INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA

ALBERT SILVA DE JESUS

Sistema IoT Resiliente com Edge, Fog e InteliIoT

Santo Antônio de Jesus - Bahia

2020

Relatório Técnico – Backend IoT para Agricultura Inteligente

1. Introdução

O presente projeto consiste no desenvolvimento de uma aplicação backend simulada para Internet das Coisas (IoT), com foco em agricultura inteligente. Diante da crescente demanda por otimização da produção agrícola e gestão eficiente de recursos, o uso de tecnologias de monitoramento tem se mostrado estratégico. No entanto, este sistema não coleta dados reais, mas simula o envio e o recebimento de informações de sensores ambientais, como temperatura, umidade e luminosidade, permitindo a experimentação e validação de soluções de IoT sem a necessidade de hardware físico.

A aplicação, construída em Java com Spring Boot, funciona como um hub central de dados, reunindo, processando e expondo informações simuladas para APIs REST, enquanto reproduz o comportamento de protocolos de comunicação amplamente utilizados em sistemas IoT, como MQTT e AMQP (RabbitMQ). Além disso, a implementação inclui mecanismos de segurança com Spring Security e autenticação via JWT, bem como a possibilidade de integração real com brokers externos, oferecendo um ambiente híbrido para estudo e testes.

O projeto visa demonstrar de forma prática e didática a arquitetura de sistemas IoT distribuídos, contemplando camadas Edge, Fog e Cloud, permitindo compreender o fluxo de dados desde a simulação da coleta até a persistência e exposição das informações. Assim, a solução oferece uma base sólida para aprendizagem e validação de técnicas de comunicação, processamento e segurança em aplicações IoT, mesmo quando os dados não são coletados em tempo real nem provenientes de dispositivos físicos.

2. Cenário Escolhido e Justificativa

O cenário escolhido para este projeto é o **monitoramento de uma fazenda inteligente**. Este ambiente foi selecionado por sua relevância direta na aplicação de conceitos de IoT. A agricultura de precisão, que utiliza dados para otimizar o uso de fertilizantes, água e pesticidas, depende fundamentalmente da coleta contínua de

informações do campo. A simulação de sensores agrícolas permite demonstrar como um sistema real pode:

- **Identificar anomalias:** Detectar rapidamente temperaturas elevadas ou baixa umidade, que podem impactar a lavoura.
- **Automatizar respostas:** Acionar sistemas de irrigação ou alertas de forma automática com base nos dados.
- **Otimizar recursos:** Analisar padrões de dados para otimizar o uso de recursos hídricos e energéticos.

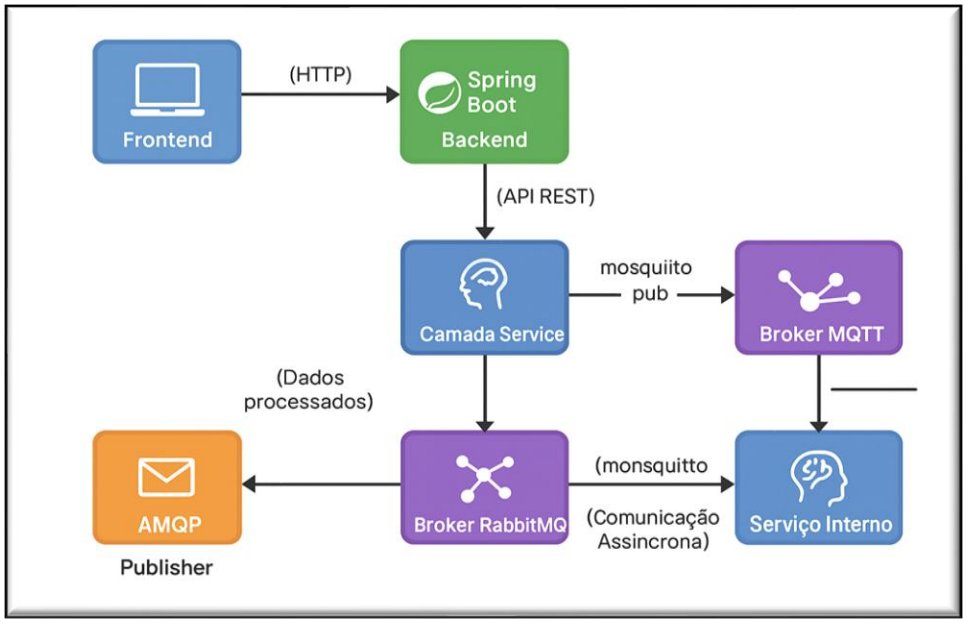
A escolha deste cenário proporciona uma base prática e funcional para a demonstração da arquitetura e das funcionalidades de um sistema IoT.

3. Arquitetura do Sistema

A arquitetura do projeto foi concebida para refletir um sistema de processamento distribuído, seguindo o modelo **Edge-Fog-Cloud**.

- **Camada Edge (Borda):** Representada pela classe SensorScheduler, que simula a leitura de dados de sensores. Esta camada, que em um ambiente real seriam microcontroladores ou sensores físicos, é responsável por gerar dados brutos em intervalos regulares.
- **Camada Fog (Névoa):** Atuando como um hub intermediário, a classe SensorDataService recebe e pré-processa os dados da camada Edge. É nesta camada que a lógica de negócio, como a detecção de alertas, é aplicada. A comunicação entre os protocolos (MQTT para entrada e AMQP para saída) também é orquestrada aqui.
- **Camada Cloud (Nuvem):** A persistência dos dados e a exposição da API REST representam a camada Cloud. Os dados são armazenados no banco de dados em memória **H2 para teste e PostegreSql**, simulando um ambiente de nuvem centralizado.

A comunicação entre essas camadas é mediada por brokers de mensageria, conforme ilustrado no diagrama a seguir:



4. Protocolos e APIs

A comunicação é o pilar central do sistema e foi implementada com três protocolos distintos para diferentes propósitos:

- **HTTP/REST:** Utilizado para a comunicação cliente-servidor. É ideal para interações diretas, como a autenticação de usuários e a consulta de dados via APIs RESTful.
- **MQTT (Message Queuing Telemetry Transport):** Protocolo leve e assíncrono, perfeito para a comunicação entre dispositivos de campo (simulados) e o backend. Sua arquitetura de publicação/assinatura permite o envio eficiente de dados de telemetria.
- **AMQP (Advanced Message Queuing Protocol):** Protocolo robusto e confiável, utilizado para a comunicação interna entre os serviços da aplicação. Sua capacidade de roteamento e garantia de entrega de mensagens o torna a escolha ideal para o processamento de alertas e eventos críticos.

As APIs desenvolvidas oferecem endpoints para:

- Autenticação (**/api/auth**): Permite o registro e login de usuários, retornando um JWT para acesso às rotas protegidas.
- Dados de Sensores (**/api/sensores**): Permite listar leituras, registrar novos dados e controlar o salvamento de alertas.
- Comunicação Direta (**/api/sensores/enviar**): Simula o envio de mensagens diretamente via MQTT ou AMQP.

- Integração (**/api/clima**): Demonstra a integração com APIs externas, como o OpenWeatherMap.

5. Mecanismos de Segurança

A segurança foi uma consideração fundamental no desenvolvimento do projeto, garantindo a integridade e a confidencialidade dos dados.

- **TLS/HTTPS:** O servidor está configurado para operar com **HTTPS** na porta **8443**, utilizando um certificado .p12 para criptografar a comunicação entre o cliente e o backend, protegendo contra interceptações.
- **JWT para Autenticação e Autorização:** A autenticação é gerenciada por **JSON Web Tokens**. Após o login, o usuário recebe um token que deve ser incluído no cabeçalho Authorization para acessar as rotas protegidas. Isso garante que apenas usuários válidos possam interagir com as funcionalidades principais.
- **Spring Security e BCrypt:** O **Spring Security** foi implementado para gerenciar a autenticação e autorização. As senhas dos usuários são armazenadas no banco de dados de forma segura, utilizando o algoritmo de criptografia **BCrypt**, que garante que as credenciais nunca sejam salvas em texto simples.

6. Integração com Conceitos de IoT Inteligentes

O projeto não se limita a um simples backend, mas incorpora ativamente conceitos-chave da arquitetura **IntelliIoT**, que distribui a inteligência computacional para otimizar o processamento de dados.

- **Edge Computing:** O SensorScheduler atua como um simulador de dispositivo Edge, pré-processando e coletando dados na "borda" da rede antes de enviá-los para o processamento centralizado.
- **Fog Computing:** A ponte entre os brokers (MQTT e AMQP) e a lógica de detecção de alertas no SensorDataService é um exemplo de **Fog Computing**. Este processamento intermediário reduz a latência e o volume de dados enviados para a nuvem.

- **Cloud:** A persistência dos dados no H2 e a exposição das APIs REST representam a camada de nuvem, onde ocorre o armazenamento a longo prazo e a análise dos dados coletados.
- **Inteligência Contextual e Escalabilidade:** A lógica de detecção de alertas adiciona uma camada de **inteligência contextual** ao sistema, permitindo respostas proativas a eventos. A arquitetura modular e o uso de filas de mensagens permitem que o sistema seja facilmente escalável para processar um número crescente de dispositivos.

7. Conclusão Crítica

O desenvolvimento deste backend de IoT para agricultura inteligente foi uma experiência de aprendizado valiosa. Ele validou a importância de uma arquitetura robusta para o gerenciamento de dados de sensores, destacando as seguintes vantagens:

- **Leveza e Eficiência:** O uso de protocolos como MQTT e a arquitetura de microserviços em Spring Boot garantem uma aplicação eficiente e de baixo consumo de recursos.
- **Segurança:** A implementação de segurança com JWT e criptografia de senhas é essencial para proteger um sistema IoT em ambientes de produção.
- **Modularidade:** A estrutura de pacotes e a separação de responsabilidades facilitam a manutenção e a adição de novas funcionalidades.
- **Uso de Protocolos Adequados:** A seleção estratégica de HTTP, MQTT e AMQP demonstra a capacidade de usar a ferramenta correta para cada tipo de comunicação, otimizando o fluxo de dados.

No entanto, algumas **limitações** foram identificadas:

- **Dependência de Simulação:** A maior parte da interação com sensores é simulada, o que impede a validação completa do sistema com hardware real e suas particularidades.
- **Execução Local:** O projeto é executado localmente, o que não reflete os desafios de um ambiente de produção em escala de nuvem.

Em última análise, este projeto atesta a capacidade de construir uma solução backend completa e segura para o ecossistema de IoT, servindo como uma base sólida para futuras implementações em escala real.