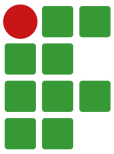

INSTITUTO FEDERAL
São Paulo
Câmpus Catanduva

Luiz Antônio Fernandes Gomes, Túlio Guirado Geraldo

Irrigação do projeto AgroflorIF

Catanduva, SP
Maio, 2021

Luiz Antônio Fernandes Gomes, Túlio Guirado Geraldo

Irrigação do projeto AgroflorIF

Monografia submetida ao Colegiado do Curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva, como parte dos requisitos para conclusão do Curso superior de Tecnologia em Análise e Desenvolvimento de Sistemas.

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, campus Catanduva

Orientador: Prof. MSc. Daniel Corrêa Lobato

Catanduva, SP

Maio, 2021

Elaborada pela biblioteca do IFSP Câmpus Catanduva
com dados fornecidos pelo autor.

Gomes, Luiz Antonio Fernandes

Geraldo, Tulio Guirado

G633i

Irrigação do projeto AgroflorIF / Luiz Antonio Fernandes Gomes, Tulio Guirado Geraldo. – Catanduva, SP: IFSP, 2021.
23 f.

Orientador: Prof. MSc. Daniel Corrêa Lobato

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - - Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Catanduva, 2021.

1. Irrigação automatizada. 2. Sensores de solo inteligentes. 3. ESP. 4. Fotovoltaico. I. Lobato, Daniel Corrêa. (Orient.). II. Título.

CDD (23. ed.): 631.7

ATA N.º 10/2021 - SIF-CTD/DAE-CTD/DRG/CTD/IFSP

Ata de Defesa de Trabalho de Conclusão de Curso - Graduação

Na presente data realizou-se a sessão pública de defesa do Trabalho de Conclusão de Curso intitulad**IRRIGAÇÃO DO PROJETO AGROFLORIF** apresentado pelos alunos **Luiz Antonio Fernandes Gomes** (CT3002454) e **Túlio Guirado Geraldo** (CT3004431) do Curso **SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS**, (*campus* Catanduva). Os trabalhos foram iniciados às 17h50 pelo Professor presidente da banca examinadora, constituída pelos seguintes membros:

Membros	IES	Presença (Sim/Não)	Aprovação/Conceito (Quando Exigido)
Daniel Corrêa Lobato (Presidente/Orientador)	IFSP Catanduva	Sim	9.00
Eros Schettini Roman (Examinador 1)	IFSP Catanduva	Sim	9.00
Lúcio Rodrigo de Carvalho (Examinador 2)	IFSP Catanduva	Sim	9.00

Observações:

A banca examinadora, tendo terminado a apresentação do conteúdo da monografia, passou à arguição dos candidatos. Em seguida, os examinadores reuniram-se para avaliação e deram o parecer final sobre o trabalho apresentado pelos alunos, tendo sido atribuído o seguinte resultado:

[X] Aprovado(a) [ ] Reprovado(a) Nota (quando exigido): 9,00 (nove pontos)

Proclamados os resultados pelo presidente da banca examinadora, foram encerrados os trabalhos e, para constar, eu lavrei a presente ata que assino juntamente com os demais membros da banca examinadora.

Câmpus Catanduva, 3 de dezembro de 2021

Avaliador externo: [ ]Sim [X]Não

Assinatura:

Documento assinado eletronicamente por:

- **Daniel Correa Lobato**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 03/12/2021 19:10:15.
- **Lucio Rodrigo de Carvalho**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 03/12/2021 19:11:13.
- **Eros Schettini Roman**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 03/12/2021 19:12:28.

Este documento foi emitido pelo SUAP em 03/12/2021. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 266704
Código de Autenticação: 2bb9b37f26

Resumo

Buscamos com esse projeto agregar e aplicar os conhecimentos adquiridos no curso de análise e desenvolvimento de sistemas do Instituto Federal de São Paulo campus Catanduva. A partir das dificuldades apresentadas no controle de irrigação e sensoriamento do projeto AgroflorIF do IFSP Catanduva, desenvolvemos um sistema automatizado com sensores baseados no microcontrolador ESP32 alimentados por painéis fotovoltaicos, contando com sistema web e um banco de dados integrado onde se torna possível ter informações sobre o solo e o controle da irrigação de qualquer lugar. O sistema criado apresentará um fácil escalonamento onde só será preciso ligar os novos sensores e configurá-lo no sistema.

Palavras-chave: Irrigação automatizada, Sensores de solo inteligentes, ESP, fotovoltaico.

Abstract

With this project, we seek to add and apply the knowledge acquired in the course on analysis and systems development at the Federal Institute of São Paulo, Catanduva campus. From the difficulties presented in the irrigation control and sensing of the Agroflorif project of the institute, we sought to develop an automated system with sensors based on the ESP32 microcontroller powered by photovoltaic panels, with a web system and an integrated database where it becomes It is possible to have information about soil and irrigation control from anywhere. The created system will present an easy scaling where it will only be necessary to connect the new sensors and configure it in the system.

Keywords: Automated Irrigation, Intelligent Ground Sensors, ESP, Photovoltaic.

Lista de figuras

Sumário

1 Introdução

Nos dias de hoje a busca por alimentação mais saudável está se tornando muito maior, por isso muitas pessoas optam por construir ambientes que permitam se obter alimentos saudáveis em casa, justamente pelo fato de ser mais barato e consideravelmente mais saudável, visto que não há a utilização de agrotóxicos ou algo do tipo. Porém, muitos não dispõem de tempo para irrigar suas plantações. Assim, nossa proposta foi desenvolver um sistema que seja capaz de automatizar a irrigação dessas hortas, permitindo o monitoramente e o controle da irrigação de forma online, utilizando um navegador Web executando em qualquer dispositivo que possua acesso à Internet em qualquer lugar do mundo, utilizando diversos conceitos de desenvolvimento para WEB e Internet das Coisas (do inglês *Internet of Things* - IoT).

Considerando o que foi exposto até aqui, nesta monografia relatamos o desenvolvimento da solução proposta. A principal hipótese de pesquisa é que possível automatizar o processo de irrigação da horta, garantindo a umidade do solo necessária para o desenvolvimento das plantas, e os resultados obtidos permitem validar essa hipótese.

Está organizada como se segue: o capítulo 2 apresenta a fundamentação teórica necessária para o desenvolvimento do trabalho; o capítulo 3 discute o método adotado para o desenvolvimento; no capítulo 4 é apresentado o desenvolvimento do projeto, e; o capítulo 5 apresenta, por fim, as conclusões deste trabalho, bem como os trabalhos futuros.

2 Fundamentação teórica

Neste capítulo serão discutidas as principais tecnologias e ferramentas utilizadas para o desenvolvimento da solução proposta. De forma geral, foi proposto...

2.1 API REST

As API's REST, é uma Application Programming Interface (interface de programação de aplicações) que segue os padrões de arquitetura REST, que permite a integração com serviços web. REST é uma sigla em inglês para transfêrencia representacional de estado, que foi criada pelo cientista da computação Roy Fielding (REDHAT, 2018). Essa arquitetura é geralmente utilizada para consulta de dados remotas de forma segura, como por exemplo, retornar o endereço a partir do CEP informado pelo usuário ou autenticar informações sensíveis por exemplo.

2.2 Git e GitHub

O Git é uma ferramenta de código aberto para controle de versão, sendo utilizado pela grande maioria dos programadores. Ela possibilita a criação de um histórico das alterações realizadas no código do projeto, garantindo também que seja possível restaurar essas alterações à qualquer momento (FERNANDES, 2018).

O GitHub é um serviço online que permite a hospedagem de repositórios Git, garantindo que seja possível o armazenamento do estado da aplicação, fora do ambiente de desenvolvimento (FERNANDES, 2018).

Esta ferramenta foi utilizada para o compartilhamento dos códigos entre os docentes, estando sujeito ao controle de versão e facilitando o desenvolvimento de forma mutua e ágil.

2.3 JavaScript e TypeScript

O JavaScript é uma linguagem de programação de computadores que é vastamente utilizada em aplicações para a WEB, permitindo que *websites* sejam mais dinâmicos e interativos. O JavaScript funciona no navegador web, hoje em dia a maioria dos sites a utilizam (LATEEF, 2018).

TypeScript é um superconjunto para o JavaScript que permite melhorar a forma que o código trabalha e deixá-lo mais conciso e legível. O TypeScript permite a adição de tipos de dados no JavaScript, tornando as aplicações que a utilizam muito mais estáveis e seguras, além de melhorar o suporte à programação orientada a objetos, garantindo uma melhor modularização da aplicação. (NOLETO, 2020).

2.4 Supabase

O Supabase é uma plataforma de servidor como serviço (do inglês *backend as a service*) de código aberto e escalável. Essa plataforma fornece serviços como o de API e banco de dados (PostgreSQL), autenticação com provedores de serviço (Google, Facebook, Twitter, Discord, entre muitos outros) e armazenamento de arquivos estáticos (COPPLESTONE, 2021). Essa plataforma permite criar e manter uma aplicação de forma mais rápida, segura, escalável e confiável em poucos segundos, além de abstrair e não necessitar da utilização da linguagem SQL para consultar o banco de dados.

2.4.1 Sistemas Gerenciadores de Bancos de Dados

Como explica Chan (2019), em uma postagem no blog *Thorn Technologies*, há dois principais tipos de banco de dados hoje em dia: os relacionais e os não relacionais.

A principal diferença entre estes dois tipos é a forma na qual os dados são armazenados e gerenciados, visto que nos relacionais há uma relação estruturada entre as linhas e colunas na tabela, o que não acontece nos bancos de dados não-relacionais, já que essa relação não existe, tornando-os mais rápidos e não sendo tão impactados quando há uma grande quantidade de dados. Outro detalhe importante é que, nos bancos de dados relacionais a definição da estrutura dos dados é extremamente importante, já nos não relacionais, essa pré-definição não é necessária. A nossa aplicação utilizou o PostgreSQL, que é um sistema gerenciador de banco de dados (SGBD) que atua como gerenciador de banco de dados relacional, implementando a linguagem de consultas estruturadas (do inglês *SQL*). Esse SGBD é um dos principais do mercado, principalmente pela disponibilidade de *drivers* para uso em diversas linguagens de programação, robustez, otimização, segurança e facilidade de instalação e uso. O PostgreSQL se destaca por suas adequações nos padrões de conformidade estabelecidos, o que garante um sistema mais robusto e otimizado para as mais diversas aplicações, inclusive quando há uma quantidade massiva de dados (SOUZA, 2020).

2.5 Vue.JS

O Vue.JS é um *framework* progressivo para a construção de interfaces de usuário (FRITZ, 2016). É escrito em JavaScript, que permite o desenvolvimento de aplicações *web* responsivas e reativas. Foi criado por Evan You em 2014 focado em acessibilidade e versatilidade. É muito utilizado para o desenvolvimento de aplicações de página única, além de diversos outros tipos de interfaces, focando na interação e experiência do usuário. Hoje em dia, o Vue é um dos *frameworks* JavaScript mais populares do mundo (GUEDES, 2020).

2.6 Nuxt.JS

É um *framework* web escrito em JavaScript baseado no *framework* Vue.JS que otimiza o desenvolvimento de aplicações de página única e também otimiza o *ranking* de SEO nos mecanismos de busca. O Nuxt.JS garante que a consistência de funcionamento da

aplicação em todas as circunstâncias que a aplicação será acessada e utilizada, como por exemplo impedir o acesso de um navegador incompatível ou que não possuem o JavaScript habilitado, como explica Petry (2020).

2.7 MicroPython

É uma implementação da linguagem de programação interpretada Python (versão 3) para operar em microcontroladores (ANDRADE, 2016). Por ser um subconjunto da linguagem Python, mantém várias de suas características, entre elas o fato de ser uma linguagem de alto nível, flexível e de fácil aprendizagem. Tem sido adotada em projetos de casa inteligente, internet das coisas e aprendizado de máquina, por ser uma linguagem de fácil aprendizagem, ela possibilitou que o mundo da programação e desenvolvimento ficasse mais acessível e simples (ROVEDA, 2020).

3 Método

O desenvolvimento da API REST e do painel de controle WEB deverá seguir as boas práticas do desenvolvimento WEB, que são a utilização do HTML versão 5 (mais recente), indentação e simplicidade, para a API, a utilização de tecnologias modernas, como por exemplo JavaScript e TypeScript.

Os dados serão armazenados utilizando um banco de dados relacional chamado PostgreSQL. Isso pelo fato de ser um banco de dados mais robusto, estável, seguro e de código aberto (do inglês *Open Source*).

Serão construídos dispositivos eletrônicos de coleta de dados utilizando plataforma de prototipação nos quais estarão conectados os sensores. A comunicação entre esses dispositivos e a central de irrigação será feita utilizando um protocolo de comunicação sem fio.

O software para os dispositivos de coleta será desenvolvido utilizando alguma linguagem compatível com a plataforma que for selecionada.

O código gerado estará sujeito a controle de versão utilizando ferramentas e ambientes comuns, como Git e GitHub.

O painel de visualização e configuração do sistema de irrigação será construído utilizando um framework para desenvolvimento Web e a solução será hospedada na nuvem.

4 Desenvolvimento

4.1 O hardware escolhido

A coleta dos dados de umidade do solo precisa ser feita de forma automatizada. Existem atualmente duas alternativas para sensores de umidade do solo: capacitivos e resistivos.

Neste projeto, adotamos sensores capacitivos por evitar a corrosão e assim aumentando significativamente a vida útil do sensor em relação ao sensores resistivos que acabam sofrendo eletrólise por ser necessário passar uma corrente pelos eletrodos ligados ao solo úmido.

Ao invés de construir os sensores, o que estaria fora do foco principal do trabalho, optamos para aquisição de sensores comerciais disponíveis no mercado global, e de baixo custo. A solução adotada foi o LILYGO@TTGO T-Higrow com ESP32 que permite a leitura da humidade do solo através do sensor capacitivo e do ar, bem como a temperatura ambiente, através do DHT11 embutido.

Os sensores incluem um microprocessador com suporte a Micropython, que nos permite dotá-lo de inteligência, como a leitura dos sensores analógicos e o pré-processamento dos dados lidos antes de armazenar nas bases de dados se necessário.

A comunicação desses sensores pode ser feita por cabo USB ou por rede WiFi. Adotamos a comunicação sem fio por ser preciso a montagem de infraestrutura com cabos que aumentariam o custo e manutenibilidade do sistema.

A alimentação dos sensores pode ser feita por baterias de lítio, com uma capacidade de 200 mAh, o que premite a operação por cerca de duas horas. Durante o desenvolvimento do software para execução no sensor, podemos adotar técnicas de *deep sleep* para reduzir o consumo da unidade, aumentando a vida útil da bateria para quarenta mil horas.

As baterias são carregadas por meio de células fotovoltaicas de 2w e 5v conectadas diretamente a porta USB do sensor. A tensão gerada pelos painéis passa por TP4054, já acoplado ao sensor, sendo um circuito integrado para gerenciamento de carga de baterias de lítio do tipo linear de corrente e tensão constantes para baterias de célula única.

Com este circuito as oscilações de energia provenientes de sombreamentos na celula fotovoltaica são estabilizadas pois o TP4054 mantem a tensão de carga fixada em 4,2V durante o carregamento e termina automaticamente o ciclo de carga quando a corrente de carga cai para 1/10 do valor programado depois que a tensão flutuante final é alcançada.

Não é necessário o uso de diodo para o bloqueio de corrente reversa pois essa proteção ocorre devido à arquitetura interna do PMOSFET presente e a segurança do controle de feedback térmico regula a corrente de carga para limitar a temperatura da bateria durante a etapa de carregamento de alta potência impedindo que a bateria sofra risco de explosão.

4.2 O software desenvolvido

Foram desenvolvidos softwares para execução no sensor, na retaguarda e nos clientes. Nos sensores foi utilizado a linguagem micro Python, enquanto que na retaguarda usamos a linguagem TypeScript, e os clientes usaram Nuxt.JS.

4.2.1 Software dos sensores

O sensor LILYGO@TTGO T-Higrow adotado permite a execução de código em micropython. Desenvolveu-se um programa que verifica se o sensor já está configurado, caso não, é criada uma rede para configuração onde se inclui o SSID e senha da rede a ser utilizada pelo sensor e gera-se um arquivo contendo as configurações e reinicializa o sensor. Caso o sensor já esteja configurado, o programa se conecta à rede WiFi configurada e verifica se o endereço físico (*MAC address*) já está registrado na API. Não estando registrado, é gerado um código único para a vinculação e ocorre uma nova reinicialização. Após configurado e vinculado, o programa efetua a leitura periódica do sensor e faz o envio dos valores lidos para o banco de dados utilizando uma API REST.

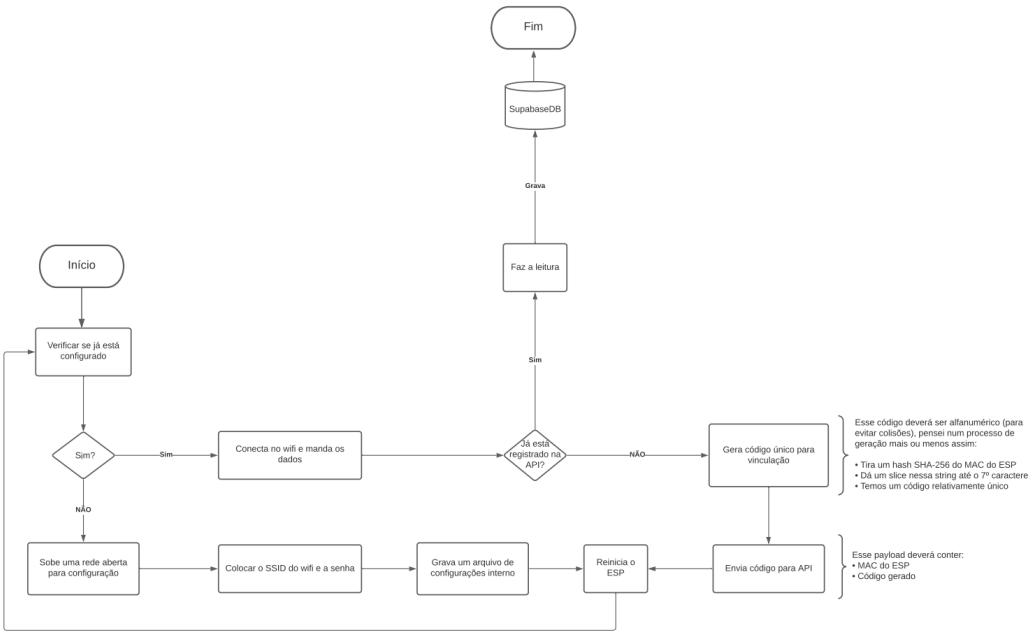

Figura 1 – Diagrama de estados do programa dos sensores

4.2.2 Software de retaguarda

O software de retaguarda é desenvolvido de acordo com o uso especificado pelo diagrama de caso de uso abaixo:

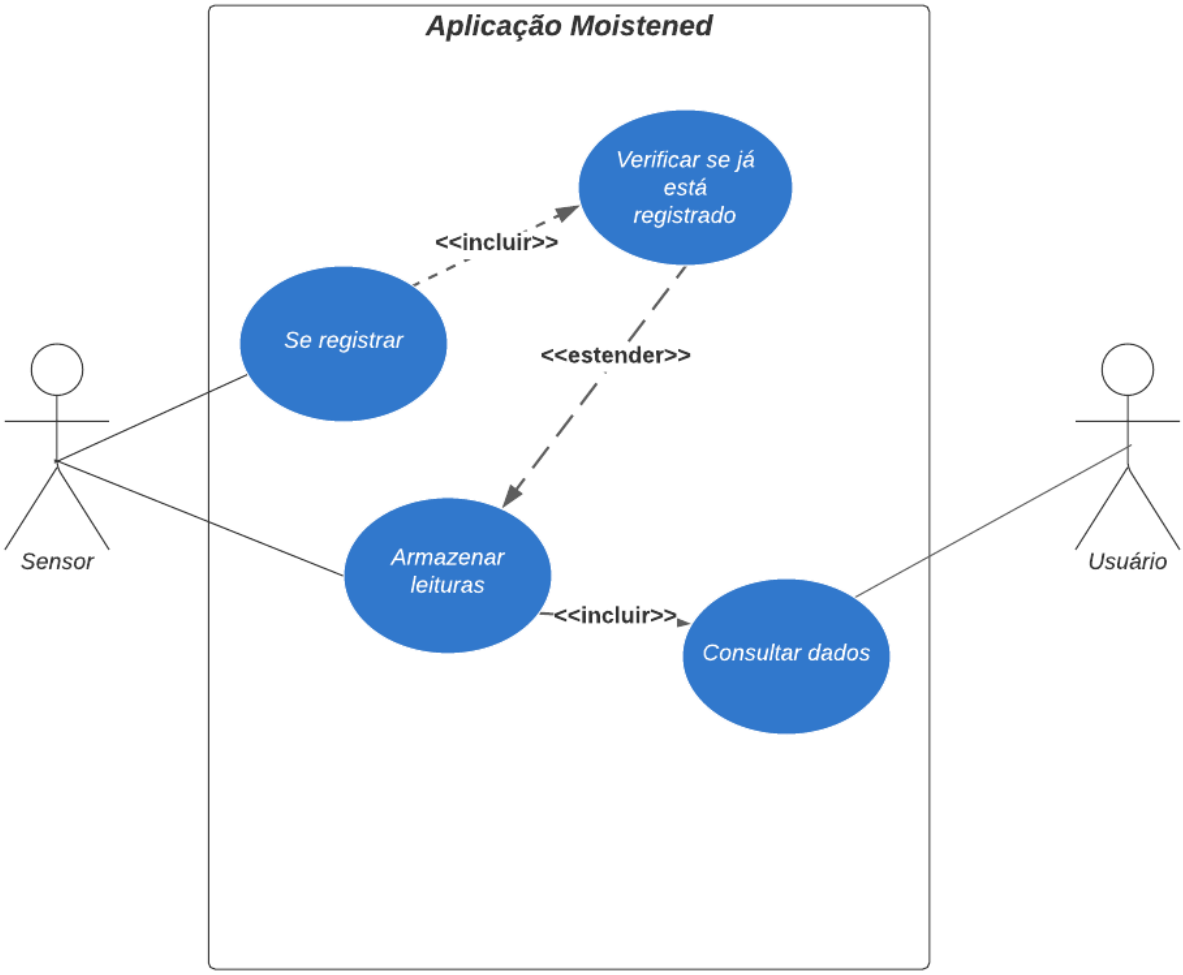

Figura 2 – Diagrama de classes do banco de dados

Conforme exposto do diagrama, a comunicação entre os sensores com a retaguarda é utilizada para armazenar as leituras no banco de dados além de permitir que o sensor se registre, garantindo a possibilidade da expansão do número de sensores utilizados para a medição da umidade do solo.

Com isso podemos utilizar a seguinte estrutura relacional de dados:

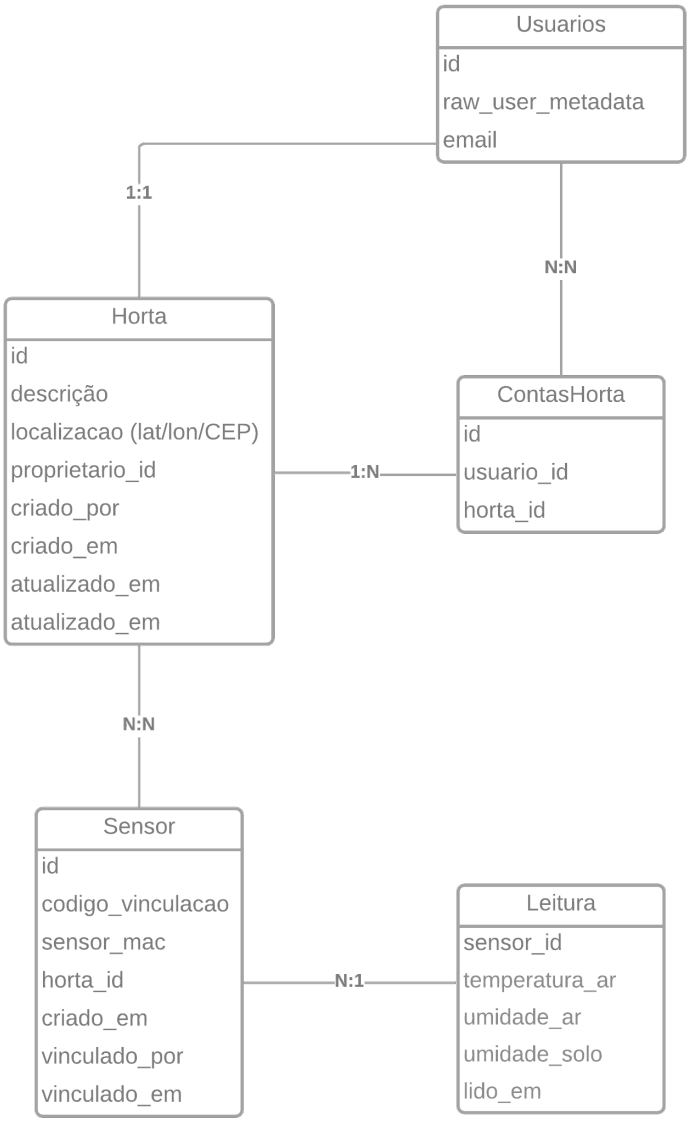

Figura 3 – Estrutura de dados do banco de dados

- Horta: tabela que armazenará as hortas cadastradas pelos usuários, cada usuário pode ter mais de uma horta cadastrada na mesma conta;
- Leitura: tabela que armazenará as leituras dos sensores;
- Sensor: tabela que armazenará os sensores disponíveis para vinculação e os vinculados;
- ContasHorta: tabela que armazenará os usuários convidados por outros usuários a visualizar a horta (vários usuários poderão visualizar uma determinada horta);
- Users: tabela que armazenará os usuários registrados no sistema;

As tabelas possuem as seguintes relações:

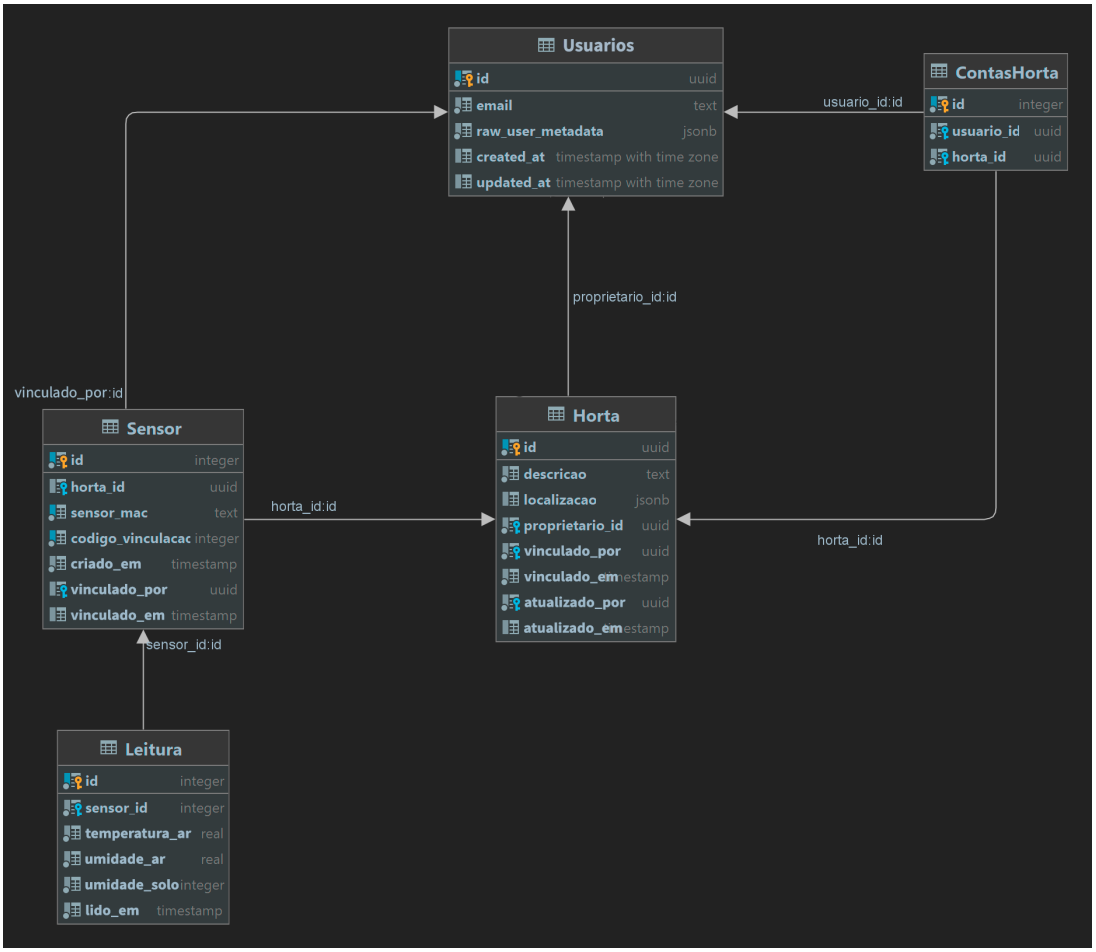

Figura 4 – Diagrama de classes do banco de dados

Para o desenvolvimento da *API REST*, que é responsável por receber as leituras dos sensores e armazená-las no banco de dados, foi definido a utilização de um supertipo da linguagem de programação *web* JavaScript conhecido por TypeScript, isso pela segurança de tipos de dados que a linguagem fornece.

Para banco de dados, o serviço chamado Supabase, uma alternativa de código aberto ao serviço similar chamado Firebase (que é desenvolvido e mantido pela Google), foi o melhor cotado, devido ao preço e ao fato de se utilizar um sistema gerenciador de banco de dados (SGBD) robusto e confiável chamado *PostgreSQL*, além da possibilidade de integração com o serviço de autenticação OAuth 2 de serviços como Google, Twitter e Facebook por exemplo.

Para disponibilizar o acesso a API publicamente, foi definido a utilização um *Framework Web* chamado Next hospedado no serviço gratuito Vercel (anteriormente Zeit).

Iniciou-se o desenvolvimento da *API REST* utilizando o framework para *web* chamado Next.JS utilizando o supertipo da linguagem JavaScript, TypeScript. Com a utilização da biblioteca do próprio *Supabase* para realizar a comunicação com o banco de dados PostgreSQL do serviço.

A *API REST* disponibiliza na rede alguns pontos de acesso (rotas), que são os mais importantes:

- `/api/dados`: Essa rota suporta somente o método HTTP POST, ela é utilizada para receber e armazenar as leituras dos sensores no banco de dados. Ela espera receber os dados das leituras (temperatura e umidade do ar e umidade do solo, além da identificação do sensor);
- `/api/sensor/register`: Essa rota suporta os métodos HTTP POST e HTTP GET, onde no GET, a partir do endereço físico da placa de wifi do sensor (MAC) é possível verificar se o sensor já está na base de dados da aplicação, visto que é possível incluir novos sensores a qualquer momento. Já o método POST espera receber os dados para registro do sensor, são eles, endereço MAC da placa de wifi do sensor e um código de vinculação (utilizado para vincular um sensor à uma horta pela interface *web*) que é gerado a partir do MAC do sensor;

4.2.3 Software no cliente

Em reunião definiu-se que seria mais rápido e mais seguro utilizar um *framework web* reativo chamado *Nuxt.JS* com um módulo chamado Vuetify.JS com os componentes necessários para a construção da interface, como botões, caixas de texto, entre outros componentes. Definiu-se também que a aplicação seria hospedada no serviço de nuvem chamado Vercel.

O framework gera os arcabouços de código necessários para as principais atividades presentes na aplicação, como visualizar os dados, cadastrar uma horta e vincular um sensor por exemplo. A autenticação de usuários é gerenciada pelo Supabase, o que permite abstrair todos o gerenciamento de credenciais ou algo do tipo, permitindo que o desenvolvedor foque em elaborar o *layout* da tela de autenticação e a lógica de autorização.

Figura 5 – Tela de autenticação do sistema

Após a autenticação estar devidamente finalizada, foi possível prosseguir para o desenvolvimento da tela inicial do projeto, a *dashboard*. Nela estão dispostos os gráficos com a umidade do solo no dia e a umidade do solo ao longo da semana vigente, sendo contados a partir de Segunda-Feira, até Domingo.

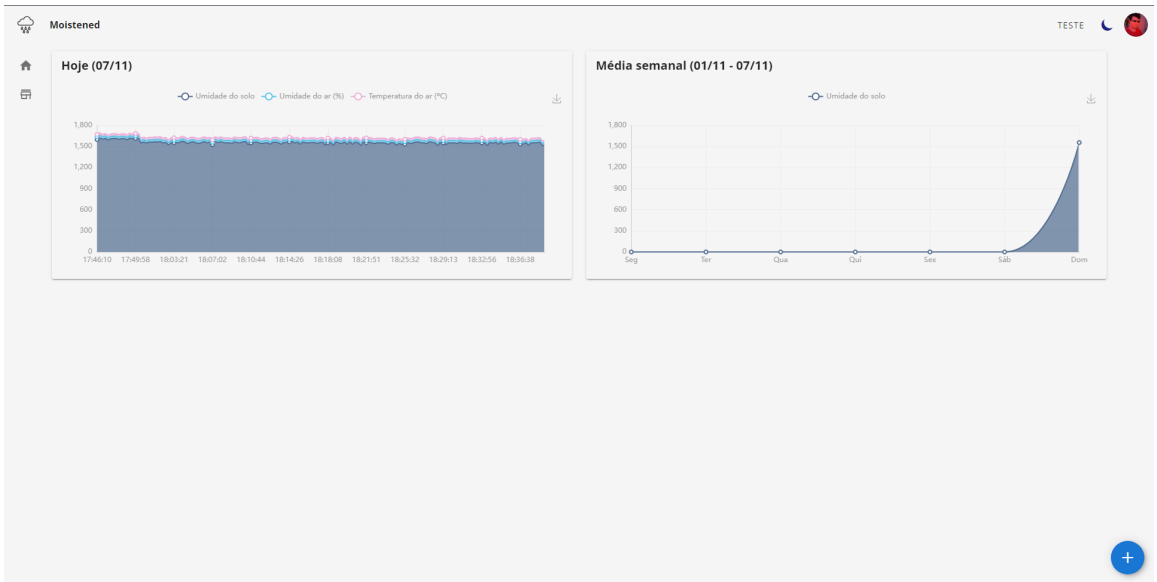

Figura 6 – *Dashboard* da aplicação (após autenticação e com uma horta selecionada)

O cadastro de hortas foi construída utilizando um componente de diálogo do Vuetify, o quê permitiu criar a interface para cadastro, com os campos necessários para o registro da Horta juntamente com uma integração com o serviço gratuito chamado *ViaCep* para preenchimento do endereço e o serviço *Nominatim* para coleta da latitude e longitude da horta.

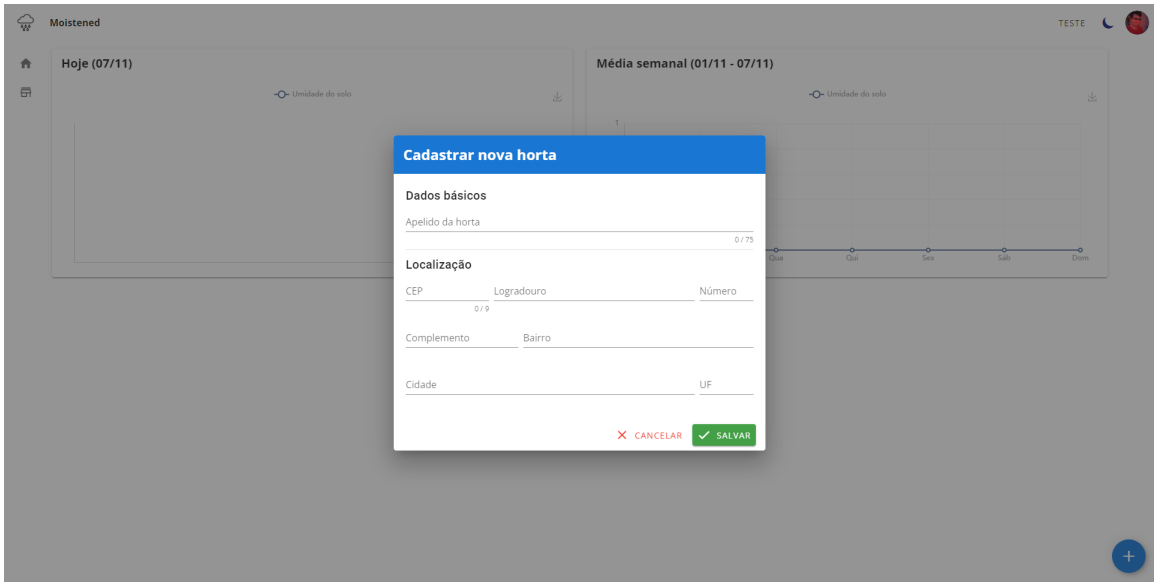

Figura 7 – Cadastro de uma nova horta

4.3 A integração entre o hardware e o software e os testes de operação

A integração entre os dispositivos e a API se deu por meio da *internet*, onde cada sensor envia suas leituras para o servidor e o servidor armazena os dados no banco de dados da aplicação. A requisição HTTP seria realizada a partir dos dispositivos utilizando a biblioteca *requests* do Python (e MicroPython) para o endereço[0] utilizando o método POST, contendo os dados da leitura, juntamente com a data e a hora.

[0] URL:<https://api.moistened.luizg.dev/api/dados>

A implementação do projeto utilizou o serviço de nuvem Vercel, integrado com o serviço de hospedagem e versionamento de código GitHub. Qualquer alteração realizada no código seria automaticamente distribuída e publicada para a *internet*. Sendo isso válido tanto para a API quanto para a interface do usuário. Já no caso dos dispositivos, a gravação do código é manual, visto a impossibilidade da gravação automática via internet, mas ainda assim o código está armazenado jutamente com o código da API.

A montagem dos sensores ocorreram de acordo com a sequência apresentada na figura 5, onde conectou-se a bateria e se fechou a capa de proteção.

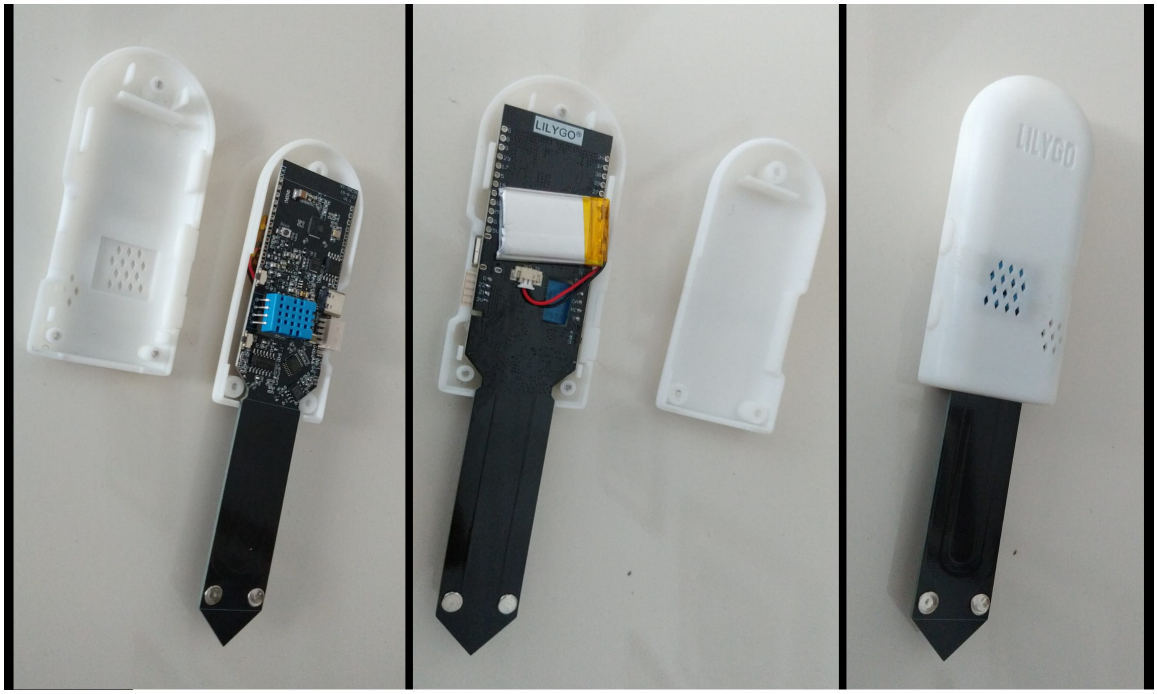

Figura 8 – Passo a passo da montagem dos sensores.

Após a gravação do código nos sensores e a configuração para a conexão com a *internet* via Wi-Fi, os sensores automaticamente começaram a enviar os dados da leitura, sendo possível vinculá-los com uma conta de usuário via interface web.

Para finalizar os testes os sensores foram aplicados ao solo para o monitoramento onde já começaram a enviar os dados ao banco de dados, porém sem os módulos fotovoltaicos que serão utilizados para o fornecimento de energia e carregamento da bateria que serão fixados individualmente a cada sensor.

Figura 9 – Sensor em funcionamento no solo.

Figura 10 – Modulo fotovoltaico utilizado para o fornecimento de energia.

5 Considerações finais

O objetivo geral desse trabalho foi desenvolver um sistema de irrigação automatizado que permita, de maneira distribuída, extrair dados do sensoriamento, salvá-los em um banco de dados para futura consulta bem como sua análise e proporcionar a irrigação de forma remota quando necessário. Nesse sentido, foi realizada uma revisão das áreas de descoberta de conhecimento e computação distribuída visando suportar a proposição do trabalho.

O protótipo desenvolvido atendeu as expectativas gerando resultados satisfatórios e permitindo a produção de dados consistentes das medições da umidade do solo bem como a temperatura e umidade do ar que podem ser expostas através de histogramas, gráficos, grafos, entre outros. A arquitetura distribuída do protótipo demonstrou flexibilidade e escalabilidade podendo ser expandida quando necessário por meio de novos sensores que podem ser adicionados ao sistema de forma simples e prática. Também é cabível melhorias no processo de controle dos solenoides para a irrigação de forma totalmente autônoma com a possibilidade de implementação de um certo nível de inteligência coletando-se os dados de uma estação meteorológica ajudando a prever a necessidade de irrigação.

É possível também escalonar a distância da comunicação dos sensores através da rede LoRa ao qual havia-se previsto inicialmente no projeto devido ao grande alcance em áreas remotas de lavouras.

Toda a fundamentação teórica e método de pesquisa foram pensadas e utilizadas de forma com que trouxessem informações e conhecimento tanto para os pesquisadores quanto para os leitores e usuários, fazendo assim com que todos saíssem favorecidos após o término do trabalho, que no contexto em geral, é muito útil e flexível para possíveis interações futuras, já que basicamente o tema todo abordado gira em torno da tecnologia que está sempre em evolução, ou seja, é possível que em um futuro distante ou não, o tema abordado no trabalho vire algo totalmente comum e padronizado no dia a dia da sociedade.

Referências bibliográficas

ANDRADE, E. *MicroPython: Python para microcontrolador - Embarcados*. 2016. Disponível em: <https://www.embarcados.com.br/micropython/>.

CHAN, M. *SQL vs. NoSQL - what's the best option for your database needs?* 2019. Disponível em: <https://www.thorntech.com/sql-vs-nosql/>.

COPPLESTONE, P. Blog, *Introduction, Supabase Documentation*. 2021. Disponível em: <https://supabase.io/docs/>.

FERNANDES, D. Journal, *Git & Github: O que é? Por que? Como iniciar?* 2018. Publisher: Rocketseat. Disponível em: <https://blog.rocketseat.com.br/iniciando-com-git-github/>.

FRITZ, C. Guide, *Introdução — Vue.js*. 2016. Disponível em: <https://br.vuejs.org/v2/guide/index.html>.

GUEDES, M. *O que é o Vue.js?* 2020. Publisher: Treinaweb. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-o-vue-js>.

LATEEF, Z. *What is JavaScript? | JavaScript Programming*. 2018. Disponível em: <https://www.edureka.co/blog/what-is-javascript/>.

NOLETO, C. *Typescript: o que é, principais conceitos e porquê usar!* 2020. Disponível em: <https://blog.betrybe.com/desenvolvimento-web/typescript/>.

PETRY, G. *Vue.js e Nuxt.js: como utilizar para construir universal apps*. 2020. Disponível em: <https://king.host/blog/2020/09/vue-js-e-nuxt-js/>.

REDHAT. Blog, *O que é API REST?* 2018. Publisher: RedHat. Disponível em: <https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api>.

ROVEDA, U. *O que é Python, para que serve e por que aprender?* 2020. Disponível em: <https://kenzie.com.br/blog/o-que-e-python/>.

SOUZA, I. de. Blog, *PostgreSQL: saiba o que é, para que serve e como instalar*. 2020. Disponível em: <https://rockcontent.com/br/blog/postgresql/>.