UNIVERSIDADE FEDERAL DO PARANÁ

RAPHAEL GONÇALVES DE CAMPOS

REPRESENTAÇÃO AUTOMATIZADA DE GEO BIG DATA EM MAPAS COROPLÉTICOS

CURITIBA

2017

RAPHAEL GONÇALVES DE CAMPOS

REPRESENTAÇÃO AUTOMATIZADA DE GEO BIG DATA EM MAPAS COROPLÉTICOS

Dissertação apresentada como requisito parcial à obtenção do grau de mestre em Ciências Geodésicas, no curso de pós-graduação em Ciências Geodésicas, setor de ciências da terra, da Universidade Federal do Paraná.

Orientadora: Profª. Drª. Silvana Philippi Camboim

CURITIBA

2017

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Campos, Raphael Gonçalves de
Representação automatizada de Geo Big Data em mapas coropléticos / Raphael Gonçalves de Campos. – Curitiba, 2017.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciências da Terra, Programa de Pós-Graduação em Ciências Geodésicas.

Orientador: Silvana Philippi Camboim

1. Big Data. 2. Dados geoespaciais. 3. Mapas coropléticos. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Ciências Geodésicas. III. Camboim, Silvana Philippi. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS DA TERRA
Programa de Pós-Graduação CIÊNCIAS GEODÉSICAS

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em CIÊNCIAS GEODÉSICAS da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **RAPHAEL GONÇALVES DE CAMPOS** intitulada: **REPRESENTAÇÃO AUTOMATIZADA DE GEO BIG DATA EM MAPAS COROPLÉTICOS**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 31 de Julho de 2017.

SILVANA PHILIPPI CAMBOIM
Presidente da Banca Examinadora (UFPR)

ANDRE LUIZ ALENCAR DE MENDONCA
Avaliador Externo (UEA)

KARINE REIS FERREIRA GOMES
Avaliador Externo (INPE)

CLAUDIA ROBBI SLUTER
Avaliador Interno (UFPR)

AGRADECIMENTOS

Agradeço à minha família pelo suporte e incentivo.

Agradeço aos meus colegas de laboratório pelo companheirismo e apoio.

Agradeço a minha orientadora Silvana Philippi Camboim pelos conhecimentos transmitidos e paciência.

Agradeço a todos os professores que de alguma forma me transmitiram conhecimentos.

Agradeço à Universidade Federal do Paraná pela oportunidade da formação acadêmica.

Agradeço à CAPES pela bolsa que me possibilitou ter dedicação exclusiva

Agradeço a todos aqueles que de alguma forma contribuíram para que esta etapa fosse concluída.

RESUMO

Os desenvolvimentos tecnológicos das últimas décadas impulsionaram um aumento considerável na produção de dados geoespaciais, não somente em volume, mas, também em velocidade e diversidade de dados. Estes dados são denominados de Geo Big Data e adicionaram novos desafios no armazenamento, processamento e análise, requisitando novos métodos e tecnologias para manipulação de dados. Grandes esforços estão sendo feitos para a manipulação do Geo Big Data, como o desenvolvimento de bancos de dados NoSQL, que são mais eficientes no tratamento das características desses dados. Algumas limitações computacionais em analisar e obter conhecimento do Geo Big Data podem ser superadas pela geovisualização. No entanto, devido nossas limitações visuais, devem ser utilizados métodos de representação que as levem em consideração. Neste trabalho, foi desenvolvido um sistema web, baseado em tecnologias livres, que trata os dados pontuais armazenados em banco de dados NoSQL para serem representados em mapas coropléticos, ou seja, agrega e conta os pontos por divisões territoriais, que estão armazenados em banco de dados tradicionais, e em seguida normaliza o resultado da contagem pela população residente em cada unidade territorial, assim, transforma-se a primitiva gráfica de pontual para área e obtém-se dados com o nível de medida numérico independentemente do valor de área de cada unidade. O sistema permite que o usuário defina parâmetros para a construção dos mapas coropléticos e produza diferentes visualizações dos dados, além de modificar automaticamente a unidade territorial que os mapas coropléticos são construídos conforme o nível de *zoom* que os dados estão sendo visualizados, assim, adequa-se a unidade territorial que é possível identificar a tal nível de zoom. Alcançamos um sistema que permite explorar dados pontuais de maneira a favorecer nossa capacidade de processamento visual.

PALAVRAS-CHAVES: Big Data. Geo Big Data. NoSQL. Mapa Coroplético.

ABSTRACT

The technological developments of the last decades had stimulated a considerable increase in the production of geospatial data, not only in volume but, also in speed and diversity of data. These data are called Geo Big Data and had added to new challenges in the storage, processing, and analysis, requesting new methods and technologies for the manipulation of data. Great efforts are being made for the manipulation of Geo Big Data, as the development of NoSQL databases, that are more efficient in the treatment of the characteristics of these data. Some computational limitations in analyzing and getting knowledge of the Geo Big Data can be surpassed by the geovisualization. However, it had our visual limitations, representation methods must be used that consider them. In this work, has developed a web system, based on free technologies, that treat the point data stored in NoSQL database to be represented in choropleth maps, that is, counts the points for territorial divisions that are stored in the traditional data base, and after that normalizes the result of the counting for the resident population in each territorial unit, thus, changed it graphical primitive of point for the area and is gotten given with the level of independent numerical measure of the area value of each unit. The system allows the user to define parameters for the construction of the choropleth maps and produces different visualizations of the data, beyond automatically modifying the territorial unit that the choropleth maps are constructed in agreement with the level of zoom that the data are being visualized, thus, adjusts it a territorial unit that is possible to identify to such level of zoom. We reach a system that allows exploring given prompts in a way to favor our capacity for visual processing.

KEYWORDS: Big Data, Geo Big Data, NoSQL, Choropleth Map.

LISTA DE SIGLAS

CONCAR - Comissão Nacional de Cartografia

CSV - *Comma-Separated Values*

EDGV - Estruturação de Dados Geoespaciais Vetoriais

GADF – *Goodness o*f Absolute Deviation

HTML - Hypertext Markup Language

ICA - *International Cartographic Association*

INDE - Infraestrutura Nacional de Dados Espaciais

IPPUC - Instituto de Pesquisa e Planejamento Urbano de Curitiba

JSON - *Javascript Object Notation*

ORDBMS - *Object-Relational Database Management System*

OSM – *OpenStreetMap*

RBDMS - *Relational Database Management System*

SQL - *Structured Query Language*

SGBD - Sistema de Gerenciamento de Banco De Dados

TMS – *Tile Map Service*

UML - *Unified Modeling Language*

VGI - *Volunteered Geographic Information*

XML - *Extensible Markup Language*

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas os avanços das tecnologias geoespaciais impulsionaram a produção de geoinformação, reforçando sua importância em diversos campos do conhecimento e criando novos desafios para sua manipulação (LI et al., 2016). Os sistemas de geoinformação são responsáveis por gerenciar e permitir a manipulação da geoinformação para diferentes propósitos. Segundo Sluter et al. (2014), há um conjunto de componentes básicos que fazem parte de qualquer sistema de geoinformação e podem ser divididos em três grupos: componentes de geovisualização, componentes de banco de dados geográficos e componentes de geoinformação.

Quanto ao componente de banco de dados geográficos, desde a década de 1980s os bancos de dados relacionais têm sido a principal tecnologia de armazenamento de diversos tipos de dados (SILBERSCHATZ; KORTH; SUDARSHAN, 2006). No contexto espacial, após o final dos anos 1990, os SGBD tornaram-se capazes de manipular a informação espacial eficientemente (QUEIROZ; MONTEIRO; CÂMARA, 2012). Como exemplo, podemos citar o PostGIS, uma extensão espacial para o SGBD PostgreSQL, foi lançado inicialmente em maio de 2001 e se encontra atualizado até hoje (POSTGIS, 2016).

Com o advento da web 2.0, os usuários passaram de apenas consumidores à produtores de dados, e com o desenvolvimento de novas plataformas móveis, redes sociais e outras tecnologias, a quantidade de dados gerados aumentou exponencialmente. Essa quantidade massiva é denominada de “Big Data” (GOODCHILD, 2016). O termo apareceu em meados dos anos 1990 (LI et al, 2016) e foi caracterizado partir de três dimensões: Volume, Velocidade e Variedade. (LANEY, 2001). O volume é referente à quantidade de dados gerados; velocidade é quão rápido os dados são gerados ou atualizados; e a variedade diz respeito aos diferentes tipos de dados envolvidos. Uma grande quantidade desses dados advém de sensores que tem sua localização definida ou podem ser georreferenciados direta ou indiretamente (HAHMANN e BURGHARDT, 2012), denominados de *Geospatial Big Data* (ROBINSON et al., 2017; LI et al., 2016; SHEKHAR et al., 2012) e referidos neste trabalho, simplificadamente, como Geo Big Data.

As características do Geo Big Data influenciam diretamente as componentes de armazenamento de um sistema de geoinformação. Essas características excedem as capacidades tradicionais dos SGBD espaciais (SHEKHAR et al. 2012), que são: a escalabilidade horizontal, ou seja, a capacidade de adicionar novos servidores para melhorar o desempenho do sistema (QUEIROZ; MONTEIRO; CÂMARA, 2012); o armazenamento e gerenciamento de grandes bancos de dados; e a eficiência das consultas devido à grande quantidade de dados (ABRAMOVA; BERNARDINO; FURTADO, 2014).

Os SGBD não relacionais, chamados de NoSQL, podem ser uma alternativa para aplicações que manuseiam o Geo Big Data, pois tem alta escalabilidade horizontal e podem manusear grande quantidade de dados com eficiência (LIZARDO; MORO; DAVIS JR, 2014). Usualmente, esses sistemas não definem a estrutura dos dados por esquemas explícitos, proporcionando flexibilidade no armazenamento, uma vez que permitem armazenar os dados da maneira desejada, sem adotar uma estrutura pré-definida.

Muitos desses sistemas têm suporte espacial ou alguma extensão que possibilita a manipulação de dados geográficos, no entanto, as funções disponíveis são limitadas. Entre elas, destacam-se, o suporte a diversos tipos de Sistemas de Referência Geodésicos; uso de diferentes projeções cartográficas; conexão com servidores de mapas; suporte a arquivos matriciais; modelos topológicos; entre outros (QUEIROZ; MONTEIRO; CÂMARA, 2012).

Os sistemas NoSQL vêm ganhando notoriedade frente aos desafios impostos pelo Big Data. Porém, muitos produtores de grandes quantidades de dados geoespaciais, como órgãos governamentais, utilizam bancos de dados tradicionais para armazenamento dos seus dados geoespaciais (CAMBOIM, 2013). Em 2008, o governo brasileiro instituiu a INDE (Infraestrutura Nacional de Dados Espaciais), que propõe um conjunto integrado de tecnologias; políticas; mecanismos e procedimentos de coordenação e monitoramento; padrões e acordos, para a manipulação dos dados geoespaciais oficiais (BRASIL, 2008). Umas das normas da INDE, a ET-EDGV (Especificações Técnicas para estruturação de dados geoespaciais digitais vetoriais) utiliza modelagem dos dados espaciais para bancos de dados orientados a objeto (CONCAR, 2007). Devido a essas diferentes formas de armazenamento, a aquisição do conhecimento através da exploração e análise de

dados geoespaciais pode ser potencializada por meio da integração dos dados armazenados em diferentes tipos de bancos de dados.

A velocidade e a grande quantidade de dados geoespaciais gerados todos os dias traz a oportunidade de se representar grande parte do mundo real e seus fenômenos com alta taxa de atualização e validade temporal. Para que os dados sejam úteis, entretanto, a construção de mapas a partir destes, deve ser possível, para que a derivação de informação seja eficaz (ROBINSON et al, 1995).

Uma das formas de se obter informação através da exploração visual dos mapas é a geovisualização, processo que utiliza as capacidades de processamento humano, associados a visão, para obter informação de representações cartográficas (MACHEAREN, 1992). No contexto do Geo Big Data, a geovisualização poderia permitir a sua análise, quando os computadores não são capazes de apresentar soluções diretas (LI et al, 2016).

Visualização geográfica pode ser definida como o uso de representações visuais concretas - em papel ou através de monitores de computador ou outras mídias - para tornar visíveis os contextos e problemas espaciais, de modo a envolver as mais poderosas habilidades de processamento de informações humanas, aquelas associadas a visão

Um grande desafio do Geo Big Data, ao representar quantidades massivas de dados em escala reduzida, está justamente na sua apresentação visual, uma vez que a formação de imagens mentais não é favorecida. No caso de dados pontuais, podemos citar dois principais problemas: coalescência e sobreposição.

A sobreposição ocorre quando os dados têm as mesmas coordenadas. Em consequência das suas precisões ou no caso de dados geocodificados, estes podem ser associados à um mesmo endereço. Já a coalescência se dá quando a distância entre os pontos, representados em uma mesma escala, é menor que a resolução da tela dos dispositivos. Neste caso, os símbolos dos pontos se tocam no processo de representação, ocasionando a supracitada condição geométrica (MCMASTER E SHEA, 1992).

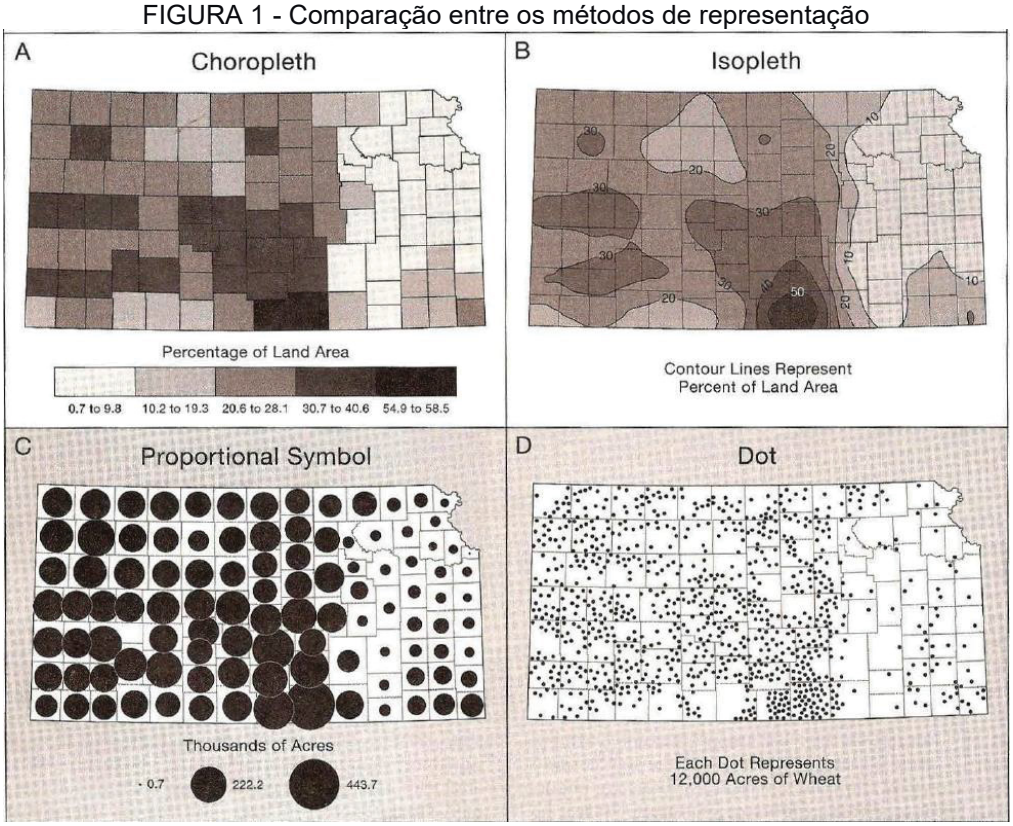
Admitindo que os problemas da representação dos dados pontuais do Geo Big Data excedem as limitações da nossa capacidade visual para exploração e análise dos dados, evidencia-se a necessidade de métodos de representação que levem em consideração o conhecimento sobre limites do nosso processamento de

informação, cognição e percepção (LI et al., 2016). Assim, esses métodos devem nos permitir visualizar padrões e diferenças, bem como derivar os significados de conjuntos de dados maciços e complexos (ROBINSON et al., 2017). Empregar os conceitos e métodos já estabelecidos da cartografia, para enfatizar padrões espaciais de um ou mais atributos geográficos, gerando assim, mapas temáticos dos dados oriundos do Geo Big Data, poderia contribuir para esse objetivo.

Dentre as técnicas de representação temática, podemos citar quatro comumente empregadas: pontos de contagem, símbolos pontuais proporcionais, isarítmica e coroplética (SLOCUM et al., 2009).

Mapas de pontos de contagem definem um único ponto para representar certa quantidade de fenômenos, posicionando-o no local de maior probabilidade de acontecer (SLOCUM, 2008). Um mapa isarítmico é a representação plana através de isolinhas de uma superfície tridimensional, que pode ser uma superfície real, como o relevo, ou uma superfície abstrata ou conceitual (DENT, 1999). O método coroplético é comumente utilizado para retratar dados coletados por unidades de enumeração. Para construir um mapa coroplético, os dados das unidades de enumeração geralmente são agrupados em classes e uma cor é atribuída a cada classe (SLOCUM et al., 2009). E por fim, os mapas de símbolos pontuais proporcionais são construídos ampliando os símbolos proporcionalmente à amplitude da ocorrência do fenômeno. A figura 1 retrata os diferentes métodos de representação supracitados.

FIGURA 1 - Comparação entre os métodos de representação



FONTE: Adaptado de Slocum et al. (2009)

No entanto, se adequarmos as quantidades massivas de dados pontuais para representá-los com as técnicas, símbolos pontuais proporcionais (Figura 1-C) e pontos de contagem (Figura 1-D), as características dessas técnicas podem gerar efeitos indesejáveis na representação, como omitir a ocorrência dos fenômenos em algumas áreas. No caso da representação isarítmica, mais especificamente nos mapas isopléticos, os dados não podem ter variações bruscas para que possam ser abstraídos como uma superfície tridimensional contínua (DENT, 1999). Já nos mapas coropléticos, se agregarmos os dados por unidades de enumeração, os mapas não representam a variação que pode ocorrer dentro das unidades de enumeração. Uma consideração importante na construção dos mapas coropléticos, é a necessidade de normalização dos dados, para que dados totais sejam ajustados ao tamanho das unidades de enumeração (SLOCUM et al.,2009). Por outro lado, isto pode ser conveniente, pois necessita a integração com outras fontes de dados, permitindo derivar diferentes informações.

Portanto, considerando-se a problemática exposta, este trabalho investiga uma metodologia para solucionar as limitações da visualização de dados pontuais do Geo Big Data em diversas escalas empregando o método coroplético.

1.1 HIPÓTESE

Para favorecer a exploração e análise visual dos dados, algumas complexidades dos dados pontuais do Geo Big Data podem ser tratadas com a generalização automática da dimensão do fenômeno, conforme as características dos dados a serem representados e em conformidade com os conceitos de comunicação cartográfica. Se a distribuição dos dados pode ser delimitada por regiões geográficas, então os dados podem ser simplificados ao número de ocorrências dentro de determinada região, tornando a área como dimensão do fenômeno e numérico o nível de medida do fenômeno, assim o fenômeno pode ser normalizado para não ser influenciado pelo valor da sua área e representado pelo método coroplético.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O objetivo deste trabalho é desenvolver um sistema que permita a criação de mapas coropléticos com base em informação espacial armazenada em bancos de dados não relacionais, de forma a contribuir para a proposição de soluções de representação automatizada de Geo Big Data.

1.1.2 Objetivos específicos

1) Entender os conceitos do Geo Big Data e dos bancos de dados NoSQL envolvidos na definição da arquitetura do sistema;
2) Projetar a arquitetura do sistema;
3) Construir o banco de dados não relacional (NoSQL);
4) Desenvolver a solução para construção dos mapas coroplético;
5) Desenvolver a interface;

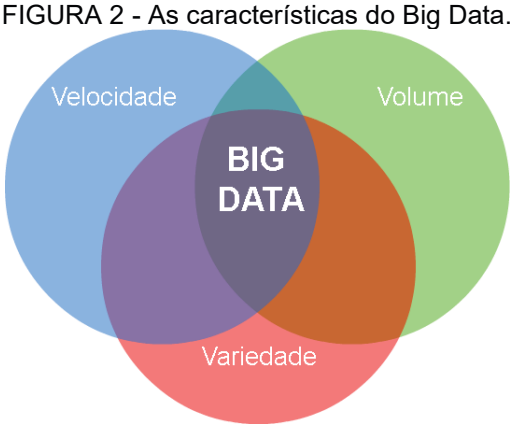
2 BIG DATA E GEO BIG DATA

Nos últimos anos, a quantidade de dados produzidos e compartilhados via web teve um crescimento elevado. Segundo uma pesquisa realizada pela universidade de Berkeley, em 2002 foram gerados 5 exabytes de dados (KIRSTEN SWEARINGEN, 2003). Já, em 2012 foram gerados 2,5 exabytes por dia (IBM, 2012), ou seja, a cada dois dias foram gerados a mesma quantidade de dados de um ano.

Entre os fatores que motivaram o aumento da produção de dados estão as mudanças nas práticas de desenvolvimento Web. Estas práticas modificaram a forma de interação dos usuários com as plataformas web, que se denominou Web 2.0 (OREILLY, 2005). Devido a isto, os usuários deixaram de ser apenas consumidores para também produzir dados;

A grande quantidade de dados, de diferentes tipos e fontes e com grande velocidade em sua geração é denominada de Big Data. O termo apareceu no meio científico na década de 1990s (LI et al, 2016) e posteriormente ganhou popularidade, atraindo o interesse das indústrias e agências governamentais, devido ao seu potencial (CHEN; MAO; LIU, 2014). Laney (2002) caracterizou o Big Data a partir de três dimensões (figura 2): Volume, Velocidade e Variedade.

FIGURA 2 - As características do Big Data.



FONTE: O autor (2016)

- **Volume –** Refere-se à quantidade de dados acumulados, resultante do armazenamento persistente dos dados, como imagens de satélites, redes sociais, VGI, entre outros (RAMOS et al., 2016).

- **Velocidade -** É o aumento do volume de dados por um intervalo de tempo (RAMOS, 2016). Esta característica também engloba a velocidade que o processamento dos dados deve ter para atender a demanda da disponibilidade dos dados quase em tempo real (DASGUPTA, 2013).
- **Variedade -** Indica vários tipos de dados (CHEN; MAO; LIU, 2014), vetoriais, raster, e-mails, geotags, vídeos, fotos, áudios, dados estruturados, semi ou não estruturados, uma grande diversidade de dados.

O Geo Big Data é a parcela do Big Data que possui uma componente espacial associada (DASGUPTA,2013), resulta, portanto, do desenvolvimento das tecnologias geoespaciais. Estimativas do McKinsey *Global Institute*, mostram que em 2009 foram gerados um Petabyte de dados de localização pessoal, e estimava-se um aumento de 20% ao ano, conforme DOBBS et al. (2011). Além dos dados que possuem uma componente espacial, muitos podem ser georreferenciados, direta ou indiretamente. Como exemplo, Hahmann e Burghardt (2012) evidenciaram que 56% a 59% dos artigos da Wikipédia alemã detém informações que permitem seu georreferenciamento.

As características do Geo Big Data desafiam o armazenamento, gerenciamento, processamento, análise e a visualização dos dados (LI et al, 2016). Segundo Shekhar et al (2012), o volume, a velocidade e a variedade excedem as capacidades dos SGBD espaciais tradicionais de manipular o Geo Big Data. Além disto, ao representar uma quantidade massiva de dados em uma escala reduzida sobrecarrega a superfície de representação, excedendo as limitações da nossa capacidade visual para exploração e análise dos dados.

3 BANCO DE DADOS NÃO RELACIONAIS (NOSQL)

Na literatura não é bem definido um significado para o termo NoSQL. A definição mais aceita para NoSQL é “não somente SQL”. (CATTEL,2010; TIWARI, 2011; QUEIROZ; MONTEIRO; CÂMARA, 2012). Este termo é utilizado para denominar bancos de dados que não utilizam o modelo relacional. Embora o termo seja relativamente novo, os bancos de dados não relacionais datam da época dos primeiros computadores (TIWARI, 2011). Como exemplo, temos os bancos de dados em redes e hierárquicos (SILBERSCHATZ; KORTH; SUDARSHAN, 2006).

No entanto, após um longo período de domínio dos bancos de dados relacionais, novos sistemas de bancos de dados não relacionais ressurgiram. Uma vez que os bancos de dados tradicionais não são eficientes para gerenciar o aumento da variedade, quantidade e velocidade da produção de dados, decorrente do advento da Web 2.0 (CATTEL, 2010; BAPTISTA et al., 2016).

Os tradicionais sistemas de bancos de dados têm limitações quanto à: escalabilidade horizontal (QUEIROZ; MONTEIRO; CÂMARA, 2012; BAPTISTA et al., 2016); o armazenamento e gerenciamento de grandes bancos de dados; e a eficiência das consultas devido à grande quantidade de dados (ABRAMOVA; BERNARDINO; FURTADO, 2014). Contudo, as principais características dos sistemas de banco de dados NoSQL, são:

- Alta escalabilidade horizontal (BAPTISTA et al., 2016; LIZARDO; MORO; DAVIS JR, 2014);
- Foram construídos para gerenciamento de grande quantidade de dados (LIZARDO; MORO; DAVIS JR, 2014).
- Não é necessária a definição de esquemas fixos para o armazenamento dos dados (QUEIROZ; MONTEIRO; CÂMARA, 2012);
- Habilidade de particionamento e replicação dos dados, provendo alta disponibilidade (BAPTISTA et al., 2016; (LIZARDO; MORO; DAVIS JR, 2014); QUEIROZ; MONTEIRO; CÂMARA, 2012);

3.1 CLASSIFICAÇÃO DOS SISTEMAS NOSQL

No meio comercial há diversos sistemas NoSQL, ambos têm características de modelo de dados e estratégia de armazenamento similares e são classificados segundo essas características (QUEIROZ; MONTEIRO; CÂMARA, 2012). Os sistemas são comumente classificados em quatro categorias: *Document Store, Key-Value Store, Column Family Store* e *Graph Database*.

- **Key - Value Store –** Armazena os dados na forma de um valor (figura 3) e indexa com uma chave para recuperá-lo. Os softwares Berkeley DB (ORACLE, 2016) e Level DB (JULIAN GRUBER, 2016) são exemplos de sistemas que utilizam esse modelo de armazenamento.

FIGURA 3 - Representação do modelo de dados key-value stores

FONTE: https://biauthority.files.wordpress.com/2012/12/image024.jpg

- **Document Store** – São sistemas projetados para armazenar documentos em formatos padronizados (figura 4), como XML e Json. (ABRAMOVA; BERNARDINO; FURTADO, 2014). Os softwares MongoDB e CouchDB enquadram-se nesse modelo de armazenamento.

FIGURA 4 - Modelo de documento do MongoDB

FONTE: https://docs.mongodb.com/manual/core/data-modeling-introduction/

- **Graph Database -** Esse tipo de banco de dados NoSQL armazena os dados e suas relações como nós e arestas de um grafo (figura 5) (BAPTISTA et al., 2016). Os softwares Neoj4 e OrientDB (ORIENTDB, 2016), são exemplos de sistemas que utilizam esse modelo de armazenamento.

FIGURA 5 - Modelo de dados json que pode ser importado para o OrienteDB.

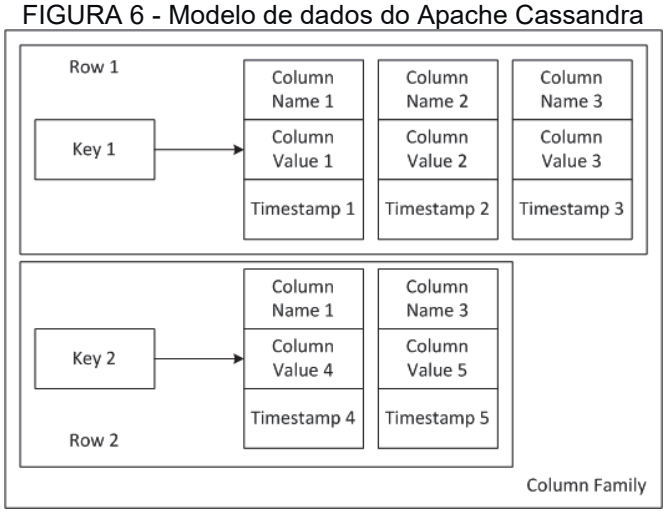
```
{
    "name"      : "Jay",
    "surname"   : "Miner",
    "job"       : "Developer",
    "creations" : [
        {
            "name"    : "Amiga 1000",
            "company" : "Commodore Inc."
        }, {
            "name"    : "Amiga 500",
            "company" : "Commodore Inc."
        }
    ]
}
```

FONTE: http://orientdb.com/docs/last/concepts.html

- **Column Family -** Os dados são armazenados como um conjunto de linhas e colunas (ABRAMOVA; BERNARDINO; FURTADO, 2014), em que cada coluna é formada por nome da coluna e valor, as colunas são agrupadas conforme seus relacionamentos formando uma linha que é identificada através de uma chave (QUEIROZ; MONTEIRO; CÂMARA, 2012) e uma família de colunas é formada por um grupo de linhas, semelhante às tabelas do modelo relacional. A figura 6

representa o modelo de dados do banco de dados Apache Cassandra, o qual utiliza o modelo Column Family (figura 6).

FIGURA 6 - Modelo de dados do Apache Cassandra



FONTE https://10kloc.wordpress.com/category/cassandra-2

3.2 SUPORTE GEOESPACIAL DOS SISTEMAS NOSQL

Os primeiros bancos de dados NoSQL não foram desenvolvidos para manusear dados geoespaciais (BAPTISTA et al., 2016). Com a evolução dos serviços baseados em localização, os sistemas começaram a ser desenvolvidos buscando dar suporte a este tipo de dado (BAAS, 2012). Sendo que, atualmente, alguns softwares possuem suporte nativo para dados geoespaciais. Como exemplo, pode-se citar o MongoDB e o Neo4j e sua extensão Neo4j-Spatial (NEO4J, 2016).

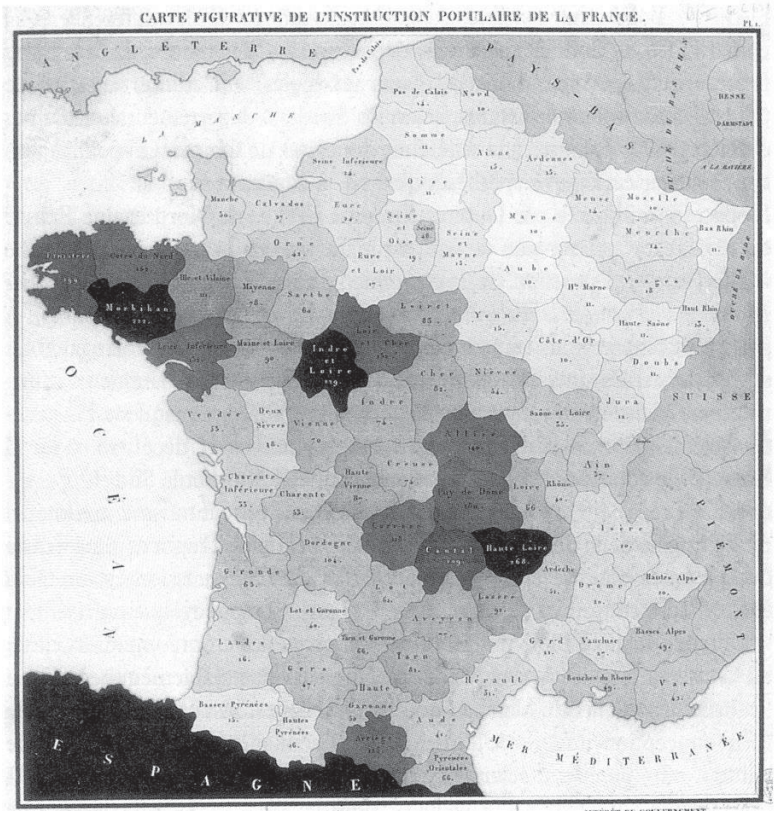
Em trabalhos realizados por Baptista *et al.* (2014) e Queiroz *et al.* (2012), foram levantadas as características do suporte espacial de alguns sistemas. As principais limitações ou inexistências de funções, são: consultas topológicas; operadores espaciais; funções de medidas; projeções cartográficas e reprojeção; suporte a diferentes sistemas geodésicos de referência; formatos de entrada e saída de dados; entre outros. Por apresentar diversas funções de mensuração e consultas topológicas, além de permitir a importação de arquivos Shapefile e OSM, o sistema melhor avaliado nas duas pesquisas, foi o Neo4j.

4 MÉTODO DE REPRESENTAÇÃO COROPLÉTICA

Mapas temáticos são representações de um ou mais atributos geográficos, com o propósito de enfatizar padrões espaciais destes atributos. Estes mapas podem ser utilizados de três maneiras básicas: prover informações específicas de locais específicos, fornecer informações gerais sobre padrões espaciais e comparar padrões entre dois ou mais mapas. Para construir mapas temáticos empregam-se diversas técnicas, dentre as quais, o método coroplético, possivelmente, é o mais utilizado (SLOCUM et al., 2009).

O primeiro mapa coroplético, atribui-se ao Barão Charles Dupin, que, em 1819, representou a distribuição de analfabetismo na França, utilizando um sombreamento gradual de branco a preto (Figura 7) (MACHEAREN, 1979). E o termo "*choropleth map"* foi introduzido pelo geógrafo americano, John Kirtland Wright em 1938 (Howarth, 2017).

FIGURA 7 - Mapa da distribuição de analfabetismo

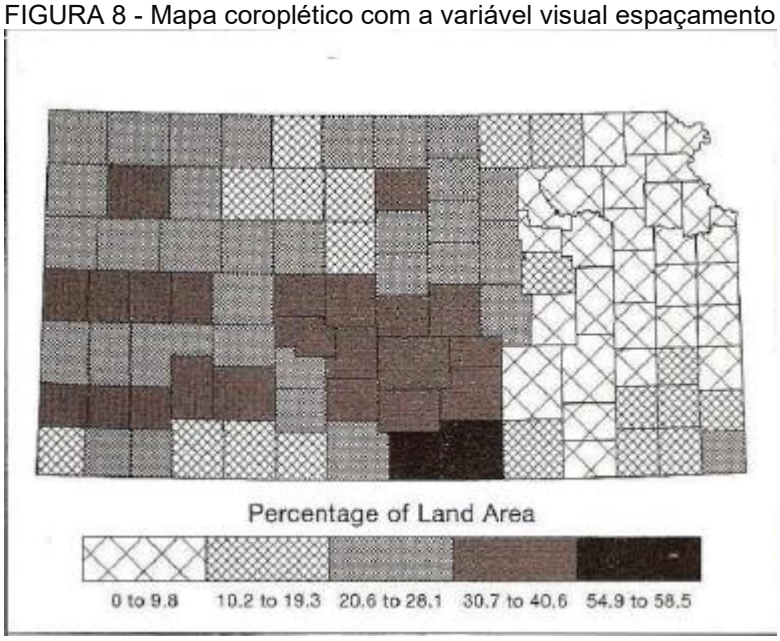


FONTE: http://math.yorku.ca/scs/gallery/images/dupin1826-map_200.jpg

O método coroplético representa uma superfície estatística com símbolos de área (ROBINSON et al. 1995). Estes símbolos coincidem com as áreas de coleta de

dados, que são normalmente áreas estatísticas ou administrativas (DENT, 1999). Eles mostram a variação nos dados quantitativos entre as unidades de enumeração (ROBINSON ET AL., 1995; SLOCUM ET AL. 2008). Esta variação é exibida usando variáveis visuais, tais como luminosidade, saturação e espaçamento (Figura 8) (SLOCUM et al., 2009).

FIGURA 8 - Mapa coroplético com a variável visual espaçamento



FONTE: Adaptado de Slocum et al., (2009)

A técnica coroplética desconsidera a variação do fenômeno dentro da unidade de enumeração, sendo assim, é adequada para fenômenos em que o valor está uniformemente distribuído dentro das áreas. Um mapa coroplético pode ser relacionado com uma representação tridimensional de prismas, em que os valores dos dados são as alturas dos prismas e os maiores valores serão representados por prismas mais altos, semelhante à figura 9.

FIGURA 9 - Modelo em 3D

FONTE: http://www.denunciabr.com.br/estatisticas/

O princípio do mapeamento coroplético é que o valor do fenômeno está uniformemente distribuído dentro da unidade de enumeração, então o topo de cada prisma é horizontal. Como consequência, a variação do fenômeno dentro dos prismas será omitida (SLOCUM et al., 2009). Outro efeito que pode ser causado pela representação coroplética, se as áreas das unidades variam excessivamente, as maiores áreas ocasionam maior impacto visual, então, é preferível que as áreas tenham tamanhos relativamente similares (ROBINSON ET AL., 1995).

Uma questão importante na seleção dos dados para o método coroplético é a normalização dos dados totais, para considerar a variação do tamanho das unidades de enumeração, pois essa variação pode alterar a percepção da distribuição do fenômeno, mascarando, por exemplo, distribuições uniformes (SLOCUM et al., 2009). Há várias abordagens para normalização dos dados, por exemplo, dividir um dado total relacionado com área por outro também relacionado, gerando uma proporção. Cada abordagem para normalizar os dados gera uma representação diferente, e em alguns casos pode ser interessante para a exploração dos dados.

5 MATERIAIS E MÉTODOS

O objetivo principal deste trabalho é automatizar a criação de representações cartográficas de Geo Big Data armazenados em bancos de dados NoSQL. Para concretizar este objetivo foi proposta a construção de um sistema de geoinformação que recupera as informações dos bancos de dados não relacionais, agrega essas informações por unidades geográficas, as normaliza por dados oficiais em um banco relacional e as representa pelo método coroplético.

O sistema de geoinformação foi construído utilizando uma infraestrutura baseada em softwares livre e alimentado por dados abertos, assim, evita-se custos com licenças e, conjuntamente, com a disponibilização dos códigos de programação, possibilita a reprodução desta pesquisa.

5.1 MATERIAIS

5.1.1 Linguagens de Programação

a) Python - É uma linguagem de programação desenvolvida sob licença de código aberto e gerenciada pela Python Software Foundation.(Python, 2017).

b) Javascript - É uma linguagem programação para Web (W3C, 2017)

5.1.2 Hardware

Foi utilizado um Notebook Samsung, com as seguintes especificações: 4Gb de memória ram; disco rígido com capacidade de armazenamento de 1Tb; Tela de 15.6 polegadas; Processador Intel CORE I5, com quatro núcleos e frequência de processamento de 2,20 GHz;

5.1.3 Software e bibliotecas

a) Oracle Virtualbox – *Software* para virtualização de máquinas virtuais;

b) OSGeo-Live – Máquina virtual baseada em Lubuntu (Sistema operacional baseado em Linux e Ubuntu), que contém um conjunto de aplicativos pré-configurados para uma variedade de casos de uso geoespacial, incluindo

armazenamento, publicação, visualização, análise e manipulação de dados. (OSGeo, 2017).

c) Biblioteca Django – Biblioteca para desenvolvimento *web* de aplicações escritas em Python;

d) Biblioteca Json - Biblioteca para processamento de objetos Json em Python;

e) Biblioteca Psycopg2 - É um adaptador do SGBD Postgres/Postgis para a linguagem de programação Python.

f) Biblioteca Pymongo - É um conjunto de ferramentas para interação com o SGBD MongoDB utilizando a linguagem de programação Python;

g) Biblioteca Lxml - Biblioteca para processamento de objetos XML e HTML em Python;

h) Biblioteca Pysal - Biblioteca escrita em Python que contém funções de análises espaciais (PYSAL, 2017);

i) Apache – Servidor web responsável por servir a aplicação para a Web;

j) Mod_tile/rendered – Módulo do servidor web Apache para servir mapas;

k) Mapnik - Software para renderizar mapas;

l) Postgres/PostGIS - Banco de dados objeto-relacional com suporte espacial (DEVELOPERS, 2017);

m) Shp2pgsql – Conversor de arquivos no formato shapefiles em instruções sql para inserção no banco de dados Postgres/Postgis (DEVELOPERS, 2017);

n) MongoDB - Banco de dados NoSQL orientados a documentos (MONGODB, 2017);

o) Leaflet - Biblioteca Javascript para mapas interativos;

5.1.4 Área de estudo

A área de estudo abrange o município de Curitiba, com área de 435.036 km² e com uma população de 1.893.997 habitantes, estimada pelo IBGE para o ano de 2016. A cidade de Curitiba dispõe de uma central de atendimento que objetiva a comunicação entre a prefeitura e o cidadão, possibilitando a solicitação de serviços e

informações de responsabilidade do poder público municipal, denominada Central 156 (ICI, 2017). Nesta central, pode-se solicitar serviços relacionados com a coleta de resíduos, iluminação pública, postos de saúde, transporte coletivo, corte de árvores, entre outros.

5.1.5 Base de dados 156

O aglomerado das solicitações geradas na central 156 compreende a base de dados 156, que é disponibilizada através do portal de dados abertos de Curitiba, local de disponibilização de documentos, informações e dados públicos para a sociedade. (CURITIBA, 2017)

A base de dados 156 é disponibilizada em formato CSV, contendo o acumulado de solicitações em um intervalo de tempo de três meses, e é acompanhado por um dicionário de dados, que contém a semântica de cada campo do arquivo (CURITIBA, 2017). Esta base de dados não tem uma componente espacial associada, porém, cada solicitação contém a informação do endereço do local em que foi solicitado o serviço, como exemplificado em alguns registros no quadro 1, assim, possibilita a geocodificação de cada serviço.

QUADRO 1 – Fragmento da base de dados 156 que contém os endereços das solicitações.

DESCRIÇÃO	LOGRADOURO_ASS	BAIRRO_ASS
ABORDAGEM SOCIAL DE RUA - CRIANÇA - ALCOOLIZADAS/DROGADAS	DESEMBARGADOR MOTTA 0	BATEL
ABORDAGEM SOCIAL DE RUA - ADULTO - PERDIDA/DESORIENTADA	DESEMBARGADOR MOTTA 1070	água VERDE
ABORDAGEM SOCIAL DE RUA - ADULTO - PESSOAS/FAMÍLIAS EM DESABRIGO NA RUA	VISCONDE DE GUARAPUAVA 1895	CENTRO
ABORDAGEM SOCIAL DE RUA - ADULTO - PESSOAS/FAMÍLIAS EM DESABRIGO NA RUA	VISCONDE DE GUARAPUAVA 2600	CENTRO
ABORDAGEM SOCIAL DE RUA - ADULTO - OPERAÇÃO INVERNO	VISCONDE DE GUARAPUAVA 2800	CENTRO
ABORDAGEM SOCIAL DE RUA - ADULTO -	VISCONDE DE GUARAPUAVA 0	CENTRO

PESSOAS/FAMÍLIAS EM DESABRIGO NA RUA		
ABORDAGEM SOCIAL DE RUA - ADULTO - PERDIDA/DESORIENTADA	VISCONDE DE GUARAPUAVA 2674	CENTRO
ABORDAGEM SOCIAL DE RUA - ADULTO - PESSOAS/FAMÍLIAS EM DESABRIGO NA RUA	VISCONDE DE GUARAPUAVA 2652	CENTRO

A base de dados 156 pertence à categoria de uma grande quantidade de dados que estão disponíveis na internet, que são passíveis de espacialização (HAHMANN E BURGHARDT, 2012). Esta base não pode ser considerada estritamente Geo Big Data, contudo, foi utilizada por ser composta por dados abertos e ter volume relativamente alto. Por consequência, escolheu-se esta, como área de estudo para o desenvolvimento do banco.

1.1.1. Bases cartográficas

Foi utilizado um conjunto de bases cartográficas para a formação do banco de dados objeto-relacional, o qual serviria para a construção dos mapas coropléticos. São elas:

- Base vetorial em formato *shapefile* dos Setores censitários, junto com os resultados do censo demográfico do Brasil no ano de 2010 de população por setor e respectiva agregada. Disponível para download no endereço virtual http://downloads.ibge.gov.br/downloads_geociencias.html.

- Base vetorial em formato shapefile das grades estatísticas, correspondentes ao censo demográfico do Brasil no ano de 2010. Disponível para download no endereço virtual http://downloads.ibge.gov.br/downloads_geociencias.html.

- Base vetorial em formato shapefile do arruamento de Curitiba, contendo os nomes dos logradouros e número dos lotes. Disponível para download no endereço virtual http://ippuc.org.br/geodownloads/geo.htm.

Tile map Service (TMS) do OSM como base cartográfica de fundo dos mapas. O serviço pode ser consumido utilizando a URL https://tile.openstreetmap.org/{z}/{x}/{y}.png.

5.2 MÉTODOS

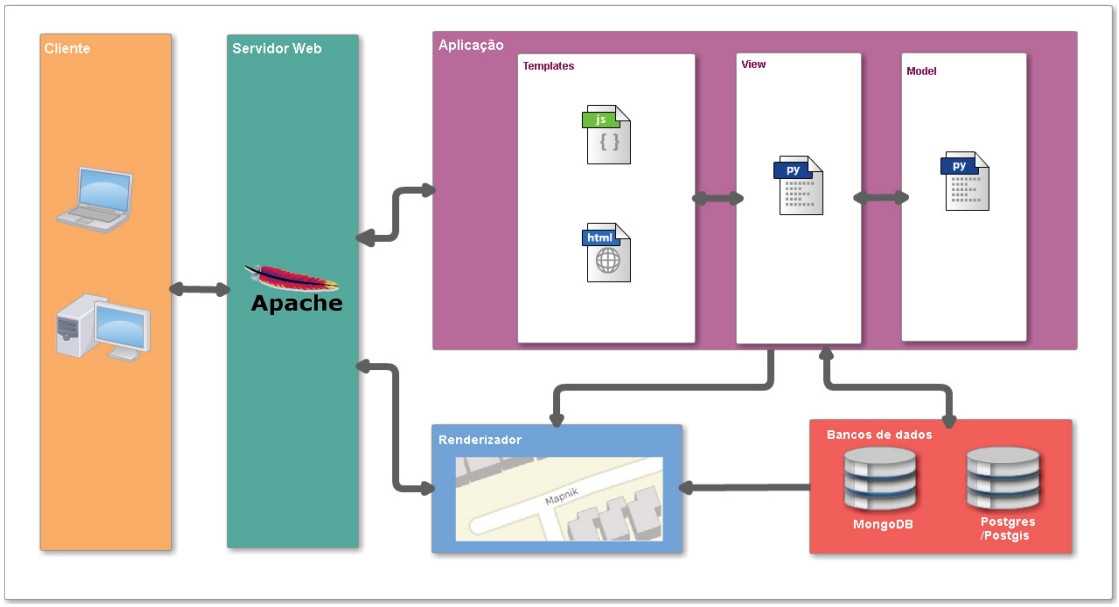
Para desenvolver o sistema de geoinformação foram realizadas as seguintes etapas:

1) Definir a arquitetura do sistema de geoinformação;
2) Definir o projeto cartográfico dos mapas coropléticos;
3) Construção dos bancos de dados;
4) Desenvolver a solução para a construção dos mapas coropléticos;

1.1.2. ARQUITETURA DO SISTEMA

O sistema foi projetado utilizando o modelo Cliente-Servidor, em que a maior parte do processamento é atribuída ao Servidor (ISRAEL et al, 1978). Na figura 10 está representada a estrutura do sistema.

FIGURA 10 - Arquitetura do sistema



FONTE: O Autor (2017)

O Cliente é um programa que se conecta a Web para acessar os servidores (ISRAEL et al, 1978). Os navegadores web, como Mozilla Firefox e Google Chrome, são os clientes do sistema e possibilitam aos usuários interagirem com o servidor.

O servidor do sistema é composto por quatro elementos principais: servidor de dados; aplicação; renderizador e servidor web. O servidor de dados é composto pelos SGBDs Postgres/Postgis e MongoDB, que armazenam as bases cartográficas e a base de dados 156, respectivamente.

A Aplicação é responsável em receber as definições selecionadas pelo usuário para a construção dos mapas coropléticos, processar essas requisições ou delegar o processamento para os outros componentes do sistema e definir qual informação será visualizado pelo sistema. Este elemento do sistema foi desenvolvido utilizando a framework Django, que utiliza um padrão de MTV (DJANGOPROJECT, 2017), o qual separa o desenvolvimento em três camadas: *Model, View e Template*.

- *Model* é a camada responsável pela modelagem dos dados da aplicação, isto é, descreve os dados que serão inseridos na aplicação e seus relacionamentos. *View* é responsável por especificar qual informação é apresentada ao usuário e contém toda a lógica da aplicação.

- *Template* é responsável pela apresentação do conteúdo gerado pela aplicação, ou seja, essa camada contém os arquivos HTML e Javascript que serão enviados para os clientes.

O renderizador é composto pelo software Mapnik e um arquivo XML, que contém todos os parâmetros utilizados pelo Mapnik para renderização dos mapas coropléticos. (Mapnik). Nesta aplicação o servidor web tem a função da comunicação do cliente com a aplicação e servir os mapas através do módulo Mod_tile.

5.2.1 Projeto cartográfico

A metodologia adotada para a realização do projeto cartográfico foi semelhante à proposta de Sluter (2008) para a sistematização do desenvolvimento

de projetos cartográficos, a qual todas as etapas estão inseridas no processo de comunicação cartográfica. Sluter (2008) propõe a seguintes etapas para a realização do projeto cartográfico:

1. Conhecer o usuário e suas necessidades;
2. Determinar os mapas a serem projetados e construídos;
3. Definição da escala e da projeção de cada mapa;
4. Coletar e analisar os dados fonte;
5. Definir a linguagem cartográfica de cada mapa;
6. Construção do mapa

5.2.1.1 Conhecer o usuário e suas necessidades

Os usuários hipotéticos do estudo de caso são os gestores e planejadores da cidade de Curitiba. Suas necessidades consistem em conhecer como ocorre a distribuição espaço-temporal das solicitações de serviços por diferentes áreas político-administrativas e suas divisões, de acordo com a modelagem prevista na ET-EDGV.

5.2.1.2 Determinar os mapas a serem projetados e construídos

Esta etapa decompôs-se em duas sub etapas: Definição das informações temáticas a serem representadas, bem como suas classificações e a definição das bases cartográficas de cada mapa (SLUTER, 2008).

Neste projeto o tema são as solicitações da central 156, que podem ter as categorias filtradas, proporcionando a exploração através diferentes representações do tema. Para definir quais categorias seriam disponibilizadas para exploração do usuário, foi analisado o dicionário de dados da base 156 (quadro 2), o qual traz a semântica de cada atributo da base 156. Dentre estas categorias, duas trazem a descrição semântica das solicitações contidas na base 156, o assunto e a descrição.

QUADRO 2 – Dicionário de dados da base 156

Nome do campo	Descrição
SOLICITACAO	Nº da solicitação efetuada na Central de Atendimento 156
TIPO	Tipo da solicitação. Ex: solicitação, elogio, reclamação,etc..
ORGAO	Órgão em que cadastrador está lotado
DATA	Data da criação da solicitação
HORARIO	Hora da criação da solicitação
ASSUNTO	Assunto ao qual a solicitação se refere
SUBDIVISAO	Subdivisão do assunto ao qual a solicitação se refere
DESCRICAO	Descrição da solicitação
LOGRADOURO_ASS	Logradouro do assunto da solicitação
BAIRRO_ASS	Bairro do assunto da solicitação
REGIONAL_ASS	Regional do assunto da solicitação
MEIO_RESPOSTA	Meio de resposta escolhido para resposta a sua solicitação
OBSERVACAO	Observações referentes a solicitação
SEXO	M para sexo masculino e F para feminino
BAIRRO_CIDADAO	Bairro do domicilio do cidadão
REGIONAL_CIDADAO	Regional do domicilio do cidadão
DATA_NASC	Data de nascimento do cidadão
TIPO_CIDADAO	Tipo do cidadão solicitante
HISTORICO	Últ imo historico da solicitação
ORGAO_RESP	Órgão responsável pela informação
RESPOSTA_FINAL	Descrição da resposta

FONTE: Portal de dados abertos de Curitiba.

Para a classificação dos dados é disposto um conjunto de métodos para o usuário, os quais foram escolhidos com base em critérios avaliados por Slocum et al (2009) apresentados no quadro 3.

QUADRO 3 – Quadro comparativo entre os métodos de classificação

MÉTODO	INTERVALOS IGUAIS	QUANTIS	DESVIO PADRÃO	QUEBRAS MAXIMAS	QUEBRAS NATURAIS	FISHER-J ENKS
Considera as distribuição dos Dados ao longo do intervalo numéricos	pobre	pobre	bom	bom	muito bom	muito bom
Fácil entendimento do método	muito bom	muito bom	muito bom	muito bom	bom	bom
Fácil de computar	muito bom	muito bom	muito bom	muito bom	muito bom	muito bom
Facilidade de entender a legenda	muito bom	pobre	bom	pobre	pobre	pobre
Valores da legenda correspondem Ao intervalo dos dados na classe	pobre	muito bom	pobre	muito bom	muito bom	muito bom
Aceitavel para dados ordinais	inaceitável	aceitável	inaceitável	inaceitável	inaceitável	inaceitável
Provém assistencia a escolha do Numero de classe	pobre	pobre	pobre	pobre	bom	muito bom

FONTE: Adaptado de Slocum et al. (2009)

Dentre os métodos avaliados, o método Fisher-Jenks tem o melhor desempenho, entretanto, ele falha em alguns critérios. Então, além do método Fisher-Jenks, foram disponibilizados os métodos que tem o melhor desempenho nestes critérios: Intervalos iguais e Quantis. Além dos diferentes métodos de classificação, foram proporcionados ao usuário a seleção de diferentes números de classes, que variam de 3 a 7 classes.

5.2.1.3 Definição das bases cartográficas

Os dados espaciais do tema foram definidos conforme a divisão brasileira, estrutura também modelada no padrão ET-EDGV. Os conceitos no quadro 3 correspondem à divisão territorial brasileira e a outras áreas de divulgação e apuração dos dados utilizados neste trabalho, bem como a conceitos associados à delimitação das unidades territoriais de coleta (IBGE, 2010).

QUADRO 3 – Divisão territorial brasileira e a outras áreas de divulgação.

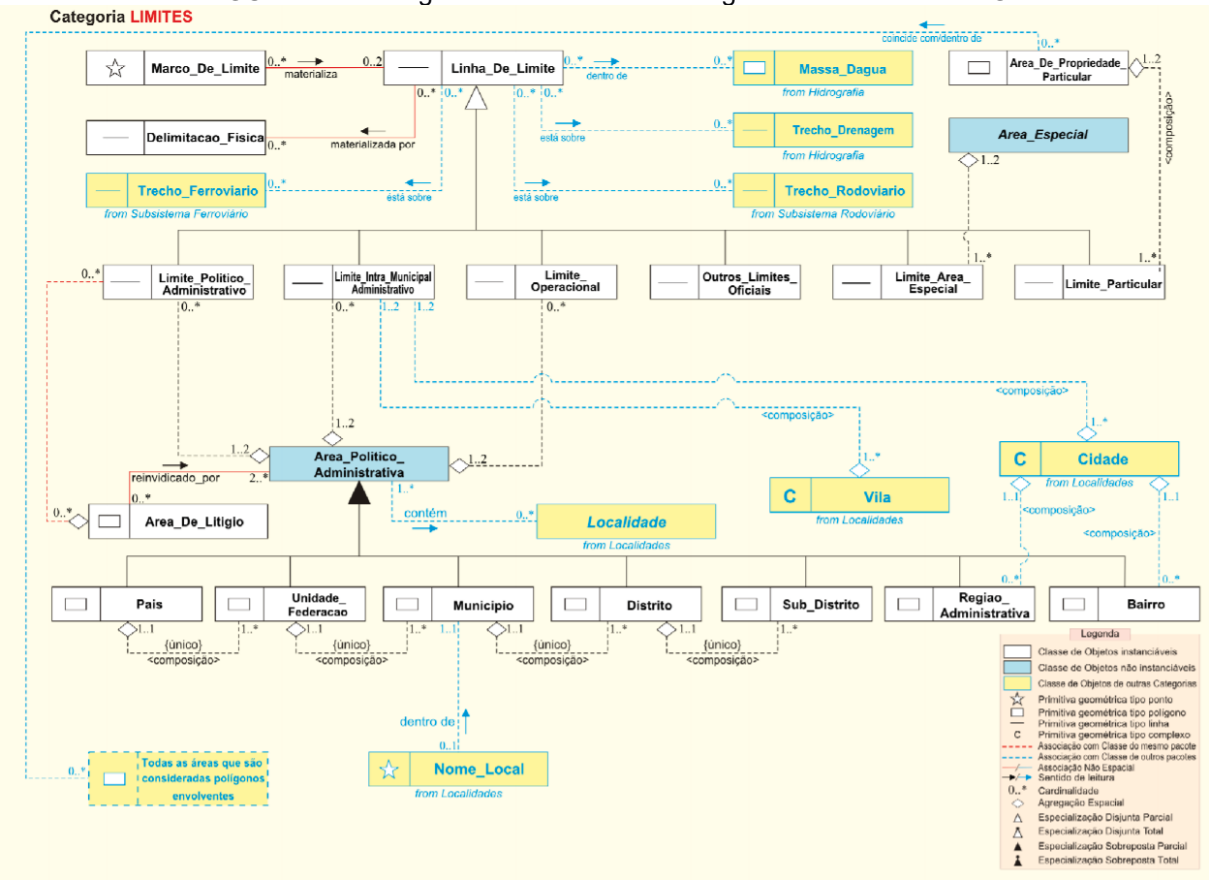
Estados	São as unidades de maior hierarquia dentro da organização político-administrativa do País, subdivididas em municípios podendo incorporar-se entre si, subdividir-se ou desmembrar-se para se anexarem a outros, ou formarem novos estados ou territórios federais, mediante aprovação da população diretamente interessada, através de plebiscito, e do Congresso Nacional, por Lei Complementar. Organizam-se e regem-se pelas constituições e leis que adotarem, observados os princípios da Constituição Federal.
Município	São as unidades autônomas de menor hierarquia dentro da organização político-administrativa do Brasil. Sua criação, incorporação, fusão ou desmembramento se faz por lei estadual, observada a continuidade territorial, a unidade histórico-cultural do ambiente urbano e os requisitos previstos em Lei Complementar estadual.
Regiões Administrativas, Subdistritos e Zonas Bairros e Similares	São unidades administrativas municipais, normalmente estabelecidas nas grandes cidades, criadas através de leis ordinárias das Câmaras Municipais e sancionadas pelo prefeito.
Unidade territorial de coleta (Setor Censitário)	É a unidade de controle cadastral formada por área contínua, situada em um único quadro urbano ou rural, com dimensão e número de domicílios ou de estabelecimentos que permitam o levantamento das informações por um único Agente Credenciado, segundo o cronograma estabelecido.

Os bairros e Subdistritos são aglomerados de Setores Censitários, os Subdistritos formam distritos que formam os municípios e formam os estados. O IBGE divulga o resultado dos Censos por essas unidades territoriais, e por isso, é uma forma conveniente de agregar os resultados:

- Recuperam as informações censitárias com facilidade.
- Agrupam-se em uma estrutura hierárquica em várias escalas.
- Estão ligadas a conceitos reais de administração pública e nomes locais, e são facilmente identificados pelo usuário (como bairros e municípios).

Na área de estudo, o município de Curitiba é dividido em áreas administrativas (subdistritos) e bairros (CURITIBA, 2017), concordante com o modelo da categoria Limites na ET-EDGV (Figura 11). Este modelo prevê a divisão dessas áreas político-administrativas em setores censitários, designados como limites operacionais no modelo (ET-EDGV, 2013).

FIGURA 11 - - Diagrama de classes da categoria limites da ET-EDGV



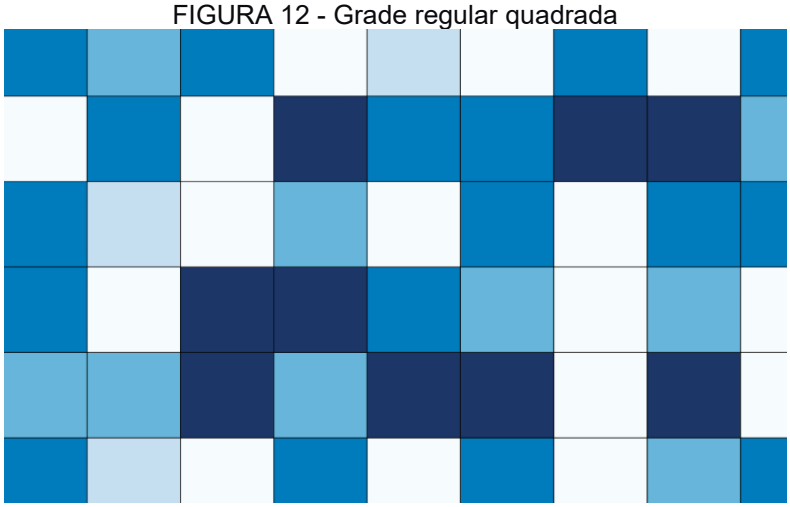
FONTE: Et-EDGV (2013)

Baseado no modelo de limites definidos pela ET-EDGV, a menor unidade territorial, com limites físicos identificáveis em campo (IBGE, 2017), são os setores censitários. Entretanto, em outubro de 2015 (IBGE, 2015), o IBGE disponibilizou bases vetoriais com o propósito de disseminação de dados estatísticos. Estas bases

são formadas por células com dimensões de 200 m por 200 m para áreas urbanas e 1 km por 1 km para áreas rurais. Isto permite representação com maior nível de detalhamento do fenômeno em ocorrências em que as dimensões dos setores forem superiores às das células da grade.

Porém, há uma provável consequência visual negativa quando representamos o fenômeno por grades regulares, que é a criação de linhas visuais que concorrem com a percepção dos padrões gerados pela distribuição dos dados. No caso das grades quadradas (com orientação para o norte), são geradas linhas horizontais e verticais (Figura 12).

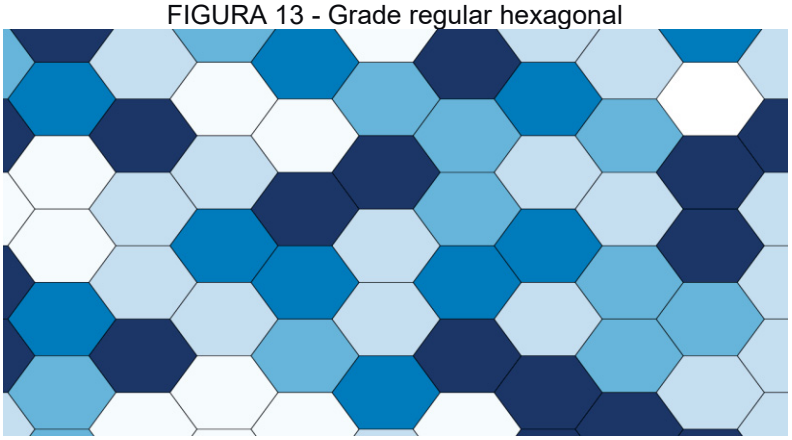
FIGURA 12 - Grade regular quadrada



Fonte: O autor (2017)

Conforme mostra a imagem, a percepção visual humana sofre forte estímulo a essas linhas, decorrente da sensação de equilíbrio gravitacional, e devem ser evitadas (CARR et al,1992). Em razão destas implicações, foi construída uma grade hexagonal com valor de área equivalente as células da grade quadrada disponibilizada pelo IBGE, apresentando como vantagem de não gerar linhas horizontais e verticais (Figura 13), o que não ocorre com outros tipos de grades.

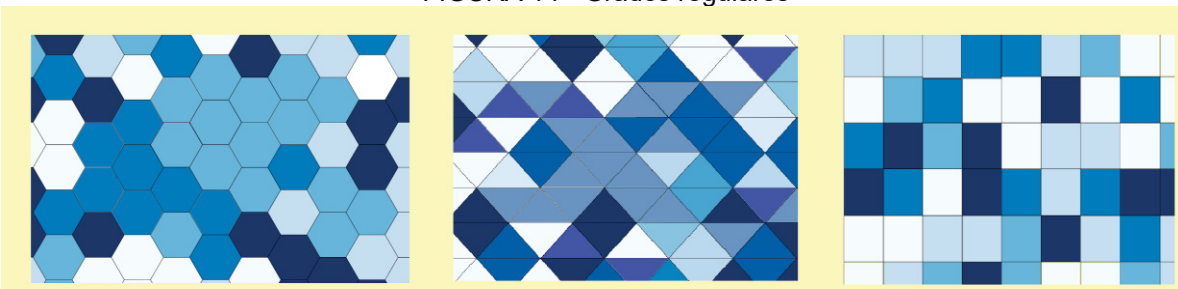
FIGURA 13 - Grade regular hexagonal



FONTE: O autor, 2017.

A figura 14 representa três possíveis tipos de grades, formadas por polígonos regulares que recobrem um plano sem sobreposições e espaços entre eles (Carr et al,1992).

FIGURA 14 - Grades regulares



FONTE: O autor (2017).

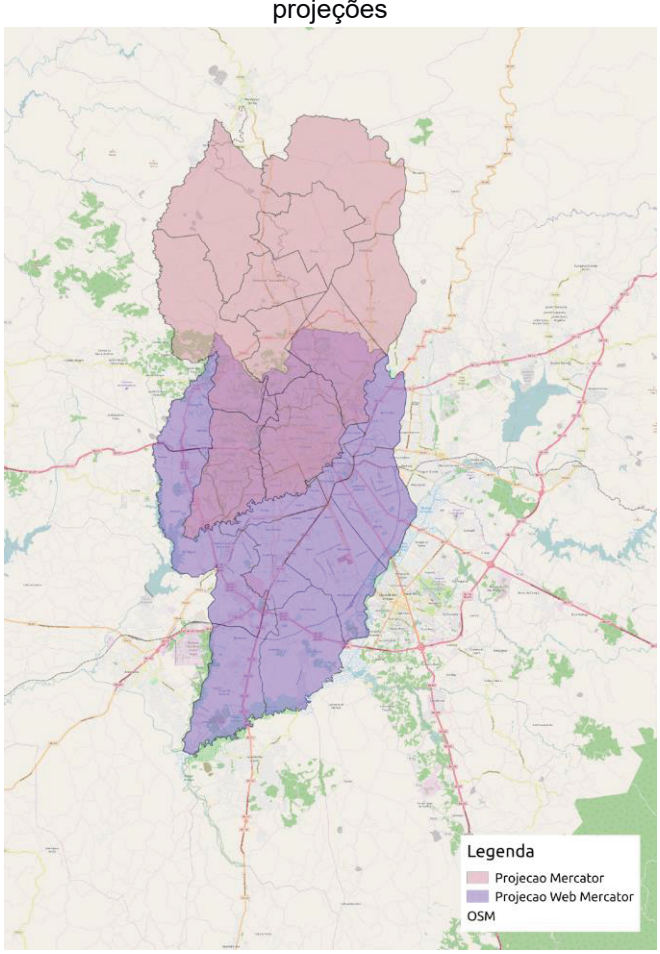
Como mapa de referência foi utilizada a base cartográfica do OSM, disponibilizada via *Tile Map Service (*TMS*)*, assim, utiliza-se dados *VGI* para entender os fenômenos representados.

5.2.1.4 Definição da projeção cartográfica e da escala

A definição da projeção cartográfica é influenciada pela base de referência utilizada na construção dos mapas devido a impossibilidade de reprojeção do TMS do OSM. Em consequência do uso do mapa de referência OSM, que emprega o uso da projeção Web Mercator (EPSG: 3857), para que as diferentes bases cartográficas se sobreponham, a definição da projeção cartográfica fica restrita a Web Mercator.

Porém, para este trabalho o ideal era utilizar uma projeção equivalente para que a 5comparação de áreas fosse favorecida. A figura 15 representa as regionais de Curitiba nas projeções de Mercator e Web Mercator sobrepondo o mapa base OSM.

FIGURA 15 - Regionais de Curitiba em diferentes projeções



FONTE: O autor (2017)

A projeção Web Mercator é uma simplificação da projeção de Mercator, ou seja, adota como superfície de referência a esfera ao invés do elipsoide, porém, não é mantida a propriedade da projeção de Mercator, a conformidade. No entanto, não há uma diferença perceptível na forma de Curitiba entre as duas projeções.

A projeção Web Mercator é adequada para representar áreas que estão contidas entre 180º W a 180ºE e 85.06ºS e 85.06º N (OPENLAYERS,2008). Esta projeção é amplamente utilizada por serviços de web mapas, como Google, Bing, OSM (EPSG:3857, 2016).

Para definir as escalas de representação, preliminarmente foram definidos os retângulos envolventes dos polígonos das bases cartográficas e determinado qual retângulo envolvente apresentava a menor dimensão, assim definiu-se o menor objeto a ser representado.

A definição da dimensão com que os menores objetos das bases cartográficas seriam representados no mapa foi fundamentada na pesquisa desenvolvida por Taura (2007), em que avalia os tamanhos mínimos de discriminação de diversos símbolos geométricos representados em papel. Entre os símbolos avaliados nesta pesquisa estão os quadrados vazados, que foram determinados perceptíveis quando tivessem lados maiores que 0,5 mm. Em consequência dessa pesquisa ser realizada para representações em papel, optou-se por extrapolar o valor de discriminação mínima de quadrados vazados em duas vezes para representação dos menores objetos. No caso dos setores censitários, também foi definido qual a geometria apresentava as maiores extensões, assim definindo qual a escala máxima da sua representação e a escala mínima dos hexágonos. O quadro 4 apresenta a menor dimensão a ser representada de cada base.

QUADRO 4 – Escalas mínimas e máximas das bases vetoriais

BASE CARTOGRÁFICA	MENOR DIMENSÃO A SER REPRESENTADA (M)	ESCALA MÍNIMA	ESCALA MÁXIMA
Hexágonos	159	1: 150.000	
Setores Censitários	40	1: 40.000	1:20.000
Bairros	532	1:500.000	
Regionais	2137	1:1.500.000	

FONTE: O autor (2017)

No entanto, o mapa base OSM tem escalas específicas para cada nível de *"zoom"*, como consequência os mapas são representados nessas escalas, sendo necessário o ajuste da escala mínima de representação de cada mapa a estas escalas. O quadro 5 apresenta o denominador da escala cartográfica de cada nível de *"zoom"* e a que nível as escalas mínimas foram ajustadas.

QUADRO 5 – Níveis de zoom dos TMS OSM em que as bases foram

NÍVEL DE *"ZOOM"*	DENOMINADOR DA ESCALA	BASE CARTOGRÁFICA
0	559.082.264	
1	279.541.132	
2	139.770.566	
3	69.885.283	
4	34.942.642	
5	17.471.321	
6	8.735.660	
7	4.367.830	
8	2.183.915	
9	1.091.958	Regionais
10	545.979	
11	272.989	Bairros
12	136.495	
13	68.247	
14	34.124	Setores Censitários
15	17.062	Hexágonos
16	8.531	
17	4.265	
18	2.133	
19	1.066	
20	533	

FONTE. adaptada DE MAPNIK (2009).

5.2.1.5 Coletar e analisar os dados fonte

Os dados foram coletados no portal de dados abertos da prefeitura de Curitiba, que são disponibilizados em formato CSV, em que cada linha corresponde

a uma solicitação realizada no portal 156. A qualidade dos dados pode ser avaliada a partir da análise dos seguintes componentes: atualidade dos dados, completude dos dados e a precisão posicional dos dados (SLUTER, 2008).

Neste caso, a completude dos dados é dependente do processo de geocodificação, ou seja, quando o processo não obtiver êxito o dado será desconsiderado, assim pode-se calcular a completude geral dos dados. Porém, como os mapas são gerados com dados resultantes de consultas feitas pelos usuários, cada consulta irá gerar um subconjunto de dados com um parâmetro de completude diferente, pois deve-se avaliar a completude a cada tema escolhido pelo usuário, com isso o parâmetro deverá ser calculado a cada consulta no sistema.

A precisão posicional dos dados deve ser calculada para cada elemento da base, em consequência do processo de geocodificação estimar o endereço no segmento de rua baseado no endereço, então podemos fazer uma aproximação da precisão do dado utilizando o comprimento do segmento que ele foi associado.

No entanto, essas avaliações não foram implementadas no sistema, exceto o cálculo da completude geral dos dados.

5.2.1.6 Definição da linguagem cartográfica

A linguagem cartográfica fica restrita aos elementos adequados à representação coroplética. Segundo a definição da ICA, *International Cartographic Association*, os mapas coropléticos são “um método de representação cartográfica que emprega cores distintas ou sombreados aplicados a áreas diferentes daquelas limitadas por isolinhas”. Além de valor de cor, outras variáveis visuais também podem ser utilizadas, como espaçamento.

Neste trabalho, foi utilizado a aplicação ColorBrewer (http://colorbrewer2.org) para definir os esquemas de cores a serem disponibilizados para o usuário. Existem diversos fatores a se considerar na escolha dos esquemas, por exemplo, o tipo de dados a representar (unipolar, bipolar, balanceado), a associação de cores com os alguns fenômenos. (SLOCUM et al., 2009). No entanto, para esta aplicação considerou-se apenas o tipo de dado unipolar, que segundo Brewer (1994), para esse tipo dado, as etapas sequenciais devem ser representadas por incrementos sequenciais em luminosidade, denominando-os esquemas sequenciais. Assim,

foram disponibilizados todos os esquemas sequenciais disponíveis na aplicação ColorBrewer.

Entretanto, os dados agregados totais não são adequados a técnica coroplética, uma vez que em muitas aplicações o tamanho variável das unidades administrativas pode alterar a percepção da distribuição do fenômeno, mascarando, por exemplo, distribuições uniformes. Dessa forma, uma importante etapa na construção dos mapas é a padronização dos dados, na qual dados totais são ajustados para os diferentes tamanhos das unidades de enumeração (SLOCUM et al., 2009). Neste trabalho, para normalizar os dados agregados, produziu-se uma medida de densidade (SLOCUM et al., 2009), a partir da divisão dos dados agregados por unidades de enumeração pelo número da população residente na mesma unidade, possibilitando sua representação pela técnica coroplética.

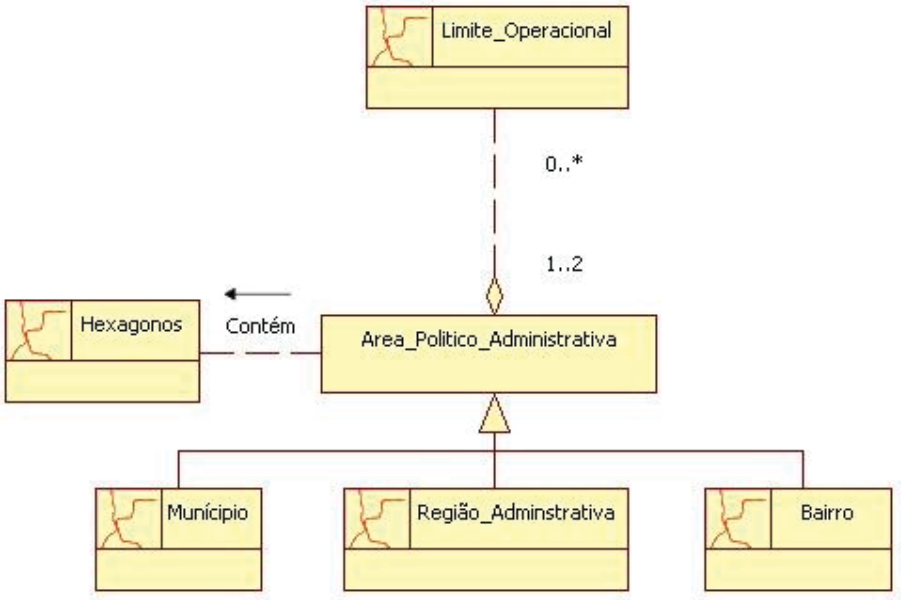
5.2.2 Construção dos bancos de dados

Foram construídos dois bancos de dados distintos, um banco de dados Postgres/Postgis para armazenar as bases cartográficas com as divisas do território e outro banco MongoDB para armazenar os dados da base 156 geocodificados.

5.2.2.1 Banco de dados Postgres/Postgis

Previamente a construção deste banco de dados, realizou-se a modelagem conceitual (figura 16) deste banco de dados utilizando o diagrama de classes da linguagem UML e em conformidade com a ET-EDGV.

FIGURA 16 - Diagrama de classes da modelagem das divisas territoriais



FONTE: O autor (2017).

Optou-se por construir as bases cartográficas dos bairros e regionais a partir da agregação dos setores censitários, considerando que os setores censitários são subdivisões dos bairros e regionais e é uma unidade geográfica em maior escala.

As bases dos hexágonos foram construídas utilizando uma função externa do Postgis, denominada de Hex_grid. Esta função tem como parâmetros: as coordenadas do canto direito inferior; coordenadas canto esquerdo superior; valor de área dos hexágonos; projeção em que os hexágonos serão calculados (projeção equivalente é indicada devido aos hexágonos terem o mesmo valor de área); projeção de saída da geometria.

5.2.2.2 Banco de dados MongoDB

A base de dados 156 não é disponibilizada com uma componente espacial, porém dispõe de informações de endereços que permite a sua espacialização. Então, para possibilitar a realização de análises e operações espaciais sobre os dados foi necessário submetê-los a um processo de associar os endereços a uma

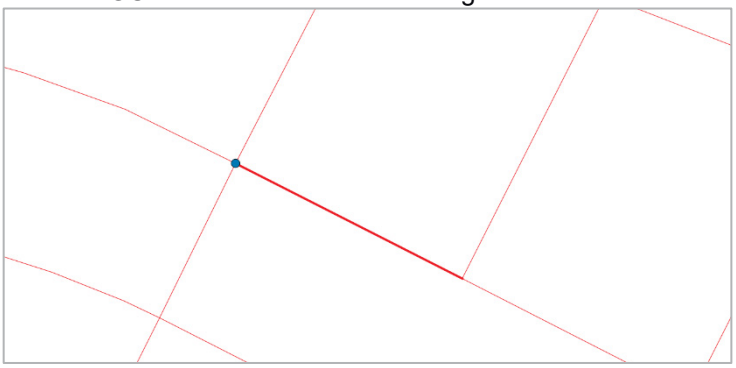
coordenada geográfica, denominado geocodificação (CASANOVA et al., 2015), para posteriormente inseri-los no banco de dados MongoDB.

Para a realização da geocodificação e inserção dos dados foi desenvolvido um script com a linguagem Python. Este script recupera o nome do logradouro, número do lote de cada informação contida na base de dados 156 e faz uma consulta no banco de dados do arruamento de Curitiba.

A consulta recupera todos os segmentos de arruamento que tem o nome do logradouro idêntico ao logradouro consultado. Cada segmento tem a informação do número do lote inicial e final, portanto é recuperado o segmento que abrange o número do lote consultado. O resultado da consulta é a geometria de uma linha, entretanto pretende-se uma precisão melhor da ocorrência do fenômeno, nesse caso foi estimada a geometria pontual. A estimação desta geometria foi realizada com as seguintes operações:

a) Determinou-se o ponto inicial do segmento (figura 17);

FIGURA 17 - Ponto inicial do segmento



FONTE: O autor (2017).

b) Estimou-se a distância estimada do ponto inicial ao lote utilizando a seguinte equação:

$$raio\ do\ buffer\ = (n - n_i) \times \frac{T}{(n_f - n_i)}$$

$n\ -\ número\ do\ lote$

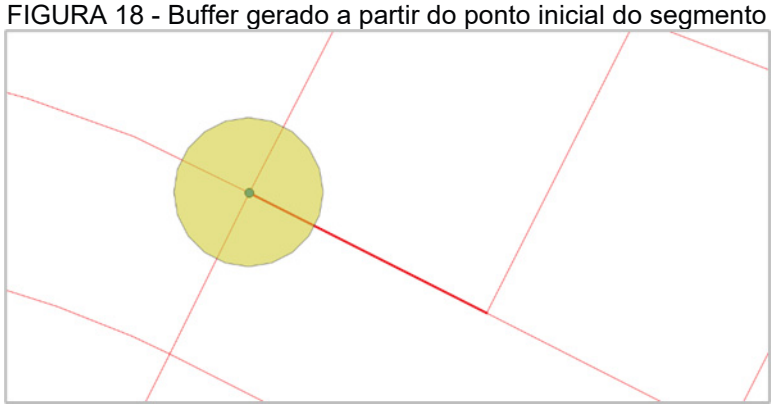
$n_i\ -\ número\ do\ lote\ inicial\ do\ segmento$

$n_f\ -\ número\ do\ lote\ final\ do\ segmento$

$T - comprimento\ do\ segmento$

c) Criou-se um buffer no ponto inicial com a distância estimada do ponto inicial ao lote (figura 18);

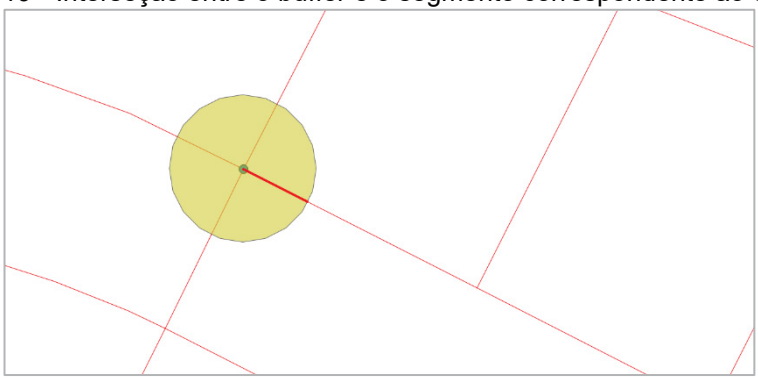
FIGURA 18 - Buffer gerado a partir do ponto inicial do segmento



FONTE: O autor (2017).

d) Foi realizada a interseção do buffer com o segmento (figura 19);

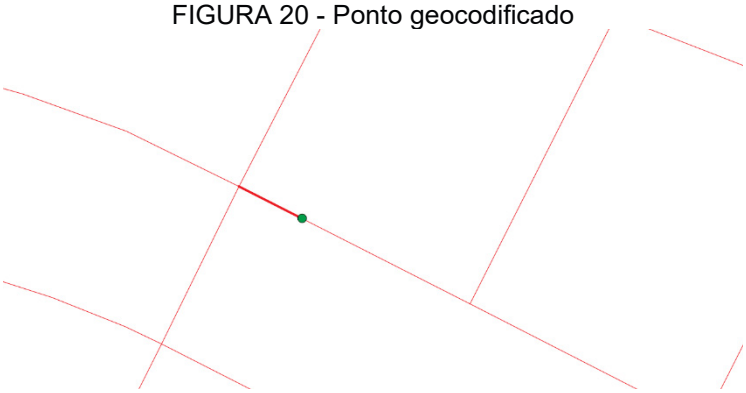
FIGURA 19 - Interseção entre o buffer e o segmento correspondente ao endereço



FONTE: O autor (2017).

e) A geometria resultante da interseção do buffer com o seguimento é o ponto estimado;

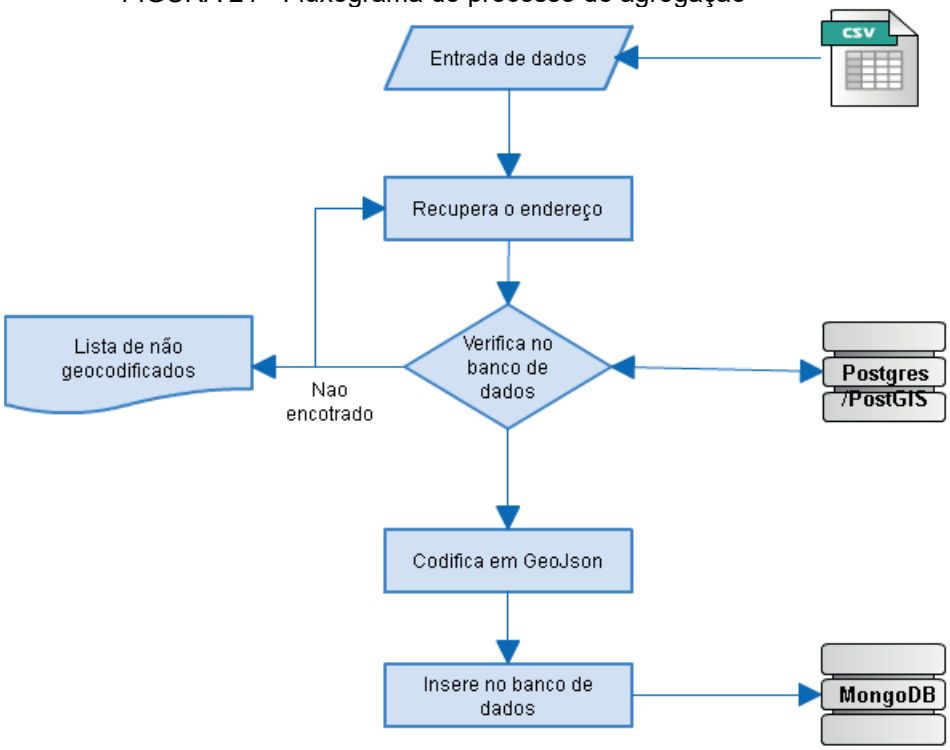
FIGURA 20 - Ponto geocodificado



FONTE: O autor (2017).

Após a geocodificação dos dados foi realizado a transformação dos dados e suas geometrias para o formato GeoJSON, formato de armazenamento do banco de dados MongoDB. Na figura 21, está representado o fluxo de processos para a realização da geocodificação e inserção no MongoDB.

FIGURA 21 - Fluxograma do processo de agregação

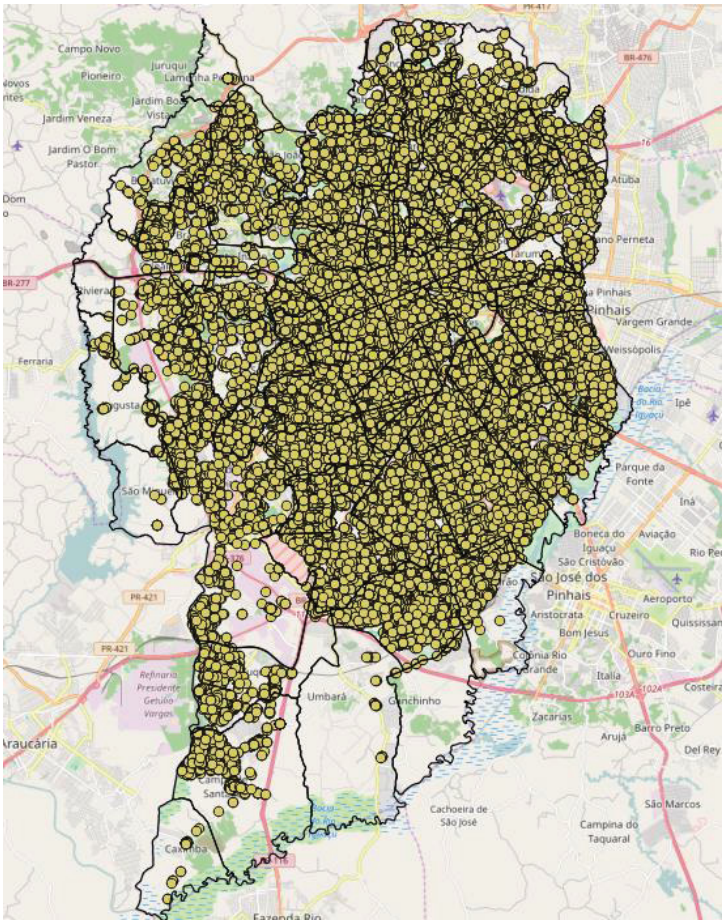


FONTE: O Autor (2016).

O processo de geocodificação foi implementado em linguagem Python e utiliza as bibliotecas Psycopg2 e PyMongo para a conexão com os bancos de dados. A base de dados utilizada no processo continha 69.583 registros, abrangendo um intervalo temporal de três meses, de 01/07/2016 a 01/10/2016. Após o

processamento, verificou-se que 58.606 registros foram geocodificados, aproximadamente 84,2% dos registros totais. A representação dos registros geocodificados é apresentada na figura 22.

FIGURA 22 - Registros geocodificados sobre a base cartográfica de Curitiba e a base OSM.



FONTE: O Autor (2016).

5.2.3 Desenvolvimento da solução para a construção dos mapas coropléticos

Para a automação da representação dos dados pontuais empregando a técnica coroplética, foram definidos os processos necessários para a construção dos mapas, a sequência dos processos é representada na figura 23.

FIGURA 23 - Sequência dos processos de construção dos mapas coropléticos

FONTE: O Autor (2016).

O processo inicial da construção dos mapas consiste no usuário definir os parâmetros referentes ao tema a ser representado, classificação dos dados e a simbologia dos mapas. Os demais processos são descritos posteriormente.

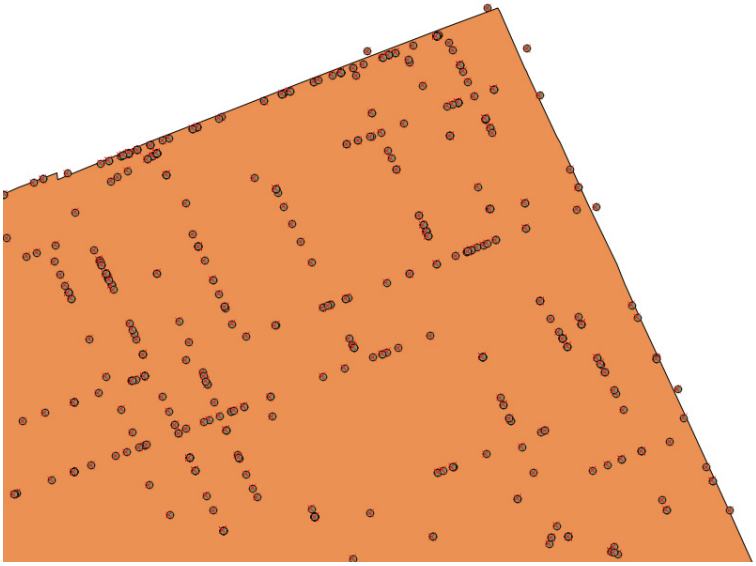
5.2.3.1 Agregação

O objetivo deste trabalho é apresentar uma representação de fenômenos pontuais numerosos, para que sejam visualizados, nas escalas apropriadas, de forma agregada, neste caso, utilizando a técnica coroplética com a dimensão área.

Para realizar a integração dos dados oficiais de área, armazenados em banco de dados relacionais, com dados pontuais da base 156, armazenadas em banco de dados não relacionais, a principal operação espacial é consultar quais pontos estão contidos em cada área.

O banco de dados MongoDB possui uma função espacial para consultar se uma geometria está dentro da outra, denominada *geoWithin*. Foi realizado um teste com esta função para consultar os pontos que estavam dentro de um polígono. No entanto, o resultado da consulta contém pontos que estão fora do polígono (Figura 24).

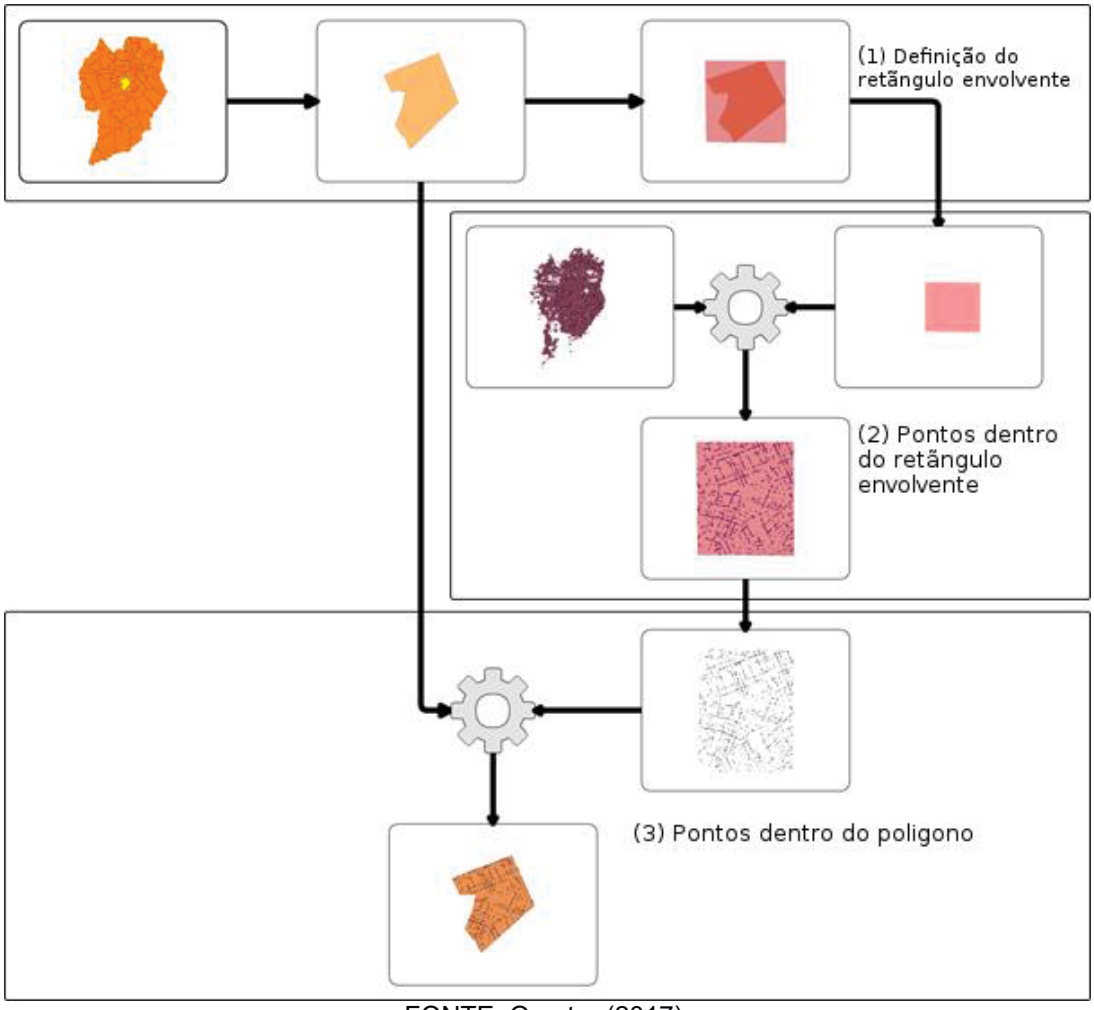
FIGURA 24 - Resultado da consulta espacial contém no MongoDB



FONTE: O Autor (2016).

Em consequência da imprecisão da função do MongoDB, não é possível a utilização desta função no processo de agregação. Então optou-se por utilizar a as funções espaciais do SGBD PostgreSQL/Postgis. Para isto, é necessário transferir os pontos para o Postgres/Postgis, porém este procedimento tem baixo desempenho em termos de tempo de execução e consome muitos recursos computacionais, devido ineficiência deste SGBD manipular quantidades massivas de dados (SHEKHAR et al., 2012). Com base nestas limitações, foram adotadas algumas estratégias para definir um conjunto de processos para otimizar o processo de agregação. A figura 25 demonstra a sequência de processos da agregação.

FIGURA 25 - Fluxo do processo de agregação



FONTE: O autor (2017).

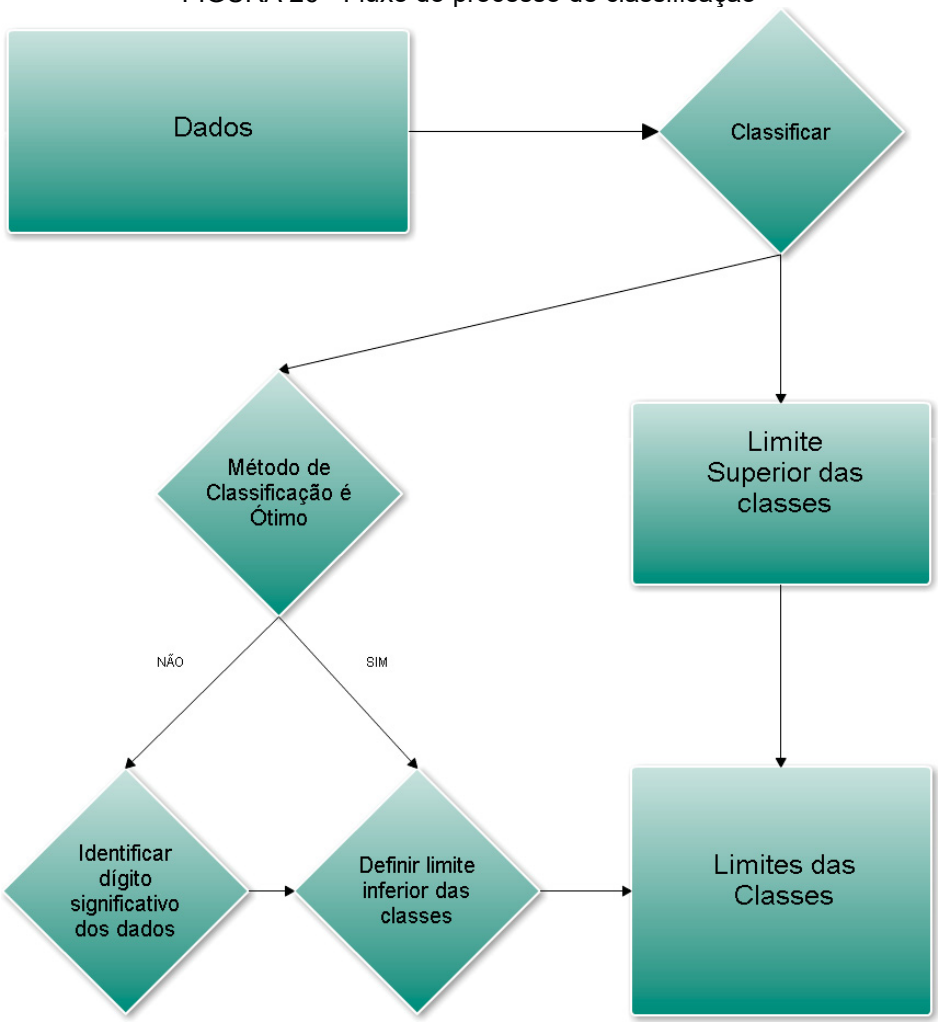
O primeiro processo é minimizar a quantidade de dados a serem transferidos para o banco de dados Postgres/Postgis, por meio de utilizar a geometria do retângulo envolvente de cada polígono, marcado como (1) na figura 26, para consultar os pontos no banco de dados MongoDB (2), assim restringe-se significativamente a quantidade de pontos para se transferir. Com os pontos transferidos para o Postgres/Postgis, consultam-se quais desses pontos estão dentro do polígono que se deseja realizar a agregação (3), na sequência conta-se a quantidade de pontos e o resultado é inserido como atributo do polígono.

5.2.3.2 Classificação

Subsequente ao processo de agregação, com a inserção do número de ocorrências por unidades geográficas nas bases cartográficas, o processo de classificação é executado.

Para realizar a classificação dos dados foi utilizado a biblioteca Pysal. Esta biblioteca contém um conjunto de funções de classificação direcionado para construção de mapas coropléticos (PYSAL, 2017). Contudo estas funções retornam apenas os limites superiores das classes, faz-se necessário processar os dados para definir o limite inferior das classes. Na figura 26 estão representadas todas as etapas para a classificação dos dados.

FIGURA 26 - Fluxo do processo de classificação



FONTE: O autor (2017).

Inicialmente recuperam-se os dados agregados das bases cartográficas e utiliza-se a função escolhida pelo usuário para classificar os dados, obtendo o limite superior das classes.

Para determinar o limite inferior, verifica-se qual tipo de método de classificação foi empregado, devido a diferença da composição do limite das classes nos diferentes métodos.

No método de classificação intervalos iguais não há “lacunas” entre as classes, ou seja, se a precisão dos dados classificados é 0,1 unidade e o limite superior de uma classe é 2,6, então o limite inferior da classe seguinte será 2,7. Isto não ocorre em outros métodos de classificação, em que os próprios dados compõem o limite inferior e superior das classes (SLOCUM et al., 2009).

Se o método de classificação selecionado for intervalos iguais, define-se qual a precisão dos dados com base no maior número de algarismos significativos entre todos os dados. Os seguintes procedimentos foram adotados para esta determinação:

a) Desmembrar o número pelo separador decimal e recuperar a parte decimal (1,223 → 223);

b) Calcular a quantidade de algarismos na parte decimal;

c) Repete-se o processo para todos os dados;

d) Adota-se como número de algarismos significativos a maior quantidade de algarismos na parte decimal entre todos os dados;

e) A precisão é definida como:

$$Se\ o\ número\ de\ algarismos\ (a)\ > 0, então\ a\ precisão\ = 10^{-a}$$
$$Se\ o\ número\ de\ algarismos\ (a) = 0, então\ a\ precisão\ = 1$$

Determinada a precisão dos dados, define-se o primeiro limite inferior como o menor valor entre os dados, e os demais são definidos pela soma do limite superior da classe anterior com a precisão dos dados. Para os demais métodos de classificação os limites inferiores das classes são definidos como:

a) O primeiro limite inferior é o menor valor entre os dados;

b) Os demais limites inferiores são definidos recuperando o menor valor compreendido entre o limite superior da classe anterior e o limite superior da classe;

c) Se o limite superior e inferior de uma classe for igual, a classe será representada pelo limite inferior;

5.2.3.3 Modificação do estilo

Com as classes definidas no processo anterior e o esquema de cores aplicado pelo usuário, a próxima etapa é modificar o arquivo XML que contém os parâmetros para a renderização dos mapas coropléticos. A modificação é executada utilizando a biblioteca Lxml, a qual permite sistemas desenvolvidos em Python manipular arquivos XML. A seguir está um fragmento do arquivo:

```
<Style cor="YlGnBu" n_classe="3"  name="YlGnBu3">
    <Rule classe="1">
                    <Filter>[cont] &gt; 0.000000 and [cont] &lt; 1.000000</Filter>
                    <PolygonSymbolizer fill="#edf8b1"/>
                    <LineSymbolizer stroke="black" stroke-width=".3"/>
            </Rule>
</Style>
```

A tag *<Style>* identifica um conjunto de parâmetros para simbolização das camadas. Para cada esquema de cor com determinado número de classes é gerado um "*Style"* que conterá um conjunto de tags *<Rule>*, em que cada "Rule" indicará uma classe do esquema de cores e as tags contidas entre a tag *<Rule>* e *</Rule>* definem a regra da classe e como ela será representada.

Na tag *filter* define-se qual atributo da base cartográfica será filtrada ([cont]), verificando quais elementos terão esse atributo entre o limite inferior (> 0.000000) e superior (< 1.000000) da classe. Estes limites serão modificados em todos os "*Styles"* nesta etapa do processo.

A tag *<PolygonSymbolizer fill="#edf8b1">* indicará qual cor, em código HTML hexadecimal, será aplicada aos polígonos e a tag *<LineSymbolizer stroke="black" stroke-width=".3"/>* define a cor e espessura das linhas limite dos polígonos.

Após modificar os limites das classes, é necessário aplicar o "*Style"* definido pelo usuário em cada camada, para isto modifica-se o conteúdo contido na tag <StyleName>, como está representado abaixo no fragmento do arquivo XML.

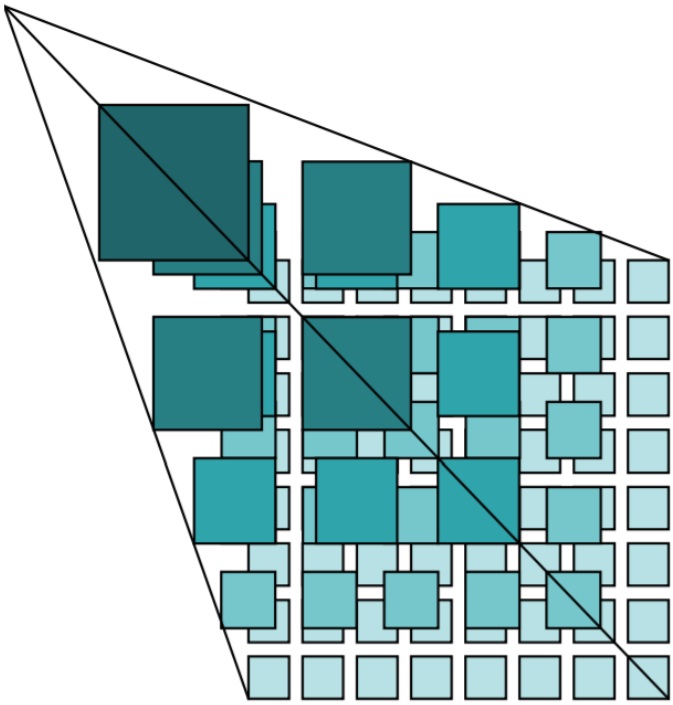
```
<Layer name="bairros" >
 <StyleName> nome do Style </StyleName>
  <Datasource>
   "Configurações  da fonte das camadas"
  </Datasource>
</Layer>
```

5.2.3.4 Renderização

O processo de renderização é executado pelo software Mapnik, que recebe requisições do módulo Mod_tile com especificações das imagens a serem geradas e as especificações da simbologia dos mapas.

O Mod_tile é um servidor de TMS (*Tile Map Service)*, o qual envia solicitações de imagens de áreas específicas do mapa em determinada escala conforme é solicitado pelo usuário via interface, usando de ferramentas de *pan* e *"zoom"* disponibilizadas por bibliotecas de mapas Javascript (Leaflet, Openlayers). Os mapas são divididos em uma grade quadrada e em escalas fixas, identificadas por níveis de "zoom", e cada elemento dessa grade é denominado *Tile* (TILES,2017). O menor nível de "zoom" é representado por um *Tile* e a cada aumento do "zoom" o número de *Tiles* é calculado como 2^{2*n} , em que n é número do nível de "zoom" (SLIPPY MAP, 2017), formando uma estrutura piramidal (Figura 27).

FIGURA 27 - Estrutura piramidal da organização dos tiles



FONTE: Adaptado de OGC (2015)

Para otimizar o serviço, o Mod_Tile armazena cada *Tile* gerado em disco e a cada solicitação do cliente para renderização de uma área do mapa, é verificado se os tiles correspondentes a essa área já foram renderizados, economizando recursos do servidor (OSGEO WIKI, 2017), além de permitir a distribuição da carga de processamento entre múltiplos servidores (CHANGESET, 2017)

O OSM emprega o Mapnik para renderizar os *tiles* e Mod_tile como servidor TMS, que foi desenvolvido para a necessidade de alto desempenho do OSM, devido a quantidade de solicitações de renderização (SLIPPY MAP, 2017). Este alto desempenho é proveitoso para gerar representações do Geo Big Data, em consequência da característica de velocidade, que congrega, além da velocidade com que os dados são gerados, a necessidade da velocidade do processamento e disponibilização dos dados (DASGUPTA, 2013).

O Mod_tile não foi desenvolvido para atualizar automaticamente as mudanças nas simbologias dos *Tiles*, o que acontece a cada pedido do cliente neste sistema proposto. Portanto, foi necessário desenvolver uma função para atualizar a simbologia no renderizador.

O Mod_tile é executado no sistema como um serviço, então é necessário reiniciá-lo para recarregar o arquivo XML com os parâmetros de construção dos mapas atualizados. No entanto, após o reinício do serviço alguns *tiles* antigos eram servidos, por efeito do armazenamento para reutilização, logo é necessário a exclusão dos *Tiles* armazenados. Após reiniciar o serviço e excluir os arquivos antigos, os *Tiles* correspondentes ao "zoom" atual não eram atualizados, em razão do armazenamento em *cache* do navegador Web. A forma encontrada de atualizar o cache foi passar um parâmetro randômico junto com a URL do TMS, assim cumprindo a atualização dos *Tiles*.

5.2.4 PROJETO DA INTERFACE

O desenvolvimento do sistema foi realizado análogo a de um sistema cartográfico especialista, o qual auxilia o usuário nas construções dos mapas (SCHMIDT, 2008), neste caso restringimos os controles do mapa para evitar o uso inadequado, de acordo com os conceitos de comunicação cartográfica. Essas restrições foram definidas no projeto cartográfico e implementadas na interface do sistema.

A interface foi modelada para prover interatividade tanto nas componentes gráficos (imagens) como nos atributos do mapa (SLUTER, 2000), a figura 28 representa a modelagem da interface em linguagem UML, por meio do diagrama de casos de uso.

FIGURA 28 - Diagrama de casos de uso do sistema

FONTE: O Autor (2017).

A interação com os atributos do mapa é a consulta de dados não gráficos nos bancos de dados (SLUTER, 2000), nesta interface corresponde a: consultar o valor do atributo em uma região, definir o atributo e o intervalo temporal a ser representado. A interação com os atributos quantitativos ocorre com a decisão do número de classes e do método de classificação, e pode objetivar tanto a associação entre os valores do fenômeno com o mapa como a análise dos dados (SLUTER, 2000). Na interação gráfica pode-se definir os esquemas de cores, definir qual camada será representada, definir a transparência do mapa coroplético, visualizar áreas específicas em diversas escalas utilizando “pan” e “"zoom"”.

O leiaute da página foi diagramado de forma que o *frame* do mapa preencha 100% da altura e 60% em largura da tela, devido a amplitude da geometria de Curitiba ser maior em latitude, o restante da tela será destinada aos controles. A interface foi desenvolvida utilizando a linguagem HTML, para estruturação da página Web (W3C, 2017), e a linguagem de programação Javascript, a fim de prover mais interatividade do usuário com a interface. O frame do mapa foi desenvolvido

utilizando a biblioteca Leaflet, escrita em Javascript para desenvolvimento de mapas interativos nas principais plataformas *desktop* e *mobile* (LEAFLET, 2017). A definição desta biblioteca foi motivada por dispor do conjunto de ferramentas interativas e a integração com as fontes de dados necessárias neste projeto, além de ter uma curva leve de aprendizagem (FARKAS, 2017).

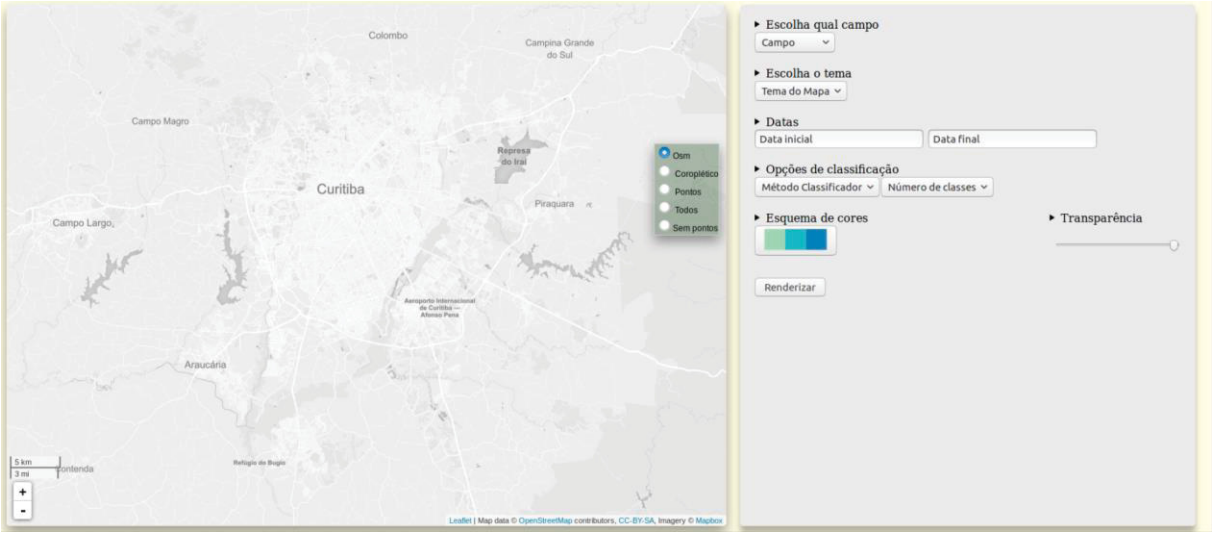
6 RESULTADOS E DISCUSSÕES

Este capítulo apresenta o funcionamento do sistema proposto, detalhando a interface, os mapas e as questões de desempenho abordadas.

6.1 INTERFACE

O acesso a interface ocorre por meio de navegadores Web em plataformas *Desktop* e *Mobile*, como Mozilla Firefox e Google Chrome, embora fosse projetada para plataformas Desktop. A Figura 29 representa o acesso inicial a interface.

FIGURA 29 - Acesso inicial a interface



FONTE: O autor (2017).

Inicialmente é apresentado a base cartográfica do OSM, com o munícipio de Curitiba abrangido no *frame* do mapa. Sobreposto ao *frame*, situam-se a escala gráfica, os controles de “*zoom” e* de camadas. O controle de “"zoom"” é fornecido pelo Leaflet e pode ser acessado por botões (Figura 30) e movimento de “rolagem” dos dispositivos desktop, conjuntamente com o controle de “pan”, acessado via movimento de “arraste”.

FIGURA 30 - Controles de zoom dos mapas.

FONTE: O autor (2017)

O Leaflet fornece um controle de camadas nativo, porém, ao se atualizar o *link* de uma camada TMS, procedimento necessário para “limpar” o *cache* do navegador (explicitado anteriormente), os comandos do controle não executam suas funções conforme esperado. Então, foi desenvolvido um controle (figura 31) para estas funções, o qual permite representar as camadas separadamente e as camadas agrupadas, com as seguintes combinações: Osm, Coroplético e Pontos (Todos); Osm e Coroplético (Sem pontos).

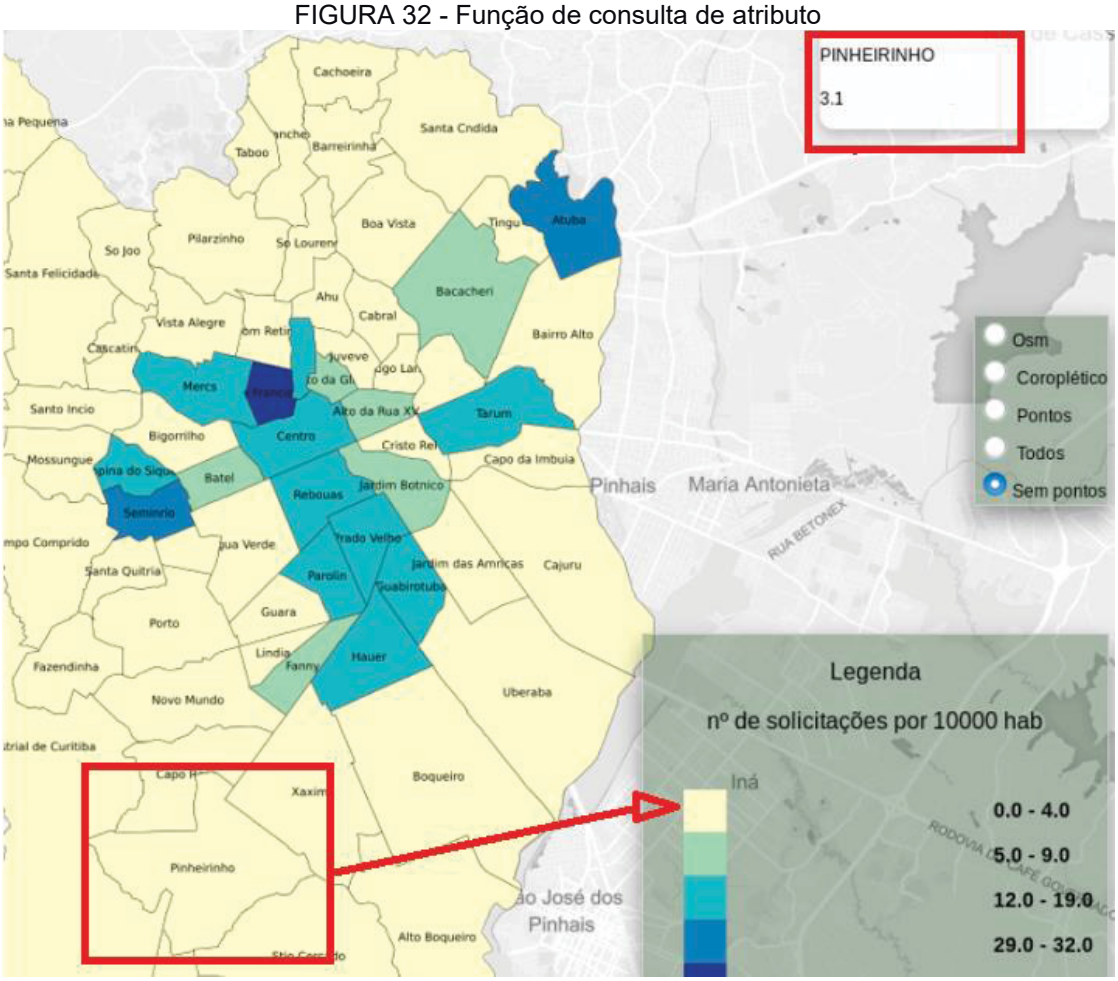
FIGURA 31 - Controle de camadas dos mapas

FONTE: O autor (2017)

Outra funcionalidade que o Leaflet disponibiliza é capturar as coordenadas geográficas do indicador do mouse sobre o mapa, isto possibilita a consulta dos atributos das unidades geográfica representadas. Neste caso, foi implementado para capturar as coordenadas no evento “Click” e subsequente consultar no Postgres/Postgis em qual polígono estão contidas. No entanto, a função deve prever qual nível de "zoom" o usuário está visualizando, para recuperar o atributo da

geometria correta. A figura 32 exibe a geometria consultada, o valor do atributo e qual classe ela pertence.

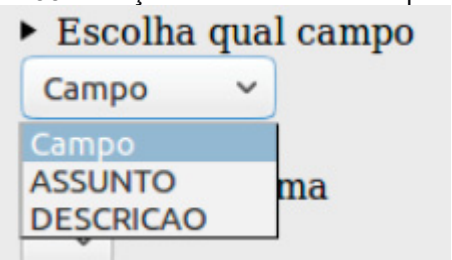
FIGURA 32 - Função de consulta de atributo



FONTE: O autor (2017)

Os controles para a construção dos mapas coropléticos estão dispostos no lado direito da interface e são agrupados conforme as suas funções: configurações do atributo, da classificação e da aparência do mapa. A primeira configuração de atributo é definir qual campo será filtrado (Figura 33)

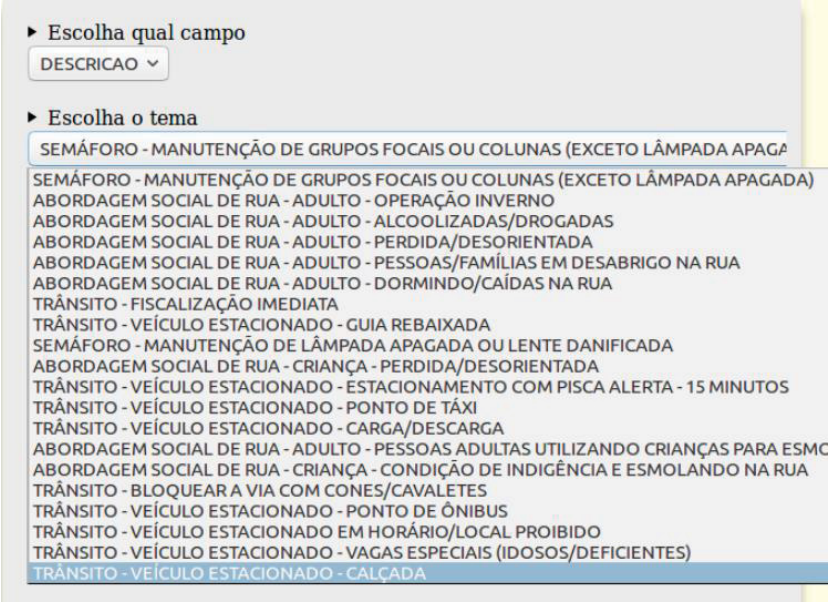
FIGURA 33 - Função de escolha do campo a filtrar



FONTE: O autor (2017)

Definido o campo a ser filtrado, o sistema consulta os valores únicos desse campo no banco de dados MongoDB e alimenta o controle da escolha do tema, conforme a Figura 34.

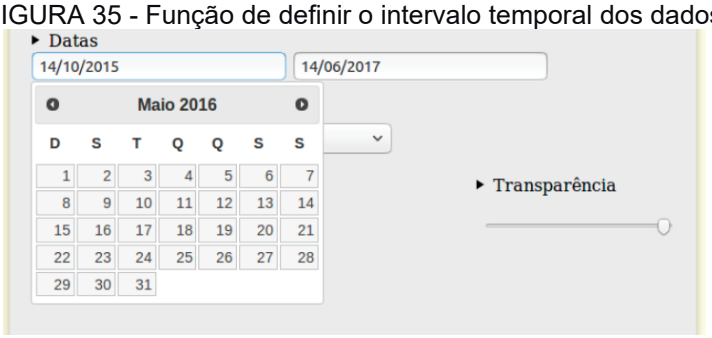
FIGURA 34 - Definição da informação a representar



FONTE: O autor (2017)

O sistema foi construído para permitir consultas temporais na base de dados 156, em consequência da disponibilidade atributo tempo na base. O controle de definição do intervalo temporal da consulta divide-se em campos de data inicial e data final. Ao clicar nos campos de datas, abre-se um menu em forma de calendário (Figura 35) que é provido pela linguagem HTML5.

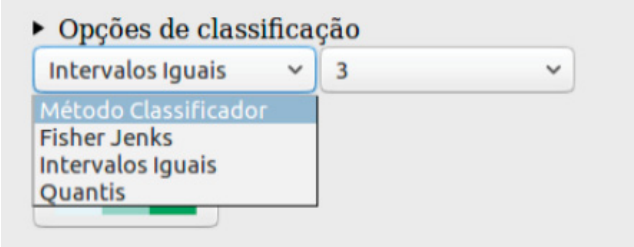
FIGURA 35 - Função de definir o intervalo temporal dos dados



FONTE: O autor (2017)

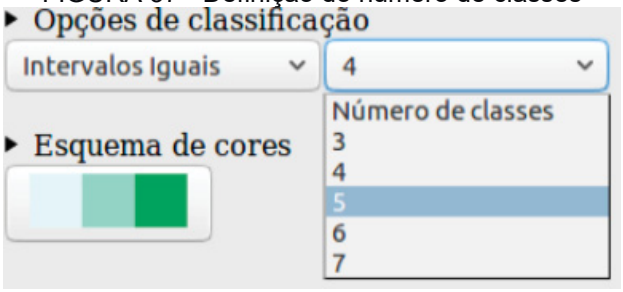
As configurações de classificação são decompostas em definir o método de classificação (Figura 36) e o número de classes (Figura 37). A definição do número de classes determina o conjunto de esquemas de cores que será apresentado ao usuário, por exemplo, se for definido 4 classes serão apresentados esquemas com 4 cores, conforme a figura 38.

FIGURA 36 - Definição do método de classificação



FONTE: O autor (2017).

FIGURA 37 - Definição do número de classes



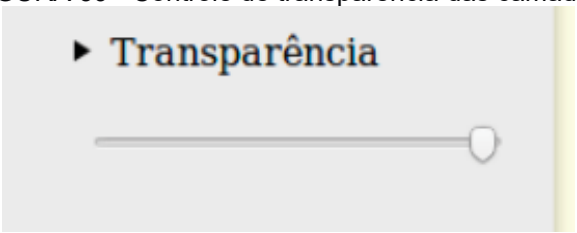
FONTE: O autor (2017).

FIGURA 38 - Definição do esquema de cores

FONTE: O autor (2017)

Para exploração da base OSM, com a sobreposição das demais camadas, a interface provê o controle de transparência das camadas, conforme a figura 39.

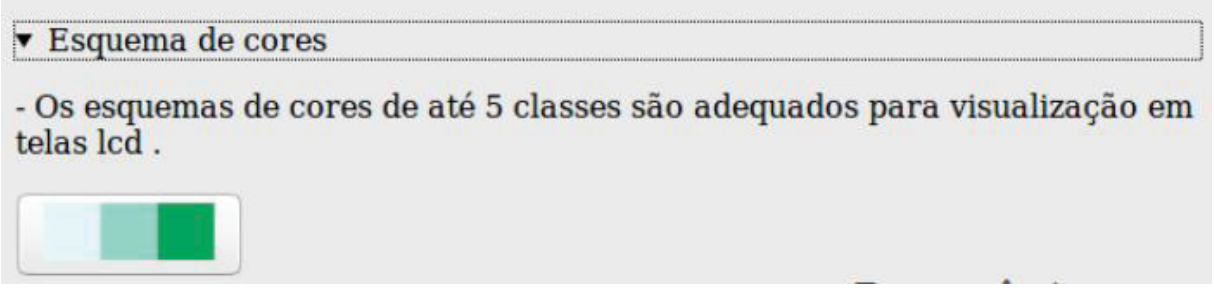
FIGURA 39 - Controle de transparência das camadas



FONTE: O autor (2017)

O sistema, além de restringir as opções do usuário, auxilia a construção dos mapas com a descrição das limitações que a seleção das opções ocasiona. (Figura 40)

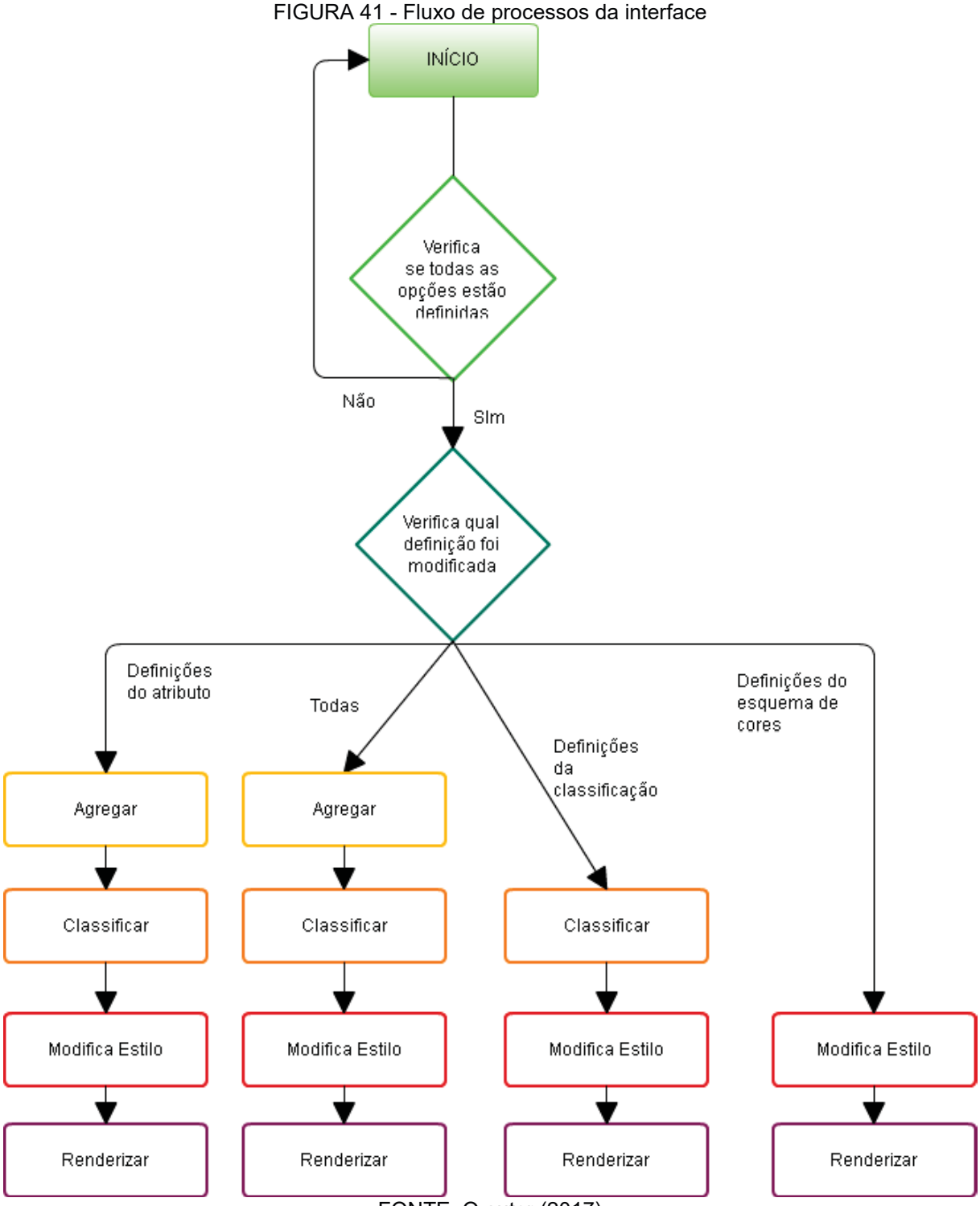
FIGURA 40 - Explicação do funcionamento de um controle



FONTE: O autor (2017)

O início da construção dos mapas é ativado no botão renderizar. Após a solicitação da construção, o sistema executa uma verificação das opções definidas na interface para determinar quais processos serão executados, assim evita-se que sejam executados todos os processos sem necessidade, por exemplo, para executar uma nova classificação dos dados não é necessário realizar a agregação novamente. Se alguma opção não for definida ou todas as opções já foram utilizadas para construir o mapa que está sendo renderizado, não é realizado nenhum processamento. A figura 41 descreve os processos que serão executados conforme a definição das opções realizadas pelo usuário.

FIGURA 41 - Fluxo de processos da interface



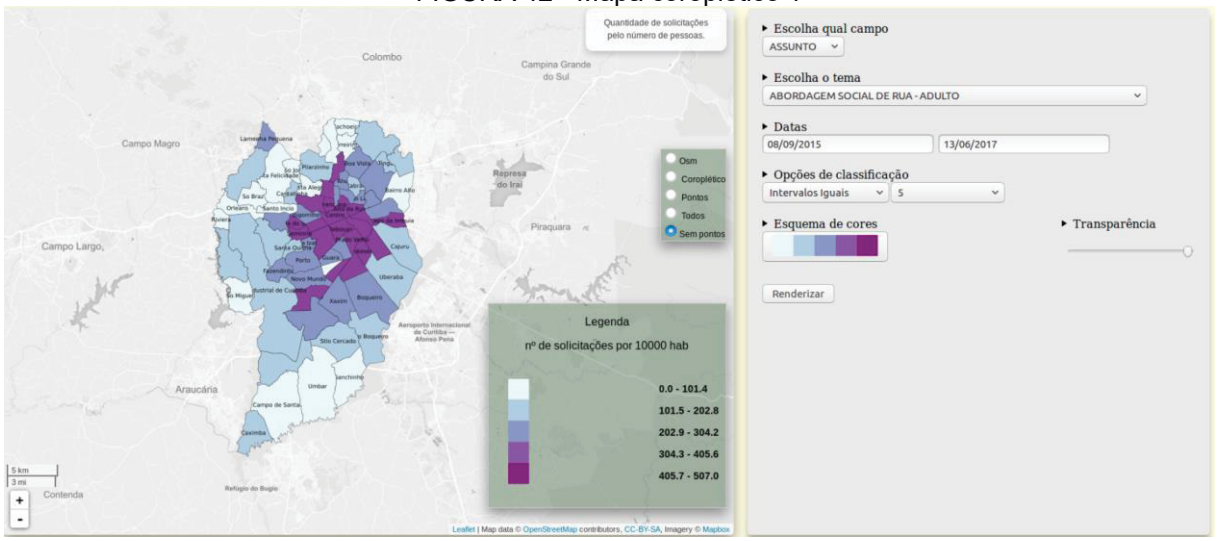
FONTE: O autor (2017)

6.2 MAPAS

Para demonstrar o resultado da renderização foi gerado um conjunto de mapas, com diversas configurações e escalas. O primeiro mapa a ser apresentado (Figura 42) foi gerado com o tema “Abordagem social de rua - Adulto”, classificado

com cinco classes e com o método intervalos iguais. Este mapa apresenta a legenda característica do método de classificação intervalo iguais, ou seja, não tem “lacunas” de valores entre as classes.

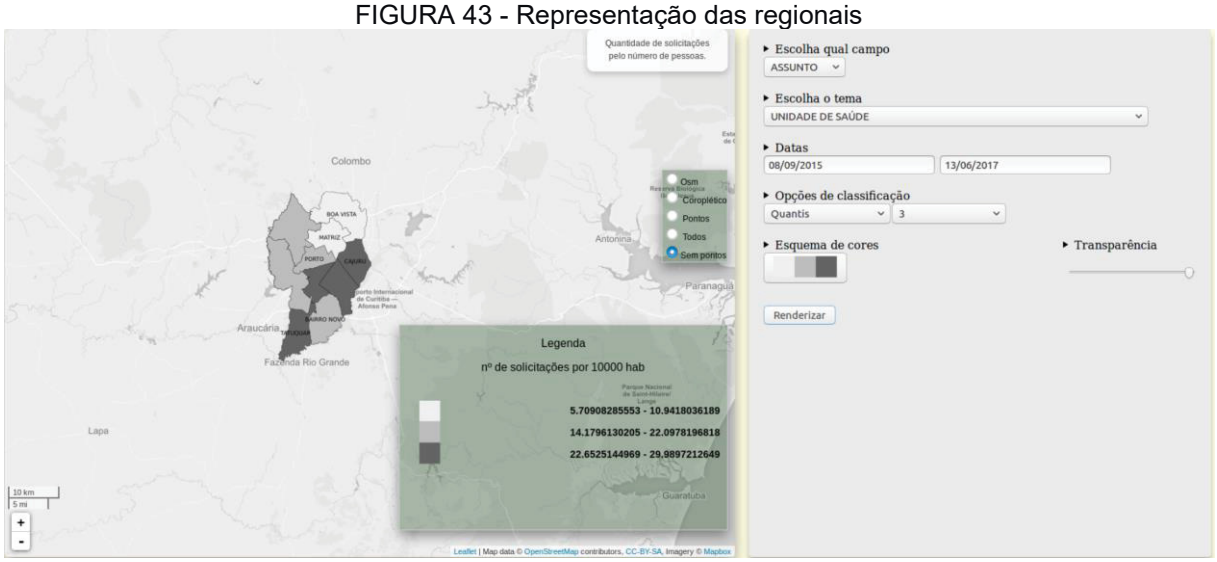
FIGURA 42 - Mapa coroplético 1



FONTE: O autor (2017)

O mapa representado na figura 43 foi gerado com o tema “Unidade de Saúde”, com três classes e método de classificação Quantis. A legenda desta representação contém “lacunas” de valores entre as classes, em consequência do método de classificação, e um número excessivo de algarismos significativos nos limites das classes, dificultando a leitura da legenda.

FIGURA 43 - Representação das regionais

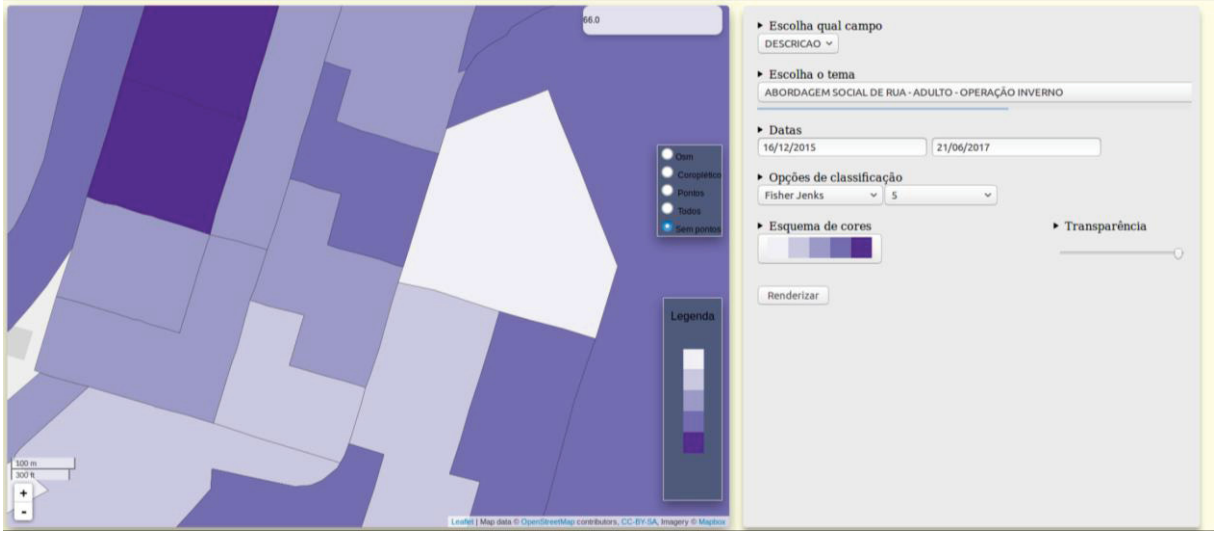


.

FONTE: O autor (2017)

O mapa seguinte (Figura 44), foi gerado com o tema “Abordagem social de rua-Adulto-Operação inverno”, com cinco classes e com o método de classificação Fisher-Jenks. Nesta representação ocorre uma falha na legenda, em que os limites das classes não são exibidos. Isto ocorre devido a uma falha no método do Leaflet que permite recuperar o nível de "zoom" no decorrer da interação com o mapa, consequentemente, o sistema não identifica qual divisão territorial está sendo representada e não pode determinar quais limites de classes deve exibir.

FIGURA 44 - Representação dos setores



FONTE: O autor (2017).

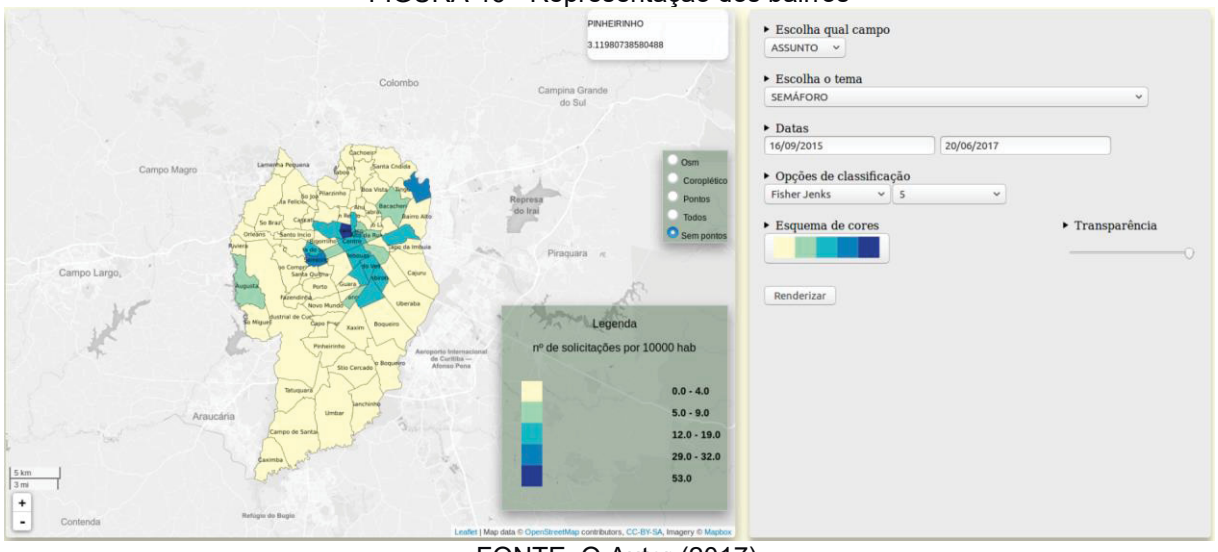
O menor nível de representação do fenômeno disponível no sistema é apresentado na figura 45, que foi gerado com o tema “Abordagem Social de rua-Adulto”, classificado com o método intervalos iguais e com cinco classes.

FIGURA 45 - Representação das grades hexagonais

FONTE: O autor (2017)

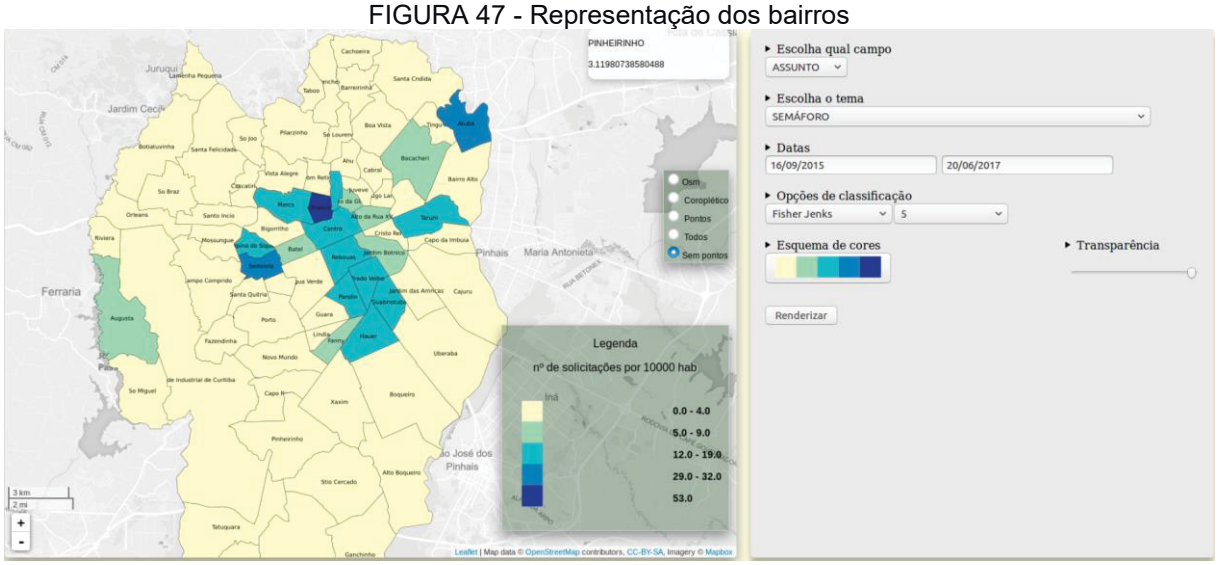
As figuras 46 e 47 apresentam a mesma divisão territorial em escalas distintas, gerado com o tema “Semáforo”, classificado com o método Fisher-Jenks e cinco classes. Os níveis de “zoom” em que os mapas estão representados são sequenciais.

FIGURA 46 - Representação dos bairros



. FONTE: O Autor (2017)

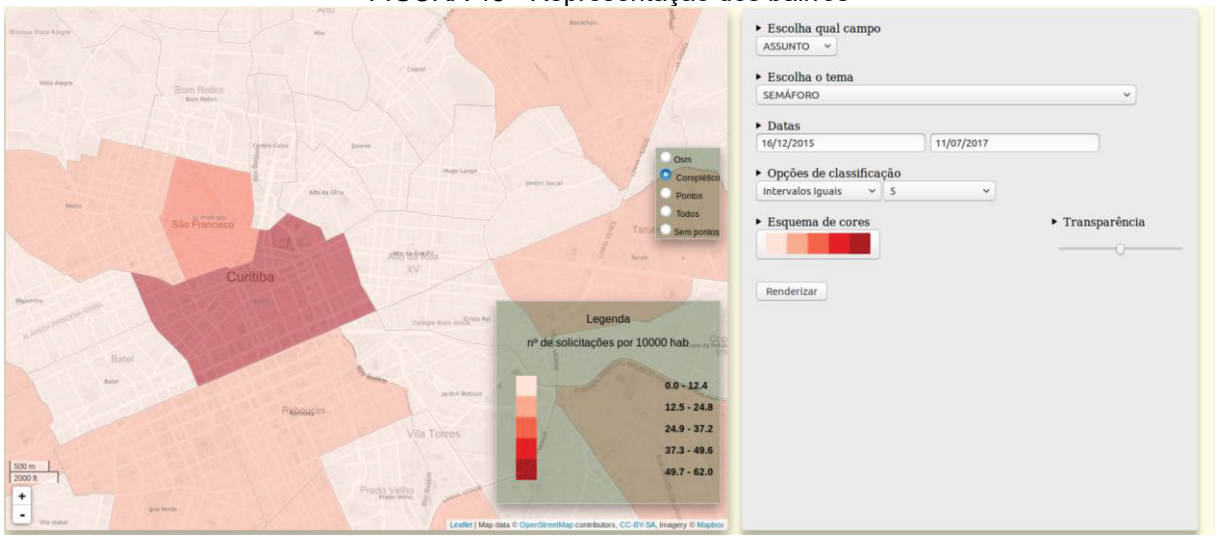
FIGURA 47 - Representação dos bairros



FONTE: O autor (2017).

O mapa da figura 48, representa a aplicação da transparência na camada do mapa coroplético, proporcionando explorar a base OSM com a sobreposição das camadas.

FIGURA 48 - Representação dos bairros



Fonte: O Autor (2017)

6.3 DESEMPENHO DO SISTEMA

O sistema foi desenvolvido em um ambiente virtual, utilizando o Oracle Virtualbox para virtualizar o sistema OSGeoLive. Esse ambiente não fornece todos os recursos de *hardware* para o sistema, porém permite configurá-lo para utilizar

uma quantidade de recursos. O quadro 6, mostra as configurações adotadas no ambiente de teste.

QUADRO 6 – Configurações do sistema Oracle Virtual box.

Processador	Nº de núcleos utilizados do processador	2
	Recurso estendidos (PAE)	habilitado
	Restrição de execução	40%
Aceleração	Interface de paravirtualização	Padrão
	Virtualização de hardware	habilitados
Placa mãe	Memória base	2048 Mb
	Recurso estendido	i/o APIC
Tela	Memória de vídeo	128 Mb
	Aceleração	Aceleração 3D
Armazenamento	Tamanho da máquina virtual	25 GB

FONTE: O autor (2017).

Para avaliar a desempenho do processo de agregação foi cronometrado o tempo necessário para executar os seus subprocessos em duas divisões territoriais: Regionais e Bairros. Os resultados são dispostos na tabela 5.

A tabela 1 apresenta a porcentagem do tempo total para realização de cada subprocesso de agregação, em diferentes definições de atributos. No subprocesso “Consultar pontos dentro do retângulo envolvente no MongoDB”, engloba-se o subprocesso de gerar um arquivo GeoJSON, composto pelo o resultado da consulta dos pontos dentro do retângulo envolvente no MongoDB, em razão da impossibilidade de desagregação dos subprocessos.

Tabela 1 - Proporções de tempo despendido em cada processo de agregação

DIVISÃO TERRITORIAL - REGIONAIS - 10 POLÍGONOS

PROCESSO	% DO TEMPO TOTAL PARA PROCESSAR 533 PONTOS	% DO TEMPO TOTAL PARA PROCESSAR 6271 PONTOS	% DO TEMPO TOTAL PARA PROCESSAR 1320 PONTOS
Recuperar retângulo envolvente no Postgres/Postgis	4,25	9,26	4,23
Consultar pontos dentro do retângulo envolvente no Mongodb	16,25	20,14	17,50
Enviar pontos para o Postgres/Postgis	28,02	33,67	35,82
Interseção dos pontos com polígono e contagem no Postgres/Postgis	49,75	33,68	40,79
Deletar pontos no Postgres/Postgis	1,74	3,26	1,64

DIVISÃO TERRITORIAL - BAIRROS - 75 POLÍGONOS

PROCESSO	% TEMPO PARA PROCESSAR 533 PONTOS	% TEMPO PARA PROCESSAR 6271 PONTOS	% DO TEMPO TOTAL PARA PROCESSAR 1320 PONTOS
Recuperar retângulo envolvente no Postgres/Postgis	10,94	12,78	11,16
Consultar pontos dentro do retângulo envolvente no Mongodb	30,93	26,75	32,43
Enviar pontos para o Postgres/Postgis	41,51	42,05	39,68
Interseção dos pontos com polígono e contagem no Postgres/Postgis	12,32	13,98	12,65
Deletar pontos no Postgres/Postgis	4,30	4,43	4,06

FONTE: O autor (2017).

A tabela 2 apresenta o tempo total para realizar o processo de agregação em diferentes definições de atributos a se representar e distintas divisões territoriais.

Tabela 2 - tempo total em segundos para o processamento da agregação

DIVISÃO TERRITORIAL	TEMPO PARA PROCESSAR 533 PONTOS (S)	TEMPO PARA PROCESSAR 6271 PONTOS (S)	TEMPO PARA PROCESSAR 1320 PONTOS (S)
REGIONAIS - 10 divisões	4,224775	7,023579	10,027734
BAIRROS - 75 divisões	21,449511	20,301935	24,164197

FONTE: O autor (2017).

7 CONCLUSÕES E RECOMENDAÇÕES

Esta pesquisa objetivou propor uma metodologia para solucionar as limitações da visualização de dados pontuais do Geo Big Data, resultantes das suas características. O resultado alcançado foi o desenvolvimento de um sistema para representação coroplética automatizada, que agrega os dados pontuais por áreas estatísticas e os normaliza pela população residente nestas áreas, desta forma cumpriu-se os objetivos propostos. Embora que os dados utilizados para alimentar o sistema podem não ser considerados estritamente Big Data, foi o suficiente para demonstrar as proposições da integração com um banco de dados não estruturado.

O sistema permite o usuário explorar os dados, através de filtros para representá-los, classificá-los com diferentes métodos e número de classes, além de prover ferramentas para interagir com as representações. No entanto, a interação do usuário com os mapas é um ponto crítico do sistema, em razão das limitações das componentes utilizadas. Como no caso da mudança da escala, a qual é determinada pelas escalas predefinidas da base OSM, podendo dificultar a visualização de uma certa região ajustada ao *frame* do mapa. A metodologia desenvolvida permite agregar dados de qualquer banco de dados não relacional que disponha do mínimo suporte a dados espaciais, ou seja, que armazene dados espaciais pontuais e execute consultas que recupere dados que estejam contidos dentro de um polígono.

7.1 RECOMENDAÇÕES

Como visto no teste de desempenho, o tempo de execução do processo de agregação é alto, aproximadamente 22 segundos para agregar por bairros, relativamente alto para aplicações que precisam de processamento e visualizações quase instantaneamente (ROBINSON et al, 2017). Os subprocessos envolvidos no intercâmbio de dados e no processamento no Postgres/Postgis consomem mais tempo em relação ao tempo total do processo. Então, é recomendável a criação de pesquisas para a otimização do processo, de modo que o método trate a característica de velocidade do Geo Big Data e que o sistema seja testado em sistemas de armazenamento distribuído.

Recomenda-se também o desenvolvimento de métodos eficazes para determinação do nível de “zoom” durante a interação com o mapa, devido ao fato que diversas outras funções do sistema dependerem desta funcionalidade, como a exibição correta da legenda e da consulta do atributo através da seleção de feições, assim favorecendo a exploração dos mapas.

O Geo Big Data apresenta um novo conjunto de desafios e oportunidades para a comunidade de pesquisa em cartografia, entretanto o contínuo desenvolvimento de soluções *open source*, combinando técnicas tradicionais de cartografia com a capacidade estendida das novas formas de armazenamento e processamento de grandes volumes de dados têm a capacidade de contribuir com novas soluções para problemas urgentes que requerem novas abordagens.

REFERÊNCIAS

ABRAMOVA, V.; BERNARDINO, J.; FURTADO, P. Which NoSQL Database? A Performance Overview. **Open Journal Of Databases (ojdb)**. [S. L.], p. 1-8. Maio 2014. Disponível em: <https://www.ronpub.com/journals/ojdb>. Acesso em: 31 out. 2016.

BAAS, B. **NoSQL spatial: Neo4j versus PostGIS.** 2012. 121 f. Tese (Doutorado) - Curso de Geographical Information Management And Applications, Universidade de Utrecht, Utrecht, 2012.

BAPTISTA, C. S. et al. NoSQL Geographic Databases: An Overview. In: POURABBAS, Elaheh. **Geographical Information Systems Trends and Technologies**. Boca Raton: Taylor & Francis Group, 2014. Cap. 4. p. 73-103.

BRASIL. **Decreto Nº 6.666, de 27 de Novembro de 2008**. Brasília, DF, Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2008/Decreto/D6666.htm>. Acesso em: 30 out. 2016.

BREWER, C. A.. Color use guidelines for mapping. In: MACEACHREN, A. M., TAYLOR, D.R.F. (ed.) **Visualization in modern cartography**, Tarrytown, NY: Elsevier Science, 1994. p. 123–147.

CALÇADO, P. Arquitetura de Camadas em Java EE. **Mundo Java**. Rio de Janeiro, v.3, n.15, 2005

CAMBOIM, S. P. **Arquitetura para integração de dados interligados abertos à INDE-BR**. 2013. 141 f. Tese (Doutorado) - Curso de Ciências Geodésicas, Geomática, Universidade Federal do Paraná, Curitiba, 2013. Disponível em: <http://acervodigital.ufpr.br/bitstream/handle/1884/32104/R - T - SILVANA PHILIPPI CAMBOIM.pdf?sequence=1>. Acesso em: 31 out. 2016.

CARR, D. B.; OLSEN, A. R.; WHITE, D. Hexagon mosaic maps for display of univariate and bivariate geographical data. **Cartography and Geographic Information Systems**, v. 19, p. 228-236, 1992.

CASANOVA, M. A. et al**. Bancos de Dados Geográficos.** Curitiba: Mundogeo, 2005. Disponível em: <http://www.inf.puc-rio.br/~casanova/Publications/Books/2005-BDG.pdf>. Acesso em: 31 out. 2016.

CATTELL, R. **Scalable SQL and NoSQL Data Stores.** Sigmod Record, [s. L.], v. 39, n. 4, p.12-27, dez. 2010.

CHEN, M.; MAO, S.; LIU, Y. **Big Data: A Survey. Mobile Networks And Applications**, [s.l.], v. 19, n. 2, p.171-209, 22 jan. 2014. Springer Science + Business Media. http://dx.doi.org/10.1007/s11036-013-0489-0.

CONCAR. **ET-EDGV: Especificações técnicas para estruturação** D. 2.0 ed. 2007. 2013 p. Disponível em:

<http://www.concar.ibge.gov.br/temp/94@EDGV_V20_10_10_2007.pdf>. Acesso em: 29 out. 2016.

CURITIBA, P. D. E. Bairros. Disponível em: <http://www.curitiba.pr.gov.br>. Acesso em: 28/6/2017.

DASGUPTA, A. **Big data: The future is in analytics. 2013**. Disponível em: <https://www.geospatialworld.net/article/big-data-the-future-is-in-analytics/>. Acesso em: 31 out. 2016.

DENT, B. D.; TORGUSON, J. S.; HODLER, T. W. **Cartography: thematic map design**. Boston: McGraw Hill, 2009.

DEVELOPERS, P. G. I. S. **PostGIS Project Steering Committee (PSC)**. Disponível em: <http://postgis.net/windows_downloads/>. Acesso em: 28/6/2017.

DOBBS, R. et al. **Big data: The next frontier for innovation, competition, and productivity**. 2011. Disponível em: <https://www.mckinsey.com/>. Acesso em: 31 out. 2016.

FARKAS, G. **Applicability of open-source web mapping libraries for building massive Web GIS clients**. Journal of Geographical Systems, v. 19, n. 3, p. 273–295, 2017.

EPSG:3857. 2016. Disponível em: <http://wiki.openstreetmap.org/wiki/EPSG:3857#History>. Acesso em: 30 out. 2016.

GOODCHILD, M. F.. **GIS in the Era of Big Data. 2016.** Disponível em: <http://cybergeo.revues.org/27647>. Acesso em: 31 out. 2016.

HAHMANN, S.; BURGHARDT, D. **How much information is geospatially referenced?** Networks and cognition. e International Journal Of Geographical Information Science. [s. L.], p. 1171-1189. nov. 2012. Disponível em: <http://www.tandfonline.com/doi/citedby/10.1080/13658816.2012.743664?scroll=top&needAccess=true>. Acesso em: 31 out. 2016.4

HOWARTH, R. J. **DICTIONARY OF MATHEMATICAL GEOSCIENCES: with historical notes**. S.l.: SPRINGER, 2017.

IBGE. **Operação censitária**. Disponível em: <http://censo2010.ibge.gov.br>. Acesso em: 28/6/2017.

ICI - INSTITUTO DAS CIDADES INTELIGENTES. Central de Atendimento 156 da Prefeitura de Curitiba. Disponível em: <http://www.central156.org.br/>. Acesso em: 4/7/2017.

JAVASCRIPT TUTORIAL. .Disponível em: <https://www.w3schools.com/js/>. Acesso em: 28/6/2017.

JULIAN, G. (Org.). **LevelUP: Fast & simple storage - a Node.js-style LevelDB wrapper.** 2016. Disponível em: <https://github.com/Level/levelup/blob/master/README.md>. Acesso em: 31 out. 2016.

KIRSTEN S. (Califórnia). Project Coordinator (Ed.). **How much information 2003?** 2003. Disponível em: <http://groups.ischool.berkeley.edu/archive/how-much-info-2003/>. Acesso em: 29 out. 2016.

LANEY, D. **3D management Controling Data Volume, Velocidade, and Variety. 2001.** Disponível em: <https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf>. Acesso em: 01 nov. 2016.

Leaflet/Leaflet. Disponível em: <https://github.com/Leaflet/Leaflet>. Acesso em: 29/6/2017.

LI, S. et al. **Geospatial big data handling theory and methods: A review and research challenges.** Isprs Journal Of Photogrammetry And Remote Sensing. [s. L.], p. 0-0. fev. 2015.

MACMASTER, R. B.; SHEA, K. S. **Generalization in digital cartography**. Washington, DC: Assoc. of American Geographers, 1992.

MAZIERO, L. T. P.. Influência **dos aspectos das interfaces na comunicação dos mapas interativos e a proposição de diretrizes para o design dessas interfaces.** 2007. 213 f. Tese (Doutorado) - Curso de Ciências Geodésicas, Geomática, Universidade Federal do Paraná, Curitiba, 2007.

Metodologia do censo demográfico 2010. Rio de Janeiro: Instituto Brasileiro de Geografia e Estatística-IBGE, 2013.

MONGODB. Disponível em: <https://www.mongodb.com/>. Acesso em: 28/6/2017.

OPENLAYERS2 Spherical Mercator. Disponível em: <http://docs.openlayers.org/library/spherical_mercator.html>. Acesso em: 16 nov. 2016.

O'REILLY What Is Web 2.0?. Disponível em: <http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html>. Acesso em: 31 out. 2016.

ORACLE Getting Started with Berkeley DB for Java. Disponível em: <http://docs.oracle.com/cd/E17076_05/html/gsg/JAVA/BerkeleyDB-Core-JAVA-GSG.pdf>. Acesso em: 31 out. 2016.

ORIENTDB Manual. Disponível em: <http://orientdb.com/docs/last/index.html>. Acesso em: 29 out. 2016.

PANTALEÃO, E. **Aplicação de técnicas de sistemas baseados em conhecimento em projeto cartográfico temático.** 104 f. Dissertação (Mestrado) - Curso de Ciências Geodésicas, Geomática, Universidade Federal do Paraná, Curitiba, 2003.

POSTGIS**.** Disponível em: <http://www.refractions.net/products/postgis/history/>. Acesso em: 31 out. 2016.

Pysal/pysal. Disponível em: <https://github.com/pysal>. Acesso em: 24/6/2017.

Python .Disponível em: <https://www.python.org/about>. Acesso em: 3/7/2017.

QUEIROZ, G R.; MONTEIRO, A. M. V. CÂMARA, G. Bancos de dados geográficos e sistemas nosql: onde estamos e para onde vamos. **Revista Brasileira de Cartografia**, [s. L.], v. 3, n. 65, p.479-492, set. 2012.

RAMOS, M. P. et al. Distributed Systems Performance for Big Data. **13th International Conference On Information Technology**. Brasil, maio 2016. p. 733-744.

ROBINSON, A. C.; DEMŠAR, U.; MOORE, A. B.; et al. Geospatial big data and cartography: research challenges and opportunities for making maps that matter. **International Journal of Cartography**, p. 1–29, 2017.

ROBINSON, A. H. et al. **Elements of cartography**. Hoboken, NJ: John Wiley & Sons, 1995.

SHEKHAR, S. et al. Spatial Big-Data Challenges Intersecting Mobility and Cloud Computing. **Proceedings of the 11th ACM International Workshop on Data Engineering for Wireless and Mobile Access.** [s. l.], jul. 2012. p. 1-6.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN. **Sistema de Banco de Dados.** 6. ed. Rio de Janeiro: Elsevier, 2006.

SLOCUM, T. A. et al. **Thematic cartography and geovisualization.** 3. ed. [s l]: Pearson Prentice Hall, 2009.

SLUTER, C. R. Uma abordagem sistêmica para o desenvolvimento de projeto cartográfico como parte do processo de comunicação cartográfica. **Portal da Cartografia**, Londrina, v.1, n.1, jun. 2008. Disponível em: <http://www.educadores.diaadia.pr.gov.br/arquivos/File/2010/artigos_teses/teses_geografia2008/artigoclaudia.pdf>. Acesso em: 29 out. 2016.

SLUTER, C. R.; VAN ELZAKKER, C. P. J. M.; IVANOVA, I. Requirements elicitation for geo-information solutions**. The Cartographic Journal**. [s. l.], p. 1-14. jul. 2014.

THE Neo4j Operations Manual v3.0. 2016. Disponível em: <http://neo4j.com/docs/operations-manual/current/>. Acesso em: 30 out. 2016.

TILE Map Service Specification. .Disponível em: <wiki.osgeo.org/wiki/Tile_Map_Service_Specification>. Acesso em: 26/6/2017.

TIWARI, S. **Professional NoSQL.** Indianapolis,: John Wiley & Sons, Inc., 2011. 386 p.

WIKIPÉDIA (Alemanha) (Ed.). **Wikipédia Alemã**. 2016. Disponível em: <https://de.wikipedia.org/w

W3C. .Disponível em: <https://www.w3.org/standards/webdesign/htmlcss>. Acesso em: 29/6/2017.

APÊNDICE 1 – CÓDIGOS DE PROGRAMAÇÃO DO SISTEMA

Os códigos aqui disponibilizados, são os fragmentos principais do código inteiro, devido a grande quantidade de linhas de programação.

1. Classe principal

Este código contém o principal funcionamento do sistema. Realiza todos o processos: Agregação, Classificação, Modificação do estilo e os procedimentos para renderizar. Porém, depende de outros códigos, com subprocesso para realizar as suas funções.

```
from .po import pontos
from .pol import poligonos
from config import configuracoes
import numpy as np
from .class_map import mapa
import time
import psycopg2

class mod_mapa(object):


    def __init__(self,teste):

        self.config=configuracoes()
        self.teste=teste

        self.str_connect=("dbname = %s user= %s host= %s password= %s")%(self.config.db_post,self.config.user_post,self.config.host_post,self.config.senha_post)

    def contar(self,campo_mongo,consulta,mongo,data_ini,data_fim):

        unidade_geo=self.config.unidades_geo
        tam_unidade_geo=len(unidade_geo)
        cont_unidade_geo=0

        while  cont_unidade_geo<tam_unidade_geo:


pol=poligonos(unidade_geo[cont_unidade_geo],7,self.str_connect,self.config.coluna_contagem)
            limite=pol.cont_pol()
            #print limite
            cont=1
            contagem=[]
            while cont < limite+1:
            #while cont< 2:

pol=poligonos(unidade_geo[cont_unidade_geo],cont,self.str_connect,self.config.coluna_contagem)
```

```
                    #ini0 = time.time()
                    po=
pontos(pol.gid,self.str_connect,self.config.db_mongo,self.config.colecao_mongo,pol.sbox(),self.config.
user_post,self.config.db_post,campo_mongo,consulta_mongo,self.config.campo_data,data_ini,data_fi
m)
                    #fim0=time.time()
                    #print "pega pontos no nosql", fim0-ini0
                    #ini1 = time.time()
                    po.consulta_sbox()
                    #fim1=time.time()
                    #print "consulta box" ,fim1-ini1
                    #ini2 = time.time()
                    po.enviar_banco()
                    #fim2=time.time()
                    #print "enviar banco",fim2-ini2
                    #ini3 = time.time()

                    #print
                    pol.contar_pts()
                    #fim3=time.time()
                    #print "contar pts",fim3-ini3
                    #ini4 = time.time()
                    po.apagar_pts()
                    #fim4=time.time()
                    #print "apagar pts",fim4-ini4
                    cont=cont+1
                    #print cont
                cont_unidade_geo=cont_unidade_geo+1
            try:
                reinicia_mapa()
            except:
                conf_render()
            #print contagem
        #contar()

        def classificar(self,n_class,met_class):
            unidade_geo=self.config.unidades_geo
            tam_unidade_geo=len(unidade_geo)
            cont_unidade_geo=0
            legenda=[]
            while  cont_unidade_geo<tam_unidade_geo:
                self.str_connect=("dbname = %s user= %s host= %s password=
%s")%(self.config.db_post,self.config.user_post,self.config.host_post,self.config.senha_post)
                conn = psycopg2.connect(self.str_connect)
                cur = conn.cursor()
                sql =( "SELECT cont from %s ;"%(unidade_geo[cont_unidade_geo]))
                cur.execute(sql)

                conn.commit()

                rows = cur.fetchall()
                cont_unidade_geo1=0
                tam_unidade_geo1=len(rows)
                dados=[]
                while cont_unidade_geo1<tam_unidade_geo1:
                    dados.append(str(rows[cont_unidade_geo1][0]))
                    cont_unidade_geo1=cont_unidade_geo1+1
                cur.close()
                conn.close()
```

```
            classes=mapa(n_class,cont_unidade_geo,met_class,dados)

            lim_sup=classes.classifica()
            #print lim_sup
            limites=classes.retorna_limites(lim_sup[0])
            classes.modifica_stilo(limites[0],limites[1])
            cont_unidade_geo=cont_unidade_geo+1
            legenda.append(limites[2])
        try:
            reinicia_mapa()
        except:
            conf_render()
        return legenda

    def reinicia_mapa(self):
        from subprocess import call

        call("(sudo rm -r /var/lib/mod_tile/default/*)",shell=True)
        call("(sudo -u user renderd -f -c /usr/local/etc/renderd.conf)",shell=True)

    def conf_render(self):

            call("(sudo mkdir /var/run/renderd/)",shell=True)
            call("(sudo chown user /var/run/renderd/)",shell=True)

    def muda_cor(self,cor):
        from lxml import etree
        tree = etree.parse("/media/sf_pasta/teste_xml/style.xml")

        root = tree.getroot()
        i=0
        for node in root.iter():
            #print node.tag
            for child in node.iter():

                if child.tag == "StyleName" and child.text != "texto":

                    if i<3:
                        i=i+1
                        child.text=cor+str(i)

        outFile = open('/media/sf_pasta/teste_xml/style.xml', 'w')
        tree.write(outFile, xml_declaration=True, encoding='utf-8')
        outFile.close()
        try:
            reinicia_mapa()
        except:
            conf_render()
    #
```

2. Classe para classificação

Este código contém os subprocessos para a classificação dos dados.

```
import pysal
import array
import numpy as np
```

```
import lxml
from lxml import etree
from config import configuracoes


class mapa(object):


    def __init__(self,n_clas,unidade_geo,met_clas,dados):
        self.n_clas=n_clas
        self.met_clas=met_clas
        self.dados=np.array(dados).astype(np.float)
        self.unidade_geo=str(unidade_geo)
        self.mult=1
        #self.cores=cores


    def classifica(self):
        #print self.dados
        if self.met_clas=='Fisher Jenks':
            intervalos= pysal.Fisher_Jenks(self.dados,self.n_clas)
            lim_maior=intervalos.bins
            ele_n_classe=intervalos.counts
            #gadf=intervalos.gadf

        if self.met_clas=='Intervalos Iguais':
            intervalos= pysal.Equal_Interval(self.dados,self.n_clas)
            lim_maior=intervalos.bins
            ele_n_classe=intervalos.counts
            #gadf=intervalos.gadf

        return [lim_maior,ele_n_classe]


    def retorna_limites(self,lim_maior):

        maior_sign=0

        for i in lim_maior:
            tam=(str(i).split('.'))[1]
            maior_sign1=len(tam)
            if maior_sign1>maior_sign:
                maior_sign=maior_sign1
        cont=0
        tam_lim_m=len(lim_maior)
        lim_menor=[min((self.dados))]


        if maior_sign>1:
            mult=10**maior_sign*(-1)
            while cont<(tam_lim_m-1):
                if cont==0:
                    lim_menor.append(-1)
                else:
                    lim_menor.append(lim_maior[cont]+mult)
                cont=cont+1


        else:
            while cont<(tam_lim_m-1):
```

```
            if cont==0:
                lim_menor.append(-1)
            else:
                lim_menor.append(int(lim_maior[cont]+1))
            lim_maior=lim_maior.astype(int)
            cont=cont+1

    lim_maior= lim_maior.tolist()

    #max(np.extract(self.dados<lim_sup[cont_limsup]
    #return [lim_menor,lim_maior]
    legenda=[]
    if self.met_clas=='Fisher Jenks':
        tam_lim=len(lim_maior)
        cont_leg=0

        while cont_leg<tam_lim:
            sup= max(np.extract( self.dados<=lim_maior[cont_leg], self.dados))
            if cont_leg==0:
                inf=min(self.dados)
            else:
                inf=min(np.extract( self.dados>lim_maior[cont_leg-1], self.dados))
            if sup==inf:
                legenda.append(sup)
            else:
                legenda.append(str(inf)+' - '+str(sup))
            cont_leg=cont_leg+1
    else:
        tam_lim=len(lim_maior)
        cont_leg=0

        while cont_leg<tam_lim:
            if cont_leg==0:
                inf=min(self.dados)
                sup=lim_maior[cont_leg]
                legenda.append(str(inf)+' - '+str(sup))
            else:
                inf=lim_menor[cont_leg]
                sup=lim_maior[cont_leg]
                legenda.append(str(inf)+' - '+str(sup))

    #print legenda
    limites=[lim_menor,lim_maior,legenda]
    return limites

def modifica_stilo(self,lim_inf,lim_sup):

    config=configuracoes()
    unid=config.unidades_geo
    tam_unidade_geo=len(self.unidade_geo)
    cont_unidade_geo=0

    while (cont_unidade_geo<tam_unidade_geo):
        tree = etree.parse("/media/sf_pasta/teste_xml/style.xml")

        root = tree.getroot()
        lim_sup=lim_sup
        lim_inf=lim_inf

        c=0
```

```
                tree = etree.parse("/media/sf_pasta/teste_xml/style.xml")

                root = tree.getroot()
                for node in root.iter():
                    #print node.tag
                    for child in node.iter():

                        if child.tag == "Style":
                            if 'n_classe' in child.attrib:
                                if child.attrib['n_classe'] == str(self.n_clas)+str(cont_unidade_geo):

                                    for parent in child.iter():
                                        if parent.tag == 'Filter':
                                            if c < len(lim_inf):
                                                #print lim_inf[c]
                                                #print lim_sup[c]
                                                parent.text=(("[cont] > %f and [cont] < %f")%(lim_inf[c]-
1,lim_sup[c]+1))
                                                c=c+1
                                            if c >= len(lim_inf):
                                                c=0

                cont_unidade_geo=cont_unidade_geo+1
                outFile = open('/media/sf_pasta/teste_xml/style.xml', 'w')
                tree.write(outFile, xml_declaration=True, encoding='utf-8')
                outFile.close()
```

3. Classe para procedimentos iniciais dos dados pontuais

Este código contém os subprocessos para a procedimento iniciais dos dados no MongoDB, como consultar pontos dentro de um retângulo envolvente, enviar os pontos para o PostGIS .

```
        import ogr
        import os
        import json
        import psycopg2
        from pymongo import MongoClient
        import time
        from datetime import datetime


        class pontos(object):



            def
__init__(self,cont,str_connect,db_mongo,colecao_mongo,sbox,user_post,db_post,campo_mongo,
consulta_mongo,campo_data,data_ini,data_fim ):

                self.gid=cont
                self.str_connect=str_connect
                self.db_mongo=db_mongo
                self.colecao_mongo=colecao_mongo
                self.campo_mongo=campo_mongo
```

```
            self.consulta_mongo=consulta_mongo
            self.campo_data=campo_data
            self.data_ini=data_ini
            self.data_fim=data_fim
            self.sbox= sbox
            self.db_post=db_post
            self.user_post=user_post


        def consulta_sbox(self):
            #ini=time.time()
            client = MongoClient()
            db = client[self.db_mongo]
            coll=db[self.colecao_mongo]

            sbox1 =self.sbox.split(',')
            a= float(str(sbox1[0]).replace("[",""))
            b=float(str(sbox1[1]).replace("]",""))
            c= float(str(sbox1[2]).replace("[",""))
            d=float(str(sbox1[3]).replace("]",""))
            box=[[a,b],[c,d]]

            query = {"geometry": { "$geoWithin":{"$box":box}},
self.campo_mongo:self.consulta_mongo, self.campo_data:{"$gt":self.data_ini , "$lt":self.data_fim } }

            consult=[]

            for doc in db.base156.find(query, {'properties.data_formatada':0}).sort('_id'):
                consult.append(str(doc))


            consult=(str(consult))
            consult=consult.replace("{u'",'{"')
            consult=consult.replace("'{'",'{"')
            consult=consult.replace("'",'"')
            consult=consult.replace(': u"',': "')
            consult=consult.replace(', "u"',", ")
            consult=consult.replace(', u"',', "')
            consult=consult.replace('u"_id"','"_id"')
            consult=consult.replace('ObjectId','')
            consult=consult.replace('(','')
            consult=consult.replace(')','')
            consult=consult.replace('}}"','}}')


            texto =('{  "type": "FeatureCollection",  "features":%s}'% (consult))

            arq = open('consulta.geojson', 'w')
            arq.write(str(texto))
            arq.close()

        def enviar_banco(self):
            command = (('ogr2ogr -f "PostgreSQL" PG:"dbname=%s user=%s" "consulta.geojson" -
nln "p%i"')% (self.db_post ,self.user_post, self.gid))
            os.system(command)

        def apagar_pts(self):
            try:
                conn = psycopg2.connect(self.str_connect)
                cur = conn.cursor()
```

```
        sql=(("DROP TABLE IF EXISTS p%i")%(self .gid))

        cur.execute(sql)
        cur.close()
        conn.commit()
        conn.close()
    #break
    except:
        print "I  am unable to connect to the database2"
```

4. Classe para procedimentos iniciais dos poligonos

Este código contém os subprocessos para a procedimento iniciais dos dados no PostGIS, como consultar quanto poligonos tem um tabela, consultar os pontos que estão contidos no poligono e recuperar o retângulos envolvente dos poligonos.

```
import psycopg2

class poligonos (object):


    def __init__(self,unidade_geo,cont,str_connect,coluna_contagem):

        self.gid=cont
        self.unidade_geo=unidade_geo
        self.str_connect=str_connect
        self.coluna_contagem=coluna_contagem

    def sbox(self):

        #try:
            conn = psycopg2.connect(self.str_connect)
            cur = conn.cursor()
            sql =( "SELECT Box2D(geom) from %s where id=%i;"%(self.unidade_geo,self.gid))
            cur.execute(sql)

            conn.commit()

            rows = cur.fetchmany()
            cur.close()
            conn.close()
            consulta=str(rows)
            tam=len(consulta)
            consulta=consulta[6:tam-3]
            consulta=consulta.replace("""","")
            consulta=consulta.replace("(","[")
            consulta=consulta.replace(")","]")
            consulta=consulta.replace(",","],[")
            consulta=consulta.replace(" ",",")
            return consulta

        #except:
            #print "sbox"
```

```
        def cont_pol(self):
            #print "entrei"
            #try:
                conn = psycopg2.connect(self.str_connect)
                cur = conn.cursor()
                sql = ("SELECT * from %s ;"%(self.unidade_geo))
                cur.execute(sql)
                rows = (cur.rowcount)

                conn.commit()
                cur.close()
                conn.close()
                return rows

            #except:
                #print "cont pol"
            #print "entrei"



        def contar_pts(self):
            #try:
                conn = psycopg2.connect(self.str_connect)
                cur = conn.cursor()
                sql = ("select relname from pg_class where relname = '%s' and relkind='r';"%
(self.unidade_geo))
                cur.execute(sql)
                rows = cur.rowcount
                if(rows==0):
                    print "Os pontos nao foram inseridos"
                else:
                    sql=("select ogc_fid from p%i,%s where %s.id=%i and
ST_Contains(geom,wkb_geometry)= true;"%(self.gid,self.unidade_geo,self.unidade_geo,self.gid))
                cur.execute(sql)
                rows = cur.rowcount
                sql=("UPDATE %s SET %s = %i WHERE id = %i;" %
(self.unidade_geo,self.coluna_contagem,rows,self.gid ))
                cur.execute(sql)
                conn.commit()
                cur.close()
                conn.close()
                return str(rows)


            #except:
```

5. Views do Django

Este código contém a camada view do Django, a qual recebe requisições da interface, envia pedidos pra classe principal e determina qual html ou template será enviado para a interface.

```
# -*- coding: utf-8 -*-
from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.http import HttpResponse
from pymongo import MongoClient
from .forms import NameForm
```

```
from .principal import mod_mapa
import json
import os

mapa=mod_mapa()

def mod_conf(campo,valor):
    filename = '/home/user/www/myteste/tenta/coro/conf1.json'
    with open(filename, 'r') as f:
        data = json.load(f)
        ant=data[campo]
        if ant != valor:
            data[campo] = valor
            result=["true",valor]
        else:
            result= ["false",ant]
    #os.remove(filename)
    str(f).encode('utf8')
    with open(filename, 'w') as f:
        json.dump(data, f, indent=4)
    return result

def post_list(request):
    print('hello')
    return render(request, 'novos/mapas.html')

def pegar_unicos(request):
    client = MongoClient()
    db = client['base156']
    coll='base156'
    vcampo=request.GET['suggest']
    valores_unicos=[]
    for doc in  db.base156.distinct(vcampo):
        valores_unicos.append(doc+"!")
    return HttpResponse(valores_unicos)


def mod_mapa (request):
    campo=request.GET['campo']
    consulta=request.GET['consulta']
    d_i=request.GET['data_ini']
    d_f=request.GET['data_fim']
    n_clas=int(request.GET['n_clas'])
    met_clas=request.GET['met_clas']
    cor=request.GET['cor']

    veri_campo=mod_conf("campo",campo)
    veri_consulta=mod_conf("consulta",consulta)
    veri_num_clas=mod_conf("num_clas",n_clas)
    veri_met_clas=mod_conf("met_clas",met_clas)
    veri_data_ini=mod_conf("data_ini",d_i)
    veri_data_fim=mod_conf("data_fim",d_f)
    veri_cor=mod_conf("cor",cor)
    legenda=[]
    #print
veri_campo,veri_consulta,veri_num_clas,veri_met_clas,veri_data_ini,veri_data_fim,veri_cor
    if (veri_campo[0]=="true" or veri_consulta[0]=="true" or veri_data_ini[0]=="true" or
veri_data_fim[0]=="true" and veri_num_clas[0]=="true" and veri_met_clas[0]=="true" and
veri_cor[0]=="true"):
```

```
            #print veri_campo[1],veri_consulta[1],veri_data_ini[1],veri_data_fim[1]
            mapa.contar(veri_campo[1],veri_consulta[1],veri_data_ini[1],veri_data_fim[1])
            legenda = mapa.classificar(veri_num_clas[1],veri_met_clas[1])
            mapa.muda_cor(veri_cor[1])
            mapa.reinicia_mapa()


        if (veri_campo[0]=="false" and veri_consulta[0]=="false" and veri_data_ini[0]=="false" and
veri_data_fim[0]=="false" and veri_num_clas[0]=="false" or veri_met_clas[0]=="false" or
veri_cor[0]=="true"):
            mapa.muda_cor(veri_cor[1])
            #mapa.reinicia_mapa()

        return HttpResponse(valores_unicos)
    mod_conf("data_ini","10/04/1950")
```

6. “Style” do Mapnik

Este código contém a um fragmento do arquivo utilizado pelo MapniK para renderizar os mapas. O arquivo é composto por um estilo com três classes e uma fonte de dados para ser renderizada.

```
    <?xml version='1.0' encoding='UTF-8'?>
    <!DOCTYPE Map>
    <Map srs="+proj=merc +a=6378137 +b=6378137 +lat_ts=0.0 +lon_0=0.0 +x_0=0.0 +y_0=0
+k=1.0 +units=m +nadgrids=@null +wktext +no_defs +over" background-color="transparent"
minimum-version="0.7.2">



    <Style cor="YlGnBu" n_classe="31" name="YlGnBu31">

        <Rule classe="1">
         <Filter>[cont] &gt; 0.000000 and [cont] &lt; 1.000000</Filter>
         <PolygonSymbolizer fill="#edf8b1"/>
         <LineSymbolizer stroke="black" stroke-width=".3"/>


        </Rule>

     <Rule classe="2">
        <Filter>[cont] &gt; 2.000000 and [cont] &lt; 5.000000</Filter>
        <PolygonSymbolizer fill="#7fcdbb"/>
        <LineSymbolizer stroke="black" stroke-width=".3"/>


        </Rule>

     <Rule classe="3">
          <Filter>[cont] &gt; 6.000000 and [cont] &lt; 16.000000</Filter>
          <PolygonSymbolizer fill="#2c7fb8"/>
          <LineSymbolizer stroke="black" stroke-width=".3"/>
```

```
        </Rule>

    </Style>


    <Layer name="setores" status="off" srs="+proj=latlong +ellps=WGS84 +datum=WGS84
+no_defs" minimum-scale-denominator="0" maximum-scale-denominator="15000000">
        <!-- Style order determines layering hierarchy -->
        <!-- Labels go on top so they are listed second -->
        <StyleName>BuPu53</StyleName>
        <Datasource>
          <Parameter name="type">postgis</Parameter>
          <Parameter name="password">postgres</Parameter>
          <Parameter name="host">localhost</Parameter>
          <Parameter name="port">5432</Parameter>
          <Parameter name="user">postgres</Parameter>
          <Parameter name="dbname">teste</Parameter>
          <Parameter name="table">lim_limite_operacional_a</Parameter>
      <Parameter name="geometry_field">geom</Parameter>
          <Parameter name="estimate_extent">false</Parameter>
      <Parameter name="srid">4326</Parameter>

        </Datasource>
    </Layer>

    </Map>
```