REQUISITOS DO SISTEMA

1. Visão Geral

Um sistema web/mobile que se conecta (via API) ao sistema da Copel, recebe dados em tempo real dos sensores espalhados na cidade e ajuda equipes e operadores a:

- **Visualizar** o mapa da rede elétrica urbana
- **Detectar** e **alertar** falhas imediatamente
- **Acompanhar** o histórico de incidentes e tempo de resposta

2. Atores / Perfis de Usuário

1. **Operador de Centro de Controle**
    - Fica no dashboard monitorando o mapa e recebendo alertas
2. **Técnico de Campo**
    - Recebe alertas no celular/tablet e consulta detalhes da falha
3. **Administrador do Sistema**
    - Gerencia contas, permissões e configurações gerais

3. Requisitos Funcionais

3.1. Integração de Dados

- **RF-01**: O sistema deve consumir uma **API REST** da Copel para obter o status dos sensores.
- **RF-02**: Deve aceitar dados de sensores (via POST) em formato JSON, com campos como `sensorId`, `timestamp`, `voltageLevel`, `status`.

3.2. Dashboard de Monitoramento

- **RF-03**: Mostrar um **mapa interativo** com localização dos sensores e cor/status (verde=normal, vermelho=falha).
- **RF-04**: Permitir filtros por região, tipo de falha e período (última hora, dia, semana).
- **RF-05**: Listar na lateral os eventos de falha mais recentes com “click para detalhes”.

3.3. Alertas e Notificações

- **RF-06**: Enviar **push notification** e/ou e-mail para operadores quando um sensor reportar falha.

- **RF-07**: Registrar cada notificação no log, com data/hora, sensor, tipo de falha e status de “lida/não lida”.

3.4. Gestão de Incidentes

- **RF-08**: Permitir ao operador **abrir um ticket** de incidente diretamente no sistema, vinculando sensor e descrição do problema.
- **RF-09**: Técnico deve poder **atualizar status** do ticket (em andamento, resolvido) e comentar no histórico.

3.5. Relatórios e Histórico

- **RF-10**: Gerar relatórios exportáveis (CSV/PDF) de incidentes, com métricas como tempo médio de resposta.
- **RF-11**: Exibir gráficos simples no front-end (ex.: número de quedas por dia).

3.6. Administração

- **RF-12**: CRUD de usuários (cadastro, edição, exclusão) com perfis e permissões.
- **RF-13**: Configurar limites de alerta (por exemplo, tensão mínima antes de considerar falha).

4. Requisitos Não-Funcionais

1. **Desempenho**
    a. Dashboard carrega dados em até 3s;
    b. Alarmes chegam ao operador em até 1 minuto após o evento.
2. **Disponibilidade**
    a. 99% de uptime do serviço backend.
3. **Segurança**
    a. Comunicação via HTTPS;
    b. Autenticação JWT para API;
    c. Controle de acesso por perfil.
4. **Escalabilidade**
    a. Preparado para suportar até 1.000 sensores simultâneos sem degradação.
5. **Usabilidade**
    a. Interface simples, com legendas claras;
    b. Mobile-friendly para técnicos de campo.

5. Tecnologias

- **Back-end**: Java & Spring Boot (API REST com Spring Web + JPA + JDBC ou H2)
- **Front-end**: HTML, CSS & Javascript puro (conhecido como VanillaJS) e, se quiser e manjar, React (ou Angular) com alguma biblioteca de mapas (Leaflet, OpenLayers)
- **Banco de Dados**: SQLite (dados de sensores, tickets, logs)

6. Histórias de Usuário (Exemplos)

1. **Como Operador**, quero ver no mapa em tempo real quais sensores estão com falha, para poder acionar a equipe rapidamente.
2. **Como Técnico**, quero receber uma notificação no celular com detalhes do incidente, para saber aonde ir e o que verificar.
3. **Como Administrador**, quero definir quais usuários podem abrir tickets e quais só podem visualizar, para manter a segurança.