Design Doc: Adoção da Arquitetura de Microsserviços no AlgaSensors

1. Visão Geral

Este documento de design detalha a adoção de uma arquitetura de microsserviços para a aplicação AlgaSensors, monitoramento de sensores, composta por três serviços:

- Device Management
- Temperature Monitoring
- Temperature Processing

A arquitetura de microsserviços permite escalabilidade, resiliência, manutenção e desenvolvimento independentes para cada **capacidade de negócio**.

2. Objetivos

- **Escalabilidade Independente**: Permitir a escalabilidade individual de cada serviço com base na sua carga de trabalho específica.
- **Isolamento de Falhas**: Garantir que problemas em um serviço não afetem os demais.
- **Desenvolvimento e Implantação Independentes**: Acelerar o ciclo de desenvolvimento e permitir implantações contínuas e independentes.
- **Manutenibilidade**: Facilitar a manutenção e evolução de cada serviço com código altamente coeso e específico.

3. Descrição dos Serviços

3.1 Capacidades de Negócio Mapeadas

Foram identificadas e mapeadas duas principais **capacidades de negócio** para a aplicação AlgaSensors:

1. **Gestão de Sensores:**
    - Envolve o cadastro, configuração e gerenciamento dos sensores, garantindo que todos os dispositivos sejam registrados, ativados, desativados e atualizados de forma centralizada.
    - **Serviço Relacionado:** Device Management Service.

2. **Monitoramento de Temperaturas:**
    - Envolve a coleta, processamento, armazenamento e análise dos dados de temperatura recebidos dos sensores, além da geração de alertas e relatórios históricos.
    - **Serviços Relacionados:** Temperature Processing Service e Temperature Monitoring Service.

A abordagem para a criação dos microsserviços foi baseada na **divisão por capacidade de negócio**. No entanto, no caso do **Temperature Processing Service**, houve uma **divisão adicional por motivos técnicos** para garantir que o processamento de dados em tempo real fosse eficiente e escalável, separando-o do **Temperature Monitoring Service**, que foca na análise e armazenamento de dados.

3.2 Serviço de Device Management

Descrição: Responsável pela **Gestão de Sensores**, este serviço cuida do cadastro, configuração e gerenciamento remoto dos sensores.

- **Funcionalidades Principais:**
    - Cadastro de sensores
    - Gerenciamento remoto de sensores

Endpoints:

- `POST /api/sensors`: Cadastra um novo sensor
- `GET /api/sensors`: Lista todos os sensores
- `GET /api/sensors/{sensorId}`: Obtém sensor específico
- `PUT /api/sensors/{sensorId}`: Atualiza sensor específico
- `DELETE /api/sensors/{sensorId}`: Remove sensor específico
- `PUT /api/sensors/{sensorId}/enable`: Ativa sensor específico
- `DELETE /api/sensors/{sensorId}/enable`: Desativa sensor específico

Objetos de negócio:

- Sensor
    - Propriedades
        - Id: TSID
        - Name: String
        - Location: String
        - IP: String
        - Protocol: String
        - Model: String
        - Enabled: Boolean

Tecnologias Sugeridas:

- Linguagem de Programação: Java
- Banco de Dados: Postgres

3.3 Serviço de Temperature Processing

Descrição: Responsável pelo recebimento dos dados dos sensores em tempo real, este serviço foi dividido do **Temperature Monitoring Service** por **motivos técnicos**, visando garantir eficiência no processamento de grandes volumes de dados.

- **Funcionalidades Principais:**
  - Recebimento de dados em tempo real dos sensores

Endpoints:

- `POST /api/sensors/{sensorId}/temperatures/data`: Recebimento de dados dos sensores via HTTP

Tecnologias Sugeridas:

- Linguagem de Programação: Java

Objetos de negócio:

- Temperature Log
  - Propriedades
    - Id: UUIDv7
    - SensorId: TSID
    - Registred At: OffsetDateTime
    - Value: Double

3.4 Serviço de Temperature Monitoring

Descrição: Responsável pela análise dos dados e geração de relatórios, este serviço faz parte da capacidade de negócio de **Monitoramento de Temperaturas**.

- **Funcionalidades Principais:**
  - Armazenamento dos dados dos sensores
  - Permitir consulta ao histórico de temperatura
  - Configuração e disparo de alertas de temperatura

Endpoints:

- `GET /api/sensors/{sensorId}/temperatures`: Lista de registro de temperaturas
- `PUT /api/sensors/{sensorId}/alert`: Atualiza configuração de alerta de
- `GET /api/sensors/{sensorId}/alert`: Consulta configuração de alerta de temperatura

- `DELETE /api/sensors/{sensorId}/alert`: Deleta configuração de alerta de temperatura
- `GET /api/sensors/{sensorId}/monitoring`: Detalhes do monitoramento
- `PUT /api/sensors/{sensorId}/monitoring/enable`: Ativa monitoramento
- `DELETE /api/sensors/{sensorId}/monitorig/enable`: Inativa monitoramento

Tecnologias Sugeridas:

- Linguagem de Programação: Java
- Banco de Dados: Postgres

Objetos de negócio:

- Temperature Log
  - Propriedades
    - Id: UUID
    - SensorId: TSID
    - Registred At: OffsetDateTime
    - Value: Double
- Sensor Monitoring
  - Id: TSID
  - Last Temperature: Double
  - Updated At: OffsetDateTime
  - Enabled: Boolean
- Sensor Alert
  - Id: TSID
  - MaxTemperature: Double
  - MinTemperature: Double

4. Comunicação entre Microsserviços

A comunicação entre os microsserviços foi projetada para refletir as **capacidades de negócio** e as **necessidades técnicas**.

Assíncrona: Utilização de um sistema de mensageria (como RabbitMQ) para comunicações onde a latência não é crítica.

- **Exemplo:** O serviço de **Temperature Processing** publica mensagens de novos dados recebidos, que são consumidos pelo serviço de **Temperature Monitoring**.

Síncrona: Utilização de HTTP em APIs RESTful para comunicações em tempo real.

- serviço de **Device Management** envia uma requisição **ture Monitoring** para desativar o monitoramento caso o sensor seja desativado.

5. Problema e Solução

Com base na entrevista com Carlos, CEO da AlgaSensors, identificamos os principais problemas enfrentados pela empresa e como a arquitetura de microsserviços proposta pode resolvê-los.

5.1 Problemas Identificados

1. **Falta de Centralização de Dados:**
   - A empresa utiliza diversos tipos de sensores, cada um com sua própria tecnologia, e o dashboard atual não consegue unificar todas essas informações.
   - **Impacto:** Dificuldade em monitorar todos os sensores de forma centralizada, resultando em ineficiência operacional.
2. **Desempenho do Dashboard:**
   - O dashboard atual fica mais lento à medida que novos sensores são adicionados.
   - **Impacto:** A experiência do usuário é prejudicada, e a capacidade de monitoramento em tempo real é comprometida.
3. **Falta de Alertas de Oscilações de Temperatura:**
   - O dashboard não envia alertas quando a temperatura atinge valores máximos ou mínimos pré-definidos.
   - **Impacto:** Risco de não detectar variações críticas de temperatura, o que pode levar a problemas operacionais ou de segurança.
4. **Falta de Histórico de Temperaturas:**
   - O dashboard atual não armazena um histórico de temperaturas, exibindo apenas a temperatura atual.
   - **Impacto:** Dificuldade em analisar tendências e tomar decisões baseadas em dados históricos.
5. **Configuração Manual e Descentralizada dos Sensores:**
   - É necessário acessar cada sensor individualmente para configurá-lo, o que é trabalhoso e pouco eficiente.
   - **Impacto:** Aumento do tempo e custo de manutenção, além de maior probabilidade de erros humanos.

5.2 Solução Proposta

A arquitetura de microsserviços proposta foi desenhada para resolver esses
rma eficiente e escalável. Abaixo, detalhamos como cada problema

5.2.1 Centralização de Dados de Diferentes Sensores

- **Solução:** O **Device Management Service** é responsável pelo cadastro e gestão de todos os sensores, independentemente da tecnologia utilizada.
- **Benefício:** Todos os sensores são gerenciados e monitorados a partir de uma única plataforma, eliminando a necessidade de dashboards separados para cada tecnologia.

5.2.2 Desempenho do Dashboard

- **Solução:** A arquitetura de microsserviços permite a escalabilidade independente de cada serviço. Por exemplo, o **Temperature Processing Service** pode ser escalado horizontalmente para lidar com um grande volume de dados de sensores, enquanto o **Temperature Monitoring Service** pode ser otimizado para consultas rápidas e geração de relatórios.
- **Benefício:** O sistema como um todo se torna mais responsivo, mesmo com a adição de novos sensores.

5.2.3 Alertas de Oscilações de Temperatura

- **Solução:** O **Temperature Monitoring Service** é responsável por configurar e disparar alertas com base nos valores máximos e mínimos de temperatura definidos para cada sensor. Ele utiliza um sistema de mensageria (como RabbitMQ) para notificar os usuários em tempo real quando os limites são atingidos.
- **Benefício:** A empresa pode detectar e responder rapidamente a variações críticas de temperatura, reduzindo riscos operacionais.

5.2.4 Histórico de Temperaturas

- **Solução:** O **Temperature Monitoring Service** armazena todos os dados de temperatura recebidos do **Temperature Processing Service** em um banco de dados temporal (como o Postgres). Isso permite a consulta de históricos de temperatura por longos períodos.
- **Benefício:** A empresa pode analisar tendências, gerar relatórios históricos e tomar decisões baseadas em dados.

5.2.5 Configuração Unificada dos Sensores

- **Solução:** O **Device Management Service** oferece uma interface única para a configuração remota de todos os sensores. Através de endpoints RESTful, os usuários podem ativar, desativar, atualizar e configurar sensores sem a necessidade de acessar cada um individualmente.
- **Benefício:** Redução do tempo e custo de manutenção, além de maior eficiência

5.3 Benefícios Gerais da Solução

- **Escalabilidade:** A arquitetura de microsserviços permite que cada serviço seja escalado de forma independente, garantindo que o sistema continue funcionando de maneira eficiente à medida que o número de sensores aumenta.
- **Resiliência:** O isolamento entre os serviços garante que falhas em um serviço não afetem os demais, aumentando a confiabilidade do sistema.
- **Manutenibilidade:** Cada serviço possui um escopo bem definido, o que facilita a manutenção e a evolução do sistema.
- **Flexibilidade:** A adoção de tecnologias modernas (como Java, Postgres e RabbitMQ) e a comunicação assíncrona e síncrona entre os serviços garantem que o sistema seja adaptável a novas necessidades e tecnologias futuras.

5.4 Próximos Passos

1. **Desenvolvimento dos Microsserviços:**
   - Priorizar o desenvolvimento do **Device Management Service** para unificar o gerenciamento dos sensores.
   - Em seguida, desenvolver o **Temperature Processing Service** para receber e processar dados em tempo real.
   - Por fim, implementar o **Temperature Monitoring Service** para análise de dados, geração de alertas e armazenamento de histórico.
2. **Integração com Sensores Existentes:**
   - Desenvolver adaptadores para integrar os diferentes tipos de sensores ao **Device Management Service** e ao **Temperature Processing Service**.
3. **Implementação de Alertas:**
   - Configurar o **Temperature Monitoring Service** para enviar alertas via e-mail, SMS ou integração com aplicativos de mensagens.

6. Conclusão

A arquitetura de microsserviços proposta resolve os principais problemas identificados na entrevista com o CEO da AlgaSensors, oferecendo uma solução escalável, resiliente e de fácil manutenção. Com a centralização de dados, alertas em tempo real, histórico de temperaturas e configuração unificada de sensores, a empresa estará melhor preparada para atender às necessidades dos clientes e crescer de forma sustentável.