UNIVERSIDADE FEDERAL DE ITAJUBÁ

CIÊNCIA DA COMPUTAÇÃO

Elaboração do Projeto de Monitoramento e Controle de Hidroponia

2020030477 - Flávio Augusto Aló Torres
2020007290 - Lucas Batista Pereira
2020002777 - Marcelo Robert Santos

COM242 - SISTEMAS DISTRIBUÍDOS

Prof. Rafael Frinhani

INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
UNIFEI - Itajubá

2023

UNIVERSIDADE FEDERAL DE ITAJUBÁ
INSTITUTO DE MATEMÁTICA E COMPUTAÇÃO
CIÊNCIA DA COMPUTAÇÃO
COM242 - SISTEMAS DISTRIBUÍDOS

Elaboração do Projeto de Monitoramento e Controle de Hidroponia

1 Introdução

1.1 Problema e Solução Apresentada

A tecnologia está presente em vários ramos industriais atuais e um destes ramos é a agricultura e plantação, onde a tecnologia auxilia o aumento da produção e da qualidade dos produtos, além de facilitar a gestão dos processos.

O projeto apresentado visa modelar e implementar uma solução para um sistema de hidroponia (um cultivo de plantas sem utilização de terra, apenas na água) auxiliada pela tecnologia, onde a monitoração e controle dos fatores que tratam o crescimento da planta são gerenciados em um sistema distribuído.

Tal sistema é composto por três componentes, o primeiro é um microcontrolador no local que irá monitorar e enviar as informações dos sensores presentes no cultivo e irá receber as operações de gestão dos sensores. O segundo é um servidor que irá manter em um banco de dados os logs das informações e comandos passados no sistema e poderá relatar esses dados para os clientes que solicitarem tais relatórios. Por fim, o terceiro é um cliente móvel que recebe as informações do microcontrolador, envia os comandos para modificar os atuadores que irão controlar as propriedades do local e pede e recebe os relatórios do servidor.

1.2 Forma de Implementação

A comunicação entre esses componentes foi realizada através de um sistema de mensagens baseada em tópicos, publicações e inscrições. Algumas funções do sistema poderiam ser realizadas através de outras formas de comunicação - como o envio do relatório diretamente para o cliente que o requisitou - porém como não serão muitos dados e para fins de simplificação, toda a comunicação é realizada por mensagens.

Quanto aos dados, eles são gerados através de sensores que representam os sensores reais da hidroponia e serão armazenados em um banco de dados local no servidor *backend*.

De forma mais específica, o microcontrolador utilizado foi um ESP8266 com código em C++, o servidor *backend* foi implementado na linguagem Java utilizando o SpringBoot como ferramenta de inclusão de API's e postgresql como SGBD. O cliente móvel é um aplicativo Android criado utilizando o *framework* Flutter que trabalha com a linguagem Dart. A comunicação entre os componentes foi feita utilizando a plataforma da AWS e o serviço Amazon MQ (*Message Queue*) sobre Active MQ, o microcontrolador e servidor *backend* utilizando o protocolo MQTT para a comunicação e o cliente móvel utiliza o protocolo Stomp através de Web Socket.

1.3 Responsabilidades

A divisão do grupo foi feita da seguinte maneira: Flávio Torres ficou responsável pela implementação do microcontrolador, Lucas Pereira ficou responsável pela implementação do servidor *backend*, e Marcelo Santos ficou responsável pela implementação do aplicativo móvel. Todos os integrantes escreveram parte da documentação ajudando na construção dos diagramas e na definição do layout.

2 Referencial Teórico

De forma geral, o problema apresentado pode ser considerado uma forma de um sistema SCADA, já que existirá um microcontrolador monitorando o ambiente, um servidor controlando logs, operações e alertas e um cliente para a visualização dos dados em tempo real baseado em uma comunicação direcionada à eventos. (Daneels & Salter, 1999)

Sendo assim, vários trabalhos envolvem sistemas similares ao atualmente proposto, mas não só isso como também o problema específico de monitoramento de uma hidroponia já foi observado em outros trabalhos. Como Palande et al. (2018) diz, alguns ambientes são de difícil cultivo e portanto um sistema de controle para cultivo sem necessidade de solo é muito bem visto para esses cenários.

A proposta de Sihombing et al. (2019) é distinta, pois não há o uso de um servidor, sendo que os cálculos e alertas são feitos diretamente no microcontrolador. A comunicação entre esse microcontrolador é feita através do software ANDES por uma conexão bluetooth. Não é uma solução para gerenciamento remoto devido à limitação de alcance da conexão bluetooth mas há a vantagem da economia de infraestrutura por não necessitar de um servidor dedicado.

O nosso trabalho se assemelha ao de Siregar et al. (2017), com um controle do sistema de forma remota e tendo a mediação de um servidor entre o cliente e o microcontrolador. Uma diferença notável é que o autor utiliza um web server para a comunicação, enquanto nós utilizamos um sistema baseado em publish-subscribe com diferentes protocolos.

Existem vários modelos de cultivo em hidroponia cada qual com seus benefícios e desafios para a automação (Lee & Lee, 2015). Foi observado que muitos utilizam do modelo de *Nutrient Film Technique* (NFT), onde as plantas ficam parcialmente submersas em uma solução de água e nutrientes, e este é o modelo utilizado para nossa implementação.

3 Modelagem

Os seguintes requisitos definem qual serão os princípios básicos para o desenvolvimento de um sistema capaz de monitorar

certas características da água em um ambiente com hidroponia. Em seguida é feito uma possível modelagem da solução apresentada.

3.1 Requisitos Funcionais

3.1.1 Frontend

- RF01: Na aplicação flutter será possível visualizar os dados do ambiente onde o microcontrolador está instalado, são eles: pH da água, luminosidade e temperatura.
- RF02: Na aplicação flutter será possível alterar o nível do pH da água onde o microcontrolador está instalado.
- RF03: Se caso o nível do pH da água for desviado de forma severa, será emitida uma notificação que aparecerá ao usuário do aplicativo. Essa notificação também será enviada mesmo que o aplicativo não esteja aberto (opcional).

3.1.2 Backend

- RF04: O servidor deverá persistir os dados enviados pelo microcomputador de forma constante em um banco de dados, são eles: pH da água, luminosidade e temperatura.
- RF05: O servidor deverá analisar e armazenar os últimos 30 registros do pH e temperatura em um buffer a fim de analisar os desvios de valores na água onde o microcomputador está instalado.
- RF06: O servidor deverá tratar, caso ocorra, um desvio severo no nível do pH da água, o servidor deverá enviar uma mensagem a aplicação flutter alertando sobre isso.
- RF07: O servidor deverá emitir um relatório contendo todos os dados coletados em um intervalo de tempo definido pela aplicação flutter, com a média do pH, temperatura e luminosidade coletados do microcontrolador do ambiente.
- RF08: O servidor deverá registrar toda e qualquer ação no sistema, mudanças de pH solicitadas pela aplicação e emissão de relatórios são alguns exemplos de ações que deverão ser armazenadas.
- RF09: O sistema deverá manter ativo uma variável que definirá o ‘target’ de pH da água, caso um usuário mude o ‘target’ do pH da água através do front-end, todos os outros usuários devem receber o mesmo ‘target’. Ou seja, assim que ocorrer a mudança de ‘target’ do pH, o servidor deverá atualizar o ‘target’ de todas as outras aplicações front-end.

3.1.3 Microcontrolador

- RF11: Enviar, a cada X segundos, os dados do PH atual da água e o PH em que está tentando manter. Além desses dados, enviar também a luminosidade e a temperatura da água.
- RF12: Receber solicitações de alteração do PH que é para ser mantido.
- RF13: Pingar soluções de ácido ou base para manter o pH da água equilibrado.

3.1.4 Infraestrutura

- RF13: Utilizar um sistema de eventos para o envio de mensagens entre dispositivos na rede.

Os diagramas desenvolvidos foram o de casos de uso, de sequência, de atividade e fluxograma, de classes, de componentes, de implantação e um modelo entidade relacionamento.

3.2 Casos de uso

Os casos de uso no sistema foram identificados e indicados na Figura 1. Os atores identificados foram uma abstração Usuário que possui as especializações Cliente e Administrador, o Servidor e o Microcontrolador. Há uma visão abstrada de como o sistema irá comportar-se em casos de solicitações de alteração de pH e temperatura da água e também quando houver um pedido de geração de relatório. Além de demonstrar as possíveis atividades no sistema.

3.3 Classes

O diagrama de classes foi realizado observando as operações necessárias para o servidor e o banco de dados. Este diagrama está na Figura 2. O diagrama é composto por dez classes, sendo duas delas classes associativas. A existência desses classes consiste no fato de que há relações ’n’ para ’n’ no diagrama, como é caso de ’Cliente’ com ’Relatório". Devido a esse tipo de cardinalidade entre relações, foram criadas as classes ’AlterarTarget’ e ’SolicitarRelatorio’ que atuam como classes associativas. Há outros detalhes do diagrama que são esclarecidos com descrições no próprio diagrama, como é o caso dos níveis de autorização de ’Administrador’.

Outro ponto relevante é o motivo pelo qual ambiente está sendo composto por outras três classes, isso ocorre para fins de modularização de classes. Para esse exemplo, o que está sendo monitorado no ambiente são luminosidade do ambiente, pH e temperatura da água, porém pode ser que no futuro seja decidido monitorar a umidade do local, para isso não seria necessário modificar a classe ’Ambiente’, apenas adicionar uma outra classe que a compõe.

Há atributos e métodos nas classes, mas é provável que boa parte seja alterado no desenvolvimento.

3.4 Modelo Entidade Relacionamento

Para facilitar o entendimento das relações entre as classes modeladas e também para a modelagem do banco de dados, foi realizado um diagrama MER apresentado na Figura 3. Uma das premissas do sistema é que toda e qualquer atividade no sistema seja salvo em um log de atividades. Para isso há relações com chaves primárias no MER, essas relações registra as ações do usuário, seja ele cliente ou administrador.

O Banco de dados também estará ativamente registrando os dados enviados no microcontrolador para o servidor, esses dados recebidos de forma constante será armazenado na tabela de nome ’Ambiente’.

3.5 Implantação

A implantação do sistema, ou seja, a interação entre cada componente foi colocada no diagrama da Figura 4. Na arquite-

tura final, o aplicativo Flutter estará funcionando em um dispositivo móvel (nos testes foram utilizados um celular Android compatível), que se comunicará via protocolo STOMP com o *broker* na cloud (AmazonMQ com ActiveMQ). Tanto o ESP8266 quanto o servidor Java recebem e enviam dados via MQTT para o broker. Para fins de simplificação, o banco de dados PostreSQL está fisicamente na mesma máquina que o servidor Java.

3.6 Componentes

Os componentes do sistema foram separados no modelo Model, View, Controller, Data, como indicado na Figura 5. Cada componente está relacionado a um grupo de casos de uso e classes do sistema. No próprio diagrama de componentes, há anotações detalhando quais classes fazem parte de quais componentes.

3.7 Fluxograma e Atividades

Para maior especificação das funções internas, foram feitos diagramas de atividade simples para os componentes de cliente (Figura 6) e de microcontrolador (Figura 7). Para o diagrama de atividades do componente cliente, são indicadas as "chamadas de funções remotas", que é o envio de mensagem para o broker esperando receber uma mensagem de retorno do servidor ou do microcontrolador (no caso de mudança dos dados objetivos).

3.8 Sequência

Os diagramas de sequência nas Figuras 8, 9, 10 e 11 contemplam a maioria dos casos de uso, sendo os principais o de envio de dados e alertas, alteração dos objetivos e login. Para os casos de uso de relatório e manutenção de equipamentos (isto é, um administrador registrar a ligação entre um cliente e um microcontrolador) a sequência é simples, apenas requisições e respostas entre o cliente e o servidor, que passam pelo broker intermediário.

4 Implementação e Resultados

Como citado anteriormente, o broker de mensagens utilizado para a comunicação é um ActiveMQ fornecido pela Amazon Web Services, denominado AmazonMQ. Tal comunicação foi pensada tanto na praticidade de uso quanto na escalabilidade, visando o escopo do projeto proposto.

Pensando em uma implantação em cenário real, cada cliente pode ficar responsável por um equipamento específico, para tanto, os tópicos do broker foram separados em dois conjuntos: aqueles que dependem de um equipamento (microcontrolador) e os que não dependem. Dos tópicos que não dependem de um equipamento foram definidos para request e reply de login e request e reply de relatório, contendo as variáveis necessárias para o tratamento com o servidor. O restante dos tópicos é contido dentro de um "supertópico" definido pelo id do equipamento, sendo que foram definidos tópicos para a stream das variáveis, a definição de dados objetivos e um tópico de alerta.

Os dados são passados através de um formato JSON, com formatação dos campos pré-acordada entre os componentes implementados. Ressalta-se que a estrutura e conteúdo dos tópicos é subjetiva, e depende do cenário específico da implementação do projeto.

Considerando as características anteriormente descritas, os detalhes de cada componente do sistema são exibidos a seguir.

4.1 Microcontrolador

Para a prototipagem da primeira versão do sistema, um módulo microcontrolador ESP8266 (desenvolvido pela empresa chinesa Espressif) foi utilizado. A vantagem desse item é o baixo custo (sendo achado facilmente por menos de R$30,00 em 2023), visto que já possui WiFi integrado, se encaixando perfeitamente no projeto. Como não possuímos recursos suficientes para comprar todos os módulos necessários, simulamos alguns deles. Para fazer o projeto real, seria necessário um módulo de detecção e monitoramento de pH da água, como por exemplo o modelo PH4502C com eletrodo, que é compatível com o ESP8266. Seria interessante levar em consideração a disposição de 2 sensores desse tipo para tentar detectar erros de medição e/ou calibragem, já que a leitura errada pode acabar ocasionando na morte das plantas cultivadas.

Figura 12: módulo PH4502C + eletrodo para aferição de pH

No cenário ideal haveria duas bombas peristálticas, que seriam usadas para jogar as soluções de ácido ou base na água do sistema, a fim de controlar o pH.

Figura 13: bomba dosadora peristáltica

Para a simulação, coletamos dados de luminosidade utilizando um módulo sensor GY-30. Os valores de pH estão sendo

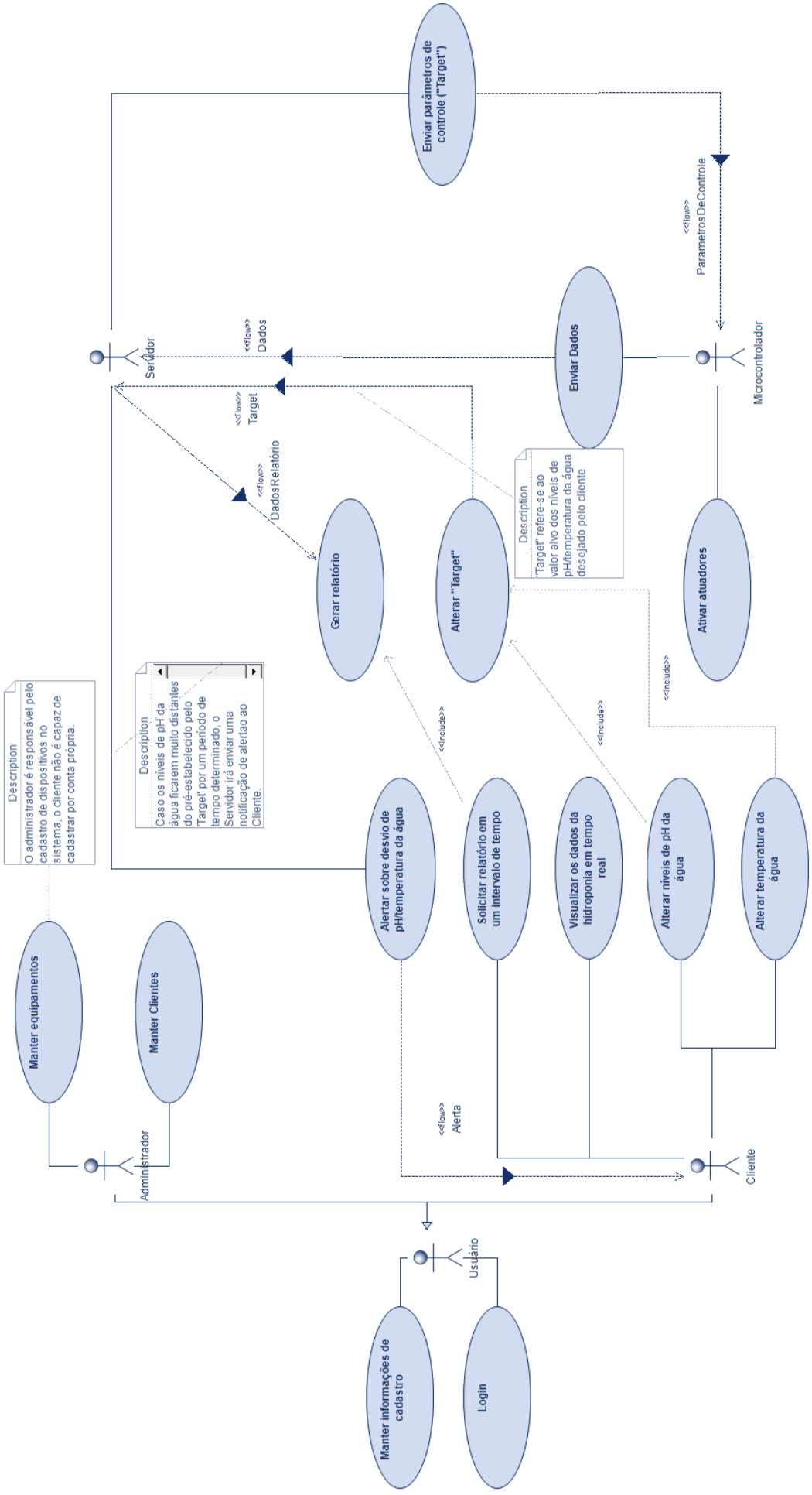

Figura 1: Casos de uso do sistema.

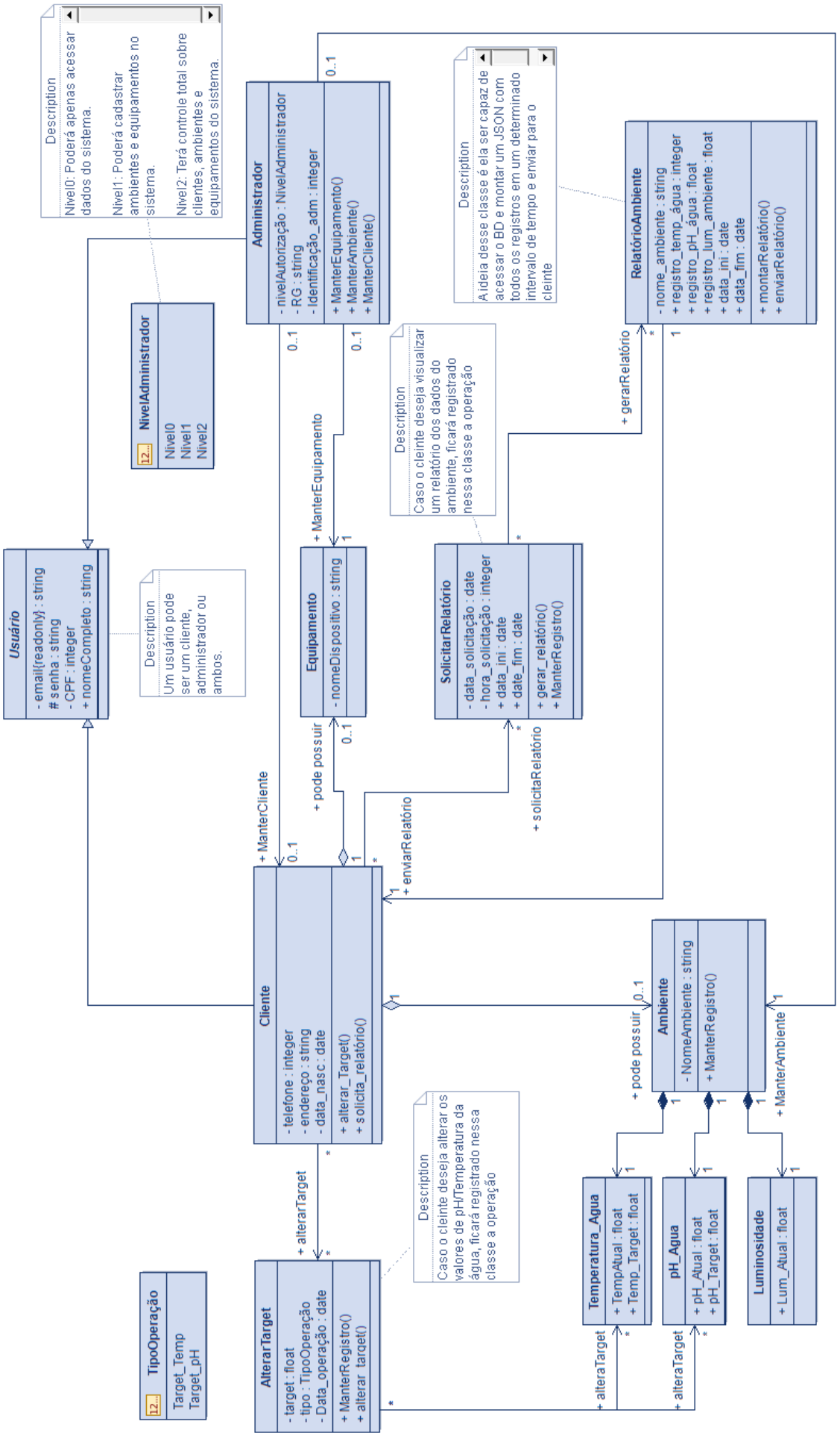

Figura 2: Diagrama de classes do sistema.

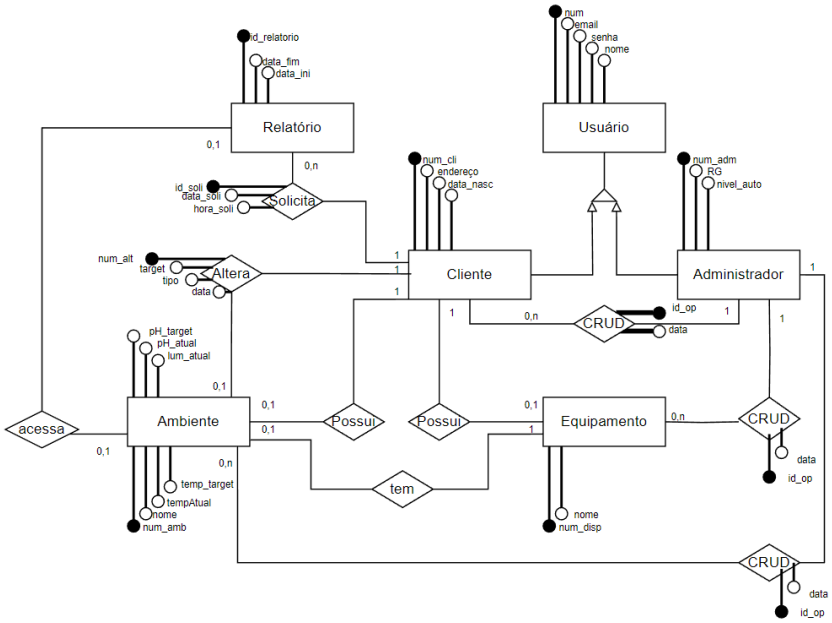

Figura 3: Modelo Entidade Relacionamento do sistema.

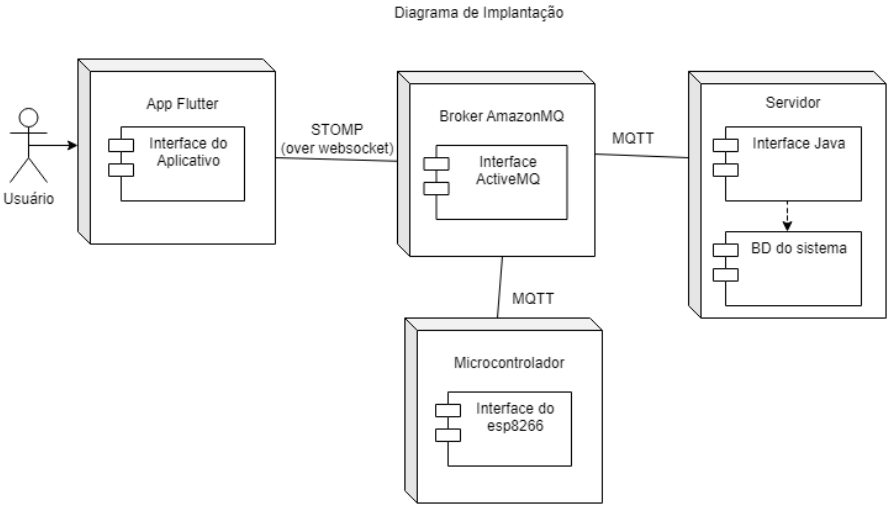

Figura 4: Diagrama de implantação do sistema.

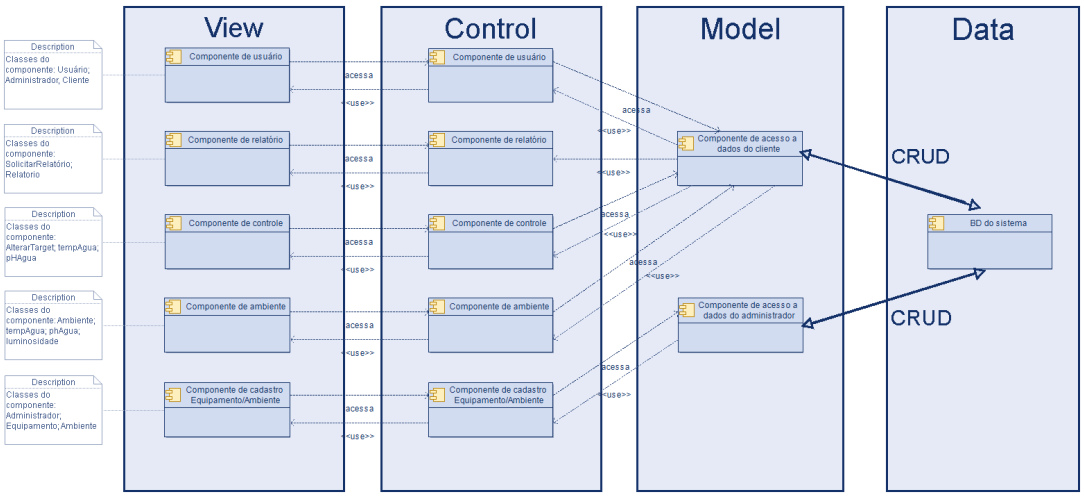

Figura 5: Diagrama de componentes do sistema

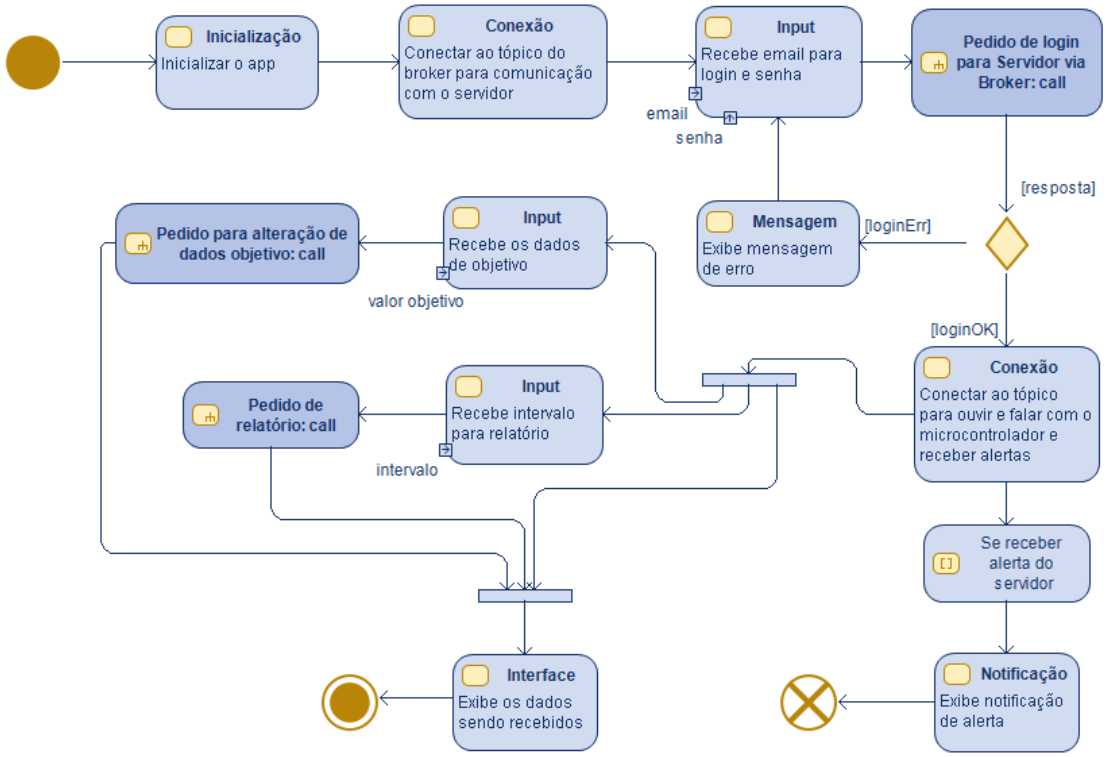

Figura 6: Diagrama de atividades para o componente cliente.

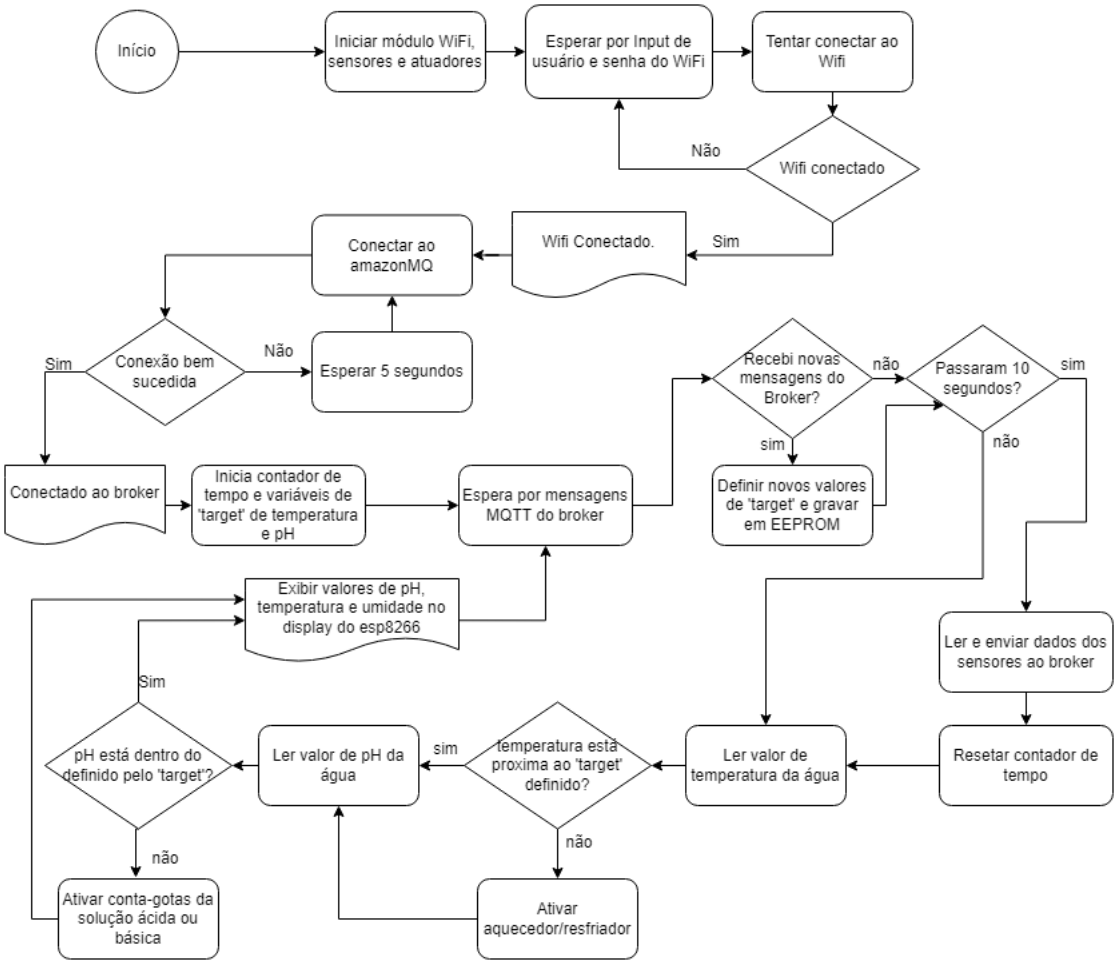

Figura 7: Fluxograma de funcionamento interno do ESP8266.

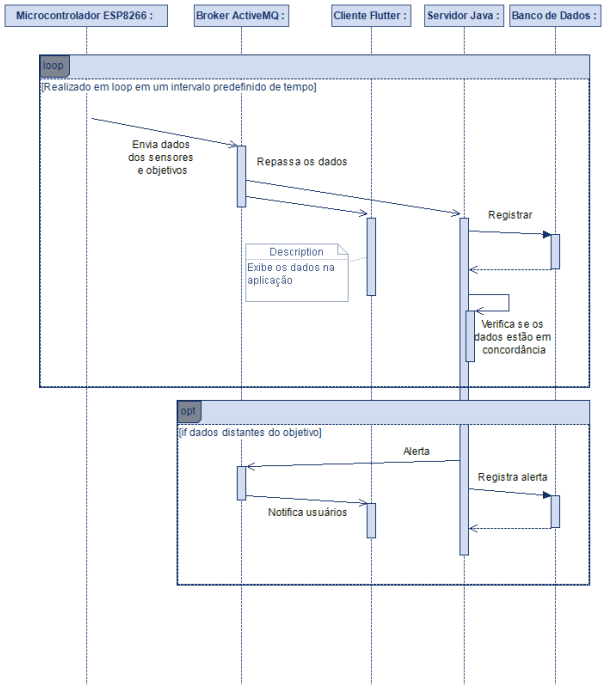

Figura 8: Diagrama de sequência de envio de dados pelo microcontrolador.

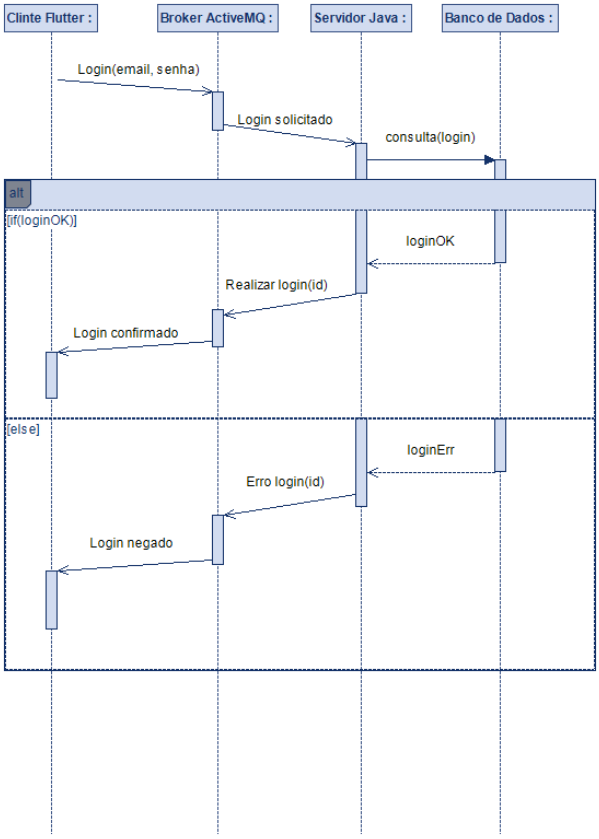

Figura 10: Diagrama de sequência de login de um cliente.

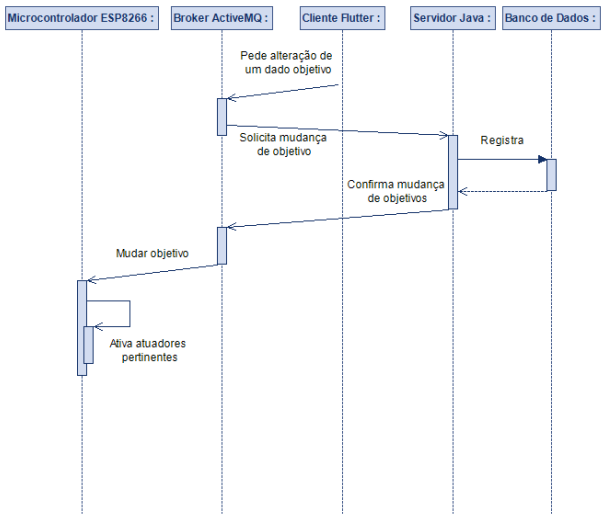

Figura 9: Diagrama de sequência de alteração de um dado objetivo pelo cliente.

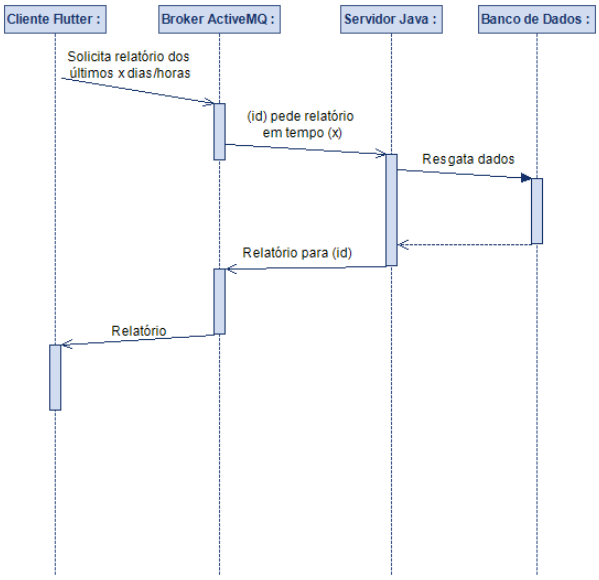

Figura 11: Diagrama de sequência de pedido de um relatório pelo cliente.

simulados por uma entrada analógica (potenciômetro), que pode ser alterado (parecido com um botão de volume). O led vermelho e verde simulam os atuadores (bomba dosadora). Um display LCD foi utilizado a fim de observar as saídas do programa.

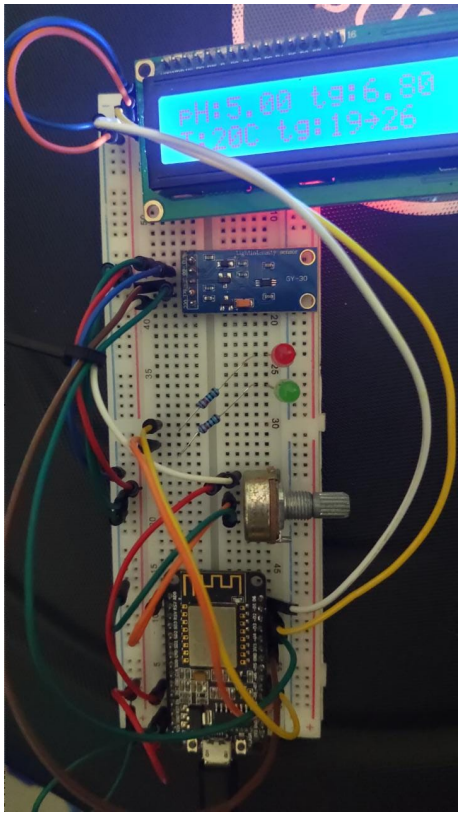

Figura 14: protótipo de simulação

O código do microcontrolador ESP8266 pode ser encontrado no repositório remoto do GitHub através do link https://github.com/Amazon-MQ-Seminario-UNIFEI/esp8266-hidroponic-activemq.

4.2 Cliente Flutter

O *framework* Flutter foi utilizado para o desenvolvimento de um aplicativo final Android que atua como cliente no sistema (uma construção Windows também foi utilizada durante o desenvolvimento), recebendo os dados e solicitando mudanças e relatório, como indicado nos diagramas. A comunicação é feita com o protocolo Stomp sobre web socket, e as mensagens são convertidas entre String JSON e Map.

Na inicialização do aplicativo e feita a conexão com o servidor, que é mantida em uma variável compartilhada entre as telas. Após a conexão o usuário é enviado para a tela de login (Figura 15), onde pode enviar as credenciais para o tópico que o servidor está preparado para ouvir e tratar. Caso ocorra um erro no login um alerta é emitido para o cliente, caso contrário uma variável compartilhada com os dados do usuário é armazenada.

Após a realização do login, o usuário é redirecionado para a tela principal, onde são vistos os dados enviados pelo microcontrolador (Figura 16). O usuário só pode acessar essa tela se estiver conectado e logado, e os tópicos em que ele é inscrito

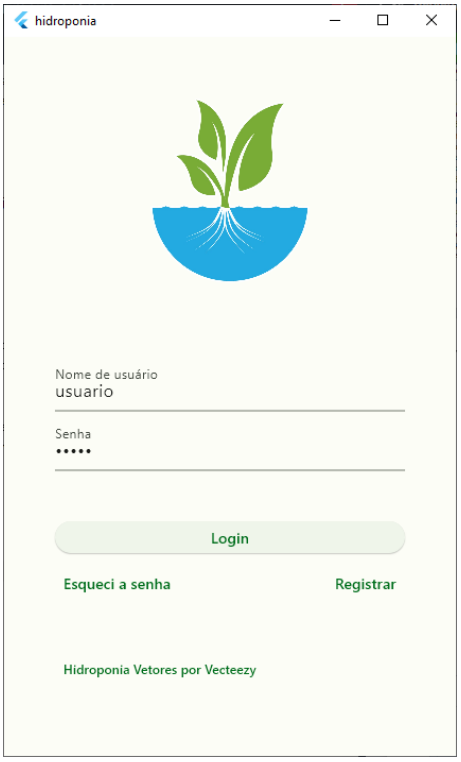

Figura 15: Página de login do cliente. Após o usuário enviar os dados há uma espera pela resposta.

são específicos do equipamento que ele é autorizado, se ocorrer um alerta ele será exibido nessa tela ou qualquer navegação partindo desta tela (Figura 17). Dessa página, o usuário pode navegar para uma página para definir novos objetivos das variáveis do ambiente ou uma página para solicitar relatório.

Na tela de modificação dos objetivos dos valores (Figura 18), o usuário pode enviar a mudança de pH objetivo e/ou temperatura mínima e máxima. Não há espera de resposta, pois foi feita a simplificação de assumir que o microcontrolador e o servidor receberam a mensagem.

Já para a solicitação de relatório, o usuário pode selecionar sobre quais variáveis ele deseja receber o relatório, e também qual a escala de busca, isto é, entre quais dias ou entre qual horário de um determinado dia (Figura 19). Após o envio da solicitação o cliente espera uma resposta no tópico correspondente para ser redirecionado para uma tela de visualização com os dados recebidos do servidor (Figura 20).

O código para a aplicação flutter pode ser encontrado no repositório remoto do GitHub através do link https://github.com/MarceloRobert/aws_mq_app.

4.3 Servidor

O servidor foi desenvolvido na linguagem Java, ele é responsável por tratar as requisições do cliente e também realiza a conexão com o banco de dados relacional PostgreSQL.

Assim que o servidor é iniciado, ele constantemente ouve os tópicos de requisição de login, requisição de alteração dos valores do ambiente e de relatório, e todos os tópicos dos microcontroladores ativos. A comunicação com o broker é feita através do protocolo MQTT e com o banco de dados através da biblioteca padrão disponibilizada pelo PostgreSQL para a comunicação com a linguagem JAVA.

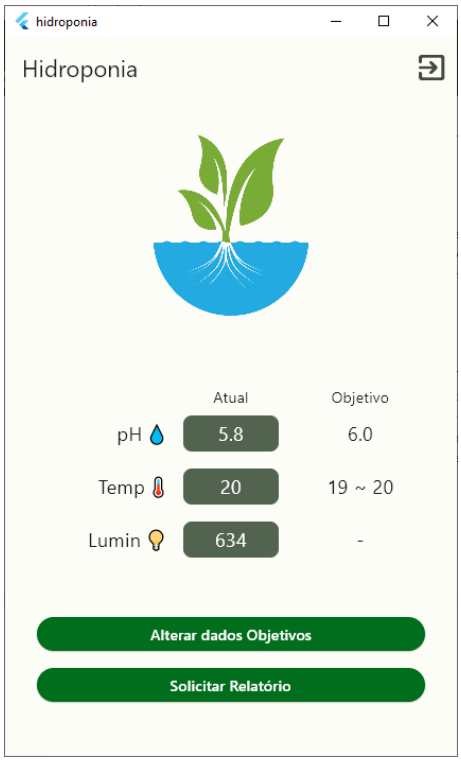

Figura 16: Página principal do cliente. Os dados são atualizados a cada intervalo predefinido de tempo.

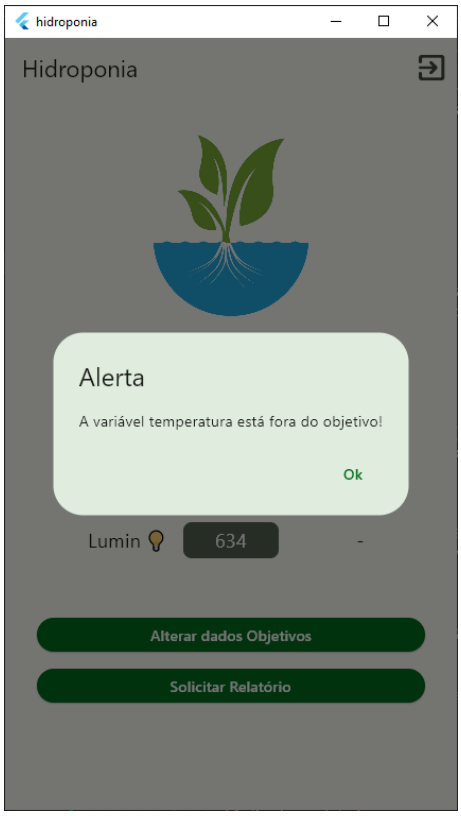

Figura 17: Exemplo de alerta ao cliente.

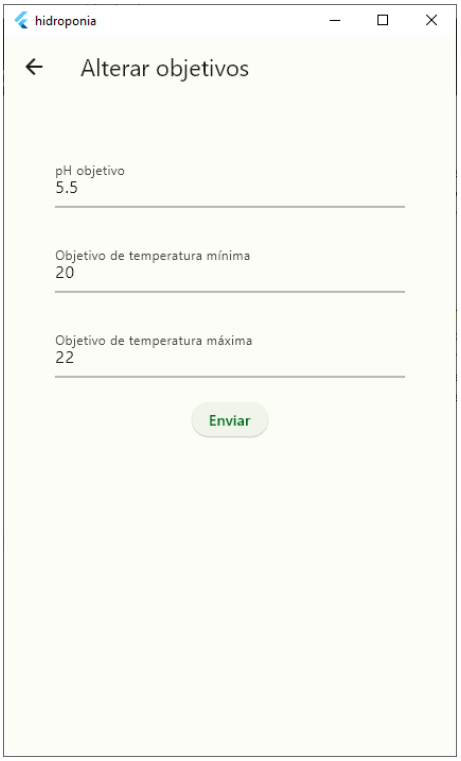

Figura 18: Página para mudança dos dados objetivos. Os dados são verificados antes de enviar para o servidor.

Figura 19: Página de solicitação do relatório. Os dados são formatados em String.

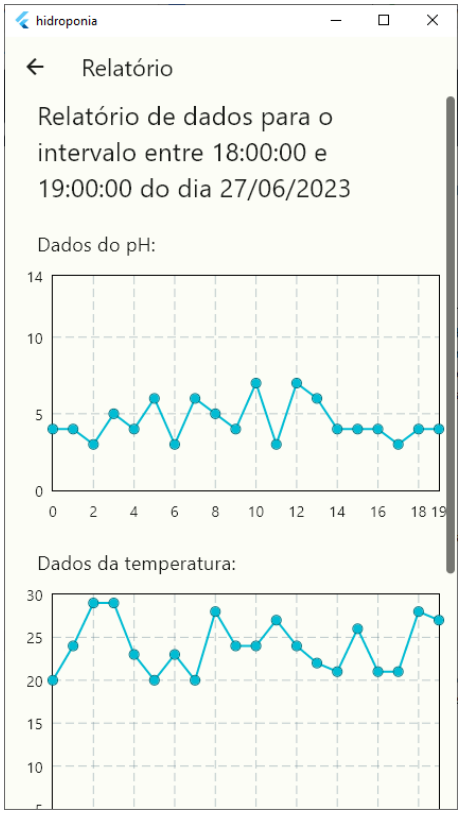

Figura 20: Página de exibição do relatório. A tela pode ser rolada para baixo para visualizar o último gráfico exibido.

A principal objetivo do servidor é realizar o armazenamento de toda e qualquer requisição do cliente ao backend; Solicitações de relatório, Solicitação de alteração de pH/temperatura da água, Stream de dados do ambiente onde o microcontrolador está instalado, tudo isso o Servidor armazena.Devido a alta rastreabilidade de dados, é possível montar diversos relatórios a partir do banco de dados atrelado ao servidor. Um dos exemplos de dados registrados pode ser verificado na Figura 21.

O servidor também verifica se o usuário que está tentando fazer login existe no BD, caso não será emitido uma mensagem ao Frontend avisando que não foi possível acessar aquele cadastro. Outra função do servidor é verificar se a temperatura e o pH da água estão dentro dos valores permitidos, caso ocorra algum desvio, o servidor irá emitir um aviso ao usuário do sistema sobre o problema identificado.

5 Conclusões

Com a realização do projeto foi possível reconhecer os desafios da automatização de um sistema de hidroponia, visualizar vários trabalhos já realizadas e implementar códigos próprios da nossa idealização, sendo realizada a comunicação entre os três componentes de forma distribuída e remota, como queríamos demostrar.

O resultado final foi um protótipo da solução proposta, podendo ser expandido e completo com as restrições apresentadas nos diagramas e implementações mais eficientes em qualquer um dos componentes. As principais melhorias podem ser feitas na comunicação entre cliente e servidor (que poderia ser feita diretamente), no armazenamento dos dados do microcontrolador (outro tipo de banco de dados poderia ser utilizado) e também na implantação do broker intermediário (implementação própria ao contrário de um serviço de terceiros).

No microcontrolador ESP8266, os principais desafios foram conseguir conciliar sensores. Além disso, conseguir realizar a conexão com o AmazonMQ exigiu um pouco de esforço. Tratar as mensagens que chegam do broker pode ser um pouco desgastante. Como projetos futuros, há a possibilidade de tirar todas as variáveis que estão *hard coded*. Por exemplo, inserir informações de WiFi pode ser feita utilizando uma página web gerada pelo próprio microcontrolador.

No cliente Flutter, os principais desafios foram em relação ao ouvir as Streams de dados e como eles seriam repassados para as telas. A solução foi utilizar construtores sobre toda a página que ouvem qualquer mensagem do tópico e as tratam antes de exibir a tela. Além disso, dados como a sessão do usuário foram colocados de forma compartilhada entre as telas, mas não há persistência de sessão do usuário, o que poderia ser implementado em trabalhos futuros.

No servidor, os principais desafios foram em relação ao entendimento das várias bibliotecas que Java disponibiliza através do gerenciador de bibliotecas Mavem. Um dos problemas enfrentado foi justamente o conflito entre dependências no Java. Outro problema foi como a comunicação entre o Servidor e o Banco de Dados era realizada, a principio foi utilizado o JPA (Java Persistence API), que transforma um objeto Java em uma tabela no Banco de Dados, acontece que como essa aplicação utiliza de mensagens para a troca de dados entre Frontend e Backend, o JPA acabou complicando o entediamento do problemas a ser resolvido, devido isso, houve uma alteração de como é persistido os dados no Banco de Dados atráves do Servidor. No caso, foi optado pela criação das tabelas no BD e a manipulação delas no Servidor Java.

Referências

Daneels, A. & Salter, W. (1999). What is scada? *7th Biennial International Conference on Accelerator and Large Experimental Physics Control Systems*. Disponível em: https://cds.cern.ch/record/532624; Acessado em 27/06/2023.

Lee, S. & Lee, J. (2015). Beneficial bacteria and fungi in hydroponic systems: Types and characteristics of hydroponic food production methods. *Scientia Horticulturae*, 195, 206–215. Disponível em https://www.sciencedirect.com/science/article/pii/S0304423815301758; Acessado em 27/06/2023.

Palande, V., Zaheer, A., & George, K. (2018). Fully automated hydroponic system for indoor plant growth. *Procedia Computer Science*, 129, 482–488. 2017 INTERNATIONAL CONFERENCE ON IDENTIFICATION, INFORMATION AND KNOWLEDGE IN THE INTERNET OF THINGS.

Sihombing, P., Zarlis, M., & Herriyance (2019). Automatic nutrition detection system (andes) for hydroponic monitoring by using micro controller and smartphone android. Em *2019 Fourth International Conference on Informatics and Computing (ICIC)* (pp. 1–6). Disponível em https://ieeexplore.ieee.org/abstract/document/8985851; Acessado em 27/06/2023.

	tempatual integer	tempmin integer	tempmax integer	phatual numeric (14,2)	phtarget numeric (14,2)	lumatual integer	amb_id integer	dataregistro date	horaregistro time without time zone
1	20	19	26	4.00	6.80	172	100	2023-06-27	18:16:19
2	24	19	26	4.00	6.80	444	100	2023-06-27	18:16:49
3	29	19	26	3.00	6.80	308	100	2023-06-27	18:17:19
4	29	19	26	5.00	6.80	85	100	2023-06-27	18:18:36
5	23	19	26	4.00	6.80	133	100	2023-06-27	18:19:06
6	20	19	26	6.00	6.80	452	100	2023-06-27	18:19:36
7	23	19	26	3.00	6.80	312	100	2023-06-27	18:20:06
8	20	19	26	6.00	6.80	505	100	2023-06-27	18:20:36
9	28	19	26	5.00	6.80	58	100	2023-06-27	18:21:06
10	24	19	26	4.00	6.80	468	100	2023-06-27	18:21:36
11	24	19	26	7.00	6.80	344	100	2023-06-27	18:25:25
12	27	19	26	3.00	6.80	502	100	2023-06-27	18:25:55
13	24	19	26	7.00	6.80	562	100	2023-06-27	18:26:25
14	22	19	26	6.00	6.80	463	100	2023-06-27	18:26:55
15	21	19	26	4.00	6.80	523	100	2023-06-27	18:27:25

Figura 21: Print de todos os registros de dados enviados pelo microcontrolador instalado em um ambiente.

Siregar, B., Efendi, S., Pranoto, H., Ginting, R., Andayani, U., & Fahmi, F. (2017). Remote monitoring system for hydroponic planting media. Em *2017 International Conference on ICT For Smart Society (ICISS)* (pp. 1–6). Disponível em: https://ieeexplore.ieee.org/abstract/document/8288884; Acessado em 27/06/2023.