UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
DEPARTAMENTO DE COMPUTAÇÃO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
TRABALHO DE CONCLUSÃO DE CURSO (2024)

DESENVOLVIMENTO DE MÓDULOS DE SOFTWARE PARA ACESSO A DADOS METEOROLÓGICOS EM TEMPO REAL

Alexandre Robson da Costa Freitas [1]
Paulo Henrique Lopes Silva [2]

RESUMO

A importância do monitoramento da temperatura e umidade e os efeitos das mesmas em nossas vidas e atividades permite compreendermos eventos e nos ajudar nas tomadas de decisões. Temos uma gama gigantesca de possíveis aplicações utilizando-se do monitoramento desses dados em diferentes áreas. Assim podemos concluir que, dada a necessidade de se construir uma aplicação web, é possível ter requisitos diversos para se construir um software e suas funcionalidades. Diante desses desafios, este artigo propõe a realização de três projetos experimentais, cada um enfocando requisitos distintos e complementares, todos com o objetivo unificado de disponibilizar dados meteorológicos em tempo real de maneira acessível e simplificada. Tais projetos serão concebidos com base em abordagens e tecnologias contemporâneas, visando oferecer soluções eficientes para cada conjunto de requisitos específicos para cada experimento.

Palavras-chave: IOT, MQTT, ESP8266, temperatura, umidade.

1 INTRODUÇÃO

A medição e coleta de dados climáticos, tais como temperatura e umidade do ar, desempenham um papel fundamental na tomada de decisões em uma ampla gama de aplicações e setores. Além disso, é notável o consumo quase imediato de informações por meio da Web, refletindo uma demanda crescente por acesso instantâneo a dados relevantes.

O contexto ressalta a necessidade de desenvolver uma solução que permita a obtenção em tempo real de dados meteorológicos cruciais para uma variedade de atividades. Nesse sentido, propõe-se a concepção de um protótipo avançado de estação meteorológica, destinado à coleta de informações como temperatura e umidade do ar em ambientes diversos. Diante da prevalência de microcontroladores e da crescente integração de dispositivos via rede com computadores convencionais, bem como da ampla adoção de aplicativos web como meio de acesso fácil para usuários finais, emerge o desafio fundamental: estabelecer uma interconexão eficaz entre o protótipo da estação meteorológica e um serviço web, viabilizando assim um acesso rápido e intuitivo a esses dados essenciais.

Estudos como os conduzidos por Lima (2018) e Awaludin (2021), trazem soluções robustas para a coleta e análise de dados meteorológicos. Esses projetos propõem arquiteturas

[1] Graduando do Bacharelado em Ciência da Computação da UFERSA
[2] Professor Associado do Departamento de Computação da UFERSA/Mossoró

integrando dispositivos IoT (do inglês *Internet of Things*) a sistemas web, com o intuito de viabilizar a obtenção de informações em tempo real de maneira eficiente e econômica. Contudo, mesmo diante desses avanços, uma lacuna persiste no panorama atual: a necessidade de construção de requisitos mais enxutos, o que poderia reduzir consideravelmente os custos de produção dessas soluções. É possível conceber uma arquitetura que atenda plenamente aos requisitos, ao mesmo tempo em que mantém um custo e escopo reduzidos, otimizando assim a eficiência e acessibilidade dessas soluções.

Como nossa proposta se baseia em uma variedade de requisitos básicos, aliada à necessidade de flexibilidade e baixo custo, a utilização de plataformas pagas, que impõem limitações em suas versões gratuitas, como a restrição de usuários ou dispositivos conectados, não é uma opção viável. Embora soluções sugeridas por Camusso (2021) e Seman (2020) tragam plataformas que ofereçam uma ampla gama de recursos robustos, frequentemente superam as necessidades essenciais e impõem barreiras, como a restrição de dispositivos ou usuários no monitoramento.

Diante desses desafios, este artigo propõe a realização de três projetos experimentais, cada um enfocando requisitos distintos e complementares, todos com o objetivo unificado de disponibilizar dados meteorológicos em tempo real de maneira acessível e simplificada. Tais projetos serão concebidos com base em abordagens e tecnologias contemporâneas, visando oferecer soluções eficientes para cada conjunto de requisitos específicos para cada experimento.

Ao término destes três experimentos, almeja-se obter artefatos de sistemas meteorológicos de tempo real que sejam eficientes, precisos, de fácil utilização e, simultaneamente, de baixo custo. Isso não apenas aprimora o entendimento do processo de desenvolvimento de software a partir de requisitos específicos, mas também destaca as vantagens de adotar um escopo simplificado para atender a essas demandas.

2 DESENVOLVIMENTO

A seguir, serão apresentadas todas as informações pertinentes ao desenvolvimento do trabalho, organizadas em duas subseções principais: a primeira discorre sobre a base teórica relacionada às tecnologias essenciais para a resolução do problema, conforme delineado na Seção 2.1; a segunda concentrar-se-á no método empregado para a implementação da solução proposta, como detalhado na Seção 2.2.

2.1 Fundamentação teórica

A monitoração contínua e precisa da temperatura e umidade é fundamental em uma variedade de campos, desempenhando um papel de aplicação crucial nas atividades modernas. Desde a agricultura, onde tais dados são utilizados para executar de forma mais eficiente e assertiva a produção de alimentos (Nascimento, 2023); na prevenção de desastres naturais, auxiliando na detecção precoce de fenômenos climáticos (Obinna, 2024); até no setor da saúde como: dos mais lógicos como o armazenamentos de insumos (sangue, medicamentos, vacinas, dentre outros) (Kurniawan, 2024), e os menos intuitivos como o próprio ambiente hospitalar dada a importância da qualidade do ar (muito relacionado com a propagação de fungos) (Abassi, 2018).

De maneira cada vez mais pervasiva, observamos um aumento significativo de dispositivos inteligentes tanto em ambientes de trabalho quanto em nossas residências. Esses dispositivos têm a capacidade de coletar uma grande quantidade de dados. Com o advento da Internet das Coisas, tornou-se possível não apenas utilizar esses dados para gerar

conhecimento, mas também para oferecer assistência aos seres humanos em suas atividades diárias (Baccour, 2021).

A motivação para a utilização de ferramentas de IoT no monitoramento de temperatura e umidade é impulsionada pela necessidade de obter dados precisos e em tempo real, além da capacidade de automatizar processos e tomar decisões informadas de forma mais eficiente. A IoT oferece uma plataforma para conectar dispositivos e sensores que podem coletar e transmitir dados sobre temperatura e umidade de forma contínua e remota. Esses dados podem ser acessados e analisados instantaneamente, permitindo uma compreensão mais abrangente das condições ambientais em diferentes locais e em diferentes momentos.

2.2 Método

Foram desenvolvidos três projetos distintos utilizando um dispositivo IoT para medição de temperatura e umidade, cada um com suas especificidades. O primeiro projeto envolveu o uso do dispositivo como um sistema simples de monitoramento, permitindo a visualização dos dados de forma direta. No segundo, o foco foi na coleta dos dados e sua disponibilização para consumo por outras aplicações, tornando a solução mais flexível e adaptável. Já no terceiro projeto, foi criada uma aplicação completa que, além de permitir a visualização dos dados, oferece funcionalidades adicionais, atendendo de maneira mais abrangente às necessidades dos usuários.

Esses três projetos foram pensados para demonstrar diferentes maneiras de utilizar a placa IoT em soluções tanto simples quanto mais complexas.

2.2.1 Arquitetura

A arquitetura de cada projeto foi desenvolvida visando a simplicidade e eficiência na comunicação entre o dispositivo IoT e as aplicações consumidoras dos dados, proporcionando a flexibilidade na escolha das soluções finais. A seguir,são descritas as arquiteturas de cada um dos projetos:

1. **Dispositivo IoT como servidor local:** Neste projeto, o ESP, equipado com sensores de temperatura e umidade, coleta os dados e os disponibiliza diretamente para os usuários através de uma página web. A comunicação ocorre dentro da rede local, e os usuários podem acessar a página simplesmente abrindo um navegador em seus dispositivos. Nessa página, eles visualizam as informações coletadas pelo ESP em tempo real.
2. **Dispositivo IoT como publicador:** Aqui, o dispositivo é configurado como um publicador de mensagens, enviando dados de temperatura e umidade para um sistema central, que centraliza e disponibiliza essas mensagens para consumo. Outros sistemas ou aplicativos podem se conectar a esse sistema central para acessar os dados. Dessa forma, o usuário final pode acessar os dados por meio da aplicação consumidora de sua escolha, interagindo de forma flexível com o sistema.
3. **Dispositivo IoT como publicador com aplicação web completa:** Neste terceiro projeto, o dispositivo continua coletando e enviando os dados para um sistema central. A diferença é que foi desenvolvida uma aplicação web completa para que os usuários possam acessar e visualizar as informações com mais detalhes. Essa aplicação oferece uma interface amigável, onde os usuários podem ver os dados em tempo real, além de acessar gráficos e relatórios detalhados. Eles também podem consultar o histórico de dados e exportá-los para outros usos.

Os três projetos desenvolvidos mostram que o dispositivo pode ser aplicado de forma eficaz tanto em soluções simples, como um monitoramento local, quanto em arquiteturas mais complexas, com publicação de dados e integração com aplicações completas. Isso evidencia a flexibilidade do dispositivo para atender a diferentes necessidades.

3 RESULTADOS

A seguir, serão detalhados os resultados alcançados com a execução do método para contemplar os objetivos deste trabalho. As informações estão organizadas da seguinte maneira: cada subseção trará a arquitetura, as ferramentas e a avaliação do sistema de cada experimento, de tal modo que, a subseção 3.1 descreve o primeiro experimento, e as subseções 3.2 e 3.3 descrevem o segundo e o terceiro experimento respectivamente.

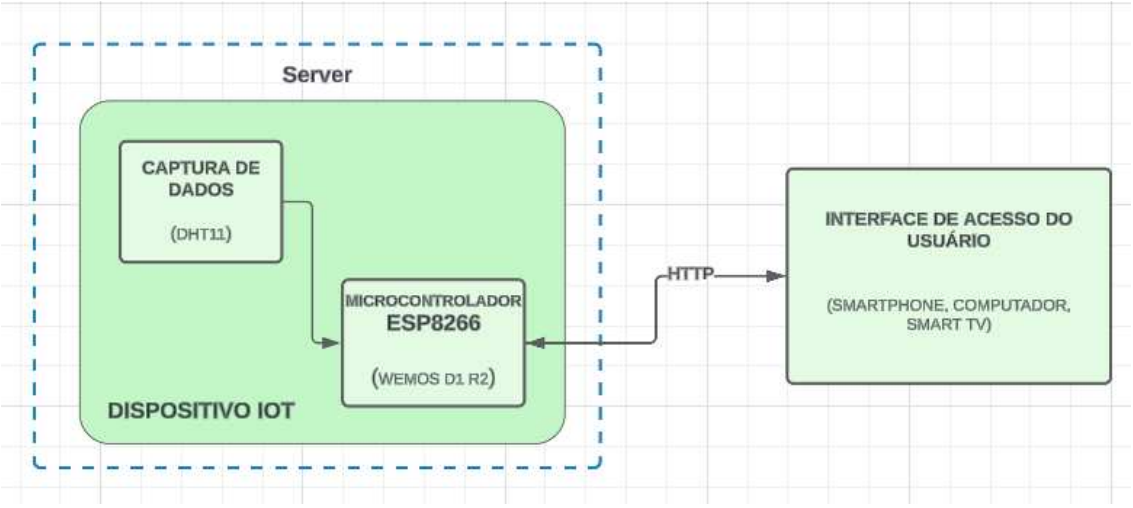
3.1 Dispositivo IoT como servidor

O primeiro experimento tem como proposta, a criação do sistema mais básico possível, a fim de disponibilizar o monitoramento em tempo real dos dados meteorológicos. Objetivamente atender a esses requisitos com o menor custo possível.

3.1.1 Arquitetura do sistema

O sistema foi desenvolvido com uma arquitetura altamente integrada, tornando-a praticamente indivisível. Essa abordagem nos permite definir toda a estrutura como uma unidade coesa, que podemos denominar de dispositivo IoT (Figura 1)

Figura 1 - Arquitetura do dispositivo IoT como servidor.



Fonte: Autoria própria (2024).

É importante observar que o dispositivo concentra todas as funções de captura, controle e disponibilização de dados. Para tornar esses dados acessíveis a partir de qualquer dispositivo com um navegador web, é estabelecida uma comunicação via protocolo HTTP. Tal comunicação se baseia em um modelo de requisição e resposta.

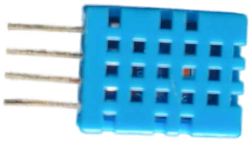
3.1.2 Ferramentas

O experimento requer um dispositivo para a captura de dados (sensor DHT11), e outro (microcontrolador ESP8266) para o controle e disponibilização das informaçẽos previamente capturadas, funcionando como um servidor.

3.1.2.1 Sensor DHT11

O DHT11 (Figura 2) é um componente amplamente adotado para medição de temperatura e umidade relativa do ar em ambientes diversos. Reconhecido por sua acessibilidade e precisão satisfatória em aplicação básica, é composto por um sensor de umidade e um termistor para aferir a temperatura. Internamente, converte essas medidas em sinais digitais que podem ser facilmente interpretados por microcontroladores, como Arduino, Raspberry Pi, entre outros. Sua interface simplificada, que utiliza apenas um pino de dados para comunicação com o microcontrolador, e outros dois sendo responsáveis pela alimentação positiva e negativa (apesar de possuir 4 pinos, 1 deles não possui funcionalidade), facilita sua integração. Além disso, apresenta baixo consumo de energia, uma taxa de leitura de dados a cada dois segundos (ASAIR, 2024).

Figura 2 - sensor de temperatura e umidade DHT11.



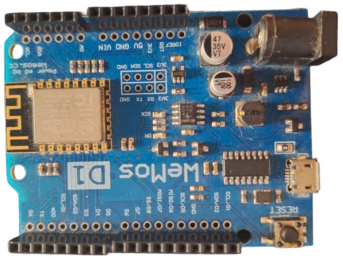
Autor: Autoria própria (2024).

Dadas as características, o DHT11 se destaca como uma solução eficiente e acessível para medição de temperatura e umidade, sendo fácil de integrar a microcontroladores e ideal para aplicação básicas.

3.1.2.2 Microcontrolador ESP8266

Optamos por utilizar o WeMos D1 R2 (Figura 3) para nosso projeto devido às excelentes características do microcontrolador ESP8266, em que se baseia.

Figura 3 - Placa WeMos D1 R2.



Autor: Autoria própria (2024).

O ESP8266 é amplamente conhecido por suas capacidades embutidas de Wi-Fi, o que o torna ideal para projetos de IoT, conectividade sem fio e automação residencial. No contexto técnico, o ESP8266 é um microcontrolador poderoso, e sua compatibilidade com a IDE Arduino facilita significativamente o desenvolvimento. Por meio do WeMos D1 R2,

encontramos uma plataforma versátil para a realização dos nossos experimentos (Silveira, 2024).

3.1.3 Considerações sobre o primeiro experimento

O sistema disponibiliza aos usuários uma interface de acesso por meio de uma página HTML, que pode ser acessada por qualquer dispositivo com acesso a um navegador de preferência do usuário. Essa interface tem a função de apresentar os dados em tempo real, com uma taxa de atualização de 5 segundos, o que permite um monitoramento eficaz e ágil (Figura 4). Essa capacidade de operar em tempo real é alcançada pelo emprego de Javascript na página, o qual efetua duas requisições assíncronas para o dispositivo em intervalos de tempo regulares.

Figura 4 - Interface de monitoramento do dispositivo IoT

Autor: Autoria própria (2024).

Destacamos como pontos positivos do desenvolvimento da aplicação sua usabilidade simplificada e custo acessível, resultante de uma implementação simples e veloz. A aplicação atende aos requisitos de acesso rápido, fácil e confiável aos dados de temperatura e umidade em tempo real.

Por outro lado, as limitações da aplicação residem na ausência de medidas robustas de segurança no acesso, além de não ser um dispositivo destinado especificamente a hospedar aplicações como um servidor web, o que inviabiliza tornar essa aplicação acessível fora de uma intranet.

Em suma, podemos concluir que, dadas as restrições mencionadas, o sistema de monitoramento de dados meteorológicos em tempo real é uma solução viável e suficiente para os requisitos estabelecidos, desde que o acesso do usuário e o dispositivo estejam na mesma rede, não permitindo acesso externo.

3.2 Disposito IoT como publisher MQTT

Para o segundo experimento, temos um sistema que mantém os objetivos do primeiro experimento, com a proposta de solucionar as limitações destacadas na seção 3.1.3, possibilitando neste, o acesso não mais limitado à uma rede interna.

O sistema propõe ser um publisher (publicador ou escritor) de mensagens utilizando-se do protocolo MQTT

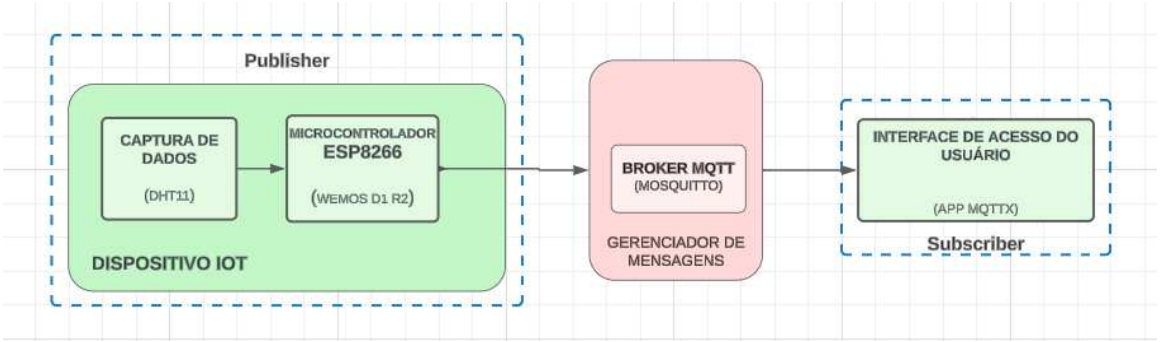
3.2.1 Arquitetura do sistema

O desenvolvimento do sistema do segundo experimento, foi construído de tal modo a dar ao dispositivo um papel de publicador dos dados, e oferecer um sistema capaz de consumir as informações em tempo real. Mantendo a estrutura arquitetural do dispositivo IoT, temos dois subsistemas (Figura 5):

a) **Dispositivo IoT**: os recursos organizacionais são os mesmos mencionados na seção 3.1.1. A distinção está no papel atribuído a este subsistema. Anteriormente, atuava como servidor; agora, desempenha exclusivamente o papel de publicador dos dados, enviando as informações coletadas.

b) **Gerenciador de mensagens**: este funciona como uma assinatura de um serviço, de tal modo que um *publisher* (publicador) cria um tópico, e dado o tópico criado, temos *subscribers* (inscritos) que assinam o tópico. Dado essa configuração, toda publicação feita no tópico é recebida para todos os *subscribers*.

Figura 5 - Arquitetura do sistema do dispositivo como publisher.



Autor: Autoria própria (2024).

Com essa nova configuração, o sistema adquire maior flexibilidade. Ao delegar ao dispositivo IoT a função exclusiva de publicação de dados, elimina-se a sobrecarga de processamento local e a dependência de conexão direta com os dispositivos consumidores. Dessa forma, a arquitetura se torna mais robusta, facilitando a integração de novos serviços e assinantes, mantendo uma transmissão de dados rápida e confiável.

3.2.2 Ferramentas

Para a realização do experimento, é essencial o uso de um broker de mensagens, para o qual foi adotado o Mosquitto, configurado em um ambiente Docker a fim de garantir a disponibilidade do serviço. Adicionalmente, emprega-se o MQTTX como consumidor de mensagens e o protocolo MQTT para assegurar uma comunicação eficiente entre as ferramentas.

3.2.2.1 Protocolo MQTT

O MQTT, do inglês *Message Queuing Telemetry Transport* ou (em português, Protocolo de Transporte de Telemetria de Filas de Mensagens), é uma solução de comunicação projetada especialmente para dispositivos com recursos limitados e redes pouco confiáveis. Nesse modelo, os dispositivos se comunicam por meio de um intermediário chamado *broker*, que facilita a troca de mensagens entre eles (Code, 2024). O protocolo

segue uma abordagem de publicação/assinatura, em que os dispositivos enviam mensagens para "tópicos" específicos, e outros dispositivos interessados se inscrevem nesses tópicos para receber as mensagens pertinentes. Essa arquitetura possibilita uma comunicação eficiente e assíncrona entre os dispositivos, onde os clientes podem ser sensores, atuadores, aplicativos ou qualquer outro tipo de sistema conectado à rede (OASIS, 2024).

3.2.2.2 Broker - Mosquitto

O Mosquitto, uma implementação de código aberto do protocolo MQTT desenvolvida pela Eclipse Foundation, destaca-se por sua leveza, eficiência e segurança, sendo uma escolha popular como broker MQTT. Possui recursos de segurança como autenticação de cliente e criptografia de dados que garantem a integridade e confidencialidade das mensagens (Mosquitto, 2024). A instalação e configuração simplificadas (Figura 6), juntamente com a documentação detalhada disponível, contribuem para a facilidade de uso. Para a configuração, foram aplicados métodos essenciais, como a especificação da porta (*listener*), a definição de persistência de dados (*persistence*), a indicação do caminho de log (*log_dest*) e o estabelecimento de uma lista de autenticação para controle de acesso (*password_file*)

Figura 6 - Arquivo de configuração do broker Mosquitto

Fonte: Autoria própria (2024)

Note que se tem um arquivo para autenticação configurável (*authentication_file*), de tal modo que em seu conteúdo pode haver uma lista de autenticações onde cada um segue o formato *username:password*.

Em suma, o Mosquitto destaca-se pela combinação de leveza, eficiência, segurança e facilidade de uso, sendo uma opção versátil para projetos de IoT e comunicação de máquina para máquina. O serviço irá funcionar em container Docker. Isso trará muita praticidade para a utilização do broker. Falaremos mais sobre essa funcionalidade na seção a seguir.

3.2.2.3 Docker e Docker Compose

O Docker é uma plataforma de código aberto que automatiza o processo de implantação de aplicativos em contêineres. Esses contêineres encapsulam o software, suas dependências e configurações, garantindo que o aplicativo seja executado de maneira consistente em qualquer ambiente (Docker, 2024).

O Docker Compose é uma ferramenta que permite definir e gerenciar aplicativos Docker multi-contêiner em um único arquivo de configuração. Ele simplifica a criação,

execução e escalonamento de aplicativos compostos por vários serviços.

O uso do Docker e do Docker Compose simplifica significativamente a implantação de brokers MQTT, como o Mosquitto, oferecendo uma maneira consistente e eficiente de gerenciar serviços em contêineres. Essa abordagem facilita a escalabilidade, a replicação e a manutenção dos serviços MQTT. Ao adotar essa abordagem, os desenvolvedores podem concentrar seus esforços na criação de aplicativos inovadores, aproveitando a confiabilidade e a flexibilidade oferecidas pelo Docker e Docker Compose. A configuração utilizada para o experimento está descrita na Figura 7.

Figura 7 - Configuração de contêineres via Docker Compose do primeiro experimento

Fonte: Autoria própria (2024)

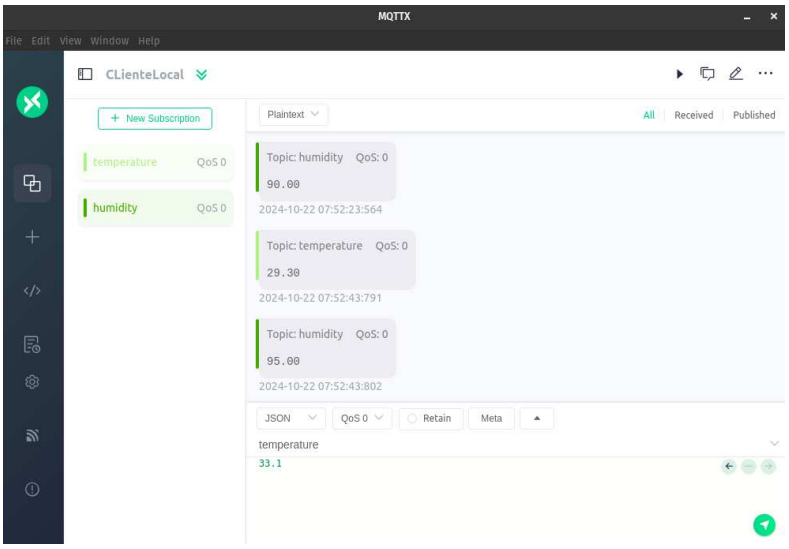
A configuração do arquivo docker-compose.yml segue a versão 3.8 do Docker Compose. O serviço broker utiliza a imagem *eclipse-mosquitto:2.0* para atuar como broker MQTT. Além disso, o arquivo de configuração personalizado do Mosquitto é montado como um volume no caminho /mosquitto/config/mosquitto.conf, caminho padrão recomendado.

3.2.3 Considerações sobre o segundo experimento

Com o objetivo de atender à exigência de fornecer monitoramento em tempo real de forma segura, independentemente do local de acesso às informações, desenvolvemos um sistema que disponibiliza esses dados via protocolo MQTT. Isso resulta em uma ampla variedade de interfaces para o consumo desses dados, incluindo aplicativos para smartphones e desktops, outros dispositivos IoT e sistemas que requerem apenas integração.

Para nossos testes, utilizamos o aplicativo desktop MQTTX, reconhecido por sua eficiência, tem como característica ser de código aberto e possuir uma interface de fácil utilização. Na Figura 8, é possível observar como esses dados são enviados pelo sistema.

Figura 8 - Interface MQTTX



Autor: Autoria própria (2024).

Nesse sistema as limitações se dão pela dependência de interface fora do ecossistema, ou seja, o sistema não se torna auto suficiente, além de depender da aplicação escolhida, pode-se ter uma interface não muito amigável. Caso tenhamos a necessidade de gerar relatórios, ou efetuar determinados estudos e análises destes dados coletados, só se torna possível dada essa persistência.

3.3 Aplicação monolítica

O último experimento tem como requisitos adicionais, a persistência de dados, com um acesso a esses dados; interface amigável; o monitoramento em tempo real dos dados meteorológicos.

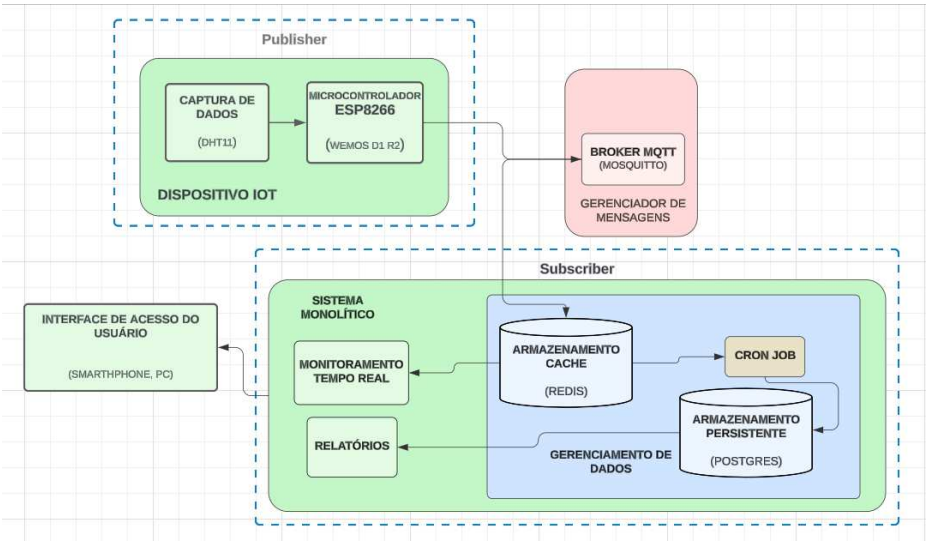
3.3.1 Arquitetura do sistema

A estrutura resultante desse sistema possui todos os componentes do que foi desenvolvido para o segundo experimento. Teremos agora como adição a arquitetura antes mostrada na seção 3.2.1, um subscriber que faz parte do ecossistema do projeto. Esse subscriber trata-se de um monolito, que possui duas subestruturas (Figura 9):

a) **Sistema monolítico:** concentra a parte lógica do gerenciamento e disponibilidade dos dados para o usuário final. Programado utilizando a linguagem Ruby On Rails. Ele disponibiliza além do login de acesso, duas funcionalidades que podem ser utilizadas pelos usuários: painel, onde se encontra o monitoramento em tempo real; e os relatórios, onde se tem acesso aos dados armazenados.

b) **Gerenciador de dados:** esse bloco possui grande responsabilidade em armazenar e disponibilizar os dados de maneira otimizada para serem acessadas pelo sistema criado. Esse subsistema está composto de três elementos e dinâmica entre si, sendo eles: um banco de dados NoSQL trabalhando como cache; outro banco de dados, agora SQL, para a persistência dos dados.

Figura 9 - Arquitetura do ecossistema com monolito



Autor: Autoria própria (2024).

A gestão de dados também envolve uma dinâmica entre os bancos de dados, que é facilitada por meio de um cron job. Este componente assíncrono é responsável por reunir os dados temporariamente armazenados em cache e salvá-los de forma persistente no banco de dados relacional. O cron job opera em uma fila e é executado a cada 5 minutos, agregando os dados meteorológicos e calculando a média de temperatura ou umidade para o intervalo de tempo especificado.

Essa abordagem foi adotada visando otimizar o desempenho, pois seria impraticável salvar todos os dados recebidos de forma persistente, considerando que são gerados a cada 2-5 segundos. Isso resultaria em um banco de dados inchado e comprometeria a geração de relatórios, especialmente para períodos de tempo mais longos.

3.3.2 Ferramentas

Nesta seção são descritas as ferramentas utilizadas para a produção do sistema automatizado de captura e monitoramento de dados meteorológicos.

3.3.2.1 Ruby On Rails

É um framework de desenvolvimento web feito em linguagem Ruby, conhecido por sua produtividade e simplicidade. Possui uma vasta coleção de bibliotecas que facilitam o desenvolvimento rápido de sistemas robustos (Rails, 2024) em um sistema monolítico, onde todas as funcionalidades são centralizadas em uma única camada e plataforma da aplicação, Ruby on Rails oferece uma arquitetura coesa e coerente, simplificando o processo de desenvolvimento, manutenção e escalabilidade.

3.3.2.2 Bancos de dados

O sistema tira proveito de dois bancos de dados (ambos sendo executados em *containers* Docker gerenciados por Docker Compose), para gerenciar a persistência e disponibilidade dos dados na aplicação:

a) **Redis:** é um banco de dados extremamente versátil, que também pode ser usado como cache e message broker. Em sua essência o Redis é um armazenamento de estrutura de dados chave-valor. Uma das características notáveis é que os dados são armazenados primariamente na memória RAM, o que torna as operações de leitura e escritas rápidas (Chen, 2016).

b) **PostgreSQL:** é um sistema de gerenciamento de banco de dados relacional de código aberto poderoso e extensível. Ele garante alta confiabilidade, integridade e consistência dos dados.

3.3.3 Considerações sobre o terceiro experimento

O sistema possui uma autenticação de login e senha para acesso do usuário, garantindo um gerenciamento melhor da equipe que possui acesso a tais dados. No que tange as funcionalidades onde os usuários conseguem interagir com os dados meteorológicos temos:

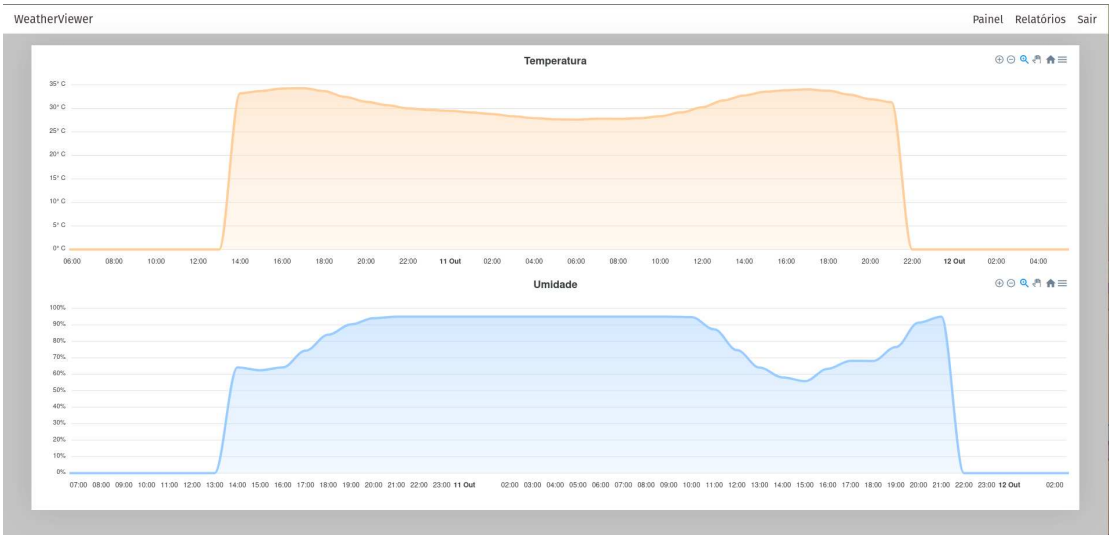
a) **Painel:** Essa página do sistema é idêntica a página de monitoramento utilizada no primeiro experimento, visite a Figura 4 na seção 3.1.3. A ideia foi trazer a mesma simplicidade e eficiência, mas agora acoplada de módulos mais seguros e robusto em recursos computacionais conforme ilustra a Figura 10.

Figura 10 - Painel de monitoramento do sistema monolítico.

Autor: Autoria própria (2024).

b) **Relatórios:** Em relatório é possível ter acesso a dois gráficos, um para os dados coletados de temperatura e outro de umidade. Além do acesso visual ao gráfico é possível exportar os dados, o que torna uma auditoria possível, a fim de compreender um certo comportamento ocorrido (Figura 11).

Figura 11 - Relatórios do sistema monolítico.



Autor: Autoria própria (2024).

Durante o experimento, os dados de temperatura e umidade foram coletados continuamente ao longo de 24 horas em residência pessoal. Essas informações foram organizadas em intervalos de 4 horas, com os valores médios de cada período apresentados (Tabela 1).

Tabela 1 - Média de valores de temperatura e umidade dado um intervalo

Data	Temperatura	Umidade
10/10/2024 (14:00) - 10/10/2024 (17:59)	33,83 ºC	66,28 %
10/10/2024 (18:00) - 10/10/2024 (21:59)	32,02 ºC	90,84 %
10/10/2024 (22:00) - 11/10/2024 (02:00)	29,54 ºC	95 %
11/10/2024 (02:00) - 11/10/2024 (06:00)	28,15 ºC	95 %
11/10/2024 (06:00) - 11/10/2024 (10:00)	29,76 ºC	91 %
11/10/2024 (10:00) - 11/10/2024 (14:00)	30,67 ºC	63,17 %

Fonte: Autoria própria (2024)

O sistema operou de forma estável, captando e transmitindo os dados em tempo real, sem interrupções. O ESP8266 demonstrou eficácia em seu papel de monitoramento, confirmando que a solução foi bem-sucedida em sua capacidade de coletar e disponibilizar as informações, validando o funcionamento da arquitetura implementada.

4 CONCLUSÕES

De acordo com os objetivos definidos para este trabalho, foram desenvolvidos três experimentos de um sistema de monitoramento em tempo real de dados meteorológicos, cada

um com diferentes escopos de requisitos, resultando em protótipos de baixo custo. Cada sistema possui em comum além de suas particularidades, uma interface que tem como funcionalidade monitorar os dados capturados pelo dispositivo IoT praticamente em tempo real.

Pode-se concluir que é possível criar módulos de software mais acessíveis, desde que haja uma consideração cuidadosa do escopo de requisitos e da aplicabilidade do ambiente em que serão implantados.

Como contribuições deste trabalho, além de um estudo de caso sobre a evolução de um software, consideramos que cada experimento representa uma progressão em relação ao anterior. Os protótipos de baixo custo podem ser aplicados em uma ampla variedade de atividades dentro de universidades e também como um produto comercial.

Para trabalhos futuros que possam ser derivados deste, há a possibilidade de refinar os requisitos, introduzindo exigências mais específicas, mas mantendo a proposta de um monitoramento genérico de dados meteorológicos. Por exemplo, é possível expandir as interfaces através da exposição de uma API REST além do monolito, ampliando ainda mais as possibilidades de conexão. Além disso, existe a oportunidade de implantar o terceiro protótipo em uma determinada área e integrar inteligência artificial (IA) para adicionar uma característica preditiva à aplicação, tornando-a não apenas receptiva aos dados coletados, mas também proativa.

REFERÊNCIAS

ABBASI, Fariba; SAMAEI, M. R. The effect of temperature on airborne filamentous fingo in the indoor and outdoor space of a hospital. **Environmental Science and Pollution Research**, 03 jan. 2018.

ASAIR. DHT11 TEMPERATURE & HUMIDITY MODULE. Disponível em: https://asairsensors.com/product/dht11-sensor/. Acesso em: 01 out. 2024.

AWALUDIN, M.; RANGAN, A. Y.; YUSNITA, A. Internet of Things (IoT) Based Temperature and Humidity Monitoring System in the Chemical Laboratory of the Samarinda Industry Standardization and Research Center. Tepian 2, n 3, p. 85-93, 2021.

BACCOUR, Emma, et al. Pervasive AI for IoT applications: A Survey on Resource-efficient Distributed Artificial Intelligence. **IEEE Communications Surveys & Tutorials**, v. 24, n. 4, 2366-2418, 2021. Disponível em: https://ieeexplore.ieee.org/document/9866918. Acesso em: 15 abr 2024.

CAMUSSO, D.; SANTOS, J. R. dos; VIAGI, A. F. Monitoramento ambiental baseado na tecnologia Internet das Coisas para pequenos avicultores / Environmental monitoring based on Internet of Things technology for small poultry farmers. **Brazilian Journal of Development**, *[S. l.]*, v. 7, n. 5, p. 51132–51146, 2021. DOI: 10.34117/bjdv.v7i5.30149. Disponível em: https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/30149. Acesso em: 16 out. 2024.

CHEN, S.; TANG, X.; WANG, H.; ZHAO, H.; GUO, M**.** Towards scalable and reliable in-memory storage system: a case study with Redis. **IEEE TRUSTCOM/BIGDATASE/ISPA**, 2016, Tianjin. **Anais** [...]. Tianjin: IEEE, 2016. p. 1660-1667.

Code, Steve. Steve's Internet Guide - Beginners Guide To The MQTT Protocol. Disponível em: http://www.steves-internet-guide.com/mqtt/. Acesso em: 05 out. 2024.

Docker, What is docker?. Disponível em: https://docs.docker.com/get-started/docker-overview/. Acesso em 16 out. 2024.

KURNIAWAN, A.; PERANGINANGIN, J. M.; HARSONO, S. B. Evaluation of Vaccine Stock Management at the Karanganyar Regency Health Service Indonesia. **Open Access Indonesian Journal of Medical Reviews**, v. 4, n. 1, p. 569-582, 2024.

LIMA, Matheus; ALVES, Francisco; JUCA, Sandro. Sistema IoT para Monitoramento de Temperatura e Umidade Ambientes e Acionamento Remoto de Cargas. *In*: ESCOLA REGIONAL DE INFORMÁTICA DO PIAUÍ (ERI-PI), 4. , 2018, Teresina. Porto Alegre: Sociedade Brasileira de Computação, 2018. p. 232 - 237.

Mosquitto. Eclipse Mosquitto - An open source MQTT broker. Disponível em: https://mosquitto.org/. Acesso em 05 out. 2024

NASCIMENTO, Danilo Carvalho; BÁLSAMO, Rayane. Agricultura Digital e Fluxo de Dados no Desenvolvimento Sustentável do Agronegócio. 2023. Disponível em: http://revistas.icesp.br/index.php/Real/article/view/4903/2609. Acesso em: 15 abr. 2024.

OASIS. OASIS Message Queuing Telemetry Transport (MQTT) TC. Disponível em: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=mqtt. Acesso em 05 out. 2024.

OBINNA, Okeke Remigius; MATHEW, Ehikhamenle. Design Analysis of an IoT based Early Flood Detection and Alerting System. **International Journal of Advanced Engineering Research and Science**, *[S. l.]*, v. 11, n. 02, 2024. Disponível em: https://i.ihspublishing.com/index.php/ijaers/article/view/130.. Acesso em: 15 abr. 2024.

Rails, Guide - Getting Started with Rails. Disponível em: https://guides.rubyonrails.org/getting_started.html. Acesso em 16 out. 2024.

SEMAN, M. T. A.; ABDULLAH, M. N.; ISHAK, M. K**.** Monitoring temperature, humidity and controlling system in industrial fixed room storage based on IoT. ***Journal of Engineering Science and Technology***, Penang, v. 15, n. 6, p. 3588-3600, 2020.

Silveira, André L. Marques da. WeMos D1 R1 Wifi ESP8266. Disponível em: http://www.um.pro.br/arduino/index.php?c=Wemos_D1_R2_Wifi_ESP8266. Acesso em: 05 out. 2024.