

Memoria del Proyecto: Sistema de Gestión de Incidencias Informáticas



3 de junio de 2025

ANDREEA DIACONU

Objetivo General

El objetivo de este proyecto es desarrollar una aplicación de consola en Java que permita a los técnicos de soporte de una empresa gestionar incidencias informáticas registradas por empleados. Se ha implementado la arquitectura Modelo-Vista-Controlador (MVC), conexión a base de datos con JDBC, y un sistema de autenticación para los técnicos.

**Estructura de la Aplicación**

1. Modelo (`modelo`): Define las clases que representan las entidades de la base de datos: `Empleado`, `Incidencia`, `Tecnico`.

2. DAO (`dao`): Se encarga del acceso a datos mediante JDBC: operaciones CRUD, búsquedas, cambios de estado, etc.

3. Controlador (`controlador`): Gestiona la lógica del negocio y coordinación entre la vista y la capa DAO.

4. Vista (`vista`): Interfaz de consola amigable con menús y submenús.

5. Utilidades (`util`): Funciones auxiliares como la conexión a la base de datos y validaciones.

**Base de Datos**

Se utiliza MySQL y las siguientes tablas:

• tecnicos: Campos: id, nombre\_usuario, password

• empleados: Campos: id, nombre, email

•incidencias: Campos: id, id\_empleado, descripcion, prioridad, estado, fecha\_creacion, fecha\_resolucion

**Autenticación**

El sistema solicita al técnico su nombre de usuario y contraseña al iniciar. La autenticación se valida contra la tabla tecnicos. Si no es válida, no se permite el acceso.

**Objetivo:**  
Garantizar que solo los usuarios autorizados (técnicos de soporte) puedan acceder al sistema de gestión de incidencias, protegiendo la información y evitando accesos no autorizados.

**Funcionamiento:**  
Al iniciar la aplicación, el sistema solicita al usuario que ingrese su **nombre de usuario** y **contraseña**. Estos datos se verifican comparándolos con los registros almacenados en la base de datos, específicamente en la tabla tecnicos.

**Detalles técnicos y flujo:**

1. **Entrada de credenciales:**  
   El sistema muestra un mensaje en consola pidiendo el ingreso de:
   * Nombre de usuario (ejemplo: admin)
   * Contraseña (ejemplo: admin123)
2. **Validación de datos:**  
   Antes de consultar la base de datos, se verifica que ambos campos no estén vacíos y cumplan con un formato básico (longitud mínima, caracteres permitidos, etc.).
3. **Consulta segura:**  
   Para validar el usuario, el sistema realiza una consulta SQL utilizando JDBC y PreparedStatement para evitar inyección SQL, por ejemplo:

SELECT \* FROM tecnicos WHERE nombre\_usuario = ? AND password = ?

Se sustituyen los signos ? por los datos ingresados por el usuario.

1. **Resultado de la consulta:**
   * Si la consulta devuelve un registro, significa que las credenciales son correctas y se permite el acceso al menú principal.
   * Si no se encuentra ningún registro que coincida, se muestra un mensaje de error como:  
     "Usuario o contraseña incorrectos. Por favor, inténtelo de nuevo."  
     y se solicita ingresar las credenciales nuevamente.
2. **Seguridad y mejoras futuras:**  
   Actualmente las contraseñas se almacenan en texto plano solo para simplificar el desarrollo y pruebas. En un sistema real, las contraseñas deben ser almacenadas cifradas con algoritmos seguros como **BCrypt** o **SHA-256 con sal** para proteger los datos en caso de brechas de seguridad.
3. **Ciclo de autenticación:**  
   El sistema puede permitir un número limitado de intentos para evitar ataques de fuerza bruta. Si se superan, podría cerrarse la aplicación o bloquear el acceso temporalmente.

**Funcionalidades Principales**

Una vez autenticado, el usuario puede navegar por los siguientes menús:

1. Gestionar Empleados (CRUD)

2. Gestionar Incidencias (submenú)

3. Ver Incidencias por Estado

4. Ver Historial de un Empleado

5. Ver Informes Generales

6. Cerrar Sesión

**Submenú de Incidencias**

• Registrar nueva incidencia (con validación de duplicados)

• Listar todas

• Cambiar estado (si se marca como “Cerrada”, se registra la fecha de resolución)

• Eliminar incidencia

• Volver al menú anterior

**Validaciones y Reglas de Negocio**

• No se permite registrar una incidencia cerrada previamente con la misma descripción.

• El cambio de estado solo permite: 'Abierta', 'En proceso', 'Cerrada'.

• Al cerrar una incidencia se registra automáticamente la fecha de resolución.

• Validaciones básicas como prioridad válida, email bien formado, y campos obligatorios.

**Informes Generales**

Menú de informes con consultas SQL agregadas que muestran:

1. Total de incidencias por empleado (abiertas y cerradas)

2. Incidencias abiertas agrupadas por prioridad

3. Tiempo promedio de resolución por empleado

4. Volver al menú principal

**Datos de Prueba**

Datos Iniciales - Inserts SQL

Para comenzar con el uso del sistema de gestión de incidencias informáticas, es necesario disponer de datos básicos insertados en la base de datos. A continuación se detallan los inserts realizados para los distintos actores del sistema:

1. Técnicos

Se insertan dos técnicos que podrán autenticarse en el sistema utilizando su nombre de usuario y contraseña. Las contraseñas deben estar almacenadas en texto plano solo a efectos de prueba, pero en un entorno real deben cifrarse.  
INSERT INTO tecnicos (nombre\_usuario, password) VALUES   
('admin', 'admin123'),  
('soporte', 'soporte456');

Propósito:  
Permitir el inicio de sesión en la aplicación desde el menú de autenticación. Solo los técnicos autenticados pueden acceder al sistema y gestionar incidencias.

2. Empleados

Se insertan tres empleados de prueba que representarán a usuarios que reportan incidencias.  
INSERT INTO empleados (nombre, email) VALUES   
('Juan Pérez', 'juan.perez@empresa.com'),  
('Ana Gómez', 'ana.gomez@empresa.com'),  
('Luis Martínez', 'luis.martinez@empresa.com');

Propósito:  
Simular empleados reales que pueden tener incidencias registradas en el sistema. Estas incidencias se asocian mediante el campo id\_empleado.

3. Incidencias

Se insertan cuatro incidencias con distintos estados, prioridades y fechas de resolución. La fecha de creación se asigna directamente y las incidencias cerradas también tienen fecha de resolución.  
INSERT INTO incidencias (id\_empleado, descripcion, prioridad, estado, fecha\_creacion, fecha\_resolucion) VALUES   
(1, 'No funciona el correo electrónico', 'Alta', 'Abierta', '2025-05-28', NULL),  
(2, 'La impresora no imprime', 'Media', 'En proceso', '2025-05-25', NULL),  
(1, 'Problemas con la conexión Wi-Fi', 'Alta', 'Cerrada', '2025-05-20', '2025-05-21'),  
(3, 'Actualización de software pendiente', 'Baja', 'Cerrada', '2025-05-18', '2025-05-22');📁 Estructura de Carpetas del Proyecto

src/  
├── controlador/  
│ └── EmpleadoControlador.java, IncidenciaControlador.java, TecnicoControlador.java  
├── dao/  
│ └── EmpleadoDAO.java, IncidenciaDAO.java, TecnicoDAO.java  
├── modelo/  
│ └── Empleado.java, Incidencia.java, Tecnico.java  
├── util/  
│ └── Conexion.java, Validaciones.java  
├── vista/  
│ └── MenuVista.java, Consola.java  
└── com.mycompany.examenfinal/  
 └── ExamenFinal.java (main)

**Herramientas Utilizadas**

Java 24

Java es el lenguaje de programación principal utilizado en este proyecto. La versión 24 aporta mejoras de rendimiento, seguridad y nuevas funcionalidades sintácticas que hacen el desarrollo más robusto y eficiente. Java ofrece independencia de plataforma (gracias a la JVM), lo que permite que la aplicación pueda ejecutarse en cualquier sistema operativo compatible.

Uso en el proyecto:  
Java ha sido empleado para toda la lógica del sistema, desde el modelo de datos hasta la interfaz por consola, aplicando el patrón de arquitectura MVC para mantener una estructura modular y fácil de mantener.

NetBeans 25

NetBeans es el entorno de desarrollo integrado (IDE) utilizado para codificar, compilar y ejecutar el proyecto. La versión 25 ofrece compatibilidad con Java 24, integración nativa con Maven, control de versiones, y un entorno visual amigable para el desarrollo ágil.

Uso en el proyecto:  
NetBeans ha permitido la gestión eficiente del código fuente, con soporte para estructuración por paquetes, ejecución de comandos Maven, y depuración durante el desarrollo de la aplicación de consola.

MySQL

MySQL es el sistema gestor de bases de datos relacional (RDBMS) utilizado para almacenar de forma persistente toda la información del sistema: empleados, técnicos e incidencias. Es gratuito, estable, ampliamente documentado y fácil de integrar con Java mediante JDBC.

Uso en el proyecto:  
Se diseñaron y crearon tres tablas (`tecnicos`, `empleados`, `incidencias`) con relaciones entre ellas para mantener la integridad referencial. Se utilizaron claves primarias, claves foráneas y restricciones para garantizar la calidad de los datos.

JDBC (Java Database Connectivity)

JDBC es una API de Java que permite la conexión y manipulación de bases de datos desde aplicaciones Java. Permite ejecutar consultas SQL y obtener resultados de forma programática.

Uso en el proyecto:  
Se utilizó para realizar todas las operaciones de persistencia (CRUD) sobre la base de datos. Se emplearon `PreparedStatement` para evitar inyecciones SQL y manejar parámetros de forma segura y eficiente.

Maven

Maven es una herramienta de automatización de compilación y gestión de dependencias para proyectos Java. Permite gestionar de forma centralizada todas las librerías necesarias, así como automatizar el proceso de compilación, ejecución y empaquetado del proyecto.

Uso en el proyecto:  
Se utilizó Maven para estructurar el proyecto según el estándar `src/main/java`, compilar automáticamente, y facilitar la integración con NetBeans y el plugin `exec-maven-plugin` para ejecutar la aplicación desde consola.

**Conclusión**

La aplicación cumple los requisitos del examen. Se ha estructurado de forma modular y clara usando el patrón MVC. Además, se han añadido informes útiles y validaciones esenciales para garantizar la fiabilidad del sistema. Su diseño permite futuras extensiones como exportar informes, interfaz gráfica o conexión segura.