Universidad Fidélitas, Costa Rica

Lenguaje de Base de Datos

Sistema de Gestión para una Biblioteca

Profesor

Leitón Jiménez Randall Alonso

Estudiantes

Aguilar Aguilar José Daniel

Salas Cerdas Darling Dayana

II Avance

I cuatrimestre

18 de marzo de 2025

Objetivo General

Desarrollar un sistema de gestión para una biblioteca que ayude a automatizar el control de libros como préstamos, devoluciones y registros esenciales como tipo de usuarios, categorías y autores.

Objetivos Específicos

- Desarrollar una interfaz en Java que le permita a los usuarios poder reservar libros e interactuar con el sistema.
- Crear una base de datos por medio de Oracle 19c para almacenar de manera segura y eficiente la información de la biblioteca.
- Registrar reportes e información sobre libros, usuarios, autores y categorías.

Introducción

Hoy en día, la automatización de procesos se ha vuelto una exigencia esencial para la optimización del tiempo y la optimización en la administración de la información. En este escenario, la creación de un sistema de administración para una biblioteca constituye una respuesta eficaz para simplificar la gestión de libros, que abarca los procedimientos de préstamos, devoluciones y la estructuración de los registros de usuarios, autores y categorías.

El propósito principal de este proyecto es diseñar e instaurar un sistema que emplee Java para la interfaz de usuario y Oracle 19c como administrador de bases de datos. Java ha sido elegido por su compatibilidad con Oracle, su habilidad para escalar y su enfoque en objetos, lo que simplifica el mantenimiento del sistema. En Oracle 19c, la base de datos asegurará un almacenamiento seguro y eficaz de los datos de la biblioteca. La implementación de este sistema potenciará la experiencia

de los usuarios al ofrecer una plataforma fácil de usar para la reserva de libros y la administración de los registros de la biblioteca. Además, se producirán informes exhaustivos acerca de libros, usuarios y categorías, lo que favorecerá la toma de decisiones y el perfeccionamiento constante de la administración de bibliotecas.

Justificación de la Interfaz

Para este proyecto se tomó la decisión de trabajar con Java para el desarrollo de la interfaz esto con varios motivos, entre ellos, uno de los más importante sería lo compatible que es con Oracle 19c. No obstante, también se consideró este lenguaje de programación debido a la capacidad que tiene y lo escalable que puede llegar a ser.

La idea de utilizar Java se debe al conocimiento adquirido durante los cuatrimestres de carrera, pero sobre todo con la intención de llegar a profundizar en las herramientas de aprendizaje. Además, se sabe perfectamente que Java es un Lenguaje de Programación en el cual se pueden llegar a desarrollar diferentes sistemas con la posibilidad de ejecutarlos en diferentes plataformas.

Otro punto favorable, o en este caso las ventajas que se quieren destacar en esta decisión es el hecho de que Java está orientada a objetos, pero ¿qué quiere decir esto? Bueno, esto significa que permite estructurar el código de una manera organizada y eficiente donde los desarrolladores podrán darle mantenimiento al sistema en caso de ser necesario. Otro aspecto importante es que al momento de investigar se pueden encontrar grandes fuentes de información o documentación sobre este lenguaje que van desde revistas hasta videos lo cual facilitará el proceso de desarrollo cuando comiencen a surgir dudas o inquietudes referente al sistema de gestión que se desarrollará para una biblioteca.

A pesar de las ventajas mencionadas el grupo de este proyecto tiene claro que se podrían presentar algunos retos entre los cuales podría destacarse el conocimiento de los miembros. Posiblemente algunos van a estar más adelantados que otros o quizá entiendan de manera más sencilla Oracle 19c, mientras que los otros tendrán que dedicar más tiempo de estudio para poder alcanzar o nivelar esos conocimientos. No obstante, siendo este uno de los posibles retos a enfrentarse, cada uno de los integrantes tiene claro que esto es un equipo en el cual se van a apoyar y ayudar en todo lo que sea necesario no solo para avanzar con el desarrollo del sistema y la base de datos, sino también para cumplir con cada uno de los objetivos propuestos de manera rápida y eficiente de acuerdo con el cronograma propuesto.

Desarrollo

El desarrollo del Sistema de Gestión para una Biblioteca se realizará siguiendo un enfoque que permita la flexibilidad, la adaptación y sobre todo que logre ser eficiente y seguro siempre buscando alinearse a las necesidades del usuario, en este caso busca basarse y seguir por completo los requerimientos funcionales para poder cumplir con los objetivos establecidos. Gracias a este enfoque se considera que tanto el grupo de desarrolladores como los usuarios van a establecer una conexión que les permitirá desarrollar un sistema no solo funcional, sino que también eficiente.

En cada proyecto que se va a desarrollar existen diferentes etapas o fases que se deben seguir para poder cumplir totalmente con el objetivo, por lo tanto, en este también hay una serie de pasos que se deben seguir según el cronograma propuesto, entre ellos:

- **Planificar y Analizar Requisitos del Usuario**

El primer paso que un desarrollador debe seguir es preguntarle al cliente cuáles son las necesidades del proyecto o bien deducir con base en la conversación establecida los requerimientos funcionales que desea para su sistemas. Con base en esto los desarrolladores van a recopilar toda la información necesaria o en este caso clave para poder ejecutar el sistema de manera eficiente.

Tomando en cuenta lo anterior es importante destacar o recordar que para este sistema es necesario elaborar o desarrollar un registro y la gestión de libros, donde además de esto se puedan administrar usuarios como por ejemplo los profesores, los estudiantes y los funcionarios. También se busca generar dentro del mismo sistema la gestión de libros prestados y las respectivas devoluciones para tener una base de datos organizada y segura. Además la idea propuesta es crear un reporte sobre los libros disponibles, sus respectivos autores y las categorías que existen.

- **Diseñar un Modelo de Base de Datos**

Uno de los requisitos de este proyecto es generar un modelo de entidad relación por medio de un esquema, por lo tanto la base de datos se va a desarrollar por medio de Oracle 19c. Es importante mencionar que este modelo o este esquema incluirá una cierta cantidad de entidades principales como usuario, libro, autor, categoría, préstamo, reserva y reporte. Para cada una de estas tablas es completamente necesario establecer una llave primaria por medio de un atributo para poder representarla.

Para poder comprender de manera más sencilla estas entidades y sus respectivos atributos se puede decir que usuario es la tabla que va a contener información sobre los usuarios por ejemplo el nombre, el correo, número de teléfonos y si es administrador o usuario. Además se establece como primary key un id.

Por otra parte la tabla o entidad libro va a representar los libros disponibles en esa biblioteca, de igual manera contará con un identificador único, pero además tomará el id de otras entidades como foreign key, en este caso sería la relación que existe entre el libro y el autor y la relación entre el libro y la categoría. También tendrá un atributo llamado disponible como indicador de si se encuentra o no en la biblioteca.

Abarcando otras entidades se puede decir que autor va a almacenar los datos de los autores y como cada una de las otras tablas tendrá un identificador único (id_autor). En cuanto a la entidad categoría va a ser quien define la clasificación de cada uno de los libros según su género y su PK será (id_categoria).

Además de esas entidades existirá la entidad préstamos que se encargará de almacenar el registro de los libros que se le han prestado a los usuarios. Para ello se va a establecer un PK llamado "id_prestamo" y también tendrá dos llaves foráneas para hacer relación con el usuario y el libro. Adicionalmente se van a establecer dos atributos más para indicar la fecha en que se prestó y la fecha en que se devolvió.

Por otra parte la entidad reserva permitirá a los usuarios reservar libros antes de solicitar el préstamo, es decir; tendrá la oportunidad de hacer una reserva, para ello se creará un PK llamado "id_reserva" y se hará uso de dos llaves foráneas con el fin de establecer una relación entre el libro, el usuario y la reserva. Además se va a agregar otro atributo llamado "fecha_reserva" que como su nombre lo indica registrará la fecha en que se realizó la reserva.

Finalmente, la entidad reporte va a generar todos los informes sobre la biblioteca. Así como las demás contará con un identificador único que en este caso será "id_reporte", a la tabla se le agregarán también otros atributos como por ejemplo el tipo de reporte que en este caso podrían ser

el historial de préstamos, las categorías que más se han consultado y uno de los más importantes; los libros disponible. El otro atributo sería “fecha_generacion” que se utilizará con el fin de tener un registro sobre cuándo se generó ese reporte.

- **Crear la Base de Datos en Oracle 19c**

  Como se mencionaba anteriormente, la base de datos se va a crear por medio de Oracle 19c con la intención de poner en práctica las habilidades desarrolladas durante el curso lectivo. En esta base se van a crear tablas de acuerdo con las entidad principales establecidas en el punto anterior, además se realizarán relaciones de acuerdo con el modelo de entidad relación esto con el fin de poder asegurar un sistema eficiente. Para poder cumplir con este punto se deben crear o establecer llaves primaria (PK) y foráneas (FK). Además de implementar constrainsts y otras funciones importantes como crear procedimientos y consultas, entre otros.

- **Desarrollar el Sistema en Java**

  Para poder llevar a cabo este proyecto es importante hacer uso de otra herramienta, en el caso de este grupo, se optó por trabajar con Java creando el proyecto en NetBeans, donde se va a configurar o establecer una conexión con la base de datos creada en Oracle 19c. En esta misma plataforma se va a desarrollar la implementación de la gestión de libros, tanto de los registros como de las consultas de los libros disponibles y todo el registro de los libros prestados y devueltos. En resumen, se podría decir que Java es una herramienta clave en el desarrollo de este Sistema ya que permitirá realizar distintas funciones y almacenar la lógica también.

- **Realizar Pruebas**

Finalmente, con el objetivo de asegurarse de que todo esté funcionando correctamente, los integrantes del grupo deberán realizar pruebas, estas no deben ser complejas, basta con demostrar e intentar insertar usuarios, registrar libros o incluso consultar las categorías o autores, entre otras otras pruebas de este tipo que se puedan realizar. A este tipo de pruebas se le conocen como pruebas unitarias ya que el usuario o el desarrollador va probando una a una estas funciones. No obstante, además de estas existen las pruebas de integración, las cuales buscan probar que exista una comunicación segura y eficiente entre programas utilizados. En este caso, los miembros realizarán pruebas para asegurarse de que exista una conexión entre Java y la base de datos que van a desarrollar por medio de Oracle 19c, básicamente lo único que tienen que verificar es que cada uno de los datos se registren de manera correcta y que cuando se realice algún tipo de consulta se vea reflejada la información que claramente se está esperando.

Otro aspecto importante que se ha considerado como equipo es la idea de ir documentando todo el código que se escriba en NetBeans con la intención de dejar un registro en caso de que alguien más necesite acceder. Además se considera que esta es una buena práctica que debe realizarse siempre que sea posible.

Requerimientos Funcionales

1. ***Gestión de Usuarios*:**
    - Registro, modificación y eliminación de usuarios.
    - Asignación de roles (administrador, usuario regular).
    - Consulta de información de usuarios.

2. ***Gestión de Libros*:**
    - Registro, modificación y eliminación de libros.
    - Clasificación por categorías y autores.
    - Consulta de disponibilidad de libros.
3. ***Préstamos y Devoluciones*:**
    - Registro de préstamos de libros.
    - Control de fechas de devolución.
    - Notificaciones de vencimiento de préstamo.
4. ***Reservas*:**
    - Permitir a los usuarios realizar reservas de libros.
    - Consulta y cancelación de reservas.
    - Notificación cuando un libro reservado está disponible.
5. ***Gestión de Reportes*:**
    - Generación de reportes sobre libros disponibles y prestados.
    - Informes sobre usuarios y su historial de préstamos.
    - Reportes de categorías y autores más consultados.
6. ***Seguridad y Control de Acceso*:**
    - Inicio de sesión con autenticación.

- Control de acceso según el rol del usuario.
- Registro de actividad de los usuarios.

7. ***Interfaz de Usuario:***
    - Desarrollo en Java con una interfaz amigable e intuitiva.
    - Fácil navegación entre las opciones del sistema.
    - Compatibilidad con distintos dispositivos.

Requerimientos No Funcionales

1. ***Seguridad***

- Implementación de autenticación de usuarios con credenciales encriptadas.
- Control de acceso basado en roles (administrador, usuario regular).
- Registro y auditoría de actividades de los usuarios en el sistema.
- Protección contra inyecciones SQL y ataques de seguridad en la base de datos.
- Respaldo y recuperación de la base de datos para evitar pérdida de información.

2. ***Rendimiento***

- La base de datos debe ser capaz de gestionar al menos 100.000 registros sin afectar el rendimiento.
- El tiempo de respuesta para la consulta de disponibilidad de libros debe ser menor a 3 segundos.
- Los reportes deben generarse en menos de 5 segundos.
- Capacidad para manejar múltiples sesiones concurrentes sin afectar el rendimiento.

3. Usabilidad

- La interfaz debe ser intuitiva y fácil de usar tanto para administradores como para usuarios regulares.
- El sistema debe proporcionar mensajes de error claros y comprensibles.
- Diseño de interfaz amigable y responsive para facilitar el uso en diferentes dispositivos.

4. Disponibilidad

- El sistema debe estar disponible al menos el 99% del tiempo.
- Implementación de un sistema de respaldo y recuperación ante fallos.
- El sistema debe soportar mantenimiento programado sin afectar la disponibilidad de las funciones esenciales.

5. Escalabilidad

- El sistema debe poder escalarse para soportar un mayor número de usuarios sin necesidad de modificaciones estructurales drásticas.
- Capacidad de expansión para futuras funcionalidades como integración con catálogos externos o acceso remoto a los libros digitales.

6. Mantenimiento

- El código debe estar documentado para facilitar su mantenimiento y futuras mejoras.
- Uso de buenas prácticas de programación orientada a objetos en Java.
- La base de datos debe permitir actualizaciones sin afectar la operatividad del sistema.

7. Compatibilidad

- Compatible con sistemas operativos Windows, MacOS y Linux.
- Integración con Oracle 19c como gestor de base de datos.
- Soporte para diferentes navegadores en caso de futuras versiones web del sistema.

8. Confiabilidad

- El sistema debe garantizar la integridad de los datos, evitando duplicaciones o pérdida de información.
- Se deben realizar pruebas de estrés y carga para asegurar la estabilidad del sistema en condiciones de alto tráfico.

9. Legalidad

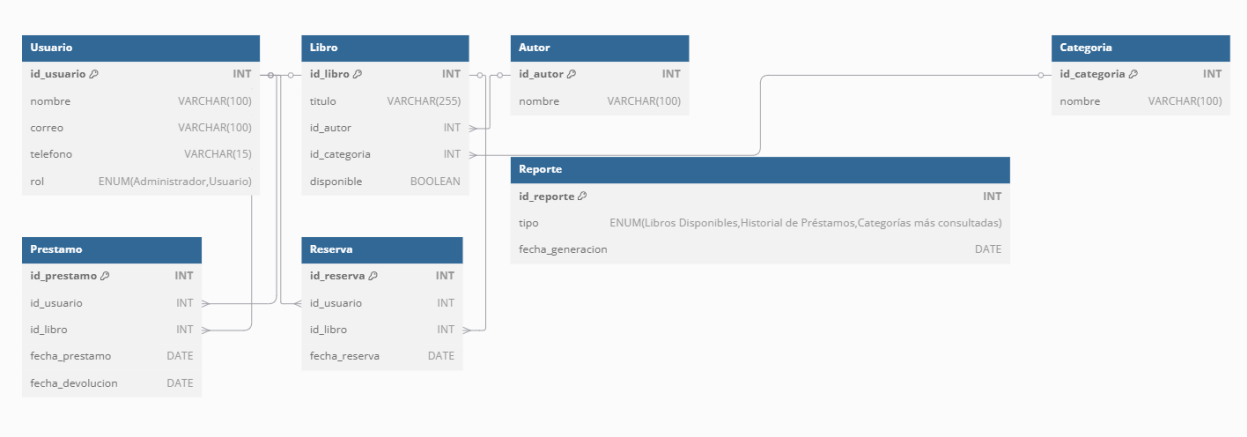
- Cumplir con normativas de protección de datos y privacidad aplicables en Costa Rica.
- Garantizar la correcta gestión de información personal de los usuarios conforme a las leyes de seguridad informática.

Cronograma

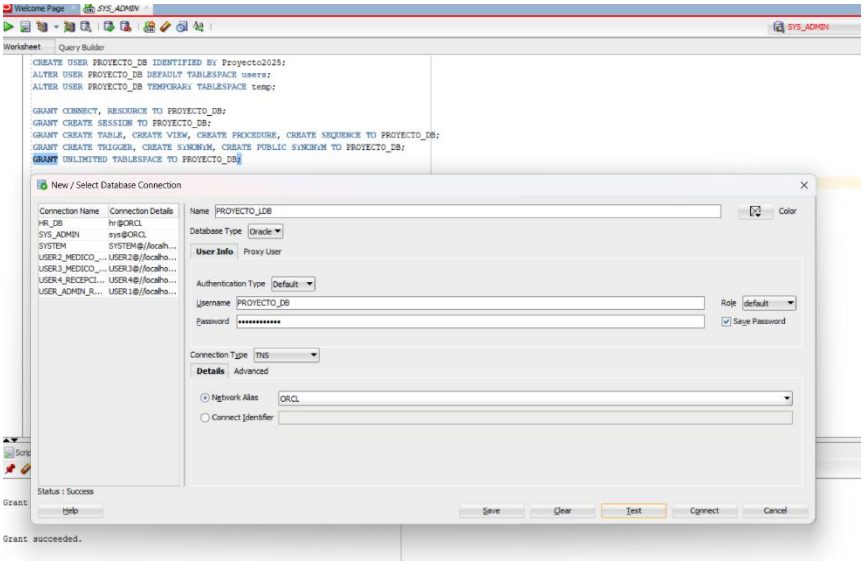
Actividad	**14 de enero 2025-15 de abril 2025**		
	28 enero-17 febrero Semana 3 a 6	**19 febrero-17 marzo Semana 6 a 10**	**19 marzo-14 abril Semana 10 a 14**
Marco conceptual			

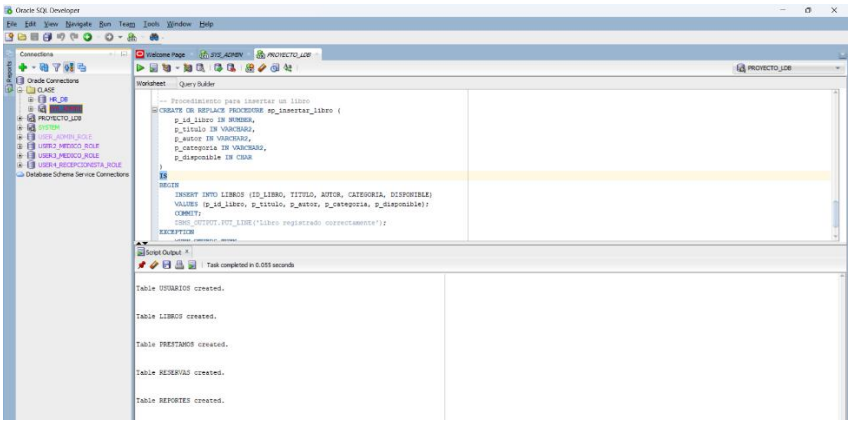
Requerimientos (funcionales no funcionales			
Objetivos generales y específicos			
Cronograma			
ER			
Requerimientos			
Propuesta de interfaz			
Definición objetos PL/SQL (diccionario de datos)			
Conexión a base de datos (código o capturas de pantalla)			
Avance de interfaz (capturas de pantalla)			
Definición objetos PL/SQL (diccionario de datos)			
Documento final			
Aplicación			

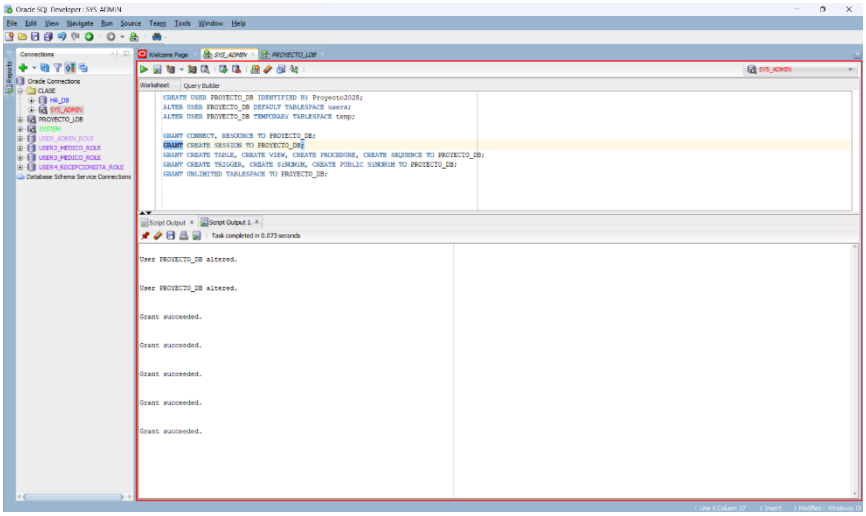
Esquema Entidad Relación:


Usuario
id_usuario INT
nombre VARCHAR(100)
correo VARCHAR(100)
telefono VARCHAR(15)
rol ENUM(Administrador,Usuario)
Libro
id_libro INT
titulo VARCHAR(255)
id_autor INT
id_categoria INT
disponible BOOLEAN
Autor
id_autor INT
nombre VARCHAR(100)
Categoria
id_categoria INT
nombre VARCHAR(100)
Reporte
id_reporte INT
tipo ENUM(Libros Disponibles,Historial de Préstamos,Categorías más consultadas)
fecha_generacion DATE
Prestamo
id_prestamo INT
id_usuario INT
id_libro INT
fecha_prestamo DATE
fecha_devolucion DATE
Reserva
id_reserva INT
id_usuario INT
id_libro INT
fecha_reserva DATE

Scripts y Esquema de la Base de Datos







Procedimientos Almacenados

Para los procedimientos almacenados, como en el segundo avance, la primera instrucción indicaba que no se podían realizar consultas directas como Select * From o inclusive Insert Into, se decidió hacer una investigación sobre un tema no visto en clase para poder implementar los procedimientos con un código más profundo. En este caso, gracias a lo aprendido en otros cursos y a herramientas de Internet o sitios web como “IBM” se pudo comprender que una manera eficiente, clara y sencilla de hacer consultas cumpliendo esta regla eran los cursores. En la biografía se puede ver la página que se estudió para poder realizar los procedimientos almacenados con base en los requisitos de este entregable. También se vieron video con el fin de comprender mejor el tema, el mismo está ubicado al final del entregable. Adicionalmente, el conocimiento se complementó con lo visto en la clase de semana 9.

```
---------------------------------
 PROCEDIMIENTOS ALMACENADOS
---------------------------------

--1. AGREGAR USUARIO

CREATE OR REPLACE PROCEDURE SP_AGREGAR_USUARIO(
    SP_NOMBRE IN VARCHAR2,
    SP_CORREO IN VARCHAR2,
    SP_TELEFONO IN VARCHAR2,
    SP_ROL IN VARCHAR2
)
AS
    CONTADOR NUMBER := 1;
```

```
   VSUM NUMBER := 0;
BEGIN
   INSERT INTO USUARIOS(NOMBRE, CORREO, TELEFONO, ROL)
   VALUES(SP_NOMBRE, SP_CORREO, SP_TELEFONO, SP_ROL);

   DBMS_OUTPUT.PUT_LINE('Usuario agregado: ' || SP_NOMBRE);

EXCEPTION
   WHEN OTHERS THEN
      DBMS_OUTPUT.PUT_LINE('Error: No se pudo agregar el usuario. Por favor, intente de nuevo');
END;

SET SERVEROUTPUT ON;

EXEC SP_AGREGAR_USUARIO('Carlos López', 'carlos.lopez@email.com', '5552345678', 'USUARIO');

--2. ELIMINAR USUARIO POR ID

CREATE OR REPLACE PROCEDURE SP_ELIMINAR_USUARIO(
   SP_ID_USUARIO IN NUMBER
)
AS
   VCOUNT NUMBER;
BEGIN
   SELECT COUNT(*) INTO VCOUNT
   FROM USUARIOS
   WHERE ID_USUARIO = SP_ID_USUARIO;
```

```
    IF VCOUNT > 0 THEN
        DELETE FROM USUARIOS
        WHERE ID_USUARIO = SP_ID_USUARIO;
      DBMS_OUTPUT.PUT_LINE('Usuario eliminado correctamente: ' || SP_ID_USUARIO);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: Usuario no encontrado.');
    END IF;
END;

EXEC SP_ELIMINAR_USUARIO(1);



--3. ACTUALIZAR DATOS USUARIO

CREATE OR REPLACE PROCEDURE SP_ACTUALIZAR_USUARIO(
    SP_ID IN NUMBER,
    SP_NOMBRE IN VARCHAR2,
    SP_CORREO IN VARCHAR2,
    SP_TELEFONO IN VARCHAR2,
    SP_ROL IN VARCHAR2
)
AS
    VCOUNT NUMBER;
BEGIN
    SELECT COUNT(*) INTO VCOUNT
    FROM USUARIOS
    WHERE ID_USUARIO = SP_ID;
```

```
    IF VCOUNT > 0 THEN
        UPDATE USUARIOS
        SET NOMBRE = SP_NOMBRE,
            CORREO = SP_CORREO,
            TELEFONO = SP_TELEFONO,
            ROL = SP_ROL
        WHERE ID_USUARIO = SP_ID;

        DBMS_OUTPUT.PUT_LINE('Datos del usuario actualizados correctamente: ' || SP_ID);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: Usuario no encontrado.');
    END IF;
END;

BEGIN
    SP_ACTUALIZAR_USUARIO(10, 'Sofía', 'sofia.fernandez@email.com', '6677889999',
'USUARIO');
END;

-- 4. AGREGAR NUEVO LIBRO
CREATE OR REPLACE PROCEDURE SP_AGREGAR_LIBRO(
    SP_ID IN NUMBER,
    SP_TITULO IN VARCHAR2,
    SP_AUTOR IN VARCHAR2,
    SP_CATEGORIA IN VARCHAR2,
    SP_DISPONIBLE IN CHAR
)
AS
```

```
    VCOUNT NUMBER;
BEGIN
    SELECT COUNT(*) INTO VCOUNT
    FROM LIBROS
    WHERE ID_LIBRO = SP_ID;

    IF VCOUNT = 0 THEN
    INSERT INTO LIBROS(ID_LIBRO, TITULO, AUTOR, CATEGORIA, DISPONIBLE)
        VALUES(SP_ID, SP_TITULO, SP_AUTOR, SP_CATEGORIA, SP_DISPONIBLE);

        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Libro agregado correctamente: ' || SP_TITULO);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: El libro con ID ' || SP_ID || ' ya existe.');
    END IF;
END;

EXEC SP_AGREGAR_LIBRO(1, 'El Gran Gatsby', 'F. Scott Fitzgerald', 'Ficción', 'S');

EXEC SP_AGREGAR_LIBRO(125, 'El conde de Montecristo', 'Alexandre Dumas', 'Aventura',
'S');

--5. ACTUALIZAR LIBRO

CREATE OR REPLACE PROCEDURE SP_ACTUALIZAR_LIBRO(
    SP_ID IN NUMBER,
    SP_TITULO IN VARCHAR2,
    SP_AUTOR IN VARCHAR2,
    SP_CATEGORIA IN VARCHAR2,
```

```
    SP_DISPONIBLE IN CHAR
)
AS
    VCOUNT NUMBER;
BEGIN
    SELECT COUNT(*) INTO VCOUNT
    FROM LIBROS
    WHERE ID_LIBRO = SP_ID;

    IF VCOUNT > 0 THEN
        UPDATE LIBROS
        SET TITULO = SP_TITULO,
            AUTOR = SP_AUTOR,
            CATEGORIA = SP_CATEGORIA,
            DISPONIBLE = SP_DISPONIBLE
        WHERE ID_LIBRO = SP_ID;

        COMMIT;
        DBMS_OUTPUT.PUT_LINE('Libro actualizado correctamente: ' || SP_TITULO);
    ELSE
        DBMS_OUTPUT.PUT_LINE('Error: El libro con ID ' || SP_ID || ' no existe.');
    END IF;
END;

EXEC SP_ACTUALIZAR_LIBRO(122, 'Alicia en el país de las maravillas', 'Lewis Carroll',
'Fantasía', 'N');

--cuando no existe el libro
```

```
EXEC SP_ACTUALIZAR_LIBRO(127, 'Alicia a través del espejo', 'Lewis Carroll', 'Fantasía',
'N');



-- 6. ELIMINAR LIBRO POR ID


CREATE OR REPLACE PROCEDURE SP_ELIMINAR_LIBRO(
   SP_ID_LIBRO IN NUMBER
)
AS
   VCOUNT NUMBER;
BEGIN
   SELECT COUNT(*) INTO VCOUNT
   FROM LIBROS
   WHERE ID_LIBRO = SP_ID_LIBRO;

   IF VCOUNT > 0 THEN
      DELETE FROM LIBROS
      WHERE ID_LIBRO = SP_ID_LIBRO;
    DBMS_OUTPUT.PUT_LINE('El libro ha sido eliminado correctamente: ' || SP_ID_LIBRO);
   ELSE
      DBMS_OUTPUT.PUT_LINE('Error: Libro no encontrado.');
   END IF;
END;


EXEC SP_ELIMINAR_LIBRO(115);


EXEC SP_ELIMINAR_LIBRO(190); -- No existe
```

```
--7. FECHA DE DEVOLUCION ACTUALIZADA

CREATE OR REPLACE PROCEDURE SP_ACTUALIZA_FECHA_DEVOLUCION(
   SP_ID_PRESTAMO IN NUMBER,
   SP_FECHA_ACTUALIZADA IN DATE
)
AS
BEGIN
   UPDATE PRESTAMOS
   SET FECHA_DEVOLUCION = SP_FECHA_ACTUALIZADA
   WHERE ID_PRESTAMO = SP_ID_PRESTAMO;
END;

--- Probamos:
INSERT INTO PRESTAMOS (ID_PRESTAMO, FECHA_DEVOLUCION)
VALUES (190, TO_DATE('2025-04-01', 'YYYY-MM-DD'));

EXEC SP_ACTUALIZA_FECHA_DEVOLUCION(190, TO_DATE('2025-05-01', 'YYYY-
MM-DD'));



--8. RESERVAR LIBRO

CREATE OR REPLACE PROCEDURE SP_RESERVAR_LIBRO(
   SP_ID_RESERVA IN NUMBER,
   SP_ID_USUARIO IN NUMBER,
   SP_ID_LIBRO IN NUMBER,
   SP_FECHA_RESERVACION IN DATE
```

```
)
AS
   V_COUNT NUMBER;
BEGIN
   SELECT COUNT(*) INTO V_COUNT
   FROM RESERVAS
   WHERE ID_USUARIO = SP_ID_USUARIO
     AND ID_LIBRO = SP_ID_LIBRO
     AND FECHA_RESERVA = SP_FECHA_RESERVACION;

   IF V_COUNT > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Lo siento. El libro ya se encuentra reservado.');
   ELSE
      INSERT INTO RESERVAS (ID_RESERVA, ID_USUARIO, ID_LIBRO,
FECHA_RESERVA)
      VALUES (SP_ID_RESERVA, SP_ID_USUARIO, SP_ID_LIBRO,
SP_FECHA_RESERVACION);

      COMMIT;
      DBMS_OUTPUT.PUT_LINE('Reserva realizada con éxito: ' || SP_ID_RESERVA);
   END IF;
END;

INSERT INTO LIBROS (ID_LIBRO, TITULO, AUTOR, CATEGORIA, DISPONIBLE)
VALUES (203, 'Sobreviviendo a las Sombras', 'Darling Salas', 'Poemario', 'N');

INTO USUARIOS (ID_USUARIO, NOMBRE, APELLIDO, CORREO, TELEFONO, ROL)
   VALUES (12, 'Miguel', 'Azofeifa', 'miguel.azofeifa@email.com', '9876543210', 'USUARIO')
```

```
EXEC SP_RESERVAR_LIBRO(1, 12, 203, TO_DATE('2025-03-20', 'YYYY-MM-DD'));


-- 9. PRESTAMO LIBRO


CREATE OR REPLACE PROCEDURE SP_PRESTAMO(
   SP_ID_PRESTAMO IN NUMBER,
   SP_ID_USUARIO IN NUMBER,
   SP_ID_LIBRO IN NUMBER,
   SP_FECHA_PRESTAMO IN DATE,
   SP_FECHA_DEVOLUCION IN DATE
)
AS
   RESERVADO NUMBER;
BEGIN
   SELECT COUNT(*) INTO RESERVADO
   FROM RESERVAS
   WHERE ID_LIBRO = SP_ID_LIBRO
     AND FECHA_RESERVA = SP_FECHA_PRESTAMO;


   IF RESERVADO > 0 THEN
      DBMS_OUTPUT.PUT_LINE('Lo siento. El libro no está disponible para préstamo porque está reservado.');
      RETURN;
   END IF;


   INSERT INTO PRESTAMOS (ID_PRESTAMO, ID_USUARIO, ID_LIBRO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
   VALUES (SP_ID_PRESTAMO, SP_ID_USUARIO, SP_ID_LIBRO, SP_FECHA_PRESTAMO, SP_FECHA_DEVOLUCION);
```

```
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Préstamo realizado con éxito: ' || SP_ID_PRESTAMO);
END;



EXEC SP_PRESTAMO(1, 12, 203, TO_DATE('2025-03-20', 'YYYY-MM-DD'),
TO_DATE('2025-04-20', 'YYYY-MM-DD'));

--10. ELIMINAR PRESTAMO LIBRO

CREATE OR REPLACE PROCEDURE SP_ELIMINAR_PRESTAMO(SP_ID IN NUMBER)
AS
BEGIN
    DELETE FROM PRESTAMOS
    WHERE ID_PRESTAMO = SP_ID;
END;

--hacemos la prueba:
INSERT INTO LIBROS (ID_LIBRO, TITULO, AUTOR)
VALUES (178, 'Lágrimas Sigilosas', 'Darling');

INSERT INTO PRESTAMOS (ID_PRESTAMO, ID_LIBRO, FECHA_PRESTAMO)
VALUES (1005, 178, TO_DATE('2025-03-18', 'YYYY-MM-DD'));

EXEC SP_ELIMINAR_PRESTAMO(1005);



--11. PRESTAMOS ACTIVOS DE UN SUUARIO
```

```
CREATE OR REPLACE PROCEDURE SP_PRESTAMOS_ACTIVOS (SP_ID_USUARIO IN
NUMBER)
AS
   CURSOR C_PRESTAMOS IS
      SELECT ID_PRESTAMO
      FROM PRESTAMOS
      WHERE ID_USUARIO = SP_ID_USUARIO
      AND FECHA_DEVOLUCION IS NULL;
BEGIN
   FOR PRESTAMO IN C_PRESTAMOS LOOP
      DBMS_OUTPUT.PUT_LINE('El prestamo: ' || prestamo.ID_PRESTAMO || ' se encuentra
activo');
   END LOOP;
END;
```

---probamos:

```
INSERT INTO PRESTAMOS (ID_PRESTAMO, ID_USUARIO, ID_LIBRO,
FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (1008, 14, 178, TO_DATE('2025-03-18', 'YYYY-MM-DD'), NULL);

INSERT INTO PRESTAMOS (ID_PRESTAMO, ID_USUARIO, ID_LIBRO,
FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (1009, 14, 178, TO_DATE('2025-03-18', 'YYYY-MM-DD'), NULL);

EXEC SP_PRESTAMOS_ACTIVOS(14);
```

```
--12. VERIFICAR DISPONIBILIDAD

CREATE OR REPLACE PROCEDURE SP_DISPONIBILIDAD (
   SP_ID_LIBRO IN NUMBER,
   SP_DISPONIBLE OUT CHAR
)
AS
   CURSOR C_LIBRO IS
      SELECT DISPONIBLE
      FROM LIBROS
      WHERE ID_LIBRO = SP_ID_LIBRO;
   VDIS CHAR(1);
BEGIN
   OPEN C_LIBRO;
   FETCH C_LIBRO
   INTO VDIS;

   IF C_LIBRO%FOUND THEN
      SP_DISPONIBLE := VDIS;
   ELSE
      SP_DISPONIBLE := 'N';
   END IF;

   CLOSE C_LIBRO;
END;

--PRUEBA:
```

```
DECLARE
   VDISP  CHAR(1);
BEGIN
   SP_DISPONIBILIDAD(SP_ID_LIBRO => 122, SP_DISPONIBLE => VDISP);
   DBMS_OUTPUT.PUT_LINE('La disponibilidad del libro es: ' || VDISP);
END;

/*
La disponibilidad del libro es: N
*/

--13. NOMBRE DE USUARIO POR ID

CREATE OR REPLACE PROCEDURE SP_NOMBRE_USUARIO(
   SP_ID_USUARIO IN NUMBER,
   SP_NOMBRE OUT VARCHAR2
)
AS
   CURSOR C_USUARIO IS
      SELECT NOMBRE
      FROM USUARIOS
      WHERE ID_USUARIO = SP_ID_USUARIO;
   VNOM VARCHAR2(50);
BEGIN
   FOR USUARIO IN C_USUARIO LOOP
      SP_NOMBRE := USUARIO.NOMBRE;
   END LOOP;
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
        SP_NOMBRE := NULL;
END;

DECLARE
    V_NOMBRE VARCHAR2(50);
BEGIN
     SP_NOMBRE_USUARIO(SP_ID_USUARIO => 14, SP_NOMBRE => V_NOMBRE);



    DBMS_OUTPUT.PUT_LINE('El nombre del usuario es: ' || V_NOMBRE);
END;

/*
El nombre del usuario es: Andrés
*/

--14. ELIMINAR RESERVA POR ID

CREATE OR REPLACE PROCEDURE SP_ELIMINAR_RESERVA (SP_ID IN NUMBER)
AS
    VCOUNT NUMBER;
BEGIN
    SELECT COUNT(*) INTO VCOUNT
    FROM RESERVAS
    WHERE ID_RESERVA = SP_ID;

    IF VCOUNT > 0 THEN
```

```
        DELETE FROM RESERVAS
        WHERE ID_RESERVA = SP_ID;

        COMMIT;

        DBMS_OUTPUT.PUT_LINE('Reserva con ID ' || SP_ID || ' eliminada exitosamente.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('La reserva con ID ' || SP_ID || ' no existe.');
    END IF;
END;

--- PRUEBA:

    INSERT INTO RESERVAS (ID_RESERVA, ID_USUARIO, ID_LIBRO,
FECHA_RESERVA)
    VALUES (2005, 13, 110, TO_DATE('2024-03-10', 'YYYY-MM-DD'));

EXEC SP_ELIMINAR_RESERVA(2005);



/*
Reserva con ID 2005 eliminada exitosamente.
*/

-- 15. LIBROS DISPONIBLES POR CATEGORIA

CREATE OR REPLACE PROCEDURE SP_LIBRO_DIS_CATE(SP_CATE IN VARCHAR2)
AS
BEGIN
```

```
    FOR LIBRO IN (SELECT TITULO FROM LIBROS
        WHERE CATEGORIA = SP_CATE
        AND DISPONIBLE = 'S')
    LOOP
        DBMS_OUTPUT.PUT_LINE('Libro disponible: ' || LIBRO.TITULO);
    END LOOP;
END;

EXEC SP_LIBRO_DIS_CATE('Novela');

/*
Libro disponible: Cien años de soledad
Libro disponible: El último recuerdo que dejó tu muerte
*/

--16. TOTAL LIBROS PRESTADOS

CREATE OR REPLACE PROCEDURE SP_LIBROS_PRESTADOS(SP_TOTAL OUT
NUMBER)
AS
    CURSOR C_PRESTAMOS IS
        SELECT COUNT(*) AS TOTAL
        FROM PRESTAMOS
        WHERE FECHA_DEVOLUCION IS NULL;
BEGIN
    OPEN C_PRESTAMOS;
        FETCH C_PRESTAMOS INTO SP_TOTAL;
    CLOSE C_PRESTAMOS;
END;
```

```
--PROBAMOS
DECLARE
  VTOTAL NUMBER;
BEGIN
  SP_LIBROS_PRESTADOS(VTOTAL);
  DBMS_OUTPUT.PUT_LINE('Total de libros prestados: ' || V_TOTAL);
END;

/*
Total de libros prestados: 2
*/

--17. ELIMINAR REPORTE POR ID

CREATE OR REPLACE PROCEDURE ELIMINAR_REPORTE(SP_ID IN NUMBER)
AS
   VCOUNT NUMBER;
BEGIN

   SELECT COUNT(*) INTO VCOUNT FROM REPORTES WHERE ID_REPORTE = SP_ID;

   IF VCOUNT > 0 THEN
      DELETE FROM REPORTES WHERE ID_REPORTE = SP_ID;
      DBMS_OUTPUT.PUT_LINE('Reporte con ID ' || SP_ID || ' eliminado correctamente.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('No se encontró reporte con ID ' || SP_ID || '.');
   END IF;
```

```
END;

--- PRUEBA

INSERT INTO REPORTES (ID_REPORTE, TIPO_REPORTE, FECHA_GENERACION)
VALUES (3004, 'Reporte de Préstamos', TO_DATE('2024-03-18', 'YYYY-MM-DD'));

EXEC ELIMINAR_REPORTE(3004);
/*
Reporte con ID 3004 eliminado correctamente.
*/

EXEC ELIMINAR_REPORTE(3005);

/*
No se encontró reporte con ID 3005.
*/


--18. PRESTAMOS POR LIBRO

CREATE OR REPLACE PROCEDURE SP_PRESTAMOS_POR_LIBRO(SP_ID_LIBRO IN
NUMBER)
AS
   CURSOR C_PRESTAMOS IS
      SELECT ID_PRESTAMO, ID_USUARIO, FECHA_PRESTAMO,
FECHA_DEVOLUCION
      FROM PRESTAMOS
      WHERE ID_LIBRO = SP_ID_LIBRO
```

```
    AND FECHA_DEVOLUCION IS NULL;
BEGIN
  FOR PRESTAMO IN C_PRESTAMOS LOOP
    DBMS_OUTPUT.PUT_LINE('Préstamo ID: ' || PRESTAMO.ID_PRESTAMO || ' - Usuario ID: ' || PRESTAMO.ID_USUARIO || ' - Fecha de préstamo: ' || TO_CHAR(PRESTAMO.FECHA_PRESTAMO, 'DD-MM-YYYY'));
  END LOOP;
END;

--- PRUEBA:

INSERT INTO PRESTAMOS (ID_PRESTAMO, ID_USUARIO, ID_LIBRO, FECHA_PRESTAMO, FECHA_DEVOLUCION)
VALUES (1010, 10, 112, TO_DATE('2024-03-01', 'YYYY-MM-DD'), NULL);

EXEC SP_PRESTAMOS_POR_LIBRO(112);

/*
Préstamo ID: 1010 - Usuario ID: 10 - Fecha de préstamo: 01-03-2024
*/

--19. USUARIOS POR ROL

CREATE OR REPLACE PROCEDURE SP_USER_POR_ROL(SP_ROL IN VARCHAR2)
AS
BEGIN
  FOR USUARIO IN (SELECT * FROM USUARIOS WHERE ROL = SP_ROL)
  LOOP
    DBMS_OUTPUT.PUT_LINE('Usuario: ' || USUARIO.NOMBRE);
```

```
    END LOOP;
END;

EXEC SP_USER_POR_ROL('USUARIO');

/*
Usuario: Luisa
Usuario: Pedro
Usuario: Laura
Usuario: Sofía
Usuario: Ricardo
Usuario: Sofía
Usuario: Donatien
Usuario: Miguel
Usuario: Martha
Usuario: Andrés
Usuario: Sara
Usuario: Katthy
Usuario: Carlos
Usuario: Mónica
*/

-- 20. TODOS LOS LIBROS

CREATE OR REPLACE PROCEDURE SP_LISTA_LIBROS
AS
    VCOUNT NUMBER := 1;
    VTOTAL NUMBER;
```

```
BEGIN
   SELECT COUNT(*) INTO VTOTAL FROM LIBROS;

   DBMS_OUTPUT.PUT_LINE('Total de libros disponibles: ' || VTOTAL || ' ');
   FOR LIBRO IN(SELECT * FROM LIBROS) LOOP
      DBMS_OUTPUT.PUT_LINE('Libro ' || VCOUNT || ': ' || LIBRO.TITULO || ' - ' ||
LIBRO.AUTOR);
      VCOUNT := VCOUNT + 1;
   END LOOP;
END;



EXEC SP_LISTA_LIBROS;

/*

Total de libros disponibles: 28

Libro 1: La sombra del viento - Carlos Ruiz Zafón
Libro 2: 1984 - George Orwell
Libro 3: Fahrenheit 451 - Ray Bradbury
Libro 4: El nombre de la rosa - Umberto Eco
Libro 5: La tregua - Mario Benedetti
Libro 6: Crónica de una muerte anunciada - Gabriel García Márquez
Libro 7: Rayuela - Julio Cortázar
Libro 8: Cumbres Borrascosas - Emily Brontë
Libro 9: El Gran Gatsby - F. Scott Fitzgerald
Libro 10: Orgullo y prejuicio - Jane Austen
Libro 11: Sobreviviendo a las Sombras - Darling Salas
```

Libro 12: Los hermanos Karamazov - Fiódor Dostoyevski

Libro 13: Ulises - James Joyce

Libro 14: El retrato de Dorian Gray - Oscar Wilde

Libro 15: El alquimista - Paulo Coelho

Libro 16: Los miserables - Victor Hugo

Libro 17: El viento en los sauces - Kenneth Grahame

Libro 18: Alicia en el país de las maravillas - Lewis Carroll

Libro 19: La metamorfosis - Franz Kafka

Libro 20: Moby Dick - Herman Melville

Libro 21: El conde de Montecristo - Alexandre Dumas

Libro 22: El Gran Gatsby - F. Scott Fitzgerald

Libro 23: Título del libro - Autor del libro

Libro 24: Lágrimas Sigilosas - Darling

Libro 25: Cien años de soledad - Gabriel García Márquez

Libro 26: El principito - Antoine de Saint-Exupéry

Libro 27: Don Quijote de la Mancha - Miguel de Cervantes

Libro 28: El último recuerdo que dejó tu muerte - Darling Salas

*/

Funciones

A continuación se muestran las funciones implementadas en el proyecto: ----------

FUNCIONES

--

```
------------------
FUNCIONES
------------------

--1. INSERTAR USUARIO

CREATE OR REPLACE FUNCTION FN_INSERTAR_USUARIO(
   SP_NOMBRE IN VARCHAR2,
   SP_CORREO IN VARCHAR2,
   SP_TELEFONO IN VARCHAR2,
   SP_ROL IN VARCHAR2
)
RETURN VARCHAR2
AS
   VALERTA VARCHAR2(80);
BEGIN
   INSERT INTO USUARIOS (NOMBRE, CORREO, TELEFONO, ROL)
   VALUES (SP_NOMBRE, SP_CORREO, SP_TELEFONO, SP_ROL);

   VALERTA := 'El usuario se ha agregado exitosamente: ' || SP_NOMBRE;

   RETURN VALERTA;
END;


--2. NOMBRE DE USUARIO POR ID

CREATE OR REPLACE FUNCTION FN_NOM_USUARIO(SP_ID_USUARIO IN NUMBER)
```

```
RETURN VARCHAR2
IS
   VNOM VARCHAR2(50);
BEGIN
   SP_NOMBRE_USUARIO(SP_ID_USUARIO, VNOM);
   IF VNOM IS NULL THEN
      RETURN 'Usuario no encontrado';
   ELSE
      RETURN VNOM;
   END IF;
END;

---
DECLARE
   VRES VARCHAR2(50);
BEGIN
   VRES := FN_NOM_USUARIO(1);
   DBMS_OUTPUT.PUT_LINE(VRES);
END;

/*
Usuario no encontrado
*/


--3. ACTUALIZAR DATOS DEL USUARIO

CREATE OR REPLACE FUNCTION FN_ACTUALIZAR_USUARIO (
```

```
    VID IN NUMBER,
    VNOM IN VARCHAR2,
    VCORREO IN VARCHAR2,
    VTEL IN VARCHAR2,
    VROL IN VARCHAR2
)
RETURN VARCHAR2
AS
BEGIN
    UPDATE USUARIOS
    SET NOMBRE = VNOM, CORREO = VCORREO, TELEFONO = VTEL, ROL = VROL
    WHERE ID_USUARIO = VID;

    RETURN 'El usuario ha sido actualizado exitosamente';
END;

--PRUEBA

DECLARE
    VRES VARCHAR2(100);
BEGIN
    VRES := FN_ACTUALIZAR_USUARIO(1, 'Juan Pérez Actualizado',
'juan.actualizado@correo.com', '555-9999', 'ADMIN');
    DBMS_OUTPUT.PUT_LINE(VRES);
END;

/*
El usuario ha sido actualizado exitosamente
*/
```

```
--4. USUARIOS POR ROL

CREATE OR REPLACE FUNCTION FN_USUARIOS_POR_ROL(SP_ROL IN VARCHAR2)
RETURN VARCHAR2
IS
   VRES VARCHAR2(255);
BEGIN
   FOR USUARIO IN (SELECT NOMBRE FROM USUARIOS WHERE ROL = SP_ROL)
   LOOP
      VRES := VRES || USUARIO.NOMBRE || ', ';
   END LOOP;

   IF VRES IS NULL THEN
      RETURN 'Lo sentimos. No hay usuarios con este rol';
   ELSE
      RETURN VRES;
   END IF;
END;

-- PRUEBA

DECLARE
   VRES VARCHAR2(255);
BEGIN
   VRES := FN_USUARIOS_POR_ROL('USUARIO');
   DBMS_OUTPUT.PUT_LINE(VRES);
```

```
END;

/*
Luisa, Pedro, Laura, Sofía, Ricardo, Sofía, Donatien, Miguel, Martha, Andrés, Sara, Katthy, Carlos, Mónica,
*/

--5. TOTAL DE RESERVAS POR USUARIO

CREATE OR REPLACE FUNCTION FN_TOTAL_RESERVAS_USUARIO(SP_ID_USUARIO IN NUMBER)
RETURN NUMBER
IS
   VCONT NUMBER := 0;
BEGIN
   FOR RESERVA IN (SELECT * FROM RESERVAS WHERE ID_USUARIO = SP_ID_USUARIO)
   LOOP
      VCONT := VCONT + 1;
   END LOOP;

   IF VCONT = 0 THEN
      RETURN 0;
   ELSE
      RETURN VCONT;
   END IF;
END;

---- PRUEBA
```

```
DECLARE
   VRESERVAS NUMBER;
BEGIN
   VRESERVAS := FN_TOTAL_RESERVAS_USUARIO(10);
   DBMS_OUTPUT.PUT_LINE('Total de reservas: ' || VRESERVAS);
END;

/*Total de reservas: 0*/

DECLARE
   VRESERVAS NUMBER;
BEGIN
   VRESERVAS := FN_TOTAL_RESERVAS_USUARIO(12);
   DBMS_OUTPUT.PUT_LINE('Total de reservas: ' || VRESERVAS);
END;

/*Total de reservas: 2*/

--6. NOMBRE DE LIBRO POR ID

CREATE OR REPLACE FUNCTION FN_NOM_LIBRO(SP_ID_LIBRO IN NUMBER)
RETURN VARCHAR2
IS
   VNOM_LIBRO VARCHAR2(100);
BEGIN
   FOR LIBRO IN (SELECT TITULO FROM LIBROS WHERE ID_LIBRO = SP_ID_LIBRO)
   LOOP
```

```
      VNOM_LIBRO := LIBRO.TITULO;
   END LOOP;

    IF VNOM_LIBRO IS NULL THEN
      RETURN 'Lo siento. El libro no fue encontrado';
   ELSE
      RETURN VNOM_LIBRO;
   END IF;
END;

--- PRUEBA

DECLARE
   VLIBRO VARCHAR2(100);
   VID_LIBRO NUMBER;
BEGIN
   VID_LIBRO := 1;
   VLIBRO := FN_NOM_LIBRO(VID_LIBRO);
   DBMS_OUTPUT.PUT_LINE('El título del libro es: ' || VLIBRO);
END;



/*El título del libro es: El Gran Gatsby*/

--7. LIBROS PRESTADOS POR CATEGORIA

CREATE OR REPLACE FUNCTION FN_PRESTADOS_CATEG(SP_CATEGORIA IN
VARCHAR2)
RETURN NUMBER
```

```
IS
   VCONT NUMBER := 0;
BEGIN
   FOR PRESTAMO IN (SELECT * FROM PRESTAMOS WHERE ID_LIBRO IN (SELECT
ID_LIBRO FROM LIBROS WHERE CATEGORIA = SP_CATEGORIA) AND
FECHA_DEVOLUCION IS NULL)
   LOOP
      VCONT := VCONT + 1;
   END LOOP;

   RETURN VCONT;
END;

SET SERVEROUTPUT ON;



--- PRUEBA:

DECLARE
   VRES NUMBER;
BEGIN
   VRES := FN_PRESTADOS_CATEG('Novela');
   DBMS_OUTPUT.PUT_LINE(VRES);
END;



/* 0 */

--8. PRESTAMOS ACTIVOS DEL USUARIO
```

```
CREATE OR REPLACE FUNCTION FN_PREST_ACTIVOS(SP_ID_USUARIO IN
NUMBER)
RETURN CHAR
IS
   VCONT NUMBER;
BEGIN
   FOR PRESTAMO IN (SELECT *
                    FROM PRESTAMOS
                    WHERE ID_USUARIO = SP_ID_USUARIO
                    AND FECHA_DEVOLUCION IS NULL)
   LOOP
      VCONT := VCONT + 1;
   END LOOP;

   IF VCONT > 0 THEN
      RETURN 'El usuario tiene ' || VCONT || 'préstamos activos';
   ELSE
      RETURN 'El usuario no tiene ningún préstamo activo';
   END IF;
END;

-- PRUEBA:

DECLARE
   VRES VARCHAR2(100);
BEGIN
   VRES := FN_PREST_ACTIVOS(1);
   DBMS_OUTPUT.PUT_LINE(VRES);
```

```
END;

/*El usuario no tiene ningún préstamo activo*/

--9. LIBROS DISPONIBLES POR CATEGORIA

CREATE OR REPLACE FUNCTION FN_CAT_DISP(SP_CATEGORIA IN VARCHAR2)
RETURN VARCHAR2
IS
   VRES VARCHAR2(250);
BEGIN
   FOR LIBRO IN (SELECT TITULO FROM LIBROS WHERE CATEGORIA =
SP_CATEGORIA AND DISPONIBLE = 'S')
   LOOP
      VRES := VRES || LIBRO.TITULO || ', ';
   END LOOP;
   IF VRES IS NULL THEN
      RETURN 'No hay libros disponibles en esta categoría';
   ELSE
      RETURN VRES;
   END IF;
END;

--PRUEBA:
SELECT FN_CAT_DISP('Novela') FROM DUAL;

/*Cien años de soledad, El último recuerdo que dejó tu muerte*/

-- 10. TOTAL DE TODOS LOS LIBROS PRESTADOS
```

```
CREATE OR REPLACE FUNCTION FN_LIBROS_PRESTADOS
RETURN NUMBER
IS
   VTOTAL NUMBER;
BEGIN
   SP_LIBROS_PRESTADOS(VTOTAL);

   IF VTOTAL <= 0 THEN
      RETURN 'No hay libros prestados';
   ELSE
      RETURN VTOTAL;
   END IF;
END;



-- PRUEBA

DECLARE
   VRES NUMBER;
BEGIN
   VRES := FN_LIBROS_PRESTADOS;

   DBMS_OUTPUT.PUT_LINE('Cantidad de libros prestados: ' || VRES);
END;

/* Cantidad de libros prestados: 3 */
```

```
-- 11. PRESTAMOS VENCIDAS


CREATE OR REPLACE FUNCTION FN_PRESTAMOS_VENCIDOS(SP_ID_USUARIO IN
NUMBER)
RETURN NUMBER
IS
   VCONT NUMBER := 0;
BEGIN
   FOR PRESTAMO IN (SELECT *
                    FROM PRESTAMOS
                    WHERE ID_USUARIO = SP_ID_USUARIO
                    AND FECHA_DEVOLUCION < SYSDATE
                    AND FECHA_DEVOLUCION IS NOT NULL)
   LOOP
      VCONT := VCONT + 1;
   END LOOP;


   RETURN VCONT;
END;


-- PRUEBA


DECLARE
   VRES NUMBER;
BEGIN
   VRES := FN_PRESTAMOS_VENCIDOS(12);
   DBMS_OUTPUT.PUT_LINE('Número de préstamos vencidos: ' || VRES);
END;
```

/*Número de préstamos vencidos: 0*/

--12. LIBRO POR TITULO

```
CREATE OR REPLACE FUNCTION FN_LIBRO_POR_TITULO(SP_TITULO IN
VARCHAR2)
RETURN VARCHAR2
IS
   VTITULO VARCHAR2(100);
BEGIN
   FOR LIBRO IN (SELECT TITULO FROM LIBROS WHERE TITULO = SP_TITULO)
   LOOP
      VTITULO := LIBRO.TITULO;
   END LOOP;

   IF VTITULO IS NULL THEN
      RETURN 'Lo siento. El libro no fue encontrado';
   ELSE
      RETURN VTITULO;
   END IF;
END;

-- PRUEBA:
DECLARE
   VRES VARCHAR2(100);
BEGIN
   VRES := FN_LIBRO_POR_TITULO('Marchitos');
   DBMS_OUTPUT.PUT_LINE(' ' || VRES);
```

```
END;

/*Lo siento. El libro no fue encontrado*/

--13. LIBROS MAS PRESTADOS

CREATE OR REPLACE FUNCTION FN_LIBROS_MAS_PRESTADOS
RETURN VARCHAR2
IS
   VRES VARCHAR2(255);
BEGIN
   FOR LIBRO IN (SELECT TITULO, COUNT(*) AS NUM_PRESTAMOS
                 FROM PRESTAMOS
                 JOIN LIBROS ON PRESTAMOS.ID_LIBRO = LIBROS.ID_LIBRO
                 WHERE PRESTAMOS.ID_LIBRO IS NOT NULL
                 GROUP BY TITULO
                 ORDER BY NUM_PRESTAMOS DESC)
   LOOP
      VRES := VRES || LIBRO.TITULO || ' - ' || LIBRO.NUM_PRESTAMOS || ' préstamos, ';
   END LOOP;

   IF VRES = '' THEN
      RETURN 'No hay libros prestados';
   ELSE
      RETURN VRES;
   END IF;
END;
```

---PRUEBA

```
DECLARE
   VRES VARCHAR2(255);
BEGIN
   VRES := FN_LIBROS_MAS_PRESTADOS;
   DBMS_OUTPUT.PUT_LINE('Libros más prestados: ' || VRES);
END;
```

/*

Libros más prestados: Lágrimas Sigilosas - 2 préstamos, Cien años de soledad - 1 préstamos, Cumbres Borrascosas - 1 préstamos, El principito - 1 préstamos, Don Quijote de la Mancha - 1 préstamos

*/

--14. TIPO DE REPORTE POR ID

```
CREATE OR REPLACE FUNCTION FN_BUSCAR_TIPO_REPORTE(SP_ID_REPORTE IN NUMBER)
RETURN VARCHAR2
IS
   VREPORTE VARCHAR2(100);
BEGIN
   SELECT TIPO_REPORTE
   INTO VREPORTE
   FROM REPORTES
   WHERE ID_REPORTE = SP_ID_REPORTE;
```

```
    IF VREPORTE IS NULL THEN
        RETURN 'Reporte no encontrado';
    ELSE
        RETURN VREPORTE;
    END IF;
END;

--15. OBTENER NOMBRE COMPLETO DEL USUAIRO
CREATE OR REPLACE FUNCTION ObtenerNombreUsuario(FN_ID_USUARIO IN
NUMBER)
RETURN VARCHAR2 AS
    VNOM VARCHAR2(200);
BEGIN
    SELECT NOMBRE || ' ' || APELLIDO INTO VNOM
    FROM USUARIOS
    WHERE ID_USUARIO = FN_ID_USUARIO;

    RETURN VNOM;
END;
```

Diccionario de Datos

Diccionario de Datos

Tabla: USUARIOS

Campo	Tipo de Dato	Restricciones
ID_USUARIO	NUMBER PRIMARY	KEY
NOMBRE	VARCHAR2(100) NOT	NULL
CORREO	VARCHAR2(100) UNIQUE	NOT NULL
TELEFONO	VARCHAR2(15)	
ROL	VARCHAR2(20) CHECK	(ROL IN ('ADMIN', 'USUARIO'))

Tabla: LIBROS

Campo	Tipo de Dato	Restricciones
ID_LIBRO	NUMBER PRIMARY	KEY
TITULO	VARCHAR2(100) NOT	NULL
AUTOR	VARCHAR2(100) NOT	NULL
CATEGORIA	VARCHAR2(50)	
DISPONIBLE	CHAR(1) DEFAULT	'S' CHECK (DISPONIBLE IN ('S', 'N'))

Tabla: PRESTAMOS

Campo	Tipo de Dato	Restricciones
ID_PRESTAMO	NUMBER PRIMARY	KEY
ID_USUARIO	NUMBER	
ID_LIBRO	NUMBER	
FECHA_PRESTAMO	DATE	
FECHA_DEVOLUCION	DATE	
CONSTRAINT	FK_PRESTAMO_USUARIO FOREIGN	KEY (ID_USUARIO) REFERENCES USUARIOS(ID_USUARIO)
CONSTRAINT	FK_PRESTAMO_LIBRO FOREIGN	KEY (ID_LIBRO) REFERENCES LIBROS(ID_LIBRO)

Tabla: RESERVAS

Campo	Tipo de Dato	Restricciones
ID_RESERVA	NUMBER PRIMARY	KEY
ID_USUARIO	NUMBER	
ID_LIBRO	NUMBER	
FECHA_RESERVA	DATE	
CONSTRAINT	FK_RESERVA_USUARIO FOREIGN	KEY (ID_USUARIO) REFERENCES USUARIOS(ID_USUARIO)
CONSTRAINT	FK_RESERVA_LIBRO FOREIGN	KEY (ID_LIBRO) REFERENCES LIBROS(ID_LIBRO)

Tabla: REPORTES

Campo	Tipo de Dato	Restricciones
ID_REPORTE	NUMBER PRIMARY	KEY
TIPO_REPORTE	VARCHAR2(50)	
FECHA_GENERACION	DATE	

Tabla: PAGOS

Campo	Tipo de Dato	Restricciones
ID_PAGO	NUMBER PRIMARY	KEY
ID_USUARIO	NUMBER	
MONTO	NUMBER(10	, 2) NOT NULL
FECHA_PAGO	DATE NOT	NULL
CONSTRAINT	FK_PAGO_USUARIO FOREIGN	KEY (ID_USUARIO) REFERENCES USUARIOS(ID_USUARIO)

Bibliografía

De Roer, D. D., & De Roer, D. D. (2022, 12 junio). Conectar Java a una base de datos Oracle. *Disco Duro de Roer* -. https://www.discoduroderoer.es/conectar-java-a-una-base-de-datos-oracle/

Qué es un diagrama entidad-relación. (s. f.). Lucidchart. https://www.lucidchart.com/pages/es/que-es-un-diagrama-entidad-relacion

Oracle Database 19c. (s. f.). https://info.quistor.com/migraci%C3%B3n/

Equipo editorial, Etecé. (2024, 24 diciembre). *Base de datos - Qué es, tipos y ejemplos*. Concepto. https://concepto.de/base-de-datos/

Solutions, V., & Jain, A. (2023, 2 enero). *Qué son los requisitos funcionales: ejemplos, definición, guía completa*. Visure Solutions. https://visuresolutions.com/es/blog/functional-requirements/#:~:text=Un%20requisito%20funcional%20es%20una,caracter%C3%ADsticas%20que%20el%20usuario%20detecta.

IBM. (n.d.). *Cursors in procedures*. IBM Documentation. https://www.ibm.com/docs/es/db2/11.5?topic=procedures-cursors

Villavicencio, G. (2022, Agosto 13). Procedimientos almacenados uso de cursor en SQLServer Parte 2[Video]. YouTube. https://www.youtube.com/watch?v=Lztys1PcMV4