**UNIVERSIDADE DE SÃO PAULO**
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
SSC0965 – STREAMING DE DADOS, MICROSSERVIÇOS E CONTÊINERES
PROF. JULIO CEZAR ESTRELLA

JULIO IGOR CASEMIRO OLIVEIRA – 11816139
VICTOR VIEIRA CUSTODIO REIS - 10734686

# PROJETO FINAL

SÃO CARLOS
2023

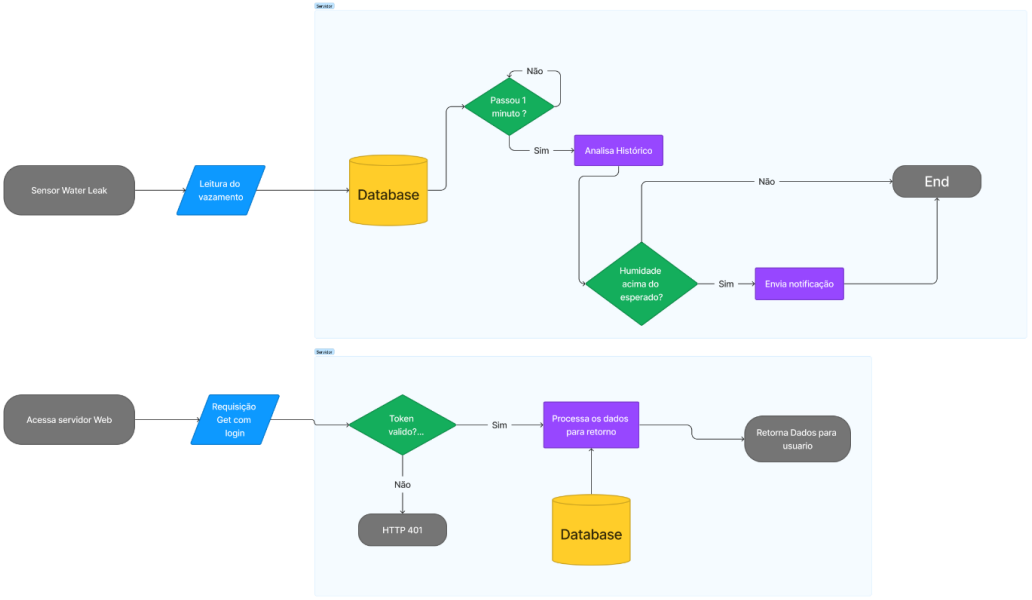
# Introdução

O presente relatório trata-se da documentação, descrição de tarefas e referências realizadas durante a execução do projeto da disciplina de SSC0965 - Streaming de dados, microsserviços e contêineres. A referida disciplina foi ministrada durante o segundo semestre de 2023, de forma presencial. A proposta inicial apresentada pelo professor trata-se da criação de uma aplicação integrada, no contexto de sistemas distribuídos. O foco era realizar a divisão dessa implementação entre quatro etapas diferentes: aplicação, broker, armazenamento e micro-serviço.

O tema do projeto vem do projeto de outra disciplina, de sigla SSC0952 - Internet das Coisas, essa ideia de tema vem de uma conversa que foi tida durante as aulas dada a presença de um dos membros do grupo nas duas disciplinas. Sendo assim, o trabalho desta disciplina acaba sendo complementar ao da outra.

Especificamente consiste em realizar o controle de vazamentos de água nos arredores das salas 1008 e 1006, sendo consideradas os laboratórios do grupo de pesquisa. Grande parte do trabalho foi realizado pelos integrantes do grupo, fisicamente, nesses laboratórios, utilizando da infraestrutura: mesas, lousa, ESP-32 e sensores. Trata-se de um evento incomum, porém, considerando a presença de vários dispositivos, aparelhos e componentes eletrônicos nos laboratórios, principalmente servidores e hosts de máquinas virtuais, é de extrema importância ficar atento caso ocorra um vazamento, para que profissionais responsáveis possam adotar soluções em tempo hábil, de modo a evitar acidentes e perda de material tecnológico. Além disso, a ocorrência desses vazamentos pode danificar estruturas do prédio e gerar demais transtornos.

O repositório da aplicação está disponível em <https://github.com/jul1co/gsmcgrad03>. Os detalhes de configuração e implantação estão na aba “Configuração” deste documento. O diagrama a seguir, ilustra o funcionamento do projeto.




# Referencial Teórico

Foi utilizado vários softwares e componentes para a realização deste projeto.

O broker para o projeto foi o Mosquitto, um servidor de mensagens MQTT, para se utilizar o protocolo que é muito bem definido para redes IoT. Também para o broker, foi usado a arquitetura Publisher/Subscriber para que o broker envie as mensagens quando tiver alguma disponível. Além disso, o Mosquitto é desenvolvido em código aberto.

Como banco de dados foi usado o PostgreSQL, por ser bem estável e que fornece suporte às diferentes funções SQL. Ele também é open source e possui um bom desempenho.

Para a técnicas de segurança foram usados o proxy para a autenticação do usuário, impedindo acesso não autorizado a terceiros e possível manipulação dos dados captados pelo sensor. Também foi implementado a criptografia para garantir que as informações fossem confiáveis.

# Desenvolvimento

- **Requisitos Funcionais**

  RF1 - O sistema deve permitir que um usuário realize cadastro no sistema. Os dados que devem ser utilizados são: nome completo, número USP, e-mail e número de telefone.

  RF2 - O sistema deve permitir que o administrador conceda acesso a usuários previamente cadastrados.

  RF3 - O sistema deve permitir que o administrador adicione mais sensores à rede IoT.

  RF4 - O sistema deve interpretar o nível de umidade do local especificado.

  RF5 - O sistema deve ser capaz de emitir um alerta de que um possível vazamento está ocorrendo.

  RF6 - O sistema deve notificar os usuários quando detectar um vazamento.

- **Requisitos de Qualidade**

  RQ1 - Uma requisição na aplicação deve ter um tempo de resposta de no máximo 5 segundos.

  RQ2 - O sistema não deve demorar mais de 1 minuto para interpretar um possível vazamento.

  RQ3 - O sistema deve estar disponível 24 horas por dia, 7 dias por semana.

  RQ4 - O sistema deve estar de acordo com a LGPD.

RQ5 - O sistema deve permitir uma comunicação segura entre sensores e gateway, protegendo contra intercepções e invasoẽs.

RQ6 - O sistema deve permitir que a interface web seja segura.

RQ7 - O sistema deve funcionar nos navegadores atuais.

RQ8 - O sistema deve ser responsivo, para que funcione bem também em dispositivos com telas menores.

RQ9 - O sistema deve ter um código-fonte bem documentado.
RQ10 - O sistema deve ter uma interface acessível e de fácil compreensão.

RQ11 - As cores da interface web devem ter um contraste agradável.

RQ12 - O sistema deve possuir recursos de prevenção de erros do usuário, como telas de confirmação.

- **Descrição dos Scripts**

O broker Mosquitto foi implementado em um docker, utilizando uma imagem disponibilizada pelos desenvolvedores do Mosquitto. Para o container, foi utilizado um arquivo de Docker Compose e, para configurar o broker, foi definido algumas diretivas , como bloquear o acesso anônimo, implementar autenticação e escutar as portas 7083 e 9001.

Na placa ESP32, foi implementado uma forma de conectar ao Wi-fi do laboratório, a lógica por trás da leitura dos dados do sensor (de forma que a placa lesse em intervalos constantes de tempo), uma forma de conectar ao broker e publicar uma mensagem a um tópico sempre que o sensor detectasse uma mudança de estado.

Também foi desenvolvido uma aplicação cliente que enviasse um email com a leitura interpretada pela placa. Foi desenvolvido uma interface para o usuário.

Foi implementada uma autenticação ao broker MQTT, tendo a função de salvaguardar quem poderia publicar ou se inscrever nos tópicos.

Diagram

## Configuração

Para realizar a reprodução de todo o projeto é preciso seguir alguns passos. Primeiramente, é necessário a utilização de servidores linux, preferencialmente as máquinas disponibilizadas pelo professor durante a disciplina. Além disso é indispensável o uso de uma ESP-32, acoplada ao sensor de umidade e em pleno funcionamento. Não se esqueça de certificar que o código disponível na pasta back-end está salvo e funcional em sua ESP-32 e que as leituras dos valores do sensor estão sendo realizadas corretamente.

Realize o acesso ao servidor, utilizando as credenciais necessárias. Clone o projeto em um diretório que seja de fácil acesso e realize o seguinte comando:

```
sudo docker-compose up
```

O docker-compose irá realizar a criação de três contêineres que serão necessários para o funcionamento da aplicação. São eles:

- frontend: servir a aplicação web responsável pelo consumo das informações disponíveis pelos sensores
- postgres: configurar corretamente a base de dados utilizada pela aplicação
- broker: realiza a configuração necessária para que o broker mqtt possa funcionar e assim permitir a comunicação entre as demais partes do projeto

Dessa forma, ao final do processo, terás todas as quatro peças que integram o projeto em funcionamento: aplicação, broker, armazenamento e micro-serviço.

## Conclusões

Em resumo, o projeto da disciplina SSC0965 - Streaming de dados, microsserviços e contêineres proporcionou uma abordagem abrangente na criação de uma aplicação integrada em sistemas distribuídos. Originado da interseção com a disciplina SSC0952 - Internet das Coisas, o trabalho focou no controle de vazamentos de água nos laboratórios, demonstrando uma aplicação prática dos conceitos teóricos.

Em síntese, a integração eficaz de teoria, prática e tecnologia resultou em uma solução sólida para o monitoramento de vazamentos de água, apesar de toda a simplicidade o projeto tem a sua utilidade. O repositório no GitHub complementa a documentação, oferecendo uma referência adicional para interessados.

# Referências


- Slides do professor Julio Cezar Estrella feitos para a disciplina “SSC0952 - Internet
- das Coisas (2023)”;
- Documento de requisitos elaborado pela profa. Rosana Teresinha Vaccare Braga para a disciplina de Análise e Projeto Orientados a Objetos (2021);
- Slides da profa. Elisa Yumi Nakagawa para a disciplina de Engenharia de Software (2022);
- https://randomnerdtutorials.com/getting-started-with-esp32/
- https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino-ide/;
- https://randomnerdtutorials.com/testing-mosquitto-broker-and-client-on-raspbbery-pi/;
- https://test.mosquitto.org/;
- https://esp32io.com/tutorials/esp32-rain-sensor;
- https://www.youtube.com/watch?v=5rHWeV0dwxo;
- https://www.youtube.com/watch?v=uXDRK6XNPNQ;
- https://www.youtube.com/watch?v=whHwoxVqJBE.
- https://www.tutorialworks.com/container-networking/
- https://github.com/eclipse/paho.mqtt.javascript?utm_source=cdnjs&utm_medium=cdnjs_link&utm_campaign=cdnjs_library
- https://www.emqx.com/en/blog/connect-to-mqtt-broker-with-websocket#get-started-with-mqtt-over-websocket