Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Sistemas de Informação

Ferramenta de Planejamento de Redes LoRaWAN

Rodrigo Torres

JUIZ DE FORA
JULHO, 2023

Ferramenta de Planejamento de Redes LoRaWAN

Rodrigo Torres

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Sistemas de Informação

Orientador: Edelberto Franco Silva

JUIZ DE FORA
JULHO, 2023

Ferramenta de Planejamento de Redes LoRaWAN

Rodrigo Torres

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM SISTEMAS DE INFORMAÇÃO.

Aprovada por:

Edelberto Franco Silva
Doutor em Ciência da Computação

Igor de Oliveira Knop
Doutor em Modelagem Computacional

Luciano Jerez Chaves
Doutor em Ciência da Computação

JUIZ DE FORA
07 DE JULHO, 2023

Resumo

Com a crescente necessidade do monitoramento por meio de sensores, a orquestração de diversos dispositivos e a tomada de decisões por meio da coleta e grandes volumes de dados, os dispositivos interligados pela chamada Internet das Coisas vêm crescendo cada vez mais com o passar dos anos. Redes para interconectar esses tipos de dispositivos estão se popularizando, e a LoRaWAN se destaca entre uma das alternativas mais promissoras. Sua grande área de de cobertura e baixo consumo de energia motivam os engenheiros de redes. Porém, atualmente esses gestores não possuem uma ferramenta apropriada, capaz de auxiliar no planejamento e na implementação desse novo tipo de rede, atendendo às suas particularidades. A dificuldade em construir um serviço capaz de garantir uma conexão estável, sem gastar uma quantidade substancial de recursos é um problema de otimização complexo. Portanto, neste trabalho propomos um sistema capaz de entregar uma solução referente à quantidade e localização de *gateways* apropriada para o cenário inserido pelo administrador de rede.

Palavras-chave: LoRa, LoRaWAN, LPWAN, Otimizador, *Datasets*, Gerador de Instâncias

Abstract

With the growing need for monitoring through sensors, the orchestration of various devices, and decision-making through the collection and large volumes of data, the devices interconnected by the Internet of Things have been growing increasingly. Networks to interconnect these devices are becoming popular, and LoRaWAN stands out as one of the more promising alternatives. Its large footprint and low power consumption motivate network engineers. Although, currently these managers do not have an appropriate tool capable of assisting in the planning and implementation of this new type of network, meeting to its particularities. The difficulty in building a service capable of guaranteeing a stable connection without spending a substantial amount of resources is a complex optimization problem. Therefore, in this work we propose a system capable of delivering a solution regarding the number and location of appropriate gateways for the scenario inserted by the network administrator.

Keywords: LoRa, LoRaWAN, LPWAN, Optimizer, Datasets, Instance Generator

Agradecimentos

A minha mãe pelo apoio em todos os anos de faculdade.

Aos meus amigos que me auxiliaram em cada matéria no decorrer do curso.

Ao professor Edelberto Franco Silva pela orientação, amizade e confiança desde o primeiro período da faculdade.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AP	*Access Point*
API	*Application Programming Interface*
BW	*Bandwidth*
CD	*Coordinating Device*
CH	*Cluster Head*
CR	*Code Rate*
CSS	*Chirp Spread Spectrum*
CAPEX	Despesa de Capital
DABC	*Discrete Artificial Bee Colony*
DFWA	*Discrete Fireworks Algorithm*
DOM	*Document Object Model*
FEC	*Forward Error Correction*
GCLS	*Grid-based Candidate Location Selection*
GW	*Gateway*
IoT	*Internet of Things*
IGW	*IoT Gateway*
JSON	*JavaScript Object Notation*
LC-BBO	*Low-complexity bio-geography-based Optimization*
LoRa	*Long Range*
LoRaWAN	*Long Range Wide Area Network*
LPWAN	*Low Power Wide Area Network*
LSM	*Local Search Methods*
NSGA-II	*Fast Non-dominated Sorting Genetic Algorithm II*
OPEX	Despesas Operacionais
ILP	*Integer Linear Program*
QoS	Qualidade de Serviço
RB	*Bit Rate*

REST	*Representational State Transfer*
RFID	*Radio Frequency Identification*
SF	*Spreading Factor*
SSGW	*Solution Specific Gateways*
SNR	*Signal-to-noise ratio*
URL	*Uniform Resource Locator*
WLAN	*Wireless Local Area Network*

1 Introdução

1.1 Apresentação do Tema

Com a crescente necessidade do monitoramento por meio de sensores, a orquestração de diversos dispositivos e a tomada de decisões por meio da coleta e grandes volumes de dados, os dispositivos interligados por meio da chamada Internet das Coisas (*Internet of Things - IoT*) vêm crescendo cada vez mais com o passar dos anos (HASAN, 2022). Com isso, sistemas baseados nos conceitos de *Smart Health* (SOLANAS et al., 2014), *Smart Farms* (YOON; LIM; PARK, 2020), *Smart Cities* (TALARI et al., 2017) e Industrias 4.0 (PEREIRA; ROMERO, 2017) estão sendo criados. Para que essas tecnologias tenham êxito, conectando diversos dispositivos ao mesmo tempo que coletam e transmitem informações, é necessário o incremento da conectividade de forma que mantenha razoável o consumo de energia. Neste cenário, se destacam as redes sem fio de longa distância, as chamadas LPWANs (*Low Power Wide Area Network*).

Um exemplo de LPWAN que aplicada aos cenários descritos, é a LoRaWAN (*Long Range Wide Area Network*). LoRaWAN é uma rede que implementa os protocolos e a arquitetura necessários para o funcionamento da tecnologia LoRa (*Long Range*). Por sua vez, a LoRa, responsável pelas camadas física e de enlace desse ambiente sem fio. Foi desenvolvida pela empresa Semtech (CORPORATION, 2022) e possui uma técnica de modulação derivada da *Chirp Spread Spectrum* (CSS). Tal modulação tem como objetivo atender a uma grande área de cobertura e interligar dispositivos IoT com baixa transferência de dados e baixo consumo de energia (SEMTECH, 2022).

1.2 Contextualização

Ao planejar uma rede de dispositivos IoT, como a rede LoRaWAN, é necessário garantir que todos os nós clientes serão atendidos e cobertos por estações base, nesse caso representados pelos GW (*Gateways*).

Porém, diferentemente de uma rede sem fio local baseada no padrão IEEE 802.11 (as, popularmente chamadas de, redes Wi-Fi) que medem a qualidade de sinal recebido por um nó cliente por meio do RSSI (*Received Signal Strength Indication*), no cenário das LPWANs, por possuírem uma grande área de cobertura, essas redes podem sofrer maior quantidade de interferência ao transferirem dados. Portanto, a métrica que melhor representa a qualidade de um sinal recebido por um nó é o *Signal-to-noise ratio* (SNR) (FIGUEIREDO; SILVA, 2020). O SNR[1], por sua vez, possui derivação do RSSI, mas levando em consideração a relação tanto da potência do sinal quanto do ruído no meio.

Além disso, é possível afirmar que vários parâmetros estão disponíveis para configuração das redes LoRaWAN, tais como: *Spreading Factor* (SF), *Code Rate* (CR) e *Bandwidth* (BW). Esses são parâmetros cruciais para definir a resistência de interferências, a taxa de transferência de dados e a distância da comunicação entre dois nós.

Finalmente, o planejamento também deve levar em conta o custo da compra de GWs. Portanto, ao finalizar o planejamento da implantação de uma rede, além de todos os nós clientes estarem cobertos por pelo menos um GW, a quantidade de *gateways* utilizados deve ser o menor possível.

1.3 Descrição do Problema

Planejar uma rede LoRaWAN é uma tarefa árdua, pois diversos fatores devem ser considerados para garantir uma rede ótima. A falta de ferramentas capazes de ajudar nesse planejamento de rede torna o objetivo ainda mais desafiador. Devido a grande área de cobertura que as redes LoRaWAN podem alcançar, problemas de interferência são capazes de ocorrer no caminho do dado ao destino. Além disso, garantir que todos dispositivos cliente sejam alcançados ao mesmo tempo que a quantidade de GWs seja mínima, requer horas de trabalho manual do administrador, aplicando técnicas de tentativa e erro. Desta forma, o administrador da rede pode acabar adquirindo uma quantidade maior de *gateways* do que realmente necessário, causando assim prejuízos, tanto financeiro quanto no incremento da complexidade do ambiente. Finalmente, não basta que os clientes estejam sendo atendidos, a rede deve garantir a qualidade de comunicação, e essa qualidade está

[1] O valor de SNR varia entre [0-120], e quanto mais próximo de 120db (Decibéis), melhor.

diretamente ligada à configuração adequada dos diversos parâmetros da LoRaWAN.

Por fim, a falta de conjuntos de dados de localização e configurações de dispositivos LoRaWAN disponíveis por meio de *datasets* que auxiliem o estudo da otimização desse tipo de rede, dificulta a criação de novos trabalhos na área.

1.4 Justificativa e Motivação

Uma rede que abrange uma extensa área e é capaz de atender uma grande quantidade de dispositivos possui a complexidade alta da criação de instâncias reais é alta. Além isso, devido a possibilidade de modificação de diversos parâmetros de configuração, como o SF, BW e o CR, para atingir o resultado esperado da rede, criar *datasets* reais seria necessário a configuração manual dos dispositivos repetidas vezes. Portanto, o desenvolvimento de uma aplicação capaz de gerar instâncias fictícias irá suprir a necessidade de *datasets* reais para futuras pesquisas de otimização de redes LoRaWAN.

Para os administradores de rede, o sistema poderia ajudar a ter uma noção melhor de como os parâmetros escolhidos podem influenciar no estado final da rede LoRaWAN, modificando-os como bem entender, em um ambiente controlado, sem a necessidade de alterar as configurações físicas dos dispositivos, até que estejam satisfeitos com os resultados.

E por último, por possuir um otimizador capaz de trazer um resultado ótimo para a arquitetura da rede, o sistema proposto pode trazer uma economia para as instituições e empresas interessadas em implementar a LoRaWAN.

1.5 Objetivos

1.5.1 Geral

O trabalho tem o intuito de gerar um sistema de planejamento de redes LoRaWAN capaz de gerar *datasets* com soluções otimizadas para que o administradores de rede possam modificar os parâmetros do LoRaWAN de forma a atender seus interesses, e possibilitar novas pesquisas a partir desses *datasets*. A aplicação será capaz de receber como entrada:

1. os parâmetros de configuração da rede LoRaWAN que o planejador da rede deseja implementar;

2. as regiões que serão atendidas pela rede delimitadas por longitude e latitude;

3. a densidade de possíveis usuários da rede de cada região.

E como saída, gerar um arquivo com a localização de possíveis nós clientes na região informada, e a localização de cada *gateway* necessário para cobrir toda a área.

1.5.2 Específicos

Para que o objetivo geral seja concluído é necessário desenvolver os seguintes tópicos:

- no estado da arte, procurar softwares geradores de *datasets*;
- desenvolver um sistema web com *frontend* com *framework* Vue para que o planejador da rede tenha uma interface amigável de utilizar;
- desenvolver o *backend* do sistema web com Django que tenha um otimizador integrado capaz de, ao receber os parâmetros de entrada, retornar uma instância com uma saída previamente aperfeiçoada;
- elaborar uma documentação de utilização do sistema desenvolvido.

1.6 Organização do trabalho

O restante do trabalho está organizado em mais quatro capítulos. O próximo capítulo possui uma explicação dos conceitos essências que serão necessários para o entendimento deste trabalho. O terceiro capítulo apresenta uma revisão da literatura, com uma análise de seis artigos relacionados a este TCC. O quarto capítulo está descrito o passo a passo de como foi projetado e desenvolvido a ferramenta proposta. Por último, o quinto capítulo apresenta as considerações finais do trabalho, bem como possibilidades de melhorias futuras.

2 Fundamentação Teórica

Este capítulo tem como objetivo apresentar a fundamentação teórica essencial para entendimento dos principais tópicos presentes neste trabalho.

Os conceitos de LoRa e LoRaWAN (Seção 2.2) são fundamentais, pois a solução apresentada ao final terá como objetivo o planejamento de uma LPWAN (Seção 2.1) definida com a arquitetura LoRaWAN que interliga dispositivos que implementam o protocolo LoRa.

É necessário também explicar sobre o que são *frameworks* e apresentar os fundamentos do *framework* de *backend* Django (Seção 2.3) e do *framework* de *frontend* Vue.js (Seção 2.4) que irão auxiliar no desenvolvimento da ferramenta proposta por esse trabalho.

Finalmente será explicado o algoritmo K-means (Seção 2.5), já que ao gerar um novo *dataset* a ferramenta proposta deve usar o algoritmo para sugerir o posicionamento de *gateways* que irá permitir a comunicação de todos os clientes.

2.1 Redes LPWAN

As *Low Power Wide Area Networks*, ou LPWANs, são redes com baixo consumo de energia, grande alcance e portanto um baixo custo de implantação, pois com apenas um GW é possível cobrir uma grande área e alcançar diversos nós cliente. Apesar desses benefícios, esse tipo de rede suporta uma baixa transferência de dados se comparada as redes *Wireless Local Area Network* (WLAN) que possuem uma alta taxa de transferência mas que possuem uma alto consumo de energia e alto custo de instalação, como é possível verificar na Figura 2.1

Devido a suas principais características, as redes LPWANs são usadas para atender o crescente uso dos dispositivos IoT que, de acordo com Hasan (2022), esse mercado crescerá em 18% em 2022, com 14,4 bilhões de dispositivos e a expectativa para 2025 é de aproximadamente 27 bilhões como é possível visualizar na Figura 2.2.

Em meio a tantos tipos de dispositivos IoT, os sensores são os que mais se be-

neficiam das LPWANs, por terem que consumir uma baixa quantidade de energia e por enviar pequenas quantidades de dados pelas redes sem fio. De acordo com a pequisa de James Brehm & Associates em 2015, mais de 80% de todos os sensores transmitem 2MB ou menos por mês (SKYSENS, 2022).

De acordo com Pasqua (2021), em 2021 as LPWANs presentes na Ásia correspondem a 66% do mercado e, como é possível ver na Figura 2.3, as principais redes LPWANs atuantes são NB-IoT, LTE-M, Sigfox e LoRa o qual será abordado nesse trabalho.

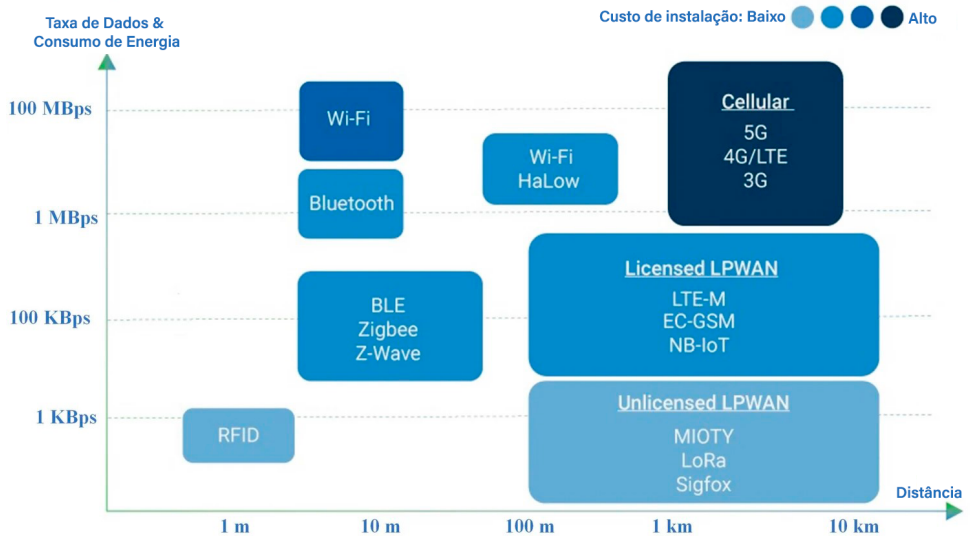

Figura 2.1: Comparativo entre redes. Fonte: (QIN et al., 2021)

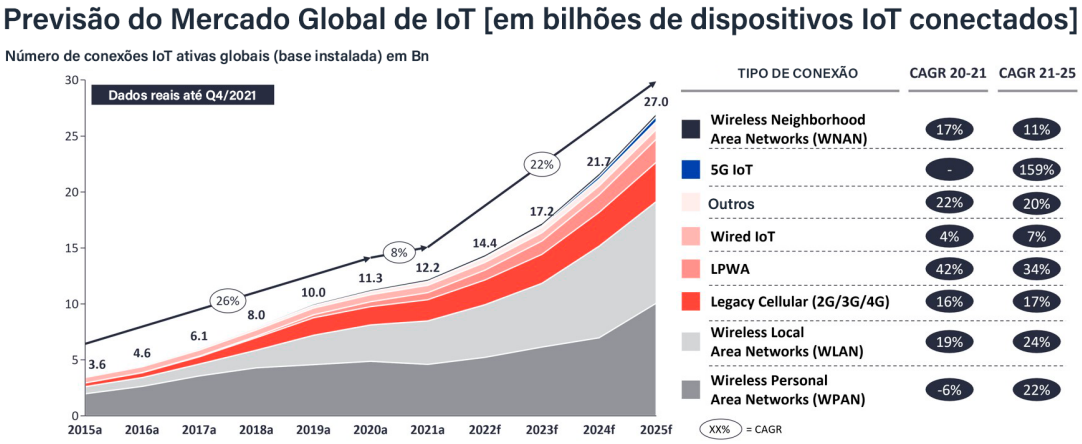

Figura 2.2: Crescimento do mercado de IoT no mundo. Fonte: (HASAN, 2022)

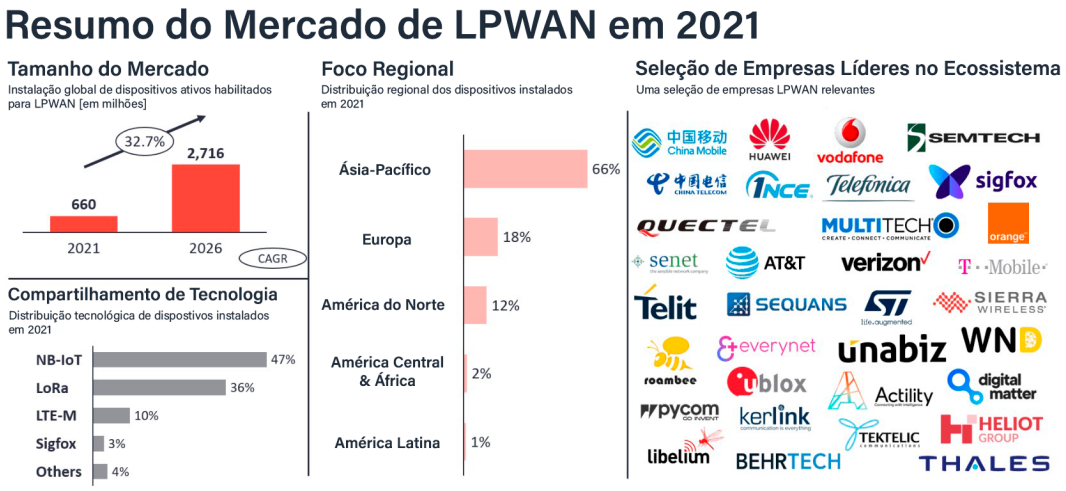

Figura 2.3: O mercado das LPWANs em 2021. Fonte: (PASQUA, 2021)

2.2 LoRa e LoRaWAN

O LoRa, ou *Long Range*, é uma tecnologia desenvolvida pela empresa francesa Semtech a qual também produz os semicondutores que a implementam, mas apenas essa tecnologia não é suficiente para criar uma rede LPWAN, então surge a LoRaWAN (*Low Power Wide Area Networking*), projetada pela LoRa Alliance, ela permite que dispositivos se comuniquem por meio do LoRa.

2.2.1 Protocolo LoRa

O protocolo LoRa implementa um esquema de modulação digital baseado na técnica de espalhamento espectral chamada CSS (*Chirp Spread Spectrum*) (FIGUEIREDO; SILVA, 2020) que garante uma plataforma sem fio de baixo consumo de energia, longo alcance, envio de pequenas quantidades de dados e baixo custo de implantação. Dependendo do país ela pode operar em diferentes faixas de banda como, por exemplo, 915 MHz na América do Norte 868 MHz na Europa.

A taxa de modulação do LoRa é definida pelo atributo chamado fator de espalhamento (SF - *Spread Factor*). O SF pode variar de 7 a 12 e está ligado diretamente a robustez da rede, pois quanto menor o SF, maior a quantidade de dados podem ser transferidas, mas a imunidade contra interferências será reduzida. Outros dois atributos também são importantes para definir a taxa de transferência da rede e intensidade do

sinal: a largura da banda (BW - *Bandwith*) e o *code rate* (CR) responsável pela correção de erro continua (FEC - *Forward Error Correction*).

Todos esses atributos descritos podem ser relacionados na Fórmula 2.1 para calcular o *bit rate* (Rb) e, na Figura 2.4, é possível visualizar os cálculos de para cada SF considerando todos os valores de CR suportados e as três maiores larguras de banda.

$$Rb = SF \times \frac{BW}{2^{SF}} \times CR \tag{2.1}$$

		Bandwidth (kHz)											
		125				250				500			
		CR											
		4/5	4/6	4/7	4/8	4/5	4/6	4/7	4/8	4/5	4/6	4/7	4/8
SF	7	5.5	4.6	3.9	3.4	10.9	9.1	7.8	6.8	21.9	18.2	15.6	13.7
	8	3.1	2.6	2.2	2.0	6.3	5.2	4.5	3.9	12.5	10.4	8.9	7.8
	9	1.8	1.5	1.3	1.1	3.5	2.9	2.5	2.2	7.0	5.9	5.0	4.4
	10	1.0	0.8	0.7	0.6	2.0	1.6	1.4	1.2	3.9	3.3	2.8	2.4
	11	0.5	0.4	0.4	0.3	1.1	0.9	0.8	0.7	2.1	1.8	1.5	1.3
	12	0.3	0.2	0.2	0.2	0.6	0.5	0.4	0.4	1.2	1.0	0.8	0.7

kbps

Figura 2.4: Relação entre SF, BW, CR e Rb. Fonte: (FIGUEIREDO; SILVA, 2020)

2.2.2 LoRaWAN e sua arquitetura

Como o LoRa é apenas um método de camada física de envio de dados, a LoRa Alliance, uma associação sem fins lucrativos criada no intuito de de impulsionar o crescimento da rede LPWAN, cria a rede LoRaWAN que implementa os protocolos para gerenciar o tráfego de dados e o gerenciamento de dispositivos que possuem o LoRa.

A arquitetura da rede LoRaWAN é composta por uma topologia estrela, onde os *gateways* são o centro das transmissões de dados responsáveis por enviar mensagens entre os nós cliente e o servidor de rede central.

Os dispositivos finais abrangem uma infinidade de funções que podem abranger desde sensores de movimento, fumaça e rastreamento de animais de estimação até dispositivos mais complexos como máquinas de venda. Esses dispositivos têm em comum a alimentação por baterias, pequena transmissão de dados e podem estar em locais geograficamente isolados.

Como é possível observar na Figura 2.5, os *gateways* estão no meio da comu-

nicação da rede e funcionam como um tradutor de pacotes, convertendo os dados vindos dos dispositivos LoRa para os servidores e vice-versa. Eles estão conectados ao servidor por meio de conexões IP padrão como redes de celulares 3G, 4G e 5G, Wi-Fi, *ethernet* ou Fibra Óptica, mas a comunicação com os nós finais é feita pelo protocolo de camada física LoRa.

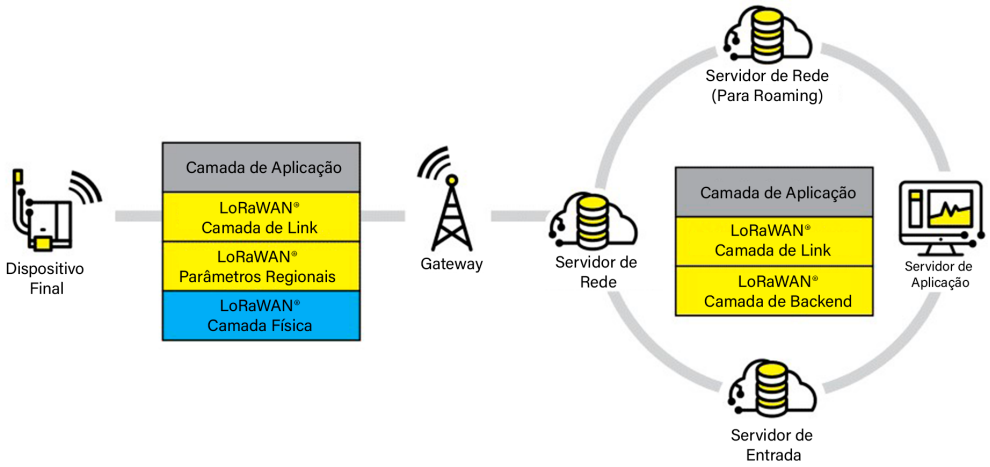

Figura 2.5: Arquitetura da rede LoRaWAN. Fonte: (ALLIANCE, 2022)

Os servidores de rede e de aplicação são responsáveis por criar e enviar mensagens de *downlink*, verificar a autenticidade e integridade dos dados recebidos, orquestrar os pacotes de *uplink*, processar e armazenar os dados recebidos pelos nós finais.

Os dispositivos finais da rede LoRaWAN podem possuir três tipos de configurações diferentes para que seja possível atender toda a gama de equipamentos IoT:

- Classe A - Dispositivos finais bidirecionais de menor potência: esse é o padrão mais básico e obrigatório para qualquer dispositivo final LoRaWAN. define a comunicação bidirecional ou de instruções de controle de rede do dispositivo com o servidor. O dispositivo final sempre inicia a comunicação e cada transmissão de *uplink* pode ser enviada a qualquer momento. Em seguida, duas janelas curtas de *downlink* são criadas para o recebimento de mensagens. Esse tipo de comunicação e oriundo do protocolo ALOHA (BACCELLI; BLASZCZYSZYN; MUHLETHALER, 2006);

  Já que a comunicação parte sempre do dispositivo final, o equipamento pode entrar em suspensão por tempo indeterminado até que seja necessário um novo envio de

pacotes. Isso classifica a Classe A com o menor consumo de energia dentre as outras classes. Caso o servidor queira se comunicar com o nó cliente, é necessário armazenar a mensagem até que uma transmissão de *uplink* seja feita. Nesse momento, é possível utilizar as duas janelas de *downlink* criadas para enviar o dado armazenado;

- Classe B - Dispositivos finais bidirecionais com latência de *downlink* determinística: os dispositivos de Classe B são caraterizados por, além de abrirem as mesmas janelas de *uplink* e *downlink* da Classe A, eles também criam novos *downlink* (*slots* de *ping*) em horários programados, os chamados *Beacons*. Essa nova funcionalidade permite esses dispositivos a comunicação com uma latência determinística. Essa latência pode ser programável em até 128 segundos. Apesar de ter um custo de energia adicional, esse consumo ainda é baixo o suficiente a ponto de ser válido a implementação dessa classe em dispositivos alimentados por bateria;

- Classe C - Dispositivos finais bidirecionais de menor latência: além de implementar as janelas da Classe A, a Classe C é caraterizada por deixar aberto o *downlink* sempre que o dispositivo não estiver transmitindo. Essa forma de operar faz com que o nó final gaste mais energia, já que a comunicação com o servidor quase sempre está ligada. Portanto, a Classe C é recomendada para equipamentos que estão ligados de forma contínua na energia.

2.3 Django

Os *web frameworks* são uma coleção de bibliotecas que auxiliam a criação de web sites. Por sua vez, essas bibliotecas possuem um conjunto de funcionalidades que todo *backend* necessita de implementar como: criação de rotas URL, conexão, gerenciamento e consultas no banco de dado, envio de dados para o *frontend* e garantia de segurança a ataques ao servidor o qual a aplicação web estiver hospedada, como ataques com *SQL Injection*. Diversos *frameworks* para *backend* estão disponíveis no mercado como, por exemplo, Express desenvolvido em JavaScript, Laravel e Symfony em PHP, Flask e Django em Python. Esse último foi escolhido para compor a ferramenta final proposta pelo trabalho e será melhor detalhada.

Django é um *web framework* de código aberto desenvolvido em Python. Sua primeira versão foi lançada em 2005 e vem sendo aprimorado continuamente pela Django Software Foundation e pela própria comunidade. Ele é utilizado por diversos sites pelo mundo, como o Instagram, Pinterest, Bitbucket e Disques. Ao mesmo tempo, por ser versátil, ele também é uma ótima escolha para pequenos projetos (VINCENT, 2021).

De acordo com Foundation (2022), os principais benefícios do Django incluem:

- rápido e simples de implementar, pois com as ferramentas que o *framework* já apresenta, o desenvolvedor só tem a necessidade de focar nas regras de negócio que sua aplicação apresenta;
- inclusão de ferramentas que lidam com problemas comuns do desenvolvimento web, como administração de conteúdo, autenticação de usuário, mapas do site, entre outros;
- garantia de segurança a problemas como *SQL injection*, *cross-site scripting*, *cross-site request forgery* e *clickjacking*;
- é um *framework* escalável para atender altas demandas de requisições e tráfego de dados;
- Totalmente versátil por ser capaz de implementar desde gerenciamento de conteúdo de redes sociais até aplicações de computação científica.

2.4 Vue.js

Assim como os *frameworks* para *backend*, os de *frontend* também possuem seu conjunto de ferramentas já implementadas para acelerar o desenvolvimento de uma aplicação web, mas tem como a principal função criar interfaces intuitivas e rápidas para o usuário final.

De acordo com Wohlgethan (2018), as tecnologias de maior tração do mercado, focadas na camada visual da aplicação web são o React[2] criado pelo Facebook, Angular[3] criado pelo Google e Vue.js criado pelo antigo empregado da Google, Evan You. Criadas

[2]https://react.dev/
[3]https://angular.io/

a partir do Javascript, as três tem premissas bastante parecidas na sua formação, mas o Vue.js[4] se destaca pela facilidade de implementação. Portanto ele foi escolhido para compor o projeto final desse trabalho.

Wohlgethan (2018) define o Vue.js como um *framework* progressivo para criação de interfaces do usuário. Lançado em 2014, o Vue foi projetado para ser facilmente integrado com outras bibliotecas e projetos existentes. Portanto é possível aplicá-lo de forma incremental. Ainda assim, o Vue também é capaz de implementar aplicações sofisticadas como as *Single-Page Applications* se utilizada em conjunto com outras bibliotecas e ferramentas.

Dentre as principais características do Vue.js estão: o tratamento da interação do usuário, a renderização declarativa que permite que os dados do DOM (*Document Object Model*) se tornem reativos, uso de diretivas para controlar condicionais e repetições. Por fim, o *framework* utiliza o sistema de componentes que proporciona a implementação de softwares formados por pequenos blocos de código altamente reutilizáveis.

2.5 K-means

As técnicas de agrupamento de dados são uma poderosa abordagem de análise exploratória utilizada para descobrir a estrutura presente em conjuntos de dados multivariados. O agrupamento de dados é um método de classificação não supervisionada, onde os *clusters* são formados com base nas similaridades e dissimilaridades entre as características intrínsecas dos dados. O agrupamento k-means é uma técnica amplamente utilizada nesse contexto.

De acordo com Morissette e Chartier (2013) o algoritmo K-means é uma técnica de particionamento que visa dividir as variáveis de um conjunto de dados em grupos não sobrepostos, chamados de *clusters*. O objetivo é alcançar alta similaridade entre os elementos de cada grupo e baixa similaridade entre grupos diferentes. O algoritmo é iterativo e funciona realocando os pontos de dados entre os *clusters* até que a convergência seja alcançada. Cada *cluster* é representado por um vetor médio chamado centróide.

Na Figura 2.6 é possível ver um exemplo da execução do algoritmo. Na esquerda,

[4]https://vuejs.org/

um conjunto de dados que não está classificado. Ao executar o K-means os dados são agrupados em 3 *clusters* diferentes, um verde, azul e um vermelho. Cada um deles possui seu centróide representado por um X. Esse centróide será de grande importância, pois será ele que definirá o local dos *gateways*.

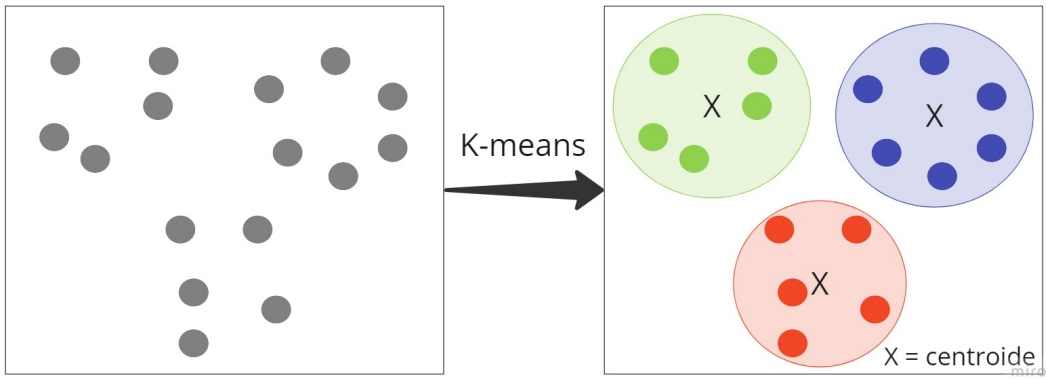

Figura 2.6: Exemplo da execução do K-means

2.6 Considerações parciais

Nesse capítulo, primeiramente, foram apresentados os conceitos de redes LPWANs e como elas estão estritamente ligadas ao mercado crescente de IoT. Posteriormente, o protocolo LoRa foi exposto. Cada um dos seus atributos foram explicados de forma a entender qual é o papel dos mesmos na robustez da rede. Pelo LoRa ser apenas um método de envio de dados, a LoRaWAN e sua arquitetura foi apresentada como forma de interligar e gerenciar todos os dispositivos IoT que implementam o protocolo LoRa.

Finalmente, devido a dificuldade de planejar uma rede LoRaWAN, o trabalho propõe a criação de uma ferramenta capaz de facilitar o planejamento da rede LPWAN e que terá como base os *web frameworks* Vue.js e Django.

3 Trabalhos Relacionados

Nesse capítulo é apresentado um conjunto de trabalhos que foram considerados para determinar os métodos de otimização que serão implementados e comparados nesse trabalho. Os métodos de otimização são de suma importância, pois são eles que definem se a aplicação proposta irá atender o planejamento adequado da rede LoRaWAN.

Os trabalhos relacionados foram pesquisados utilizando palavras-chave "*optimization*", "*gateway*", "IoT", "*planning*" e "*network*". Essas palavra foram selecionados para procurar temas relacionados a cobertura de área de rede, disposição de *gateways*, propagação de sinal em ambiente inteligentes, como Smart Cities, Smart Homes e Smart Campus e no planejamento de redes com enfoque em dispositivos IoT.

Além disso, o busca tentou se concentrar em artigos publicados a partir de 2017, mas foram encontrados artigos igualmente relevantes em datas anteriores. A pesquisa foi feita apenas pelo Google Scholar e esse, por sua vez, direcionava para o banco de dados IEEE.

Cada análise identificou as principais características dos cenários, os objetivos, os algoritmos apresentados responsáveis pela otimização das redes apresentadas e os resultados alcançados de cada trabalho relacionado. Assim é possível relacionar e identificar semelhanças ao projeto proposto e extrair algoritmos candidatos a serem implementados no sistema final.

3.1 Planejamento de redes WLAN utilizando algoritmo genéticos

No trabalho Lima, Carrano e Takahashi (2012), o principal problema abordado se refere ao balanceamento de carga em redes IEEE 802.11 onde, quando um número de clientes muito grande está concentrado em uma área pequena, a qualidade do serviço e a taxa de transferência tende a cair. A alta quantidade de usuários na rede leva o aumento de

custos de instalação e à degradação da rede, devido principalmente à interferência entre os canais limitados os quais os clientes são atribuídos.

O principal objetivo de Lima, Carrano e Takahashi (2012) é fazer a distribuição adequada dos usuários entre os APs para maximizar a força do sinal garantindo assim uma distribuição de carga otimizada. Para isso, são consideradas dois critérios a minimização do número de APs e a minimização do desequilíbrio de carga na rede. É importante mencionar que o desequilíbrio é medido a partir da intensidade do sinal, ou seja, quando maior a intensidade do sinal maior o equilíbrio.

A proposta de algoritmo utilizada em Lima, Carrano e Takahashi (2012) é o algoritmo genético baseado no *Fast Non-dominated Sorting Genetic Algorithm II* (NSGA-II). As soluções obtidas por esse algoritmo obedecem obrigatoriamente a três restrições: cobertura mínima, capacidade máxima de largura de banda dos APs e disponibilidade de canal.

Os resultados obtidos por Lima, Carrano e Takahashi (2012) são comparados a um algoritmo de agrupamento K-means em dois diferentes cenários. As soluções obtidas pelo algoritmo genético superaram o K-means em todas as instâncias testadas proporcionando a eficiência do uso de recursos da rede, provendo melhor equilíbrio de carga e minimizando a interferência para o intervalo de 13 a 22 APs.

3.2 Planejamento de redes IoT para Cidades Inteligentes

No trabalho Karthikeya, Vijeth e Murthy (2016) descreve a dificuldade de planejar uma rede IoT em uma *Smart City*. Nesse tipo de cenário, diversas tecnologias trabalham em conjunto para integrar diferentes aplicações, e cada uma dessas aplicações são coordenadas por um *Coordinating Device* (CD). Esse dispositivo recebe nomes diferentes dependendo de qual tecnologia emprega, por exemplo: *Cluster Head* (CH) em redes de sensores, Identificador de Radiofrequência (RFID) ou *Access Point* (AP) em redes Wi-Fi. Cada CD deve estar conectado a um *Solution Specific Gateways* (SSGW) e estes, por sua vez, conectados a um *IoT Gateway* (IGW). Os SSGW são responsáveis pela interconversão

entre as tecnologias e os IGW tem o papel de transmitir os dados para a internet.

O foco de Karthikeya, Vijeth e Murthy (2016) está na minimização do total de *gateways* que seriam utilizados em uma *Smart City* com o intuito de diminuir o custo de instalação da rede, pois se torna inviável que cada CD tenha um IGW individualmente conectado a ele. Esse tipo de abordagem aumentaria o custo da rede desnecessariamente além de subutilizar um IGW.

O algoritmo proposto em Karthikeya, Vijeth e Murthy (2016) é nomeado de *Network Intersection based CandidaTe Gateway Location Selection* ou simplesmente algoritmo NewIoTGateway-Select e tem como principais pontos o cálculo dos locais candidatos para instalação de SSGWs e IGWs, seleção dos locais ideias para instalação dos mesmos e o tratamento de falhas para os *gateways* e no link de internet incorporando algoritmos de k-cobertura e k-conectividade.

Em suma, no trabalho de Karthikeya, Vijeth e Murthy (2016) os resultados do algoritmo proposto são comparados com o *Grid based Candidate Location Selection* (GCLS) em diferentes tipos de densidade de rede. Ao final, o artigo conclui que em cenários com um grande número de CDs o NewIoTGatewaySelect se sai melhor do que o GCLS com uma melhoria de quase 20%.

3.3 Posicionamento de *gateways* em uma LPWAN utilizando algoritmo Fuzzy C-Means

Em Matni et al. (2019) é descrito o cenário de uma arquitetura LoRa. Redes LoRa possuem a topologia de uma LPWAN e por isso conseguem abranger uma área com dezenas de quilômetros, portanto com apenas um *gateway* é possível atender vários dispositivos de uma só vez. Porém, aumentar o número de *gateways*, afim de compartilhar o número de dispositivos, pode melhorar a Qualidade de Serviço (QoS) aumentando a qualidade de comunicação entre os dispositivos IoT e os *gateways*. No entanto, o custo de implantação e manutenção da rede irá aumentar, afetando diretamente a Despesa de Capital (CAPEX) e as Despesas Operacionais (OPEX).

O objetivo de Matni et al. (2019) inclui a proposta do posicionamento ideal

de *gateways* LoRa por toda a cobertura da rede de forma a melhorar a Qualidade de Serviço e, ao mesmo tempo, diminuir o CAPEX e o OPEX garantindo o equilíbrio dessas características.

Para Matni et al. (2019) a utilização do algoritmo Fuzzy C-Means foi necessário para aplicar o conceito de PLACE, que divide o cenário de LPWANs em *clusters*. Cada *cluster* pode ser traduzido como o alcance de um *gateway* da rede proposta. O agrupamento leva em consideração a similaridade entre dispositivos IoT. Diferente de outros agrupamentos hierárquicos que, quando o cluster é formado não é possível modificar o seu posicionamento, esse método modifica o posicionamento de *gateways* e de dispositivos a cada iteração até que uma organização ótima seja atingida.

Os resultados de Matni et al. (2019) foram comparados a um algoritmo de colocação de *gateway* aleatório. Nessa comparação, foi possível alcançar um economia de até 36% no custo de instalação mantendo uma taxa de transmissão equiparada entre as soluções. Além disso, o algoritmo proposto conseguiu diminuir em nove a quantidade de *gateways*, reduzindo de 25 para 16 no cenário de teste

3.4 Planejamento de uma infraestrutura em nuvem para sistemas de assistência médica

No trabalho Ali et al. (2022) é introduzido uma infraestrutura de IoT para um ambiente de Assistência Médica 4.0 onde é necessário o processamento de registros médicos eletrônicos de forma segura e uma transmissão de dados para a nuvem. Porém, os serviços de nuvem tradicionais podem violar regulamentos de saúde diminuindo a confiança dos pacientes hospitalares e comprometendo a segurança cibernética. Além disso, as nuvens tradicionais não possibilitam um controle total do usuário, podendo sofrer de eventos inexplicáveis de falha de rede. Portanto, a alternativa de computação em névoa (*FOG computing*) se torna uma alternativa viável para diminuir os problemas que ocorrem nas nuvens tradicionais já que essa tecnologia distribui a infra-estrutura trazendo os recursos de armazenamento mais perto da rede local e apenas alguns dados são realmente repassados para a nuvem principal, melhorando a qualidade da comunicação e reduzindo a quantidade de dados

roteados.

Os principais objetivos de Ali et al. (2022) incluem o planejamento da capacidade e cobertura da rede, minimização do custo de *hardware* e infra-estrutura e também a diminuição computacional e operacional. Em suma, melhorar a Qualidade de serviço da rede ao mesmo tempo que reduz o custo necessário para se implementar a arquitetura.

Em Ali et al. (2022) três algoritmos evolutivos baseados em *swarm intelligence* são propostos: o *Discrete Fireworks Algorithm* (DFWA) com três métodos de pesquisa (LSM), o *Discrete Artificial Bee Colony* (DABC) também com 3 LSM e o *Low-complexity bio-geography-based Optimization* (LC-BBO). Devido a esses algoritmos reproduzirem a ideia de um exame, onde os nós se comportam coletivamente de forma descentralizada e auto-organizada, a similaridade do funcionamento de uma arquitetura *FOG Computing* pode ser traçada.

Oito instâncias diferentes foram utilizadas para extrair os resultados de Ali et al. (2022). Além dos três algoritmos evolucionários propostos mais um algoritmo é adicionado na comparação, o Algoritmo Genético. Ao comparar os quatro algoritmos propostos, o DFWA-3-LSM superou os outros em termos de custo médio de planejamento, atingindo valores até 18% menor que seus concorrentes.

3.5 Planejamento e otimização de uma rede para Campus Inteligente

O trabalho de Jr et al. (2022) traz o contexto de redes IoT aplicadas em *Smart Campus*. A dificuldade apresentada está na interoperabilidade e a heterogeneidade ao planejar uma rede nesse contexto, pois diversos tipos de dados são trafegados por meio de aplicações distintas e cada uma delas é processada em locais diferentes. Portanto é necessário a implantação de *gateways* capazes de suportar diversas tecnologias de comunicação. Além disso, esses *gateways* devem prover uma boa largura de banda e possuir uma boa eficiência energética. Para atender todos esses requisitos, o trabalho utiliza um projeto de gateway IoT chamado SOFTWAY4IoT que implementa um software de rede virtualizado capaz de integrar a computação em nuvem com a computação em névoa.

Além do desafio da escolha do *gateway*, Jr et al. (2022) enfatiza que outros desafios devem ser levados em consideração ao planejar uma rede para *Smart Campus*. A escolha da tecnologia capaz de fazer a comunicação entre os clientes deve levar em consideração as características arquitetônicas do ambiente, o número de elementos que acessam a rede ao mesmo tempo e a mobilidade dos clientes na área de cobertura. Dito isso, a distribuição de *gateways* dentro desses ambientes é um grande desafio por compreender diversos fatores que interferem no posicionamento o quantificação.

O objetivo descrito em Jr et al. (2022) é propor um método para planejamento e otimização da implantação de *gateways* IoT tendo em vista o uso do *gateway* SOFTWAY4IoT. O trabalho visa minimizar o número de *gateways* necessários e maximizar a área de cobertura, considerando as restrições de capacidade e alcance da comunicação e do enlace de dados. As tecnologias escolhidas para no estudo foram: LoRa, BLE e WI-Fi. A primeira foi escolhida devido a sua grande área de cobertura e seu baixo consumo de energia. Já a BLE, apesar de possuir um curto alcance ela se equipara ao baixo consumo de energia do LoRa. Por último, o Wi-Fi foi escolhido por ser uma das tecnologias mais populares em redes sem fio, possui um alcance que está entre as duas anteriores e tem um alto consumo de energia.

Nesse trabalho, Jr et al. (2022) propõe também mais de um tipo de algoritmo. Nesse caso, quatro tipos são apresentados para tentar resolver o problema de posicionamento de *gateways* em uma área predefinida. O primeiro utiliza um modelo de otimização de programação linear, o segundo método usa o agrupamento K-means e o terceiro o agrupamento PSO-Simples e o último PSO-Híbrido. Mas antes de verificar o posicionamento, a proposta segue primeiro para a definição da quantidade de *gateways* utilizando um modelo de programação linear.

Para comparar os métodos e definir os resultados, o trabalho Jr et al. (2022) utiliza como métrica o coeficiente de silhueta com o intuito de ranquear o melhor dos algoritmos. Três cenários reais foram aplicados nos experimentos. O trabalho conclui que todos os modelos propostos podem ser usados para o planejamento de uma rede sem fio, desde que o cenário de implementação seja apropriado ao algoritmo, mas de forma geral os modelos K-means e PSO-Hibrido apresentaram melhores coeficientes de silhueta.

3.6 Planejamento eficiente de *gateways* em redes voltadas para Internet das Coisas

No último trabalho analisado, Gravalos et al. (2016) cita também o problema do planejamento de redes IoT que atende as diversas aplicações com diferentes padrões de dados e tecnologias de comunicação. Como alternativa se destaca o planejamento da arquitetura de *FOG computing* devido a necessidade de processar dados com eficiência e baixo custo. Além disso, nessa arquitetura, os *gateways* podem exercer um papel importante em executar a agregação e processamento de dados antes que a informação chegue ao destino final. Portanto, ao implementar uma infraestrutura de computação em névoa deve ser levado em conta a eficiência da largura de banda e o custo de instalação.

No trabalho de Gravalos et al. (2016) é enfatizado que o objetivo é minimizar o número de *gateways* junto com dispositivos finais de IoT correspondentes com o intuito de otimizar o custo geral de instalação garantindo que a Qualidade de Serviço não seja prejudicada.

Em Gravalos et al. (2016), por se tratar de um problema formulado como *Integer Linear Program* (ILP) o autor propõe apenas um algoritmo de própria autoria para tentar resolver o problema abordado. Esse algoritmo define primeiro os candidatos a localização dos *gateways* e posteriormente define o posicionamento ideal dos dispositivos de transmissão e também dos próprios *gateways*. A saída desse algoritmo garante que todos os nós foram atendidos por um *gateway*, que nenhum tráfego será perdido e que seja mínima o custo total da implantação da infraestrutura de rede IoT.

Como resultado, o algoritmo proposto por Gravalos et al. (2016) foi executado em diferentes cenários simulados de planejamento de rede e foi constatado que à medida que a topologia ficava mais densa a rede se tornava mais eficiente. Por fim, o autor conclui que o algoritmo, de forma geral, conseguiu apresentar uma economia de até 52% no custo total de instalação e, em casos mais específicos, a economia chegou a 60%.

3.7 Considerações Finais

Este capítulo expôs os principais trabalhos relacionados aos assuntos tratados por esse trabalho. A Tabela 3.1 apresenta um resumo contendo, as principais características dos trabalhos citados de forma a facilitar a comparação entre eles.

Tabela 3.1: Comparativo das principais características dos trabalhos relacionados

Autores / Trabalhos	**Cenário**	**Objetivo**	**Algoritmo**	**Ajuste de parâmetros**	**Arquitetura de Comunicação**
(LIMA; CARRANO; TAKAHASHI, 2012)	-	Minimizar *gateways*	Algoritmo Genético	Não	Wi-Fi
(KARTHIKEYA; VIJETH; MURTHY, 2016)	*Smart City*	Minimizar *gateways*	NewIoTGateway-Select	Não	Wi-Fi, RFID, Bluetooth e ZigBee
(MATNI et al., 2019)	*Asistência médica 4.0*	Balancear o QoS com o custo	Fuzzy C-Means	Não	Lora
(ALI et al., 2022)	*Smart Campus*	Balancear o QoS com o custo	Algoritmos evolutivos	Sim	*Fog Computing*
(JR et al., 2022)	*Smart Campus*	Minimizar *gateways*	LP, K-means e PSO	Sim	LoRa, BLE e Wi-Fi
(GRAVALOS et al., 2016)	-	Balancear o QoS com o custo	LP	Não	*Fog Computing*

Em resumo, os trabalhos listados apresentam soluções diversas para problemas bem parecidos. Todos eles querem melhorar a qualidade do planejamento da rede de alguma forma e os resultados mostraram que cada trabalho teve sucesso nos seus experimentos. Com isso, é possível dizer que a otimização do planejamento de redes voltadas para dispositivos IoT é totalmente possível.

Todos os cenários descritos em cada trabalho visam melhorar o ecossistema de uma rede IoT, seja ela implementada em um local menor, como uma *Smart Campus*, ou em um cenário maior, como em uma *Smart City*. Portanto é possível concluir que a aplicabilidade da solução proposta por esse trabalho é bastante ampla.

Importante ressaltar que apenas dois trabalhos procuram ajustar os parâmetros de configuração da rede em tempo de execução do algoritmo, os outros apenas fixam os valores.

Ao analisar esses trabalhos um algoritmo foi escolhido para a implementação do otimizador na solução de software proposta por esse trabalho. Os algoritmos de autoria própria nos trabalhos Karthikeya, Vijeth e Murthy (2016) e Gravalos et al. (2016) foram desconsideradas na escolha. O K-means foi escolhido por ser um algoritmo que aparece como um dos melhores candidatos em cenário de rede LoRa no trabalho Jr et al. (2022).

Além disso, ele foi utilizado em Lima, Carrano e Takahashi (2012) para avaliar a eficácia do principal algoritmo do trabalho.

4 Metodologia e resultados

Na Figura 4.1 é possível observar, de forma simplificada, a arquitetura desenvolvida. O sistema foi focado em uma arquitetura REST (*Representational State Transfer*) onde o Cliente (*frontend*) consome a API (*backend*) por meio da comunicação de dados em JSON (*JavaScript Object Notation*).

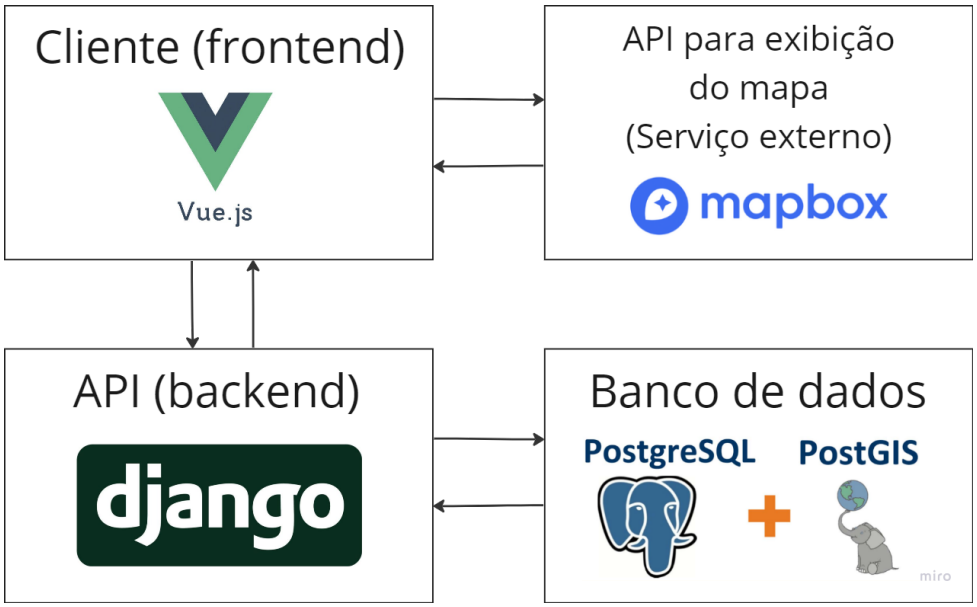

Figura 4.1: Arquitetura da Solução em Alto Nível.

Para a criação do sistema proposto, primeiramente, foram definidos os requisitos funcionais e não funcionais. Posteriormente o *backend* em Django foi criado juntamente com a integração com o banco de dados PostgreSQL e sua extensão PostGIS. Por último, o desenvolvimento do *frontend* em Vue.js com a comunicação com a API do MapBox foi focado. Nesse capítulo cada etapa de criação foi detalhada e, ao fim, um Diagrama de Sequência foi gerado para demonstrar o fluxo dos dados entre as camadas do sistema. O código pode ser acessado em ⟨https://github.com/Rodrigo947/gerador-otimizador-instancias-lorawan⟩.

4.1 Requisitos funcionais e não funcionais

Primeiramente, para a criação da aplicação proposta, foi necessário definir o comportamento esperado em termos de entrada, processamento e saída, além de como seria sua usabilidade e confiabilidade. Portanto os seguintes requisitos funcionais e não funcionais da aplicação foram definidos:

4.1.1 Requisitos funcionais

- RF01: O sistema deve exibir um mapa interativo que permita o usuário localizar qualquer região do mundo;
- RF02: O usuário deve conseguir desenhar no mapa todas as regiões que serão atendidas pela solução;
- RF03: O usuário deve conseguir definir os pesos que cada área terá na geração de clientes;
- RF04: O sistema deve permitir que o usuário insira os seguintes parâmetros: seed, quantidade de clientes, potência de transmissão, ganho da antena, frequência e SF;
- RF05: Ao gerar a instância o sistema deve exibir a solução encontrada no mapa;
- RF06: Um arquivo .JSON deve ser disponibilizado para download com todas as informações da instância gerada.

4.1.2 Requisitos não funcionais

- RNF01: O usuário deve conseguir utilizar o sistema pelo navegador Google Chrome;
- RNF02: O sistema deve ser responsivo;
- RNF03: A instância gerada sempre deve possuir uma solução válida, ou seja, todos os clientes devem estar atendidos por um *gateway*;
- RNF04: Todos os dados inseridos pelo usuário devem ser validados de acordo com a Tabela 4.1 antes que a instância seja gerada.

Tabela 4.1: Parâmetros e validações

Parâmetro	Tipo	Validação
Peso das regiões	Porcentagem	A soma deve ser igual a 100%
Seed	Número inteiro	Valor >= 1
Quantidade de clientes	Número inteiro	Valor >= 1
Potência de transmissão	Número inteiro em mW	Apenas os valores 5, 20, 50 e 100 são permitidos [1]
Ganho da antena	Número decimal em dBi	Valor >= 0,01
Frequência	Número inteiro em MHz	Apenas os valores 433 e 915 são permitidos [2]
SF	Número inteiro	Apenas os valores 7, 8, 9, 10, 11 e 12 são permitidos [3]

- RNF05: Ao exibir a solução no mapa o sistema deve possuir mecanismos que facilitem a visualização da mesma.

4.1.3 Diagrama de atividades

Definido os requisitos da aplicação, um diagrama de atividades (Figura 4.2) foi criado para representar o fluxo do sistema. Nele é possível observar que a RF01 é responsável pela primeira interação do usuário no sistema, seguido das RFs 2, 3 e 4 que são as responsáveis pela entrada de dados. Caso todas as entradas estejam válidas o sistema segue para as RFs 5 e 6, finalizando o fluxo.

[1] As potências foram restringidas a apenas quatro valores, pois esse trabalho leva em consideração o estudo dos limites distância do Islam, Ray e Pasandideh (2020)

[2] De acordo com Committee (2020), apenas os dois valores de frequência citados estão disponíveis no Brasil.

[3] De acordo com Figueiredo e Silva (2020), o SF pode variar dentre os seis disponíveis.

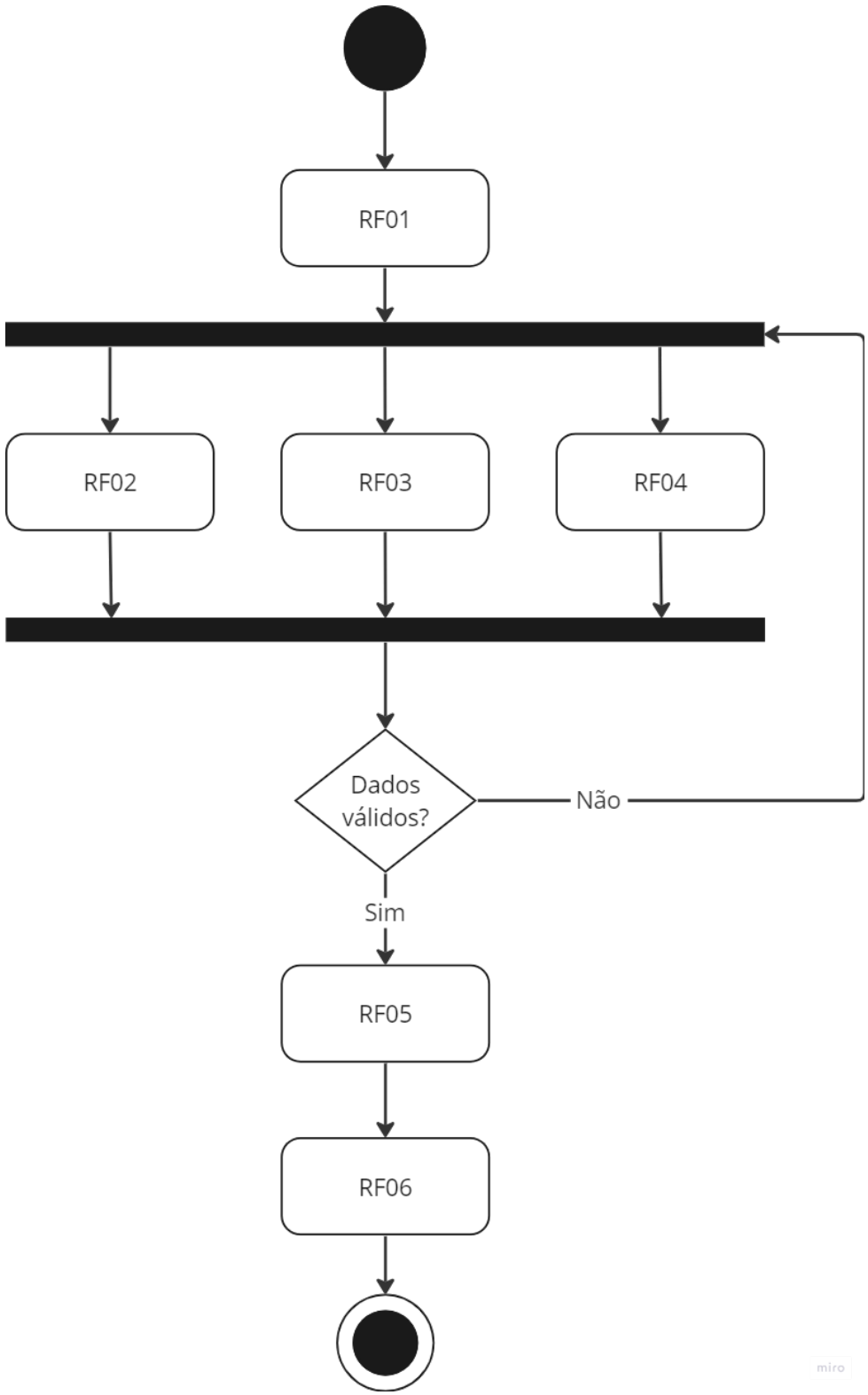

Figura 4.2: Diagrama de Atividades.

4.2 Desenvolvimento do *backend*

O desenvolvimento do *backend* pode ser dividido nas seguintes etapas: validações dos dados, geração dos clientes, validação dos *gateways* e geração dos *gateways*. Cada uma das etapas será detalhado nesse capítulo.

4.2.1 Validações dos dados

Todas as validações citadas na Tabela 4.1 foram validadas no *backend* por meio de um recurso do *framework* Django chamado *Serializers*[5]. Nele é possível programar suas próprias validações de dados e, no momento que o usuário enviar a requisição, se alguma validação falhar, o erro é enviado para o *frontend* e nenhum outro processo é feito.

Além das validações básicas, existia mais uma validação que era necessário realizar: o polígono que o usuário desenha no *frontend*. O desenho de um polígono não poderia ter linhas que sobrepõe outras linhas do mesmo polígono. Exemplos de sobreposição podem ser observadas na Figura 4.3.

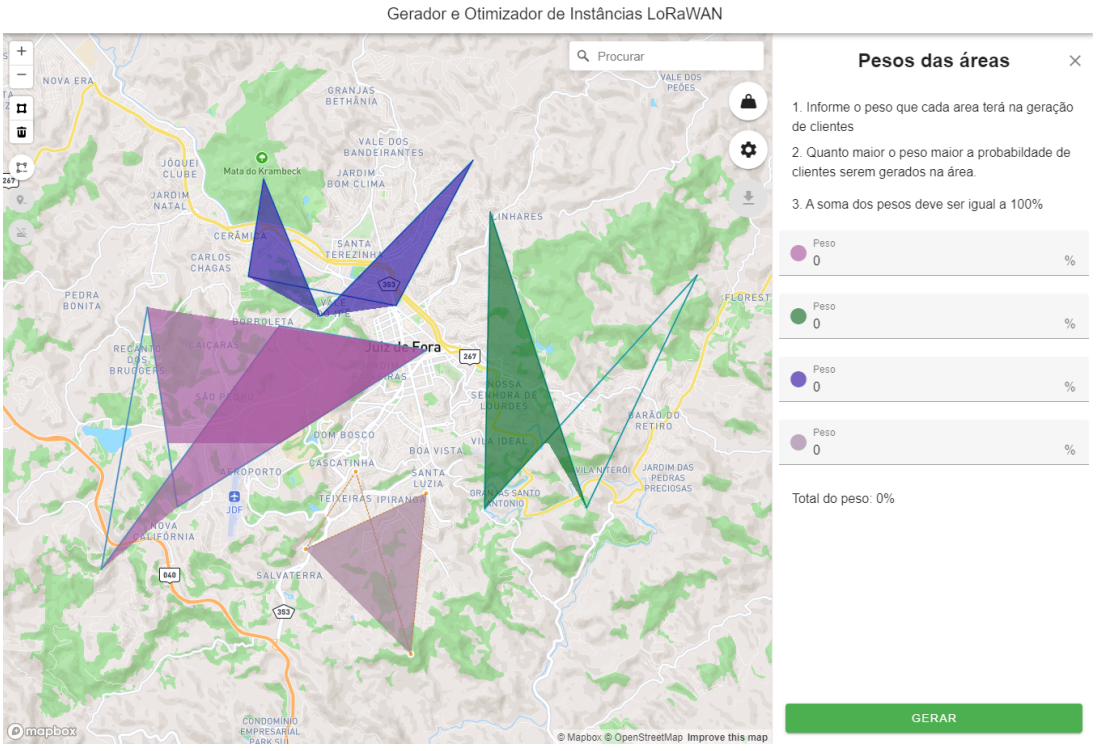

Figura 4.3: Polígonos com linhas sobrepostas.

[5]https://www.django-rest-framework.org/api-guide/serializers/

Para resolver esse problema foi utilizado o banco de dados relacional PostgreSQL[6] em conjunto com sua extensão que oferece suporte para armazenar, indexar e consultar dados geográficos, o PostGIS[7]. Com o uso da função **St_IsValid**[8], que recebe como parâmetro as coordenadas dos pontos do polígono, é possível saber se o desenho feito pelo usuário é válido ou não.

4.2.2 Geração dos clientes

Com todos os dados corretos o sistema armazena as coordenadas e os pesos das áreas e segue para a geração de clientes da instância. Primeiramente é criado um dicionário para auxiliar na distribuição dos clientes. O preenchimento do dicionário segue a lógica do seguinte exemplo: supomos que o usuário adicione três áreas (Figura 4.4), a primeira com 50%, a segunda com 30% e terceira com 20%, totalizando os 100%. O algoritmo (Algoritmo 1) irá criar um dicionário onde as chaves de 0 a 49 terão como valor o identificador da primeira área, as chaves 50 a 79 serão vinculadas ao identificador da segunda área e o identificador da terceira área será preenchido nas chaves 80 a 99.

[6]https://www.postgresql.org/
[7]https://postgis.net/
[8]https://postgis.net/docs/ST_IsValid.html

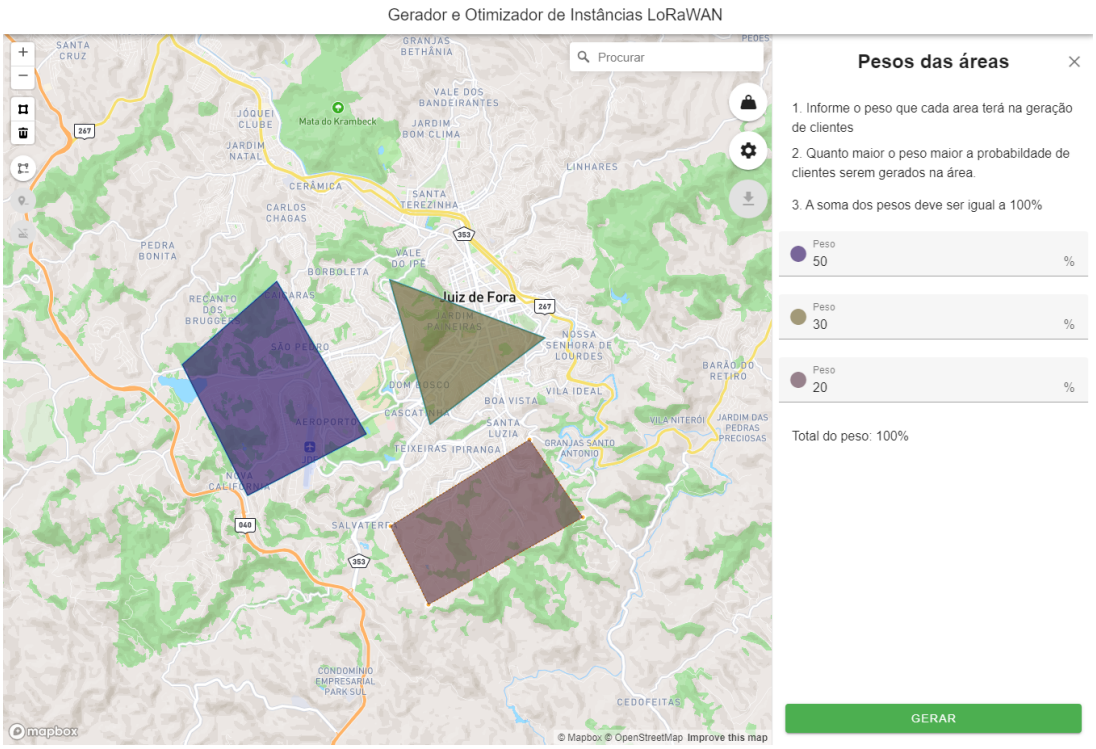

Figura 4.4: Exemplo de áreas informadas pelo usuário e seus pesos.

Algoritmo 1: Algoritmo de criação do dicionário de pesos

Output: *Dicionário de pesos*

```
1  dicionarioDePesos ← {}
2  inicioDaFaixa ← 0
3  for area ← primeiraArea to ultimaArea do
4      idDaArea ← area.getId()
5      pesoDaArea ← area.getPeso()
6      for n ← inicioDaFaixa to inicioDaFaixa + pesoDaArea do
7          dicionario[n] ← idDaArea
8          n ← n + 1
9      end for
10     inicioDaFaixa ← inicioDaFaixa + pesoDaArea
11 end for
12 return dicionarioDePesos;
```

A partir do dicionário a quantidade de clientes é distribuída utilizando uma função de geração de números pseudo-aleatórios que recebe a *seed* informada pelo usuário no

frontend. Assim um novo dicionário é gerado onde cada chave representa o identificador da área e o valor representa a quantidade de clientes.

Ao final, o PostGIS é utilizado novamente para gerar as coordenadas dos clientes. Utilizando a função **ST_GeneratePoints**[9], ao informar as coordenadas da área, a quantidade de clientes e a *seed* nos parâmetros da função, a extensão gera pontos pseudo-aleatórios que estão contidos na área informada. Os dados dos clientes são armazenados bem como sua relação com a área a qual pertence. O Algoritmo 2 mostra a lógica utilizada nessa etapa.

Algoritmo 2: Algoritmo de criação dos clientes

Output: *Clientes e suas coordenadas*

```
1  dictClientesPorArea ← {}
2  n ← 0
3  while n < quantidadeClientes do
4      idArea ← dicionarioDePesos[gerarNumeroRandomico0a99()]
5      dictClientesPorArea[idDaArea] ← dictClientesPorArea[idArea] + 1
6      n ← n + 1
7  end while
8  area ← dictClientesPorArea.proximaArea()
9  while area ≠ null do
10     areaCoords ← area.getCoordenadas()
11     quantClientes ← area.getQuantidadeClientes()
12     clientes ← ST_GeneratePoints(areaCoords, quantClientes, seed)
13     salvaClientes(clientes)
14     area ← dictClientesPorArea.proximaArea()
15 end while
```

4.2.3 Validação dos *gateways*

Antes de gerar um *gateway* é preciso definir que tipo de validação será feita para dizer se os clientes serão atendidos. No sistema duas verificações foram feitas: se a distância física entre o *gateway* e o cliente é válida e se o *gateway* irá sofrer uma interferência (SNR) que

[9] https://postgis.net/docs/ST_GeneratePoints.html

irá inibir a comunicação com o *gateway*.

Esse trabalho utilizou os valores aproximados do trabalho Islam, Ray e Pasandideh (2020) e retirou a Tabela 4.2 para a validação da distância.

Tabela 4.2: Limites de distância de acordo com o SF e a Potência de Transmissão (mW)

SF/ Potência de Transmissão (mW)	5	20	50	100
7	2000 metros	2500 metros	3200 metros	3800 metros
8	2400 metros	3000 metros	3600 metros	4400 metros
9	2600 metros	3500 metros	4200 metros	5000 metros
10	3000 metros	4000 metros	5000 metros	6000 metros
11	3500 metros	4800 metros	6000 metros	7400 metros
12	4000 metros	5800 metros	7300 metros	9000 metros

Já para validar a interferência foi considerado o cenário onde todos os clientes que estão ligados ao *gateway* estão tentando comunicar ao mesmo tempo. Nesse cenário, para calcular o SNR do *gateway* é necessário primeiro calcular o somatório de todas as potências de envio de dados que os clientes exercem para se comunicar. A potência de cada cliente pode ser calculada pela Fórmula 4.1, baseada na equação de Friss em espaço livre (FRIIS, 1946; AREF; SIKORA, 2014), onde PC é a potência recebida do cliente, P a potência (mW), GA o ganho da antena (dBi), D (metros) a distância física entre cliente e *gateway* e F a frequência (MHz). Cada um desses parâmetros são informados pelo usuário no *frontend*.

$$PC = P \times GA^2 \times \left(\frac{300}{4 \times \pi \times D \times F} \right)^2 \tag{4.1}$$

Por fim, o SNR do *gateway* para um cliente é calculado utilizando a Fórmula 4.2, baseada no teorema de Shannon-Hartley (SHANNON, 1948), onde P é a potência informada pelo usuário e TPC é o somatório de todas as potências recebidas dos outros clientes que estão comunicando com o *gateway*.

$$SNR = 20 \times \log_{10} \left(\frac{P}{TPC} \right) \tag{4.2}$$

Com o SNR calculado a Tabela 4.3 é consultada para identificar se o valor do

SNR irá impossibilitar a comunicação. A Tabela foi retirada do trabalho Jouhari et al. (2023) e leva em consideração o SF para indicar o limite.

Tabela 4.3: Limites de SNR por SF. Fonte: (JOUHARI et al., 2023)

SF	7	8	9	10	11	12
Limite de SNR	-7,5	-10	-12,5	-15	-17,5	-20

4.2.4 Geração dos *gateways*

Para a geração da localização dos *gateways* da instância o Algoritmo 3 foi criado e tem como principal recurso o uso do K-means para a otimização da quantidade de *gateways*.

Na primeira execução apenas um *cluster* é gerado, ou seja, todos os clientes são alocados em apenas um *gateway* que está sempre localizado no centro do *cluster*. Caso esse *gateway* seja inválido uma nova chamada ao algoritmo K-means é feita, mas agora gerando dois *clusters*. Nessa segunda iteração cada *gateway* terá seu próprio conjunto de clientes e, caso algum deles não seja válido, uma nova chamada do K-means é feita com apenas os clientes daquele determinado *gateway* e uma nova divisão de dois *clusters* é feita e assim sucessivamente até que todos os *gateways* gerados sejam válidos.

Algoritmo 3: Algoritmo de criação dos gateways

Output: *Gateways e suas coordenadas*

```
1  gatewaysValidos ← []
2  criarGateways(clientes, 1)
3  Function criarGateways(clientes, quantidadeClusters):
4      clusters ← kmeans(clientes, quantidadeClusters)
5      for cluster ← clusters[n] to n = clusters.size() do
6          if gatewayValido(cluster) then
7              gatewaysValidos ← cluster
8          else
9              criarGateways(cluster.getClientes(), 2)
10         end if
11     end for
```

4.3 Desenvolvimento do *frontend*

Para o desenvolvimento do *frontend* o primeiro passo era mostrar o mapa iterativo. Para isso, a API do MapBox[10] foi implementada juntamente com o *framework* Vue.js[11]. Como é possível ver na Figura 4.5, a API, além de exibir o mapa, possibilita a procura de localizações no campo superior e o desenho de áreas com as ferramentas da lateral esquerda.

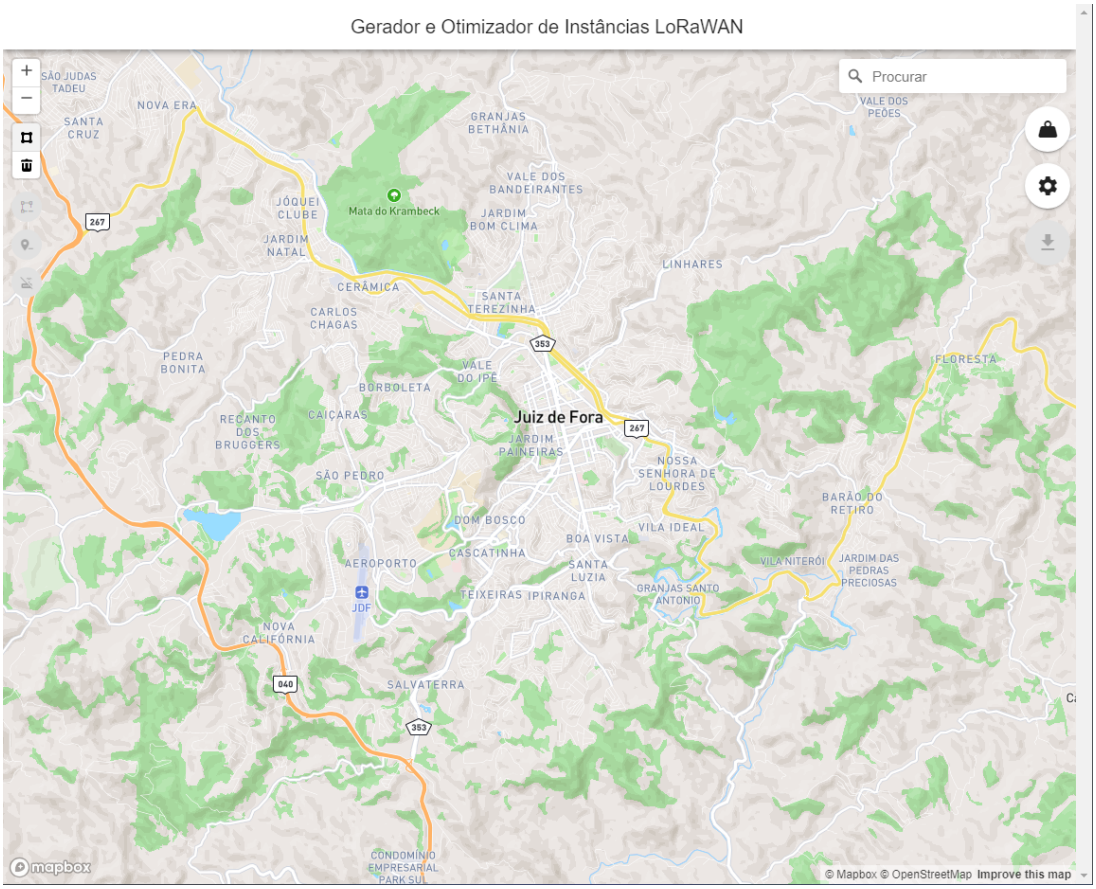

Figura 4.5: Tela inicial

Um menu lateral foi criado (Figuras 4.6 e 4.7) para que o usuário possa informar os pesos de cada região e os parâmetros de configuração da rede e da instância. Ao clicar no botão de GERAR, caso alguma configuração informada não esteja de acordo com as validações, um aviso aparecerá na tela informando o erro (Figura 4.8).

[10] https://www.mapbox.com/
[11] https://vuejs.org/

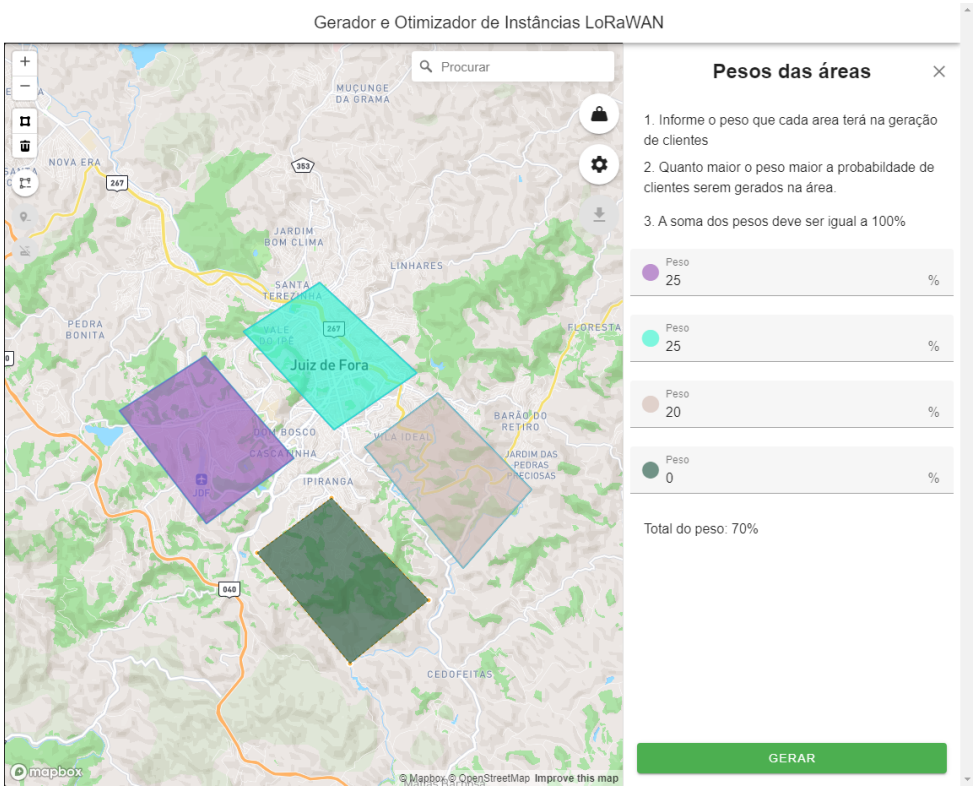

Figura 4.6: Tela de pesos

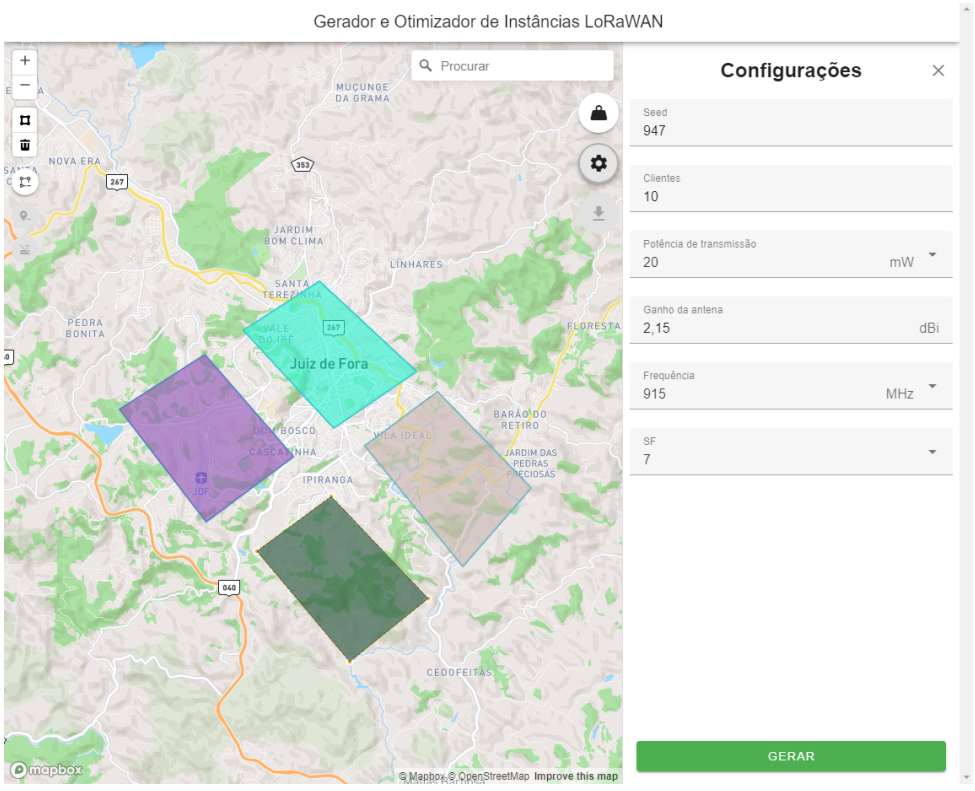

Figura 4.7: Tela de configurações dos parâmetros

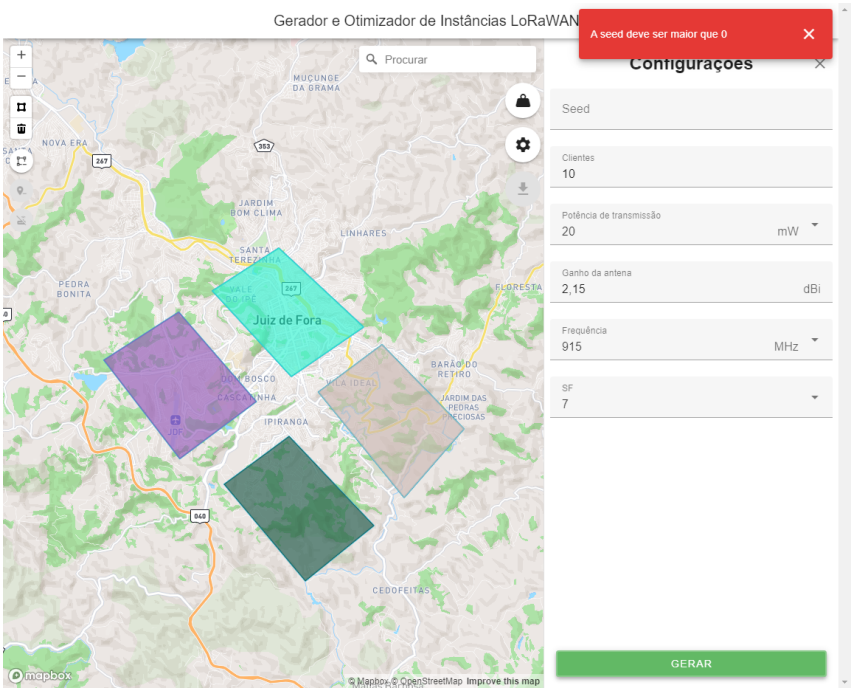

Figura 4.8: Aviso de erro

Na Figura 4.9 é possível visualizar como foi feita a visualização do resultado. Cada cliente é representado por um ícone de marcador e os *gateways* por um ícone de um roteador. O cliente possui a cor igual ao *gateway* o qual está ligado, facilitando a visualização dos grupos formados.

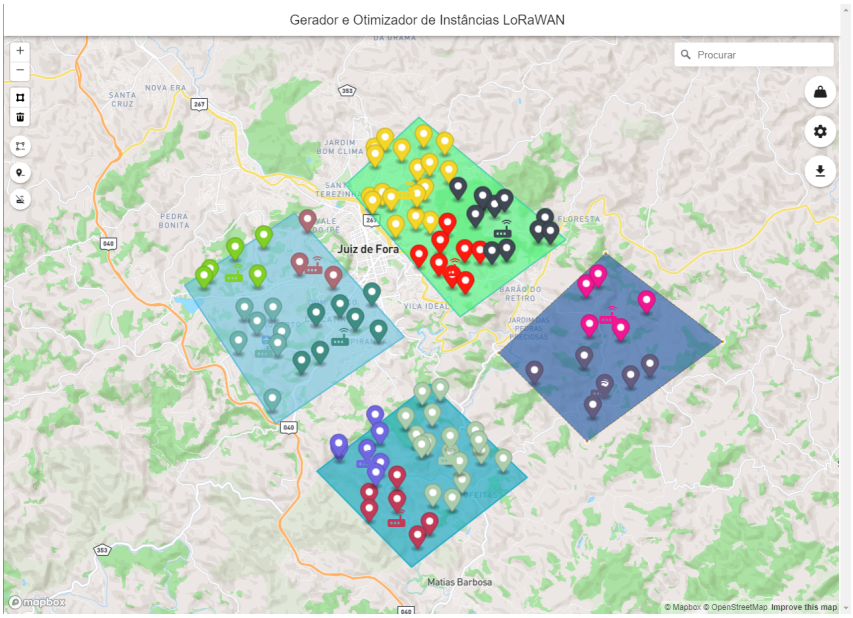

Figura 4.9: Visualização do resultado

Por fim, para melhorar a usabilidade da visualização dos resultados, novos botões são habilitados na esquerda da tela para esconder ou mostrar as áreas desenhadas e os ícones dos *gateways* e clientes. O botão de download do arquivo .JSON também é habilitado na direita da tela caso o usuário queira baixar a instância gerada.

O arquivo da instância possui todos os parâmetros informadas pelo usuário (Figura 4.10) bem como as coordenadas dos pontos que formam as áreas desenhadas e seus pesos (Figura 4.11).

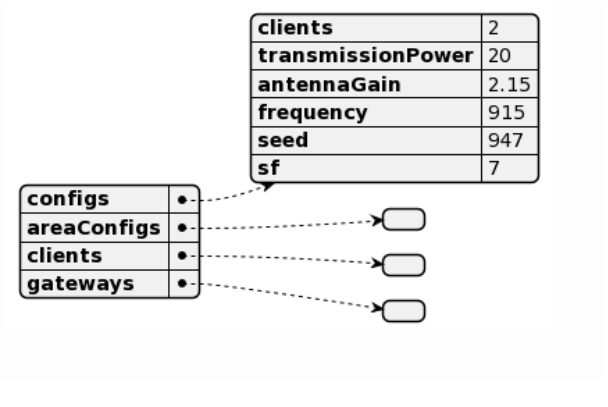

Figura 4.10: Representação em diagrama dos parâmetros informados no arquivo .JSON

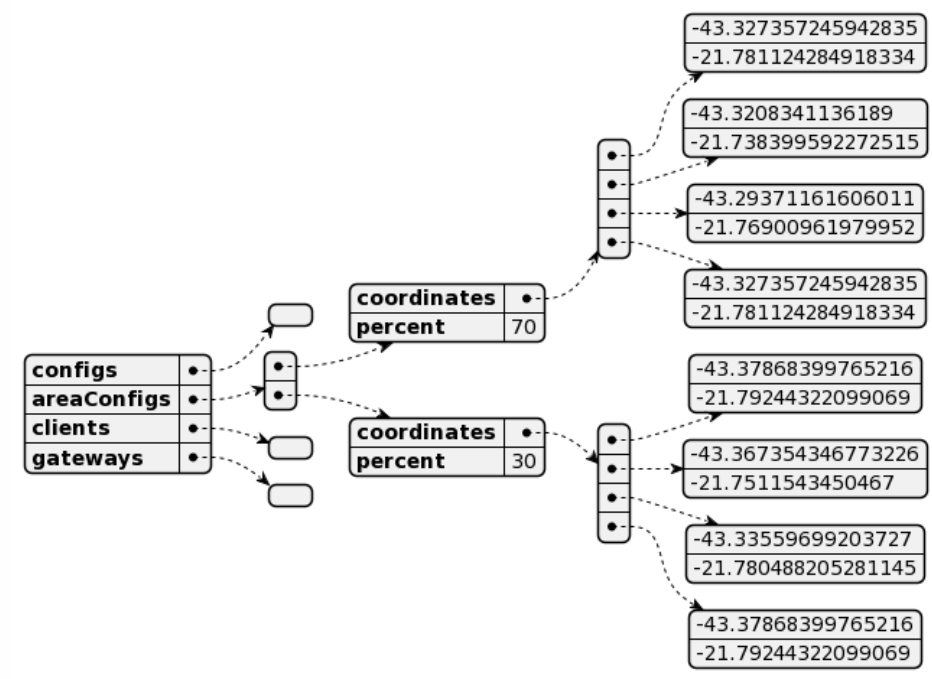

Figura 4.11: Representação em diagrama das áreas informadas no arquivo .JSON

No final do arquivo .JSON as coordenadas dos clientes (Figura 4.12) e dos *gateways* (Figura 4.13) gerados pelo *backend* também estão presentes.

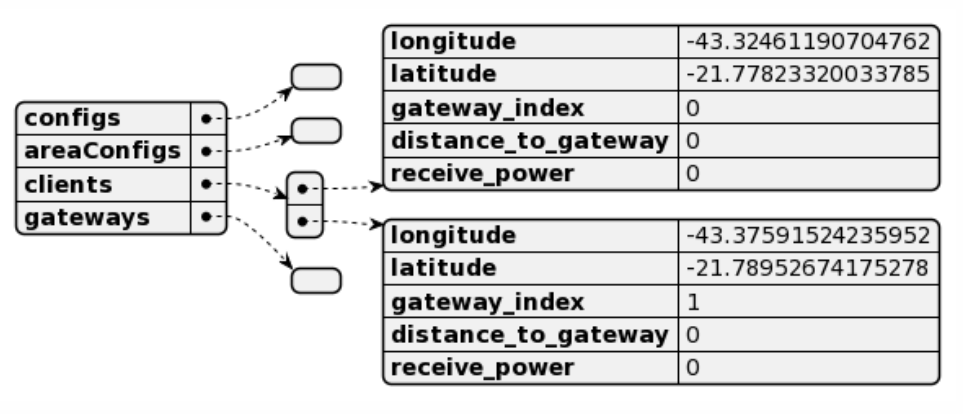

Figura 4.12: Representação em diagrama dos clientes informados no arquivo .JSON

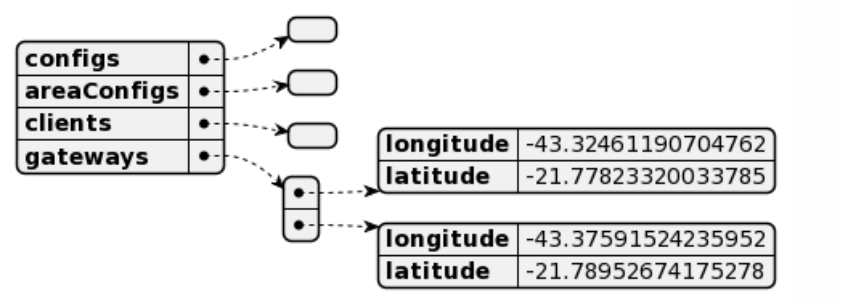

Figura 4.13: Representação em diagrama dos *gateways* informados no arquivo .JSON

4.4 Diagrama de Sequência

Ao finalizar o desenvolvimento, foi possível gerar um diagrama de sequência do principal fluxo da aplicação. A Figura 4.14 demonstra o fluxo de dados entre as camadas do sistema. Nele é possível visualizar a comunicação e interação entre o usuário e objetos bem como identificar a ordem das mensagens trocadas entre os elementos.

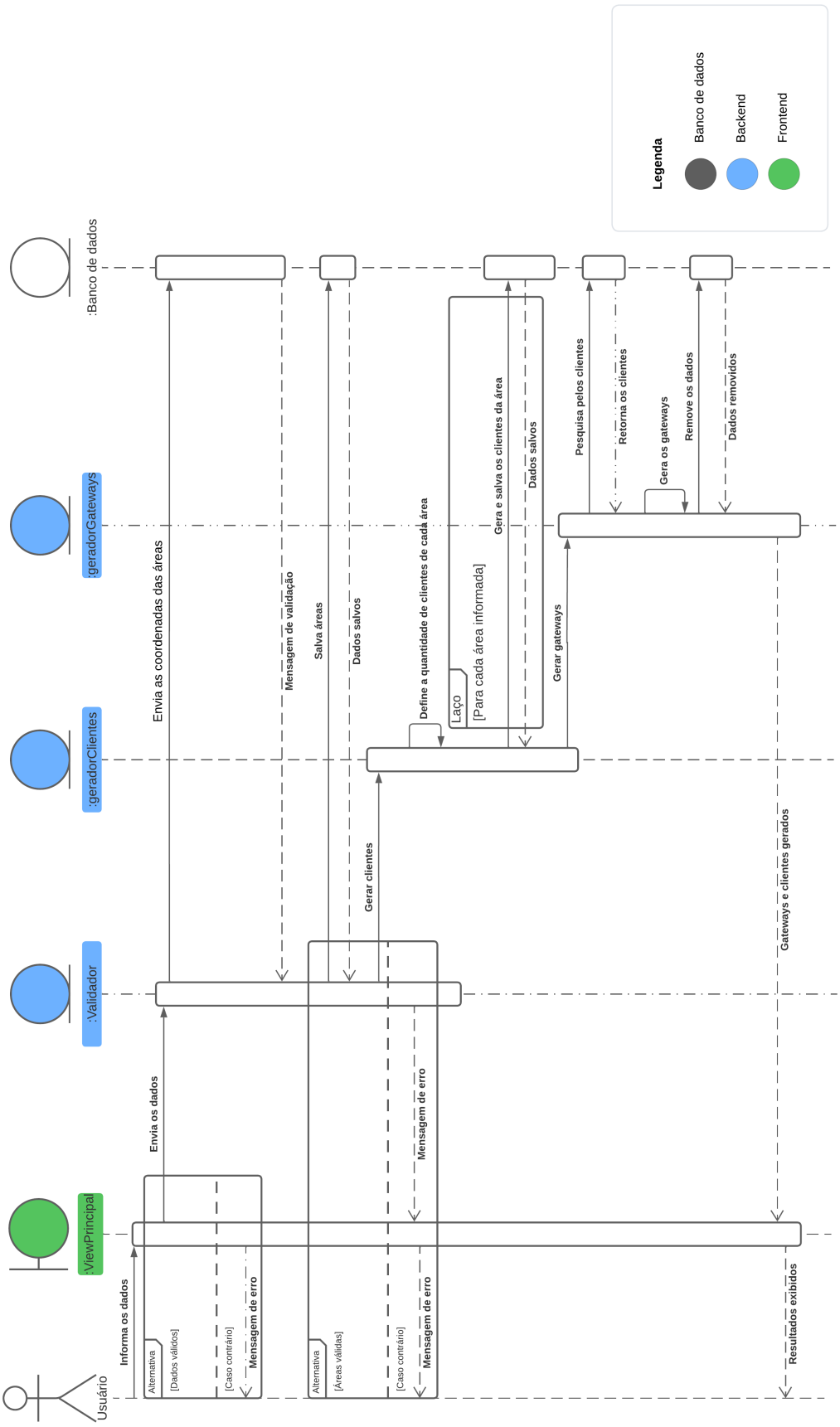

Figura 4.14: Diagrama de sequência

5 Conclusões e trabalhos futuros

Este trabalho de conclusão de curso teve como finalidade desenvolver um sistema capaz de auxiliar os administradores de rede a projetar uma LoRaWAN entendo melhor a rede e diminuindo os custos necessários para implantação. Além disso, o sistema é capaz de fornecer *datasets* que irão auxiliar futuras pesquisas de algoritmos de otimização de redes LPWANs.

O sistema foi desenvolvido respeitando diversos padrões existentes na rede LoRaWAN e um conjunto de tecnologias foram utilizadas para trazer a melhor usabilidade para o usuário final. O código pode ser acessado em ⟨https://github.com/Rodrigo947/gerador-otimizador-instancias-lorawan⟩.

Durante a criação do sistema, dificuldades foram encontradas para desenvolver a ferramenta que o usuário conseguisse desenhar uma área válida no mapa. Portanto, a API do Mapbox e a extensão PostGIS foram cruciais para que o *frontend* tivesse todas as devidas funcionalidades propostas. Além disso, a dificuldade de entregar *gateways* em locais válidos da instância, levou a intensa pesquisa de como validar um *gateway* em uma rede LoRAWAN e, com o algoritmo K-means, a estratégia de colocar um *gateway* no centro de um *cluster* facilitou a otimização da quantidade de *gateways* da rede.

O trabalho trouxe contribuições importantes no estado da arte relacionado a redes LoRaWAN. A disponibilidade de uma aplicação, capaz de gerar instâncias nos mais diversos cenários, possibilita a criação de outros trabalhos que foquem na otimização da rede. Além disso, devido a grande quantidade de parâmetros, atualmente há uma certa dificuldade de configurar a rede de forma correta. Com a aplicação, os administradores da rede podem testar as configurações e observar como se comportam, facilitando o entendimento e implantação.

Para trabalhos futuros novos parâmetros podem ser adicionados ao algoritmo de otimização que prezem pela economia de energia ou/e considerar, no cálculo do SNR, a interferência dos elementos de uma cidade já que, nesse trabalho, é considerado um ambiente aberto sem interferência do meio.

Bibliografia

ALI, H. M.; LIU, J.; BUKHARI, S. A. C.; RAUF, H. T. Planning a secure and reliable iot-enabled fog-assisted computing infrastructure for healthcare. *Cluster Computing*, Springer, v. 25, n. 3, p. 2143–2161, 2022.

ALLIANCE, L. *What is LoRaWAN® Specification*. 2022. ⟨https://lora-alliance.org/about-lora-alliance/⟩. Acessado: 30/10/2022.

AREF, M.; SIKORA, A. Free space range measurements with semtech lora™ technology. In: IEEE. *2014 2nd international symposium on wireless systems within the conferences on intelligent data acquisition and advanced computing systems*. [S.l.], 2014. p. 19–23.

BACCELLI, F.; BLASZCZYSZYN, B.; MUHLETHALER, P. An aloha protocol for multihop mobile wireless networks. *IEEE transactions on information theory*, IEEE, v. 52, n. 2, p. 421–436, 2006.

COMMITTEE, L. A. T. *RP2-1.0.1 LoRaWAN® Regional Parameters*. 2020. ⟨https://resources.lora-alliance.org/technical-specifications/rp2-1-0-1-lorawan-regional-parameters⟩. Acessado: 20/06/2023.

CORPORATION, S. S. C. 2022. ⟨https://www.semtech.com/⟩. Acessado: 19/10/2022.

FIGUEIREDO, L. M.; SILVA, E. F. Cognitive-lora: adaptation-aware of the physical layer in lora-based networks. In: IEEE. *2020 IEEE Symposium on Computers and Communications (ISCC)*. [S.l.], 2020. p. 1–6.

FOUNDATION, D. S. *Why Django?* 2022. ⟨https://www.djangoproject.com/start/overview/⟩. Acessado: 30/10/2022.

FRIIS, H. T. A note on a simple transmission formula. *Proceedings of the IRE*, IEEE, v. 34, n. 5, p. 254–256, 1946.

GRAVALOS, I.; MAKRIS, P.; CHRISTODOULOPOULOS, K.; VARVARIGOS, E. A. Efficient gateways placement for internet of things with qos constraints. In: IEEE. *2016 IEEE Global Communications Conference (GLOBECOM)*. [S.l.], 2016. p. 1–6.

HASAN, M. *State of IoT 2022: Number of connected IoT devices growing 18% to 14.4 billion globally*. 2022. ⟨https://iot-analytics.com/number-connected-iot-devices/⟩. Acessado: 30/10/2022.

ISLAM, N.; RAY, B.; PASANDIDEH, F. Iot based smart farming: Are the lpwan technologies suitable for remote communication? In: IEEE. *2020 IEEE International Conference on Smart Internet of Things (SmartIoT)*. [S.l.], 2020. p. 270–276.

JOUHARI, M.; SAEED, N.; ALOUINI, M.-S.; AMHOUD, E. M. A survey on scalable lorawan for massive iot: Recent advances, potentials, and challenges. *IEEE Communications Surveys & Tutorials*, IEEE, 2023.

JR, D. F.; OLIVEIRA, J. L.; SANTOS, C.; FILHO, T.; RIBEIRO, M.; FREITAS, L. A.; MOREIRA, W.; OLIVEIRA-JR, A. Planning and optimization of software-defined and virtualized iot gateway deployment for smart campuses. *Sensors*, MDPI, v. 22, n. 13, p. 4710, 2022.

KARTHIKEYA, S. A.; VIJETH, J.; MURTHY, C. S. R. Leveraging solution-specific gateways for cost-effective and fault-tolerant iot networking. In: IEEE. *2016 IEEE Wireless Communications and Networking Conference*. [S.l.], 2016. p. 1–6.

LIMA, M. P.; CARRANO, E. G.; TAKAHASHI, R. H. Multiobjective planning of wireless local area networks (wlan) using genetic algorithms. In: IEEE. *2012 IEEE Congress on Evolutionary Computation*. [S.l.], 2012. p. 1–8.

MATNI, N.; MORAES, J.; ROSÁRIO, D.; CERQUEIRA, E.; NETO, A. Optimal gateway placement based on fuzzy c-means for low power wide area networks. In: IEEE. *2019 IEEE Latin-American Conference on Communications (LATINCOM)*. [S.l.], 2019. p. 1–6.

MORISSETTE, L.; CHARTIER, S. The k-means clustering technique: General considerations and implementation in mathematica. *Tutorials in Quantitative Methods for Psychology*, v. 9, n. 1, p. 15–24, 2013.

PASQUA, E. *5 things to know about the LPWAN market in 2021*. 2021. ⟨https://iot-analytics.com/5-things-to-know-lpwan-market/⟩. Acessado: 30/10/2022.

PEREIRA, A. C.; ROMERO, F. A review of the meanings and the implications of the industry 4.0 concept. *Procedia Manufacturing*, Elsevier, v. 13, p. 1206–1214, 2017.

QIN, J.; LI, Z.; WANG, R.; LI, L.; YU, Z.; HE, X.; LIU, Y. Industrial internet of learning (iiol): Iiot based pervasive knowledge network for lpwan—concept, framework and case studies. *CCF Transactions on Pervasive Computing and Interaction*, Springer, v. 3, n. 1, p. 25–39, 2021.

SEMTECH. *What is LoRa?* 2022. ⟨https://www.semtech.com/lora/what-is-lora⟩. Acessado: 19/10/2022.

SHANNON, C. E. A mathematical theory of communication. *The Bell system technical journal*, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948.

SKYSENS. *Skysens Network Technology*. 2022. ⟨https://www.skysens.io/technology-network⟩. Acessado: 30/10/2022.

SOLANAS, A.; PATSAKIS, C.; CONTI, M.; VLACHOS, I. S.; RAMOS, V.; FALCONE, F.; POSTOLACHE, O.; PÉREZ-MARTÍNEZ, P. A.; PIETRO, R. D.; PERREA, D. N. et al. Smart health: A context-aware health paradigm within smart cities. *IEEE Communications Magazine*, IEEE, v. 52, n. 8, p. 74–81, 2014.

TALARI, S.; SHAFIE-KHAH, M.; SIANO, P.; LOIA, V.; TOMMASETTI, A.; CATALÃO, J. P. A review of smart cities based on the internet of things concept. *Energies*, MDPI, v. 10, n. 4, p. 421, 2017.

VINCENT, W. S. *Django for Beginners: Build websites with Python and Django*. [S.l.]: WelcomeToCode, 2021.

WOHLGETHAN, E. *Supportingweb development decisions by comparing three major javascript frameworks: Angular, react and vue. js*. Tese (Doutorado) — Hochschule für Angewandte Wissenschaften Hamburg, 2018.

YOON, C.; LIM, D.; PARK, C. Factors affecting adoption of smart farms: The case of korea. *Computers in Human Behavior*, Elsevier, v. 108, p. 106309, 2020.