MATHEUS GUILHERME GONÇALVES
PAULO MASSAYOSHI HIRAMI
RAFAEL AUGUSTO CONCEIÇÃO BAPTISTA DAS NEVES

AGROSMART: APLICAÇÕES EM AUTOMAÇÃO E IOT PARA AGRICULTURA URBANA E PERIURBANA

De acordo:

Carlos Eduardo Cugnasca

São Paulo

2021

MATHEUS GUILHERME GONÇALVES
PAULO MASSAYOSHI HIRAMI
RAFAEL AUGUSTO CONCEIÇÃO BAPTISTA DAS NEVES

AGROSMART: APLICAÇÕES EM AUTOMAÇÃO E IOT PARA AGRICULTURA URBANA E PERIURBANA

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro de Computação.

Orientador:

Prof. Dr. Carlos Eduardo Cugnasca

São Paulo
2021

FICHA CATALOGRÁFICA

Guilherme Gonçalves, Matheus.

Massayoshi Hirami, Paulo.

Conceição Baptista das Neves, Rafael.

Agrosmart: Aplicações em Automação e IoT para Agricultura Urbana e Periurbana - São Paulo, 2021

Orientador: Prof. Dr. Carlos Eduardo Cugnasca

Monografia - Escola Politécnica da USP

1.Agricultura Urbana; 2. Internet das Coisas; 3. Sistemas Embarcados; 4. Computação em Nuvem; 5. Aplicação Mobile.

AGRADECIMENTOS

Agradecemos a todos que nos apoiaram nessa jornada em especial nosso professor orientador Carlos Eduardo Cugnasca por toda disposição, ajuda e experiência; à nossas famílias pelo suporte incondicional; à nossas namoradas Raissa Pena e Larissa de Medeiros Botecchia por toda ajuda e apoio moral fornecido; aos docentes que tivemos o prazer de acompanhar suas aulas, mais especificamente aos professores: Selma Shin Shimizu Melnikoff, Moacyr Martucci Junior, Pedro Luiz Pizzigatti Corrêa, Bruno de Carvalho Albertini e Reginaldo Arakaki por darem a oportunidade de abordar o tema deste projeto em disciplinas ministradas por eles; aos amigos Arthur Strini e Pedro Birindelli pelo suporte técnico; à empresa Pink Farms por nos receber e nos apresentar sua fazenda vertical; e a todos os amigos e colegas que nos acompanharam durante essa jornada.

Gostaríamos também de agradecer e dedicar este trabalho de maneira especial à memória de José Cláudio Gonçalves, pai do aluno Matheus Guilherme Gonçalves, que foi uma das vítimas do Covid em 2021.

RESUMO

Este projeto de formatura teve como objetivo conceber e implementar uma solução completa para o controle e automação da prática doméstica de agricultura urbana. Levantou-se o panorama do agronegócio brasileiro, abordando-se vantagens e diferenciais da agricultura urbana e periurbana e levando em conta o modelo de negócio de fazendas verticais e os cultivos comunitários urbanos, buscando-se conceber um sistema capaz de monitorar, controlar e automatizar cultivos de pequeno porte. O protótipo desenvolvido, que envolve conceitos de Internet das Coisas, Sistemas Embarcados, Computação em Nuvem e Aplicações Móveis, é capaz de medir e controlar fatores importantes para o bom crescimento de uma planta, como iluminação, umidade, condutibilidade elétrica (correlacionado com a fertilidade) e temperatura. Após as medições, os dados são processados em um dispositivo embarcado, que automatiza os processos de iluminação e irrigação, corrigindo seus valores para seus patamares ideais Todos os dados são coletados e armazenados na nuvem e disponibilizados ao usuário final, a partir de uma interface de um dispositivo de aplicação móvel.

Palavras-chave: Agricultura Urbana. Automação. Internet das Coisas. Sistemas Embarcados. Computação em Nuvem. Aplicação Móvel.

ABSTRACT

The Project proposes a complete solution on controlling and automating domestic practices of urban agriculture. After briefly understanding the Brazilian agribusiness panorama, pointing the advantages and differentials of an urban and peri-urban agriculture and also regarding the business model of vertical farms and urban Community crops. The Project brings a system capable of monitoring, controlling and automating small crops. Involving concepts of: Internet of Things, Embedded Systems, Cloud Computing and Mobile Applications, the system is able to measure important factors to condition a good growth of a plant such as: lighting, humidity, electrical conductivity for fertility, and temperature. After measurements, the data is processed on an embedded device, which automates processes like illuminating and irrigating, fixing those parameters. And then, all that data is sent and stored on clouding services, which are available on a mobile application.

Keywords: Urban Agriculture. Automation. Internet of Things, Embedded Systems. Cloud Computing. Mobile Aplication.

LISTA DE TABELAS

Tabela 1 – APIs.. 88

LISTA DE FIGURAS

Figura 1 - Diagrama de Requisitos 16
Figura 2 - Diagrama de definição de blocos SysML 20
Figura 3 - Módulo de sensores Xiaomi Flora 30
Figura 4 - Leitura do sensor feita por um Raspberry Pi 31
Figura 5 - Módulo de atuadores, microcontrolador e relés 31
Figura 6 - Modulo de atuadores com foco no microcontrolador 33
Figura 7 - Sistema de hardware local. 34
Figura 8 - Diagrama exemplificando os recursos do Mongoose OS 36
Figura 9 - Diagrama exemplificando todos os recursos do AWS IoT *Greengrass* 40
Figura 10 - Diagrama de alto nível dos principais recuros AWS 43
Figura 11 - Diagrama de alto nível representando os principais elementos do sistema 48
Figura 12 - Diagrama de blocos SysML representando a hierarquia do sistema 56
Figura 13 - Diagrama exemplificando os 4 estágios da arquitetura IoT 56
Figura 14 - Diagrama da arquitetura do sistema embarcado 59
Figura 15 - Grupo criado para testes 61
Figura 16 - Dispositivo de núcleo criado 62
Figura 17 - Componentes instalados no dispositivo de núcleo 62
Figura 18 - Pseudo código 63
Figura 19 - Logs de execução do componente local 63
Figura 20 - Cliente teste MQTT na nuvem evidenciando a sincronização entre as mensagens geradas nos tópicos MQTT locais e os tópicos da nuvem. 64
Figura 21 - Rule criada na nuvem para acionar uma função lambda específica 65
Figura 22 - Função lambda criada para processar as leituras dos eventos e salvá-las no banco de dados 65
Figura 23 - Pseudocódigo da função lambda 66
Figura 24 - Dados do sensor e das medidas realizadas armazenados no banco de dados 66
Figura 25 - handler do Dynamo Stream 68
Figura 26 - Funcionamento do health score 69
Figura 27 - Algoritmo do health score 69

Figura 28 - Gerenciamento de projetos embarcados 70
Figura 29 - Comando ‘mos build’ 72
Figura 30 - Comando ‘mos flash’ 73
Figura 31 - comando 'mos wifi' 74
Figura 32 - comando 'mos aws-iot-setup' 75
Figura 33 - Continuação figura 33 76
Figura 34 - Mockups do protótipo 80
Figura 35 - Continuação dos mockups 81
Figura 36 - Índice principal da modelagem de dados 83
Figura 37 - Primeiro índice secundário 84
Figura 38 - Segundo índice secundário 84
Figura 39 - Esquema de criação de usuário 86
Figura 40 - Esquema de autenticação de usuário 86
Figura 41 - Diretórios do projeto Flutter 90
Figura - 42 Diretório lib do projeto Flutter 91
Figura 43 - Camada de *views* 92
Figura 44 - Estufa vista por fora 93
Figura 45 - Estufa por dentro e posicionamento do Raspberry 94
Figura 46 - Sensor, mecanismo de irrigação e resultados do crescimento 95
Figura 47 - Módulo de atuadores 96
Figura 48 - Painel de luz e reservatório de água e bomba de irrigação 97
Figura 49 - Resultados da integração da horta com o sistema 98
Figura 50 - Fluxo de autenticação 99
Figura 51 - Fluxo de criação de planta 100
Figura 52 - Fluxo de cadastro de sensor 100
Figura 53 - Fluxo de cadastro de atuador 101
Figura 54 - Fluxo de pareamento do sensor 102
Figura 55 - Fluxo pareamento de Atuador 102
Figura 56 - Históricos de medição para Planta 103
Figura 57 - Histórico de medição do Sensor 103
Figura 58 - Histórico de medições do Atuador 104

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
AWS	Amazon Web Services
BLE	Bluetooth Low Energy
CaaS	Container as a Service
CLI	Command Line Interface
CORS	Cross-origin Resource Sharing
CRUD	Create, Read, Update, Delete
FaaS	Function as a Service
FPGA	Field-Programmable Gate Arrays
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IaC	Infrastructure as Code
IoT	Internet of Things
JSON	JavaScript Object Notation
MQTT	Message Queuing Telemetry Transport
OLTP	Online Transaction Processing
OS	Operating System
PaaS	Platform as a Service
RESTful	Representational State Transfer
RM-ODP	Reference Model of Open Distributed Processing
ROM	Read Only Memory
RTOS	Real Time Operating System
SaaS	Software as a Service
SDK	Software Development Kit
TI	Tecnologia da Informação
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus

SUMÁRIO

1. INTRODUÇÃO 9

1.1. Objetivo 9

1.2. Motivação 9

1.3. Justificativa 10

1.4. Organização 10

2. ASPECTOS CONCEITUAIS 12

2.1. Agronegócio no Brasil 12

2.2. Agricultura Urbana e Fazendas Verticais 13

2.2.1. Subsistema de iluminação 17

2.2.2. Subsistema de irrigação 17

2.2.3. Subsistema de temperatura 18

2.2.4. Subsistema de qualidade do substrato 19

2.2.5. Subsistema de qualidade do ar 19

2.3. Agricultura Urbana no contexto Doméstico, Comunitário e Institucional 21

2.4. IoT e Sistemas Embarcados 24

2.5. Computação em Nuvem 25

2.6. Arquitetura *Serverless* 26

2.6.1. Microsserviços 26

2.6.2. Container Stateless 26

2.7. Infraestrutura como Código 27

2.8. Conclusão 28

3. TECNOLOGIAS UTILIZADAS 29

3.1. Hardware 29

3.1.1. Sensores 29
3.1.2. Atuadores 31
3.1.3. *Gateway* de Sensores e Atuadores 32
3.1.4. Dispositivo de borda 33
3.2. Software 35
3.2.1. Mongoose OS e Raspberry Pi OS 35
3.2.2. *Serverless Framework* e *Cloudformation* 37
3.2.3. AWS 37
3.2.4. Node.js 43
3.2.5. Google *Firebase* 43
3.2.6. Flutter 44
4. METODOLOGIA DE TRABALHO 45
4.1. Pesquisa inicial 45
4.2. Estudo sobre viabilidade de projeto 45
4.3. Elaboração de proposta 46
4.4. Elaboração de monografia 48
5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA 49
5.1. Requisitos Funcionais e não funcionais 49
5.2. Visão Empresa 50
5.3. Visão Informação 50
5.4. Visão Computação 51
5.5. Visão Engenharia 51
5.6. Visão Tecnologia 51
5.6.1. Sensores e Atuadores 51
5.6.2. Dispositivos de borda 51
5.6.3. *Backend* 52

5.6.4. *Frontend* 52
6. PROJETO E IMPLEMENTAÇÃO 53
6.1. Metodologia de projeto de Sistemas Embarcados 53
6.1.1. Aplicação Dedicada 53
6.1.2. Interação com o Ambiente 53
6.1.3. Robustez 53
6.1.4. Eficiência garantida por Projeto 53
6.1.5. Etapas do Projeto 54
6.2. Hierarquia do Sistema 55
6.3. Quatro estágios da arquitetura IoT 56
6.4. Arquitetura do sistema embarcado alvo 58
6.5. Aplicação Cliente 76
6.5.1. Questionário de opinião pública 77
6.6. Backend 81
6.6.1. Modelagem de dados 82
6.6.2. Implementação 85
6.7. Frontend 89
7. RESULTADOS E TESTES 93
7.1. Resultados 93
7.1.1. Sistema Embarcado 93
7.1.2. Aplicação Móvel 98
8. CONSIDERAÇÕES FINAIS 105
8.1. Conclusões do projeto de formatura 105
8.1.1. Matheus 105
8.1.2. Paulo 105
8.1.3. Rafael 106

8.2. Contribuições 107
8.3. Perspectivas de continuidade 107
REFERÊNCIAS 109

1. INTRODUÇÃO

1.1. Objetivo

O objetivo deste trabalho de conclusão de curso foi conceber e construir um sistema para auxiliar no monitoramento e controle de sistemas de produção vegetal doméstica de plantas alimentícias, no contexto da agricultura urbana.

1.2. Motivação

A agricultura brasileira se caracteriza pela predominância da produção em locais afastados do consumo. Isso traz diversas desvantagens, como o desperdício e encarecimento do produto, aumento da poluição, impacto ambiental, aumento da necessidade de processamento dos alimentos e prejuízos à saúde dos consumidores.

Sabendo que as expansões do agronegócio resultam em problemas ambientais e sociais, uma alternativa que vem sendo explorada é a agricultura urbana e periurbana, que visa complementar as atividades agrícolas desenvolvidas no meio rural com o diferencial de estar integrada aos sistemas econômicos e ecológicos urbanos (LEGNAIOLI, 2021). Ela traz vários benefícios em relação ao modelo tradicional de agricultura, como a redução do desperdício com transporte, armazenamento e água, produção de alimentos mais nutritivos e menos processados, revitalização de espaços urbanos ociosos, contribuição para o microclima e biodiversidade, redução do uso de pesticidas e contribuição para economia, saúde e segurança alimentar.

Uma das principais manifestações da agricultura urbana são as fazendas verticais, que aliam a ideia de produção agrícola nas cidades com a utilização de tecnologia para impulsionar sua produtividade. Segundo os autores do trabalho 29 do ENGEMA 2020 (CUGNASCA et al., 2020) As fazendas verticais são um tipo de fazenda caracterizada pelo cultivo dentro de casa, nas quais se busca maximizar a produção e eficiência por metro quadrado, através de plantações em múltiplos níveis no eixo vertical.

Mesmo com as fazendas verticais, seu escopo continua comercial e o consumo pelo usuário final se dá por meio indireto a partir de mercados, feiras, hortifrútis e restaurantes. Sendo assim,

a motivação deste projeto é de aproximar ainda mais o consumidor final dos conceitos da agricultura urbana e facilitar o processo de consumo direto de plantas, a partir do cultivo doméstico através de uma aplicação para monitoramento e controle de hortas domésticas, utilizando conceitos similares aos das fazendas verticais, em escopo reduzido que se adapte aos requisitos culturais, sociais, ambientais e financeiro de uma plantação caseira.

1.3. Justificativa

Com todas as vantagens e desvantagens elencadas, chega-se à conclusão que a agricultura urbana auxilia positivamente como um modelo complementar à agricultura rural, principalmente nas grandes metrópoles, apresentando maior sustentabilidade ambiental e social. Um de seus maiores benefícios é o aproveitamento de espaços ociosos em regiões metropolitanas, contribuindo para geração de conhecimento, empregos na região, colheitas com maior qualidade e tempo de prateleira por conta da proximidade com o consumidor final e manutenção de um bioclima mais saudável nos ambientes urbanos (LEGNAIOLI, 2021).

Assim como as fazendas verticais se aproveitam de prédios e galpões ociosos para sua produção em escala comercial, é possível aproveitar espaços domésticos ociosos para produção em escopo reduzido. Utilizando os mesmos mecanismos de automação e controle das fazendas verticais, de maneira simplificada e com menor custo, é possível construir cultivos de pequeno porte dentro de casas e apartamentos, otimizando ecologicamente espaços urbanos e contribuindo para o crescimento da agricultura urbana e periurbana.

Portanto, este projeto visa integrar o uso de tecnologias para construção de um aplicativo capaz de informar medidas dos atributos mais importantes para o desenvolvimento de uma planta captado por sensores e atuadores que auxiliam na automatização deste processo, facilitando o cultivo para usuários desde os mais leigos até experientes em agricultura.

1.4. Organização

Este documento está organizado em capítulos que abordam as etapas de desenvolvimento de um projeto, dos aspectos conceituais à implementação do protótipo.

O Capítulo dois discorre sobre aspectos conceituais relacionados ao projeto, como o agro-negócio brasileiro, agricultura urbana em suas vertentes comerciais e domésticas, Internet das Coisas, Sistemas Embarcados, Computação em Nuvem, Arquitetura *Serverless* e Infraestrutura como Código.

O Capítulo três trata das tecnologias empregadas no projeto. Dividindo-se em duas partes: elementos de Hardware do Sistema Embarcado e elementos de Software presentes na aplicação e servidor.

O Capítulo quatro apresenta a metodologia de trabalho empregada no desenvolvimento do trabalho, começando pelas pesquisas iniciais e visitas técnicas realizadas para contextualizar o domínio do problema. São feitos também os estudos sobre viabilidade do projeto, elaboração de proposta e elaboração da monografia.

O Capítulo cinco abarca as especificações de requisitos do sistema, através das cinco visões: Visão de Empresa, Informação, Computação, Engenharia e Tecnologia.

No Capítulo seis está a documentação da implementação e testes do protótipo, assim como todos os resultados do trabalho e por fim estão apresentadas as referências que viabilizaram a realização do projeto.

No Capítulo sete estão os resultados, testes e avaliações do projeto com uma demonstração do projeto e avaliações dele. No Capítulo oito estão as considerações finais do projeto e por fim as Referências Bibliográficas usadas na monografia.

2. ASPECTOS CONCEITUAIS

Neste capítulo serão discutidos os principais aspectos conceituais que permeiam o objeto de estudo deste projeto. São eles: um breve resumo do cenário do agronegócio brasileiro, seguido por definições de agricultura urbana no contexto de fazendas verticais e hortas domésticas, comunitárias e institucionais, e, por fim, elementos de tecnologias relacionados ao projeto, como Internet das Coisas, Sistemas Embarcados, Computação em Nuvem, Arquitetura *Serverless* e Infraestrutura como Código.

2.1. Agronegócio no Brasil

Desde os anos de 1980, a produção agropecuária brasileira se desenvolveu de tal forma que o Brasil se tornou um grande fornecedor de alimentos para o mundo. O agronegócio tem sido reconhecido como um vetor crucial do crescimento econômico brasileiro. Em 2019, por exemplo, a soma de bens e serviços gerados no agronegócio chegou a R$1,55 trilhão ou 21,4% do PIB do país. Dentre os segmentos da economia, a maior parcela é a do ramo agrícola, que corresponde a 68% deste valor (R$ 1,06 trilhão), e a pecuária corresponde a 32%, ou R$ 494,8 bilhões (SUPERINTENDÊNCIA TÉCNICA DA CNA E CEPEA , 2019).

Por outro lado, conforme apontam dados do Sistema Nacional de Informações sobre o Saneamento (SNIS), vinculado ao Ministério das Cidades, a média diária de consumo de água de cada brasileiro é de 150 litros, o que corresponde a uma média anual de 10,4 trilhões de litros. Desse total, a agricultura recebe pouco mais de 7 trilhões de litros, dos quais 3 trilhões acabam desperdiçados, seja por irrigações executadas de maneira incorreta, ou ainda pela falta de controle do produto (SNIS, 2020).

Além disso, a agricultura brasileira usou 539,9 mil toneladas de pesticidas em 2017, segundo os dados mais recentes do Instituto Brasileiro de Meio Ambiente (IBAMA). Isso representou um gasto de US$ 8,8 bilhões, de acordo com a associação que representa os fabricantes, a ANDEF (MORAES, 2019).

Desta forma, é possível perceber uma dicotomia no desenvolvimento do agronegócio no Brasil: por um lado, o crescimento deste setor impulsiona de forma significativa a economia do

país, sendo responsável por uma fatia considerável do PIB e pelo superávit cambial no balanceamento dos negócios feitos pelo país; porém, por outro lado, traz sérias consequências ambientais e sociais caso a lógica de desenvolvimento deste setor continuar sendo ditada de maneira expansionista e latifundiária. Assim, além de se pensar em maneiras de se conduzir o agronegócio de maneira mais sustentável e igualitária, é importante pensar em alternativas e outros modelos de negócios, como produzir na própria cidade parte dos alimentos consumidos nos grandes centros urbanos.

2.2. Agricultura Urbana e Fazendas Verticais

O crescimento populacional nas cidades traz consigo muitos desafios de gerenciamento público, dentre os quais a necessidade crescente de fornecer alimentos às famílias residentes nesses ambientes. Os índices de pobreza das populações urbanas também têm crescido, assim como a dificuldade de acesso à alimentação básica, e este cenário piorou com o avanço da pandemia do Coronavírus no país.

Para procurar alternativas que auxiliem no combate à fome e desnutrição nos centros urbanos, surgiu o conceito de agricultura intraurbana e periurbana, que é definido por agências das Nações Unidas, tais como United Nations Development Programme (UNDP) e Food and Agriculture Organization of United Nations (FAO), como a utilização de pequenas superfícies situadas dentro das cidades ou em suas respectivas periferias para produção agrícola e criação de pequenos animais, destinados ao consumo próprio ou à venda em mercados locais. Assim, a prática da agricultura urbana que compreende o exercício de diversas atividades relacionadas à produção de alimentos e conservação dos recursos naturais dentro dos centros urbanos ou em suas respectivas periferias, surge como estratégia efetiva de fornecimento de alimentos, de geração de empregos, além de contribuir para segurança alimentar e melhoria da nutrição dos habitantes das cidades (MACHADO et al., 2002).

Um modelo de agricultura urbana que vem sendo aplicado no mundo são as fazendas verticais. Elas podem ser definidas como culturas em ambientes fechados, com utilização de células climáticas, nas quais o plantio é feito na forma vertical, com controle de luz, umidade e temperatura, dentre outros, para maximizar a produção em ambiente limitado (LEBLANC, 2019). Assim, pode-se dizer que as fazendas verticais utilizam vários conceitos de tecnologia

para aumentar a eficiência na produção de alimentos e assim se tornar um modelo viável e mais sustentável de agricultura.

Este modelo tem diversas vantagens em relação ao modelo tradicional de agricultura, como o cultivo perene, proteção contra variações climáticas, ausência da necessidade de pesticidas no cultivo, economia de água (até 95%), uso de fontes renováveis de energia (como solar e eólica, entre outras), crescimento urbano sustentável e melhoria no microclima das cidades, colheitas confiáveis, alta produtividade, diminuição do custo de produção e transporte, diminuição das perdas em todos os ciclos do cultivo, eliminação da necessidade de tornar terras naturais em terras agrícolas, uso de espaços urbanos ociosos, estímulo ao conhecimento e trabalhos rurais em áreas urbanas e emprego de tecnologias. As limitações das fazendas verticais em relação ao modelo tradicional de agricultura é a demanda maior de energia, custos iniciais maiores, monitoramento constante e automação complexa, polinização manual e limitação de culturas (CUGNASCA et al., 2020).

Conforme apresentado nas vantagens desta modalidade de agricultura, a sustentabilidade é o foco primordial das fazendas verticais. Os altos custos iniciais de implantação e o maior consumo energético são justificados pelos ganhos econômicos, ambientais e sociais de longo prazo que este tipo de cultura tem a oferecer, principalmente quando se usa fontes de energia renovável para o funcionamento das fazendas. Assim, pode-se afirmar que o modelo de negócios de uma fazenda vertical está fundamentado em um tripé de sustentabilidade, pois sua implementação implica em ganhos ambientais, sociais e econômicos, que são notados no aumento de produtividade, qualidade dos alimentos, redução de custo e impacto ambiental no gerenciamento de insumos da fazenda e integração do local de produção de alimentos na cidade, o que fomenta a geração de emprego, pesquisa e desenvolvimento e melhoria da qualidade de vida nos centros urbanos (CUGNASCA et al., 2020).

O sistema de controle de uma fazenda vertical consiste prioritariamente no controle das necessidades básicas de cultivo e crescimento de uma planta. São eles: luz, temperatura, água, umidade e nutrientes. Para verticalizar o processo deve-se levar em conta toda a logística de produção, atendendo de maneira estratégica todos os andares de uma fazenda vertical. O sistema de controle irá variar de acordo com o tipo de plantio que se deseja implementar.

Existem basicamente três tipos de plantio:

- **Hidroponia**: tipo predominante, envolve plantio em soluções nutritivas livres de solo. As raízes das plantas são submersas na solução nutritiva. A solução é frequentemente monitorada e controlada para garantir a correta manutenção dos componentes químicos nas devidas concentrações (UFU, 2021).
- **Aeroponia**: técnica desenvolvida pela NASA em 1990, através de um programa que buscava maneiras eficientes de se cultivar no espaço. É definido pelo plantio em um ambiente nebuloso, sem solo e com um uso muito reduzido de água. É o sistema mais eficiente, reduzindo o consumo de água em até 90% em comparação com a hidroponia(CABALLERO, 2021).
- **Aquaponia**: um sistema aquapônico utiliza o conceito de hidroponia adicionando peixes no tanque de solução nutritiva. Os resíduos dos peixes, altamente nutritivos para as plantas, são usados como fonte de nutrientes para o sistema. As plantas, por sua vez, filtram e purificam esses resíduos, tornando a água reaproveitável para a reposição no tanque dos peixes (EMBRAPA, 2021).

Fazendas verticais exigem boa conectividade à rede e um bom sistema tecnológico para seu funcionamento e, de modo geral, estão mais próximas dos centros urbanos, por isso o termo “fazendas verticais” pode também ser entendido como “fazendas digitais” (CUG-NASCA et al., 2020). Dado esse cenário, é necessário entender quais áreas da tecnologia são aplicadas nesse método de agricultura.

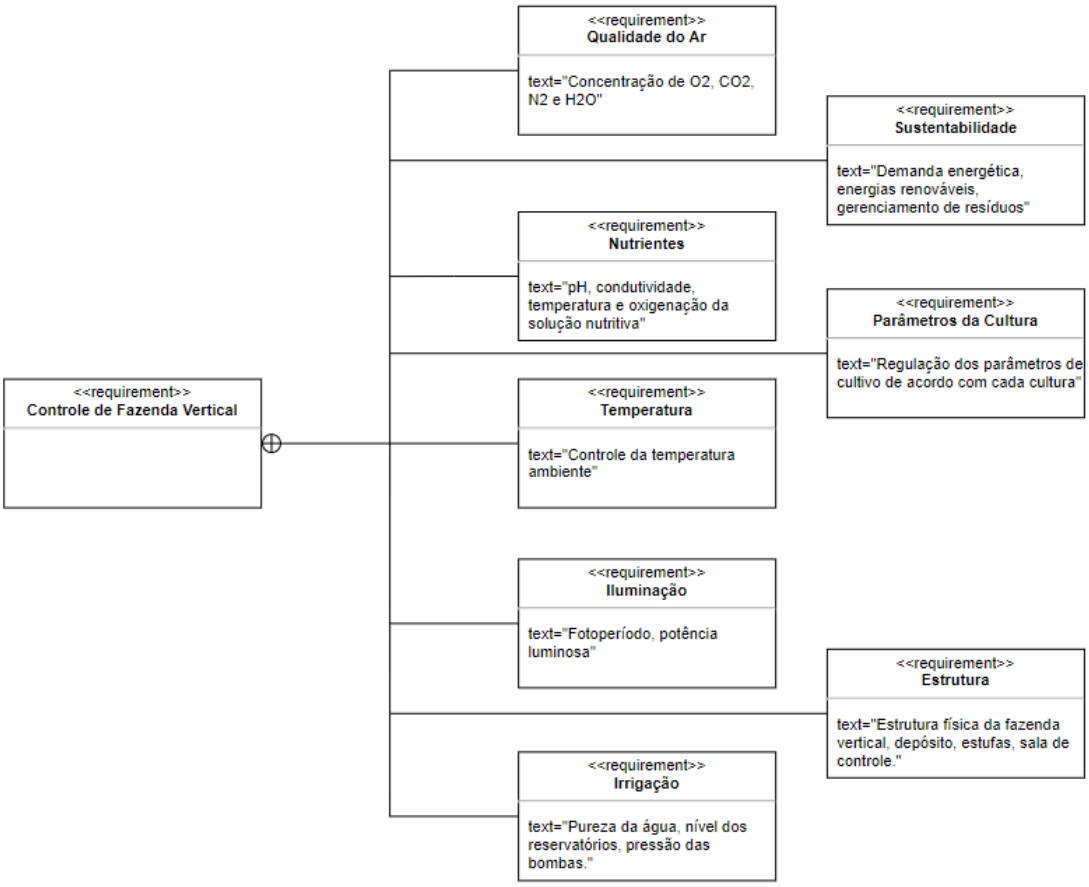
Realizar monitoramento e automação de processos em uma fazenda vertical é necessário, visto que se trata de um sistema de alta complexidade. A Internet das Coisas (*Internet of Things* - IoT e Sistemas Embarcados vêm sendo comumente utilizados para garantir a estabilidade e tolerância a falhas para aplicações dessa natureza. Dentre as possíveis funções a serem automatizadas e controladas estão:

- controle de luminosidade;
- controle da qualidade do substrato;
- aplicação de fertilizantes;

- monitoramento da qualidade do ar;
- monitoramento da qualidade do substrato;
- controle da qualidade do meio aquoso (em hidroponia);
- controle de temperatura;
- histórico de desenvolvimento do plantio;
- visualização e cruzamento de dados do plantio.

Assim, os principais requisitos de sistema para controle de uma fazenda vertical podem ser resumidos num diagrama de requisitos SysML apresentado na Figura 1.

Figura 1 - Diagrama de Requisitos



Fonte: os autores.

Como o cultivo de uma plantação exige o monitoramento e controle de diversos parâmetros, o sistema de uma fazenda vertical abarca diversos subsistemas, tornando o sistema completo consideravelmente complexo. Nesta seção, serão considerados apenas os subsistemas

responsáveis pelo controle e monitoramento das estufas, que são o núcleo do modelo de negócio das fazendas verticais. O sistema de uma fazenda vertical pode ser dividido em cinco subsistemas principais, que serão listados abaixo.

2.2.1. Subsistema de iluminação

Responsável pelo gerenciamento da iluminação das estufas. Pode abarcar tanto iluminação natural quanto artificial. Para lidar com a iluminação natural, é conveniente que as paredes do edifício sejam transparentes, assim como o teto. Caso essas estruturas não possam ser de material transparente (tipicamente vidro), é interessante implantar estruturas que possam ser abertas durante o dia para aproveitar a luz natural. Nesses casos, pode-se fazer um controle automático da abertura dessas estruturas, levando em consideração parâmetros como luminosidade, chuva e qualidade do ar.

Para o controle da luminosidade artificial, é necessário ter um controlador que acione e desligue os painéis de luz de acordo com o fotoperíodo das culturas. Para realizar esta tarefa, são necessários sensores de iluminância, que medem a intensidade do feixe luminoso em lux. Assim, o controlador irá fazer a leitura dos sensores de iluminância e, a partir da intensidade luminosa lida e das especificações da cultura alvo, acionar os painéis luminosos. Painéis de luz mais avançados possuem a funcionalidade de controle de potência luminosa. Para esses casos, é interessante adequar o acionamento dos painéis de luz com a potência necessária para alcançar a iluminância adequada para a cultura alvo.

2.2.2. Subsistema de irrigação

Responsável pelo gerenciamento da irrigação das plantas, assim como do nível de água nos reservatórios. A irrigação nas fazendas verticais é feita através de um sistema de tubulação interligado aos reservatórios de água. Para fazer com que a água chegue até as plantas, precisa-se de um conjunto de bombas de água para fornecer a pressão necessária que garanta um fluxo contínuo na tubulação. Assim, um controlador fica responsável por checar o nível de água no reservatório, e caso seja preciso realizar a irrigação, aciona o sistema de bombas para fornecer água às plantas.

O método de irrigação varia conforme o tipo de plantação. Na hidroponia e aquaponia, essa água (já com os nutrientes) é depositada nas canaletas de plantio, onde as raízes das plantas

ficam submersas na solução. Já na aeroponia, existem aspersores nos terminais da tubulação que fornecem a mesma solução nutritiva para as raízes da planta que estão suspensas no ar, através da nebulização dessa solução. O sistema de canaletas serve apenas para coletar a água residual deste processo.

O subsistema de irrigação também é responsável por lidar com o nível dos reservatórios de água. Existem diversas fontes de água para os reservatórios numa fazenda vertical. Um deles é a coleta de água da chuva, que normalmente é feita na cobertura e arredores dos prédios, e armazenada em reservatórios auxiliares da fazenda. A água residual do processo de irrigação também pode ser reaproveitada nas fazendas verticais. Este reaproveitamento é feito através de duas maneiras, pela drenagem das canaletas de plantio e pela umidade do ar que tende a aumentar pela transpiração das plantas.

Para coletar a umidade do ar, são usados condensadores, que servem tanto para reaproveitar a água de transpiração das plantas quanto para regular a umidade do ambiente. Além disso, podem ser usadas fontes externas de água para alimentar o sistema caso nenhuma das outras fontes tenha água o suficiente para a realimentação dos reservatórios. Antes de realimentar os reservatórios, é importante checar a qualidade da água e submetê-la a um processo de filtragem caso não esteja de acordo com os parâmetros de qualidade do sistema.

2.2.3. Subsistema de temperatura

Responsável pelo controle de temperatura ambiente da fazenda vertical, este subsistema abarca instrumentos de medição como o termômetro e termostato e abarca boa parte da estrutura que envolve a fazenda. Primeiramente o ambiente deve ser isolado termicamente do exterior por meio de uma câmara de controle, e por meio de um sistema de refrigeração industrial como um ar-condicionado e um controlador de temperatura, torna-se possível o controle da temperatura interna propícia para o cultivo levantados na Tabela 1. Para o controle de temperatura inicialmente é feita a medição de forma automática e caso a temperatura esteja acima dos parâmetros ideais o ar-condicionado é acionado ou diminui-se a temperatura do termostato para que o ambiente atinja a temperatura ideal e por fim o ar-condicionado pode parar ou diminuir sua operação.

2.2.4. Subsistema de qualidade do substrato

Este subsistema é responsável pela qualidade do substrato, envolvendo os parâmetros necessários para o crescimento saudável das culturas da fazenda vertical. Como os métodos de plantio nas fazendas usam uma solução nutritiva como substrato (água mais nutrientes), este subsistema possui bastante importância na manutenção da saúde das plantas, e impactam diretamente no rendimento e qualidade do produto.

Os quatro parâmetros mais importantes de controle para garantir a qualidade do substrato são a temperatura da solução, o pH, a condutividade elétrica e a concentração de oxigênio. Assim, são necessários medidores de pH, condutividade, temperatura e oxigenação em meio líquido. Além disso, para o controle desses parâmetros, são necessários um controlador para aquecer e resfriar a solução, mantendo dentro dos parâmetros desejados, uma bomba para fornecer oxigênio para a solução e outras duas bombas, uma para controlar o pH da solução através da adição de substâncias para alterar o pH para o valor alvo, e outra para controlar a concentração de nutrientes da solução, principalmente nitrogênio, fósforo e potássio, adicionando essas substâncias quando a solução estiver com valores abaixo do esperado.

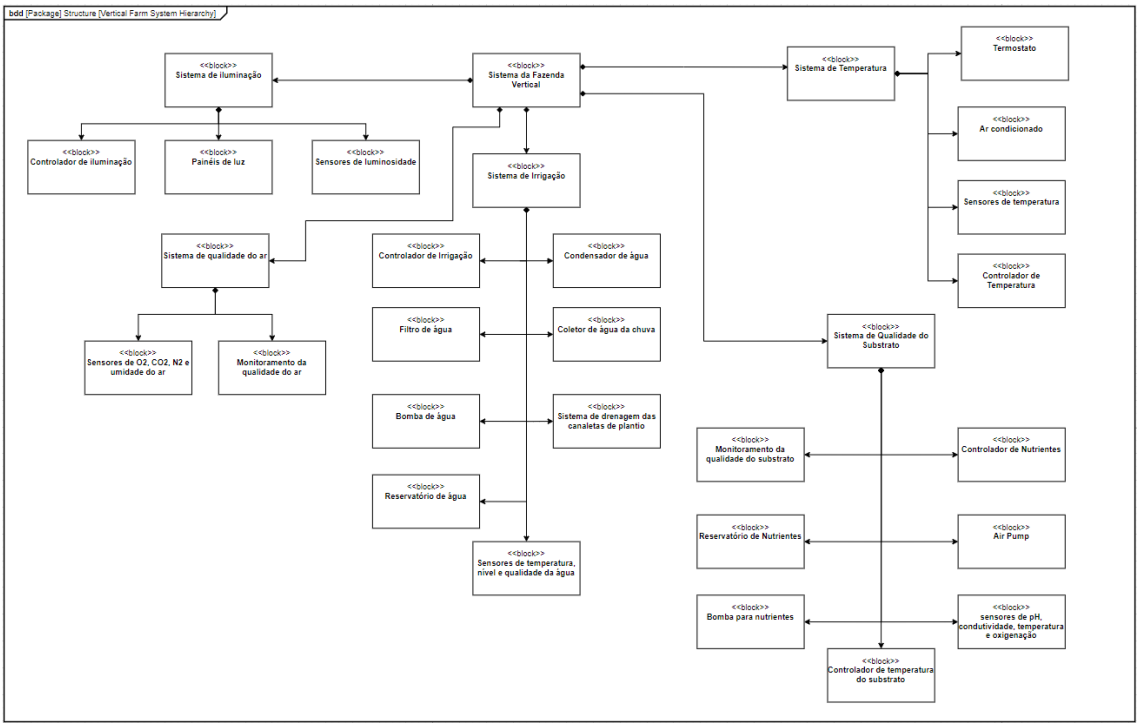
A manutenção da qualidade do substrato é essencial para o correto funcionamento de uma fazenda vertical, e problemas de qualidade podem comprometer uma colheita inteira. Por isso, além da automatização desses parâmetros, é essencial um monitoramento rigoroso deles, além de rotinas de inspeção e correção manual para casos não abordados pela automação.

2.2.5. Subsistema de qualidade do ar

Fator importante para o crescimento das plantas, uma vez estando em ambiente controlado deve-se dar ainda mais importância para este subsistema. Uma vez que a iluminação resulta em diferentes intervalos em razão do fotoperíodo, os níveis de oxigênio, gás carbônico e nitrogênio devem ser monitorados, assim como a pressão do ambiente interno que são controladas por exaustores responsáveis pela entrada e saída de ar. O conjunto de sensores destes gases possibilita que a qualidade do ar possa ser monitorada.

Na Figura 2 encontra-se um diagrama de definição de blocos para o sistema da fazenda vertical. Este diagrama abarca apenas os subsistemas mais importantes para o funcionamento das estufas onde são cultivadas as plantas da fazenda. Não foram levados em consideração os sistemas que auxiliam na colheita, armazenamento, distribuição, geração de energia, gerenciamento de resíduos dentre outros, pois aumentariam o escopo do trabalho além do desejado. Porém, o escopo apresentado no diagrama é suficiente para entender os principais subsistemas que compõem o controle de uma fazenda vertical, assim como seus principais componentes.

Figura 2 - Diagrama de definição de blocos SysML



Fonte: os autores.

A seguir será contemplada a agricultura urbana de porte doméstico, que será o alvo de estudo e implementação deste projeto. Os conceitos tecnológicos apresentados nesta seção serão usados como base para definir o escopo de implementação do sistema alvo.

2.3. Agricultura Urbana no contexto Doméstico, Comunitário e Institucional

Outra manifestação da agricultura urbana é o cultivo doméstico e comunitário. De acordo com Hogdson, Campbell e Bailkey, hortas domésticas são cultivos de alimentos privados, localizados na frente ou nos fundos da casa, em pátios internos, na laje, em varandas, soleiras portões, paredes ou porões de residências privadas uni ou multifamiliares, acompanhada por indivíduos ou empresas de jardinagem. As hortas comunitárias, segundo os mesmos autores, são plantações de alimentos ou plantas ornamentais de pequena a média escala em lotes contínuos ou descontínuos, localizados em propriedades públicas ou privadas, em áreas residenciais, operada e administrada coletivamente por um grupo.

As atividades de jardinagem e os produtos são utilizados para consumo ou educação, entretanto, eles também podem ser vendidos no local ou fora dele, dependendo de regulação dos governos locais e das metas da horta, enquanto um esforço coletivo. Já as hortas institucionais são hortas de alimentos ou pomares de pequenos a grandes, localizadas em propriedades institucionais públicas ou privadas (escolas, hospitais, organizações religiosas, locais de trabalho) em áreas residenciais, comerciais ou mistas, conduzidas por uma organização ou negócio.

O processo de jardinagem é utilizado para propósitos educacionais, terapêuticos e serviços comunitários - incluindo, mas não limitado a educação nutricional, consciência ambiental e orientação religiosa. Os produtos são utilizados para doação ou consumo. Dependendo da regulação dos governos locais, eles também podem ser vendidos no local ou fora dele, para especificamente para apoiar financeiramente as atividades de jardinagem.

Na dimensão política social, com finalidade não comercial, horta doméstica e horta comunitária compreenderam outras modalidades de agricultura urbana familiar. Em ambos os casos, a produção é voltada ao autoconsumo, a trocas e eventualmente à venda. Como elas visam prover a subsistência, a economia de gastos propiciada pelo autoconsumo e trocas devem impactar a renda familiar. A modalidade horta comunitária envolve programas e projetos de apoio a indivíduos e famílias em condições de vulnerabilidade social, na qual a agricultura urbana e periurbana é um meio de promover sua inclusão social. Os projetos são promovidos e coordenados por diferentes níveis de governo (PIRES, 2016).

Para analisar a importância e influência da Agricultura Urbana e Periurbana (AUP) no desenvolvimento econômico e social, foram realizados estudos pela Universidade Estadual de Maringá (UEM), liderados por Vicente Chiaramonte Pires, para levantar dados acerca das hortas comunitárias que atuam na Região Metropolitana de Maringá. Trata-se de pesquisa qualitativa, a partir de dados coletados com a aplicação de questionários e abordagem interpretativa.

Os resultados desta pesquisa revelam a importância econômica e social da agricultura urbana em seus mais variados aspectos. Conforme os dados coletados, 73% dos entrevistados são agricultores de hortas comunitárias, 18% de hortas institucionais e 9% de hortas caseiras. Cerca de 90% dos respondentes produzem apenas produtos orgânicos. Da cultura produzida, 61% são produtores de verduras e legumes, 24% produzem ervas medicinais, 6% plantas ornamentais e 9% outros tipos de produtos. Além do consumo dos produtos que plantam, 19% dos agricultores responderam que realizam trocas de produtos entre as famílias, 4% transformam os seus produtos agregando valor, 19% comercializam, 3% prestam serviços de apoio e assessorias técnicas e 5% realizam doações dos seus produtos (PIRES, 2016).

Ainda de acordo com este estudo da UEM, as comercializações dos produtos ocorrem no próprio local da horta em sua maioria, o que é importante para os produtores, pois ficam livres dos custos de remoção e transporte, além do fato dos clientes consumirem os produtos frescos, colhidos na hora da compra. Desta forma, os agricultores ganham poder de mercado na medida em que expandem sua atuação para mercados e restaurantes. Porém, a mudança do foco de atuação da troca para a venda pode ocasionar problemas entre os produtores, pelo fato deles não possuírem habilidades administrativas, além de que com esta mudança o propósito da AUP deixa de ser atendido. Quanto a origem dos recursos, 48% responderam que os insumos e equipamentos são doados pela prefeitura do município, 37% responderam que os investimentos nas hortas são feitos com recursos próprios, 7% disseram receber recursos da UEM, 4% de empresas e outros 4% disseram que emprestam recursos para investir em sua produção.

Outro fator da pesquisa diz respeito à contribuição da AUP na renda do agricultor. Dos agricultores pesquisados, 91% consideram que a renda gerada pela agricultura urbana implica em um complemento de renda, sendo que apenas 9% não possuem outra fonte de renda e tiram da agricultura urbana o sustento da família. A contribuição gerada pela agricultura urbana para

estes agricultores não ultrapassa R$250,00 mensais, o que mostra que na região metropolitana de Maringá essa atividade ainda se mostra bastante principiante e que com o apoio do Ceraup e Prefeituras envolvidas, a agricultura urbana ainda pode se expandir em volume inimaginável. Mesmo ainda recebendo pouco com a atividade de agricultura urbana, os agricultores, em sua maioria (45%), são seus próprios financiadores, 21% disseram ser financiados pela prefeitura, 15% por empresas parceiras, 9% por ONGs e 9% pelo governo estadual/federal. Em termos de renda, verifica-se que há um aumento médio de R$ 234,67 na renda familiar, o que resulta num aumento agregado das famílias pesquisadas em torno de R$ 118.037,33 por mês (PIRES, 2016).

Com relação aos benefícios da AUP, a pesquisa revela que o principal benefício é a contribuição para a alimentação saudável da família (26%), seguido pelo envolvimento e interação da comunidade (23%), contribuição para a oferta de alimentos na cidade (12%), adição de valor ao produto (11%), emprego seguro aos envolvidos (6%), geração e distribuição de renda (5%) e participação de vendas no mercado (4%). Adicional aos elencados para respostas, muitos responderam que o envolvimento com a agricultura urbana muda a vida de uma pessoa, que o contato com a terra e o plantio alivia o stress da vida corrida do dia a dia.

Além disso, houve um relato dizendo que muitos dos envolvidos nas hortas são aposentados e usam esses projetos como um motivo para continuar ativos, ocupar a mente e não cair em depressão, sem contar a renda extra. Outro relato é de um agricultor que não tira renda extra nenhuma em sua atividade, planta para consumir e doar, e usa sua horta como processo terapêutico, que ajudou inclusive na redução de medicações para depressão. Assim, pode-se auferir que a agricultura urbana contribui para a vida dos agricultores tanto de forma social como econômica, pois além de agregar renda e disponibilidade financeira, agrega alimentação saudável, qualidade de vida, envolvimento e interação entre as pessoas em torno de um propósito comum, conceito principal de capital social e empoderamento (PIRES, 2016).

Outros estudos realizados indicam que o interesse dos brasileiros em jardinagem e horta urbana aumentaram após a pandemia de Covid-19. Este fato indica que formas mais sustentáveis de produção e consumo de alimentos vem despertando o interesse dos moradores das cidades, o que pode impulsionar ainda mais a agricultura urbana no ambiente familiar e comunitário (RIBEIRO, 2020).

Porém, o cultivo nas cidades possui barreiras de adoção relacionadas com o tempo, espaço, custo e conhecimento da população para manter hortas domésticas ou comunitárias nos espaços urbanos. A partir dessa dificuldade, surge a motivação do projeto, que é usar tecnologia e automação para diminuir a barreira de entrada da agricultura urbana de porte doméstico, permitindo com que mais pessoas consigam plantar parte de seu consumo alimentício em casa sem que essa atividade demande tempo e conhecimento.

Portanto, diferentemente das fazendas verticais, onde os recursos alocados visam a automação da produção de alimentos em um nível comercial de alta performance, com a existência de grandes investimentos que viabilizam a sofisticação de sua infraestrutura tecnológica, nas hortas domésticas e comunitárias o cenário é distinto, já que seu maior objetivo é a cultura de subsistência, engajamento social e familiar, alimentação saudável, complemento e distribuição de renda e melhoria na qualidade de vida dos praticantes.

Assim, o escopo deste projeto será pautado pelas demandas da agricultura doméstica, que requer um custo reduzido, agregação de informação acerca do plantio, interfaces para a interação entre os usuários e automação de alguns processos para melhorar a segurança e eficiência do plantio.

Portanto, foram tomadas decisões de projeto alinhadas com o escopo das necessidades, como a escolha de realizar o plantio em solo, que é predominante nesta modalidade, o monitoramento dos parâmetros essenciais para o cultivo, como luminosidade, temperatura, umidade do solo e fertilidade do solo, a automação do processo de iluminação e irrigação e o desenvolvimento de um aplicativo onde os usuários possam monitorar sua horta, realizar automações para facilitar o gerenciamento da mesma e interagir com outros usuários para a formação de comunidades de agricultura urbana.

2.4. IoT e Sistemas Embarcados

Sistemas embarcados são aplicações de propósito específico que são tipicamente programadas em microprocessadores e embutidas no contexto ou objeto que se deseja controlar. Podem realizar tarefas pré-determinadas e receber otimizações em sistema ou em tamanho, estão

presentes em sistemas independentes e em partes de um sistema de escopo maior (EMBARCADOS, 2013).

Funcionam como dispositivos gerenciados microcontroladores, circuitos integrados, FPGA e entre outros, o sistema de processamento aplicado é integrado a componentes dedicados de maneira a compor a interface interna e externa do sistema. A parte de *software*, que é responsável por comandar as ações executadas pelos dispositivos e que é chamada de *firmware*, é alocada na memória *flash* ou na memória ROM e precisa ser projetada de maneira a executar com pouco recurso computacional. Por fim, sua estrutura básica é composta por: sensores, conversores A/D e D/A, atuadores e processador, podendo variar de acordo com o objetivo do sistema.

Atualmente possuem variadas aplicações no mercado e recentemente vêm sendo bastante explorados no contexto de oferecer dispositivos com conectividade, como automação industrial, residencial, agrícola, cuidados de saúde, *wearables*, etc.

Internet of Things (IoT) ou Internet das Coisas é o nome dado a uma rede de objetos que possuem conexão com a internet e entre si e que realizam coleta, processamento e troca de dados. A ideia é fornecer uma conectividade presente a qualquer momento e aumentar a integração entre dispositivos eletrônicos e o cotidiano de uma cidade, por exemplo. Toda essa conexão onipresente se faz possível por conta de aplicações em sistemas embarcados voltados para funções do dia a dia de uma pessoa e não apenas para a indústria e com redes bastante estáveis e com boa velocidade como o 4G e até o 5G (AMAZON 2021a).

Toda essa rede criada fornece maior integração entre dispositivos e pessoas e promete revolucionar o campo das comunicações e da computação. O que se espera como resultado disso são cidades e casas inteligentes, fornecendo automação de algumas atividades e previsão e monitoramento de eventos, entre outras possibilidades.

2.5. Computação em Nuvem

A Computação em Nuvem é uma rede de computadores espalhados por todo o mundo e voltados para aplicações de computação como: serviços, banco de dados, *software*, funções, infraestrutura etc. por meio de servidores alocados ao redor do mundo e por se comunicarem

de maneira uniforme, rápida e segura. Atualmente vem sendo bastante aplicada em modelos de oferecer serviços de variados tipos do campo da computação (*Software as a Service* - SaaS) e são bastante utilizados por empresas que buscam entregar aplicações voltadas para o cliente: aplicativos, sites, jogos etc., se fazem coerentes para esses usos por oferecem: escalabilidade, centralização de dados e ausência de barreiras geográficas. Tem como um de seus princípios cobrar apenas pelos recursos que são utilizados e oferecem recursos flexíveis e menor resiliência a mudanças, por serem serviços bastante mutáveis de modo geral (AMAZON, 2021b).

2.6. Arquitetura *Serverless*

É um modelo de execução em que o provedor de serviços em nuvem (*AWS* e *Google Cloud*, por exemplo) realiza o gerenciamento das execuções de trechos de código (funções de um modo geral) com recursos que serão alocados de maneira dinâmica, e a cobrança do serviço é aplicada apenas pelo que foi utilizado na execução da função. A ativação da execução pode ser realizada de várias maneiras (depende do serviço e provedor a ser utilizado) e geralmente é realizada em um contêiner *stateless*. Dentro dessa arquitetura, o código executado geralmente é uma função, por isso esse modelo é comumente chamado de Funções como Serviço ou *Functions as a Service* (FaaS) (AMAZON, 2021c).

2.6.1. Microsserviços

É praticamente um pré-requisito para a utilização de sistemas *serverless*. Para enviar uma função para o provedor de nuvem é fortemente recomendado que a separação de todo o projeto contido no modelo *serverless* seja realizada em forma de funções e essa separação acaba sendo melhor organizada se feita de maneira a separar cada serviço de acordo com um contexto e escopo específico (TRILLES et al., 2020).

2.6.2. Container Stateless

São contêineres independentes que podem ser executados ou interrompidos a qualquer momento (a depender da função no caso do *serverless*) e que podem ser executados em apenas um nó do *cluster*. Funções executadas nesse tipo de contêiner ficam disponíveis apenas enquanto a

requisição estiver em curso e que após o término, o contêiner será apagado (TRILLES et al., 2020).

2.7. Infraestrutura como Código

Antes de definir o que Infraestrutura como Código é, convém definir o termo *DevOps*. O mercado de Tecnologia da Informação (TI) necessita cada vez mais de agilidade em seus processos. Esta necessidade foi a responsável pela emergência do uso de técnicas e metodologias ágeis, que visam encurtar o ciclo de desenvolvimento de aplicações e relacionar as atividades de desenvolvimento de software com as operações de TI. Esta tendência em usar táticas de engenharia de software para reduzir o espaço, tempo e recursos entre o desenvolvimento de software e as atividades de operação, assim como a distância técnica e organizacional entre esses dois tipos de time de software é conhecida como *DevOps* (ARTAC et al., 2017).

A partir de técnicas e práticas específicas de desenvolvimento de software, como a reutilização de ferramentas padrão como versionamento de código e gerenciamento de revisão de código, aplicadas no contexto de *DevOps*, foi criado o conceito de Infraestrutura como Código (IaC) (ARTAC et al., 2017).

O projeto de infraestrutura de software é a fase do ciclo de vida de um sistema que define e configura a infraestrutura necessária para a operação daquele software. Está tipicamente associado a *scripts* de instalação e configuração que realizam a instanciação e ligação entre máquinas, instalação e configuração de software e *middleware* para estas máquinas e instalação e configuração dos serviços necessários para a operação do software. Para facilitar este processo de projeto, o *DevOps* promove o uso de ferramentas típicas de software, como código fonte, aplicado para o projeto de infraestrutura. Desta forma, o conjunto de *scripts*, código de automação e de configuração, modelos, dependências do projeto e parâmetros de configuração operacional podem ser expressos usando uma mesma linguagem padronizada, para facilitar o gerenciamento da infraestrutura de software. Esta prática é conhecida como Infraestrutura como Código (ARTAC et al., 2017).

Portanto, IaC é um método que visa automatizar a infraestrutura de um projeto do campo da tecnologia. O objetivo é permitir que operação e alteração de uma infraestrutura seja garantida, gerenciada e oferecida sem que seja necessário ter acesso ao hardware (servidor físico, por exemplo) ou a ferramentas e interfaces de configuração. A maneira para prover essa automação é criar uma camada de código entre os recursos de infraestrutura e o projeto. Essa camada fica responsável por prover e implantar a estrutura, e se comunicar com os serviços contidos no projeto. Essa metodologia aplica-se a variados tipos de projeto e é preparada para garantir segurança, qualidade e disponibilidade em seu contexto de uso.

2.8. Conclusão

Neste capítulo, foram apresentados os principais tópicos conceituais que permeiam a motivação, ideação, parametrização, desenvolvimento e implementação deste projeto. A partir do panorama da agricultura urbana em suas diversas vertentes, foi delimitado o escopo de atuação do projeto. Listando as principais ferramentas e tecnologias que auxiliarão em sua implementação, foi possível apresentar uma introdução dos principais conceitos arquitetônicos e metodológicos que serão usados no desenvolvimento da solução proposta. O próximo capítulo será dedicado ao aprofundamento das tecnologias que serão usadas para a composição do sistema objeto deste projeto de conclusão de curso.

3. TECNOLOGIAS UTILIZADAS

Neste capítulo serão apresentadas as principais tecnologias que foram utilizadas neste projeto. Como se trata de uma aplicação típica de IoT, foram necessários recursos de hardware e software. Assim, primeiramente serão descritos os elementos de hardware, assim como os elementos de *firmware* e por fim os elementos de software.

3.1. Hardware

Para realizar a automação de cultivos domésticos, foi necessário um conjunto de dispositivos alocados no ambiente de plantação para realizar a instrumentação, controle e gerenciamento dos principais parâmetros locais da plantação, além dos dispositivos remotos disponibilizados pelo provedor de nuvem para o armazenamento e processamento refinado desses parâmetros. Os componentes de hardware que serão usados na localidade do cultivo serão apresentados abaixo.

3.1.1. Sensores

Os parâmetros de cultivo a serem monitorados são: temperatura do ar, luminosidade, umidade do solo e fertilidade (condutibilidade elétrica) do solo. Para colher esses parâmetros, foi escolhido o sensor Xiaomi Flora, produzido pela empresa chinesa Xiaomi, que realiza a leitura dos quatro parâmetros através de um único dispositivo como pode ser visto na Figura 3.

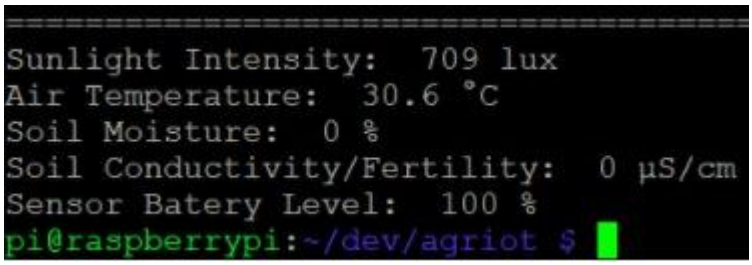
Figura 3 - Módulo de sensores Xiaomi Flora

Fonte: Página de venda dispositivo Xiaomi Flora.

Este sensor é conveniente pois, além de realizar as medições, o dispositivo também agrega os dados lidos e transmite-os através de um módulo de comunicação *Bluetooth Low Energy* (BLE). Além disso, possui uma bateria de lítio própria com uma vida útil de 1,5 a 2 anos, a depender do uso.

A intensidade de luz é medida em lux (fluxo luminoso por unidade de área ou iluminância), a temperatura do ar é medida em graus Celsius, a umidade do solo assim como o restante de carga da bateria são medidos em porcentagem relativa e a condutibilidade do solo é medida em micro Siemens por centímetro (µS/cm) ilustrados na Figura 4.

Figura 4 - Leitura do sensor feita por um Raspberry Pi



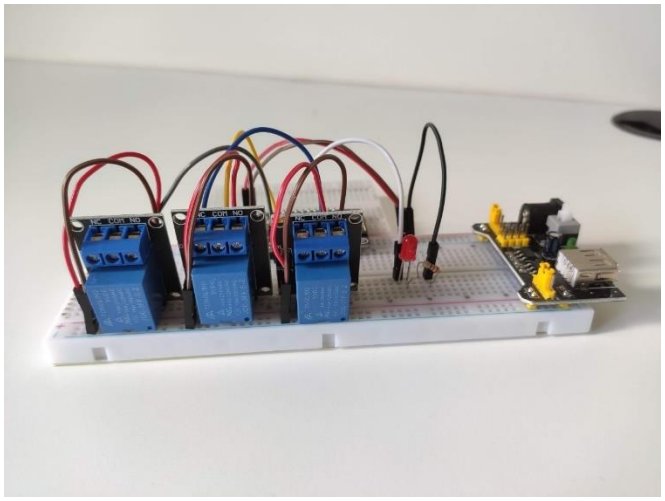
Fonte: Captura de tela do terminal de comando.

Estes parâmetros são utilizados para o monitoramento e controle da plantação, servindo de base para o sistema e o usuário tomarem decisões e ações no gerenciamento do cultivo.

3.1.2. Atuadores

O sistema terá um módulo de atuadores responsável pelo controle da irrigação, iluminação e ventilação do ambiente de cultivo, para casos de miniestufas. Todos os atuadores serão controlados por relés. Assim o sistema terá inicialmente três módulos relés, um controla a bomba de água para irrigação, um controla o painel de iluminação e o final controla o dispositivo de ventilação. A montagem dos relés (componentes azuis) pode ser vista à esquerda na Figura 5.

Figura 5 - Módulo de atuadores, microcontrolador e relés



Fonte: os autores.

Os atuadores são acionados pelo sistema, através da leitura dos parâmetros do sensor e execução da lógica de controle de cada um, de acordo com os parâmetros lidos, ou através de *timers* pré-programados. O relé que controla a irrigação é acionado quando o sensor de umidade do solo indicar uma leitura abaixo do limiar de umidade aceitável. O relé que controla a ventilação será acionado periodicamente, ou se o sensor de temperatura indicar uma leitura acima do limiar de temperatura aceitável. O relé que controla a iluminação será acionado de acordo com o fotoperíodo da plantação, e o sensor de luminosidade serve para checar se a iluminação está funcionando e com a iluminância adequada para a cultura.

3.1.3. *Gateway* de Sensores e Atuadores

O dispositivo Xiaomi Flora já possui internamente conversores analógico-digitais para seus sensores, assim como um *gateway* agregador e transmissor de dados e a comunicação é feita através da tecnologia BLE. Desta forma, não foi preciso implementar a transformação, agregação e transmissão dos dados captados pelo sensor através de um módulo de rede de sensores e conversores A/D, apenas a comunicação por BLE com o dispositivo de borda, através de um *dongle* que suporta este protocolo de comunicação.

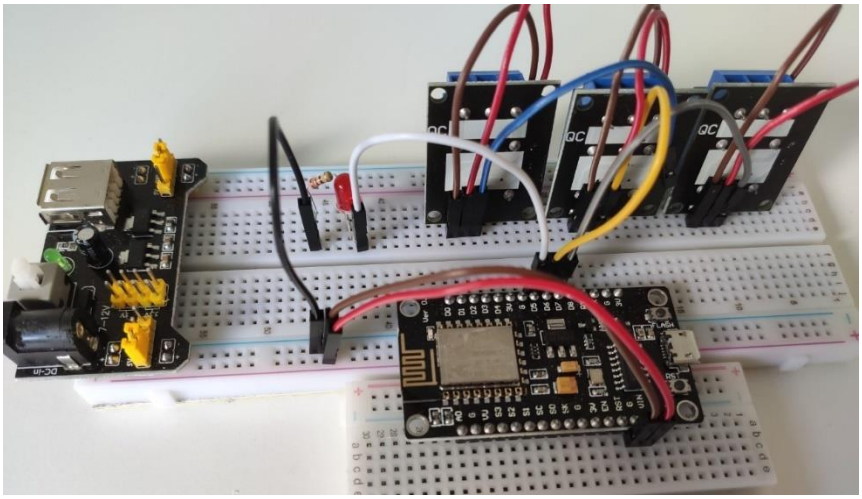
O módulo de atuadores, contudo, não possui nenhum tipo de *gateway* para comunicação com o dispositivo de borda, pois será projetado e implementado inteiramente pelo grupo para o projeto. Os principais requisitos elencados para o módulo de atuadores são a possibilidade de comunicação sem fio, pois o dispositivo de borda não estará necessariamente próximo aos atuadores, e a capacidade de executar algumas lógicas mais simples sem a necessidade de comunicação com o dispositivo de borda ou com a nuvem, para manter funcionalidades mais sensíveis mesmo com a perda de conexão.

Além disso, é interessante que o módulo de atuadores possa ter conexão direta com a nuvem, sem que haja uma dependência com o dispositivo de borda para transmitir e receber informações do provedor de nuvem. Considerando estes requisitos, o dispositivo escolhido como gateway para o módulo de atuadores do projeto é o microcontrolador NodeMCU LoLin ESP8266 V3 (Figura 6). Suas principais especificações de hardware são:

- Processador 32-bit RISC, operando a 80 MHz (com suporte para RTOS);

- 64 kB de memória SRAM;
- 4 MB de memória *flash;*
- Antena PCB (Wi-Fi 802.11 b/g/n);
- 11 pinos I/O digitais e 1 pino analógico (GPIO, UART, SPI e I2C);
- Chip USB Serial CH340.

Figura 6 - Modulo de atuadores com foco no microcontrolador



Fonte: os autores.

Assim, o microcontrolador será responsável por controlar os relés do módulo de atuadores. Por suas especificações, o dispositivo é capaz de executar lógicas locais pelo acionamento de *timers*, que serão usados para o controle de iluminação e de ventilação periodicamente. Além disso, possui uma interface de comunicação Wi-Fi, que será usada para comunicação tanto com o dispositivo de borda quanto com a nuvem, para que o acionamento de atuadores possa ser feito por qualquer camada de arquitetura de redes de aplicação. No protótipo, o módulo de atuadores está sendo alimentado por uma fonte de alimentação para *protoboards*.

3.1.4. Dispositivo de borda

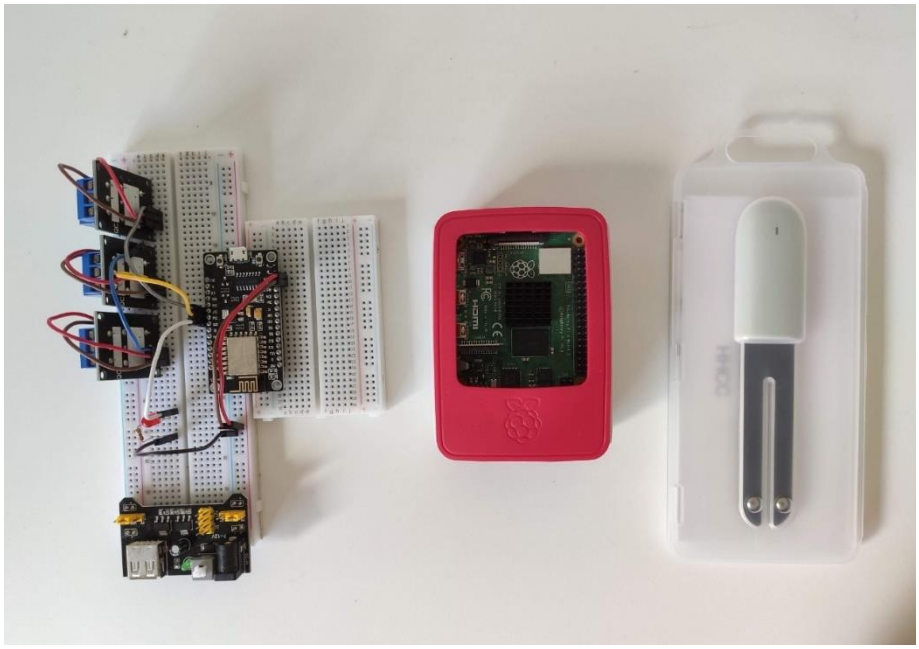
O dispositivo de borda deverá ser capaz de se comunicar com o módulo de sensores através do protocolo BLE e com o módulo de atuadores através do protocolo Wi-Fi. Além disso, deverá ser capaz de se comunicar com o provedor de nuvem tanto para enviar os dados das leituras dos sensores e detalhes de funcionamento dos atuadores, como para receber comandos

e notificações das aplicações hospedadas em nuvem e atuar de acordo com as mensagens recebidas. Deverá também ser capaz de processar parte da carga da aplicação na borda antes de enviar para a nuvem ou agir localmente caso necessário, para economizar recursos de processamento remoto e agilizar alguns processos que podem ser desencadeados localmente.

Portanto, o dispositivo de borda deverá ter compatibilidade com os protocolos de comunicação, um poder de processamento suficiente para lidar com as demandas locais e remotas e compatibilidade com os firmwares do provedor de nuvem. Para atender estes requisitos, será usado um Raspberry Pi como dispositivo de borda (Figura 7 ao meio). Suas principais especificações são (Raspberry Pi 4):

- Processador *Quad core* Cortex-A72 (ARM v8) 64-bit SoC @1,5GHz;
- 2, 4 ou 8GB LPDDR4-3200 SDRAM;
- 2,4GHz e 5,0GHz IEEE 802.11ac *wireless*, Bluetooth 5.0, BLE;
- *Slot* para cartão de memória micros SD para carregamento do sistema operacional e armazenamento interno.

Figura 7 - Sistema de hardware local.



Fonte: os autores.

Com o dispositivo de borda definido, todos os elementos de *hardware* presentes na solução do projeto foram apresentados. A seguir, serão apresentados os softwares que serão utilizados nestes dispositivos e na camada do provedor de nuvem, e a arquitetura de rede por trás da aplicação como um todo.

3.2. Software

Como se trata de um sistema típico de Internet das Coisas, o espectro de *softwares* a serem empregados na solução é bastante amplo. Devem ser definidos os sistemas operacionais e frameworks de desenvolvimento a serem utilizados nos dispositivos locais da aplicação, os serviços do provedor de nuvem que serão implementados nos dispositivos da borda da aplicação e no núcleo da nuvem para a construção de APIs, bancos de dados e ferramentas de análise e monitoramento, as linguagens de programação a serem utilizadas em cada plataforma alvo, as ferramentas de gerenciamento de infraestrutura de software e os serviços para hospedagem, autenticação e desenvolvimento do aplicativo cliente do sistema.

3.2.1. Mongoose OS e Raspberry Pi OS

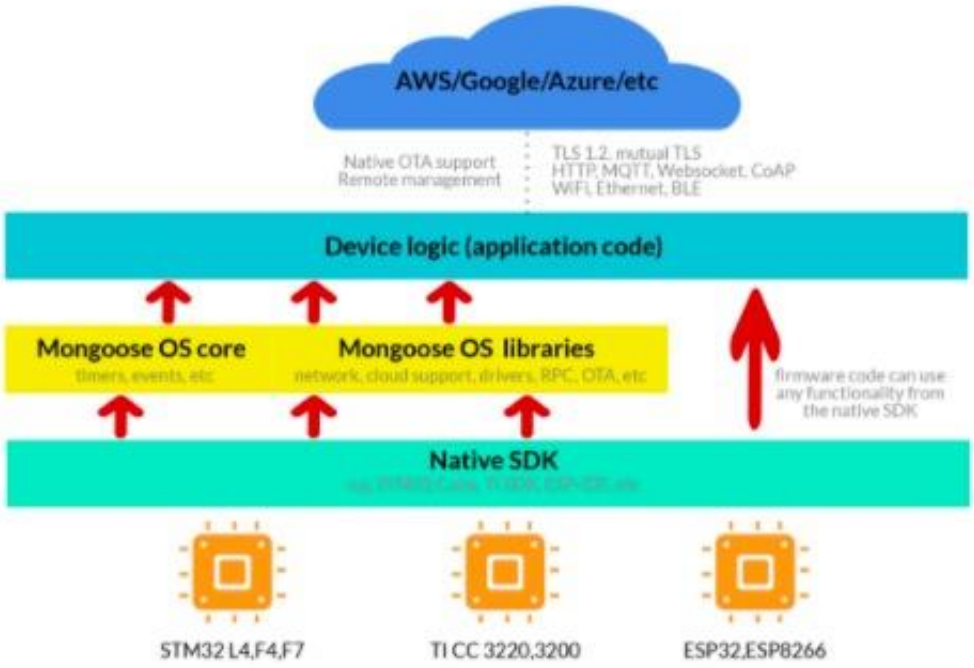
Para gerenciar os recursos do microcontrolador ESP8266 será usado o Mongoose OS, que é um *framework* de desenvolvimento de *firmware* para aplicações IoT, focado em microcontroladores de baixa potência. Seu objetivo é reduzir o tempo de desenvolvimento de *firmware*, disponibilizando um conjunto de componentes reutilizáveis, independente de plataforma. O Mongoose OS foi desenvolvido em cima do sistema operacional FreeRTOS e utiliza em seu núcleo a biblioteca de rede *Mongoose Networking Library* (MONGOOSE, 2021).

Sua arquitetura é composta por 3 componentes principais (Figura 8):

- Uma ferramenta de gerenciamento de dispositivos e de *build* de *firmware*, chamada MOS;
- Um conjunto de ferramentas para o processo de *build*, através de um contêiner Docker que contém o SDK de hardware da plataforma alvo junto com os arquivos da biblioteca Mongoose OS. As configurações de build são editáveis através de um arquivo de configuração MOS.yml, localizado no diretório raiz da aplicação;

- Uma coleção de *apps* e bibliotecas para auxiliar no desenvolvimento do *firmware.*

Figura 8 - Diagrama exemplificando os recursos do Mongoose OS



Fonte: página inicial Mongoose OS.

Desta forma, a biblioteca Mongoose OS possui as ferramentas necessárias para expor o SDK nativo da plataforma alvo ao desenvolvedor, juntamente com um conjunto de bibliotecas proprietário para configurar *timers*, eventos, interfaces de rede, suporte aos principais provedores de nuvem, *drivers*, *Remote Procedure Calls*, *Over The Air updates* e outras ferramentas de desenvolvimento. Possibilita também o desenvolvimento do código de aplicação em C ou em *javascript* embarcado, ou mJS.

Para o Raspberry Pi será usado o sistema operacional Raspberry Pi OS, que é baseado na distribuição Debian do Linux, oferecido e mantido como sistema operacional oficial da *Raspberry Foundation*, otimizado para o hardware do Raspberry Pi. Possui mais de 35.000 pacotes de software nativos, sendo também um projeto mantido pela comunidade de desenvolvedores de forma ativa, com ênfase em estabilidade, performance e interoperabilidade entre os diversos recursos e aplicações do sistema (RASPBERRY, 2021).

3.2.2. *Serverless Framework* e *Cloudformation*

O AWS *Cloudformation* é a ferramenta de provisão de infraestrutura de nuvem através de código para os serviços da plataforma AWS. Fornece uma linguagem descritiva para a configuração de recursos de nuvem através de pilhas, permitindo o gerenciamento de recursos físicos através de estruturas lógicas, de acordo com o escopo da aplicação. Seus benefícios são: automatização de boas práticas, agilidade e escalabilidade para provisionar infraestrutura de *software*, integração com outros serviços da AWS e proximidade entre o código de infraestrutura e o código de aplicação (AMAZON, 2021c).

Já o *Serverless Framework* é um projeto *open source* que disponibiliza um *framework* para gerenciamento de infraestruturas *serverless*, com ferramentas adequadas para agilizar o processo de desenvolvimento de *software* nesse ecossistema (SERVERLESS, 2021). Para gerenciar o ecossistema *serverless* da AWS, o *Serverless Framework* fornece uma linguagem de configuração própria e uma CLI que facilita o tratamento dos recursos da AWS como Lambda, API Gateway e DynamoDB, transpilando (convertendo o código fonte em outro por meio de um processo de compilação) os recursos para o *CloudFormation* para que possam ser provisionados no ambiente de nuvem.

Desta forma, essas duas tecnologias de provisionamento de Infraestrutura como Código serão usadas em conjunto para gerenciar toda a infraestrutura de nuvem da aplicação, desde *softwares* rodando nos dispositivos de borda como o AWS IoT Core e o *Greengrass*, até os que comporão a infraestrutura da camada de nuvem do projeto, como Lambda, API Gateway, DynamoDB e outros. A seguir, serão apresentados com mais detalhes os *softwares* da AWS a serem utilizados no projeto.

3.2.3. AWS

A AWS é a maior plataforma de provisionamento de infraestrutura e *software* de nuvem do mundo, sendo pioneira em muitos conceitos e arquiteturas utilizadas no ambiente de nuvem pública. Fornece soluções para os diversos paradigmas de desenvolvimento em nuvem, como *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), *Software as a Service* (SaaS),

Container as a Service (CaaS) e *Function as a Service* (FaaS). O projeto focará no paradigma de *Funcion as a Service* (AMAZON, 2021d).

A camada de nuvem da aplicação será desenvolvida no paradigma arquitetural *Serverless*. Este paradigma permite um desenvolvimento mais ágil, uma vez que grande parte da infraestrutura por trás do código é gerenciada pelo provedor de nuvem. É fortemente baseado em contêineres efêmeros sem estado, gerenciados pelo provedor de nuvem no paradigma FaaS, fornecendo escalabilidade automática, alta disponibilidade, arquitetura compatível com desenvolvimento em microsserviços e um modelo de cobrança sob demanda.

Os principais serviços de computação sem servidor da AWS que serão utilizados no projeto serão apresentados a seguir:

3.2.3.1. AWS IoT e AWS *Greengrass*

O AWS IoT e AWS *Greengrass* são serviços que oferecem um *framework* de ferramentas que permitem levar os recursos da AWS para os dispositivos de borda (AMAZON, 2021e). Garantem que o dispositivo possa tratar dados localmente e conte com armazenamento em nuvem. Também permitem aos dispositivos de borda executarem funções Lambda.

Suas principais vantagens são a rápida resposta para eventos locais, operação constante mesmo com conexão intermitente com a *internet* e redução de custos de operação de aplicações IoT.

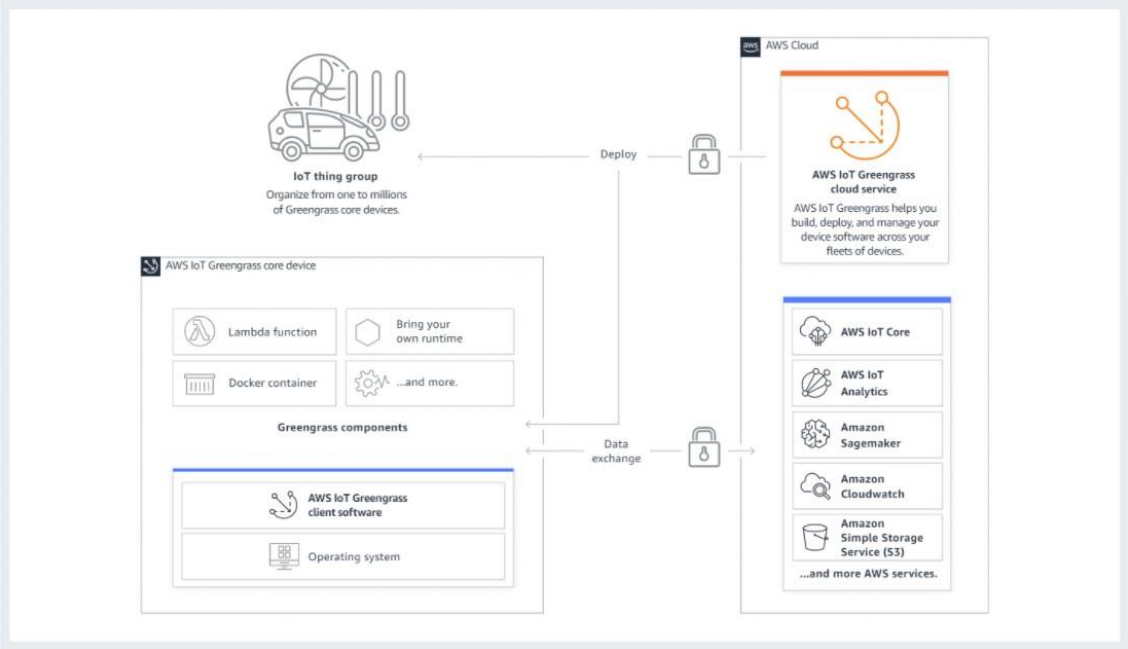
Os principais conceitos por trás do AWS IoT e Greengrass são (Figura 9):

- Grupo: Agrupamento de recursos, dispositivos e regras que definem o escopo do sistema embarcado. Sua principal função é organizar os diversos componentes de hardware e software presentes no ambiente local do sistema embarcado, assim como a configuração deles;
- Lambdas: Funções semelhantes às executadas em nuvem, porém podendo atuar no dispositivo de borda;
- Dispositivo de Núcleo: Trata-se do dispositivo de borda, ou nó principal do grupo. Tipicamente o dispositivo com maior poder computacional, é o hospedeiro do software de núcleo

do *Greengrass* e responsável pelo gerenciamento dos demais dispositivos e componentes, assim como pela comunicação com a nuvem;

- Componentes: Aplicações de software usadas para a implementação de funcionalidades no dispositivo de núcleo, infraestrutura de mensageria entre os componentes e outras utilidades para auxílio no desenvolvimento local do sistema. A AWS fornece diversos componentes públicos, como uma interface gráfica e um CLI para o software do *Greengrass*, *brokers* MQTT para serem hospedados no dispositivo de núcleo, componentes de sincronização entre os *brokers* MQTT locais e os da nuvem, componentes para auxiliar na autenticação das requisições dentre outros. Além disso, existe uma interface para a criação de componentes locais, que podem executar lógicas arbitrárias implementadas pelos desenvolvedores;
- **Dispositivos**: outros componentes de *hardware* que interagem com o dispositivo de núcleo e podem estar no mesmo grupo, responsáveis por diversas tarefas dentro do escopo da aplicação embarcada;
- ***Deployments***: ferramentas de controle de versionamento do grupo e seus componentes. Sempre que se deseja realizar uma mudança em algum componente ou configuração de um grupo, realiza-se uma revisão dos parâmetros e a liberação de uma nova versão através de um *deployment*, que é responsável pela rotina de atualização e versionamento do grupo;
- **MQTT *Brokers*:** o protocolo MQTT define dois tipos de entidades na rede: um *message broker* e inúmeros clientes. O broker é um servidor que recebe todas as mensagens dos clientes e, em seguida, roteia essas mensagens para os clientes de destino relevantes. Um cliente é qualquer coisa que possa interagir com o *broker* e receber mensagens. Um cliente pode ser um sensor de IoT em campo ou um aplicativo em um data center que processa dados de IoT. No contexto da AWS, os fluxos de mensagens MQTT são separados em tópicos, o que auxilia na organização da comunicação entre os diversos componentes;
- ***Rules***: as regras no AWS IoT permitem que os dispositivos interajam com os demais serviços da AWS. As regras são analisadas e as ações são realizadas com base no fluxo de tópicos do MQTT.

Figura 9 - Diagrama exemplificando todos os recursos do AWS IoT *Greengrass*



Fonte: página inicial Greengrass IoT.

É possível instalar o SDK do AWS IoT *Greengrass* para configurar um Raspberry Pi como dispositivo de núcleo do *Greengrass*, fornecendo ferramentas para leitura dos sensores, pré-processamento dos dados, comunicação com a nuvem e orquestração dos atuadores. A placa ESP8266 também pode ser conectada à nuvem através do SDK do AWS IoT, permitindo a comunicação direta entre a nuvem e o módulo de atuadores. A comunicação entre a nuvem, o dispositivo de borda e o módulo de atuadores será feita através de tópicos MQTT, que são gerenciados também pelo SDK do *Greengrass*.

3.2.3.2. AWS Lambda

O AWS Lambda é um serviço de computação sem servidor que permite a execução de código sem o provisionamento ou gerenciamento de servidores. Usa o conceito de computação efêmera para provisionar contêineres que executam código para praticamente qualquer tipo de aplicação ou serviço de *backend*, tudo sem precisar de administração. Basta fazer o upload do código como um arquivo ZIP ou imagem de contêiner, e o Lambda aloca de maneira automática

e precisa o poder de execução de computação e executa o código com base na solicitação ou evento de entrada, para qualquer dimensão de tráfego (AMAZON, 2021f).

O Lambda é capaz de se comunicar com vários serviços da AWS e é a peça fundamental no paradigma de *Funcion as a Service* de seu ecossistema *serverless*. Possui suporte nativo a várias linguagens de programação através do SDK da AWS, mas no projeto serão usados principalmente os tempos de execução do Python e Node.js. Os lambdas são usados principalmente para a construção de APIs de acesso ao banco de dados, e podem ser agrupados logicamente em serviços, formando em seu conjunto uma arquitetura de microsserviços altamente distribuída por contexto e responsabilidade.

3.2.3.3. AWS API Gateway

O Amazon API Gateway é um serviço gerenciado que permite que desenvolvedores criem, publiquem, mantenham, monitorem e protejam APIs em qualquer escala. APIs agem como a "porta de entrada" para aplicativos acessarem dados, lógica de negócios ou funcionalidade de serviços de *backend*. Usando o API *Gateway*, é possível criar APIs RESTful e APIs WebSocket que habilitam aplicativos de comunicação bidirecionais em tempo real. O API Gateway dá suporte a cargas de trabalho conteinerizadas e sem servidor, além de aplicativos da web (AMAZON, 2021g).

O API *Gateway* administra todas as tarefas envolvidas no recebimento e processamento de até centenas de milhares de chamadas de API simultâneas, inclusive gerenciamento de tráfego, suporte de CORS, controle de autorização e acesso com fluxo controlado e monitoramento e gerenciamento de versões de API. A precificação é feita através das chamadas de API recebidas e pela quantidade transferida de dados de saída. Além disso, com o modelo de definição de preço em camadas do API *Gateway*, é possível reduzir os custos à medida que o uso da API é escalado.

Na aplicação, o API *Gateway* será responsável por receber as requisições da aplicação cliente, checar a autenticação e autorização e engatilhar lambdas responsáveis pela execução da lógica da requisição, seja ela um acesso ao banco de dados ou a algum dispositivo de borda da aplicação.

3.2.3.4. AWS DynamoDB

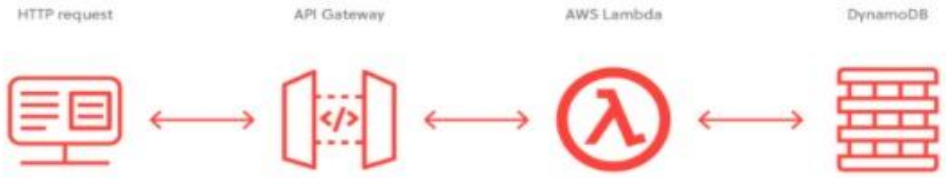
O Amazon DynamoDB é um banco de dados não relacional de valores-chave e documentos que oferece desempenho em milissegundos de um dígito em qualquer escala. É um banco de dados totalmente gerenciado, multirregional, multiativo e durável com segurança, *backup* e restauração integrados e armazenamento em cache na memória para aplicativos em escala de *Internet*. O DynamoDB pode processar mais de 10 trilhões de solicitações por dia e comportar picos de mais de 20 milhões de solicitações por segundo. Possui uma arquitetura compatível com o ecossistema serverless e com infraestrutura como código. A conexão é feita através do protocolo HTTP e o permissionamento de acesso pode ser bastante modularizado.

Caracteriza-se por uma estrutura desnormalizada e um *schema* fraco, sendo mais adequado para aplicações de processamento de transações online (OLTP). Internamente, é organizado em partições, que são a unidade fundamental de armazenamento das tabelas do banco. Através das partições, o DynamoDB provisiona uma arquitetura desnormalizada e permite *node sharding* e escalabilidade horizontal a níveis de trilhões de solicitações por dia sem perder performance (AMAZON, 2021h).

Oferece uma API que permite operar em itens, partições ou na tabela inteira de uma instância em execução. Na aplicação, será usado como armazenamento persistente e de larga escala, servindo para guardar as informações referentes aos usuários e às plantações.

Após a apresentação das principais tecnologias de nuvem que serão utilizadas, pode-se dizer que a camada de nuvem da aplicação será composta primordialmente por essas 4 tecnologias, formando uma arquitetura *serverless* altamente distribuída e integrada, permitindo a comunicação eficiente entre os dispositivos de borda, a infraestrutura de cloud e a aplicação cliente como visto na Figura 10.

Figura 10 - Diagrama de alto nível dos principais recuros AWS



Fonte: os autores.

3.2.4. Node.js

NodeJS é um software de código aberto que foi desenvolvido para ser um ambiente de execução para a linguagem JavaScript. Tem como objetivo desvincular a execução de código de um browser e fornece escalabilidade, flexibilidade e baixo custo. Atualmente é amplamente utilizado no mercado de desenvolvimento de software e é tido como a escolha natural para utilizar JavaScript em aplicações voltadas para clientes (NODEJS, 2021).

3.2.5. Google *Firebase*

O Google Cloud Platform é uma suíte de computação em nuvem oferecida pelo Google, funcionando na mesma infraestrutura que a empresa usa para seus produtos dirigidos aos usuários, dentre eles o Buscador Google e o Youtube. Assim como a AWS, possui vários serviços de nuvem disponíveis, mas este projeto se restringirá ao uso do Google *Firebase*.

O Google *Firebase* é uma plataforma para o desenvolvimento de aplicações *mobile* e *web*. Possui ferramentas para construção de código, desenvolvimento de processos de negócio e manutenção de qualidade. Apresenta seus recursos através de um SDK modularizado, permitindo a importação e uso de suas ferramentas de forma independente. O *Firebase* será utilizado para hospedar a aplicação cliente do projeto. Desta forma, os principais recursos que serão aprimorados no projeto são o módulo de Hospedagem e o módulo de Autenticação de usuários (GOOGLE, 2021a).

O módulo de Hospedagem do *Firebase* oferece hospedagem rápida e segura para uma aplicação *Web*, conteúdo estático e dinâmico e microsserviços. Ele permite exibir conteúdo por

meio de uma conexão segura, enviar conteúdo rapidamente, emular e até compartilhar alterações antes de publicar e implantar novas versões com um comando. O aplicativo cliente para o monitoramento da plantação será desenvolvido em Flutter e hospedado através do módulo de hospedagem do *Firebase*.

O módulo de Autenticação do *Firebase* fornece serviços de *backend*, SDKs fáceis de usar e bibliotecas de IU prontas para autenticar usuários no aplicativo. Ele oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook e Twitter, entre outros.

O *Firebase Authentication* é estreitamente integrado a outros serviços do *Firebase* e aproveita os padrões do setor, como OAuth 2.0 e OpenID Connect, para que possa ser facilmente integrado a um *backend* personalizado. Na aplicação, o módulo de autenticação do *Firebase* será utilizado em conjunto com um Lambda para realizar a autenticação do usuário quando ele faz uma requisição ao API Gateway por meio do aplicativo cliente.

3.2.6. Flutter

Flutter é um arcabouço desenvolvido pela Google para auxiliar o desenvolvimento de aplicações mobile com uso da linguagem Dart. Possui melhor desempenho se comparado a outras possibilidades para desenvolvimento *mobile*, pois o código desenvolvido é compilado para a linguagem base do dispositivo (Android ou iOS), isso permite que funcionalidades nativas do dispositivo sejam utilizadas de maneira menos custosa. Atualmente é bastante utilizada no mercado, não só por seu desempenho, mas também por ser reconhecida por possuir baixa curva de aprendizagem (GOOGLE, 2021b).

4. METODOLOGIA DE TRABALHO

A metodologia a ser empregada é baseada no conceito de prototipação e de espiral evolucionária.

4.1. Pesquisa inicial

A concepção do tema deste projeto vem desde o trabalho final da disciplina Laboratório Digital II, onde foi desenvolvido um sistema que visava automatizar irrigação em plantações e realizar monitoramento de luminosidade. Partindo da premissa desse trabalho, foram realizadas pesquisas nos temas de agricultura urbana, automação e soluções IoT para plantações residenciais, os resultados mostraram que atualmente existem poucas aplicações que buscam atender as necessidades de controle e monitoramento de uma horta doméstica, a maioria das propostas existentes são focadas apenas em monitoramento ou automação e poucas oferecem integração com nuvem. Em outras palavras, a pesquisa revelou que a maioria das aplicações tecnológicas voltadas para agricultura se concentram no agronegócio, tanto no Brasil, quanto mundialmente.

Mesmo com esse cenário de soluções voltadas para grandes produções agrícolas, o resultado da pesquisa trouxe duas empresas localizadas na cidade de São Paulo que realizam agricultura urbana e automatizada, são elas: *Pink Farms* e Fazenda Cubo. A primeira é uma fazenda vertical que aplica hidroponia e automatizações em irrigação e em aplicações de nutrientes para produção de variados tipos de alface e de *microgreens*. A segunda é uma fazenda indoor que também possui automatização em irrigação e tem como foco a produção de hortaliças. Ambas as empresas são voltadas para produção em grande escala (ainda que bem menor se comparada com o agronegócio) e visam comercializar o seu plantio.

Toda a pesquisa mostra que a ideia do projeto ainda é pouco explorada, visto que ainda há poucas soluções completas voltadas para a agricultura urbana e periurbana e as aplicações existentes buscam vender o resultado do plantio e a ideia deste projeto é oferecer ferramentas para que uma pessoa possa obter bons resultados com seu próprio plantio.

4.2. Estudo sobre viabilidade de projeto

O projeto se divide em três principais partes: sensores e atuadores, comunicação de dados com um servidor na nuvem e interface móvel. Para analisar a viabilidade do projeto como um

todo é preciso entender essas partes e como será a interação entre elas. Os sensores e atuadores são os componentes que ficam aplicados na plantação, têm como principal função receber e enviar dados e realizar comandos de controle (regar e fertilizar, por exemplo). A parte de comunicação fica responsável por receber e enviar dados dos sensores e atuadores e da interface mobile. A parte de interface com usuário é onde os dados da planta serão visualizados e onde pode-se ser criada alguma ação de controle para a planta.

Por propor um protótipo de hardware e um de software para a interação direta com o usuário final, o projeto requer considerações quanto a montagem e usabilidade. Os sensores e atuadores precisam de fácil entendimento para a aplicação na plantação e devem exigir pouca manutenção, além de serem capazes de executar variadas funções por um longo período. A interface mobile deve prover uma experiência de uso bastante amigável e simples para o contexto da aplicação.

O projeto se mostra viável, pois já existem soluções para as três principais partes citadas e a maioria dessas soluções já possuem possibilidades de se comunicarem de maneira independente (no caso de sensores a atuadores, por exemplo). A viabilidade se prova também com o fato de já existirem no mercado variadas soluções de monitoramento e atuação para plantações no agronegócio e o projeto, dentre outras funcionalidades, visa trazer isso para um escopo menor.

4.3. Elaboração de proposta

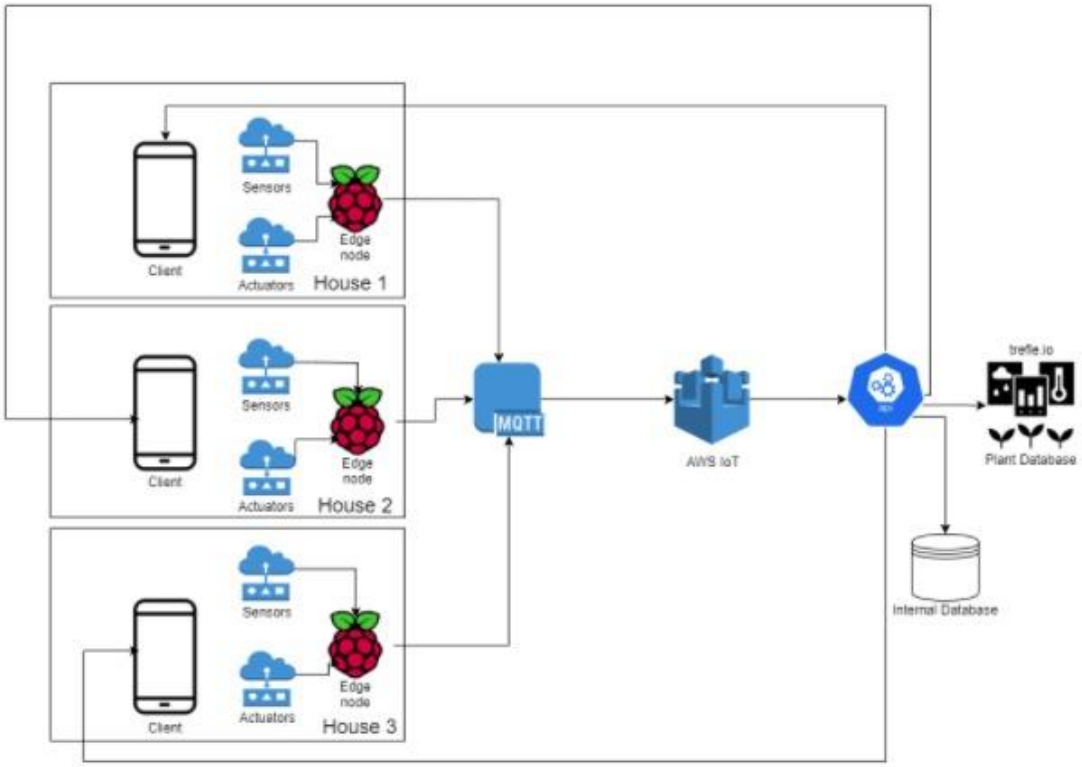
A ideia do projeto consiste em levantar os principais requisitos de uma plantação doméstica num ambiente urbano, descobrir quais são os pontos de maior dificuldade na manutenção de uma horta nesta vertente agrícola e procurar soluções que automatizam este processo, de preferência com baixo custo e sem dependência de interferência do usuário final. Para isso, foi elencado um conjunto de sensores que fazem o monitoramento dos principais parâmetros que influenciam no crescimento saudável de uma planta, assim como possíveis atuadores que farão a manutenção desses parâmetros em níveis aceitáveis para o crescimento saudável da cultura.

Com estes requisitos levantados, foi feita uma aplicação que coleta os dados da plantação alvo, compara os valores lidos com os parâmetros recomendáveis para aquela cultura e aciona os atuadores para aproximarem os parâmetros lidos dos recomendáveis, caso necessário.

Como o escopo do projeto é a automação de hortas domésticas, optou-se por utilizar o método de cultivo em solo, pois é mais comum neste ambiente do que o cultivo hidropônico. O cultivo em solo é o tipo mais tradicional de plantio, usando-se terra como substrato para o desenvolvimento da planta. Para o crescimento saudável da cultura, devem ser controlados parâmetros do solo como umidade, pH, adubos e fertilizantes (Macronutrientes e Micronutrientes). Além disso, outros parâmetros não relacionados ao solo devem ser monitorados, como a luminosidade e temperatura do ambiente.

Por fim, foi desenvolvida uma aplicação cliente para fazer a interface entre o usuário e o sistema. A ideia é construir uma interface simples e informativa, que possua uma camada de autenticação para a segurança e privacidade dos usuários e apresente informações relevantes acerca da cultura que o usuário está cultivando e os principais parâmetros controlados pelo sistema. Um primeiro esboço da arquitetura pode ser visto na Figura 11.

Figura 11 - Diagrama de alto nível representando os principais elementos do sistema



Fonte: os autores.

4.4. Elaboração de monografia

A documentação de todo projeto será realizada de maneira contínua ao desenvolvimento dos protótipos de *hardware* e *software*, além das partes tradicionais exigidas de uma monografia, foi necessário documentar toda a parte de prototipação de ambos os casos, processo de codificação e os conceitos de arquitetura utilizados.

O código em si pode ser considerado um documento do projeto, então utilizou-se uma ferramenta de gerenciamento de código Git, para garantir a qualidade, funcionamento e separação do material desenvolvido, sendo o Github o mais popular sistema de controle de versões do mercado.

5. ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Os requisitos de sistema serão especificados usando como base o modelo proposto pelo *Reference Model of Open Distributed Processing* (RM-ODP). Dividindo-se a arquitetura de projeto em cinco pontos de vista diferentes.

5.1. Requisitos Funcionais e não funcionais

No Capítulo 2, foram apresentadas as principais características de um cultivo doméstico, e a partir delas, foram levantados os principais requisitos deste tipo de cultivo. A partir dessas ideias desenvolvidas, foram especificados os requisitos funcionais e não funcionais do sistema.

Os principais requisitos funcionais são:

- Coletar e fornecer informações sobre o cultivo da cultura alvo;
- Realizar medições de temperatura, luminosidade, umidade e fertilidade do solo periodicamente;
- Adicionar atuadores para o controle dos parâmetros da plantação;
- Automatizar o acionamento dos atuadores;
- Disponibilizar dados em tempo real;
- Coletar, processar e agregar dados de medição a longo prazo;
- Criar uma plataforma para que o usuário final possa interagir com sua plantação.

Os principais requisitos não funcionais são:

- Consumo de energia e eficiência energética dos dispositivos;
- Segurança e privacidade dos dados na camada de borda e na nuvem;
- Autenticação e autorização para acesso a dados no servidor;
- Funcionamento do sistema mesmo com conexão intermitente com a internet e o provedor de nuvem;
- Segurança de operação no controle dos atuadores (principalmente irrigação);
- Interface amigável e intuitiva.

5.2. Visão Empresa

Atualmente existem diversas opções de cultivos domésticos, desde as mais simples e arcaicas sem interferências da tecnologia até modelos mais robustos com total controle de operações e monitoramento. Para uma persona mediana com conhecimentos básicos sobre tecnologias usadas no dia a dia, existem diversos ensinamentos seja por cursos, internet ou programas de TV sobre a criação e cultivo de plantas dentro de casa. Há também produtos que facilitam a instalação e montagem com sensoriamentos através de *timers* e dispositivos mais simples de custos relativamente altos reduzindo a um nicho mais entusiasta.

A proposta deste projeto é de criar um protótipo mais abrangente capaz de atrair desde usuários mais simples como também os entusiastas em automação e monitoramento de suas culturas. Uma forma de amplificar o público é tornando os serviços oferecidos pelo projeto independentes. Ou seja, o usuário pode optar por ter todos os componentes de monitoramento, automação e aplicação, ou usar apenas o que lhe for conveniente.

Tendo isso em vista, para se tornar possível, o componente de monitoramento será feito a partir do dispositivo Xiaomi Flora (para próximos passos seria um embarcado de fabricação própria), o componente de automação será feito a partir de atuadores controlados por um Raspberry pi que também receberá dados do Xiaomi Flora, e por fim a aplicação será voltada para que o usuário final tenha acesso aos dados dos seus dispositivos assim como a capacidade de controlá-los, será implementada também uma interface capaz de permitir a interação e compartilhamento de informações entre usuários da plataforma.

5.3. Visão Informação

Para o nível de informação, o sistema será descrito em termos de estruturas de informações, fluxo de informações e restrições relacionados com a manipulação das mesmas, e relacionando-se com o projeto em questão o fluxo se inicia nas medições de temperatura, umidade e fertilidade do solo, os dados serão enviados diretamente para o dispositivo Raspberry na qual serão feitas análises necessárias para possíveis correções dos valores usando os atuadores e paralelamente, as informações processadas serão enviadas para nuvem e serão disponibilizadas para o usuário em uma plataforma *web* e móvel.

5.4. Visão Computação

O nível computacional descreve o sistema de informações em termos de operações e características computacionais do processo de mudança de informações, ou seja, como o sistema irá interagir entre cada um dos módulos.

5.5. Visão Engenharia

A visão de engenharia descreve o sistema de informações em termos de recursos de engenharia para suportar a natureza distribuída do processamento, e como especificado anteriormente, serão utilizados os serviços AWS para comunicação do dispositivo de borda Raspberry Pi 4 será usado o AWS *Greengrass* e AWS IoT, já para o *backend*, as comunicações serão feitas pelo AWS Lambda e API Gateway e os dados serão armazenados no AWS DynamoDB, garantindo estabilidade e responsividade das informações.

5.6. Visão Tecnologia

A elaboração do projeto contará com diversas tecnologias já citadas anteriormente na seção de tecnologias utilizadas, desta forma, especificando a implementação dos componentes nas visões anteriores, as tecnologias e dispositivos utilizados poderão ser separadas da seguinte forma:

5.6.1. Sensores e Atuadores

Para o protótipo serão usados sensores Xiaomi Flora capazes de captar temperatura, umidade e fertilidade do solo. Enquanto os atuadores serão capazes de controlar a irrigação, iluminação e ventilação utilizando relés para cada funcionalidade. E por fim, o *gateway* destas informações com o dispositivo de borda será feito com o microcontrolador ESP8266 utilizando o sistema operacional Mongoose OS.

5.6.2. Dispositivos de borda

O dispositivo de borda utilizado no projeto deverá ter compatibilidade com os protocolos de comunicação e com poder computacional suficiente para as demandas locais imediatas,

e compatibilidade com os firmwares do provedor de nuvem, sendo assim o dispositivo utilizado será o Raspberry Pi 4 com o Raspberry Pi OS.

5.6.3. *Backend*

O *backend* do projeto será estruturado principalmente pelos serviços Serverless Framework e AWS *Cloudformation*, para comunicação com o dispositivo de borda será usado o AWS IoT e AWS *Greengrass* e farão parte da infraestrutura os serviços API Gateway e Lambda. Os dados serão armazenados no banco de dados não relacional DynamoDB e autenticação feita usando os serviços Google *Firebase*.

Toda essa implementação será feita em linguagem de programação Python e NodeJS.

5.6.4. *Frontend*

E por fim, o *frontend* será implementado com o kit de desenvolvimento de interface de usuário, criado pelo Google, chamado Flutter, e hospedado no módulo de hospedagem do Google Firebase.

6. PROJETO E IMPLEMENTAÇÃO

Este capítulo é dedicado a apresentar o planejamento e execução do processo de modelagem, codificação e testes do sistema, desde a parte de prototipação e plano de testes, até a parte de desenvolvimento do código aplicado em hardware e software para compor o sistema como um todo.

6.1. Metodologia de projeto de Sistemas Embarcados

Serão utilizados conceitos levantados em Projeto de Sistemas Embarcados (Cugnasca, 2018) tendo como foco o Projeto baseado em Microcontroladores, devendo-se levar em conta inicialmente as seguintes características:

6.1.1. Aplicação Dedicada

Realiza uma ou poucas funções não alteráveis, mas sim configuráveis.

6.1.2. Interação com o Ambiente

Escolha dos sensores e atuadores para o projeto.

6.1.3. Robustez

O sistema embarcado deve apresentar boa estabilidade de funcionamento, alta disponibilidade e segurança em relação à operação.

6.1.4. Eficiência garantida por Projeto

Deve-se levar em conta características que representam um bom projeto de sistemas embarcados tais como:

6.1.4.1. dimensões reduzidas

O projeto deve ter um tamanho compacto ocupando o menor espaço possível.

6.1.4.2. baixo custo

A proposta do projeto de criar um protótipo de baixo custo capaz de ser mais acessível ao público.

6.1.4.3. projeto eficiente

Sendo voltado para uso doméstico devera-se ponderar a relação custo-benefício para que seja possível criar um protótipo simples e eficiente.

6.1.4.4. interface simples

A interface com o usuário deverá ser informativa e simples para trazer uma segurança ao usuário de que o sistema está operando com êxito.

6.1.5. Etapas do Projeto

São cinco etapas propostas para o desenvolvimento do projeto que usa como base o livro *Computer as Components de* Waine Wolf, são eles:

6.1.5.1. Requisitos

Um levantamento dos Requisitos Funcionais e Não Funcionais.

6.1.5.2. Especificação

Descrição formal e sem ambiguidades.

6.1.5.3. Arquitetura

Destaque dos principais componentes e módulos do sistema por meio de diagramas

6.1.5.4. Componentes

Escolha de componentes que estão comercialmente disponíveis para projeção e implementação respeitando as interfaces de conexão que foram escolhidas e seguidas de testes.

6.1.5.5. integração do sistema

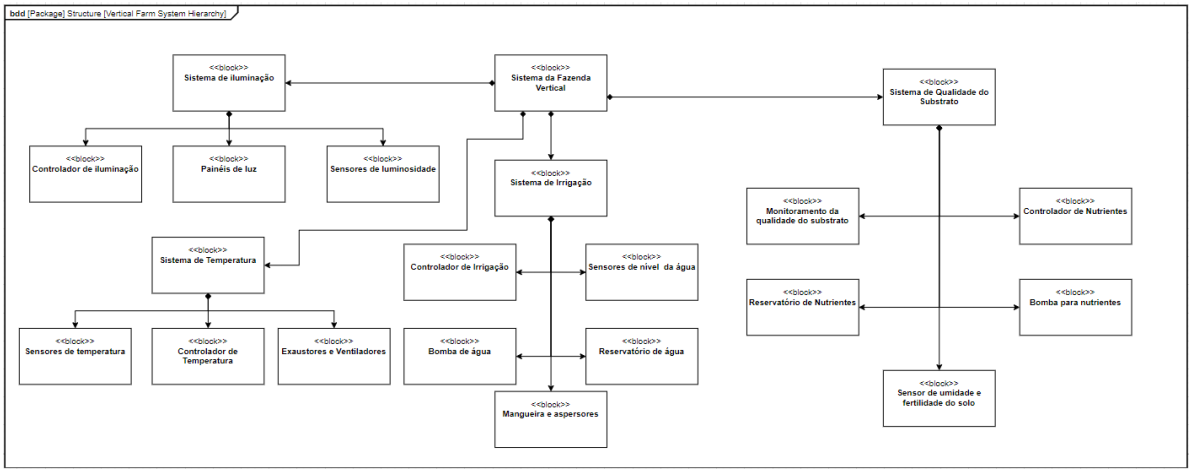
Por fim, a integração é a parte mais complexa pois, além da integração entre os dispositivos embarcados deve-se haver também uma integração com o sistema IoT e posteriormente com a aplicação móvel.

6.2. Hierarquia do Sistema

Na seção de fazendas verticais do capítulo 2 foi apresentado um diagrama de blocos representando a hierarquia do sistema de controle de uma fazenda vertical. A partir deste diagrama, foi elaborado um novo diagrama de blocos para representar o sistema a ser desenvolvido, realizando as devidas simplificações e adequações de escopo. O novo diagrama possui basicamente o mesmo sistema de iluminação, porém com dimensões bastante reduzidas. O sistema de temperatura é composto por sensores de temperatura e, caso necessário, exaustores e ventiladores, que podem ser controlados por um controlador de temperatura.

O sistema de irrigação foi simplificado, uma vez que o método de cultivo será em solo e o gerenciamento de abastecimento de água será mais simples. É composto por um controlador de irrigação, um sensor de nível de água, um reservatório de água, uma bomba de água, mangueira e aspersores. O sistema de qualidade de substrato também foi modificado, uma vez que o cultivo será feito em solo. Ele será composto por um monitor da qualidade do solo e sensores de umidade e fertilidade do solo. Caso o gerenciamento de nutrientes também precise ser automatizado, serão necessários um controlador de nutrientes, um reservatório de nutrientes e uma bomba para os nutrientes como visto na Figura 12.

Figura 12 - Diagrama de blocos SysML representando a hierarquia do sistema

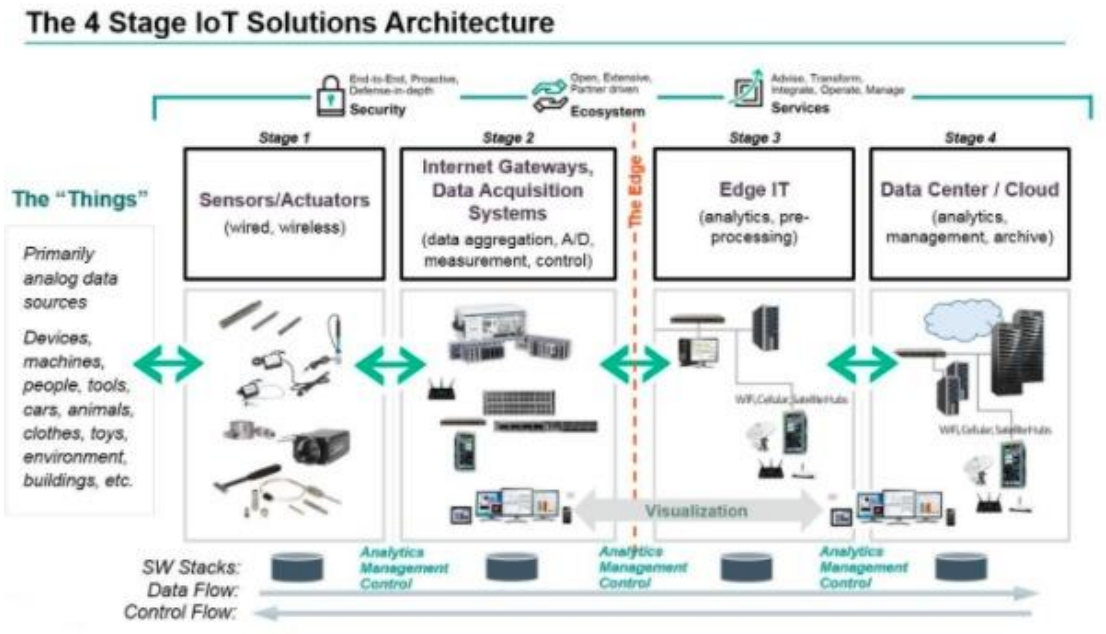


Fonte: os autores.

6.3. Quatro estágios da arquitetura IoT

Como este projeto é uma aplicação típica de internet das coisas, pretende-se abordar a solução do problema seguindo os quatro estágios de uma arquitetura IoT, conforme a Figura 13:

Figura 13 - Diagrama exemplificando os 4 estágios da arquitetura IoT



Fonte: Medium: 4 *Stages of IoT architecture explained in simple words.*

O primeiro estágio consiste em elencar os sensores e atuadores que serão utilizados no projeto. Para o cultivo em solo, os principais parâmetros que devem ser controlados são a luminosidade, a temperatura ambiente, a umidade do solo e a fertilidade do solo, através da condutividade elétrica dele. Para realizar o sensoriamento do projeto, foi escolhido o dispositivo Xiaomi Flora, que reúne num único aparelho sensores de luminosidade, temperatura, umidade e condutividade do solo, uma bateria de lítio e um módulo de comunicação BLE.

Os principais atuadores numa automação de plantio em solo são uma bomba de água para irrigação, uma placa de luz para iluminação e um sistema de exaustão e ventilação, caso o plantio seja feito num ambiente fechado. O escopo inicial do projeto abarcará apenas o sistema de irrigação e o de iluminação, que são os mais essenciais para a saúde da planta. Para o controle dos atuadores, foi desenvolvido um protótipo que envolve 2 relés e um microcontrolador esp8266.

O segundo estágio consiste em elencar os dispositivos de aquisição de dados, responsáveis pela computação de névoa. Com relação aos sensores, o próprio Xiaomi Flora já possui o agregador de dados integrado, que expõe as medidas através do protocolo BLE. Com relação à comunicação com os atuadores, será usado o microcontrolador esp8266, que possui um módulo Wi-Fi para a transmissão e recepção de dados.

Para programar o firmware do módulo de controle dos atuadores, será usado o sistema operacional Mongoose OS. Ele é um *framework* de componentes voltado para o desenvolvimento de aplicações IoT, tem compatibilidade com a placa esp8266 e com a AWS. O Mongoose OS possibilita a programação de componentes locais da placa, como *timers* e controle de pinos, assim como o gerenciamento de servidores MQTT e de comunicação com a nuvem. Desta forma, será possível compor uma solução para os atuadores que pode ser controlada localmente, pelo dispositivo de borda ou pela nuvem. A flexibilidade no tratamento do módulo de atuadores é conveniente, pois se trata do ponto mais sensível do projeto em termos de segurança.

O terceiro estágio corresponde aos componentes e mecanismos de computação de borda. Neste estágio será usado um Raspberry Pi, que orquestrará a comunicação entre o módulo de sensores, o módulo de atuadores e a nuvem. Como se trata de um dispositivo com um poder computacional maior, parte do pré-processamento dos dados e da lógica da aplicação

poderá ser feita de forma local, para que os dados sejam convenientemente estruturados antes de serem enviados à nuvem.

Assim, o Raspberry fará leituras periódicas dos parâmetros dos sensores através de seu módulo BLE, analisará os valores lidos e poderá tomar ações locais de acordo com a lógica programada no dispositivo de borda, como o acionamento do módulo do módulo de irrigação caso a umidade do solo esteja baixa, o acionamento do módulo de iluminação para ligar ou desligar a placa de LED, de acordo com o fotoperíodo da planta e o acionamento dos demais atuadores que possam ser implementados, de acordo com a lógica de cada um. Além disso, o dispositivo pode encaminhar solicitações de notificação ao usuário para a nuvem caso seja necessária a intervenção ou decisão dele.

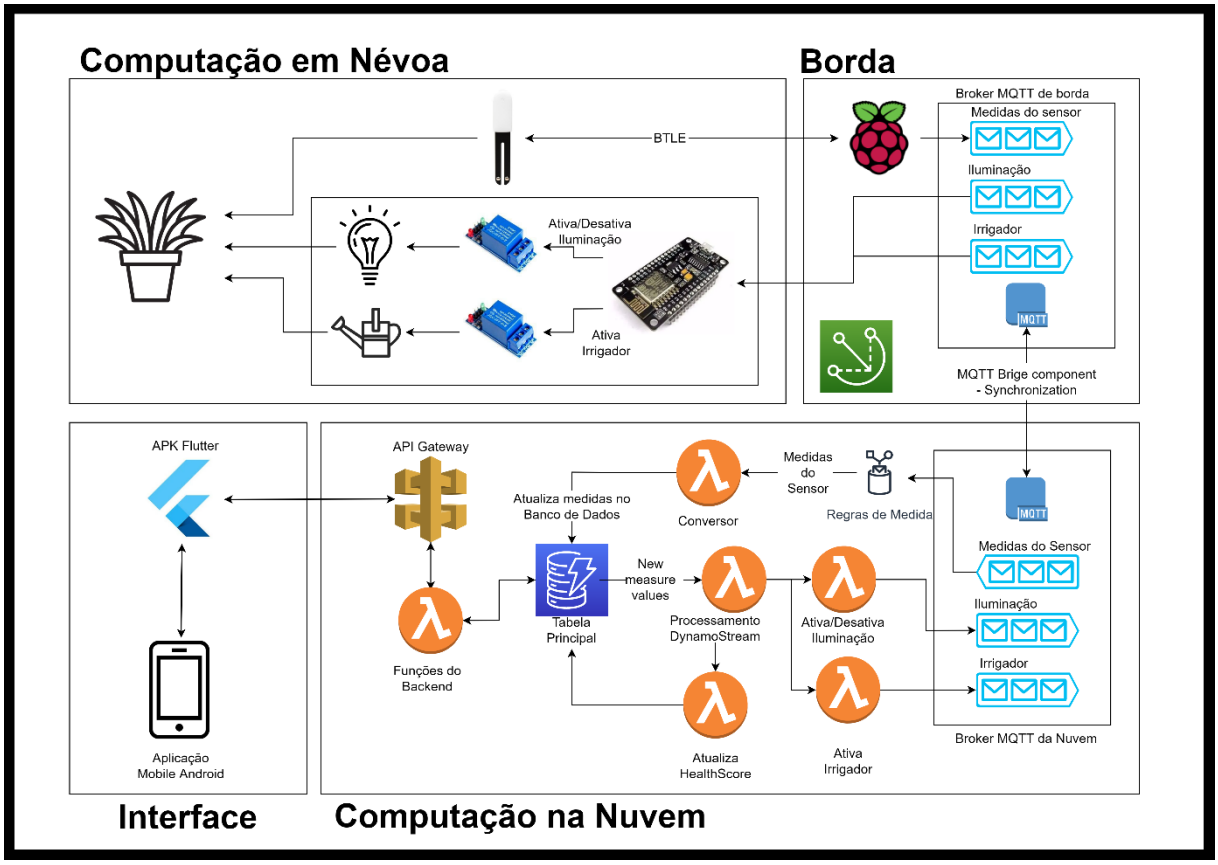
Também é função do dispositivo de borda enviar os dados que deverão ser armazenados em bancos de dados na nuvem, como o histórico de leituras para a geração de *dashboards*, o estado dos atuadores e todas as outras informações locais relevantes. Este dispositivo também poderá receber solicitações do provedor de nuvem, sejam elas dos usuários ou de alguma lógica mais robusta executada remotamente, e deverá validá-las e agir de acordo com elas.

O quarto estágio corresponde ao processamento e análise dos dados na nuvem. Para a computação em nuvem, foi escolhida a AWS, pois possui serviços relacionados com Internet das Coisas como AWS IoT e *Greengrass*, além de outras ferramentas para o desenvolvimento de aplicações *Web*, como o banco de dados DynamoDB, ferramentas para construção de APIs como API *Gateway* e Lambda, além de um ecossistema de softwares e utilitários para o processamento e análise de dados em larga escala.

6.4. Arquitetura do sistema embarcado alvo

Utilizando-se a arquitetura base para aplicações IoT apresentada no tópico anterior, construiu-se um diagrama com os principais componentes presentes na solução proposta, contendo as principais interações entre eles, assim como a separação hierárquica de acordo com sua atuação.

Figura 14 - Diagrama da arquitetura do sistema embarcado



Fonte: os autores.

Na camada descrita como *Fog* estão os sensores, atuadores e interfaces responsáveis pela aquisição dos dados e acionamento dos dispositivos. Na camada descrita como *Edge* está localizado o dispositivo de borda, responsável pelo acionamento da leitura do sensor e encaminhamento dos dados para a nuvem. Além disso, é responsável por rotinas locais de controle dos atuadores e por repassar comandos da nuvem para o módulo de atuadores.

Por fim, a camada *Cloud* é responsável pelo processamento e armazenamento dos dados recebidos do dispositivo de borda, assim como pelo acionamento de rotinas baseadas nas leituras recebidas, como atualização da pontuação de saúde da planta, atualização nos parâmetros dos atuadores no banco de dados e ativação dos módulos de irrigação e iluminação, caso necessário, através de mensagens enviadas ao módulo de atuadores.

Para explicar o funcionamento do sistema, serão usados os conceitos apresentados no capítulo 3, que abordou os componentes de hardware e software utilizados. O fluxo típico da aplicação se dá através da leitura periódica das medidas do sensor, feita pelo dispositivo de borda através do protocolo BLE. Após cada medição, os parâmetros são tratados e enviados a um tópico MQTT local no formato de notação de objetos do JavaScript (JSON). Foi implementado um mecanismo de sincronização entre os *brokers* MQTT da borda e da nuvem, assim é possível mapear mensagens com origem na borda e destino na nuvem e com origem na nuvem e destino na borda.

Desta forma, toda leitura do sensor enviada ao tópico MQTT local será automaticamente sincronizada com um tópico de mesmo nome no *broker* MQTT da nuvem. Uma vez que a leitura do sensor chegue no tópico da nuvem, ele engatilha uma Regra do AWS IoT que encaminha a mensagem para uma função lambda, responsável por validações de dados e armazenamento deles no banco de dados. Uma vez que a nova medida do sensor é escrita no banco de dados, é criado um evento que recebe os parâmetros nova medição e os encaminha para uma função lambda orquestradora dos fluxos dependentes dos valores da leitura, como atualização da pontuação de saúde da planta e acionamento dos módulos de irrigação e iluminação caso necessário, através de outros tópicos MQTT.

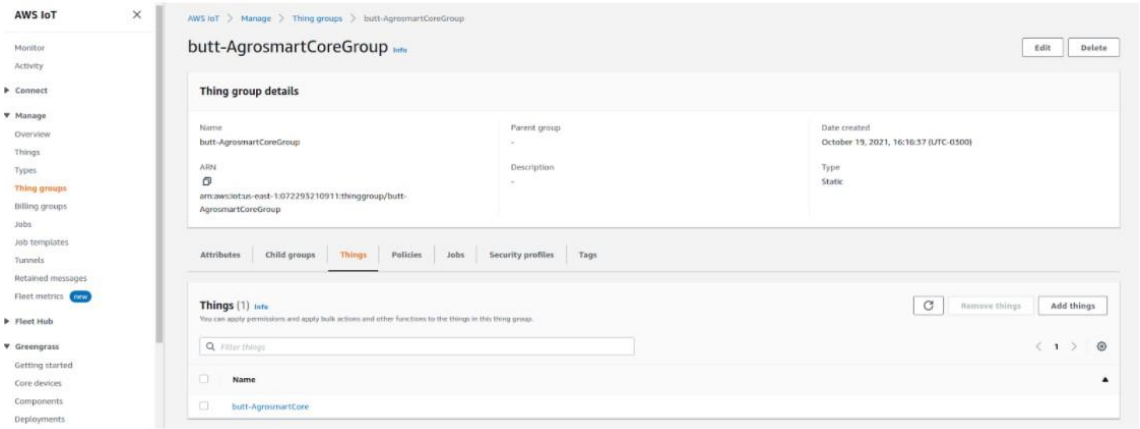
O acionamento dos atuadores é feito pelo caminho inverso dos dados dos sensores, os comandos de controle deles partem de funções da nuvem através dos tópicos MQTT neste ambiente, e é feita uma sincronização entre os tópicos remotos com os tópicos locais do ambiente de borda. Assim, o microcontrolador responsável pelo acionamento dos atuadores atua como subscritor nestes tópicos, realizando as rotinas de irrigação e iluminação conforme a lógica programada localmente. Esta ação fecha o ciclo de monitoramento e controle da aplicação.

No contexto do sistema proposto, a camada de borda da aplicação é representada por um grupo do AWS *Greengrass*. o Raspberry Pi é usado como dispositivo de núcleo e o esp8266 como um outro dispositivo dentro do mesmo grupo. O Raspberry Pi é o hospedeiro do sdk do *Greengrass*, responsável pelo gerenciamento dos componentes de borda. No dispositivo de borda foram instalados os seguintes componentes da AWS:

- aws.greengrass.Cli: Fornece uma interface de linha de comando para lidar com os recursos do dispositivo de borda;
- aws.greengrass.LocalDebugConsole: Fornece uma interface gráfica local para lidar com os recursos do dispositivo de borda;
- aws.greengrass.clientdevices.Auth: Auxilia no gerenciamento de permissões dos dispositivos clientes;
- aws.greengrass.clientdevices.mqtt.Bridge: Responsável pela sincronização dos tópicos de brokers MQTT locais com os tópicos correspondentes no *broker* MQTT da nuvem;
- aws.greengrass.Nucleus: Componente responsável pelo gerenciamento das principais configurações do dispositivo de núcleo;
- aws.greengrass.clientdevices.mqtt.Moquette: Permite a criação de *brokers* MQTT locais no dispositivo de núcleo.

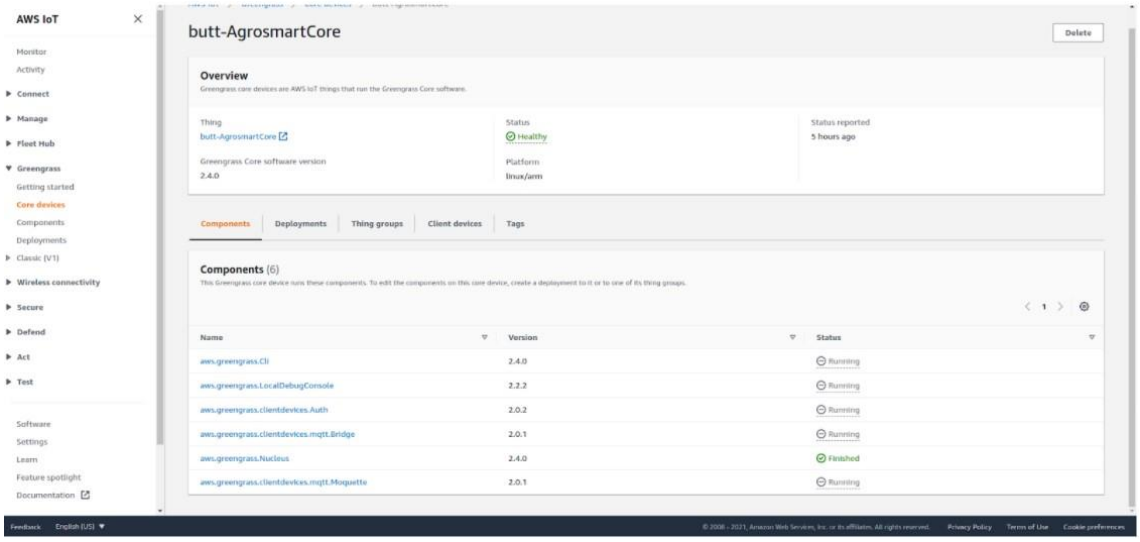
As figuras 15 e 16 mostram o grupo criado, assim como o dispositivo de núcleo e seus componentes através do console da AWS na nuvem.

Figura 15 - Grupo criado para testes



Fonte: captura de tela do console AWS IoT.

Figura 16 - Dispositivo de núcleo criado



Fonte: captura de tela do console AWS IoT.

Utilizando a interface de linha de comando instalada, é possível listar os componentes localmente no dispositivo de núcleo (Figura 17).

Figura 17 - Componentes instalados no dispositivo de núcleo

Fonte: captura de tela do console de comando.

Examinando os componentes da última imagem, percebe-se a presença de um componente novo, chamado *sensor.poller*. Trata-se de um componente local desenvolvido para realizar as leituras do sensor Xiaomi Flora e enviá-las para o tópico MQTT local '*sensor/measures*'.

Ele foi desenvolvido em Python, usando a dependência awsiotsdk para a comunicação com os tópicos MQTT locais e as dependências bluepy e miflora para comunicação BLE com o sensor. O componente é formado por um arquivo de configuração e um script. O arquivo de configuração é responsável por definir o tipo de instância do componente, assim como o ciclo de vida dele, no caso a instalação das dependências e execução do script associado. O script desenvolvido possui o pseudocódigo como visto na Figura 18:

Figura 18 - Pseudo código

```
while True:
    instantiate Mi Flora poller;
    make sensor measures & format to JSON;
    publish the JSON message on 'sensor/measures' MQTT topic;
    sleep(5 min);
```

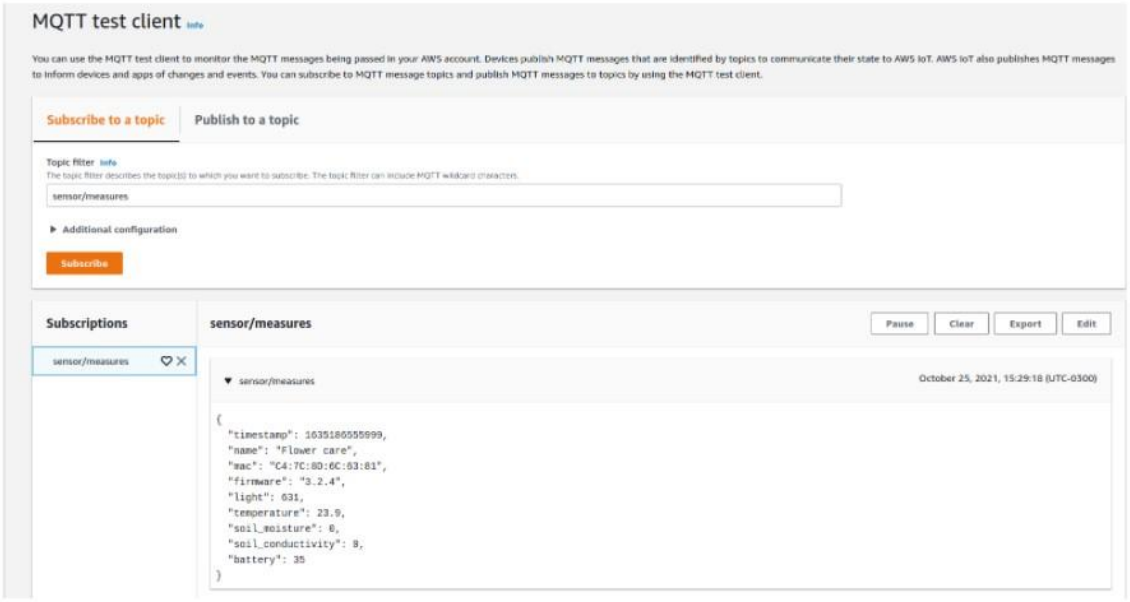
Fonte: os autores.

Outro componente importante para a rotina de leitura do sensor é o MQTT *bridge*, responsável pela sincronização entre os *brokers* MQTT locais e o *broker* da nuvem. A partir dele, é possível mapear todas as mensagens encaminhadas para um tópico local específico para o seu correspondente na nuvem. Por exemplo, toda mensagem enviada localmente no tópico '*sensor/measures*' será automaticamente sincronizada com o tópico '*sensor/measures*' na nuvem. As Figuras 19 e 20 ilustram a execução do componente local que realiza a leitura do sensor e a sincronização das mensagens enviadas ao tópico '*sensor/measures'* com o correspondente na nuvem.

Figura 19 - Logs de execução do componente local

Fonte: captura de tela do console de comando.

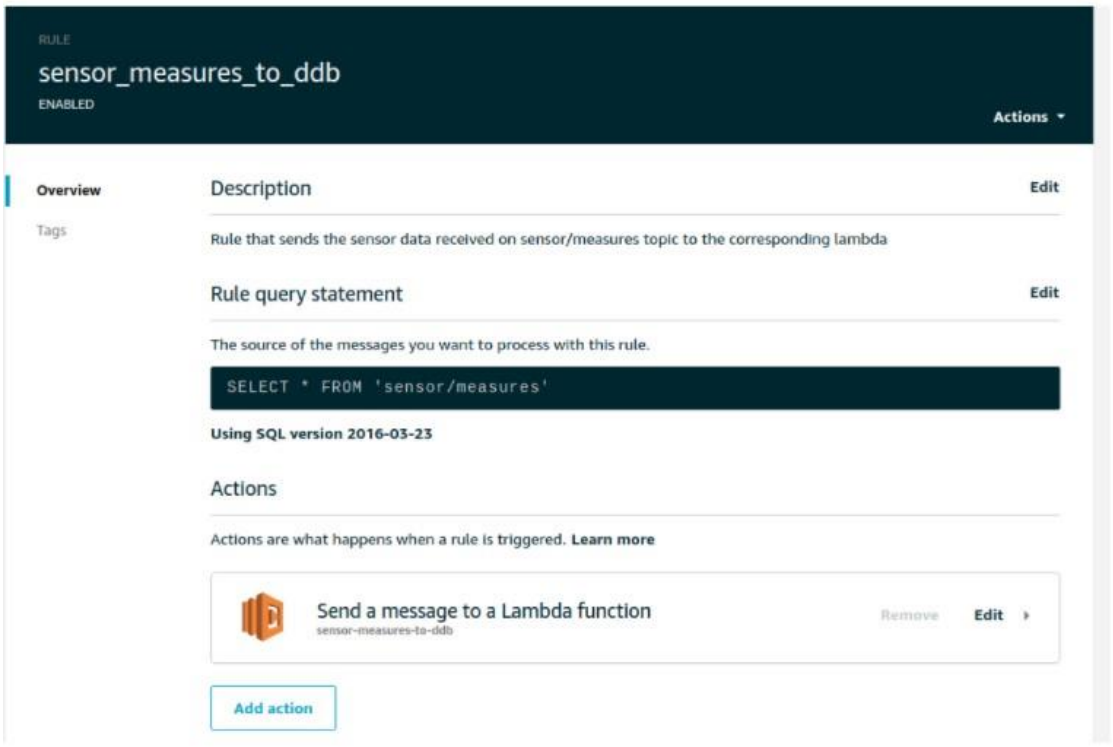
Figura 20 - Cliente teste MQTT na nuvem evidenciando a sincronização entre as mensagens geradas nos tópicos MQTT locais e os tópicos da nuvem.



Fonte: captura de tela do console AWS.

Uma vez que as mensagens chegam no tópico na nuvem, é possível configurar *Rules* (Figuras 21 e 22) para acionar outros serviços da AWS com este evento. No caso das leituras do sensor, sempre que uma mensagem chegar no tópico '*sensor/measures*' na nuvem, este evento acionará uma *Rule*, que por sua vez enviará todo o conteúdo da mensagem recebida para uma função lambda, que processará a mensagem de acordo com a lógica implementada em seu código. As imagens abaixo ilustram a integração entre o tópico '*sensor/measures'*, a *rule* e a função lambda associadas.

Figura 21 - Rule criada na nuvem para acionar uma função lambda específica



Fonte: captura de tela do console AWS.

Figura 22 - Função lambda criada para processar as leituras dos eventos e salvá-las no banco de dados

Fonte: captura de tela do console Lambda AWS.

A função lambda '*sensor-measures-to-ddb*' é responsável por receber os parâmetros de leitura do sensor e armazená-los no banco de dados. Seu pseudocódigo está representado na Figura 23.

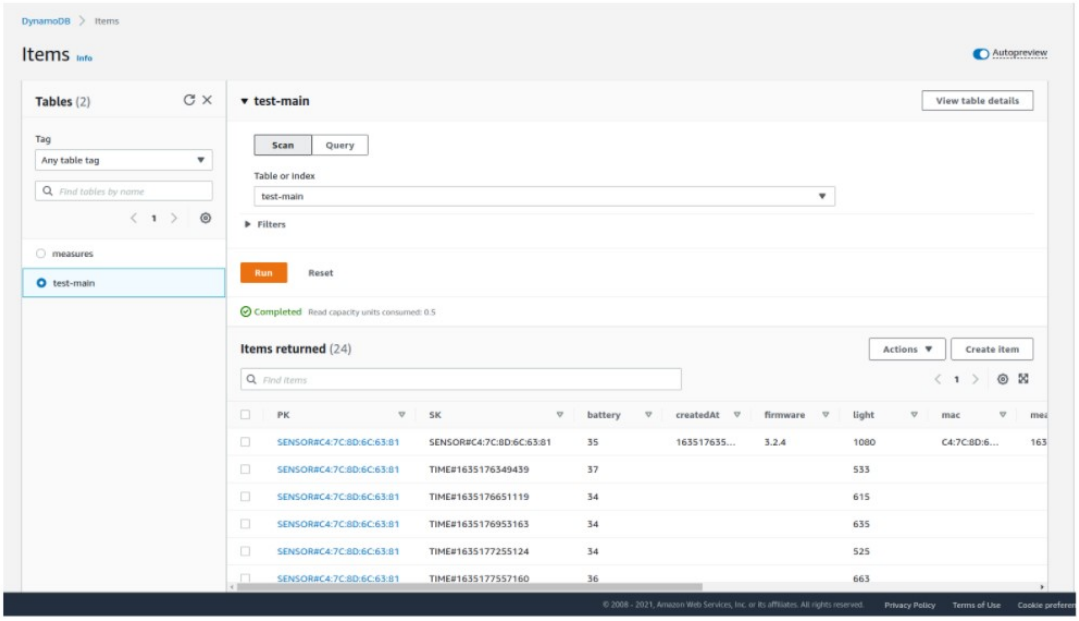
Figura 23 - Pseudocódigo da função lambda

```
1
2  handle(event) {
3      log(event);
4      sensor = getSensorParams(event.mac);
5      if (exists sensor) updateSensorDoc(event);
6      else createSensorDoc(event);
7      createMeasureDoc(event);
8      return success;
9  }
```

Fonte: os autores.

Como pode ser visto na Figura 23, a função é responsável pela atualização da última leitura do sensor assim como pela criação da medida como uma entidade separada. A entidade do sensor e as entidades das medidas podem ser vistas através do console da AWS, como ilustra a Figura 24.

Figura 24 - Dados do sensor e das medidas realizadas armazenados no banco de dados



Fonte: captura de tela do console DynamoDB AWS.

Como se está usando um banco de dados não relacional para o armazenamento dos dados, decidiu-se criar uma entidade separada para o sensor e outra entidade para cada medição

realizada. Desta forma, os dados do sensor assim como a última medição realizada podem ser acessados facilmente e de maneira eficiente através da entidade do sensor. Já o histórico de medidas pode ser facilmente filtrado por *timestamp*, o que é útil para a montagem dos gráficos, processamento do algoritmo de atualização de *Health Score* e mesmo para análise de dados futuras e rotinas de limpeza de dados antigos.

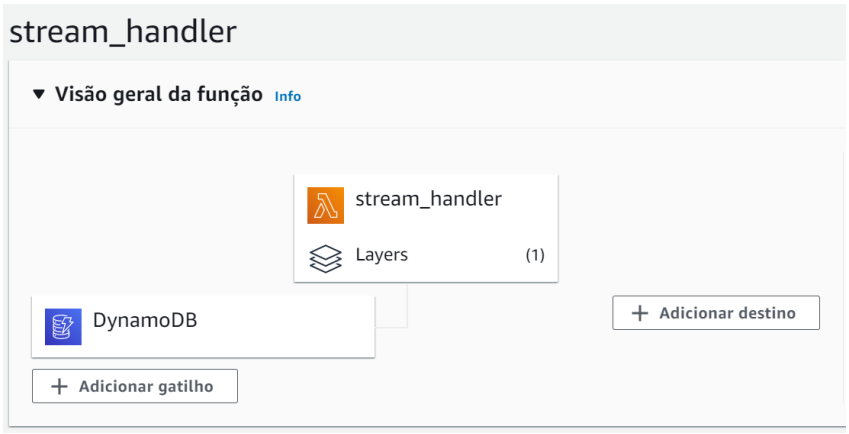
O banco de dados usado para armazenar as leituras do sensor possui uma funcionalidade chamada DynamoDB *Streams*, que permite a geração de eventos através de uma *stream* de dados com o histórico das operações de escrita nas tabelas do banco. Desta forma, toda vez que uma nova leitura de um sensor for armazenada no banco de dados, será gerado um evento que engatilhará uma função lambda para a análise e processamento dos dados lidos. No caso do sistema embarcado, esses eventos serão utilizados para decidir se algum atuador deve ser acionado. Caso seja necessário o acionamento de algum atuador, serão encaminhadas mensagens para o *broker* MQTT da nuvem, que serão sincronizadas com o *broker* MQTT de borda no dispositivo de núcleo do *greengrass* e disponibilizadas para o módulo de atuadores.

Para realizar o controle da irrigação, caso a umidade do solo lida esteja abaixo da faixa de umidade de solo recomendada para a planta que está sendo cultivada, será enviada uma mensagem para o tópico MQTT de irrigação, que irá acionar a bomba de irrigação por um período delimitado através do microcontrolador do módulo de atuadores. Para o gerenciamento da iluminação da horta, será levado em consideração o fotoperíodo da planta que está sendo cultivada, que é basicamente a divisão de horas de luz e escuridão que a planta deve ter por dia para um crescimento saudável.

Desta forma, a função deverá averiguar o horário da leitura, compará-la com o fotoperíodo cadastrado no banco e checar se o ambiente está iluminado ou não através da medida de luminosidade do sensor. Caso a horta esteja iluminada, porém a planta deveria estar no período de escuridão, será enviada uma mensagem para o tópico MQTT de iluminação que realizará o desligamento da placa de LED através do microcontrolador do módulo de atuadores. Caso a horta esteja escura, porém a planta deveria estar no período de luz, será enviada uma mensagem para o acionamento da placa de LED.

Para a implementação desta lógica, foi habilitada a *Stream* de dados da tabela principal do banco de dados e criada uma função lambda que é acionada pela *stream* de dados do banco. Assim, toda vez que acontecer uma operação de escrita na tabela principal do banco de dados, esta função será acionada com um evento informando o documento alterado e o tipo de alteração (criação, atualização ou deleção). Com isso, é possível identificar a criação de novos documentos de medida do sensor, buscar a planta relacionada a essas medições no banco de dados para descobrir seu fotoperíodo e faixas recomendadas de umidade do solo e iluminação e utilizar esses dados para a atualização da pontuação de saúde da planta e o acionamento dos atuadores caso necessário.

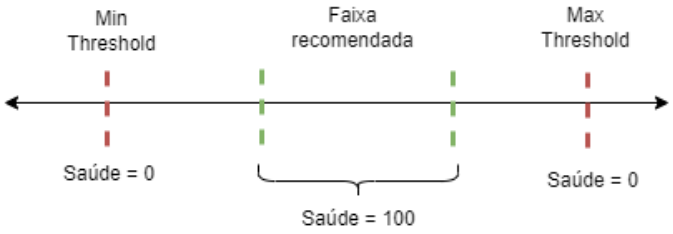
Figura 25 - handler do Dynamo Stream



Fonte: captura de tela do console Lambda AWS.

A pontuação de saúde da planta procura sintetizar a saúde da planta medindo o desvio da faixa recomendada de cada atributo através do histórico de medições do sensor. A nota de saúde pode variar de 0 a 100. Se a medida realizada estiver dentro da faixa recomendada, o score de saúde dela será 100. Se a medida estiver fora do intervalo de *threshold* mínimo e máximo pré-definidos, sua saúde será 0. Nas regiões intermediárias, será calculado o desvio percentual da faixa recomendada e este número será usado como score de saúde da planta. A pontuação de saúde geral da planta é calculada pela média aritmética da saúde de cada parâmetro monitorado como pode ser visto na Figura 26.

Figura 26 - Funcionamento do health score



Fonte: os autores.

O código para o cálculo do *health score* pode ser visto na Figura 27.

Figura 27 - Algoritmo do health score

```
const calculateHealthScore = ({ lowThresh, min, max, highThresh, measure }) => {
    if (lowThresh === min || highThresh === max) return -1;
        let score = 0;
        if (measure >= min && measure <= max) score = 100;
        else if (measure < min) {
                score = Math.round(((measure - lowThresh) / (min - lowThresh)) * 100);
                if (score < 0) score = 0;
}
                else {
                    score = 100 - Math.round(((measure - max) / (highThresh - max)) * 100);
                    if (score < 0) score = 0;
                }
                return score;
            };

```

Fonte: os autores.

Para decidir se algum atuador deve ser acionado, a função compara as novas medidas recebidas com a faixa recomendada das medidas para a planta alvo. Se a umidade do solo estiver abaixo da faixa recomendada, é enviada uma mensagem para o tópico MQTT da nuvem requisitando a ação de irrigação para o módulo de atuadores, que está subscrito neste mesmo tópico. Se a iluminação da horta não estiver de acordo com o fotoperíodo da planta alvo, é enviada uma mensagem para o tópico MQTT da nuvem requisitando o acionamento ou desativação da placa de LED, que também será consumida pelo módulo de atuadores.

Como já foi dito anteriormente, a irrigação e iluminação do sistema serão automatizadas através de um módulo de atuadores, composto por um microcontrolador conectado à rede, um conjunto de módulos relés, uma placa de iluminação LED, uma bomba de água e um reservatório. O microcontrolador usado é o ESP8266 da empresa Espressif, que possui um módulo *Wi-Fi* e pinos de comunicação GPIO, dentre outras funcionalidades. O sistema operacional usado

na placa é o Mongoose OS, que fornece uma plataforma de desenvolvimento IoT para microcontroladores de baixo consumo. Um dos diferenciais deste sistema operacional é a existência de bibliotecas e ferramentas para auxiliar o desenvolvimento de dispositivos conectados à rede, como integração com as principais plataformas de nuvem, gerenciamento de eventos e *timers* e comunicação em diversos protocolos de rede.

Para este projeto, foram usadas as bibliotecas de integração com o serviço AWS IoT e *Greengrass*, comunicação com os pinos GPIO da placa e gerenciamento de mensagens MQTT para comunicação com a nuvem. Com estas ferramentas, construiu-se uma função para gerenciar eventos engatilhados pelas mensagens MQTT enviadas pela nuvem para o acionamento dos atuadores de irrigação e iluminação. Assim, sempre que uma mensagem for enviada ao tópico de irrigação, a função engatilhará um evento que aciona o relé da bomba de irrigação por 1 minuto e o desliga na sequência. Da mesma forma, sempre que uma mensagem for enviada ao tópico de iluminação, a função engatilhará outro evento que liga ou desliga o painel de LED, dependendo do conteúdo da mensagem recebida.

O Mongoose OS disponibiliza uma ferramenta para desenvolvimento e implantação de projetos para microcontroladores de baixa potência. A partir da página web do projeto, é possível baixar um programa executável que instancia um servidor local e uma aplicação cliente para o gerenciamento de projetos embarcados como visto na Figura 28.

Figura 28 - Gerenciamento de projetos embarcados

Fonte: captura de tela do console Mongoose OS.

Um projeto simples do Mongoose OS é composto por um arquivo de configuração chamado *mos.yml* e um diretório para a declaração de códigos e *scripts*. Através do arquivo de configuração é possível definir as bibliotecas utilizadas, assim como a versão de cada uma,

variáveis de ambiente além de outras configurações gerais. No arquivo de configuração é definido o atributo *device.id*, que será necessário para a identificação do microcontrolador na nuvem. Para a implementação da lógica de controle do módulo de atuadores, foi criada uma função que atua como subscritora no tópico MQTT utilizado para o controle da iluminação e irrigação.

Sempre que uma mensagem chegar neste tópico, seu conteúdo é analisado pela função. Se for uma mensagem para ativar a irrigação, a função ativa o pino de controle do relé de irrigação por um minuto e o desativa na sequência. Se for uma mensagem para ativar a iluminação, a função verifica o parâmetro da mensagem que indica se a luz deve ser ligada ou desligada e aciona o pino controlador do relé de iluminação de acordo. Após a execução dessas tarefas, uma mensagem é publicada no tópico MQTT de resposta, indicando o id do módulo de atuadores, a tarefa executada e o novo estado do componente. Esta mensagem será consumida por uma função lambda na nuvem para atualizar o estado do módulo de atuadores no banco de dados, sincronizando assim o estado do dispositivo físico com sua representação na nuvem.

Por fim, também foi implementado um botão para desativar o módulo de atuadores através do aplicativo, que envia uma mensagem para o desligamento de todos os atuadores e altera uma variável de controle no documento de atuadores para que as lógicas de acionamento dos atuadores não sejam levadas em consideração enquanto o módulo estiver inativado pelo usuário.

Para realizar a implantação desta função no microcontrolador do módulo de atuadores, é necessário primeiramente realizar o *build* do projeto, que irá gerar um arquivo binário com o código executável. O comando para realizar o build é *'mos build –arch esp8266'* (Figura 29). Com o código do projeto compilado, o próximo passo é fazer o *flash* do código executável na memória do microcontrolador, através do comando *'mos flash'*.

Figura 29 - Comando ‘mos build’

```
$ mos build --arch esp8266 --platform esp8266
Connecting to https://build.mongoose-os.com, user test
Uploading sources (3770 bytes)
Firmware saved to C:\mos\mongoose-os\fw\examples\c_mqtt\build\fw.zip
Command completed.
```

Fonte: captura de tela do console de comando.

Figura 30 - Comando ‘mos flash’

```
$ mos flash
Loaded c_mqtt/esp8266 version 1.0 (20211212-205822)
Opening COM3 @ 115200...
Connecting to ESP8266 ROM, attempt 1 of 10...
  Connected, chip: ESP8266EX
Running flasher @ 921600...
  Flasher is running
Flash size: 4194304, params: 0x024f (dio,32m,80m)
Deduping...
     2320 @ 0x0 -> 0
   262144 @ 0x8000 -> 12288
   679088 @ 0x100000 -> 61440
      128 @ 0x3fc000 -> 0
Writing...
     4096 @ 0x7000
    12288 @ 0x10000
    16384 @ 0x101000
    36864 @ 0x107000
     8192 @ 0x112000
     4096 @ 0x3fb000
Wrote 81920 bytes in 1.13 seconds (565.17 KBit/sec)
Verifying...
     2320 @ 0x0
     4096 @ 0x7000
   262144 @ 0x8000
   679088 @ 0x100000
     4096 @ 0x3fb000
      128 @ 0x3fc000
Booting firmware...
All done!
Command completed.
```

Fonte: captura de tela do console de comando.

Após o *flash* do programa na memória do microcontrolador, deve-se conectar o dispositivo à rede *Wi-Fi* e realizar a implantação do módulo programado como um dispositivo na

nuvem. Estas ações são executadas através dos comandos '*mos wifi "SSID" "password"*' e *'mos aws-iot-setup –aws-region "region" –aws-iot-policy "policy_name"'* (Figura 31 a 33). Com estas configurações feitas, o microcontrolador já está sincronizado com os serviços da nuvem e preparado para receber as mensagens MQTT e acionar os atuadores de acordo com a lógica implantada.

Figura 31 - comando 'mos wifi'

Fonte: captura de tela do console de comando.

Com a lógica de acionamento dos atuadores implementada, é fechada a malha de controle do sistema, já que o sensor percebe as mudanças na umidade do solo e iluminação de acordo com a atuação do módulo de atuadores. Desta forma, o sistema embarcado apresentado é capaz de monitorar os principais parâmetros de saúde das plantas de uma horta e controlar sua iluminação e irrigação a fim de maximizar os parâmetros de saúde dela.

Nesta seção foram apresentados os detalhes de implementação referentes ao fluxo de leitura do sensor, armazenamento e processamento dos dados em nuvem e lógica de acionamento dos atuadores.

Figura 32 - comando 'mos aws-iot-setup'

```
$ mos aws-iot-setup --aws-region us-east-1 --aws-iot-policy mos-default
AWS region: us-east-1
Connecting to the device...
  esp8266 F6CFA2EFAF9A running c_mqtt
Current MQTT config: {
  "clean_session": true,
  "client_id": "",
  "cloud_events": true,
  "enable": false,
  "keep_alive": 60,
  "max_qos": 2,
  "max_queue_length": 5,
  "pass": "",
  "pub": "/response/butt-esp8266",
  "reconnect_timeout_max": 60,
  "reconnect_timeout_min": 2,
  "recv_mbuf_limit": 3072,
  "require_time": false,
  "server": "broker.mqttdashboard.com:1883",
  "ssl_ca_cert": "",
  "ssl_cert": "",
  "ssl_cipher_suites": "",
  "ssl_key": "",
  "ssl_psk_identity": "",
  "ssl_psk_key": "",
  "sub": "/request/butt-esp8266",
  "user": "",
  "will_message": "",
  "will_retain": false,
  "will_topic": "",
  "ws_enable": false,
  "ws_path": "/mqtt"
}
```

Fonte: captura de tela do console de comando.

Figura 33 - Continuação figura 33

```
Updating config:
  aws.thing_name =
  mqtt.enable = true
  mqtt.server = a3vu8mva2aarx9-ats.iot.us-east-1.amazonaws.com:8883
  mqtt.ssl_ca_cert = ca.pem
  mqtt.ssl_cert = aws-butt-esp8266.crt.pem
  mqtt.ssl_key = aws-butt-esp8266.key.pem
Setting new configuration...
Command completed.
```

Fonte: captura de tela do console de comando.

6.5. Aplicação Cliente

Para fazer a interface com usuário, um aplicativo móvel foi desenvolvido. A ideia deste aplicativo não se restringe apenas a usuários que procuram apenas automatizar uma cultura, mas também auxiliar e incentivar para que se crie um interesse maior em começar a cultivar uma plantação em casa, seja automatizada ou não. São fornecidos dados técnicos recomendados para cada tipo de planta dentro do escopo da aplicação, para aumentar o conhecimento sobre o cultivo dela. O aplicativo fornece também de maneira mais didática e simples um passo a passo de uma montagem do protótipo produzido pelo grupo, assim como o produto pronto como um todo.

Como funcionalidade principal, esta interface coleta as medições e informar para o usuário, seja em tempo real, ou também através de *dashboards* ilustrativos os dados coletados ao longo do tempo. É importante notar que além de uma produção para consumo próprio, o protótipo possibilita também que seja produzido em uma escala comercial ou doação para possíveis parceiros interessados nos insumos produzidos pelos usuários, pois apesar de o nosso foco ser voltado para a agricultura urbana, estão sendo considerados escalabilidade tanto para o aplicativo quanto para o protótipo de controle e monitoramento.

Outra possível implementação para tornar o aplicativo mais atrativo é a de compartilhar resultados entre os usuários da plataforma por meio de publicações, visando aproximar todos usuários e dados gerados com uma usabilidade próxima de uma rede social. Porém, esta será

uma funcionalidade secundária, sendo o foco principal do projeto o monitoramento e automação da plantação.

Portanto, a função da aplicação cliente é fornecer informação simplificada e de qualidade para usuários que desejam manter uma horta em casa, assim como mecanismos básicos para a visualização dos parâmetros lidos pelos sensores e para o controle dos atuadores, este podendo ser automatizado pelo próprio sistema, ou sob demanda com comandos do usuário. Além disso, o aplicativo deve fornecer também uma seção de notificações para que o sistema notifique o usuário sobre informações relevantes de seu plantio.

6.5.1. Questionário de opinião pública

Para entender melhor as demandas dos usuários potenciais da plataforma, foi realizado um estudo de usuário através de um questionário que procurou traçar o perfil dos interessados na aplicação. O questionário coletou informações básicas como faixa etária, grau de escolaridade e dividiu o conjunto de respostas em dois grupos: as pessoas que já possuíam horta em casa e as que não possuíam. 57 pessoas participaram do estudo. 70,2% dos entrevistados tinham até 25 anos e 89,5% tinham pelo menos o ensino superior incompleto. Com relação ao tipo de entrevistados, 61,4% não possuíam uma horta doméstica e 38,6% já possuíam.

Das pessoas que responderam já ter uma horta caseira, constatou-se que as principais plantas cultivadas foram manjericão (15 resultados), cebolinha (13 resultados) e salsinha (12 resultados). Dos objetivos para se manter uma horta caseira, o principal escolhido foi para se ter uma alimentação mais saudável, seguido de fins terapêuticos ou entretenimento e contribuição com a sustentabilidade e o meio ambiente. 63,6% dos entrevistados afirmaram ter gastado até 100 reais como investimento inicial na horta caseira, sendo que mais de 95% gastaram até 500 reais. O gasto mensal para manutenção da horta foi semelhante, com 63,6% gastando até 25 reais para manutenção mensal e mais de 95% gastando até 50 reais. Com relação ao nível de conhecimento sobre cultivo e manutenção de hortas caseiras, 27,3% dos entrevistados afirmaram não ter conhecimento suficiente e sofrer perdas constantes de plantas, 54,5% afirmaram conhecer apenas o básico e apenas 18,2% afirmaram ter experiência e familiaridade com o assunto. Já o tempo

médio de dedicação semanal à horta foi relativamente baixo, com 50% dos entrevistados dedicando até 1 hora por semana, 40,9% dedicando entre 1 e 3 horas por semana e apenas 9,1% dedicando entre 3 e 5 horas por semana.

Quando se perguntou sobre o uso de automação nas hortas domésticas, 86,4% dos entrevistados afirmaram não utilizar, sendo que 40,9% disseram que não possuem interesse em automatizá-las. Apenas 13,6% dos entrevistados afirmaram usar algum tipo de automação em suas hortas. Com relação ao comércio dos insumos plantados, mais de 80% dos usuários afirmaram ter interesse na compra de produtos de outros donos de hortas caseiras, porém mais de 80% também afirmou não ter interesse em vender o que produz. Por fim, 68,2% dos usuários afirmaram comprar os insumos para sua horta em lojas físicas, 13,6% afirmaram comprar pela internet e 18,2% disseram que compra em ambos.

Já as pessoas que afirmaram não ter uma horta caseira, 60% disseram que gostariam de ter uma, mas não possuem tempo, conhecimento ou orçamento para realizar o projeto, 31,4% disseram que gostariam de ter uma horta caseira, mas não sabem como fazer para ter uma ou ainda não tomaram iniciativa de fazer e apenas 8,6% afirmaram não ter interesse no assunto. Com relação ao investimento inicial, 71,4% dos entrevistados afirmaram estar dispostos a pagar de 100 a 300 reais, e 23% disseram que pagariam até 100 reais. Já sobre os investimentos para manutenção semanal, 62,9% afirmaram estar dispostos a pagar até 25 reais e 31,4% afirmaram que podem pagar de 25 a 50 reais.

O nível de conhecimento para manter uma horta doméstica foi parecido com o outro grupo, com 45,7% dos entrevistados não possuindo conhecimento nenhum, 34,3% conhecendo apenas o básico e 17,1% tendo alguma familiaridade com o assunto. O tempo semanal disponível para se dedicar a horta também é escasso nesse grupo, com 54,3% dos participantes possuindo até 1 hora por semana e 40% possuindo entre 1 e 3 horas por semana. Neste grupo, 77,1% afirmaram ter interesse em uma plataforma para compra de produtos produzidos em hortas caseiras e 51,4% afirmaram ter interesse também em vender os produtos que viessem a cultivar.

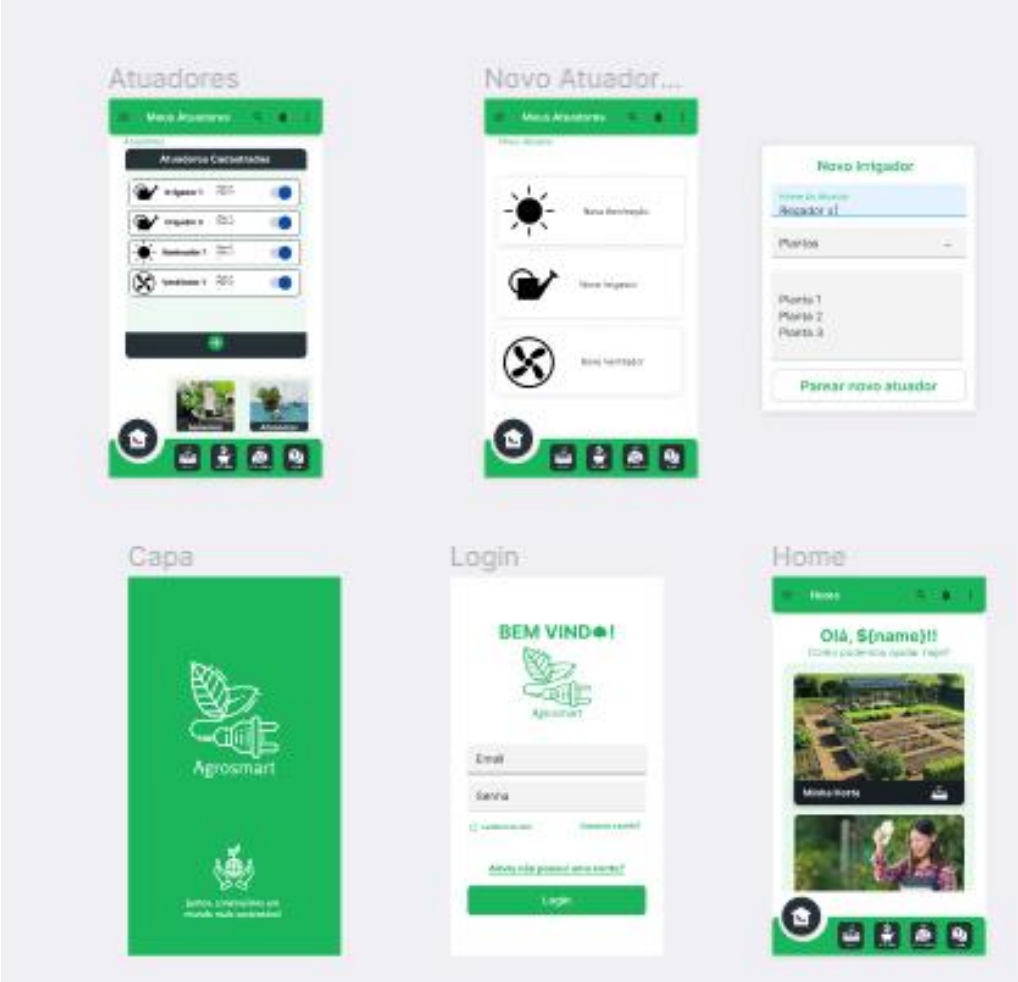
Como resultado da pesquisa, pode-se validar a hipótese de que as pessoas possuem pouco tempo, pouco conhecimento e um orçamento reduzido para manter hortas caseiras. Percebeu-se também que a maioria dos usuários gostaria de uma solução de automação e monitoramento

de suas hortas, e existe um interesse em ambos os grupos no comércio dos insumos produzidos, tanto na venda, mas principalmente na compra. Por fim, nota-se que a maior parte dos entrevistados que não cultivam uma horta em casa possuem interesse em ter, e as pessoas que já cultivam uma horta caseira afirmaram fazê-lo para manter uma alimentação mais saudável, por fins terapêuticos ou para contribuir com o meio ambiente.

Levando em consideração os resultados do estudo, decidiu-se optar por uma interface minimalista e simples, que auxilie o usuário a manter suas plantas saudáveis através de monitoramento e automação, visando compensar a falta de tempo e conhecimento deles na manutenção de suas hortas. Através da interface, os usuários serão capazes se cadastrar e fazer *login* na plataforma, cadastrar suas plantas de acordo com a espécie, cadastrar sensores e atuadores e relacionar suas plantas com os componentes cadastrados. A interface trará informações sobre a espécie cadastrada, os valores atuais de medição do sensor, o histórico de leituras através de gráficos e uma pontuação de saúde para a planta.

A seguir, serão apresentadas alguns *mockups* de telas desenvolvidas para o aplicativo que ilustram as principais funcionalidades da aplicação cliente.

Figura 34 - Mockups do protótipo



Fonte: captura de tela da plataforma Figma.

Figura 35 - Continuação dos mockups

Fonte: captura de tela da plataforma Figma.

6.6. Backend

Pautado em aplicações com a ferramenta Serverless Framework, o *backend* deste projeto utiliza arquitetura de microsserviços, NodeJS como interpretador para execução, *TypeScript* como linguagem de programação e DynamoDB como banco de dados. Os serviços são determinados de acordo com as necessidades da aplicação cliente e do sistema embarcado.

Os primeiros passos para a estrutura do projeto foram: infraestrutura de microsserviços, aplicação do *serverless* framework e modelagem de entidades para o banco de dados. Os dois primeiros passos tiveram prioridade, pois fazem parte da estrutura do sistema, o último passo

foi iniciado e será abordado na seção a seguir. Após a aplicação da modelagem no DynamoDB, foram criados e fornecidos *endpoints* para que sejam consumidos pela aplicação cliente do projeto, de maneira a tornar o *backend* uma API a ser consumida pela interface móvel.

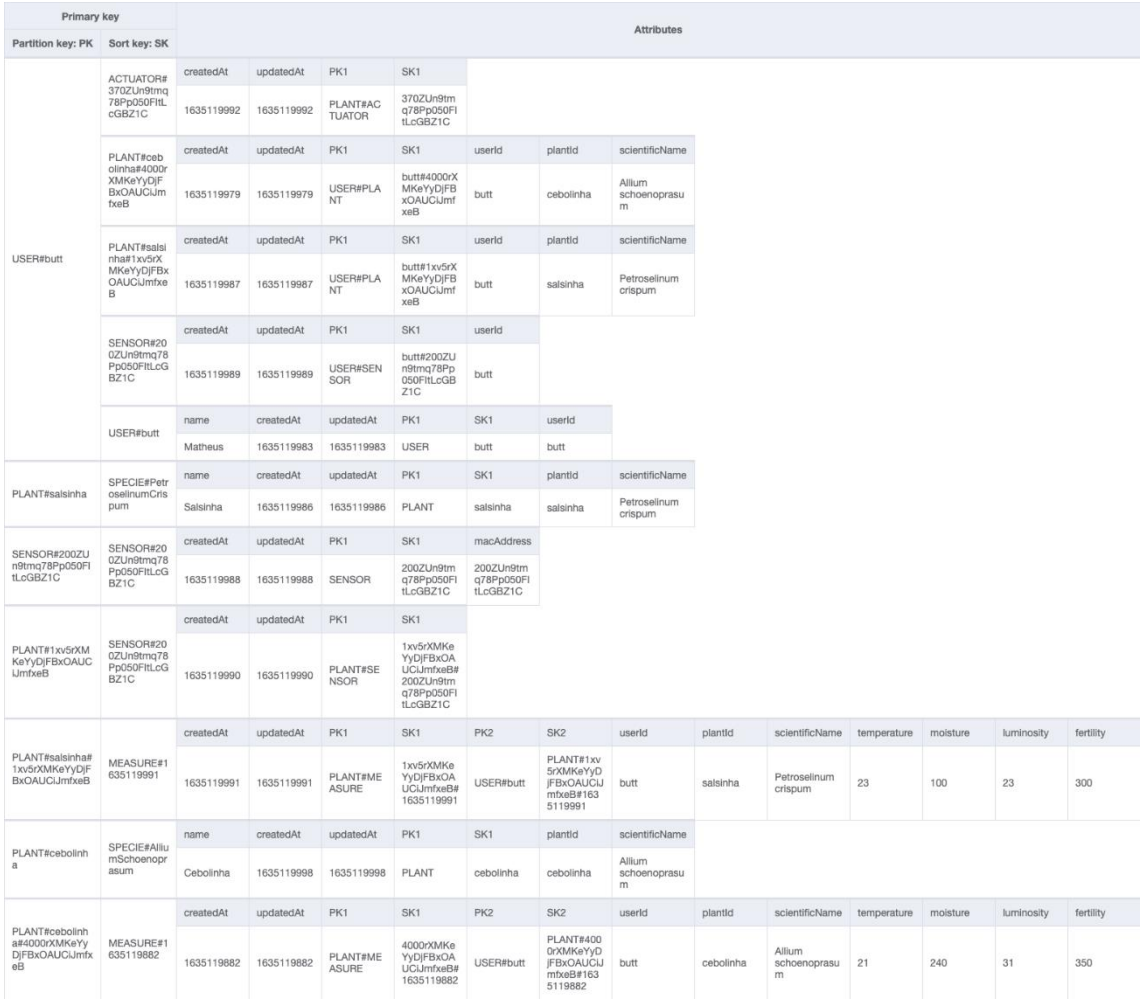
6.6.1. Modelagem de dados

Para a modelagem inicial dos dados a serem recebidos pelo sistema embarcado e pela interação com o usuário (pensando no escopo inteiro do projeto), foi utilizada a ferramenta NoSQL Workbench da Amazon, que oferece uma visualização amigável para a modelagem e verificação dos padrões de acesso de um banco não relacional, através de entradas que podem ser determinadas pelo usuário ou serem importadas a partir de arquivo no formato JSON, por exemplo.

Para o sistema em questão foram consideradas as seguintes entidades: USER, PLANT, SENSOR e ACTUATOR e as seguintes relações: USER#PLANT, USER#ACTUATOR, USER#SENSOR, PLANT#MEASURE e PLANT#SENSOR. Esses documentos foram determinados a partir das necessidades iniciais do projeto e de funcionalidades futuras, que podem ser desenvolvidas em uma eventual continuidade do sistema. Os padrões de acesso para a interação com o usuário também foram levados em consideração, alguns atributos (nome, nome científico, por exemplo) foram adicionados com o objetivo de fornecer informações de reconhecimento de entidades a partir de alguma ação no aplicativo.

Os padrões de acesso determinados foram pensados em permitir separação entre informações e em agregação de dados para o usuário. Foi necessário criar três índices (principal e mais dois secundários) para permitir que informações chaves pudessem ser acessadas apenas por uma busca, como exemplo na Figura 36.

Figura 36 - Índice principal da modelagem de dados

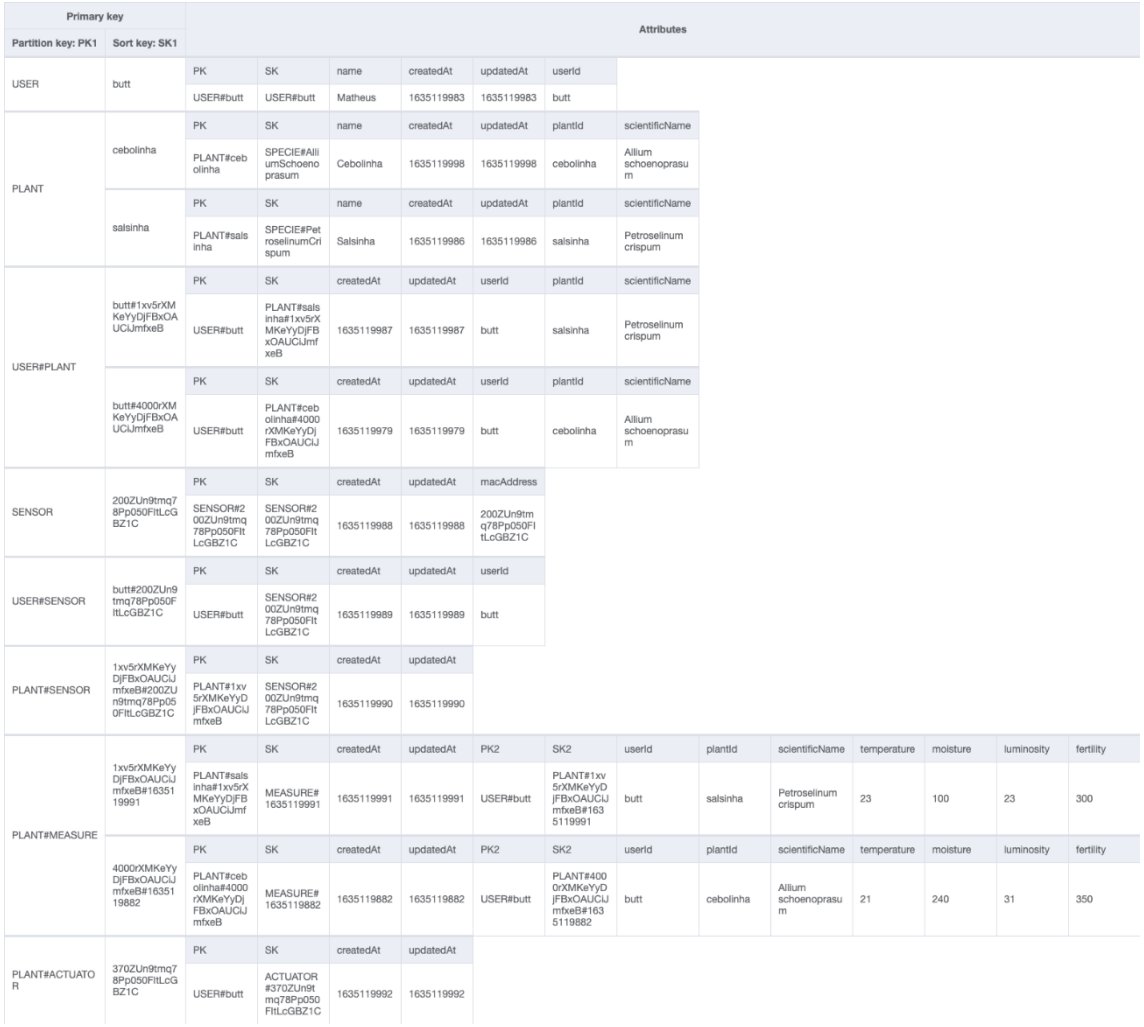


Primary key		Attributes												
Partition key: PK	Sort key: SK													
USER#butt	ACTUATOR#370ZUn9tmq78Pp050FItLcGBZ1C	createdAt	updatedAt	PK1	SK1									
		1635119992	1635119992	PLANT#ACTUATOR	370ZUn9tmq78Pp050FItLcGBZ1C									
	PLANT#cebolinha#4000rXMKeYyDjFBxOAUCiJmfxeB	createdAt	updatedAt	PK1	SK1	userId	plantId	scientificName						
		1635119979	1635119979	USER#PLANT	butt#4000rXMKeYyDjFBxOAUCiJmfxeB	butt	cebolinha	Allium schoenoprasum						
	PLANT#salsinha#1xv5rXMKeYyDjFBxOAUCiJmfxeB	createdAt	updatedAt	PK1	SK1	userId	plantId	scientificName						
		1635119987	1635119987	USER#PLANT	butt#1xv5rXMKeYyDjFBxOAUCiJmfxeB	butt	salsinha	Petroselinum crispum						
	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	createdAt	updatedAt	PK1	SK1	userId								
		1635119989	1635119989	USER#SENSOR	butt#200ZUn9tmq78Pp050FItLcGBZ1C	butt								
	USER#butt	name	createdAt	updatedAt	PK1	SK1	userId							
		Matheus	1635119983	1635119983	USER	butt	butt							
PLANT#salsinha	SPECIE#PetroselinumCrispum	name	createdAt	updatedAt	PK1	SK1	plantId	scientificName						
		Salsinha	1635119986	1635119986	PLANT	salsinha	salsinha	Petroselinum crispum						
SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	createdAt	updatedAt	PK1	SK1	macAddress								
		1635119988	1635119988	SENSOR	200ZUn9tmq78Pp050FItLcGBZ1C	200ZUn9tmq78Pp050FItLcGBZ1C								
PLANT#1xv5rXMKeYyDjFBxOAUCiJmfxeB	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	createdAt	updatedAt	PK1	SK1									
		1635119990	1635119990	PLANT#SENSOR	1xv5rXMKeYyDjFBxOAUCiJmfxeB#200ZUn9tmq78Pp050FItLcGBZ1C									
PLANT#salsinha#1xv5rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119991	createdAt	updatedAt	PK1	SK1	PK2	SK2	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		1635119991	1635119991	PLANT#MEASURE	1xv5rXMKeYyDjFBxOAUCiJmfxeB#1635119991	USER#butt	PLANT#1xv5rXMKeYyDjFBxOAUCiJmfxeB#1635119991	butt	salsinha	Petroselinum crispum	23	100	23	300
PLANT#cebolinha	SPECIE#AlliumSchoenoprasum	name	createdAt	updatedAt	PK1	SK1	plantId	scientificName						
		Cebolinha	1635119998	1635119998	PLANT	cebolinha	cebolinha	Allium schoenoprasum						
PLANT#cebolinha#4000rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119882	createdAt	updatedAt	PK1	SK1	PK2	SK2	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		1635119882	1635119882	PLANT#MEASURE	4000rXMKeYyDjFBxOAUCiJmfxeB#1635119882	USER#butt	PLANT#4000rXMKeYyDjFBxOAUCiJmfxeB#1635119882	butt	cebolinha	Allium schoenoprasum	21	240	31	350

Fonte: captura de tela da ferramenta NoSQL Workbench.

A partir da leitura dos índices secundários (Figuras 37 e 38), pode-se notar o tipo de retorno das informações agregadas de acordo com cada determinação de responsabilidade, o primeiro índice secundário funciona como identificador do documento, informa os nomes de entidades envolvidos e seus respectivos valores identificadores, já o segundo índice secundário é um agregador de medidas de plantas de usuários.

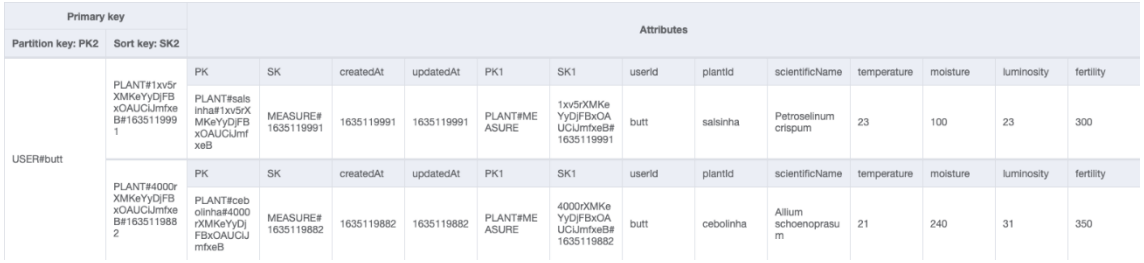
Figura 37 - Primeiro índice secundário



Primary key		Attributes												
Partition key: PK1	Sort key: SK1													
USER	butt	PK	SK	name	createdAt	updatedAt	userId							
		USER#butt	USER#butt	Matheus	1635119983	1635119983	butt							
PLANT	cebolinha	PK	SK	name	createdAt	updatedAt	plantId	scientificName						
		PLANT#cebolinha	SPECIE#AlliumSchoenoprasum	Cebolinha	1635119998	1635119998	cebolinha	Allium schoenoprasum						
	salsinha	PK	SK	name	createdAt	updatedAt	plantId	scientificName						
		PLANT#salsinha	SPECIE#PetroselinumCrispum	Salsinha	1635119986	1635119986	salsinha	Petroselinum crispum						
USER#PLANT	butt#1xv5rXMKeYyDjFBxOAUCiJmfxeB	PK	SK	createdAt	updatedAt	userId	plantId	scientificName						
		USER#butt	PLANT#salsinha#1xv5rXMKeYyDjFBxOAUCiJmfxeB	1635119987	1635119987	butt	salsinha	Petroselinum crispum						
	butt#4000rXMKeYyDjFBxOAUCiJmfxeB	PK	SK	createdAt	updatedAt	userId	plantId	scientificName						
		USER#butt	PLANT#cebolinha#4000rXMKeYyDjFBxOAUCiJmfxeB	1635119979	1635119979	butt	cebolinha	Allium schoenoprasum						
SENSOR	200ZUn9tmq78Pp050FItLcGBZ1C	PK	SK	createdAt	updatedAt	macAddress								
		SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	1635119988	1635119988	200ZUn9tmq78Pp050FItLcGBZ1C								
USER#SENSOR	butt#200ZUn9tmq78Pp050FItLcGBZ1C	PK	SK	createdAt	updatedAt	userId								
		USER#butt	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	1635119989	1635119989	butt								
PLANT#SENSOR	1xv5rXMKeYyDjFBxOAUCiJmfxeB#200ZUn9tmq78Pp050FItLcGBZ1C	PK	SK	createdAt	updatedAt									
		PLANT#1xv5rXMKeYyDjFBxOAUCiJmfxeB	SENSOR#200ZUn9tmq78Pp050FItLcGBZ1C	1635119990	1635119990									
PLANT#MEASURE	1xv5rXMKeYyDjFBxOAUCiJmfxeB#163511 9991	PK	SK	createdAt	updatedAt	PK2	SK2	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		PLANT#salsinha#1xv5rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119991	1635119991	1635119991	USER#butt	PLANT#1xv5rXMKeYyDjFBxOAUCiJmfxeB#1635119991	butt	salsinha	Petroselinum crispum	23	100	23	300
	4000rXMKeYyDjFBxOAUCiJmfxeB#163511 9882	PK	SK	createdAt	updatedAt	PK2	SK2	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		PLANT#cebolinha#4000rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119882	1635119882	1635119882	USER#butt	PLANT#4000rXMKeYyDjFBxOAUCiJmfxeB#1635119882	butt	cebolinha	Allium schoenoprasum	21	240	31	350
PLANT#ACTUATOR	370ZUn9tmq78Pp050FItLcGBZ1C	PK	SK	createdAt	updatedAt									
		USER#butt	ACTUATOR#370ZUn9tmq78Pp050FItLcGBZ1C	1635119992	1635119992									

Fonte: captura de tela da ferramenta NoSQL Workbench.

Figura 38 - Segundo índice secundário



Primary key		Attributes											
Partition key: PK2	Sort key: SK2												
USER#butt	PLANT#1xv5rXMKeYyDjFBxOAUCiJmfxeB#1635119991	PK	SK	createdAt	updatedAt	PK1	SK1	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		PLANT#salsinha#1xv5rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119991	1635119991	1635119991	PLANT#MEASURE	1xv5rXMKeYyDjFBxOAUCiJmfxeB#1635119991	butt	salsinha	Petroselinum crispum	23	100	23	300
	PLANT#4000rXMKeYyDjFBxOAUCiJmfxeB#1635119882	PK	SK	createdAt	updatedAt	PK1	SK1	userId	plantId	scientificName	temperature	moisture	luminosity	fertility
		PLANT#cebolinha#4000rXMKeYyDjFBxOAUCiJmfxeB	MEASURE#1635119882	1635119882	1635119882	PLANT#MEASURE	4000rXMKeYyDjFBxOAUCiJmfxeB#1635119882	butt	cebolinha	Allium schoenoprasum	21	240	31	350

Fonte: captura de tela da ferramenta NoSQL Workbench.

A modelagem apresentada pode sofrer alterações caso haja andamento no projeto, um dos motivos de optar por banco não relacional é poder alterar de maneira menos custosa a estrutura de alguma entidade do banco e as informações contidas nela. A partir deste conhecimento, tarefas como adaptar os índices e os padrões de acesso podem ser realizadas de acordo com o crescimento do sistema.

6.6.2. Implementação

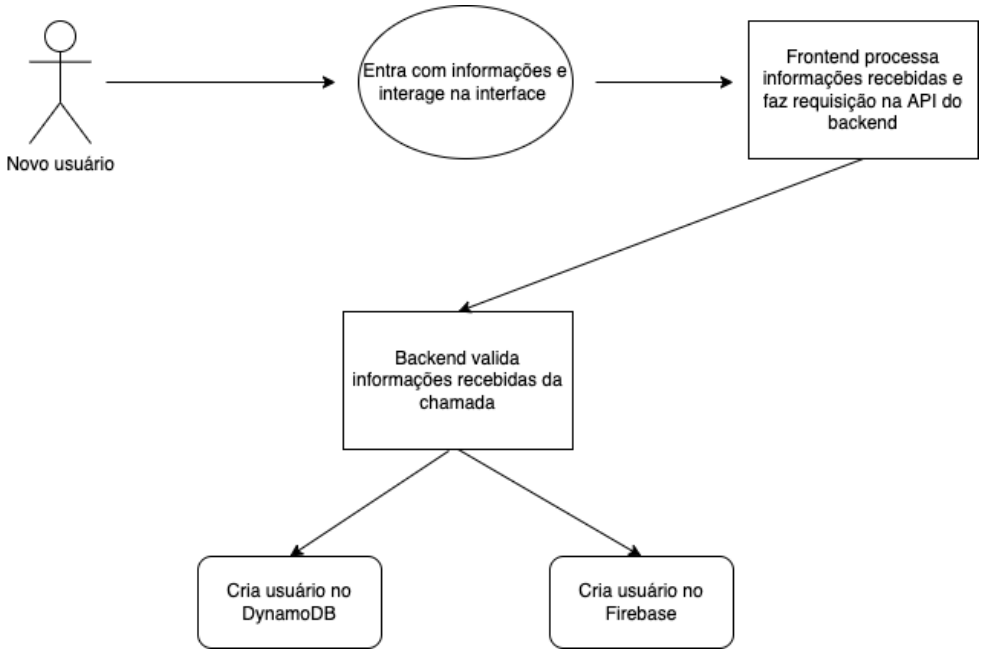
Dada a modelagem do banco de dados, os próximos passos a serem tratados no desenvolvimento são: autenticação e codificação de API para fluxo de CRUD e listagens para cada entidade. Esta seção aborda sobre a parte técnica aplicada e evolução de cada um desses tópicos.

6.6.2.1. autenticação

A maior parte do controle de autenticação será controlado pela *interface mobile*, onde o *firebase* irá atuar como ferramenta auxiliar, já que oferece um consistente e conhecido sistema de autenticação. O *backend* é responsável por duas principais tarefas: validar as informações de autenticação no fluxo de criação de usuário e por validar o *token* de autenticação quando houver uma requisição na API do *backend* por parte *frontend*.

A tarefa de criação de usuário (Figura 39) consiste em validar as informações recebidas na requisição, mapear os dados para as criações do usuário no *firebase* e no DynamoDB e criar no *firebase* uma assinatura personalizada para o usuário, que irá contribuir para a criação do *token* de autenticação (Figura 40).

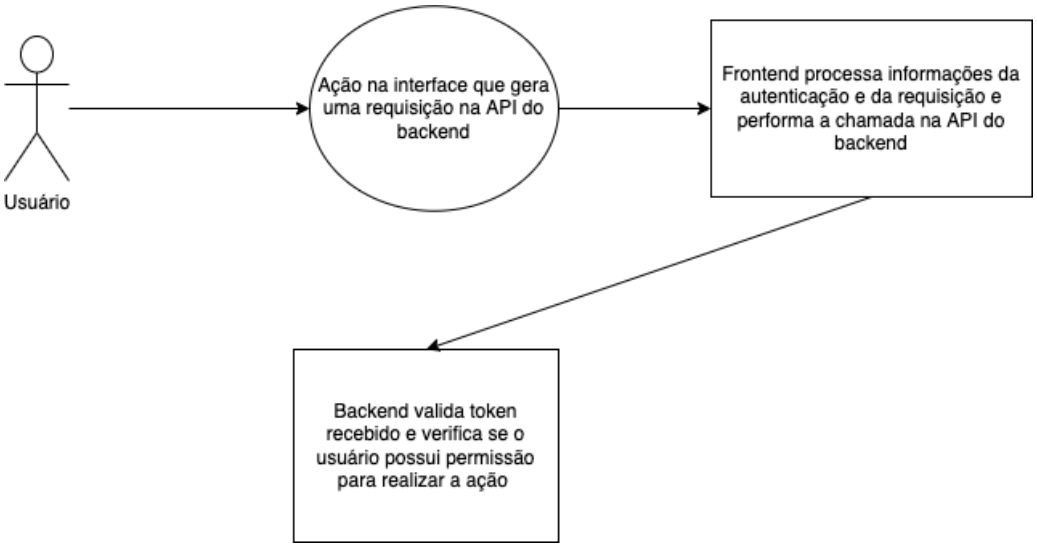
Figura 39 - Esquema de criação de usuário



Fonte: os autores.

A tarefa de validar o token de autenticação consiste em comparar informações alocadas no sistema com as recebidas na requisição e permitir ou proibir o acesso de acordo com o papel e do tipo de usuário.

Figura 40 - Esquema de autenticação de usuário



Fonte: os autores.

A autenticação é verificada em todos os *endpoints* da API do *backend*, exceto no fluxo de criação de usuário. Validar as informações de autenticação em todos os fluxos possíveis da API é importante para garantir segurança, consistência de dados e privilégio mínimo para usuários dentro do sistema.

6.6.2.2. *backend* como API

Da perspectiva da *interface mobile* o *backend* pode atuar, dentre outras possibilidades, como uma API que fornece pontos de comunicação (*endpoints*) para serem consumidos de acordo com as interações do usuário e de eventos programados. Uma possível abordagem para o desenvolvimento dessa comunicação é criar um *endpoint* para cada operação (criar, ler, atualizar e deletar) sobre cada entidade definida na modelagem do banco de dados, ainda são necessários *endpoints* de listagens que podem ser aplicados a cada índice de cada entidade. Utilizando abordagem mais prática, a API vem sendo desenvolvida de acordo com as necessidades e com o andamento do *frontend*, ou seja, um *endpoint* já é criado pensando na integração e na utilidade que a funcionalidade vai oferecer para o usuário. Por isso, nem todas as entidades vão possuir *endpoints* para cada operação e nem para listagem de todos os índices.

Para este ponto do projeto, as entidades que são mais importantes para a integração com o *frontend* são: USER, PLANT, ACTUATOR e SENSOR. Foram criados *endpoints* que performam operação de criar e ler para todas as entidades e as relações USER#PLANT, USER#PLANT e PLANT#SENSOR, USER#ACTUATOR e PLANT#ACTUATOR receberam foco nesta parte inicial, pois são muito importantes para o fluxo básico de funcionamento do sistema. Ainda foram criados *endpoints* de listagem para algumas entidades e para algumas funcionalidades (listar plantas ou sensores pertencentes a um usuário, por exemplo). O modelo a ser seguido para cada *endpoint* pode ser visto na Tabela 1:

Tabela 1 - APIs

Nome	Método HTTP	Auth	Url relativa	Parâmetros
createUser	*POST*	Sim	*user-prefix/stage/*	*email*: texto *phone*: texto *fullName*: texto *picture*: número
updateUser	*PATCH*	Sim	*user-prefix/stage*	*name:*texto *picture: texto*
createUserPlant	*POST*	Sim	*user-prefix/stage/user-plant*	*name*: texto *displayName*: texto *picture*: texto
readUserPlant	*GET*	Sim	*user-prefix/stage/user-plant*	*plantId*: texto
updateUserPlant	*PATCH*	Sim	*user-prefix/stage/user-plant*	*plantId*: texto *picture:* texto *name*: texto
createUserSensor	*POST*	Sim	*user-prefix/stage/user-sensor*	*macAddress*: texto
readUserSensor	*GET*	Sim	*user-prefix/stage/user-sensor*	*macAddress*: texto
listUserPlants	*GET*	Sim	*user-prefix/stage/user-plant*	
listUserSensors	*GET*	Sim	*user-prefix/stage/user-sensor*	
listUserActuators	*GET*	Sim	*user-prefix/stage/user-actuator*	
createPlant	*POST*	Sim	*plant-prefix/stage*	*name*: texto *scientificName*: texto
readPlant	*GET*	Sim	*plant-prefix/stage*	*name*: texto
createPlantSensor	*POST*	Sim	*plant-prefix/stage/plant-sensor*	*macAddress*: texto *plantId*: texto
readPlantSensor	*GET*	Sim	*plant-prefix/stage/plant-sensor*	*macAdress*: texto *plantId*: texto
CreatePlantActuator	*POST*	Sim	*plant-prefix/stage/plant-actuator*	*actuatorId*: texto *plantId*: texto
readPlantActuator	*GET*	Sim	*plant-prefix/stage/plant-actuator*	*actuatorId*: texto *plantId*: texto
listPlants	*GET*	Sim	*plant-prefix/stage/list-plant*	
createSensor	*POST*	Sim	*sensor-prefix/stage*	*macAdress*: texto
readSensor	*POST*	Sim	*sensor-prefix/stage*	*macAdress*: texto
createActuator	*POST*	Sim	*actuator-prefix/stage*	*actuatorId*: texto

				displayName: texto
readActuator	*POST*	Sim	*actuator-prefix/stage*	*actuatorId*: texto
actuatorActivation	*PATCH*	Sim	*actuator-prefix/stage/activation*	*actuatorId:* texto *active:* booleano

Fonte: os autores.

Sendo:

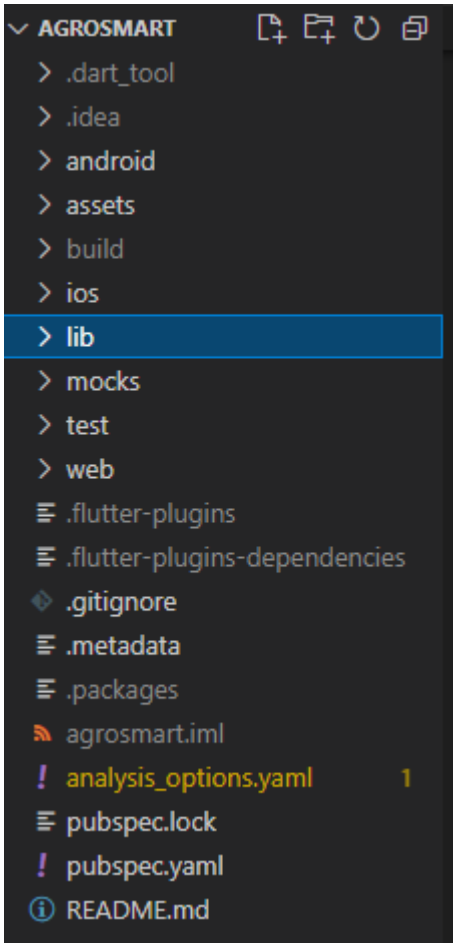
- Método HTTP: método de chamada podendo ser do tipo POST, GET, PATCH ou DELETE
- Autenticação: indica se para esta ação o usuário necessita estar autenticado ou não
- Url relativa: o caminho relativo que identifica o *endpoint*
- Parâmetros de chamada: os parâmetros que a chamada deve receber, seguido do tipo de dado (texto, numérico ou booleano), a chamada pode não ter parâmetros.

Pode-se observar que os *endpoints* criados envolvem fluxos comuns a uma aplicação web e que os parâmetros envolvidos são, em sua maioria, intrínsecos a cada entidade que for modelada (a entidade *USER*, por exemplo, possui nome e documento). Nas *urls* relativas o parâmetro *user-prefix* (válido também para *sensor-prefix* e *plant-prefix*) é a identicação para o microsserviço, decodificada em *url* para o acesso do *frontend* e o parâmetro *stage* se refere ao ambiente em que a aplicação está sendo executada (*development* ou *production*, por exemplo). As listagens foram desenvolvidas com base em necessidades da aplicação *mobile*, já que antes da integração ser realizada, é preciso que o *backend* forneça o *endpoint* de maneira coerente com a funcionalidade no *frontend*.

6.7. Frontend

A estrutura do *frontend* se baseia na estrutura determinada pelo Google para o desenvolvimento de aplicações *mobile* junto da ferramenta Flutter e linguagem Dart. Com a prototipação feita no Figma já planejando o formato *mobile* os *mockups* (Figuras 34 e 35) facilitaram a implementação das telas. O diretório raiz de um projeto em Flutter se separa nas seguintes pastas (Figura 41):

Figura 41 - Diretórios do projeto Flutter

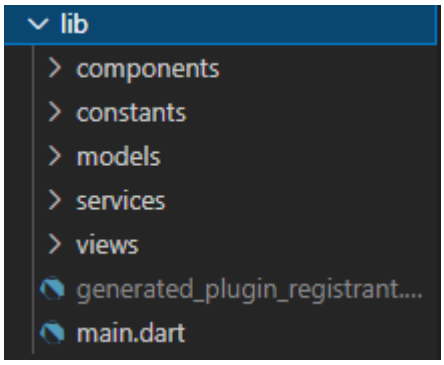


Fonte: captura de tela do programa VSCode.

- Diretórios relacionados à plataforma:
    - Android, iOS e web: referentes às plataformas da qual o app está disponível
    - Build, .dart_tool, .idea: responsável pela compilação do projeto
    - Pubspec.yaml: arquivo para informar todas as bibliotecas adicionais assim como as versões utilizadas de cada uma delas
- Diretórios relacionados ao desenvolvimento:
    - *Assets*: contém todas imagens usadas como ícones ou substitutos do sistema
    - *Mocks:* contém o formato das respostas esperadas pelas APIs que inicialmente serviram de molde antes da integração e posteriormente poderão ser usados nos testes
    - *Test*: relaciona com os testes automatizados do sistema

- E, por fim, o diretório mais importante da implementação chamado lib (Figura 42).

Figura - 42 Diretório lib do projeto Flutter

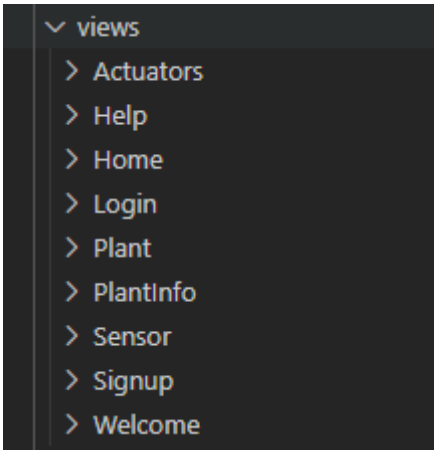


Fonte: captura de tela do programa VSCode.

  - Em *componentes:* estão componentes que são reaproveitados em mais de uma tela
  - *Constants:* encontram-se tanto valores constantes como cores e tamanhos padrões para manter a aplicação consistente, há também o arquivo de rotas que fazem parte das integrações APIs
  - *Models:* para todo tido de objeto criado e para facilitar a manipulação dos dados recebidos pelo *backend,* são criados modelos de objetos específicos como o de planta, sensores e atuadores que facilitam o acesso a seus atributos
  - *Services:* responsável pelas integrações de todas APIs do *backend*
  - *Main:* assim como em outras linguagens de programação, é o trecho responsável por unir todas as outras funções do sistema
  - *Views* (Figura 43)*:* Separação e implementação das telas, usando-se na maioria componentes nativos da seguinte forma:
    - *Actuators:* agrupamento das telas de criação e listagem de atuadores (Figura 53)
    - *Help:* tela com informações dos desenvolvedores e do projeto
    - *Home:* tela de menu que pode ser vista na Figura 50
    - *Login:* tela de login que pode ser vista na Figura 50
    - *Plant:* Todo o fluxo de plantas desde a listagem, criação e detalhamento (Figura 51)

- *PlantInfo:* tela específica do detalhamento de plantas cadastradas com os parâmetros ideais por espécie (Figura 51 ao centro)
- *Sensor:* agrupamento das telas de criação e listagem de sensores (Figura 52)
- *Signup:* tela de cadastro que pode ser vista na Figura 50
- *Welcome: splash screen,* (tela de carregamento de início).

Figura 43 - Camada de *views*



Fonte: captura de tela do programa VSCode.

7. RESULTADOS E TESTES

7.1. Resultados

7.1.1. Sistema Embarcado

Após a implementação do sistema embarcado, foi realizada a montagem do protótipo de monitoramento e controle de uma horta doméstica. Como se deseja controlar a iluminação do ambiente de forma a simular períodos de luz e escuridão, optou-se por realizar um cultivo indoor, através de uma estufa fechada (Figura 44).

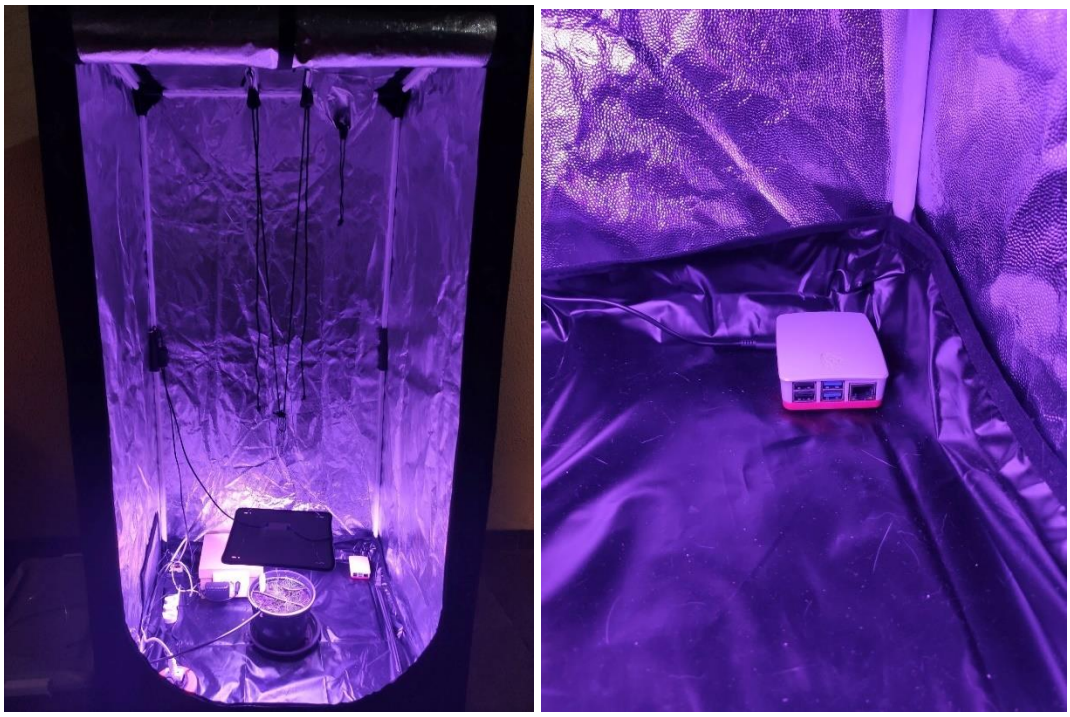
Figura 44 - Estufa vista por fora

Fonte: os autores.

Dentro da estufa estão o vaso da planta, o sensor Xiaomi Flora, o dispositivo de borda Raspberry Pi, o módulo de atuadores composto por um microcontrolador ESP8266 e dois relés, instalação das fiações, uma placa de LED, um suporte regulável para a placa e uma mangueira

para irrigação instalada no vaso da planta (Figura 45 à esquerda). E no canto direito da estufa se encontra o dispositivo de borda, responsável pela leitura periódica do sensor e comunicação com a nuvem (Figura 45 à direita).

Figura 45 - Estufa por dentro e posicionamento do Raspberry



Fonte: os autores.

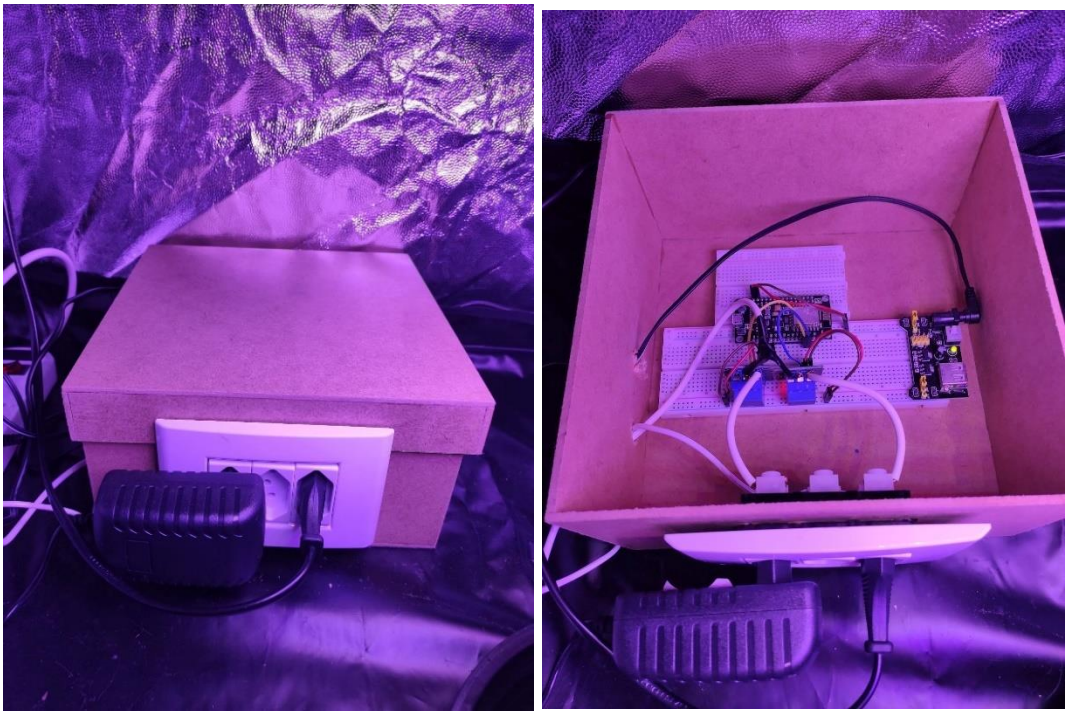
No centro da estufa está o vaso de cebolinha com a mangueira de irrigação e o sensor instalados. A cebolinha estava sendo cultivada há uma semana no momento da Figura 46, assim é possível ver as primeiras plantas nascendo. A mangueira é feita de um micro tubo de PVC próprio para irrigação e está ligada a uma bomba do lado de fora da estufa. Sua extremidade é formada por um círculo com várias perfurações por onde a água é distribuída de maneira uniforme pelo vaso. O sensor fica espetado na terra e realiza medições de temperatura, luminosidade, umidade e condutividade elétrica do solo.

Figura 46 - Sensor, mecanismo de irrigação e resultados do crescimento

Fonte: os autores.

Para a construção do módulo de atuadores, foi utilizada uma caixa de MDF, um conjunto de três tomadas embutíveis, um *plug* de tomada, cabos de energia, protoboards, fonte e cabos de conexão para a *protoboard*, um módulo ESP8266 e dois módulos relés. Foi feita uma abertura na caixa para o encaixe das tomadas e dois furos para a passagem dos cabos de alimentação da fonte da protoboard e das tomadas (Figura 47 à esquerda). Internamente, os dois relés foram ligados em paralelo à fonte de alimentação da rede elétrica através das tomadas para o controle da placa de LED e da bomba de irrigação. O circuito de chaveamento dos relés e o microcontrolador estão ligados à fonte de alimentação da protoboard. O pino de acionamento dos relés está conectado aos pinos GPIO do microcontrolador para o controle programático do acionamento da iluminação e da irrigação (Figura 47 à direita).

Figura 47 - Módulo de atuadores



Fonte: os autores.

A placa de LED está suspensa por um suporte regulável no topo da estufa, permitindo aproximar a fonte de luz do vaso de planta. Em testes realizados, a intensidade luminosa medida com a placa próxima da planta foi até cinco vezes maior que a intensidade luminosa medida com a placa no topo da estufa. E ao lado externo da estufa, se encontram o reservatório de água e a bomba responsável pela irrigação da planta (Figura 48 à direita). Ela está interligada ao reservatório por uma mangueira e realiza o bombeamento da água para a estufa quando acionada pelo relé que controla a irrigação.

Figura 48 - Painel de luz e reservatório de água e bomba de irrigação

Fonte: os autores.

Com todos os elementos montados (Figura 49), foi feita a instalação elétrica da estufa através de extensões. Elas estão sendo mantidas do lado interno da estufa para serem protegidas do sol e de respingos de água, pois ela está instalada no quintal. Assim, alimentando o dispositivo de borda, o módulo de atuadores, a placa de LED e a bomba de água, a montagem está completa.

Figura 49 - Resultados da integração da horta com o sistema

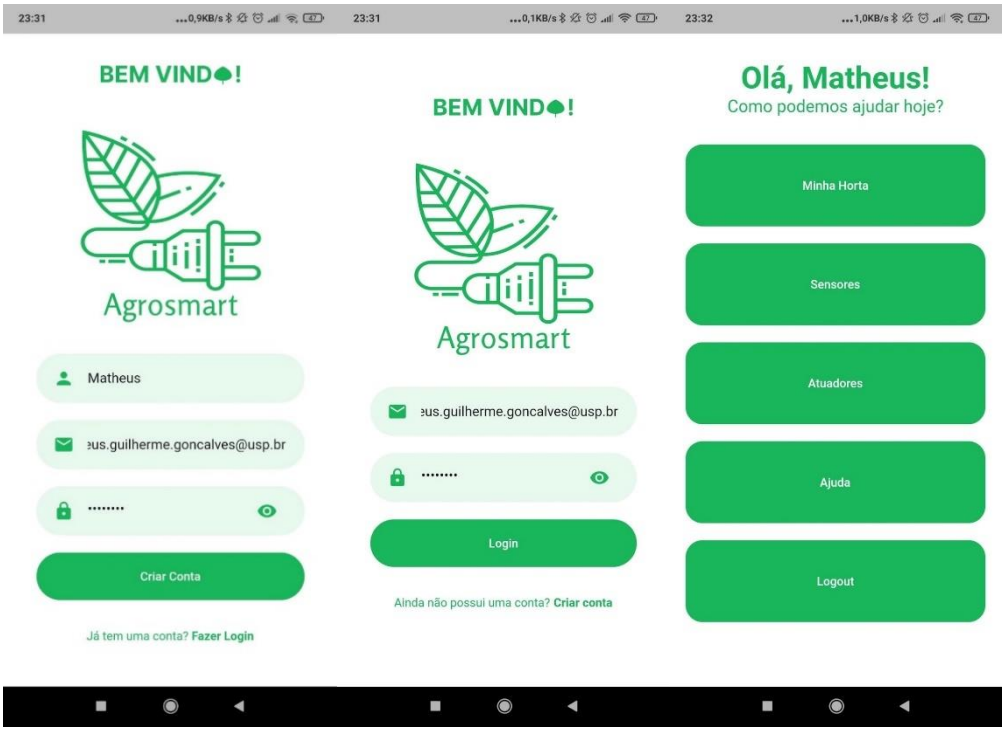
Fonte: os autores.

7.1.2. Aplicação Móvel

Com o sistema embarcado implementado e a montagem feita, foi possível cadastrar a horta no aplicativo feito para o monitoramento da plantação através do celular.

Primeiramente, o usuário deve fazer um cadastro na plataforma e realizar o login para poder utilizar o sistema (Figura 50 à esquerda e meio). Após o login, ele é apresentado com as principais funcionalidades do aplicativo, representadas pelas seções Minha Horta, Sensores e Atuadores (Figura 50 à direita).

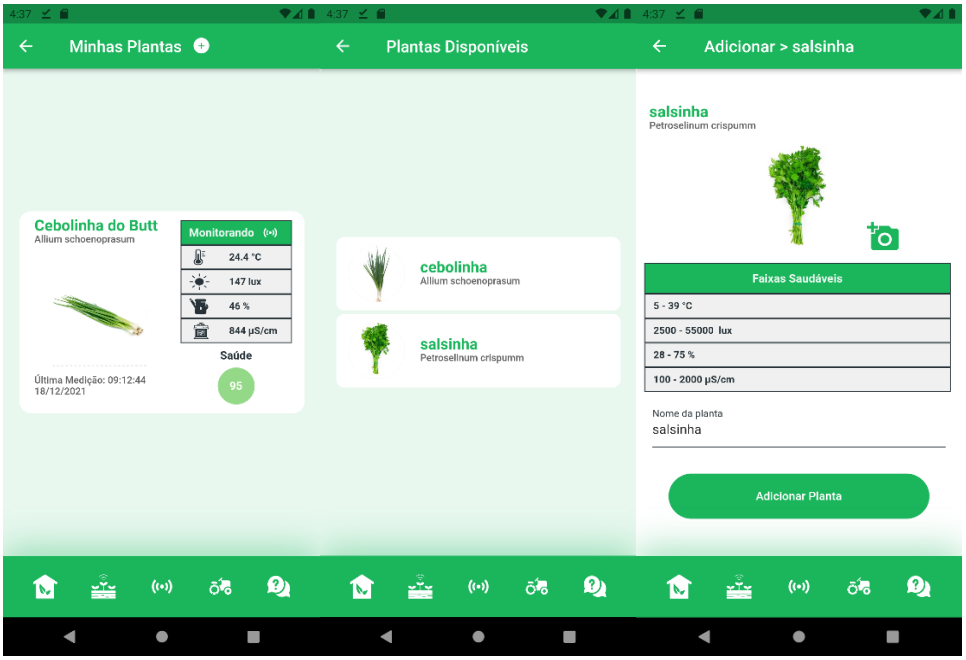
Figura 50 - Fluxo de autenticação



Fonte: captura de tela do aplicativo Agrosmart.

Na seção “Minha horta” (Figura 51), é possível visualizar as plantas cadastradas, cadastrar uma nova planta, ver os detalhes de uma planta específica e associar um sensor e um atuador à planta.

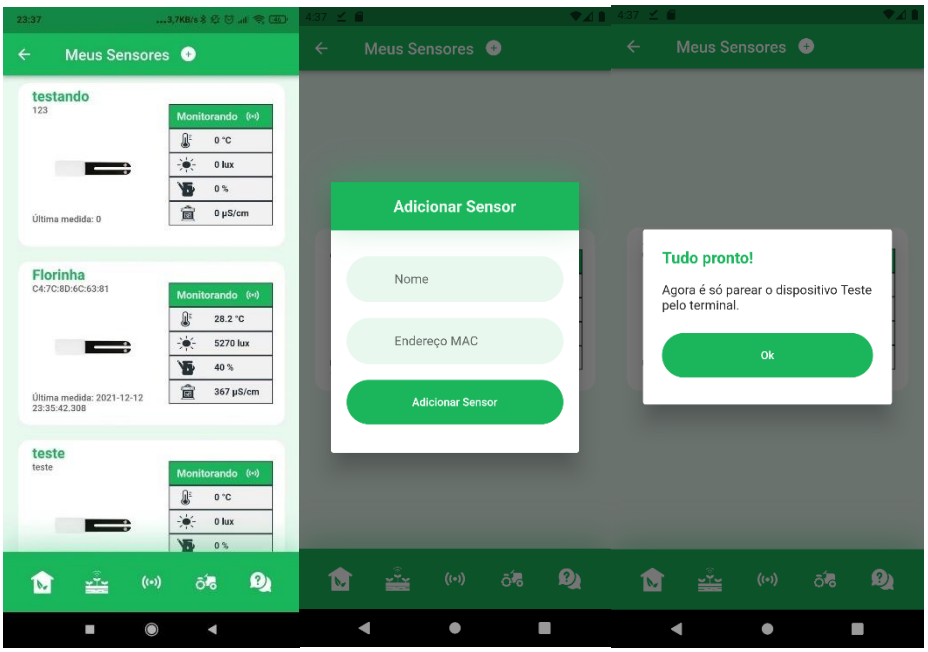
Figura 51 - Fluxo de criação de planta



Fonte: captura de tela do aplicativo Agrosmart.

Na seção Sensores (Figura 52), é possível visualizar os sensores cadastrados e suas medições quando ativadas e cadastrar um novo sensor.

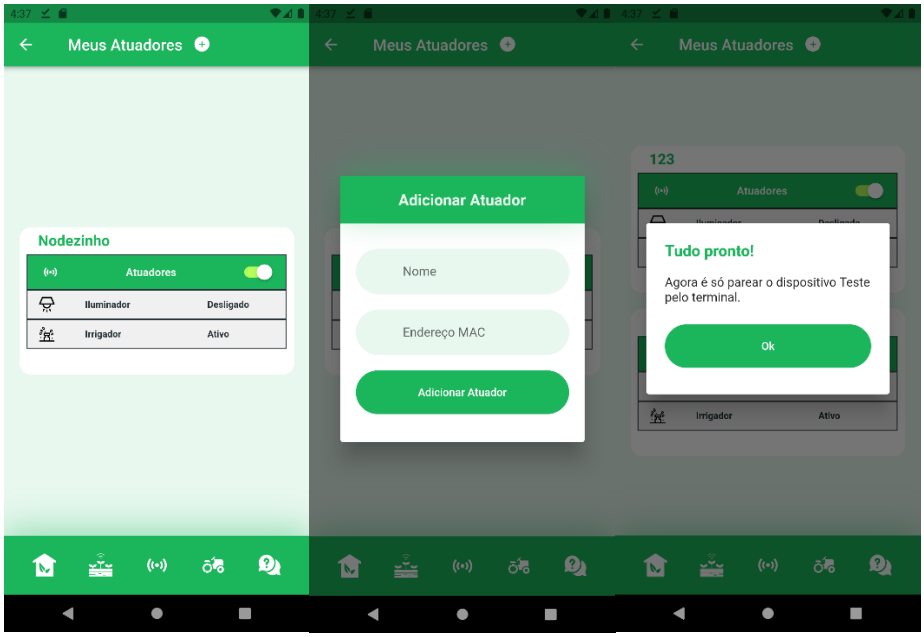
Figura 52 - Fluxo de cadastro de sensor



Fonte: captura de tela do aplicativo Agrosmart.

Na seção de Atuadores (Figura 53), é possível visualizar os atuadores cadastrados e os estados de cada módulo (iluminação e irrigação) e cadastrar um novo atuador.

Figura 53 - Fluxo de cadastro de atuador

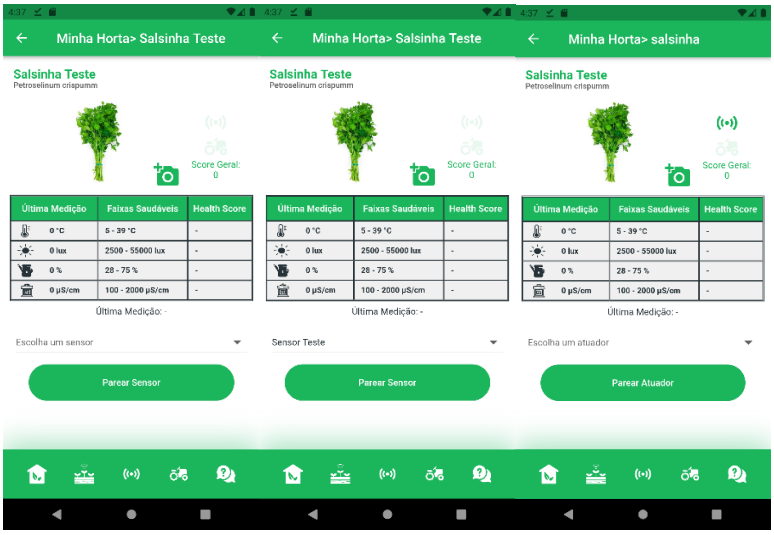


Fonte: captura de tela do aplicativo Agrosmart.

Após o cadastro do protótipo no aplicativo e a associação entre a planta, o sensor e o módulo de atuadores, é possível visualizar as medições do sensor e o estado dos atuadores, conforme ilustram as Figuras 52 e 53.

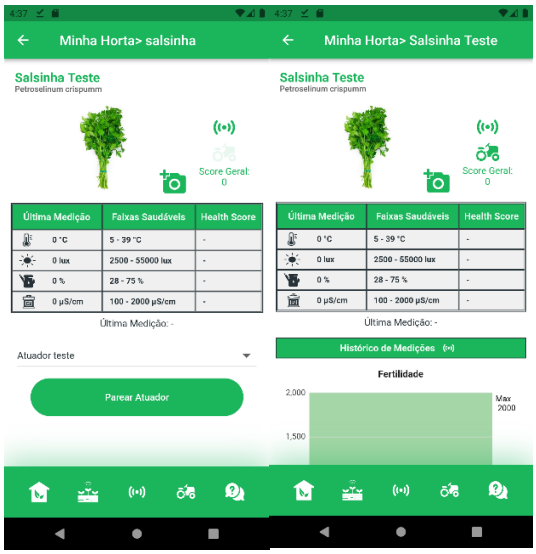
Após cadastrar os sensores e atuadores (Figuras 54 e 55), e realizado o pareamento dos mesmos a partir dos scripts do dispositivo de borda, é possível visualizá-los automaticamente na aplicação. São disponibilizados dados dos históricos de medições de maneira propícia para cada fluxo.

Figura 54 - Fluxo de pareamento do sensor



Fonte: captura de tela do aplicativo Agrosmart.

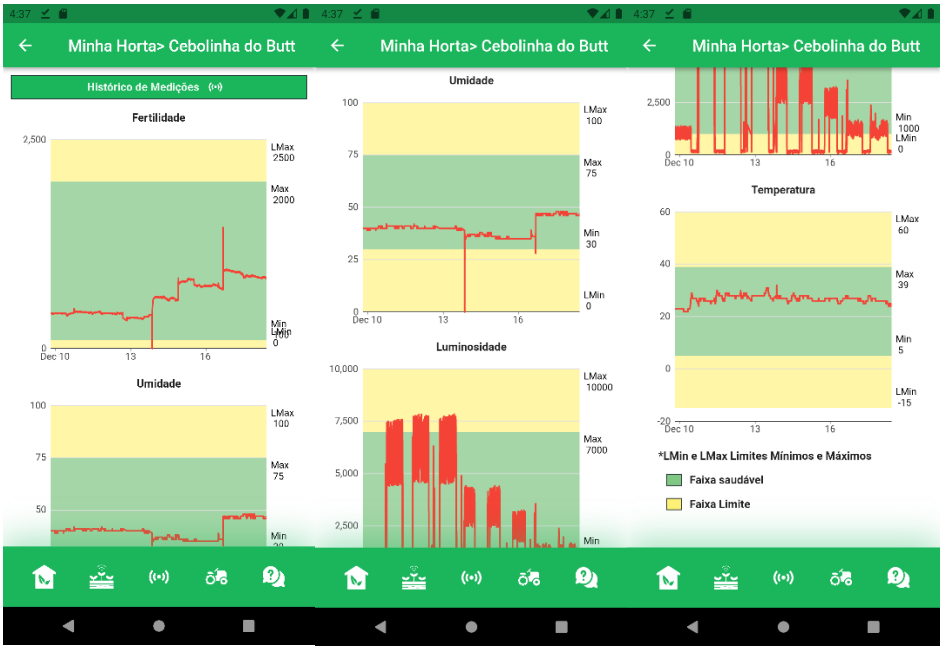
Figura 55 - Fluxo pareamento de Atuador



Fonte: captura de tela do aplicativo Agrosmart.

No fluxo da planta (Figura 56), encontra-se gráficos com o histórico de medições de cada sensor junto aos intervalos essenciais sendo em verde a margem ideal enquanto em amarelo tem-se os limites aceitáveis, onde quanto mais longe da região verde menor seu *score*.

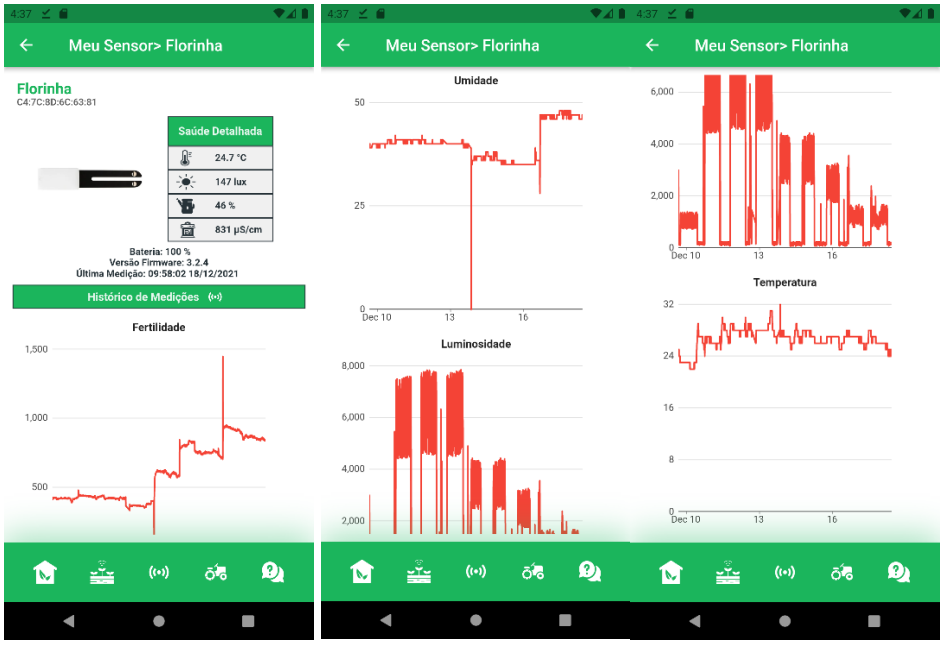
Figura 56 - Históricos de medição para Planta



Fonte: captura de tela do aplicativo Agrosmart.

Para os sensores os históricos têm gráficos semelhantes sem os limites ideais. Acarretam uma melhor escala como visto na Figura 56:

Figura 57 - Histórico de medição do Sensor



Fonte: captura de tela do aplicativo Agrosmart.

E por fim, o histórico de ativações dos atuadores estão disponíveis em formato de tabela (Figura 57):

Figura 58 - Histórico de medições do Atuador

Fonte: captura de tela do aplicativo Agrosmart.

8. CONSIDERAÇÕES FINAIS

8.1. Conclusões do projeto de formatura

8.1.1. Matheus

Este projeto de formatura procurou abordar de maneira consistente muitos dos tópicos aprendidos ao longo do curso de Engenharia da Computação. Para sua elaboração e execução, foi preciso utilizar vários conceitos aprendidos em matérias como engenharia de software e de sistemas de computação, arquitetura de computadores, de redes e de sistemas operacionais, sistemas embarcados, interação humano computador e muitas outras. A experiência com desenvolvimento obtida durante os módulos de estágio também foi decisiva para a completude deste projeto. Assim, pode-se dizer que este projeto é fruto de toda a experiência acadêmica e profissional adquirida durante o curso de graduação em Engenharia da Computação na Escola Politécnica da USP.

O projeto de formatura também exigiu uma forte colaboração e entendimento entre os membros da equipe e o orientador, principalmente no contexto da pandemia, onde não foi possível realizar reuniões presenciais para o planejamento e execução do projeto. Desta forma, foi possível exercitar uma prática cada vez mais comum em nossa profissão, que é o trabalho remoto em equipe. Pelo andamento e resultados obtidos, pode-se dizer que a equipe soube administrar de maneira muito satisfatória o trabalho remoto conjunto e pôde realizar um projeto de conclusão de curso com qualidade e chegar em resultados satisfatórios para o protótipo.

Particularmente, a implementação do sistema embarcado e sua integração com a nuvem foi um processo de muito aprendizado para mim. Eu já tinha tido experiência com a implementação de sistemas Web, porém nunca tinha feito um projeto embarcado conectado à nuvem e integrado com outros sistemas. Assim, o desenvolvimento do protótipo do sistema embarcado foi a parte mais complexa, porém mais proveitosa para mim, pois estava fora de minha zona de conforto e contribuiu para a expansão de meus conhecimentos na área da computação.

Como já foi reforçado, o tema deste projeto é muito abrangente e desafiador, porém os benefícios do mesmo para a sociedade são numerosos e multifacetados. Assim, pretende-se continuar a implementação do sistema aqui proposto, visando a construção de um produto com maturidade comercial que ajude a impulsionar a agricultura urbana de porte doméstico no Brasil.

8.1.2. Paulo

O projeto foi uma oportunidade para colocar em prática várias áreas exploradas durante o curso assim como em experiências profissionais, servindo como uma revisão final para todos os trabalhos na poli. Foi muito interessante também poder aplicar uma ideia que nasceu de um tema livre no laboratório de sistemas digitais. Sendo um sistema completo, envolvendo sistema embarcado, camada de nuvem e aplicação *mobile*, foi muito interessante também trabalhar em grupo com cada um tomando frente onde tinha mais experiência e se sentia mais à vontade, e compartilhando esse conhecimento.

Um trabalho em grupo entre amigos que também já foram colegas de trabalho em estágio já contava com um conhecimento de como cada um tinha seu método de trabalho, entretanto, este projeto evoluiu ainda mais essa sinergia entre nós que foi posto à prova no momento de integração dos sistemas. Sendo a interface *mobile* um desafio para todos nós por não termos experiência na área, foi particularmente um desafio a mais para mim. Tomar essa frente, aprender do zero e aplicar todo esse conhecimento foi difícil, mas conseguir chegar nos resultados foi muito satisfatório.

Continuar melhorando o projeto com certeza estará em meus planos, e criar um produto sustentável que contribua para sociedade está em minhas metas de vida.

8.1.3. Rafael

A escolha por um projeto de engenharia que tenha várias frentes de desenvolvimento e que é voltado para pessoas se mostrou bastante acertada e me aproximou de campos de conhecimento pouco explorados durante a graduação: idealização de interface móvel, execução de um sistema embarcado e infraestrutura de *backend*. São áreas que vão contribuir muito para o andamento da minha carreira de engenheiro de computação e que tive contato com a teoria e a prática.

O desenvolvimento de todas as frentes do projeto foi bastante fluido e teve interação dos três integrantes do grupo. A integração entre as frentes de trabalho era uma preocupação no início do projeto, já que o sistema só se provaria com as partes funcionando de maneira inde-

pendente e integradas entre si, porém durante o período de integração foi onde apareceram algumas necessidades que trouxeram melhorias significativas para o projeto e que nossas ideias realmente se provaram coerentes.

Poder trabalhar em grupo trouxe a possibilidade de optar por um projeto grande e também a possibilidade de atuar em um projeto de engenharia da maneira que se faz profissionalmente, com características como: trabalho em grupo, diferentes opiniões sobre o mesmo tema, divisão de tarefas, prazos, validação de funcionalidades, testes e documentação. O grupo em si foi naturalmente se desenvolvendo durante a graduação e a escolha dos integrantes foi bastante natural, já que nos conhecemos na sala de aula, fizemos diversos trabalhos da poli, trabalhamos juntos durante um módulo de estágio do regime quadrimestral e criamos uma amizade profissional e pessoal que se mostrou bastante proveitosa para o projeto, pois não enfrentamos problemas durante toda realização deste trabalho de conclusão de curso.

8.2. Contribuições

Este projeto tem potencial para contribuir no auxílio do cultivo de pequenas hortas oferecendo domínio das informações essenciais para o cultivo como umidade, luminosidade, temperatura e fertilidade do solo. O protótipo conta também com a automatização da irrigação que pode ser um dos principais problemas enfrentados por usuários não familiarizados com plantações, assim como o fator de iluminação, que para ambientes internos acaba sendo negligenciado.

Este projeto aborda também questões sociais e econômicas acerca da agricultura no Brasil, onde a conscientização sobre os benefícios de práticas de agricultura urbana também deve ser levada em conta.

8.3. Perspectivas de continuidade

Ao definir e especificar o escopo do projeto, o grupo levantou e priorizou os fatores mais importantes para o protótipo, porém estas decisões não diminuem as propostas que ficaram fora do escopo. Assim como a ideia já existe desde os primeiros módulos acadêmicos, o projeto

continuará a ser implementado dentro das margens de custo grátis das plataformas utilizadas como AWS, *Firebase* etc.

Pode-se listar então, sem ordens de prioridade, novos recursos, por exemplo:

- Automatização do pareamento com os dispositivos embarcados via aplicação;
- Implementação de fatores de correção da ventilação do sistema e controle de temperatura;
- Implementação da automatização dos fatores de fertilidade;
- Substituição do Xiaomi Flora por um sensor totalmente fabricado pelo grupo;
- Criar uma biblioteca de plantas com os parâmetros de cultivo para fornecer aos usuários uma lista de espécies abrangente no fluxo de cadastro de plantas;
- Melhorias de usabilidade, responsividade e projeto da interface;
- Detalhamento sobre informações das plantas assim como a de usuários:
- Criação de dashboards para tornar a leitura de dados mais amigável;
- Refatorações de código;
- Pesquisa mais profunda sobre técnicas de plantio doméstico, de preferência com profissionais da área, para melhorar os processos de monitoramento e controle do sistema;
- Adicionar uma câmera ao protótipo para registrar o crescimento da planta, possibilitar ao usuário o acompanhamento remoto de seu ambiente de cultivo e mesmo utilizar algoritmos de visão computacional e aprendizado de máquina para tentar determinar a fase de crescimento da planta, período ideal para colheita e possíveis problemas de saúde através de seu aspecto físico;
- Adaptação do protótipo desenvolvido para um produto comercialmente viável.

REFERÊNCIAS

AMAZON **API Gateway**. Disponível em: <https://aws.amazon.com/pt/apigateway/>. Acesso em 30 jul. 2021.

_____. **AWS**. O que é computação em nuvem? Disponível em: <https://aws.amazon.com/pt/what-is-cloud-computing/>. Acesso em 25 ago. 2021.

_____. **CloudFormation**. Disponível em: <https://aws.amazon.com/pt/cloudformation/>. Acesso em 30 jul. 2021.

_____. **DynamoDB**. Disponível em: <https://aws.amazon.com/pt/dynamodb/>. Acesso em 30 jul. 2021.

_____. **GreenGrass**. Disponível em: <https://aws.amazon.com/pt/greengrass/>. Acesso em 30 jul. 2021.

_____. **IoT**. Disponível em: <https://aws.amazon.com/pt/iot/>. Acesso em 30 jul. 2021.

_____. **Lambda**. Disponível em: <https://aws.amazon.com/pt/lambda/>. Acesso em 30 jul. 2021.

_____. **Serverless**. Disponível em: <https://aws.amazon.com/pt/serverless/>. Acesso em 30 jul. 2021.

ARTAC, Matej; BOROVSAK, Tadej; NITTO, Elisabetta Di; GUERRIERO, Michele; TAMBURRI, Damian Andrew. **DevOps**: Introducing Infrastructure-as-Code. Politecnico di Milano. 2017. Disponível em: <https://www.researchgate.net/publication/318124847_DevOps_Introducing_Infrastructure-as-Code>. Acesso em 20 jul. 2021.

CABALLERO, Luiza. **Aeroponia**: como funciona essa técnica. Disponível em: <https://www.ecycle.com.br/aeroponia/>. Acesso em 30 jul.2021.

CUGNASCA, Carlos Eduardo; FARIA, José Sinézio Rebello; NISHIDA, Julio Cesar Candia; Dias, Eduardo Mario; Oliveira, Jairo Cardoso e. **Fazendas Verticais**: descrição, panorama mundial e perspectivas para o futuro. ENGEMA Disponível em: <http://engemausp.submissao.com.br/22/arquivos/29.pdf>. Acesso em: 15 jul. 2021.

EMBARCADOS, Equipe. **Sistema Embarcado**: O que é? Qual a sua importância? 2013. Disponível em: <https://www.embarcados.com.br/sistema-embarcado/>. Acesso em 10 dez 2021.

EMBRAPA. **Aquaponia residêncial**. Disponível em: <https://www.embrapa.br/e-campo/aquaponia-residencial>. Acesso em 30 jul. 2021.

GOOGLE. **Firebase**. Disponível em: <https://firebase.google.com/>. Acesso em 30 jul. 2021.
_____. **Flutter**. Disponível em: <https://docs.flutter.dev/>. Acesso em 30 jul. 2021.

LEBLANC, R. What you should know about vertical farming - is it the future of agriculture? Sustainable Businesses, p. LeBlanc, R. (n.d.). No Title. Retrieved April 19, 2019.

LEGNAIOLI, Stela. **Agricultura Urbana Orgânica**: entenda por que é uma boa ideia. ECycle. Disponível em: <https://www.ecycle.com.br/agricultura-urbana/>. Acesso em: 15 jun. 2021.

MACHADO, Altair Toledo; MACHADO, Cyntia Torres de Toledo. **Documentos 48**: Agricultura Urbana. Embrapa, Planaltina, jun 2002. Disponível em: <https://www.agriculturaurbana.org.br/>. Acesso em: 15 jun. 2021.

MONGOOSE OS**. Who we are**. Disponível em: <https://mongoose-os.com/about.html>. Acesso em 30 jul. 2021.

MORAES, Rodrigo Fracalossi de. **Agrotóxicos no Brasil**: padrões de uso, política da regulação e prevenção da captura regulatória. IPEA. Disponível em: <http://repositorio.ipea.gov.br/bitstream/11058/9371/1/td 2506.pdf>. Acesso em: 20 jul. 2021.

NODE.JS. **About Node.Js.** Disponível em: <https://nodejs.org/en/about/>. Acesso em 30 jul. 2021.

PIRES, Vicente Chiaramonte. **Agricultura Urbana como Fator de Desenvolvimento Sustentável**: Um Estudo na Região Metropolitana de Maringa. Revista Pesquisa & Debate, São Paulo, 2016.

RASPBERRY PI. **About us.** Disponível em: <https://www.raspberrypi.org/about/>. Acesso em 30 jul. 2021.

RIBEIRO, Cassiano**. Pandemia de Covid-19 aumenta interesse dos brasileiros em jardinagem e horta urbana.** Revista Globo Rural, 2020. Disponível em: < https://revistagloborural.globo.com/Colunas/Cassiano-Ribeiro/noticia/2020/06/pandemia-de-covid-19-aumenta-interesse-dos-brasileiros-em-jardinagem-e-horta-urbana.html> . Acesso em: 20 jul. 2021.

SERVERLESS. **About.** Disponível em: <https://www.serverless.com/about>. Acesso em 30 jul. 2021.

SNIS, Sistema Nacional de Informações sobre Saneamento. **Do SNIS ao SINISA**: Informações para planejar o abastecimento de água. Ministério do Desenvolvimento Regional, Brasília, dez. 2020. Disponível em <http://www.snis.gov.br/downloads/cadernos/2019/DO_SNIS_AO_SINISA_AGUA_SNIS_2019.pdf>. Acesso em: 20 jul. 2021.

SUPERINTENDÊNCIA TÉCNICA DA CNA E CEPEA. **PIB do Agronegócio cresce 3,81% em 2019**. CNA Brasil. Disponível em: <https://www.cnabrasil.org.br/boletins/pib-do-agronegocio-cresce-3-81-em-2019>. Acesso em: 15 jun. 2021.

TRILLES, Sergio; PÉREZ, Alberto González; HUERTA, Joaquín. **An IoT Plataform Based on Microservices and Serverless Paradigms for Smart Farming Purposes**. Institute of New Imaging Technologies, 2020.

UFU, ICIAG. **Hidroponia**. Disponível em: <http://www.fruticultura.iciag.ufu.br/hidropo.htm>. Acesso em 30 jul. 2021.