UNIVERSIDADE DE ÉVORA
CURSO DE ENGENHARIA INFORMÁTICA
2024/2025

Relatório de Trabalho de Sistemas Distribuídos

Miguel Aleixo - 51653
Pedro Freire - 52215

Objetivo

O objetivo deste trabalho foi desenvolver um sistema distribuído para monitorização ambiental no Novo Hospital de Évora. Este sistema visa garantir o controlo eficiente da temperatura e humidade em diversas áreas do hospital, assegurando o conforto, a preservação de equipamentos sensíveis e o cumprimento de normas de controlo ambiental. O sistema inclui a receção de dados de dispositivos IoT, processamento, armazenamento e a disponibilização de funcionalidades para consulta e gestão.

Descrição do Sistema

Arquitetura Geral

A solução proposta foi implementada como um sistema distribuído baseado em microserviços, utilizando um broker MQTT para comunicação entre dispositivos IoT e o servidor. A base de dados utilizada foi PostgreSQL, e a API RESTful permitiu a interação com os dados armazenados.

Persistência em Base de Dados

Entidades e Estrutura

Foram desenvolvidas as seguintes entidades para garantir a persistência dos dados:

1. **IoTDevice**
    - Representa os dispositivos IoT registados no sistema.
    - Atributos principais: `id`, `nome`, `sala`, `serviço`, `piso`, `edifício`.
2. **Metricas**
    - Representa as métricas de temperatura e humidade.
    - Atributos principais: `id`, `dispositivo`, `temperatura`, `humidade`, `timestamp`.
3. **Users**
    - Representa os utilizadores do sistema, incluindo administradores.
    - Atributos principais: `id`, `username`, `password`, `role`.

Serviços de Persistência

1. **IoTDeviceService**
    - Métodos principais: `saveDevice`, `findDeviceById`, `updateDevice`, `deleteDevice`.
2. **MetricaService**
    - Métodos principais: `saveMetric`, `findMetricsByDevice`, `getAggregatedMetrics`.

Componentes do Sistema

Controladores

IoTDeviceController

- Endpoint para gerir dispositivos IoT.
- Métodos: `registerDevice`, `listDevices`, `updateDevice`, `deleteDevice`.

MetricaController

- Endpoint para gerir métricas ambientais.
- Métodos: `submitMetric`, `getMetrics`, `filterMetrics`.

Menu de Métricas

MetricMenuController

- Facilita a interação do utilizador com o sistema para consultar métricas de forma estruturada.
- Permite filtrar por sala, serviço, piso e edifício.

AdminClient

Funções Administrativas

- Gerir permissões de utilizadores.
- Consultar e aprovar dispositivos IoT.
- Operações protegidas por autenticação robusta.

Simulação de Dispositivos

Sim.java

- Classe que simula o envio de dados por dispositivos IoT.
- Gera métricas realistas de temperatura e humidade de forma periódica.

Listener.java

- Classe que consome mensagens do broker MQTT e valida os dados antes de enviá-los para o servidor.

Desenvolvimento

Fluxo de Dados

1. Dispositivos IoT enviam métricas para o broker MQTT.
2. O `Listener.java` consome as mensagens e verifica a validade dos dispositivos.
3. Dados válidos são armazenados na base de dados através do `MetricaService`.
4. O `MetricMenuController` permite a consulta das métricas processadas.

API RESTful

Foram implementados endpoints para gerir dispositivos e métricas. Os controladores `IoTDeviceController` e `MetricaController` garantem a interação do cliente com o sistema de forma segura e eficiente.

Justificação das Escolhas

- **Broker MQTT**: Leve e ideal para comunicação entre dispositivos IoT.
- **PostgreSQL**: Adequado para armazenamento seguro e consultas complexas.
- **Sim.java**: Garante a geração de dados realistas para simulação.
- **Listener.java**: Essencial para validar e encaminhar dados ao servidor.

Observações sobre o Desenvolvimento

Durante o desenvolvimento, enfrentámos desafios como:

- Implementar a logica do broker, algo que revelou se mais dificil do que inicialmente esperado.
- O parelismo de execução da parte de emissão de metricas e da parte da consulta destas.
- A aplicação da segurança dos endpoints.

Apesar dos desafios, os objetivos foram alcançados com sucesso.

Execução

Para executar o projeto utilizamos o seguinte comando:

mvn spring-boot:run

Para iniciar a base de dados utilizamos:

sudo psql bd1 -U user1 -h localhost

Para iniciar o broker:

./activemq start

Conclusão

Este trabalho permitiu aprofundar conhecimentos sobre sistemas distribuídos, em particular no contexto da integração de dispositivos IoT. Além disso, proporcionou a oportunidade de aplicar boas práticas no desenvolvimento de software, como a

utilização de microserviços, a segurança de comunicação e a implementação de APIs RESTful.