INSTITUTO DE INFORMÁTICA

PHYTOFLOW

SISTEMA DE IRRIGAÇÃO INTELIGENTE

Documento de Arquitetura (v1.0)

JAIR SOUZA MEIRA RODRIGUES
GABRIEL MILHOMEM CUNHA
GABRIEL CARDOSO DE CASTRO
VITOR LIMA RIBEIRO

Sumário

Introdução

O Sistema de Irrigação por Nebulização Automático foi projetado para otimizar o uso de água em cultivos, garantindo a irrigação precisa e eficiente de acordo com as condições ambientais. A arquitetura é baseada em uma solução escalável e segura, utilizando tecnologias modernas para IoT e processamento em nuvem. Todo o backend é desenvolvido em Python, aproveitando a ampla compatibilidade e a capacidade de integrar facilmente serviços de análise e processamento de dados.

A arquitetura está organizada em camadas, cada uma com um propósito específico, visando a modularidade, segurança e escalabilidade. O sistema se comunica com sensores e atuadores locais via microcontroladores, realiza processamento de dados em tempo real e batch, armazena e organiza dados para análises, e apresenta as informações em uma aplicação móvel amigável.

1. Visão Geral da Arquitetura

1.1 Componentes Principais

1. **Camada de Dispositivos (Edge Layer):** Esta camada é composta por sensores e atuadores responsáveis por monitorar e atuar sobre o ambiente. Os microcontroladores ESP8266 Node MCU são utilizados para coletar dados dos sensores e enviar comandos aos atuadores de irrigação, enquanto o gateway IoT gerencia a comunicação com a nuvem, fornecendo redundância e caching local.
2. **Camada de Comunicação:** A comunicação entre os dispositivos e a nuvem é realizada por meio de uma rede WiFi Mesh e utiliza o protocolo MQTT para troca de mensagens. O broker escolhido é o Apache Pulsar, que garante a alta disponibilidade e confiabilidade das mensagens trocadas. Um API Gateway centraliza as requisições externas, oferecendo autenticação e caching para melhorar o desempenho.
3. **Camada de Backend:** Todo o backend é implementado em Python, utilizando uma arquitetura de microsserviços. Estes microsserviços lidam com processamento de dados em tempo real, controle de irrigação, alertas, gerenciamento de dispositivos e integração com serviços de previsão do tempo. Há também um pipeline de processamento em batch para análises mais profundas e geração de relatórios.
4. **Camada de Dados:** Esta camada é responsável pelo armazenamento e gerenciamento dos dados coletados pelo sistema. Ela utiliza um banco de dados time series, um data lake para armazenamento bruto e processamento, um cache distribuído para melhorar a performance e um banco relacional para gerenciar informações estruturadas.
5. **Camada de Apresentação:** Esta camada fornece a interface de interação com o usuário, concentrando-se em um aplicativo móvel desenvolvido em React Native. O app permite a visualização de dados em tempo real, envio de

comandos manuais de irrigação e o recebimento de notificações push, além de funcionalidades offline.

1.2 Integrações Externas

O sistema possui integrações com serviços de previsão do tempo, um banco de dados de informações de plantas para apoiar a tomada de decisões, plataformas de monitoramento e sistemas de energia solar para otimizar o uso de recursos energéticos.

2. Arquitetura Detalhada

2.1 Camada de Dispositivos

1. **Sensores:** Sensores IoT para monitorar temperatura, umidade, luz e raios UV. Eles se conectam ao sistema via MQTT e operam com uma frequência de leitura configurável, ajustável de 30 segundos a 5 minutos. Em caso de falha de conexão, há um buffer local para armazenamento temporário de dados. A configuração inclui um modo de economia de energia para aumentar a eficiência.
2. **Microcontroladores:** Utiliza-se ESP8266 Node MCU, um microcontrolador com suporte para atualização de firmware OTA e com um watchdog embutido para auto recuperação em casos de falhas.
3. **Gateway IoT:** Também baseado no ESP8266 Node MCU, o gateway gerencia a comunicação local e o envio de dados para a nuvem, além de manter um cache local e realizar processamentos de borda para decisões imediatas.

2.2 Camada de Comunicação

1. **Rede WiFi Mesh:** Utiliza o protocolo IEEE 802.11s para formar uma rede mesh confiável, com múltiplos pontos de acesso e redundância automática. Implementa QoS para priorizar dados críticos.
2. **Mensageria:** O broker de mensagens é o Apache Pulsar, configurado com tópicos específicos por tipo de sensor e qualidade de serviço (QoS) ajustável. Também permite a retenção de mensagens para manter o último estado conhecido dos sensores.
3. **API Gateway:** Um gateway, como o Kong, é utilizado para centralizar o acesso, oferecendo controle de taxa de requisições, autenticação, autorização e caching para melhorar o desempenho e a segurança.

2.3 Camada de Backend

O backend, desenvolvido inteiramente em Python, é composto por uma arquitetura de microsserviços que cobre as principais funcionalidades do sistema.

Entre esses serviços, encontram-se o **Sensor Data Service**, que gerencia e processa os dados coletados pelos sensores; o **Irrigation Control Service**, responsável pelo controle dos atuadores de irrigação; o **Alert Service**, que emite notificações e alertas baseados em condições críticas; o **Device Management Service**, que lida com a gestão e configuração de dispositivos; e o **Weather Integration Service**, que integra os dados de previsão do tempo para otimizar a irrigação.

Além disso, há um pipeline de **Processamento em Batch** para a coleta, limpeza, agregação e análise de dados, e geração de relatórios. Para as **Análises Preditivas**, o backend conta com um módulo simples que realiza treinamentos de modelos, geração de previsões, engenharia de features e gestão de um registro de modelos.

2.4 Camada de Dados

1. **Time Series DB:** Utiliza-se um banco de dados de séries temporais, o Timescale DB, com políticas de retenção de dados configuradas e downsampling automático para otimizar o armazenamento.
2. **Data Lake:** O data lake armazena dados brutos, limpos, features processadas, modelos treinados e relatórios gerados. É utilizado como base para análises e armazenamento de longo prazo.
3. **Cache Distribuído:** O Redis é utilizado para gerenciar o cache de dados de sensores, configuração e autenticação, melhorando a eficiência do sistema.
4. **Banco Relacional:** Armazena informações estruturadas, como dados de usuários, dispositivos, culturas, configurações e alertas.

2.5 Camada de Apresentação

1. **App Mobile:** O aplicativo móvel é desenvolvido em React Native, com suporte a cache offline, notificações push e integração com geolocalização.

3. Aspectos Técnicos

3.1 Segurança

- JWT para autenticação
- HTTPS/TLS para todas comunicações
- Certificados X.509 para dispositivos
- Criptografia em repouso
- Firewall por camada
- IDS/IPS

3.2 Escalabilidade

- Containers Docker
- Orquestração Kubernetes
- Auto-scaling horizontal
- Particionamento de dados
- Cache distribuído

3.3 Disponibilidade

- Multi-zona
- Failover automático
- Health checks
- Circuit breakers
- Retry policies
- Backup/Restore

3.4 Monitoramento

- Métricas de negócio
- Métricas técnicas
- Logs centralizados
- APM
- Alerting
-