Proyecto de Semestre

# 1. Introducción

El objetivo de este proyecto es desarrollar un Sistema de Tickets de Servicio en Java, con una interfaz gráfica amigable programada en JavaFX. Este sistema permitirá la gestión eficiente de tickets de soporte dentro de una organización, asegurando que las solicitudes sean atendidas de manera organizada.

Los estudiantes deberán implementar los siguientes casos de uso:

1. Configurar Parámetros del Sistema
2. Gestionar Roles y Permisos
3. Registrar Departamentos o Áreas de Trabajo
4. Registrar Usuarios Iniciales
5. Definir Estados de los Tickets
6. Configurar Flujos de Trabajo de los Tickets
7. Gestionar Tickets
8. Consultar Lista de Solicitudes Pendientes
9. Cambiar Estado de un Ticket
10. Agregar Nota a un Ticket

Este proyecto se deberá realizar de forma individual.

# 2. Requisitos Técnicos y Conceptos a Implementar

El sistema deberá estar programado en Java utilizando JavaFX para la interfaz gráfica y deberá cumplir con los siguientes conceptos clave:

**Nota Importante:** Todo el código generado para la implementación de su proyecto debe ser de forma obligatoria de su total autoría. Si se detectase el plagio o uso de código de un tercero, representará la anulación total de la ponderación del presente proyecto.

## 2.1. Programación Orientada a Objetos (POO)

* Abstracción: Definir clases que representen las entidades clave del sistema (Ticket, Usuario, Técnico, Administrador, Departamento, etc.).
* Encapsulamiento: Todos los atributos de las clases deben ser privados y accederse a través de métodos públicos (getters y setters).
* Herencia: Implementar una jerarquía de clases donde Administrador, Técnico y Usuario hereden de una clase base Persona.
* Polimorfismo: Métodos como cambiarEstadoTicket() pueden comportarse de manera distinta dependiendo del rol del usuario que los invoque.

## 2.2. Manejo de Excepciones

Manejo adecuado de errores en operaciones como creación de usuarios, asignación de tickets, acceso a la base de datos y validación de datos.

## 2.3. Serialización de Objetos a Archivos

Implementar la funcionalidad de guardar y cargar tickets en archivos como respaldo, utilizando lectura y escritura de objetos en archivos.

## 2.4. Arreglos de Datos Dinámicos

Utilizar estructuras como ArrayList y HashMap para almacenar tickets, usuarios y departamentos en memoria.

## 2.5. Bases de Datos (PostgreSQL en la Nube)

Todos los datos del sistema deben almacenarse en una base de datos PostgreSQL en la nube.

Se deben realizar consultas eficientes para filtrar y buscar tickets por estado, prioridad o departamento.

## 2.6. Pilas y Colas

Colas: Implementar colas de atención en cada departamento para organizar los tickets según su orden de llegada.

Pilas: Utilizar pilas para almacenar el historial de cambios de estado de cada ticket, permitiendo deshacer cambios si es necesario.

# 3. Entregas Parciales

El proyecto se desarrollará en 13 semanas, con 6 entregas parciales para asegurar un avance progresivo.

| **Fecha** | **Entrega** |
| --- | --- |
| 14 de marzo | Análisis del sistema y modelado UML (Identificación de clases y relaciones, diagrama de clases, diagramas de casos de uso). |
| 28 de marzo | Interfaz gráfica inicial en JavaFX (Diseño de pantallas principales, navegación entre vistas). |
| 11 de abril | Implementación de POO y manejo de excepciones (Clases, encapsulamiento, herencia y polimorfismo, validación de datos). |
| 25 de abril | Persistencia de datos y base de datos en la nube (Implementación de PostgreSQL, consultas, CRUD de tickets y usuarios). Diagrama entidad-relación. |
| 09 de mayo | Manejo de estructuras dinámicas, pilas y colas (Implementación de colas para tickets, historial de cambios con pilas). |
| 23 de mayo | Optimización y pruebas finales (Corrección de errores, pruebas, refinamiento de la interfaz gráfica). |
| **24 de mayo** | **Entrega del proyecto final** |

# 4. Entregables Finales

Los estudiantes deberán entregar:

* Código fuente en un repositorio público en GitHub.
* Documentación técnica:
  + Manual de usuario.
  + Manual técnico.
* Modelado UML del sistema.
* Diagrama Entidad-Relación
* Ejecutable del sistema.