Logotipo

El contenido generado por IA puede ser incorrecto.

**CATEDRATICO**

LOUIS CARLOS ZELAYA VALLEJO

**ALUMNOS**

DAMNY MEZA ROMERO

201210070008

DIMAS BELSASAR HERNANDEZ HERRERA

202410110004

**CARRERA**

INGENIERIA ENCOMPUTACION

**Sistema de Biblioteca**

INDICE

[Introducción 2](#_Toc203514355)

[Explicación y funcionalidad 3](#_Toc203514356)

[Descripción de validaciones implementadas 7](#_Toc203514357)

[Validaciones implementadas en el formulario Registrar Devolución 7](#_Toc203514358)

[Validaciones implementadas en el formulario Registro de libros 9](#_Toc203514359)

[Validaciones implementadas en el formulario Registro de préstamo 11](#_Toc203514360)

[Validaciones implementadas en el formulario Registro de usuario 13](#_Toc203514361)

[Uso de las funciones CRUD 15](#_Toc203514362)

[Implementación de la conexión DAO + JDBC 16](#_Toc203514363)

# Introducción

El presente proyecto consiste en el desarrollo de un **Sistema de Gestión de Biblioteca** con interfaz gráfica de usuario (GUI) implementada en Java utilizando Swing y NetBeans. Este sistema ha sido diseñado como parte del segundo parcial de la asignatura, con el objetivo de aplicar los conceptos de programación orientada a objetos, acceso a bases de datos y buenas prácticas de desarrollo de software.

El sistema permite la gestión integral de una biblioteca, facilitando la administración de información clave como **libros, usuarios, préstamos y devoluciones**. Para ello, se ha construido una interfaz visual amigable y funcional, que se conecta en tiempo real a una base de datos MySQL mediante JDBC, garantizando la persistencia y seguridad de los datos.

La arquitectura del proyecto sigue el patrón **DAO (Data Access Object)**, separando claramente la lógica de acceso a datos de la lógica de presentación y control. Cada módulo principal del sistema cuenta con su propia ventana (JFrame) que implementa las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre las tablas correspondientes, permitiendo al usuario interactuar de manera sencilla a través de componentes gráficos como JTable, JTextField, JComboBox, JButton y JDialog para formularios de registro y edición.

Este proyecto no solo cumple con los requisitos técnicos y funcionales establecidos, sino que también promueve la organización del código y la escalabilidad del sistema, sentando las bases para futuras mejoras o ampliaciones.

# Explicación y funcionalidad

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.Registro de devolución

Esta pantalla permite registrar la devolución de un libro en la biblioteca. El usuario ingresa el ID del préstamo, el nombre, la fecha de devolución y cualquier observación relevante. Al hacer clic en "Guardar", la devolución queda registrada en el sistema; el botón "Cerrar" permite salir del formulario sin guardar cambios.

Registro de libros

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Este formulario permite ingresar la información de un nuevo libro en la biblioteca. El usuario debe completar los campos de título, categoría, año de publicación, ISBN y editorial. Al hacer clic en "Insertar", el libro se registra en la base de datos; el botón "Salir" cierra el formulario sin guardar cambios.

Registro de préstamo

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Este formulario permite registrar un nuevo préstamo de libro en la biblioteca. El usuario debe ingresar los datos del libro, el usuario que lo solicita, la fecha de préstamo y la fecha de devolución prevista. Al hacer clic en "Guardar" o "Insertar", el préstamo queda registrado en la base de datos; el botón "Salir" o "Cerrar" permite salir del formulario sin guardar cambios.

Registro de usuario

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Esta pantalla permite gestionar los usuarios de la biblioteca. Muestra una tabla con los datos de cada usuario (id, nombre, apellido, tipo y correo electrónico) y ofrece botones para agregar, actualizar o eliminar usuarios. Así, facilita el control y la administración eficiente de los usuarios registrados en el sistema.

Listado de devolución

Tabla

El contenido generado por IA puede ser incorrecto.

Esta pantalla muestra un listado de todas las devoluciones registradas en la biblioteca. Permite visualizar el ID, el ID del préstamo, el nombre del usuario, la fecha de devolución y las observaciones asociadas a cada devolución. Además, ofrece botones para agregar, actualizar o eliminar registros, facilitando la gestión y control de las devoluciones en el sistema.

Listado de libros

Tabla

El contenido generado por IA puede ser incorrecto.

Esta pantalla muestra el listado de todos los libros registrados en la biblioteca. Permite visualizar los datos principales de cada libro, como el id, título, categoría, año de publicación, ISBN y editorial. Además, cuenta con botones para insertar, actualizar o eliminar libros, facilitando la gestión eficiente del inventario de la biblioteca.

Listado de prestamos

Tabla

El contenido generado por IA puede ser incorrecto.

Esta pantalla muestra todos los préstamos registrados en la biblioteca, incluyendo información como el ID del préstamo, ejemplar, usuario, empleado responsable, fechas de préstamo, vencimiento y devolución. Además, permite agregar, actualizar o eliminar préstamos mediante los botones correspondientes, facilitando así la gestión y el control de los préstamos de libros en el sistema.

Listado de usuarios

Interfaz de usuario gráfica, Tabla

El contenido generado por IA puede ser incorrecto.

Esta pantalla muestra el listado de todos los usuarios registrados en la biblioteca, incluyendo su id, nombre, apellido, tipo (docente o estudiante) y correo electrónico. Permite agregar, actualizar o eliminar usuarios mediante los botones correspondientes, facilitando así la gestión eficiente de los usuarios en el sistema.

# Descripción de validaciones implementadas

## Validaciones implementadas en el formulario Registrar Devolución

En esta parte colocamos las validaciones que consideramos necesarias para asegurar la integridad y coherencia de los datos ingresados en el formulario de registro de devolución:

1. **ID:**
   * Validamos que el campo no sea editable por el usuario, ya que corresponde a un identificador único generado automáticamente por el sistema.
   * Nos aseguramos de que siempre esté presente antes de guardar la devolución.
2. **ID Préstamo:**
   * Verificamos que el campo no esté vacío, ya que es obligatorio para asociar la devolución a un préstamo existente.
   * Validamos que solo acepte valores numéricos, evitando caracteres no permitidos.
   * Comprobamos que el ID de préstamo exista en la base de datos antes de registrar la devolución.
3. **Nombre:**
   * Validamos que el campo no esté vacío, ya que es necesario identificar al usuario o responsable de la devolución.
   * Limitamos la longitud máxima del nombre para evitar errores de almacenamiento y presentación.
   * No permitimos el ingreso de caracteres especiales no válidos.
4. **Fecha Devolución:**
   * Nos aseguramos de que la fecha no esté vacía, pues es un dato obligatorio.
   * Validamos que la fecha ingresada no sea anterior a la fecha del préstamo ni posterior a la fecha actual.
   * Comprobamos que el formato de fecha DD-MM-AAAA sea válido.
5. **Observaciones:**
   * Permitimos que el campo sea opcional, pero en caso de completarse, validamos que no exceda un número máximo de caracteres.
   * No aceptamos caracteres especiales que puedan afectar la integridad de la base de datos (por ejemplo, comillas simples o dobles sin escape).
6. **Botón Guardar:**
   * Solo permitimos el guardado si todos los campos obligatorios cumplen con las validaciones anteriores.
   * Mostramos mensajes de error claros y específicos en caso de que alguna validación no se cumpla.
7. **Botón Cerrar:**
   * Permitimos cerrar el formulario en cualquier momento, descartando los datos no guardados y evitando la pérdida accidental de información ya registrada.

## Validaciones implementadas en el formulario Registro de libros

En esta parte colocamos las validaciones necesarias para garantizar la calidad y consistencia de los datos ingresados en el formulario de registro de libros:

1. **Id:**
   * Validamos que este campo no sea editable por el usuario, ya que corresponde a un identificador único generado automáticamente por el sistema.
   * Nos aseguramos de que el identificador se asigne correctamente al momento de insertar un nuevo libro.
2. **Título:**
   * Verificamos que el campo no esté vacío, ya que el título es obligatorio para identificar el libro.
   * Limitamos la longitud máxima del título para evitar problemas de almacenamiento y presentación.
   * No permitimos el ingreso de caracteres especiales no válidos que puedan afectar la integridad de los datos.
3. **Id\_categoria:**
   * Validamos que el campo no esté vacío, asegurando que cada libro esté correctamente clasificado.
   * Comprobamos que solo acepte valores numéricos y que el identificador de categoría exista en la base de datos.
   * No permitimos el ingreso de caracteres alfabéticos o símbolos.
4. **Año\_Publicacion:**
   * Nos aseguramos de que el campo no esté vacío, ya que el año de publicación es un dato relevante.
   * Validamos que solo acepte valores numéricos y que el año esté dentro de un rango lógico (no mayor al año actual ni menor a un año razonable para publicaciones).
   * Comprobamos que el formato sea correcto DD-MM-AAAA.
5. **ISBN:**
   * Verificamos que el campo no esté vacío, ya que el ISBN es obligatorio para la identificación internacional del libro.
   * Validamos que el ISBN tenga el formato correcto (por ejemplo, 10 o 13 dígitos, sin letras ni símbolos).
   * Comprobamos que el ISBN no esté duplicado en la base de datos para evitar registros repetidos.
6. **Editorial:**
   * Validamos que el campo no esté vacío, ya que la editorial es un dato requerido.
   * Limitamos la longitud máxima del nombre de la editorial.
   * No permitimos caracteres especiales no válidos.
7. **Botón Insertar:**
   * Solo permitimos la inserción del libro si todos los campos obligatorios cumplen con las validaciones establecidas.
   * Mostramos mensajes de error específicos en caso de que alguna validación no se cumpla, facilitando la corrección por parte del usuario.
8. **Botón Salir:**
   * Permitimos cerrar el formulario en cualquier momento, descartando los datos no guardados y asegurando que no se inserten registros incompletos o incorrectos.

## Validaciones implementadas en el formulario Registro de préstamo

En esta parte colocamos las validaciones necesarias para asegurar la correcta captura y coherencia de los datos en el formulario de registro de préstamo:

1. **Id:**
   * Validamos que este campo no sea editable por el usuario, ya que corresponde a un identificador único generado automáticamente por el sistema.
   * Nos aseguramos de que el identificador se asigne correctamente al momento de insertar un nuevo préstamo.
2. **Id\_Ejemplar:**
   * Verificamos que el usuario seleccione un ejemplar válido de la lista desplegable.
   * Nos aseguramos de que el ejemplar seleccionado esté disponible para préstamo (es decir, que no esté actualmente prestado o reservado).
   * Validamos que el campo no quede vacío.
3. **Id\_Usuario:**
   * Validamos que el usuario seleccione un usuario válido de la lista desplegable.
   * Comprobamos que el usuario seleccionado exista en la base de datos y esté habilitado para realizar préstamos.
   * Nos aseguramos de que el campo no quede vacío.
4. **Id\_Empleado:**
   * Verificamos que se seleccione un empleado válido de la lista desplegable.
   * Nos aseguramos de que el empleado exista en la base de datos y esté habilitado para registrar préstamos.
   * Validamos que el campo no esté vacío.
5. **Fecha Préstamo:**
   * Validamos que la fecha de préstamo no esté vacía, ya que es un dato obligatorio.
   * Comprobamos que la fecha ingresada tenga el formato correcto DD-MM-AAAA.
   * Nos aseguramos de que la fecha de préstamo no sea posterior a la fecha actual.
6. **Fecha Vencimiento:**
   * Verificamos que la fecha de vencimiento no esté vacía.
   * Validamos que la fecha de vencimiento sea posterior a la fecha de préstamo.
   * Comprobamos que el formato de la fecha sea correcto DD-MM-AAAA.
7. **Fecha Devolución:**
   * Permitimos que este campo quede vacío al momento de registrar un nuevo préstamo, ya que la devolución puede no haberse realizado aún.
   * Si se ingresa una fecha de devolución, validamos que no sea anterior a la fecha de préstamo ni posterior a la fecha actual.
   * Comprobamos el formato correcto de la fecha DD-MM-AAAA.
8. **Botón Insertar:**
   * Solo permitimos la inserción del préstamo si todos los campos obligatorios cumplen con las validaciones establecidas.
   * Mostramos mensajes de error claros y específicos en caso de que alguna validación no se cumpla, facilitando la corrección por parte del usuario.
9. **Botón Cerrar:**
   * Permitimos cerrar el formulario en cualquier momento, descartando los datos no guardados y asegurando que no se inserten registros incompletos o incorrectos.

## Validaciones implementadas en el formulario Registro de usuario

En esta parte colocamos las validaciones necesarias para garantizar la integridad y veracidad de los datos ingresados en el formulario de registro de usuario:

1. **Id:**
   * Validamos que este campo no sea editable por el usuario, ya que corresponde a un identificador único generado automáticamente por el sistema.
   * Nos aseguramos de que el identificador se asigne correctamente al momento de insertar un nuevo usuario.
2. **Nombre:**
   * Verificamos que el campo no esté vacío, ya que el nombre es obligatorio para la identificación del usuario.
   * Limitamos la longitud máxima del nombre para evitar problemas de almacenamiento y presentación.
   * No permitimos el ingreso de caracteres especiales no válidos o números.
3. **Apellido:**
   * Validamos que el campo no esté vacío, ya que el apellido es obligatorio para la correcta identificación del usuario.
   * Limitamos la longitud máxima del apellido.
   * No permitimos caracteres especiales no válidos ni números.
4. **Tipo:**
   * Nos aseguramos de que el usuario seleccione un tipo válido de la lista desplegable (por ejemplo: docente, estudiante, administrativo, etc.).
   * Validamos que el campo no quede vacío y que el valor seleccionado corresponda a una categoría existente en el sistema.
5. **Email:**
   * Verificamos que el campo no esté vacío, ya que el correo electrónico es obligatorio para la comunicación y gestión de usuarios.
   * Validamos que el formato del email sea correcto [usuario@dominio.com](mailto:usuario@dominio.com).
   * Comprobamos que el email no esté duplicado en la base de datos para evitar registros repetidos.
   * Limitamos la longitud máxima para evitar problemas de almacenamiento.
6. **Botón Insertar:**
   * Solo permitimos la inserción del usuario si todos los campos obligatorios cumplen con las validaciones establecidas.
   * Mostramos mensajes de error claros y específicos en caso de que alguna validación no se cumpla, facilitando la corrección por parte del usuario.
7. **Botón Salir:**
   * Permitimos cerrar el formulario en cualquier momento, descartando los datos no guardados y asegurando que no se inserten registros incompletos o incorrectos.

# Uso de las funciones CRUD

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Sitio web

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

# Implementación de la conexión DAO + JDBC

Para establecer la conexión entre nuestro proyecto de biblioteca y la base de datos, implementamos una clase llamada ConnectionDB

En esta clase centralizamos todos los parámetros necesarios para la conexión, como la URL de la base de datos, el usuario y la contraseña. Por ejemplo, configuramos la conexión para que utilice la base de datos llamada “biblioteca”, el usuario “alumno” y la contraseña correspondiente a nuestro entorno de MySQL.

Para optimizar y gestionar las conexiones, utilizamos la librería HikariCP, que permite manejar el pool de conexiones de manera eficiente y segura. Gracias a esto, el sistema puede abrir y cerrar conexiones automáticamente según la demanda, evitando problemas de rendimiento y facilitando la escalabilidad de la aplicación.

En cuanto al acceso a los datos, aplicamos el patrón DAO (Data Access Object). Cada vez que la aplicación necesita realizar una operación sobre la base de datos (como insertar, consultar, actualizar o eliminar registros), lo hace a través de clases DAO específicas para cada entidad. Por ejemplo, el DAO de libros utiliza la conexión proporcionada por ConnectionDB para ejecutar las operaciones necesarias en la base.

Esta estructura permite mantener el código organizado y modular. La interfaz gráfica se encarga únicamente de la interacción con el usuario, mientras que las clases DAO gestionan toda la lógica relacionada con la base de datos. De esta manera, si en algún momento es necesario modificar la estructura de la base de datos o cambiar detalles de la conexión, solo es necesario actualizar el DAO o la clase de conexión, sin afectar el resto del sistema.

En resumen, la integración de la clase de conexión junto con el patrón DAO nos permitió separar de forma clara la lógica de acceso a datos del resto de la aplicación, logrando así un sistema más limpio, mantenible y fácil de escalar.