BUAP

GESTOR DE NOTAS ESTUDIANTILES

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

PROGRAMACIÓN DISTRIBUIDA

EMMANUEL JIMENEZ PÉREZ
PAVEL TAMANIS RODRIGUEZ
ERIK FLORES PALACIOS

Tabla de Contenidos

EL OBJETIVO

Crear un **software gestor de notas estudiantiles** es esencial por diversas razones, especialmente en entornos educativos que buscan eficiencia, organización y precisión en la gestión académica. A continuación, se presentan los motivos principales:

CENTRALIZACIÓN DE LA INFORMACIÓN

- Permite almacenar, actualizar y consultar las calificaciones de los estudiantes en una única plataforma.

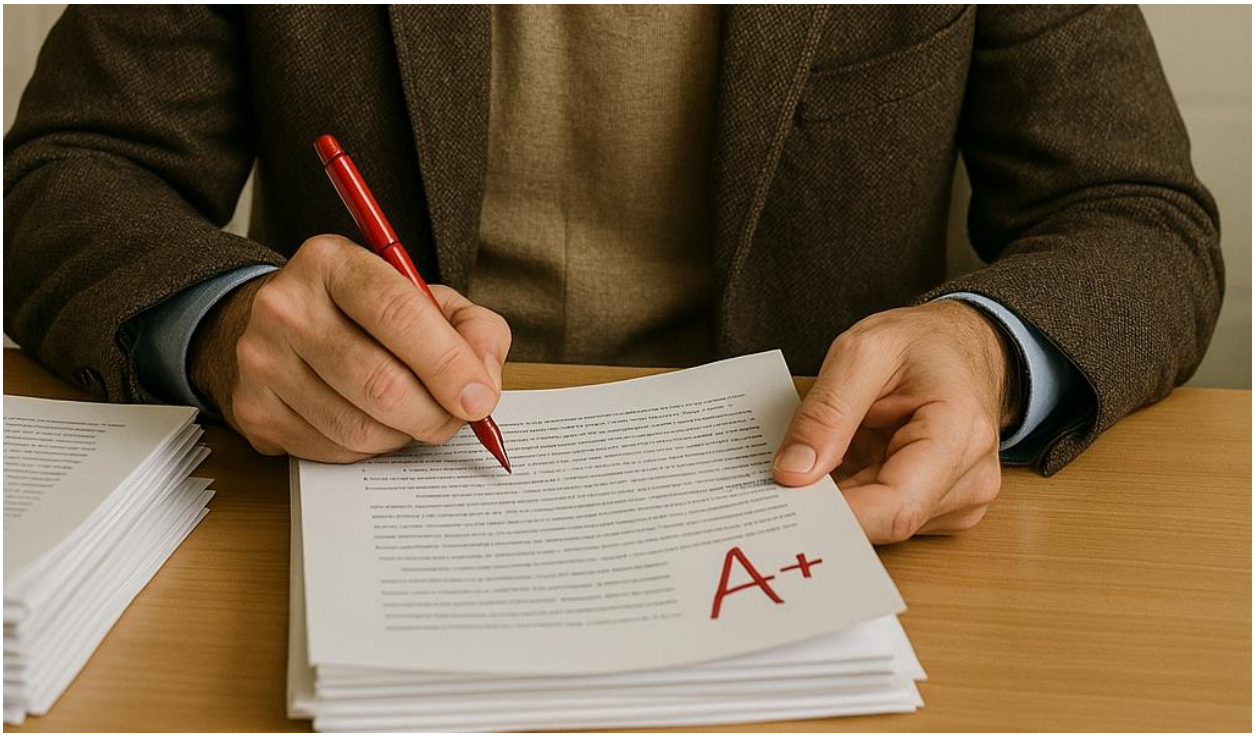
- Facilita el acceso a los datos por parte de docentes, estudiantes y administrativos, evitando duplicidad de registros.

AUTOMATIZACIÓN DE PROCESOS

- Elimina la necesidad de cálculos manuales al generar promedios, históricos académicos y reportes.
- Facilita el ingreso de calificaciones, reduciendo errores humanos.

GENERACIÓN DE REPORTES Y ANÁLISIS

- Proporciona estadísticas académicas, como rendimiento por asignatura o comparativas entre períodos.
- Ayuda a identificar patrones de desempeño y áreas que requieren mejoras.



ACCESO REMOTO Y DISTRIBUIDO

- Utilizando tecnologías como **RMI**, permite el acceso desde múltiples ubicaciones, ideal para instituciones con varias sedes.

- Los estudiantes pueden consultar sus notas en tiempo real desde cualquier dispositivo conectado.

SEGURIDAD Y PRIVACIDAD DE DATOS

- Garantiza el manejo seguro de la información académica, evitando accesos no autorizados.

- Implementa roles y permisos para que solo usuarios autorizados puedan realizar modificaciones.

HISTORIAL ACADÉMICO Y TRAZABILIDAD

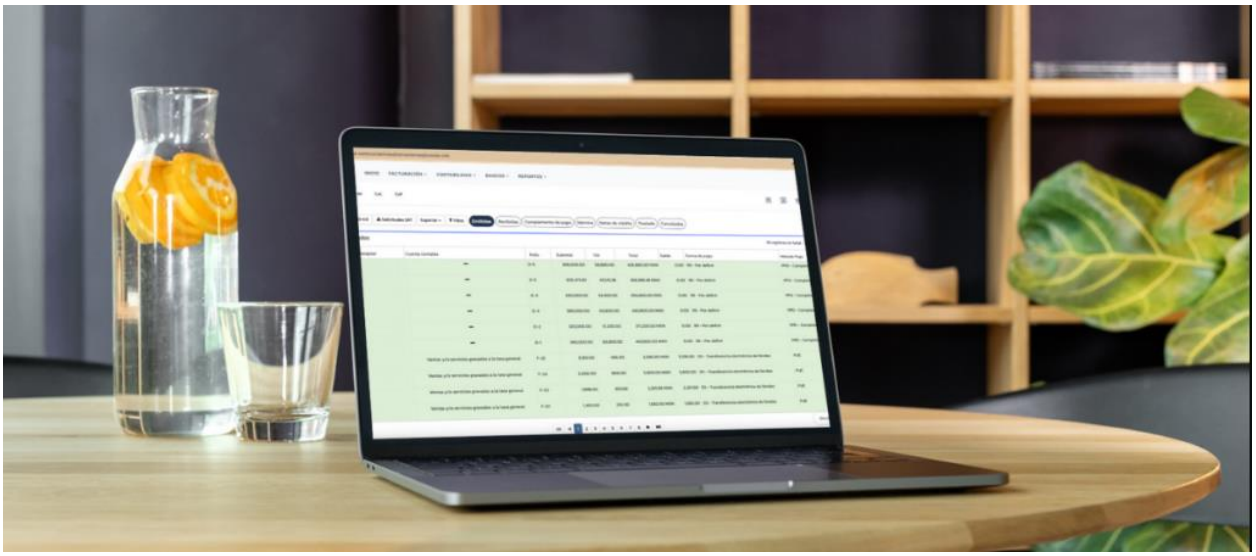
- Almacena de manera histórica las calificaciones, permitiendo a los estudiantes y docentes acceder a registros antiguos.

- Facilita la auditoría de cambios en las notas, identificando quién y cuándo realizó modificaciones.

AHORRO DE TIEMPO Y RECURSOS

- Reduce la carga administrativa, permitiendo que los docentes dediquen más tiempo a la enseñanza en lugar de la gestión de notas.

- Minimiza el uso de papel y recursos físicos al digitalizar la gestión académica.



ACTUALIZACIÓN Y ESCALABILIDAD

- El sistema puede ampliarse fácilmente para incluir nuevas funcionalidades o integrar módulos adicionales, como asistencia o gestión de tareas.

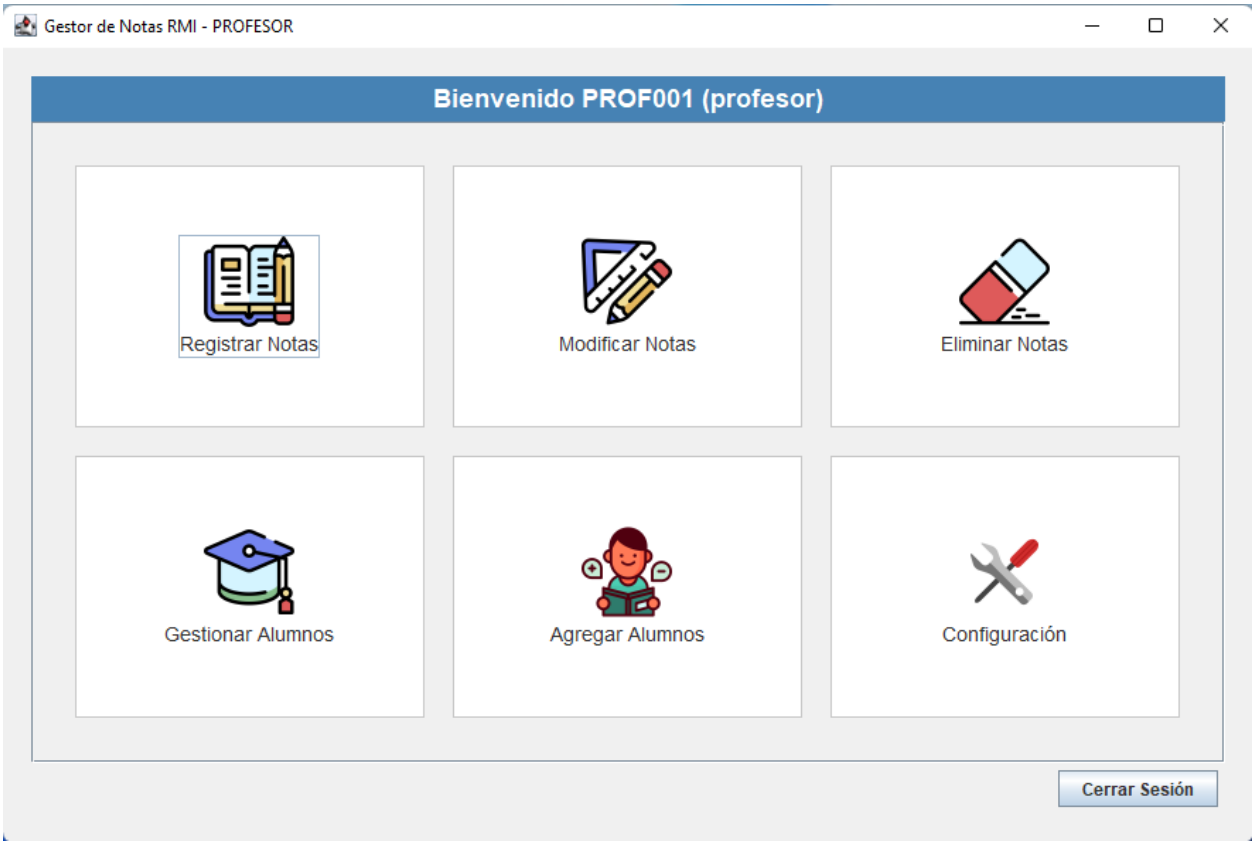
- Actualizaciones en tiempo real garantizan que todos los usuarios trabajen con información actualizada.

MANEJO DE NUESTRO SOFTWARE

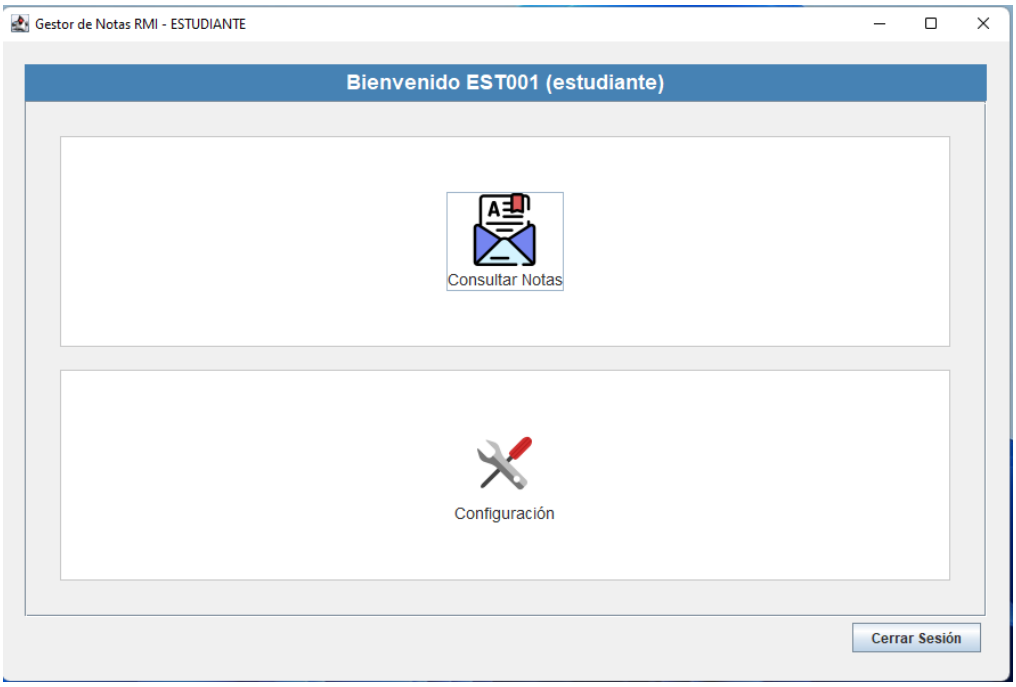
Usando principalmente: **RMI**, **Mysql** (Xampp), **java** y encriptación **Hash**, conseguimos la siguiente interfaz donde podemos iniciar sesión ***como profesor***:

Donde podremos:

- Registrar calificaciones
- Modificar calificaciones
- Eliminar calificaciones
- Administrar alumnos inscritos
- Agregar nuevos alumnos.
- Actualizar contraseña de profesor



O ***como alumno*** donde será posible consultar nuestras calificaciones, así como actualizar nuestra contraseña:



RMI, MYSQL Y HASH

- **Protocolo RMI (Remote Method Invocation)**: Permite a los programas escritos en Java invocar métodos de objetos que están en diferentes máquinas de forma remota, facilitando la comunicación entre aplicaciones distribuidas.

- **MySQL**: Sistema de gestión de bases de datos relacional de código abierto que utiliza SQL (Structured Query Language) para gestionar y manipular datos.

- **Hash**: Función que convierte una entrada (como datos o contraseñas) en una cadena de longitud fija, generalmente utilizada para almacenar contraseñas de forma segura o para acceder rápidamente a datos en estructuras como tablas hash.

LA BASE DE DATOS

La base de datos **"gestor_notas"** está diseñada para gestionar usuarios y calificaciones en un entorno académico.

Cuenta con la siguiente estructura:

1) **Usuarios:**
   - Almacena información de usuarios (profesores y estudiantes) con atributos como nombre, matrícula (única), contraseña y tipo (profesor o estudiante).
   - Se añadió un campo salt para mejorar la seguridad de las contraseñas.
2) **Estudiantes:**
   - Contiene información básica de los estudiantes, como nombre y matrícula (única).
3) **Notas:**
   - Registra las calificaciones de los estudiantes, especificando el estudiante (mediante un ID), la materia y la nota obtenida.
   - La columna id_estudiante está vinculada mediante una clave foránea a la tabla de estudiantes.

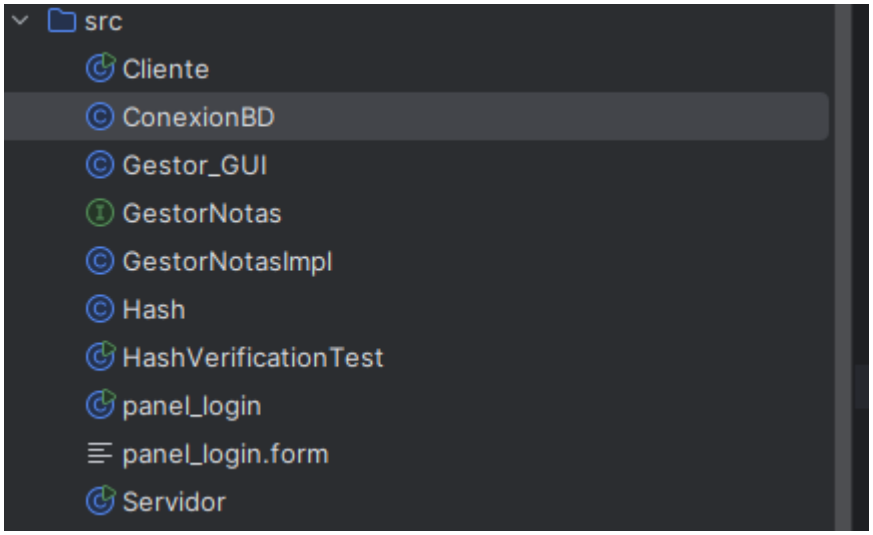
```
show tables' at line 1
MariaDB [gestor_notas]> show tables;
+------------------------+
| Tables_in_gestor_notas |
+------------------------+
| estudiantes            |
| notas                  |
| usuarios               |
+------------------------+
3 rows in set (0.001 sec)
```

LAS CLASES SERVIDOR Y CONEXIÓN

1) **Servidor:** Esta clase Java implementa un servidor RMI (Remote Method Invocation).
    - **Registro RMI:** Utiliza la clase LocateRegistry para crear un registro RMI en el puerto 1099.
    - **Instancia del objeto remoto:** Crea una instancia del objeto remoto GestorNotasImpl, que implementa la interfaz GestorNotas.
    - **Vinculación en el registro:** Usa el método rebind para asociar el nombre "GestorNotas" al objeto remoto, haciéndolo accesible para clientes RMI.
    - **Mensaje de confirmación:** Indica en la consola que el servidor está en funcionamiento en el puerto 1099.
    - **Manejo de excepciones:** Captura cualquier excepción durante el proceso y la imprime en la consola.

2) **Conexión:** Este código Java define una clase llamada ConexionBD que establece una conexión a una base de datos MySQL.

    - **Conexión JDBC:** Utiliza el controlador JDBC para conectarse a una base de datos MySQL.
    - **Parámetros de conexión:** Define la URL de la base de datos (gestor_notas en localhost y puerto 3306), el usuario (root) y la contraseña (vacía).
    - **Método de conexión:** El método estático conectar() intenta crear una conexión mediante DriverManager.getConnection().
    - **Manejo de excepciones:** Si ocurre un error, lo imprime en la consola y retorna null.



LA CLASE HASH

- Se implementa el algoritmo PBKDF2, para base de datos, deriva una clave a partir de una contraseña, la técnica que aplica es encriptación basada en iteraciones.

- Se implementa el algoritmo HMAC-SHA256, para verificar la integridad y autenticidad de un mensaje.

- Se implementa el algorimo SHA-256 genera una huella digital de 256 bits.

- Evitamos problemas como: rainbow tables, que es una base de datos que contiene hashes, que permite descrifrar contraseñas.

LA CLASE CLIENTE

Esta clase Java implementa un **cliente RMI (Remote Method Invocation)** que interactúa con el servidor para gestionar estudiantes y notas.

Funcionamiento del Cliente:

1. **Conexión al servidor RMI:**
   - ❖ Se conecta al registro RMI ubicado en el puerto 1099 en el host localhost.
   - ❖ Busca el objeto remoto llamado "GestorNotas".

2. **Autenticación de usuario:**
    - ❖ Solicita matrícula y contraseña al usuario.
    - ❖ Utiliza el método remoto login() para verificar las credenciales.
    - ❖ Si el inicio de sesión es exitoso, obtiene el tipo de usuario (estudiante o profesor).
3. **Funcionalidades según el tipo de usuario:**
    - ❖ **Estudiante:** Muestra sus notas mediante el método verNotas().
    - ❖ **Profesor:** Muestra un menú con opciones administrativas:
        1. Agregar estudiante.
        2. Ver notas de un estudiante.
        3. Modificar una nota (comentado en el código).
        4. Eliminar una materia de un estudiante.
        5. Modificar información de un alumno.
        6. Registrar una nueva nota.
        7. Salir del sistema.
    - ❖ Ejecuta la acción correspondiente según la opción seleccionada.
4. **Manejo de excepciones:**
    - ❖ Cualquier error se imprime en la consola.

LA CLASE LOGIN

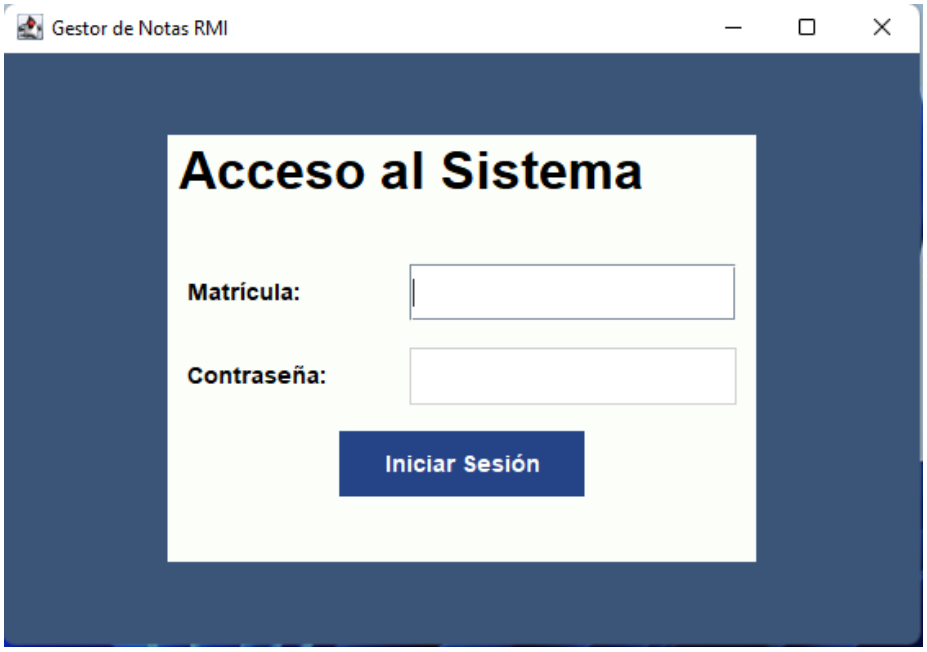
Esta clase Java define una interfaz gráfica de usuario (GUI) utilizando **Swing** para un sistema de gestión de notas con acceso mediante autenticación.

Componentes principales:

1. **Conexión RMI:**
    - Utiliza el registro RMI en localhost (puerto 1099) para obtener el objeto remoto GestorNotas.
    - Muestra un mensaje de error si no puede conectarse al servidor.
2. **Configuración de la Ventana:**
    - Título: **"Gestor de Notas RMI"**
    - Tamaño: 800x600 píxeles.
    - Fondo: Color verde claro (RGB: 223, 248, 235).
    - Panel principal con márgenes y fondo azul oscuro.
3. **Formulario de Inicio de Sesión:**
    - **Etiquetas:** Matrícula y Contraseña.
    - **Campos de texto:** JTextField para la matrícula y JPasswordField para la contraseña.

- **Botón:** "Iniciar Sesión" con animación de hover (cambio de color).
- **Estilo:** Fondo blanco para el formulario, colores contrastantes en etiquetas y botón.
- **Panel con título:** "Acceso al Sistema" con borde y estilo personalizados.

4. **Acciones del Botón de Login:**
   - Verifica si los campos están vacíos.
   - Realiza la autenticación usando el objeto remoto gestorNotas.login().
   - Muestra un mensaje de bienvenida con el tipo de usuario si el login es exitoso.
   - Si falla, muestra un mensaje de error.
   - Si es exitoso, abre la interfaz principal del gestor (Gestor_GUI) y cierra la ventana de login.
5. **Interactividad:**
   - El botón de inicio de sesión cambia de color al pasar el mouse.
   - La interfaz se inicializa en el método main() usando SwingUtilities.invokeLater() para asegurar la creación de la GUI en el hilo de eventos.



LA CLASE GESTOR NOTAS

Esta clase Java implementa el servicio remoto de gestión de notas utilizando **RMI (Remote Method Invocation)** y gestiona la información académica almacenada en una base de datos MySQL.

La clase GestorNotasImpl extiende UnicastRemoteObject e implementa la interfaz GestorNotas, proporcionando métodos para la gestión de estudiantes, notas y autenticación. Se comunica con la base de datos a través de la clase ConexionBD.

Componentes Principales:

1. **Conexión RMI:**
   - ✓ La clase hereda de UnicastRemoteObject, lo que permite que sus métodos se puedan invocar de forma remota.
   - ✓ Registra el servicio GestorNotas en el registro RMI.
2. **Materias Predefinidas:**
   - ✓ Una lista estática de materias escolares, utilizada para la validación de materias ingresadas.

Métodos Implementados:

1. **Autenticación (Login):**
   - ✓ Verifica si el usuario está registrado en la base de datos.
   - ✓ Utiliza contraseñas con hash y salt para mayor seguridad.
   - ✓ Admite compatibilidad con contraseñas antiguas sin hash durante la migración.
2. **Obtener Tipo de Usuario:**
   - ✓ Recupera el tipo de usuario (por ejemplo, **"estudiante"** o **"profesor"**) a partir de la matrícula.
3. **Agregar Estudiante:**
   - ✓ Valida los campos obligatorios (nombre y matrícula).
   - ✓ Genera una contraseña por defecto (usando hash y salt).
   - ✓ Verifica que la matrícula no esté registrada previamente.
   - ✓ Inserta el estudiante en las tablas **estudiantes** y **usuarios**.
4. **Ver Notas:**
   - ✓ Obtiene una lista de notas de un estudiante utilizando su matrícula.
   - ✓ Consulta la tabla **notas** de la base de datos.
5. **Modificar Nota:**
   - ✓ Modifica la calificación de un estudiante en una materia específica.
   - ✓ Verifica la existencia del estudiante y la nota antes de actualizar.

6. **Eliminar Materia:**
    - ✓ Elimina la calificación de una materia específica asociada a un estudiante.
    - ✓ Valida la existencia del estudiante y la materia antes de proceder.
7. **Registrar Nota:**
    - ✓ Agrega una nueva nota para un estudiante, validando previamente que la materia sea válida y que no exista previamente.
    - ✓ Lanza una excepción si la nota no está en el rango permitido (0.0 a 10.0).
8. **Modificar Alumno:**
    - ✓ Actualiza el nombre y la matrícula del estudiante en las tablas **estudiantes** y **usuarios**.
9. **Obtener Estudiantes con Notas:**
    - ✓ Devuelve una lista con todos los estudiantes que tienen notas registradas, ordenada por nombre y materia.
10. **Validación de Existencia de Matrícula:**
    - ✓ Comprueba si una matrícula ya está registrada en la base de datos en las tablas **estudiantes** o **usuarios**.
11. **Validación de Contraseñas:**
    - ✓ Verifica y cambia contraseñas de estudiantes y profesores.

Tratamiento de Errores:

- Manejo extensivo de excepciones, especialmente relacionadas con:
    - Problemas de conexión a la base de datos.
    - Fallas en la autenticación.
    - Problemas con la modificación o eliminación de registros.
- Uso de RemoteException para manejar errores durante la comunicación remota.
- Uso de transacciones para garantizar la consistencia de la base de datos al agregar estudiantes.

LA CLASE GESTOR GUI

La clase Gestor_GUI muestra una interfaz gráfica que permite a los usuarios (profesores y estudiantes) interactuar con el sistema de gestión de notas. La interfaz cambia según el **tipo de usuario** (profesor o estudiante) y ofrece opciones específicas para cada rol.

Componentes Principales:

1. **Inicialización del GUI:**
   - ✓ El constructor recibe el objeto remoto gestorNotas, la matrícula del usuario y su tipo.
   - ✓ Configura la ventana principal con título, tamaño fijo y centrado en pantalla.
   - ✓ Llama al método initUI() para construir la interfaz gráfica.

Estructura de la Interfaz:

1. **Panel Principal (mainPanel):**
   - ✓ Utiliza un diseño **BorderLayout**.
   - ✓ Incluye tres secciones: cabecera, área de botones y pie de página.
   - ✓ Fondo de color claro (gris claro).
2. **Cabecera (headerPanel):**
   Muestra un mensaje de bienvenida con la matrícula y el rol del usuario.
   Fondo azul (Color: 70, 130, 180) y texto en blanco.
   Fuente en **negrita** para resaltar el mensaje.

3. **Panel de Botones (buttonPanel):**
   - ✓ Organización en una **rejilla (GridLayout)** de 2 filas y 3 columnas.
   - ✓ Diferenciación de funciones según el tipo de usuario:
     - **Profesor:**
       - Registrar Notas
       - Modificar Notas
       - Eliminar Notas
       - Gestionar Alumnos
       - Agregar Alumnos
     - **Estudiante:**
       - Consultar Notas
       - Configuración
   - ✓ Cada botón muestra un ícono (si está disponible) y una etiqueta descriptiva.
   - ✓ Los botones se estilizan con un fondo blanco, borde sutil y espacio de relleno (padding).
4. **Pie de Página (footerPanel):**
   - ✓ Incluye el botón "Cerrar Sesión" alineado a la derecha.
   - ✓ El botón activa el método cerrarSesion() al ser presionado.

Creación de Botones (crearBtn):

- Recibe el panel de destino, el texto y la ruta del ícono.
- Carga el ícono desde los recursos si está disponible.
- Ajusta el tamaño de la imagen a **64x64 píxeles**.
- Añade el botón al panel después de configurar su apariencia y asignarle la acción correspondiente.

Manejo de Acciones:

- Los botones activan el método manejarAccion(texto), el cual ejecuta la acción adecuada según el botón presionado.
- La acción varía según el texto del botón, diferenciando entre funcionalidades para profesores y estudiantes.

CONCLUSIÓN

Desarrollar un software gestor de notas estudiantiles es fundamental para cualquier institución educativa que desee modernizar su gestión académica. No solo optimiza el tiempo y los recursos, sino que también contribuye a una mejor comunicación, transparencia y toma de decisiones basada en datos.