**UNIVERSIDADE CATÓLICA DE ANGOLA FACULDADE DE ENGENHARIA**

**Documento de Implemtação**

# **Sistema de Gestão de HelpDesk**

**Grupo 13:**

27254 - Nelson Dalva

#### **1. Análise Geral do Projeto**

O projeto consiste na construção de um sistema de gestão de Help Desk que atenda aos seguintes requisitos:

* **Backend** : Desenvolvido em Java 17 com Spring Boot.
* **Banco de Dados** : PostgreSQL.
* **Mensageria** : Apache Kafka para comunicação assíncrona.
* **Logs** : Implementação de uma solução de logs para rastrear todos os eventos ocorridos no sistema.
* **Portfolio Estruturado** : Uso de uma tabela normalizada (como a lista de medicamentos) para representar categorias de tickets ou serviços oferecidos.

#### **2. Definição das Classes**

As classes principais do sistema podem ser organizadas conforme os processos de negócio descritos na memória descritiva. Aqui está uma sugestão:

##### **2.1 Classes Principais**

1. **Usuário** :
   * Atributos: **id**, **nome**, **email**, **senha,status**.
   * Métodos: Autenticação, autorização.
2. **Colaborador (Subclasse de Usuário)** :
   * Atributos adicionais: setor.
   * Métodos: Registrar ticket, visualizar status de tickets.
3. **UsuarioSuporte (Subclasse de Usuário** :
   * Atributos adicionais: **tipo** (técnico, supervisor, administrador), **especialidade**: lista de especialidades do técnico.**Métodos** :
   * Métodos: Atualizar status de ticket, registrar ações, gerenciar usuários (apenas administradores).
4. **Ticket** :
   * Atributos: **id**, **descricao**, **categoria**, **prioridade**, **status**, **dataHoraCriacao**, **colaboradorId**, **tecnicoResponsavelId**.
   * Métodos: Criar ticket, atualizar status, atribuir técnico, registrar feedback.
5. **Categoria** :
   * Atributos: **id**, **nome**, **nivel** (ex.: nível 1, nível 2, etc.), **categoriaPaiId** (para hierarquia), **prioridadePadrao**.
   * Métodos: Obter subcategorias, listar categorias por nível.
6. **Log** :
   * Atributos: **id**, **mensagem**, **dataHora**, **evento, ticketId** (ex.: "Ticket criado", "Status alterado").
   * Métodos: Registrar log, consultar logs.
7. **SLA** :
   * Atributos: **id**, **categoriaId**, **tempoMaximoResposta**, **tempoMaximoResolucao**.
   * Métodos: Validar SLA, gerar alertas.
8. **Relatorio** :
   * Atributos: **id**, **periodo**, **metricas** (ex.: tempo médio de resolução, taxa de conformidade com SLA).
   * Métodos: Gerar relatório, exportar dados.

#### **3. Requisitos Funcionais**

Os requisitos funcionais devem estar alinhados com os objetivos específicos descritos na memória descritiva. Aqui estão alguns exemplos:

##### **3.1 Registro de Ticket**

* Permitir que colaboradores internos registrem tickets de suporte.
* Cada ticket deve conter uma descrição do problema, data de abertura, categoria atribuída e prioridade.

##### **3.2 Triagem e Alocação**

* O sistema deve realizar a triagem e atribuir automaticamente os tickets para a equipe ou técnico responsável com base em critérios como especialização técnica, carga de trabalho e prioridade.

##### **3.3 Escalonamento de Tickets**

* Caso o problema não seja resolvido, o ticket deve ser escalonado para um nível superior (ex.: supervisor).

##### **3.4 Resolução e Encerramento**

* Permitir que técnicos atualizem o status dos tickets e registrem ações realizadas.
* Solicitar confirmação do solicitante após a resolução e coletar feedback.

##### **3.5 Monitoramento de SLAs**

* O sistema deve monitorar o cumprimento de SLAs (Acordos de Nível de Serviço) e gerar alertas caso os prazos sejam excedidos.

##### **3.6 Geração de Relatórios**

* Supervisores devem ter acesso a relatórios de desempenho contendo métricas como tempo médio de resolução, taxa de conformidade com SLA e produtividade da equipe.

#### **3.7 Logs de Eventos**

* Todos os eventos ocorridos no sistema devem ser registrados e processados via Apache Kafka para fins de auditoria e monitoramento.

#### **3.8 Gerenciamento de Usuários e Perfis**

* O sistema deve permitir o cadastro, edição e remoção de usuários, bem como a definição de perfis de acesso (colaborador, técnico, supervisor, administrador).

#### **4. Lógica de Funcionamento**

A lógica de funcionamento do sistema segue os fluxos de negócio descritos anteriormente, detalhando como cada componente interage para garantir um processo eficiente e automatizado. Cada etapa é explicada com clareza, incluindo os mecanismos utilizados e as regras aplicadas.

**1 Registro de Ticket** :

**Descrição** :  
Um colaborador (usuário interno) submete uma solicitação com descrição, contexto e impacto percebido. O sistema determina automaticamente a categoria e prioridade do ticket com base em regras pré-definidas.

**Mecanismos Utilizados** :

1. **Processamento de Linguagem Natural (NLP)** :
   1. O sistema analisa a descrição do problema usando técnicas de NLP para identificar palavras-chave relevantes.
   2. Exemplo: Se a descrição contém "falha no servidor", o sistema atribui a categoria "Infraestrutura" e a prioridade "Alta".
2. **Regras Armazenadas em Arquivo Excel** :
   1. As regras para mapear palavras-chave a categorias e prioridades estão armazenadas em um arquivo Excel separado.
   2. O sistema utiliza Apache POI para ler esse arquivo dinamicamente.
   3. Estrutura do arquivo:

|  |  |  |  |
| --- | --- | --- | --- |
| **1** | **Palavra-Chave** | **CategoriaId** | **Prioridade** |
| **2** | falha no servidor | 1 | Alta |
| **3** | problema de impressão | 11 | Média |
| **4** | lentidão no sistema | 2 | Baixa |

**Armazenamento no Banco de Dados** :

1. O ticket é armazenado no banco de dados PostgreSQL com todas as informações geradas automaticamente (categoria, prioridade, data/hora, etc.).

**Registro de Logs via Kafka** :

1. Um evento é enviado ao Apache Kafka informando a criação do ticket (ticket-created), permitindo auditoria e monitoramento posterior.

**2 Triagem e Alocação** :

**Descrição** :  
O sistema avalia a especialização técnica, carga de trabalho atual e prioridade do ticket para direcioná-lo ao técnico ou equipe mais adequada. Caso nenhum técnico esteja disponível, o ticket é colocado em uma fila de espera.

**Mecanismos Utilizados** :

**Cálculo de Afinidade Técnica** :

1. O sistema consulta o banco de dados para verificar a especialidade de cada técnico cadastrado.
2. Exemplo: Um técnico especializado em "Redes" será preferido para tickets relacionados a problemas de conectividade.

**Avaliação da Carga de Trabalho** :

1. O sistema calcula o número de tickets atribuídos a cada técnico no momento.

**Priorização Baseada em SLA** :

Tickets com prioridade alta recebem precedência na alocação.

O SLA associado à categoria do ticket define os prazos máximos de resposta e resolução.

Exemplo de SLA:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | ID\_SLA | Categoria | Tempo\_Maximo\_Resposta | Prioridade |
| 2 | 1 | Infraestrutura | 4 horas | Alta |
| 3 | 2 | Software | 24 horas | Média |

**Fila de Espera** :

Se nenhum técnico estiver disponível, o ticket é colocado em uma fila de espera até que um técnico fique disponível.

**3 Escalonamento de Tickets** :

1. **Monitoramento de SLA** :
   * O sistema verifica periodicamente o tempo decorrido desde a criação do ticket.
   * Se o prazo estiver próximo de expirar, um alerta é gerado e enviado ao técnico responsável.
   * Se o prazo for excedido, o ticket é escalonado automaticamente.
2. **Identificação de Complexidade** :
   * Durante a atualização do status, o técnico pode marcar o ticket como "Complexo".
   * Tickets marcados como complexos são automaticamente escalonados para supervisores ou equipes especializadas.
3. **Notificação Automática** :
   * O supervisor recebe uma notificação via sistema informando que um ticket foi escalonado.

#### **4 Resolução e Encerramento**

* **Descrição** :  
  O técnico responsável executa as ações necessárias para resolver o problema. Após a resolução, o solicitante confirma a solução e fornece feedback. O ticket é encerrado, e um relatório é gerado automaticamente.
* **Mecanismos Utilizados** :
  1. **Atualização de Status** :
     + O técnico atualiza o status do ticket conforme o progresso (ex.: "Em andamento", "Esperando resposta do cliente").
     + Todas as alterações de status são registradas no log (ticket-updated).
  2. **Confirmação do Solicitante** :
     + Após a resolução, o sistema envia uma notificação ao solicitante solicitando confirmação.
     + O solicitante pode confirmar a solução ou reabrir o ticket caso o problema persista.
  3. **Coleta de Feedback** :
     + O sistema solicita ao solicitante que avalie o atendimento fornecido (ex.: satisfação geral, clareza das comunicações).
     + O feedback é armazenado no banco de dados para análise futura.
  4. **Encerramento do Ticket** :
     + Quando o solicitante confirma a solução, o ticket é marcado como "Encerrado".
     + Um evento é registrado no log (ticket-closed) e um relatório é gerado automaticamente.

#### **5 Monitoramento de SLAs**

* **Descrição** :  
  O sistema monitora continuamente o cumprimento de SLAs (Acordos de Nível de Serviço) e gera alertas caso os prazos sejam excedidos.
* **Mecanismos Utilizados** :
  1. **Verificação Periódica** :
     + O sistema verifica regularmente o tempo decorrido desde a criação de cada ticket.
     + Para cada ticket, o sistema compara o tempo decorrido com o prazo definido no SLA.
  2. **Geração de Alertas** :
     + Se o prazo estiver próximo de expirar, o sistema envia um alerta ao técnico responsável.
     + Se o prazo for excedido, o sistema escalona o ticket automaticamente.
  3. **Relatórios de Conformidade** :
     + O sistema gera relatórios consolidados mostrando a taxa de conformidade com SLAs (número de tickets resolvidos dentro do prazo).

#### **6 Geração de Relatórios**

* **Descrição** :  
  Supervisores acessam dados consolidados sobre o desempenho da equipe, incluindo métricas como tempo médio de resolução, taxa de conformidade com SLA e produtividade.
* **Métricas Calculadas** :
  + **Tempo Médio de Resolução** :
    - Fórmula: (SOMA(tempo\_resolucao) / total\_tickets)
  + **Taxa de Conformidade com SLA** :
    - Fórmula: (tickets\_dentro\_prazo / total\_tickets) \* 100
  + **Produtividade da Equipe** :
    - Fórmula: (total\_tickets\_resolvidos / numero\_tecnicos)
  + **Feedback Médio** :
    - Fórmula: (SOMA(feedback) / total\_feedbacks)
* **Exportação** :
  + Os relatórios podem ser exportados para formatos como PDF ou Excel para compartilhamento com stakeholders.

#### **5. Diagrama de Entidade e Relacionamentos**

Aqui está um esboço do diagrama de entidades:

#### **Entidades Principais**

1. **Usuário**:
   * Relacionamento **[1:1 com Colaborador e UsuárioSuporte]**: Um usuário pode ser um **Colaborador** (quem cria tickets) ou um **Usuário de Suporte** (quem atende tickets).
2. **Colaborador**:
   * **Relacionamento [1:N com Ticket]**: Um colaborador pode criar vários tickets.
3. **Usuário de Suporte**:
   * **Relacionamento [1:N com Ticket]**: Um usuário de suporte pode ser responsável por vários tickets.
4. **Ticket** :

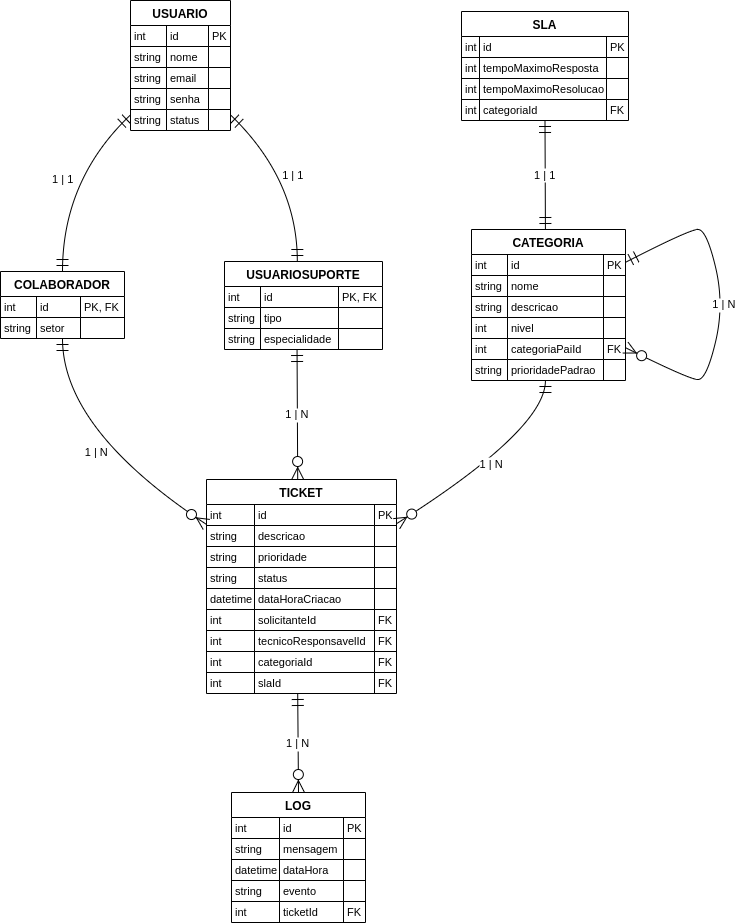
**Relacionamentos**:

* **[1:N com Categoria]**: Cada ticket pertence a uma única categoria, mas uma categoria pode ter vários tickets.
* **[1:1 com SLA]**: Cada ticket herda o SLA associado à sua categoria.
* **[1:N com Log]**: Um ticket pode gerar vários logs ao longo do seu ciclo de vida.

1. **Categoria** :
   * **Relacionamento [1:N com Subcategorias]**: Uma categoria pode ter várias subcategorias em uma estrutura hierárquica.
2. **SLA** :
   * **Relacionamento [1:1 com Categoria]**: Cada categoria tem um único SLA associado, que define os tempos de resposta e resolução.
3. **Log** :
   * **Relacionamento [1:N com Ticket]**: Um log está associado a um ticket específico, registrando eventos e ações realizadas.

#### **Exemplo de Relacionamentos**

* **Usuário → Colaborador/UsuárioSuporte** : 1:1 (Cada usuário pode ser um colaborador ou um usuário de suporte).
* **Colaborador → Ticket** : 1:N (Um colaborador pode criar vários tickets).
* **UsuárioSuporte → Ticket** : 1:N (Um usuário de suporte pode ser responsável por vários tickets).
* **Categoria → Subcategoria** : 1:N (Uma categoria pode ter várias subcategorias).
* **Categoria → Ticket** : 1:N (Uma categoria pode ter vários tickets).
* **SLA → Categoria** : 1:1 (Cada categoria tem um SLA associado).
* **SLA → Ticket** : 1:N (O SLA se aplica a vários tickets de uma mesma categoria).
* **Ticket → Log** : 1:N (Um ticket pode gerar vários logs ao longo do tempo).



#### **6. Mensageria com Apache Kafka (Logs e Eventos)**

#### **6.1** **Eventos a Serem Registrados**

* **Abertura de Tickets** : Quando um ticket é criado.
* **Atualização de Status** : Sempre que o status do ticket muda.
* **Fechamento de Tickets** : Quando o ticket é resolvido e fechado.
* **Erros e Falhas do Sistema** : Qualquer erro crítico detectado durante a execução.

#### **6.2 Kafka Topics**

* **ticket-created**: Armazena eventos de criação de tickets.
* **ticket-updated**: Armazena eventos de atualização de status.
* **ticket-closed**: Armazena eventos de fechamento de tickets.
* **system-logs**: Armazena logs gerais do sistema.

**6.3 Funcionalidades**

A implementação de uma solução de logs usando Apache Kafka pode ser usada para:

* Rastrear todos os eventos ocorridos no sistema (criação de tickets, atualizações de status, alocações de técnicos, etc.).
* Armazenar logs em um tópico Kafka para análise posterior.
* Garantir que os logs sejam persistentes e acessíveis mesmo em ambientes distribuídos.
* Os logs e eventos são armazenados em tópicos Kafka para facilitar o processamento assíncrono.
* Os dados podem ser consumidos posteriormente para auditoria, monitoramento e análise de desempenho.

#### **7. Portfolio Estruturado**

O sistema utilizará um portfolio estruturado para organizar categorias de tickets em níveis hierárquicos. Essa abordagem permite maior flexibilidade e clareza na classificação de problemas.

#### **Estrutura do Portfolio**

* **Nível 1** : Categorias principais (ex.: Hardware, Software, Rede).
* **Nível 2** : Subcategorias (ex.: Problemas de Impressoras, Falhas em Servidores).
* **Nível 3** : Subcategorias específicas (ex.: Falhas em Impressoras Laser, Erros de Conexão Wi-Fi).

#### **Importação via Excel**

* O sistema fará o uso da biblioteca Apache POI para ler um arquivo Excel contendo as categorias.
* O arquivo será importado para o banco de dados PostgreSQL, permitindo que o sistema utilize a hierarquia de categorias dinamicamente.

#### **Exemplo de Hierarquia**

* **Hardware** :
  + **Impressoras** :
    - **Impressoras Laser**
    - **Impressoras Jet**
  + **Servidores** :
    - **Falhas em Servidores de Aplicação**
    - **Falhas em Servidores de Banco de Dados**

#### **Funcionalidade**

* Micro-serviços na camada Controller permitem retornar listas de categorias por nível ou subcategorias associadas a qualquer item do portfólio.