VirreyFlix

Ejercicio con Hibernate

Revisión de EjemploHibernate2

Revisa el ejemplo 2, y mira cómo se definen las asociaciones entre entidades, y los tipos que hay (@OneToOne, @OneToMany, @ManyToOne, @ManyToMany), así como se ha realizado un menú para llevar a cabo la gestión de diferentes entidades separadas, así cómo se ha realizado una clase EjemploMenu.java, en la cual se implementa un menú clásico por consola para acceder a los datos según el usuario va solicitando.

Creación de base de datos Hibernate VirreyFlix

- Crea un proyecto nuevo llamado VirreyFlix.
- Asocia las dependencias Hibernate configura los archivos necesarios.
- Genera las entidades de las tablas que hay más abajo. Primero define las @Entity sin contar con las relaciones con todos sus campos. Después, ve analizando las relaciones que hay entre cada una de ellas. En cuanto a los owner, es decir, quién contiene a quién: perfil contiene a usuario, serie contiene a episodios, y desde perfil se generará el historial de visionados.

```
-- Tabla de usuarios (Clientes que usan la plataforma, cada usuario tiene
un perfil)
CREATE TABLE Usuario (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    email VARCHAR(100) UNIQUE
);
-- Tabla de perfiles (Cada usuario tiene un único perfil)
CREATE TABLE Perfil (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100),
    edad INT,
    usuario_id INT UNIQUE,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id)
);
-- Tabla de series (Contenido disponible en la plataforma)
CREATE TABLE Serie (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(200),
    genero INT,
    calificacion_edad INT
);
-- Tabla de episodios (Cada serie tiene varios episodios)
CREATE TABLE Episodio (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(200),
    duracion INT, -- Duración en minutos
    serie_id INT,
    FOREIGN KEY (serie_id) REFERENCES Serie(id)
);
```

```
-- Tabla de historial de reproducciones (Relación entre perfiles y
episodios vistos)
CREATE TABLE Historial (
    id INT AUTO_INCREMENT PRIMARY KEY,
    fecha_reproduccion DATETIME,
    perfil_id INT,
    episodio_id INT,
    FOREIGN KEY (perfil_id) REFERENCES Perfil(id),
    FOREIGN KEY (episodio_id) REFERENCES Episodio(id)
);
   -- Tabla Genero (de esta solo quiero que hagáis la @Entity)
   CREATE TABLE genero (
     id INT AUTO_INCREMENT,
     nombre VARCHAR(100)
   )
   -- Tabla Serie_Genero (solo @JoinTable)
   CREATE TABLE serie_genero (
     serie_id int,
     genero_id int,
     PRIMARY KEY (idSerie, idGenero),
     FOREIGN KEY (serie_id) REFERENCES Serie(id),
     FOREIGN KEY (genero_id) REFERENCES Genero(id),
   )
```

Creación del menú de aplicación

Implementa un menú que permita:

- Para cada entidad (usuario, perfil, serie, episodio, capítulo e historial) un CRUD:
  - Agregar una entidad.
  - Modificar datos de la entidad a través de su ID.
  - Eliminar a través de su ID.
  - Mostrar los datos a través de su ID.
- Realizar las siguientes consultas:
  - Mostrar todos los usuarios junto con la información de sus perfiles.
  - Mostrar todas las series.
  - Mostrar las series que existen en función de una edad introducida por el usuario.
  - Mostrar todos los capítulos de una serie a partir de un ID de serie.
  - Mostrar los capítulos que ha visto un usuario a partir de un ID: título de la serie, nombre del capítulo, duración, y fecha de reproducción.
  - Añadir al historial de un usuario a través de su ID, todos los de una serie a través de su ID de golpe (con la fecha de hoy). Si hubiera registrado algún capítulo de esa serie ya, solo se actualizaría la fecha del día (o se borraría y insertaría uno nuevo).
  - Mostrar series por género introducido por el usuario, con la duración media de sus capítulos.
  - Mostrar las 5 series más vistas del catálogo.