UNIVERSIDAD MARIANO GÁLVEZ DE GUATEMALA

Centro Universitario San Juan Sacatepéquez

Facultad de Ingeniería

Curso: Programación 1

Jornada: Fin de Semana

Catedrático: Miguel Catalán

Proyecto Final Programación 1 -Manual Técnico.

Nombre del estudiante: Amada Noemi Cárcamo Renderos

Numero de carnet: 7590-24-16729

Sección: “B”

24 de mayo del 2025

Manual Técnico

Introducción: Este manual técnico está dirigido a desarrolladores, administradores de sistemas y personal responsable para dar el mantenimiento, actualización y ampliación del sistema de gestión de tickets. El documento describe con detalle la arquitectura del sistema, la estructura del código fuente, la configuración de la base de datos (PostgreSQL), el entorno de desarrollo necesario, buenas prácticas de seguridad, y sugerencias para su escalabilidad y mantenimiento.

El sistema ha sido diseñado con un enfoque modular usando el patrón Modelo-Vista-Controlador: es una forma de organizar el código de una aplicación para separarla en tres componentes principales, facilitando su mantenimiento y escalabilidad:

1. **Modelo**: Maneja los datos y la lógica del negocio. Se encarga de acceder a la base de datos y procesar la información.
2. **Vista**: Es la interfaz gráfica (GUI) que el usuario ve y con la que interactúa.
3. **Controlador**: Actúa como intermediario entre la vista y el modelo. Recibe acciones del usuario desde la vista, las procesa, y actualiza el modelo o la vista según corresponda.

Descripción General del Sistema Creación de Tickets

El Sistema de Tickets de Servicio es una aplicación de escritorio desarrollada en Java con JavaFX y respaldada por una base de datos en la nube en PostgreSQL. Su propósito principal es gestionar de forma eficiente las solicitudes de soporte técnico dentro de una organización. Este sistema automatiza el proceso de registro, asignación, seguimiento y resolución de incidencias reportadas por los usuarios, permitiendo mantener un control ordenado y transparente del flujo de atención.

Objetivos Principales

- Facilitar la creación de tickets por parte de los usuarios.
- Permitir la asignación automática o manual de tickets a técnicos dentro de departamentos específicos.
- Dar seguimiento al estado de los tickets mediante flujos de trabajo configurables.
- Habilitar a los técnicos para gestionar, resolver y documentar cada incidencia.
- Proporcionar a los administradores herramientas de control, configuración y auditoría de todo el sistema.

Roles del Sistema

- **Administrador:** controla configuraciones generales, roles, permisos, departamentos y flujos de trabajo.
- **Técnico:** gestiona tickets asignados, cambia estados, agrega notas.
- **Usuario:** crea tickets, consulta el estado de sus solicitudes y puede agregar notas complementarias.

Funcionalidades Clave

1. Configuración del sistema: definir parámetros como nombre de la empresa, idioma, zona horaria, prioridades, etc.
2. Gestión de roles y permisos: creación de perfiles con acciones específicas.
3. Registro de departamentos: áreas responsables de atender tickets.
4. Registro de usuarios: alta de usuarios con asignación de roles y departamentos.
5. Definición de estados de tickets: permite establecer pasos del flujo (pendiente, en proceso, resuelto…).
6. Gestión de flujos de trabajo: configuraciones dinámicas de transiciones válidas entre estados.
7. Gestión de tickets: creación, asignación, cambio de estado, cierre y cancelación de tickets.
8. Historial: seguimiento detallado de cambios en cada ticket.
9. Notas: interacción entre usuarios y técnicos mediante mensajes.
10. Uso de pilas y colas: estructura para mantener orden y seguimiento del historial y atención por orden de llegada.

Integración con PostgreSQL

- Todos los datos del sistema se almacenan en una base de datos PostgreSQL en la nube.
- Se implementan consultas SQL para listar tickets por filtros: estado, prioridad, fecha o departamento.
- Se usan claves foráneas y restricciones para mantener la integridad de los datos.

Tecnologías y conceptos implementados

- POO (Programación Orientada a Objetos): clases como Ticket, Usuario, Técnico, Administrador, Departamento.
- Herencia y polimorfismo: roles que heredan de la clase Persona.
- Manejo de excepciones: para validar datos, errores de conexión, duplicidad, etc.
- Estructuras dinámicas: uso de ArrayList para representar listas de tickets, usuarios y acciones.
- Colas: organización de atención en departamentos.
- Pilas: almacenamiento de historial de cambios en los tickets.

Arquitectura y Tecnología

- Lenguaje y herramientas
- Lenguaje principal: Java 8+
- Interfaz gráfica: Java Swing o Scene Builder para una apariencia moderna
- Base de datos: PostgreSQL (versiones 12 a 17 compatibles)
- Acceso a datos: JDBC

- Entorno de desarrollo: NetBeans / IntelliJ IDEA / Eclipse

Patrón de diseño

- MVC (Modelo-Vista-Controlador): separación clara de responsabilidades.
- DAO (Data Access Object): se recomienda implementar para encapsular el acceso a datos.
- JAVA FX y Scene Builder

Base de Datos

Tecnología usada

- PostgreSQL

Principales tablas

- **usuario**: gestiona los datos de los usuarios, credenciales y su rol.
- **rol**: define los tipos de usuarios (administrador, técnico, usuario).
- **permiso / rol_permiso**: gestión de permisos asignados a roles.
- **ticket**: contiene los detalles de cada solicitud creada por los usuarios.
- **estado_ticket**: define los posibles estados por los que puede pasar un ticket.
- **nota**: observaciones asociadas a tickets.
- **departamento**: área responsable del ticket.
- **configuracion_sistema**: parámetros generales como idioma, zona horaria, etc.
- **historial / historial_cambios**: guarda el historial.

Relaciones clave

- Un usuario pertenece a un departamento y tiene un rol.
- Un ticket es creado por un usuario y puede ser asignado a otro.
- rol_permiso asocia roles con sus permisos específicos.

Configuración del Entorno

1. **Herramientas necesarias**

- NetBeans 12.0 o superior
- JDK 8 o superior
- MySQL Server
- Librerías
- FXML
- Scene Builder

2. **Clonación del Proyecto:** Descargar el proyecto desde Github que se necesita clonar para el repositorio. Para ello necesitamos su URL que encontrarás en GitHub dentro del desplegable desde el que descargas el código en un fichero ZIP.

https://github.com/Amada07/Proyecto-Semestre-Programacion1.git

3. Configuración con la base de datos

- Crea la base de datos ejecutando el script SQL.
- Carga el proyecto en NetBeans.
- Configura el archivo ConexionBD.java con tus credenciales:

```
String url = "jdbc: mysql://localhost:3306/sistema_tickets";
String user = "usuario";
String password = "tu_contraseña";
```

4. Configuración y ejecución del sistema

- Abre el proyecto en NetBeans o el IDE de tu preferencia.
- Verifica que la conexión JDBC a SQL esté correctamente configurada.
- Ejecuta la clase principal del sistema.

5. Compilación y ejecución

En el IDE

- Ejecuta el archivo Main.java.
- Verifica que la conexión con la base de datos se realice correctamente.

```
javac -d bin src/**/*.java
java -cp bin Main
```

Diagrama Entidad Relación de la Base de Datos