

**MATEUS NAKAJO DE MENDONÇA
ERIC RODRIGUES PIRES**

SISTEMA WEB PARA INSTALAÇÃO DE ERBS

São Paulo
2018

**MATEUS NAKAJO DE MENDONÇA
ERIC RODRIGUES PIRES**

SISTEMA WEB PARA INSTALAÇÃO DE ERBS

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

São Paulo
2018

**MATEUS NAKAJO DE MENDONÇA
ERIC RODRIGUES PIRES**

SISTEMA WEB PARA INSTALAÇÃO DE ERBS

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

Orientador:

Bruno de Carvalho Albertini

São Paulo
2018

Dedico à minha família e meus amigos, que me ajudaram e incentivaram durante a minha graduação.

-- Mateus

Dedico este trabalho à minha mãe Telma, ao meu pai Flávio, e ao meu irmão Caio, que sempre me apoiaram nos estudos e trabalhos.

-- Eric

AGRADECIMENTOS

Ao Prof. Bruno de Carvalho Albertini, pela orientação constante na realização deste trabalho e sincero desprendimento.

Ao Prof. Antonio Fischer de Toledo, pelos textos trazidos e pela disposição a nos ajudar.

À Escola Politécnica da Universidade de São Paulo, pela oportunidade de realização do curso de bacharelado em Engenharia de Computação.

Ao Dr. Wilian França Costa, pela indicação de dados utilizados no projeto.

Ao Milton Kaoru Kashiwakura e o SIMET do NIC.br, pelo interesse e auxílio no desenvolvimento.

“O maior bem do Homem é uma mente inquieta.”

-- Isaac Asimov

RESUMO

Este projeto de formatura tem como objetivo criar um sistema capaz de calcular posições para a instalação de Estações Radiobase (ERBs) de forma que a cobertura da rede de ERBs seja máxima. A partir da região dada como entrada, o sistema obterá seus dados geográficos através de um Sistema de Informações Geográficas (SIG) e utilizará programação específica para a otimização da posição de instalação. Para interface com o usuário do sistema, criaremos uma aplicação Web responsiva que permita selecionar a região na qual se pretende instalar uma ERB e mostra as posições ideais para instalação.

Palavras-Chave – Estações Radiobase, Otimização, Sistema de Informações Geográficas, Aplicação Web.

ABSTRACT

This term paper intends to achieve a system capable of calculating the position to install cellular Base Stations (BS) in order to maximize the coverage network. For a given input region, the system will collect geographic data through a Geographical Information System (GIS) and utilize specific programming to optimize the placement position. For interfacing with the system user, we will develop a responsive Web application that allows the selection of a region on which we intended to place a BS, and show the ideal points for installation.

Keywords – Base Stations, Optimization, Geographical Information System, Web Application.

LISTA DE FIGURAS

1	Diagrama de Gantt.	23
2	Árvore de pré-requisitos do sistema.	24
3	Mapa de ERBs sem clusterização.	33
4	Mapa de ERBs com clusterização.	33

LISTA DE TABELAS

LISTA DE SÍMBOLOS

ERB: Estação Radiobase

MVP: Produto Viável Mínimo (do inglês *Minimum Viable Product*)

SIG: Sistema de Informações Geográficas

UX: Experiência de Usuário

TI: Tecnologia da Informação

TCC: Trabalho de Conclusão de Curso

CSV: Valores Separados por Vírgula (do inglês *Comma-Separated Values*)

UF: Unidade Federativa

CGI: Identificador Global de Célula (do inglês *Cell Global Identifier*)

MCC: Código de País Móvel (do inglês *Mobile Country Code*)

MNC: Código de Rede Móvel (do inglês *Mobile Network Code*)

SUMÁRIO

1	Introdução	12
1.1	Objetivo	12
1.1.1	Sistema de Informações Geográficas	13
1.1.2	Interface Web	13
1.2	Motivação	14
1.3	Justificativa	14
1.4	Organização do Trabalho	15
2	Aspectos Conceituais	17
2.1	Representação Geográfica	17
2.2	Algoritmos de Otimização	17
2.2.1	Método Taguchi	18
3	Tecnologias Utilizadas	20
3.1	Sistema Web	20
3.2	Bases de Dados	21
4	Metodologia do Trabalho	22
5	Especificação de Requisitos do Sistema	24
5.1	Casos de Uso	25
5.1.1	Atores	25
5.1.2	Requisitos funcionais	25
5.1.3	Requisitos não-funcionais	26
5.1.4	Descrição dos casos de uso	26

6	Projeto e Implementação	31
6.1	<i>Back-end</i>	31
6.1.1	Modelos de ERBs	31
6.1.2	Clusterização	32
6.1.3	Métodos Numéricos	36
6.1.4	Interface de Programação de Aplicações	36
6.1.5	Bancos de Dados Externos	37
6.1.6	Instalação de Servidor	37
6.2	<i>Front-end</i>	39
6.2.1	Projeto de Telas	39
6.2.2	Implementação com Vue.js e Vuetify	40
6.2.3	Integração com <i>Back-end</i>	41
7	Testes e Avaliação	42
7.1	Testes de Métodos Numéricos	42
7.2	Resultados	43
8	Considerações Finais	44
8.1	Conclusões do Projeto de Formatura	44
8.2	Contribuições	44
8.3	Perspectivas de Continuidade	44
8.3.1	Plano de Negócio	44
8.3.2	Dados	44
8.3.3	Descrição dos Casos de Uso Adicionais	44
	Referências	50

1 INTRODUÇÃO

Na revolução da informação em que vivemos hoje, onde cada vez mais pessoas estão conectadas à rede, o acesso à Internet tem se tornado cada vez mais essencial no dia-a-dia, até mesmo a populações fisicamente isoladas de regiões urbanas. Empresas bem conhecidas, como Vivo e Claro, vêm se empenhando para garantir melhor acesso a mais pessoas, mas se deparam com problemas de engenharia nesta tarefa.

A extensão territorial e a densidade demográfica desigual do Brasil são dois dentre vários fatores que tornam problemas de telecomunicação mais complexos. Há necessidade de se aumentar uma rede celular tanto em cobertura (para áreas com pouca densidade de antenas), quanto em capacidade (áreas com infraestrutura já existente, mas que não suporta a demanda local).

A dimensão deste problema gera um grande potencial de mercado para empresas terceirizadas, voltadas à instalação de Estações Radiobase (ERBs) para compartilhamento ou aluguel de células telefônicas às grandes empresas de telecomunicação. Dessa forma, há demanda do mercado por ferramentas que simplifiquem e/ou automatizem a tarefa de estudo de localização de ERBs.

1.1 Objetivo

O objetivo deste projeto de formatura é criar um produto mínimo viável (MVP) de um sistema que permita calcular posições para a instalação de antenas de telefonia de forma a maximizar o alcance delas.

Há duas necessidades para a realização deste projeto. A primeiro envolve coletar, apresentar e utilizar dados geográficos utilizados nos cálculos de instalação de novas antenas. Isso pode ser realizado por um sistema de informações geográficas.

A segunda necessidade seria que tal sistema tenha uma interface prática e responsiva para os usuários, de forma a ser utilizada tanto em computadores de escritório quanto em

campo pelo celular. Portanto, uma interface web que apresente os dados requisitados é ideal para o projeto.

A seguir, abordaremos com mais detalhes os dois módulos que serão utilizados para cumprir as necessidades do sistema.

1.1.1 Sistema de Informações Geográficas

Um SIG (Sistema de Informações Geográficas) é um sistema computacional capaz de obter, gravar, gerir, analisar e visualizar dados geográficos. Seu uso permite tomar decisões, analisar estatísticas e resolver problemas de otimização a partir de dados geográficos. O SIG é uma base que pode ser usada tanto no dia-a-dia para encontrar lojas próximas, quanto para profissionais rastrearem padrões de migração, monitorarem desmatamento ou planejamento urbano, por exemplo.

No nosso projeto, usaremos um framework de SIG para gravar e exibir a posição de ERBs (Estações Radiobase) atuais, o relevo e os consumidores atendidos na área da rede atual. Com essas informações, determinaremos as posições ótimas de ERBs de modo a maximizar a área de cobertura do sistema de telefonia. Para tanto, aplicaremos técnicas de programação linear, uma vez que estamos diante de um problema de otimização cuja função a ser otimizada é linear em relação às variáveis de entrada.

1.1.2 Interface Web

Para interação dos dados geográficos e de otimização com o usuário, desenvolveremos um *front-end* Web que permita selecionar a região na qual se pretende instalar alguma ERB. Esta interface se comunicará com o *back-end* do SIG, para obter e calcular os dados desejados.

O design deverá ser responsivo, podendo ser utilizado em plataformas mobile ou desktop, e simples, com controles com experiência de usuário (UX) intuitiva para verificar a posição ótima de instalação de antenas em determinada área escolhida pelo usuário. Para isso, a interface deverá exibir um mapa com as informações do SIG, que permita ao usuário selecionar uma área desejada. Os dados serão calculados no *back-end* e exibidos ao usuário na tela. Para isso, o *front-end* deverá obter dados e atualizar a tela dinamicamente.

1.2 Motivação

Com uma análise preliminar do setor, verificamos o mercado de instalação e aluguel de torres telefônicas no Brasil para comparar as tecnologias utilizadas em estudos de ERBs. Assim, pesquisamos serviços similares da concorrência. Um dos produtos encontrados, chamado Atoll, é um software de planejamento de células e posições de ERBs, similar ao que desejamos desenvolver, porém com funcionalidades estendidas como manutenção e melhoria de locais pré-estabelecidos, e parâmetros avançados de especificação das antenas, além de módulos para outras tecnologias de telecomunicação como Wi-Fi [1].

Porém, a ferramenta parece muito voltada à instalação urbana e análise de infraestrutura pré-existente, sem foco em uma eventual expansão. Por isso, vemos como que há necessidade do mercado por uma ferramenta voltada à ampliação de uma rede de ERBs, e que seja de fácil uso pelos instaladores de antenas do setor comercial. A utilização de um *front-end* web também contornará problemas relacionados à instalação de programas no computador, como dificuldades para usuários menos instruídos no uso de computadores, ou políticas de tecnologia da informação (TI) em empresas.

Um novo projeto na área de telecomunicações também pode ser de interesse ao público geral. Alguns casos de uso alternativos incluem a estimativa de posição do dispositivo pelas antenas encontradas, e a localização de antenas a partir do próprio celular do usuário. Eles permitiriam, respectivamente, uma geolocalização de baixa potência, ou que haja mais engajamento dos usuários para a melhoria de infraestrutura em sua região ou para avaliar alternativas dentre empresas de telecomunicação concorrentes.

1.3 Justificativa

Como discorrido anteriormente, falta uma solução voltada ao setor brasileiro de telecomunicações que seja automatizada e simples. Para verificar a necessidade da ferramenta a ser desenvolvida, pesquisamos sobre a situação atual para o mercado de antenas e eventuais *stakeholders* que seriam beneficiados por essa ferramenta.

Como explica o presidente da consultoria Teleco, há problemas com as leis atuais para instalação de novas antenas e, portanto, a adoção de novas tecnologias [2]. Com a chegada do 5G no Brasil, espera-se que haja maior disposição dos agentes de Estado por agilidade no processo de licitação de novas antenas, já que há necessidade dos usuários por um acesso à Internet melhor e mais veloz. Nesse sentido, a ferramenta pode acelerar

a instalação de novas antenas.

Sobre potenciais clientes, verificamos a existência de empresas no Brasil para localizar antenas, alugar terrenos para a instalação de antenas ou alugar antenas para empresas de telecomunicação. A maior parte destas empresas foca em um contexto urbano, enquanto que há interesse das empresas de telecomunicação e dos governos estaduais em uma expansão também para áreas rurais.

MyTower é um portal de locação e venda de imóveis para operadoras de telecomunicação [3]. Ele permite que o usuário cadastre seu imóvel e o anuncie para as operadoras após aprovação. O portal então faz a intermediação entre o anunciante e a operadora.

A Skysites é uma empresa que oferece soluções na área da infraestrutura de telecomunicação [4]. Ela gere um portfólio de sítios para instalação de equipamentos de telecomunicação (torres, *small cells*, *rooftops*), além de prover soluções customizadas para empresas de telecomunicação e compartilhar torres entre diferentes empresas. Outros serviços são redes para cobertura *indoor* e pequenas ERBs para melhorar a cobertura em ambiente urbano, as *small cells*.

Dessa forma, concluímos que o MVP é essencial para garantir um diferencial de mercado entre as várias empresas que crescem nesse setor, sejam tanto atores principais como as operadoras móveis, quanto terceiros. A automatização desses estudos de instalação pode cortar custos em análise de risco e permitir um processo mais ágil de projeto de novas antenas.

1.4 Organização do Trabalho

Para o projeto de formatura, organizaremos este Trabalho de Conclusão de Curso (TCC) em oito capítulos:

- No capítulo “**Introdução**” deste trabalho, definimos o problema, a motivação da realização deste sistema, e o que buscamos alcançar neste projeto.
- No capítulo “**Aspectos Conceituais**”, realizamos a contextualização dos conceitos de suporte para a execução do trabalho e a revisão da literatura de base.
- No capítulo “**Tecnologias Utilizadas**”, listamos as ferramentas, algoritmos e dados necessários para o desenvolvimento do sistema.

- No capítulo “**Metodologia do Trabalho**”, definimos os processos e fases no desenvolvimento de funcionalidades deste sistema, como concepção, estudo, projeto, implementação e teste.
- No capítulo “**Especificação de Requisitos do Sistema**”, definimos os requisitos para o desenvolvimento do sistema.
- No capítulo “**Projeto e Implementação**”, definimos os processos de implementação das tecnologias e requisitos levantados.
- No capítulo “**Testes e Avaliação**”, estão documentados os resultados do sistema através de testes feitos em diversos níveis.
- No capítulo “**Considerações Finais**”, relatamos a experiência de projeto de TCC e questões adicionais que restaram para uma avaliação futura.

2 ASPECTOS CONCEITUAIS

Com o MVP definido no capítulo anterior, iremos nos aprofundar na base teórica das tecnologias para a seleção de ponto ótimo de instalação de antenas. Dentre estes aspectos, discorreremos neste capítulo sobre a representação e modelagem dos dados geográficos em SIG e os algoritmos de otimização que podem ser utilizados.

2.1 Representação Geográfica

2.2 Algoritmos de Otimização

Em consulta à literatura pré-existente sobre o problema de otimização de instalação de ERBs, nos deparamos com várias abordagens distintas para o mesmo problema, em diferentes níveis de abstração. Há várias técnicas empregadas, desde programação não-linear a algoritmos evolutivos, algoritmos de polinização a programação inteira mista. Foi realizada uma comparação das tecnologias para verificar a que mais se adequa ao nosso caso de uso.

A princípio, nos voltamos a três alternativas: LEE et al. (2015) [5] utiliza conceitos básicos de telecomunicações, através de uma fórmula para calcular a satisfação dos usuários do sistema a partir de medidas de qualidade de banda, se baseando em um algoritmo evolutivo para otimizar a cobertura da rede.

Já KARULKAR & OH (2016) [6] se baseia em uma abordagem de limites geográficos impostos no processo de projeto de antenas, utilizando programação não-linear para identificar a posição ótima.

Por fim, WEGMANN et al. (2011) [7] utilizam o método Taguchi, descrito em detalhes a seguir.

2.2.1 Método Taguchi

O Método Taguchi é um método estatístico que tem como objetivo diminuir o número de experimentos necessários para se ajustar um conjunto de parâmetros e assim se obter um aumento de performance no sistema. No trabalho em questão, os parâmetros a serem otimizados são as coordenadas (latitude e longitude) de cada ERB a ser instalada. Como tais parâmetros são contínuos, criamos níveis associados, que correspondem (em cada parâmetro) a valores distribuídos uniformemente dentro do intervalo avaliado.

O primeiro passo do Método Taguchi é escolher um *array* ortogonal que possua o número de colunas iguais ao número de parâmetros analisados, e possua a quantidade de níveis adequada. Um *array* ortogonal é uma tabela, na qual existe um número t (chamado de força do *array* ortogonal), o qual para qualquer seleção de t colunas na tabela, todas as tuplas aparecem o mesmo número de vezes.

Em seguida, mapeamos cada nível a um valor do parâmetro. Sejam min_t e max_t os valores máximo e mínimo do intervalo em estudo de um parâmetro x_t , e s o número de níveis escolhido.

$$V_t^{(m)} = \frac{min_t + max_t}{2}$$

Definimos:

$$\beta_t^{(m)} = \frac{max_t - min_t}{s + 1}$$

Na m -ésima iteração, a função de mapeamento é a seguinte:

$$f_t^m(l) = \begin{cases} V_t^{(m)} - (s/2 - l) \cdot \beta_t^{(m)}, & 1 \leq l \leq [s/2] - 1 \\ V_t^{(m)}, & l = [s/2] \\ V_t^{(m)} + (l - s/2) \cdot \beta_t^{(m)}, & [s/2] + 1 \leq l \leq s \end{cases}$$

Onde s é a quantidade de níveis escolhida e l é o número do nível.

Depois disso, mapeamos cada valor da imagem da função objetivo para a razão sinal-ruído. A fórmula é:

$$SN_i = 10 \cdot \log_{10}(y_i^2) [dB]$$

Depois, calculamos o valor médio de SN para cada parâmetro para cada nível. Para cada parâmetro x_t , o nível que tiver maior SN médio, é denotado $V_t^{(best, m)}$

Em seguida, verificamos o critério de parada. Se ele não estiver satisfeito, os melhores valores de cada parâmetro são usados como valores centrais para a próxima iteração.

Além disso, o valor de β_t é reduzido de um fator $\epsilon < 1$.

$$\beta_t^{(m+1)} = \epsilon \beta_t^{(m)}$$

3 TECNOLOGIAS UTILIZADAS

Anteriormente, abordamos os dois componentes principais para compor nosso sistema: uma interface Web e um *back-end* SIG. Além dessas ferramentas, utilizaremos também dados específicos para o SIG na implementação de um dos algoritmos apresentados em “**Aspectos Conceituais**”. Neste capítulo, discutimos as decisões tomadas pelo grupo acerca de *frameworks* e bases de dados utilizadas no trabalho. Elas serão utilizadas no trabalho descrito pelo próximo capítulo.

3.1 Sistema Web

No projeto, utilizaremos o *framework* Django, escrito em Python. A principal razão para isso é que os integrantes já possuem familiaridade com ambos Python e Django, portanto facilitando o desenvolvimento pela transferência de conhecimentos prévios. Além disso, o *framework* possui prototipagem relativamente simples para orientação a objetos, autenticação por padrão e fácil integração com banco de dados. A integração de *front-end* com o *back-end* deverá ser feita com o módulo de API REST do Django.

O Django possui funcionalidades de SIG pelo módulo GeoDjango, que utiliza como banco de dados o PostGIS (baseado em PostgreSQL). Serão armazenados dados públicos de localização de ERBs, relevo e densidade populacional. Ele será também responsável pelos cálculos realizados para a localização de novas antenas no mapa.

Para interação com o usuário por um mapa interativo, utilizaremos a biblioteca Leaflet, escrita em JavaScript. Ela se comunicará aos dados pela API REST a ser desenvolvida, tanto para requisições quanto para exibições.

Optamos também pelo *framework* JavaScript conhecido por Vue para realizar o controle *single-page app* do nosso sistema, e a interface gráfica Vuetify para elaborar as telas do nosso sistema.

3.2 Bases de Dados

Todos os dados usados para o trabalho vieram de fontes abertas, e serão utilizados no SIG para exibição em telas ou cálculos de otimização.

Primeiramente, utilizamos o Mapa de ERBs Brasil presente no portal Telebrasil [8]. Essa base contém uma lista de ERBs do Brasil de novembro de 2017, com informação de operadora, endereço, e posição geográfica de cada ERB. Essas informações são essenciais para o cálculo da posição ótima da ERB para maximizar a cobertura da célula de determinada operadora.

Também utilizamos a base do OpenCellID, da empresa Unwired Labs [9]. Essa base contém uma lista de ERBs do mundo inteiro, com o CGI de cada ERB. Os dados foram obtidos através da colaboração de usuários do aplicativo LocationAPI da Unwired Labs. O LocationAPI trata-se de um serviço de geolocalização que não depende de GPS. Dessa forma, com a base da OpenCellID, podemos estimar a posição de um celular a partir das ERBs às quais ele está conectado.

Outra base de dados em estudo foi o Google Earth Engine [10], uma API específica para dados geográficos públicos do Google, como relevo e densidade populacional. Devido à extensão destes dados, e à impraticidade de armazenamento em banco próprio, será realizada dependência desta base através de sua API, que possui implementação em Python. Esta implementação será discutida adiante.

4 METODOLOGIA DO TRABALHO

Neste capítulo, discorreremos sobre como foi feito o desenvolvimento do MVP apresentado na introdução.

Em uma primeira etapa, definimos com o orientador a proposta e o escopo deste projeto, propondo a pesquisa a ser realizada tanto da perspectiva de implementações do algoritmo de otimização quanto de requisitos necessários. Levantados tais requisitos no capítulo **“Especificação de Requisitos do Sistema”**, a próxima etapa se baseou em projetar a realização de cada um destes requisitos de acordo com as prioridades definidas, isto é, iniciar um processo de decisões definitivas para o andamento do trabalho.

Tomadas estas decisões, finalmente, realizou-se a prototipagem das nossas funcionalidades, dividida entre os dois integrantes do grupo de forma paralela para permitir o andamento simultâneo de diferentes partes do sistema, com reuniões frequentes entre os integrantes e o orientador para avaliar o progresso. Após certa familiaridade com os desafios colocados, iniciou-se o desenvolvimento definitivo do sistema na especificação final, etapa que levou mais tempo neste projeto.

Junto com o desenvolvimento, foi realizado o *deployment* do servidor de produção, onde os testes e as demonstrações podem ser feitos, de forma a verificar as condições reais de implementação do sistema. Ele foi atualizado a cada nova funcionalidade concluída.

Por fim, antes de realizar a apresentação do trabalho final, foi realizada uma bateria de testes para verificar o funcionamento correto do sistema após o desenvolvimento, sem a elaboração de novas funcionalidades, garantindo que continuamos dentro das nossas previsões de projeto.

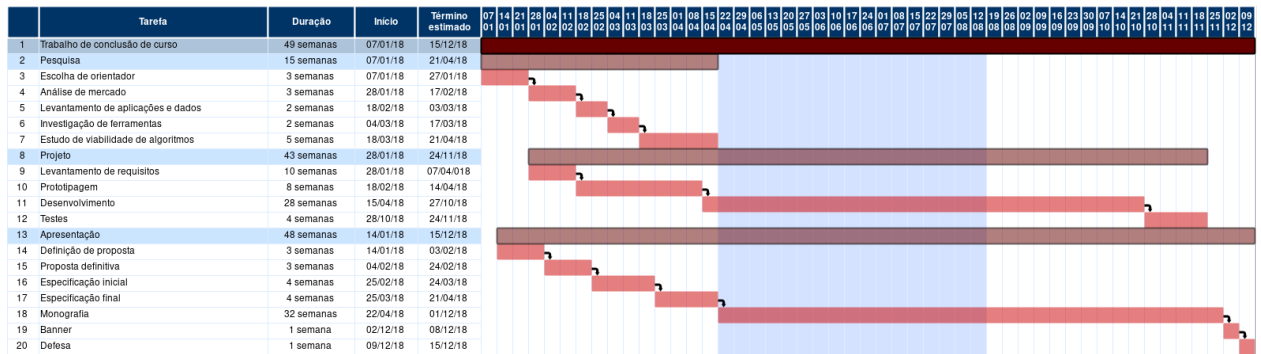


Figura 1: Diagrama de Gantt.

Desta forma, nos organizamos para realizar a implementação do sistema, separando as tarefas a serem realizadas nas categorias: pesquisa, projeto, e apresentação. Utilizamos o cronograma oficial da disciplina de TCC para elaborar um diagrama de Gantt na Figura 1 com a sequência de tarefas a serem divididas pelo grupo ao longo do ano para cada semana. Indicamos também o período extra-escolar em fundo azul, quando haverá desenvolvimento do sistema de forma mais lenta e sem cronograma específico.

5 ESPECIFICAÇÃO DE REQUISITOS DO SISTEMA

Tomando-se como base as tecnologias anteriormente citadas, definimos com mais detalhe as necessidades do sistema para priorizar o trabalho descrito no capítulo de “Metodologia do Trabalho”.

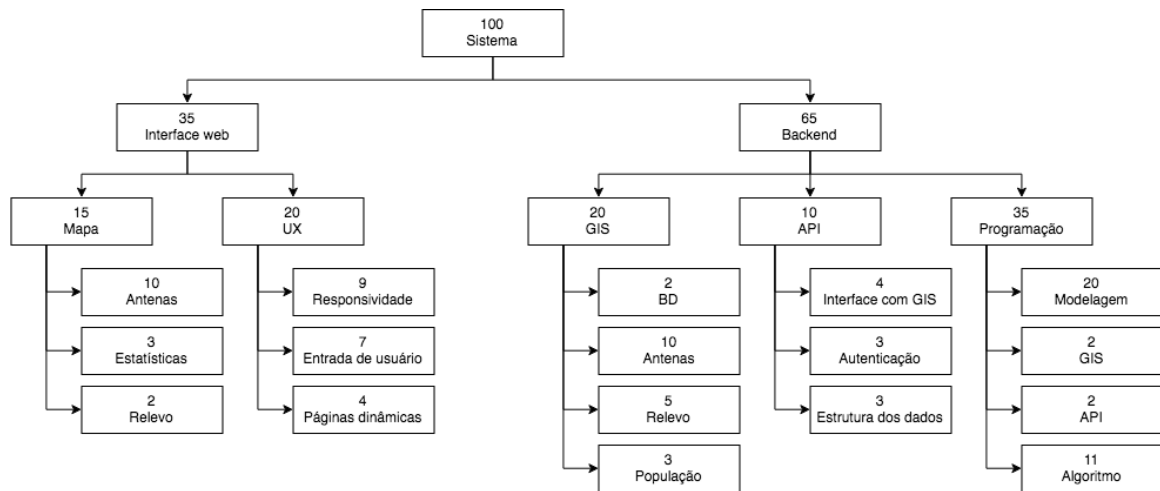


Figura 2: Árvore de pré-requisitos do sistema.

Para definir os requisitos do nosso sistema, foi elaborada uma árvore de pré-requisitos, listando a prioridade total dada a cada componente do sistema final na Figura 2. A divisão do trabalho será feita nos componentes de *front-end* e *back-end*, como discorrido anteriormente.

Como evidenciado pela figura 2, a ênfase deste projeto será no *back-end*; em especial, na parte de modelagem e programação relacionadas ao cálculo de otimização da posição de antenas. As outras duas partes relevantes do *back-end* tratam, respectivamente, da instalação e uso do banco de dados do SIG, e da comunicação externa de dados via API.

Embora tenha uma ênfase menor, o *front-end* da interface web também será um requisito fundamental de projeto, separado na experiência do usuário e na visualização do mapa.

Definido o escopo, começamos a pensar, junto com nosso orientador, sobre os requisitos do projeto. A estratégia utilizada foi o *brainstorming*, leitura de obras de referência na literatura de otimização de antenas, e pesquisa de softwares com escopo parecido. A ideia inicial era ter como ator principal do sistema o funcionário cadastrado de uma empresa de instalação de antenas, porém depois consideramos que seria complementar ao projeto considerar um usuário não-comercial, assim como um usuário administrador. Nessa fase, perguntamo-nos como o nosso projeto poderia ajudar esses atores, qual informação ele deveria produzir. Esse processo nos levou à modelagem de diversos casos de uso de interesse do sistema. Alguns foram deixados de lado por fugirem ao escopo do MVP, e estão listados na seção “**Perspectivas de Continuidade**” do último capítulo.

5.1 Casos de Uso

5.1.1 Atores

- Administrador
- Funcionário cadastrado de Empresa de instalação de antenas
- Usuário não-comercial

5.1.2 Requisitos funcionais

- Efetuar *login*
- Efetuar *logout*
- Cadastrar usuário no sistema
- Exibir mapa com ERBs
- Exibir mapa com cobertura celular estimada
- Exibir localização ideal para instalação de uma nova ERB
- Adicionar nova ERB
- Fazer *login* por OAuth2
- Acessar API
- Exibir ERBs às quais o celular do usuário está conectado

- Exibir mapa com qualidade do sinal por operadora
- Estimar geolocalização do usuário
- Listar usuários

5.1.3 Requisitos não-funcionais

- Funcionalidades e código bem documentados
- Interface acessível e simples
- Ser responsivo
- Ser dinâmico
- Ser rápido
- Ser seguro

5.1.4 Descrição dos casos de uso

Caso de Uso 1: Efetuar *login* no sistema.

Descrição: Este caso de uso descreve o processo de autenticação no sistema.

Evento iniciador: Usuário informa seu nome de usuário e senha.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário na página de login.

Sequência de Eventos:

1. Usuário informa seu nome de usuário e sua senha.
2. Sistema autentica o usuário e senha.
3. Sistema exibe página inicial com opções correspondentes ao nível de privilégio do usuário.

Pós-condições: Usuário logado no sistema.

Extensões:

1. Usuário ou senha estão incorretos: Sistema exibe mensagem de erro (passo 2).

2. Login por OAuth2: Sistema recebe token de autenticação externa para o acesso (passo 1).

Inclusões: --

Caso de Uso 2: Efetuar *logout* no sistema.

Descrição: Este caso de uso descreve o processo de *logout* no sistema.

Evento iniciador: Usuário clica no botão de *logout*.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário logado no sistema.

Sequência de Eventos:

1. Usuário clica no botão de *logout*.
2. Sistema exibe página inicial para usuários não-logados.

Pós-condições: Usuário deslogado do sistema.

Extensões: --

Inclusões: --

Caso de Uso 3: Exibir mapa

Descrição: Este caso de uso descreve o processo de exibição de um mapa centrado na localização do usuário.

Evento iniciador: Usuário requisita exibição de mapa.

Pré-condições: Usuário logado no sistema.

Sequência de Eventos:

1. Usuário requisita exibição de mapa.
2. Sistema pede que o usuário permita acessar sua geolocalização.
3. Usuário permite que sistema acesse sua geolocalização.
4. Sistema mostra um mapa centrado no usuário.

Pós-condições: Mapa com ERBs apresentado.

Extensões: Usuário não permite: Sistema mostra mapa centrado numa localização padrão. (passo 4)

Inclusões: Busca de mapa no OpenStreetMap (passo 4)

Caso de Uso 4: Exibir mapa com ERBs.

Descrição: Este caso de uso descreve a exibição das ERBs em um mapa.

Evento iniciador: Usuário clica na opção “Mapa de ERBs”.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema.

Sequência de Eventos:

1. Usuário clica na opção “Mapa de ERBs”.
2. Sistema exibe mapa com ERBs da região.

Pós-condições: Mapa com ERBs apresentado.

Extensões: --

Inclusões: Caso de uso 3 “Exibir mapa” (passo 2)

Caso de Uso 5: Exibir mapa centrado em localização dada

Descrição: Este caso de uso descreve a exibição do mapa com ERBs centrado em uma localização dada pelo usuário.

Evento iniciador: Usuário insere a latitude e longitude e clica em “Buscar”.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema e na página com mapa de ERBs.

Sequência de Eventos:

1. Usuário insere a latitude e longitude e clica em “Buscar”.
2. Sistema mostra o mapa de ERBs centrado na localização dada.

Pós-condições: Mapa com ERBs centrado na localização dada apresentado.

Extensões: Latitude ou longitude inválida: Sistema mostra uma mensagem de erro (passo 2)

Inclusões:

1. Busca de mapa no OpenStreetMap (passo 2)
2. Caso de uso 4 “Exibir mapa com ERBs” (pré-condição)

Caso de Uso 6: Exibir ERBs no mapa por operadora.

Descrição: Este caso de uso descreve a exibição do mapa com ERBs para uma operadora definida pelo usuário.

Evento iniciador: Usuário seleciona uma operadora.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema e na página com mapa de ERBs.

Sequência de Eventos:

1. Usuário seleciona uma operadora.
2. Sistema mostra o mapa de ERBs da operadora dada pelo usuário.

Pós-condições: Mapa com ERBs da operadora escolhida.

Extensões: --

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (pré-condição)

Caso de Uso 7: Encontrar local de instalação de ERBs. **Descrição:** Este caso de uso descreve a exibição de posições otimizadas

para instalação de ERBs. **Evento iniciador:** Usuário clica na opção “Calcular posição para instalação”.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema e na página “Mapa de ERBs”.

Sequência de Eventos:

1. Usuário clica na opção “Calcular posição para instalação”.
2. Sistema exibe um formulário com campos: “Quantidade de ERBs a serem instaladas” e “Parâmetros a serem considerados no cálculo”.
3. Usuário preenche o formulário e clica em avançar.
4. Sistema solicita para o usuário selecionar a região de interesse.
5. Usuário seleciona a região de interesse.
6. Sistema informa que operação pode demorar alguns momentos.
7. Sistema indica os locais calculados no mapa.
8. Sistema mostra número de usuários atendidos por cada nova antena.

Pós-condições: Mapa com posições otimizadas apresentadas.

Extensões: Operação demora muito tempo: Sistema informa que não foi possível calcular a posição (passo 7)

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (pré-condição).

6 PROJETO E IMPLEMENTAÇÃO

Definidos os requisitos e as tecnologias a serem utilizadas, e os conceitos abordados em cada algoritmo, iniciaram-se as implementações conforme o capítulo “**Metodologia do Trabalho**”. As partes relevantes do sistema e as soluções encontradas estão mencionadas abaixo.

6.1 *Back-end*

A utilização do *back-end* no sistema foi, no geral, para a obtenção de dados existentes em banco e a computação de novas posições ótimas para as antenas. Apesar de não estar diretamente visível ao usuário, esta parte do sistema se mostrou também com dificuldades intrínsecas que nós precisamos enfrentar. Nas seções a seguir, detalharemos as medidas tomadas em cada etapa.

6.1.1 Modelos de ERBs

Anteriormente, comentamos sobre a criação de dois modelos de ERBs, cada um proveniente de uma dentre duas bases de dados. Uma base delas utiliza dados oficiais da Anatel para indicar as antenas oficialmente registradas, portanto que pertencem a uma única operadora legalmente [8]. Estas antenas foram denominadas “Antenas Apropriadas” (“*Owned Base Stations*”), pois indicam as operadoras às quais pertencem.

A outra base apresenta antenas distinguíveis por um identificador global, portanto acaba exibindo uma única antena compartilhada como várias entradas do banco de dados para cada respectiva operadora [9]. Como possuem um identificador de padrão global, estes dados foram chamados de “Antenas Identificadas” (“*Identified Base Stations*”) no código. Tanto os dados dessa base quanto da anterior são representados como arquivos *Comma-Separated Values* (CSV) em suas origens.

Para cada modelo, criamos um *script* que parseia os dados de entrada e popula as

entradas no banco de dados. Estes *scripts* utilizaram o suporte nativo do Django a utilitários da linha de comando para facilitar o uso.

Os dados da Anatel foram extraídos e desestruturados em latitude e longitude, operadora, Unidade Federativa (UF), município e endereço. Também foram criados modelos adicionais para agregar antenas por operadora e por UF.

Para a base de dados aberta, foram utilizados como parâmetros a latitude e a longitude, os campos do *Cell Global Identity* (CGI), que identifica cada antena, o tipo de rádio, e o sinal médio detectado. Como esta base possui dados globais de antenas, filtramos apenas para os do Brasil; para isso, utilizamos um dos campos do CGI, o *Mobile Country Code* (MCC) – que é único por país –, como filtro das linhas do arquivo CSV. O MCC do Brasil é 724 [11].

Com estes dois scripts, conseguimos importar todos os dados relevantes destes arquivos para o PostGIS. Porém, em etapas posteriores do projeto, foi necessário escolher entre um modelo principal para implementar a solução. Para chegar a um consenso, conversamos dentro do grupo, com o orientador e com Milton Kaoru Kashiwakura, Diretor de Projetos Especiais e Desenvolvimento do NIC.br.

Devido à proeminência do compartilhamento de antenas entre empresas de telecomunicação, optamos por utilizar a segunda base nas etapas subsequentes do projeto como análise mais realista da cobertura celular no Brasil. Como não há uma correlação direta das antenas com cada operadora ao contrário da outra base, utilizou-se um campo específico do CGI, denominado *Mobile Network Code* (MNC) – que especifica uma única operadora dentro de um país [11] –, para verificar a qual empresa pertence cada antena nas etapas subsequentes do trabalho. Assim, foi criado um modelo no banco de dados específico para guardar os valores de MNC e respectivas operadoras, pois será útil para filtrar estes dados.

6.1.2 Clusterização

Um dos principais desafios na implementação foi realizar a exibição de múltiplas antenas como pontos em um mapa. Como muitas ERBs ficam próximas das outras e, portanto, difíceis de serem visualizadas separadamente, utilizou-se um *plugin* próprio da biblioteca de mapa para a clusterização de pontos, facilitando a identificação da quantidade de antenas em uma região, com exibição variável com o zoom.

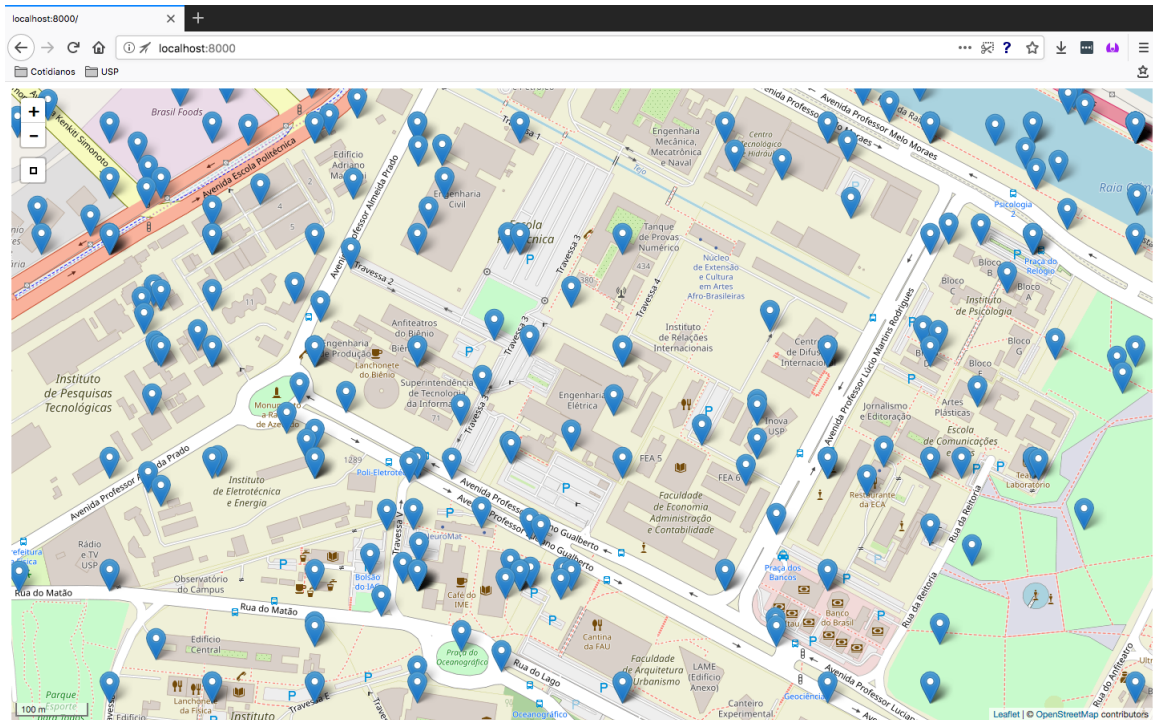


Figura 3: Mapa de ERBs sem clusterização.

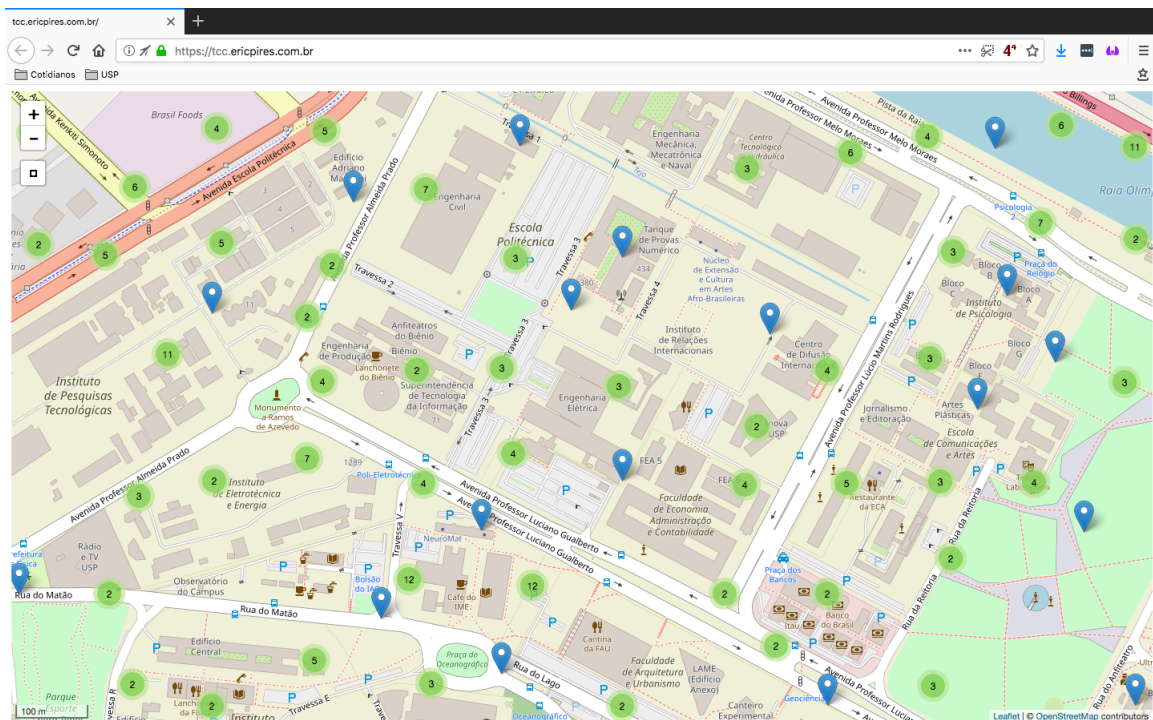


Figura 4: Mapa de ERBs com clusterização.

Com o uso desse plugin, a exibição de até cerca de 500 antenas na tela tornou-se possível. Porém, mais antenas do que isso causavam vários problemas relacionados a

uma grande quantidade de dados, como demora na transmissão por HTTP, requisições intensivas ao servidor, e problemas de memória no navegador do cliente. Algumas medidas foram tomadas tanto no *back-end* quanto no *front-end* para tentar amenizar estes efeitos, como otimizar scripts mas sem gerar vantagens muito perceptíveis.

Dessa forma, resolveu-se utilizar algoritmos de clusterização no *back-end*, e então realizar a integração com o *front-end*. A primeira tentativa envolveu a utilização de uma biblioteca *open-source* específica para o GeoDjango, chamada “**Anycluster**” [12]. Embora tenha sido feita para uma configuração de servidor similar à utilizada no projeto, a instalação e documentação se mostraram confusas e incompletas. E mesmo com tentativas de ajuste manual no código, incluindo refatoração das partes não-essenciais da aplicação, não foi possível integrar este código à nossa base. Além disso, a análise do código indica que não há nenhuma forma de cache dos dados de clusterização, o que não resultaria em velocidades adequadas para uma aplicação de visualização em mapa.

Procurou-se então uma segunda alternativa de clusterização, com implementação própria para este projeto. A solução envolve utilizar o *tiling* de grade que o OpenStreetMap utiliza para carregar as imagens. A clusterização foi feita com fórmulas do PostGIS específicas para clusterização de pontos. Para o cache dos *clusters*, utilizou o cache padrão do Django. Os resultados gerados pelo programa eram corretos, mas o uso alto de memória e o tempo para clusterizar antes de cache inviabilizam o uso no servidor. Foi realizado um aumento na capacidade da máquina do host, mas não houve ganhos consideráveis. Além disso, os dados de *cluster* ficam com aparência estranha, onde a separação em grids fica muito perceptível por não ser natural.

A terceira tentativa utilizou a mesma ideia anterior, porém para um dado nível de zoom, seriam utilizados os *clusters* para um zoom mais próximo do mapa, portanto usando 4 vezes mais dados. Ainda houve alguns *clusters* estranhos na visualização em mapa, embora consideravelmente melhores no geral; mas combinado com o uso de dados muito alto tanto em processamento quanto em rede, uma utilização com muitos clientes se tornaria inviável.

Em seguida, na quarta tentativa, optamos por salvar os *clusters* em banco de dados, onde são indexados por zoom como nas tentativas anteriores. Para acelerar o algoritmo de cálculo de *clusters*, utiliza-se os *clusters* de um zoom menor para gerar os de zooms maiores. Foram feitas algumas iterações com diferentes funções de clusterização do PostGIS para obter resultados melhores. A utilização de modelos ao invés de cache resultou em uma fácil integração com o *front-end* já existente, dada a facilidade do Django para

lidar com modelos na API. Porém, os resultados obtidos foram inesperados, com clusters baseados na distância ponto-a-ponto ao invés de ao redor de um centro de massa. Dessa forma, lugares com muitas antenas lado-a-lado, em uma área com alta densidade de pontos como a cidade de São Paulo, aparecem como um único *cluster*, independentemente do zoom, e às vezes muito longe do centro, deixando muitas áreas vazias.

A quinta tentativa se baseou nas demais, e também no conceito de *Geohash* (anotação de cada região em um mapa com caracteres alfanuméricos, dividindo cada região sucessiva do globo em 16 partes). Foi selecionada uma precisão (i.e. um número de caracteres do *geohash*) diferente para cada zoom, seguindo-se como referência a biblioteca “*django-geohash-cluster*” [13], sendo que a precisão máxima é de 7 caracteres, equivalente a uma precisão de aproximadamente 76 metros no Equador, o que é o suficiente para níveis de zoom. Além disso, o código do *front-end* foi modificado especificamente para fins de performance, assim obtendo clusterizações com visualização mais natural. Graças ao suporte do GeoDjango para inserir o geohash em queries, foi possível realizar a implementação em um algoritmo simples, sem escrever as queries manualmente. A principal desvantagem deste algoritmo foi a extrema demora para realizar as clusterizações em comparação com os demais algoritmos; porém, o resultado final com a integração de clusterização *front-end* e *back-end* foi melhor em desempenho. Outra desvantagem foi que, como não há correspondência direta entre níveis de zoom do mapa e precisões de *geohash*, determinados níveis de zoom demoram mais tempo para ler os dados. Além disso, há algumas pequenas falhas visuais perto das bordas, que foram corrigidas posteriormente. Por fim, foram realizadas alterações na API para tornar a transição entre *clusters* e ERBs na exibição do mapa opaca para o usuário final – ou seja, o *script* de exibição do mapa do navegador enxerga um único *endpoint* e não diferencia quando o servidor opta por enviar dados de antenas ao invés de *clusters* para zooms menores.

Este algoritmo foi expandido para também permitir a filtragem por operadora de celular. Esta filtragem se baseia no campo MNC do identificador de cada estação radiobase, que distingue a empresa responsável. Para filtros mais precisos, como tipo de tecnologia celular, não será utilizada clusterização, com carregamento e exibição diretamente no mapa – já que inclui um número menor de antenas a serem exibidas, e varia com o caso de uso de cada usuário, portanto sendo um processo menos repetitivo para o servidor que não se beneficia de mecanismo de cache.

Vale notar que, para a implementação de filtragem por operadora, os *clusters* de todas as antenas utilizam um campo nulo para representar a operadora. Isso é importante porque, ao exibir os *clusters* sem filtro específico de operadora, na realidade, deverá ser

feito filtro por operadora igual ao valor nulo. Esta distinção terá maior importância quando o cliente requisitar dados de clusters pela API, como descrito mais adiante.

6.1.3 Métodos Numéricos

Em um primeiro momento, utilizamos três métodos para comparação entre algoritmos: Taguchi, Basinhopping e SQSQP. A função de otimização foi desacoplada da implementação do método numérico. Desse modo, podemos combinar quaisquer métodos numéricos implementados com quaisquer funções de otimização.

Para os métodos Basinhopping e SQSQP, usamos a implementação disponível na biblioteca “`scipy`” [14].

Já o método Taguchi foi implementado de acordo com a especificação especificada no capítulo “**Aspectos Conceituais**”. Os arrays ortogonais usados pelo método foram implementados como uma tabela, uma vez que seu cálculo é uma operação complexa, e os casos de uso preveem o uso de poucos parâmetros no método Taguchi. Usamos a biblioteca “`numpy`” [15] para as operações com *array*, uma vez que ela provê operações eficientes nessas estruturas de dados.

6.1.4 Interface de Programação de Aplicações

A integração do *back-end* com o *front-end* foi feita feita por uma API HTTP, que abstrai os dados e as funcionalidades necessários para a realização dos casos de uso. Resumidamente, ela deverá ser responsável por três tarefas específicas:

- Realizar *login* de usuários no sistema.
- Exibir dados de mapas na tela.
- Gerar otimização de posição de antena.

Desta forma, foram expostos *endpoints* para cada uma das tarefas. A utilização da biblioteca Django REST Framework facilitou o processo de integração da API ao *back-end* desenvolvido. O cliente de *front-end* – no caso, o Vue – poderá descobrir estes *endpoints* a partir de um outro *endpoint* específico que expõe estas informações, de forma a tornar refatorações do *back-end* dinâmicas.

Algumas alterações tiveram que ser feitas tanto na configuração do Django quanto na criação de *endpoints* de API para a realização esperada dos casos de uso. A primeira se

refere ao *login* de usuários, pois necessitou de reconfiguração da autenticação do Django. Esta autenticação foi modificada para permitir que, por um *endpoint* específico de *login*, um usuário realize uma requisição POST com seu nome de usuário e senha e receba um *token*, que pode ser utilizado nas requisições autenticadas subsequentes. Como a autenticação padrão do Django não utiliza *tokens*, mas sim parâmetros de sessão específicos, alteramos esta configuração.

Outra modificação se refere à exibição de dados no mapa. Devido ao esquema de clusterização por *geohash* descrito anteriormente, optou-se por uma abordagem opaca na API, aonde o usuário deverá indicar apenas as coordenadas da tela, o tamanho de zoom atual, e o filtro de operadora. Dessa forma, foi descrita lógica adicional neste *endpoint* para converter o tamanho de zoom à precisão de geohash correspondente. Além disso, o filtro de operadora foi complementado com o caso geral de não filtrar por operadora, pois a modelagem dos *clusters* no banco de dados requer um filtro específico para *clusters* sem filtro de operadora.

6.1.5 Bancos de Dados Externos

Alguns dados adicionais mencionados anteriormente, que não serão persistidos no banco de dados, são referentes a população e relevo. Ao longo de nossa pesquisa, nos foi indicado o uso da Google Earth Engine [10], que é uma plataforma de dados geoespaciais disponível publicamente para ferramentas de análise científica e pesquisa. Dentro desta plataforma, dois *datasets* estudados para uso no sistema são “USGS/SRTMGL1_003” para topografia, e “WorldPop/POP” para densidade populacional.

O Google Earth Engine possui uma API oficial em Python, que foi estudada e implementada no *back-end*. A implementação criada permite persistir dados no cache do servidor, para evitar que valores frequentemente utilizados nos algoritmos de otimização (i.e. uma única área analisada repetidamente) sejam baixadas novamente a cada execução.

6.1.6 Instalação de Servidor

O *back-end* do sistema foi instalado em um servidor remoto para apresentações e testes. Utilizou-se o serviço DigitalOcean para a virtualização de servidor, onde subiu-se a instância por SSH em uma distribuição Ubuntu. As chaves SSH foram distribuídas aos integrantes do grupo e ao orientador para permitir o acesso. Nesta máquina, instalamos o repositório em Python, o banco de dados PostgreSQL, e as aplicações adicionais ne-

cessárias para o GeoDjango/PostGIS. Os passos tomados na configuração a seguir foram, fora de ordem:

- Criar e ativar um ambiente virtual do Python (`virtualenv`).
- Instalar dependências em Python do projeto.
- Obter a extensão do PostGIS, e as bibliotecas GEOS, PROJ.4, GDAL, GeoIP2.
- Criar um *schema* no banco de dados para a aplicação.
- Carregar dados de antenas apropriadas e identificadas.
- Gerar diretório de arquivos estáticos do Django.
- Obter arquivos de localização via IP da biblioteca GeoIP2.
- Registrar-se no Google Earth Engine, gerar *token* de autenticação, e salvar no ambiente de usuário.
- Criar usuários administradores para os membros do grupo e para demonstração.
- Habilitar cache em disco por banco de dados.

Estes passos foram reproduzidos em um **README** do repositório, para facilitar a criação de novos ambientes de desenvolvimento.

Ainda neste servidor virtual, foram instalados o Nginx e o Gunicorn, para o *proxy* e o acesso HTTP, respectivamente. Também foi utilizado um nome de domínio para o servidor, que foi configurado no Nginx. Além disso, por questão de segurança, também configurou-se o acesso HTTPS ao servidor, através do Let's Encrypt e sua configuração para *proxy*.

Por fim, para facilitar o deployment da aplicação para testes, configurou-se o repositório local do Git como um repositório remoto, permitindo que seja atualizado por SSH com um único comando; e as regras de *pull* do repositório foram modificadas para aplicar novas alterações no código instantaneamente – como migração de banco de dados, coleta de arquivos estáticos, novos *endpoints*, dentre outros.

Uma medida tomada por segurança e por limitações técnicas na instalação do *back-end* foi a utilização de ambientes de desenvolvimento – isto é, a definição manual de variáveis do Django, sensíveis e/ou específicas por máquina. Para isto, foi criado um novo arquivo,

`config.settings.py`, que é importado pelo arquivo de configurações padrão do Django (`settings.py`) e é ignorado pela ferramenta Git. Dentre as variáveis deste arquivo, estão:

- Uma *flag* de debug, utilizada pelo Django para exibir informações sensíveis de erro quando habilitada.
- Uma chave secreta, utilizada para gerar *hashes* de segurança.
- Lista de *hostnames* aceitos para conexões de entrada, que foi definida para os IPs da máquina e o nome de domínio.
- A conexão ao banco de dados, com usuário, senha e *schema*.
- Dados de acesso a autenticação social em serviços como GitHub e Facebook.

Alguns dados e parâmetros acima foram colocados como opcionais, para facilitar a instalação de ambientes de desenvolvimento. Por exemplo, a biblioteca GeoIP2, que tenta estimar a localização do usuário pelo IP, foi colocada no código como dependência opcional, devido ao tamanho dos arquivos consumidos e à falta de necessidade de testes para uso. Os dados de acesso para serviços sociais também foram tornados opcionais por razões semelhantes.

6.2 *Front-end*

Para elaborar o *front-end*, usamos o *framework* Vue.js. O front-end da nossa aplicação foi responsável pela parte visual, pela comunicação do cliente com a API em Django, pela exibição dos dados no mapa, e pelo recebimento das entradas do usuário. Nas seções a seguir, detalharemos como planejamos e implementamos o cada uma dessas partes.

6.2.1 Projeto de Telas

O planejamento do *front-end* começou com o desenho das telas. Pensamos em telas que implementassem os casos de uso, além de serem fáceis de usar tanto em *desktop* como em *smartphones*. Criamos um diagrama de telas para representar a sequência em que o usuário acessará as telas.

A primeira tela projetada foi a tela de mapas. Inicialmente, ela teria o mapa com os clusters, e um menu lateral com opções de filtro por operadora e tecnologia, opções de

remoção, adição e exportação de ERBs e um botão para executar o algoritmo de posição ideal a partir de parâmetros entrados pelo usuário. Criamos os desenhos com papel e lápis, visto que tratam-se de um projeto de baixa fidelidade.

Depois de algumas iterações, removemos algumas funcionalidades e adicionamos outros filtros.

Outra tela projetada foi a tela inicial, com imagens do sistema funcionando, textos explicativos e um formulário de contato. A ideia dessa tela é fornecer uma ideia inicial das *features* do nosso projeto.

Pensamos também na tela de *login*, na qual o usuário entra com seu nome de usuário e senha. Essa seria a tela à qual ele seria redirecionado caso não estivesse logado.

6.2.2 Implementação com Vue.js e Vuetify

Usamos o *framework* Vue.js no front-end do nosso projeto. Ele é um dos *frameworks* JavaScript mais populares da atualidade. Seu uso aumenta muito a produtividade e organização do código, porque ele divide o sistema em componentes pequenos e reusáveis. Outro benefício é o *single page application* que o Vue.js viabiliza. Com o *single page application*, quando o usuário muda de tela, a página não é recarregada, apenas os componentes diferentes que são trocados. Dessa forma, os componentes comuns a várias telas (como *header* e *footer*) nunca são recarregados. Isso propicia uma experiência mais fluida para o usuário.

Para tratar do visual do sistema, usamos a biblioteca Vuetify. Ela provê componentes com visual característico do Material Design para o Vue.js. O Material Design é uma linguagem visual criada pela Google que oferece um visual intuitivo que se aproxima do design do mundo real com tinta e papel, trabalhando com camadas e sombras que dão uma sensação tátil para o usuário. O uso do Vuetify facilitou muito o desenvolvimento, pois não precisamos nos preocupar muito com o estilo de cada elemento HTML.

Com relação às telas criadas, a tela de mapas usa um menu lateral, com opções de filtro, um botão para executar o algoritmo de otimização, e um botão para exibir mapa de calor. As opções de menu são seleções retangulares, por círculo, tecnologia e cidades. Os filtros não implementados foram desabilitados. Com isso, a tela final ficou pouco diferente da planejada na etapa de projeto de telas.

Ao escolher a opção de menu de seleção, podemos selecionar um retângulo ou circunferência no mapa.

A tela de *login* foi um formulário simples com um campo de usuário e senha.

6.2.3 Integração com *Back-end*

O *front-end* foi responsável por fazer as chamadas de API. Usamos o `axios.js`, uma biblioteca JavaScript bastante popular usada para fazer requisições HTTP. Com o `axios`, os retornos da API são convertidos automaticamente em JSON, formato adequado para manipulação dentro do JavaScript.

Para a autenticação, na página de *login*, enviamos o usuário e a senha digitados pelo usuário para o *endpoint* de *login*. Caso válidos, o *back-end* retorna um *token* necessário para realizar outras consultas. Usamos o gerenciador de estado `Vuex` para armazenar esse token no escopo global. O `Vuex` permite que todos os componentes da aplicação compartilhem o mesmo estado e que a mudança de estado seja previsível e reflita para todos os componentes.

Possuindo esse *token*, podemos fazer as chamadas de API restritas aos usuários autenticados, e assim implementar nossos casos de uso. Isso foi feito adicionando-o ao cabeçalho `Authorization` do HTTP. Caso o usuário tentasse acessar a página de mapas e o *token* não estivesse salvo no `Vuex`, o usuário seria redirecionado à página de login.

7 TESTES E AVALIAÇÃO

7.1 Testes de Métodos Numéricos

Realizamos testes para verificar a corretude dos métodos numéricos que usamos, o SLSQP, o Basinhopping e o Taguchi. Usamos funções de teste para otimização com mínimo global conhecido, e verificamos se era igual à saída do método numérico. A seguir estão os testes do método Basinhopping.

Nome da função	(x, y) do mínimo	(x, y) dos mínimos conhecidos
Easom	[100.0, 100.0]	[[3.14, 3.14]]
Himmelblau's	[-3.78, -3.28]	[[3, 2], [-2.81, 3.13], [-3.78, -3.28]]
Matyas	[-0.0, -0.0]	[[0, 0]]
Mccormick	[2.59, 1.59]	[[0.55, -1.55]]
Schaffer n.4	[98.96, 98.02]	[[0, 1.25]]
Three-hump camel	[0.0, -0.0]	[[0, 0]]

A seguir estão os testes do método SLSQP.

Nome da função	(x, y) do mínimo	(x, y) dos mínimos conhecidos
Easom	[100.0, 100.0]	[[3.14, 3.14]]
Himmelblau's	[-2.85, 3.08]	[[3, 2], [-2.81, 3.13], [-3.78, -3.28]]
Matyas	[-0.65, -0.65]	[[0, 0]]
Mccormick	[2.51, 1.51]	[[0.55, -1.55]]
Schaffer n.4	[100.0, 100.0]	[[0, 1.25]]
Three-hump camel	[0.0, -0.0]	[[0, 0]]

7.2 Resultados

As fontes do trabalho foram disponibilizadas como código aberto no GitHub, e foram separadas em dois repositórios: um para o *back-end* Django, com instruções de instalação para desenvolvimento local [16]; e um para o *framework* de *front-end* escrito em Vue [17]. Além disso, esta monografia em si também está disponível no GitHub [18].

8 CONSIDERAÇÕES FINAIS

8.1 Conclusões do Projeto de Formatura

8.2 Contribuições

8.3 Perspectivas de Continuidade

Ao longo da especificação de requisitos e da implementação, alguns aspectos foram deixados de lado por fugirem do escopo do MVP. Porém, eles serão interessantes em um produto final, agregando funcionalidades desejáveis aos *stakeholders*. Além disso, também será interessante detalhar o processo de criação de empresa para a distribuição do produto.

8.3.1 Plano de Negócio

8.3.2 Dados

Mais duas bases de dados que foram estudadas para uso no projeto são: a G-Econ [19], da Universidade de Yale, que apresenta os dados de paridade de poder de comparar geograficamente; e o SIMET-NIC [20], com o acesso e qualidade da Internet no Brasil. Estas duas bases de dados, em conjunto com os anteriores, podem permitir uma análise aprofundada de parâmetros ótimos para a instalação de novas antenas e avaliação da estrutura pré-existente; porém, a relação destes dados não cabe ao nosso caso de uso principal. Elas serão mais interessantes em uma análise futura, ao trabalhar em conjunto com os *stakeholders* para análises de instalação com mais fatores.

8.3.3 Descrição dos Casos de Uso Adicionais

Os casos de uso a seguir foram elaborados na etapa de “Especificação de Requisitos do Sistema”, porém foram deixados para uma implementação futura, já que não afetam

a demonstração do MVP.

Caso de Uso 8: Cadastrar usuário no sistema.

Descrição: Este caso de uso descreve o cadastro de usuários no sistema

Evento iniciador: Usuário clica no botão “Cadastrar usuário”.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário não logado no sistema.

Sequência de Eventos:

1. Usuário clica no botão “Cadastrar usuário”.
2. Sistema exibe página de cadastro.
3. Usuário digita nome, e-mail, senha e confirmação de senha.
4. Sistema valida dados e envia e-mail de confirmação para usuário.

Pós-condições: Usuário cadastrado no sistema com confirmação de e-mail pendente.

Extensões:

1. Usuário já cadastrado: Sistema exibe mensagem de erro (passo 4)
2. Senha não é igual a confirmação de senha: Sistema exibe mensagem de erro (passo 4)

Inclusões: Buscar usuário (passo 4)

Caso de Uso 9: Confirmação de e-mail.

Descrição: Este caso de uso descreve o processo de confirmação de um e-mail.

Evento iniciador: Usuário acessa a URL correspondente à confirmação de seu e-mail.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário cadastrado no sistema com e-mail não confirmado.

Sequência de Eventos:

1. Usuário acessa a URL correspondente à confirmação de seu e-mail.
2. Sistema confirma a validade do e-mail do usuário.
3. Sistema redireciona o usuário para página inicial.

Extensões: URL de confirmação inválida: Sistema mostra uma mensagem de erro e redireciona o usuário para a página de *login* (passo 2).

Inclusões: --

Caso de Uso 10: Adicionar nova ERB

Descrição: Este caso de uso descreve o processo de adição de novas ERBs na base de dados.

Evento iniciador: Usuário clica na opção “Adicionar ERB”.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema.

Sequência de Eventos:

1. Usuário clica na opção “Adicionar ERB” e clica em um ponto no mapa.
2. Sistema exibe formulário com dados a serem cadastrados sobre a nova ERB.
3. Usuário preenche formulário e clica em “Salvar”.
4. Sistema salva nova ERB.
5. Sistema exibe mapa com ERBs.

Pós-condições: Mapa com ERBs apresentado e nova ERB salva na base de dados.

Extensões: Dados inválidos: Sistema exibe mensagem de erro e exibe formulário novamente (passo 4).

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (pré-condição e passo 5).

Caso de Uso 11: Remover ERB

Descrição: Este caso de uso descreve o processo de remoção de uma ERB da base de dados.

Evento iniciador: Usuário clica na opção “Remover ERB”.

Atores: Administrador, Funcionário.

Pré-condições: Usuário logado no sistema.

Sequência de Eventos:

1. Usuário clica na opção “Remover ERB” e seleciona uma ERB no mapa.
2. Sistema pergunta se usuário realmente deseja remover ERB.

3. Usuário responde “Sim”.
4. Sistema remove ERB.
5. Sistema exibe mapa com ERBs.

Pós-condições: Mapa com ERBs apresentado e ERB escolhida pelo usuário removida da base de dados.

Extensões: --

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (pré-condição e passo 5).

Caso de Uso 12: Exibir ERBs às quais o celular do usuário está conectado

Descrição: Este caso de uso descreve o processo de exibição de ERBs às quais o usuário está conectado

Evento iniciador: Usuário clica na opção “Antenas conectadas”.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário logado em um celular, na página inicial do sistema.

Sequência de Eventos:

1. Usuário clica na opção “Antenas conectadas”.
2. Sistema exibe um mapa com as ERBs conectadas em destaque.

Pós-condições: Mapa com ERBs conectadas apresentado.

Extensões: Celular não está conectado a ERB nenhuma: Sistema exibe mensagem de erro (passo 2).

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (passo 2).

Caso de Uso 13: Exibir mapa com qualidade do sinal por operadora.

Descrição: Este caso de uso descreve o processo de exibição de qualidade de sinal por operadora.

Evento iniciador: Usuário clica na opção “Qualidade de sinal”.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário logado, na página inicial do sistema.

Sequência de Eventos:

1. Usuário clica na opção “Qualidade de sinal”.

2. Sistema exibe um mapa centrado na localização atual do usuário.
3. Usuário clica em um ponto do mapa.
4. Sistema exibe informações de qualidade de sinal por operadora.

Pós-condições: Informações de qualidade de sinal apresentadas.

Extensões: --

Inclusões: Caso de uso 3 “Exibir mapa” (passo 2)

Caso de Uso 14: Estimar geolocalização do usuário.

Descrição: Este caso de uso descreve o processo de estimar a geolocalização do usuário a partir das antenas às quais ele está conectado.

Evento iniciador: Usuário clica na opção “Estimar geolocalização”.

Atores: Administrador, Funcionário ou Usuário não-comercial.

Pré-condições: Usuário logado em um celular, na página inicial do sistema.

Sequência de Eventos:

1. Usuário clica na opção “Estimar geolocalização”.
2. Sistema exibe um mapa centrado na localização estimada do usuário.
3. Sistema mostra as antenas aos quais o usuário está conectado em destaque.

Pós-condições: Localização estimada do usuário e antenas conectadas são mostradas na tela.

Extensões:

1. Número de ERBs conectadas são insuficientes para estimar posição exata: Sistema exibe um mapa com a localização estimada do usuário representada por um círculo. (passo 2)
2. Usuário não está conectado a nenhuma ERB: Sistema exibe mensagem de erro. (passo 2)

Inclusões: Caso de uso 4 “Exibir mapa com ERBs” (passo 2).

Caso de Uso 15: Listar usuários do sistema

Descrição: Este caso de uso descreve o processo de listar usuários do sistema.

Evento iniciador: Administrador clica na opção “Listar usuários”.

Atores: Administrador

Pré-condições: Administrador logado no sistema, na página inicial.

Sequência de Eventos:

1. Administrador clica na opção “Listar usuários”.
2. Sistema busca usuários e os exibe em uma lista.

Pós-condições: Lista de usuários apresentada.

Extensões: --

Inclusões: Buscar usuários no sistema (passo 2)

Caso de Uso 16: Modificar papel de usuário.

Descrição: Este caso de uso descreve o processo de modificar papel de usuário.

Evento iniciador: Administrador seleciona a opção “Funcionário” no campo “Papel do usuário”

Atores: Administrador

Pré-condições: Administrador logado no sistema, na página listar usuários.

Sequência de Eventos:

1. Administrador seleciona um usuário de interesse.
2. Administrador seleciona a opção “Funcionário” no campo “Papel do usuário”.
3. Administrador clica em “Aplicar”.
4. Sistema modifica usuário escolhido para ele ter papel de Funcionário.

Pós-condições: Usuário escolhido tem papel de Funcionário.

Extensões: --

Inclusões: Caso de uso 15 “Listar usuários do sistema” (pré-condição)

REFERÊNCIAS

- [1] Forsk. *Atoll LTE / LTE-A Planning Software — Forsk* Disponível em: <http://www.forsk.com/ltelte-pro>. Acesso em: 01º de março de 2018.
- [2] TUDE, Eduardo. *Os desafios para a ampliação dos serviços de telecomunicações nos municípios*: Workshop. [22 de agosto de 2018]. Fiesp, São Paulo.
- [3] MyTower. *MyTower - Aluguel e Venda de Terrenos e Topos para Operadoras de Telecom* Disponível em: <http://www.mytower.com.br/>. Acesso em: 01º de março de 2018.
- [4] Skysites. *Skysites* Disponível em: <http://skysites.com/>. Acesso em: 01º de março de 2018.
- [5] LEE, S.; LEE, S.; KIM, K.; KIM, YH. *Base Station Placement Algorithm for Large-Scale LTE Heterogeneous Networks*. PLoS ONE 10(10), 2015.
- [6] KARULKAR, S. A.; OH, JY. *Optimal Placement of Base Station for Cellular Network Expansion*. Issues in Information Systems, volume 17, edição II, pg. 215-221, 2016.
- [7] WEGMANN, A.; VIERING I.; KLEIN, A. *A Joint Optimization of Antenna Parameters in a Cellular Network Using Taguchi's Method* IEEE 73rd Vehicular Technology Conference (VTC Spring), 2011.
- [8] Telebrasil. *Mapa de ERBs Brasil (antenas)*. Disponível em: <http://www.telebrasil.org.br/panorama-do-setor/mapa-de-erbs-antenas>. Acesso em: 31 de janeiro de 2018.
- [9] Unwired Labs. *OpenCellID - Largest Open Database of Cell Towers & Geolocation by Unwired Labs*. Disponível em: <https://opencellid.org/>. Acesso em: 01º de março de 2018.
- [10] Google Earth. *Google Earth Engine*. Disponível em: <https://earthengine.google.com/>. Acesso em: 16 de março de 2018.
- [11] Interactive Digital Media GmbH. *Mobile Country Codes (MCC) and Mobile Network Codes (MNC)*. Disponível em: <http://www.mcc-mnc.com>. Acesso em: 10 de outubro de 2018.
- [12] biodiv (GitHub). *anycluster*. Disponível em: <https://github.com/biodiv/anycluster>. Acesso em: 6 de maio de 2018.
- [13] EugeneOskin (GitHub). *django-geohash-cluster*. Disponível em: <https://github.com/EvgeneOskin/django-geohash-cluster>. Acesso em: 8 de setembro de 2018.
- [14] *SciPy.org*. Disponível em: <https://www.scipy.org/>. Acesso em: 13 de março de 2018.

- [15] *NumPy*. Disponível em: <https://www.numpy.org/>. Acesso em: 13 de março de 2018.
- [16] EpicEric (GitHub). *base_stations_django*. Disponível em: https://github.com/EpicEric/base_stations_django. Acesso em: 21 de fevereiro de 2018.
- [17] EpicEric (GitHub). *base_stations_vue*. Disponível em: https://github.com/EpicEric/base_stations_vue. Acesso em: 22 de julho de 2018.
- [18] EpicEric (GitHub). *tcc*. Disponível em: <https://github.com/EpicEric/tcc>. Acesso em: 27 de janeiro de 2018.
- [19] Yale University. *Geographically based Economic data -- Brazil*. Disponível em: <https://gecon.yale.edu/brazil>. Acesso em: 24 de março de 2018.
- [20] Núcleo de Informação e Coordenação do Ponto BR. *Sistema de Medição de Tráfego Internet*. Disponível em: <http://simet.nic.br/mapas-app.html>. Acesso em: 24 de março de 2018.