Sistemas de Monitorização e Controlo em Tempo Real

1. Introdução

A atividade laboratorial tem como propósito o desenvolvimento de sistemas de monitorização e controlo em tempo real, recorrendo à implementação de mecanismos de streaming de dados, bases de dados de séries temporais, técnicas de processamento de dados em tempo real e utilização de ferramentas de visualização de dados.

O nosso cenário de implementação consiste na monitorização da qualidade do solo. Pretende-se simular a operação de um sensor de qualidade do solo (https://www.jxct-iot.com/product/showproduct.php?id=197), que recolhe os seguintes parâmetros: temperatura, humidade, pH, condutividade elétrica (EC) e níveis de macronutrientes (NPK – azoto, fósforo e potássio). Os dados são transmitidos em tempo real pelo RabbitMQ e posteriormente, processados e armazenados pelo InfluxDB. O Grafana possibilitará a visualização dos dados e a avaliação do seu comportamento ao longo do tempo, através de uma dashboard web, semelhante à da Figura 1.

Figura 1. Dashboard web de monitorização da qualidade do solo.

2. Objetivos e competências a desenvolver

Ao concluir esta atividade laboratorial, os estudantes deverão ser capazes de:

- Compreender os conceitos e componentes de sistemas de streaming de dados e processamento de eventos.
- Configurar e integrar bases de dados de séries temporais para armazenar dados com marcações temporais.
- Utilizar ferramentas de visualização de dados para análise em tempo real.

- Desenvolver capacidades de construir uma infraestrutura de monitorização integrada.

3. Instruções
A atividade será dividida em três etapas:

Etapa 1: Revisão da literatura e estado da arte
Faça uma breve revisão da literatura e levantamento do estado da arte sobre os temas em exploração nesta atividade laboratorial e principais tecnologias associadas, nomeadamente: (1) Streaming de dados (Apache Kafka, RabbitMQ); (2) Bases de dados de séries temporais (InfluxDB, Prometheus); (3) Técnicas de processamento de dados em tempo real (Kapacitor); e (4) Visualização de dados (Grafana, Chronograf). Para cada um dos tópicos deverão ser apresentados os conceitos fundamentais, cenários de aplicação e principais tecnologias. O enquadramento teórico não deverá exceder as oito páginas.

Etapa 2: Configuração dos containers Docker para o RabbitMQ, InfluxDB e Grafana
1. Crie uma diretoria denominada `lp2425_lab06`, e aceda a essa diretoria.
2. Crie o ficheiro `docker-compose.yaml` para configurar o RabbitMQ, o InfluxDB e o Grafana.

```
version: '3'
services:
  rabbitmq:
    image: rabbitmq:3-management
    ports:
      - "5672:5672"
      - "15672:15672"
  influxdb:
    image: influxdb:latest
    ports:
      - "8086:8086"
    environment:
      - INFLUXDB_DB=soil_data
  grafana:
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    depends_on:
      - influxdb
```

3. Execute o Docker Compose através do comando `docker-compose up -d`.
4. Confirme se o RabbitMQ (`http://localhost:15672`; username: `guest`; password: `guest`), o InfluxDB (`http://localhost:8086`) e o Grafana (`http://localhost:3000`; username: `admin`; password: `admin`) foram corretamente instalados.
5. Configure o InfluxDB através da interface web com os seguintes dados:
   - Username: `admin`
   - Password: `lp2425lab06`
   - Organization: `soil_org`
   - Bucket: `soil_data`
6. Copie o API token gerado, será necessário para os passos 3.3 e 4.1.

Etapa 3: Envio dos dados em tempo real com o RabbitMQ e armazenamento com o InfluxDB

1. Na diretoria criada no passo 2.1., crie um ambiente virtual através do comando `python -m venv venv` e em seguida ative-o (`venv\Scripts\activate`).
2. Instale as dependências necessárias através do comando `pip install pika influxdb_client`.
3. Crie um script em Python (`simulate_data.py`) que simule a recolha de dados e enviar dados para o RabbitMQ. Utilize o ficheiro `rabbitmq_send.py` como ponto de partida, o qual apresenta um exemplo para onde é criada uma queue e o envio de uma mensagem utilizando o RabbitMQ. Este ficheiro deverá ser adaptado de modo a simular a coleta de dados de um sensor de monitorização do solo. Deverá ser feita uma nova coleta de dados e respetivo envio a cada cinco segundos.
4. Crie um outro script em Python (`consume_data.py`) para receber as mensagens do RabbitMQ e inseri-las no InfluxDB. Utilize os ficheiros `rabbitmq_receive.py` e `influxdb_write.py` como ponto de partida. O primeiro script cria um callback para que fica indefinidamente à espera da receção de novas mensagens, enquanto o segundo apresenta um exemplo para a armazenar dados no InfluxDB.
5. Em dois terminais, execute os scripts criados nos passos anteriores (3.3 e 3.4).

Etapa 3: Configuração da dashboard com o Grafana

1. Adicione o InfluxDB como data source no Grafana (`Connections > Data Sources`), com as seguintes configurações:
   - Data source: `influxdb`
   - Query language: `Flux`
   - URL: `http://influxdb:8086`
   - Auth: `No auth`
   - Organization: `soil_org`
   - Token: API Token obtido no passo 2.6.
   - Default Bucket: `soil_bucket`
2. Crie uma dashboard para visualizar as métricas dos dados do solo (Figura 1). Abaixo encontra-se um exemplo query para obter os dados da temperatura. Para cada um dos parâmetros deverão ser criados dois paneis, um Time Series e um Gauge. As unidades deverão ser configuradas em conformidade com o parâmetro, bem como deverão ser devidos thresholds de modo a fornecer informação contextual aos utilizadores.

```
from(bucket: "soil_data")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "soil_metrics")
  |> filter(fn: (r) => r["_field"] == "temperature")
```