UNIVERSIDADE DE ÉVORA

UNIVERSIDADE DE ÉVORA

CURSO DE ENGENHARIA INFORMÁTICA

Sistemas Distribuidos- 2º Trabalho

Fidelis Silva nº 54761
Enoque Massau nº 53235

14 de Janeiro de 2025

---

1. Introdução

O desenvolvimento de sistemas distribuídos para a Internet das Coisas (IoT) tem se tornado essencial em ambientes complexos, como hospitais, onde o monitoramento constante de variáveis ambientais é crucial para garantir a segurança, o conforto e a eficiência operacional. Este relatório descreve o projeto de um Sistema de Monitorização Ambiental especificamente projetado para o novo Hospital de Évora.

O objetivo principal deste sistema é monitorizar e gerir a temperatura e a humidade em diversas áreas do hospital. Para isso, o sistema utiliza dispositivos IoT para capturar dados em tempo real, que são posteriormente processados, armazenados e disponibilizados para consulta e gestão através de uma interface RESTful.

O projeto incorpora várias tecnologias modernas, como Spring Boot para a construção do servidor, PostgreSQL para o armazenamento de dados e MQTT como protocolo de comunicação para os dispositivos IoT. O sistema também inclui um mecanismo de autenticação baseado em tokens JWT para proteger o acesso às funcionalidades críticas.

Este relatório documenta os principais componentes do sistema, detalhando os desafios enfrentados e as soluções implementadas. A integração de múltiplos módulos, incluindo autenticação, gestão de dispositivos IoT, processamento de dados e consultas agregadas, reflete a complexidade e a robustez do sistema desenvolvido para atender às necessidades específicas do Hospital de Évora.

---

2. Metodologia - Planeamento e Definição de Requisitos

Identificação dos Requisitos

O projeto iniciou-se com a identificação dos requisitos funcionais e não funcionais, alinhados à necessidade de monitorizar a temperatura e a humidade em diversas áreas do hospital descritos no enunciado.

Requisitos Funcionais:

- **Monitorização Ambiental**: O sistema deve receber e armazenar dados de temperatura e humidade provenientes de dispositivos IoT instalados nas diferentes áreas do hospital.
- **Gestão de Dispositivos IoT**: Deve permitir o registo, consulta, atualização e remoção de dispositivos IoT, garantindo a associação destes a áreas específicas do hospital.
- **Receção de Dados via MQTT**: Os dispositivos IoT enviam dados de monitorização para o sistema através de um broker MQTT.
- **Consulta de Métricas Ambientais**: Disponibilização de métricas de temperatura e humidade para consulta, com filtros de localização e intervalo de datas.

Requisitos Não Funcionais:

- **Segurança**: Implementação de mecanismos de autenticação e autorização.
- **Confiabilidade**: Garantir a integridade e consistência dos dados recebidos e armazenados.

Análise de Casos de Uso

Com base nos requisitos especificados, os casos de uso principais foram definidos:

- **Gestão de Dispositivos IoT**: Permite que administradores registrem, atualizem, consultem e removam dispositivos IoT.
- **Receção de Dados via MQTT**: Captura dados de temperatura, humidade e timestamp enviados pelos dispositivos IoT para o broker MQTT.
- **Consulta de Métricas Ambientais**: Permite aos utilizadores consultarem dados agregados de temperatura e humidade, com filtragem por sala, serviço, piso ou edifício e intervalo de datas.

---

3. Implementação

3.1. Configuração do Ambiente

- **Servidor/Database**: Foi criado um arquivo `.properties` para configurar o servidor e a base de dados, utilizando Spring Boot.
- **Dependências**: Geridas pelo Maven, com bibliotecas essenciais como Google Gson para manipulação de JSON e Java HttpClient para comunicação HTTP.
- **Banco de Dados**: Utilizou-se PostgreSQL para persistência dos dados dos dispositivos e métricas.

3.2. Arquitetura do Sistema

A arquitetura foi projetada de forma modular, dividida em pacotes para facilitar a manutenção e a expansão. Os principais pacotes e suas funcionalidades são descritos abaixo:

- **Pacote com.iotproject.admin**: Responsável pela lógica administrativa da aplicação, contendo subpacotes para autenticação, gestão de dispositivos IoT, consulta de métricas e utilitários.
    - **com.iotproject.admin.auth**: Gestão da autenticação de administradores, incluindo validação de credenciais e controle de acesso.
    - **com.iotproject.admin.device**: Gestão das operações de dispositivos IoT, como registro e atualizações.
    - **com.iotproject.admin.metric**: Gerencia a consulta e processamento de métricas ambientais.
    - **com.iotproject.admin.util**: Contém utilitários auxiliares que facilitam operações comuns.

- **Pacote com.iotproject.config**: Contém classes de configuração, como `AdminUserInitializer.java` e `DataInitializer.java`, que inicializam o sistema e os dados básicos.
- **Pacote com.iotproject.controller**: Exposição de endpoints RESTful para interação com o cliente.
- **Pacote com.iotproject.model**: Define as entidades principais do sistema, como dispositivos, métricas e usuários.
- **Pacote com.iotproject.repository**: Interfaces JPA para persistência de dados.
- **Pacote com.iotproject.service**: Lógica de negócios ou serviços e regras de processamento de dados.
- **Pacote com.iotproject.mqtt**: Comunicação com dispositivos IoT via MQTT.
- **Pacote com.iotproject.security**: Implementação de segurança, com autenticação e autorização usando JWT.
- **Pacote com.iotproject.payload**: Classes de transferência de dados entre cliente e servidor.

---

4. Justificativas das Escolhas

Escolha do Protocolo MQTT para Comunicação IoT

O protocolo MQTT foi escolhido para a comunicação entre os dispositivos IoT e o servidor devido às suas características específicas que o tornam ideal para sistemas distribuídos com muitos dispositivos, como o nosso caso. O MQTT é um protocolo de mensagens leve e eficiente, projetado para funcionar de forma confiável mesmo em redes de baixa largura de banda e com dispositivos de recursos limitados. Além disso, ele é altamente escalável, permitindo uma comunicação eficaz em tempo real, o que é crucial em um sistema de monitorização.

Uma das vantagens do MQTT é a sua arquitetura de publicação/assinatura, que facilita a receção de dados de múltiplos dispositivos IoT de forma centralizada e eficiente. A utilização de um broker MQTT centralizado facilita a gestão de mensagens, garantindo que os dados dos sensores sejam transmitidos de forma contínua para o servidor sem sobrecarregar a rede.

Escolha do Banco de Dados PostgreSQL

Optamos por utilizar PostgreSQL como base de dados devido à sua fiabilidade, confiabilidade e escalabilidade demonstradas em projetos anteriores. O PostgreSQL é uma base de dados relacional open-source que oferece excelente desempenho, capacidade de lidar com grandes volumes de dados e é altamente compatível com tecnologias como o Spring Boot, que foi utilizado para construir o servidor do sistema.

Além disso, o PostgreSQL oferece suporte completo a transações, o que garante a integridade dos dados, além de ser bem documentado e amplamente utilizado no mercado. Esta escolha facilita a manutenção do sistema, garantindo que as métricas ambientais e os dados dos dispositivos IoT sejam armazenados de forma segura e eficiente. A escolha do JPA (Java Persistence API) para interagir com o banco de dados também foi uma decisão

mecionada pelo o professor que visou simplificar a criação e manutenção das tabelas, além de permitir que a estrutura da base de dados seja facilmente alterada quando necessário.

Escolha do JWT para Autenticação e Autorização

No início do projeto, foi considerado o uso do Basic Security do Spring Boot, que funciona com cookies. No entanto, durante os testes, verificou-se que o sistema de cookies não era ideal para garantir a segurança desejada, pois os cookies não eram eliminados corretamente ao encerrar a sessão no navegador ou no Insomnia. Isso implicava que, caso um usuário deixasse a sessão aberta e o cookie permanecesse na máquina, seria possível acessar os endpoints do sistema sem a devida autenticação.

Por essa razão, optamos por utilizar JWT (JSON Web Tokens) para a autenticação e autorização no sistema. O JWT oferece uma abordagem mais robusta, onde os tokens são gerados após o login do usuário e contêm as informações de autenticação e autorização. Estes tokens são enviados em cada requisição, garantindo que o usuário esteja autenticado e autorizado a acessar os endpoints do sistema.

Fluxo de Execução

1. O usuário configura a base de dados no arquivo `application.properties`, incluindo as credenciais do banco de dados (usuário, senha e etc).
2. Após configurar, o usuário executa o comando `mvn clean install` para preparar o projeto e, em seguida, executa `mvn spring-boot:run` para iniciar o servidor.
3. O servidor é executado, o usuário é criado e a base de dados é conectada com sucesso.
4. O usuário pode então executar o `AdminClient com (“ java -cp target/IOT_PROJECT-1.0-SNAPSHOT.jar com.iotproject.admin.AdminClient”)`, onde estará disponível todas as funcionalidades mencionadas, como o registo de dispositivos e a simulação do MQTT.
5. O simulador de dispositivos (MQTT Listener) é executado juntamente com o Spring Boot, ficando à escuta do tópico esperado para receber os dados dos dispositivos IoT.

---

5. Conclusão

O projeto implementado para o novo Hospital de Évora mostra-se robusto e eficiente, utilizando tecnologias modernas como MQTT, PostgreSQL e JWT. A solução proposta garante a monitorização e gestão eficiente das condições ambientais no hospital, oferecendo um certo nível de segurança e escalabilidade para suportar um número crescente de dispositivos IoT. O desenvolvimento modular e as escolhas tecnológicas feitas ao longo do projeto permitem que o sistema seja facilmente mantido e expandido conforme necessário.Este projeto reflete uma abordagem profissional, utilizando as melhores práticasde desenvolvimento de sistemas distribuídos, garantindo a implementação de um sistema de monitorização ambiental eficiente e seguro para as necessidades do Hospital de Évora.