Fausto Fili

Proyecto final - Java

Noviembre de 2024

CRUD Usuarios

Cómo Ejecutar la Aplicación

Requisitos Previos:

- Asegúrate de tener instalada la última versión de Java JRE o JDK en tu computadora. Podes descargarla desde Oracle Java Downloads.
- Tener configurada y funcionando la base de datos MySQL.

Configuración de la Base de Datos:

- Importa el archivo SQL proporcionado para crear la base de datos y la tabla.
- Asegúrate de que la base de datos se llame sistema_crud y esté ejecutándose en el servidor localhost en el puerto 3306.

Credenciales predeterminadas para MySQL:

- Usuario: root
- Contraseña: 1234
- Si necesitas modificar las credenciales o el nombre de la base de datos, actualiza la clase DatabaseConnection en el archivo fuente antes de compilar.

Ejecutar el CRUD_Proyecto_Final.jar:

Navega a la carpeta **dist** de tu proyecto donde se encuentra el archivo ejecutable .jar.

Ejecutar desde NetBeans (opcional):

- Abre el proyecto en NetBeans.
- Haz clic en el botón de ejecución o presiona F6.

Descripción de las Funcionalidades Implementadas

Esta aplicación es un sistema de gestión de registros que incluye operaciones básicas (CRUD: Crear, Leer, Actualizar, Eliminar) y búsquedas. A continuación, se describen las funcionalidades y clases principales:

Gestión de Registros:

- Permite agregar, actualizar y eliminar registros almacenados en una base de datos MySQL.

Búsqueda Avanzada:

- Busca registros por diferentes campos: nombre, apellido, correo electrónico o teléfono.

Validaciones:

- Los campos obligatorios no pueden estar vacíos.
- El correo electrónico debe ser válido (contener `@`).
- El número de teléfono solo puede contener dígitos.

Interfaz Gráfica (Swing):

- La aplicación cuenta con una interfaz intuitiva para gestionar los registros, construida con Java Swing.

Clases y Sus Funciones

1. Clase `RegistroApp` (Ventana Principal)

- **Propósito:**
  - Es la ventana principal de la aplicación.
  - Permite visualizar todos los registros almacenados y acceder a las funciones de insertar, actualizar, eliminar y buscar registros.
- **Características:**
  - Carga todos los registros desde la base de datos al iniciar.
  - Proporciona botones para realizar las acciones CRUD y abrir ventanas secundarias (modales).
  - Muestra los registros en una tabla no editable.

2. Clase `InsertarVentana`

- **Propósito:**
  - Ventana modal que permite al usuario agregar un nuevo registro.
- **Características:**
  - Incluye campos para ingresar nombre, apellido, email y teléfono.
  - Realiza validaciones antes de insertar:
    - Todos los campos son obligatorios.
    - El correo debe ser válido.
    - El teléfono solo permite números.
  - Inserta el registro en la base de datos utilizando el método `insertarRegistro` de la clase `RegistroService`.

3. Clase `ActualizarVentana`

- **Propósito:**
  - Ventana modal que permite al usuario actualizar un registro existente.
- **Características:**
  - Carga los datos actuales del registro seleccionado en los campos de texto.
  - Permite editar los datos y validarlos antes de guardar:
    - Todos los campos son obligatorios.
    - El correo debe contener un `@`.
    - El teléfono solo permite números.
  - Actualiza el registro en la base de datos utilizando el método `actualizarRegistro` de `RegistroService`.

4. Clase `BusquedaVentana`

- **Propósito:**
  - Ventana modal para buscar registros por nombre, apellido, email o teléfono.
- **Características:**
  - Permite al usuario ingresar un criterio de búsqueda y muestra los resultados en una tabla.
  - Realiza búsquedas parciales utilizando `LIKE` en los campos relevantes de la base de datos.
  - Si no se encuentra ningún resultado, muestra una tabla vacía.

5. Clase `RegistroService`

- **Propósito:**
  - Actúa como capa de servicio para interactuar con la base de datos.
- **Métodos:**
  - `obtenerRegistros`: Recupera todos los registros de la tabla.
  - `insertarRegistro`: Inserta un nuevo registro.
  - `actualizarRegistro`: Actualiza un registro existente por su ID.
  - `eliminarRegistro`: Elimina un registro por su ID.
  - `buscarRegistros`: Realiza búsquedas por nombre, apellido, email o teléfono.

6. Clase `Registro`

- **Propósito:**
  - Representa un registro individual como objeto en la aplicación.
- **Características:**
  - Contiene atributos como `id`, `nombre`, `apellido`, `email` y `telefono`.
  - Proporciona métodos `getters` y `setters` para acceder y modificar los datos.

7. Clase `DatabaseConnection`

- **Propósito:**
  - Gestiona la conexión con la base de datos utilizando el patrón Singleton.
- **Características:**
  - Asegura que solo haya una conexión activa a la base de datos.
  - Contiene métodos para establecer y recuperar la conexión.