H2O Level - Um sistema de monitoramento de níveis de água em reservatórios domésticos utilizando o microcontrolador ESP32

Trabalho de Conclusão de Curso

João Brás da Hora Nascimento Júnior

Antonio Carlos dos Santos Souza
Orientador

Instituto Federal da Bahia – IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
Agosto 2025

SUMÁRIO

1. Visão Geral

1.1 Declaração do Problema

Segundo pesquisa publicada em 2023 pelo Instituto Trata Brasil [1], que atua nas áreas de proteção dos recursos hídricos e avanços no saneamento básico, a Região Metropolitana de Salvador (RMS - composta por 13 municípios) perde cerca de 55,8% de toda água potável destinada ao sistema de distribuição. Percentual este, superior aos 40,3% da média nacional.

Figura 01 - Indicadores de Saneamento da Bahia, Salvador e da Região Metropolitana de Salvador, ano base 2021

Localidade	Índice de Atendimento Total de Água	Índice de Atendimento Total de Esgoto (%)	Índice de Esgoto Tratado Referido à Água Consumida (%)	Índice de Perdas na Distribuição (%)
Região Metropolitana de Salvador	97,4%	75,6%	83,5%	55,8%
Bahia	81,0%	41,4%	46,7%	39,7%
Salvador	98,8%	88,4%	100,0%	56,6%

Fonte: Trata Brasil (2023)

Os dados apontados são referentes apenas às perdas na distribuição, não considerando os desperdícios que ocorrem nas residências, o que faria este percentual ser ainda maior.

Aliado a este cenário de perdas, na cadeia de abastecimento, a população da RMS é afetada por suspensões recorrentes no fornecimento de água [2][3][4], sendo necessária, como medida de mitigação de danos, a instalação de reservatórios residenciais, garantindo um fornecimento de água contínuo.

Inclusive, no dia dessa apresentação de defesa de TCC (05/08/2025), a EMBASA – Empresa Baiana de Águas e Saneamento S.A, está fazendo uma grande operação de manutenção na Barragem de Pedra do Cavalo e em todo o sistema de distribuição, envolvendo cerca de 500 Técnicos, em 50 equipes, afetando 12 municípios baianos, dentre eles, Salvador. Isso demonstra a alta importância de um sistema eficiente de abastecimento hídrico.

Através do monitoramento dos reservatórios residenciais será possível detectar danos estruturais e evitar perdas em decorrência de transbordamentos, esvaziamentos parciais e totais. Por consequência, gastos adicionais serão poupados com faturas de consumo e contratação de carros pipas.

A motivação na escolha do problema abordado no presente Trabalho de Conclusão de Curso (TCC) se deu em decorrência das crescentes preocupações mundiais, referentes à sustentabilidade e preservação dos recursos hídricos. Sendo o desperdício de água um dos grandes problemas atuais,

os seus efeitos podem ser minimizados com a solução proposta neste projeto; que tem as seguintes justificativas:

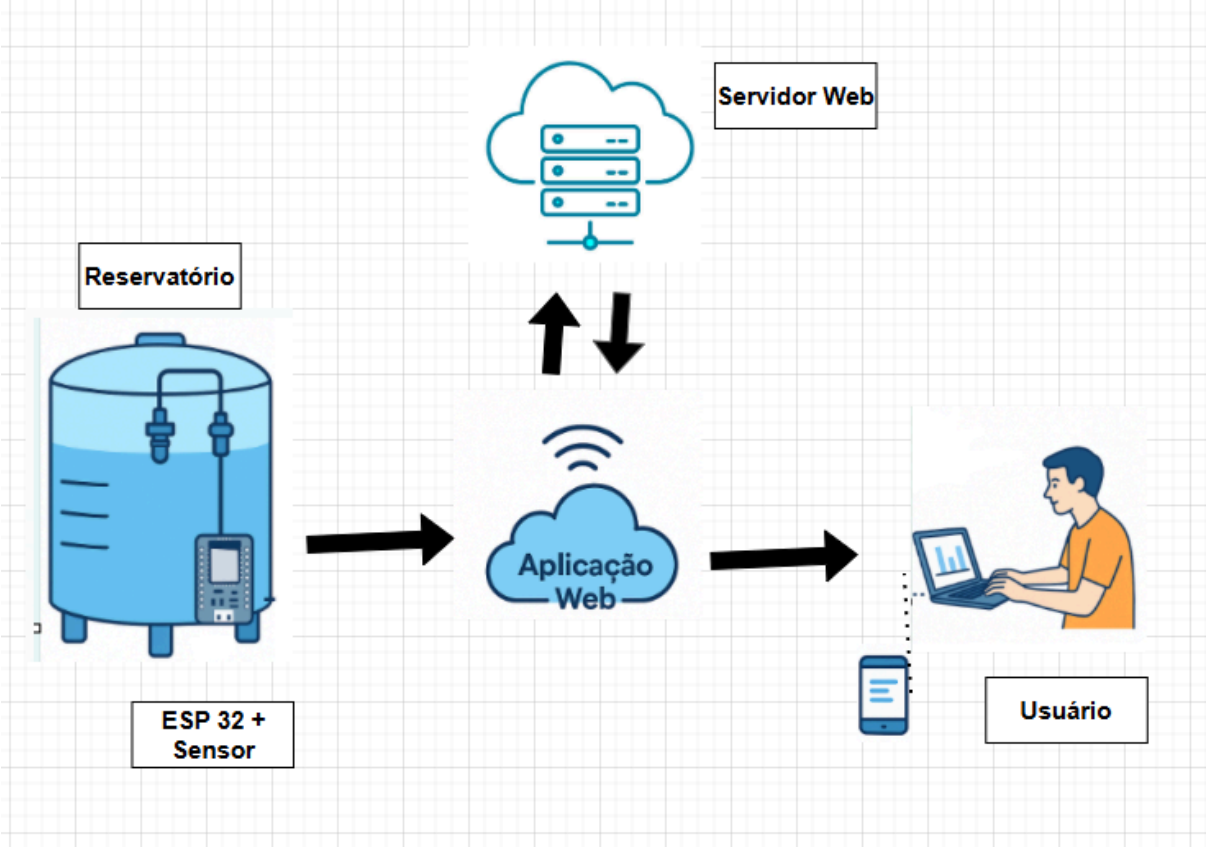
- Sustentabilidade e economia de recursos: visando controlar eficientemente e conservar os recursos hídricos, diminuindo assim desperdícios e custos adicionais.
- Desafio técnico e interdisciplinar: o projeto engloba várias áreas de conhecimento, como eletrônica (sensores e microcontroladores), programação (software gerenciador e visualização dos dados) e sustentabilidade (gestão de recursos hídricos).
- Tecnologia acessível: Utiliza componentes de baixo custo, sendo acessível para toda população.

O objetivo deste projeto é oferecer informações para tomada de decisões visando um maior controle e uso consciente dos recursos hídricos nas residências, minimizando desperdícios; gerando um impacto positivo no cotidiano das pessoas.

1.2 Proposta de Solução de Software

A solução tem como proposta desenvolver um sistema inteligente de monitoramento em tempo real, para reservatórios de água, que permita: acompanhar continuamente os níveis de água, detectar precocemente eventos críticos (baixo nível e transbordamentos) e armazenar dados referentes ao consumo de água nas residências.

Figura 02 - Arquitetura da solução

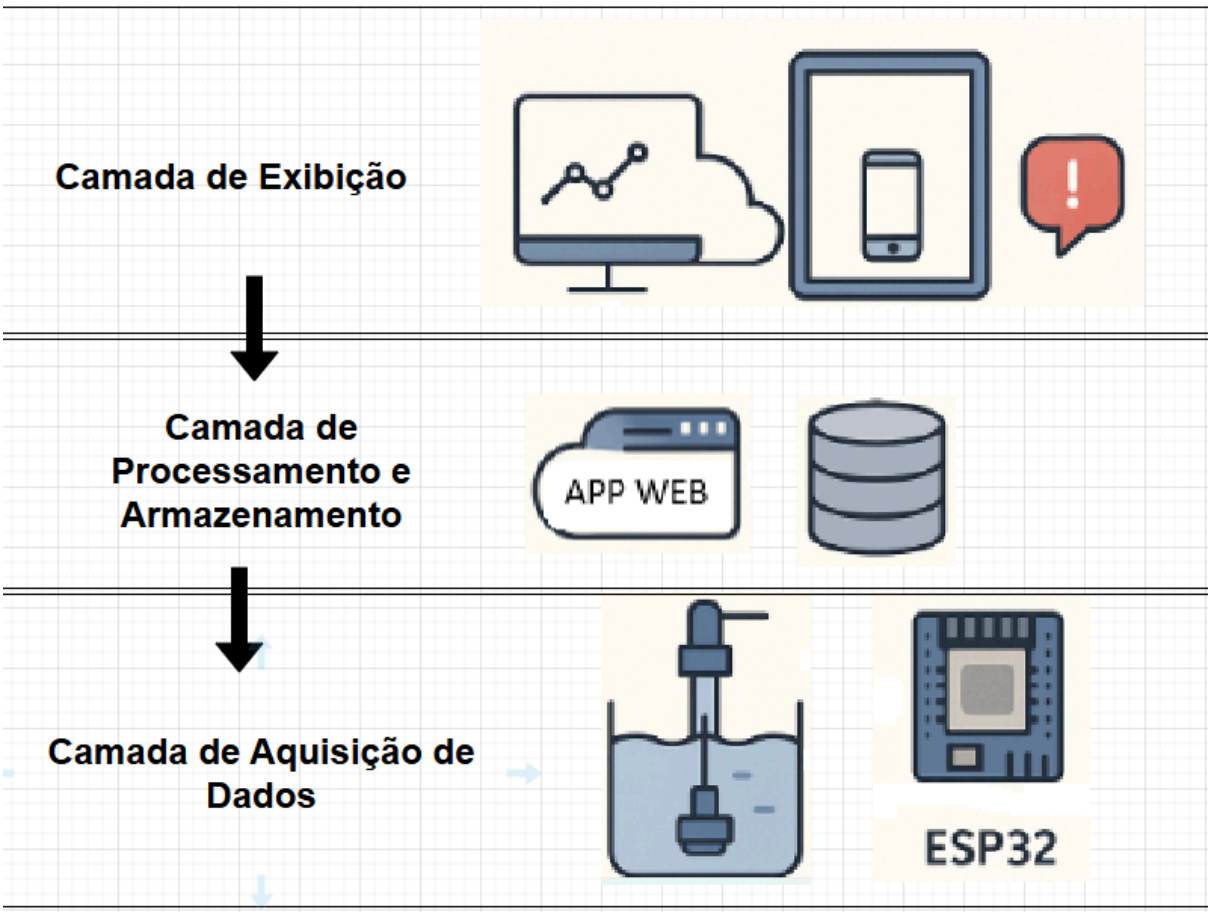


Fonte: Elaboração própria

O Sistema possui três camadas principais:

1) **Camada de aquisição de dados**: responsável pela captura dos dados referentes ao nível de água. Isso será possível através do uso do Sensor HC-SR04 (sensor de distância ultrassônico, responsável por medir os níveis de água) e do microcontrolador ESP32 (receberá os dados captados pelo sensor HC-SR04 e os enviará via WI-FI para o servidor web).
2) **Camada de processamento e armazenamento**: será responsável por armazenar os dados e histórico para análises; processamento dos dados em tempo real para alertas; garantia de informações confiáveis e sempre disponíveis.
3) **Camada de apresentação:** exibirá as informações de forma clara, possibilitando tomadas rápidas de decisão.

Figura 03 - Fluxograma dos relacionamentos entre as camadas destacando seus elementos componentes.



Fonte: Elaboração própria

O presente projeto visa solucionar problemas de falta de controle dos níveis de água em reservatórios, das seguintes maneiras: automatizando o monitoramento, dispensando assim inspeções manuais e minimizando erros humanos; disponibilidade de dados históricos para monitoramento, ajudando a prever padrões de consumo e soluções eficientes; acesso remoto aos dados, possibilitando o acompanhamento do reservatório de qualquer local.

1.3 Tecnologias Adotadas

1.3.1 Camada de aquisição de dados:

1.3.1.1 ESP32 [10][11]
Microcontrolador dual core de 32 bits com Wi-Fi e Bluetooth integrados. Pode ser alimentado via USB e fontes de 3,3V ou 5V. Possui 34 pinos digitais e 15 canais analógicos de 12 bits. Possui memória flash interna e suporte a cartões microSD. Apresenta criptografia para as informações enviadas. O microcontrolador ESP32 foi escolhido por possuir módulo Wi-Fi integrado, facilitando a coleta e transmissão dos dados obtidos, via sensor, para o Banco de Dados MongoDB.

1.3.1.2 HC-SR04 [12]
Sensor de Distância Ultrassônico. Apresenta ótima precisão nas medições realizadas entre 2cm e 4m. O circuito foi desenvolvido utilizando um emissor e um receptor acoplados aos seus 4 pinos (VCC, ECHO, Trigger, GND). Funciona calculando a diferença de tempo de emissão da onda sonora, partindo do pino Trigger, até a recepção pelo pino ECHO; permitindo assim que se calcule a profundidade dos reservatórios. O sensor foi escolhido devido ao seu baixo custo e eficiência, ao ser implantado no modelo criado para demonstração do projeto;

1.3.1.3 Arduino IDE [13]
Software gratuito e de código aberto, que oferece uma interface gráfica amigável de desenvolvimento para programação de microcontroladores Arduino. Utiliza uma linguagem baseada em C/C++, conhecida como Program. Permite a comunicação de placas arduino, via USB, possibilitando o upload de código, diretamente para o Hardware. Esta plataforma também é compatível com o ESP32. A Arduino IDE foi a plataforma de desenvolvimento escolhida para a programação do microcontrolador ESP32, por ser intuitiva e possuir uma grande variedade de recursos e bibliotecas à disposição.

1.3.2 Camada de processamento e armazenamento de dados

1.3.2.1 Node.js & Express.js [14]
Servidor backend com API REST. O Express é um framework para aplicativos web do Node.js; leve e flexível, ideal para aplicações em tempo real e de fácil integração com frontends modernos. A escolha deste framework levou em consideração a sua grande variedade de recursos disponíveis, simplificação na construção da aplicação e desempenho satisfatório na execução do projeto.

1.3.2.2 MongoDB [15]
Banco de dados NoSQL. Sistema de gerenciamento de banco de dados não relacional, baseado em software livre. Utiliza documentos flexíveis, ao invés de linhas e tabelas para o processamento e armazenagem dos dados. Armazenará os dados de forma flexível (JSON-like). É escalável e se mostrou uma ótima opção para registros temporais dos históricos dos sensores.

1.3.2.3 Mongoose [16]
Biblioteca ORM (Object Relational Mapping) para MongoDB. Abordagem mais simples de modelagem dos dados no MongoDB, dentro do Node.js.

1.3.2.4 Axios [17]
Cliente HTTP baseado em promessas, para navegadores e Node.js. Funciona em ambos com a mesma base de código. No servidor usa o código nativo do Node.js - o módulo HTTP, enquanto no cliente (navegador) usa XMLHttpRequests. Foi utilizado tanto no frontend quanto no ESP32 (via HTTPClient) para enviar e receber dados da API.

1.3.2.4 Nodemailer [18]

Módulo do Node.js responsável pelo envio de emails, de forma simples e eficiente. É compatível com os principais serviços de email existentes e protocolos, como o SMTP, OAuth2 e Sendmail. Sua escolha foi baseada na integração nativa com o Node.

1.3.3 Camada de apresentação

1.3.3.1 React.js [19]
Framework Javascript usado para criar interfaces de usuário em aplicativos Web. É uma biblioteca simples de usar, flexível e escalável. Ideal para quem busca agilidade no desenvolvimento; além de oferecer suporte a gráficos. Foi escolhido devido a sua simplicidade, fácil integração com o Node e escalabilidade, já pensando em expansões futuras do projeto.

1.3.3.2 Chart.js [20]
Biblioteca Javascript de código aberto que permite a visualização de dados e criação de gráficos interativos na Web. Foi utilizada por oferecer uma maior quantidade de recursos na visualização do gráfico.

1.3.3.3 Bootstrap [21]
Ferramenta gratuita de estilização de interfaces para desenvolvimento HTML, CSS e JS. Facilita a criação de layouts responsivos e permite o acesso à sites, sem alteração no código, via desktop e dispositivos móveis.

1.3.3.4 CSS [22]
Padrão que define como os dados devem ser expostos nos navegadores e seus respectivos aspectos como: fonte, fundo, texto, cor dos links, margens e disposição dos itens na tela. Foi utilizado a fim de padronizar a estilização dos menus e itens.

De forma resumida, as ferramentas listadas foram selecionadas tomando como base os seguintes critérios:

- Baixo custo: as tecnologias escolhidas possuem funcionalidades gratuitas, facilitando a ampla implantação.
- Escalabilidade: permite futuras expansões para reservatórios múltiplos e adição de uma maior quantidade de sensores.
- Facilidade de integração: todas as três camadas se comunicam com protocolos modernos (HTTP, REST, JSON), o que facilita o desenvolvimento da solução.
- Portabilidade: frontend responsivo e acessível por qualquer plataforma.
- Fácil manutenção: código estruturado por critérios de responsabilidade.

1.4 Trabalhos Relacionados

1.4.1 Monitoramento do nível de água em reservatórios residenciais utilizando sensor ultrassônico [6]

O Trabalho de Conclusão de Curso desenvolvido por Gabriel Belizário Alves, pelo Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, descreve um projeto de monitoramento de níveis de água em reservatórios residenciais, do ponto de vista da engenharia elétrica e eletrônica.

O projeto faz uso de uma placa arduino e o sensor ultrasônico HC-SR04. Além disso, faz uso de relês, protoboard, leds, resistores, regulador de tensão e bateria.

Descreve em detalhes os componentes eletrônicos utilizados, os códigos na programação do arduino, as fórmulas utilizadas na obtenção dos volumes de água, bem como, diferentes medidas de capacidades de reservatórios de água comercializados.

O projeto não possui uma aplicação que possa ser acessada remotamente e o cenário escolhido para os testes, é um ambiente real de uso. Não possui conectividade Wi-Fi, nem qualquer tipo de acesso a aplicações locais ou Web.

De forma resumida, o projeto foca nas obtenções dos volumes dos reservatórios, que só podem ser observados localmente, não tendo os seus dados salvos, para consultas posteriores e auxílio na tomada de decisões.

Comparada a solução que apresento, as únicas semelhanças se dão através da presença do mesmo sensor ultrassônico e do método de coleta dos níveis de água do reservatório.

1.4.2 Monitoramento do nível de água e detecção de falhas através do sistema IoT [5]

O projeto segue a mesma linha do trabalho descrito no item 1.4.1; inclusive, utilizando a maior parte dos materiais. Se diferencia pelo detalhamento da parte hidráulica, foco na automação e integração com a Web, através do Módulo GSM/GPRS SIM800L conectado a placa Arduino, que permite conexão com a internet, via rede 2G; além do envio de mensagens de texto e ligações.

Não há uma aplicação integrada ao projeto, apenas limites pré estabelecidos, que ao serem atingidos, ligações e mensagens de texto são enviadas para um número previamente configurado, informando a situação dos níveis do reservatório.

Possui um interruptor que ao ser acionado, torna o projeto funcional, podendo ser monitorado localmente ou de forma automatizada, a medida que os limites fixados sejam atingidos, ocasionando no disparo de mensagens de texto e ligações.

Em comparação a solução do presente TCC, ambas possuem o mesmo método de coleta dos níveis de água dos reservatórios. Já o mecanismo de envio de mensagens será utilizado de forma distinta; sem a necessidade de um módulo/chip, apenas para cumprir tal função. A solução que proponho trará um sistema Web, capaz de fazer o gerenciamento dos dados e enviar notificações ao usuário, através dele.

1.4.3 Dispositivos de medição de consumo de água usando conceitos de IoT [8]

No trabalho de Conclusão de Curso desenvolvido por Mohamad Sadeque Abou Ali, pela Universidade Federal do Rio Grande do Norte, o diferencial em relação aos projetos apresentados nos itens 1.4.1 e 1.4.2, se dá pelo uso de um sistema de supervisão, chamado de Smart Automation using IoT (SAIOT), que faz toda a monitoração dos parâmetros via servidor Web. Apresenta dois sensores de distância: o HC-SR04, utilizado nos dois trabalhos descritos anteriormente e o sensor JSN-SR04T, que possui um limite de medição muito maior, comparado ao HC-SR04, operando entre 20 e 600 cm de distância, possuindo precisão de 2mm.

O projeto também descreve o monitoramento da bomba e coleta de parâmetros sobre o seu funcionamento, como corrente elétrica que alimenta o sistema, vibração e acionamento.

As principais diferenças deste projeto para os dois anteriores listados são:

- A utilização da placa Wemos D1 mini pro, que pode ser programada usando MicroPython, Arduino e nodeMCU.
- A integração com ferramentas Web e sistema de monitoramento automatizado, o que confere a possibilidade de gerenciamento remoto e coleta de dados para análises posteriores.

Embora a solução abordada por mim, no presente trabalho, tenha o mesmo direcionamento; a que proponho tem ênfase apenas no monitoramento dos níveis de água dos reservatórios e construção do sistema que fará todo o gerenciamento inteligente dos dados; enquanto a proposta apresentada por Mohamed tem foco no monitoramento dos componentes físicos e sua correta funcionalidade, voltando-se mais para a engenharia.

1.4.4 LoRaWAN Performance Analysis for a Water Monitoring and Leakage Detection System in a Housing Complex [9]

O artigo apresenta um estudo sobre o monitoramento da água e detecção de desperdícios, num complexo residencial. Isso se torna possível através da aplicação dos conceitos da Long-Range Wide-Area Network (LoRaWAN) ou Redes de Longo Alcance em Áreas Abertas.

Essa abordagem trata cada residência como um nó na rede, ou seja, um ponto de conexão. Desta forma, possibilita o escalonamento do monitoramento, uma vez que para monitorar mais residências, basta interconectar com outras LoRaWANs.

Cada casa (nó da rede) é composta por um hidrômetro, um sensor de pressão e uma válvula inteligente, monitorados via técnica de baixa pressão.

O estudo também verificou o consumo de energia do sistema, as taxas de transferência de dados e a estabilidade das conexões. Mostrando-se 100% eficiente num cenário sem vazamentos, numa área de 1 km². Já num cenário de vazamentos, apresentou performance de 88% de taxa de detecção numa área de 25 km².
A solução apresentada funciona através da ativação da válvula inteligente, que suspende o fornecimento e notifica o responsável pela residência, através de uma rede de baixo consumo energético.

Esse artigo contribui com a presente solução, ao se pensar em expansões futuras. Pois, é um modelo teórico, trazendo conceitos de interconexão de redes e gerenciamento de nós múltiplos. O que facilitará bastante a gestão de múltiplos usuários e reservatórios, futuramente.

1.4.5 Sistema para monitoramento de reservatórios de água - AquaMeasure [7]

O AquaMeasure foi criado em 2017, inicialmente, como um trabalho de conclusão de curso desenvolvido por Ivo Martins de Souza e Rodolfo Francisco de Oliveira, no Instituto Federal de São Paulo. Posteriormente, se tornou uma aplicação proprietária, de mesmo nome, possuindo versões Web, Android e IOS.

Foi desenvolvido usando Java, num ambiente Eclipse.

Dos cinco projetos apresentados, este é o que mais se assemelha a solução que proponho, apresentando funcionalidades similares.

O que diferencia a minha solução é que utilizo o microcontrolador ESP32, sendo mais atual e possuindo mais recursos e segurança que o utilizado no AquaMeasure (ESP8266); será uma solução

gratuita e de código aberto, onde, qualquer pessoa poderá torná-la funcional, adquirindo o microcontrolador, jumpers e sensor de distância; o seu escopo será reduzido para apenas um reservatório, a fim de facilitar a sua utilização, reduzindo ou eliminando a necessidade de custos com suporte técnico; além de atualizar as tecnologias utilizadas no desenvolvimento do TCC do AquaMeasure (todas elas descritas no tópico 1.3).

Abaixo, segue uma tabela comparativa entre o presente projeto, os trabalhos relacionados e os recursos que cada um apresenta.

Figura 04 - Tabela comparativa entre os trabalhos relacionados

Recursos / Trabalhos Relacionados	Medição dos Niveis	Conectividade à Web	Notificações	Automação	Sistema Integrado
Monitoramento Nível [6]	X				
Monitoramento IOT [5]	X	X	X	X	
Dispositivo Medição [8]	X	X		X	X
LoRaWAN Performance [9]	X	X	X		
AquaMeasure [7]	X	X	X	X	X
H2O Level	X	X	X	X	X

Fonte: Elaboração própria

2. Requisitos

2.1 Requisitos Funcionais

Fazem alusão aos serviços, comportamentos e funcionalidades específicas que a aplicação deve possuir, ou seja, como ela deve reagir a entradas específicas e se portar em determinadas situações, para atender as necessidades e expectativas do usuário.
Segue a lista dos Requisitos Funcionais da aplicação:

- Como usuário, quero me registrar na aplicação para acessar as funcionalidades do sistema.
- Como usuário, quero fazer login para acessar meus dados e configurações.
- Como usuário, quero configurar os parâmetros do reservatório de água, para que os alertas recebidos estejam de acordo com a minha necessidade.
- Como usuário, preciso visualizar o nível de água atual no reservatório para acompanhar a situação em tempo real.
- Como usuário, quero visualizar graficamente os dados de variação do nível da água para entender o comportamento do reservatório ao longo do tempo.

- Como usuário, quero receber notificações sobre a situação do reservatório para agir rapidamente e evitar prejuízos.
- Como desenvolvedor, quero corrigir bugs no sistema ao serem detectados, a fim de manter o bom funcionamento da aplicação.
- Como desenvolvedor, quero poder disponibilizar melhorias e atualizações na aplicação para atender as demandas dos usuários.
- Como desenvolvedor, quero investigar e resolver falhas de execução no sistema para garantir sua estabilidade.
- Como sistema, quero medir automaticamente o nível de água num intervalo de tempo pré-definido a fim de manter os dados atualizados.
- Como sistema, quero armazenar os dados das medições num banco de dados para consultas posteriores e análises dos históricos.
- Como sistema, quero verificar automaticamente se o nível de água está dentro dos parâmetros pré-definidos para enviar alertas.
- Como sistema, preciso me comunicar com o sensor, por meio do microcontrolador ESP32, para coletar os dados físicos do nível da água.
- Como sistema, quero me reconectar automaticamente sempre que houver falhas na comunicação com a rede para evitar perda de dados e prejuízos ao funcionamento.

2.2 Requisitos Não-Funcionais

Descrevem as restrições, qualidades e padrões técnicos que a aplicação deve obedecer, sem adentrar nas funcionalidades específicas, sendo essenciais para garantir a sua eficiência.
Segue a lista dos Requisitos Não-Funcionais da aplicação:

RNF01 O processo de cadastro deve ser simples e intuitivo, não exigindo mais que 3 etapas.
RNF02 O sistema deve utilizar autenticação via login e senha, com criptografia HTTPS nas comunicações
RNF03 O sistema deve permitir a configuração dinâmica do limite do reservatório de água.
RNF04 O sistema deve apresentar as informações do nível de água em tempo real com um atraso máximo de 5 segundos após a coleta dos dados pelo sensor.
RNF05 O sistema deve gerar um gráfico de variação do nível da água de forma dinâmica e responsiva, exibindo os dados históricos armazenados.
RNF06 O sistema deve garantir a entrega de todas as notificações críticas via e-mail.
RNF07 O desenvolvedor deve fazer uso de rotinas de atualizações da aplicação, ferramentas de depuração, como o Debugger, a fim de se identificar e solucionar bugs da aplicação.
RNF08 O sistema deve permitir atualizações remotas do firmware do microcontrolador e da aplicação web, minimizando intervenções do usuário.
RNF09 O código do sistema deve seguir boas práticas de desenvolvimento, aliadas ao uso de ferramentas de depuração de código.
RNF10 O sistema deve ser capaz de se conectar automaticamente à rede para coletar e enviar dados do sensor ao backend.
RNF11 Os dados históricos de nível de água devem ter a sua consistência garantida em casos de falha de comunicação com o sensor e devem ser armazenados por 12 meses no banco de dados.
RNF12 O sistema deve verificar o nível atual e comparar com os parâmetros estabelecidos pelo usuário para enviar os alertas.
RNF13 A comunicação entre o sistema e o microcontrolador deve ser feita por meio de protocolos seguros (HTTP/REST API).
RNF14 O sistema deve tentar se reconectar automaticamente a rede a cada 30 segundos em caso de falha, até que a operação seja bem sucedida.

3. Design

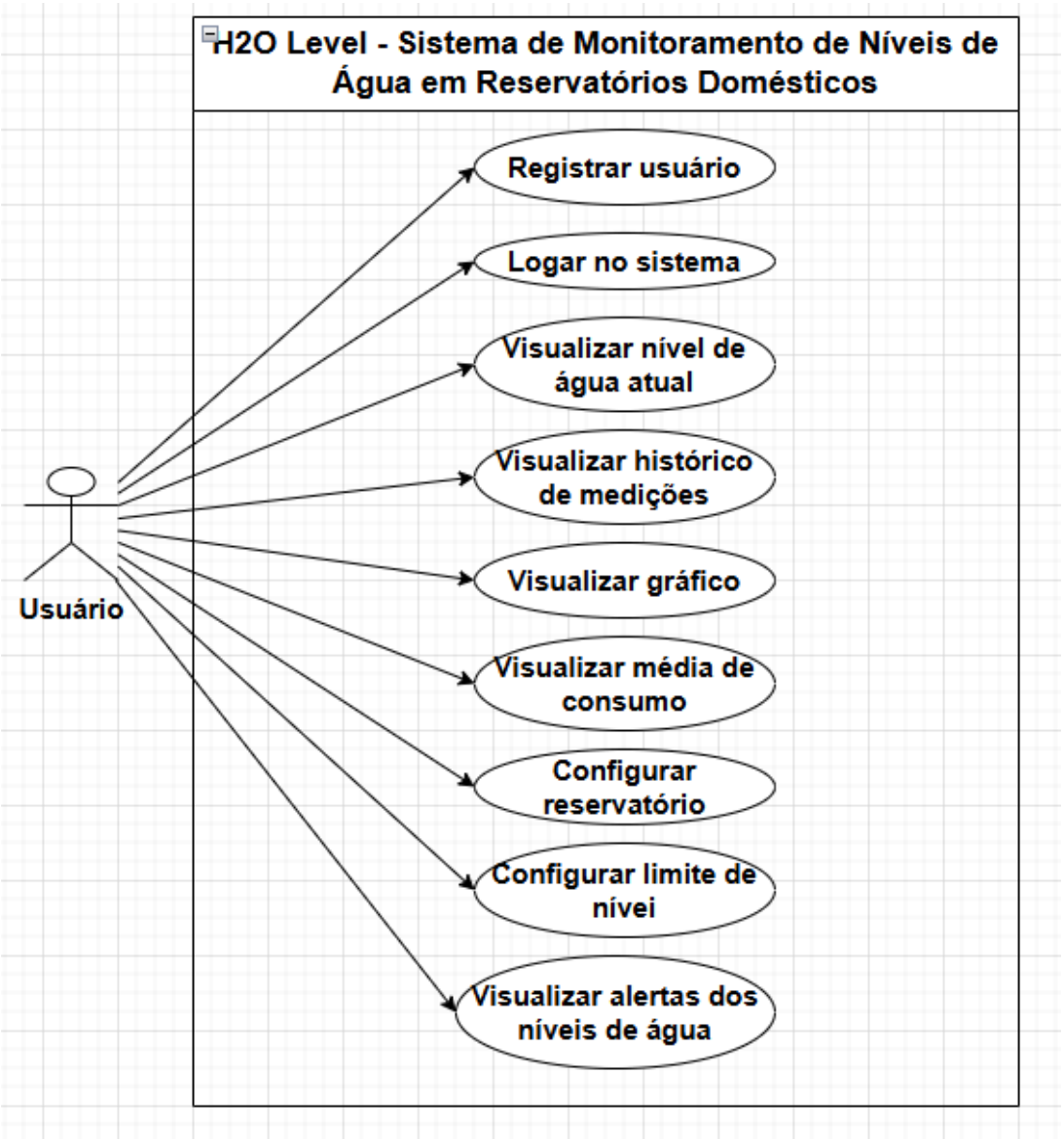
3.1 Projeto UML

Os projetos UML (Unified Modeling Language), ou Linguagem de Modelagem Unificada, utilizam uma linguagem padrão para modelar e documentar sistemas, principalmente aqueles orientados a objetos. Foi criada com o intuito de facilitar o planejamento, design e a manutenção de sistemas complexos.

A linguagem UML utiliza diagramas gráficos para representar a estrutura e o comportamento de um sistema. Permite criar diagramas que mostram, de forma clara, como os diferentes componentes de um sistema se relacionam e interagem, ajudando a visualizar desde a arquitetura do sistema até os fluxos de interação entre objetos.

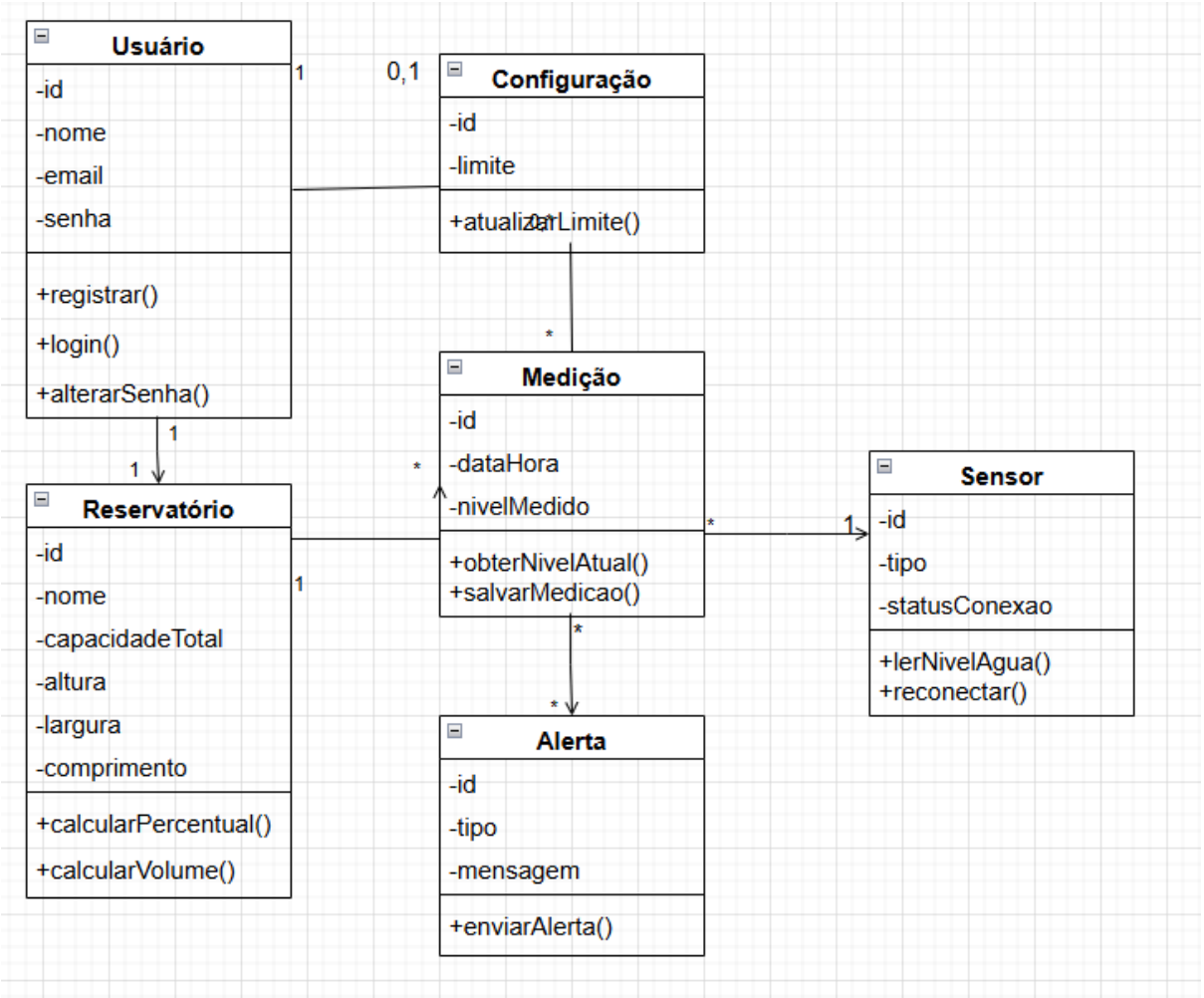
3.1.1 Diagrama de casos de uso

O diagrama de caso de uso serve para demonstrar o comportamento, definir o contexto e os requisitos de um sistema inteiro ou apenas das suas partes importantes. Descreve funções de alto nível e o escopo, ao permitir a visualização das interações entre o sistema e seus atores.



3.1.2 Diagrama de classes

O diagrama de classes serve para modelar objetos e a estrutura estática de um sistema. Representa as cópias do sistema ou subsistema, exibindo os relacionamentos entre os objetos, suas ações e serviços que fornecem.

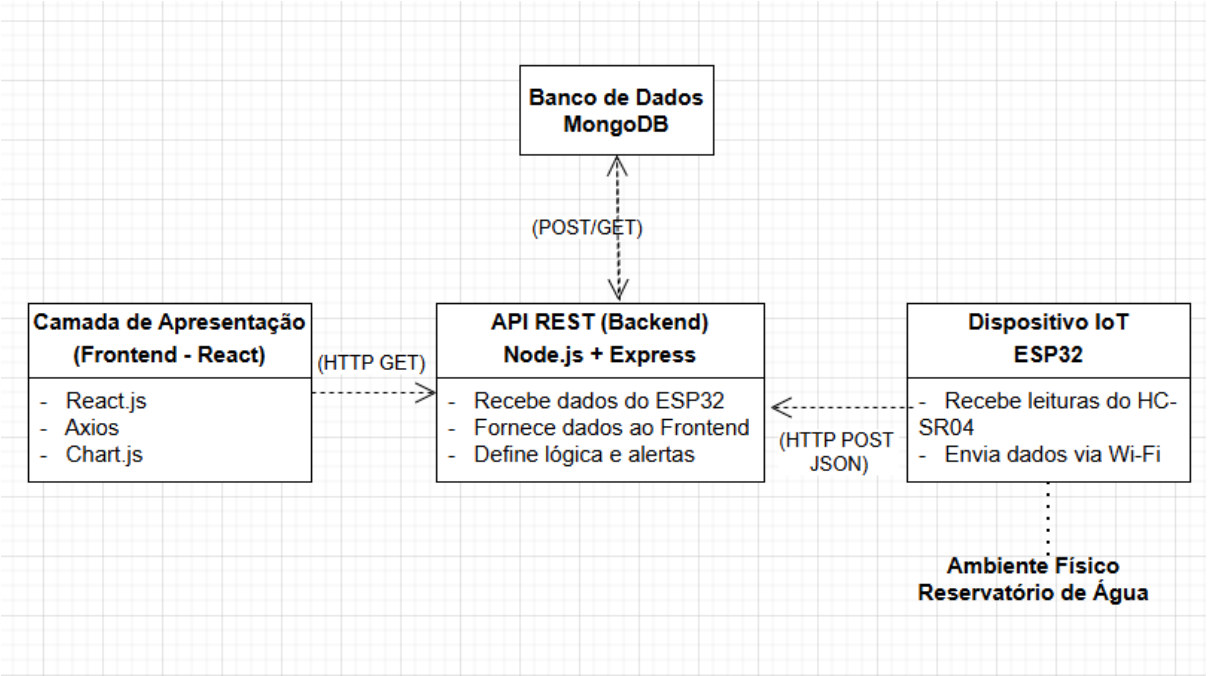


3.2 Visão Arquitetural

A arquitetura escolhida para o projeto foi a de três camadas, por possuir um ótimo desempenho no desenvolvimento de aplicações baseadas no modelo cliente-servidor. Neste modelo, as camadas (tiers) das aplicações são divididas em:

1) Camada de aquisição de dados, responsável por coletar e enviar os dados para o servidor.
2) Camada de processamento e armazenamento de dados, onde acontece a aplicação da lógica do sistema e armazenamento dos dados recebidos.
3) Camada de apresentação ou interface com o usuário, onde os dados são exibidos para visualização pelo usuário.

Figura 05 - Visão arquitetural do sistema

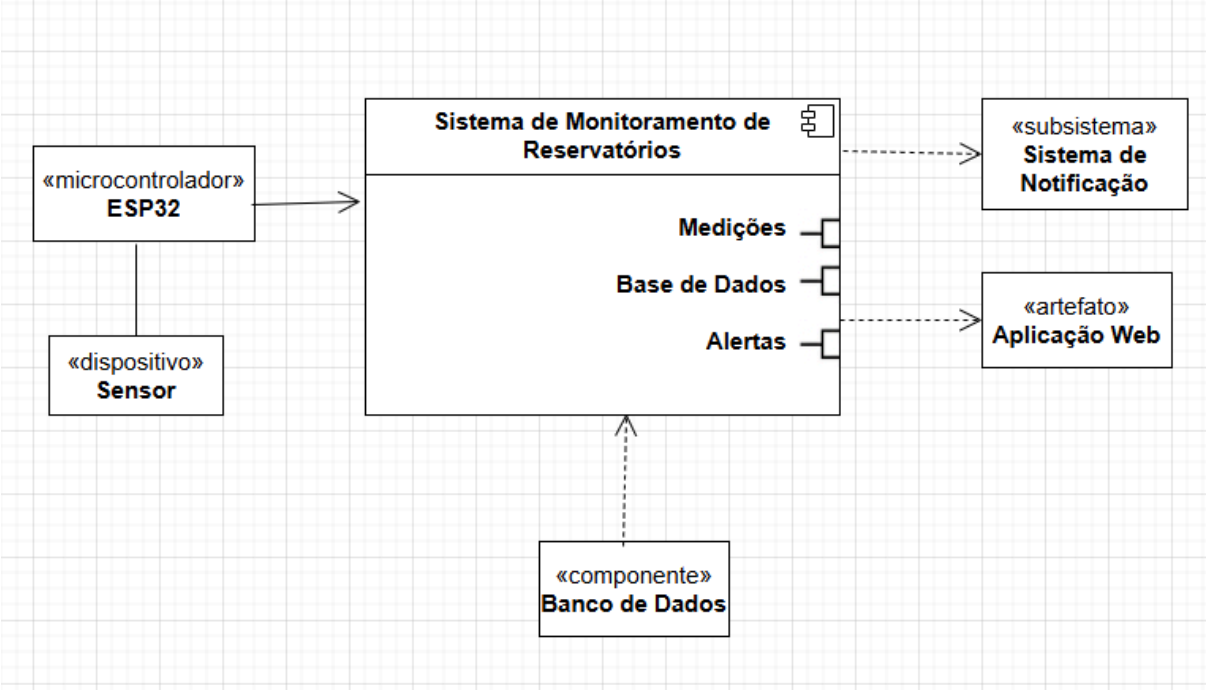


Fonte: Elaboração própria

As camadas possuem a sua própria infraestrutura, o que possibilita o desenvolvimento de cada uma delas simultaneamente, por equipes diferentes. Suas principais vantagens são:

- Separação de responsabilidades, o que aumenta a organização e facilita a manutenção.
- Escalabilidade melhorada, facilitando o dimensionamento de partes específicas da aplicação, como interfaces e backend.
- Maior confiabilidade, pois as camadas são desenvolvidas separadamente e funcionam como módulos independentes, que se conectam por endpoints.
- Facilidade na realização de testes.
- Segurança aprimorada: como o backend e o frontend não se comunicam diretamente, e sim, através do Axios, este, acaba conferindo uma camada extra de proteção. Outro ponto são as validações via token, para se ter acesso aos dados da aplicação.

Figura 06 - Visão dos componentes do sistema



Fonte: Elaboração própria

Abaixo, farei uma descrição detalhada de cada camada, suas responsabilidades, funcionamento e tecnologias envolvidas:

1) **Camada de aquisição de dados**

Responsável por coletar os dados referentes ao nível de água no reservatório.

Possui dois componentes principais:

- Sensor Ultrassônico HC-SR04: efetua medições periódicas no reservatório, para obter o volume total de água disponível.
- Microcontrolador ESP32: possui duas funções: acessar o sensor e se conectar a rede Wi-Fi para enviar as informações coletadas através deste, para o servidor.

Esta camada funciona da seguinte forma:

1. O microcontrolador ESP32 aciona o sensor para que este realize uma medição da distância entre o nível de água atual e o topo do reservatório.
2. O ESP32 processa o valor obtido e calcula o nível da água, em centímetros.
3. Realiza o envio das medições ao servidor junto com um timestamp, para registrar a data e hora de cada medição.

2) **Camada de processamento e armazenamento**

Responsável por coletar e manipular os dados das medições.

Os seus componentes são:

- Servidor Node + Express: recebe os dados enviados pelo microcontrolador ESP32 e caso necessário, gera os alertas. Também disponibiliza os endpoints que possibilitam ao frontend acessar os dados armazenados no banco de dados.
- Banco de dados MongoDB: armazena as leituras dos sensores (timestamps, valores medidos, ids, etc.), configuração do alerta (limite no qual as notificações serão enviadas), reservatórios, usuários, autenticação e os alertas gerados.

O funcionamento desta camada se inicia com:

1. Envio de um documento JSON pelo microcontrolador ESP32, que é recebido pelo servidor.
2. O Servidor Node fará a validação e registrará as medições, verificando o limite configurado pelo usuário, e assim, a necessidade de gerar alertas ou não.
3. Os dados coletados serão armazenados no MongoDB e ficarão disponíveis para serem acessados pelo React, via Axios.

3) Camada de apresentação

Exibe as informações da aplicação para o usuário.

Esta camada possui três grandes funcionalidades que fazem uso das seguintes tecnologias: frontend (React), gráfico (Chart;js e Rechart.js) e comunicação com a API (Axios).

A interface web exibirá os seguintes recursos:

- Tela de login e cadastro dos usuários.
- Painel (dashboard) com o nível atual do reservatório.
- Histórico e gráfico das medições.
- Configuração do reservatório e usuário.
- Média de consumo.

Quanto ao sistema de notificações, estas serão enviadas via e-mail.

A aplicação poderá ser acessada em qualquer dispositivo que possua conectividade com a internet através dos navegadores modernos, desde que estejam conectados na mesma rede do servidor.

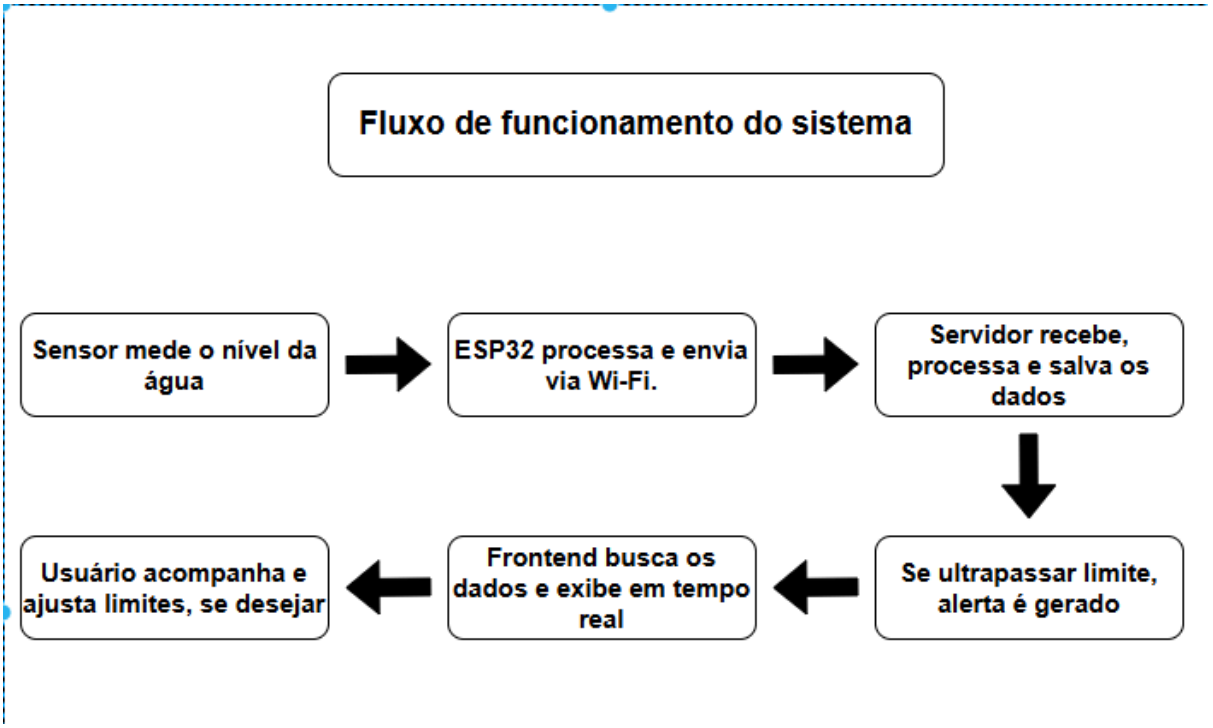
O funcionamento desta camada começa com a realização do login pelo usuário ou cadastramento, caso não possua um login. Após isso, a Interface acessa a API, exibindo o nível atual do reservatório, histórico de medições e o gráfico, além das telas de configuração e média de consumo. O usuário poderá configurar o limite do alerta e as notificações serão enviadas, caso estejam fora do limite definido..

Figura 07 - Tabela das tecnologias usadas no sistema

Pilha tecnológica do sistema	
Camada	**Tecnologia**
Aquisição	ESP32, HC-SR04, C++
Backend/API	Node.js (Express)
Banco de Dados	MongoDB
Frontend	React.js, Chart.js, Axios
Notificações	Nodemailer
Comunicação	HTTP REST

Fonte: Elaboração própria

Figura 08 - Fluxo de funcionamento do sistema



Fonte: Elaboração própria

3.3 Modelo de Banco de Dados

O banco de dados utilizado no projeto é o MongoDB. Este banco tem como particularidade ser orientado a documentos, ou seja, é um banco do tipo NoSQL. A sua escolha se deu devido ao grande volume de dados gerados pelas medições, que apresenta uma melhor performance neste modelo, de orientação a documentos ao invés do modelo entidade-relacionamento, que é o mais comumente usado.

Modelo lógico:

O modelo lógico é responsável por descrever, de forma conceitual, as entidades existentes no projeto, quais informações estas armazenam e as formas como se relacionam umas com as outras.

As entidades do banco são:

- **Usuário** - pessoas que acessam o sistema para visualizar informações (dados) e personalizar configurações..
- **Reservatório** - Objeto monitorado pelo sensor e local onde os dados são captados.
- **Medição** - registros enviados pelo microcontrolador ESP32, contendo os dados a serem analisados e armazenados.

Os relacionamentos ocorrem da seguinte forma:

- Um usuário pode possuir um reservatório.
- Cada reservatório pode gerar múltiplas medições.

Modelo físico:

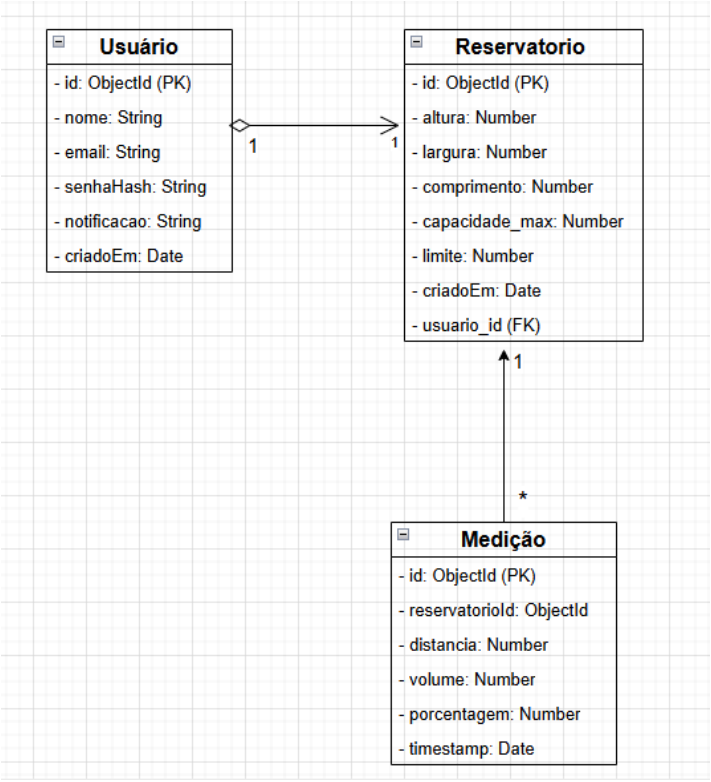
O modelo físico é responsável por estruturar a forma como os dados serão armazenados nos documentos JSON. Os usuários possuirão seus próprios documentos e, de forma aninhada, os reservatórios terão os seus dados armazenados como subdocumentos dentro dos documentos do usuário. As demais entidades serão mantidas em coleções separadas, a fim de otimizar a organização e buscas de dados.

Os critérios que definiram a escolha desta modelagem são:

- Dados aninhados (reservatório no usuário) - resulta em rápidos acessos. O usuário e seu reservatório são consultados juntos.
- Coleções separadas para medições e alertas - favorece a escalabilidade. Esses dados são separados para evitar documentos ainda maiores, uma vez que são gerados muito rapidamente. Desta forma, otimiza o tratamento dos dados.
- Referências por ID - Confere precisão na identificação dos dados requisitados.
- Indexação - índices são adicionados aos campos timestamp, TankId, e tipo para melhorar a performance nas buscas.

O MongoDB traz como vantagens a flexibilidade para alterações na sua estrutura, uma leitura otimizada dos dados mais usados juntos (usuário + reservatórios) e praticidade caso aumentem o volume dos dados gerados e o armazenamento de medições.

Figura 09 - Modelo lógico do banco de dados



Fonte: Elaboração própria

4. Testes de Software

4.1Projeto de Testes

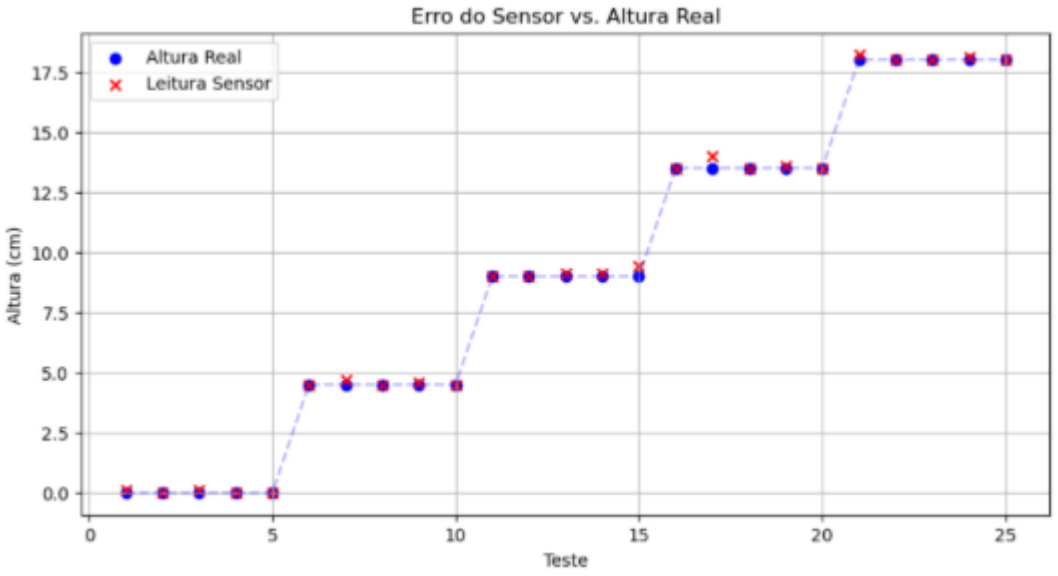
Como forma de validação do projeto, foram empregados 08 casos de testes. Sendo que 04 deles foram realizados manualmente e outros 04, automatizados, utilizando a ferramenta Artillery. Agora, farei a descrição de cada um e as conclusões individuais obtidas.

Testes Manuais:

1) Teste de Precisão do Sensor

O objetivo deste teste foi verificar se as medições coletadas pelo sensor coincidiam com os níveis reais.
O sensor HC-SR04 possui taxa de variação aceitável de até 3 mm para mais ou para menos. Ele é afetado por variações na temperatura e sons do ambiente.
Os testes foram realizados em 05 conjuntos de 05 testes cada, totalizando 25, nos seguintes níveis: 0% (vazio), 25%, 50%, 75% e 100% (cheio); Os dados obtidos estão demonstrados no gráfico de dispersão abaixo:

Figura 10 - Teste de precisão do sensor



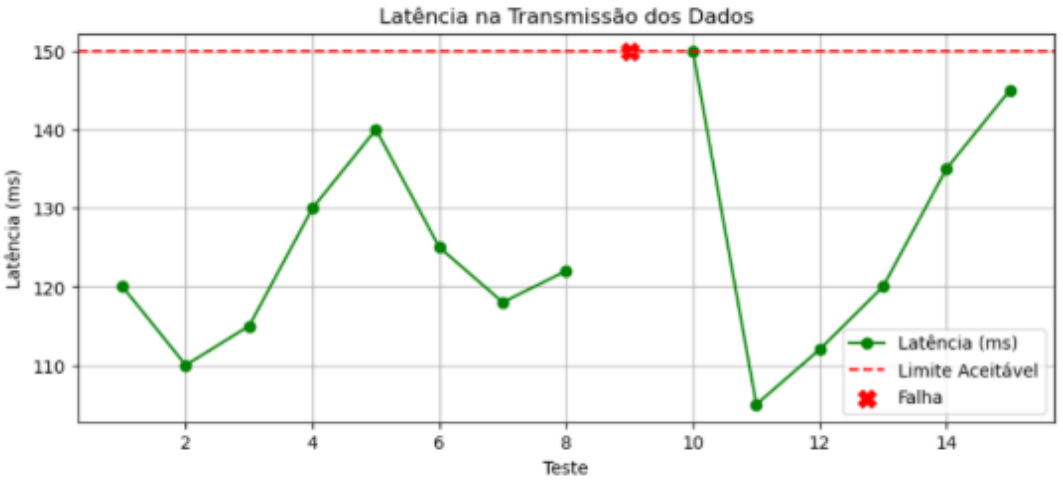
Fonte: Elaboração própria

Dos 25 testes, apenas 02 apresentaram variações maiores que os 3 mm aceitáveis. Foram os testes 15 e 17. Sendo assim, o sensor apresentou 92% de taxa de sucesso.
As conclusões obtidas são que o sensor apresenta um ótimo desempenho, em termos de precisão, em reservatórios de maior capacidade; pois, como as variações esperadas são diminutas, não afetam de forma consistente os resultados obtidos em situações reais. Já nos reservatórios de menores capacidades, essas imprecisões podem gerar variações significativas. No modelo utilizado nos testes, que possui 3,25 litros, somente a variação esperada do sensor gerou, em alguns casos, uma imprecisão de 1,6%. Enquanto num reservatório de 1000 litros, esta imprecisão geraria aproximadamente 0,2% de variação.

2) Teste de Transmissão de Dados

Este teste teve como objetivo medir a porcentagem de sucesso no envio das medições via Sensor HC-SR04 + Microcontrolador ESP32 para o Servidor Node.
O Microcontrolador possui uma taxa de alcance que varia de 20 a 100 metros, a depender do uso de antenas, obstáculos próximos e interferências de outras redes. Sua taxa de transmissão de dados é de 150 mbps.
Seguem os dados coletados. apresentados num gráfico de dispersão:

Figura 11 - Teste de transmissão dos dados



Fonte: Elaboração própria

Foram realizados 15 testes, sendo que houve apenas 01 falha de transmissão, conferindo uma taxa de 93% de sucesso. A média de tempo nas transmissões foi de 125 ms, sendo que o limite fixado foi de 150 ms.
A conclusão deste caso de teste é que o sensor precisa de uma rede estável para funcionar adequadamente e estar posicionado dentro da distância operacional informada. Atendendo a esses requisitos e não havendo oscilações na rede, a probabilidade de erro será na faixa dos 7% ou menor.

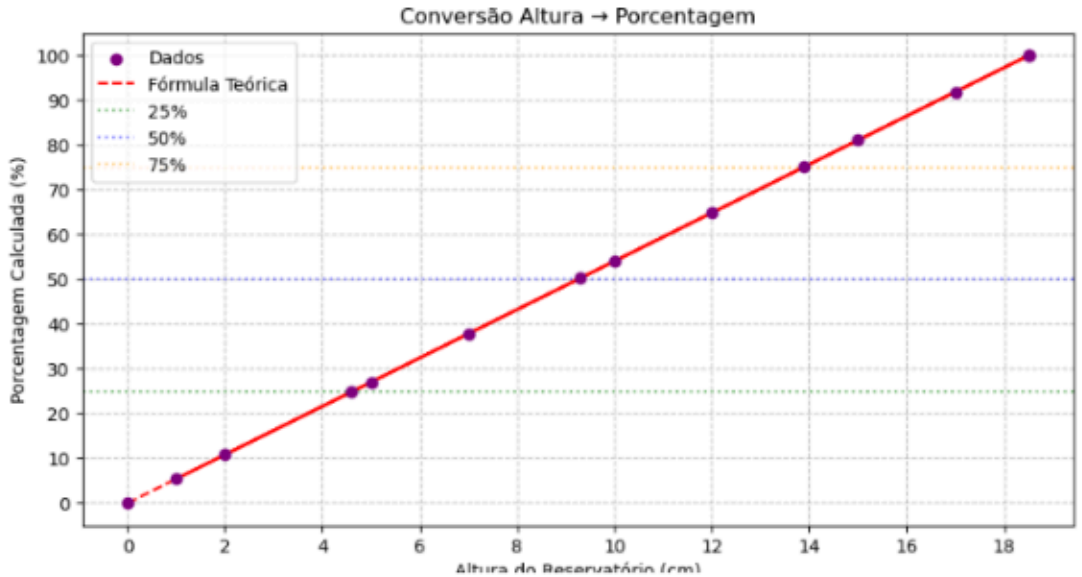
3) Teste de Cálculo da Porcentagem

O objetivo foi verificar se o cálculo dos volumes e porcentagens estavam coincidindo com os valores esperados.

- O volume total do reservatório é calculado a depender do tipo de sólido (retangular, cilíndrico ou pré definido). A fórmula de volume dos sólidos retangulares é: altura x largura x comprimento. Dos circulares: $\pi \times r^2 \times$ altura. Os pré-preenchidos, já possuem os valores definidos.
- A altura do nível de água no reservatório foi calculada usando a fórmula: distância = altura do reservatório - distância obtida pelo sensor.
- O volume do nível d'água atual foi obtido utilizando uma regra de três simples, já que a proporção é linear e sabemos as medidas do volume total do reservatório, altura do reservatório e altura do nível da água.
- A porcentagem de água atual do reservatório é obtida de forma análoga, já que conhecemos o volume total do reservatório, volume atual e a porcentagem total.

Abaixo, o gráfico de dispersão contendo os resultados:

Figura 12 - Teste de cálculo de porcentagem



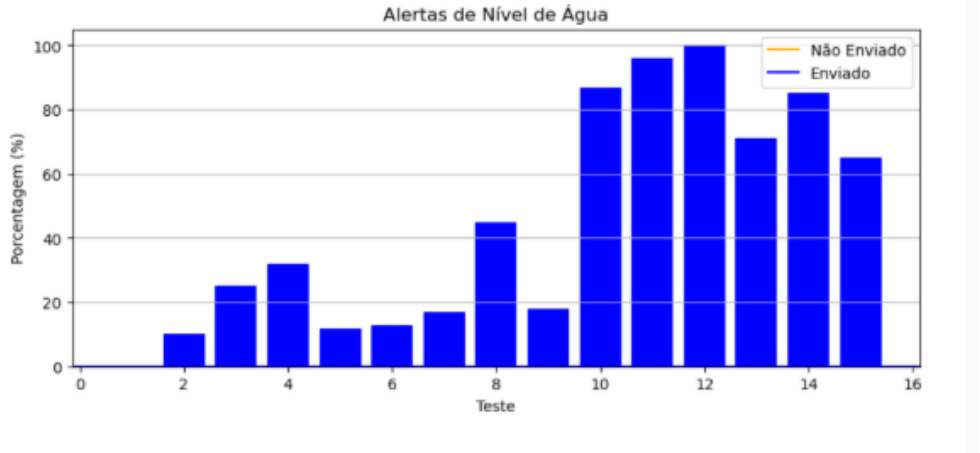
Fonte: Elaboração própria

A taxa de sucesso foi de 100%. O Sistema calculou corretamente as conversões e operações em todas as requisições.

4) Teste do Envio de Alertas por E-mail

O objetivo foi verificar se todos os alertas de nível de água gerados foram enviados corretamente, estando a função de envio dos alertas ativada.
O teste retornou 100% de sucesso. Todas as notificações foram entregues, variando apenas o tempo médio de resposta: 200 ms. Este tempo é considerado o limite aceitável nos servidores Node.
Segue o gráfico com os resultados:

Figura 13 - Teste de envio de alertas por e-mail



Fonte: Elaboração própria

A conclusão deste teste é que todas as notificações foram geradas e encaminhadas pelo Nodemailer, sem perdas. O que irá oscilar são os tempos de resposta, já que dependerá da estabilidade da rede do servidor e requisições existentes, estabilidade do cliente de e-mail configurado para enviar as notificações, etc.

Testes Automatizados:

Os 04 testes seguintes foram realizados utilizando a ferramenta Artillery. Ela se mostrou excelente e de fácil integração com o servidor Node. Além disso, é uma plataforma escalável, flexível e fácil de usar, ideal para testes de carga de nível de produção.

5) Teste de Carga

O objetivo do teste foi avaliar o desempenho do sistema sob condições de carga esperada e acima do esperado, simulando o comportamento num cenário de muitos usuários e requisições ocorrendo simultaneamente.
Obtive os seguintes dados:

Em 20 minutos de duração:

- 97.200 usuários criados
- 580.575 requisições enviadas
- 482 requisições/seg
- 401 ms de tempo médio de resposta
- Taxa de Sucesso de usuários criados: 99,46% (96.675)
- Falharam: 525

Todos os erros encontrados foram gerados num cenário de estresse, que detalho no próximo caso de teste. Foi a partir de 62 usuários criados por segundo que o sistema começou a apresentar lentidão. As conclusões obtidas no teste são de que o sistema conseguiu responder à grande volume de requisições com consistência, embora o tempo médio de resposta tenha crescido progressivamente. Se mostrou ideal para cargas moderadas e previsíveis (Cerca de 50 usuários/seg)

6) Teste de Estresse

O objetivo deste teste é avaliar a robustez e capacidade de um sistema, em suportar condições extremas e fora do comum. Isso inclui identificar pontos de falha e garantir que o sistema continue funcionando de forma satisfatória, mesmo sob pressão.
Os dados coletados foram:

Em 15 minutos de duração:

- Pico de 482 requisições/seg
- 525 erros de conexão
- Taxa de sucesso: 66,7%
- Falha: 193.350 requisições (33,3%)

Foi neste cenário de estresse que, nos picos de requisições (100 usuários criados por segundo), o sistema não conseguiu atender todas as solicitações. O servidor acabou quebrando e parou de atender as requisições.
A conclusão é que o sistema se mostrou eficiente em cenários de usos moderados e médios. Mantendo-se estável e sem recusas de conexões.

7) **Teste de Tempo de Resposta**

O objetivo do teste foi avaliar a rapidez com que o sistema recebeu às solicitações do usuário. Ou seja, tentou medir o desempenho do sistema em termos de quão rápido ele reage a diferentes condições e cargas de trabalho
Obtive os seguintes dados:

Cenário de estresse:

- Tempo médio de resposta: 401 ms
- Erro nas respostas do cliente: Média de 520 ms
- Erro nas respostas do servidor: Média de 213 ms
- Pico máximo de resposta: 3 segundos

Cenário de uso de moderado a médio:
:

- Tempo médio de resposta: 2.2 ms
- 95% responderam em até 5 ms

A conclusão deste teste é que a maioria das requisições foram aceitas num tempo aceitável. Porém, há evidência de lentidão em algumas requisições que, combinadas ao volume, ou seja, no cenário de estresse, impactaram na performance.

8) **Teste de Estabilidade**

Tem como objetivo medir o tempo que cada usuário levou para ter a sua requisição atendida. O tempo que levou para aceitar a requisição, processamento e resposta.
Os dados coletados foram:

Cenário de estresse:

- Tempo médio de sessão: 2415 ms
- Pico máximo das sessões: 7312 ms

Cenário de uso de moderado a médio:

- Tempo médio de sessão: 26.4 ms
- Pico máximo: 115.4 ms

Como este teste está interligado ao anterior, as conclusões são similares.

De forma resumida o sistema se saiu bem nos testes e se mostrou estável e confiável, em cenários de uso moderado e médio. A recomendação é que o Sistema seja utilizado num cenário de até 50 usuários enviando requisições por segundo; porém, analisando o escopo no qual funcionará, o Sistema não será submetido a sequer 10% da carga recomendada.

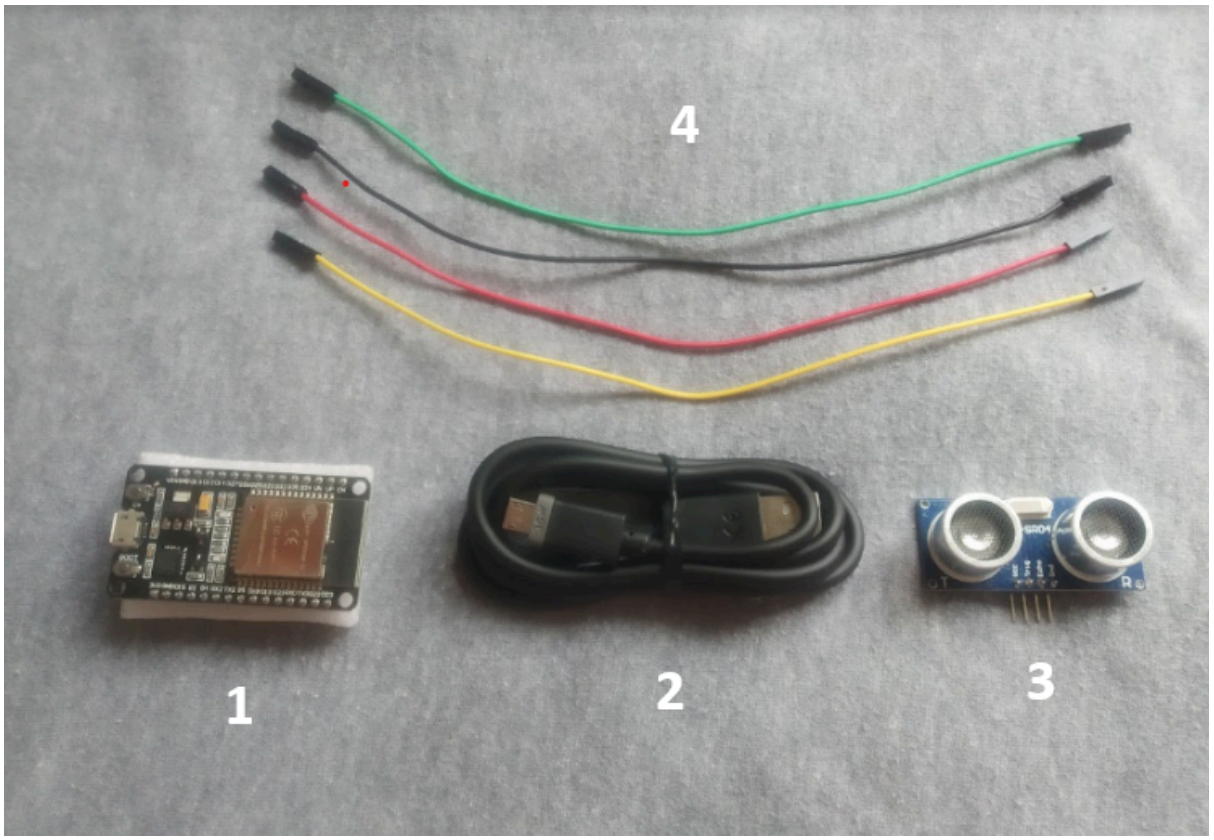
5. Implantação

5.1 Projeto de Implantação

O sistema foi pensado para ser executado no localhost dos usuários. Para tal, precisaremos dos seguintes hardwares e softwares:

Hardware:

Figura 14 - Hardware da Camada de aquisição de dados

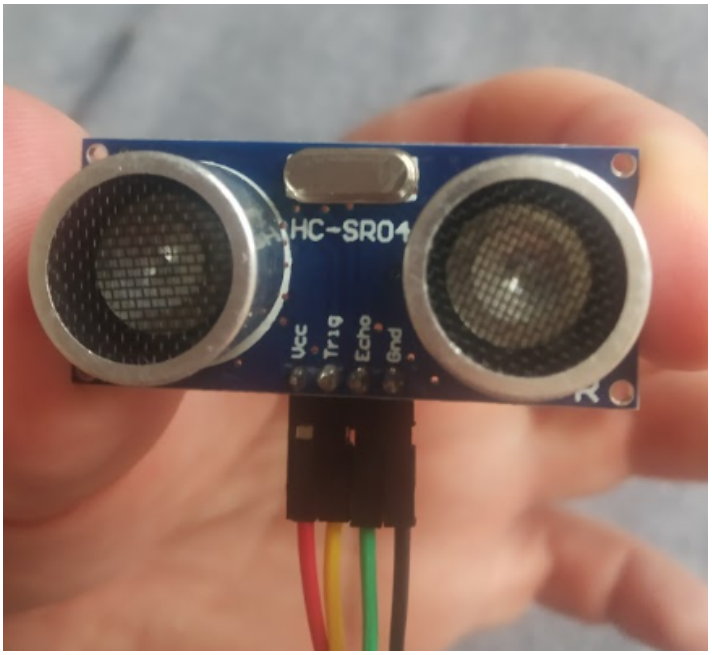


Fonte: Elaboração própria

1. 01 Microcontrolador ESP32.
2. 01 Cabo USB-C para alimentação e/ou comunicação com a ArduinoIDE.
3. 01 Sensor Ultrassônico HC-SR04.
4. 04 Jumpers fêmea-fêmea.

A escolha das cores não precisa seguir nenhum parâmetro. Os pinos digitais não precisam ser os mesmos.

Figura 15 - Jumpers conectados ao sensor



Fonte: Elaboração própria

Figura 16 - Jumpers conectados ao microcontrolador

Fonte: Elaboração própria

Utilizei uma garrafa como modelo, para demonstrar o funcionamento do sistema. Segue imagem:

Figura 17 - Modelo composto por reservatório + sensor + microcontrolador para simular ambiente real.

Fonte: Elaboração própria

Agora que já temos todo hardware e as devidas conexões efetuadas, seguiremos para o software.

Software:

Sistema H2O Level - Link para download: https://github.com/joaobrasjr/h2o-level. Este link permite o acesso ao diretório contendo todo o código do sistema (Programação do Microcontrolador ESP32, frontend e backend). Após o download do diretório, siga as instruções contidas no arquivo Readme do projeto.

ArduinoIDE - pode ser obtida no link: https://docs.arduino.cc/software/ide-v2/tutorials/getting-started/ide-v2-downloading-and-installing/. Esta IDE de programação do Arduino também é compatível com o microcontrolador ESP32. Foi por

ela que desenvolvi o código do microcontrolador e enviei as instruções para o mesmo. É uma interface de desenvolvimento gratuita e a mais completa e intuitiva, de programação, para o microcontrolador ESP32. Qualquer alteração no código, seja de lógica, redefinição de parâmetros, etc, necessitará que o código seja reinserido no ESP32. Caso não precise alterar o código contido nele, o funcionamento do ESP32 independe da IDE. Basta conectá-lo a uma fonte de energia e ele começará a seguir a programação.

Visual Studio Code - pode ser obtido no link: https://code.visualstudio.com/download. É um editor de código gratuito, robusto, com suporte a várias linguagens de programação, plugins, ferramentas de debug e várias outras funcionalidades. Utilizei este editor de código para construir o sistema. Caso opte por este mesmo editor, será por ele que iniciaremos os servidores backend e frontend.

MongoDB - Pode ser obtido no link: https://www.mongodb.com/docs/manual/installation/.Este banco de dados será responsável por armazenar todas as informações geradas pelo sistema H2O Level. Além de ser gratuito, simples e estável, foi escolhido por ter uma ótima integração com modelo de dados não relacionais, ou seja, que usam formulários de dados aninhados (minha lógica do banco de dados).

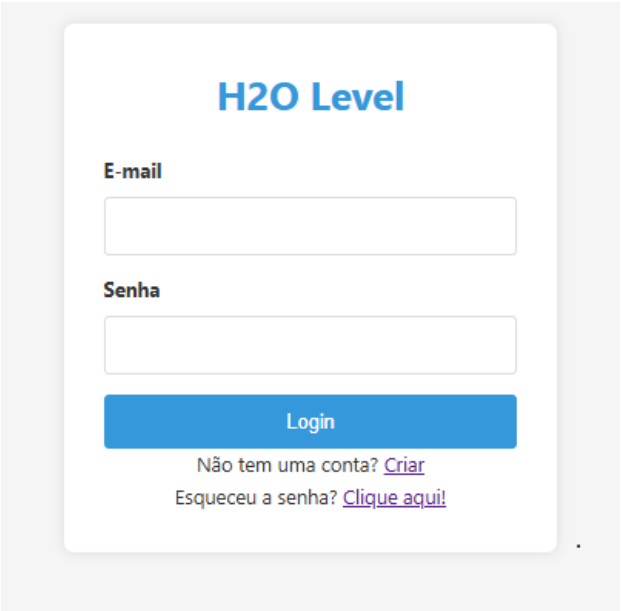
Estas são as tecnologias necessárias para o funcionamento do Sistema. Os passos seguintes deverão ser seguidos para se chegar a tela inicial do sistema:

1. Execute em modo administrador o MongoDB Compass e conecte o banco de dados.
2. Execute o Visual Studio Code, carregue o diretório do projeto (esta ação só precisa ser realizada uma vez, pois, sempre que o Visual Studio Code for inicializado, já entrará no diretório do último projeto acessado) e inicialize os servidores backend e frontend.
3. Após isso, o Sistema será exibido numa aba do navegador padrão do dispositivo.

6. Manual do Usuário

Seguindo os passos listados anteriormente, acessaremos a primeira tela do sistema: a tela de Login.

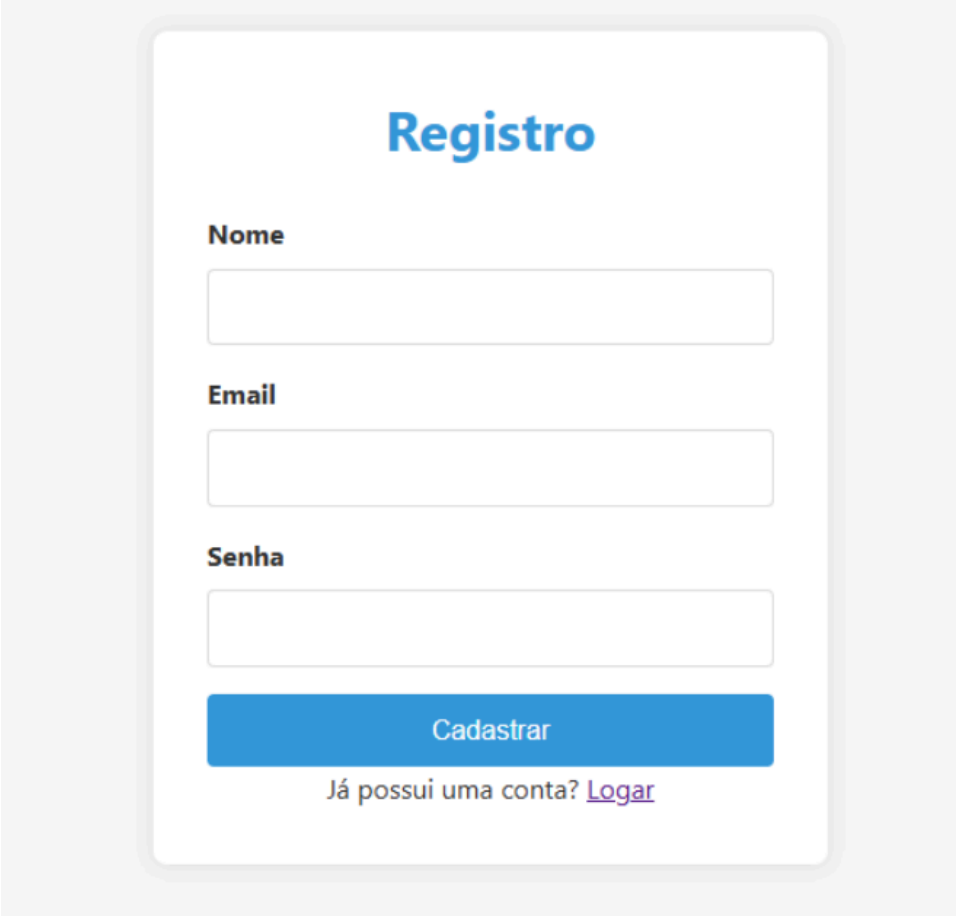
Figura 18 - Tela de Login



Fonte: Elaboração própria

Se o usuário já possuir um registro, basta informar o e-mail e senha cadastrados para ter acesso às funcionalidades do Sistema. Caso não, clique em “Criar” e será redirecionado para a tela de Registro.

Figura 19 - Tela de Registro



Fonte: Elaboração própria

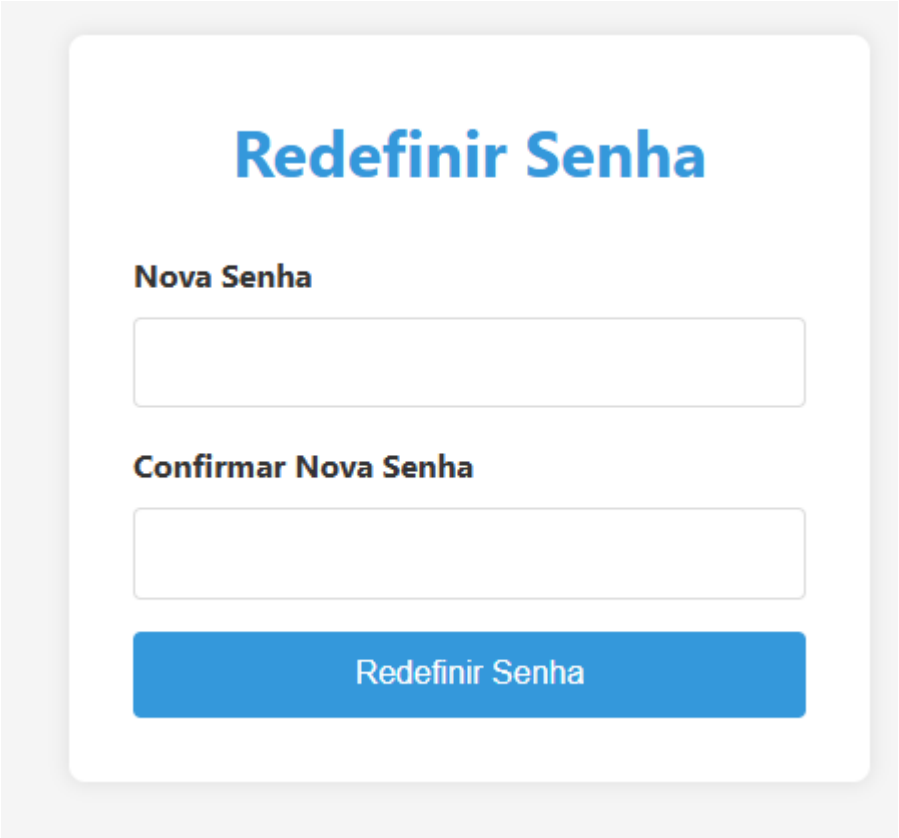
Nesta tela, é possível cadastrar um usuário informando um nome, e-mail e senha. Após concluir esta ação, o usuário será cadastrado no Sistema e redirecionado à tela de Login. Ou, caso prefira, pode clicar em “Logar” que será redirecionado manualmente. Outra funcionalidade que pode ser acessada via tela de login é a de redefinição de senha. Para acessá-la, clique em “Clique aqui” e você será redirecionado para a seguinte tela:

Figura 20 - Tela de Recuperação de Senha

Fonte: Elaboração própria

Informe o e-mail de um usuário previamente cadastrado no sistema e após a conferência, verificando se o usuário existe, será enviado um e-mail contendo um link para redefinição de senha, que nos levará para a seguinte tela:

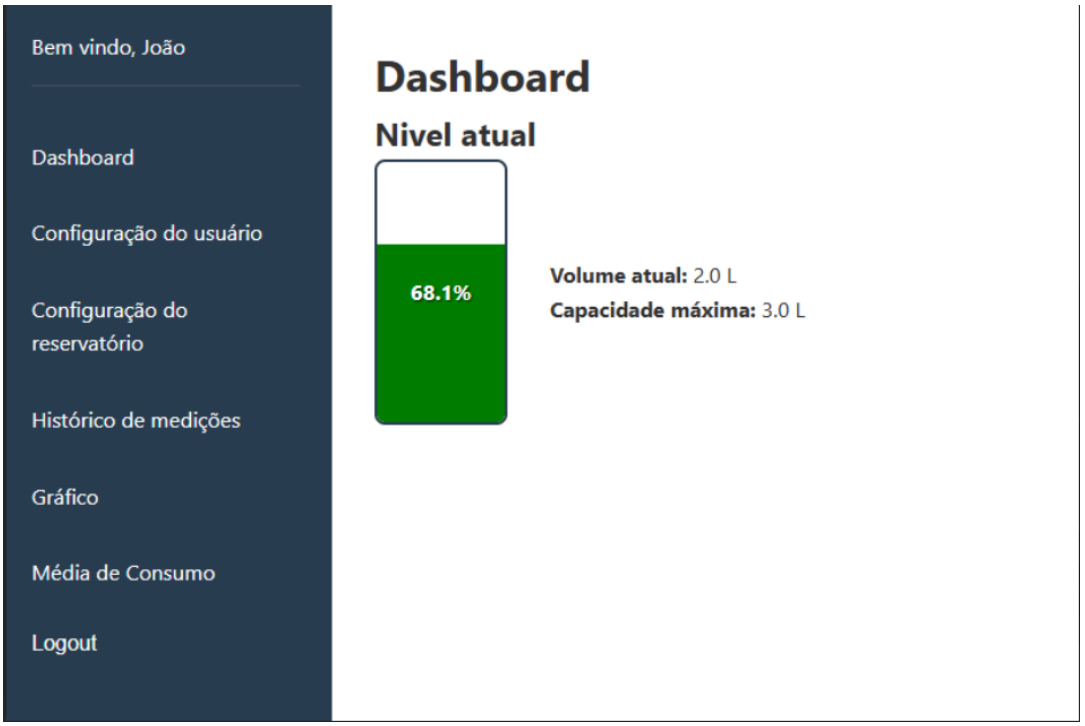
Figura 21 - Tela de Redefinição de Senha



Fonte: Elaboração própria

Digite a nova senha, confirme e clique no botão “Redefinir Senha” e a nova senha substituirá a antiga. Após esta ação, você será redirecionado para a tela de login. Concluindo-o, estaremos na tela principal do sistema, a Dashboard.

Figura 22 - Tela Principal - Dashboard



Fonte: Elaboração própria

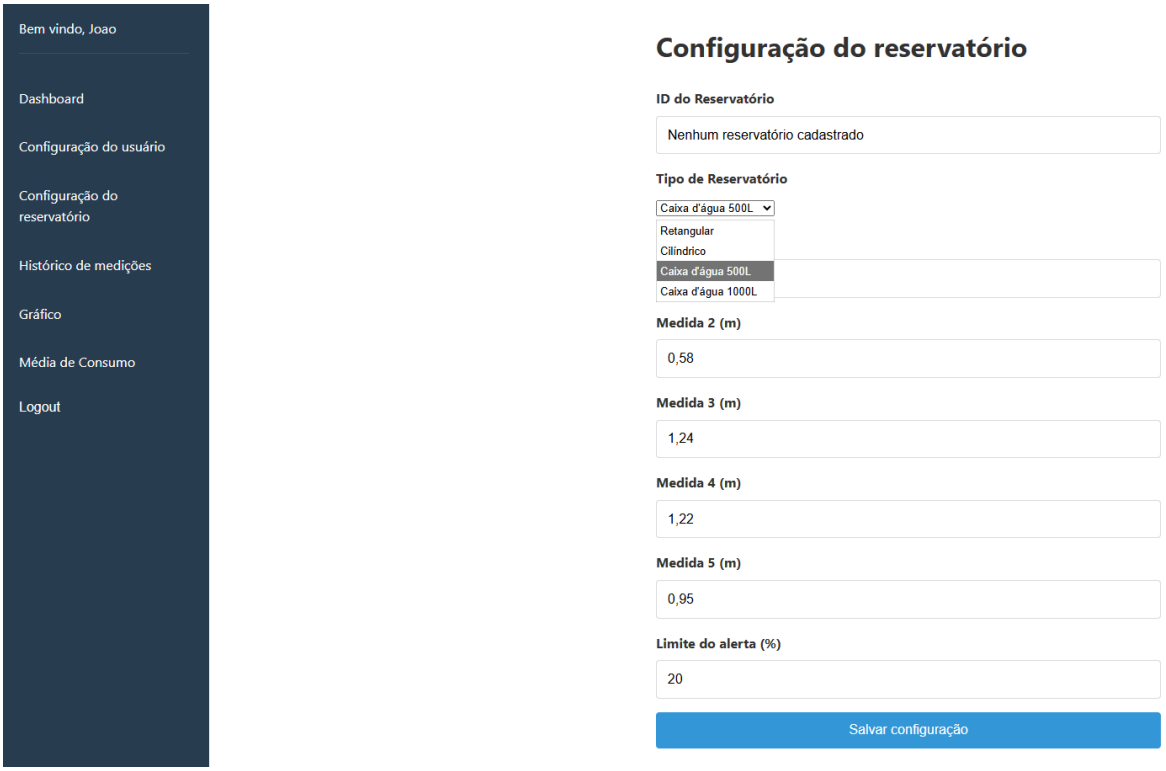
Esta tela mostra a principal funcionalidade do Sistema, que é o monitoramento em tempo real, onde exibe os dados do nível atual do reservatório, em porcentagem e litros; e capacidade máxima. No menu à esquerda, estão as funcionalidades do sistema. A primeira delas é a de configuração do usuário.

Figura 23 - Tela de Configurações do Usuário

Fonte: Elaboração própria

Aqui é possível alterar a senha de acesso, ativar/desativar o envio do alerta por e-mail e excluir a conta do usuário, bem como todos os dados vinculados a ele. A tela seguinte é a de configuração do reservatório.

Figura 24 - Tela de Configuração do reservatório



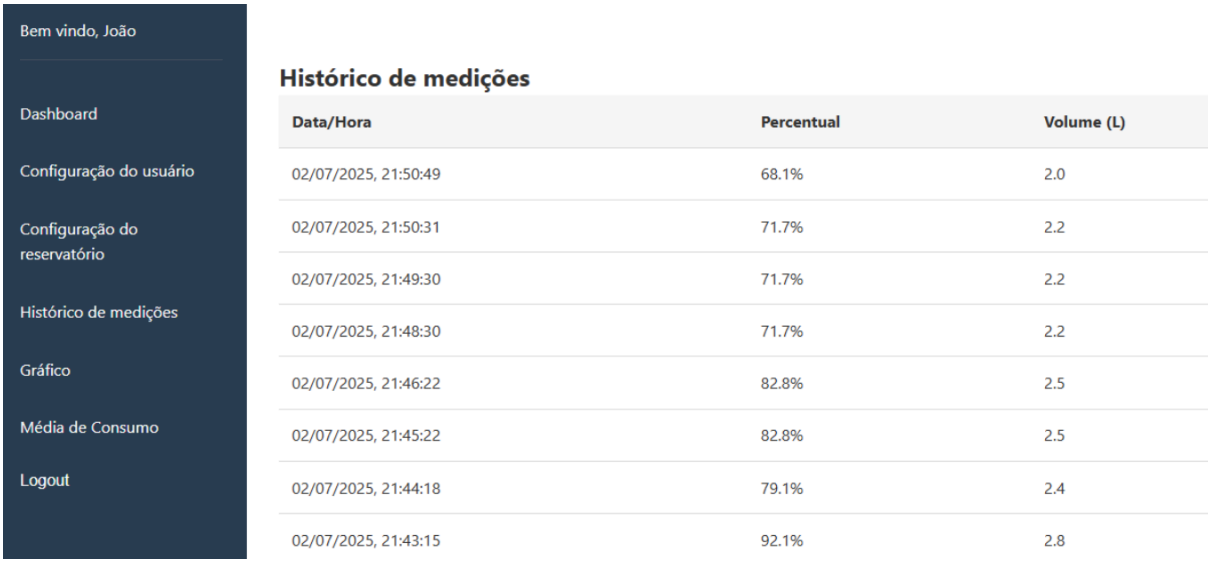
Fonte: Elaboração própria

Esta tela permite a criação do reservatório no banco de dados, bastando selecionar o modelo compatível com o da sua residência e informar as dimensões. Os modelos de 500 e 1000 litros já possuem as dimensões preenchidas com base nos reservatórios azuis amplamente vendidos e encontrados em qualquer casa de materiais de construção . Após a criação do reservatório, digitando as dimensões ou apenas selecionando-as e clicando em “Salvar configuração”, será possível visualizar o ID do reservatório. Este ID permitirá o monitoramento e envio das medições para o servidor e, consequentemente, banco de dados.
Também é possível definir a porcentagem do nível de água do reservatório com que, ao ser ultrapassada, os alertas serão enviados. Como a lógica do Sistema prevê a criação de apenas um reservatório, ao alterar quaisquer das medidas, o reservatório será atualizado.

O passo seguinte é acessar o software ArduinoIDE e carregar o arquivo contido no diretório do Sistema, chamado esp32_prog. Lá no final, na string do Payload, insira a sequência contida no campo “ID do Reservatório”, compile e envie o código para o ESP32. Após isso, posicione o sensor no reservatório, conecte-o a uma fonte de energia e ele começará a enviar as medições.

Na tela principal, da Dashboard, agora será possível visualizar a última medição. No menu, Histórico de medições, será possível acessar o histórico das 100 últimas medições, ordenados das mais recentes para as mais antigas e,acessar um botão no final da tela, onde será possível apagar todo o histórico de medições do usuário logado no sistema, conforme tela abaixo:

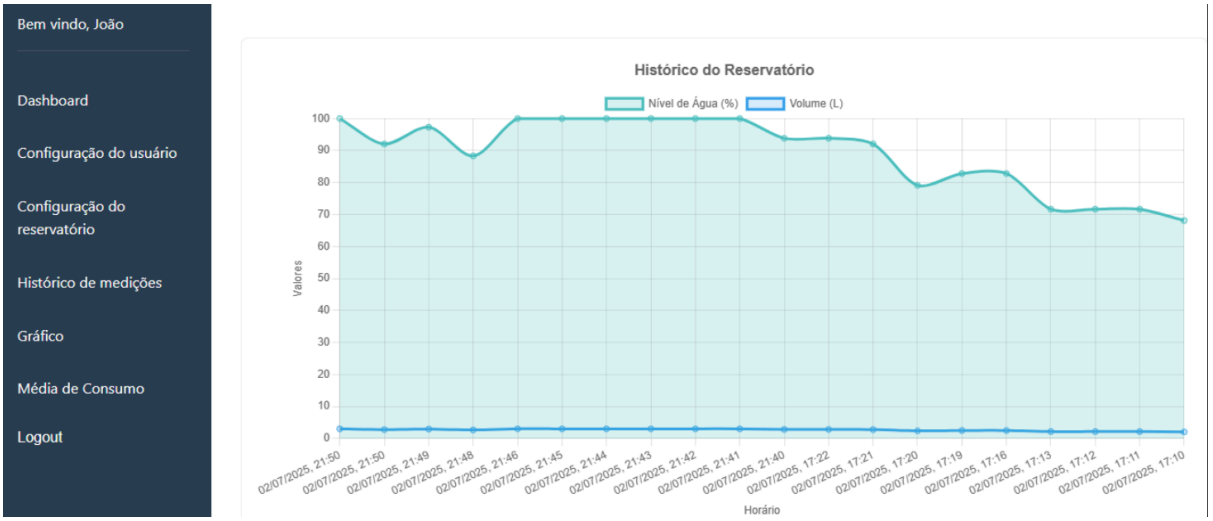
Figura 25 - Tela de Histórico de medições



Fonte: Elaboração própria

Em Gráfico, são exibidos os dados do histórico de medições de forma visual, mostrando as 30 últimas medições, em ordem decrescente. Os dados são obtidos através de uma varredura no banco de dados, buscando por todas as medições associadas ao ID do usuário e usando os elementos da biblioteca chart.js para construção e exibição do gráfico.

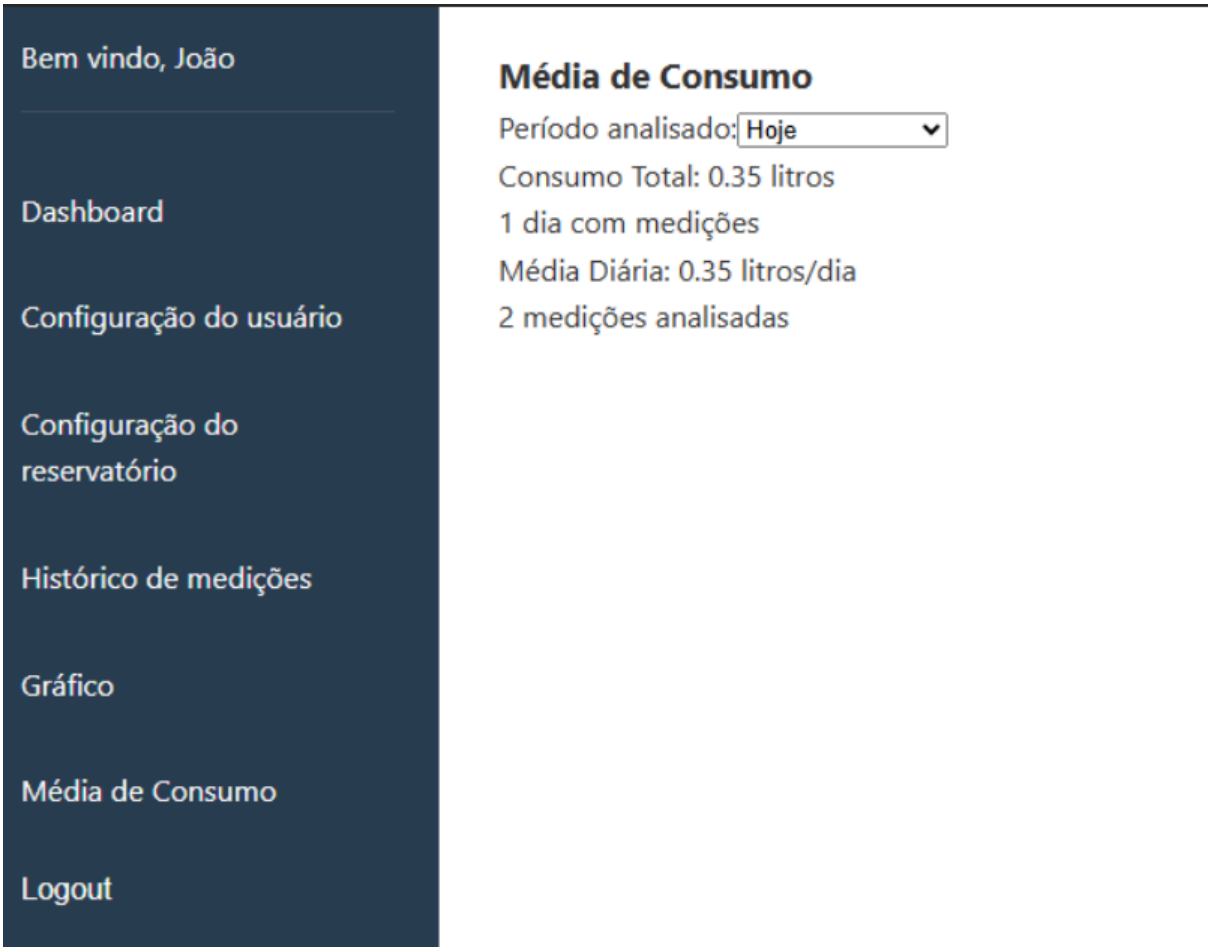
Figura 26 - Tela do Gráfico



Fonte: Elaboração própria

Por fim, temos a última funcionalidade, a Média de Consumo. Onde será possível observar o consumo total do dia, dos últimos 7 dias e últimos 30 dias. Além de uma média de consumo diária.

Figura 27 - Tela da Média de consumo



Fonte: Elaboração própria

A opção Logout desloga o usuário e retorna a tela de login.

Essas são todas as funcionalidades do Sistema H2O Level.

Considerações Finais

O processo de desenvolvimento do Sistema H2O Level permitiu demonstrar, de forma prática, o uso da tecnologia no combate ao desperdício de água e uso consciente dos recursos hídricos, nas residências. Suas tecnologias, em conjunto, possibilitaram o desenvolvimento de uma solução funcional, capaz de monitorar reservatórios de água em tempo real, detectar problemas estruturais precocemente, emitir alertas de nível e permitir acesso a dados históricos de consumo, ajudando na tomada de decisão sobre a melhor forma de se utilizar a água.
Apesar dos resultados positivos, o H2O Level apresenta espaço para melhorias com acréscimo de novas funcionalidades e expansão de algumas já presentes, como as notificações.
Conclui-se que o H2O Level representa uma importante medida para o uso racional da água, no ambiente doméstico, contribuindo para a sustentabilidade e maior conscientização dos seus usuários sobre a importância de se preservar este recurso indispensável à sobrevivência de qualquer ser vivo.

Projetos Futuros

Nas etapas de desenvolvimento do H2O Level, várias ideias de aprimoramentos e novas funcionalidades foram surgindo, como:

- Aplicativos móveis que não tenham limitações quanto a só poderem acessar o Sistema, se estiverem na mesma rede do servidor.
- Implementação de notificações Push.
- Integração com assistentes virtuais e inteligências artificiais.
- Integração com válvulas e bombas, automatizando assim o abastecimento dos reservatórios.
- Monitoramento de múltiplos reservatórios.
- Relatórios detalhados de consumo, sugerindo comparações com períodos específicos e ações para economia.

Essas melhorias e novas funcionalidades possuem o potencial de expandir ainda mais o impacto do H2O Level, tornando-o relevante para diferentes perfis de usuários e integração com outros serviços e aplicações.

Agradecimentos

Agradeço a todos os professores do curso, que com toda dedicação e ensinamentos, foram fundamentais para a conclusão deste projeto.
Agradeço também a todos os colegas do curso, por toda ajuda e companheirismo ao longo da formação.
Um agradecimento especial ao meu orientador, Professor AC, que embarcou neste projeto desde o início. Sempre disponível, dando ideias e me guiando com paciência e sabedoria até finalmente concluí-lo.
Agradeço a minha namorada, por todo apoio, ajuda, incentivos diversos e por sempre me manter motivado a correr atrás dos meus sonhos e em especial, concluir este Curso.
Por fim e não menos importante, sou profundamente grato a 3 pessoas, que sempre fizeram parte da minha vida e me apoiaram incondicionalmente em todas as escolhas que fiz. Minha Madrinha, que mesmo a distância, sempre torceu por mim e me apoiou nas mais variadas situações; Minha Mãe, que por muitas vezes abdicou de si para que eu pudesse realizar meus objetivos e sempre me fez sentir que não há amor maior que o de Mãe! Obrigado por tudo, Minha Mãe… Por tanto apoio e pela diferença que a Senhora faz na minha vida até hoje, E finalizando, Minha Avó, que infelizmente não está mais aqui, cujo maior sonho era me ver formado. Seu sonho finalmente se realizou, Minha Avó! Sei que a Senhora olhou por mim e acompanhou essa trajetória, de onde quer que esteja! Obrigado por todos os ensinamentos e tudo que a Senhora sempre fez por mim.

Referências

[1] TRATA BRASIL. Mais da metade da água potável é desperdiçada na região metropolitana de Salvador. Disponível em: https://tratabrasil.org.br/mais-da-metade-da-agua-potavel-e-desperdicada-na-regiao-metropolitana-de-salvador/. Acesso em: 03 abr. 2025.
[2] G1. Falta de água atinge bairros de Salvador. Disponível em: https://g1.globo.com/ba/bahia/noticia/2025/03/05/falta-agua-salvador.ghtml. Acesso em: 03 abr. 2025.

[3] CORREIO24HORAS. Sete bairros de Salvador terão fornecimento de água suspenso nesta quarta.Disponível em: https://www.correio24horas.com.br/minha-bahia/sete-bairros-de-salvador-terao-fornecimento-de-agua-suspensos-nesta-quarta-0425. Acesso em: 26 abr. 2025.
[4] CORREIO24HORAS. Fornecimento de água será interrompido em 60 bairros de Salvador nesta quarta. Disponível em: https://www.correio24horas.com.br/minha-bahia/fornecimento-de-agua-sera-interrompido-em-60-bairros-de-salvador-nessa-quarta-1024. Acesso em: 26 abr. 2025.
[5] BERMUDES, Larissa Pereira; VIEIRA, João Paulo da Silva; SILVA, Mirella Gonçalves da Fonseca Miranda da; FREITAS, Alexandre Adler Cunha de. Monitoramento de nível de água e detecção de falhas através do sistema IoT. Vitória, 2024. Disponível em: https://multivix.edu.br/wp-content/uploads/2024/09/MONITORAMENTO-DE-NIVEL-DE-AGUA-E-DETECCAO-DE-FALHAS-ATRAVES-DO-SISTEMA-IOT.pdf. Acesso em: 15 abr. 2025.
[6] ALVES, Gabriel Belizário. Monitoramento do nível de água em reservatórios residenciais utilizando sensor ultrassônico. João Pessoa, 2023. Disponível em: https://repositorio.ifpb.edu.br/jspui/bitstream/177683/3459/1/MONITORAMENTO%20DO%20N%C3%8DVEL%20DE%20%C3%81GUA%20EM%20RESERVAT%C3%93RIO%20RESIDENCIAIS%20UTILIZANDO%20SENSOR%20ULTRASS%C3%94NICO%20-%20GABRIEL%20BELIZ%C3%81RIO%20ALVES.pdf. Acesso em: 15 abr. 2025.
[7] SOUZA, Ivo Martins de. Sistema para monitoramento de reservatórios de água – AquaMeasure. Hortolândia, 2017. Disponível em: https://hto.ifsp.edu.br/portal/images/thumbnails/images/IFSP/Cursos/Coord_ADS/Arquivos/TCCs/2017/TCC_Ivo_Martins_de_Souza_A1420216.pdf. Acesso em: 23 abr. 2025.
[8] ALI, Mohamad Sadeque Abou. Dispositivos de medição de consumo de água usando conceitos de IoT. Natal, 2022. Disponível em: https://repositorio.ufrn.br/jspui/bitstream/123456789/50593/1/TCC_Mohamad.pdf. Acesso em: 19 abr. 2025.
[9] ALGHAMDI, Atheer M.; KHAIRULLAH, Enas F.; AL MOJAMED, Mohammad. LoRaWAN Performance Analysis for a Water Monitoring and Leakage Detection System in a Housing Complex. Sensors, Jeddah, 2022. Disponível em: https://www.mdpi.com/1424-8220/22/19/7188. Acesso em: 16 abr. 2025.
[10] ESPRESSIF. Getting Started with ESP-IDF. Disponível em: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/index.html. Acesso em: 04 abr. 2025.
[11] USINAINFO. ESP32 – Tutorial com primeiros passos. Disponível em: https://www.usinainfo.com.br/blog/esp32-tutorial-com-primeiros-passos/. Acesso em: 04 abr. 2025.
[12] HOW TO MECHATRONICS. Ultrasonic Sensor HC-SR04 with Arduino. Disponível em: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/. Acesso em: 06 abr. 2025.
[13] ARDUINO. Software. Disponível em: https://www.arduino.cc/en/software/. Acesso em: 06 abr. 2025.
[14] EXPRESS. Express – Framework web Node.js. Disponível em: https://expressjs.com/pt-br/. Acesso em: 06 abr. 2025.
[15] IBM. O que é MongoDB? Disponível em: https://www.ibm.com/br-pt/topics/mongodb. Acesso em: 06 abr. 2025.
[16] MONGOOSE. Getting Started. Disponível em: https://mongoosejs.com/#getting-started. Acesso em: 07 abr. 2025.

[17] AXIOS. Introdução ao Axios. Disponível em: https://axios-http.com/ptbr/docs/intro. Acesso em: 07 abr. 2025.
[18] NODEMAILER. Using Gmail. Disponível em: https://nodemailer.com/usage/using-gmail. Acesso em: 04 jul. 2025.
[19] REACT. Learn React. Disponível em: https://react.dev/learn. Acesso em: 07 abr. 2025.
[20] CHART.JS. Getting Started. Disponível em: https://www.chartjs.org/docs/latest/getting-started/. Acesso em: 08 abr. 2025.
[21] BOOTSTRAP. Introdução ao Bootstrap 5.3. Disponível em: https://getbootstrap.com/docs/5.3/getting-started/introduction/. Acesso em: 08 abr. 2025.
[22] W3SCHOOLS. CSS Introduction. Disponível em: https://www.w3schools.com/css/css_intro.asp. Acesso em: 04 jul. 2025.