Proyecto: HelpDesk U - Mesa de Ayuda con Bag of Words BoW

Primer Avance

Integrantes:

Santiago Carballo Alvarez

Amanda Padilla Leiva

Institución Educativa:

Universidad Cenfotec

Curso:

Programación Orientada a objetos

Profesor: Álvaro Cordero

Fecha de entrega: 31 de Octubre de 2025

Descripción del dominio

El dominio elegido corresponde a un sistema de gestión de tickets de soporte técnico que integra aspectos emocionales y técnicos para mejorar la atención al usuario. El propósito del sistema es registrar, clasificar y analizar tickets enviados por usuarios, asignándolos a un departamento correspondiente y detectando el estado de ánimo del texto usando un enfoque de Bag of Words (BoW).

Este dominio incluye usuarios que reportan problemas técnicos, departamentos que atienden los tickets, y los propios tickets, que contienen la descripción del problema, el usuario asociado y el departamento responsable. Además, incorpora diccionarios de palabras que agrupan categorías emocionales o técnicas, y un módulo de análisis BoW que permite reconocer emociones y temas técnicos en el texto de un ticket.

El modelo se implementa siguiendo una arquitectura por capas. La capa de Dominio incluye las entidades principales: Usuario, Departamento, Ticket, Diccionario, Palabra y analisisBow. La capa de Memoria contiene los repositorios (UsuarioRepo, ticketRepo, DepartamentoRepo, DiccionarioRepo y DataStore) que gestionan los datos en listas estáticas. La capa de UI (Interfaz de usuario) presenta menús de consola (MenuUsuario, MenuDepartamento, MenuTicket, MenuDiccionario y MenuPrincipal) que permiten interactuar con el sistema. Finalmente, la capa App contiene el punto de entrada del programa (main).

Justificación de las relaciones modeladas

El diagrama UML refleja cuatro tipos principales de relaciones según su naturaleza conceptual: composición, asociación, agregación y dependencia. Cada una cumple un rol específico dentro del modelo de datos y la interacción entre clases.

Composición (*-- compone)

Las composiciones representan contenedores que poseen otros objetos como parte esencial de su estructura. En este sistema, la clase Diccionario compone la clase Palabra, ya que cada palabra pertenece a un diccionario y no puede existir fuera de él. De manera similar, los repositorios (UsuarioRepo, ticketRepo, DepartamentoRepo y DiccionarioRepo) componen sus entidades, pues almacenan y controlan su ciclo de vida en memoria.

Asociación (-- se asocia)

Las asociaciones se utilizan cuando dos clases interactúan directamente sin dependencia jerárquica. En este caso, la clase Ticket se asocia tanto con Usuario como con Departamento, ya que cada ticket pertenece a un usuario y está asignado a un departamento. También la clase analisisBow se asocia con Diccionario, puesto que utiliza dos instancias (emocional y técnica) para realizar su análisis de texto.

Agregación (o-- colabora con)

Las agregaciones modelan colaboraciones importantes entre componentes, sin llegar a representar pertenencia. Se redujeron a las más significativas para mantener claridad en el diagrama. Por ejemplo, MenuUsuario, MenuDepartamento, MenuTicket y MenuDiccionario colaboran con sus respectivos repositorios para realizar operaciones de registro y consulta. Además, MenuPrincipal colabora con los menús secundarios, actuando como el controlador central que gestiona el flujo de la aplicación.

Dependencia (--> depende de)

Las dependencias reflejan un uso temporal o creación de objetos. La clase main depende de MenuPrincipal, ya que instancia y ejecuta la interfaz principal al inicio del programa. Este tipo de relación muestra que el flujo de ejecución del sistema inicia en App y se extiende hacia la capa de interfaz de usuario.

Instrucciones de ejecución

Abrir el proyecto en IntelliJ IDEA.
Es importante conservar la estructura de paquetes de la siguiente forma:

```
Dominio/
memoria/
UI/
app/
```

1.

Ejecutar el archivo principal.
Desde IntelliJ, abrir y ejecutar el archivo:

```
app/main.java
```

Este archivo contiene el método main, que inicializa el menú principal mediante la instrucción:

```
new MenuPrincipal().iniciar();
```

2.
3. **Navegar por los menús del sistema.**
    - **Usuarios:** permite registrar y listar usuarios del sistema.
    - **Departamentos:** permite registrar y listar departamentos.
    - **Tickets:** permite crear y listar tickets, asociándolos a un usuario y un departamento.
    - **Diccionarios:** permite crear diccionarios y agregar palabras emocionales o técnicas.
4. **Finalizar la ejecución.**
   En cualquier momento, desde el menú principal, el usuario puede seleccionar la opción “0” para salir del sistema.

Conclusión

El sistema implementa un modelo de soporte técnico inteligente, combinando relaciones bien estructuradas en UML para reflejar jerarquías reales y una arquitectura clara por capas. La composición asegura la integridad en la gestión de objetos internos, la asociación representa las interacciones directas entre usuarios, tickets y departamentos, la agregación simplifica la comunicación entre la capa lógica (repositorios) y la capa de presentación (interfaz), y la dependencia muestra la dirección del flujo de ejecución desde el punto de entrada hasta la interfaz principal.

En conjunto, el modelo UML no solo describe las relaciones entre las clases del código, sino que demuestra una arquitectura modular, reutilizable y extensible. Este diseño permite mantener un equilibrio entre organización, claridad y capacidad de ampliación, garantizando una base sólida para futuros desarrollos o integraciones más complejas dentro del sistema de gestión de tickets.