UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Gustav Shigueo Nicioka Asano 11212355
Marcos Antonio Nobre Coutinho 10716397
Pedro Fernando Christofoletti dos Santos 11218560
Altair Fernando Pereira Junior 9391831

Publicar leituras de temperatura e umidade de um sensor DHT11 via MQTT com o ESP32

São Carlos
2022

RESUMO

Em suma, para desenvolver o projeto de leituras de temperatura e umidade de um sensor DHT11 foi escolhido a interface e web como endpoint para que o aplicativo possa ser acessado por qualquer dispositivo que possua navegador instalado. Portanto, a aplicação foi desenvolvida utilizando o ReactJS para a interface e o NodeJS para o backend.

Assim, a aplicação foi desenvolvida de modo que o DHT11 envia sinais de temperatura e umidade para o ESP32, onde esses sinais são publicados em um tópico específico na corretora Mosquitto, broker que usa o protocolo MQTT e serve como mediador para que a comunicação ocorra. É através dessa mediação que os dados da leitura de temperatura e umidade chegam a aplicação NodeJS que por fim faz a comunicação para ser impressa na aplicação em ReactJS.

Os códigos utilizados nesse trabalho pode ser encontrado no gitlab a partir do endereço: https://gitlab.com/icmc-ssc0952/2022/giotgrad09

1 INTRODUÇÃO

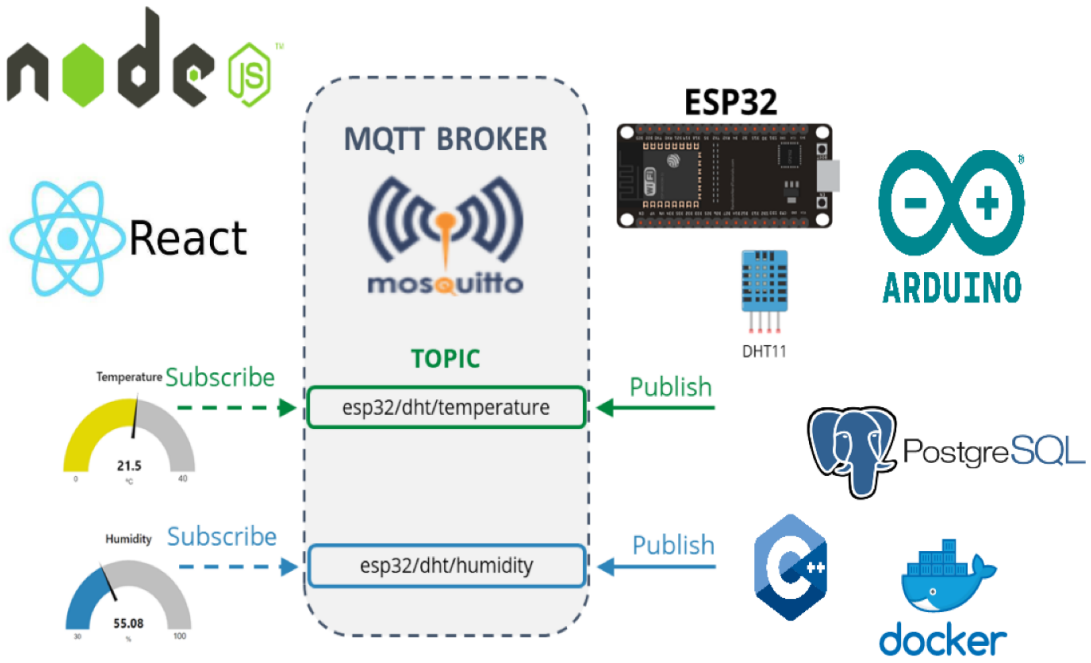
Como proposto na disciplina SSC0952 - Internet das Coisas, esse projeto une hardware + software para medir e conferir a temperatura e umidade de uma sala do ICMC. Foi disponibilizado pelo professor uma máquina virtual e um ESP32, que é uma série de microcontroladores de baixo custo e baixo consumo de energia com microcontrolador integrado, Wi-Fi e Bluetooth. A série ESP32 emprega um microprocessador Tensilica Xtensa LX6 com duas variações dual-core e single-core e inclui uma antena integrada RF tipo balun, amplificador de potência, receptor de baixo ruído amplificado, filtros, gerenciamento de energia dos módulos.

Para começar, temos a série ESP32 com um sensor de temperatura e umidade do tipo DHT11 acoplado e usando uma alimentação de 5V 2.5A (carregador de celular). Usamos a plataforma Arduino IDE em linguagem C++ em conjunto com as bibliotecas: Async MQTT Client Library e DHT library from Adafruit para publicar as leituras do sensor em dois tópicos (umidade e temperatura).

Para isso, foi instalado o Mosquitto Broker na máquina virtual disponibilizada pelo professor (LaSDPC). Também usamos SGDB PostgreSQL para armazenamento das informações de usuário e logs. Quanto ao responsável por ler os tópicos e executar queries no banco, temos uma api em Node Js utilizando o framework Express.js que lê os dois tópicos do broker e que também conta com um sistema de autenticação utilizando da tecnologia RFC do JSON Web Token, bem como o query builder Knex para execução das queries no banco. Essa API é consumida por uma interface web.

Por fim, toda essa estrutura está dentro de um container do Docker.

Figura 1 - Tecnologias utilizadas para o projeto



2 APLICAÇÃO

A interface escolhida é a web, pois assim a aplicação pode ser acessada por qualquer dispositivo que possua um browser instalado (computadores, smartphones, tablets, TVs, etc.). Utilizaremos a biblioteca de javascript React.js, seguindo o padrão “Single Page Application”, também contaremos com o conjunto de pacotes da Material UI do Google para o design.

O DHT11 enviará sinais de temperatura e umidade para a ESP32, e estes sinais serão publicados num tópico específico no broker do Mosquitto. O nosso back-end em NodeJS, que estará consumindo as mensagens publicadas no tópico, poderá ler essas informações e enviar para o nosso front-end em ReactJS para que os usuários autenticados possam ter conhecimento dessas informações.

3 BROKER

O Mosquitto é um broker leve que pode ser utilizado em microcontroladores até grandes computadores. Ele utiliza o protocolo MQTT e servirá como mediador para que a comunicação aconteça, sendo responsável em receber, filtrar, decidir quem precisa e publicar as mensagens para os clientes inscritos. Podendo ser usado localmente ou com a nuvem para acesso a diferentes dispositivos com conexão à internet.

Por utilizar o protocolo MQTT, possui as mesmas desvantagens como um ciclo de transmissão lento, não possui suporte a transmissão de vídeo e a segurança é bem simples com poucas autenticações, apenas para usuários e senhas com formato de texto.

3.1 Configuração do broker

Para usarmos o eclipse mosquitto no docker, é preciso primeiramente criar três pastas: config, data, log.

Dentro da pasta config tem-se o arquivo “mosquitto.conf”, onde é armazenado as configurações do broker. Onde deverá ser colocado onde os dados de conexão, inscrição e as mensagens nesse caso será guardado, nessa caso será colocado na pasta “data”. Enquanto os logs do broker são guardados na pasta de log. Como estamos utilizando uma porta diferente da padrão (1883) deve ser mudado a porta de listener. Para segurança, os usuários e senhas serão guardados em um arquivo na pasta config.

Como uma forma de facilitar a criação do container foi utilizado o comando docker-compose. Onde é necessário o arquivo “docker-compose.yaml”, onde é posto os parâmetros utilizados para fazer o container como o nome, a imagem do docker utilizada, se sempre vai reiniciar, o volume a ser montado e as portas que serão abertas.

4 ARMAZENAMENTO

O PostgreSQL é um programa open-source de dados que usa a linguagem SQL, que possui uma arquitetura no modelo cliente-servidor.

As vantagens do PostgreSQL é sua fácil utilização, possui tipos de dados definidos pelo usuário, por ser open-source tem suporte da comunidade e tem suporte ao ACID (**A**tomicidade, **C**onsistência, **I**solamento e **D**urabilidade).

Enquanto suas desvantagens se destaca o mal desempenho e velocidade comparado com outras ferramentas.

5 MICRO-SERVIÇO

Seleção da linguagem e do framework de desenvolvimento: como decidimos por utilizar uma plataforma web, devido à sua praticidade (é possível acessar de diversos dispositivos diferentes), e dado que JavaScript é amplamente utilizado para a web, escolhemos então utilizar frameworks JavaScript para o desenvolvimento da interface web deste projeto. Para o front-end, utilizamos ReactJS e para o back-end utilizamos NodeJS.

Como citado logo acima, utilizamos a biblioteca ReactJS para criar a interface que o usuário vai utilizar para a leitura da temperatura e umidade da sala. A ideia é de que os dados sejam apresentados de forma intuitiva exibindo simultaneamente a data em que está acontecendo a leitura. Assim, o número em ºC da temperatura e o número em % da umidade foram centralizados dentro da interface.

Desta forma, para implementar foi criado um arquivo App.jsx e começar a aplicação da interface com a chamada de API do Node para ler os dados coletados, além de definir as classes de cada componente.

Com a estrutura montada, classes e variáveis declaradas, no Index.css definimos os aspectos visuais que queremos para cada componente como background, cor, fonte, alinhamento e tamanho. Ou seja, o refinamento da interface para que ela seja atrativa e intuitiva para melhorar a usabilidade do usuário, no caso a leitura da temperatura e da umidade.

6 SEGURANÇA

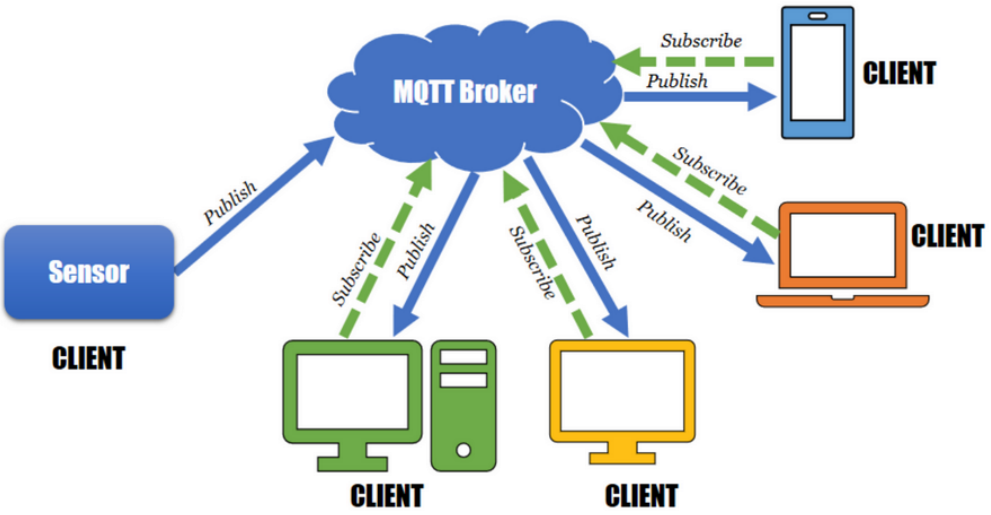
O protocolo MQTT (Message Queuing Telemetry Transport) é um protocolo de mensagens entre máquinas open-source desenvolvido pela IBM. Por ser um protocolo leve e simples é muito utilizado em aplicações para a IoT, o que permite utilizá-lo em locais com internet limitada. Também não sofre sobrecarregamento,

pode ser instalado na maior partes dos hardwares, possui um consumo baixo de energia e algum nível de segurança para os dados.

Ele funciona a partir de uma conexão TCP/IP que depois o cliente pode ser autenticado por um certificado SSL/TLS ou por um usuário e senha. Após a autenticação, o cliente pode se inscrever e publicar mensagens e operações.

As desvantagens são um ciclo de transmissão lento, não possui suporte a transmissão de vídeo e a segurança é bem simples com poucas autenticações, apenas para usuários e senhas com formato de texto.

Figura 2 - Esquema de funcionamento do broker MQTT



Para que haja a conexão com broker, deverá ser feito um usuário e senha que estão guardados no arquivo “mosquitto.passwd” na pasta de configuração do broker no container.

Quando a segurança da api, como dito anteriormente, está sendo utilizado de autenticação via JWT, e as senhas são criptografadas no banco de dados.

7 RESULTADOS

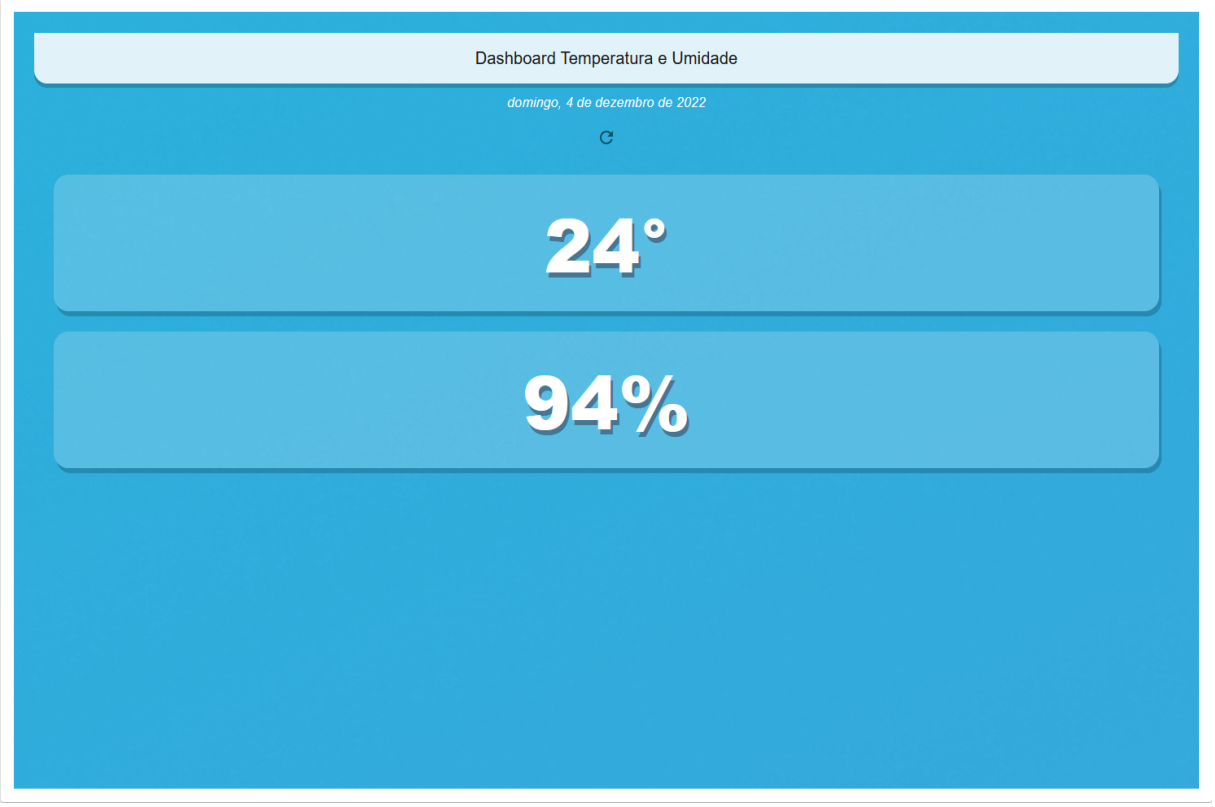
Para poder acessar o sistema de consulta de temperatura e umidade pode-se utilizar o seguinte endereço web: http://andromeda.lasdpc.icmc.usp.br:8423/

Assim chegando na tela de login em que deve ser inserido o endereço de email e senha cadastrado.

Figura 3 - Tela de Login

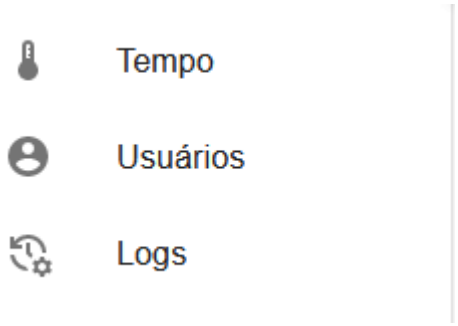
Após a validação do login, o usuário já vai se deparar com o quadro mostrando a temperatura e a umidade do ambiente medida pelo sensor conectado ao ESP32 naquele momento.

Figura 4 - Quadro de temperatura e umidade.



Ao lado esquerdo tem-se um menu que leva às seguintes opções: “Tempo”, “Usuários” e “Logs”.

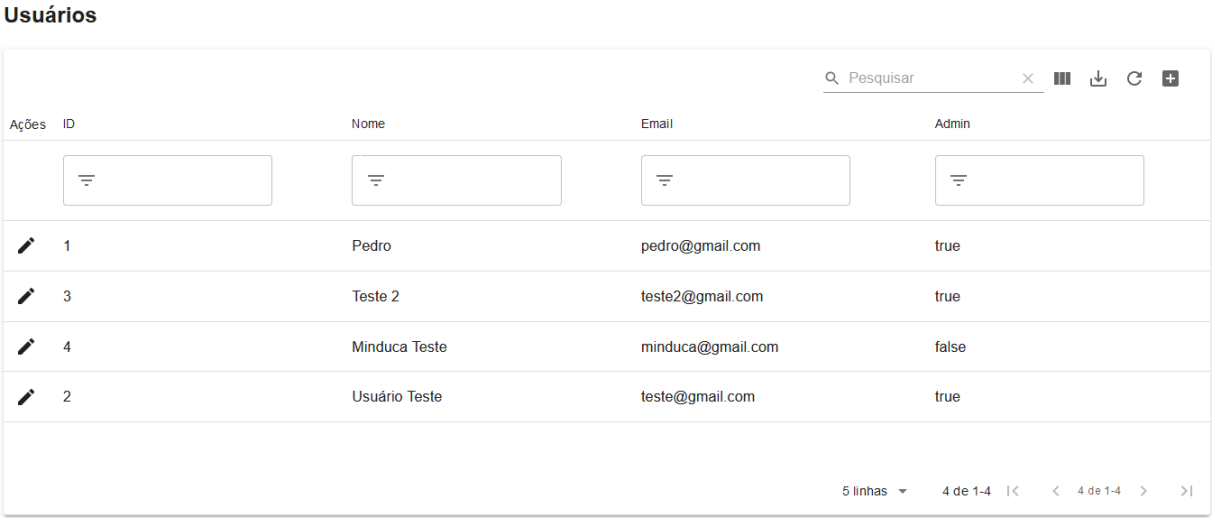
Figura 5 - Menu do lado esquerdo do site



Na opção de usuários aparece a lista de usuários cadastrados apenas se o usuário que fez o login for administrador. Nesta tela, mostra o nome, o email e se o usuário é administrador ou não. Pode-se buscar e alterar informações dos usuários já existentes, baixar a lista no formato CSV e adicionar novos usuários.

Para a criação de novos usuários é necessário que a senha possua um tamanho de 6 a 20 caracteres contendo pelo menos uma letra maiúscula e minúscula e um número.

Figura 6 - Lista de usuários



Na opção de Logs aparecem os registros que ocorreram para o administrador, mostrando quem foi o autor, o tipo de registro, o tipo de ação e a mensagem. Também é possível baixar no formato CSV.

Figura 7 - Lista de Registros