CESUR

Tu Centro Oficial de FP

CESUR

CFGS DAM

PROYECTO FIN DE CICLO

Diseño e Implementación de una Aplicación para la Gestión de una Biblioteca

Para obtener el título de:

Técnico Superior en Desarrollo de Aplicaciones Multiplataforma

Alumno:

Gabriel Fernández Magán

Tutora:

María Jesús Rodríguez Sánchez

Curso Académico:

2024/25

ÍNDICE:

1. Introducción

Esta memoria de proyecto de fin de ciclo para el **CFGS en Desarrollo de Aplicaciones Multiplataforma**, describe el desarrollo de un **Sistema de Gestión de Biblioteca**. Una aplicación informática diseñada para centralizar y automatizar las operaciones fundamentales de una biblioteca. El objetivo es modernizar la administración de recursos bibliográficos y usuarios, mejorando la eficiencia operativa y la experiencia **para el personal de la biblioteca.**

La aplicación servirá como un punto único de control para todo el ciclo de vida de los libros y la interacción con los usuarios, desde la adquisición o registro de nuevos títulos hasta la gestión de préstamos, devoluciones y bajas, así como la administración de los perfiles de los usuarios y la generación de informes relevantes a modo de tablas y ventanas de visualización de datos.

1.1. Motivación

La motivación para desarrollar este sistema son múltiples y responden a la necesidad de modernización y eficiencia en el entorno bibliotecario actual:

- **Superar limitaciones actuales:** Muchas bibliotecas aún dependen de registros manuales, hojas de cálculo o sistemas antiguos que son propensos a errores, lentos y difíciles de escalar.
- **Aumento de la eficiencia operativa:** Reducir la carga de trabajo manual del personal, permitiéndoles dedicar más tiempo a tareas de mayor valor, como la atención al usuario, o la selección y organización de contenido y eventos.
- **Mejorar el servicio al usuario:** Ofrecer un servicio más rápido, preciso y transparente a los usuarios de la biblioteca.
- **Digitalización y sostenibilidad:** Contribuir a la digitalización de los procesos, reduciendo el uso de papel y mejorando la sostenibilidad.
- **Generación de conocimiento:** Transformar datos brutos en información valiosa para la gestión estratégica de la biblioteca.

1.2. ¿Qué Soluciones Aporta?

El sistema de gestión de biblioteca aportará soluciones clave para los desafíos habituales en la administración manual o con sistemas obsoletos:

- **Optimización de la gestión de recursos:** Centraliza la información de todos los libros (disponibilidad y estado), facilitando búsquedas rápidas y precisas.
- **Automatización de procesos:** Simplifica el registro de altas y bajas de usuarios y libros, y automatiza el seguimiento de préstamos y devoluciones, reduciendo errores y el tiempo dedicado a tareas administrativas.
- **Control de préstamos y devoluciones:** Permite registrar con precisión quién tiene qué libro y cuándo debe devolverlo, con funcionalidades para notificaciones de vencimiento, así como el establecimiento de posibles sanciones temporales a usuarios por retrasos en las devoluciones.
- **Mejora de la experiencia del personal de la biblioteca:** Facilita la búsqueda de libros por parte del personal (a través de un módulo de consulta) y agiliza el proceso de préstamo.
- **Acceso a información en tiempo real:** Proporciona al personal de la biblioteca una visión clara e instantánea del inventario, la disponibilidad de libros y el estado de los préstamos.
- **Toma de decisiones basada en datos:** A través de la visualización de datos e informes, la biblioteca puede identificar los libros más populares, los usuarios más activos, los periodos de mayor demanda, etc., lo que ayuda en la adquisición de nuevo material y en la planificación de los servicios ofrecidos..
- **Reducción de pérdidas y deterioro:** Al tener un registro exacto de cada libro y su historial de préstamos, se facilita la identificación de extravíos o daños.

2. Objetivos

Este proyecto tiene como objetivo principal **potenciar la eficiencia y la efectividad del personal de la biblioteca** en la gestión diaria de sus recursos y usuarios.

A través de un módulo administrativo robusto e intuitivo, se busca optimizar los procesos de registro y seguimiento de libros y préstamos, facilitar la administración de la información de los usuarios y proporcionar una visibilidad operativa clara; todo ello para mejorar la experiencia tanto del bibliotecario como, en última instancia, del usuario final.

- **Optimizar la Gestión de Recursos de la Biblioteca:**
    - **Objetivo:** Mejorar la eficiencia del personal de la biblioteca en el registro, actualización y seguimiento del estado de los libros y ejemplares, simplificando las tareas administrativas manuales.

    - **Justificación:** El CRUD de libros y la gestión de ejemplares son fundamentales para una administración eficaz del inventario de la biblioteca por parte de los bibliotecarios.
- **Mejorar el Control y Seguimiento de Préstamos:**
    - **Objetivo:** Establecer un sistema robusto que permita al personal de la biblioteca un registro preciso y en tiempo real de préstamos y devoluciones, agilizando el proceso y reduciendo la posibilidad de errores.
    - **Justificación:** Las funcionalidades de registro, devolución y renovación son el corazón de la operación de préstamos y son críticas para el trabajo diario de los bibliotecarios.
- **Fomentar la Responsabilidad del Usuario y Gestionar Retrasos:**
    - **Objetivo:** Implementar un sistema que permita a los bibliotecarios gestionar eficazmente las notificaciones de préstamos vencidos y aplicar sanciones temporales a los usuarios, promoviendo la devolución puntual de los libros.
    - **Justificación:** Las notificaciones internas y el sistema de sanciones temporales son herramientas clave para que los bibliotecarios mantengan el orden y la disponibilidad de los recursos.
- **Facilitar la Administración y Consulta de Información de Usuarios:**
    - **Objetivo:** Proporcionar al personal de la biblioteca herramientas intuitivas para gestionar eficientemente los datos de los usuarios, su estado y el historial de préstamos, permitiendo un acceso rápido y sencillo a la información relevante.
    - **Justificación:** El CRUD de usuarios y la visualización del historial de préstamos son esenciales para que los bibliotecarios puedan atender y gestionar a los usuarios de la biblioteca.
- **Proveer Visibilidad Operacional para el Personal Administrativo:**
    - **Objetivo:** Ofrecer un dashboard administrativo que presente de forma clara y accesible la información operativa clave de la biblioteca, permitiendo a los bibliotecarios obtener una visión general del estado de la biblioteca y de las operaciones en curso.
    - **Justificación:** El dashboard y las funcionalidades de reporting básico consolidan la información necesaria para que los bibliotecarios tomen decisiones informadas y supervisen el funcionamiento de la biblioteca.
    - Estos objetivos están diseñados para reflejar los beneficios directos para el personal de la biblioteca, haciendo su trabajo más eficiente y efectivo en la gestión diaria de los recursos y usuarios.

3. Alcance del Proyecto

El alcance inicial de este proyecto se centrará, como hemos mencionado previamente, en el desarrollo de un **Módulo Administrativo** robusto e intuitivo para el personal de la biblioteca, que cubra las funcionalidades esenciales de gestión de la biblioteca.

Funcionalidades Incluidas (Módulo Administrativo):

- **Gestión de usuarios:**
  - **CRUD (Crear, Leer, Actualizar, Eliminar) de usuarios:** Registro completo de datos personales, estado (activo/inactivo), y sanciones temporales a usuarios.
  - **Visualización de historial de préstamos por usuario.**
  - **Funcionalidad de búsqueda** de usuarios por varios criterios (Id, Email).
- **Gestión de libros/ejemplares:**
  - **CRUD de libros:** Registro de metadatos del libro (título, autor, ISBN, género, año de publicación, etc.) y gestión de múltiples ejemplares para cada título.
  - **Estado del ejemplar:** Asignación y actualización del estado (disponible, prestado, dado de baja).
  - **Funcionalidad de búsqueda** de libros por varios criterios (Id, ISBN).
- **Gestión de préstamos y devoluciones:**
  - **Registro de préstamos:** Asociación de un ejemplar de libro a un usuario, con fecha de préstamo, fecha de devolución esperada y fecha de devolución real.
  - **Registro de devoluciones:** Marcar un préstamo como devuelto, actualizando la disponibilidad del ejemplar.
  - **Renovaciones de préstamo.**
  - **Sistema de notificaciones internas:** Para préstamos vencidos.
  - **Sistema de sanciones temporales a usuarios:** Por retrasos en las devoluciones de préstamos, al usuario se le impedirá adquirir nuevas reservas/préstamos de libros durante un período de tiempo.
  - **Funcionalidad de búsqueda** de préstamos por diversos criterios (Id de préstamo, Id de libro, ISBN, Id de usuario, Email).
- **Visualización de Datos y Reporting Básico:**
  - **Dashboard administrativo:** Vista general de información clave en tablas y ventanas de visualización de datos (información de libros y ejemplares, libros prestados, usuarios activos, etc.).

Funcionalidades Excluidas (para una fase posterior):

Para mantener el proyecto en un tamaño y complejidad manejable durante su fase inicial, las siguientes funcionalidades **NO** se incluirán en el alcance primario del proyecto, pero serán consideradas para futuras iteraciones:

- **Ampliación de la información:** posibilidad de inclusión de información relevante no reflejada en la versión inicial, como por ejemplo ubicación física de cada ejemplar en la biblioteca o gestión de peticiones de adquisición de nuevos ejemplares, volúmenes y obras.
- **Módulo de usuario/autoservicio:** Interfaz para que los usuarios puedan buscar libros, ver su historial de préstamos, o solicitar reservas por sí mismos.
- **Autenticación y roles:** Junto al módulo de autoservicio, extensión de la aplicación para ser usada tanto por parte del personal de gestión, como por los usuarios de la biblioteca. Necesidad de diseñar e implementar un sistema de gestión de usuarios y roles con identificación de acceso seguro.
- **Sistema de notificaciones automáticas:** Envío de correos electrónicos, SMS o alertas internas a usuarios sobre préstamos vencidos o reservas disponibles.
- **Gestión de adquisiciones:** Proceso de gestión del stock y catalogación avanzada de nuevos libros. Inclusión de peticiones de adquisición y Wishlists.
- **Módulo de reservas/listas de espera.**
- **Integración con sistemas RFID o códigos de barras avanzados.**
- **Análisis de datos avanzado/Business Intelligence (BI):** Más allá de los informes y descriptores básicos.
- **Módulo de reserva de salas de exposición, charlas o eventos.**

La definición de este alcance inicial permite un desarrollo centrado y la entrega de un producto mínimo viable (MVP) que cubra las necesidades más críticas de la gestión de la biblioteca, sentando las bases para futuras expansiones.

4. Análisis de Características

La aplicación se construiría con las siguientes características clave:

- **Gestión de usuarios:**
    - **Altas y bajas:** Registro de nuevos usuarios con sus datos personales (nombre, ID, contacto). Dar de baja o suspender usuarios.

- **Perfiles de usuario:** Consulta, búsqueda y edición de la información de cada usuario.
- **Historial de préstamos:** Visualización del historial de libros prestados por cada usuario.

- **Gestión de libros:**
  - **Altas y bajas:** Registro de nuevos libros (título, autor, ISBN, género, ejemplares disponibles, estado). Dar de baja libros por pérdida, deterioro o descarte.
  - **Consulta de información:** Consulta, búsqueda y edición de los datos de cada libro.
  - **Gestión de Inventario:** Actualización automática del estado y número de ejemplares disponibles.
- **Gestión de préstamos y devoluciones:**
  - **Registro de préstamos:** Asociación de un libro a un usuario con fecha de préstamo, fecha de devolución prevista y fecha de devolución real.
  - **Registro de devoluciones:** Marcar un libro como devuelto, liberando su disponibilidad.
  - **Control de Fechas:** Notificaciones sobre préstamos vencidos.
  - **Renovaciones y modificaciones:** Posibilidad de extender el período de préstamo.
  - **Historial de préstamos:** potente motor de búsqueda de préstamos por diversos criterios.
- **Visualización de datos y reporting:**
  - **Dashboards:** Paneles de control con datos clave (libros prestados, usuarios activos, etc.).
  - **Informes personalizables:** Generación de reportes en ventanas de visualización de datos con información relevante de usuarios, libros y préstamos.
  - **Búsqueda avanzada:** Funcionalidad para filtrar y ordenar datos de préstamos, usuarios y libros.

4.1. Requisitos Funcionales

Estos requisitos describen las operaciones y funcionalidades específicas que el sistema debe proporcionar a los usuarios (bibliotecarios).

I. **Gestión de Libros (CRUD):** [RFI.A-F]

A. El sistema debe permitir a los bibliotecarios **añadir** nuevos libros al catálogo, incluyendo título, autor, ISBN, género, año de publicación y el número de ejemplares disponibles.
B. El sistema debe permitir a los bibliotecarios **actualizar** la información de libros existentes (título, autor, ISBN, género, año de publicación, número de ejemplares y disponibilidad).
C. El sistema debe permitir a los bibliotecarios **eliminar** libros del catálogo.
D. El sistema debe permitir **buscar** libros por su ID o ISBN.
E. El sistema debe **mostrar** una lista completa de todos los libros registrados en el catálogo.
F. El sistema debe **mostrar los detalles completos** de un libro seleccionado, incluyendo su ID, título, autor, ISBN, género, año de publicación, número de ejemplares y estado de disponibilidad, junto con una portada si está disponible.

II. **Gestión de Usuarios (CRUD):** [RFII.A-F]
A. El sistema debe permitir a los bibliotecarios **añadir** nuevos usuarios, incluyendo nombre, apellidos, email, estado de actividad y sanciones adquiridas.
B. El sistema debe permitir a los bibliotecarios **actualizar** la información de usuarios existentes.
C. El sistema debe permitir a los bibliotecarios **eliminar** usuarios del sistema.
D. El sistema debe permitir **buscar** usuarios por su ID o email.
E. El sistema debe **mostrar** una lista completa de todos los usuarios registrados.
F. El sistema debe **mostrar los detalles completos** de un usuario seleccionado.

III. **Gestión de Préstamos y Reservas:** [RFIII.A-G]
A. El sistema debe permitir a los bibliotecarios **registrar nuevos préstamos** de libros a usuarios, especificando el libro, el usuario y la fecha de devolución prevista.
B. El sistema debe permitir a los bibliotecarios **registrar la devolución** de un libro prestado, actualizando el estado de la reserva y marcando el ejemplar del libro como disponible.
C. El sistema debe permitir a los bibliotecarios **cancelar** una reserva existente si está en estado "PENDIENTE".
D. El sistema debe permitir a los bibliotecarios **finalizar** una reserva existente si está en estado "PENDIENTE".
E. El sistema debe permitir a los bibliotecarios **eliminar** reservas que no estén en estado "PENDIENTE".
F. El sistema debe **mostrar** una lista de todas las reservas y préstamos.

G. El sistema debe permitir **buscar** reservas por ID de reserva, ID de libro, ISBN de libro, ID de usuario o email de usuario.

IV. **Validaciones de Negocio para Préstamos/Reservas:** [RFIV.A-K]

A. El sistema debe **validar** que el libro y el usuario existan antes de crear o actualizar una reserva/préstamo.
B. El sistema debe **impedir** la creación de un préstamo/reserva si el libro no está marcado como disponible.
C. El sistema debe **impedir** la creación de un préstamo/reserva si el usuario no está activo.
D. El sistema debe **impedir** la creación de un préstamo/reserva si el usuario está sancionado y su sanción no ha expirado.
E. El sistema debe **limitar** el número de préstamos/reservas "PENDIENTE" que un usuario puede tener simultáneamente (actualmente 5).
F. El sistema debe **impedir** que un usuario tenga más de una reserva/préstamo en estado "PENDIENTE" para el *mismo libro* al mismo tiempo.
G. El sistema debe **establecer automáticamente** la fecha de reserva al día actual si no se especifica.
H. El sistema debe **establecer automáticamente** la fecha de devolución prevista a 15 días desde la fecha de reserva si no se especifica, o bien, notificar al bibliotecario la necesidad de introducir el dato.
I. El sistema debe **establecer el estado inicial** de una nueva reserva/préstamo como "PENDIENTE".
J. El sistema debe **comprobar y gestionar** los ejemplares disponibles, de forma que cuando se registre un préstamo, se devuelva o se cancele; el número de ejemplares para el libro de la reserva se actualizará automáticamente.
K. El sistema debe **impedir** realizar nuevos préstamos de un libro si no se dispone de suficientes ejemplares de ese libro.

V. **Gestión de Sanciones:** [RFV.A-B]

A. El sistema debe **aplicar una sanción** a un usuario si devuelve un libro con retraso, calculando la duración de la sanción en función de los días de retraso.
B. El sistema debe **eliminar automáticamente** la sanción de un usuario si la fecha de fin de sanción ya ha pasado.

VI. **Interfaz de Usuario y Navegación:** [RFVI.A-C]

A. El sistema debe proporcionar una **barra de menú** superior con opciones para navegar entre las secciones de "Gestión de Libros", "Gestión de Usuarios" y "Gestión de Préstamos/Reservas".

B. El sistema debe incluir un **menú "Acerca de"** que muestre información del proyecto y del autor, incluyendo un logo.
C. El sistema debe **mostrar mensajes informativos o de error** claros al usuario a través de diálogos (JOptionPane) para las operaciones CRUD y validaciones.

4.2. Requisitos No Funcionales

Estos requisitos definen las cualidades del sistema y las restricciones bajo las cuales debe operar, asegurando una experiencia de usuario eficiente y un sistema robusto.

I. **Usabilidad:** [RNFI.A-D]
   A. **Interfaz Intuitiva:** La interfaz de usuario (GUI) debe ser intuitiva y fácil de usar para los bibliotecarios, permitiendo la realización de tareas comunes con un mínimo de clics y una curva de aprendizaje reducida.
   B. **Navegación Clara:** La navegación entre las diferentes secciones del sistema (Libros, Usuarios, Reservas) debe ser clara y consistente a través de la barra de menú.
   C. **Feedback al Usuario:** El sistema debe proporcionar retroalimentación clara y oportuna al usuario sobre el éxito o fracaso de las operaciones (ej., "Libro añadido con éxito", "Error al actualizar reserva").
   D. **Atajos de Teclado:** Los menús principales deben tener atajos de teclado (mnemónicos) para una navegación más rápida.
II. **Rendimiento:** [RNFII.A-C]
   A. **Tiempo de Respuesta:** Las operaciones de búsqueda y carga de datos en las tablas (libros, usuarios, reservas) deben ejecutarse de forma fluida y con un tiempo de respuesta aceptable para el usuario (ej., menos de 2−3 segundos para listas de tamaño moderado).
   B. **Eficiencia de Base de Datos:** Las interacciones con la base de datos MariaDB deben ser eficientes, minimizando la latencia y el uso de recursos del servidor.
   C. **Carga de Imágenes:** La carga de las portadas de los libros debe ser rápida y no debe ralentizar la visualización de los detalles del libro.
III. **Fiabilidad:** [RNFIII.A-C]
   A. **Manejo de Errores:** El sistema debe manejar adecuadamente las excepciones (ej., NumberFormatException, IllegalArgumentException, IllegalStateException) y notificar al usuario de manera comprensible en caso de fallos.

B. **Persistencia de Datos:** Todos los datos (libros, usuarios, reservas) deben ser almacenados de forma persistente en la base de datos MariaDB, asegurando que la información no se pierda al cerrar la aplicación.
C. **Estabilidad:** La aplicación debe ser estable y no debe colgarse ni cerrarse inesperadamente durante su uso normal.

IV. **Seguridad:** [RNFIV.A-B]
A. **Integridad de Datos:** La base de datos debe mantener la integridad de los datos mediante el uso de claves primarias, claves foráneas (implícitas por la lógica de negocio), restricciones NOT NULL y UNIQUE (ej., para ISBN).
B. **Validación de Entrada:** El sistema debe validar las entradas del usuario para prevenir datos inconsistentes o maliciosos (ej., asegurar que los años de publicación y el número de ejemplares sean números válidos).

V. **Mantenibilidad:** [RNFV.A-C]
A. **Arquitectura Modular (MVC):** El sistema debe seguir el patrón de diseño Modelo-Vista-Controlador (MVC) para asegurar una clara separación de responsabilidades, facilitando la identificación y corrección de errores, así como la implementación de nuevas funcionalidades.
B. **Código Limpio y Comentado:** El código fuente debe ser claro, legible y estar adecuadamente comentado para facilitar su comprensión y mantenimiento por parte de otros desarrolladores.
C. **Tecnologías Estándar:** El uso de Java 21 (LTS), Swing y MariaDB, que son tecnologías maduras y bien documentadas, contribuye a la facilidad de mantenimiento.

VI. **Portabilidad:** [RNFVI.A-B]
A. **Multiplataforma:** Al estar desarrollada en Java y Swing, la aplicación debe ser ejecutable en diferentes sistemas operativos (Windows, macOS, Linux) sin necesidad de modificaciones en el código fuente.
B. **Base de Datos Estándar:** El uso de MariaDB asegura que la base de datos pueda ser alojada en diversos entornos de servidor.

VII. **Escalabilidad:** [RNFVII.A-B]
A. **Gestión de Datos:** La elección de MariaDB permite que el sistema pueda manejar un volumen creciente de libros, usuarios y reservas sin una degradación significativa del rendimiento en etapas iniciales y medias.
B. **Diseño Modular:** La arquitectura MVC facilita la expansión y adición de nuevas funcionalidades sin afectar drásticamente el código existente.

VIII. **Experiencia de Usuario (GUI):** [RNFVIII.A-B]

A. **Inicio Maximizada:** La ventana principal de la aplicación debe iniciarse en modo maximizado, ocupando toda la superficie de la pantalla, pero manteniendo los controles de ventana y la barra de tareas del sistema operativo visibles.
B. **Tamaño Mínimo:** La ventana principal debe tener un tamaño mínimo definido para asegurar que todos los componentes sean visibles y la interfaz sea utilizable incluso si el usuario intenta redimensionarla a un tamaño muy pequeño.

4.3. Diagramas de Arquitectura y Casos de Uso

A continuación, se muestran los casos de uso principales del ***Sistema de Gestión de Biblioteca***, describiendo las interacciones clave entre los usuarios (bibliotecarios) y el sistema. Cada caso de uso representa una funcionalidad específica que el sistema debe ofrecer para permitir una administración eficiente de libros, usuarios y préstamos; asegurando que las operaciones diarias de la biblioteca se realicen de manera fluida y controlada.

- **Diagrama de arquitectura del sistema:**

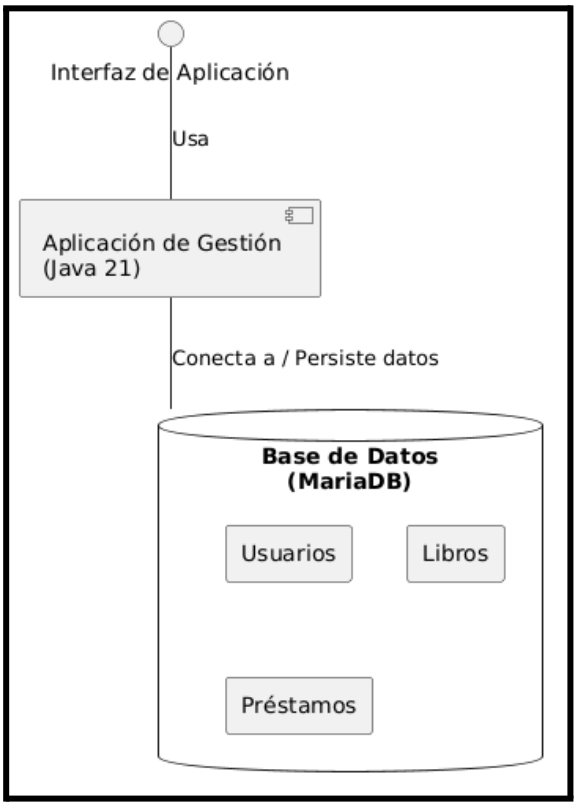

Figura 1. Visión general de los componentes y su organización

- **Caso de Uso, Gestión de Libros:**

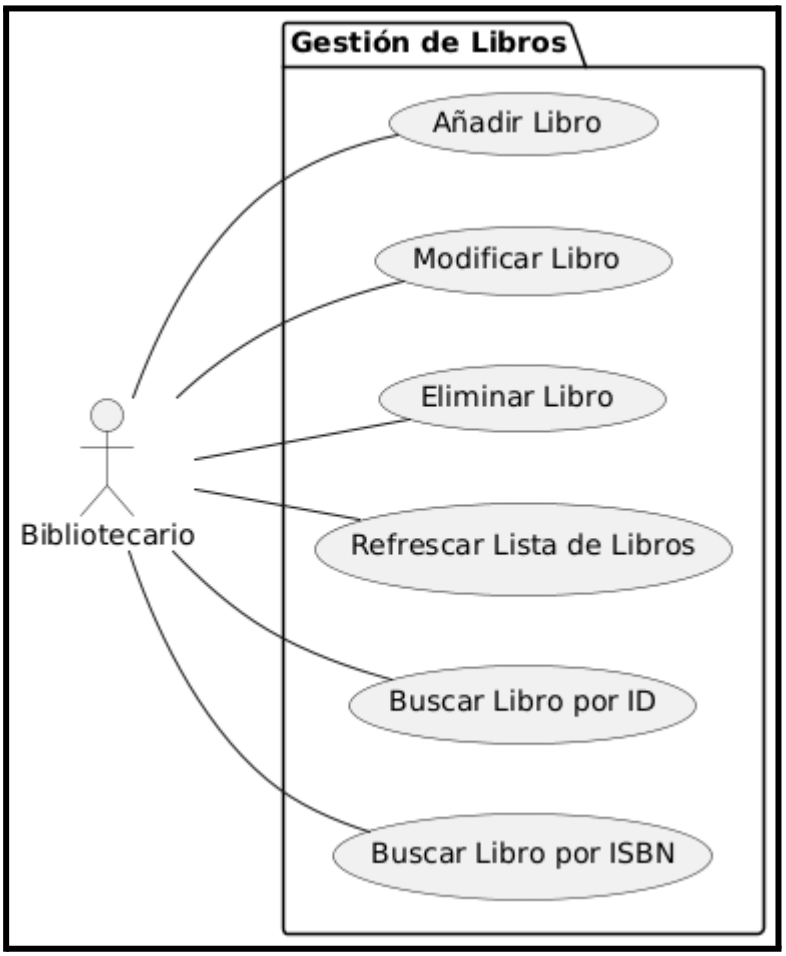

Figura 2. Interacciones para la administración y consulta del catálogo de libros.

- **Caso de Uso, Gestión de Usuarios:**

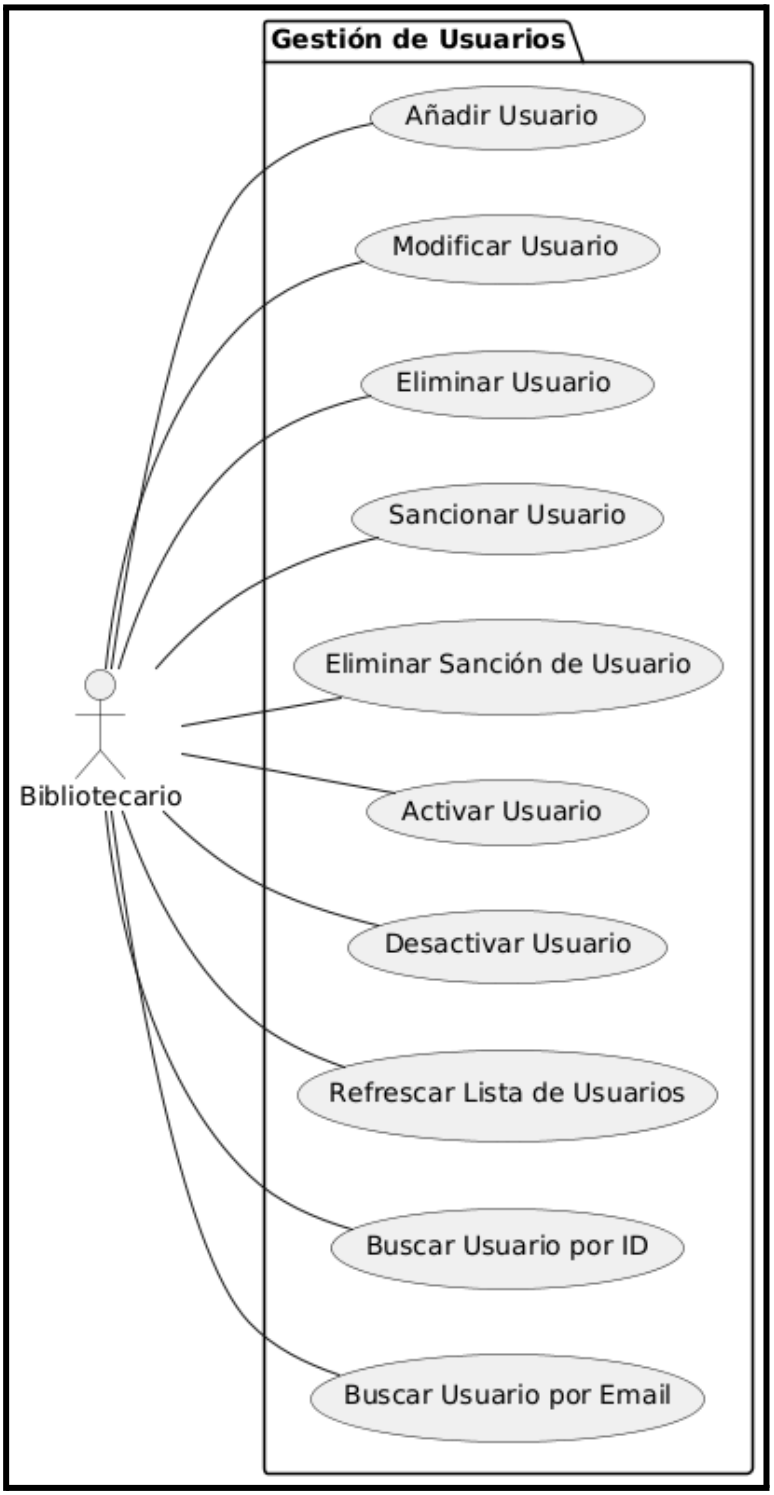

Figura 3. Gestión de la información y estados de los usuarios del sistema.

- **Caso de Uso, Gestión de Reservas/Préstamos:**

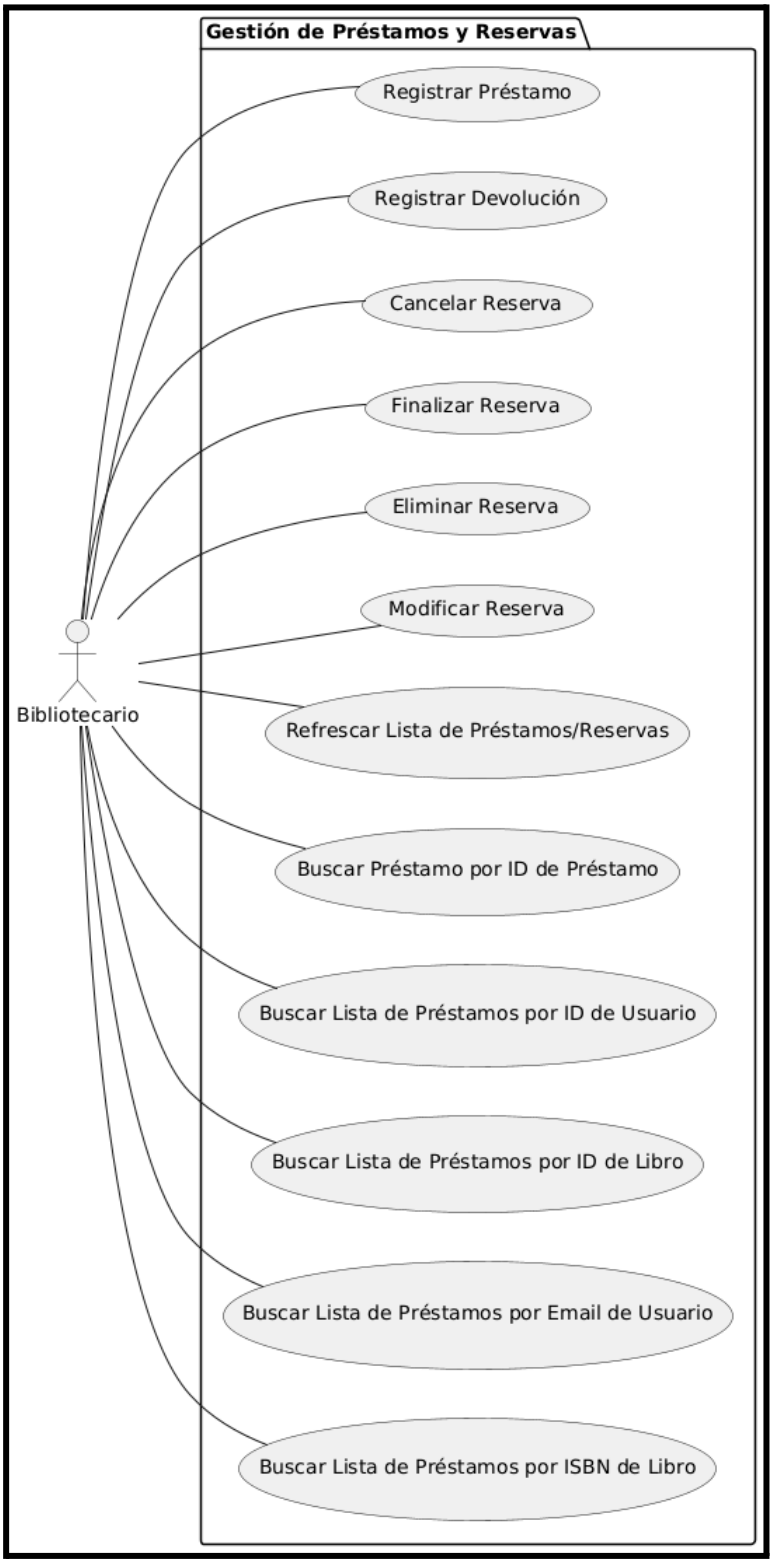

Figura 4. Flujos para el registro, seguimiento y finalización de las transacciones de libros.

5. Tecnologías Clave Utilizadas en el Proyecto

A continuación, se detalla el conjunto de tecnologías que darán vida a este sistema de gestión. Se priorizará la elección de soluciones accesibles a la par de profesionales, y en mi elección particular, se dará preferencia a entornos, APIs, y aplicaciones de código abierto:

- **Modelado y Diseño:**
    - **Modelio:** La fase de diseño y arquitectura del software se llevará a cabo utilizando el software **Modelio**. Esta potente herramienta de modelado permitirá crear diagramas UML y otros modelos visuales detallados, asegurando una planificación exhaustiva y una comprensión clara de la estructura y el comportamiento del sistema antes de la codificación.
- **Entorno de Desarrollo e Implementación:**
    - **NetBeans IDE:** La implementación y desarrollo del código fuente se realizará íntegramente en **NetBeans IDE**, aprovechando su robusto conjunto de herramientas para la programación en Java.
    - **Java 21:** El lenguaje de programación principal para todo el sistema será **Java 21**, la versión LTS más reciente (Long-Term Support). Su fiabilidad, rendimiento y versatilidad garantizan una aplicación estable y eficiente.
- **Gestión de Base de Datos:**
    - **MariaDB:** Para el almacenamiento y la gestión persistente de todos los datos (usuarios, libros, préstamos, etc.), se empleará **MariaDB**. Esta base de datos relacional de código abierto es reconocida por su velocidad, robustez y compatibilidad.
    - **XAMPP:** La administración y configuración de la base de datos MariaDB se realizará a través de **XAMPP**. Esta distribución de Apache permite un entorno de desarrollo local ágil y completo, facilitando la interacción con la base de datos y otras funcionalidades web si fueran necesarias en el futuro.

La combinación de estas tecnologías me permitirá construir un sistema de gestión de biblioteca que no solo cumpla con los requisitos funcionales, sino que también destaque por su

diseño estructurado (*la aplicación será diseñada siguiendo el* ***patrón MVC***), su implementación eficiente y su capacidad para ofrecer un valor real en el entorno bibliotecario contemporáneo.

6. Desarrollo e Implementación

6.1. Modelo de Datos

El modelo de datos de la base de datos del proyecto, biblioteca_db, está diseñado para gestionar los recursos (libros) y los usuarios de una biblioteca, así como las operaciones de préstamo y devolución asociadas. Se trata de un **modelo relacional** simple y eficiente, que establece relaciones claras entre las entidades principales para asegurar la integridad y coherencia de los datos.

Sistema de Gestión de Bases de Datos (SGBD)

El SGBD elegido para esta implementación es **MariaDB**. Este SGBD es una bifurcación (fork) de MySQL, conocida por su rendimiento, robustez y compatibilidad.

Descripción de Tablas y Columnas

La base de datos consta de tres tablas principales: libros, usuarios y reservas.

1. Tabla libros

Almacena la información detallada de cada libro facilitado por la biblioteca.

- **id**:
  - **Tipo:** INT
  - **Clave:** PRIMARY KEY (Clave Primaria)
  - **Restricciones:** AUTO_INCREMENT (se genera automáticamente y es único para cada libro).
  - **Descripción:** Identificador único para cada libro.
- **titulo**:
  - **Tipo:** VARCHAR(255)
  - **Restricciones:** NOT NULL
  - **Descripción:** Título del libro.
- **autor**:

  - **Tipo:** VARCHAR(255)
  - **Restricciones:** NOT NULL
  - **Descripción:** Nombre del autor del libro.
- **isbn**:
  - **Tipo:** VARCHAR(20)
  - **Restricciones:** UNIQUE, NOT NULL
  - **Descripción:** Número Estándar Internacional de Libro (International Standard Book Number), único para cada edición de un libro.
- **genero**:
  - **Tipo:** VARCHAR(100)
  - **Restricciones:** NULL (opcional)
  - **Descripción:** Género literario al que pertenece el libro (ej. "Ficción", "Ciencia Ficción", "Historia").
- **anio_publicacion**:
  - **Tipo:** INT
  - **Restricciones:** NULL (opcional)
  - **Descripción:** Año en que el libro fue publicado.
- **numero_ejemplares**:
  - **Tipo:** INT
  - **Restricciones:** NOT NULL, DEFAULT 1
  - **Descripción:** Cantidad total de ejemplares de este título que posee la biblioteca.
- **disponible**:
  - **Tipo:** BOOLEAN
  - **Restricciones:** DEFAULT TRUE
  - **Descripción:** Indica si al menos un ejemplar del libro está disponible para préstamo (TRUE) o si todos están prestados o dados de baja (FALSE).

2. Tabla usuarios

Almacena la información de los usuarios registrados en la biblioteca.

- **id**:
  - **Tipo:** INT
  - **Clave:** PRIMARY KEY (Clave Primaria)
  - **Restricciones:** AUTO_INCREMENT
  - **Descripción:** Identificador único para cada usuario.
- **nombre**:

  - **Tipo:** VARCHAR(255)
  - **Restricciones:** NOT NULL
  - **Descripción:** Nombre del usuario.
- **apellidos**:
  - **Tipo:** VARCHAR(255)
  - **Restricciones:** NOT NULL
  - **Descripción:** Apellidos del usuario.
- **email**:
  - **Tipo:** VARCHAR(255)
  - **Restricciones:** UNIQUE, NOT NULL
  - **Descripción:** Dirección de correo electrónico del usuario, utilizada como identificador único.
- **telefono**:
  - **Tipo:** VARCHAR(20)
  - **Restricciones:** NULL (opcional)
  - **Descripción:** Número de teléfono de contacto del usuario.
- **activo**:
  - **Tipo:** BOOLEAN
  - **Restricciones:** NOT NULL, DEFAULT TRUE
  - **Descripción:** Indica si la cuenta del usuario está activa (TRUE) o inactiva (FALSE).
- **sancionado**:
  - **Tipo:** BOOLEAN
  - **Restricciones:** NOT NULL, DEFAULT FALSE
  - **Descripción:** Indica si el usuario está actualmente sancionado (TRUE) o no (FALSE).
- **fecha_fin_sancion**:
  - **Tipo:** DATE
  - **Restricciones:** NULL (opcional)
  - **Descripción:** Fecha en la que finaliza la sanción del usuario, si está sancionado.

3. Tabla reservas

Registra los préstamos y devoluciones de libros, actuando como una tabla de unión entre libros y usuarios.

- **id**:
  - **Tipo:** INT

    - **Clave:** PRIMARY KEY (Clave Primaria)
    - **Restricciones:** AUTO_INCREMENT
    - **Descripción:** Identificador único para cada registro de préstamo/reserva.
- **libro_id**:
    - **Tipo:** INT
    - **Clave:** FOREIGN KEY (Clave Foránea, referencia a libros.id)
    - **Restricciones:** NOT NULL, ON DELETE CASCADE
    - **Descripción:** Identificador del libro que ha sido prestado/reservado. ON DELETE CASCADE significa que si un libro es eliminado de la tabla libros, todos sus registros de reserva asociados en esta tabla también serán eliminados.
- **usuario_id**:
    - **Tipo:** INT
    - **Clave:** FOREIGN KEY (Clave Foránea, referencia a usuarios.id)
    - **Restricciones:** NOT NULL, ON DELETE CASCADE
    - **Descripción:** Identificador del usuario que ha realizado el préstamo/reserva. ON DELETE CASCADE significa que si un usuario es eliminado de la tabla usuarios, todos sus registros de reserva asociados en esta tabla también serán eliminados.
- **fecha_reserva**:
    - **Tipo:** DATE
    - **Restricciones:** NOT NULL
    - **Descripción:** Fecha en la que se realizó el préstamo o reserva.
- **fecha_devolucion_prevista**:
    - **Tipo:** DATE
    - **Restricciones:** NOT NULL
    - **Descripción:** Fecha límite en la que se espera que el libro sea devuelto.
- **fecha_devolucion_real**:
    - **Tipo:** DATE
    - **Restricciones:** NULL (opcional)
    - **Descripción:** Fecha real en la que el libro fue devuelto. Será NULL si el libro aún no ha sido devuelto.
- **estado**:
    - **Tipo:** VARCHAR(50)
    - **Restricciones:** NOT NULL, DEFAULT 'PENDIENTE'
    - **Descripción:** Estado actual del préstamo (ej. 'PENDIENTE', 'DEVUELTO', 'RETRASO', 'CANCELADO').

Relaciones entre Entidades

El modelo de datos establece las siguientes relaciones clave:

- **libros y reservas (Uno a Muchos):**
  - Un libro puede tener **cero o muchas** reservas (préstamos) asociadas a él.
  - Cada reserva está asociada a **un único** libro.
  - La relación se implementa a través de la **clave foránea** libro_id en la tabla reservas, que referencia a libros.id.
- **usuarios y reservas (Uno a Muchos):**
  - Un usuario puede tener **cero o muchas** reservas (préstamos) asociadas a él.
  - Cada reserva está asociada a **un único** usuario.
  - La relación se implementa a través de la **clave foránea** usuario_id en la tabla reservas, que referencia a usuarios.id.

Modelo Relacional de las Tablas con Estructuras, Claves y Restricciones

A continuación, se presenta el **diagrama Entidad - Relación** de la estructura relacional:

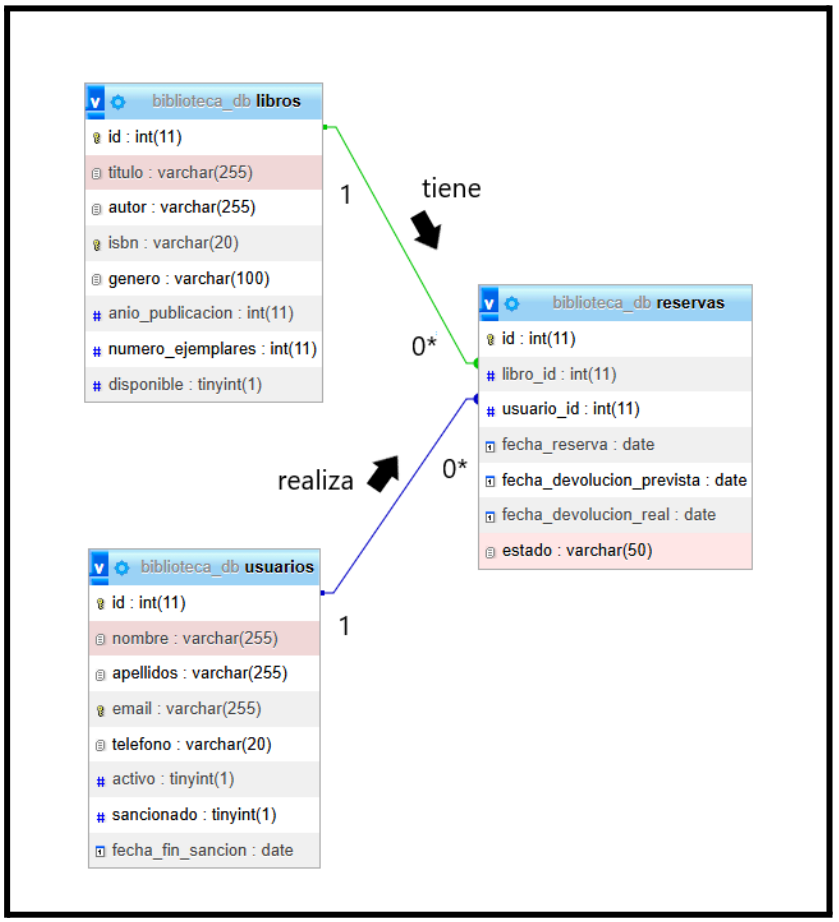

Figura 5. Diagrama E-R de la base de datos del proyecto biblioteca_db.

Implementación de la Base de Datos en Lenguaje de Consulta

La implementación de la base de datos se realiza utilizando **SQL (Structured Query Language)**, que es el lenguaje estándar para interactuar con bases de datos relacionales. Las sentencias CREATE DATABASE, CREATE TABLE, CREATE USER, GRANT y USE son ejemplos de SQL DDL (Data Definition Language) y DCL (Data Control Language) utilizados para definir la estructura de la base de datos y gestionar los permisos.

Motor de Base de Datos Elegido

Como se mencionó anteriormente, el motor de base de datos elegido para esta implementación es **MariaDB**. Este motor es una excelente opción para aplicaciones web y de gestión, ofreciendo un buen equilibrio entre rendimiento, facilidad de uso y características avanzadas.

Asignación de Dependencias de Maven (en NetBeans)

```
<dependencies>
    <dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>3.3.3</version> </dependency>
</dependencies>
```

Este fragmento de código en el fichero `pom.xml` sirve para que Maven descargue e incluya el **driver JDBC de MariaDB** en el proyecto Java. Gracias a esto, la aplicación Java podrá establecer la conexión y comunicarse con la base de datos de la aplicación en MariaDB.

6.2. Estructura del Proyecto

Ventajas de usar el patrón MVC

El patrón MVC es ampliamente utilizado en el desarrollo de aplicaciones por sus múltiples beneficios:

- **Separación de Responsabilidades:** Divide la aplicación en tres componentes interconectados, lo que facilita que cada parte se enfoque en una tarea específica. Esto reduce la complejidad y hace el código más manejable.
- **Reusabilidad:** Los componentes pueden ser más fácilmente reutilizados en diferentes partes de la aplicación o incluso en otros proyectos.
- **Mantenibilidad:** Los cambios en una capa (por ejemplo, la interfaz de usuario) tienen un impacto mínimo en las otras capas (la lógica de negocio o los datos), lo que simplifica las actualizaciones y correcciones de errores.
- **Desarrollo Colaborativo:** Permite que diferentes equipos o desarrolladores trabajen en distintas partes de la aplicación simultáneamente sin interferir constantemente entre sí.

Estructura de Directorios y Funcionalidades

A continuación, se detalla la estructura del proyecto, explicando la función de cada directorio y fichero:

- **ProyectoDAM/**:
  - **Función:** Directorio raíz del proyecto. Contiene todos los archivos y subdirectorios relacionados con el proyecto.
- **pom.xml**:
  - **Función:** Archivo de configuración de Maven (Project Object Model). Define las dependencias del proyecto, plugins, versiones, y la forma en que se construye el proyecto.
- **resources/ (raíz del proyecto)**:
  - **Función:** Contiene recursos generales del proyecto que no son específicos del código fuente ni de los recursos de la aplicación.
  - **resources/BBDD:** Carpeta para almacenar todo lo referente a la BBDD.
  - **biblioteca_db.sql:** Es el fichero de la base de datos del proyecto.
  - **consultas_BBDD.txt**: Fichero de texto para almacenar consultas SQL que moldean y dan estructura a la BBDD (creación de tablas, usuarios, etc) o scripts.
  - **readme.txt**: Archivo de texto con información básica del proyecto.
- **src/**:
  - **Función:** Directorio principal que contiene el código fuente y los tests del proyecto.
  - **main/**: Contiene el código fuente principal de la aplicación.
  - **main/java/**: Contiene el código fuente Java de la aplicación.
  - **main/java/com/cesur/biblioteca/**: Paquete raíz de la aplicación, siguiendo la convención de nombres de paquetes Java (dominio inverso).

- **Gestorbiblioteca.java**: Clase principal de la aplicación.
- **Función:** El punto de entrada principal de la aplicación Java. Contiene el método main() que inicia la aplicación.

- **main/java/com/cesur/biblioteca/controlador/ (MVC: Controller)**:
  - **Función:** Contiene la lógica de control que maneja las interacciones del usuario y actualiza el Modelo y la Vista. Actúa como intermediario entre la Vista y el Modelo.
  - **LibroController.java**: Maneja las operaciones relacionadas con los libros (ej. añadir, editar, eliminar libros desde la interfaz).
  - **ReservaController.java**: Gestiona las operaciones de préstamos y devoluciones.
  - **UsuarioController.java**: Controla las acciones relacionadas con los usuarios.
- **main/java/com/cesur/biblioteca/dao/ (Data Access Object)**:
  - **Función:** Capa de acceso a datos. Contiene clases que interactúan directamente con la base de datos (*MariaDB*). Abstrae la lógica de persistencia del resto de la aplicación.
  - **ConexionBBDD.java**: Gestiona la conexión a la base de datos.
  - **LibroDAO.java**: Implementa las operaciones CRUD para la entidad Libro.
  - **LibroDAOInterface.java**: Interfaz que define los métodos que LibroDAO debe implementar, promoviendo la abstracción y el desacoplamiento.
  - **ReservaDAO.java**: Implementa las operaciones CRUD para la entidad Reserva.
  - **ReservaDAOInterface.java**: Interfaz que define los métodos que ReservaDAO debe implementar, promoviendo la abstracción y el desacoplamiento.
  - **UsuarioDAO.java**: Implementa las operaciones CRUD para la entidad Usuario.
  - **UsuarioDAOInterface.java**: Interfaz que define los métodos que UsuarioDAO debe implementar, promoviendo la abstracción y el desacoplamiento.
- **main/java/com/cesur/biblioteca/modelo/ (MVC: Model)**:
  - **Función:** Representa la lógica de negocio y los datos de la aplicación. Contiene las clases de entidad (**POJOs**) que mapean a las tablas de la base de datos.

- **EstadoReserva.java**: Declaración de tipo enum que define los estados posibles de una reserva.
- **Libro.java**: Clase que representa la entidad Libro con sus atributos.
- **Reserva.java**: Clase que representa la entidad Reserva con sus atributos y relaciones.
- **Usuario.java**: Clase que representa la entidad Usuario con sus atributos.

- **main/java/com/cesur/biblioteca/servicio/ (Service Layer)**:
  - **Función:** Capa de lógica de negocio. Contiene la lógica de alto nivel que coordina las operaciones entre los DAOs y los Controladores. Aquí se aplican las reglas de negocio.
  - **LibroService.java**: Implementa la lógica de negocio para los libros.
  - **LibroServiceInterface.java**: Interfaz que define los métodos que LibroService debe implementar.
  - **ReservaService.java**: Implementa la lógica de negocio para las reservas.
  - **ReservaServiceInterface.java**: Interfaz que define los métodos que ReservaService debe implementar.
  - **UsuarioService.java**: Implementa la lógica de negocio para los usuarios.
  - **UsuarioServiceInterface.java**: Interfaz que define los métodos que UsuarioService debe implementar.
- **main/java/com/cesur/biblioteca/vista/ (MVC: View)**:
  - **Función:** Se encarga de la presentación de los datos al usuario y de capturar sus interacciones. No contiene lógica de negocio.
  - **MainFrame.java**: La ventana principal de la interfaz gráfica de usuario (GUI).
  - **main/java/com/cesur/biblioteca/vista/libro/**: Contiene componentes de la interfaz de usuario específicos para la gestión de libros.
  - **LibroDetailDialog.java**: Diálogo para ver o editar los detalles de un libro.
  - **LibroManagementPanel.java**: Panel para listar y gestionar libros (ej. tabla de libros, botones de añadir/editar/eliminar/etc).
  - **main/java/com/cesur/biblioteca/vista/reserva/**: Contiene componentes de la interfaz de usuario específicos para la gestión de reservas.
  - **ReservaDetailDialog.java**: Diálogo para ver o editar los detalles de una reserva.
  - **ReservaManagementPanel.java**: Panel para listar y gestionar reservas (ej. tabla de reservas, botones de Registrar Préstamo, Devolver Préstamo, etc).

        - **main/java/com/cesur/biblioteca/vista/usuario/**: Contiene componentes de la interfaz de usuario específicos para la gestión de usuarios.
        - **UsuarioDetailDialog.java**: Diálogo para ver o editar los detalles de un usuario.
        - **UsuarioManagementPanel.java**: Panel para listar y gestionar usuarios (ej. tabla de usuarios, botones de añadir/editar/eliminar/etc).
    - **main/resources/**:
        - **Función:** Contiene recursos estáticos de la aplicación que serán empaquetados con el JAR (imágenes, archivos de configuración, etc.).
        - **cesur_logo.png**: Imagen del logo de Cesur.
        - **main/resources/portadas/**: Directorio para almacenar las imágenes de portada de los libros.
        - **main/resources/fotos_usuarios/**: Directorio para almacenar las imágenes de perfil de los usuarios.
    - **test/**:
        - **Función:** Contiene el código fuente de los tests unitarios y de integración para asegurar la calidad y el correcto funcionamiento de la aplicación.
        - **test/java/**: Directorio para los archivos de código fuente de los tests Java.
- **target/**:
    - **Función:** Directorio generado por Maven durante el proceso de construcción del proyecto. Contiene los archivos compilados (.class), los JARs, y otros artefactos generados. Es un directorio temporal que se puede eliminar y regenerar.

Esta estructura conforma la jerarquía típica de un proyecto Java MVC, como puede apreciarse, promueve la organización, la separación de responsabilidades y la facilidad de mantenimiento.

6.3. Diagrama de Clases

Véase documento adjunto, **Diagrama de Clases.pdf**.

7. Repositorio Github del Proyecto

Enlace a repositorio Github del proyecto: https://github.com/GabrielFM16/ProyectoDAM

8. Interfaz Gráfica de Usuario

- **Vista de la Ventana Principal:**

Imagen 1. Ventana Principal.

- **Vista de la Ventana de Gestión de Libros:**

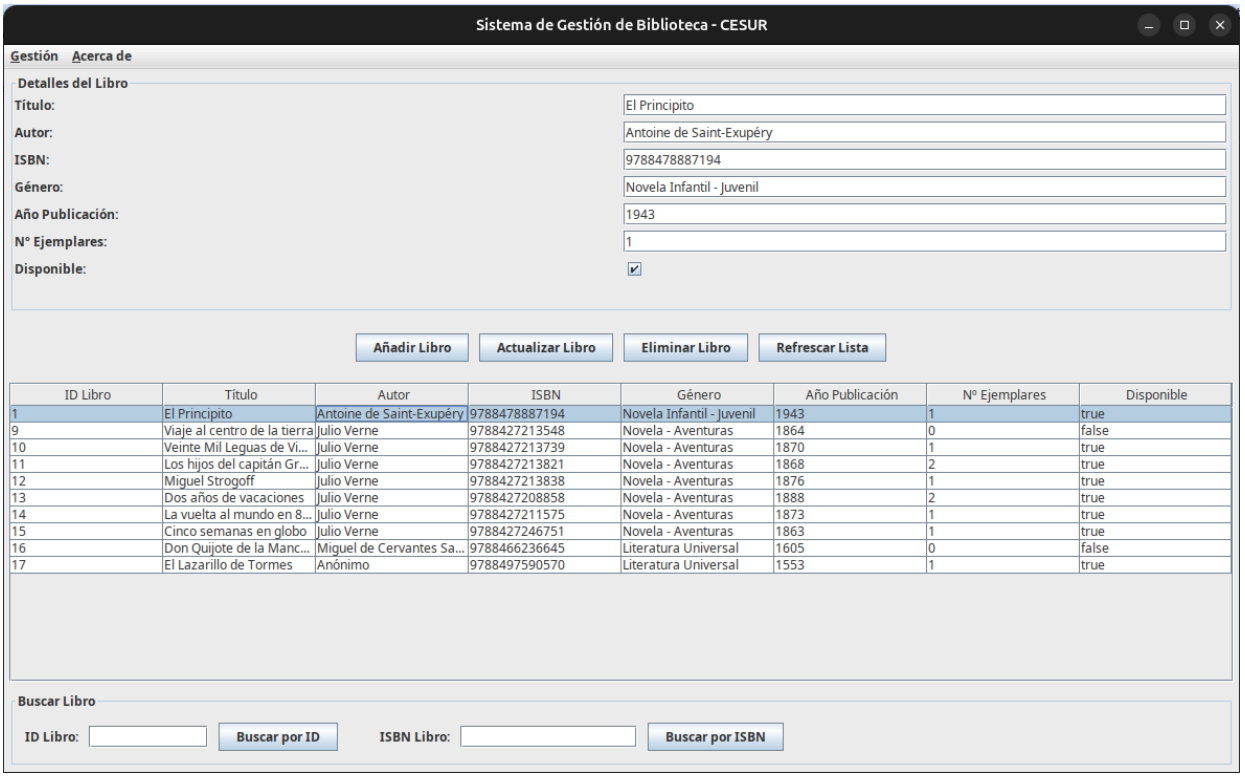

Imagen 2. Ventana de Gestión de Libros.

- **Vista de la Ventana de Visualización de Datos de Libro:**

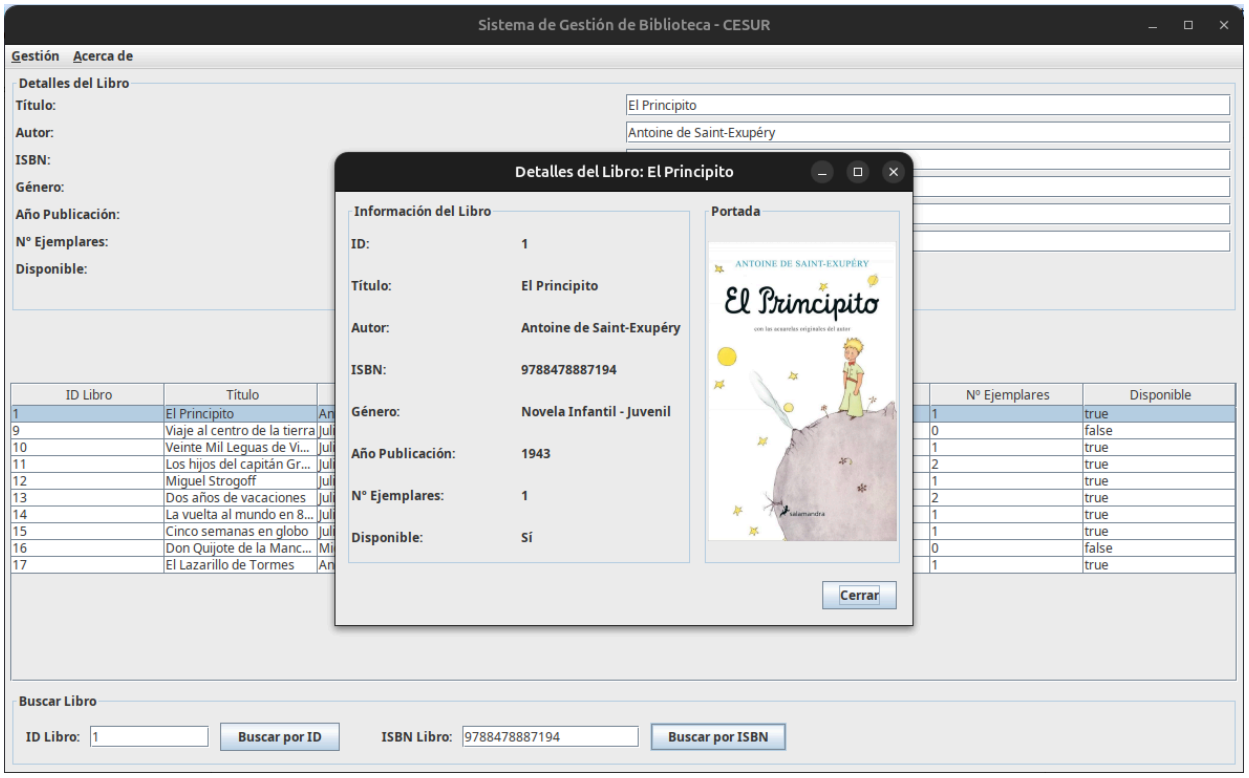

Imagen 3. Ventana de Detalle de Libro.

- **Vista de la Ventana de Gestión de Usuarios:**

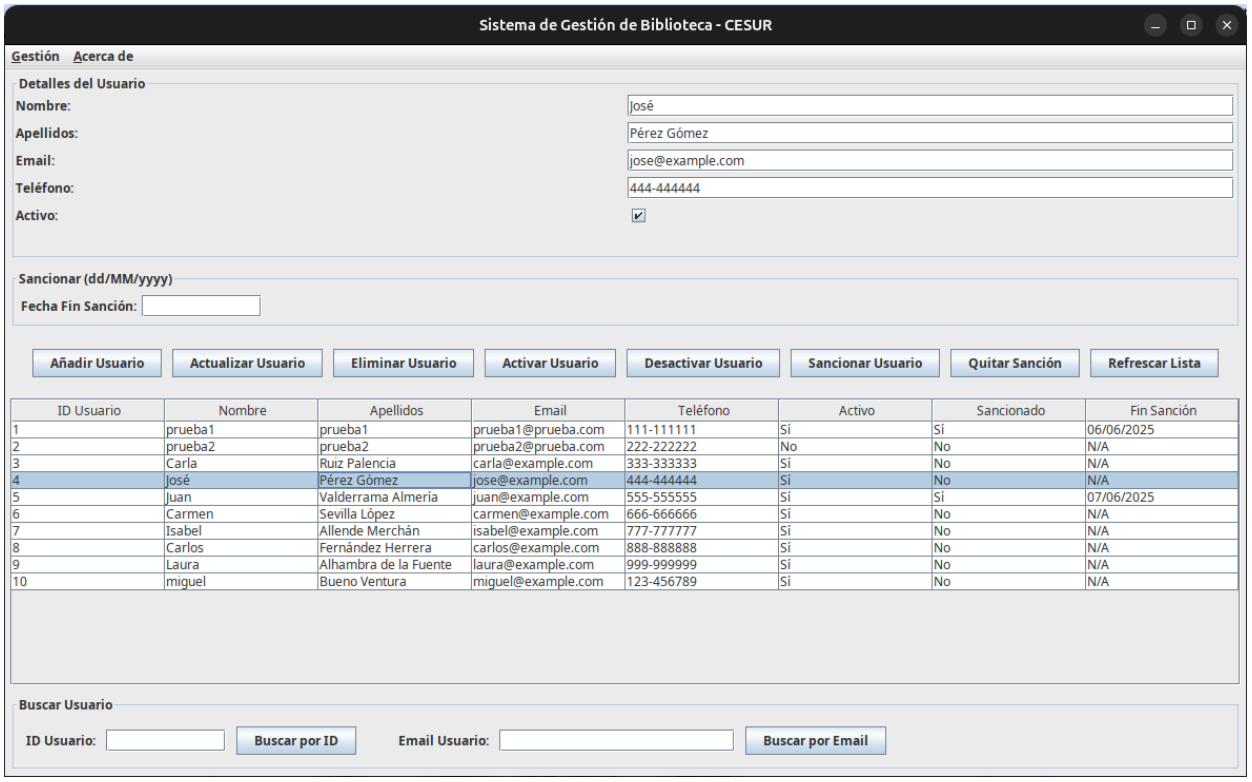

Imagen 4. Ventana de Gestión de Usuarios.

- **Vista de la Ventana de Visualización de Datos de Usuario:**

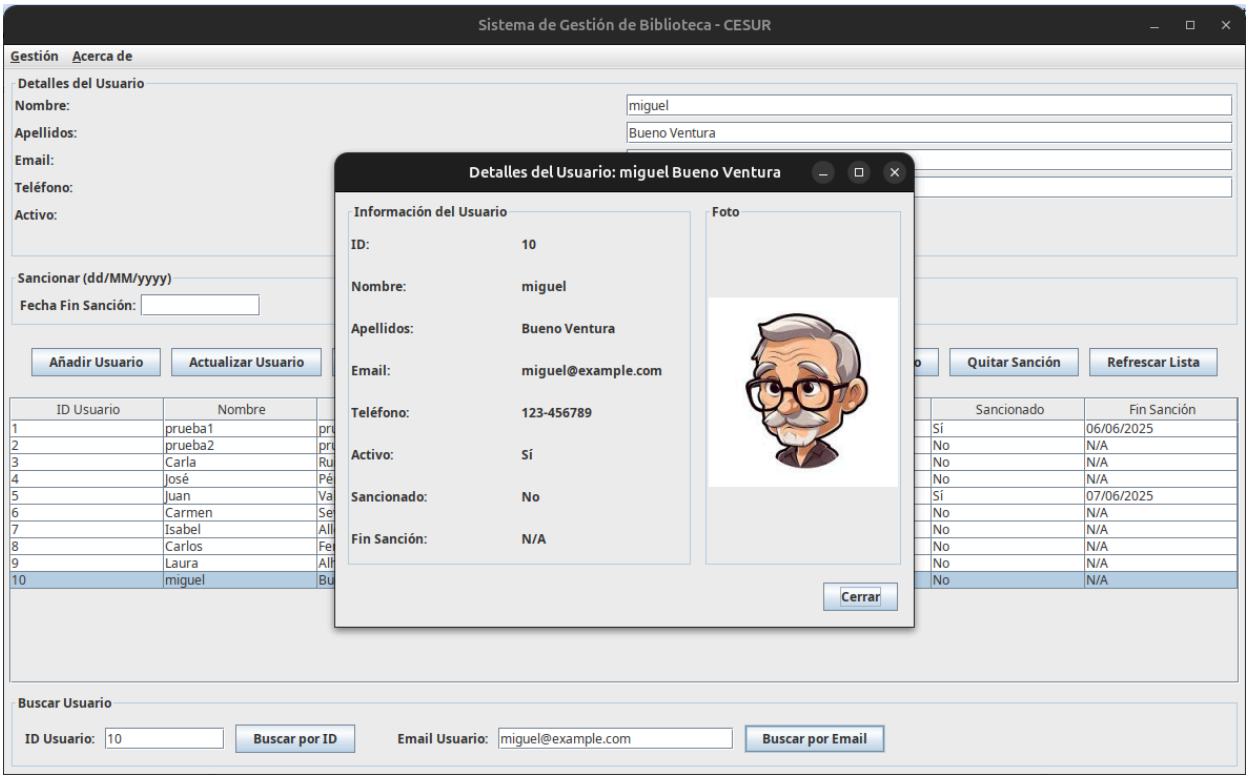

Imagen 5. Ventana de Detalle de Usuario.

- **Vista de la Ventana de Gestión de Préstamos:**

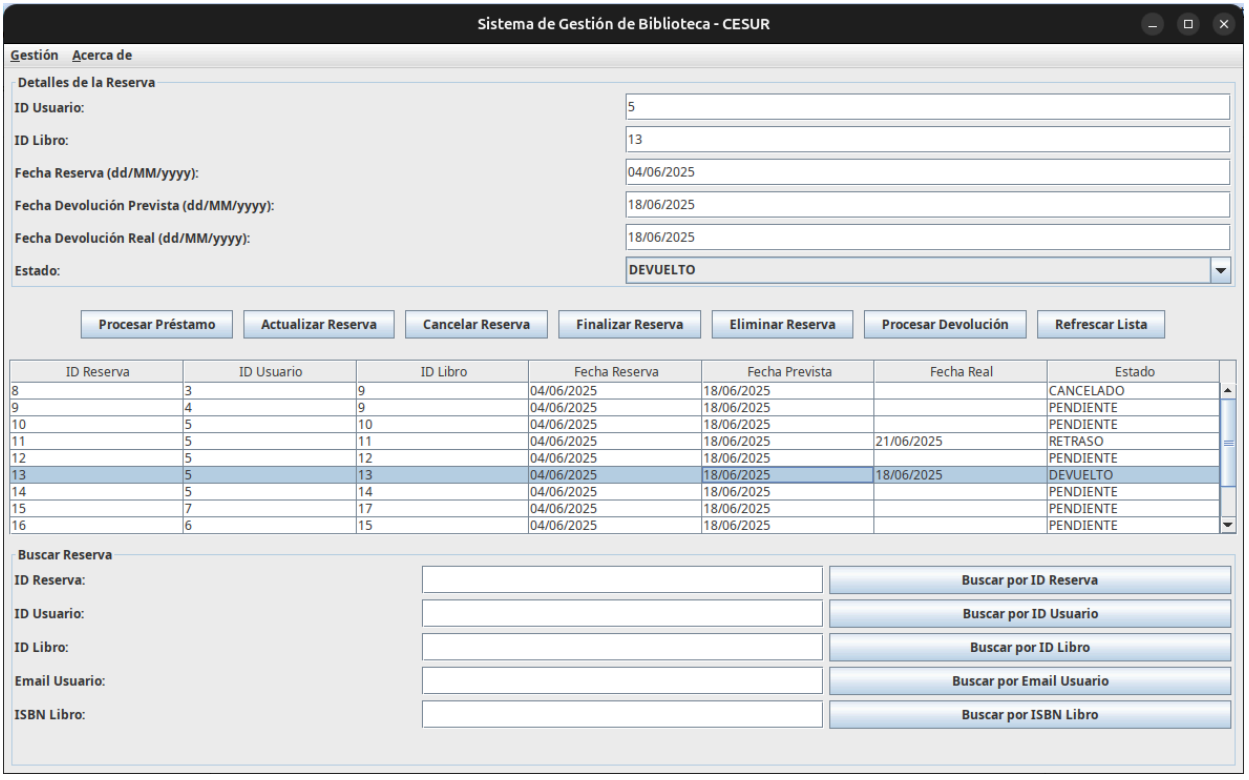

Imagen 6. Ventana de Gestión de Reservas.

- **Vista de la Ventana de Visualización de Datos de Préstamo:**

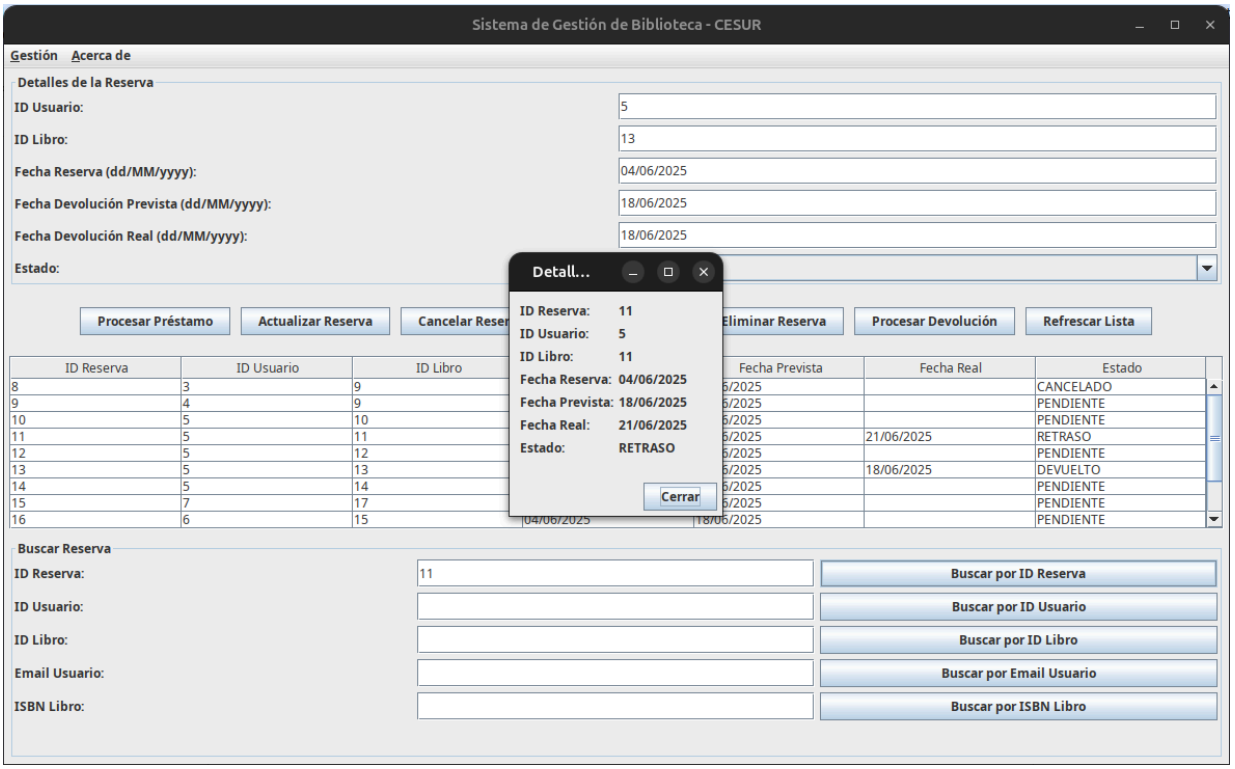

Imagen 7. Ventana de Detalle de Préstamo.

- **Vista de la Ventana de Acerca de/Información de la Aplicación:**

- Imagen 8. Ventana de Acerca de/Información.

9. Bibliografía y Webgrafía

Principalmente documentación de java en **Oracle y W3Schools**. Así como uso de buscadores y foros para solventar dudas.

- https://www.w3schools.com/java/
- https://docs.oracle.com/en/java/
- https://stackoverflow.com/questions/tagged/java

Anexo 1: Generación de un JAR Ejecutable "Fat JAR" en NetBeans (Proyecto Maven)

Este anexo te guiará a través del proceso de crear un archivo .jar que contenga todo lo necesario para que la aplicación Java se ejecute en cualquier lugar, incluyendo el código compilado y todas sus dependencias (como librerías externas). Este tipo de JAR se conoce como "Fat JAR" o "Uber JAR" y es ideal para la distribución.

Contexto: Este procedimiento está diseñado para **proyectos Maven en NetBeans**. La clase principal, la que contiene el método public static void main(String[]args), en mi caso particular, se llama com.cesur.biblioteca.GestorBiblioteca.

Pasos para crear el "Fat JAR":

1. **Modifica el archivo pom.xml:**
    - En NetBeans, abre la ventana **Projects** (Proyectos).
    - Expande el proyecto Maven.
    - Dentro de la carpeta **Project Files**, haz doble clic en pom.xml para abrirlo en el editor de código.
    - **Inserta la siguiente sección <build>** justo antes de la etiqueta de cierre </project>. Este bloque XML le dice a Maven que use el maven-assembly-plugin para empaquetar tus dependencias:

      <build> <plugins> <plugin> <artifactId>maven-assembly-plugin</artifactId> <version>3.6.0</version> <configuration> <archive> <manifest> <mainClass>com.cesur.biblioteca.GestorBiblioteca</mainClass> </manifest> </archive> <descriptorRefs> <descriptorRef>jar-with-dependencies</descriptorRef> </descriptorRefs> </configuration> <executions> <execution> <id>make-assembly</id> <phase>package</phase> <goals> <goal>single</goal> </goals> </execution> </executions> </plugin> </plugins> </build>

    - **Guarda** los cambios en el archivo pom.xml.
2. **Construye de nuevo el Proyecto:**
    - En la ventana **Projects** de NetBeans, haz clic derecho sobre el proyecto.
    - Selecciona la opción **"Clean and Build"** (Limpiar y Construir).

3. **Localiza el "Fat JAR":**
    - Una vez que el proceso de construcción finalice (verás un mensaje de "BUILD SUCCESS" en la ventana **Output**), el archivo JAR listo para distribuir estará en la carpeta target dentro del directorio de tu proyecto.
    - El nombre del archivo será similar a: **ProyectoDAM-1.0-SNAPSHOT-jar-with-dependencies.jar**.

Cómo ejecutar el "Fat JAR":

Este JAR es completamente autónomo. Puedes copiarlo a cualquier sistema que tenga un **Java Runtime Environment (JRE)** instalado y ejecutarlo desde la terminal (en Linux, macOS o Windows) con el siguiente comando:

java -jar ProyectoDAM-1.0-SNAPSHOT-jar-with-dependencies.jar

(Recuerda reemplazar ProyectoDAM-1.0-SNAPSHOT-jar-with-dependencies.jar con el nombre exacto de tu archivo JAR si es diferente).

Anexo 2: Importación de una Base de Datos en phpMyAdmin

PhpMyAdmin es una herramienta basada en web muy popular y fácil de usar para gestionar bases de datos MySQL y MariaDB. Es ideal para entornos de desarrollo local, como los proporcionados por XAMPP, WAMP o MAMP.

Requisitos previos:

- Un navegador web (Chrome, Firefox, Edge, etc.).
- Acceso a tu instalación de phpMyAdmin (ya sea local o en un servidor remoto).
- El archivo de tu base de datos en formato **.sql**, que generalmente se obtiene exportando una base de datos existente.

Pasos para importar tu base de datos:

1. **Accede a phpMyAdmin:**
    - Abre tu navegador web.

- Dirígete a la URL de tu instalación de phpMyAdmin. Si es local, la más común es http://localhost/phpmyadmin.
- Inicia sesión con tus credenciales de usuario y contraseña (las predeterminadas suelen ser root sin contraseña en muchas configuraciones locales, usuario "bibliotecario" y contraseña "12345" en mi caso particular).

2. **Selecciona o crea una Base de Datos:**
   - En la barra lateral izquierda de phpMyAdmin, verás una lista de tus bases de datos existentes.
   - **Si ya tienes una base de datos** donde quieres importar los datos, simplemente haz clic en su nombre para seleccionarla.
   - **Si necesitas crear una nueva base de datos** para la importación:
     - Haz clic en la pestaña **"Bases de datos"** en la parte superior de la interfaz.
     - En el campo "Crear base de datos", introduce el **nombre** que desees para tu nueva base de datos (por ejemplo, biblioteca_db).
     - Elige la **codificación de caracteres** adecuada (generalmente utf8mb4_unicode_ci es una buena práctica).
     - Haz clic en el botón **"Crear"**.
3. **Importa el archivo .sql:**
   - Una vez que hayas seleccionado la base de datos (o estés dentro de la nueva base de datos que creaste), haz clic en la pestaña **"Importar"** en la parte superior.
   - En la sección "Fichero para importar":
     - Haz clic en el botón **"Examinar..."** (o "Choose File", dependiendo de la versión de phpMyAdmin).
     - Navega en tu sistema de archivos hasta donde guardaste tu archivo **.sql** y selecciónalo.
   - **Opciones de Importación:** Normalmente, los valores por defecto son correctos. Asegúrate de que el "Formato" esté configurado como "SQL".
   - Haz clic en el botón **"Continuar"** (o "Go") en la parte inferior de la página.
4. **Verifica la Importación:**
   - phpMyAdmin procesará el archivo SQL. Una vez que termine, deberías ver un mensaje de éxito, como "Importación finalizada con éxito, X consultas ejecutadas".
   - Ahora, en la barra lateral izquierda, expande el nombre de tu base de datos para ver todas las tablas que se han importado.

¡Con estos dos procesos completos, tu aplicación Java debería estar lista para ser distribuida y ejecutada, y tu base de datos configurada en phpMyAdmin!