SSC0952: Internet das Coisas

Solução funcional de aplicação integrada (software e hardware)

Dezembro/2020

Time 1

Grupo 1

Eduardo Molina
Fernanda Federici
Vinicius Ribeiro
Luiz Adorno

Grupo 2

Breno Cunha Queiroz
Guilherme Mota Petrucci
Henrique Hiram Libutti Núñez
Natan Bernardi Cerdeira

Grupo 7

Bruno Germano
Dennis Lemke Green
Matheus Tomieiro de Oliveira
Matheus Lopes Rigato

Grupo 12

Eduardo Zaboto
Paulo Bodnarchuki
Pedro Oliveira
Victor Pereira
Vinicius Genésio

Grupo 22

Ana Clara Amorim Andrade
Lucas Yuji Matubara
Matheus Godoy Bolsarini
Pedro Pastorello Fernandes
Vinicius Eduardo de Araujo

Introdução

- Advento da computação
  - Integração de dispositivos à rede;
  - Surge o conceito de IoT;
  - Interligar dispositivos e sensores de forma a monitorá-los e controlá-los.
- Projeto prático para controlar dispositivos em uma sala no ICMC/USP
  - Monitorar sensores diversos
  - Controlar temperatura de ares-condicionados presentes
  - Tratar o fluxo de dados, desde os sensores, até a aplicação.

Justificativa

- A teoria trabalhada em aula nem sempre é suficiente para a compreensão plena;
- Trabalho prático é essencial;
- Aplicar os conceitos no mundo real traz consigo complicações;
- As complicações são necessárias para compreender os problemas reais;
- Por ser um assunto tão atual, este trabalho é valioso para a nossa futura experiência na academia e/ou no mercado de trabalho.

Objetivos do projeto

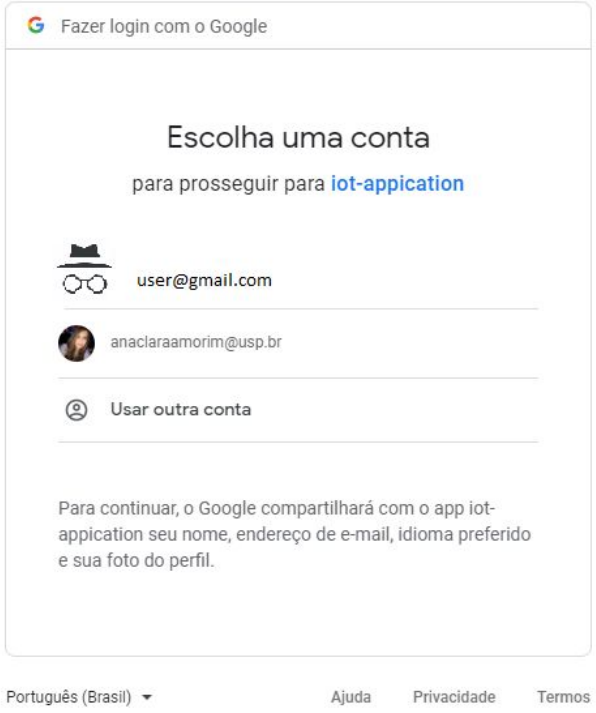
- Apresentar e desenvolver as etapas de criação de uma arquitetura em IoT
- Estudar e conhecer as etapas fundamentais desta arquitetura:
  - Aplicação
  - Broker
  - Micro-serviço
  - Armazenamento
  - Segurança

Metodologia: **Aplicação**

- ➢ Para o desenvolvimento da aplicação criamos uma interface de login para o usuário e outra para alterar as possíveis configurações do ar condicionado.
- ➢ Utilizamos o NodeJS, Express, e MQTT.js para o back-end, que serve a pasta /public/ e algumas rotas GET e POST para possibilitar a interação do front-end com o servidor.
- ➢ Já para o front-end, utilizamos o HTML, CSS, JavaScript, e o ajax.

- ❖ Na interface criamos funções no JavaScript para:
  - acesso restrito a usuários com domínio de email **@usp.br**
  - apresentar o display do ar condicionado
  - congelar a tela para ter um tempo de resposta do broker
  - validar os parâmetros de temperatura
  - apresentar respostas para os comandos enviados

Front-end: **Aplicação**



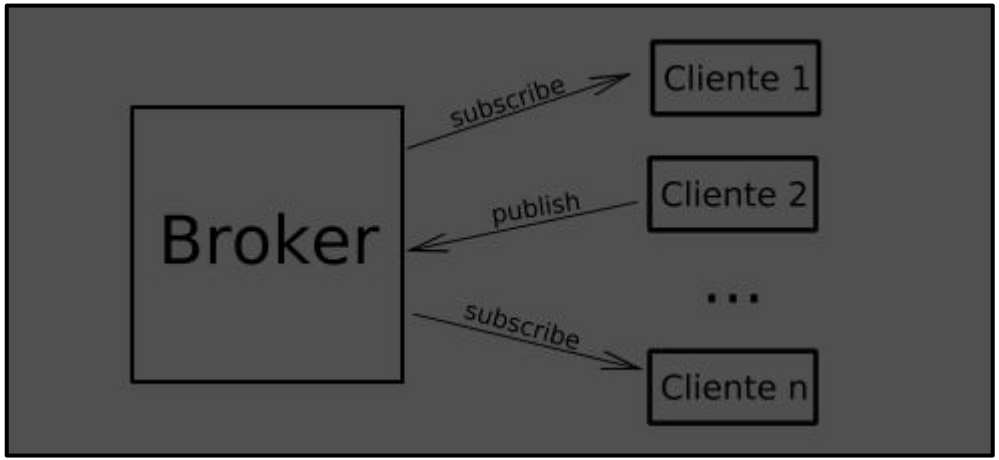
Front-end: **Aplicação**

Back-end: **Aplicação**

- O servidor foi desenvolvido em NodeJS + ExpressJS
- Serve requisições utilizando HTTPS
- Sistema de login, sessão e autenticação
- Foram criados semáforos utilizando *promises*, *async* e *await* para sincronizar mensagens e respostas

Metodologia: **Broker**

- Escolha do MQTT como protocolo;
- Repassa mensagens publicadas em um tópicos para assinantes daquele tópico;
- Tópicos não são fixos, e os próprios clientes escolhem o tópico qual assinam ou publicam.



Metodologia: **Broker**

- Utilização do Mosquitto
  - Configurações simplificadas;
  - Documentação abundante;
  - Baixa curva de aprendizado.

Metodologia: **Broker**

- Configuração do Mosquitto
  - Instalação em uma VM do ICMC;
  - Roda como um serviço no systemd.
  - Criação de um link simbólico para configurações.

```
pid_file /var/run/mosquitto.pid
persistence true
persistence_location /var/lib/mosquitto/
log_dest file /var/log/mosquitto/mosquitto.log
include_dir /etc/mosquitto/conf.d


#Configuracoes especificas
port 1821
allow_anonymous false
```

Endereço do Broker:
andromeda.lasdpc.icmc.usp.br

Porta:
8021

Metodologia: **Broker**

- Segurança
  - Autenticação por meio de usuário e senha
  - Criptografia utilizando TLS (Self-Signed)

```
#Autenticacao
password_file /home/ssc952-t1/broker/users/users.passwd

#Criptografia
cafile /home/ssc952-t1/broker/cert/ca.crt
certfile /home/ssc952-t1/broker/cert/server.crt
keyfile /home/ssc952-t1/broker/cert/server.key
require_certificate true
```

Metodologia: **Broker**

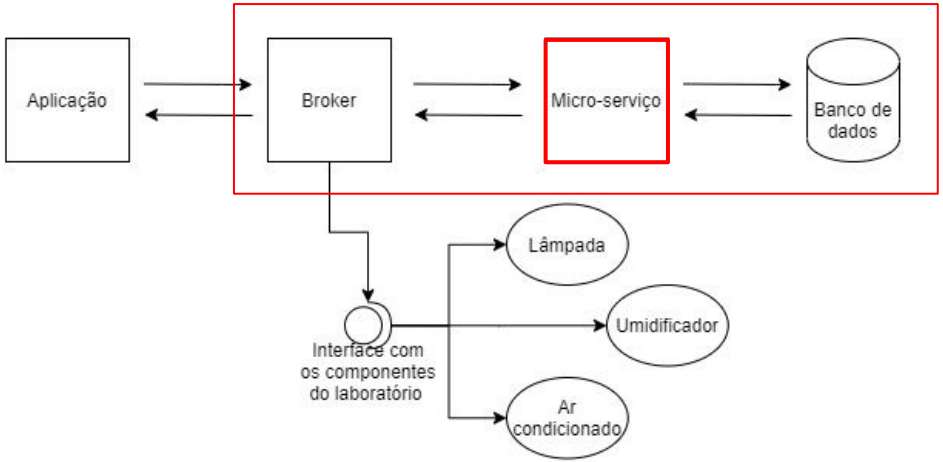
- Fácil obtenção dos certificados para uso dos clientes;

```
ssc952-t1@tau02-vm1:~/broker/cert/get$ ls
ca.crt  client.crt  client.key

$ scp -P 2321 ssc952-t1@andromeda.lasdpc.icmc.usp.br:broker/cert/get/* .
```

Metodologia: **Micro serviço**

O **funcionamento** do micro serviço está baseado nas **integrações** necessárias para que as mensagens recebidas do **broker** passem pelo micro serviço e sejam armazenadas no **banco de dados**.



Metodologia: **Micro serviço**

As etapas para que essa integração seja realizada são:

1. Carregamento das informações de configuração

   Caminhos para arquivos de segurança (certificados), usuário e senha do broker, além de host e porta em que o broker está alocado.

   CONEXÃO COM O BANCO DE DADOS

2. Utilização do método *connect* passando como argumento o nome do database

   E em caso de deploy para produção, passamos também o usuário, senha e host do banco.

Metodologia: **Micro serviço**

CONEXÃO COM O BROKER

3. Com a utilização da biblioteca Paho, estabelecemos callbacks de on_connect, on_message e on_disconnect.

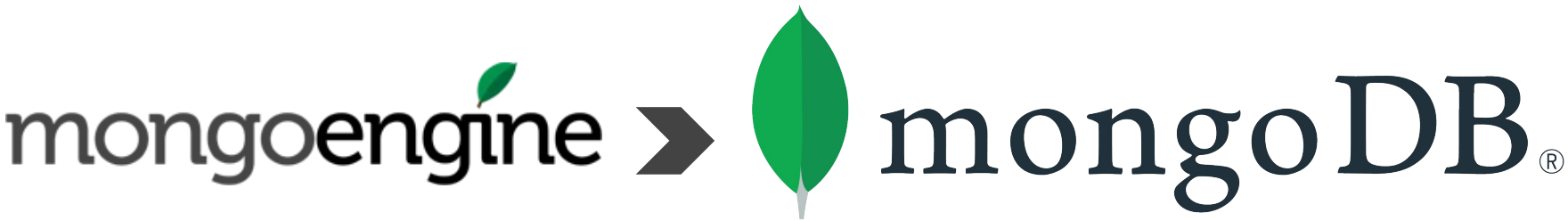
   Conseguimos manipular como as ações serão tratadas e com isso armazenar os dados importantes no banco de dados.

AO FINAL DESTAS 3 ETAPAS O MICRO SERVIÇO JÁ ESTÁ RODANDO!

Por fim, utilizamos a biblioteca logging para termos o log de todas as informações.

Metodologia: **Armazenamento**

- Utilizou-se na VM3, o software de banco de dados orientado a documentos MongoDB, conectado à biblioteca *mongoengine* no módulo de microsserviços;
- A base de dados consiste em um registro de mensagens compostas por um campo “Timestamp”, um campo “Action” – utilizado para distinguir mensagens de nova conexão, desconexão, ou inserção de dados –, e um campo “Payload”, referente a qualquer informação a mais da ação, como a mensagem enviada para o acionamento de um sensor.



Metodologia: **Armazenamento**

logs Collection

```
timestamp: <integer>
action:    <string>
payload:   <document>
```

Exemplo:

```
{
    _id: <ObjectId>,
    timestamp: 1536075328,
    action: "connect",
    payload: {}
}
```

- O terceiro campo, quando não estiver vazio (como no exemplo de conexão ao lado), é composto por um tópico e um payload, no seguinte formato: {'topic': msg_topic, 'payload': msg_payload};
- A *msg_payload* chega criptografada (base64) do módulo de microsserviços;
- Também há autenticação de usuário (*time1*, com permissão de leitura e escrita na *dbTime1*) e senha;
- Para conectar ao mongoshell pelo terminal, temos: `mongo -u "time1" --authenticationDatabase "dbTime1" -p`

Metodologia: **Segurança**

A segurança foi projetada para esta solução nas seguintes instâncias:

- Criptografia na comunicação entre as partes (aplicação, broker, dispositivos IoT).
- Criptografia das informações armazenadas (Banco de dados).
- Validação da entrada do usuário no lado da aplicação.

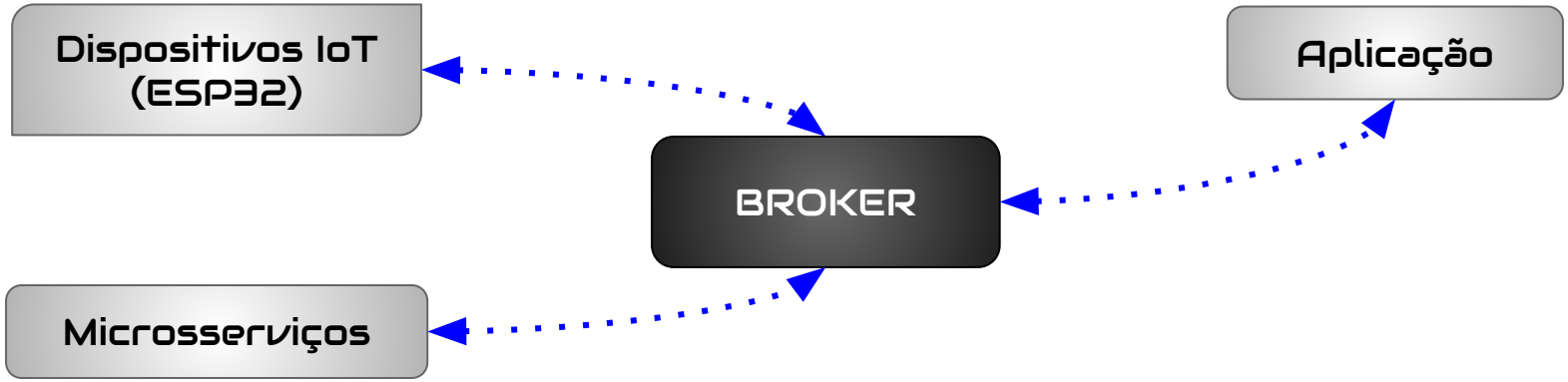
Metodologia: **Segurança**

Segurança entre Broker e clientes

- Broker e clientes utilizam o MQTT com uma camada de segurança oferecida por Tunneled Layer Security (TLS).
- O Broker exige uma certificação de autenticidade dos clientes, o que impede que qualquer cliente se conecte e se inscreva em tópicos.

Metodologia: **Segurança**

Segurança entre Broker e clientes

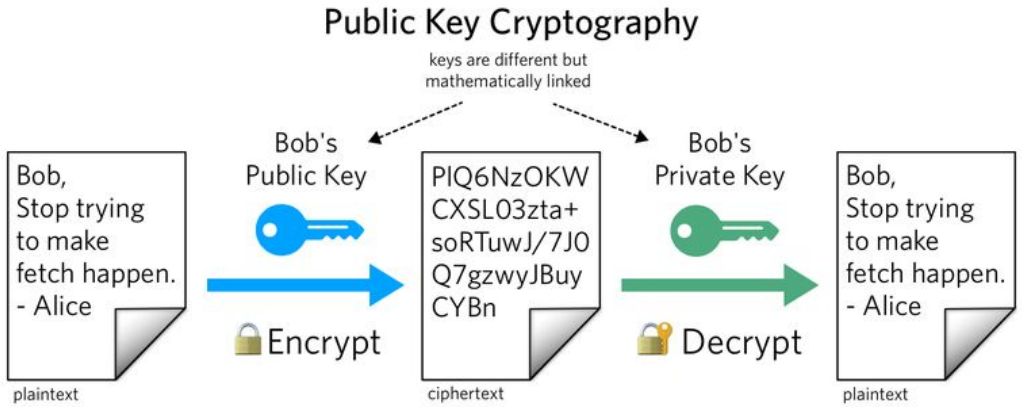


Representa uma comunicação
com criptografia e autenticação.

Metodologia: **Segurança**

Criptografia das informações armazenadas (Banco de dados)

Todas as mensagens armazenadas no banco de dados estão sendo encriptadas utilizando RSA. Desta forma caso alguém externo consiga acessar o banco de dados, os dados ainda estarão protegidos.



Metodologia: **Segurança**

Segurança entre usuário final e aplicação

Todas as entradas do usuário para controle do ar condicionado são validadas antes de serem enviadas para o broker. Esta checagem de dados válidos (dentro do intervalo permitido de temperatura), ocorre tanto no lado do cliente, como do servidor que hospeda aplicação.

A comunicação entre o dispositivo do usuário final e o servidor web da aplicação utiliza HTTPS.

Resultados e Discussões

Conclusão

- Implementação prática traz consigo problemas reais.
- Para uma boa integração, a comunicação entre os grupos foi essencial.
- Uma boa modularização é crucial para aumentar a eficiência de todo o projeto.
- O tratamento da segurança dos dados é indispensável em projetos reais.

Agradecimentos

- Ao professor Julio Cezar Estrella, por nos acompanhar e ensinar esse semestre.
- A nossos familiares, pais e irmãos, pelo suporte nas dificuldades da atual situação.
- A cada integrante dos grupos, que colaboraram para a finalização desse projeto.