NOTA IMPORTANTE

- El proyecto se da por finalizado, seguirá recibiendo actualizaciones.
- El proyecto se encuentra en una versión estable, por lo que no se realizarán cambios significativos
- El proyecto con interfaz gráfica se iniciará en otro repositorio, tendrá las mismas funcionalidades y se conectará de la misma forma a phpMyAdmin

PLANEACIÓN

- Conectar el proyecto a SpringBoot ✗
- Conectar el proyecto a una base de datos ✓
- Crear interfaz gráfica ⚠️
- Crear un diagrama de clases ✓
- Cambiar los arrays por listas ✓
- Crear base de datos simulada con archivo .txt ✓
- Crear una clase para el préstamo de un libro a un usuario ✓

Tabla de Contenido

DESCRIPCIÓN ACTUAL

Este repositorio contiene un software de gestión de biblioteca que se conecta a una base de datos MySQL a través de JDBC para gestionar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de libros. También incluye persistencia en archivos `.txt` como respaldo.

El proyecto utiliza **XAMPP** para gestionar la base de datos MySQL con **phpMyAdmin** y el conector JDBC para la interacción con la base de datos desde Java.

MODO DE USO

Requisitos

Para ejecutar este proyecto, es necesario:

1. **Conector JDBC**:
   Descargue e instale el conector JDBC desde la página oficial de MySQL:
   MySQL Connector/J

2. **Base de datos MySQL**:
   - Crear una base de datos con el nombre `biblioteca`.
   - Dentro de esta base de datos, cree las tablas necesarias como `libros`.

3. **Entorno de desarrollo**:
   - Un IDE que soporte Java (Eclipse, IntelliJ IDEA, NetBeans, etc.).
   - Incluir el archivo JAR del conector JDBC en su proyecto para la conexión a la base de datos.

Clases Principales

En su IDE, deberá crear las siguientes clases:

- `Biblioteca.java`: Clase principal del proyecto que gestiona las operaciones de la biblioteca.
- `ConexionDB.java`: Clase responsable de la conexión con la base de datos MySQL y la ejecución de las consultas.
- `Libro.java`: Clase que representa los objetos de tipo libro en el sistema.
- `GestorUsuarios.java`: Clase responsable de registrar usuarios.
- `Prestamo.java`: Clase que representa los objetos de tipo préstamo de libros.
- `Usuario.java`: Clase que representa a los usuarios.

Detalles de Clases

ConexionDB.java

Clase responsable de la conexión con la base de datos MySQL y la ejecución de las consultas. Incluye métodos para manejar registros de libros y préstamos

Biblioteca.java

Clase principal del proyecto que gestiona las operaciones de la biblioteca.

GestorUsuarios.java

Maneja la lógica para la creación de usuarios en la base de datos.

Prestamo.java

Clase que representa los objetos de tipo préstamo de libros. Ahora incluye seguimiento de fechas de devolución y estado del préstamo. La clase `Prestamo` ha sido ampliada para incluir:

- Atributo `fechaDevolucion` para registrar cuándo se devuelve un libro.
- Atributo `devuelto` para indicar si el libro ha sido devuelto.
- Métodos adicionales para gestionar estos nuevos atributos (`getFechaDevolucion`, `setFechaDevolucion`, `isDevuelto`).

Usuario.java

Clase responsable de registrar usuarios.

Libro.java

Clase que representa los objetos de tipo libro en el sistema.

Instalación

1. **Instalar MySQL y XAMPP**:
    - Instale XAMPP y active **MySQL** y **Apache**.
2. **Configurar la base de datos**:
    - Cree la base de datos `biblioteca` y las tablas necesarias como `libros`, `prestamos`, `usuarios`.
3. **Configuración del proyecto**:
    - Asegúrese de que el conector JDBC esté correctamente configurado en su IDE.
    - `ConexionDB.java` manejará la conexión a la base de datos.

Funcionalidades del programa

- **Gestión de libros**: Agregar, editar y eliminar libros en la base de datos.
- **Persistencia**: Genera un archivo `.txt` con los libros registrados.
- **Conexión a base de datos**: Mediante JDBC.
- **Creación de usuarios**: Permite crear usuarios y administradores.
- **Gestión de préstamos y devoluciones**: Ahora, los préstamos incluyen seguimiento de las fechas de devolución y estado de los préstamos.

Limitaciones

- Limitaciones en la lógica de ingreso de usuarios y manejo de la base de datos.
- Más funcionalidades pueden ser añadidas en futuras versiones.

Soporte

- Para dudas o problemas, puede ponerse en contacto por cualquier medio disponible.
- Reportar Error
- PQRS
- La carpeta "GUIA DE IMAGENES" incluye imágenes de ayuda para la instalación y configuración.

ERRORES ANTERIORES SOLUCIONADOS

- Error con el `Scanner` (`NoSuchElementException`) ✓
- Error con arrays (`InputMismatchException`) ✓
- Error en el guardado de libros ✓
- Error cuando se muestra un libro en la terminal ✓
- Error con la carga de los archivos `.txt` ✓
- Los libros editados no se reflejaban en el archivo `.txt` ✓
- Error al cargar los libros generando excepciones ✓
- Error al editar la fecha de publicación de un libro (`NumberFormatException`) ✓
- Problema con el guardado persistente de libros ✓
- Solucionado error en las consultas de la tabla "devoluciones" ✓

ERRORES CONOCIDOS

- Se ha identificado un error (no fatal) donde se muestra un mensaje de que los registros no fueron actualizados en la base de datos, cuando en realidad si se actualizaron, se ha identificado como error lógico y será solucionado en la siguiente actualización

CAMBIOS

08/08/2024 - 3:00 PM

- Se realizaron cambios en la estructura de las funciones.
- Se asignaron nombres más claros y fueron enviadas a las clases correspondientes.

08/08/2024 - 10:50 PM

- Guardado de libros en un archivo `.txt` como base de datos temporal.

09/08/2024 - 6:00 PM

- Cambios importantes en la estructura de las funciones.
- Añadido el guardado de libros mediante archivos `.txt`.

27/08/2024 - 1:33 PM

- Cambios en las funciones `cargarLibros`, `guardarLibros` y `aTexto` (de la clase `Libro`).
- Cambio de arrays a listas.

04/09/2024 - 5:30 PM

- Cambio de `System.out` por `LOGGERS` para mayor organización del código.

07/09/2024 - 10:30 PM

- Añadida una nueva clase y una API en PHP para la conexión a la base de datos local en `phpMyAdmin`.
- Funciona el método de agregar en la base de datos.

10/09/2024 - 10:37 AM

- Implementados métodos básicos del CRUD en la base de datos.
- Eliminado PHP debido a dificultades con su conexión.
- `ConexionDB.java` ahora realiza toda la conexión.
- Instaladas dependencias de MySQL Connector/J.

11/09/2024 - 10:45 AM

- Arreglado el método para editar libros en la base de datos.
  05:47 PM
- Añadido el login de usuarios con su respectiva conexión a la base de datos.

14/09/2024 - 11:03 PM

- Implementado el préstamo de libros.
- Nueva tabla `libros` en la base de datos.
- Actualizada la documentación con Javadoc.

14/09/2024 - 5:34 PM

- Actualizada la documentación.
- Mejorada la guía de imágenes.

15/09/2024 - 9:21 PM

- Implementado sistema de préstamos (beta).
- Actualizada toda la documentación.
- Mejorado el login de usuarios.
- Añadido sistema de búsqueda de libros.

18/09/2024 - 7:51 PM

- Implementado sistema de préstamos (beta 2).
- eliminada tabla "devoluciones"
- Implementado sistema de busqueda para usuarios con libros prestados

19/09/2024

- Documentación al día -**Prestamo.java:** Añadidos atributos para la gestión de fechas de devolución y estado de préstamos. Mejorado el seguimiento y la gestión de los préstamos de libros.

-**ConexionBD.java:** Mejorada la conexión y gestión de la base de datos para incluir métodos adicionales relacionados con préstamos y devoluciones.

Diagrama de Clases

```
classDiagram
    class Libro {
        -int id
        -String titulo
        -String autor
        -int fechaPublicacion
        -int numPaginas
        -boolean disponible
        -String isbn
        -String descripcion
        +Libro(int id, String titulo, String autor, int fechaPublicacion, int
numPaginas, boolean disponible, String isbn, String descripcion)
        +getTitulo() String
        +setTitulo(String titulo) void
        +getAutor() String
        +setAutor(String autor) void
        +getfechaPublicacion() int
        +setfechaPublicacion(int fechaPublicacion) void
        +getNumPaginas() int
        +setNumPaginas(int numPaginas) void
        +isDisponible() boolean
        +setDisponible(boolean disponible) void
        +getIsbn() String
        +setIsbn(String isbn) void
        +getDescripcion() String
        +setDescripcion(String descripcion) void
        +getId() int
        +setId(int id) void
        +aTexto() String
        +aLibro(String texto) Libro
        +editarLibro(Scanner scanner) void
        +cambiarDisponibilidad() void
        +toString() String
    }

    class Biblioteca {
        -LinkedList~Libro~ biblioteca
        +mostrarLibros(LinkedList~Libro~ biblioteca) void
        +posicionLibro(LinkedList~Libro~ biblioteca) void
        +eliminarLibro(LinkedList~Libro~ biblioteca, Scanner teclado) void
        +guardarLibros(LinkedList~Libro~ biblioteca) void
        +cargarLibros(LinkedList~Libro~ biblioteca) void
        +agregarLibro(LinkedList~Libro~ biblioteca, Scanner teclado) void
        +editarLibroPorId(LinkedList~Libro~ biblioteca, Scanner teclado) void
        +editarLibro(LinkedList~Libro~ biblioteca, Scanner teclado) void
        +cambiarEstado(LinkedList~Libro~ biblioteca, Scanner teclado) void
        +main(String[] args) void
    }
```

```
    class ConexionBD {
        -String URL
        -String USER
        -String PASSWORD
        -getConnection() Connection
        +crearLibro(Libro libro) void
        +leerLibro(int id) Libro
        +actualizarLibro(int id, Libro libro) void
        +eliminarLibro(int id) void
        +obtenerSiguienteId() int
    }

    class Usuario {
        -int id
        -String nombreUsuario
        -String contrasena
        -String email
        -boolean esAdministrador
        +Usuario(int id, String nombreUsuario, String contrasena, String email,
boolean esAdministrador)
        +getId() int
        +setId(int id) void
        +getNombreUsuario() String
        +setNombreUsuario(String nombreUsuario) void
        +getContrasena() String
        +setContrasena(String contrasena) void
        +getEmail() String
        +setEmail(String email) void
        +esAdministrador() boolean
        +setEsAdministrador(boolean esAdministrador) void
        +toString() String
    }

    class GestorUsuarios {
        -Connection conexion
        +GestorUsuarios(Connection conexion)
        +registrarUsuario(Usuario usuario) boolean
        +autenticarUsuario(String nombreUsuario, String contrasena) Usuario
    }

    class Prestamo {
        -int id
        -String nombreUsuario
        -String documento
        -int idLibro
        -String isbnLibro
        -String tituloLibro
        -String autorLibro
        -Date fechaPrestamo
        +Prestamo(int id, String nombreUsuario, String documento, int idLibro,
String isbnLibro, String tituloLibro, String autorLibro, Date fechaPrestamo)
        +getId() int
        +setId(int id) void
        +getNombreUsuario() String
```

```
        +setNombreUsuario(String nombreUsuario) void
        +getDocumento() String
        +setDocumento(String documento) void
        +getIdLibro() int
        +setIdLibro(int idLibro) void
        +getIsbnLibro() String
        +setIsbnLibro(String isbnLibro) void
        +getTituloLibro() String
        +setTituloLibro(String tituloLibro) void
        +getAutorLibro() String
        +setAutorLibro(String autorLibro) void
        +getFechaPrestamo() Date
        +setFechaPrestamo(Date fechaPrestamo) void
        +toString() String
    }

    Biblioteca -- Libro : manages
    Biblioteca -- ConexionBD : uses
    ConexionBD -- Libro : manages
    GestorUsuarios -- Usuario : manages
    GestorUsuarios -- ConexionBD : uses
    Prestamo -- Libro : references
    Prestamo -- Usuario : references
```