BASES DE DATOS

Curso: *SQL*

Tema: *Proyecto Final*

Alumno: *Roberto Agustín Mejía Collazos*

INDICE

Contenido

PROYECTO CODERHOUSE: BIBLIOTECA....3
1. Introducción....3
2. Objetivos....3
3. Situación problemática....3
4. Modelo de negocio....4
5. Diagrama de Entidad Relación....5
5.1. Boceto (Excalidraw)....5
5.2. Diagrama 2 (DrawSQL)....6
5.3. Diagrama 3 (Workbench 8.0)....7
6. Listado de tablas con descripción de estructuras....8
7. Scripts de creación de base de datos y tablas....8
7.1 CREACIÓN DE BASE DE DATOS....8
7.2 CREACIÓN DE TABLAS....9
7.3 Scripts de inserción de datos....10
8. Lista de Vistas....14
9. Lista de Funciones....14
10. Lista de Stored procedures....14
11. Lista Triggers....14
12. Scripts de Vistas....15
13. Script de Funciones....16
14. Scripts de Stored Procedures....17
15. Scripts de Triggers....18
16. Roles....19
17. Usuarios....19
18. Tablas, Views, Stored Procedures y funciones en Workbench....20
19. Herramientas y tecnologías usadas....20
20. Futuras líneas....21
21. Referencias Bibliográficas....21

PROYECTO CODERHOUSE: BIBLIOTECA

1. Introducción

El presente informe detalla el desarrollo de un proyecto de gestión de bibliotecas implementado en MySQL. El objetivo principal del sistema es facilitar la administración de los recursos bibliográficos, como libros, autores, géneros, editoriales, miembros y préstamos, mediante un diseño de base de datos relacional que optimiza la gestión y consulta de información. La base de datos está estructurada para soportar la compleja relación entre los diferentes elementos, garantizando así una gestión eficiente y coherente de la biblioteca.

2. Objetivos

Desarrollo de un sistema de gestión: Crear una base de datos que permita administrar la información relacionada con los libros, autores, géneros, editoriales, miembros y préstamos en una biblioteca de manera eficiente.

Optimización de consultas: Diseñar un modelo de base de datos que soporte consultas rápidas y precisas, permitiendo a los administradores recuperar y manipular información de manera efectiva.

Integridad de los datos: Implementar restricciones y claves foráneas para asegurar la integridad referencial de los datos y evitar inconsistencias.

Facilitar la escalabilidad: Estructurar la base de datos de manera que pueda ser escalable y adaptarse a futuras expansiones de la biblioteca.

3. Situación problemática

En muchas bibliotecas tradicionales, la gestión de los recursos bibliográficos y el control de préstamos se realizan manualmente o con sistemas obsoletos, lo que genera ineficiencias, pérdida de tiempo, y errores humanos. Estas ineficiencias pueden llevar a la pérdida de libros,

errores en el seguimiento de los préstamos, y dificultades en la recuperación de información. Ante esta problemática, surge la necesidad de implementar un sistema de gestión automatizado que permita llevar un control más preciso y eficiente de los recursos bibliográficos, mejorando así la calidad del servicio que se brinda a los usuarios.

4. Modelo de negocio

El modelo de negocio del sistema de gestión de bibliotecas se basa en ofrecer una herramienta digital robusta que permita a las bibliotecas optimizar sus operaciones diarias. El sistema permite:

Gestión de inventarios: Administrar la información relacionada con los libros, incluyendo su título, año de publicación, género y editorial.

Registro de autores y editoriales: Mantener una base de datos actualizada de autores y editoriales, lo que facilita la búsqueda y organización de los libros.

Administración de miembros: Gestionar la información de los miembros de la biblioteca, permitiendo un control preciso sobre el registro de nuevos miembros y el seguimiento de los actuales.

Control de préstamos: Facilitar la administración de préstamos y devoluciones, garantizando que los libros se manejen de manera adecuada y reduciendo el riesgo de pérdidas.

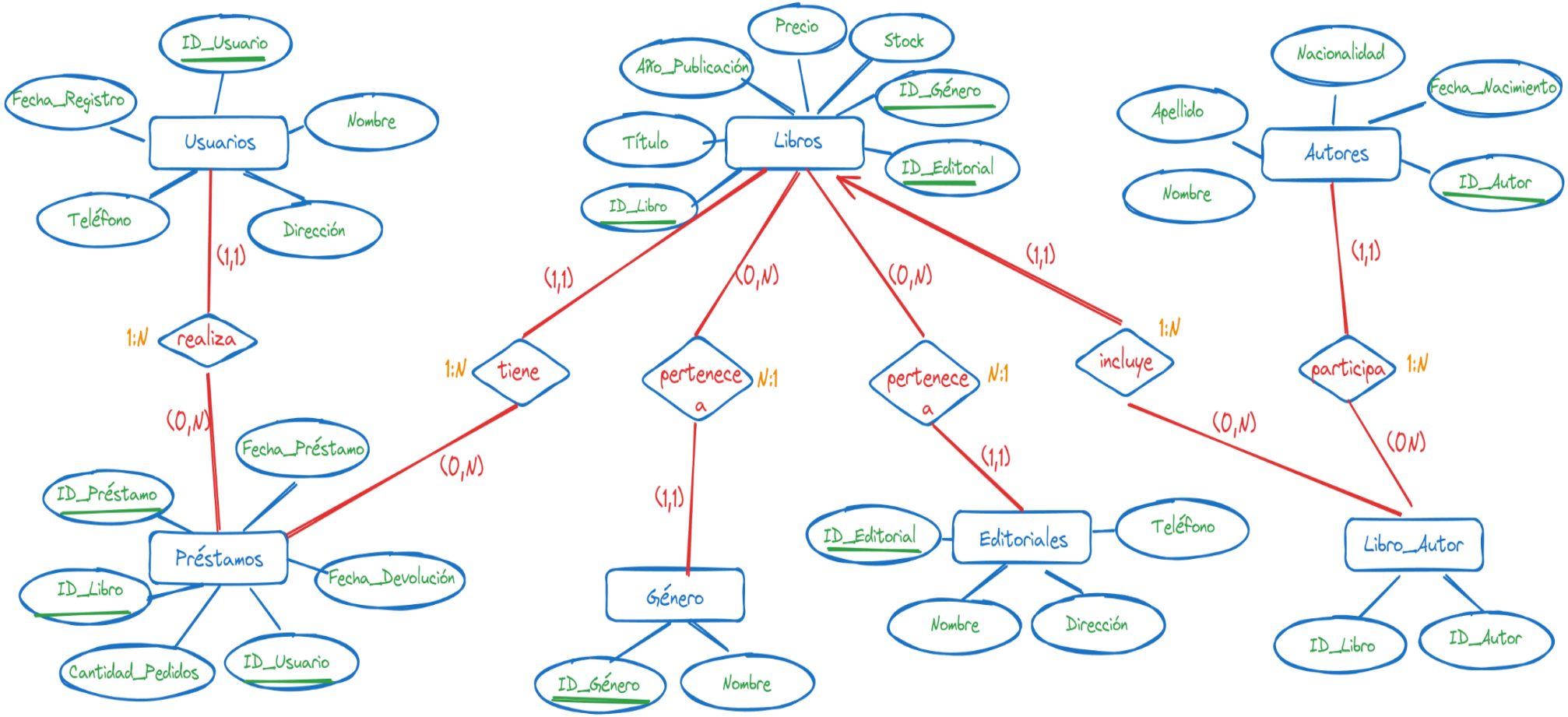
El sistema está diseñado para ser escalable, permitiendo la inclusión de nuevas funcionalidades conforme las necesidades de la biblioteca evolucionen. Este enfoque asegura que la inversión en la implementación del sistema sea sostenible a largo plazo.

ENLACE DEL REPOSITORIO:

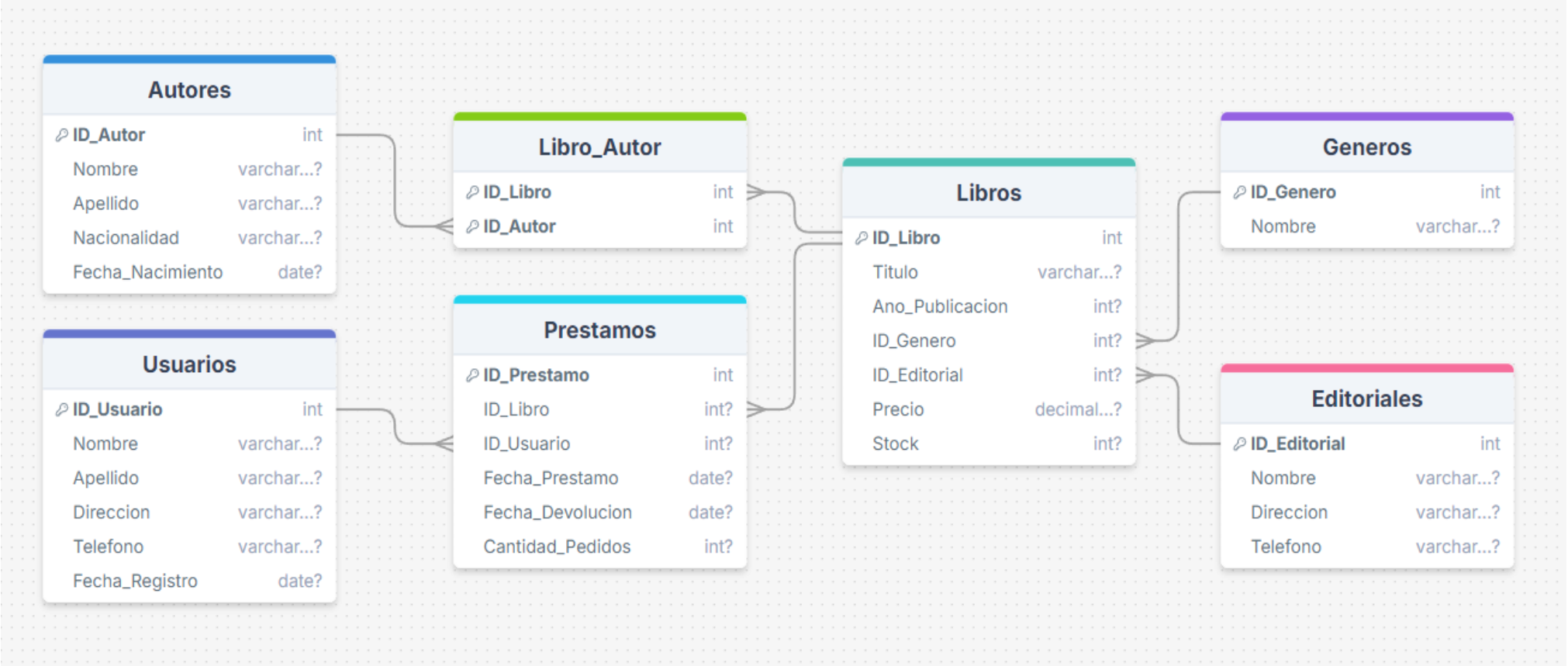
https://github.com/robermejia/coderhouse_sql_59410/tree/main

5. Diagrama de Entidad Relación

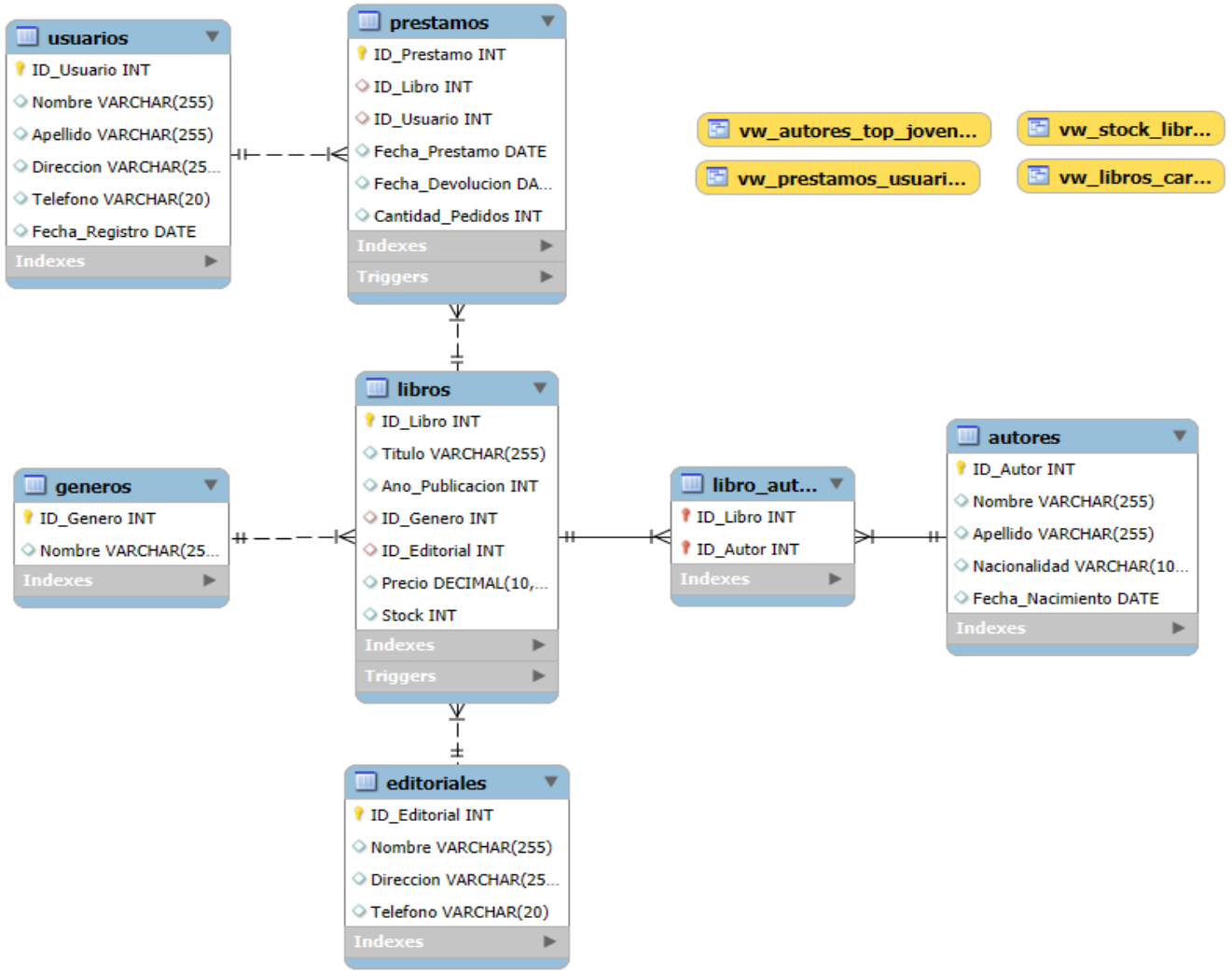
5.1. Boceto (Excalidraw)



5.2. Diagrama 2 (DrawSQL)



5.3. Diagrama 3 (Workbench 8.0)



6. Listado de tablas con descripción de estructuras

TABLA	CAMPOS	DETALLE CAMPO	PK	FK	TIPO DE DATO
Editoriales	ID_Editorial	*Identificador único de la editorial*	PK		INT
	Nombre	*Nombre de la editorial*			VARCHAR(255)
	Direccion	*Dirección de la editorial*			VARCHAR(255)
	Telefono	*Número de teléfono de la editorial*			VARCHAR(18)
Genero	ID_Genero	*Identificador único del género*	PK		INT
	Nombre	*Nombre del género*			VARCHAR(255)
Usuarios	ID_Usuario	*Identificador único del usuario*	PK		INT
	Nombre	*Nombre del usuario*			VARCHAR(255)
	Direccion	*Dirección del usuario*			VARCHAR(255)
	Telefono	*Número de teléfono del usuario*			VARCHAR(18)
	Fecha_Registro	*Fecha de registro del usuario*			DATE
Autores	ID_Autor	*Identificador único del autor*	PK		INT
	Nombre	*Nombre del autor*			VARCHAR(255)
	Apellido	*Apellido del autor*			VARCHAR(255)
	Nacionalidad	*Nacionalidad del autor*			VARCHAR(255)
	Fecha_Nacimiento	*Fecha de nacimiento del autor*			DATE
Libros	ID_Libro	*Identificador único del libro*	PK		INT
	ID_Genero	*ID del género al que pertenece el libro*		FK	INT
	ID_Editorial	*ID de la editorial del libro*		FK	INT
	Titulo	*Título del libro*			VARCHAR(255)
	Año_Publicacion	*Año de publicación del libro*			INT
	Precio	*Precio del libro*			INT
	Stock	*Cantidad de libros disponibles*			INT
Prestamos	ID_Prestamo	*Identificador único del préstamo*	PK		INT
	ID_Libro	*ID del libro asociado al préstamo*		FK	INT
	ID_Usuario	*ID del usuario asociado al préstamo*		FK	INT
	Fecha_Prestamo	*Fecha de realización del préstamo*			DATE
	Fecha_Devolucion	*Fecha prevista de devolución del préstamo*			DATE
	Cantidad_Pedidos	*Cantidad de pedidos*			INT
Libro_Autor	ID_Libro	*ID del libro asociado al autor*	FK		INT
	ID_Autor	*ID del autor asociado al libro*	FK		INT

7. Scripts de creación de base de datos y tablas

7.1 CREACIÓN DE BASE DE DATOS

```sql
-- ELIMINACIÓN DE BASE DE DATOS
DROP DATABASE IF EXISTS proyecto_biblioteca;
-- CREACIÓN DE BASE DE DATOS
CREATE DATABASE proyecto_biblioteca;
-- POSICIONARSE EN LA BD
USE proyecto_biblioteca;
```

7.2 CREACIÓN DE TABLAS

```sql
-- CREACIÓN DE TABLAS
CREATE TABLE IF NOT EXISTS Editoriales (
    ID_Editorial INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(255),
    Direccion VARCHAR(255),
    Telefono VARCHAR(20)
);
CREATE TABLE IF NOT EXISTS Generos (
    ID_Genero INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(255)
);
CREATE TABLE IF NOT EXISTS Usuarios (
    ID_Usuario INT AUTO_INCREMENT PRIMARY KEY,
    Nombre VARCHAR(255),
    Apellido VARCHAR(255),
    Direccion VARCHAR(255),
    Telefono VARCHAR(20),
    Fecha_Registro DATE
);
CREATE TABLE IF NOT EXISTS Autores (
    ID_Autor INT PRIMARY KEY,
    Nombre VARCHAR(255),
    Apellido VARCHAR(255),
    Nacionalidad VARCHAR(100),
    Fecha_Nacimiento DATE
);

CREATE TABLE IF NOT EXISTS Libros (
    ID_Libro INT PRIMARY KEY,
    Titulo VARCHAR(255),
    Ano_Publicacion INT,
    ID_Genero INT,
    ID_Editorial INT,
    Precio DECIMAL(10, 2),
    Stock INT,
    FOREIGN KEY (ID_Genero) REFERENCES Generos(ID_Genero),
    FOREIGN KEY (ID_Editorial) REFERENCES Editoriales(ID_Editorial)
```

```
);
CREATE TABLE IF NOT EXISTS Prestamos (
    ID_Prestamo INT AUTO_INCREMENT PRIMARY KEY,
    ID_Libro INT,
    ID_Usuario INT,
    Fecha_Prestamo DATE,
    Fecha_Devolucion DATE,
    Cantidad_Pedidos INT,
    FOREIGN KEY (ID_Libro) REFERENCES Libros(ID_Libro),
    FOREIGN KEY (ID_Usuario) REFERENCES Usuarios(ID_Usuario)
);
CREATE TABLE IF NOT EXISTS Libro_Autor (
    ID_Libro INT,
    ID_Autor INT,
    PRIMARY KEY (ID_Libro, ID_Autor),
    FOREIGN KEY (ID_Libro) REFERENCES Libros(ID_Libro),
    FOREIGN KEY (ID_Autor) REFERENCES Autores(ID_Autor)
);
-- VISUALIZAR TODAS LAS TABLAS DE LA BASE DE DATOS
SHOW TABLES;
```

7.3 Scripts de inserción de datos

```
-- INSERCIONES MULTIPLES
INSERT INTO Editoriales (Nombre, Direccion, Telefono) VALUES
('Editorial Planeta', 'Av. Javier Prado 123, Lima', '01-2345678'),
('Penguin Random House', 'Calle de las Letras 45, Madrid', '+34 910 123 456'),
('HarperCollins', '195 Broadway, New York', '+1 212-207-7000'),
('Simon & Schuster', '1230 Avenue of the Americas, New York', '+1 212-698-7000'),
('Alfaguara', 'Calle Alfonso XII, 62, Madrid', '+34 917 595 300'),
('Anagrama', 'Calle Pedró Martell, 19, Barcelona', '+34 933 687 850'),
('Random House Mondadori', 'Av. Diagonal, 662-664, Barcelona', '+34 934 928 840'),
('Editorial SM', 'Calle de Impresores, 2, Boadilla del Monte', '+34 917 596 400'),
('Santillana', 'Calle del Tambre, 50, Madrid', '+34 913 984 500'),
('Edebé', 'Calle de Provença, 386, Barcelona', '+34 934 535 500'),
('Minotauro', 'Calle de Espronceda, 32, Barcelona', '+34 933 663 600'),
('Roca Editorial', 'Calle de Roger de Llúria, 118, Barcelona', '+34 932 082 040');
```

```sql
INSERT INTO Generos (Nombre) VALUES
('Novela'),
('Ciencia Ficción'),
('Historia'),
('Fantasía'),
('Biografía'),
('Romance'),
('Misterio'),
('Terror'),
('Aventura'),
('Poesía'),
('Ensayo'),
('Drama');

INSERT INTO Usuarios (Nombre, Apellido, Direccion, Telefono, Fecha_Registro)
VALUES
('Carlos', 'López', 'Calle Falsa 123, Lima', '987654321', '2024-01-15'),
('María', 'Pérez', 'Av. Universitaria 678, Lima', '912345678', '2024-02-20'),
('José', 'Martínez', 'Av. Pardo y Aliaga 120, Lima', '987654322', '2024-03-10'),
('Lucía', 'Gómez', 'Jirón Miraflores 567, Lima', '987654323', '2024-04-05'),
('Andrés', 'Ramírez', 'Calle Las Lomas 876, Lima', '987654324', '2024-05-12'),
('Ana', 'Torres', 'Av. Brasil 450, Lima', '987654325', '2024-06-18'),
('Jorge', 'Rodríguez', 'Calle Los Sauces 34, Lima', '987654326', '2024-07-25'),
('Elena', 'Gutiérrez', 'Av. Benavides 223, Lima', '987654327', '2024-08-01'),
('Ricardo', 'Méndez', 'Calle Los Olivos 789, Lima', '987654328', '2024-08-15'),
('Isabel', 'Fernández', 'Jirón Lampa 234, Lima', '987654329', '2024-09-20'),
('Rosa', 'García', 'Av. San Felipe 453, Lima', '987654330', '2024-10-05'),
('Diego', 'Herrera', 'Calle Las Flores 123, Lima', '987654331', '2024-11-10');

INSERT INTO Autores (ID_Autor, Nombre, Apellido, Nacionalidad, Fecha_Nacimiento)
VALUES
(1, 'Gabriel', 'García Márquez', 'Colombiana', '1927-03-06'),
(2, 'Isaac', 'Asimov', 'Rusa-Americana', '1920-01-02'),
(3, 'Mario', 'Vargas Llosa', 'Peruana', '1936-03-28'),
(4, 'J.K.', 'Rowling', 'Británica', '1965-07-31'),
(5, 'George', 'Orwell', 'Británica', '1903-06-25'),
(6, 'Julio', 'Verne', 'Francesa', '1828-02-08'),
(7, 'Jane', 'Austen', 'Británica', '1775-12-16'),
```

```
(8, 'Ernest', 'Hemingway', 'Americana', '1899-07-21'),
(9, 'Agatha', 'Christie', 'Británica', '1890-09-15'),
(10, 'H.P.', 'Lovecraft', 'Americana', '1890-08-20'),
(11, 'Miguel de', 'Cervantes', 'Española', '1547-09-29'),
(12, 'Virginia', 'Woolf', 'Británica', '1882-01-25');

INSERT INTO Libros (ID_Libro, Titulo, Ano_Publicacion, ID_Genero, ID_Editorial,
   Precio, Stock) VALUES
(1, 'Cien Años de Soledad', 1967, 1, 1, 150.50, 10),
(2, 'Fundación', 1951, 2, 2, 120.75, 15),
(3, 'La Casa Verde', 1966, 1, 3, 95.25, 8),
(4, 'Harry Potter y la Piedra Filosofal', 1997, 4, 4, 180.30, 20),
(5, '1984', 1949, 6, 5, 88.90, 12),
(6, 'Viaje al Centro de la Tierra', 1864, 4, 6, 75.60, 5),
(7, 'Orgullo y Prejuicio', 1813, 5, 7, 68.40, 7),
(8, 'El Viejo y el Mar', 1952, 1, 8, 105.80, 18),
(9, 'Asesinato en el Orient Express', 1934, 7, 9, 110.20, 14),
(10, 'La Llamada de Cthulhu', 1928, 8, 10, 95.75, 9),
(11, 'Don Quijote de la Mancha', 1605, 1, 11, 65.90, 25),
(12, 'Al Faro', 1927, 12, 12, 82.45, 6);
```

```
INSERT INTO Prestamos (ID_Libro, ID_Usuario, Fecha_Prestamo, Fecha_Devolucion,
Cantidad_Pedidos) VALUES
(1, 1, '2024-08-10', '2024-08-20', 2),
(2, 2, '2024-08-15', '2024-08-25', 1),
(3, 3, '2024-07-01', '2024-07-15', 3),
(4, 4, '2024-07-20', '2024-08-05', 0),
(5, 5, '2024-06-25', '2024-07-05', 5),
(6, 6, '2024-05-15', '2024-05-25', 2),
(7, 7, '2024-04-10', '2024-04-20', 1),
(8, 8, '2024-03-18', '2024-03-28', 4),
(9, 9, '2024-02-20', '2024-03-01', 0),
(10, 10, '2024-01-12', '2024-01-22', 2),
(11, 11, '2024-12-05', '2024-12-15', 3),
(12, 12, '2024-11-01', '2024-11-11', 1);
```

```
INSERT INTO Libro_Autor (ID_Libro, ID_Autor) VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(5, 5),
(6, 6),
(7, 7),
(8, 8),
(9, 9),
(10, 10),
(11, 11),
(12, 12);
```

8. Lista de Vistas

VISTAS	DESCRIPCIÓN	OBJETIVO	TABLAS QUE LO COMPONEN
vw_prestamos_usuarios	*Muestra los usuarios que han tenido prestamos*	Cuidar sus datos	Usuarios
vw_libros_caros	*Muestra el top de libros más caros*	Ver top de precios	Libros
vw_autores_top_jovenes	*Muestra los autores más jovenes*	Ver autores nuevos	Autores
vw_stock_libros	*Muestra el stock de libros*	Ver stock de libros	Libros

9. Lista de Funciones

FUNCIONES	DESCRIPCIÓN	OBJETIVO	TABLAS QUE LO COMPONEN
fn_agregar_dias_prestamo	*Simula ampliación de fecha de devolución del prestamo del libro*	Simular ampliación de prestamo	Prestamos
fn_fecha_nacimiento_autor	*Muestra fecha de nacimiento de los autores segun el ID*	Ver fecha de nacimiento del autor	Autores

10. Lista de Stored procedures

STORED PROCEDURES	DESCRIPCIÓN	OBJETIVO	TABLAS QUE LO COMPONEN
sp_prestamo_stock	*Realiza prestamos e indica si ya hay libros disponibles*	Muestra el stock de libros	Libros
sp_tipo_lector	*Muestra en que categoria esta el usuario segun la compra*	Mostrar la categoría de comprador	Prestamos
sp_libro_azar	*Muestra 1 libro al azar*	Muestra 1 libro al azar	Libros

11. Lista Triggers

TRIGGERS	DESCRIPCIÓN	OBJETIVO	TABLAS QUE LO COMPONEN
tg_nuevo_libro_autor	*Agrega un libro a Libro_Autor cuando se inserta nuevo registro*	Muestra el stock de libros	Libros
tg_actualizar_stock_prestamos	*Agrega la fecha de devolución hasta la fecha actual*	Ampliar fecha de devolcuión	Prestamos, Libros

12. Scripts de Vistas

```
-- ########################## VISTAS ##########################
-- ========= VISTA 1 - PRESTAMOS DE LIBROS =========
DROP VIEW IF EXISTS vw_prestamos_usuarios; -- Eliminando la vista si es que existe
-- CREANDO LA VISTA "prestamos_usuarios_vw"
CREATE VIEW vw_prestamos_usuarios AS
SELECT usuarios.ID_Usuario, usuarios.Nombre,
MAX(prestamos.Fecha_Devolucion) AS Ultima_Fecha_Devolucion
FROM usuarios INNER JOIN prestamos ON usuarios.ID_Usuario = prestamos.ID_Usuario
GROUP BY usuarios.ID_Usuario;
```

```
-- ========= VISTA 2 - LIBROS MÁS CAROS =========
DROP VIEW IF EXISTS vw_libros_caros; -- Eliminando la vista si es que existe
CREATE VIEW vw_libros_caros AS
SELECT ID_Libro, Titulo, Precio
FROM libros ORDER BY Precio DESC LIMIT 5;
```

```
-- ========= VISTA 3 - AUTOR MÁS JOVEN =========
DROP VIEW IF EXISTS vw_autores_top_jovenes; -- Eliminando la vista si es que
existe
CREATE VIEW vw_autores_top_jovenes AS
SELECT ID_Autor, Nombre, Nacionalidad, Fecha_Nacimiento
FROM autores ORDER BY ABS(DATEDIFF(Fecha_Nacimiento, NOW())) LIMIT 5;
```

```
-- ========= VISTA 4 - STOCK DE LIBROS =========
DROP VIEW IF EXISTS vw_stock_libros; -- Eliminando la vista si es que existe
CREATE VIEW vw_stock_libros AS
SELECT COUNT(ID_Libro) AS STOCK_LIBROS FROM Libros;
```

13. Script de Funciones

```
-- ######################### FUNCIONES ##########################
-- FUNCIÓN SIMULACIÓN DE AMPLIACIÓN DE FECHA DE DEVOLUCIÓN
DROP FUNCTION IF EXISTS fn_agregar_dias_prestamo;
DELIMITER //
CREATE FUNCTION fn_agregar_dias_prestamo (prestamo_id INT, dias INT)
RETURNS VARCHAR(255)
READS SQL DATA
BEGIN
    DECLARE fecha_prestamo DATE;
    DECLARE fecha_limite DATE;
    SELECT Fecha_Devolucion INTO fecha_prestamo
    FROM prestamos
    WHERE ID_Prestamo = prestamo_id;
    SET fecha_limite = fecha_prestamo + INTERVAL dias DAY;
    RETURN CONCAT('Fecha de devolución: ', DATE_FORMAT(fecha_prestamo, '%Y-%m-%d'),
                ' - Nueva fecha límite: ', DATE_FORMAT(fecha_limite, '%Y-%m-%d'));
END;
//
DELIMITER ;
```

```
-- FUNCIÓN VER FECHA DE NACIMIENTO DE AUTORES POR SU ID
DROP FUNCTION IF EXISTS fn_fecha_nacimiento_autor;
DELIMITER //
CREATE FUNCTION fn_fecha_nacimiento_autor (autor_id INT)
RETURNS DATE
READS SQL DATA
BEGIN
    DECLARE fecha_autor DATE;
    SELECT Fecha_Nacimiento INTO fecha_autor
    FROM autores
    WHERE ID_Autor = autor_id;
    RETURN fecha_autor;
END;
//
DELIMITER ;
```

14. Scripts de Stored Procedures

```
-- ################## STORES PROCEDURES ######################
-- ======= PROCEDIMIENTO 1 - PRÉSTAMO DE STOCK =======
DROP PROCEDURE IF EXISTS sp_prestamo_stock;
DELIMITER //
CREATE PROCEDURE sp_prestamo_stock(IN ID_Miembro INT, IN ID_Libro INT, OUT cantidad INT)
BEGIN
    DECLARE cantidad_libros INT;
    SELECT Stock INTO cantidad_libros FROM libros WHERE libros.ID_Libro = ID_Libro;
    IF cantidad_libros > 0 THEN
        -- Actualizar el stock en la tabla libros
        UPDATE libros SET Stock = Stock - 1 WHERE libros.ID_Libro = ID_Libro;

        SET cantidad = cantidad_libros - 1;
        SELECT "Se ha realizado el préstamo exitosamente." AS Mensaje;
    ELSE
        SELECT "No es posible realizar préstamos." AS Mensaje;
    END IF;
END //
DELIMITER ;
```

```
-- ======= PROCEDIMIENTO 2 - CATEGORÍA DE USUARIOS =======
DROP PROCEDURE IF EXISTS sp_tipo_lector;
DELIMITER //
CREATE PROCEDURE sp_tipo_lector(IN ID_Usuario INT)
BEGIN
    SET @cantidad = (SELECT Cantidad_Pedidos FROM prestamos
                    WHERE prestamos.ID_Usuario = ID_Usuario);
    CASE
        WHEN @cantidad = 5 THEN
            SELECT "Fanático" AS Mensaje;
        WHEN @cantidad = 4 THEN
            SELECT "Aficionado" AS Mensaje;
        WHEN @cantidad = 3 THEN
            SELECT "Promedio" AS Mensaje;
        ELSE
            SELECT "Nuevo" AS Mensaje;
    END CASE;
END //
DELIMITER ;
```

```
-- ======= PROCEDIMIENTO 3 - LIBROS AL AZAR =======
DROP PROCEDURE IF EXISTS sp_libro_azar; -- Eliminando el procedimiento si es que
existe
DELIMITER //
CREATE PROCEDURE sp_libro_azar()
BEGIN
    SET @iterador = 0;
    WHILE @iterador < 3 DO
        SELECT ID_Libro, Titulo FROM libros ORDER BY RAND() LIMIT 1;
        SET @iterador = @iterador + 1;
    END WHILE;
END //
DELIMITER ;
```

15. Scripts de Triggers

```
-- ########################## TRIGGERS ##########################
-- ###################### TRIGGERS 1 ########################
DELIMITER //
CREATE TRIGGER tg_nuevo_libro_autor
AFTER INSERT ON Libros
FOR EACH ROW
BEGIN
-- Verificar si el libro ya existe en la tabla Libro_Autor
    IF NOT EXISTS (SELECT 1 FROM Libro_Autor WHERE ID_Libro = NEW.ID_Libro) THEN
    -- Insertar un nuevo registro en la tabla Libro_Autor
    INSERT INTO Libro_Autor (ID_Libro, ID_Autor) VALUES (NEW.ID_Libro, 1); -- Aquí
puedes cambiar el ID_Autor por el que consideres
    END IF;
END //
DELIMITER ;
```

```
-- ################################  TRIGGERS 2 ################################
DELIMITER //
CREATE TRIGGER tg_actualizar_stock_prestamo
AFTER INSERT ON Prestamos
FOR EACH ROW
BEGIN
-- Actualizar el stock del libro en la tabla Libros
    UPDATE Libros
    SET Stock = Stock - NEW.Cantidad_Pedidos
    WHERE ID_Libro = NEW.ID_Libro;
END //
DELIMITER ;
```

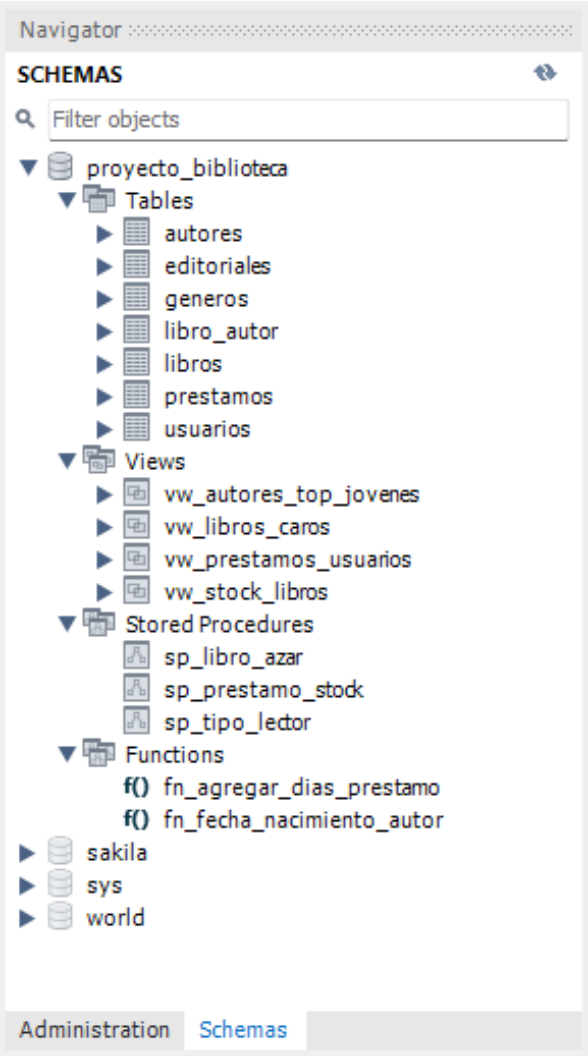
16. Roles

```
-- ######################## ROLES ##########################
-- Eliminar roles si existen
DROP ROLE IF EXISTS role_proyecto_biblioteca_admin;
DROP ROLE IF EXISTS role_proyecto_biblioteca_reader;

-- #### ROL 1 (ACCESO COMPLETO A BD "proyecto_biblioteca") ######################
CREATE ROLE role_proyecto_biblioteca_admin;
GRANT ALL PRIVILEGES ON proyecto_biblioteca.* TO role_proyecto_biblioteca_admin;

-- #### ROL 2 (SOLO LECTURA EN LA TABLA "libros" DE BD "proyecto_biblioteca") ###
CREATE ROLE role_proyecto_biblioteca_reader;
GRANT SELECT ON proyecto_biblioteca.libros TO role_proyecto_biblioteca_reader;
```

17. Usuarios

```
-- ################################### USERS ################################
SELECT Host, User FROM mysql.user; -- Muestra todos los usuarios;
SELECT USER(); -- Ver el usuario actual
FLUSH PRIVILEGES; -- ACTUALIZAR PRIVILEGIOS
-- ######################################## USERS
##############################################
DROP USER IF EXISTS 'coderhouse'@'%', 'coderhouse_alumno'@'%',
'coderhouse_docente'@'%', 'coderhouse_invitado'@'%';
-- ========= USER 1 (SIN PERMISOS) ===============
CREATE USER 'coderhouse'@'%'
IDENTIFIED BY 'coderhouse';
SHOW GRANTS FOR 'coderhouse'@'%'; -- Ver permisos
-- ========= USER 2 (CON DERECHOS RESTRINGIDOS) ===============
CREATE USER 'coderhouse_invitado'@'%' IDENTIFIED BY 'coderhouse';
GRANT role_proyecto_biblioteca_reader TO 'coderhouse_invitado'@'%';
SHOW GRANTS FOR 'coderhouse_invitado'@'%'; -- Ver permisos
-- ========= USER 3 (CON ACCESO A UNA BASE DE DATOS) ===============
CREATE USER 'coderhouse_alumno'@'%' IDENTIFIED BY 'coderhouse';
GRANT role_proyecto_biblioteca_admin TO 'coderhouse_alumno'@'%';
SHOW GRANTS FOR 'coderhouse_alumno'@'%'; -- Ver permisos
-- ========= USER 4 (CON TODOS LOS DERECHOS) ===============
CREATE USER 'coderhouse_docente'@'%' IDENTIFIED BY 'coderhouse';
GRANT role_proyecto_biblioteca_admin TO 'coderhouse_docente'@'%';
SHOW GRANTS FOR 'coderhouse_docente'@'%'; -- Ver permisos a todas las bd
```

18. Tablas, Views, Stored Procedures y funciones en Workbench



19. Herramientas y tecnologías usadas

- *MySQL Workbench*
- *MySQL 8.0 Command Line Client*
- *Visual Studio Code*
- *Windows PowerShell*
- *GitHub*

20. Futuras líneas

Podría considerar lo siguiente:

Mejoras en la escalabilidad: En el futuro, podría añadir funcionalidades para soportar bibliotecas más grandes o redes de bibliotecas. Esto incluiría la creación de tablas adicionales para manejar múltiples ubicaciones, o el uso de particionamiento de tablas para mejorar la eficiencia en bases de datos grandes.

Optimización de consultas: A medida que los datos crezcan, podría implementar índices adicionales o realizar optimizaciones más avanzadas en las consultas SQL para garantizar que las operaciones sigan siendo rápidas. También podrías considerar la implementación de bases de datos en la nube para aprovechar la elasticidad y los recursos escalables.

Seguridad y auditoría: Implementar mejores prácticas de seguridad para la base de datos, como cifrado de datos sensibles (por ejemplo, la información personal de los usuarios), controles de acceso avanzados, y auditoría de operaciones para monitorear quién accede y modifica la base de datos.

Integración con otras plataformas: Pienso integrar un sistema de gestión de bibliotecas con otros servicios o plataformas, como una API RESTful que permita la interacción con aplicaciones móviles, o el uso de sistemas de recomendaciones basados en algoritmos para mejorar la experiencia de los usuarios.

Mejora de la experiencia de usuario: Implementar funcionalidades avanzadas para los usuarios finales, como recomendaciones personalizadas de libros basadas en el historial de préstamos o herramientas de análisis de tendencias de uso.

21. Referencias Bibliográficas

- ***Ocaña, A. (2024, septiembre 24). CoderHouse SQL 59430. https://almondine-stealer-d91.notion.site/0fe6b4bc1c354ef99f0f88380a7e924a?v=ac85a3a9ef1c4616bcfff66e99ce10ea***

- ***Diagramaweb.com. (2020, September 29). Diagrama entidad-relación. https://diagramaweb.com/entidad-relacion***

- ***CódigoFacilito. (2022, enero 1). Curso Profesional de Base de Datos. https://codigofacilito.com/cursos/base-datos-profesional***