Um serviço para o desenvolvimento de aplicações de visualização de dados de saúde georreferenciados

Eduardo Gonçalves Pinheiro

Monografia apresentada ao
Instituto de Matemática e Estatística
da Universidade de São Paulo
para obtenção do título de
Bacharel em Ciência da Computação

Orientador: Prof. Dr. Paulo Meirelles
Coorientador: Prof. Dr. Fabio Kon

Durante o desenvolvimento deste trabalho o autor
recebeu auxílio financeiro da USP e do CNPq

São Paulo
24 de Novembro de 2018

Um serviço para o desenvolvimento de aplicações de visualização de dados de saúde georreferenciados

Eduardo Gonçalves Pinheiro

Esta é a versão original da
monografia entregue como parte
do trabalho final de MAC0499.

Autorizo a reprodução e divulgação total ou parcial deste trabalho, por qualquer meio convencional ou eletrônico, para fins de estudo e pesquisa, desde que citada a fonte.

Agradecimentos

Life is never completely
without its challenges.
— Stan Lee

Este trabalho só foi possível graças a um grupo muito grande de pessoas, cujo envolvimento em minha vida formaram a pessoa que sou hoje.

Agradeço primeiramente aos meus pais, *Cicero da Silva Pinheiro* e *Valdeilda Gonçalves da Silva Pinheiro*, pelo amor incondicional, suporte, carinho, amizade, e por todo sacrifício que me proporcionou a oportunidade de estar aqui.

A toda a minha família que sempre esteve ao meu lado em todos os momentos e cujo amor não pode ser expresso em palavras.

Aos meus orientadores *Paulo Meirelles* e *Fabio Kon*, por guiar o desenvolvimento deste projeto e auxiliar no meu crescimento pessoal.

Aos técnicos da SMS-SP, *Breno Aguiar* e *Marcelo Failla*, que atuaram ativamente durante o desenvolvimento, oferecendo feedback rápido e valioso para o projeto.

Aos alunos de iniciação científica *Camila Kodaira*, *Gabriely Pereira* e *Yurick Honda* que fazem parte deste projeto e auxiliaram imensamente em seu desenvolvimento.

A todos os professores do IME que fizeram parte da minha graduação e compartilharam conhecimento valioso para toda a vida.

Aos professores da FITO, do SESI, e do ANGLO que foram fundamentais para minha formação. Em especial ao Mestre *Kao Yung Ming*, meu primeiro professor de programação, seu trabalho exemplar me inspirou a seguir este caminho.

A todos os membros do Laboratório de Sistemas do CCSL, aprendi muito durante o tempo que ali estive.

Agradeço a todos os amigos que estão presentes na minha vida, e fazem tudo isso valer

a pena.

E por fim, agradeço aos seguintes desenvolvedores pelas suas contribuições ao projeto: *Débora Ciriaco, Fernanda de Camargo, Fernando Lemes, Lucas Gasparino, Lucas Helfstein, Lucas Kanashiro, Lucas Moura, Macartur Sousa, Murilo Ribeiro, Rosangela Pereira, e Waldir Caro.*

Resumo

Eduardo Gonçalves Pinheiro. **Um serviço para o desenvolvimento de aplicações de visualização de dados de saúde georreferenciados:** . Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2018.

As tecnologias de Cidades Inteligentes emergem como uma solução para enfrentar problemas comuns em grandes centros urbanos, usando os recursos da cidade de forma mais eficiente e proporcionando serviços de melhor qualidade para os cidadãos. Investigar estratégias e desenvolver aplicações para visualização espacial de grandes quantidades de dados de cidades, como São Paulo, a partir de múltiplas fontes heterogêneas se faz cada vez mais necessário na área da Computação. Nesta monografia, como estudo de caso para o desenvolvimento de uma aplicação de clusterização de dados, trabalhamos com dados públicos do Sistema Único de Saúde (SUS), fornecidos pela Secretaria Municipal de Saúde de São Paulo (SMS-SP). A grande quantidade de dados heterogêneos sobre saúde nas grandes cidades torna necessário a criação de novas formas de visualização interativa e georreferenciada. Nesse sentido, há diversas oportunidades de colaborações inerentes ao desenvolvimento dessas soluções, em especial para que pesquisadores da área da Saúde e da Ciência da Computação possam contribuir com formas inovadoras para coleta, armazenamento, gestão, visualização e análise de grandes quantidades de dados de saúde de populações urbanas. Esta monografia propõe uma arquitetura de referência para armazenamento de dados georreferenciados e apresenta uma aplicação piloto a ser utilizada pela SMS-SP.

Palavras-chave: Arquitetura de referência, RESTFul API, Cidades Inteligentes, Visualização de dados, Saúde pública

Lista de Abreviaturas

API	Interface de programação de aplicações (*Application Programming Interface*)
SMS-SP	Secretaria Municipal de Saúde de São Paulo
SIH	Sistema de Informações sobre Internações Hospitalares
SIM	Sistema de Informações sobre Mortalidade
SINASC	Sistema de Informação sobre Nascidos Vivos
CID	Classificação Estatística Internacional de Doenças e Problemas Relacionados com a Saúde
HTTP	Protocolo de Transferência de Hipertexto (*Hypertext Transfer Protocol*)
OSRM	*Open Source Routing Machine*
URL	Localizador Uniforme de Recursos (*Uniform Resource Locator*)
SQL	Linguagem de Consulta Estruturada (*Structured Query Language*)
NoSQL	Não Somente SQL (*Not Only SQL*)
MST	Árvores Geradoras de Custo Mínimo (*Minimum spanning tree*)
JSON	JavaScript Object Notation
GEOJSON	Geospatial Data Interchange Format Based on JavaScript Object
CSV	*Comma-separated values*
XLS	*Microsoft Excel spreadsheet file*
IME	Instituto de Matemática e Estatística
CCSL	Centro de Competência em Software Livre
USP	Universidade de São Paulo

Lista de Figuras

Sumário

Capítulo 1

Introdução

O impacto da Internet, recentemente ampliado com as tecnologias móveis e celulares inteligentes, tem sido crucial no dia-a-dia das pessoas, no comportamento das novas gerações, nos movimentos políticos e sociais (Kon e Blair, 2011), nos negócios, no comércio, na educação (Cooper e Sahami, 2013), e em áreas específicas tais como a indústria de software (Wasserman, 2011). Recentemente, com a popularização das tecnologias da Computação em Nuvem (Zhang *et al.*, 2010) e da Computação Móvel e com o crescente uso de Redes de Sensores e de tecnologias de Big Data (Hu *et al.*, 2014; Jagadish *et al.*, 2014), uma infinidade de novas possibilidades de aplicações estão surgindo, compondo a chamada "Internet do Futuro"(Batista *et al.*, 2016).

A Internet do Futuro integrará sistemas de grande porte construídos a partir da composição de milhares de serviços distribuídos selecionados a partir de milhões de possibilidades, sendo executados em uma grande quantidade de máquinas físicas e virtuais, manipulando grandes quantidades de dados multimídia (textos, imagens, áudio, vídeos, etc.) gerados a partir de inúmeras fontes possivelmente interagindo diretamente com o mundo físico através de sensores e atuadores e da Internet das Coisas (IoT) (Sheng *et al.*, 2013; Zorzi *et al.*, 2010). Essa Internet do Futuro será o elemento chave que possibilitará a realização de Cidades Inteligentes, um novo paradigma destinado a abordar os problemas dos grandes centros urbanos através da integração das tecnologias citadas com a infraestrutura da cidade.

O rápido crescimento das cidades ao redor do mundo criou centros urbanos densamente povoados caracterizados por organizações estruturais, sociais e econômicas complexas. De acordo com o Departamento de Assuntos Econômicos e Sociais das Nações Unidas (Nations, 2014), a proporção da população mundial que vive em áreas urbanas aumentou de 30% em 1950 para 54% em 2014 e, até 2050, espera-se chegar a 66%. Assim, as soluções de cidades inteligentes buscam superar os desafios desse rápido crescimento urbano ao redor do mundo provendo meios que ajudem no uso dos recursos da cidade de forma eficiente e proporcionando serviços de qualidade para seus cidadãos.

As soluções de software para cidades inteligentes contribuem não só para o dia-a-dia das cidades mas também para o planejamento de longo prazo e para a concepção de políticas públicas (Anttiroiko *et al.*, 2014). Entretanto, existem diversos aspectos que

devem ser desenvolvidos e estudados para transformar as tecnologias atuais em soluções efetivas para cidades inteligentes. Por exemplo, a integração de tecnologias como a Internet das Coisas, Big Data e Computação em Nuvem não é trivial, contendo tantas oportunidades quanto desafios. Outro fator importante é a falta de validações científicas e práticas das soluções atuais, dificultando uma análise sobre o impacto tecnológico, social e econômico na adoção das soluções existentes e prejudicando a realização de comparações mais adequadas. Por fim, vale destacar a importância do uso de tecnologias abertas, tais como padrões abertos, dados abertos e software livre. Iniciativas de cidades inteligentes centradas em soluções proprietárias dificultam o compartilhamento dos avanços, a aplicação das soluções desenvolvidas em outros contextos e forçam outros grupos de P&D a "reinventarem a roda".

No contexto da saúde, temos um crescimento da quantidade de dados gerados e armazenados por estabelecimentos de saúde. Cria-se assim o problema de filtrar informações a partir de diferentes e imensas bases de dados. Técnicas sofisticadas de visualização são muito utilizadas para contornar esse problema (Keim *et al.*, 2013). Entre elas a clusterização de dados tem se mostrado efetiva e eficaz (Jain *et al.*, 1999) para produzir visualizações que realçam as informações relevantes dos dados, viabilizando análises posteriores.

Nesta monografia, propomos uma arquitetura de referência para armazenamento e utilização de dados georreferenciados para auxiliar o desenvolvimento de aplicações de visualização e análise de dados geolocalizados. Além disso apresentamos como estudo de caso um sistema Web interativo de visualização de dados, chamado GeoMonitor da Saúde, desenvolvido durante este trabalho de conclusão de curso. Esta aplicação utiliza dados da saúde pública com geolocalização fornecidos pela Secretaria Municipal de Saúde de São Paulo (SMS-SP). Esperamos que esse sistema ofereça a profissionais de saúde pública a possibilidade de realizar análises de forma a compreender padrões de utilização dos serviços de saúde pública que não seriam visíveis sem o auxílio de um software que se utiliza de técnicas de visualização e clusterização de dados.

Os demais capítulos desta monografia estão organizados como a seguir. No Capítulo 2, apresentamos uma visão geral de métodos de clusterização, e apresentamos o método utilizado na aplicação desenvolvida. No Capítulo 3, descrevemos a proposta de arquitetura para um serviço que recebe dados georeferenciados e oferece a aplicações de visualização, dados filtrados e agrupados de acordo com suas necessidades. No Capítulo 4, é descrita a aplicação GeoMonitor da Saúde, uma aplicação desenvolvida que faz uso da *API* disponibilizada pelo serviço proposto. O Capítulo 5 expõe considerações finais sobre o trabalho, discutindo alguns passos futuros para a sua evolução.

Capítulo 2

Clusterização de dados

Análise de dados é parte fundamental da Ciência da Computação. Com o recente avanço na capacidade de armazenamento de dados e na ascensão de tecnologias de Big Data, manipular dados com o objetivo de extrair informações se tornou uma necessidade (Hu *et al.*, 2014). Dentro dessa realidade, clusterização de dados se tornou uma prática muito usada, como método de simplificar dados perdendo o mínimo possível de informação.

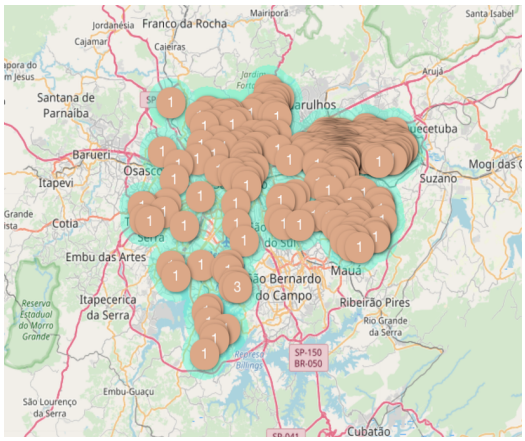

Figura 2.1: *Dados não clusterizados*

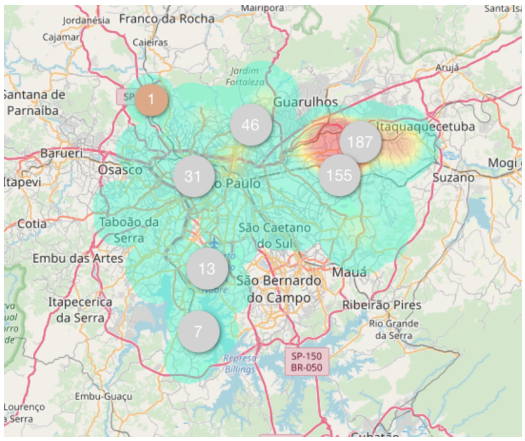

Figura 2.2: *Dados clusterizados*

Clusterização é a classificação não supervisionada dos dados através de seus padrões em grupos onde essas características são semelhantes (Jain *et al.*, 1999) baseando-se em alguma métrica de distância estabelecida, com o objetivo de minimizar a distância entre objetos no mesmo cluster e maximizar a distância entre os diferentes clusters (Amatriain *et al.*, 2011), com essa classificação podemos abstrair informações que não foram caracterizadas como relevantes e realçar as características que o cluster possui. Como ilustração, as Figuras 2.1 e 2.2 representam dados de internações hospitalares por aborto na cidade de São Paulo. Na Figura 2.2 vemos facilmente a concentração dos casos na zona Leste da cidade.

2.1 Componentes de uma tarefa de Clusterização

Jain *et al.* (1999) descrevem os passos necessários para um algoritmo de clusterização:

1. Representação dos padrões (opcionalmente incluindo extração e/ou seleção de características): faz referência ao número de classes e padrões disponíveis, e também ao número, tipo e escala das características disponíveis ao algoritmo de clusterização, algumas dessas informações podem não ser controláveis por quem utilizará o algoritmo.
    - **Seleção de características:** é o processo de identificar o subgrupo de características mais efitivo para o algoritmo de clusterização.
    - **Extração de características:** é o uso de uma ou mais transformações dos dados de entrada para produzir novas características mais interessantes.
2. Definição da medida de aproximação apropriada aos dados: normalmente é calculada pela função pré-definida de distância entre os objetos. Uma variedade de funções de distância são usadas de acordo com o domínio dos dados. Um exemplo de função muito usada é a distância euclidiana.
3. Clusterização ou agrupamento: existem diversos métodos para se realizar a clusterização, conforme a apresentado na Seção 2.1.1.
4. Abstração dos dados (se necessário): é o processo de extrair uma representação simples e compacta dos dados. Simplicidade em termos de análise automatizada (Uma máquina pode processar os dados), ou em termos de compreensão humana (A representação encontrada é intuitiva e de fácil compreensão).
5. Avaliação do resultado (se necessário).

2.1.1 Tipos de Clusterização

A Figura 2.3 apresenta uma visão geral das técnicas mais utilizadas para clusterização. Jain *et al.* (1999) separa técnicas de clusterização em dois conjuntos distintos ***Hierarchical*** e ***Partitional***. A seguir definimos cada um dos conjuntos, e descrevemos alguns dos principais algoritmos de cada grupo.

As técnicas do tipo ***Hierarchical*** organizam o conjunto de dados em uma estrutura hierárquica de acordo com a proximidade entre os objetos. O resultado de um algoritmo de clusterização hierárquica é geralmente uma árvore binária ou dendrograma [1] (Cassiano, 2015).

- *Single Link:* a distância entre dois *clusters* é a menor distância entre pares de objetos.
- *Complete Link:* a distância entre dois *clusters* é a maior distância entre pares de objetos.

[1] Árvore que iterativamente divide a base de dados em subconjuntos menores

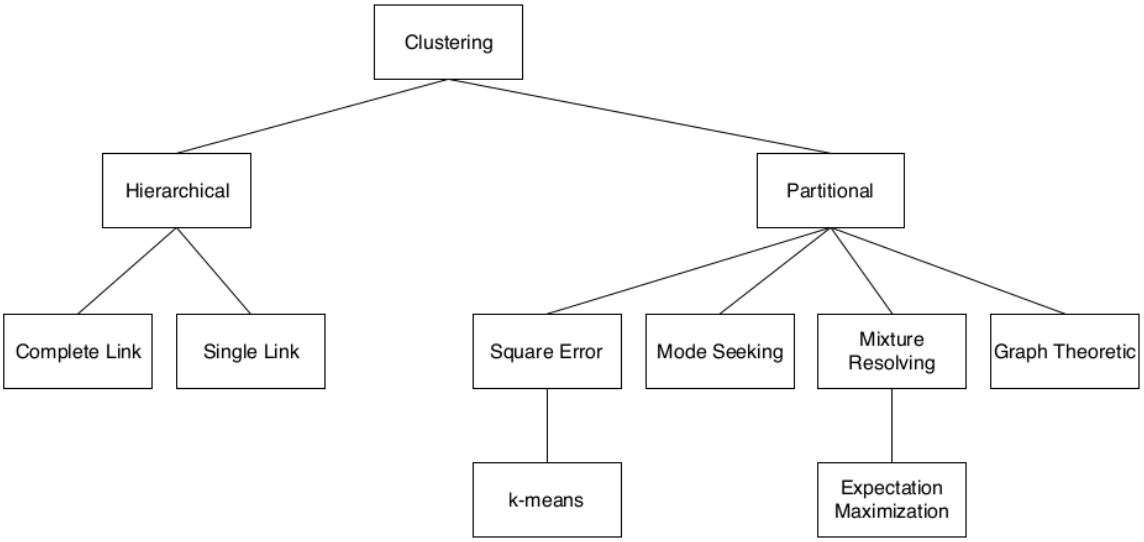

Figura 2.3: *Tipos de Clusterização*

Em ambos, *Single Link* e *Complete Link*, *clusters* são agrupados para formar *clusters* maiores hierarquicamente de acordo com o críterio de distância minima. Nos algoritmos *Complete Link* os *clusters* são compactos e fortemente ligados. Por outro lado, no caso dos algoritmos *Single Link* pode ocorrer um efeito de encadeamento, ou seja, o *cluster* pode ficar longo e "deformado"(Jain *et al.*, 1999). Aplicando as técnicas no mesmo conjunto de dados, com dois grupos distintos {1, 2} e dados com ruído {*}, vemos o resultado nas Figuras 2.4 e 2.5.

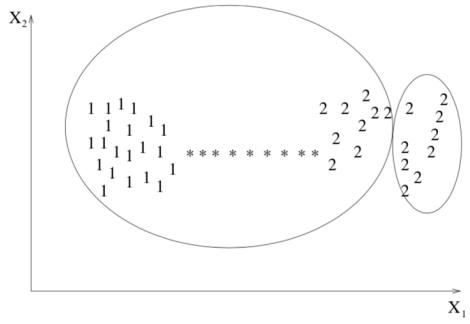

Figura 2.4: *Clusterização Hierárquica Single Link*

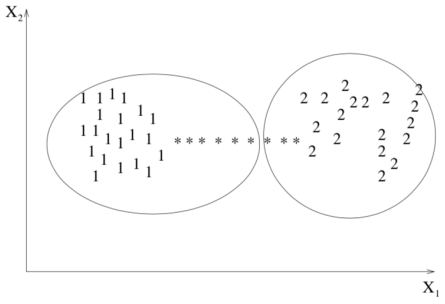

Figura 2.5: *Clusterização Hierárquica Complete Link*

Nas técnicas do tipo ***Partitional***, os algoritmos de clusterização dividem a base de dados em k grupos, escolhendo k objetos como sendo centros, os demais objetos são então divididos entre os k grupos de acordo com a medida de aproximação ao centro definida (Cassiano, 2015).

- *Square Error:* o mais intuitivo e mais utilizado entre os algoritmos particionais. A função de distância para a clusterização θ, com conjunto de características χ (Contendo K clusters) é definida pela função (Jain *et al.*, 1999):

$$e^2(\chi, \theta) \quad = \quad \sum_{j=1}^{K} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|^2$$

- *Graph Theoretic:* são construídas árvores geradoras de custo mínimo (*Minimum spanning tree*) a partir dos dados. Arestas de maior comprimento são removidas para gerar os *clusters* (Jain *et al.*, 1999).

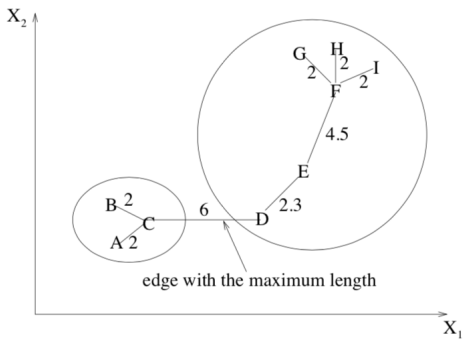

Figura 2.6: *Clusterização Graph Theoretic*

A Figura 2.6 descreve o processo de clusterização *Graph Theoretic* para nove pontos. A aresta CD de maior comprimento, pode ser desfeita criando dois *clusters* {A, B, C} e {D, E, F, G, H, I}. O processo pode ser feito novamente, quebrando a aresta EF criando dois *clusters* {D, E} e {F, G, H, I} e assim por diante até um certo comprimento definido.

2.2 Clusterização Geoespacial

Assim como no problema geral de clusterização discutido no Seção 2.1, no problema de clusterização geoespacial buscamos agrupar objetos semelhantes no mesmo cluster enquanto maximizamos a distância entre *clusters*; nesse caso, o algoritmo toma essa decisão baseado na distância geográfica entre os objetos (Wang *et al.*, 2010).

O algoritmo utilizado no GeoMonitor da Saúde é o algoritmo denominado Clusterização gulosa hierárquica (*Hierarchical greedy clustering*) que está na categoria de algoritmos de clusterização hierárquica *Complete Link* (vide Seção 2.1.1), disponibilizado na biblioteca de clusterização geoespacial "Leaflet.makercluster" [2].

Como definido por Agafonkin, 2016, a base do algoritmo de clusterização segue os seguintes passos:

1. Começa com um ponto qualquer dos dados.
2. Encontra todos os pontos dentro de um raio específico.
3. Forma um cluster com esses pontos.
4. Encontra um novo ponto que não faz parte de nenhum cluster.
5. Cria um novo cluster a partir dele.
6. Repita o processo até visitar todos os pontos.

[2] https://github.com/Leaflet/Leaflet.markercluster

Como a clusterização é feita em um mapa onde é possível alterar o nível de aproximação (*zoom*), seria necessário executar o algoritmo para todos os níveis de aproximação existentes, o que seria muito custoso em termos de desempenho, para contornar esse problema temos a abordagem hierárquica do algoritmo, reutilizando os cálculos efetuados.

A Figura 2.7 apresenta esse processo. Os valores z18, z17, z16, z15, e z14 na figura representam os diversos níveis de aproximação. Tendo calculado os clusters no nível z18, podemos agrupar os clusters resultantes em clusters no nível z17, da mesma forma encontramos os clusters no nível z16 e assim por diante. Num processo hierárquico podemos criar clusters a partir de clusters, minimizando consideravelmente a quantidade de dados a ser processada em cada nível.

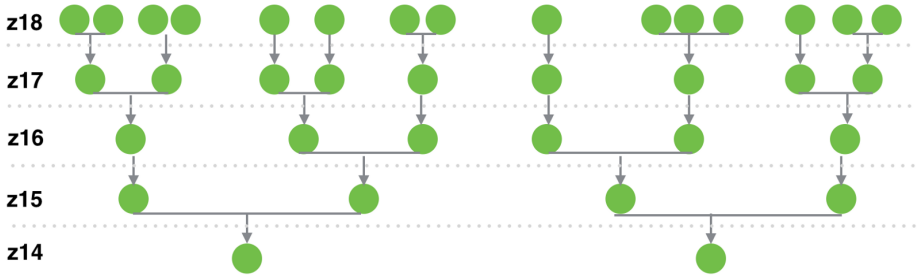

Figura 2.7: *Clusterização hierárquica*

2.3 Clusterização no GeoMonitor da Saúde

No caso do GeoMonitor da Saúde, o processo de anonimização (vide Seção 3.2.2) tem o efeito de reduzir o problema de clusterização das internações hospitalares (554.202 no ano de 2015), no problema de clusterizar setores censitários [3] (18.953 na cidade de São Paulo), pois como vemos no algoritmo descrito por Agafonkin, 2016, pontos com a mesma localização serão agrupados no mesmo cluster. Dessa forma, a clusterização do GeoMonior da Saúde é feita em duas etapas:

1. Agrupamos os dados de acordo com seu setor censitário, etapa feita pelo módulo de agrupamento, descrito na Seção 3.2.2.
2. Utilizamos uma versão modificada do algoritmo de clusterização que atribui a cada ponto a ser clusterizado um peso de acordo com a quantidade de internações que ocorreram no setor censitário representado pelo ponto. O algoritmo padrão assume peso unitário a todos os pontos.

Como podemos ver na Figura 2.8, no algoritmo padrão a clusterização não leva em conta a quantidade de internações que ocorreram em um mesmo ponto, assim somando os valores dos clusters teríamos apenas 16.868 (número de setores censitários em que

[3] Setor censitário é a unidade territorial de controle cadastral da coleta do censo do IBGE, constituída por áreas contíguas, respeitando-se os limites da divisão político-administrativa, dos quadros urbano e rural legal e de outras estruturas territoriais de interesse, além dos parâmetros de dimensão mais adequados à operação de coleta.

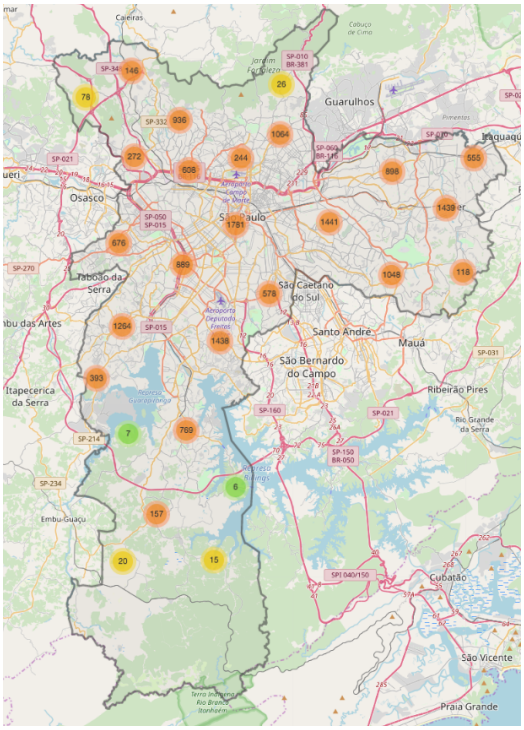

Figura 2.8: *Método de Clusterização padrão da biblioteca Leaflet.markercluster*

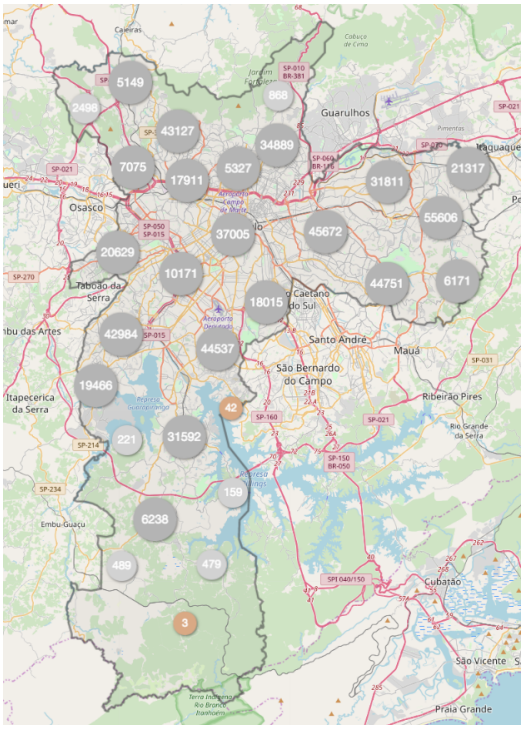

Figura 2.9: *Método de Clusterização Modificado do GeoMonitor da Saúde*

ocorreram pelo menos uma internação) pontos nos clusters. Já na Figura 2.9 ao somar os valores temos o resultado correto de 554.202 internações.

No próximo capítulo, apresentamos a proposta de um *serviço* para base de dados geolocalizados, descrevendo uma arquitetura que atuará para fornecer dados a aplicações de visualização. O *serviço* em sua arquitetura, define um módulo de *Agrupamento* dos dados que visa auxiliar algoritmos de clusterização como os vistos neste capítulo e utilizados na aplicação *GeoMonitor da Saúde*.

Capítulo 3

Proposta de serviço para base de dados geolocalizados da saúde

A Secretaria municipal de saúde de São Paulo recebe dados de muitas fontes diferentes. Esses dados não são padronizados, dificultando assim a sua utilização por aplicações de visualização de dados, já que sua heterogeneidade implica na necessidade de implementação de aplicações especificas para cada base de dados existente. Isso dificulta também a adoção dessas tecnologias por gestores públicos.

Com o objetivo de mitigar esse problema, propomos a arquitetura de um serviço para base de dados, inicialmente da saúde, prevendo expansão para dados georreferenciados de qualquer fonte. Essa arquitetura busca abstrair as camadas de manipulação dos dados com o objetivo de oferecer a aplicações uma *API* de fácil utilização para inserção e consulta de dados independente da sua origem. A partir disso, a mesma aplicação poderá apresentar análises de dados da saúde, segurança e mobilidade urbana.

3.1 Decisões de *Design*

Dado o contexto em que a aplicação foi desenvolvida, alguns princípios fundamentais foram estabelecidos:

- **Software Livre:** utilização de componentes de software existentes é uma pratica fundamental de engenharia de software para aumentar produtividade, eficiência e qualidade de software. No desenvolvimento deste sistema, damos preferência à utilização de ferramentas, bibliotecas e *frameworks* de *Software* Livre já existentes.
- **Padrões abertos:** utilizar padrões bem definidos e amplamente usados tanto na academia quanto na indústria, facilita a integração de novos desenvolvedores bem como a utilização do sistema por *software* cliente. Desta forma, o uso de padrões é sempre recomendado, priorizando padrões abertos.
- **Modularidade do projeto:** buscando facilitar o ingresso de novos desenvolvedores, o projeto é modularizado de modo que modificações locais causem apenas impactos

locais, mantendo a complexidade baixa mesmo com a evolução da aplicação.

- **Modelo de dados não-relacional:** para atingir o objetivo de criar uma plataforma capaz de abstrair o formato dos dados recebidos e oferecer às aplicações dados no mesmo formato, o modelo dos dados deve ser capaz de se adaptar as diferenças encontradas e, com isso, optamos por modelos não-relacionais de dados.

3.2 Arquitetura

A arquitetura proposta está descrita na Figura 3.1, apresentando três principais componentes: as aplicações, o *serviço* [1], e as bases de dados. Considerando o nível de desacoplamento proposto, os componentes podem ser desenvolvidos e mantidos por desenvolvedores diferentes, comunicando-se via chamadas *REST*.

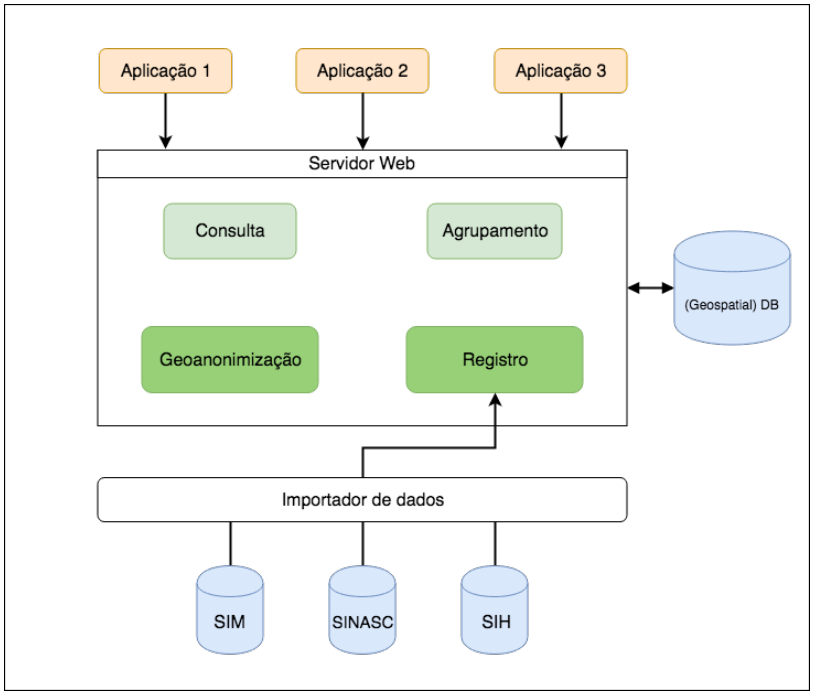

Figura 3.1: *Arquitetura Proposta*

As aplicações fazem referência as diferentes formas de visualização que serão utilizadas a partir dos dados disponibilizados. O *serviço* é composto de quatro módulos (*Consulta*, *Agrupamento*, *Geoanonimização* e *Registro*) que atuam em conjunto para receber dados geolocalizados, processá-los e disponibilizá-los a aplicações por meio de requisições *REST*.

Entre as aplicações, destacamos o *GeoMonitor da Saúde* (vide Capítulo 4), desenvolvido para visualizar dados georreferenciados referentes aos sistemas de informação de mortalidade (SIM), de nascidos vivos (SINASC), e internações hospitalares (SIH) fornecidos pela

[1] https://gitlab.com/interscity/health-dashboard/datahealth-api

SMS-SP. O desafio de criar uma mesma visualização para diferentes bases de dados foi a motivação para a elaboração da arquitetura descrita a seguir.

3.2.1 Aplicações

Nesta camada está a interface com os usuários. A flexibilidade fornecida por esta arquitetura torna possível que o desenvolvedor desta camada possa criar visualizações complexas para dados georreferenciados e, utilizando a *API*, utilizar como fonte dados variados sem a necessidade de fazer grandes modificações em sua aplicação. A comunicação entre aplicações e o *serviço* é feita via chamadas *REST*, onde parâmetros são enviados em formato *JSON* e as respostas do *serviço* são em *GEOJSON*.

3.2.2 Serviço

O *serviço*, principal componente do projeto, faz a comunicação entre os dados e as aplicações que buscam fazer análises sobre eles. Para isso, os dados recebidos são tratados e armazenados de modo que a aplicação cliente recebe, ao fazer uma requisição, dados formatados em *GEOJSON*. Com isso diferentes bases de dados podem ser utilizados pela mesma aplicação de visualização.

Os módulos internos do *serviço* estão divididos em dois grupos:

- **Tratamento dos dados:** os módulos ***Registro*** (vide Seção 3.2.2) e ***Geoanonimização*** (vide Seção 3.2.2) tratam os dados recebidos a partir dos *importadores de dados*. São responsáveis por checar se o formato está correto, bem como verificar se os dados são geolocalizadas, além disso outro papel fundamental é o de garantir que não seja possível identificar o local de origem dos dados [2].
- **Serviço a aplicações:** os módulos ***Consulta*** (vide Seção 3.2.2) e ***Agrupamento*** (vide Seção 3.2.2) lidam com as requisições de aplicações clientes que buscam os dados disponíveis no serviço. Eles são responsáveis pelo processamento desses dados com o objetivo de garantir o mesmo formato de resposta para a aplicação que fez a requisição.

A seguir estão descritos os módulos do *serviço*.

Registro

Com a intenção de criar um *serviço* genérico para atender a diversas bases de dados, o módulo de Registro foi projetado para fornecer um ponto de entrada de novos registros neste *serviço*. Um registro é um dado georreferenciado que é armazenado pelo serviço para ser disponibilizado às aplicações. A Figura 3.2 ilustra a estrutura de um registro em que é

[2] O processo de geolocalização possuí grande precisão, tornando possível identificar o endereço original, o módulo de *Geoanonimização* tem a função de aplicar um deslocamento impossibilitando assim a identificação do paciente.

necessário definir o tipo, coordenadas e suas propriedades. No *GeoMonitor da Saúde* (vide Capítulo 4) uma internação hospitalar é um registro.

```
{
    "type": "Feature",
    "geometry": {
        "type": "Point",
        "coordinates": [long, lat]
    },
    "properties": {
        "prorpiedade_1": "valor_propriedade_1",
        "propriedade_2": "valor_propriedade_2",
        "propriedade_n": "valor_propriedade_n"
    }
}
```

Figura 3.2: *Exemplo de um registro*

Este módulo proporciona uma interface para consultar, alterar e remover registros existentes. Podendo ser utilizado diretamente por meio de requisições *REST* ou por meio de aplicações facilitadoras aqui denominadas *importadores de dados* (vide Seção 3.2.3), esta utilização é feita nos seguintes *endpoints*:

- ***POST /record***

  Salva o registro.

- ***GET /record/:_id***

  Retorna as informações do registro que possui o id passado como parâmetro.

- ***PUT /record/:_id***

  Altera valores de um registro de acordo com os valores passados na requisição.

- ***DELETE /record/:_id***

  Remove o registro que possui o valor id.

Para tornar possível o registro de base de dados diferentes com campos distintos, utilizaremos internamente um banco de dados *NoSQL* orientado a documentos. Nesse caso, no formato *GEOJSON*. Com isso ***POST /record*** recebe documentos no formato descrito na Figura 3.2.

- **type:** indica para o *parser* interno se o documento contém informações de um ponto "Feature" ou de um conjunto de pontos "FeatureCollection".
- ***geometry:*** neste campo estão contidas informações da geolocalização do registro.
  - ***type:*** o formato *GEOJSON* é capaz de identificar sete formatos geométricos. Os dados utilizados são do formato *Point*, ou seja são pontos geométricos.

  - ***coordinates:*** identifica as coordenadas [3] do ponto registrado.

- ***properties:*** todas informações referentes ao registro ficaram armazenadas neste campo.

O campo ***properties*** é o responsável por atingir o nível de abstração dos dados desejado. Se a base SIH possui os campos (ESPECIALIDADE, CNES, COMPLEXIDADE) e a base SIM (DIARIAS, P_IDADE, CAR_INTERNAÇÃO), é possível salvar os dados de ambas com a mesma chamada alterando apenas os campos do objeto enviado em *properties*. O banco de dados *NoSQL* nos dá a capacidade de fazer consultas utilizando as propriedades, apesar dos valores serem dinâmicos.

Entre as informações enviadas em ***properties*** damos destaque à propriedade ***database*** que identifica a base de dados a que o registro pertence, facilitando o consumo desses dados por aplicações que utilizarão o serviço como fonte de dados. Portanto, é essencial que todos os registros sejam identificados quanto a sua origem.

A aplicação é capaz de receber múltiplos registros em uma única chamada. Com isso, a quantidade de chamadas efetuadas ao inserir os dados é reduzida. Isso é alcançado utilizando o *type* "FeatureCollection" do *GEOJSON*, conforme exemplificado na Figura 3.3.

Após validar o documento recebido e verificar que ele possui o formato correto, o módulo de *Geoanonimização* (vide Seção 3.2.2) é acionado antes de concluir o registro do dado.

Esse formato de registro de dados, portanto, estende o escopo dos dados que podem ser visualizados nas aplicações. De forma geral, qualquer dado geolocalizado pode ser adicionado à aplicação, possibilitando que uma mesma aplicação de visualização seja utilizada para dados de fontes diferentes.

Geoanonimização

Parte fundamental da utilização de dados da Saúde é impossibilitar a identificação do paciente por terceiros. Atualmente técnicos da SMS-SP tem o papel de fazer a anonimização dos dados, ela é feita deslocando o local de origem do paciente para o centróide do setor censitário [4] em que ele reside. O processo atual é demorado e custoso para a SMS-SP. Desta forma, o serviço proposto oferece um módulo de *Geoanonimização* que tem por objetivo fazer todo o processo de anonimização dos dados de forma automática assim que os dados são recebidos. O módulo proposto recebe uma coordenada geográfica e retorna o centróide do setor censitário em que ela se encontra.

[3] Coordenadas no sistema WGS 84

[4] Setor censitário é a unidade territorial de controle cadastral da coleta do censo do IBGE, constituída por áreas contíguas, respeitando-se os limites da divisão político-administrativa, dos quadros urbano e rural legal e de outras estruturas territoriais de interesse, além dos parâmetros de dimensão mais adequados à operação de coleta.

[5] https://mapshaper.org/

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [long1, lat1]
            },
            "properties": {
                "propriedade1": "valor_propriedade1",
                "propriedade2": "valor_propriedade2",
                ...
            }
        },
        {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": [long2, lat2]
            },
            "properties": {
                "propriedade1": "valor_propriedade1",
                "propriedade2": "valor_propriedade2",
                ...
            }
        }
    ]
}
```

Figura 3.3: *Exemplo de GEOJSON com múltiplos pontos*

Para efetuar essa tarefa, utilizamos dados do Censo Demográfico de 2010 que fornece os polígonos que identificam os setores censitários de todo o Brasil. Separamos os dados referentes à cidade de São Paulo. Esses arquivos foram convertidos para o formato *GEOJSON* possibilitando a criação de uma base de dados georreferenciada que identifica os setores censitários como objetos geométricos. Nessa base, é possível fazer consultas georreferenciadas. Com a coordenada recebida no modulo de *registro*, é feita uma consulta que resulta no setor censitário em que a coordenada se encontra. Na nossa base de dados, a coordenada salva é o centróide do polígono resultado desta consulta, ou seja, o centróide do setor censitário garantindo, assim, a anonimidade do dado. É importante mencionar que o Censo Demográfico possui dados de todo Brasil, assim esse processo pode ser efetuado para qualquer cidade do Brasil.

Consulta

O módulo de *consulta* foi projetado para entregar às aplicações dados com todas as suas informações, sem nenhum tipo de tratamento, além da aplicação dos filtros passados na requisição. Ele surge da necessidade que certas aplicações possuem de apresentar

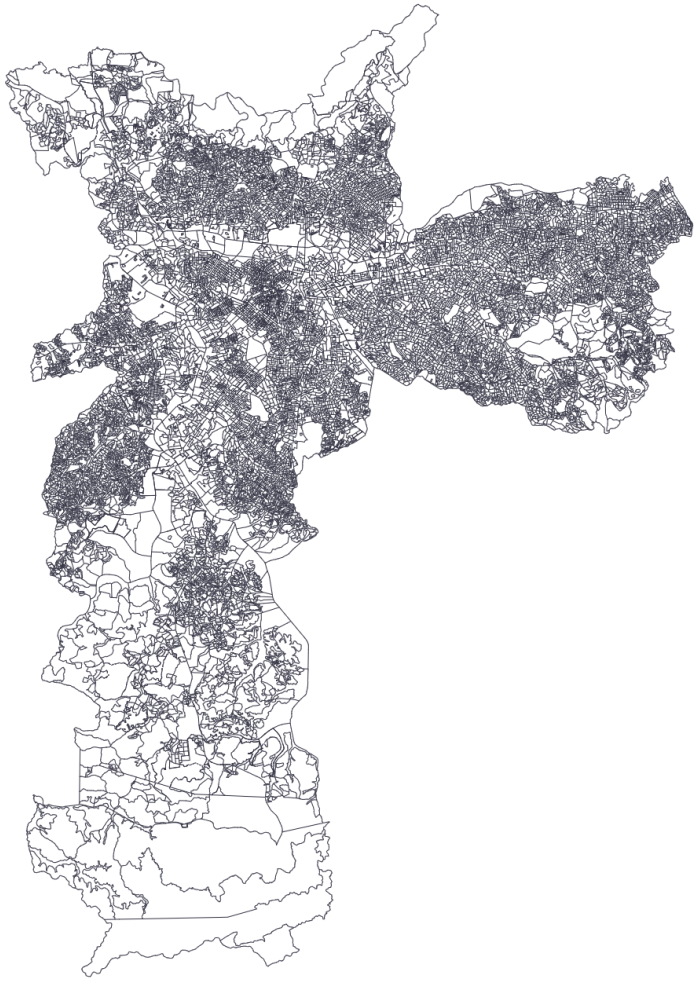

Figura 3.4: *Setores Censitários da cidade São Paulo* [5]

informações completas sobre os dados, ou se a aplicação deseja fazer uma análise dos dados que o formato oferecido pelo módulo de *Agrupamento* (vide Seção 3.2.2) não é capaz de atender. A requisição é feita no *endpoint /query*, descrito abaixo:

- ***POST /query*** este *endpoint* pode ser usado para buscar registros que correspondem aos filtros especificados na chamada. Os filtros podem ser da seguinte forma:

  - ***properties:*** filtra os dados de acordo com as propriedades. Este filtro será usado para criar consultas dinâmicas no banco de dados, portanto os valores passados como parâmetros devem coincidir com valores que os registros salvos possuem. Os seguintes operadores são aceitos:

    * **eq** - o registro deve ter valor igual ao parâmetro.
    * **gt** - o registro deve ter valor maior que o parâmetro.
    * **gte** - o registro deve ter valor maior que ou igual ao parâmetro.

* **lt** - o registro deve ter valor menor que o parâmetro.
* **lte** - o registro deve ter valor menor que ou igual ao parâmetro
* **in** - o valor deve estar contido na lista de valores especificada na chamada.
* **ne** - o registro deve ter valor diferente do parâmetro.

A requisição terá o formato visto na Figura: 3.5

```
{
    "properties": {
        "database": "SIM",
        "CNES": ["eq", "201030"],
        "P_IDADE": ["gt", 26]
    }
}
```

Figura 3.5: *Requisição aceita pelo endpoint /group*

– ***location:*** para receber registros de acordo com uma localização, informando os parâmetros lat e long. Como resposta, teremos os registros que estão no ponto específico ou, enviando também um parâmetro *radius*, os registros num raio de distância do ponto informado, como mostrado na Figura 3.6.

```
{
    "properties": {
        "database": "SIH",
        "location": [long, lat],
        "radius": 6.5
    }
}
```

Figura 3.6: *Dados da requisição aceita pelo endpoint /query geolocalizado*

A resposta dessa requisição será um arquivo *GEOJSON* com os registros que se enquadram nos filtros enviados, como ilustrado na Figura 3.7.

Agrupamento

O módulo de *agrupamento* foi desenvolvido com o objetivo de auxiliar aplicações que farão análises dos dados registrados na aplicação. Ele alcança esse objetivo fornecendo ao cliente uma série de operadores capazes de agrupar os dados de várias formas em velocidade superior a que as aplicações seriam capazes.

O módulo de *agrupamento* faz parte, juntamente com o módulo de *consulta*, da integração com as aplicações que utilizam os dados. O *endpoint* a ser utilizado é o */group*, descrito abaixo:

```
{
    "type": "FeatureCollection",
    "features": [
        {
            "type": "Feature",
            "geometry": {"type": "Point", "coordinates": [
                long1, lat1]},
            "properties": {"database": "SIH", "propriedade1":
                "valor_propriedade1", "propriedade2": "
                valor_propriedade2" ...}
        },
        {
            "type": "Feature",
            "geometry": {"type": "Point", "coordinates": [
                long2, lat2]},
            "properties": {"database": "SIM", "propriedade1":
                "valor_propriedade1", "propriedade2": "
                valor_propriedade2" ...}
        },
        {
            "type": "Feature",
            "geometry": {"type": "Point", "coordinates": [
                long3, lat3]},
            "properties": {"database": "SINASC", "propriedade1
                ": "valor_propriedade1", "propriedade2": "
                valor_propriedade2" ...}
        }
    ]
}
```

Figura 3.7: *Resposta do endpoint /query*

- ***POST /group*** da mesma forma que */query* este *endpoint* recebe um *JSON* com filtros. Adicionalmente, o usuário pode enviar um campo pelo qual os dados serão agrupados e um operador de *agrupamento*. Os seguintes operadores são aceitos:

  - **avg:** a média dos valores do campo especificado.
  - **max:** o maior valor para o campo especificado.
  - **min:** o menor valor para o campo especificado.
  - **sum:** a soma dos valores do campo especificado.
  - **count:** a quantidade de registros por valor distinto do campo especificado.

  O *endpoint /group* aceita como entrada arquivos JSON com o formato visto na Figura 3.8.

  Assim como */query* a resposta deste *endpoint* é um arquivo *GEOJSON* no formato apresentado na Figura: 3.9

```
{
    "properties": {
            "database": "SIH",
            "CNES": ["eq", 291245]
    },
    "group": ["count", ["coordinates"]]
}
```

Figura 3.8: *Dados da requisição aceita pelo endpoint /group*

```
{
    "type": "FeatureCollection",
    "features": [
        {"type": "Feature", "geometry": {"type": "Point", "
            coordinates": [lat1, long1], "properties": {"
            counter": 10}}},
        {"type": "Feature", "geometry": {"type": "Point", "
            coordinates": [lat2, long2], "properties": {"
            counter": 5}}},
        {"type": "Feature", "geometry": {"type": "Point", "
            coordinates": [lat3, long3], "properties": {"
            counter": 20}}},
        {"type": "Feature", "geometry": {"type": "Point", "
            coordinates": [lat4, long4], "properties": {"
            counter": 2}}},
        {"type": "Feature", "geometry": {"type": "Point", "
            coordinates": [lat5, long5], "properties": {"
            counter": 1}}}
    ]
}
```

Figura 3.9: *Resposta do endpoint /group*

Note que, nesse caso, cada ponto possui, em propriedades, o valor de *counter*. Naquele ponto foram encontrados *counter* registros. Esse *agrupamento* por coordenadas auxilia, por exemplo, na clusterização efetuada na aplicação GeoMonitor da Saúde (vide Capítulo 4) diminuindo o número total de pontos.

3.2.3 Importador de dados

A SMS-SP recebe os dados em formatos variados (*CSV*, *Shapefile*, *xls*, etc.). No entanto, o serviço proposto, em seu *endpoint* de registro, recebe os dados no formato *GEOJSON*. Para facilitar o processo de registro dos dados, uma pequena aplicação foi criada para converter os dados para o formato adequado. Com isso, o trabalho de inserção dos dados é automatizado, facilitando o trabalho de quem atualiza a aplicação e evitando o registro de dados incorretos.

O *importador de dados* proposto recebe dados de duas formas distintas:

- **Arquivo:** o *importador* será capaz de ler um arquivo do tipo *CSV* (vide Figura 3.10) ou *Shapefile*, e gerar automaticamente documentos *GEOJSON* e registrá-los na aplicação.

CD_GEOCODI	ESPECIALID	CMPT	DT_EMISSAO	CEP
355030842000059	5	201504	2015-02-22	05508-090
355030842000059	3	201505	2015-03-24	05508-220
355030842000059	3	201508	2015-07-25	05501-100
355030842000059	7	201503	2015-03-10	05001-020
355030842000059	3	201504	2015-03-30	04001-120
355030842000059	3	201505	2015-04-21	05511-010
355030842000059	2	201506	2015-05-17	05502-222
355030842000059	3	201504	2015-03-30	05534-187
355030842000059	3	201510	2015-10-05	01518-070
355030842000059	1	201506	2015-04-13	05928-110

Figura 3.10: *Exemplo de Arquivo CSV aceito pelo importador de dados*

- **Input do usuário:** a aplicação terá uma página onde o usuário poderá descrever o registro a ser adicionado. Com esses dados o *importador* fará o processo de registro na aplicação.

O *importador de dados* também deve ser capaz de geolocalizar registros. Um requisito para o desenvolvimento da arquitetura é integrar o processo de *geoanonimização* à aplicação. Parte desse processo envolve receber um endereço (CEP, logradouro, nome etc.) e encontrar as coordenadas (longitude, latitude) desse endereço. Essa função é desempenhada pelo *importador*, pois o documento no formato *GEOJSON* deve conter coordenadas, que serão tratadas pelo módulo de *Geoanonimização*. Com isso, o usuário pode cadastrar no *importador* um registro no formato apresentado na Figura 3.11.

```
1       {
2           "ENDERECO": "Rua do Matao, 1010",
3           "properties": {
4               "ESPECIALID": 1, "CMPT": 201508, "DT_EMISSAO": "
                    2015-06-22"
5           }
6       }
```

Figura 3.11: *Entrada do Importador de dados*

O *importador*, portanto, criará um arquivo *GEOJSON* no formato ilustrado na Figura 3.12.

Dado o arquivo descrito na Figura 3.10, o *script* descrito na Figura 3.13 atua como um *importador de dados*. O *script* lê cada linha do arquivo, com o valor do CEP encontrado, utiliza a biblioteca *geocoder* [6] para encontrar as coordenadas daquele local. Depois do processo de geolocalização, os dados são formatados em *GEOJSON* e enviados para a aplicação com uma requisição *POST* no *endpoint /record*.

[6] http://www.rubygeocoder.com/

```
{
    "type": "Feature,
    "geometry": {
        "type": "Point",
        "coordinates": [-46.7319998, -23.559667],
        "properties": {
            "ESPECIALIDADE": 1, "CMPT": 201508, "DT_EMISSAO":
                "2015-06-22"
        }
    }
}
```

Figura 3.12: *Saída do importador de dados*

```
require 'json'
require 'geocoder'
require 'csv'
require 'rest-client'

CSV.foreach(procedure_csv_path, :headers => true) do |row|
    cep = row[4]
    location = Geocoder.search(cep).first

    values = {}
    values[:properties] = {}
    values[:properties][:database] = "SIH"
    values[:properties][:CD_GEOCODI] = row[0]
    values[:properties][:ESPECIALID] = row[1]
    values[:properties][:CMPT] = row[2]
    values[:properties][:DT_EMISSAO] = row[3]

    point = {}
    point[:type] = "Point"
    point[:coordinates] = [location[0].to_f, location[2].to_f]

    RestClient::Request.execute(method: :post,
                                url: ENV["API_URL"],
                                payload: values.to_json,
                                headers: {"Content-Type" => "
                                    application/json"}
                               )
end
```

Figura 3.13: *Script para registrar dados na aplicação*

3.3 Detalhes de Implementação

Esta seção apresenta escolhas técnicas feitas durante o desenvolvimento da aplicação.

3.3.1 *RESTFul API*

Este padrão de arquitetura foi adotado dada sua grande aceitação por parte da comunidade de desenvolvedores tornando assim, a aplicação mais atrativa a potenciais desenvolvedores que buscam *APIs* abertas para serem base de seus aplicativos. Além disso, facilita o acoplamento de aplicações já existentes que utilizam esse padrão.

3.3.2 *Ruby On Rails* [7]

É um *framework* para desenvolvimento de aplicações *Web* e *serviços*. Foi escolhido dada a familiaridade do grupo de sistemas do IME-USP de desenvolvimento com a linguagem *Ruby*, além dos seguintes fatores:

- Software Livre com grande comunidade ativa de desenvolvedores.
- Modo *API-Only*[8] que facilita o processo de criação de um serviço.
- Grande quantidade de *gems* [9] tornando o reuso de código fácil e intuitivo.

3.3.3 *MongoDB* [10]

Banco de dados orientado a documentos permite maior flexibilidade quanto aos dados que cada registro deve possuir, sem perder a capacidade de manipulação dos dados (consultas, agrupamentos, contagem etc). Assim, o sistema de gerenciamento de banco de dados escolhido foi o *MongoDB*, pelos motivos elencados a seguir:

- Software Livre com comunidade de desenvolvedores grande e em expansão.
- Facilidade de integração com *Ruby on Rails*.
- Módulo para dados geolocalizados que melhora o desempenho da aplicação e fornece consultas geolocalizadas.

[7] https://rubyonrails.org/

[8] https://guides.rubyonrails.org/api_app.html

[9] *Gem* é a denominação de um programa ou biblioteca *ruby* disponibilizado para uso pela comunidade de desenvolvedores.

[10] https://www.mongodb.com/

3.3.4 *GEOJSON* [11]

O padrão *GEOJSON* foi adotado por sua facilidade de lidar com objetos geolocalizados. A aplicação receberá dados majoritariamente desta forma. Além disso, com este padrão podemos interagir com diversos sistemas já existentes que fazem sua comunicação por este formato. Facilitando o desenvolvimento de aplicações no *front-end*, e possibilitando que aplicações já existentes passem a utilizar a *API* disponibilizada.

3.4 Utilização do *serviço* para base de dados geolocalizados da saúde

Nesta seção, apresentamos o fluxo dos dados entre os diferentes componentes da arquitetura proposta. Usaremos como exemplo a aplicação *GeoMonitor da Saúde* utilizando a base de dados SIH.

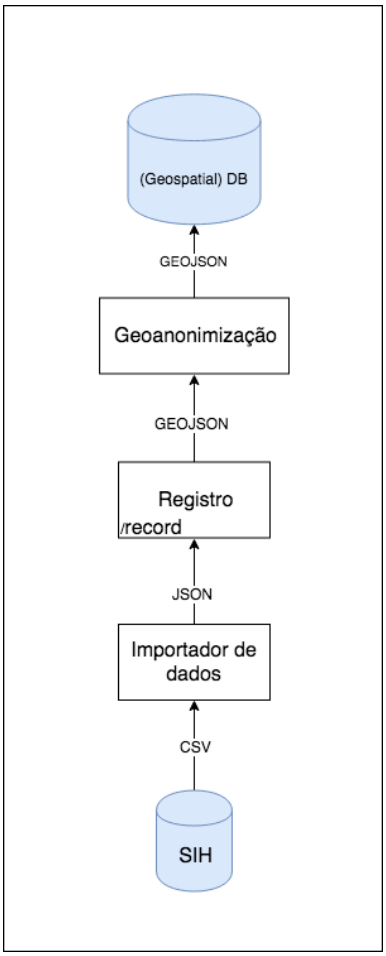

Figura 3.14: *Fluxo de registro de dados na aplicação*

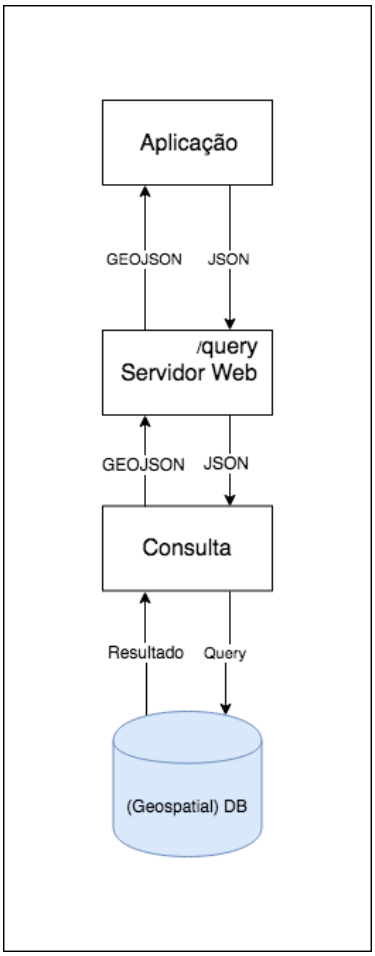

Figura 3.15: *Fluxo de consulta de dados na aplicação*

[11] http://geojson.org/

A Figura 3.14 apresenta a entrada de dados no serviço. Os dados referentes ao SIH estão armazenados no formato *CSV*, eles são lidos pelo *Importador de dados* que cria arquivos *GEOJSON*, e forma requisições para o *endpoint /record*. Nesse ponto, os dados entram no serviço, aqui o módulo de *Geoanonimização* é utilizado para anonimizar os dados que então são salvos na base de dados.

Quando uma aplicação faz uma requisição por dados nos *endpoints /query* ou */group*, temos o fluxo de dados como ilustrado na Figura 3.15. A requisição é validada quanto a seu formato; a *query* é formulada pelo módulo que foi acionado e executada pela instância do banco de dados; o resultado é então formatado em *GEOJSON* e enviado na resposta da requisição.

Capítulo 4

Uma aplicação para visualização espacial dos dados da saúde

Este capítulo descreve uma aplicação [1] desenvolvida para visualizar dados da saúde utilizando a *API* oferecida pelo serviço descrito no capítulo anterior como base para suas análises.

4.1 GeoMonitor da Saúde

A Secretaria Municipal de Saúde recebe uma grande quantidade de dados que descrevem as diversas atuações do Sistema Único de Saúde (SUS) em relação à sociedade. Nesses dados estão informações cruciais para a melhoria na qualidade do sistema de saúde pública da cidade. Para a análise de grandes volumes de dados, necessitamos de ferramentas sofisticadas, uma vez que a mera observação dos dados não atende a esse objetivo. Dentro desse contexto, a aplicação *GeoMonitor da Saúde* foi criada com o objetivo de extrair dos dados informações "sutis" que uma análise crua dos dados não seria capaz de identificar.

A plataforma apresenta os dados georreferenciados das internações custeadas pelo SUS. A ferramenta também apresenta informações demográficas, oriundas do Censo de 2010 realizado pelo IBGE e das projeções populacionais disponibilizadas pela Fundação SEADE[2], estratificadas por sexo, faixa etária e raça/cor segundo as unidades territoriais administrativas da Prefeitura de São Paulo e da Secretaria Municipal da Saúde. Com o GeoMonitor, é possível realizar buscas de internações hospitalares a partir de certos filtros (estabelecimento de ocorrência, subprefeitura, raça/cor, faixa etária, distância, etc.) e observar o mapeamento do resultado ao longo de São Paulo. Além disso, o sistema tem a capacidade de, a partir dos dados georreferenciados, criar clusters e mapas de calor que são apresentados no mapa da cidade.

[1] http://interscity.org/apps/saude

[2] Fundação vinculada à Secretaria de Planejamento e Gestão do Estado de São Paulo, é hoje um centro de referência nacional na produção e disseminação de análises e estatísticas socioeconômicas e demográficas: http://seade.gov.br.

4.2 Dados utilizados

Os dados utilizados foram cedidos pela Secretaria Municipal da Saúde de São Paulo a partir das bases públicas referentes aos Sistemas de Informação de mortalidade (SIM), de Nascidos Vivos (SINASC) e Hospitalar (SIH). Até a presente data, no contexto desta monografia, a SMS-SP nos forneceu a base de dados SIH. Todos os dados fornecidos à equipe do IME-USP foram anonimizados, i.e., não houve informações pessoais sobre pacientes específicos e o domicílio do paciente não foi fornecido na forma de endereço, mas apenas com a identificação do setor censitário correspondente [3].

Esse primeiro conjunto de dados do SIH, fornecido pela SMS-SP, foi dividido entre dados dos estabelecimentos e dados das internações. Dos estabelecimentos de saúde, teremos as informações como CNES, latitude e longitude, quantidade de leitos, telefone, classificação, distrito administrativo, subprefeitura, supervisão técnica de saúde e coordenadoria regional de saúde a qual pertence. Das internações hospitalares teremos dados sobre o setor censitário do paciente, estabelecimento de saúde, especialidade, CID, data, distância viária entre setor censitário do paciente e o estabelecimento de saúde e informações do paciente (Idade, sexo, nível de instrução e grupo étnico).

4.3 Aplicação

Para o desenvolvimento da aplicação, foram adotadas as boas práticas de desenvolvimento de software livre que já vem sendo utilizadas pelo grupo de sistemas do IME-USP nos últimos 20 anos. Uma dessas técnicas, o método ágil [4] de desenvolvimento, foi aplicado da seguinte forma: Iterações de curta duração, normalmente duas semanas, eram planejadas e executadas pelos desenvolvedores. Seguidas de homologação por parte da SMS-SP que verificava o trabalho executado e criava novas demandas. Elaborando com isso, uma nova iteração.

O repositório https://gitlab.com/interscity/health-dashboard/health-smart-city, licenciado sob a Mozilla Public License 2.0 [5], foi criado para o desenvolvimento da aplicação, e sua organização em torno do código-fonte desenvolvido. Nele, está documentado todo o trabalho desenvolvido e todo o trabalho futuro para a evolução da aplicação, assim como reuniões com representantes da SMS-SP que acompanharam o desenvolvimento.

4.3.1 Home

Na página inicial (vide Figura 4.1), temos algumas informações sobre os dados presentes na aplicação e uma lista de atalhos para os serviços disponíveis.

[3] Processo que no futuro poderá ser efetuado pelo módulo de Geoanonimização descrito na Seção 3.2.2
[4] http://www.manifestoagil.com.br/
[5] https://www.mozilla.org/en-US/MPL/2.0/

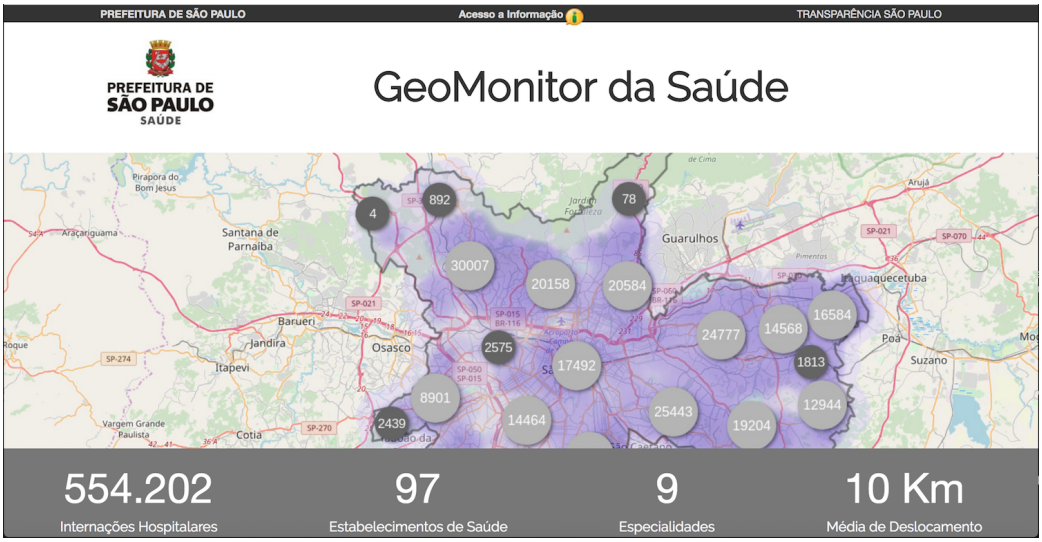

Figura 4.1: *Página Home*

4.3.2 Dados Gerais

Na página de Dados Gerais (vide Figura 4.2), são apresentados ao usuário análises feitas sobre os dados. Dentre elas estão:

- Ranking dos estabelecimentos por internações efetuadas
- Gráfico indicando a quantidade de internações por mês
- Gráfico mostrando a distância viária entre o paciente e o estabelecimento de saúde utilizado por especialidade.
- Gráfico indicando a porcentagem e valor absoluto de internações por especialidade.
- Seção onde o usuário pode escolher uma variável, e a partir dela é gerado um histograma de seus valores.

4.3.3 Busca avançada

Na página de busca avançada (vide Figura 4.3), o usuário pode filtrar os dados a partir dos campos da base de dados. Por exemplo, se o usuário quer apenas as internações em que a especialidade do leito seja “Cuidados Prolongados”, ele pode fazer tal filtro e observar o resultado no mapa onde as internações, que se enquadram na sua busca, serão identificadas.

A partir dos dados filtrados, a plataforma gera duas visualizações distintas: a primeira clusterizando os dados de acordo com sua posição no mapa e a segunda gerando um mapa de calor de acordo com a concentração do pontos no mapa. A partir dessas duas visualizações, é possível identificar regiões da cidade onde um grande número de pessoas utilizaram o SUS pelo motivo especificado nos filtros. Identificando assim, necessidades de recursos hospitalares que não estão sendo atendidas com base nos dados analisados.

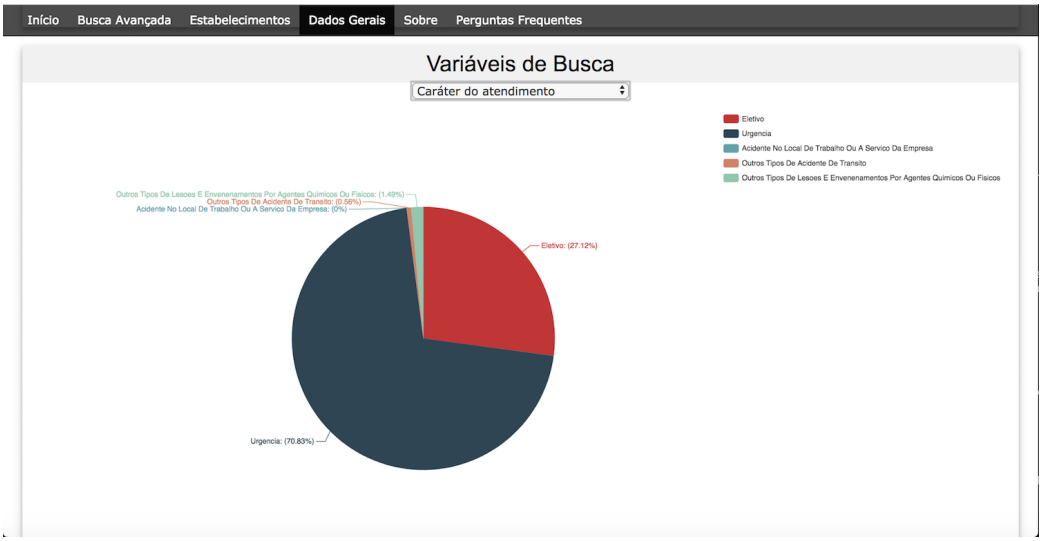

Figura 4.2: *Página de Dados Gerais*

O usuário pode escolher o raio de convergência a ser utilizado pelos algoritmos que irão gerar as visualizações, podendo assim fazer tanto análises locais como análises mais abrangentes.

Também nessa página, o usuário tem a opção de visualizar no mapa os seguintes limites administrativos:

- Município de São Paulo
- Coordenadoria Regional de Saúde
- Supervisão Técnica de Saúde
- Prefeitura Regional
- Distritos Administrativos
- Áreas de Abrangência de UBS
- Áreas de Cobertura da Estratégia Saúde da Família

Com isso um gestor público de Saúde, que atua em alguma dessas áreas, pode visualizar as internações que ocorreram dentro de seu limite de atuação, facilitando o seu processo de tomada de decisão.

Por fim, o usuário tem ainda as seguintes opções de utilização dos dados:

- Download dos dados de acordo com os filtros aplicados
- Gerar a página de Dados Gerais apenas com os dados filtrados
- Imprimir o mapa que está sendo visualizado.

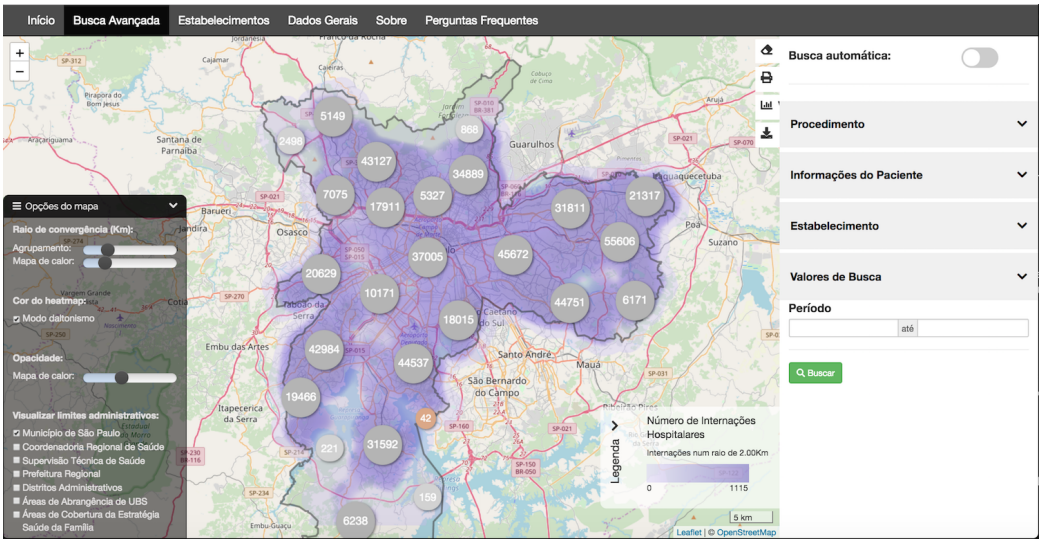

Figura 4.3: *Página de Busca Avançada*

4.3.4 Estabelecimentos

Na página de estabelecimentos (vide Figura 4.4), é apresentado ao usuário todos os estabelecimentos de saúde da cidade de São Paulo que efetuaram internações hospitalares financiadas pelo SUS. Ao selecionar um deles, o usuário tem informações sobre o estabelecimento, bem como a localização no mapa do setor censitário dos pacientes que utilizaram tal estabelecimento. Assim como na página de "Busca Avançada", são apresentados clusters das internações hospitalares e mapa de calor de acordo com a concentração delas.

Para facilitar a visualização, a página também mostra os raios de concentração das internações hospitalares: por exemplo, o primeiro raio indica a região de residência de 25% dos pacientes que se deslocaram até o estabelecimento escolhido; o segundo raio 50% e; o terceiro raio 75%; dessa forma, facilitando a identificação de estabelecimentos em que pacientes necessitam se locomover por maiores distâncias para serem atendidos.

4.3.5 Sobre

Na página Sobre (vide Figura 4.4), é apresentada uma visão geral do projeto, informações sobre o desenvolvimento, dados utilizados e uma pequena apresentação do sistema e dos membros do projeto.

4.3.6 Perguntas Frequentes

Na página de Perguntas Frequentes (vide Figura 4.6), estão esclarecidas dúvidas recorrentes sobre a utilização da aplicação.

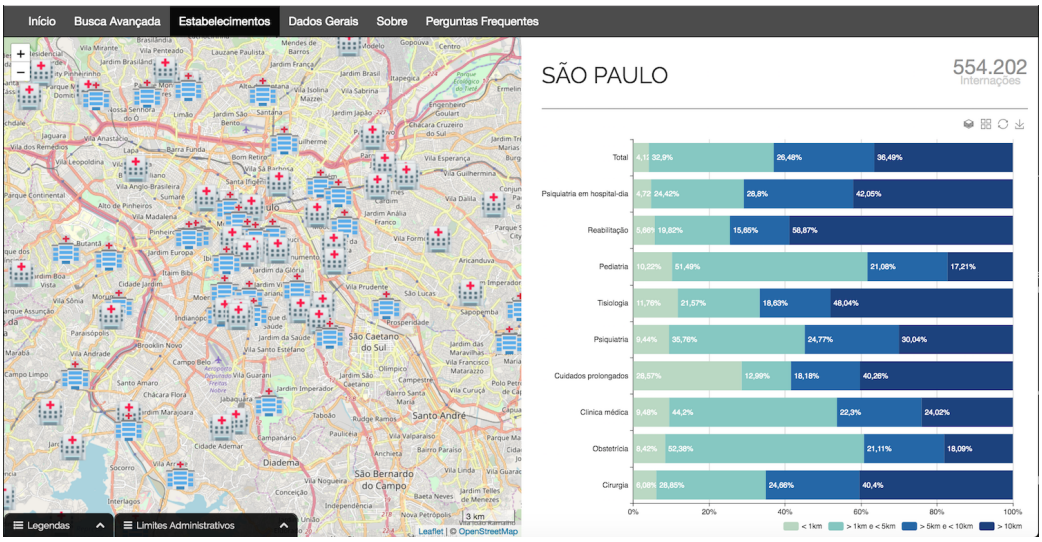

Figura 4.4: *Página de Estabelecimentos*

Figura 4.5: *Página Sobre*

Figura 4.6: *Página de Perguntas Frequentes*

4.4 Detalhes de implementação

Para implementação do GeoMonitor, selecionamos uma série de ferramentas e bibliotecas, sob licença de *Software* Livre, que permitiu o desenvolvimento de uma aplicação de visualização de dados georreferenciados da saúde, de acordo com o que estava disponível no estado-da-prática das tecnologias envolvidas, conforme apresentadas a seguir.

4.4.1 Open Street Maps

Seguindo os princípios que guiaram o desenvolvimento deste projeto, utilizamos a *API* do Open Street Maps [6] para criar suas visualizações. Essa *API* é licenciada como *Software* Livre, desenvolvida por uma grande comunidade. Utilizamos a biblioteca Leaflet [7], também *Software* Livre, para gerir os elementos que são apresentados no mapa.

4.4.2 Cluster

Com um volume muito grande de dados que são recebidos, torna-se necessário uma forma inteligente de apresentá-los. A clusterização é um método muito comum para lidar com esse tipo de problema, de forma que, os dados geolocalizados tornam a visualização dos clusters extremamente intuitiva e informativa. Geramos os clusters utilizando a biblioteca Leaflet.markercluster [8].

[6] http://openstreetmap.org
[7] http://leafletjs.com
[8] https://github.com/Leaflet/Leaflet.markercluster

4.4.3 Mapa de Calor

Outra forma muito comum de visualização de dados são mapas de calor, que facilitam a identificação de áreas com grande concentração de dados. Utilizamos esse método para apresentar ao usuário áreas no mapa de grande concentração de internações. Os mapas de calor são gerados com a biblioteca Heatmap.js [9].

4.4.4 Gráficos

Para criação de gráficos dinâmicos e adaptativos que podem ter seu conteúdo alterado de acordo com a necessidade do usuário, utilizamos a biblioteca echarts [10].

4.4.5 Distância Percorrida

Inicialmente, utilizamos a Distância Euclidiana para realizar as análises da plataforma, porém para aproximar essas análises à realidade, a plataforma foi alterada para calcular a distância viária percorrida pelo paciente, ou seja, a distância de acordo com as vias públicas disponíveis. Isso foi alcançado com a ajuda do *software Open Source Routing Machine (OSRM)* [11], uma aplicação *Software* Livre que, utilizando dados viários públicos, gera rotas de deslocamento similares às encontradas em sistemas proprietários, como o Google Maps [12]. O OSRM possui uma API que aceita requisições com os pontos geográficos de origem e destino respondendo com as informações relevantes (distância, rota, número de ruas, etc).

Utilizando uma instância própria do *software*, geramos primeiramente todos os pares de origem e destino que a base de dados possuía. Como os dados estão geoanonimizados, obtivemos o valor de 166.600 pares distintos, via um *script* para realizar os pedidos e guardar as respostas. Com essa informação, todos os registros da base de dados receberam a distância percorrida de acordo com seus dados.

[9] http://patrick-wied.at/static/heatmapjs/?utm_source=gh
[10] http://echarts.baidu.com
[11] http://project-osrm.org/
[12] https://google.com/maps

4.5 Exemplos de utilização

A partir do **GeoMonitor da Saúde** podemos encontrar padrões que não seriam visíveis sem uma ferramenta de visualização de dados. Alguns exemplos:

- Concentração de casos de Leucemia Línfoide na região leste da cidade de São Paulo.
- Casos de HIV se concentram no centro da cidade.

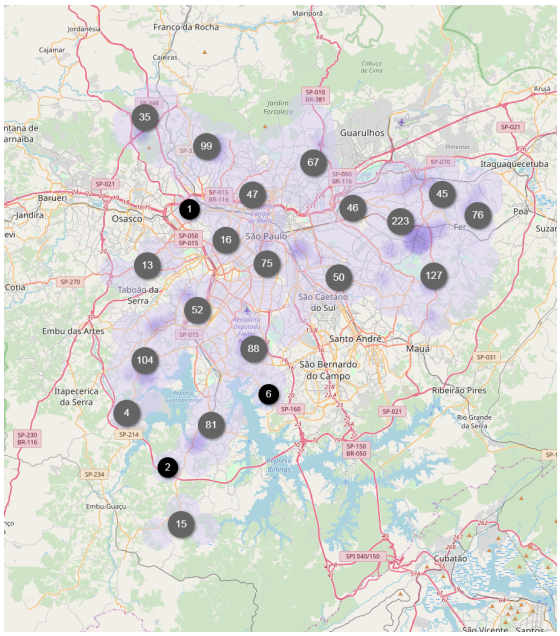

Figura 4.7: *Internações relacionadas a casos de Leucemia Linfóide*

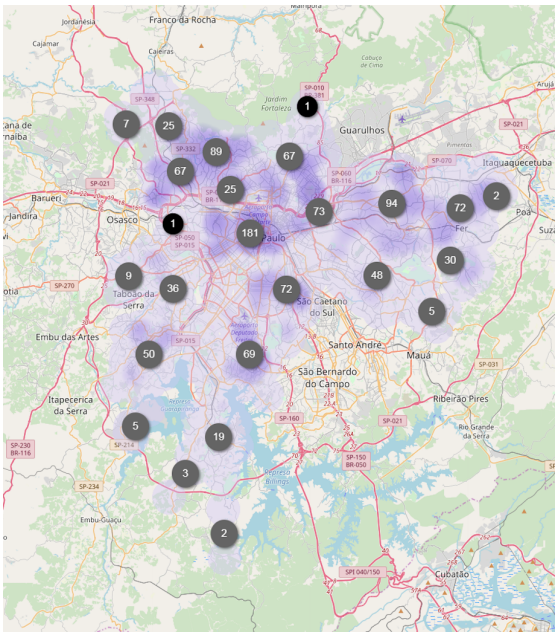

Figura 4.8: *Internações relacionadas a casos de HIV*

A aplicação oferece 25 filtros distintos ao usuário. Tornando possível criar uma grande quantidade de combinações de busca que podem ser usadas para encontrar anomalias nos dados. E assim, fazer um uso mais inteligente dos recursos de saúde.

Capítulo 5

Considerações Finais

A elaboração da aplicação proposta neste trabalho apresentou diversos desafios, entre eles: lidar com a grande quantidade de dados, elaborar formas informativas de visualização, manter uma aplicação *online* com usuários reais, coordenar uma equipe de desenvolvedores, e outros problemas que surgiram ao longo do trabalho. Esses desafios foram superados, resultando na implementação de uma arquitetura para uma plataforma de armazenamento de dados geolocalizados. Essa aplicação é capaz de abstrair detalhes dos dados e oferecer dados processados, de uma forma que facilita a realização de diversas análises visuais, o que possibilitou uma primeira implementação [1] desta arquitetura.

Além disso, a aplicação GeoMonitor da Saúde de visualização de dados, desenvolvida para rodar a partir da implementação desenvolvida durante este trabalho de conclusão de curso, foi um primeiro resultado para colocarmos a plataforma em funcionamento para a SMS-SP, de forma que os técnicos da SMS-SP puderam avaliar, continuamente, cada funcionalidade disponibilizada, ao final de cada ciclo de desenvolvimento.

Caso torne-se um sistema oficial da SMS-SP, a plataforma tem um potencial de impactar a elaboração de políticas públicas na área de Saúde e nas tomadas de decisões quanto ao consumo de serviços financiados pelo Sistema Único de Saúde. Por isso, disponibilizamos como resultado prático desta monografia uma primeira versão de um software livre de alta qualidade e referência para visualização de grandes quantidades de dados sobre serviços de saúde de uma grande metrópole, como a cidade de São Paulo.

O desenvolvimento de software, assim como a interação entre desenvolvedores e usuários, é sempre um desafio. Neste projeto não foi diferente, mas ambos os lados tiveram uma experiência positiva ao superar tal desafio. Do lado das atividades de desenvolvimento foi possível ter um retorno rápido e interação contínua com a SMS-SP, que participou ativamente do processo de desenvolvimento, testando e homologando a aplicação. Com isso, o trabalho desenvolvido neste projeto era constantemente validado, diminuindo a complexidade das mudanças necessárias para atender as demandas de evolução da plataforma proposta. Para a SMS-SP, esta primeira versão disponível em http://interscity.org/apps/saude, atualizada constantemente durante a evolução deste trabalho, permitiu uma aproximação ao processo de desenvolvimento de software, assegurando, assim, que a aplicação proposta

[1] https://gitlab.com/interscity/health-dashboard/datahealth-api

estivesse em uma versão já de acordo com suas necessidades práticas.

Referências

[Agafonkin 5] Vladimir Agafonkin. *Clustering millions of points on a map with Supercluster*. 2016. URL: https://blog.mapbox.com/clustering-millions-of-points-on-a-map-with-supercluster-272046ec5c97 (acesso em 14/11/2018) (citado nas pgs. 6, 7).

[Amatriain *et al.* 5] Xavier Amatriain, Alejandro Jaimes*, Nuria Oliver e Josep M. Pujol. "Data mining methods for recommender systems". Em: *Recommender Systems Handbook*. Ed. por Francesco Ricci, Lior Rokach, Bracha Shapira e Paul B. Kantor. Boston, MA: Springer US, 2011, pgs. 39–71. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_2. URL: https://doi.org/10.1007/978-0-387-85820-3_2 (citado na pg. 3).

[Anttiroiko *et al.*] Ari-Veikko Anttiroiko, Pekka Valkama e Stephen J. Bailey. "Smart cities in the new service economy: building platforms for smart services". Em: *AI & SOCIETY* 29.3 (ago. de 2014), pgs. 323–334. ISSN: 1435-5655. DOI: 10.1007/s00146-013-0464-0. URL: http://dx.doi.org/10.1007/s00146-013-0464-0 (citado na pg. 1).

[Batista *et al.* 5] D. M. Batista *et al.* "Interscity: addressing future internet research challenges for smart cities". Em: *7th International Conference on the Network of the Future*. IEEE, nov. de 2016 (citado na pg. 1).

[Cassiano 5] Keila Mara Cassiano. "Análise de séries temporais usando análise espectral singular (ssa) e clusterização de suas componentes baseada em densidade". Tese de doutorado. Pontifícia Universidade Cat ólica do Rio de Janeiro - PUC-Rio, 2015. URL: https://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=24787@1&msg=28# (citado nas pgs. 4, 5).

[Cooper e Sahami] Steve Cooper e Mehran Sahami. "Reflections on stanford's moocs". Em: *Commun. ACM* 56.2 (fev. de 2013), pgs. 28–30. ISSN: 0001-0782. DOI: 10.1145/2408776.2408787. URL: http://doi.acm.org/10.1145/2408776.2408787 (citado na pg. 1).

[Hu *et al.*] H. Hu, Y. Wen, T. S. Chua e X. Li. "Toward scalable systems for big data analytics: a technology tutorial". Em: *IEEE Access* 2 (2014), pgs. 652–687. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2332453 (citado nas pgs. 1, 3).

[Jagadish *et al.*] H. V. Jagadish *et al.* "Big data and its technical challenges". Em: *Commun. ACM* 57.7 (jul. de 2014), pgs. 86–94. ISSN: 0001-0782. DOI: 10.1145/2611567. URL: http://doi.acm.org/10.1145/2611567 (citado na pg. 1).

[Jain *et al.* 5] A. K. Jain, M. N. Murty e P. J. Flynn. "Data clustering: a review". Em: *ACM Comput. Surv.* 31.3 (set. de 1999), pgs. 264–323. ISSN: 0360-0300. DOI: 10.1145/331499.331504. URL: http://doi.acm.org/10.1145/331499.331504 (citado nas pgs. 2–6).

[Kon e Blair] Fabio Kon e Gordon Blair. "The internet's deep impact—letter from the editors-in-chief". Em: *Journal of Internet Services and Applications* 2.1 (jul. de 2011), pgs. 1–2. ISSN: 1869-0238. DOI: 10.1007/s13174-011-0022-2. URL: http://dx.doi.org/10.1007/s13174-011-0022-2 (citado na pg. 1).

[Keim *et al.* 5] Daniel Keim, Huamin Qu e Kwan-Liu Ma. "Big-data visualization". Em: *IEEE Computer Graphics and Applications* 33.4 (2013), pgs. 20–21. ISSN: 0272-1716. DOI: 10.1109/MCG.2013.54 (citado na pg. 2).

[Nations] United Nations. *World Urbanization Prospects: The 2014 Revision*. Rel. téc. ST/ESA/SER.A/352. New York: Department of Economic e Social Affairs of the United Nations, 2014 (citado na pg. 1).

[Sheng *et al.*] X. Sheng, J. Tang, X. Xiao e G. Xue. "Sensing as a service: challenges, solutions and future directions". Em: *IEEE Sensors Journal* 13.10 (out. de 2013), pgs. 3733–3741. ISSN: 1530-437X. DOI: 10.1109/JSEN.2013.2262677 (citado na pg. 1).

[Wang *et al.* 5] Xin Wang, Wei Gu, Danielle Ziebelin e Howard Hamilton. "An ontology-based framework for geospatial clustering". Em: *International Journal of Geographical Information Science* 24.11 (2010), pgs. 1601–1630. DOI: 10.1080/13658811003702147. eprint: https://doi.org/10.1080/13658811003702147. URL: https://doi.org/10.1080/13658811003702147 (citado na pg. 6).

[Wasserman] Anthony I. Wasserman. "How the internet transformed the software industry". Em: *Journal of Internet Services and Applications* 2.1 (jul. de 2011), pgs. 11–22. ISSN: 1869-0238. DOI: 10.1007/s13174-011-0019-x. URL: http://dx.doi.org/10.1007/s13174-011-0019-x (citado na pg. 1).

[Zhang *et al.*] Qi Zhang, Lu Cheng e Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges". Em: *Journal of Internet Services and Applications* 1.1 (maio de 2010), pgs. 7–18. ISSN: 1869-0238. DOI: 10.1007/s13174-010-0007-6. URL: http://dx.doi.org/10.1007/s13174-010-0007-6 (citado na pg. 1).

[Zorzi *et al.*] M. Zorzi, A. Gluhak, S. Lange e A. Bassi. "From today's intranet of things to a future internet of things: a wireless- and mobility-related view". Em: *IEEE Wireless Communications* 17.6 (dez. de 2010), pgs. 44–51. ISSN: 1536-1284. DOI: 10.1109/MWC.2010.5675777 (citado na pg. 1).