Instrucciones de Tixi para lo del registro:

SELECT nombre\_usuario,contraseña

FROM Usuario

WHERE nombre\_usuario="" AND contraseña=""

DATOS DEL USUARIO SE PRUEBA:

VALUES (

'usuario\_prueba',

'prueba@ejemplo.com',

'1234',

'2025-07-14',

'2000-01-01',

'No binario',

1,

1,

0

);

CODIGO PARA USAR EL USUARIO DE PRUEBA:

import sqlite3

#Conectar a la base

conn = sqlite3.connect("Database\_Bloom\_Music\_definitiva.db")

cursor = conn.cursor()

#Usuario de prueba

email = "prueba@ejemplo.com"

password = "1234"

#Verificar login

cursor.execute("""

SELECT id, nombre, email

FROM Usuario

WHERE email = ? AND contraseña = ?

""", (email, password))

user = cursor.fetchone()

if user:

print(f"Login exitoso: {user}")

else:

print("Usuario o contraseña incorrectos")

conn.close()

Login:

* A partir de un usuario y contraseña, verificar si el usuario y contraseña se encuentran en la base de datos.

SELECT id\_usuario, nombre\_usuario, email

FROM Usuario

WHERE email = ? AND contraseña = ?;

* Nuevamente, a partir de un usuario y contraseña, verificar si ambos atributos de usuario existen en la base de datos.

Registro:

* Al ingresar el email, se tiene que verificar que este no esté ya guardado en la base de datos.

SELECT id\_usuario

FROM Usuario

WHERE email = ?;

* En caso de que el email no esté en uso, debe poder guardar el email y contraseña en la base de datos, junto con los demás atributos del usuario (nombre, género, fecha de nacimiento).

INSERT INTO Usuario (nombre, email, contraseña, genero, fecha\_nacimiento)

VALUES (?, ?, ?, ?, ?);

Inicio:

* Lupa: Al buscar el nombre de la canción, álbum o artista, este se tiene que comprobar si está en la música subida dentro de la base de datos.

SELECT Cancion.id, Cancion.titulo, Artista.nombre AS artista, Album.nombre AS album

FROM Cancion

JOIN Artista ON Cancion.id\_artista = Artista.id

JOIN Album ON Cancion.id\_album = Album.id

WHERE Cancion.titulo LIKE '%' || ? || '%'

OR Artista.nombre LIKE '%' || ? || '%'

OR Album.nombre LIKE '%' || ? || '%';

* Cargar música: El usuario debe ser capaz de cargar su propia música descargada a la aplicación y esta debe quedar en la base de datos.

INSERT INTO Cancion (titulo, id\_artista, id\_album, genero, duracion, id\_usuario)

VALUES (?, ?, ?, ?, ?, ?);

* Historial de reproducción:

#Registrar que un usuario escuchó una canción

INSERT INTO Reproduccion (id\_usuario, id\_cancion, fecha)

VALUES (?, ?, DATE('now'));

#Mostrar historial de canciones escuchadas por un usuario

SELECT c.titulo, a.nombre AS artista, r.fecha

FROM Reproduccion r

JOIN Cancion c ON r.id\_cancion = c.id

JOIN Artista a ON c.id\_artista = a.id

WHERE r.id\_usuario = ?

ORDER BY r.fecha DESC;

* Canciones favoritas o likeadas:

#Marcar canción como favorita

INSERT INTO Favorito (id\_usuario, id\_cancion)

VALUES (?, ?);

#Obtener canciones favoritas de un usuario

SELECT c.titulo, a.nombre AS artista

FROM Favorito f

JOIN Cancion c ON f.id\_cancion = c.id

JOIN Artista a ON c.id\_artista = a.id

WHERE f.id\_usuario = ?;

* Actualizar datos del usuario:

UPDATE Usuario

SET contraseña = ?, nombre = ?, genero = ?

WHERE id = ?;

* Eliminar usuario o playlist:

#Eliminar playlist

DELETE FROM Playlist WHERE id = ? AND id\_usuario = ?;

#Eliminar usuario (y en cascada se eliminan sus playlists/favoritos si está configurado)

DELETE FROM Usuario WHERE id = ?;

* Playlist: Al usuario crear una lista de reproducción, esta debe quedar guardada en la base de datos junto con los atributos de las playlist (nombre, fecha de creación, orden).

Crear playlist

INSERT INTO Playlist (nombre, fecha\_creacion, id\_usuario)

VALUES (?, DATE('now'), ?);

Agregar canción a playlist

INSERT INTO Playlist\_Cancion (id\_playlist, id\_cancion, orden)

VALUES (?, ?, ?);

* Filtro: Debo poder filtrar las canciones según su género (atributo de canciones) y de ahí mostrar lo nuevo de los géneros más escuchados en la sección de NUEVO.

Filtrar canciones por género

SELECT id, titulo, genero

FROM Cancion

WHERE genero = ?;

Mostrar lo nuevo de los géneros más escuchados

SELECT c.id, c.titulo, c.genero

FROM Cancion c

JOIN (

SELECT genero, COUNT(\*) AS reproducciones

FROM Reproduccion

JOIN Cancion ON Reproduccion.id\_cancion = Cancion.id

WHERE fecha >= DATE('now','-30 days')

GROUP BY genero

ORDER BY reproducciones DESC

LIMIT 3

) top\_generos ON c.genero = top\_generos.genero

ORDER BY c.id DESC;

* Tendencia: En la sección de TENDENCIA dentro de la sección de NUEVO, se encontrarán aquellas 10 canciones más veces reproducidas en el último mes, en base a todos los usuarios.

Top 10 canciones más reproducidas en el último mes

SELECT c.id, c.titulo, COUNT(r.id) AS total\_reproducciones

FROM Reproduccion r

JOIN Cancion c ON r.id\_cancion = c.id

WHERE r.fecha >= DATE('now','-30 days')

GROUP BY c.id, c.titulo

ORDER BY total\_reproducciones DESC

LIMIT 10;

CÓDIGO PARA LOS FILTROS.

* Cumbia:

SELECT \* FROM Canciones WHERE Genero = 'Cumbia';

* Hiphop:

SELECT \* FROM Canciones WHERE Genero = 'HipHop';

* Jazz:

SELECT \* FROM Canciones WHERE Genero = 'Jazz';

* Latina:

SELECT \* FROM Canciones WHERE Genero = 'Latina';

* Pop:

SELECT \* FROM Canciones WHERE Genero = 'Pop';

* Rock:

SELECT \* FROM Canciones WHERE Genero = 'Rock';

* Salsa:

SELECT \* FROM Canciones WHERE Genero = 'Salsa';

* Alternativa/ mismo a tendencia que está arriba:

SELECT c.\*, COUNT(h.Id\_cancion) as Reproducciones

FROM Canciones c

JOIN Historial\_Reproduccion h ON c.Id\_canciones = h.Id\_cancion

GROUP BY c.Id\_canciones

ORDER BY Reproducciones DESC

LIMIT 10;