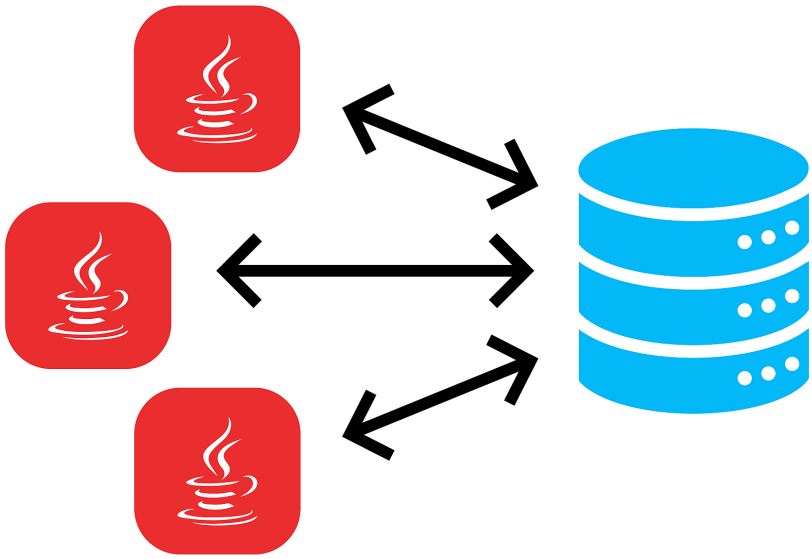
Taller UD10_3: Patron DAO (CRUD completo)



1. **Introducción**
2. **Esquema de la BBDD**
3. **Implementación paso a paso**
4. **Actividades**
5. **Fuentes de información**

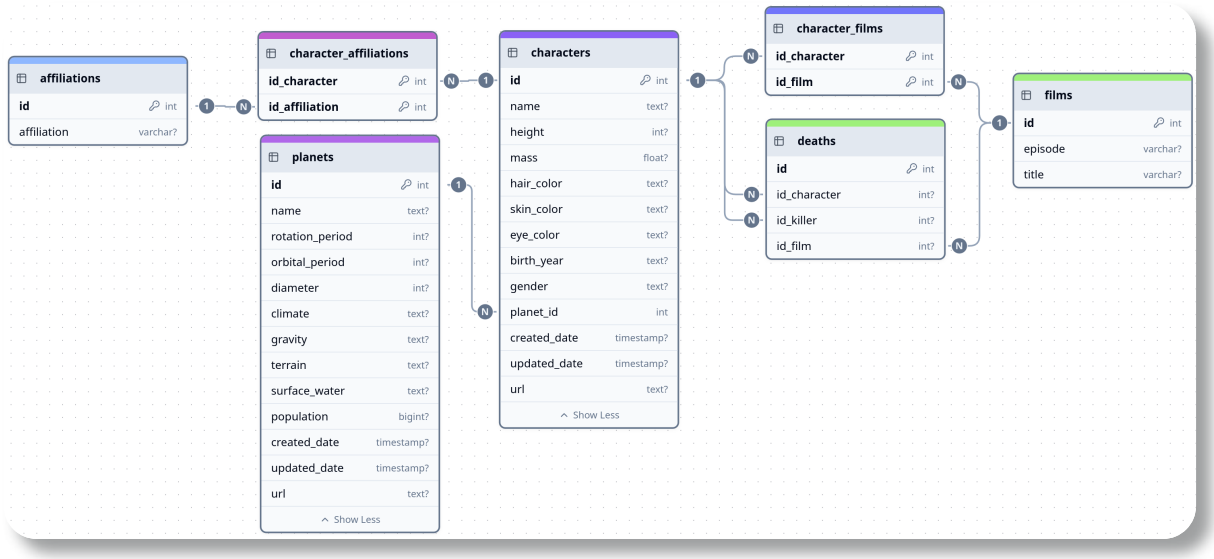
1. Introducción

En el apartado correspondiente de teoria, ya estudiamos los patrones `Singleton` y `DAO`.

> **Importante** Para realizar el ejemplo que seguiremos en este apartado usaremos la BBDD starwars. Pero tu deberás replicar este ejemplo para la BBDD que creaste en el anterior taller UD10_T2_AWS_IntelliJ_ES.

En programación existen una serie de estándares denominados Patrones de Diseño que debes conocer para poder programar según estos patrones y no reinventar la rueda.

2. Esquema de la BBDD



3. Implementación paso a paso

Paso 0: Definir la clase `DbConnect` (`Singleton`), usando HikariCP

Para simplificar la implementar y conexión a la base de datos podemos crear la clase `DbConnect` (como ya hemos implementado anteriormente en algunos ejercicios):

```java
package es.martinezpenya.ejemplos.UD10.starwars;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;
import java.sql.Connection;
import java.sql.SQLException;

public class DBConnect {

    private static HikariDataSource dataSource;

    // Configuración del pool de conexiones
    static {
        HikariConfig config = new HikariConfig();
        config.setJdbcUrl("jdbc:mariadb://databasedmp.cipxbdkxiaqy.us-east-1.rds.amazonaws.com:6000/starwars"); // URL de la base de datos
        config.setUsername("admin"); // Usuario de la base de datos
        config.setPassword("123456Ab$"); // Contraseña de la base de datos
        config.setMaximumPoolSize(20); // Número máximo de conexiones en el pool
        config.setMinimumIdle(2); // Número mínimo de conexiones inactivas en el pool
        config.setIdleTimeout(3000); // Tiempo de espera para conexiones inactivas (en milisegundos)
        config.setMaxLifetime(180000); // Tiempo máximo de vida de una conexión (en milisegundos)
        config.setConnectionTimeout(300000); // Tiempo de espera para obtener una conexión (en milisegundos)

        dataSource = new HikariDataSource(config);
    }

    // Método para obtener una conexión del pool
    public static Connection getConnection() throws SQLException {
        return dataSource.getConnection();
    }

    // Método para cerrar el pool de conexiones (opcional)
    public static void closeDataSource() {
        if (dataSource != null) {
            dataSource.close();
        }
    }
}
```

Paso 1: Definir las clases de las entidades

Definiremos las clases `Film`, `Planet`, `Character` y `Characterfilm`, pero tu deberás hacerlo con todas tus tablas (o al menos 3 de ellas, y 2 deben estar relacionadaS), deberás incluir todos los campos de las tablas, los constructores que estimes oportuno, los getters y setters y un metodo `toString()`:

Film

```java
package es.martinezpenya.ejemplos.UD10.starwars;

public class Film {
    private int id;
    private String episode;
    private String title;

    public Film(int id, String episode, String title) {
        this.id = id;
        this.episode = episode;
        this.title = title;
    }
```

```
13
14      public Film(int id, String title) {
15          this.id = id;
16          this.title = title;
17      }
18
19      // Getters y setters
20      public int getId() {
21          return id;
22      }
23
24      public void setId(int id) {
25          this.id = id;
26      }
27
28      public String getEpisode() {
29          return episode;
30      }
31
32      public void setEpisode(String episode) {
33          this.episode = episode;
34      }
35
36      public String getTitle() {
37          return title;
38      }
39
40      public void setTitle(String title) {
41          this.title = title;
42      }
43
44      @Override
45      public String toString() {
46          return "Film{" +
47                  "id=" + id +
48                  ", episode='" + episode + '\'' +
49                  ", title='" + title + '\'' +
50                  '}';
51      }
52  }
```

Planet

```
1   package es.martinezpenya.ejemplos.UD10.starwars;
2
3   public class Planet {
4       private int id;
5       private String name;
6       private String rotation_period;
7       private String orbital_period;
8       private String diameter;
9       private String climate;
10      private String gravity;
11      private String terrain;
12      private String surface_water;
13      private String population;
14      private String created_date;
15      private String updated_date;
16      private String url;
17
18      public Planet(int id, String name, String rotation_period, String orbital_period,
    String diameter, String climate, String gravity, String terrain, String surface_water,
    String population) {
19          this.id = id;
20          this.name = name;
21          this.rotation_period = rotation_period;
22          this.orbital_period = orbital_period;
23          this.diameter = diameter;
24          this.climate = climate;
25          this.gravity = gravity;
26          this.terrain = terrain;
27          this.surface_water = surface_water;
28          this.population = population;
29      }
```

```
30
31      public Planet(int id, String name) {
32          this.id = id;
33          this.name = name;
34      }
35
36      // Getters y setters
37      public int getId() {
38          return id;
39      }
40
41      public void setId(int id) {
42          this.id = id;
43      }
44
45      public String getName() {
46          return name;
47      }
48
49      public void setName(String name) {
50          this.name = name;
51      }
52
53      public String getRotation_period() {
54          return rotation_period;
55      }
56
57      public void setRotation_period(String rotation_period) {
58          this.rotation_period = rotation_period;
59      }
60
61      public String getOrbital_period() {
62          return orbital_period;
63      }
64
65      public void setOrbital_period(String orbital_period) {
66          this.orbital_period = orbital_period;
67      }
68
69      public String getDiameter() {
70          return diameter;
71      }
72
73      public void setDiameter(String diameter) {
74          this.diameter = diameter;
75      }
76
77      public String getClimate() {
78          return climate;
79      }
80
81      public void setClimate(String climate) {
82          this.climate = climate;
83      }
84
85      public String getGravity() {
86          return gravity;
87      }
88
89      public void setGravity(String gravity) {
90          this.gravity = gravity;
91      }
92
93      public String getTerrain() {
94          return terrain;
95      }
96
97      public void setTerrain(String terrain) {
98          this.terrain = terrain;
99      }
100
101     public String getSurface_water() {
102         return surface_water;
103     }
104
```

```
    public void setSurface_water(String surface_water) {
        this.surface_water = surface_water;
    }

    public String getPopulation() {
        return population;
    }

    public void setPopulation(String population) {
        this.population = population;
    }

    public String getCreated_date() {
        return created_date;
    }

    public void setCreated_date(String created_date) {
        this.created_date = created_date;
    }

    public String getUpdated_date() {
        return updated_date;
    }

    public void setUpdated_date(String updated_date) {
        this.updated_date = updated_date;
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    @Override
    public String toString() {
        return "Planet{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", rotation_period='" + rotation_period + '\'' +
                ", orbital_period='" + orbital_period + '\'' +
                ", diameter='" + diameter + '\'' +
                ", climate='" + climate + '\'' +
                ", gravity='" + gravity + '\'' +
                ", terrain='" + terrain + '\'' +
                ", surface_water='" + surface_water + '\'' +
                ", population='" + population + '\'' +
                ", created_date='" + created_date + '\'' +
                ", updated_date='" + updated_date + '\'' +
                ", url='" + url + '\'' +
                '}';
    }
}
```

Character

```
package es.martinezpenya.ejemplos.UD10.starwars;

import java.time.LocalDateTime;

public class Character {
    private int id;
    private String name;
    private int height;
    private double mass;
    private String hair_color;
    private String skin_color;
    private String eye_color;
    private String birth_year;
    private String gender;
    private Planet planet_id;
    private LocalDateTime created_date;
```

```
    private LocalDateTime updated_date;
    private String url;

    public Character(int id, String name, int height, double mass, String hair_color,
String skin_color, String eye_color, String birth_year, String gender, Planet planet_id)
{
        this.id = id;
        this.name = name;
        this.height = height;
        this.mass = mass;
        this.hair_color = hair_color;
        this.skin_color = skin_color;
        this.eye_color = eye_color;
        this.birth_year = birth_year;
        this.gender = gender;
        this.planet_id = planet_id;
    }

    public Character(int id, String name) {
        this.id = id;
        this.name = name;
    }

    // Getters y setters
    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getHeight() {
        return height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public double getMass() {
        return mass;
    }

    public void setMass(double mass) {
        this.mass = mass;
    }

    public String getHair_color() {
        return hair_color;
    }

    public void setHair_color(String hair_color) {
        this.hair_color = hair_color;
    }

    public String getSkin_color() {
        return skin_color;
    }

    public void setSkin_color(String skin_color) {
        this.skin_color = skin_color;
    }

    public String getEye_color() {
        return eye_color;
    }
```

```
90
91      public void setEye_color(String eye_color) {
92          this.eye_color = eye_color;
93      }
94
95      public String getBirth_year() {
96          return birth_year;
97      }
98
99      public void setBirth_year(String birth_year) {
100         this.birth_year = birth_year;
101     }
102
103     public String getGender() {
104         return gender;
105     }
106
107     public void setGender(String gender) {
108         this.gender = gender;
109     }
110
111     public Planet getPlanet_id() {
112         return planet_id;
113     }
114
115     public void setPlanet_id(Planet planet_id) {
116         this.planet_id = planet_id;
117     }
118
119     public LocalDateTime getCreated_date() {
120         return created_date;
121     }
122
123     public void setCreated_date(LocalDateTime created_date) {
124         this.created_date = created_date;
125     }
126
127     public LocalDateTime getUpdated_date() {
128         return updated_date;
129     }
130
131     public void setUpdated_date(LocalDateTime updated_date) {
132         this.updated_date = updated_date;
133     }
134
135     public String getUrl() {
136         return url;
137     }
138
139     public void setUrl(String url) {
140         this.url = url;
141     }
142
143     @Override
144     public String toString() {
145         return "Character{" +
146                 "id=" + id +
147                 ", name='" + name + '\'' +
148                 ", height=" + height +
149                 ", mass=" + mass +
150                 ", hair_color='" + hair_color + '\'' +
151                 ", skin_color='" + skin_color + '\'' +
152                 ", eye_color='" + eye_color + '\'' +
153                 ", birth_year='" + birth_year + '\'' +
154                 ", gender='" + gender + '\'' +
155                 ", planet='" + planet_id + '\'' +
156                 ", created_date=" + created_date +
157                 ", updated_date=" + updated_date +
158                 ", url='" + url + '\'' +
159                 '}';
160
161     }
162 }
```

CharacterFilm

```java
package es.martinezpenya.ejemplos.UD10.starwars;

public class CharacterFilm {
    private Character character;
    private Film film;

    public CharacterFilm(Character character, Film film) {
        this.character = character;
        this.film = film;
    }

    public Character getCharacter() {
        return character;
    }

    public void setCharacter(Character character) {
        this.character = character;
    }

    public Film getFilm() {
        return film;
    }

    public void setFilm(Film film) {
        this.film = film;
    }

    @Override
    public String toString() {
        return "CharacterFilm{" +
                "character=" + character +
                ", film=" + film +
                '}';
    }
}
```

Repite el proceso anterior para todas las clases de tu BD.

Paso 2: Definir la interface DAO

Luego, definiremos la interfaz DAO para después implementarlas en cada entidad.

DAO

```java
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.SQLException;
import java.util.ArrayList;

public interface DAO<T> {
    void crear(T entidad) throws SQLException;;          // C: Create
    T obtener(int id) throws SQLException;;              // R: Read
    ArrayList<T> obtenerTodos() throws SQLException;;    // R: Read (all)
    void actualizar(T entidad) throws SQLException;;     // U: Update
    void eliminar(int id) throws SQLException;;          // D: Delete
}
```

Paso 3: Implementar las clases DAO

Luego, implementaremos las clases DAO para cada entidad e interactuando con la base de datos.

FilmDAO

```java
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.*;
import java.util.ArrayList;

public class FilmDAO implements DAO<Film> {
    private static final String INSERT_QUERY = "INSERT INTO films (id, episode, title)
VALUES (?, ?, ?)";
```

```
    private static final String SELECT_BY_ID_QUERY = "SELECT * FROM films WHERE id = ?";
    private static final String SELECT_ALL_QUERY = "SELECT * FROM films";
    private static final String UPDATE_QUERY = "UPDATE films SET episode = ?, title = ? WHERE id = ?";
    private static final String DELETE_QUERY = "DELETE FROM films WHERE id = ?";
    private Connection con;

    public FilmDAO () {
        try {
            con = DBConnect.getConnection();
        } catch (SQLException e) {
            System.out.println("ERROR al conectar: " + e.getMessage());
        }
    }

    @Override
    public void crear(Film film) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(INSERT_QUERY, Statement.RETURN_GENERATED_KEYS)) {

            pst.setInt(1, film.getId());
            pst.setString(2, film.getEpisode());
            pst.setString(3, film.getTitle());

            int filasInsertadas = pst.executeUpdate();
            /* El siguiente fragmento serviria para el caso en que la tabla tuviera un campo autoincremental
              con pst.getGeneratedKeys() obtenemos el id generado y se lo podemos asignar al objeto
              if (filasInsertadas > 0) {
                ResultSet rs = pst.getGeneratedKeys();
                if (rs.next()) {
                    int id = rs.getInt(1);
                    film.setId(id);
                }
            }
            */
        }
    }

    @Override
    public Film obtener(int id) throws SQLException {
        Film usuario = null;
        try (PreparedStatement pst = con.prepareStatement(SELECT_BY_ID_QUERY)) {

            pst.setInt(1, id);
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                int idDevuelto = rs.getInt("id");
                String episode = rs.getString("episode");
                String title = rs.getString("title");
                usuario = new Film(id, episode, title);
            }
        }
        return usuario;
    }

    @Override
    public ArrayList<Film> obtenerTodos() throws SQLException {
        ArrayList<Film> films = new ArrayList<>();
        try (Statement st = con.createStatement();
             ResultSet rs = st.executeQuery(SELECT_ALL_QUERY)) {
            while (rs.next()) {
                int id = rs.getInt("id");
                String episode = rs.getString("episode");
                String title = rs.getString("title");
                Film film = new Film(id, episode, title);
                films.add(film);
            }
        }
        return films;
    }

    @Override
    public void actualizar(Film usuario) throws SQLException {
```

```
        try (PreparedStatement pst = con.prepareStatement(UPDATE_QUERY)) {
            pst.setString(1, usuario.getEpisode());
            pst.setString(2, usuario.getTitle());
            pst.setInt(3, usuario.getId());
            pst.executeUpdate();
        }
    }

    @Override
    public void eliminar(int id) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(DELETE_QUERY)) {
            pst.setInt(1, id);
            pst.executeUpdate();
        }
    }
}
```

PlanetDAO

```
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.*;
import java.util.ArrayList;

public class PlanetDAO implements DAO<Planet> {
    private static final String INSERT_QUERY = "INSERT INTO planets (id, name, rotation_period, orbital_period, diameter, climate, gravity, terrain, surface_water, population) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    private static final String SELECT_BY_ID_QUERY = "SELECT * FROM planets WHERE id = ?";
    private static final String SELECT_ALL_QUERY = "SELECT * FROM planets";
    private static final String UPDATE_QUERY = "UPDATE planets SET name = ?, rotation_period = ?, orbital_period = ?, diameter = ?, climate = ?, gravity = ?, terrain = ?, surface_water = ?, population = ? WHERE id = ?";
    private static final String DELETE_QUERY = "DELETE FROM planets WHERE id = ?";
    private Connection con;

    public PlanetDAO () {
        try {
            con = DBConnect.getConnection();
        } catch (SQLException e) {
            System.out.println("ERROR al conectar: " + e.getMessage());
        }
    }

    public void crear(Planet planet) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(INSERT_QUERY, Statement.RETURN_GENERATED_KEYS)) {

            pst.setInt(1, planet.getId());
            pst.setString(2, planet.getName());
            pst.setString(3, planet.getRotation_period());
            pst.setString(4, planet.getOrbital_period());
            pst.setString(5, planet.getDiameter());
            pst.setString(6, planet.getClimate());
            pst.setString(7, planet.getGravity());
            pst.setString(8, planet.getTerrain());
            pst.setString(9, planet.getSurface_water());
            pst.setString(10, planet.getPopulation());

            int filasInsertadas = pst.executeUpdate();
            if (filasInsertadas > 0) {
                ResultSet rs = pst.getGeneratedKeys();
                if (rs.next()) {
                    int id = rs.getInt(1);
                    planet.setId(id);
                }
            }
        }
    }

    public Planet obtener(int id) throws SQLException {
        Planet planet = null;
        try (PreparedStatement pst = con.prepareStatement(SELECT_BY_ID_QUERY)) {
```

```
            pst.setInt(1, id);
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                int idDevuelto = rs.getInt("id");
                String name = rs.getString("name");
                String rotation_period = rs.getString("rotation_period");
                String orbital_period = rs.getString("orbital_period");
                String diameter = rs.getString("diameter");
                String climate = rs.getString("climate");
                String gravity = rs.getString("gravity");
                String terrain = rs.getString("terrain");
                String surface_water = rs.getString("surface_water");
                String population = rs.getString("population");
                planet = new Planet(id, name, rotation_period, orbital_period, diameter,
climate, gravity, terrain, surface_water, population);
            }
        }
        return planet;
    }

    @Override
    public ArrayList<Planet> obtenerTodos() throws SQLException {
        ArrayList<Planet> planets = new ArrayList<>();
        try (Statement st = con.createStatement();
             ResultSet rs = st.executeQuery(SELECT_ALL_QUERY)) {
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                String rotation_period = rs.getString("rotation_period");
                String orbital_period = rs.getString("orbital_period");
                String diameter = rs.getString("diameter");
                String climate = rs.getString("climate");
                String gravity = rs.getString("gravity");
                String terrain = rs.getString("terrain");
                String surface_water = rs.getString("surface_water");
                String population = rs.getString("population");
                Planet planet = new Planet(id, name, rotation_period, orbital_period,
diameter, climate, gravity, terrain, surface_water, population);
                planets.add(planet);
            }
        }
        return planets;
    }

    public void actualizar(Planet planet) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(UPDATE_QUERY)) {

            pst.setString(1, planet.getName());
            pst.setString(2, planet.getRotation_period());
            pst.setString(3, planet.getOrbital_period());
            pst.setString(4, planet.getDiameter());
            pst.setString(5, planet.getClimate());
            pst.setString(6, planet.getGravity());
            pst.setString(7, planet.getTerrain());
            pst.setString(8, planet.getSurface_water());
            pst.setString(9, planet.getPopulation());
            pst.setInt(10, planet.getId());

            pst.executeUpdate();
        }
    }

    public void eliminar(int id) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(DELETE_QUERY)) {

            pst.setInt(1, id);
            pst.executeUpdate();
        }
    }
}
```

CharacterDAO

```java
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.*;
import java.util.ArrayList;

public class CharacterDAO implements DAO<Character> {
    private static final String INSERT_QUERY = "INSERT INTO characters (id, name, height, mass, hair_color, skin_color, eye_color, birth_year, gender, planet_id) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
    private static final String SELECT_BY_ID_QUERY = "SELECT * FROM characters WHERE id = ?";
    private static final String SELECT_ALL_QUERY = "SELECT * FROM characters";
    private static final String UPDATE_QUERY = "UPDATE characters SET name = ?, height = ?, mass = ?, hair_color = ?, skin_color = ?, eye_color = ?, birth_year = ?, gender= ?, planet_id = ? WHERE id = ?";
    private static final String DELETE_QUERY = "DELETE FROM characters WHERE id = ?";
    private Connection con;

    public CharacterDAO() {
        try {
            con = DBConnect.getConnection();
        } catch (SQLException e) {
            System.out.println("ERROR al conectar: " + e.getMessage());
        }
    }

    public void crear(Character character) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(INSERT_QUERY, Statement.RETURN_GENERATED_KEYS)) {

            pst.setInt(1, character.getId());
            pst.setString(2, character.getName());
            pst.setInt(3, character.getHeight());
            pst.setDouble(4, character.getMass());
            pst.setString(5, character.getHair_color());
            pst.setString(6, character.getSkin_color());
            pst.setString(7, character.getEye_color());
            pst.setString(8, character.getBirth_year());
            pst.setString(9, character.getGender());
            pst.setInt(10, character.getPlanet_id().getId());

            int filasInsertadas = pst.executeUpdate();
            if (filasInsertadas > 0) {
                ResultSet rs = pst.getGeneratedKeys();
                if (rs.next()) {
                    int id = rs.getInt(1);
                    character.setId(id);
                }
            }
        }
    }

    @Override
    public Character obtener(int id) throws SQLException {
        Character character = null;
        try (PreparedStatement pst = con.prepareStatement(SELECT_BY_ID_QUERY)) {

            pst.setInt(1, id);
            PlanetDAO planetDAO = new PlanetDAO();
            ResultSet rs = pst.executeQuery();
            if (rs.next()) {
                int idDevuelto = rs.getInt("id");
                String name = rs.getString("name");
                int height = rs.getInt("height");
                double mass = rs.getDouble("mass");
                String hair_color = rs.getString("hair_color");
                String skin_color = rs.getString("skin_color");
                String eye_color = rs.getString("eye_color");
                String birth_year = rs.getString("birth_year");
                String gender = rs.getString("gender");
                Planet planet = planetDAO.obtener(rs.getInt("planet_id")); //buscamos el planeta por su id
                character = new Character(id, name, height, mass, hair_color, skin_color, eye_color, birth_year, gender, planet);
            }
```

```
        }
        return character;
    }

    @Override
    public ArrayList<Character> obtenerTodos() throws SQLException {
        ArrayList<Character> characters = new ArrayList<>();
        PlanetDAO planetDAO = new PlanetDAO();
        try (Statement st = con.createStatement();
             ResultSet rs = st.executeQuery(SELECT_ALL_QUERY)) {
            while (rs.next()) {
                int id = rs.getInt("id");
                String name = rs.getString("name");
                int height = rs.getInt("height");
                double mass = rs.getDouble("mass");
                String hair_color = rs.getString("hair_color");
                String skin_color = rs.getString("skin_color");
                String eye_color = rs.getString("eye_color");
                String birth_year = rs.getString("birth_year");
                String gender = rs.getString("gender");
                int planet_id = rs.getInt("planet_id");
                Planet planet = null;
                if (!rs.wasNull()){
                    planet = planetDAO.obtener(planet_id); //buscamos el planeta por su
id
                }
                Character character = new Character(id, name, height, mass, hair_color,
skin_color, eye_color, birth_year, gender, planet);
                characters.add(character);
            }
        }
        return characters;
    }

    @Override
    public void actualizar(Character entidad) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(UPDATE_QUERY)) {
            pst.setString(1, entidad.getName());
            pst.setInt(2, entidad.getHeight());
            pst.setDouble(3, entidad.getMass());
            pst.setString(4, entidad.getHair_color());
            pst.setString(5, entidad.getSkin_color());
            pst.setString(6, entidad.getEye_color());
            pst.setString(7, entidad.getBirth_year());
            pst.setString(8, entidad.getGender());
            pst.setInt(9, entidad.getPlanet_id().getId());
            pst.setInt(10, entidad.getId());
            pst.executeUpdate();
        }
    }

    @Override
    public void eliminar(int id) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(DELETE_QUERY)) {
            pst.setInt(1, id);
            pst.executeUpdate();
        }
    }

}
```

CharacterFilmDAO

```
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.*;
import java.util.ArrayList;

public class CharacterFilmDAO implements DAO<CharacterFilm> {
    private static final String INSERT_QUERY = "INSERT INTO character_films
(id_character, id_film) VALUES (?, ?)";
    private static final String SELECT_ALL_QUERY_BY_CHARACTER = "SELECT * FROM
character_films WHERE id_character = ?";
```

```
    private static final String SELECT_ALL_QUERY_BY_FILM = "SELECT * FROM character_films WHERE id_film = ?";
    private static final String DELETE_QUERY_BY_CHARACTER = "DELETE FROM character_films WHERE id_character = ?";
    private static final String DELETE_QUERY_BY_FILM = "DELETE FROM character_films WHERE id_film = ?";
    public static final String DELETE_QUERY_BY_CHARACTER_AND_FILM = "DELETE FROM character_films WHERE id_character = ? AND id_film = ?";

    private Connection con;

    public CharacterFilmDAO() {
        try {
            con = DBConnect.getConnection();
        } catch (SQLException e) {
            System.out.println("ERROR al conectar: " + e.getMessage());
        }
    }
    @Override
    public void crear(CharacterFilm entidad) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(INSERT_QUERY, Statement.RETURN_GENERATED_KEYS)) {

            pst.setInt(1, entidad.getCharacter().getId());
            pst.setInt(2, entidad.getFilm().getId());

            ResultSet rs = pst.executeQuery();
        }
    }

    @Override
    public CharacterFilm obtener(int id) throws SQLException {
        //No tiene sentido en una tabla N a N
        return null;
    }

    @Override
    public ArrayList<CharacterFilm> obtenerTodos() throws SQLException {
        //No tiene sentido en una tabla N a N
        return null;
    }

    public ArrayList<CharacterFilm> obtenerTodosPorCharacter(int id_character) throws SQLException {
        ArrayList<CharacterFilm> characterFilms = new ArrayList<>();
        try (PreparedStatement pst = con.prepareStatement(SELECT_ALL_QUERY_BY_CHARACTER)) {
            pst.setInt(1, id_character);
            ResultSet rs = pst.executeQuery();
            CharacterDAO characterDAO = new CharacterDAO();
            Character character = characterDAO.obtener(id_character);
            FilmDAO filmDAO = new FilmDAO();
            while (rs.next()) {
                int id_film = rs.getInt("id_film");
                Film film = filmDAO.obtener(id_film);
                characterFilms.add(new CharacterFilm(character, film));
            }
        }
        return characterFilms;
    }

    public ArrayList<CharacterFilm> obtenerTodosPorFilm(int id_film) throws SQLException {
        ArrayList<CharacterFilm> characterFilms = new ArrayList<>();
        try (PreparedStatement pst = con.prepareStatement(SELECT_ALL_QUERY_BY_FILM)) {
            pst.setInt(1, id_film);
            ResultSet rs = pst.executeQuery();
            CharacterDAO characterDAO = new CharacterDAO();
            FilmDAO filmDAO = new FilmDAO();
            Film film = filmDAO.obtener(id_film);
            while (rs.next()) {
                int id_character = rs.getInt("id_character");
                Character character = characterDAO.obtener(id_character);
                characterFilms.add(new CharacterFilm(character, film));
            }
```

```
        }
        return characterFilms;
    }

    @Override
    public void actualizar(CharacterFilm entidad) throws SQLException {
        //No tiene sentido en una tabla N a N
    }

    @Override
    public void eliminar(int id) throws SQLException {
        //No tiene sentido en una tabla N a N
    }

    public void eliminarPorCharacterYFilm(int idCharacter, int idFilm) throws
SQLException {
        try (PreparedStatement pst =
con.prepareStatement(DELETE_QUERY_BY_CHARACTER_AND_FILM)) {
            pst.setInt(1, idCharacter);
            pst.setInt(2, idFilm);
            pst.executeUpdate();
        }
    }

    public void eliminarPorCharacter(int id) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(DELETE_QUERY_BY_CHARACTER)) {
            pst.setInt(1, id);
            pst.executeUpdate();
        }
    }

    public void eliminarPorFilm(int id) throws SQLException {
        try (PreparedStatement pst = con.prepareStatement(DELETE_QUERY_BY_FILM)) {
            pst.setInt(1, id);
            pst.executeUpdate();
        }
    }
}
```

Paso 4: Implementar la lógica de la aplicación

Finalmente, implementaremos la lógica de la aplicación en una clase principal `Main` donde podremos interactuar con los DAOs y la base de datos.

```
package es.martinezpenya.ejemplos.UD10.starwars;

import java.sql.Connection;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        try (Connection con = DBConnect.getConnection()) {
            while (true) {
                menuPrincipal();
                int opcion = entrada.nextInt();
                entrada.nextLine(); // Limpiar el buffer del entrada
                switch (opcion) {
                    case 1:
                        gestionarFilms();
                        break;
                    case 2:
                        gestionarCharacters();
                        break;
                    case 3:
                        gestionarCharacterFilms();
                        break;
                    case 4:
                        gestionarPlanetas();
                        break;
                    case 0:
                        System.out.println("Saliendo...");
```

```
                        entrada.close();
                        System.exit(0);
                    default:
                        System.out.println("Opción inválida. Intenta de nuevo.");
                }
            }
        } catch (SQLException e) {
            System.out.println("Error en la conexión con la base de datos: " +
e.getMessage());
        }
    }

    private static void gestionarCharacterFilms() {
        Scanner entrada = new Scanner(System.in);
        CharacterFilmDAO characterFilmDAO = new CharacterFilmDAO();
        while (true) {
            menuCharacterFilms();
            int opcion = entrada.nextInt();
            entrada.nextLine(); // Limpiar el buffer del entrada
            switch (opcion) {
                case 1:
                    System.out.print("Introduce el id del character: ");
                    int id_character = entrada.nextInt();
                    entrada.nextLine();
                    System.out.print("Introduce el id del film: ");
                    int id_film = entrada.nextInt();
                    entrada.nextLine();
                    Character character = null;
                    Film film = null;
                    try {
                        character = new CharacterDAO().obtener(id_character);
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el character: " +
e.getMessage());
                    }
                    try {
                        film = new FilmDAO().obtener(id_film);
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el film: " +
e.getMessage());
                    }
                    if (character == null || film == null) {
                        System.out.println("No se pudo crear el CharacterFilm porque el
Character o el Film no existen.");

                    } else {
                        CharacterFilm nuevoCharacterFilm = new CharacterFilm(character,
film);
                        try {
                            characterFilmDAO.crear(nuevoCharacterFilm);
                            System.out.println("CharacterFilm creado");
                        } catch (SQLException e) {
                            System.out.println("Error al crear characterFilm: " +
e.getMessage());
                        }
                    }
                    break;
                case 2:
                    System.out.print("Introduce el ID del Character para ver todos los
Films en los que aparece: ");
                    int idCharacter = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        ArrayList<CharacterFilm> characterFilms =
characterFilmDAO.obtenerTodosPorCharacter(idCharacter);
                        if (!characterFilms.isEmpty()) {
                            System.out.println("Listado de CharacterFilm:");
                            for (CharacterFilm cf : characterFilms) {
                                System.out.println(cf);
                            }
                        } else {
                            System.out.println("No hay CharacterFilm registrados.");
                        }
                    } catch (SQLException e) {
```

```
                        System.out.println("Error al obtener todos los CharacterFilm del Character: " + e.getMessage());
                    }
                    break;
                case 3:
                    System.out.print("Introduce el ID del Film para ver todos los Characters que aparecen: ");
                    int idFilm = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        ArrayList<CharacterFilm> characterFilms = characterFilmDAO.obtenerTodosPorFilm(idFilm);
                        if (!characterFilms.isEmpty()) {
                            System.out.println("Listado de CharacterFilm:");
                            for (CharacterFilm cf : characterFilms) {
                                System.out.println(cf);
                            }
                        } else {
                            System.out.println("No hay CharacterFilm registrados.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener todos los CharacterFilm del Film: " + e.getMessage());
                    }
                    break;
                case 4:
                    System.out.print("Introduce el ID del Character para eliminar todos los CharacterFilm asociados: ");
                    int idCharacterEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        characterFilmDAO.eliminarPorCharacter(idCharacterEliminar);
                        System.out.println("CharacterFilm eliminados correctamente.");
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar CharacterFilm: " + e.getMessage());
                    }
                    break;
                case 5:
                    System.out.print("Introduce el ID del Film para eliminar todos los CharacterFilm asociados: ");
                    int idFilmEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        characterFilmDAO.eliminarPorFilm(idFilmEliminar);
                        System.out.println("CharacterFilm eliminados correctamente.");
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar CharacterFilm: " + e.getMessage());
                    }
                    break;
                case 6:
                    System.out.print("Introduce el ID del Character del CharacterFilm a eliminar: ");
                    int idCharacterCharacterFilmEliminar = entrada.nextInt();
                    System.out.print("Introduce el ID del Film del CharacterFilm a eliminar: ");
                    int idFilmCharacterFilmEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {

                     characterFilmDAO.eliminarPorCharacterYFilm(idCharacterCharacterFilmEliminar, idFilmCharacterFilmEliminar);
                        System.out.println("CharacterFilm eliminado correctamente.");
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar CharacterFilm: " + e.getMessage());
                    }
                    break;
            }
        }
    }

    private static void gestionarFilms() {
        Scanner entrada = new Scanner(System.in);
```

```
        FilmDAO filmDAO = new FilmDAO();
        while (true) {
            menuFilms();
            int opcion = entrada.nextInt();
            entrada.nextLine(); // Limpiar el buffer del entrada
            switch (opcion) {
                case 1:
                    System.out.print("Introduce el id del film: ");
                    int id = entrada.nextInt();
                    entrada.nextLine();
                    System.out.print("Introduce el episode del film: ");
                    String episode = entrada.nextLine();
                    System.out.print("Introduce el title del film: ");
                    String title = entrada.nextLine();
                    Film nuevoFilm = new Film(id, episode, title); // El ID se genera
automáticamente
                    try {
                        filmDAO.crear(nuevoFilm);
                        System.out.println("Usuario creado con ID: " +
nuevoFilm.getId());
                    } catch (SQLException e) {
                        System.out.println("Error al crear usuario: " + e.getMessage());
                    }
                    break;
                case 2:
                    System.out.print("Introduce el ID del film a consultar: ");
                    int idFilm = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Film film = filmDAO.obtener(idFilm);
                        if (film != null) {
                            System.out.println("Film encontrado:");
                            System.out.println(film);
                        } else {
                            System.out.println("No se encontró ningún film con ese ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el film: " +
e.getMessage());
                    }
                    break;
                case 3:
                    try {
                        ArrayList<Film> films = filmDAO.obtenerTodos();
                        if (!films.isEmpty()) {
                            System.out.println("Listado de Usuarios:");
                            for (Film u : films) {
                                System.out.println(u);
                            }
                        } else {
                            System.out.println("No hay films registrados.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener todos los films: " +
e.getMessage());
                    }
                    break;
                case 4:
                    System.out.print("Introduce el ID del film a actualizar: ");
                    int idActualizar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Film filmActualizar = filmDAO.obtener(idActualizar);
                        if (filmActualizar != null) {
                            System.out.print("Introduce el nuevo episode del film (deja
en blanco para mantener el actual): ");
                            String nuevoEpisode = entrada.nextLine();
                            if (!nuevoEpisode.isEmpty()) {
                                filmActualizar.setEpisode(nuevoEpisode);
                            }
                            System.out.print("Introduce el nuevo title del film (deja en
blanco para mantener el actual): ");
                            String nuevoTitle = entrada.nextLine();
                            if (!nuevoTitle.isEmpty()) {
                                filmActualizar.setTitle(nuevoTitle);
```

```
                            }
                            filmDAO.actualizar(filmActualizar);
                            System.out.println("Film actualizado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún Film con ese ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al actualizar Film: " +
e.getMessage());
                    }
                    break;
                case 5:
                    System.out.print("Introduce el ID del film a eliminar: ");
                    int idEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Film filmEliminar = filmDAO.obtener(idEliminar);
                        if (filmEliminar != null) {
                            // eliminar primero los character_films y deaths asociados al
film
                            //
Character_filmDAO.eliminarCharacter_filmsPorFilm(idEliminar);
                            // luego eliminar el film
                            filmDAO.eliminar(idEliminar);
                            System.out.println("Film eliminado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún film con ese ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar film: " + e.getMessage());
                    }
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Opción inválida. Intenta de nuevo.");
            }
        }
    }

    private static void gestionarPlanetas() {
        Scanner entrada = new Scanner(System.in);
        PlanetDAO planetDAO = new PlanetDAO();
        while (true) {
            menuPlanets();
            int opcion = entrada.nextInt();
            entrada.nextLine(); // Limpiar el buffer del entrada
            switch (opcion) {
                case 1:
                    System.out.print("Introduce el id del planeta: ");
                    int id = entrada.nextInt();
                    entrada.nextLine();
                    System.out.print("Introduce el nombre del planeta: ");
                    String name = entrada.nextLine();
                    System.out.print("Introduce el rotation_period del planeta: ");
                    String rotation_period = entrada.nextLine();
                    System.out.print("Introduce el orbital_period del planeta: ");
                    String orbital_period = entrada.nextLine();
                    System.out.print("Introduce el diameter del planeta: ");
                    String diameter = entrada.nextLine();
                    System.out.print("Introduce el climate del planeta: ");
                    String climate = entrada.nextLine();
                    System.out.print("Introduce el gravity del planeta: ");
                    String gravity = entrada.nextLine();
                    System.out.print("Introduce el terrain del planeta: ");
                    String terrain = entrada.nextLine();
                    System.out.print("Introduce el surface_water del planeta: ");
                    String surface_water = entrada.nextLine();
                    System.out.print("Introduce el population del planeta: ");
                    String population = entrada.nextLine();
                    Planet nuevoPlanet = new Planet(id, name, rotation_period,
orbital_period, diameter, climate, gravity, terrain, surface_water, population); // El ID
se genera automáticamente
                    try {
                        planetDAO.crear(nuevoPlanet);
```

```
                        System.out.println("Planeta creado con ID: " +
nuevoPlanet.getId());
                    } catch (SQLException e) {
                        System.out.println("Error al crear planeta: " + e.getMessage());
                    }
                    break;
                case 2:
                    System.out.print("Introduce el ID del planeta a consultar: ");
                    int idPlanet = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Planet planet = planetDAO.obtener(idPlanet);
                        if (planet != null) {
                            System.out.println("Planeta encontrado:");
                            System.out.println(planet);
                        } else {
                            System.out.println("No se encontró ningún planeta con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el planeta: " +
e.getMessage());
                    }
                    break;
                case 3:
                    try {
                        ArrayList<Planet> planets = planetDAO.obtenerTodos();
                        if (!planets.isEmpty()) {
                            System.out.println("Listado de Planetas:");
                            for (Planet u : planets) {
                                System.out.println(u);
                            }
                        } else {
                            System.out.println("No hay planetas registrados.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener todos los planetas: " +
e.getMessage());
                    }
                    break;
                case 4:
                    System.out.print("Introduce el ID del planeta a actualizar: ");
                    int idActualizar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Planet planetActualizar = planetDAO.obtener(idActualizar);
                        if (planetActualizar != null) {
                            System.out.print("Introduce el nuevo nombre del planeta (deja
en blanco para mantener el actual): ");
                            String nuevoName = entrada.nextLine();
                            if (!nuevoName.isEmpty()) {
                                planetActualizar.setName(nuevoName);
                            }
                            System.out.print("Introduce el nuevo rotation_period del
planeta (deja en blanco para mantener el actual): ");
                            String nuevoRotation_period = entrada.nextLine();
                            if (!nuevoRotation_period.isEmpty()) {

 planetActualizar.setRotation_period(nuevoRotation_period);
                            }
                            System.out.print("Introduce el nuevo orbital_period del
planeta (deja en blanco para mantener el actual): ");
                            String nuevoOrbital_period = entrada.nextLine();
                            if (!nuevoOrbital_period.isEmpty()) {
                                planetActualizar.setOrbital_period(nuevoOrbital_period);
                            }
                            System.out.print("Introduce el nuevo diameter del planeta
(deja en blanco para mantener el actual): ");
                            String nuevoDiameter = entrada.nextLine();
                            if (!nuevoDiameter.isEmpty()) {
                                planetActualizar.setDiameter(nuevoDiameter);
                            }
                            System.out.print("Introduce el nuevo climate del planeta
(deja en blanco para mantener el actual): ");
                            String nuevoClimate = entrada.nextLine();
```

```
                            if (!nuevoClimate.isEmpty()) {
                                planetActualizar.setClimate(nuevoClimate);
                            }
                            System.out.print("Introduce el nuevo gravity del planeta
(deja en blanco para mantener el actual): ");
                            String nuevoGravity = entrada.nextLine();
                            if (!nuevoGravity.isEmpty()) {
                                planetActualizar.setGravity(nuevoGravity);
                            }
                            System.out.print("Introduce el nuevo terrain del planeta
(deja en blanco para mantener el actual): ");
                            String nuevoTerrain = entrada.nextLine();
                            if (!nuevoTerrain.isEmpty()) {
                                planetActualizar.setTerrain(nuevoTerrain);
                            }
                            System.out.print("Introduce el nuevo surface_water del
planeta (deja en blanco para mantener el actual): ");
                            String nuevoSurface_water = entrada.nextLine();
                            if (!nuevoSurface_water.isEmpty()) {
                                planetActualizar.setSurface_water(nuevoSurface_water);
                            }
                            System.out.print("Introduce el nuevo population del planeta
(deja en blanco para mantener el actual): ");
                            String nuevoPopulation = entrada.nextLine();
                            if (!nuevoPopulation.isEmpty()) {
                                planetActualizar.setPopulation(nuevoPopulation);
                            }
                            planetDAO.actualizar(planetActualizar);
                            System.out.println("Planeta actualizado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún Planeta con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al actualizar Planeta: " +
e.getMessage());
                    }
                    break;
                case 5:
                    System.out.print("Introduce el ID del planeta a eliminar: ");
                    int idEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Planet planetEliminar = planetDAO.obtener(idEliminar);
                        if (planetEliminar != null) {
                            // todo: Revisar si quiero eliminar Characters al eliminar su
Planeta Origen.
                            planetDAO.eliminar(idEliminar);
                            System.out.println("Planeta eliminado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún planeta con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar planeta: " +
e.getMessage());
                    }
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Opción inválida. Intenta de nuevo.");
            }
        }
    }

    private static void gestionarCharacters() {
        Scanner entrada = new Scanner(System.in);
        CharacterDAO characterDAO = new CharacterDAO();
        while (true) {
            menuCharacters();
            PlanetDAO planetDAO = new PlanetDAO();
            int opcion = entrada.nextInt();
            entrada.nextLine(); // Limpiar el buffer del entrada
            switch (opcion) {
```

```
                case 1:
                    System.out.print("Introduce el id del character: ");
                    int id = entrada.nextInt();
                    entrada.nextLine();
                    System.out.print("Introduce el name del character: ");
                    String name = entrada.nextLine();
                    System.out.print("Introduce el height del character: ");
                    int height = Integer.parseInt(entrada.nextLine());
                    System.out.print("Introduce el mass del character: ");
                    int mass = Integer.parseInt(entrada.nextLine());
                    System.out.print("Introduce el hair_color del character: ");
                    String hair_color = entrada.nextLine();
                    System.out.print("Introduce el skin_color del character: ");
                    String skin_color = entrada.nextLine();
                    System.out.print("Introduce el eye_color del character: ");
                    String eye_color = entrada.nextLine();
                    System.out.print("Introduce el birth_year del character: ");
                    String birth_year = entrada.nextLine();
                    System.out.print("Introduce el gender del character: ");
                    String gender = entrada.nextLine();
                    System.out.print("Introduce el id del planeta del character: ");
                    int planet_id = Integer.parseInt(entrada.nextLine());
                    Character nuevoCharacter = null;
                    try {
                        nuevoCharacter = new Character(id, name, height, mass,
skin_color, hair_color, eye_color, birth_year, gender, planetDAO.obtener(planet_id));
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el planeta: " +
e.getMessage());
                    }
                    try {
                        characterDAO.crear(nuevoCharacter);
                        System.out.println("Character creado con ID: " +
nuevoCharacter.getId());
                    } catch (SQLException e) {
                        System.out.println("Error al crear character: " +
e.getMessage());
                    }
                    break;
                case 2:
                    System.out.print("Introduce el ID del character a consultar: ");
                    int idCharacter = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Character character = characterDAO.obtener(idCharacter);
                        if (character != null) {
                            System.out.println("Character encontrado:");
                            System.out.println(character);
                        } else {
                            System.out.println("No se encontró ningún character con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener el character: " +
e.getMessage());
                    }
                    break;
                case 3:
                    try {
                        ArrayList<Character> characters = characterDAO.obtenerTodos();
                        if (!characters.isEmpty()) {
                            System.out.println("Listado de Characters:");
                            for (Character u : characters) {
                                System.out.println(u);
                            }
                        } else {
                            System.out.println("No hay characters registrados.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al obtener todos los characters: " +
e.getMessage());
                    }
                    break;
                case 4:
                    System.out.print("Introduce el ID del character a actualizar: ");
```

```
                    int idActualizar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Character characterActualizar =
characterDAO.obtener(idActualizar);
                        if (characterActualizar != null) {
                            System.out.print("Introduce el nuevo name del character (deja
en blanco para mantener el actual): ");
                            String nuevoName = entrada.nextLine();
                            if (!nuevoName.isEmpty()) {
                                characterActualizar.setName(nuevoName);
                            }
                            System.out.print("Introduce el nuevo height del character
(deja en blanco para mantener el actual): ");
                            String nuevoHeight = entrada.nextLine();
                            if (!nuevoHeight.isEmpty()) {

 characterActualizar.setHeight(Integer.parseInt(nuevoHeight));
                            }
                            System.out.print("Introduce el nuevo mass del character (deja
en blanco para mantener el actual): ");
                            String nuevoMass = entrada.nextLine();
                            if (!nuevoMass.isEmpty()) {

 characterActualizar.setMass(Double.parseDouble(nuevoMass));
                            }
                            System.out.print("Introduce el nuevo hair_color del character
(deja en blanco para mantener el actual): ");
                            String nuevoHair_color = entrada.nextLine();
                            if (!nuevoHair_color.isEmpty()) {
                                characterActualizar.setHair_color(nuevoHair_color);
                            }
                            System.out.print("Introduce el nuevo skin_color del character
(deja en blanco para mantener el actual): ");
                            String nuevoSkin_color = entrada.nextLine();
                            if (!nuevoSkin_color.isEmpty()) {
                                characterActualizar.setSkin_color(nuevoSkin_color);
                            }
                            System.out.print("Introduce el nuevo eye_color del character
(deja en blanco para mantener el actual): ");
                            String nuevoEye_color = entrada.nextLine();
                            if (!nuevoEye_color.isEmpty()) {
                                characterActualizar.setEye_color(nuevoEye_color);
                            }
                            System.out.print("Introduce el nuevo birth_year del character
(deja en blanco para mantener el actual): ");
                            String nuevoBirth_year = entrada.nextLine();
                            if (!nuevoBirth_year.isEmpty()) {
                                characterActualizar.setBirth_year(nuevoBirth_year);
                            }
                            System.out.print("Introduce el nuevo gender del character
(deja en blanco para mantener el actual): ");
                            String nuevoGender = entrada.nextLine();
                            if (!nuevoGender.isEmpty()) {
                                characterActualizar.setGender(nuevoGender);
                            }
                            System.out.print("Introduce el nuevo id del planeta del
character (deja en blanco para mantener el actual): ");
                            String nuevoPlanet_id = entrada.nextLine();
                            if (!nuevoPlanet_id.isEmpty()) {
                                Planet p;
                                if
((p=planetDAO.obtener(Integer.parseInt(nuevoPlanet_id)))!=null) {
                                    characterActualizar.setPlanet_id(p);
                                }
                            }
                            characterDAO.actualizar(characterActualizar);
                            System.out.println("Character actualizado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún Character con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al actualizar Character: " +
e.getMessage());
```

```
                    }
                    break;
                case 5:
                    System.out.print("Introduce el ID del character a eliminar: ");
                    int idEliminar = entrada.nextInt();
                    entrada.nextLine(); // Limpiar el buffer del entrada
                    try {
                        Character characterEliminar = characterDAO.obtener(idEliminar);
                        if (characterEliminar != null) {
                            characterDAO.eliminar(idEliminar);
                            System.out.println("Character eliminado correctamente.");
                        } else {
                            System.out.println("No se encontró ningún character con ese
ID.");
                        }
                    } catch (SQLException e) {
                        System.out.println("Error al eliminar character: " +
e.getMessage());
                    }
                    break;
                case 0:
                    return;
                default:
                    System.out.println("Opción inválida. Intenta de nuevo.");
            }
        }
    }

    // MENÚ PRINCIPAL: menuPrincipal()
    private static void menuPrincipal() {
        System.out.println("\nMenú Principal:");
        System.out.println("1. Gestionar Films");
        System.out.println("2. Gestionar Characters");
        System.out.println("3. Gestionar Character_films");
        System.out.println("4. Gestionar Planets");
        System.out.println("0. Salir");
        System.out.print("Selecciona una opción:  ");
    }

    // MENÚ SECUNDARIO: menuFilms()
    private static void menuFilms() {
        System.out.println("\nMenú de Films:");
        System.out.println("1. Crear film");
        System.out.println("2. Consultar film por ID");
        System.out.println("3. Listar todos los films");
        System.out.println("4. Actualizar film");
        System.out.println("5. Eliminar film");
        System.out.println("0. Volver al menú principal");
        System.out.print("Selecciona una opción:  ");
    }

    //MENU SECUNDARIO: menuPlanets()
    private static void menuPlanets() {
        System.out.println("\nMenú de Planets:");
        System.out.println("1. Crear planeta");
        System.out.println("2. Consultar planeta por ID");
        System.out.println("3. Listar todos los planetas");
        System.out.println("4. Actualizar planeta");
        System.out.println("5. Eliminar planeta");
        System.out.println("0. Volver al menú principal");
        System.out.print("Selecciona una opción:  ");
    }

    // MENÚ SECUNDARIO: menuCharacters()
    private static void menuCharacters() {
        System.out.println("\nMenú de Characters:");
        System.out.println("1. Crear character");
        System.out.println("2. Consultar character por ID");
        System.out.println("3. Listar todos los characters");
        System.out.println("4. Actualizar character");
        System.out.println("5. Eliminar character");
        System.out.println("0. Volver al menú principal");
        System.out.print("Selecciona una opción:  ");
    }

```

```
630     // MENÚ SECUNDARIO: menuCharacterFilms()
631     private static void menuCharacterFilms() {
632         System.out.println("\nMenú de CharacterFilm:");
633         System.out.println("1. Crear characterFilm");
634         System.out.println("2. Consultar characterFilm por Character");
635         System.out.println("3. Consultar characterFilm por Film");
636         System.out.println("4. Eliminar characterFilm por Character");
637         System.out.println("5. Eliminar characterFilm por Film");
638         System.out.println("6. Eliminar characterFilm por Character y Film");
639         System.out.println("0. Volver al menú principal");
640         System.out.print("Selecciona una opción:  ");
641     }
642 }
```

Consideraciones

- **HikariCP y SQL Queries:** Hemos utilizado HikariCP para conectarnos y realizar operaciones en la base de datos. Es importante manejar excepciones y cerrar correctamente las conexiones y recursos.
- **Patrón DAO:** Este patrón nos ayuda a mantener un código organizado y aislado, separando la lógica de acceso a datos de la lógica de negocio.
- **Lógica de la aplicación:** En la clase `Main`, hemos implementado un menú interactivo que permite al usuario gestionar *films*, *characters*, *planets* y *character_films* utilizando los métodos proporcionados por los DAO's.
- **Adaptabilidad:** Puedes expandir este ejemplo completando las tablas que faltan, añadiendo más funcionalidades o haciendo ajustes según los requisitos de tu aplicación.
- **Fuera de la norma**: Fíjate en la lógica de la tabla `character_films` (tabla N a N), fíjate que no sigue exactamente el patrón DAO, pero eso no es un problema, la interfaz DAO marca un mínimo, pero no es completa, la podemos expandir con nuevas funcionalidades.
- **Eliminación y actualización en cascada**: En este ejemplo no se han contemplado estos casos, se podrían gestionar directamente con el gestor de las bases de datos, al definir las tablas y su comportamiento, o bien simular esta funcionalidad desde sentencias SQL en la lógica del programa.
- **Autonuméricos**: El ejemplo que se ha desarrollado más arriba no tiene ninguna tabla con un campo autoincremental (como suelen tener algunas claves primarias), pero en el código fuente del método `crear` de la clase `FilmDAO` puedes ver un fragmento comentado sobre como podrias gestionar el echo de que no sepas el identificador del registro que insertas en la BBDD hasta que realmente se haya insertado, como recuperarlo y mostrarlo al usuario.
- **Tablas Relacionadas**: Fíjate en el comportamiento que tiene la clase `Character` respecto del `Planet` (o el resto de relaciones), no solo se muestra el `planet_id` de la tabla `Character`, sino que se recupera también el resto de información del planeta.
- **Localización de la BBDD**. En este caso he apostado por alojar la BBDD en un servidor AWS, pero si lo prefieres puedes hacerlo localmente. Eso si, asegurate que le das toda la información necesaria al docente para que pueda reproducir tu entorno y poder evaluar que tu proyecto funciona como se espera.
- **Rendimiento**: En según que casos, si las tablas tienen muchísima información pueden provocar salidas muy lentas, o incluso errores de *timeout*.

4. Actividades

Teniendo todo lo anterior en cuenta debes...

1. Crear un nuevo proyecto con un nombre identificativo que incluya tu nombre y/o apellidos. En mi caso podria llamarse "`StarWarsDavid`".
2. Sigue las instrucciones para crear las clases básicas para cada una de las tablas de tu BBDD.
3. Genera la clase de conexión a la BBDD.
4. Define la interfaz para gestionar el patrón `DAO`.
5. Implementa las clases que a su vez implementan la interfaz `DAO` para cada una de las tablas.
6. Genera una clase con el `main` que hará que todo funcione con menús y gestión completa del CRUD de tu BBDD.
7. Añade un archivo `.sql` que permita reconstruir la BBDD desde cero, debe incluir tanto el DDL como la inserción de información en las tablas.

Envía a la tarea de Aules:

1. Una memoria en fichero **pdf** explicando los pasos seguidos (con capturas), explicando la estructura de la BBDD, las tablas que vas a implementar, y toda la información que creas relevante, así como tu opinión personal sobre el proyecto (dificultades, futuras ampliaciones, limitaciones conocidas, etc.)
2. Un fichero **zip** con el proyecto gestionado con `maven` en **IntelliJ**. Asegurate de incluir todo lo necesario para que el profesor pueda reproducir y poner en funcionamiento el proyecto. No olvides el fichero `.sql` (o similar) para poder generar la BBDD de nuevo. Este archivo generalmente lo puedes obtener exportando la BBDD o generando una copia de seguridad en SQL.

5. Fuentes de información

- Wikipedia
- Programación (Grado Superior) - Juan Carlos Moreno Pérez (Ed. Ra-ma)
- Apuntes IES Henri Matisse (Javi García Jimenez?)
- Apuntes AulaCampus
- Apuntes José Luis Comesaña
- Apuntes IOC Programació bàsica (Joan Arnedo Moreno)
- Apuntes IOC Programació Orientada a Objectes (Joan Arnedo Moreno)
- https://arturoblasco.github.io/prg