}';
    }
}
```

PilotoDAO.java:

```
package dao;

import modelo.Piloto;
import modelo.Escuderia;
import org.hibernate.Session;
import org.hibernate.Transaction;
import util.HibernateUtil;

import java.util.List;
```

```
public class PilotoDAO {

   public void guardar(Piloto p) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.persist(p);
       tx.commit();
       session.close();
   }

   public Piloto obtener(Long id) {
       try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
           return session.find(Piloto.class, id);
       }
   }

   public List<Piloto> listar() {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       List<Piloto> lista = session.createQuery("FROM Piloto",
Piloto.class).list();
       session.close();
       return lista;
   }

   public void actualizar(Piloto p) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.merge(p);
       tx.commit();
       session.close();
   }

   public void eliminar(Piloto p) {
       Session session =
HibernateUtil.getSessionFactory().openSession();
       Transaction tx = session.beginTransaction();
       session.remove(p);
       tx.commit();
       session.close();
   }

   public void asignarEscuderia(Long idPiloto, Long idEscuderia) {
```

```
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();

        Piloto p = session.find(Piloto.class, idPiloto);
        Escuderia e = session.find(Escuderia.class, idEscuderia);

        if (p != null && e != null) {
            p.getEscuderias().add(e);
            e.getPilotos().add(p);
            session.merge(p);
        }

        tx.commit();
        session.close();
    }

    public List<Escuderia> getEscuderiasDePiloto(Long idPiloto) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        List<Escuderia> lista = session
                .createQuery("SELECT e FROM Piloto p JOIN
p.escuderias e WHERE p.id = :id",
                        Escuderia.class)
                .setParameter("id", idPiloto)
                .list();
        session.close();
        return lista;
    }
}
```

EscuderiaDAO.java:

```
package dao;

import modelo.Escuderia;
import modelo.Piloto;
import org.hibernate.Session;
import org.hibernate.Transaction;
import util.HibernateUtil;

import java.util.List;

public class EscuderiaDAO {

    public void guardar(Escuderia e) {
```

```
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.persist(e);
        tx.commit();
        session.close();
    }

    public Escuderia obtener(Long id) {
        try (Session session =
HibernateUtil.getSessionFactory().openSession()) {
            return session.find(Escuderia.class, id);
        }
    }

    public List<Escuderia> listar() {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        List<Escuderia> lista = session.createQuery("FROM
Escuderia", Escuderia.class).list();
        session.close();
        return lista;
    }

    public void actualizar(Escuderia e) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.merge(e);
        tx.commit();
        session.close();
    }

    public void eliminar(Escuderia e) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        Transaction tx = session.beginTransaction();
        session.remove(e);
        tx.commit();
        session.close();
    }

    public List<Piloto> getPilotosDeEscuderia(Long idEscuderia) {
        Session session =
HibernateUtil.getSessionFactory().openSession();
        List<Piloto> lista = session
                .createQuery("SELECT p FROM Escuderia e JOIN
e.pilotos p WHERE e.id = :id",
```

```
                        Piloto.class)
                .setParameter("id", idEscuderia)
                .list();
        session.close();
        return lista;
    }
}
```

HibernateUtil.java:

```
package util;

import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;

public class HibernateUtil {

    private static final SessionFactory sessionFactory =
buildSessionFactory();

    private static SessionFactory buildSessionFactory() {
        try {
            return new
Configuration().configure().buildSessionFactory();
        } catch (Exception e) {
            System.err.println("Error en SessionFactory: " + e);
            throw new RuntimeException(e);
        }
    }

    public static SessionFactory getSessionFactory() {
        return sessionFactory;
    }
}
```

GenerarBD.java:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

public class GenerarBD {

    public static void main(String[] args) {
        String url = "jdbc:sqlite:f1.db";
```

```
        try (Connection conn = DriverManager.getConnection(url);
             Statement stmt = conn.createStatement()) {

            // Eliminar tablas si existen
            stmt.executeUpdate("DROP TABLE IF EXISTS
piloto_escuderia");
            stmt.executeUpdate("DROP TABLE IF EXISTS pilotos");
            stmt.executeUpdate("DROP TABLE IF EXISTS escuderias");

            // Crear tabla pilotos
            stmt.executeUpdate("""
                CREATE TABLE pilotos (
                    id_piloto INTEGER PRIMARY KEY AUTOINCREMENT,
                    nombre TEXT NOT NULL,
                    nacionalidad TEXT NOT NULL
                );
            """);

            // Crear tabla escuderias
            stmt.executeUpdate("""
                CREATE TABLE escuderias (
                    id_escuderia INTEGER PRIMARY KEY AUTOINCREMENT,
                    nombre TEXT NOT NULL,
                    sede TEXT NOT NULL
                );
            """);

            // Crear tabla intermedia
            stmt.executeUpdate("""
                CREATE TABLE piloto_escuderia (
                    piloto_id INTEGER NOT NULL,
                    escuderia_id INTEGER NOT NULL,
                    PRIMARY KEY (piloto_id, escuderia_id),
                    FOREIGN KEY (piloto_id) REFERENCES
pilotos(id_piloto),
                    FOREIGN KEY (escuderia_id) REFERENCES
escuderias(id_escuderia)
                );
            """);

            // Insertar pilotos
            stmt.executeUpdate("""
                INSERT INTO pilotos (nombre, nacionalidad) VALUES
                ('Fernando Alonso', 'España'),
                ('Lewis Hamilton', 'Reino Unido'),
                ('Max Verstappen', 'Países Bajos'),
                ('Charles Leclerc', 'Mónaco'),
                ('Carlos Sainz', 'España');
```

```
            """);

            // Insertar escuderías
            stmt.executeUpdate("""
                INSERT INTO escuderias (nombre, sede) VALUES
                ('Ferrari', 'Maranello, Italia'),
                ('Mercedes', 'Brackley, Reino Unido'),
                ('Red Bull Racing', 'Milton Keynes, Reino Unido'),
                ('McLaren', 'Woking, Reino Unido'),
                ('Renault/Alpine', 'Enstone, Reino Unido');
            """);

            // Insertar relaciones piloto ↔ escudería
            stmt.executeUpdate("""
                INSERT INTO piloto_escuderia VALUES
                (1, 4),   -- Alonso en McLaren
                (1, 5),   -- Alonso en Renault/Alpine
                (2, 2),   -- Hamilton en Mercedes
                (2, 4),   -- Hamilton en McLaren
                (3, 3),   -- Verstappen en Red Bull
                (4, 1),   -- Leclerc en Ferrari
                (5, 1),   -- Sainz en Ferrari
                (5, 5);   -- Sainz en Renault/Alpine
            """);

            System.out.println("Base de datos f1.db generada
correctamente.");

        } catch (Exception e) {
            System.err.println("Error al generar la base de datos:
" + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

hibernate.cfg.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
    <session-factory>
```

```
        <property
name="hibernate.connection.driver_class">org.sqlite.JDBC</property
>
        <property
name="hibernate.connection.url">jdbc:sqlite:f1.db</property>
        <property
name="hibernate.dialect">org.hibernate.community.dialect.SQLiteDia
lect</property>
        <property name="hibernate.show_sql">true</property>
        <property name="hibernate.format_sql">true</property>
        <property name="hibernate.hbm2ddl.auto">update</property>

        <mapping class="modelo.Piloto"/>
        <mapping class="modelo.Escuderia"/>
    </session-factory>
</hibernate-configuration>
```

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

    <groupId>org.example</groupId>
    <artifactId>act2ut3</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>26</maven.compiler.source>
        <maven.compiler.target>26</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <hibernate.version>7.1.0.Final</hibernate.version>
        <sqlite.jdbc.version>3.50.3.0</sqlite.jdbc.version>
        <slf4j.version>2.0.17</slf4j.version>
    </properties>

    <dependencies>
        <!-- Hibernate core (incluye implementación JPA) -->
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-core</artifactId>
            <version>${hibernate.version}</version>
        </dependency>
```

```
        <!-- Community Dialects (SQLite) -->
        <dependency>
            <groupId>org.hibernate.orm</groupId>
            <artifactId>hibernate-community-dialects</artifactId>
            <version>${hibernate.version}</version>
        </dependency>

        <!-- SQLite JDBC -->
        <dependency>
            <groupId>org.xerial</groupId>
            <artifactId>sqlite-jdbc</artifactId>
            <version>${sqlite.jdbc.version}</version>
        </dependency>

        <!-- SLF4J Simple -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>${slf4j.version}</version>
        </dependency>
    </dependencies>

</project>
```

Acceso al zip del proyecto

https://github.com/RafaelMayor/AED/tree/main/act2ut3