INSTITUTO FEDERAL DE SANTA CATARINA

OSVALDO DA SILVA NETO

Sistema de Monitoramento Microclimático Para o Setor do Agronegócio Utilizando Tecnologia Lorawan

São José - SC

março/2022

SISTEMA DE MONITORAMENTO MICROCLIMÁTICO PARA O SETOR DO AGRONEGÓCIO UTILIZANDO TECNOLOGIA LORAWAN

Trabalho de conclusão de curso apresentado à Coordenadoria do Curso de Engenharia de Telecomunicações do campus São José do Instituto Federal de Santa Catarina para a obtenção do diploma de Engenheiro de Telecomunicações.

Orientador: Mário Noronha Neto, Dr

Coorientador: Hamilton Avinco

São José - SC

março/2022

Osvaldo da Silva Neto
Sistema de Monitoramento Microclimático Para o Setor do Agronegócio Utilizando Tecnologia Lorawan/ Osvaldo da Silva Neto. – São José - SC, março/2022
Orientador: Mário Noronha Neto, Dr

Monografia (Graduação) – Instituto Federal de Santa Catarina - IFSC
Campus São José
Engenharia de Telecomunicações, março/2022.

1. monitoramento climático. 2. Lorawan. 3. iot. I. Mário de Noronha Neto. II. Instituto Federal de Santa Catarina. III. Campus São José. IV. Desenvolvimento de Uma Solução de Monitoramento de Dados MicrometeorológicoPara o Setor do Agronegócio Brasileiro

OSVALDO DA SILVA NETO

SISTEMA DE MONITORAMENTO MICROCLIMÁTICO PARA O SETOR DO AGRONEGÓCIO UTILIZANDO TECNOLOGIA LORAWAN

Este trabalho foi julgado adequado para obtenção do título de Engenheiro de Telecomunicações, pelo Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, e aprovado na sua forma final pela comissão avaliadora abaixo indicada.

São José - SC, 30 de março de 2021:

Mário Noronha Neto, Dr, Dr.
Orientador
Instituto Federal de Santa Catarina

Richard Demo Souza, Dr.
Universidade Federal de Santa Catarina

Arliones Stevert Hoeller Jr, Dr.
Instituto Federal de Santa Catarina

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Agradeço a minha esposa, Rafaela, pelo apoio e esforço nesta etapa tão importante, só ela sabe o quanto foi batalhado para completar este desafio.

À meus pais Osvaldo e Sandra, por acreditarem e nunca desistirem de mim, eles sabem que ser um engenheiro é um sonho de infância prestes a ser realizado.

À minha sogra Elisabeth pelo apoio e ajuda em todos os momentos difíceis, sem ela talvez este sonho não teria sido realizado.

Ao meu orientador Professor Mario Noronha Neto e ao coorientador Hamilton Marques Avinco, que me auxiliaram no desenvolvimento do trabalho de conclusão de curso.

A todo o corpo docente do IFSC, sejam professores ou qualquer outro, pelo acolhimento e apoio durante o curso.

Aos professores que com todo o carinho sempre me atenderam e sanaram todas as dúvidas, nunca me deixando sem respostas.

RESUMO

O agronegócio no Brasil representa uma fatia importante do Produto Interno Bruto (PIB) brasileiro mesmo assim ele sofre de uma carência tecnológica muito grande. Pensando nisso, a proposta apresenta desenvolver uma solução para auxiliar o agricultor na sua tomada de decisões. A ideia consiste em desenvolver um sistema onde serão capturados dados considerados importantes para desenvolvimento de um determinado cultivo e apresentá-los de forma amigável ao agricultor. O sistema irá utilizar os dados capturados de uma estação meteorológica para formar uma base de dados que poderão ser utilizados para futuros projetos. Com os dados coletados será desenvolvido uma interface gráfica para apresentar ao agricultor uma situação em tempo real ou até mesmo que remeta ao passado, fazendo com que o mesmo possa tomar suas decisões baseada em dados realísticos. Vale ressaltar que os dados coletados para desenvolvimento do sistema serão umidade do ar e solo, temperatura, incidência de raios solares, direção e velocidade do vento e pressão atmosférica. A proposta foi desenvolvida com uma estação micrometeorológica comercial e toda a comunicação com a estação será feita utilizando a tecnologia Lora®.

Palavras-chave: monitoramento climático. Lorawan. dados meteorológicos. IOT.

ABSTRACT

Agribusiness in Brazil represents an important part of the Brazilian PIB, yet it suffers from a very large technological shortage. With this in mind, the proposal presents the development of a solution to assist the farmer in his decision-making. The idea is to develop a system where data considered important for the development of a particular crop will be captured and presented in a friendly way to the farmer. The system will use data captured from a weather station to form a database that can be used for future projects. With the collected data, a graphical interface will be developed to present the farmer with a situation in real time or even that refers to the past, allowing him to make his decisions based on realistic data. It is worth mentioning that the data collected for the development of the system will be air and soil humidity, temperature, incidence of sunlight, wind direction and speed and atmospheric pressure. The proposal was developed with a commercial micrometeorological station and all communication with the station will be done using Lora® technology.

Keywords: monitoring. Lorawan. weather data. IOT.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

A agroindústria é um conjunto de atividades relacionadas à transformação de matérias-primas agropecuárias originadas da agricultura, pecuária, aquicultura ou silvicultura (TOCANTINS, 2020), seus produtos tem como principal característica servir de alimento para o consumidor final.

A agricultura é um dos principais eixos da economia no Brasil, onde segundo IBGE (2020) o setor cresceu 2,4% somente no ano de 2019 e colaborou com 5,2% do PIB. Sendo um país com dimensões continentais e uma variedade climática diversificada, o setor do agronegócio brasileiro proporciona diversas oportunidades de negócios e geração de riquezas ao país.

Os destaques nacionais estão para os grãos com 232,6 milhões de toneladas e frutas com 43,1 milhões de toneladas no ano de 2019/2020 (EMBRAPA, 2021). Segundo IBGE (2018) no censo Agro realizado em 2017 as lavouras com maiores rendimentos foram a soja, seguida da cana de açúcar, o milho e o café. Juntas estas três culturas somam uma área de aproximadamente 19,5 milhões de hectares e mais de 406,4 milhões de toneladas produzidas.

Os dados dos Censos Agropecuários realizados entre 2006 e 2007 mostraram que o Brasil se destacou na produtividade, colaborando com uma taxa de crescimento médio anual de aproximados 4,3%, superior a outros países, tais como: Argentina (2,7%), Chile (3,1%), Estados Unidos (1,9%) e China (3,3%) (FILHO; GASQUES, 2020).

Mesmo tendo o agro como um dos principais colaboradores do PIB, por muitos anos o setor sofreu com a falta de tecnologia.

Apesar da mudança de cultura por parte dos agricultores e empresários com relação à inserção de tecnologias no agro, ainda existe um grande desafio para "conectar"o setor. Tal desafio é a falta infraestrutura de comunicação para cobrir a imensa área rural brasileira.

Quaisquer melhorias em telecomunicações necessitam de um investimento elevado, este fator se torna um dos principais empecilhos na busca por investimentos em tecnologia no setor agrícola. No ano de 2021 as áreas urbanas encontravam-se totalmente cobertas, sendo que 91,2% possuem sinal 3g ou 4g disponível. Este dado não reflete a situação no campo, a área rural contava apenas com 23% de área coberta. O problema da cobertura de algum sinal de rede móvel traz consequências negativas para o setor, impactando diretamente no seu desenvolvimento (ANATEL, 2020).

As redes Low Power Wide Area Networks (LPWAN) nos ajudam a solucionar este problema da conectividade em IOT, com tecnologias que permitem comunicações a longas distâncias, principalmente em áreas rurais onde o alcance atinge aproximadamente 12 km.

A proposta do projeto consiste em oferecer uma ferramenta que integrará a rede de sensores à internet. O monitoramento vai proporcionar uma ideia concreta do que vem acontecendo no dia a dia da região através da leitura constante dos dados oferecidos pela estação meteorológica. A ideia do projeto é implementar uma interface amigável, onde o agricultor através de poucos cliques e sem sair de casa possa observar a situação em que a região se encontra. O principal objetivo será oferecer uma melhora nos índices de produtividade e redução nos custos operacionais através da leitura e estudo dos dados.

1.1 Objetivo Geral

Consiste em desenvolver uma solução de monitoramento de dados microclimáticos para setor do agronegócio utilizando uma estação meteorológica comercial com a tecnologia LoRaWAN.

1.2 Objetivos Específicos

- Coletar informações da estação meteorológica;
- Implementar uma comunicação LoRaWAN por meio da plataforma de comunicação NLT(Everynet) ou privada.
- Desenvolver um sistema para armazenamento de dados;
- Desenvolver uma Interface de Programação de Aplicação (API) para consumo destes dados;
- Implementar uma interface gráfica onde o usuário possa analisar os dados capturados;
- Propor um modelo de negócio para a solução.

1.3 Organização do texto

O trabalho será dividido em seis capítulos, no Capítulo 1, contendo conteúdos introdutórios, no Capítulo 2 serão abordados temas que servirão como base no decorrer da pesquisa e por fim, no Capítulo 3 será apresentado todo o desenvolvimento do projeto, no Capítulo 4 será descrito todos os procedimento para implementar o projeto, no seção 4.3 serão apresentados os resultados de uma breve análise do projeto e por fim no Capítulo 5 será descrito as considerações finais do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos acerca da temática deste trabalho e as tecnologias necessárias aplicadas para desenvolvimento da solução.

2.1 Agricultura de Precisão (AP)

A agricultura aparece como o resultado de um longo processo de evolução que afetou muitas sociedades, esta expansão permitiu um forte crescimento da população mundial, ocasionando assim uma necessidade de otimização ainda maior.

A Agricultura de Percisão (AP) é novidade na atividade agrícola brasileira e seu princípio nada mais é do que aplicar a tecnologia no campo.

Em 2012, o Ministério da Agricultura, Pecuária e Abastecimento (MAPA), ao instituir a Comissão Brasileira de Agricultura de Precisão (CBAP), definiu a Agricultura de Precisão como "um sistema de gerenciamento agrícola baseada na variação espacial e temporal da unidade produtiva e visa ao aumento de retorno econômico, à sustentabilidade e à minimização do efeito ao ambiente".

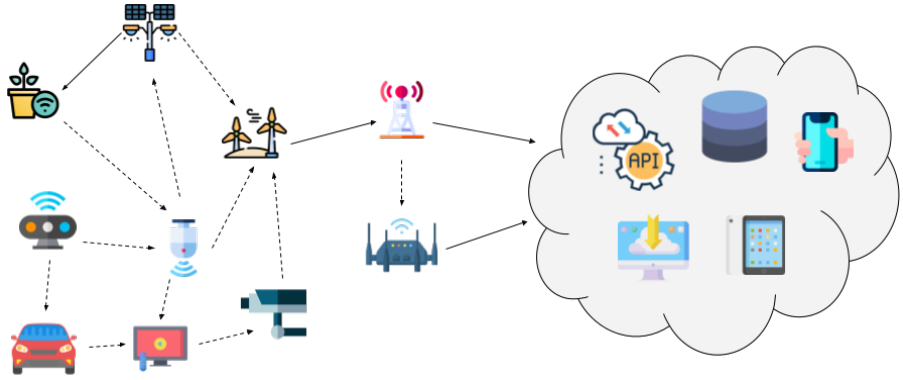
Por exigir um nível de tecnologia avançado a aplicabilidade deste conceito ainda é novidade na maioria das agriculturas no Brasil. Entretanto, o avanço é inegável, houve amadurecimento e o mercado se estabeleceu trazendo os resultados que são sustentados cientificamente.(BERNARDI et al., 2014)

A demanda para este serviço surgiu com a necessidade de produzir mais alimentos em um menor tempo, diminuir a mão de obra e recursos utilizados e preparar-se para possíveis mudanças climáticas e ataques de pragas, vindo a se tornar um grande aliado para o campo.

2.1.1 Agricultura de Precisão (AP) e Internet das Coisas (IOT)

O avanço da tecnologia trouxe mudanças significativas para a humanidade, melhorando consideravelmente nossas experiências junto ao mundo digital, no agro não é diferente. Um dos fatores principais para este avanço foi a possibilidade de conectar objetos do cotidiano ao mundo digital. A Figura 1 ilustra uma ideia geral da arquitetura IOT.

Figura 1 – Ilustração rede IOT



Fonte: Próprio autor utilizando imagens de Flaticon <https://www.flaticon.com/br/>

Para Lamparelli (2016) AP é o conjunto de técnicas que permitem o gerenciamento localizado dos cultivos.

> O uso de ferramentas adequadas da agricultura de precisão contribui para a diminuição de perdas na agricultura. Por meio de tais ferramentas de agricultura de precisão é possível obter dados provenientes da análise da propriedade subdividida em pequenas áreas (informações geográficas georreferenciadas), relativos a irrigação, propriedades físicas do solo, necessidade de aplicação de defensivos. Quanto mais subdividida a propriedade rural, mais útil será a informação georreferenciada. O controle das variáveis que influenciam o cultivo depende do maior detalhamento das informações. (LAMPARELLI, 2016)

A junção de IOT e tecnologia no campo vem proporcionando vantagens, auxiliando o agricultor a minimizar as perdas e potencializando os ganhos econômicos e ambientais na lavoura. Essa ferramenta permite uma exploração mais controlada e precisa de insumos, aumentando a lucratividade e reduzindo os impactos.

2.2 Low Power Wide Area Networks (LPWAN)

As LPWAN são tecnologias sem fio que possuem a característica de trabalharem em redes de máquinas para máquinas. Possuem a característica de conectar dispositivos que necessitam de uma pequena taxa de transmissão sem se preocupar com a distância e tendo a capacidade de manter-se ativo por um longo período de tempo, por apresentarem características singulares as LPWAN se adequam muito bem a sistema de IOT.

A rede de IOT envolve um conjunto de requisitos que competem entre si, como custo de ponto final, consumo de energia, largura de banda, latência, densidade de conexão, custo operacional, qualidade de serviço e faixa de frequência da conexão. Atualmente, nenhuma tecnologia de rede otimiza tudo isso de maneira única, mas as novas tecnologias de rede de IOT fornecerão mais opções e flexibilidade. (GISERMAN; GALVES; JAOLINO, 2021)

Com o avanço do IOT surgiram novas tecnologias LPWAN que viabilizam a comunicação, dentre elas podemos citar: LoRa, SigFox, NB-Iot, LTE-M e outras.

2.2.1 Arquitetura

Na Figura 2 temos um modelo da arquitetura de rede LPWAN na qual *gateways* são a ponte entre os dispositivos (sensores) e os servidores de internet. Os *gateways* se ligam aos servidores através de uma conexão simples de Protocolo de Internet (IP), enquanto os dispositivos usam a rede sem fio para falar com um ou mais *gateways*. As comunicações geralmente são bi-direcionais, mas também permitem mensagens de *multicast* para upgrade e outras mensagens *broadcast* para reduzir o tempo de comunicação.

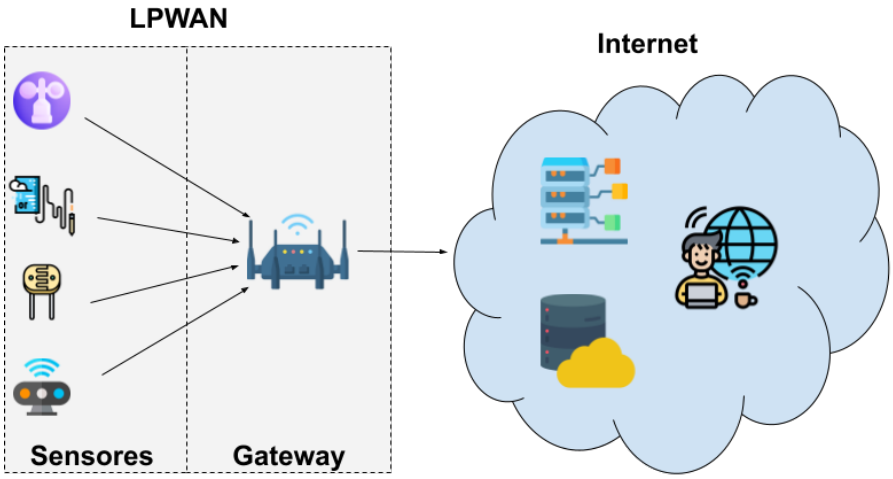
2.2.2 Lora®

A tecnologia LoRa® é uma camada física ou um tipo de modulação utilizada para criar os links de comunicações de longa e média distância, é de propriedade da Semtech e utiliza o conceito de espalhamento espectral e permite comunicação a longas distâncias (em áreas urbanas 3-4 Km de alcance, e em áreas rurais, aproximadamente 12 Km), com consumo mínimo de energia.

Esta tecnologia implementa uma taxa de transmissão utilizando do espalhamento ortogonal, o que permite ao sistema trocar taxa por alcance ou potência de forma dinâmica. (SEMTECH, 2015)

Suas principais aplicações são sistema de IOT, como sensores e monitores remotos, sobretudo aqueles operados a bateria, de mensagens curtas e em alguns casos em locais de difícil acesso.

Figura 2 – Arquitetura de Rede LPWAN



Fonte: Próprio autor utilizando imagens de Flaticon

2.2.3 LoraWAN ™

O protocolo juntamente com a arquitetura são os maiores responsáveis por determinar a vida útil da bateria de um sensor além da capacidade, qualidade e segurança da rede.

2.2.3.1 Modos de Operação

Na tecnologia LoraWAN ™ existem três modos de operação que podem ser utilizados de acordo com a particularidade de cada necessidade, são eles classes A, B e C.

Classe A (Sensores): temos um protocolo muito parecido com o ALOHA, em que sempre que o nodo da rede tem dados para transmitir ele pode acessar a rede mandando o dado. A comunicação entre módulo e *gateway* é bidirecional, porém para que o módulo receba dados deve primeiro fazer uma transmissão.

Classe B (atuadores): temos um modo de operação muito parecido com o protocolo *slotted ALOHA*, em que servirá para obter um momento previsível para fazer a operação de *downlink*. Já no *uplink* temos a comunicação não sincronizada. Os módulos desta classe são geralmente operados a baterias, a comunicação entre módulo e *gateway* é bidirecional, e nesta classe pode-se fazer agendamentos de janelas de recepção para comunicação, assim o *gateway* sempre sabe quando o módulo pode receber dados.

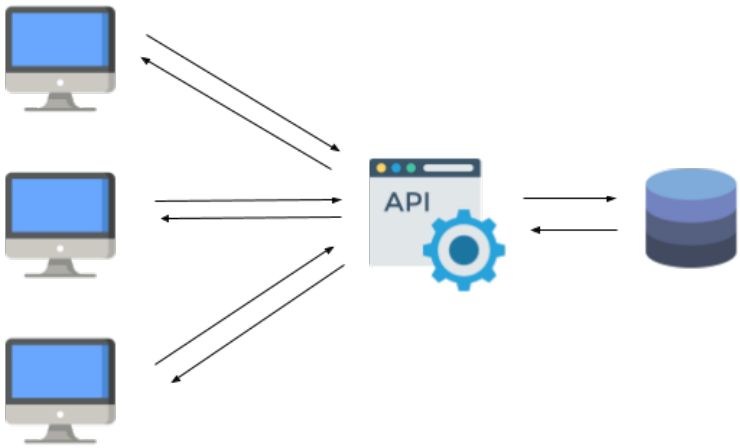
Classe C : temos a total liberdade de recepção de dados nos *endpoints*, ficando com a janela de recepção fechada somente quando ele próprio estiver transmitindo, ou seja, fazendo a operação de *uplink*. Os módulos desta classe não são recomendados a serem operados a bateria, pois como sempre estão atuando o consumo de energia é maior. Então o *gateway* pode enviar mensagens instantaneamente e o módulo sempre receberá.

2.3 Interface Programação de Aplicação (API)

O termo API nada mais é que um conjunto de funcionalidades utilizadas por desenvolvedores, que agem como uma interface entre o sistemas e os desenvolvedores. É importante ressaltar que em todos os casos temos que seguir as suas restrições arquiteturais (COMER, 2006).

Nas aplicações cliente/servidor (Figura 3) utiliza-se o modelo de divisão de cargas de trabalho entre fornecedores e recursos, um servidor é responsável pela execução de um ou mais serviços enquanto o cliente apenas os consome.

Figura 3 – Modelo de arquitetura Cliente Servidor



Fonte: Próprio autor utilizando imagens de Flaticon

Devido à popularidade que os serviços web tem tomado, é comum que as organizações disponibilizem seus serviços através de uma API, sendo necessário a definição de um modelo para integrar o sistema.

2.3.1 Tipos de Protocolo

Ao optar por utilizar o desenvolvimento de um projeto utilizando APIs, deve-se seguir alguns protocolos que servirão para definir as regras. Segundo Castellani (2020) existem três tipos mais utilizados de protocolos para API.

O protocolo Simple Object Access Protocol (SOAP) segue a ideia de padronizar a maneira como os aplicativos devem usar conexões de rede. Possui regras rígidas e consome muitos recursos.

O protocolo Remote Procedural Call (RPC) é o tipo mais simples de API, seu objetivo é que o cliente execute o código em um servidor. Apesar de semelhante ao Representational State Transfer (REST), existem algumas diferenças importantes, são fortemente acopladas, tornando difícil sua manutenção.

Para Fielding (2000) REST é um estilo de arquitetura, ou seja não existe um padrão oficial, para serem consideradas REST devem estar em conformidade com seis restrições de arquitetura, são elas, *Stateless*, interface uniforme, cliente-servidor, cache e em camadas.

> API REST é uma aplicação cliente/servidor que envia e recebe dados através do protocolo *Protocolo de Transferência de Hipertexto (HTTP)* utilizando de padrões de comunicação como *Extensible Markup Language (XML)* e *JavaScript Object Notation (JSON)* e pode ser implementada na linguagem desejada. Uma API é desenvolvida para permitir a interoperabilidade entre aplicações, por exemplo, considerando dois sistemas, um *desktop* e um outro mobile, ambos podem consumir a mesma API para montar suas interfaces, ou seja, a mesma API pode ser utilizada por diferentes sistemas. (BRITO, 2021)

Já para Brito (2021), o modelo de arquitetura REST nada mais é do que um conjunto de padronização com que o aplicativo deverá seguir, ou seja, busca-se uma maior efetividade no código para

se ter produtividade no desenvolvimento das aplicações. Neste caso temos uma separação das atribuições, na parte do cliente são enviadas as requisições, enquanto o servidor faz o processamento e envia a resposta.

2.4 Protocolo de Transferência de Hipertexto (HTTP)

Segundo GRIGORIK (2013), HTTP é um dos protocolos de aplicativos mais utilizados nos dias de hoje. Desde sua publicação, este protocolo tem sido usado como base para o crescimento sem precedentes da Internet. O HTTP está presente todos os dias entregando notícias, vídeos e outros aplicativos da Web dos quais dependemos em bilhões de dispositivos de todas as formas e tamanhos, de computadores a pequenos dispositivos da Web.

2.4.1 Métodos ou Verbos HTTP

O HTTP é o protocolo utilizado na transferência de páginas HyperText Markup Language (HTML). Ao se estabelecer uma conexão, os dados são enviados de um servidor web que armazena os dados para o cliente.

O protocolo HTTP trabalha com requisição e resposta e tem suporte a diversos métodos ou verbos HTTP, estes métodos são responsáveis pela comunicação entre cliente/servidor. Abaixo temos os principais verbos e suas funções:

- ***GET*** : Utilizado para obter informações. Ocorre o envio de determinado dado do servidor para o cliente;
- ***POST*** : Utilizado para envio de dados através do corpo da requisição do cliente para o servidor, possibilitando ao servidor o armazenamento dos mesmos;
- ***PUT*** : Cria ou altera um determinado dado no servidor;
- ***DELETE*** : Requisita a remoção de um dado do servidor;
- ***PATCH*** : Requisita alteração de somente determinados campos em um elemento no servidor, evitando o envio desnecessário de todo o elemento;
- ***HEAD*** : Retorna o cabeçalho de uma resposta, sem o corpo;
- ***OPTIONS*** : Utilizado para que um cliente possa descobrir quais as opções de requisição permitidas para um determinado recurso em um servidor.

2.4.2 Códigos ou Status

Para o projeto será utilizado o conceito de cliente servidor, neste caso, ao solicitar uma requisição sempre teremos uma resposta, esta pode ser em diversos formatos como XML (ver exemplo código 2.1) ou JSON (ver exemplo código 3.7) que é o formato mais adotado por ser leve, legível e de fácil interpretação.

Além disso, o protocolo HTTP dispõe de diversos códigos que devem ser incluídos na resposta, indicando o resultado do processamento.

Segundo o site Devmedia (2020), os códigos iniciados em '2' indicam que a operação foi bem sucedida. Nessa categoria temos, por exemplo, código 200 (OK), indicando que o método foi executado com sucesso; 201 (Created) quando um novo recurso foi criado no servidor; e 204 (No Content) quando a requisição foi bem sucedida, mas o servidor não precisa retornar nenhum conteúdo para o cliente.

Códigos iniciados em '4' indicam algum erro, por exemplo, o código 400 (*Bad Request*) indica que a requisição não pôde ser compreendida pelo servidor, o 404 (*Not Found*) indica que o recurso não foi localizado.(DEVMEDIA, 2020)

Os códigos que indicam erro do lado do servidor. Nesse caso, eles iniciam com '5', como o 500 (*Internal Server Error*), que indica que ocorreu um erro internamente no servidor que o impediu de processar e responder adequadamente a requisição.(DEVMEDIA, 2020)

Os métodos são ações permitidas dentro de uma API, enquanto os códigos são os retornos padrões de uma API, ambos fazem parte das definições do protocolo HTTP.

Código 2.1 – Exemplo de formatação XML

```xml
<CATALOG>
   <CD>
       <TITLE>Empire Burlesque</TITLE>
       <ARTIST>Bob Dylan</ARTIST>
       <COUNTRY>USA</COUNTRY>
       <COMPANY>Columbia</COMPANY>
       <PRICE>10.90</PRICE>
       <YEAR>1985</YEAR> </CD>
   <CD>
       <TITLE>Hide your heart</TITLE>
       <ARTIST>Bonnie Tyler</ARTIST>
       <COUNTRY>UK</COUNTRY>
       <COMPANY>CBS Records</COMPANY>
       <PRICE>9.90</PRICE>
       <YEAR>1988</YEAR> </CD>
   <CD>
       <TITLE>Greatest Hits</TITLE>
       <ARTIST>Dolly Parton</ARTIST>
       <COUNTRY>USA</COUNTRY>
       <COMPANY>RCA</COMPANY>
       <PRICE>9.90</PRICE>
       <YEAR>1982</YEAR> </CD>
</CATALOG>
```

Código 2.2 – Exemplo de formatação JSON

```json
{
   "primeiroNome": "João",
   "ultimoNome": "Silva",
   "genero": "male",
   "idade": 28,
   "endereco": {
       "rua": "Avenida Rio Branco",
       "cidade": "Florianópolis",
       "state": "SC"
   },
   "telefones": [
       { "tipo": "casa", "numero": "7349282382" }
   ]
}
```

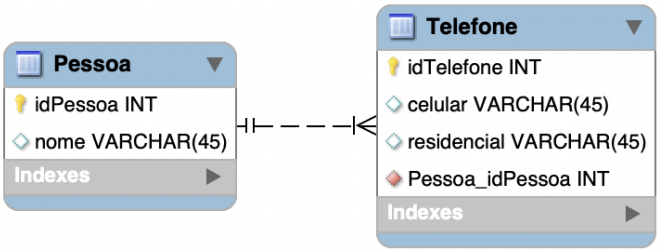
2.5 Banco de Dados

Os bancos de dados são uma estrutura de armazenamento das informações de maneira simples e segura, garantindo a integridade no acesso as informações de todo o sistema. Existem vários modelos de armazenamento de dados, porém os mais comuns são os relacionais e não relacionais.(LIMA; MOURA, 2017)

2.5.1 Relacionais

Os bancos relacionais, são estruturas onde existe um relacionamento entre si, este modelo segue a ideia de armazenamento dos dados em tabelas onde estas possuem colunas com informações, por exemplo na Figura 4, temos um relacionamento entre duas entidades (Pessoa e Telefone).

Figura 4 – Tabelas Banco Relacional



Fonte: Próprio autor utilizando a ferramenta MysqlWorkbench

Ao observar as tabelas é possível notar um relacionamento, cada entidade Pessoa possui um ou mais itens da entidade Telefone, vale ressaltar que este relacionamento pode ser um pra muitos, um pra um ou muitos pra muitos.

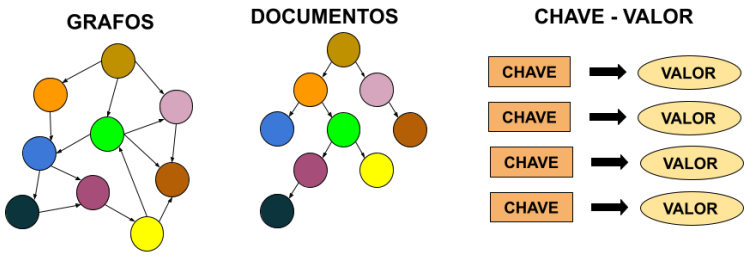
> Um banco de dados é uma coleção de dados estruturados. Ele pode ser qualquer coisa desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede corporativa. Para adicionar, acessar, e processar dados armazenados em um banco de dados de um computador, você necessita de um sistema de gerenciamento de bancos de dados como o Servidor MySQL. Como os computadores são muito bons em lidar com grandes quantidades de dados, o gerenciamento de bancos de dados funciona como a engrenagem central na computação, seja como utilitários independentes ou como partes de outras aplicações.(MYSQLAB, 2006)

2.5.2 Não Relacionais

Os bancos de dados não relacionais surgiram com a ideia de solucionar alguns problemas como escalonamento e armazenamento, diante disto foram desenvolvidas algumas estruturas para suprir esta necessidade, são elas:

- **Chave - valor** : o armazenamento dos dados é realizado através de uma única chave;
- **Orientado a documentos** : o armazenamento dos dados é realizado por uma estrutura de documentos, onde cada documento é uma coleção e todos possuem seu identificador único, cada documento não possui relação com os demais;
- **Banco de dados em grafos** : necessita de uma relação forte, atuam de maneira eficiente ao integrar o banco com aplicações que necessitam de uma relação.

Figura 5 – Modelo Banco Não Relacional

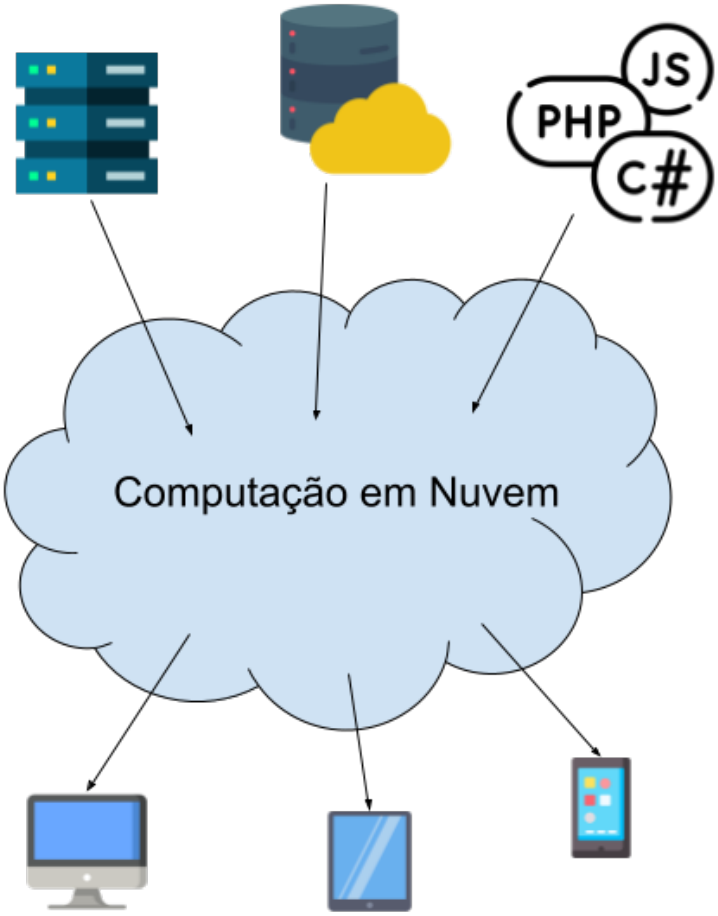


Fonte: Próprio autor utilizando Google docs

2.6 Computação em Nuvem

A computação em nuvem é uma tecnologia que permite acesso remoto a serviços computacionais como programas, arquivos e recursos por meio da internet, é isso que permite a oferta de serviços sob demanda: só se paga por aquilo que, de fato, é consumido, ver Figura 6.

Figura 6 – Computação em Nuvem



Fonte: Próprio autor utilizando imagens de Flaticon

Essas vantagens fazem com que a computação em nuvem seja cada vez mais atraente e disponível para qualquer usuário e empresas de todos os tamanhos, tornando a gestão de tecnologia da informação muito mais eficiente.

Outra característica importante é a possibilidade do profissional atuar remotamente, desde que ele tenha um computador e conexão à internet, ele poderá utilizar as soluções da nuvem.

Com o avanço desta tecnologia podemos acessar arquivos pessoais em qualquer lugar do mundo, sem necessitar de um dispositivo de armazenamento pessoal. Isto acontece devido aos seus dados estarem armazenados em um servidor remoto, um exemplo clássico deste recurso é o Google Docs, onde é possível criar, editar e armazenar arquivos em nuvem.

> A computação em nuvem é o fornecimento de serviços de computação, incluindo servidores, armazenamento, bancos de dados, rede, software, análise e inteligência, pela Internet para oferecer inovações mais rápidas, recursos flexíveis e economias de escala. Você normalmente paga apenas pelos serviços de nuvem que usa, ajudando a reduzir os custos operacionais, a executar sua infraestrutura com mais eficiência e a escalonar conforme as necessidades da sua empresa mudam.(AZURE, 2021)

Tecnologias desta natureza já vêm sendo utilizadas há um bom tempo, um exemplo clássico disso são os serviços de e-mail, podemos acessar os arquivos em qualquer lugar e utilizar um programa remoto para ler os arquivos sem a necessidade de um processamento local da informação.

Para AWS (2021c) as organizações de todos os tipos, portes e setores usam a nuvem para uma grande variedade de casos de uso, como backup de dados, recuperação de desastres, e-mail, desktops virtuais, desenvolvimento e teste de software, análises de big data e aplicativos web voltados ao cliente.

As principais vantagens da computação em nuvem são agilidade, escalabilidade e o acesso compartilhado de arquivos e recursos para um grupo de usuários. De certo modo esta tecnologia permite uma economia considerável em termos de recursos financeiros, pois gasta-se menos com aquisição e manutenção de hardwares.

Da mesma forma que esta tecnologia vem ganhando espaço, há notícias de pessoas se especializando em invadir sistemas vulneráveis, fato que trás preocupação no quesito segurança e faz com que as empresas de computação em nuvem investem muito em segurança.

2.7 Microclimas

O clima influencia diretamente no crescimento das culturas, no rendimento das plantações, na ocorrência de pragas, na necessidade de água e fertilizantes, além de todas as demais atividades agrícolas, desde o plantio até a colheita, o agricultor precisa estar atento às estações do ano e possíveis imprevistos climáticos como geadas, granizos, etc.

De acordo com Ciência e Clima (2018) o microclima é o clima de um lugar específico, produzido sob a influência da topografia local e da vegetação, pode apresentar temperaturas mais frias ou mais quentes do que a atmosfera livre.

O microclima se caracteriza pelas variações climáticas que ocorrem numa região em específico, refletindo as alterações no ambiente e possui uma característica peculiar de vulnerabilidade, ou seja, podem ocorrer algumas mudanças que o tornará diferente das características do clima de uma determinada região que o cerca.

Segundo Broto (2021) o estudo de Microclimas no agro procura entender e avaliar como os diversos componentes do clima interagem e podem afetar a agricultura. Esses estudos visam principalmente aos diferentes microclimas no ambiente onde são desenvolvidas as culturas no campo. Assim, focam com mais rigor o conjunto de fenômenos climáticos e suas influências na camada de ar mais próxima da lavoura e da superfície do solo.

Portanto os microclimas são diretamente impactados por características peculiares, localizados onde afetam diretamente uma determinada região, como por exemplo uma topologia do terreno, o tipo de solo ou fatores externos.

Adquirir um entendimento dos microclimas locais é importante para o desenvolvimento e manejo mais adequado das culturas, saber quais as características e como podem ser trabalhados ajudam a favorecer e facilitar na condução da lavoura.

2.8 Análise de Mercado

A análise de mercado é um dos pontos principais no desenvolvimento de um bom plano de negócios, a partir dela é possível entender as necessidades do público alvo, os melhores fornecedores e o perfil da concorrência.

Para Pereira (2021) a análise de mercado é uma das atividades mais determinantes para o sucesso de um empreendimento, e caso ela seja negligenciada, as consequências podem ser drásticas.

Em outras palavras pode-se afirmar que esta análise é o processo de coleta de informações relacionadas ao mercado de atuação de uma empresa, procedimento que oferece uma avaliação para estabelecer quão atrativo está um determinado mercado. Nela são analisados dados sobre o segmento e o contexto em que a organização irá atuar, seu potencial público-alvo, a relação do seu produto com seus fornecedores, e ainda o posicionamento da concorrência, sendo possível ter uma ideia de quais os riscos atuais e futuros que a empresa enfrentará.

2.8.1 Modelo de Negócio Canvas

O Canvas se refere a uma ferramenta visual que tem como objetivo organizar as ideias sobre um determinado negócio, permitindo que um modelo de negócios organizado, prático e fácil seja criado, aumentando a produtividade do time e possibilitando que a empresa se posicione à frente dos concorrentes em diversos cenários.

Segundo Camargo (2019), o Canvas é uma solução muito simples e eficiente que ajuda gestores de negócios a visualizarem de maneira estratégica todas as questões mais importantes da sua empresa.

Desenvolvido por Alexander Osterwalder o diagrama permite uma visualização estratégica na criação de seu negócio, o objetivo é que você monte o canvas em conjunto, assim tem-se uma observação pontual de cada pessoa dentro da equipe.

Para Osterwalder e Pigneur (2010) um diagrama de canvas padrão segue a premissa de reunir todos os interessados em uma mesma estrutura visual. Na Figura 7 temos uma ilustração que define muito bem o que o autor cita.

Analisando a Figura 7 percebe-se que o *canvas* é um mapa visual que contém todos os pilares de um negócio. São nove os quadrantes a serem preenchidos (ABSTARTUPS, 2019).

- **Parcerias**: são *stakeholders* [1], partes interessadas, externas que de alguma forma são essenciais para seu negócio e viabilizam que você foque no seu negócio, como por exemplo um investidor;
- **Atividades Chave**: é o que você precisa fazer para que seu negócio consiga operar;

[1] O termo é muito utilizado nas áreas de comunicação, administração e tecnologia da informação, cujo objetivo é designar as partes interessadas de um planejamento estratégico ou plano de negócios.São os *stakeholders* que legitimam as ações de uma organização e tem um papel de influência para a gestão e os resultados dessa mesma organização. (SIGNIFICADOS.COM, 2019)

Figura 7 – Conceitos Principais do Canvas

Fonte: Próprio autor utilizando imagens de Flaticon

- **Recursos Chave**: são os bens essenciais que são fundamentais para o negócio;
- **Proposta de Valor**: é o que você faz para o cliente escolher o seu produto, um conjunto de benefícios que entregam valor para o cliente;
- **Relacionamento**: a forma como você se relaciona com cada cliente;
- **Segmento de Clientes**: saiba quem são seus clientes, quais as suas necessidades e objetivos;
- **Canais**: meios com que a proposta de valor é comunicada;
- **Custo**: toda estrutura de recursos, atividades ou parcerias lhe gera custos;
- **Fontes e Receita**: define como cada setor gera valor para o negócio. Cabe a ressalva de que alguns setores podem não gerar dinheiro, diretamente, mas são o coração do negócio.

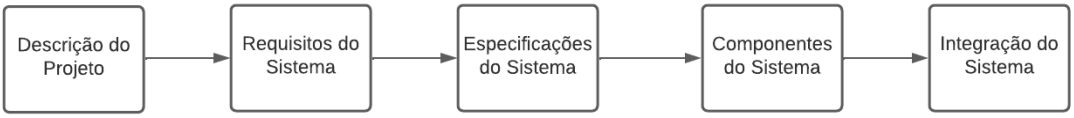
Em tese estes nove elementos representam as quatro principais áreas do negócio: cliente, oferta, infraestrutura e viabilidade financeira. O Canvas propõe criar um modelo de negócios a partir da proposta de valor e organizar sua estrutura baseado em elementos chave.

3 DESENVOLVIMENTO

Este projeto consiste em desenvolver uma plataforma de monitoramento das variáveis climáticas em tempo real, nesta seção será detalhado o desenvolvimento do projeto.

Para realizar o desenvolvimento, utilizamos a metodologia de Mínimo Produto Viável (MVP)[1], pois temos a intenção de validar esta solução o mais rápido possível em clientes reais, para depois, com o retorno deles, trabalharmos as melhorias relatadas na utilização da solução. A Figura 8 mostra o fluxo das etapas para o desenvolvimento, as quais serão descritas a seguir.

Figura 8 – Etapas de Desenvolvimento



Fonte: Próprio Autor - Desenvolvido em Ludichart

3.1 Descrição do Projeto

A proposta do projeto consiste em oferecer uma ferramenta onde a parte interessada poderá monitorar em tempo real as variáveis climáticas de uma determinada região.

Através de uma rede de sensores será feita a leitura das variáveis para apresenta-los em uma interface amigável e de fácil entendimento, para concretizar esta ideia foi necessário um bom embasamento teórico e técnico, pois o projeto envolve diversas tecnologias e seu desenvolvimento engloba uma série de componentes que serão descritos nos próximos capítulos.

Devido a agricultura ter uma dependência direta com o clima, este projeto vem com o intuito de servir como uma ferramenta adicional na gestão do agronegócio, ajudando os envolvidos a melhorar seus indicadores através de uma gestão estratégica.

A produção agro vem passando por mudanças drásticas nos últimos anos, acreditava-se que a produtividade estava diretamente relacionada ao tamanho das propriedades, já que para aumentar a produção era necessário um aumento na área, porém a realidade do produtor rural acabou modificando este cenário, hoje, com a tecnologia, é possível ter um aumento considerável na produtividade sem necessidade de ampliar a extensão de territorial de cultivo.

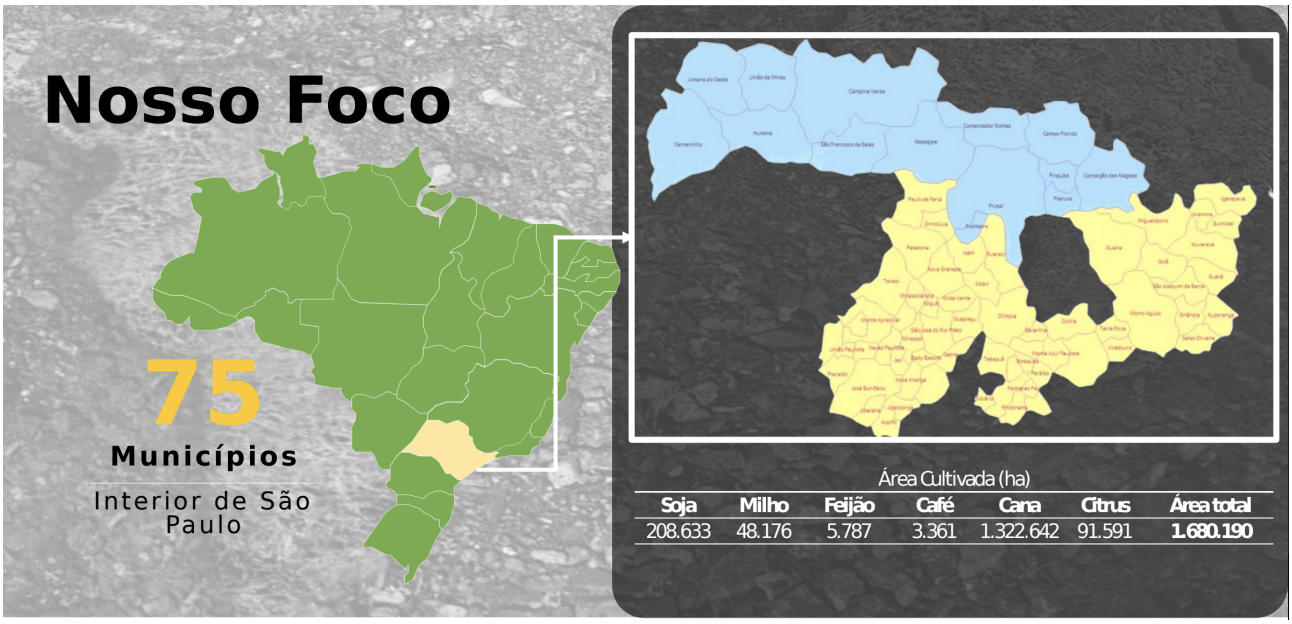
Tendo em vista que o sistema será focado para o cultivo e plantação, a ferramenta desenvolvida irá oferecer uma solução para captação e apresentação dos dados que são relevantes para a agricultura. O projeto tem como objetivo tornar-se mais uma das soluções tecnológicas para o agro.

Para delimitar o escopo, a proposta visa solucionar problemas enfrentados por agricultores de pequeno e médio porte, localizados no interior de São Paulo, onde segundo (AVINCO, 2022) temos mais de 1,6 milhões de hectares de cultivos e mais de 5 mil produtores deste porte. A seguir, para entender

[1] O MVP serve para medir a viabilidade de um negócio. São feitos testes com o público e ao receber o feedback são feitas alterações no desenvolvimento da empresa. O produto mínimo viável nada mais é uma versão simplificada do produto ou serviço fornecido, de uma maneira que possam ser feitos testes de mercado. (SEBRAE, 2019)

melhor qual será o papel do projeto no agronegócio, foi desenvolvido um modelo de negócio canvas, o qual auxilia estrategicamente o desenvolvimento do projeto e nos dá uma ideia geral dos agentes e ferramentas envolvidas. Com a construção desse modelo, conseguimos pensar, discutir e compreender melhor o potencial do negócio, bem como buscar trazer valor para o cliente com a solução proposta.

Figura 9 – Nosso Foco



Fonte: Hamilton Marques Avinco

3.1.1 Desenvolvimento Modelo Canvas

O projeto visa auxiliar o setor a implementar novas soluções para alguns problemas enfrentados no campo. Para entender melhor qual será o papel do projeto no agronegócio foi desenvolvido um modelo canvas que auxilia estrategicamente o desenvolvimento do projeto e nos da uma ideia geral dos agentes e ferramentas envolvidas.

Para um bom entendimento do canvas é sugerido uma leitura de dentro para fora, sua principal função é simplificar o desenvolvimento do negócio, logo abaixo será detalhado os tópicos mencionados pelo canvas da Figura 10.

- **Oferta de Valor**

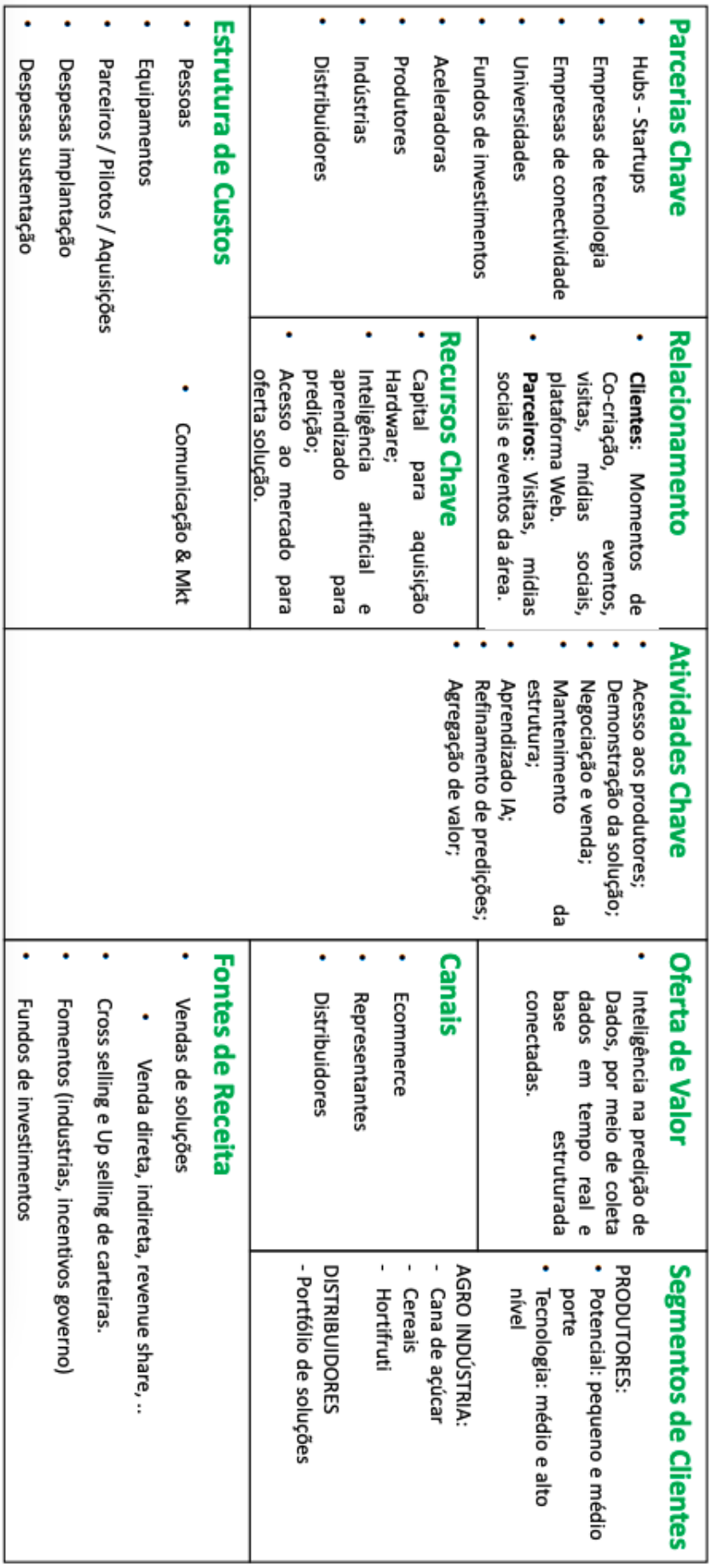
  A ideia do projeto é oferecer uma plataforma onde o usuário poderá monitorar em tempo real toda a situação climática de uma região previamente determinada, como consequência, a ferramenta poderá auxiliar o usuário na tomada de decisões.

  Através de uma gestão estratégica o projeto oferecerá melhora nos índices de produtividade relacionados a agricultura. A partir deste momento, o contratante terá um controle mais efetivo sobre as principais variáveis que influenciam no plantio e cultivo da sua cultura.

  Através da predição e um controle mais efetivo sobre a situação climática de uma determinada região o cliente poderá tomar decisões estratégicas e fundamentais no sucesso da safra.

- **Atividades Chaves**

Figura 10 – Busnies Model Canvas



Fonte: Hamilton Marques Avinco

Oferecer um sistema de monitoramento das principais variáveis climáticas através de uma rede de sensores instalados em campo e implementar uma rede de comunicação LPWAN para comunicação destes dispositivos com o mundo digital.

Através da captação dos dados oferecer um sistema de aprendizado de máquina para oferecer um sistema de predição climática agregando valores ao produto.

- **Recursos Chave**

  Recursos financeiros estão entre os principais neste momento, aquisição de hardwares serão fundamentais para o desenvolvimento no negócio.

  Outros recursos para o projeto serão de infraestrutura (hardwares e softwares) e também humanos. Os recursos computacionais para desenvolvimento de uma inteligência artificial serão terceirizados com empresa especializada na área, parte dos recursos de hardwares deverão ser adquiridos com empresas homologadas.

- **Canais**

  Os principais canais de comunicação serão por meio de telefonemas, e-mails e mensagens. Através deles serão feitos e registrados todos os contatos com clientes e possíveis interessados do projeto.

  Ecomerce, representantes e distribuidores estão entre os principais meios para mostrar nosso produto aos possíveis clientes.

- **Relacionamento**

  Uma boa relação com clientes e parceiros sem dúvida é uma das chaves para o sucesso, para estabelecer este canal será feito um link de comunicação direto entre a equipe e demais envolvidos, através de ligações telefônicas, mensagens, e-mails e visitas em loco.

- **Parcerias Chave**

  Fundos de investimentos poderão se tornar a principal fonte de fomento neste momento, busca por recursos estão entre os tópicos fundamentais para o desenvolvimento do projeto, além disso o produto necessita de uma melhora em diversas partes, portanto pesquisa de desenvolvimento no negócio serão bem aceitas.

  Por termos a conectividade como um dos principais gargalos do projeto empresas de conectividade serão bons parceiros.

- **Segmento de Mercado**

  A ferramenta irá proporcionar ao usuário um sistema de apoio na tomada de decisões, através do monitoramento das variáveis climáticas em tempo real, portanto toda e qualquer pessoa com a intenção de se informar do clima de uma determinada região poderá contratar o serviço.

  Como produtores da área agro estão entre os mais interessados em acompanhar o clima vemos clientes em potencial nesta área, sendo eles pequenos, médio ou grande porte.

- **Estrutura dos Custos**

  Como principais custos no projeto tem-se o armazenamento dos dados e softwares em nuvem, a contratação de uma empresa especializada em comunicação Lora, recursos humanos e uma possível estrutura física para o pessoal trabalhar e desenvolver novos recursos.

- **Fontes de Receita**

  Existem algumas abordagens para este tema, a locação de todos os equipamentos ou a venda deles, isto ainda não está definido, o cliente pagará uma mensalidade conforme o número de dispositivos

instalados, assim como o tráfego de dados que este cliente irá demandar. Outros parceiros interessados podem entrar no negócio como investidores.

Fazendo uma análise do canvas, pode-se observar que a proposta visa oferecer uma solução de auxilio na tomada de decisões aos interessados em monitorar as variáveis climáticas (principalmente da área agro), mesmo sendo uma solução interessante e fundamental na administração do agronegócio existem diversos fatores que podem ser decisivos no sucesso do projeto, que podem ser visualizados a seguir.

A solução exige um esforço considerável para se consolidar no mercado, um dos maiores problemas observados será a captação de cliente, por este motivo será necessário recorrer a profissionais da área como possíveis parceiros. A cobertura de sinal na área rural também impacta diretamente na qualidade do serviço, levando em conta a necessidade da estação estar localizada em locais remotos e a necessidade de troca de mensagens serem fatores importantes no projeto, foi levantado como solução a implementação de uma rede LPWAN privada, mais especificamente Lora, os motivos da escolha foram a capacidade de alcance e o fluxo de mensagens entre a estação e a camada de rede serem baixos.

Outro fator preocupante na análise são os recursos humanos, sendo um projeto que envolve diversas tecnologias, é necessário a contratação de diversos profissionais, o que encarece o projeto.

A resistência que alguns empreendedores da área possuem quando se fala em aplicar tecnologia no agro precisa ser observada, isso foi analisado através da leitura de alguns blogs e notícias, além de conversas com diversos empreendedores. Este cenário vem se modificando, alguns veem a necessidade de investimento em tecnologia e pesquisa o que acaba tornando a ideia ainda mais atrativa para o agro.

Existem outros fatores que auxiliam na concretização do projeto, como a baixa competitividade na área, em uma pesquisa a nível nacional existem poucas empresas que atuam nesta área e nenhuma delas utiliza as tecnologias de comunicação Lora, em sua grande parte utilizam tecnologias de comunicação que demandam um custo alto de implementação. Uma das vantagens de utilizar a tecnologia Lora será a cobertura que um *gateway* [2] pode oferecer e o baixo custo de sua implementação.

Viabilizar o projeto exigirá um esforço enorme de toda e equipe, entrar neste mercado de trabalho não é algo simples, um projeto deste nível envolve diversas tecnologias por estes motivos deve-se estar preparado e um bom embasamento técnico e teórico das ferramentas necessárias para concluir o projeto podem definir o seu futuro.

O setor de tecnologia está aquecido e desenvolver tecnologia para o agro será fundamental para a humanidade, melhorar métodos de cultivos, aperfeiçoar técnicas, compartilhar o conhecimento, estão entre os fatores fundamentais no melhoramento da maneira com que se conhece a agricultura.

3.2 Requisitos do Sistema

O processo de desenvolvimento compreende um conjunto de atividades que tem como objetivo atender aos requisitos especificados pelo projeto (cliente, usuário), ao atende-los temos uma precondição básica para o sucesso, por isto é necessário um bom levantamento dos requisitos. Como parte fundamental no processo de desenvolvimento, o levantamento dos requisitos deve ser executado pensando nas necessidades do cliente.

De acordo com IEEE os requisitos são definidos como:

[2] gateway é um hardware ou software de rede usado em redes de telecomunicações que permite que os dados fluam de uma rede discreta para outra, os gateways são distintos dos roteadores ou switches, pois se comunicam usando mais de um protocolo para conectar várias redes e podem operar em qualquer uma das sete camadas do modelo de interconexão de sistemas abertos.(EN, 2022)

- Uma condição ou capacidade necessitada por um usuário para resolver um problema ou alcançar um objetivo.
- Uma condição ou capacidade que deve ser satisfeita ou possuída por um sistema ou componente do sistema para satisfazer um contrato, um padrão ou uma especificação.
- Uma representação documentada de uma condição ou capacidade.

Um software mal especificado, sem dúvidas irá causar problemas para toda a equipe, desenvolver, estudar e atender os requisitos é tarefa fundamental para o sucesso, caso contrário o projeto tem grande chance de sofrer modificações, ocasionando assim custos desnecessários.

3.2.1 Levantamento dos Requisitos

Nesta seção será descrito o levantamento dos requisitos necessários para desenvolver um produto minimamente viável, como parte fundamental no projeto deve-se atentar aos mínimos detalhes. Abaixo segue o requisito do sistema, nele consta um levantamento de uma série de fatores relevantes para a confecção do projeto.

Requisitos Funcionais

Os requisitos funcionais são importantes no desenvolvimento de softwares, é importante construir um modelo que seja claro e objetivo. Para se obter requisitos funcionais de qualidade o responsável deve estar atento às necessidades levantadas pelo cliente. Nesta seção será definido os requisitos funcionais do sistema, onde o projeto começará a se materializar e tomar forma.

O sistema deve possuir um controle de acesso para usuários e cada usuário irá possuir no mínimo um tipo de permissão. O sistema irá contar com 3 tipos de permissões 'admin', 'user' e 'super-admin'. A permissão de 'super-admin' terá acesso a todo o sistema, podendo editar, excluir e criar dados, a permissão de 'admin' será destinada aos usuários que terão acesso livre aos seus dados podendo editar, deletar e criar qualquer dado que seja pertinente a sua conta, já a permissão de 'user' será delegada aos usuários que possuem acesso apenas para leitura dos dados pertinente a sua conta.

O usuário terá acesso a uma interface onde existirá um painel demonstrativo das variáveis capturadas pelo sistema, vale ressaltar que não teremos um período definido para esta amostragem, ou seja, o usuário irá ter acesso a qualquer dado que remeta ao passado em qualquer momento, desde que este pertença a sua conta.

Mensagens de erros devem ser apresentadas em caso de problemas com os cadastros ou atualizações, assim como mensagens de sucesso quando cadastrados ou atualizados com sucesso.

O sistema deverá possuir uma integração com a estação meteorológica, sendo esta responsável pela captura dos dados a serem apresentados, vale ressaltar que uma estação pode pertencer a vários usuários. O sistema deve ser capaz de armazenar uma quantidade massiva de dados originados da estação meteorológica.

Para os usuários terem acesso ao painel demonstrativo os mesmos devem estar vinculado a uma empresa e cada empresa vinculada a uma ou mais estação. Somente será liberado acesso para os usuários pertinentes aquela empresa, o mesmo vale para a empresa, somente será liberado acesso aos dados das estações pertinentes aquela empresa.

A comunicação entre a estação meteorológica e o sistema se dará por meio da tecnologia Lora, toda esta comunicação será terceirizada ou implementada pela equipe dependendo do caso.

Com algumas premissas já definidas podemos relatar alguns casos de usos para deixar ainda mais claro o objetivo do sistema.

- **Usuário com permissão 'super-user' cadastra empresa**

  Apenas usuários com a permissão de 'super-user' podem cadastrar uma empresa, cada empresa deve conter um endereço completo, nome, cnpj, telefone, razão social e web site. Usuários com esta permissão também poderão vincular outros usuários às empresas a eles pertencentes.

- **Usuário com permissão 'admin' e 'user' cadastra empresa**

  Como um dos requisitos é de que apenas usuários com permissão de 'super-admin' efetuem o cadastro o sistema deverá bloquear todo o conteúdo para cadastro de nova empresa.

- **Usuário comum se cadastra**

  Tem-se a intensão de oferecer uma amostra do que pode ser visto no sistema à qualquer usuário, portanto ele poderá cadastrar-se no sistema sem nenhuma restrição, para cadastro dos usuários será necessário possuir os dados de nome e sobrenome, senha, email, telefone, endereço completo e cpf, ao se cadastrar todo usuário já possuirá a permissão de 'user' por padrão.

- **Usuário acessa painel de amostras**

  Com o usuário logado o sistema deverá fornecer uma aba onde deve ser possível entrar com um intervalo de datas para se obter um gráfico com as amostragens capturadas pela estação meteorológica.

- **Tabela das variáveis climáticas**

  A estação meteorológica deve ser confiável e possuir a capacidade de captura dos seguintes dados:

Tabela 1 – Dados capturados pela estação meteorológica.

Descrição	**Unidade**
Temperatura	°c
Umidade	%
Velocidade Vento	m/s
Sentido Vento	°
Radiação	W/m^2
Pluviômetro	mm
Luminosidade	lx
Pressão Barométrica	hPa

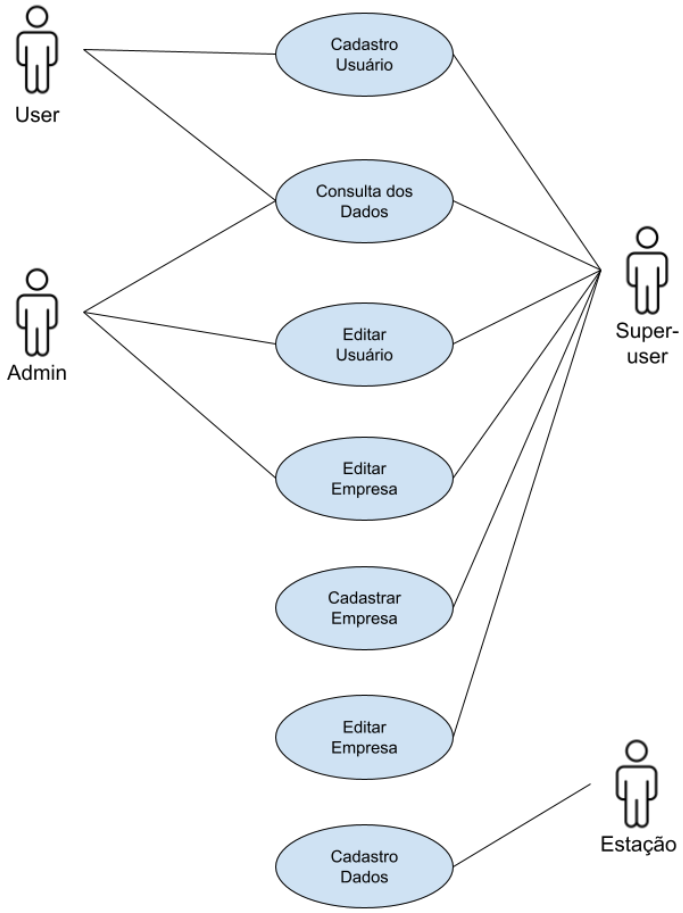
Agora que temos uma ideia clara do projeto podemos definir alguns requisitos funcionais do sistema (ver Tabela 3).

Tabela 2 – Requisitos Funcionais

Identificação	**Autor**	**Permissão**	**Objetivo**
Efetua cadastro	Usuário	-	Este caso o usuário deve ser capaz de cadastrar seus dados
Efetua cadastro	Usuário	user	Este caso o usuário **não** deve ser capaz de cadastrar nenhum dados
Efetua cadastro	Usuário	admin	Este caso o usuário deve ser capaz de cadastrar dados pertinentes ao seu domínio(empresa)
Efetua cadastro	Usuário	super-user	Este caso o usuário deve ser capaz de cadastrar dados em qualquer parte do sistema
Edita cadastro	Usuário	user	Este caso o usuário deve ser capaz de editar seus dados
Edita cadastro	Usuário	super-user	Este caso o usuário deve ser capaz de editar qualquer dado do sistema
Edita cadastro	Usuário	admin	Este caso o usuário deve ser capaz de editar os dados pertinentes ao seu domínio(empresa)
Cadastra informação de variável climática	Sistema	-	Este caso o sistema deve cadastrar as variáveis climáticas
Cadastra tipo de informação de variável climática	Sistema	-	Este caso o sistema deve capturar e cadastar as variáveis de temperatura, umidade, velocidade e sentido do vento, rediação, pluviômetro, luminosidade e pressão barométrica

Na Figura 11 temos um diagrama com as funções que cada autor irá desempenhar no sistema, podemos identificar que além de existir uma distinção de usuários, existe outro autor externo que será o responsável por alimentar o sistema com os dados capturados.

Figura 11 – Diagrama UML



Fonte: Próprio autor

Requisitos Não Funcionais

Assim como os requisitos funcionais os não funcionais possuem um papel fundamental no desenvolvimento do projeto, apesar de não estarem ligados diretamente ao funcionamento do produto, eles ajudam a definir como o sistema irá executar suas tarefas. Os requisitos não funcionais são praticamente todas as necessidades que devem ser atendidas para que os funcionais possam operar de maneira eficiente. Para uma primeira versão vamos definir algumas premissas que servirão de base para definir alguns requisitos básicos(BARRETT, 2002).

- **Geral**

  Todas as versões de softwares básicos, frameworks, servidores e quaisquer outros recursos utilizados pela solução deverão ser totalmente compatíveis entre si.

  O sistema será dividido em três partes, IOT, administração do sistema e interface gráfica.

  Na parte de IOT vamos ter a comunicação entre a estação meteorológica e o sistema, já na parte administrativa do sistema será destinada ao armazenamento e tratamento dos dados coletados e por fim a interface gráfica irá apresentar os dados.

- **Produto**

O sistema deverá apresentar uma interface simples e de fácil entendimento para a iteração com o usuário.

A montagem dos equipamentos a serem instalados em campo serão feitas pelo próprio cliente com o auxílio de um manual a ser desenvolvido pela equipe.

- **Servidores**

  Toda a infra de servidores será feita em serviços terceirizados como AWS, Azure ou Google.

- **Banco de dados**

  A solução deverá armazenar os dados pertinentes aos usuários em um banco relacional de preferência MYSQL, já os dados referentes as variáveis que o sistema deve capturar deverão ser armazenados em um banco de dados não relacional como MongoDb.

- **Segurança**

  A solução deverá possuir um sistema de autenticação em todas as etapas do processo, assim como garantir que alguns usuários tenham acesso a determinadas áreas autorizadas pelo sistema.

  Toda a autenticação deverá ser feita via *token.* [3]

- **Energia**

  Um dos pontos críticos do sistema é a alimentação da estação meteorológica, na grande parte dos casos não existirá fonte de energia próxima. Para solucionar este problema o sistema deverá contar com um sistema de energia responsável por alimentar a estação de forma continua e eficiente.

- **Desempenho**

  O sistema deve ser capaz de suportar um número mínimo de 10 (dez) usuários simultâneos, além de ser capaz de receber dados de no mínimo 20 (vinte) estações meteorológicas.

  As estações deverão enviar dados atualizados a cada 10 (dez) minutos no máximo.

- **Comunicação**

  A comunicação entre a estação meteorológica e o gateway será feita via Lora, todas as demais via protocolo IP.

  Toda a comunicação Lora será implementada pela equipe ou terceirizada por uma empresa especializada, dependendo do caso.

- **Confiabilidade**

  O sistema não poderá ficar fora por mais de 24 horas seguidas.

  A estação meteorológica não deverá ficar sem transmitir dados para o sistema por mais de 48 horas seguidas.

  O sistema deve garantir o acesso as informações pertinentes a cada autorização definida nos requisitos funcionais.

[3] tokens são dispositivos físicos que auxiliam o usuário quanto à segurança pessoal ao gerar uma senha temporária de proteção para as contas que ele utiliza. Normalmente o processo é feito através de um aparelho semelhante a um chaveiro, que cria senhas especiais com um único clique, ideais para transações bancárias pela internet. (FONSECA, 2009).

Tabela 3 – Requisitos não Funcionais

Classificação	Tipo	Descrição
Essencial	Integrado	O sistema deverá ser totalmente integrado, frameworks e dispositivos deverão se comunicar perfeitamente.
Essencial	Integridade	O sistema deverá ser dividido em componentes (IOT e core).
Desejável	Usabilidade	O usuário deverá clicar no máximo 5 vezes para realizar operação de consulta aos dados.
Essencial	Infraestrutura	Todos os recursos computacionais de produção deverão ser alocados em servidores terceirizados.
Essencial	Segurança	Os dados dos usuários devem ser protegidos, e só devem ser acessados com autorização.
Essencial	Desempenho	O sistema deverá garantir um número mínimo de 10 usuários simultâneos e 20 estações meteorológicas no mínimo.
Essencial	Energia	A estação deverá operar com bateria capaz de garantir no mínimo 48 horas sem o sistema de alimentação de recarga ativo.
Essencial	Confiabilidade	O sistema não deverá ficar fora de operabilidade por mais de 24 horas.
Essencial	Confiabilidade	A estação não deverá permanecer por mais de 48 horas sem comunicação com o sistema.
Essencial	Comunicação	A comunicação entre a estação e o sistema se dará por meio da tecnologia Lora.
Essencial	Produto	Todas as ferramentas utilizadas no desenvolvimento deverão ser gratuitas, exceto em modo de produção.

3.3 Especificações do Sistema

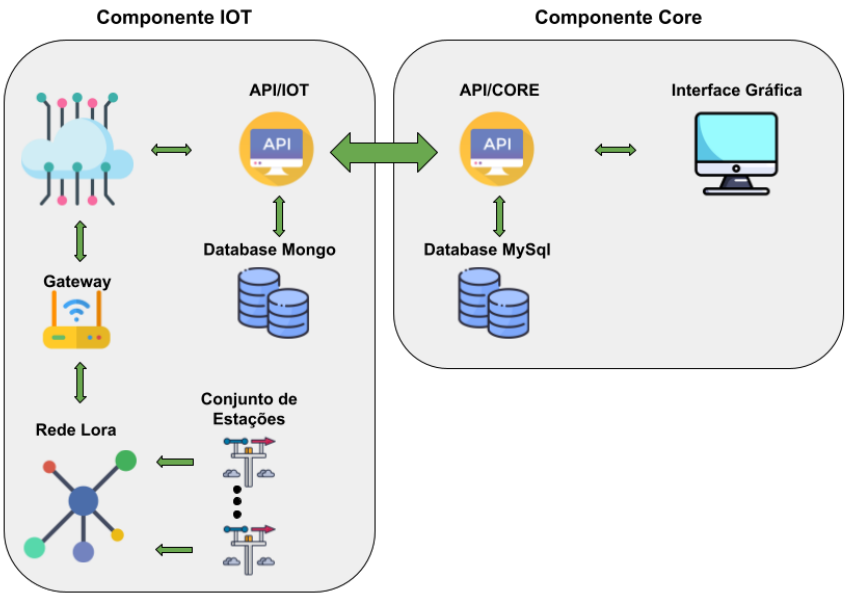
Com os requisitos bem definidos e alguns cenários descritos na sessão 3.2, tem-se todos os dados necessários para elaborar as especificações do projeto. Visando cumprir todos os requisitos para o desenvolvimento do projeto optou-se em desenvolver o sistema na forma de componentes.

O sistema será composto por dois componentes, como ilustrado na Figura 12, o componente de Iot será responsável pela comunicação e armazenamento dos dados originados pelas estações já o componente core será responsável pela administração do controle e acesso ao sistema principal.

A escolha por dividir o sistema se deu pela necessidade de simplificar o desenvolvimento, além de dividir a carga de trabalho e garantir uma escalabilidade do sistema. O componente IOT possui uma complexidade maior, sua arquitetura conta com diversas tecnologias o que acaba tornando-o sua implementação mais complexa, já o componente core conta com uma arquitetura mais simplificada, porém isto não significa que seu desenvolvimento seja simples, nele teremos toda a administração do sistema, como controle de acesso e armazenamento dos dados pertinentes a administração do sistema.

Como pode ser observado na Figura 12 os componentes se comunicarão entre si, a troca de informação é constante e a entrega de mensagens deve ser garantida, havendo algum problema em um dos componentes, a aplicação irá se corromper comprometendo todo o sistema.

Figura 12 – Diagrama do Sistema

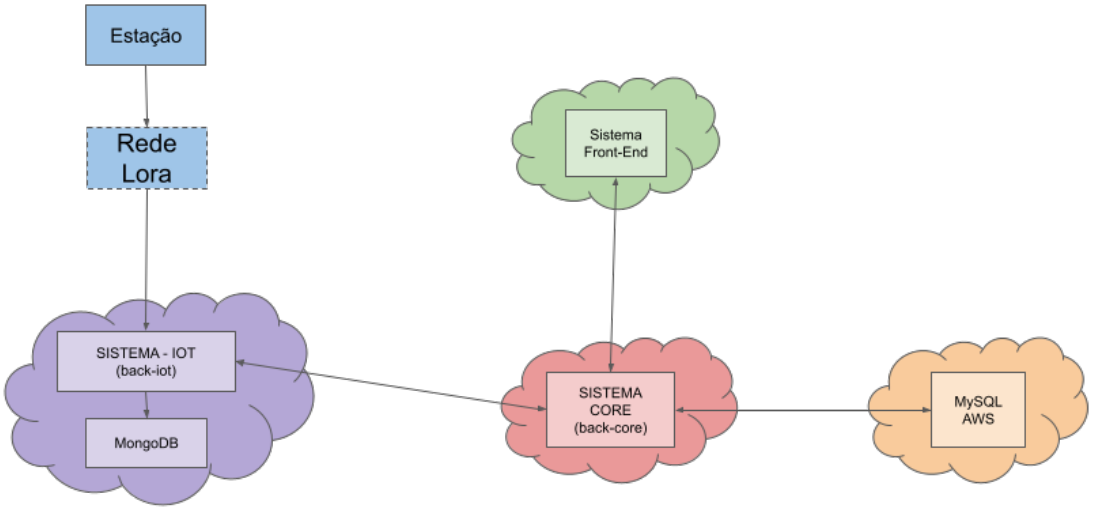


Fonte: Próprio autor

3.3.1 Arquitetura do Sistema

O projeto engloba diversos conceitos como desenvolvimento em softwares, montagem de hardwares e comunicação entre componentes. Um estudo minucioso e detalhado dos requisitos foi desenvolvido para se chegar a uma arquitetura que atendesse aos requisitos do sistema. A Figura 13 demonstra como será a arquitetura do sistema.

Figura 13 – Arquitetura do Sistema



Fonte: Próprio autor

Esta arquitetura foi desenvolvida para atender os requisitos do sistema, a escolha por dividir o projeto em diversas partes ocorreu para simplificar o desenvolvimento e deixar mais claro o papel de cada

sistema no projeto.

A estação será a responsável por capturar os dados e enviá-los por meio da tecnologia lora para o sistema de IOT, o sistema de IOT ficará responsável por armazenar e disponibilizar os dados capturados pela estação, estes dois componentes juntos formam o que apelidou-se de "Componente IOT", já na outra parte, temos o sistema core que ficará responsável por armazenar e administrar os dados pertinentes a usuários além de ter um acesso direto ao sistema IOT, o sistema Front-End será responsável por oferecer uma interface gráfica para o usuário, ele terá acesso direto ao sistema core, estes últimos sistemas formarão o que chamamos de "Componente Core".

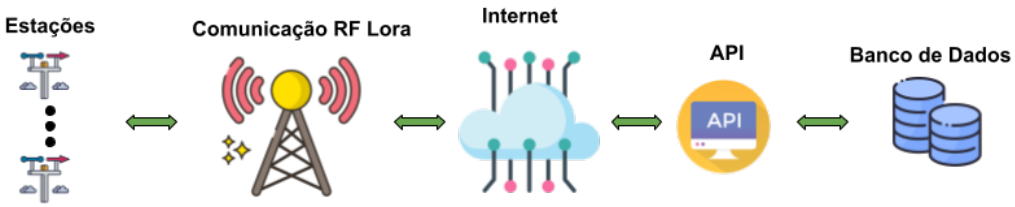
3.4 Componentes do Sistema

A modelagem do sistema foi desenvolvida para atender as necessidades dos requisitos levantados no projeto, a escolha por dividir o projeto em componentes se deu pela necessidade de oferecer uma divisão de carga e deixar claro qual será o papel de cada componente no projeto. Nas próximas sessões será detalhado a tarefa de cada componente e por fim a integração entre eles.

3.4.1 Componente IOT

O componente IOT tem como principal objetivo fazer a integração com a rede de sensores, nele teremos API, banco de dados, rede Lora e hardwares, o funcionamento e conjunto destes subcomponentes garantirão a integridade do componente. Por possuir diversas tecnologias embarcadas, nota-se que o desenvolvimento deste componente não será algo simples, é necessário um estudo detalhado e um embasamento técnico que garantirá a implementação correta dos dispositivos.

Figura 14 – Diagrama Componente IOT



Fonte: próprio autor

Na Figura 14 temos um diagrama detalhado dos subcomponentes envolvidos, em alguns casos em particular é possível simplificar ainda mais o componente, nos deparar com esta situação será praticamente nula nos dias atuais pois a cobertura de sinal lora em áreas rurais ainda é pouca e caso isto aconteça a única modificação será a exclusão do Gateway, ou seja, a mensagem será encaminhada diretamente da estação para a rede Lora.

A arquitetura do componente iot foi desenvolvida pensando nas necessidades que o sistema exigirá. Neste componente temos a comunicação entre a estação e o gateway implementada em uma rede privada ou pública (dependendo do caso) utilizando a tecnologia de transmissão Lora, a mensagem é entregue a uma API(sistema IOT) utilizando o protocolo *TCP/IP* [4] como protocolo de transporte, por fim após o recebimento da mensagem na API é feito um tratamento e armazenamento da mensagem.

[4] TCP (Transmission Control Protocol) e IP (Internetworking Protocol). O IP trataria do roteamento de datagramas, enquanto o TCP seria responsável pelas funções de nível mais alto, como segmentação, remontagem e detecção de erros. O protocolo de interligação em rede tornou-se conhecido como TCP/IP. (FOROUZAN, 2010)

Existem duas possibilidades de implementar este componente, em locais onde já existe uma cobertura de rede ofertada por uma empresa especializada como *Everynet* [5] pode-se desenvolver uma parceria eliminando assim alguns componentes de hardware, já por outro lado caso não existe nenhuma rede pública acessível a solução exige a utilização de uma rede privada.

Estação

Tendo a Khomp como parceira do IFSC-SJ e com seus produtos em grante parte já homologados na rede Lora Everynet optamos por desenvolver o projeto utilizando a estação meteorológica Khomp.

A Estação Meteorológica constata parâmetros meteorológicos e climáticos através dos 7 sensores (temperatura, umidade, nível pluviométrico, luminosidade, índice UV, velocidade e direção do vento), fornecendo dados meteorológicos abrangentes para monitoramento do ambiente para aplicações do agronegócio, funcionalidades industriais e comerciais. O sistema é resultado da identificação das necessidades de clientes e parceiros que precisam monitorar os tipos de grandezas que são coletadas pelos sensores(KHOMP, 2019a).

Figura 15 – Estação Meteorológica Khomp

Fonte: Khomp (KHOMP, 2019a)

Rede Lora

A rede Lora será desenvolvida de forma privada ou por uma empresa especializada. Para escolher a opção será feito um levantamento *in loco*, onde não haver uma cobertura de sinal lora oferecida por alguma empresa será desenvolvido um projeto de implantação de rede privada.

[5] A infraestrutura da Everynet é feita para permitir serviços IoT robustos de baixa potência de longa distância (LPWA), com os menores custos compartilhados e o menor tempo para gerar receita.(EVERYNET, 2020)

Tabela 4 – Especificações da Estação

Parâmetro	Intervalo	Precisão	Unidade
Volume de chuva	0 a 15	1	mm
	15 a 6553,5	7%	mm
Iluminação	1 a 128.000	15%	lux
Direção Vento	0 a 359	12,5%	°
Velocidade Vento	0 a 50	10%	m/s
Temperatura	-40 a 60	1%	°C
Umidade	10 a 99	5%	%

Tabela 5 – Outras Especificações da Estaçao

Parâmetro	Descrição
Alimentação	3,3 VDC à 5 VDC, 100 mA
Aquisição de dados	a cada 16 s
Dimensões	135x97x26mm
Peso	805g
Temperatura de Operação	-40 a 60 °C
Umidade de Operação	10–90% (não condensado)

Gateway

Para cenários onde a rede Lora não cobre a região será necessário o desenvolvimento de uma rede privada, com um gateway de comunicação adquirido por fornecedores parceiros. Neste caso vamos utilizar um gateway da Khomp ITG 200. O ITG 200 é um gateway de telemetria desenvolvido para integração de soluções de monitoramento IoT – Internet of Things. Ele recebe e transmite dados coletados através de Endpoints sem fio, provendo escalabilidade à solução IoT(KHOMP, 2019b).

Figura 16 – Gateway Itg200 Khomp

Fonte: Khomp (KHOMP, 2019a)

Banco de Dados

Como o sistema Iot conta com uma base de dados massiva onde as informações são dinâmicas e a probabilidade de mudança na estrutura dos dados é grande, optou-se em utilizar um banco de dados não relacional orientado a documento, que permite a modificação e manipulação dos dados de forma simples além de possuir uma ótima performance na pesquisa por dados o que é fundamental para o sucesso do projeto.

Para atender aos requisitos vamos desenvolver uma estrutura que armazena os dados pertinentes as variáveis climáticas e alguns outros que podem ser relevantes para o projeto, como a relação sinal ruído, status da conexão e outros.

Como estamos tratando de uma base de dados massiva, concentrar estes dados em uma única base não seria indicado, teríamos uma perda de performance nas consultas, pensando nisso decidiu-se dividir o banco em diversas coleções.

O MongoDB é um banco de dados orientado a documentos, ou seja, os dados são armazenados como documentos, podendo ser descritos como dados no formato de chave-valor, por aceitar uma dinâmica maior entre os valores armazenados na base, este modelo atende perfeitamente as necessidades do sistema iot.

Como as informações recebida pelo sistema possuem um certo padrão, decidiu-se dividir os dados em grupos. No modelo do código a seguir temos um payload enviado para a API do sistema.

Código 3.1 – Exemplo Payload

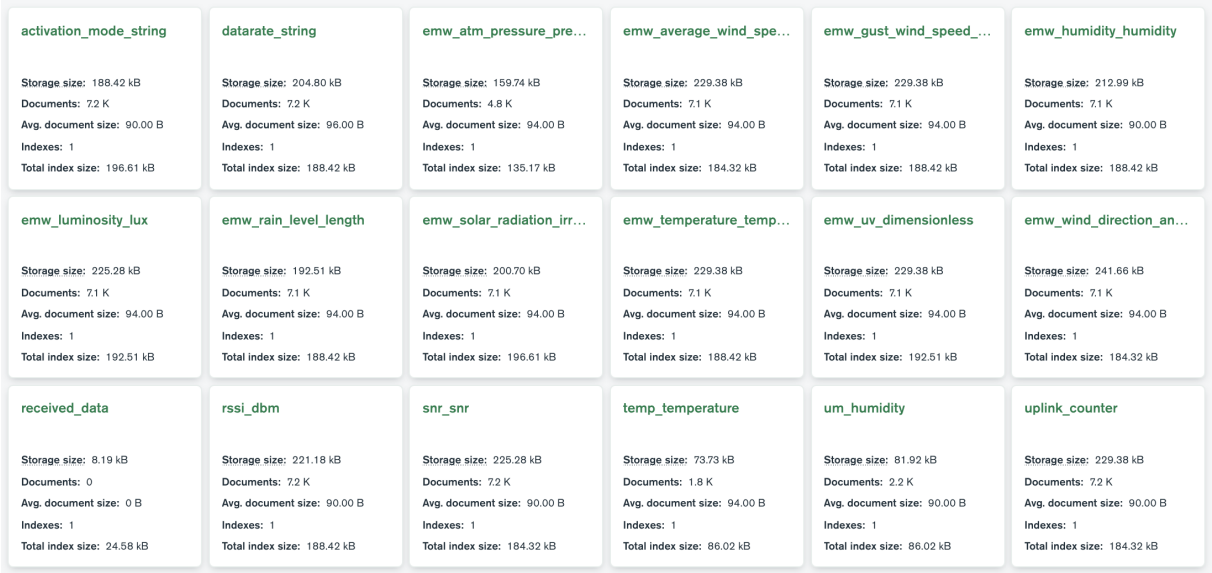
```
38 {
39     "seq": 7,
40     "data": [
41         {
42             "time": 1643723563514032,
43             "unit": 655361,
44             "value": 71.7,
45             "dev_id": "f803320100027b40",
46             "ref": "um_humidity"
47         }, {
48             "time": 1643723563434775,
49             "unit": 2224179556,
50             "value": 301.39,
51             "dev_id": "f803320100027b40",
52             "ref": "temp_temperature"
53         }, {
54             "time": 1643723563352591,
55             "unit": 1310720,
56             "value": 10.0,
57             "dev_id": "f803320100027b40",
58             "ref": "snr_snr"
59         }
60     ]
61 }
```

A estratégia para definir as coleções no sistema partiu de um estudo detalhado dos valores recebidos no *payload*. Os dados que interessam estão dentro do vetor chamado "data"e a sequência de dados obedece um padrão de informação com chave e valor. A sequência de dados será utilizada para gerar e adicionar os dados nas tabelas, como por exemplo, na linha 44 temos um campo "ref", este será o dado responsável por gerar ou adicionar dados nas tabelas, neste caso em específico, teremos uma tabela com o nome "um_humidity"responsável por armazenar as informações pertinentes a umidade de todas as estações do sistema, analisando a segunda posição do vetor "data"temos no campo "ref"com a informação "emp_temperature", neste caso será criado uma coleção com este nome e armazenado um documento com as informações.

A Figura 17 mostra as coleções utilizadas no sistema.

Seguindo esse princípio, todas as informações originadas da estação serão armazenadas em alguma

Figura 17 – Database Sistema IOT



Fonte: MongoDb Compass

coleção, seja ela variáveis climáticas ou do próprio sistema.

API

Também conhecida como Sistema Iot, será destinado para a manipulação e recebimento dos dados originados da estação. Do ponto de vista da API pouco importa o que aconteceu antes da entrega da mensagem, o que interessa neste ponto é o formato da mensagem que será recebida assim como o tratamento dado a esta mensagem.

Desenvolvimento da API

Para facilitar o trabalho vamos utilizar um framework chamado django-restframework, por possuir diversas ferramentas integradas como serialização, serviço de autenticação e outros ele ajuda muito o desenvolvimento de API‘s.

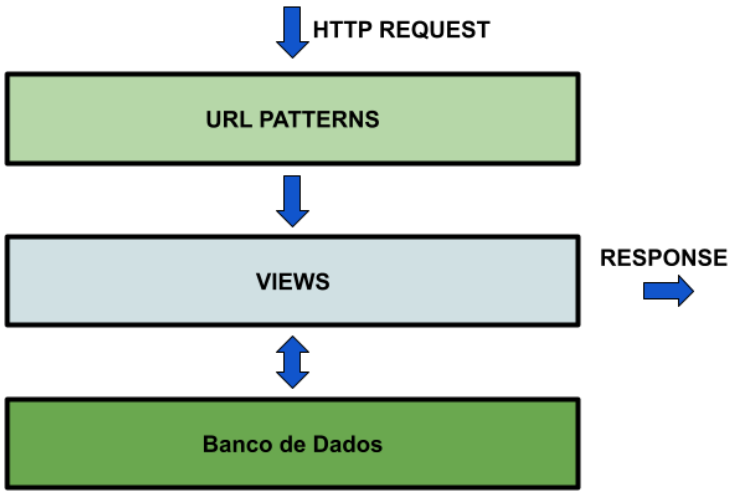
Esta API é responsável por receber, armazenar e controlar os dados originados das estações meteorológicas, além de oferecer uma comunicação direta entre componentes através de *endpoints* definidos.

A API foi desenvolvida utilizando o conceito de camadas. A primeira camada tem a responsabilidade de processar as requisições vindas dos outros componentes, como o django rest-framework trabalha com disparo de eventos, esta primeira camada dispara um evento que aciona a camada inferior, o viewset, aqui se inicia o processamento dos dados e também é onde se envia a resposta da requisição. Descendo Uma camada temos o banco de dados, para este caso em específico a camada Views terá acesso direto a camada de Banco de Dados, fugindo um pouco da metodologia utilizada neste *framework*.

3.4.2 Componente Core

O Componente Core é responsável por administrar, controlar e gerenciar todo o acesso ao sistema. Este componente tem a característica de atuar próximo ao usuário controlando o acesso, fornecendo informações e efetuando cadastros no sistema. A arquitetura do componente core segue o modelo de *web*

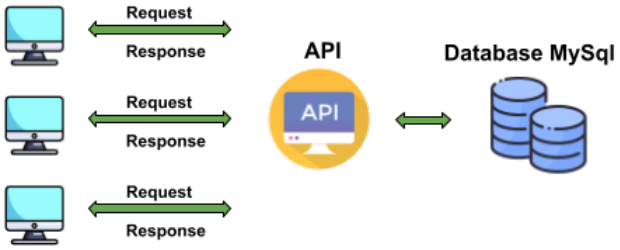
Figura 18 – Arquitetura API IOT



Fonte: próprio autor

service, dividida em duas partes, a primeira será chamada de *backend* [6], sua principal tarefa será fazer a ponte de comunicação entre a interface de usuário e o banco de dados, além de fazer validações e a manipulação dos dados, a outra parte temos o *frontend* [7], nele será desenvolvida a interface gráfica para iteração do usuário com o sistema.

Figura 19 – Arquitetura Componente Core



Fonte: próprio autor

O componente core seguirá o modelo de uma arquitetura cliente-servidor, onde o processamento da informação é dividido em módulos ou processos distintos, neste caso o servidor mantém as informações e o controle sobre ela enquanto o cliente as consome. Devido a necessidade de dividir o componente em alguns subcomponentes, nesta situação em particular vamos ter dois outros sistemas, chamados de Sistema Core e Sistema Front-End.

[6] O Back End trabalha em boa partes dos casos fazendo a ponte entre os dados que vem do navegador rumo ao banco de dados e vice-versa, sempre aplicando as devidas regras de negócio, validações e garantias em um ambiente onde o usuário final não tenha acesso e possa manipular algo. (SOUTO, 2019)

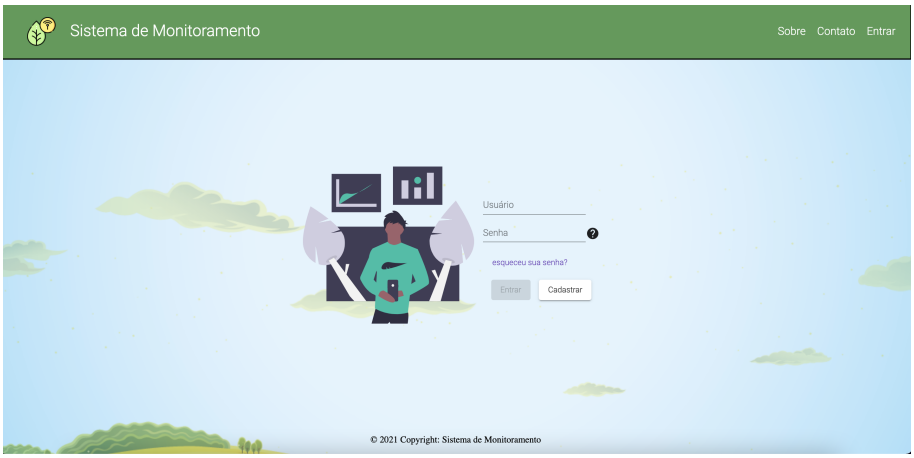
[7] Podemos classificar como a parte visual de um site, aquilo que conseguimos interagir. Quem trabalha com Front End é responsável por desenvolver por meio de código uma interface gráfica, normalmente com as tecnologias base da Web (HTML, CSS e JavaScript). (SOUTO, 2019)

Sistema Front-End

No sistema front-end acontece toda a iteração com o usuário, ele é o responsável por controlar a interface gráfica e oferecer uma experiência agradável ao usuário. Para o desenvolvimento do front-end foi utilizado um framework conhecido como *angular* [8] por facilitar e agilizar o processo.

Desenvolver uma interface amigável não é simples, é necessário um estudo detalhado mas, como se trata de um produto minimamente viável, uma interface simples e de fácil manuseio já atende o projeto, as Figura 20, Figura 21, Figura 22, Figura 23 e Figura 24, ilustram como ficou a interface gráfica do sistema, vale ressaltar que o acesso às informações serão diferentes dependendo das permissões, por exemplo, um usuário com a permissão de administrados poderá visualizar todos os dados pertinentes a sua empresa, já um usuário com a permissão de user terá acesso apenas aos seus dados pessoais e por fim um usuário super-admin terá acesso aos dados de todas as empresas e usuários.

Figura 20 – Interface Gráfica - Tela de Login



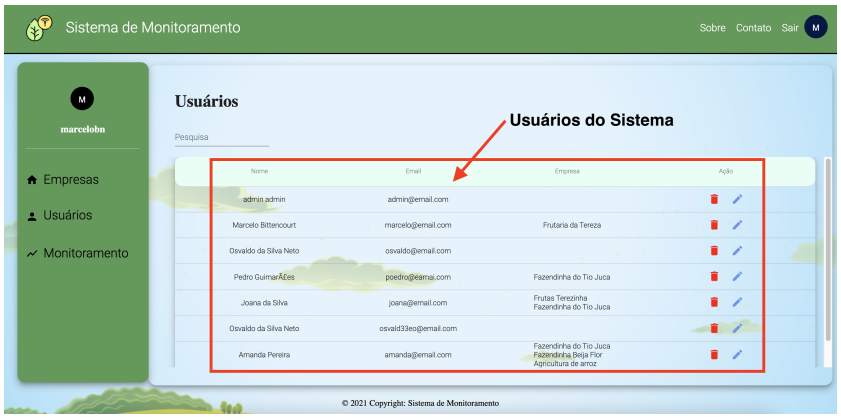
Fonte: próprio autor

Figura 21 – Interface Gráfica - Tela de Empresas

Fonte: próprio autor

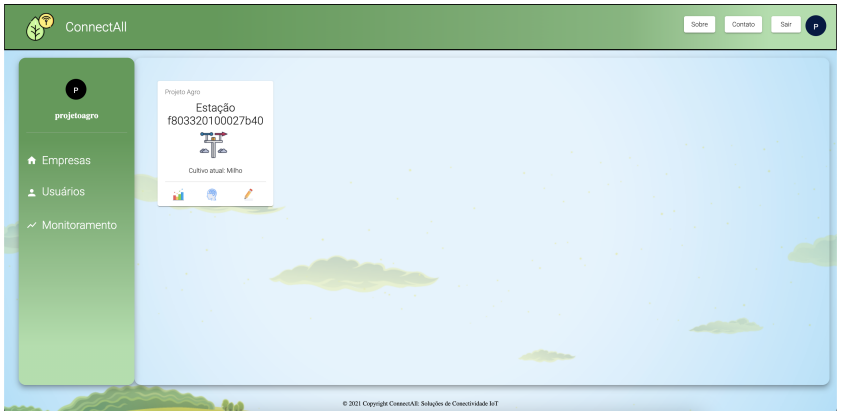
[8] Angular é uma estrutura de design de aplicativos e plataforma de desenvolvimento para criar aplicativos de página única eficientes e sofisticados.(ANGULAR, 2020)

Figura 22 – Interface Gráfica - Tela de Empresas



Fonte: próprio autor

Figura 23 – Interface Gráfica - Tela das Estações



Fonte: próprio autor

Figura 24 – Interface Gráfica - Tela dos Gráficos

Fonte: próprio autor

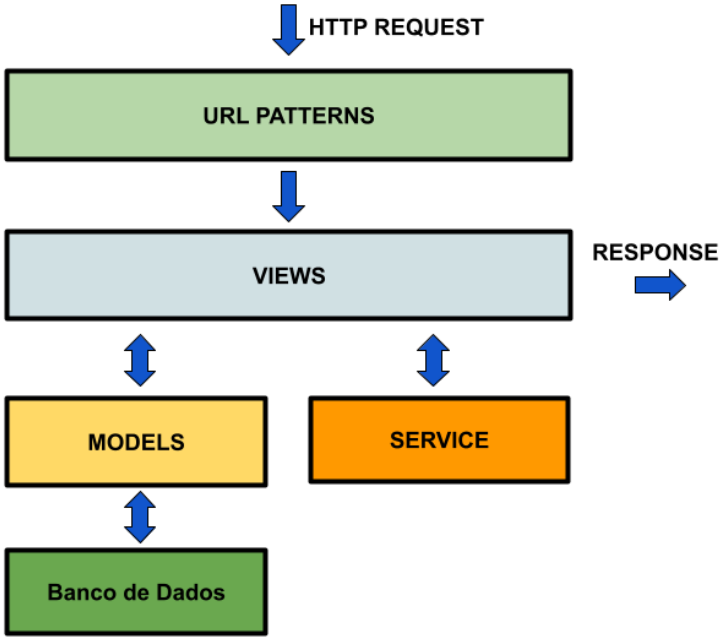
A interface gráfica necessita de alguns ajustes, mas para fins de desenvolvimento de um produto minimamente viável a solução está dentro do esperado, todo o código do desenvolvimento esta armazenado no Github e pode ser acessado através do link https://github.com/projeto-agro-tcc/osvaldo-frontend.git.

Sistema Back-End

No desenvolvimento do sistema backend será utilizado o framework *"django rest framework"*, que além de utilizar uma linguagem popular também é possível fazer todo o *CRUD* [9], outra ferramenta que auxilia bastante é a interface de administração sendo possível fazer cadastro, gerar tokens de autenticação e outras.

Este sistema é responsável por armazenar e controlar o acesso a informação, ele será o coração de todo o sistema, implementando as lógicas para interagir com o sistema front-end e o componente de IOT.

Figura 25 – Arquitetura Sistema Back-end - CORE



Fonte: próprio autor

O projeto foi implementado utilizando o conceito de camadas do *django restframework*. A primeira camada tem a responsabilidade de processar as requisições vindas dos usuários, supõe-se que o usuário deseja acessar a rota "/empresas", o framework irá processar a informação e encaminha-la para a camada inferior apropriada.

Código 3.2 – Arquivo URL

```
62 router = routers.DefaultRouter()
63 router.register('empresas', EmpresasViewSet, basename='Empresas')
64 router.register('usuarios', UsuariosViewSet, basename='Usuarios')
65 router.register('estacoes', EstacoesViewSet, basename='Estacoes')
66
67 urlpatterns = [
```

[9] CRUD (acrónimo do inglês Create, Read, Update and Delete) são as quatro operações básicas (criação, consulta, atualização e destruição de dados) utilizadas em bases de dados relacionais (RDBMS) fornecidas aos utilizadores do sistema. (WIKIPEDIA, 2011)

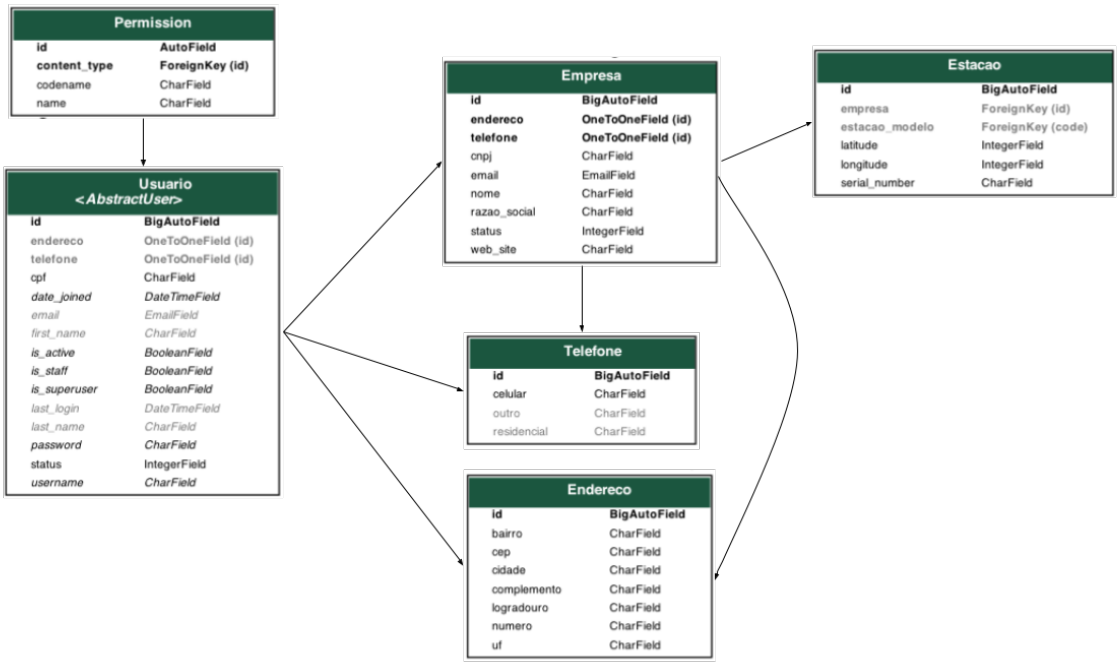
```
68     path('', include(router.urls)),
69     path('admin/', admin.site.urls),
70     path('auth/', CustomAuthToken.as_view()),
71 ]
```

Note que no código existem três rotas registradas no projeto (empresas, usuraios, estacoes), cada uma é responsável por formar uma parte de recursos do sistema.

Após a camada de rota processar o request, será disparado um evento que acionará o viewset apropriado da camada View, aqui se inicia o processamento dos dados e também é onde se envia a resposta da requisição.

Agora, vamos descer um pouco mais e explorar a camada Model da arquitetura, esta camada é a fonte de armazenamento vamos descrever, em forma de classes (ver Figura 26), as entidade as do nosso sistema. Um modelo é a descrição do dado que será gerenciado pela sua aplicação, nele temos os campos e comportamentos. No fim, cada modelo vai equivaler à uma tabela no banco de dados.

Figura 26 – Diagrama de Clases



Fonte: próprio autor - utilizando ferramenta Pygraphviz

Em paralelo a camada de Model temos a camada de service, que será responsável por auxiliar o processamento de atividades extras do sistema, como controle de acesso, tratamento de exceções e outros.

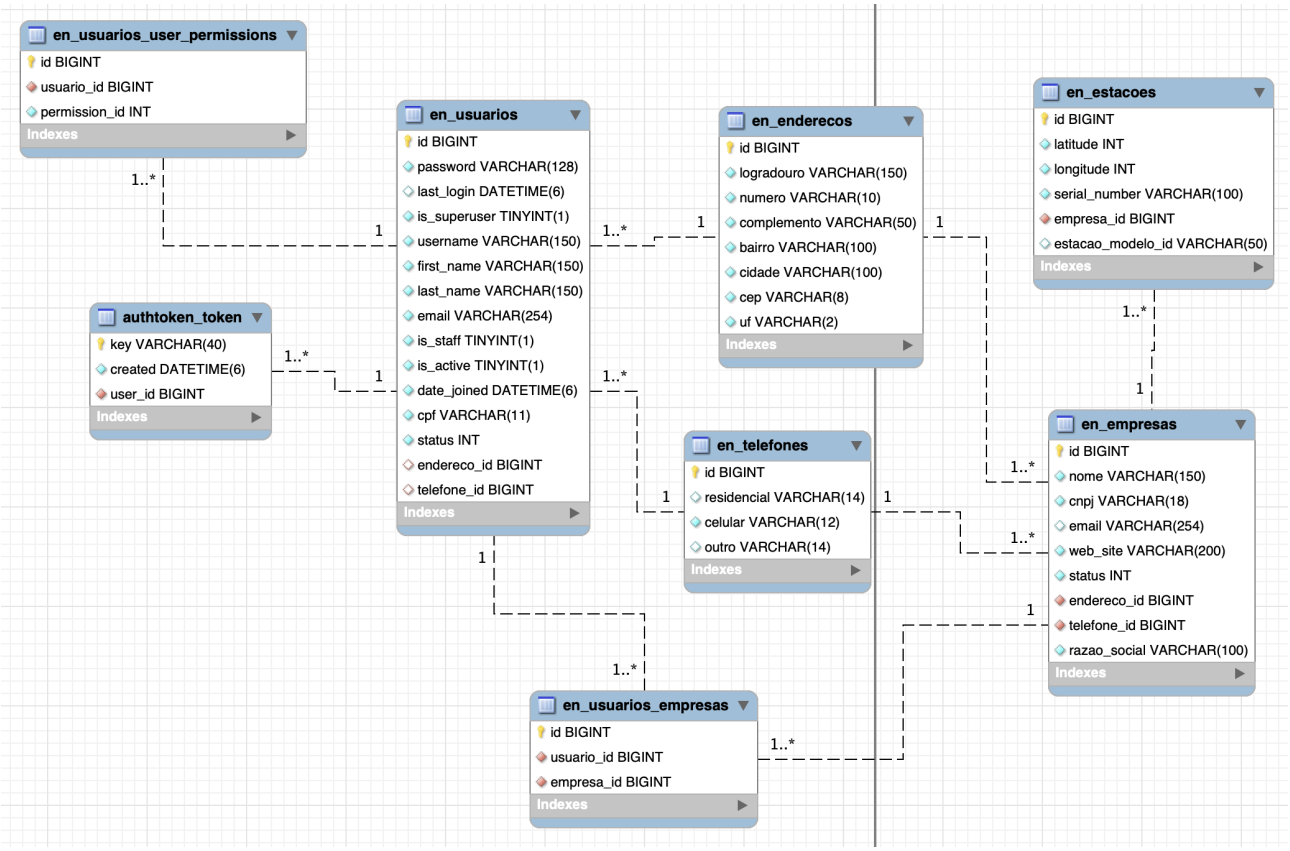
Modelagem do Banco de Dados

Modelagem de dados é o ato de explorar estruturas orientadas a dados. Do ponto de vista de um desenvolvedor, modelagem de dados é conceitualmente similar à modelagem de classes. Com a modelagem de dados identificamos tipos de entidades da mesma forma que na modelagem de classes identificamos classes.(NETO, 2011)

Para desenvolvimento do sistema core constatou-se a necessidade de trabalhar com o *mapeamento*

objeto-relacional [10], por este motivo optou-se por desenvolver a modelagem utilizando o banco de dados relacional. Podemos observar na Figura 28 a estrutura e os relacionamentos do banco.

Figura 27 – Estrutura Banco de Dados



Fonte: Workbench - desenvolvido por próprio autor

Para chegar a esta estrutura foi necessário uma análise minuciosa dos requisitos, só então a partir dela o banco foi modelado, o trabalho de modelagem iniciou-se pensando nos parâmetros e comportamentos que cada objeto terá no decorrer do projeto.

Dividiu-se o projeto em três principais objetos: usuários, empresas e estações. Para confecção da tabela de usuários (en_usuarios) optou-se por armazenar os principais dados como nome, sobrenome, senha, email, cpf, endereço, telefone, além de outros que o framework necessita. Na tabela de empresa (en_empresas) entramos com os dados de nome fantasia (nome), cnpj, email, web site, status (ativo ou inativo), razão social, endereço e telefone. Na tabela de estações (en_estacoes) armazenamos os dados pertinentes a cada estação como coordenadas (latitude e longitude), número serial (serial_number) e empresa.

Além destas tabelas temos outras que servirão de complemento para dar suporte no desenvolvimento e escalabilidade do projeto, como por exemplo as tabelas de endereço (en_endereco) e telefone (en_telefone) que possuem relacionamento com empresas e usuários.

[10] O mapeamento objeto-relacional (ORM ou MOR) é uma técnica de muito utilizada para converter dados entre bancos relacionais e linguagens orientadas a objeto, como é o caso do PHP, C++, Java, Python, Ruby, entre outras. (ORM, 2019)

3.5 Integração do Sistema

Para garantir um bom funcionamento do hardware é também necessário o software, onde ambos devem operar em plena sintonia. O software tem a função de interpretar e fornecer as instruções ao hardware, capacitando na realização das operações.

Como o projeto conta com uma série de hardwares e sistemas de softwares, foi necessário desenvolver uma integração que seja cautelosa e ao mesmo tempo eficiente. No total tem-se duas API's (IOT e Core) e um conjunto de hardwares (Gateway e Estação).

Figura 28 – Integrações do Sistema

Fonte: próprio autor

O uso de API's é capaz de viabilizar uma série de vantagens ao integrar sistemas, desta maneira é possível oferecer novos serviços e melhorar os já existentes, essas ferramentas garantem um grau maior de eficiência, uma vez que possibilitam automatizar atividades que levariam muito tempo para serem executadas manualmente.

3.5.1 Integração A

Esta integração ocorre por meio de uma comunicação direta entre as estações e o gateway é implementada utilizando a tecnologia lora como meio de comunicação. O protocolo Lora utiliza uma arquitetura da rede em formato de estrela, no geral esta topologia é composta por diversos dispositivos finais (*endpoints*), gateways e servidores de redes e aplicação.

No projeto a comunicação acontecerá utilizando a faixa de frequência utilizada na Austrália e adotada também no Brasil de 915 MHz, operando com 8 canais conforme manual do dispositivo.

3.5.2 Integração B

A integração entre o gateway e a API/IOT é consolidada utilizando como camada de transporte o protocolo *TCP/IP* na camada de aplicação temos o protocolo *http* enviando a mensagem a uma API REST.

A mensagem partirá do gateway em direção a API/IOT seguindo um padrão de troca de mensagens especificado pelo fabricante. Na API/IOT existe dois endereços responsáveis pela comunicação com o gateway.

Para o endereço <http://.../api/v1_2/json/itg/connection_status>, é enviado uma mensagem periódica informando o estado da conexão e a interface de rede está sendo usada para envio das mensagens, no Código 3.3 pode-se ver o padrão da mensagem enviada.

Código 3.3 – Exemplo Payload - Conection Status

```
72 {
73     "seq": "int",
74     "signal": "int"
75 }
```

Para o endereço <http://.../api/v1_2/json/itg/data>, é enviado as mensagens com os dados coletados pelos sensores, no Código 3.4 pode-se ver o formato das mensagens.

Código 3.4 – Exemplo Payload - Data

```
{
    "seq": "int",
    "data":[
        {
            "time" : "unsigned int",
            "unit" : "unsigned int",
            "value" : "double",
            "dev_id" : "string",
            "ref": "string",
            "error" : "int"
        },{
            "time" : "unsigned int",
            "unit" : "unsigned int",
            "value" : "double",
            "dev_id" : "string",
            "ref": "string",
            "error" : "int"
        }
    ]
}
```

As mensagens enviadas para a API-IOT seguem um padrão estipulado pelo fabricante do equipamento, ao receber esta mensagem a API-IOT da inicio ao processo de armazenamento das mensagens em um banco de dados.

Por padrão todas as mensagens devem ser respondidas ao *gateway* contendo o seguinte corpo de mensagem (Ver Código 3.5), onde "seq"representa a sequência recebida, "status"o código Http da requisição e "message"contendo informações adicionais.

Código 3.5 – Exemplo Payload - Data

```
{
    "seq": "int",
    "status": "int",
    "message": "string"
}
```

3.5.3 Integração C e D

Nas integrações C e D foi consolidada utilizando o protocolo TCP/IP na camada de transporte e o protocolo Http na camada de aplicação. As comunicações entre as Api‘s se deu utilizando a arquitetura REST permitindo que o sistema acessem e manipulem os dados entre si através de alguns endpoints disponíveis.

Código 3.6 – Arquivo Rotas - API-IOT

```
#Rotas disponíveis para a comunicação entre gateway e Api-iot
router.register('api/v1_2/json/itg', DataViewSet, basename='data')

#Rotas disponíveis para a comunicação com a API-Core
router.register('iot', IotViewSet, basename='iot')
```

Código 3.7 – Arquivo Rotas - API-Core

```
106 #Rotas disponíveis para comunicação com o Front-end
107 router.register('empresas', EmpresasViewSet, basename='Empresas')
108 router.register('usuarios', UsuariosViewSet, basename='Usuarios')
109 router.register('estacoes', EstacoesViewSet, basename='Estacoes')
110 router.register('emw', EmwViewSet, basename='Emw')
```

Com a conclusão deste capítulo, foi possível observar todas as etapas de desenvolvimento da solução bem como a integração entre elas. No capítulo seguinte, apresentaremos a implementação desses serviços, bem como testes e resultados que conseguimos obter até o momento.

4 IMPLEMENTAÇÃO DOS SERVIÇOS

A implementação é o processo em que se inicia a utilização de um novo produto, esse período está ligado ao funcionamento adequado do sistema, mas também à etapa de escolha da solução ideal, assim, consegue-se garantir que esse produto é o mais adequado às necessidades. Nesta etapa será descrito todo o processo para implementação dos serviços.

4.1 Hardwares

Existem dois hardwares no sistema, o primeiro é o gateway e o segundo é a estação meteorológica composta por diversos outros componentes. Abaixo vamos detalhar qual a responsabilidade que cada um deles exercerá além de como configurá-los para ter um bom funcionamento.

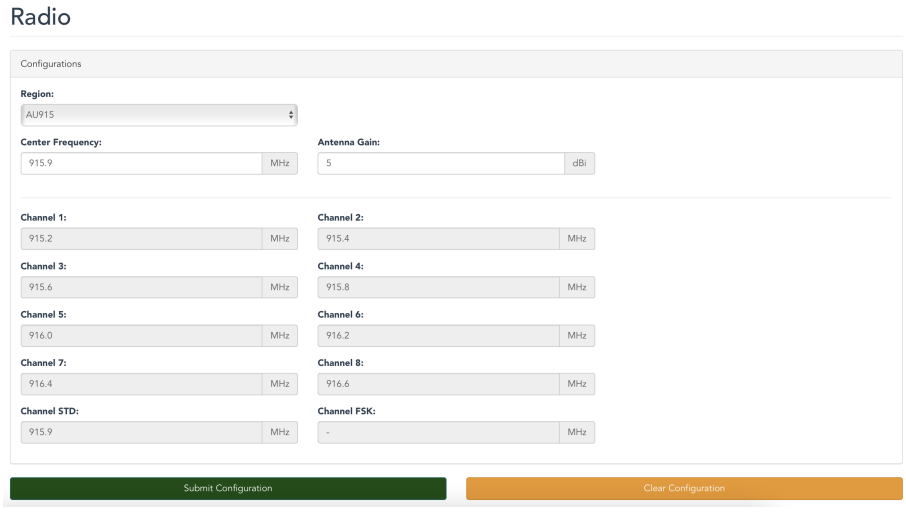
4.1.1 Gateway

O gateway escolhido para implementação do projeto foi o ITG-200 fabricado pela Khomp, este equipamento tem a tarefa de encaminhar as mensagens enviadas pelas estações meteorológicas ao sistema Iot. O gateway exerce um papel fundamental no projeto, graças a ele é possível explorar uma tecnologia *Low Power/Lora* com a possibilidade de utilizar dispositivos que consomem pouca energia e possuem a necessidade de se comunicar por uma distância considerável.

Para utilizar este equipamento foi necessário configurá-lo de acordo com manual, inserindo algumas informações como o dispositivo a conectar, encaminhamento das mensagens, frequência de operação e outros. Na sequência vamos descrever os passos executados para configuração do equipamento.

O primeiro passo foi configurar as frequências que o equipamento irá operar, no Brasil foi definido como padrão as mesmas adotadas na Austrália, portanto foi cadastrado o aparelho para operar em 915,9MHz, com oito canais saltando em 200KHz iniciando em 915,2MHz e finalizando em 916,6MHz.

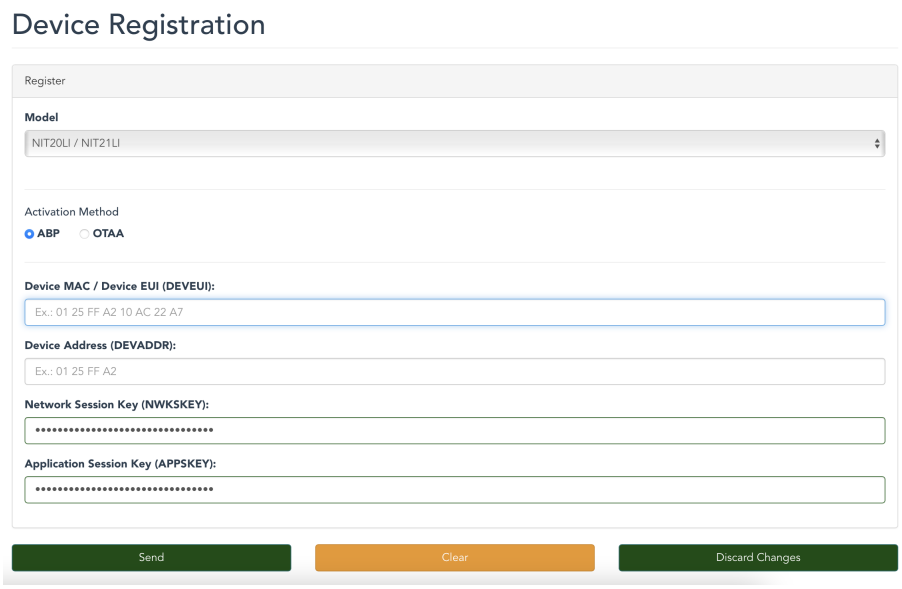
Figura 29 – Configuração Frequências ITG200



Fonte: Interface configuração ITG200

Com as frequências já definidas, nosso próximo passo foi inserir o *endpoint* na lista de dispositivos do equipamento, somente após este procedimento é possível autenticar o dispositivo na rede *lora*. Como estamos trabalhando com equipamentos Khomp, neste caso apenas será necessário entrar com o endereço MAC [1] do dispositivo no campo *Device MAC/Device EUI (DEVEUI)* os demais serão preenchidos automaticamente.

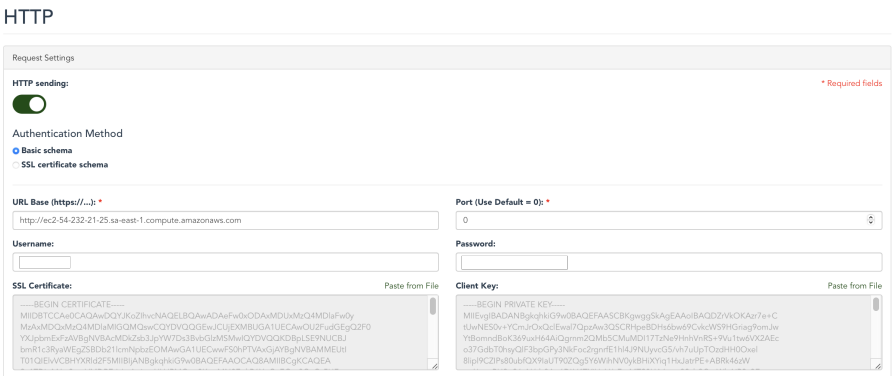
Figura 30 – Configuração Dispositivos ITG200



Fonte: Interface configuração ITG200

Por fim configuramos a integração do gateway com um provedor externo, neste caso a API-IOT, neste caso devemos ter a certeza de que a Api esteja funcionando corretamente e seu endereço seja conhecido. Neste caso em particular todas as mensagens serão enderaçadas para <http://ec2-54-232-21-25.sa-east-1.compute.amazonaws.com> seguindo as regras do protocolo http na camada de aplicação e o protocolo TCP/IP na camada de transporte.

Figura 31 – Configuração Integração ITG200



Fonte: Interface configuração ITG200

[1] Um endereço de controle de acesso ao meio (endereço MAC) de um dispositivo é um identificador único atribuído a uma interface de rede (ou Network Interface Controller - NIC). Para comunicações dentro de um segmento de rede, é usado como endereço de rede para a maioria das tecnologias de rede IEEE 802, incluindo Ethernet, Wi-Fi e Bluetooth.(WIKIPEDIA, 2021)

4.1.2 Estação Meteorológica

A estação meteorológica utilizada foi fornecida pela Khomp, para garantir um bom funcionamento do projeto a estação necessita de uma série de hardwares adicionais, nas Figura 32 e Figura 33 pode-se observar os componentes envolvidos e sua nomenclatura.

Figura 32 – Estação Meteorológica Khomp

Fonte: Foto retirada pelo próprio autor

Figura 33 – Case Estação Meteorológica Khomp

Fonte: Foto retirada pelo próprio autor

O sistema de alimentação da estação meteorológica é composto por painel solar, controlador de carga e bateria. A energia solar é convertida em energia elétrica através do painel solar e armazenada na

bateria (ver especificações nas Tabela 6, Tabela 7 e Tabela 8). O diagrama esquemático do sistema de alimentação pode ser visto na Figura 34.

Tabela 6 – Especificações do Painel Solar

Parâmetro	**Especificação**
Máxima Potência	10 W
Voltagem de Máxima Potência	19 V
Corrente de Máxima Potência	0,53 A
Eficiência do Painel	11,34%
Coeficiente de Temperatura da Potência(Pm)	< -0,423 % / °C
Coeficiente de Temperatura da Corrente(Isc)	< -0,039 % / °C
Coeficiente de Temperatura da Voltagem(Voc)	< -0,307 % / °C
Temperatura de operação	45 °C
Dimensões (CxLxA)	245 x 375 x 25 mm
Peso	1,34 Kg

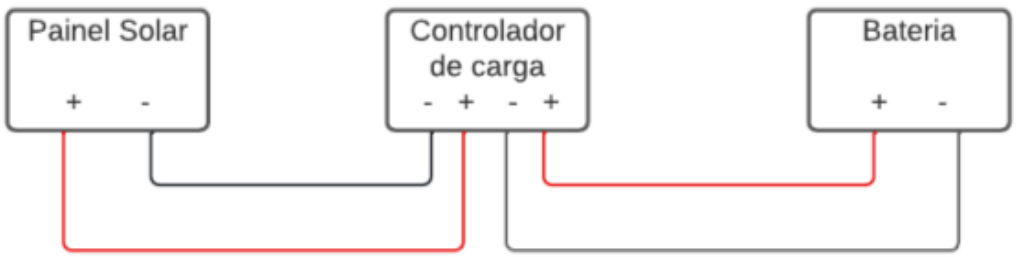
Tabela 7 – Especificações do Controle de Carga

Parâmetro	**Especificação**
Corrente nominal	5 A
Tensão de operação	12 V
Potência	60 W
Máxima tensão das baterias	16 V
Autoconsumo	< 6 mA
Temperatura de operação	-35 °C a 55 °C
Porta USB frontal	5 V / 1,2 A
Dimensões (CxLxA)	101,2x67x21,8 mm
Peso	80 g

Tabela 8 – Especificações da Bateria

Parâmetro	**Especificação**
Corrente máxima	56 A
Tensão total	12 V
Quantidade de células	6
Máxima tensão das baterias	16 V
Temperatura de operação	-20 °C a 60 °C
Dimensões (CxLxA)	151×100×65 mm
Peso	1800 g

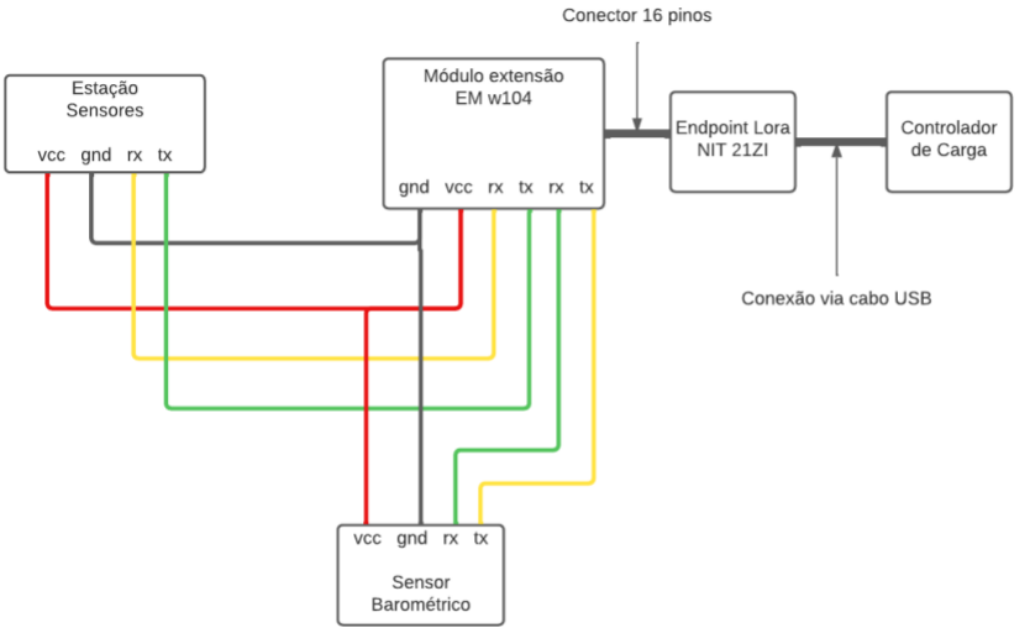
Figura 34 – Diagrama Esquemático - Sistema Alimentação



Fonte: Próprio autor, desenvolvida em Ludichart

Na outra parte estão todos os componentes responsáveis pela leitura e transmissão dos dados climáticos. Na estação tem-se os sensores de velocidade e sentido do vento, sensor pluviométrico, sensor de temperatura e umidade, o sensor de pressão atmosférica é instalado a parte. No case existem outros dispositivos como o módulo de extensão (EM W104) que é responsável por processar os dados da estação e o *endpoint* Lora, NIT 21ZI(Ver especificações Tabela 9) responsável por encaminhar a mensagem ao *gateway* mais próximo. Na Figura 35 ilustra-se o diagrama esquemático da estação.

Figura 35 – Diagrama Esquemático - Estação



Fonte: Próprio autor, desenvolvida em Ludichart

Tabela 9 – Especificações do NIT 21ZI

Parâmetro	**Especificação**
Potência de Transmissão	10 dBm
Consumo dormindo	6 μA
Consumo trnsmitindo	6 mA
Alimentação	5 a 12 V
Taxa de dados	250 kbps máximo
Dimensões (CxLxA)	69x63x38 mm

4.2 Softwares

Para manter o sistema operando com garantia, optou-se por utilizar os serviços de hospedagem da *AWS*, por ser uma empresa já consolidada no mercado e oferecer diversas soluções necessárias para o projeto como o *elastic beanstalk* [2] e *S3 bucket* [3] que ajudam bastante na hora de fazer o *deploy*.

[2] O AWS Elastic Beanstalk é um serviço de fácil utilização para implantação e escalabilidade de aplicações e serviços da web desenvolvidos com Java, .NET, PHP, Node.js, Python, Ruby, Go e Docker em servidores familiares como Apache, Nginx, Passenger e IIS. (AWS, 2021b)

[3] O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade, disponibilidade de dados, segurança e performance líderes do setor. Clientes de todos os portes e setores podem armazenar

No sistema existe uma conta "super-admin", para acessa-la basta entrar no endereço <http://front-core-projetoagro.s3-website-sa-east-1.amazonaws.com/login>, utilizando como usuário "projetoagro"e senha "monitoramento2021".

4.3 Resultados

Nesta seção serão apresentados os resultados e uma validação dos dados capturados pela estação.

4.3.1 Cenário do MVP

Como o projeto envolve uma parte de hardware e existe a necessidade de uma comunicação entre a estação meteorológica e o *gateway* foi elaborado um cenário que não significa o que será encontrado em campo, porém, foi o mais próximo da realidade que consegui-se desenvolver.

Figura 36 – Localização dos Equipamentos

Fonte: google earth adaptada

O conjunto de soluções se demonstrou eficiente, tem-se noção de que a realidade enfrentada não seria esta, praticamente não existe obstruções do sinal e a distância é curta, a realidade é outra, em campo tem-se distâncias quilométricas e na maioria dos casos a visão entre a estação e o *gateway* não é direta, causando impacto na qualidade do sinal.

4.3.2 Transmissão dos dados

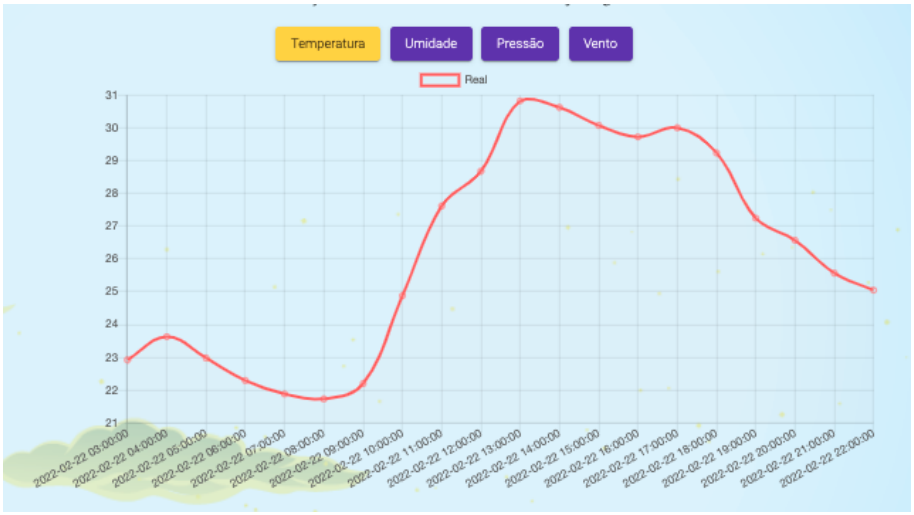
Diante deste cenário o conjunto da solução apresentou resultados satisfatórios, no período em que a estação ficou operando o sistema se demonstrou eficiente, não deixando que nenhuma mensagem fosse perdida, em um prazo de 30 dias houve uma troca de 35,4 mil mensagens, sendo estas gerando mais de 139,9 mil documentos armazenados.

e proteger qualquer quantidade de dados de praticamente qualquer caso de uso, como data lakes, aplicações nativas da nuvem e aplicações móveis. (AWS, 2021a)

4.3.3 Validação dos Dados

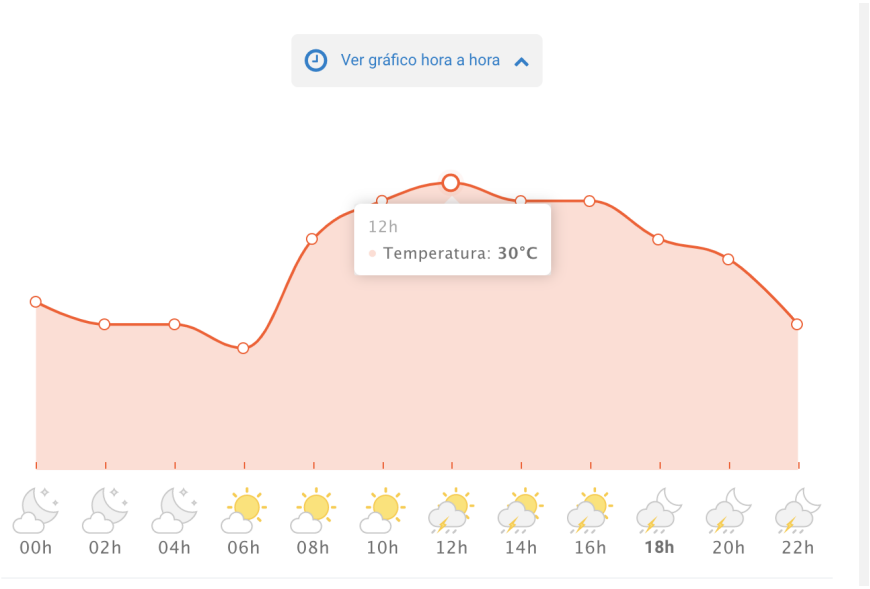
A estação se demonstrou confiável em termos de integração com o sistema, porém em relação os dados capturados não foi possível afirmar sua precisão com exatidão, na Figura 38 temos um demonstrativo dos dados sendo apresentados por um serviço de monitoramento bastante conhecido (Clima Tempo) e na Figura 37 temos os dados capturados pela estação, ambos disponíveis para realizar um comparativo.

Figura 37 – Gráfico da Temperatura



Fonte: sorftware desenvolvido <http://front-core-projetoagro.s3-website-sa-east-1.amazonaws.com/>

Figura 38 – Gráfico Temperatura - Clima Tempo



Fonte: climatempo - adaptada <https://www.climatempo.com.br/>

Como podemos ver existe uma semelhança no comportamento dos gráficos, isto nos deixa a entender que o sistema está operando corretamente, porém, para ter esta certeza é necessário extrair dados de sensores calibrados e capturando os dados na mesma região, para então compará-los.

Por estes fatores não foram calculado os erros obtidos na estação, este comportamento acabou se repetindo com as demais variáveis.

O fabricante da estação não nos forneceu nenhum certificado de calibração dos sensores e tão pouco algum dado que comprove a veracidade dos dados, portanto nos garantiu que o dispositivo captura os dados dentro das margens de erros citadas na Tabela 4.

4.3.4 Análise dos Sistemas

Os sistemas se demonstraram eficientes funcionando perfeitamente como esperado, as consultas e apresentações dos dados, seguiram o fluxo previsto demonstrando boa integração entre os sistemas.

Não foi possível executar um teste de carga e performance nas Api's, porém o requisito do projeto consistia em garantir um acesso de 20 usuários simultâneos, isto a própria aplicação nos garantirá. O tempo de resposta se demostrou dentro do esperado, fazendo com que o usuário obtenha os resultados praticamente ao clicara na opção, salvo em alguns casos onde exige um maior processamento.

Um ponto dos requisitos não atendido no projeto, foi a não autenticação entre o sistema core e o sistema iot, deixando o sistema vulnerável, podendo ocasionar problemas sérios, porém os bancos estão protegidos com senhas, o que acaba dificultando em certo ponto os ataques.

5 CONCLUSÃO

O agronegócio no Brasil sofre muito com carência de tecnologia, apesar do setor estar passando por modificações, ainda existem diversos outros fatores dificultam ainda mais este avanço, um exemplo clássico disso é a falta de cobertura de algum sinal nas áreas rurais.

O que mais chama a atenção é o fato de termos a agricultura como um dos principais eixos da economia no Brasil e a tecnologia ainda ser novidade para muitos produtores, mesmo com programas de incentivo do governo federal, nota-se que a procura por tecnologia ainda é baixa e aplica-la em suas propriedades ainda não é algo trivial.

Para se ter uma ideia o receita bruta anual de um produtor de pequeno porte não deve passar de R$500.000,00, já um produtor de médio porte vai de R$500.000,00 até R$2.400.000,00 e um produtor de grande porte acima de R$2.400.000,00 (MINISTéRIODAECONOMIA, 2020), portanto estamos falando de empresas que movimentam quantias consideráveis no mercado, mesmo assim para parte dos agricultores investimentos em tecnologia são considerados supérfluo, ainda não tão natural, pois é necessário ao fornecedor do serviço/tecnologia demonstrar que sua solução gera de fato um ganho/otimização na produção e que o custo da solução tecnológica será recuperado com sua implantação.

Por possuir um custo elevado em hardwares, a necessidade de um investimento no projeto é crucial para a proposta se tornar economicamente viável. De acordo com o estudo de caso realizado neste projeto (Apêndice C), fazendo um levantamento cada cliente terá um custo inicial na casa dos R$25.000,00, portanto viabilizar esta ideia vai além de vender uma solução para o monitoramento agro, precisa-se converter todo o investimento inicial em resultados para o agricultor.

Este projeto tem a intenção de se tornar uma ferramenta adicional para o agronegócio no Brasil, monitorar as variáveis climáticas é fundamental para o desenvolvimento da cultura e tomar decisões baseadas em precisões e não em intuições deve estar em destaque na hora de ofertar o produto.

Outro fator de destaque no projeto é o meio de comunicação escolhido, por possuir a característica de ofertar uma cobertura de sinal em uma escala quilométrica e possibilitar com apenas um *gateway* cobrir uma área de aproximadamente 12 km de raio, estes fatores acabaram sendo fundamentais na escolha desta tecnologia.

Um fato observado em uma análise de mercado foi a constatação de poucas empresas utilizarem esta tecnologia, o que acaba acendendo um alerta, se esta tecnologia é tão boa, por que não é muito utilizada pelas empresas. Uma possibilidade de resposta pode ser que fazer a cobertura desta tecnologia na área rural brasileira não seja atrativo comercial para empresas que oferecem serviços de conectividade LoRaWAN.

O projeto ficou dividido em camadas, deixando claro qual a responsabilidade de cada componente, a escolha de uma arquitetura desta forma possibilita um ganho em escalabilidade, tornando possível o desenvolvimento de futuras funcionalidades. A distribuição dos sistemas possibilita uma evolução tecnológica mais simples, onde cada sistema pode ser aprimorado de maneira individual, apesar de estarem todos interligados e existir uma dependência entre eles o seu aprimoramento não está diretamente relacionado.

Já a divisão do projeto por componentes proporcionou uma visão ampla de sua complexidade, desenvolver e manter um projeto desta magnitude exige um conhecimento em diversas áreas como

hardwares, computação em nuvem, desenvolvimento e integração de sistemas.

A escolha dos frameworks de desenvolvimento foi estratégica, pela familiaridade com as linguagens e segundo hotframeworks (2022), estarem entre os mais utilizados nos dias de hoje, facilitando na procura por conteúdo de suporte ao desenvolvimento. A estação demonstrou um bom funcionamento, a veracidade dos dados e o consumo baixo de energia foram destaques para nos provar que ela atende aos requisitos do projeto, com testes massivos em diversas regiões ela demonstrou um funcionamento de forma sólida, operando por 24 horas sem interrupções.

A opção por utilizar um servidor AWS para a instalação dos servidores premitiu que o projeto se beneficie dos avanços tecnológicos que a plataforma oferece, ajudando na integração e armazenamento dos dados, porém a maior vantagem é a utilização de equipamentos terceirizados evitando a necessidade de adquiri-los e administrá-los, além disso, foi possível executar os serviços em contas de teste para validação da solução.

Por fim, o projeto se demonstrou-se confiável, o processo de validação atendeu as exigências descritas nos requisitos e a experiência de navegação pela plataforma foi agradável, mesmo precisando de uma melhora na interface gráfica atendeu as exigências do projeto oferecendo um entendimento claro das necessidades do cliente.

Apesar dos resultados terem sido satisfatórios, a solução ainda exige muito esforço para chegar ao nível de um produto comercial, tendo, em um próximo passo, que passar por uma prova de conceito em um cliente real. Além disso, temos consciência de que fazer esse produto chegar ao agricultor dependerá também do retorno de seu impacto na produção e retorno financeiro que a solução vai oferecer ao cliente. É importante destacar também que o TCC do aluno Marcelo Bittencourt será complementar a este TCC a funcionalidade de predição dos parâmetros e será incorporada a este produto para torná-lo mais atrativo ao produtor.

Para futuros trabalhos, além das melhorias nos componentes do sistema, pode pensar em ampliar as formas de conectividade da estação, como por exemplo, a utilização de módulos de comunicação celular ou via satélite. Isto traria mais flexibilidade ao produto pois não deixamos a solução presa a uma única tecnologia de comunicação.

Por se tratar de um projeto utilizando do conceito MVP, optou-se por desenvolver diagramas ilustrativos (ver Figura 12, Figura 13 e Figura 14), porém em trabalhos futuros deve-se levar em conta uma possível mudança, apresentando diagramas padronizados de acordo com normas técnicas.

REFERÊNCIAS

ABSTARTUPS. *Business Model Canvas: Aprenda Na Prática + Baixe Modelo*. Abstartups, 2019. Disponível em: <https://abstartups.com.br/abseducacao-business-model-canvas/>. Citado na página 34.

ANATEL. *Agência Nacional de Telecomunicações*. 2020. Disponível em: <https://informacoes.anatel.gov.br/paineis/infraestrutura/panorama>. Citado na página 23.

ANGULAR. *Introduction to the Angular Docs*. 2020. Disponível em: <https://angular.io/docs>. Citado na página 55.

AVINCO, H. M. *Havinco Consultoria ME*. 2022. Havinco Consultoria ME. Citado na página 37.

AWS. *Amazon S3*. 2021. Disponível em: <https://aws.amazon.com/pt/s3/>. Citado na página 68.

AWS. *AWS Elastic Beanstalk*. 2021. Disponível em: <https://aws.amazon.com/pt/elasticbeanstalk/>. Citado na página 67.

AWS. *Quem usa a computação em nuvem*. Amazon Web Services, 2021. Disponível em: <https://aws.amazon.com/pt/what-is-cloud-computing/>. Citado na página 33.

AZURE. *O que é computação em nuvem*. Azure, 2021. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-cloud-computing/#uses>. Citado na página 33.

BARRETT, M. L. PUTTING NON-FUNCTIONAL REQUIREMENTS TO GOOD USE*. *Department of Computer and Information Sciences East Tennessee State University*, December 2002. Citado na página 45.

BERNARDI, A. C. d. C. et al. *Agricultura de Precisão - Resultados de um Novo Olhar*. [S.l.]: Embrapa, 2014, 2014. Citado na página 25.

BRITO, M. *Spring Boot Da API REST aos Microservices*. 1. ed. Alfenas: Michelli Brito, 2021. Acesso em: 21 ago 2021. Citado na página 28.

BROTO. *Agrometeorologia: como interpretar as previsões de clima e tempo no campo*. Broto, 2021. Disponível em: <https://blog.broto.com.br/agrometeorologia/>. Citado na página 33.

CAMARGO, R. *O que é Canvas? E como pode auxiliar em seus projetos?* Robson Camargo Projetos e Negócios, 2019. Disponível em: <https://robsoncamargo.com.br/blog/O-que-e-Canvas>. Citado na página 34.

CASTELLANI, S. *axway-API*. Axway Blog, 2020. Disponível em: <https://blog.axway.com/amplify-products/api-management/different-types-apis>. Citado na página 28.

CIêNCIA; CLIMA. *Importância do microclima para a adaptação ao aquecimento global*. Ciencia e clima, 2018. Disponível em: <https://cienciaeclima.com.br/importancia-microclima-adaptacao-aquecimento-global/>. Citado na página 33.

COMER, D. E. *Redes de Computadores e Internet*. 6. ed. Rio Grande do Sul/RS: Bookman, 2006. Acesso em: 21 ago 2021. Citado na página 27.

DEVMEDIA. *HTTP: Verbos*. Devmedia, 2020. Disponível em: <https://www.devmedia.com.br/http-verbos/41224>. Citado 2 vezes nas páginas 29 e 30.

EMBRAPA. *Embrapa em números*. 2021. Disponível em: <https://www.embrapa.br/embrapa-em-numeros>. Citado na página 23.

EN, W. *Gateway*. 2022. Disponível em: <https://en.wikipedia.org/wiki/Gateway_(telecommunications)>. Citado na página 41.

EVERYNET. *Revolutionize IoT in Brazil*. 2020. Disponível em: <https://www.everynet.com/everynet-home-br>. Citado na página 50.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. *IEEE Wireless Communications*, October 2000. Citado na página 28.

FILHO, J. E. R. V.; GASQUES, J. G. *Uma Jornada Pelos Contrastes do Brasil: Cem anos do Censo Agropecuário*. [S.l.]: IPEA, IBGE, 2020, 2020. Citado na página 23.

FONSECA, W. *O que é token?* 2009. Disponível em: <https://www.tecmundo.com.br/senha/3077-o-que-e-token-.htm>. Citado na página 46.

FOROUZAN, B. A. *Protocolo TCP/IP*. 3. ed. Instituto de Matemática e Estatística da Universidade de São Paulo: AMGH Editora Ltda, 2010. Acesso em: 21 ago 2010. Citado na página 49.

GISERMAN, L.; GALVES, B.; JAOLINO, A. *Conclusões Finais*. www.gta.ufrj.br, 2021. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/lora/conclusoes.html>. Citado na página 26.

GRIGORIK, I. *Making the Web Faster with HTTP 2.0*. 1. ed. California: ILYA GRIGORIK, 2013. Acesso em: 21 ago 2021. Citado na página 29.

HOTFRAMEWORKS. *Find your new favorite web framework*. 2022. Disponível em: <https://hotframeworks.com/>. Citado na página 72.

IBGE. *Censo Agropecuário*. 2018. Disponível em: <https://www.ibge.gov.br/estatisticas/economicas/agricultura-e-pecuaria/21814-2017-censo-agropecuario.html?edicao=25757&t=destaques>. Citado na página 23.

IBGE. *PIB do setor agropecuário cresce 1,3% em 2019*. 2020. Disponível em: <https://www.gov.br/pt-br/noticias/financas-impostos-e-gestao-publica/2020/03/pib-do-setor-agropecuario-cresce-1-3-em-2019>. Citado na página 23.

IEEE. *IEEE 90*. 2021. Disponível em: <https://standards.ieee.org/standard/90-0.html>. Citado na página 41.

KHOMP. *Estação Meteorológica*. 2019. Disponível em: <https://www.khomp.com/wp-content/uploads/2020/08/Estacao_Meteorologica_-_PT_v3.pdf>. Citado 2 vezes nas páginas 50 e 51.

KHOMP. *ITG 200 Everynet Indoor*. 2019. Disponível em: <https://www.khomp.com/pt/produto/itg-200-everynet-indoor/>. Citado na página 51.

LAMPARELLI, R. A. C. *Agricultura de precisão*. Agência Embrapa de Informação Tecnológica, 2016. Disponível em: <http://www.agencia.cnptia.embrapa.br/gestor/cana-de-acucar/arvore/CONTAG01_72_711200516719.html>. Citado na página 26.

LIMA, J. F. D.; MOURA, L. P. D. *Implementando um Mural Eletrônico em PHP: Uma Aplicação Voltada a uma Instituição de Ensino Superior*. 2017. UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ. Citado na página 31.

MINISTéRIODAECONOMIA. *Ministério da Agricultura, Pecuária e Abastecimento*. 2020. Disponível em: <https://www.gov.br/agricultura/pt-br/assuntos/noticias/cmn-aprova-elevacao-dos-limites-da-receita-anual-para-efeito-da-classificacao-do-produtor-rural>. Citado na página 71.

MYSQLAB. *Manual de Referência do MySQL*. MySQLAB, 2006. Disponível em: <https://downloads.mysql.com/docs/refman-4.1-pt.a4.pdf>. Citado na página 31.

NETO, A. C. D. *Modelagem de Dados Tutorial*. 2011. Disponível em: <https://www.devmedia.com.br/modelagem-de-dados-tutorial/20398>. Citado na página 58.

ORM. *Mapeamento objeto-relacional: como funciona e técnicas*. 2019. Disponível em: <https://blog.geekhunter.com.br/mapeamento-objeto-relacional/>. Citado na página 59.

OSTERWALDER, A.; PIGNEUR, Y. *Business Model Generation: A Handbook for Visionaries, Game Changers, and Challengers*. Wiley, 2010. (The Strategyzer Series). ISBN 9780470901038. Disponível em: <https://books.google.com.br/books?id=fklTInjiPQAC>. Citado na página 34.

PEREIRA, M. *Análise de mercado: o que é e para que serve?* Sebrae, 2021. Disponível em: <https://comunidadesebrae.com.br/blog/analise-de-mercado-o-que-e-e-para-que-serve>. Citado na página 34.

SEBRAE. *Protótipo e MVP*. 2019. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/ufs/pr/artigos/prototipo-e-mvp,6e3fcfda70d8d610VgnVCM1000004c00210aRCRD>. Citado na página 37.

SEMTECH. *Semtech Corporation*. Semtech Corporation, 2015. Disponível em: <https://www.frugalprototype.com/wp-content/uploads/2016/08/an1200.22.pdf>. Citado na página 26.

SIGNIFICADOS.COM. *Significado de Stakeholders*. Robson Camargo Projetos e Negócios, 2019. Disponível em: <https://www.significados.com.br/stakeholder/>. Citado na página 34.

SOUTO, M. *O que é front-end e back-end?* 2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end?gclid=CjwKCAiAz--OBhBIEiwAG1rIOjFI7Q7everxWgUjzaQAQZSMRcikq31blQ8jQrO2nGW-1glequuLqRoCy88QAvD_BwE>. Citado na página 54.

TOCANTINS, I. de Desenvolvimento Rural do Estado do. *Agroindústria*. 2020. Disponível em: <https://www.to.gov.br/ruraltins/agroindustria/4j6ipzekiniz>. Citado na página 23.

WIKIPEDIA. *CRUD*. 2011. Disponível em: <https://pt.wikipedia.org/wiki/CRUD>. Citado na página 57.

WIKIPEDIA. *Endereço MAC*. 2021. Disponível em: <https://pt.wikipedia.org/wiki/Endere%C3%A7o_MAC>. Citado na página 64.

Apêndices

APÊNDICE A – APÊNDICE A - CÓDIGOS SISTEMA CORE

Abaixo será listado os códigos para o desenvolvimento do sistema core, desenvolvidas na linguagem python os códigos representam as camadas listadas no sistema back-end. O acesso a todos os códigos está disponível em <https://github.com/projeto-agro-tcc/backend-core>.

Código A.1 – Arquivo URL

```
router = routers.DefaultRouter()
router.register('empresas', EmpresasViewSet, basename='Empresas')
router.register('usuarios', UsuariosViewSet, basename='Usuarios')
router.register('estacoes', EstacoesViewSet, basename='Estacoes')

urlpatterns = [
    path('', include(router.urls)),
    path('admin/', admin.site.urls),
    path('auth/', CustomAuthToken.as_view()),
]
```

Código A.2 – ViewSet Usuários

```
class UsuariosViewSet(ModelViewSet):
    queryset = Usuario.objects.all()
    serializer_class = UsuarioSerializer
    authentication_classes = (TokenAuthentication,)

    def create(self, request, *arg, **kwargs):
        try:
            user = UsuarioService.from_dto(request.data)
            UsuarioService.save_usuario(user)
            my_group = Group.objects.get(name='less_user')
            my_group.user_set.add(user)
            Token.objects.create(user=user)
            serializer = UsuarioSerializer(user)
            response = {'message': 'User Created', 'result': serializer.data}
            return Response(response, status=status.HTTP_200_OK)
        except Exception as err:
            raise CustomValidation(err, 'detail', status_code=status.HTTP_400_BAD_REQUEST)

    @authenticated_user
    def update(self, request, *args, **kwargs):
        try:
            user = Usuario.objects.filter(id=kwargs['pk'])[0]
            user = UsuarioService.from_dto_update(request.data, user)
            UsuarioService.save_usuario(user)
            serializer = UsuarioSerializer(user)
            response = {'message': 'User Updated', 'result': serializer.data}
            return Response(response, status=status.HTTP_200_OK)
        except Exception as err:
            raise CustomValidation(err, 'detail', status_code=status.HTTP_400_BAD_REQUEST)

    @authenticated_user
```

```
152     @allowed_users_by_group(allowed_roles=['admin', 'super_user'])
153     def list(self, request, *args, **kwargs):
154         queryset = Usuario.objects.filter(is_active=1)
155         serializer = UsuarioSerializer(queryset, many=True)
156         return Response(serializer.data)
157 
158     @authenticated_user
159     @allowed_users_by_group(allowed_roles=['admin', 'super_user'])
160     def destroy(self, request, *args, **kwargs):
161         try:
162             user = Usuario.objects.filter(id=kwargs['pk'])[0]
163             UsuarioService.delete_user(user)
164             return Response(status=status.HTTP_200_OK)
165         except Exception as err:
166             raise CustomValidation(err, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
```

Código A.3 – ViewSet Empresas

```
167 class EmpresasViewSet(ModelViewSet):
168     queryset = Empresa.objects.all()
169     serializer_class = EmpresaSerializer
170     authentication_classes = (TokenAuthentication,)
171 
172     @authenticated_user
173     @allowed_users_by_group(allowed_roles=['admin', 'super_user'])
174     def create(self, request, *arg, **kwargs):
175         try:
176             empresa = EmpresaService.from_dto(request.data)
177             EmpresaService.save_empresa(empresa)
178             serializer = EmpresaSerializer(empresa)
179             response = {'message': 'Empresa Created', 'result': serializer.data}
180             return Response(response, status=status.HTTP_200_OK)
181         except Exception as error:
182             raise CustomValidation(error, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
183 
184     @authenticated_user
185     @allowed_users_by_group(allowed_roles=['admin', 'high_user', 'super_user'])
186     def update(self, request, *args, **kwargs):
187         try:
188             empresa = Empresa.objects.filter(id=kwargs['pk'])[0]
189             empresa = EmpresaService.from_dto_update(request.data, empresa)
190             EmpresaService.save_empresa(empresa)
191             serializer = EmpresaSerializer(empresa)
192             response = {'message': 'Empresa Updated', 'result': serializer.data}
193             return Response(response, status=status.HTTP_200_OK)
194         except Exception as error:
195             raise CustomValidation(error, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
196 
197     @authenticated_user
198     @allowed_users_by_group(allowed_roles=['admin', 'super_user'])
199     def destroy(self, request, *args, **kwargs):
200         try:
201             empresa = Empresa.objects.filter(id=kwargs['pk'])[0]
202             EmpresaService.delete_empresa(empresa)
203             return Response(status=status.HTTP_200_OK)
204         except Exception as err:
205             raise CustomValidation(err, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
206 
207     # Verifica se usuário é super admin
```

```
208     # caso sim retorna todas as empresas do contrário retorna suas empresas cadastradas
209     @authenticated_user
210     def list(self, request, *args, **kwargs):
211         role_list = request.user.groups.all()
212         for role in role_list:
213             if role.name in ['admin', 'super_user']:
214                 queryset = Empresa.objects.filter(status=1)
215                 serializer = EmpresaSerializer(queryset, many=True)
216                 return Response(serializer.data)
217         else:
218             queryset = Empresa.objects.filter(status=1).filter(usuario=request.user)
219             serializer = EmpresaSerializer(queryset, many=True)
220             return Response(serializer.data)
221 
222     @authenticated_user
223     @action(detail=False, methods=['GET'])
224     def get_users_by_idempresa(self, request):
225         print("Endpoint acessado")
226         response = {'message': 'It is working'}
227         return Response(response, status=status.HTTP_200_OK)
```

Código A.4 – ViewSet Estações

```
228 class EstacoesViewSet(ModelViewSet):
229     queryset = Estacao.objects.all()
230     serializer_class = EstacaoSerializer()
231     authentication_classes = (TokenAuthentication,)
232 
233     def list(self, request, *args, **kwargs):
234         queryset = Estacao.objects.all()
235         serializer = EstacaoSerializer(queryset, many=True)
236         return Response(serializer.data)
237 
238     @action(detail=False, methods=['GET'])
239     def get_data(self, request):
240         response = requests.get("http://127.0.0.1:7000/endpoints/get_data/")
241         response = {'response': response}
242         return Response(response, status=status.HTTP_200_OK)
```

Código A.5 – Model Estação

```
243 class Estacao(models.Model):
244     latitude = models.IntegerField()
245     longitude = models.IntegerField()
246     serial_number = models.CharField(max_length=100, unique=True)
247     empresa = models.ForeignKey(Empresa, on_delete=models.CASCADE, blank=True, related_name='estacoes'
        )
248     estacao_modelo = models.ForeignKey(EstacaoModelo, on_delete=models.CASCADE, blank=True, null=True)
249 
250     class Meta:
251         db_table = "en_estacoes"
252 
253     def __str__(self):
254         return self.serial_number
```

Código A.6 – Model Endereço

```
255 class Endereco(models.Model):
```

```
256     logradouro = models.CharField(max_length=150, null=False)
257     numero = models.CharField(max_length=10, null=False)
258     complemento = models.CharField(max_length=50, null=False)
259     bairro = models.CharField(max_length=100, null=False)
260     cidade = models.CharField(max_length=100, null=False)
261     cep = models.CharField(max_length=8, null=False)
262     uf = models.CharField(max_length=2, null=False)
263 
264     class Meta:
265         db_table = "en_enderecos"
266 
267     def __str__(self):
268         return self.logradouro
```

Código A.7 – Model Empresa

```
269 class Empresa(models.Model):
270     nome = models.CharField(max_length=150)
271     cnpj = models.CharField(max_length=18, unique=True)
272     razao_social = models.CharField(max_length=100)
273     email = models.EmailField(null=True, unique=True)
274     web_site = models.CharField(max_length=200, unique=True)
275     endereco = models.OneToOneField(Endereco, on_delete=models.CASCADE, null=False)
276     telefone = models.OneToOneField(Telefone, on_delete=models.CASCADE, null=False)
277     status = models.IntegerField(default=1)
278 
279     class Meta:
280         db_table = "en_empresas"
281 
282     def __str__(self):
283         return self.nome
```

Código A.8 – Model Telefone

```
284 
285 class Telefone(models.Model):
286     residencial = models.CharField(max_length=14, null=True, blank=True)
287     celular = models.CharField(max_length=12, null=False)
288     outro = models.CharField(max_length=14, null=True, blank=True)
289 
290     class Meta:
291         db_table = "en_telefones"
292 
293     def __str__(self):
294         return self.celular
```

Código A.9 – Model Usuario

```
295 class Usuario(AbstractUser):
296     cpf = models.CharField(max_length=11, unique=True)
297     status = models.IntegerField(default=1)
298     endereco = models.OneToOneField(Endereco, on_delete=models.CASCADE, null=True, blank=True)
299     telefone = models.OneToOneField(Telefone, on_delete=models.CASCADE, null=True, blank=True, unique=
        True)
300     empresas = models.ManyToManyField(Empresa, blank=True, null=True)
301 
302     class Meta:
303         db_table = "en_usuarios"
```

```
304
305     def __str__(self):
306         return self.username
```

Código A.10 – Service Usuario

```
307 class UsuarioService:
308
309     def __init__(self):
310         pass
311
312     def from_dto(objDto):
313         error = UsuarioService.validate_user_and_password(objDto)
314         if error:
315             raise CustomValidation(error, 'detail', status_code=status.HTTP_409_CONFLICT)
316
317         try:
318             user = Usuario()
319             user.endereco = EnderecoService.from_dto(objDto)
320             user.telefone = TelefoneService.from_dto(objDto)
321             user.cpf = objDto['cpf']
322             user.email = objDto['email']
323             user.username = objDto['username']
324             user.first_name = objDto['first_name']
325             user.last_name = objDto['last_name']
326             user.set_password(objDto['password'])
327             return user
328         except Exception as error:
329             raise CustomValidation(error, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
330
331
332     def from_dto_update(objDto, user):
333         try:
334             user.cpf = objDto['cpf']
335             user.email = objDto['email']
336             user.username = objDto['username']
337             user.first_name = objDto['first_name']
338             user.last_name = objDto['last_name']
339             endereco = EnderecoService.from_dto_update(objDto, user.endereco)
340             telefones = TelefoneService.from_dto_update(objDto, user.telefone)
341             TelefoneService.save_telefones(telefones)
342             EnderecoService.save_endereco(endereco)
343             user.endereco = endereco
344             user.telefone = telefones
345             return user
346         except Exception as error:
347             raise error
348
349     def validate_user_and_password(objDto):
350         error = []
351         if Usuario.objects.filter(username=objDto['username']).exists():
352             error.append("Username is not unique")
353         if Usuario.objects.filter(email=objDto['email']).exists():
354             error.append("Email is not unique")
355         if Usuario.objects.filter(cpf=objDto['cpf']).exists():
356             error.append("CPF is not unique")
357         return error
358
359     def save_usuario(usuario):
```

```
360         try:
361             TelefoneService.save_telefones(usuario.telefone)
362             EnderecoService.save_endereco(usuario.endereco)
363             usuario.save()
364         except:
365             raise CustomValidation("Erro ao salvar usuário", 'detail', status_code=status.
    HTTP_409_CONFLICT)
366 
367     def delete_user(usuario):
368         try:
369             usuario.is_active = False
370             usuario.save()
371         except:
372             raise CustomValidation("Erro ao deletar usuário", 'detail', status_code=status.
    HTTP_409_CONFLICT)
```

Código A.11 – Service Empresa

```
373 class EmpresaService:
374 
375     def __init__(self):
376         pass
377 
378     def from_dto(objDto):
379         error = EmpresaService.validate_empresa(objDto)
380         if error:
381             raise CustomValidation(error, 'detail', status_code=status.HTTP_409_CONFLICT)
382 
383         try:
384             empresa = Empresa()
385             empresa.endereco = EnderecoService.from_dto(objDto)
386             empresa.telefone = TelefoneService.from_dto(objDto)
387             empresa.nome = objDto['nome']
388             empresa.cnpj = objDto['cnpj']
389             empresa.email = objDto['email']
390             empresa.web_site = objDto['web_site']
391             return empresa
392         except Exception as error:
393             raise CustomValidation(error, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
394 
395     def from_dto_update(objDto, empresa):
396         try:
397             empresa = Empresa.objects.filter(id=empresa.id)[0]
398             empresa.endereco = EnderecoService.from_dto_update(objDto, empresa.endereco)
399             empresa.telefone = TelefoneService.from_dto_update(objDto, empresa.telefone)
400             empresa.nome = objDto['nome']
401             empresa.cnpj = objDto['cnpj']
402             empresa.email = objDto['email']
403             empresa.web_site = objDto['web_site']
404             return empresa
405         except Exception as error:
406             raise CustomValidation(error, 'detail', status_code=status.HTTP_400_BAD_REQUEST)
407 
408     def save_empresa(empresa):
409         try:
410             TelefoneService.save_telefones(empresa.telefone)
411             EnderecoService.save_endereco(empresa.endereco)
412             empresa.save()
413         except Exception as error:
```

```
414             raise CustomValidation(error, 'detail', status_code=status.HTTP_409_CONFLICT)
415
416     def validate_empresa(objDto):
417         error = []
418         if Empresa.objects.filter(cnpj=objDto['cnpj']).exists():
419             error.append("CNPJ is not unique")
420         if Empresa.objects.filter(email=objDto['email']).exists():
421             error.append("email is not unique")
422         if Empresa.objects.filter(web_site=objDto['web_site']).exists():
423             error.append("web_site is not unique")
424         return error
425
426     def delete_empresa(empresa):
427         try:
428             empresa.status = 0
429             empresa.save()
430         except:
431             raise CustomValidation("Erro ao deletar empresa", 'detail', status_code=status.
    HTTP_409_CONFLICT)
```

Código A.12 – Service Telefone

```
432
433 class TelefoneService:
434
435     def __init__(self, residencial=None, celular=None, outro=None):
436         self.residencial = residencial
437         self.celular = celular
438         self.outro = outro
439
440     def from_dto(objdto):
441         try:
442             telefones = Telefone()
443             telefones.residencial = objdto['residencial']
444             telefones.celular = objdto['celular']
445             telefones.outro = objdto['outro']
446             return telefones
447         except:
448             raise CustomValidation("Erro ao parse telefones", 'detail', status_code=status.
    HTTP_400_BAD_REQUEST)
449
450     def from_dto_update(objdto, telefone):
451         try:
452             telefones = Telefone.objects.filter(id=telefone.id)[0]
453             telefones.residencial = objdto['residencial']
454             telefones.celular = objdto['celular']
455             telefones.outro = objdto['outro']
456             return telefones
457         except:
458             raise CustomValidation("Erro ao parse telefones", 'detail', status_code=status.
    HTTP_400_BAD_REQUEST)
459
460     def save_telefones(telefones):
461         try:
462             telefones.save()
463         except:
464             raise CustomValidation("Erro ao salvar telefone", 'detail', status_code=status.
    HTTP_409_CONFLICT)
```

Código A.13 – Service Endereço

```
465 class EnderecoService:
466
467     def __init__(self, logradouro=None, numero=None, complemento=None, bairro=None, cidade=None, cep=
        None, uf=None):
468         self.logradouro = logradouro
469         self.numero = numero
470         self.complemento = complemento
471         self.bairro = bairro
472         self.cidade = cidade
473         self.cep = cep
474         self.uf = uf
475
476     def from_dto(objDto):
477         try:
478             endereco = Endereco()
479             endereco.logradouro = objDto['logradouro']
480             endereco.numero = objDto['numero']
481             endereco.complemento = objDto['complemento']
482             endereco.bairro = objDto['bairro']
483             endereco.cidade = objDto['cidade']
484             endereco.cep = objDto['cep']
485             endereco.uf = objDto['uf']
486             return endereco
487         except:
488             raise CustomValidation("Erro ao parse endereço", 'detail', status_code=status.
        HTTP_400_BAD_REQUEST)
489
490     def from_dto_update(objDto, endereco):
491         try:
492             endereco = Endereco.objects.filter(id=endereco.id)[0]
493             endereco.logradouro = objDto['logradouro']
494             endereco.numero = objDto['numero']
495             endereco.complemento = objDto['complemento']
496             endereco.bairro = objDto['bairro']
497             endereco.cidade = objDto['cidade']
498             endereco.cep = objDto['cep']
499             endereco.uf = objDto['uf']
500             return endereco
501         except:
502             raise CustomValidation("Erro ao parse endereço", 'detail', status_code=status.
        HTTP_400_BAD_REQUEST)
503
504     def save_endereco(endereco):
505         try:
506             endereco.save()
507         except:
508             raise CustomValidation("Erro ao salvar endereço", 'detail', status_code=status.
        HTTP_409_CONFLICT)
```

APÊNDICE B – APÊNDICE B - CÓDIGOS SISTEMA IOT

Abaixo será listado os principais códigos para o desenvolvimento do sistema core, desenvolvidas na linguagem python os códigos representam as camadas listadas no sistema IOT. O acesso a todos os códigos está disponível em <https://github.com/projeto-agro-tcc/backend-iot>.

Código B.1 – Arquivo URL

```
509 router = routers.DefaultRouter()
510 router.register('empresas', EmpresasViewSet, basename='Empresas')
511 router.register('usuarios', UsuariosViewSet, basename='Usuarios')
512 router.register('estacoes', EstacoesViewSet, basename='Estacoes')
513
514 urlpatterns = [
515     path('', include(router.urls)),
516     path('admin/', admin.site.urls),
517     path('auth/', CustomAuthToken.as_view()),
518 ]
```

Código B.2 – ViewSet Data

```
519 class DataViewSet(ModelViewSet):
520
521
522     @action(detail=False, methods=['POST'])
523     def data(self, request, *args, **kwargs):
524         db_handle, mongo_client = get_db_handle()
525         collection = get_collection_handle(db_handle, RECEIVED_DATADB)
526         collection.insert_one(request.data)
527         response = {
528             "seq": request.data['seq'],
529             "status": 200,
530             "message": 'It is working'
531         }
532         return Response(response)
533
534
535     @action(detail=False, methods=['POST'])
536     def connection_status(self, request, *arg, **kwargs):
537         response = {
538             "seq": request.data['seq'],
539             "status": 200,
540             "message": 'It is working'
541         }
542         return Response(response)
543
544     @action(detail=False, methods=['POST'])
545     def signal_status(self, request, *arg, **kwargs):
546         response = {
547             "seq": request.data['seq'],
548             "status": 200,
549             "message": 'It is working'
```

```
550         }
551         return Response(response)
```

Código B.3 – ViewSet IOT

```
552 class IotViewSet(ModelViewSet):
553     serializer_class = EmwSerializer
554     iot_service = IotService()
555 
556     @action(detail=False, methods=['GET'])
557     def findbyparams(self, request, *args, **kwargs):
558         time_to_start = request.query_params.get('timetostart')
559         time_to_end = request.query_params.get('timetoend')
560         dev_id = request.query_params.get('dev_id')
561         colection = COLLECTIONS_EMW[request.query_params.get('var')]
562         if (time_to_start and dev_id and colection) is not None:
563             if time_to_end is None:
564                 time_to_end = datetime.now().timestamp()
565             response = self.iot_service.getDataByParams(time_to_start, time_to_end, dev_id, colection)
566             result_data = EmwSerializer(json.loads(response), many=True).data
567             return Response(result_data, status=status.HTTP_200_OK)
568         else:
569             return Response({"detail": "Verify the fields"}, status=status.HTTP_400_BAD_REQUEST)
```

Código B.4 – Service IOT

```
570 class IotService:
571 
572     def __init__(self):
573         pass
574 
575     def getDataByParams(self, time_to_start, time_to_end, dev_id, colection):
576         db_handle, mongo_client = get_db_handle()
577         collection = get_collection_handle(db_handle, colection)
578         serialized_obj = dumps(list(collection.find({'dev_id': dev_id,
579                                                      'time': {'$lt': int(time_to_end), '$gt': int(
    time_to_start)}})))
580         return serialized_obj
```

APÊNDICE C – APÊNDICE C - BUSNESS CASE

Abaixo será listado alguns indicativos de viabilidade do projeto. Todos os cálculos referente a esta etapa foram desenvolvidos em conjunto com o orientador Sr. Hamilton Marques Avinco.

Figura 39 – Premissas

Bases	Unidade	Valor
Ticket - Gateway (Setup + Ano01)	onetime	R$ 18.000,00
Ticket - Estação (Mensal - a partir Ano 02)	mensal	R$ 550,00

Penetração de 6% em áreas de culturas (raio 5km cobertura)

Capex	Unidade	Valor
Custo - Gateway	pc	R$ 6.020,88
Custo - Estação	pc	R$ 2.123,84

Opex Operação	Unidade	Valor A1	Valor A2	Valor A3	Valor A4	Valor A5
Custo - Frete	kit	R$ 150,00	R$ 165,00	R$ 181,50	R$ 199,65	R$ 219,62
Custo - Instalação	kit	R$ 400,00	R$ 440,00	R$ 484,00	R$ 532,40	R$ 585,64
Custo - Infra Estrtutura Cliente	kit	R$ 800,00	R$ 880,00	R$ 968,00	R$ 1.064,80	R$ 1.171,28
Custo - Infra Nuvem	mensal	R$ 500,00	R$ 550,00	R$ 605,00	R$ 665,50	R$ 732,05

Despesas Pessoal	Unidade	Valor A1	Valor A2	Valor A3	Valor A4	Valor A5
Profissional IA	mensal	R$ 3.000,00	R$ 3.300,00	R$ 3.630,00	R$ 3.993,00	R$ 4.392,30
Profissional Eng. HW / SW Estação	mensal	R$ 3.000,00	R$ 3.300,00	R$ 3.630,00	R$ 3.993,00	R$ 4.392,30
Despesas Vendas	mensal	R$ 3.000,00	R$ 3.300,00	R$ 3.630,00	R$ 3.993,00	R$ 4.392,30
Comissionamento Vendas Diretas - Setup	% receita	1,50%				
Comissionameto Vendas Parceiros - Setup	% receita	1,50%				
Comissionamento Vendas Diretas - Mensal	% receita	2,00%				
Comissionameto Vendas Parceiros - Mensal	% receita	2,00%				

Fonte: Hamilton Marques Avinco

Figura 40 – Premissas

	Ano 01	Ano 02	Ano 03	Ano 04	Ano 05	Acumulado
Receita Bruta	360.000,00	1.242.800,00	3.679.400,00	8.463.700,00	14.633.350,00	**28.379.250,00**
Receita Liquida	316.800,00	1.093.664,00	3.237.872,00	7.448.056,00	12.877.348,00	**24.973.740,00**
Opex Total	151.800,00	253.412,00	520.583,00	1.038.394,15	1.558.011,90	**3.522.201,05**
Ebit	- 81.242,06	180.665,34	907.396,25	2.722.003,67	6.567.977,36	**10.296.800,56**
% Ebit	-22,57%	14,54%	24,66%	32,16%	44,88%	**36,28%**
Capex Total	228.052,19	570.130,47	1.531.207,55	2.972.823,17	3.339.335,61	**8.641.548,98**
Depreciação	18.189,88	89.456,19	278.685,20	714.835,01	1.412.023,13	**2.513.189,41**
Ebitda	- 63.052,19	270.121,53	1.186.081,45	3.436.838,69	7.980.000,49	**12.809.989,97**
% Ebitda	-17,51%	21,73%	32,24%	40,61%	54,53%	**45,14%**

VPL - 5 anos	**R$ 4.495.652,23**
Ebitda Positivo (m)	**24**

Fonte: Hamilton Marques Avinco