

Wancharle Sebastião Quirino

SEARCHLIGHT: Melhorando a visualização de informações crowdsourcing em Mapas Web através de Zoom seletivo

Vitória - ES, Brasil

7 de Junho de 2013

Wancharle Sebastião Quirino

SEARCHLIGHT: Melhorando a visualização de informações crowdsourcing em Mapas Web através de Zoom seletivo

Monografia apresentada para obtenção do
Grau de Bacharel em Engenharia de Computação pela Universidade Federal do Espírito Santo.

Orientador: Celso Alberto Saibel Santos

Universidade Federal do Espírito Santo

Centro Tecnológico

Departamento de Informática

Vitória - ES, Brasil

7 de Junho de 2013

Wancharle Sebastião Quirino

SEARCHLIGHT: Melhorando a visualização de informações crowdsourcing em Mapas Web através de Zoom seletivo

Monografia apresentada para obtenção do Grau de Bacharel em Engenharia de Computação pela Universidade Federal do Espírito Santo.

Trabalho aprovado. Vitória - ES, Brasil, 7 de junho de 2013:

Celso Alberto Saibel Santos
Orientador

José Gonçalves Pereira Filho
Convidado 1

Magnos Martinello
Convidado 2

Vitória - ES, Brasil
7 de Junho de 2013

*Este trabalho é dedicado aos meus pais, que sempre estiveram comigo,
por toda a sua dedicação.*

Lista de ilustrações

Figura 1	Sobreposição de informações no mapa do site PortoAlegre.cc	9
Figura 2	MapATL não possui sobreposição de informação devido ao filtro por zoom	10
Figura 3	Entre ZOOM A e ZOOM C existem muitos níveis intermediários e não apenas 1 (ZOOM B).	10
Figura 4	Informações necessárias em níveis superiores, de zoom, visualizadas em níveis inferiores	11
Figura 5	O centro do mundo na latitude 0 e longitude 0 reside em algum lugar a oeste da costa da África	14
Figura 6	A cada incremento no zoom o mapa dobra a resolução das imagens . .	15
Figura 7	Busca como estratégia para redução do número de marcadores	16
Figura 8	Filtro como estratégia para redução do número de marcadores	17
Figura 9	Otimização Visual como estratégia para redução do número de marcadores	18
Figura 10	Estes dois marcadores não serão agrupados pois residem em quadrados diferentes da grade	20
Figura 11	Usando tamanho e cor do marcador para representar a dimensão do grupo	21
Figura 12	Usando gráficos para representar agrupamentos	22
Figura 13	Principais algoritmos de agrupamentos de pontos	23
Figura 14	Exemplo de utilização da API MarkerClusterer	26
Figura 15	Página do github.com que hospeda o código fonte deste projeto.	28
Figura 16	Arquitetura do framework Searchlight.	30
Figura 17	Processamento de dados do framework Searchlight.	31
Figura 18	Estruturas de dados do framework Searchlight.	32
Figura 19	Website com documentação e exemplos do framework Searchlight.	34
Figura 20	Filtro por categoria fornecido pelo framework Searchlight	36
Figura 21	No framework Searchlight os agrupamentos de marcadores eliminam o zoom arbitrário	36
Figura 22	Balões de Resumo do agrupamento clicado	37
Figura 23	Exemplo de Foco em Grupo por categoria segurança	38
Figura 24	Exemplo de fonte de dados utilizada para gerar mapas automaticamente	39

Sumário

1	Introdução	7
1.1	Definição do Problema	8
1.1.1	Sobreposição de Informações	8
1.1.2	Zoom arbitrário	9
1.1.2.1	Informações arbitrárias	11
1.2	Objetivos	11
1.3	Contribuições	12
1.4	Estrutura da Monografia	12
2	Fundamentação Teórica	13
2.1	Fundamentos de Mapas Web	13
2.1.1	Coordenadas Geográficas	13
2.1.2	Zoom	14
2.2	Estratégias para lidar com muitos marcadores	15
2.2.1	Reduzindo o número de marcadores	16
2.2.1.1	Busca	16
2.2.1.2	Filtro	17
2.2.1.3	Otimização Visual	18
2.2.2	Agrupamento/Clustering	19
2.2.2.1	Agrupamento Por Grade	19
2.2.2.2	Agrupamento Por Distância	19
2.2.2.3	Agrupamento Por Região	20
2.2.2.4	Estilos de visualização de agrupamento	21
2.3	Algoritmos de agrupamento	22
2.3.1	Métodos baseados em grade	22
2.3.1.1	WaveCluster	23
3	Desenvolvimento	25
3.1	Considerações Iniciais	25
3.1.1	Agrupamento de Pontos	26
3.2	Ferramentas Utilizadas	27
3.2.1	Github	27
3.2.2	Leaflet.js	28
3.2.3	TableTop.js	29
3.2.4	Rapydscript	29
3.3	Arquitetura do Sistema	29

3.4	Processamento de dados	31
4	Solução Desenvolvida	34
4.1	Website do projeto	34
4.2	Recursos desenvolvidos	35
4.2.1	Filtro por categorias	35
4.2.2	Agrupamento de marcadores	35
4.2.3	Balões de Resumo e Foco em grupo	37
4.2.4	Geração e Compartilhamento automático de Mapas	38
5	Considerações Finais	40
5.1	Conclusões	40
5.2	Dificuldades Encontradas	40
5.3	Limitações e Perspectivas	41
	Referências	43

1 Introdução

O estado da comunicação mundial atualmente, permite as pessoas, de qualquer país, comunicar e trocar conhecimento dos mais diversos assuntos. Essa facilidade de comunicação também facilita a união de indivíduos, que não se conhecem e com realidades sociais completamente opostas, em um projeto com metas em comum.

Um exemplo, dessa união improvável, é o caso de um carpinteiro sul-africano que, ao perder os dedos numa serra, e ficar insatisfeito com os preços e qualidade das próteses disponíveis no mercado, resolveu iniciar um projeto open source de mão biônica com a ajuda de um técnico de efeitos especiais que mora nos EUA. Juntos, esses dois indivíduos que até então não se conheciam, conseguiram criar um projeto de mão biônica por 150 dólares. Após isso, o projeto ganhou visibilidade e patrocínio e já está sendo testado por algumas crianças deficientes da África do Sul¹.

Essa melhoria de comunicação, obviamente, também afeta o campo empresarial. Em algumas empresas, o sistema de "Outsourcing", que é um sistema de aquisição de conhecimento ou tecnologia, começou a ser substituído pelo sistema de "Crowdsourcing", que no campo empresarial, funciona como uma espécie de concorrência: a empresa lança uma espécie de edital informando quanto pode pagar por determinada solução e quem quer que seja poderá oferecer a resposta adequada ao que se procura, independente de ser uma, duas, ou cem pessoas, contanto que se resolva o problema.

De modo geral Crowdsourcing é a prática de obtenção de serviços, idéias ou conteúdo solicitando contribuições de um grande grupo de pessoas e, especialmente, a partir de uma comunidade on-line, ao invés de funcionários ou fornecedores tradicionais². Existem diversos tipos de crowdsourcing, mas neste trabalho iremos nos focar no crowdsourcing conhecido como *Wisdom of the Crowd*. Ele é um tipo de crowdsourcing que coleta grandes quantidades de informação e as agrupa para obter uma visão completa e precisa sobre um determinado tema. Essa visão, dependendo do tema de estudo, pode ser representada por um mapa de crowdsourcing.

A produção de mapas crowdsourcing é geralmente feita de forma automática, usualmente temos algum software e/ou site que coleta as informações e as agrupa por meio de algum algoritmo desenvolvido especificamente para um conjunto mapa-domínio.

A coleta dessas informações pode ocorrer de várias formas, tanto manual como

¹ Notícia sobre projeto de mão biônica <http://meiobit.com/115807/>

² Definição completa de crowdsourcing <http://en.wikipedia.org/wiki/Crowdsourcing>

automática. Por exemplo, no site [PortoAlegre.cc](#) os usuários podem adicionar novas informações através do site, ou seja, ela é feita de forma manual. Alguns sistemas podem coletar informações de forma automática usando programas de computadores, aplicativos de smartphones ([THIAGARAJAN et al., 2010](#)) e até mesmo da Internet³.

Uma vez coletada, essa informação é analisada e exibida em um mapa, mas o mapa em si não é o produto final, e sim algumas informações específicas retiradas dele. Por exemplo, podemos ter mapas que mostrem os congestionamentos no trânsito de uma cidade e um sistema ([THIAGARAJAN et al., 2009](#)) que através desse mapa consegue identificar uma rota mais eficiente com menor consumo de gasolina.

Em alguns casos, mapas de crowdsourcing possuem informações posicionadas em regiões muito próximas entre si, que devido a quantidade elevada, acabam poluindo a visualização e dificultando a compreensão do mapa. Esse problema pode ser resolvido quando os mapas oferecem mecanismos para agrupar e filtrar essas informações. Um mecanismo ideal é o zoom contextual ou zoom em grupo, que filtra informações irrelevantes, em determinados níveis de zoom, deixando o mapa mais leve e compreensível.

O Projeto Searchlight pretende ser uma ferramenta para auxiliar e melhorar a visualização de mapas de crowdsourcing.

O escopo do projeto atinge a criação de uma ferramenta que visualize mapas de crowdsourcing em um navegador de Internet, tanto desktop quanto mobile, usando recursos de visualização de mapas já disponíveis em HTML5 mas que ainda não possuem zoom contextual e outras opções úteis que permitam uma melhor visualização do mapa.

1.1 Definição do Problema

Mapas de crowdsourcing tendem a mostrar uma enorme quantidade de informação. Essa característica faz com que, em alguns casos, a visualização e a compreensão do mapa seja comprometida.

Ao trabalhar visualização de informações em mapas de crowdsourcing, geralmente encontramos 2 problemas. O primeiro é a sobreposição de informações. O segundo é o excesso de aplicação de zooms para se obter um zoom específico. Este excesso ocorre devido ao uso arbitrário de uma escala rígida de zoom. Portanto, este segundo problema será chamado de Zoom Arbitrário.

1.1.1 Sobreposição de Informações

O site [PortoAlegre.cc](#) é um exemplo da importância do mapas de crowdsourcing no contexto governamental e na sociedade. Por meio desse site os cidadãos de Porto Alegre

³ Mapa com informações coletadas do Twitter <http://trendsmap.com>

podem relatar os problemas de sua cidade para que as autoridades tomem as devidas providências.

Um dos principais objetivos do portal é identificar as áreas prioritárias em que o governo deveria atuar. Mas a sobreposição de informações dificulta essa tarefa, como pode ser observado na [Figura 1](#).

Figura 1 – Sobreposição de informações no mapa do site PortoAlegre.cc



Fonte: [PortoAlegre.cc](#) (2013)

Esse problema fica evidente na [Figura 1](#) quando consideramos, a possibilidade, que um grupo de 5 marcadores reunidos numa região específica, podem sobrepor dezenas ou até milhares de outros marcadores. Ou seja, o mapa não consegue mostrar, com clareza e precisão, as áreas de maior ocorrência de determinado incidente.

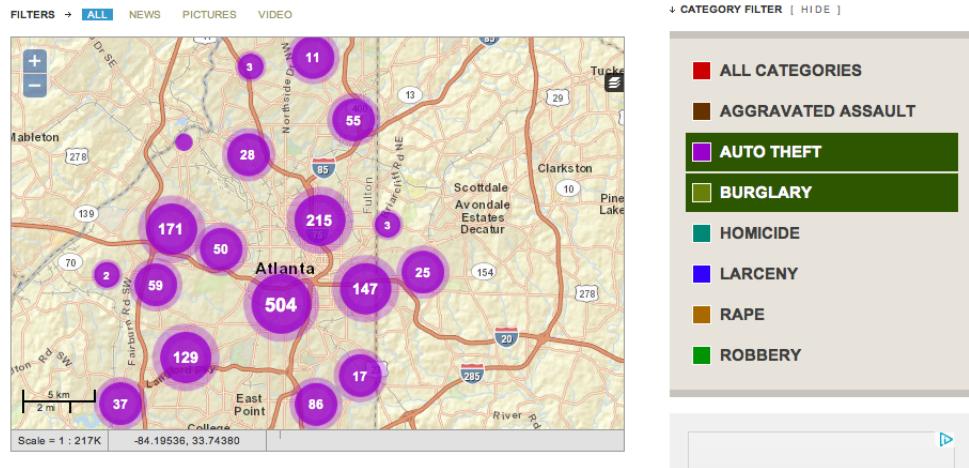
O portal fornece um filtro por categorias, que diminui de forma significativa a quantidade de informação exibida. Entretanto, isto infelizmente não resolve o problema, pois a sobreposição ainda pode ocorrer com informações de uma mesma categoria.

1.1.2 Zoom arbitrário

Alguns sites criam mecanismos que minimizam o problema da sobreposição de informações. Como exemplo, temos o [MapATL](#) (2013) mostrado na [Figura 2](#) que mostra um mapa com a taxa de crimes na cidade norte-americana de Atlanta.

O site fornece filtros por categoria, data e zoom. Mas o principal responsável pela eliminação da sobreposição de informação é o filtro por zoom. Esse filtro agrupa todos os marcadores que estão sobrepostos, no zoom atual, em um único marcador que exibe a informação somada dos marcadores que o compõem.

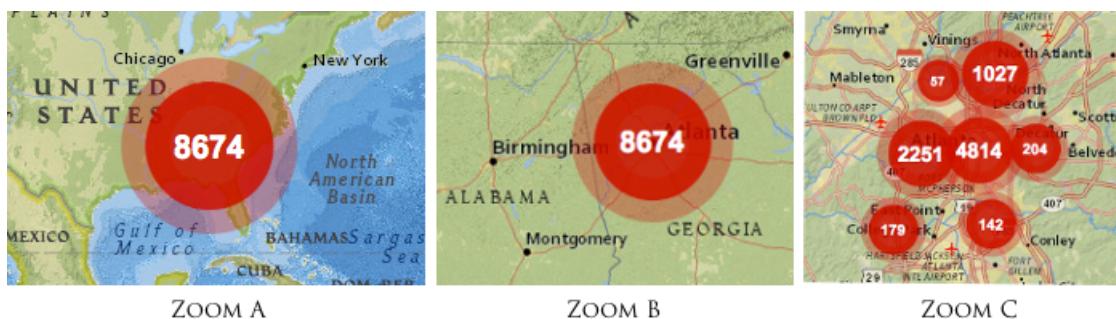
Figura 2 – MapATL não possui sobreposição de informação devido ao filtro por zoom



Fonte: MapATL (2013)

Segundo (SILVA, 2010, 42,44) esse tipo de agrupamento, baseado em grades, pode ser implementado a partir do algoritmo WaveCluster(SHEIKHOLESLAMI; CHATTERJEE; ZHANG, 2000).

Figura 3 – Entre ZOOM A e ZOOM C existem muitos níveis intermediários e não apenas 1 (ZOOM B).



Fonte: MapATL (2013)

Esse algoritmo resolve o problema de sobreposição de informações. Porém o problema do zoom arbitrário, ilustrado na Figura 3, permanece.

O usuário precisa aplicar vários zooms para ir do ZOOM A para o ZOOM C. Isso implica em gasto de tempo e de interações desnecessárias com o servidor de aplicações para exibir os zooms intermediários, quando o ideal seria exibir apenas o ZOOM B após a primeira interação, uma vez que não haverá uma modificação importante na informação obtida pela visualização. Em outras palavras, não há nenhum ganho em termos de visualização pois os dados permanecerão agrupados formando um círculo com o número de ocorrências (8674) em todas as interações antes do ZOOM C.

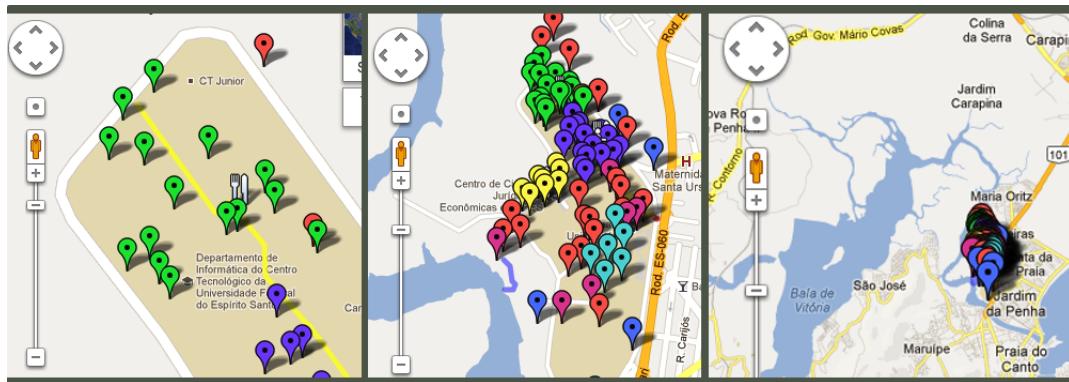
Esse excesso de interações desnecessárias entre o cliente (visualização) e o servidor (processador das interações e dados), prejudica a usabilidade de mapas em dispositivos móveis pois, geralmente, eles possuem pouca banda de Internet.

Uma abordagem para esse problema é o uso de um zoom seletivo que siga uma hierarquia espacial invés de simplesmente dobrar a resolução espacial.

1.1.2.1 Informações arbitrárias

Um outro problema relacionado ao zoom arbitrário é a exibição de informações desnecessárias em qualquer nível de zoom. Na [Figura 4](#) podemos observar o problema, no qual a arbitrariedade está na exibição de informações, e não no número de zooms. Os

Figura 4 – Informações necessárias em níveis superiores, de zoom, visualizadas em níveis inferiores



Fonte: <http://maps.google.com>

marcadores desse mapa exibem informações sobre os departamentos internos da UFES, mas essas informações são úteis apenas em certos níveis de zoom. Em níveis mais baixos, onde a área dessa universidade seja desprezível, a informação perde seu valor e fica poluindo visualmente o mapa. Neste caso, um único marcador representando o grupo de informações seria mais útil.

1.2 Objetivos

Com base nas informações da definição do problema, o objetivo deste trabalho é desenvolver um framework que dê suporte à implementação das seguintes funcionalidades: agrupar marcadores de forma inteligente eliminando a sobreposição; fornecer um zoom seletivo que não exiba níveis intermediários desnecessários; esconder marcadores quando sua informação não seja mais necessária ao nível observado.

O objetivo principal desse trabalho é facilitar a visualização de informação em mapas crowdsourcing. Isso devido a importância sócio-econômica das informações que

geralmente são exibidas nessa classe de mapas. A ferramenta também poderia implementar recursos que facilitassem a divulgação dessas informações. Por isso, o objetivo secundário é fornecer um mecanismo para gerar mapas a partir de planilhas de dados, e compartilhá-los sem que o autor precise programar ou ter algum conhecimento de programação. Para este trabalho a divulgação dos mapas é um objetivo secundário, mas poderia ser melhor abordado em trabalhos futuros.

1.3 Contribuições

A contribuição deste trabalho é o desenvolvimento de um framework para exibição de mapas em páginas Web facilitando a visualização de informações crowdsourcing.

Além disso também foi criado um website do projeto ([QUIRINO, 2013](#)) que explica e documenta o framework. O website fornece também um pagina de geração e compartilhamento de mapas por pessoas que não sabem programar.

1.4 Estrutura da Monografia

Este trabalho está organizado da seguinte forma, após esta breve introdução:

No capítulo 2, Fundamentação Teórica, apresenta-se uma explicação básica sobre os elementos comuns em mapas geográficos, usados na web; uma pesquisa sobre as estratégias para lidar com mapas que possuem muitos marcadores e uma pequena explicação sobre os principais algoritmos para agrupamento de pontos.

No capítulo 3, Desenvolvimento, é explicado como o projeto foi desenvolvido, quais foram as ferramentas utilizadas, e o motivo da escolha de determinadas tecnologias.

No capítulo 4, Solução Desenvolvida, é apresentado em detalhes a solução que foi desenvolvida e como utilizá-la.

No capítulo 5, Conclusão, são apresentadas as dificuldades encontradas no decorrer do projeto, trabalhos futuros e conclusão geral sobre o projeto.

2 Fundamentação Teórica

Neste capítulo são apresentados os principais conceitos teóricos necessários para o desenvolvimento deste trabalho, que são: os fundamentos de mapas Web, as estratégias para lidar com muitos marcadores e os algoritmos para agrupamentos de pontos.

2.1 Fundamentos de Mapas Web

Entender como funciona o processo de exibição de um mapa Web pode parecer um pouco complexo quando não se conhece o funcionamento do sistema de coordenadas geográficas do mapa. Essa complexidade aumenta quando o processo de zoom também é considerado. Porém, ao compreender o funcionamento desses elementos percebe-se que a exibição de mapas dentro de um navegador Web é um processo relativamente simples.

2.1.1 Coordenadas Geográficas

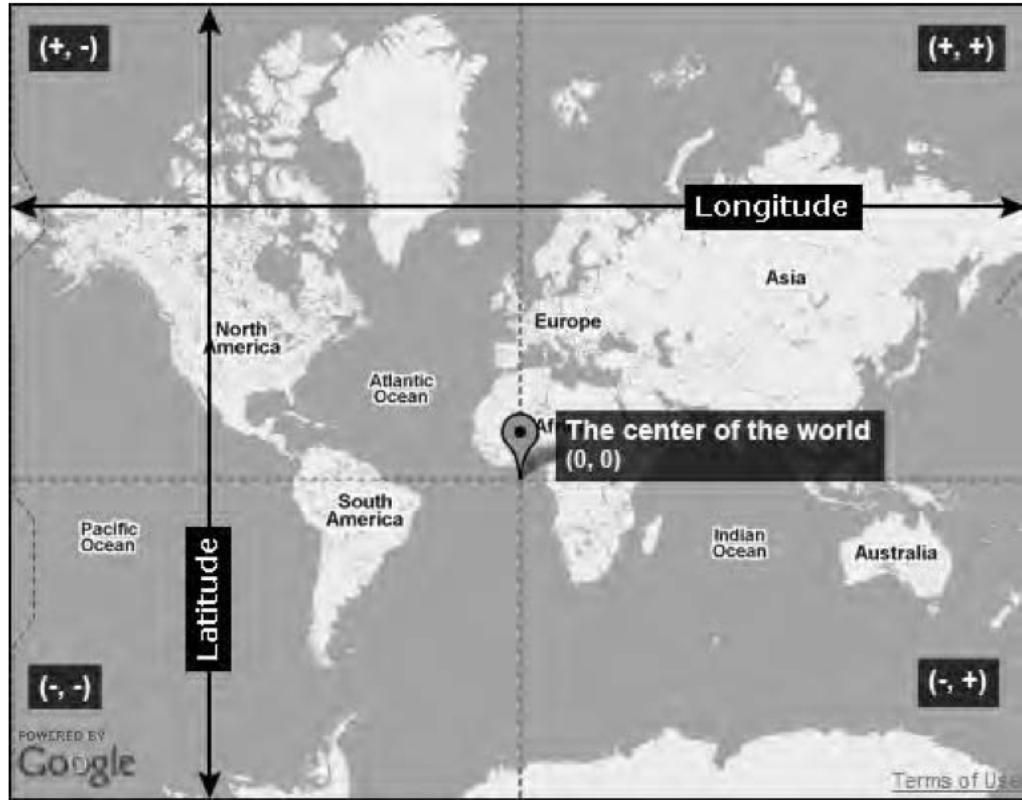
Coordenadas geográficas são usadas para expressar localizações no mundo. Existem vários sistemas de coordenadas diferentes. O sistema de coordenadas usado pelo Google Maps é o Word Geodetic System 84 (WGS84), que é o mesmo sistema que o Global Position System(GPS) usa.

As coordenadas são expressas usando o conceito de latitude e longitude. No Equador a latitude é 0 e, consequentemente, tudo abaixo do Equador (hemisfério sul) possui uma latitude negativa, e tudo acima possui uma latitude positiva. Similarmente também existe uma linha zero para longitude. Ela é conhecida como meridiano, e por razões históricas passa por Greenwich, Inglaterra. Cada posição que é localizada a leste desta linha tem um número positivo e tudo a oeste tem um número negativo ([SVENNERBERG, 2010](#), p. 4).

A [Figura 5](#) ilustra esses conceitos e mostra que é possível representar o mapa do mundo em uma imagem retangular projetada sobre o plano cartesiano.

Caso seja necessário um maior grau de detalhamento, do mapa, basta que se aumente a resolução da imagem mantendo as proporções e relações entre as coordenadas (x,y), do plano cartesiano, com as coordenadas (longitude, latitude) que são usadas pelo mapa.

Figura 5 – O centro do mundo na latitude 0 e longitude 0 reside em algum lugar a oeste da costa da África



Fonte: ([SVENNERBERG, 2010](#), p. 5)

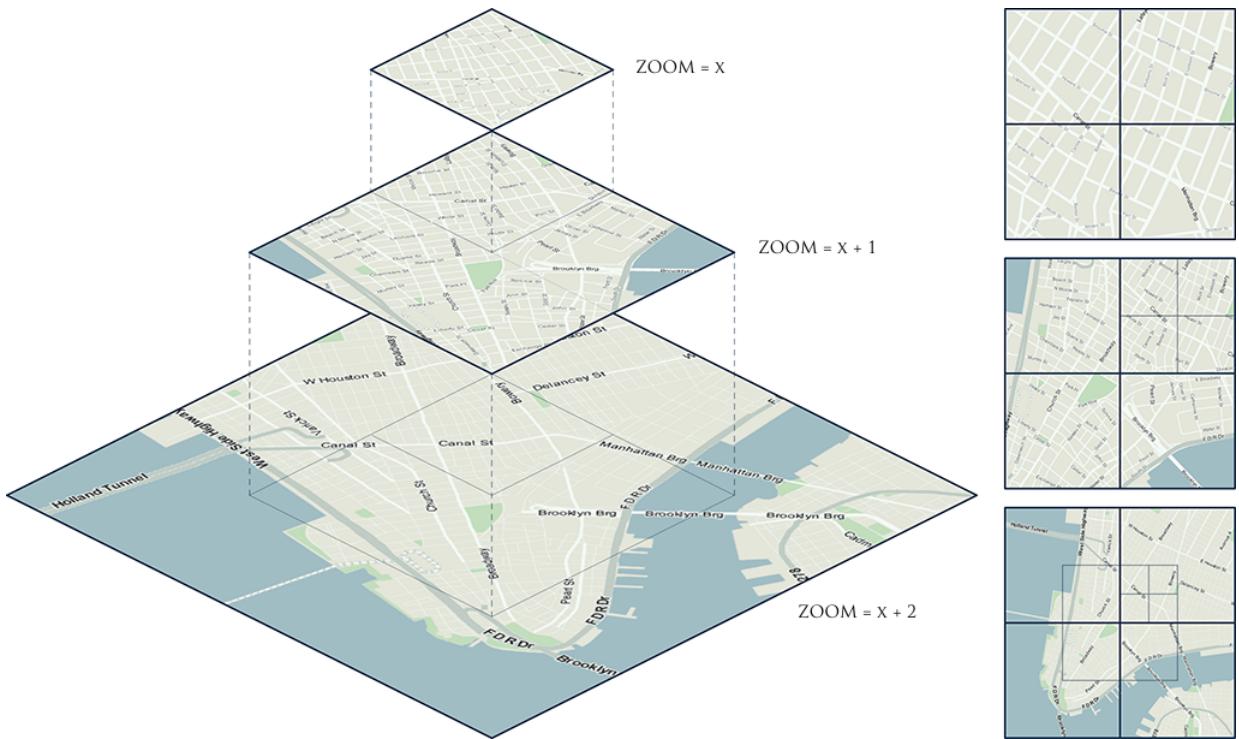
2.1.2 Zoom

As coordenadas de longitude e latitude servem para localizar um ponto numa imagem retangular, através de suas correspondentes x e y do plano cartesiano. Porém mapas também fornecem uma terceira coordenada conhecida como coordenada de zoom.

Esta coordenada controla qual o tamanho da imagem que será usada para projetar o mapa sobre o plano cartesiano. Usualmente, um mapa com coordenada zoom, ou nível de zoom, igual a zero possui uma imagem de tamanho 256x256 pixels. A cada incremento, da coordenada de zoom, dobra-se o tamanho da imagem usada para representar o mapa, e consequentemente o nível de detalhes. De forma que para zoom=1 a imagem terá 512x512 pixels, para zoom=2 terá 1024x1024 e assim por diante, mantendo-se a mesma quantidade de informações por unidade de área (pixel). Em outras palavras, mais informações por unidade de área significa menos detalhes sendo exibidos na imagem que representa o mapa. A [Figura 6](#) exemplifica esse processo.

Portanto para localizar uma informação georreferenciada em um mapa Web, de forma detalhada, precisamos de três coordenadas: latitude, longitude e zoom. Além disso, como o tamanho da imagem da projeção varia de acordo com o nível de zoom, o mapa

Figura 6 – A cada incremento no zoom o mapa dobra a resolução das imagens



Fonte: <http://workshops.opengeo.org/suiteintro/geowebcache/basics.html>

precisa ajustar os valores das coordenadas latitude e longitude para que mantenham suas proporções para os diversos níveis de zoom.

2.2 Estratégias para lidar com muitos marcadores

Um problema comum ao se trabalhar com mapas de crowdsourcing é a enorme quantidade de marcadores utilizados para representar os dados do mapa. Isto geralmente afeta a performance da aplicação para visualização do mapa (e das suas informações), pois quanto mais marcadores são inseridos mais lenta fica a exibição do mapa.

É difícil calcular exatamente o número máximo de marcadores que um mapa suporta antes de começar a ficar lento, pois a velocidade de exibição do mapa depende tanto do navegador quanto da plataforma em que é exibido. Por exemplo, um mapa pode ser exibido bem rápido em um navegador como o Google Chrome e ao mesmo tempo ser lento quando exibido no Internet Explorer ([SVENNERBERG, 2010](#), p. 177).

Portanto, é necessário um estudo sobre as estratégias para se lidar com o problema de exibição de muitos marcadores em um mapa. Em ([SVENNERBERG, 2010](#), capítulo 9) o autor comenta sobre duas estratégias básicas para esse problema. A primeira, e mais óbvia, é reduzir o número de marcadores. A segunda estratégia consiste em agrupar os

marcadores por algum grau de similaridade.

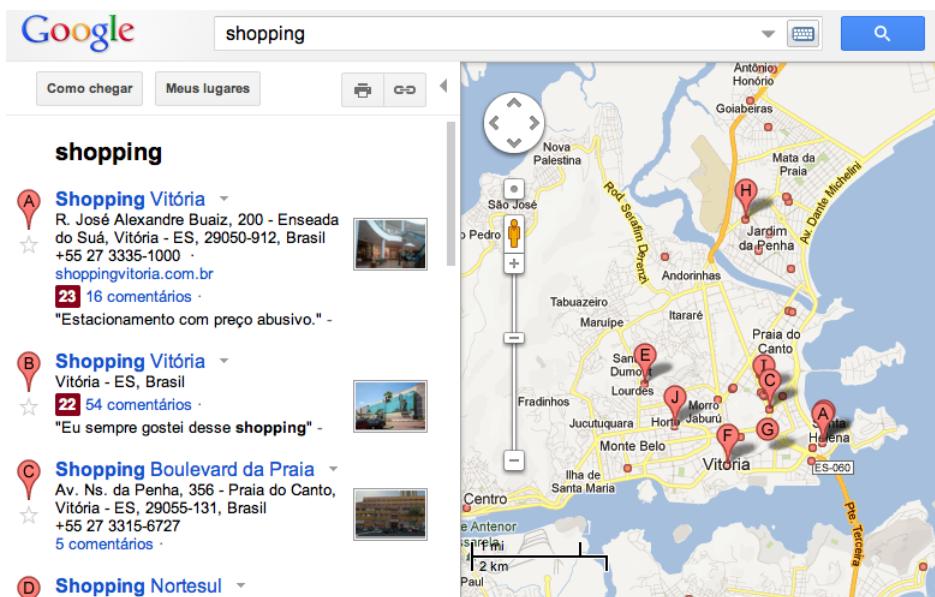
2.2.1 Reduzindo o número de marcadores

Existem muitos meios de se reduzir o número de marcadores, entre eles destacam-se as reduções via busca, filtro e otimização visual.

2.2.1.1 Busca

Para se reduzir o número de marcadores exibidos no mapa pode-se fornecer um mecanismo de pesquisa ou busca no mapa. Desta forma, mesmo que o mapa possua milhões de marcadores, somente aqueles que satisfazem os critérios da busca são exibidos.

Figura 7 – Busca como estratégia para redução do número de marcadores



Fonte: <http://maps.google.com>

A Figura 7 exemplifica isso ao mostrar o exemplo de busca no Google Maps. O Google Maps possui um catálogo enorme de informações, sobre diversas regiões geográficas. Mostrar todas essas informações ao mesmo tempo deixaria o mapa muito poluído, e praticamente inutilizável. Para que isso não aconteça é implementado o mecanismo de filtragem a partir da satisfação de critérios de busca.

Na Figura 7 observa-se a exibição de uma busca pela palavra chave shopping. Nesta mesma região de interesse existem estabelecimentos de outras categorias como supermercados, lojas, padarias etc. Mas o Google Maps exibe apenas os marcadores relativos a categoria shopping, que é a palavra chave da busca. Ao fazer isso o mapa fica mais limpo e compreensivo.

2.2.1.2 Filtro

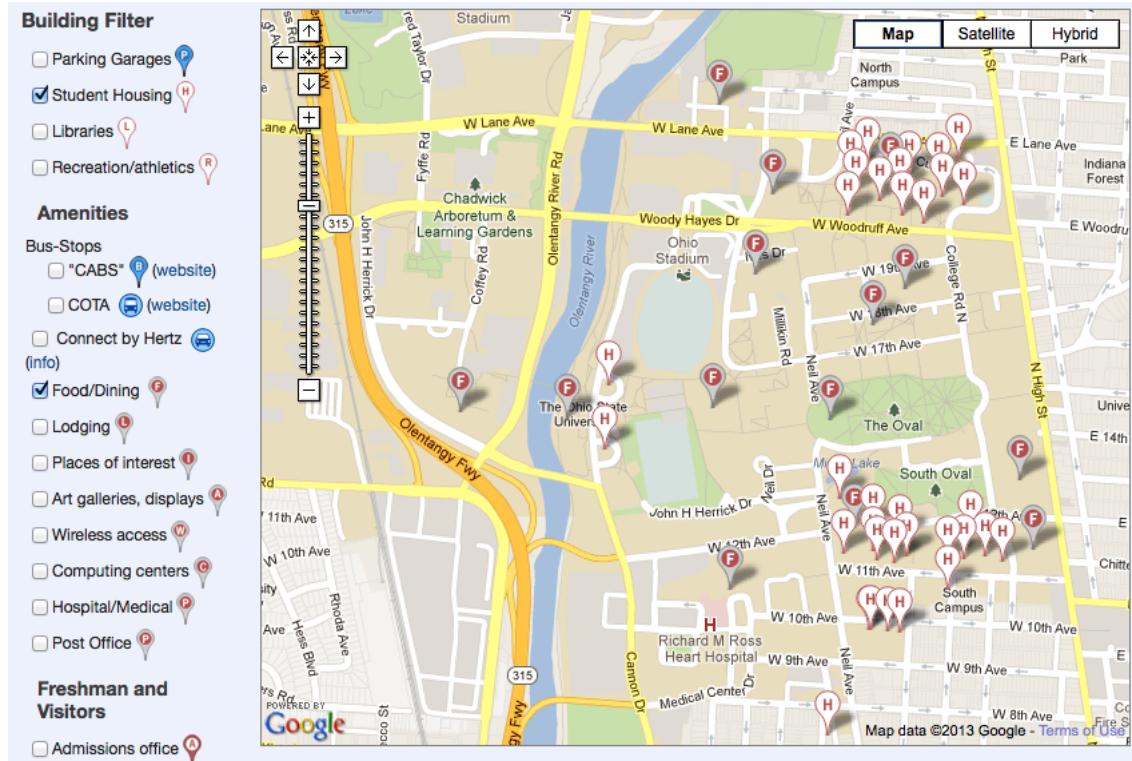
De forma similar ao mecanismo de busca, um mapa também pode possuir um mecanismo para filtrar grupos de marcadores de acordo com algum critério de seleção do grupo. Desta forma, somente os marcadores que pertençam aos grupos marcados no filtro serão exibidos no mapa.

Diferente do método de busca, este método permite ser mais específico e direto, pois cada campo do filtro é relacionado diretamente com alguma categoria contida nas informações. No método de busca esta relação é menos direta e depende do algoritmo de busca utilizado.

Como exemplo, ao utilizar-se do método de filtro para marcar duas categorias, obrigatoriamente, deve ser retornado o resultado para as duas categorias que foram marcadas. Já com o método de busca isto nem sempre é verdade, pois dependendo do algoritmo de busca ele pode entender que a busca por “categoria 1 categoria 2” é uma busca por duas categorias ou uma busca por uma categoria conhecida como “categoria 1 categoria 2”, que obviamente não seria encontrada.

A [Figura 8](#) mostra um mapa com filtro à esquerda. Observa-se que apenas os marcadores dos grupos “Student Housing” e “Food/Dining” são exibidos no mapa.

Figura 8 – Filtro como estratégia para redução do número de marcadores



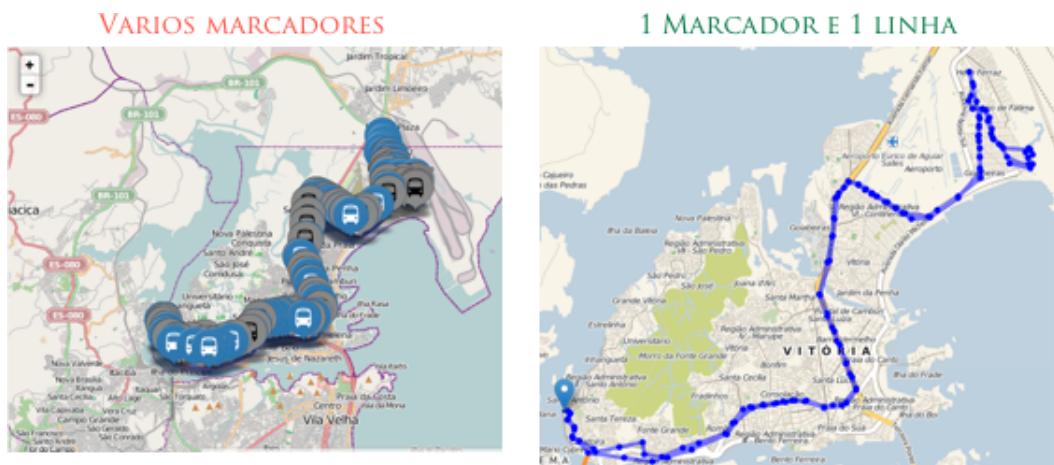
Fonte: <http://www.osu.edu/map/google.php>

2.2.1.3 Otimização Visual

Nem sempre é preciso usar marcadores para representar dados de um mapa. Em alguns mapas faz mais sentido usar polígonos ou grupo de polígonos para representar um dado ao invés de simplesmente um ponto para o marcador. Por exemplo, ao exibir uma rota não é necessário um marcador para cada vértice da rota, pois precisa-se apenas do desenho de uma linha, passando por cada vértice, para que a rota seja representada.

A [Figura 9](#) mostra um exemplo desse tipo de otimização. No mapa à esquerda, ela exibe uma rota com marcadores em cada vértice do trajeto percorrido. No mapa à direita, ela exibe a mesma rota só que sem colocar marcadores nos vértices do trajeto, deixando apenas um marcador para exibir o ponto de início do trajeto.

Figura 9 – Otimização Visual como estratégia para redução do número de marcadores



Fontes: <http://wancharle.github.io/Searchlight/> e <https://dl.dropboxusercontent.com/u/877911/UFES/grafos/grafos.html>

Nesse exemplo, cada vértice da rota representa um ponto de parada para os ônibus que percorrem esse trajeto. Inicialmente, pode se achar necessário a exibição de marcadores nesses vértices como mecanismos de interação com os dados que eles representam. Por exemplo, os usuários do transporte público poderiam querer saber quais são os horários que determinados ônibus passam naquele ponto específico, e com um simples clique no marcador o mapa poderia exibir essa informação.

Porém, para que este tipo de interação seja possível o mapa teria que exibir os marcadores em cada vértice, que como mostrado na [Figura 9](#) não é o melhor caminho. Felizmente, existe algumas alternativas para esse processo que não exigem a presença de um marcador. Por exemplo, ao clicar na linha da rota o mapa pode fazer uma busca, baseada em proximidade, e localizar o ponto que o usuário deseja saber mais informações. Após isso o mapa pode até exibir um marcador temporário como reforço visual do local que está sendo exibida a informação.

2.2.2 Agrupamento/Clustering

A segunda estratégia citada por (SVENNERBERG, 2010, capítulo 9) para reduzir o número de informações exibidas por um mapa, é conhecida como clustering, ou em português, agrupamento.

A estratégia de agrupamento de marcadores consiste em agrupar marcadores que estejam próximos, por algum critério de proximidade, e exibir apenas um marcador para cada grupo e um marcador para cada marcador individual, que não pertença a nenhum grupo. Ou seja, em vez de exibir um marcador individual para cada marcador são exibidos marcadores para grupos e marcadores individuais.

Nessa estratégia os grupos variam de acordo com o zoom. Ou seja, ao fazer zoom em grupo, o grupo se divide em subgrupos menores e, se for o caso, em marcadores individuais.

O processo contrário também ocorre ao diminuirmos o zoom de um grupo. Neste caso o grupo pode se agrupar com grupos vizinhos e formar um novo grupo como mais elementos, de forma, que em alguns mapas, no zoom=0 o mapa exibe apenas um marcador representando o grupo com todos os marcadores do mapa.

Existem diversos métodos para agrupamento de marcadores, mas os mais comuns são implementados usando algum dos seguintes parâmetros para agrupamento: agrupamento por grade, por distância, ou por região.

2.2.2.1 Agrupamento Por Grade

Agrupamento baseado em grade é provavelmente a abordagem mais comum para agrupar marcadores. Esta abordagem, divide o mapa em uma grade e agrupa todos os marcadores de cada quadrado em um grupo.

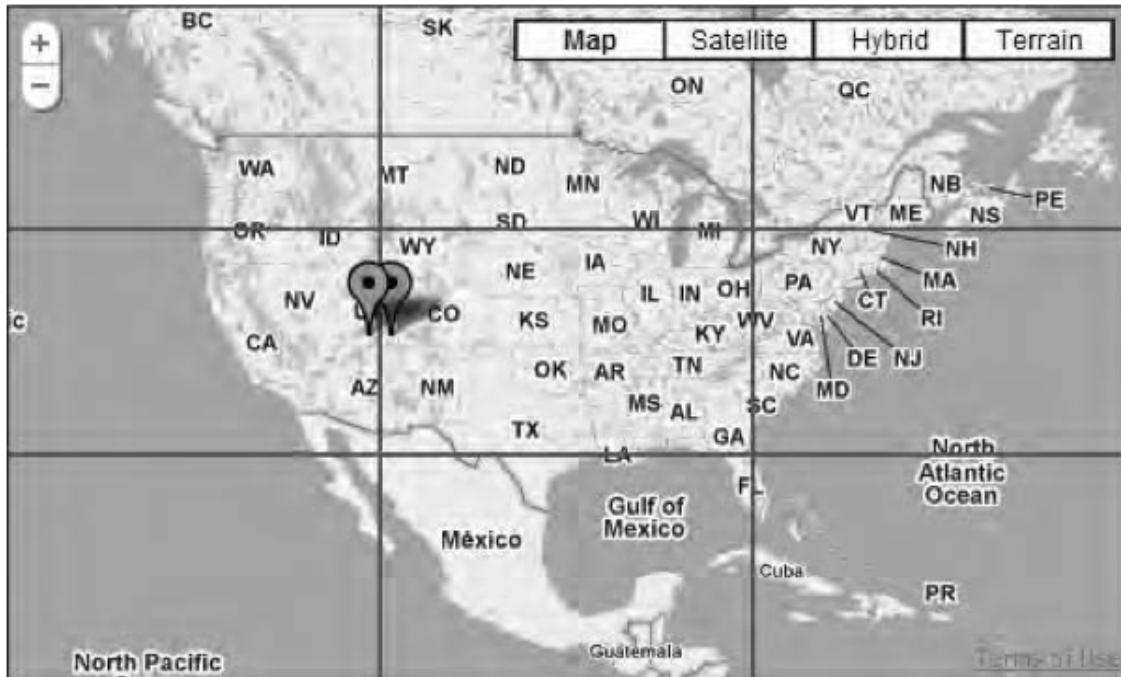
Apesar de ser uma técnica eficiente, ela possui limitações óbvias, pois pode levar a resultados indesejados. Por exemplo, considere dois marcadores próximos mas localizados em quadrados diferentes da grade. Neste caso, eles não serão agrupados no mesmo grupo. Para exemplificar isso, veja a [Figura 10](#).

2.2.2.2 Agrupamento Por Distância

Nesta técnica, não ocorre o problema comentado no agrupamento por grade, pois nela é observado cada marcador e para cada um é procurado marcadores vizinhos, se os marcadores forem próximos o suficiente eles são colocados no mesmo grupo.

Alguns podem achar esta técnica problemática para certos mapas. Pois por agrupar marcadores próximos entre si, os grupos não possuem uma localização fixa como acontece no agrupamento por grade. Assim no agrupamento por distância os grupos

Figura 10 – Estes dois marcadores não serão agrupados pois residem em quadrados diferentes da grade



Fonte: ([SVENNERBERG, 2010](#), figura 9.6)

podem aparecer em posições aleatórias que podem não fazer sentido para o usuário que observa o mapa.

2.2.2.3 Agrupamento Por Região

No agrupamento por região, definem-se diferentes regiões geográficas como países, estados, cidades. E todos os marcadores, de cada região, podem ser agrupados em um grupo para representar a região. Além disso, também é possível definir em que nível de zoom os grupos serão quebrados em subgrupos.

A vantagem é que fica fácil criar grupos que fazem mais sentido para o usuário. Pois, um agrupamento que siga a ordem País > Estado > Cidade é mais natural para o usuário do que um que considere apenas a proximidade.

A desvantagem dessa técnica é o esforço para implementá-la ([SVENNERBERG, 2010](#), p. 182) já que a definição dos grupos não pode ser facilmente automatizada. Essa dificuldade ocorre devido à natureza de como é organizada a hierarquia das regiões, que varia de país para país, sendo necessário em alguns casos o uso de tabelas para conversão¹.

¹Até a definição do que é uma autoestrada precisa de uma tabela de conversão entre países: http://wiki.openstreetmap.org/wiki/Highway:International_equivalence

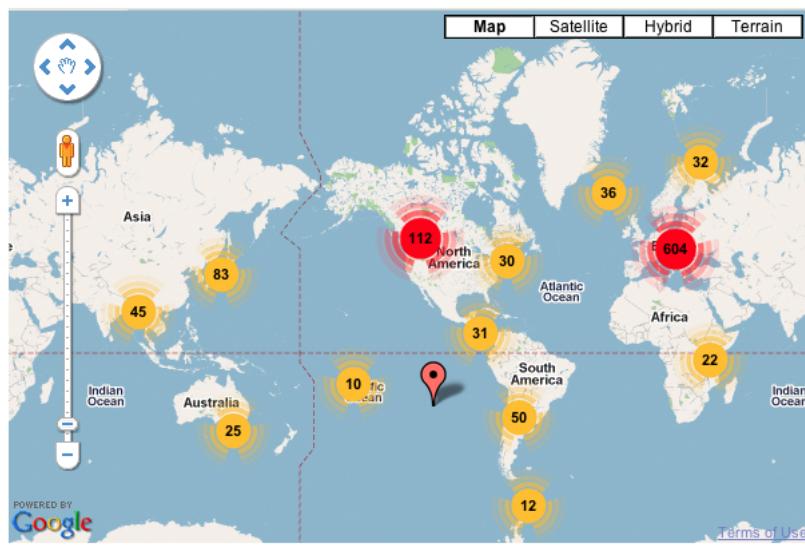
2.2.2.4 Estilos de visualização de agrupamento

A representação mais simples de um agrupamento pode ser feita por meio de um marcador. Porém, o marcador por si só não expressa muita informação. Não é possível saber a dimensão do grupo ou a natureza de seus elementos observando apenas um marcador.

Uma alternativa é usar marcadores com tamanhos diferentes de acordo com o tamanho do grupo. Mas, neste caso, tem-se o problema de ter que classificar um tamanho padrão para determinados marcadores. Pode-se, também, usar cores para representar a dimensão do grupo, onde grupos menores tenderiam para o verde e grupos maiores para o vermelho.

Uma solução bastante comum é uma mistura das soluções anteriores. Onde usa-se um marcador com tamanho diferente de acordo com o número de elementos e cores para maximizar a percepção. Além disso, acrescenta-se uma legenda exibindo o número total de elementos do grupo. A Figura 11 ilustra um exemplo de uso desta técnica de agrupamento.

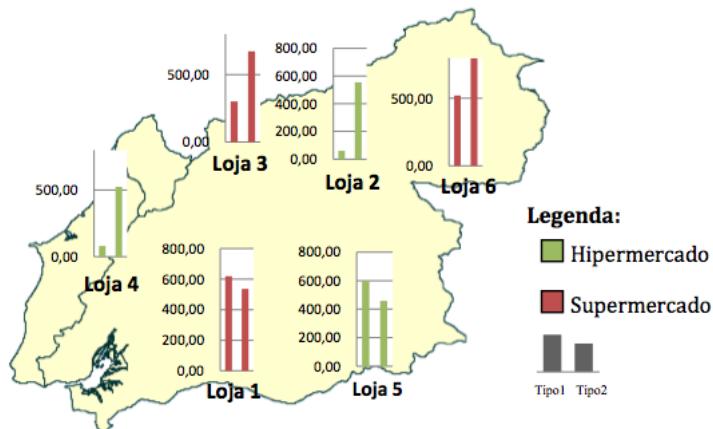
Figura 11 – Usando tamanho e cor do marcador para representar a dimensão do grupo



Fonte: <https://developers.google.com/maps/articles/toomanymarkers>

Alguns mapas, possuem dados que permitem uma visualização mais complexa. Para esses tipos de mapa é possível substituir o marcador por uma imagem, gráfico ou forma que represente o grupo. Por exemplo, uma mapa que mostre as compras de todos os clientes de uma rede de supermercados poderia agrupar os dados inicialmente por loja e na visualização exibir um gráfico de vendas para representar cada grupo. Esse exemplo, pode ser observado na Figura 12.

Figura 12 – Usando gráficos para representar agrupamentos



Fonte: ([SILVA, 2010](#), figura 65)

Além disso, os gerentes dessas lojas, poderiam efetuar um zoom no gráfico da loja que gerenciam e ver os subgrupos divididos por bairros, com cada grupo exibindo o gráfico de vendas do bairro, permitindo aos gerentes a criação de uma estratégia de marketing localizada.

2.3 Algoritmos de agrupamento

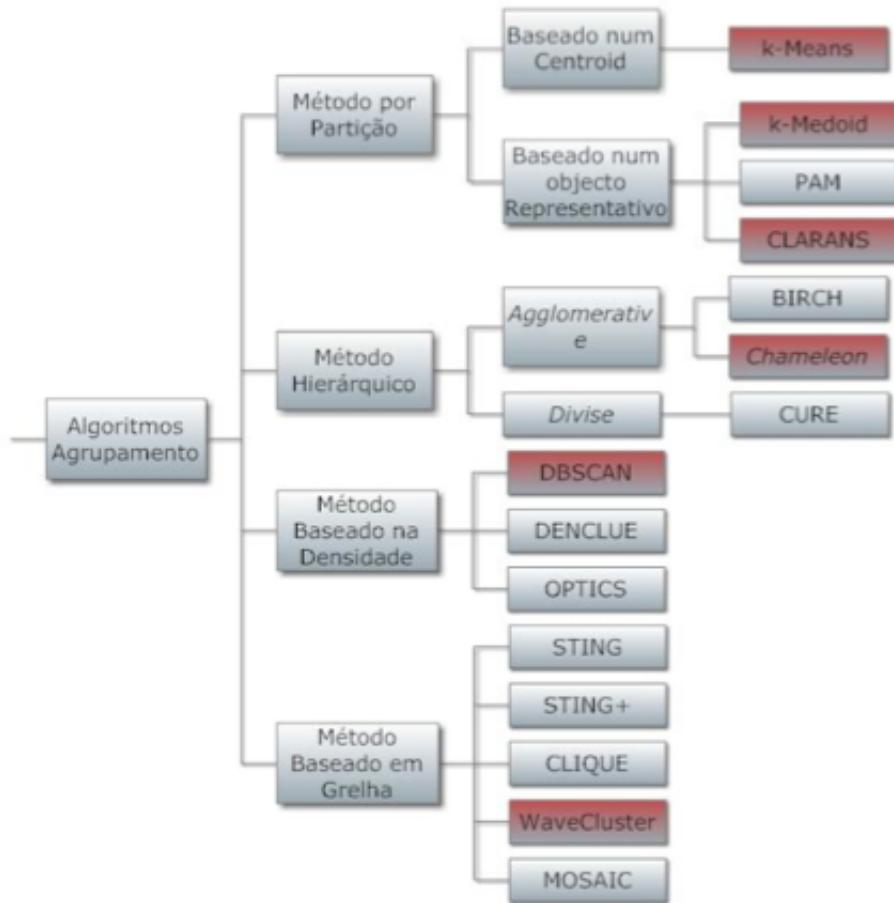
Basicamente os algoritmos de agrupamentos de pontos podem ser classificados em quatro categorias: (i) método por partição (ii) método hierárquico (iii) método baseado em densidade (iv) método baseado em grade. Isto é ilustrado na [Figura 13](#) onde ([SILVA, 2010](#), p. 35) destaca os potenciais melhores algoritmos para este tipo de problema. Com exceção dos algoritmos k-Means e k-Medoid que foram introduzidos apenas por motivos históricos ([SILVA, 2010](#), p. 36).

Observa-se que em ([SILVA, 2010](#), capítulo 2) é feito um estudo mais aprofundado sobre os principais algoritmos de agrupamentos de pontos. Por isso, nesta seção será descrito apenas o algoritmo utilizado na solução implementada neste trabalho, isto é, o algoritmo WaveCluster, que é baseado no método de grade.

2.3.1 Métodos baseados em grade

Como já foi comentado na [Subseção 2.2.2](#), agrupamentos baseados em grade são aqueles que dividem o espaço do mapa em uma grade e agrupa os marcadores de cada quadrado ou célula da grade. Ou seja, algoritmos baseados em grade são aqueles que usam o agrupamento por grade como parâmetro de agrupamento. Pode se observar na [Figura 13](#) que os principais algoritmos são: STING, STING+, CLIQUE, WaveCluster e

Figura 13 – Principais algoritmos de agrupamentos de pontos



Fonte: ([SILVA, 2010](#), figura 15)

MOSAIC. E, apesar de não ser o mais indicado para o contexto SOLAP+ que ([SILVA, 2010](#)) estudava, o melhor entre os algoritmos baseados em grade é o WaveCluster.

2.3.1.1 WaveCluster

Uma boa abordagem para agrupamentos deve ser eficiente e detectar grupos de formas arbitrárias. Ela também tem que ser insensível a dados discrepantes e à ordem de entrada dos dados.

O algoritmo WaveCluster atende todos esses requisitos. Sua definição completa é encontrada em ([SHEIKHOLESLAMI; CHATTERJEE; ZHANG, 2000](#)) mas um resumo é exibido no [algoritmo 1](#).

O WaveCluster possui complexidade $O(n)$. Além disso, por causa do uso das técnicas de processamento de sinais a propriedade de multi-resolução, ou agrupamento nos níveis hierárquicos, também é aplicada ao WaveCluster.

Entrada Vetores de características dos objetos de dados multidimensionais
Saida Objetos agrupados

1. Quantizar o espaço de características, atribuir objetos as células da grade.
2. Aplicar a transformação *wavelet* no espaço de características quantizado.
3. Encontrar os componentes conectados (clusters) nas sub-bandas
4. Transformar o espaço original, em diferentes níveis.
5. Atribuir rótulos às células.
6. Criar a tabela de pesquisa.
7. Mapear objetos aos clusters/grupos.

Algoritmo 1: WaveCluster

O seu uso não é limitado apenas a dados espaciais, o algoritmo WaveCluster pode ser aplicado a qualquer conjunto de atributos com valores numéricos ordenados.

Observe que o algoritmo utiliza-se de uma transformada chamada *wavelet*. A transformada wavelet é uma ferramenta que permite decompor um sinal em diferentes componentes de frequências, permitindo assim, estudar cada componente separadamente em sua escala correspondente².

A implementação da transformada wavelet não faz parte do escopo deste trabalho. Além disso, o framework Searchlight utiliza uma ferramenta que realiza essa transformação a partir de um conjunto de valores de entrada. A Subseção 3.1.1 descreve essa ferramenta e o motivo de sua escolha como componente responsável por realizar o agrupamento de marcadores.

² Informações sobre a transformada wavelet em: <http://www.dainf.ct.utfpr.edu.br/~ionildo/wavelet/cap3.htm>

3 Desenvolvimento

Neste capítulo será detalhado o processo de desenvolvimento do framework Searchlight, e as ferramentas utilizadas para a construção do mesmo.

3.1 Considerações Iniciais

Durante a fase de pesquisa deste projeto, descobriu-se uma variedade de recursos que poderiam ser usados na elaboração de um framework para visualização de mapas. Muitos destes recursos já estavam no escopo do projeto, outros foram adicionados e alguns ficaram fora do escopo e colocados na lista de trabalhos futuros.

Inicialmente a meta do framework era apenas encontrar uma maneira de visualizar mapas de crowdsourcing de forma mais limpa e sucinta através de técnicas de agrupamento aliadas a hierarquia de zoom. Mas no decorrer das pesquisas percebeu-se que seria interessante adicionar mais alguns recursos, como filtro por categoria e foco em grupo.

Também foi adicionado ao projeto a possibilidade de gerar e compartilhar um mapa web sem precisar escrever linhas de código. Para isso é necessário fornecer um endereço web no qual o gerador de mapas possa buscar os dados geográficos do mapa. Desta forma, o usuário não precisa mais saber programar para poder usufruir dos recursos do framework e a sua única preocupação fica depositada sobre o conteúdo do mapa.

Em relação ao conteúdo usado pelo compartilhador de mapas, o framework atualmente suporta o armazenamento numa planilha eletrônica ou em um arquivo de formato JSON¹. Quando o conteúdo é oriundo de planilha eletrônica é necessário que esta esteja armazenada no Google Docs e seja pública. De forma similar, quando o conteúdo é armazenado em um arquivo JSON é necessário que o website que hospeda o arquivo suporte o protocolo JSONP. Caso contrário, o usuário deverá instalar o framework Searchlight no website que deseja visualizar o mapa.

Em relação a API de mapas utilizada pelo projeto, o objetivo inicial era usar a API do Google Maps para a visualização dos mapas. Porém, durante a fase de pesquisa, circulava notícias que o Google estaria mudando seus termos de serviço e poderia tornar pago² o uso da API. Além disso, no mesmo período, a Apple anunciou que iria abandonar o uso do Google Maps em seus smartphones e adotaria uma solução própria. Isto trouxe

¹Formato muito usado para troca de dados entre websites <http://www.json.org>

²Uma notícia sobre o inicio das cobranças do Google Maps <http://goo.gl/f1E3K>

a necessidade de procurar uma alternativa ao Google Maps. A alternativa encontrada foi o uso da biblioteca open-source Leaflet.js ([LEAFLET, 2013](#)) que será descrita no decorrer deste capítulo.

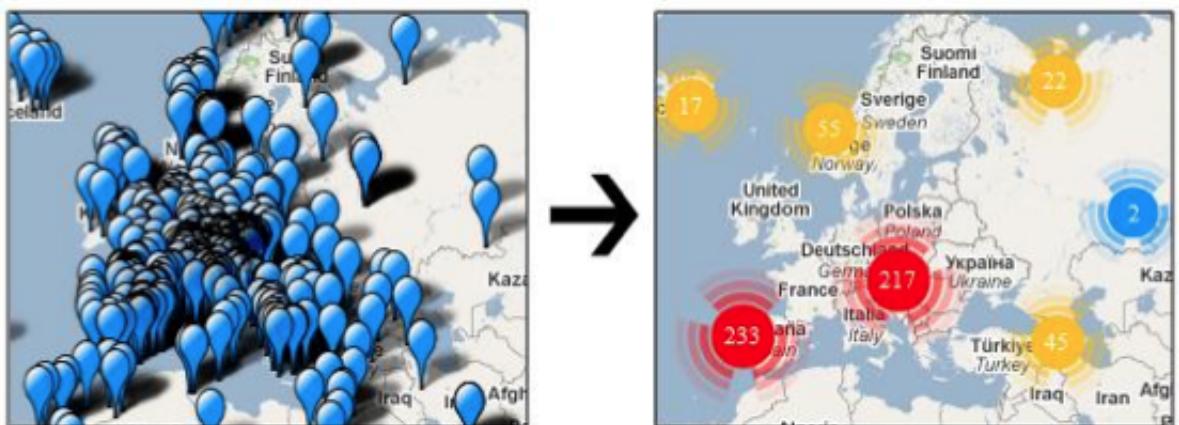
3.1.1 Agrupamento de Pontos

Durante a fase de pesquisa deste trabalho, foi feito uma busca por ferramentas acadêmicas e comerciais que já fizessem o uso de algoritmos de agrupamento de pontos. Foram encontradas duas ferramentas promissoras: MarkerClusterer e Leaflet.MarkerCluster.

MarkerClusterer ([SVENNERBERG, 2010](#), p. 188) é uma ferramenta fornecida pelo google que estende o uso da API do Google Maps para o uso de algoritmos de agrupamentos.

MarkerClusterer é uma solução baseada em grade. Ela agrupa marcadores de acordo com sua distância ao centro de um grupo. Quando um marcador é adicionado, ele pesquisa sua posição em todos os grupos. Caso não seja colocado em nenhum grupo, um novo grupo é criado para este marcador. A [Figura 14](#) demonstra a aplicação da ferramenta nos marcadores de um mapa.

Figura 14 – Exemplo de utilização da API MarkerClusterer



Fonte: ([SILVA, 2010](#), figura 20)

Assim como a biblioteca MarkerClusterer estende a API do Google Maps para uso de agrupamento de marcadores, o plugin Leaflet.MarkerCluster([LEAFLET.MARKERCLUSTER, 2013](#)) serve como uma extensão para a API Leaflet.

Este plugin, faz basicamente as mesmas funções da ferramenta MarkerCluster, porém com algumas melhorias. Por exemplo, ao fazer zoom o mapa exibe uma pequena animação do processo de agrupamento; ao se passar o mouse sobre um marcador do grupo

o mapa exibe um polígono que mostra os limites alcançados pelo grupo; os grupos que não são visíveis na visualização atual do mapa são retirados do mapa para aumentar a performance. Além dessas melhorias, uma que merece destaque é a sua capacidade de customização.

Ao contrário da biblioteca MarkerClusterer, o plugin Leaflet.MarkerCluster possui um documentação bastante útil que pode ser encontrada em ([LEAFLET.MARKERCLUSTER, 2013](#)).

É importante observar, que inicialmente, procurou-se por uma documentação mais abrangente da biblioteca MarkerCluster do Google Maps, mas até o momento da confecção desse trabalho não foi possível encontrar nada além de alguns poucos exemplos de uso. Isto, e outros fatores, contribuíram para a escolha da biblioteca Leaflet.js como API de visualização de mapas do framework Searchlight.

3.2 Ferramentas Utilizadas

Um framework, em desenvolvimento de software, é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica. Um framework pode atingir uma funcionalidade específica, por configuração, durante a programação de uma aplicação. Ao contrário das bibliotecas, é o framework quem dita o fluxo de controle da aplicação, chamado de Inversão de Controle³.

De modo geral, um framework é conjunto de ferramentas que trabalham em conjunto. O que une as ferramentas são as regras do framework que tem como objetivo prover as funcionalidades que motivaram a criação do dele.

O framework Searchlight tem como objetivo facilitar a criação e a visualização de mapas de crowdsourcing por meio de automatização do processo de criação do mapa. O público alvo abrange programadores e usuários sem nenhum conhecimento de programação. Para atingir este objetivo o framework faz uso do seguinte conjunto de ferramentas: GitHub, Leaflet.js, Tabletop.js e RapydScript.

3.2.1 Github

GitHub é um Serviço de Web Hosting Compartilhado para projetos que usam o controle de versionamento Git. Mais do que isso, GitHub incorpora recursos sociais e de pesquisa que na prática o transformam em uma rede social para programadores. GitHub incorpora elementos tanto do facebook como do twitter. Por exemplo é possível favoritar um projeto, seguir um programador, ver quais são as tendências de programação, quais projetos estão se destacando, quais projetos se relacionam entre si e muitos outros recursos.

³Definição completa em: <http://pt.wikipedia.org/wiki/Framework>

Neste projeto o github foi usado para hospedar o código fonte da solução desenvolvida e o código latex desta monografia. Todos esses dados podem ser acessados através do endereço <https://github.com/wancharle/Searchlight>.

Na Figura 15 podemos observar a página do código fonte do projeto. A partir dela um programador pode reportar um bug, solicitar permissões para contribuir com o projeto, copiar o código fonte e muitos outros recursos.

Figura 15 – Pagina do github.com que hospeda o código fonte deste projeto.

The screenshot shows a GitHub repository page for 'Searchlight'. At the top, there are tabs for 'Code', 'Network', 'Pull Requests (0)', 'Issues (0)', 'Wiki', 'Graphs', and 'Settings'. Below the tabs, a summary message reads: 'Trabalho de Conclusão de Curso apresentado ao Departamento de Informática do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Bacharel em Engenharia de Computação. — [Read more](#)'. Underneath this, there are buttons for 'Clone in Mac', 'ZIP', 'HTTP', 'SSH', and a text field with the URL 'git@github.com:wancharle/Searchlight.git'. To the right of the URL is a 'Read+Write access' button. Below these controls, there are dropdown menus for 'branch: master', 'Files', 'Commits', 'Branches (2)', and 'Tags'. The main content area is titled 'Searchlight / +' and shows a list of commits. The commits are:

- wancharle authored 27 minutes ago: correções cap 2 [wancharle] (latest commit 2ae0cafe8)
- monografia 27 minutes ago: correções cap 2 [wancharle]
- site 8 days ago: adicionando eventos, e estrutura [wancharle]
- site_src 8 days ago: adicionando eventos, e estrutura [wancharle]
- .gitignore 2 months ago: finalizando seção sobreposição de informação. [wancharle]
- .merge-ghpages 2 months ago: mudando o header [wancharle]
- README.md 2 months ago: fix [wancharle]

At the top right of the commit list, it says '52 commits'.

Além de hospedar os códigos fontes do projeto, o GitHub também participa ativamente do framework Searchlight. Seu papel no framework é realizado através da ferramenta GitHub Pages. Essa ferramenta estende o uso do GitHub para que ele se comporte como um servidor de páginas web simples. Isto permite ao framework Searchlight hospedar a página de geração e compartilhamento de mapas sem que seja preciso contratar um servidor web apenas para esse propósito.

3.2.2 Leaflet.js

Leaflet é uma biblioteca open-source feita em JavaScript para visualização de mapas interativos. Seu design é baseado em simplicidade, performance e usabilidade. Devido a isso é bastante amigável a dispositivos móveis como tablets e smartphones.

Leaflet é uma biblioteca moderna e robusta, que tira vantagem dos recursos de navegadores modernos como o uso de HTML5 e CSS3 mas ainda mantém suporte aos

navegadores antigos. É bastante confiável e sua estabilidade já foi testada por grandes sites da Web como Foursquare⁴ e Flickr.

Neste projeto, a biblioteca Leaflet é utilizada para fornecer os recursos de exibição e interação com mapas na web. Além disso, o framework a extensão Leaflet.Spin, para controlar o processo de carregamento de dados, e Leaflet.MarkerCluster, para agrupar marcadores.

3.2.3 TableTop.js

Tabletop([SOMA, 2013](#)) é uma biblioteca JavaScript que se comunica com o Google Docs e permite a leitura de planilhas eletrônicas por websites via Javascript.

O framework Searchlight utiliza a biblioteca Tabletop na geração e compartilhamento de mapas. A função da biblioteca, neste caso, é ler as planilhas com as informações do mapa e converter os dados para o formato JSON que é um dos formatos de dados mais utilizado em JavaScript.

3.2.4 Rapydscript

Rapydscript ([TSEPKOV, 2013](#)) é um pré-compilador para JavaScript com sintaxe bem próxima da linguagem Python. Sua principal vantagem, é a facilidade de se programar orientado a objetos de forma similar à Python.

A maior parte do framework Searchlight é escrita usando a sintaxe do Rapydscript. As partes restantes são escritas em JavaScript nativo.

Decidiu-se usar um pré-compilador para JavaScript, pois a linguagem JavaScript, apesar de ser uma linguagem com suporte a orientação a objetos, não possui uma sintaxe agradável para criação de classes de objetos. Devido a isso, é possível encontrar na internet algumas dezenas de pré-compiladores de javascript⁵.

3.3 Arquitetura do Sistema

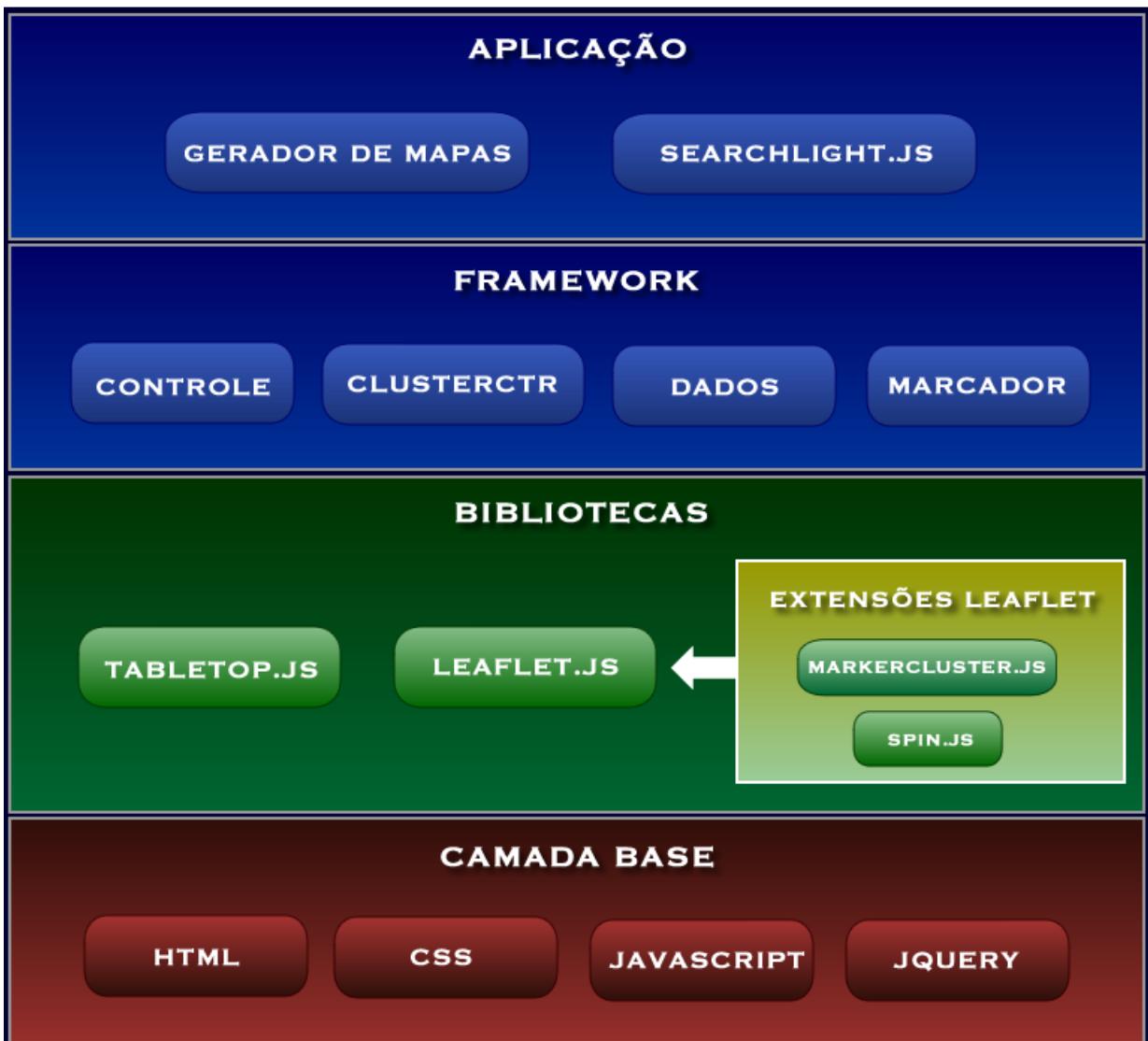
A [Figura 16](#) mostra as camadas que fazem parte da arquitetura do framework Searchlight. A arquitetura do sistema, tal como mostrada na figura, é assim explicada:

- **Camada de Aplicação:** é a camada que disponibiliza os serviços do framework aos usuários. Sendo que para usuários programadores a camada disponibiliza a biblioteca Searchlight.js que permite a integração customizada do framework ao website privado do usuário. Porém, para usuários sem conhecimentos de programação, a

⁴Foursquare abandona Google Maps em prol de leaflet.js <http://goo.gl/JXPCW>

⁵O site <http://altjs.org> exibe uma lista dos principais pré-compiladores de JavaScript.

Figura 16 – Arquitetura do framework Searchlight.



camada também disponibiliza o gerador de mapas. O gerador de mapas, permite a geração automática de mapas e o seu compartilhamento através da internet sem que seja preciso programar.

- **Camada Framework:** é a camada que implementa os serviços oferecidos para os usuários através da camada de aplicação. A camada framework implementa as classes Controle, ClusterCtr e Dados. Essas classes são responsáveis por controlar a interface de usuário; controlar os agrupamentos, filtros e foco; e pelo processamento de dados do framework.
- **Camada Bibliotecas:** é a camada que implementa serviços básicos que são utilizados pela camada framework como visualização de mapas, agrupamentos de marcadores e processamento de dados . Os seus principais componentes são a biblioteca

Leaflet, com suas extensões, e a biblioteca Tabletop. Sendo que a biblioteca Leaflet é responsável pelas funções visuais do mapa e a biblioteca Tabletop pelo processamento de dados de planilhas do Google Docs.

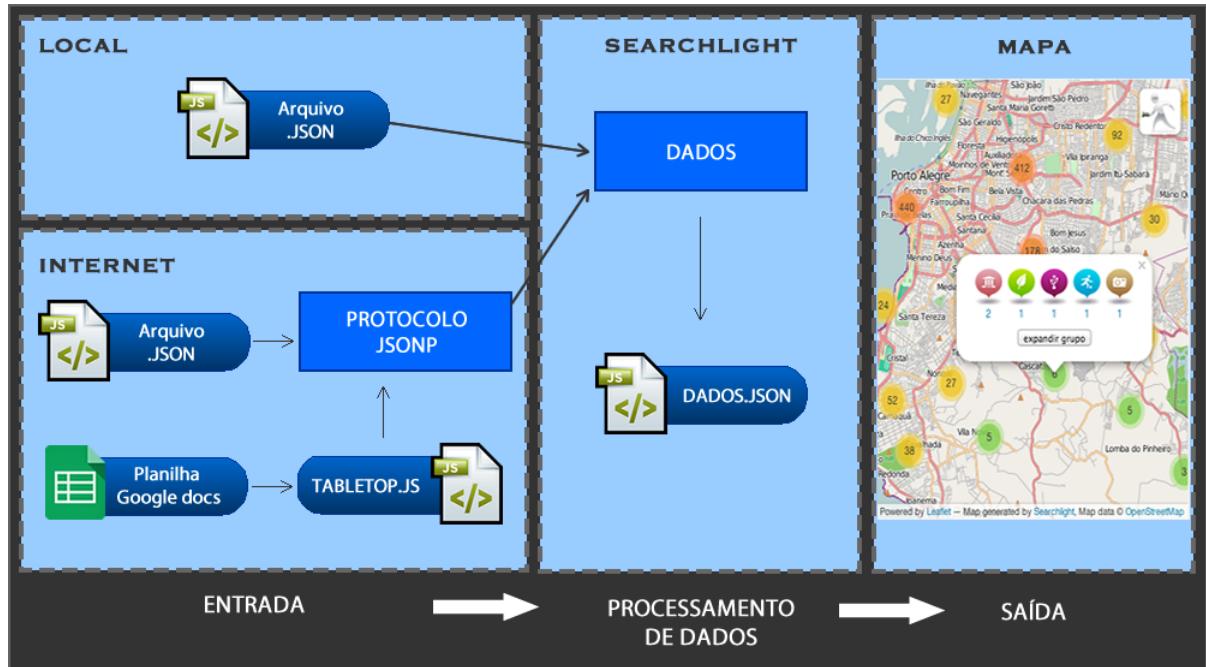
- **Camada Base:** é camada que fornece os componentes estruturais para a construção das camadas superiores. Sendo que os seus principais componentes são o jQuery, para fornecer compatibilidade com diversos navegadores, e o grupo HTML, CSS e JavaScript que são os blocos estruturais de todo o framework.

3.4 Processamento de dados

O processamento de dados do framework Searchlight é feito em duas etapas. Na primeira etapa, o framework faz a leitura dos dados e converte no formato JSON. Na segunda etapa, ele organiza os dados para a utilização nas classes do framework.

Por convenção, utiliza-se o formato JSON como fonte de dados. A entrada dos dados pode ser feita localmente ou pela internet. Dados que não estejam no formato JSON devem ser processados conforme é mostrado na [Figura 17](#).

Figura 17 – Processamento de dados do framework Searchlight.



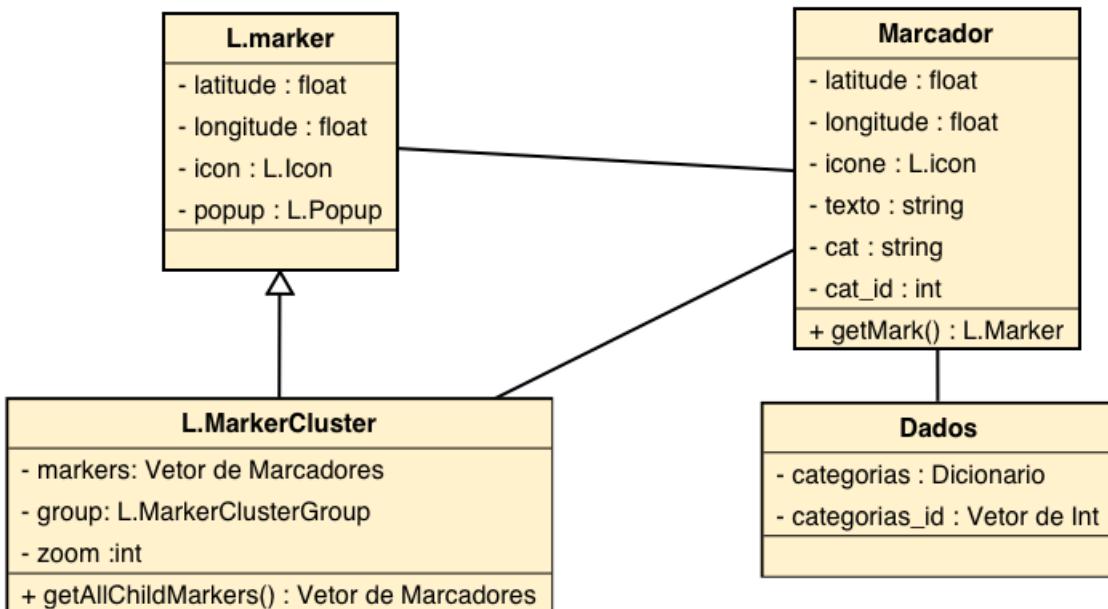
Vale ressaltar, que os dados fornecidos localmente são utilizados diretamente pelo framework. Porém, os dados que são fornecidos via internet precisam ser processados e seguir as regras do protocolo JSONP.

Arquivos de dados que não estejam no formato JSON podem ser utilizados desde que sejam convertidos e entregues via protocolo JSONP. O framework Searchlight faz

essa conversão automaticamente para planilhas do Google Docs. Neste caso, é utilizado a biblioteca Tabletop para converter os dados da planilha para o formato JSON.

Ao passar pela primeira etapa, com os dados já em formato JSON, o framework inicia o processo de organização desses dados. A [Figura 18](#) exibe um diagrama com as estruturas de dados utilizadas e suas classes simplificadas.

Figura 18 – Estruturas de dados do framework Searchlight.



De modo geral, a principal estrutura de dados utilizada pelo framework é a árvore de clusters criada pela biblioteca *Leaflet.MarkerCluster*. Esta árvore é representada na [Figura 18](#) pela classe *L.MarkerCluster*. Esta classe herda os dados da classe *L.Marker* que representa marcadores comuns que são exibidos num mapa pela biblioteca *Leaflet*.

Cada nó dessa árvore é um marcador, cada nível define um novo conjunto de agrupamentos de marcadores, e cada agrupamento está relacionado a um zoom específico. Isso faz com que todos os nós da árvore sejam marcadores virtuais, ou seja, representem apenas agrupamentos de marcadores e não informações concretas.

As folhas da árvore representam marcadores reais, ou seja, que exibem dados concretos e da fonte dados do mapa. As folhas, são representados na [Figura 18](#) pela classe *Marcador*. Essa classe, é responsável por armazenar informações básicas sobre o marcador e informações extras como categoria do marcador, texto do popup e ícone.

A classe *Dados* é responsável por armazenar todos os dados processados na primeira etapa e separá-los em categorias. Para isso utiliza-se um dicionário de categorias. Dessa forma, a chave do dicionário é o nome da categoria, e o valor é uma lista de marcadores que pertencem a essa categoria. Além disso, como recurso para a otimização

do framework, também é armazenado uma lista de identificadores numéricos para cada categoria.

Assim, por meio dessas estruturas de dados é possível controlar o agrupamento de marcadores, popups, zoom e o filtro de categorias. Além disso, o framework faz uso de outras estruturas de dados que não precisam de pré-processamento. Por exemplo, para desenvolver o recurso de fazer e desfazer ações de zoom é necessário a utilização de algumas variáveis de controle e uma estrutura de pilha para salvar as ações do usuário.

4 Solução Desenvolvida

Neste capítulo será detalhado o framework Searchlight e os recursos que o mesmo oferece para seus usuários.

4.1 Website do projeto

Pensando em ilustrar os recursos do framework Searchlight, para os usuários do framework, foi criado um website com informações do projeto. O endereço dele é <http://wancharle.github.io/Searchlight/>. Através dele podemos ter acesso ao endereço do código fonte do projeto, que se encontra no GitHub, permitindo assim que outros programadores contribuam para o mesmo.

Figura 19 – Website com documentação e exemplos do framework Searchlight.



Este website contém o link da API Searchlight.js para os programadores que querem utilizar o framework de forma customizada. Nele é possível visualizar exemplos de filtros de categorias, agrupamento de marcadores, foco em grupo, geração automática de mapas e compartilhamento de mapas, que são os principais recursos e meios de utilização do framework. Dentre eles, destacam-se a geração e compartilhamento automático de mapas que possuem páginas específicas para sua utilização.

Em suma, o website documenta o projeto como um todo. Fala sobre os objetivos e metas do projeto. É possível conferir uma parte do dele na [Figura 19](#).

4.2 Recursos desenvolvidos

A [Seção 2.2](#) descreveu duas estratégias que podem ser utilizadas para se lidar com o problema de muitos marcadores num mapa. O framework fornece recursos que se baseiam nessas duas estratégias.

4.2.1 Filtro por categorias

O filtro por categoria é um dos recursos do framework Searchlight que se baseia na estratégia de redução de marcadores. Através dele é possível selecionar quais categorias de informações devem ser exibidas pelo mapa.

Para utilizar o filtro por categorias basta que o usuário acesse o menu Searchlight, localizado no campo superior direito do mapa, faça a escolha de quais categorias devem ser exibidas e clique no botão atualizar mapa.

A [Figura 20](#) exibe o processo de visualização do filtro por categorias. Como pode ser observado, o filtro exibe um número ao lado de cada categoria. Este número representa o quantidade de marcadores que pertencem a categoria.

É importante ressaltar, que o filtro de categorias é gerado através da leitura dos dados do mapa. Se os dados utilizados pelo mapa não forem classificados por categoria o filtro não é exibido.

4.2.2 Agrupamento de marcadores

Outra recurso interessante fornecido pelo framework Searchlight é o agrupamento de marcadores. Por meio, da biblioteca Leaflet.MarkerCluster o framework agrupa os marcadores de acordo com o nível de zoom. Isto resolve somente o problema de sobreposição de informações, que é um dos problemas que motivaram a criação deste projeto.

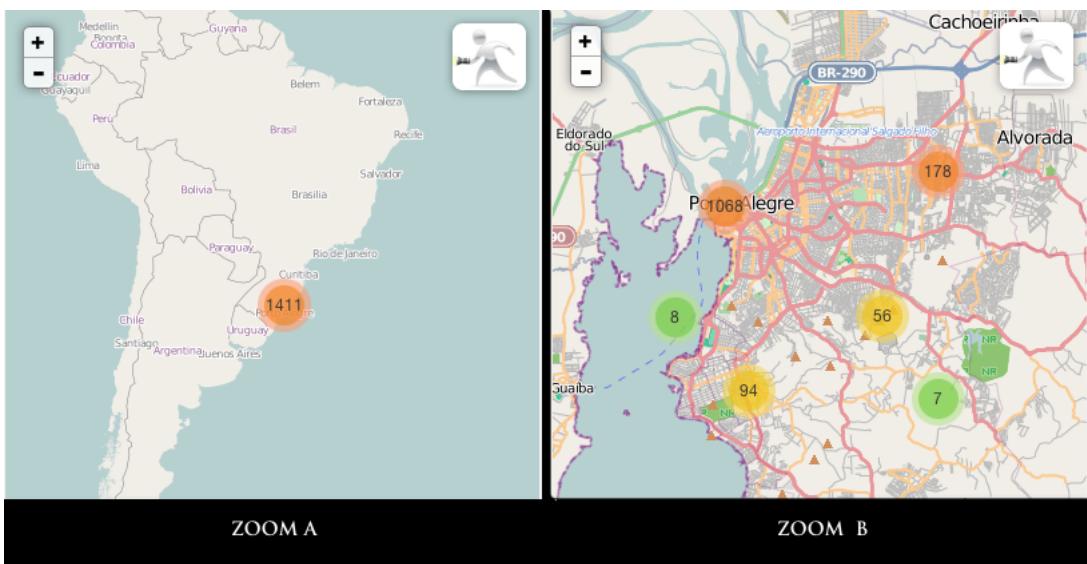
Outro problema que também é resolvido pelo agrupamento de marcadores é o problema de zoom arbitrário. Quando o usuário clica em um agrupamento de marcadores o

Figura 20 – Filtro por categoria fornecido pelo framework Searchlight



mapa calcula um nível de zoom que possibilita ver todos os subelementos do agrupamento que foi clicado. A Figura 21 exemplifica isso. Com apenas um clique no marcador central o mapa muda do zoom A para o zoom B sem precisar de zooks intermediários como no caso analisado na Figura 3.

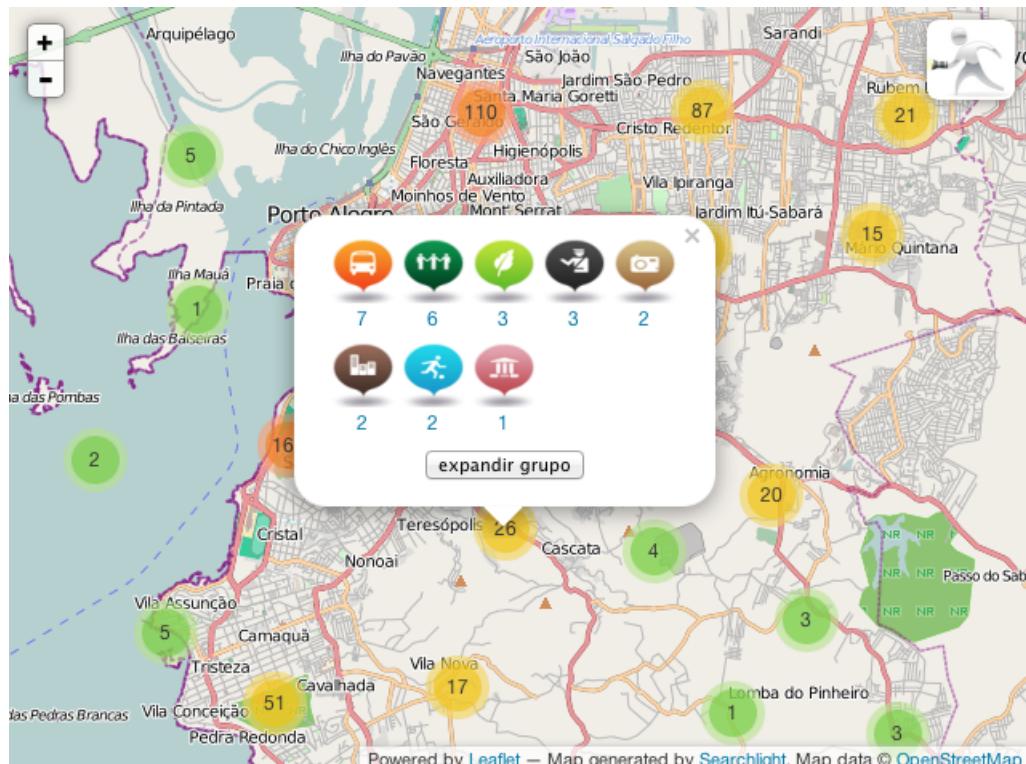
Figura 21 – No framework Searchlight os agrupamentos de marcadores eliminam o zoom arbitrário



4.2.3 Balões de Resumo e Foco em grupo

Em mapas com um elevado número de categorias seria interessante que o framework exibisse algum recurso visual para resumir a informação de cada agrupamento de marcadores. No framework Searchlight isto foi implementado através de balões de resumo.

Figura 22 – Balões de Resumo do agrupamento clicado

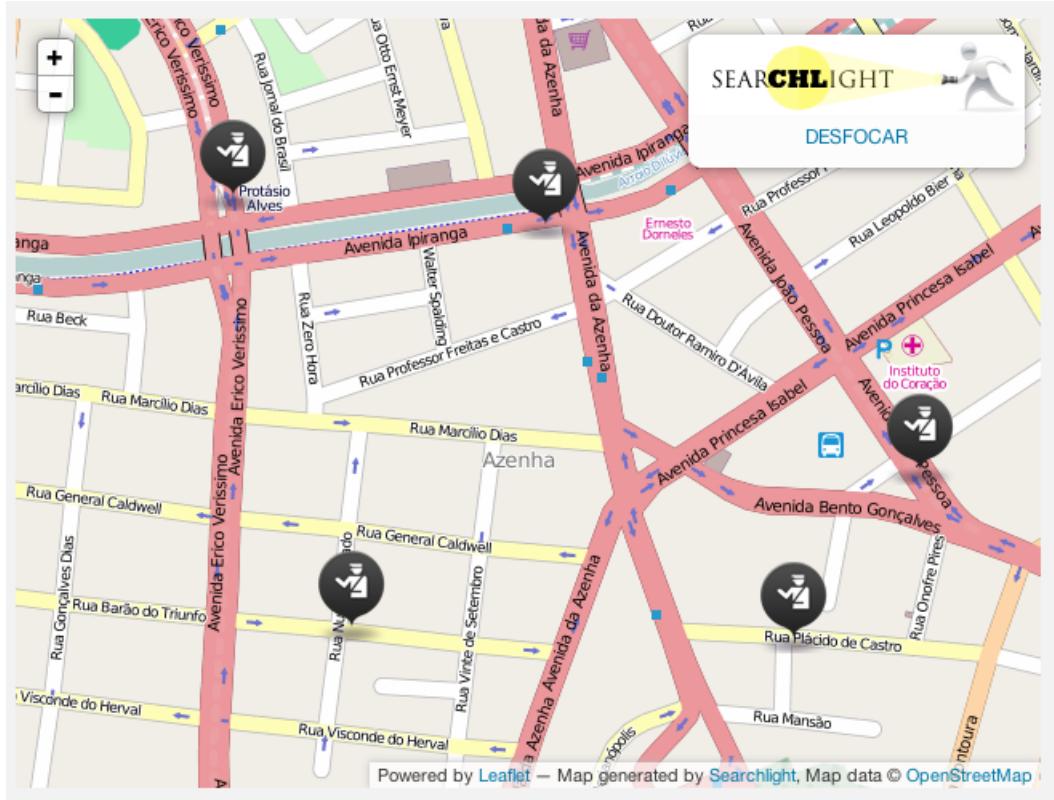


A Figura 22 exibe um balão de resumo. Como pode se observar, o Balão de resumo mostra um pequeno relatório sobre os elementos pertencentes ao agrupamento. É exibido uma listagem de ícones de categorias que pertencem ao agrupamento que foi clicado, com o total de elementos da categoria a baixo do ícone.

Balões de Resumo fornecem duas formas de interação. A primeira forma de interação é com o botão “expandir grupo” que basicamente faz um zoom no grupo. A segunda forma de interação é com os ícones de categorias. Através dele é possível utilizar o recurso “Foco em Grupo” que basicamente faz um zoom no grupo, mas exibindo apenas os elementos pertencentes ao agrupamento e a categoria que foi clicada.

O recurso de foco em grupo é útil para situações que se deseja visualizar apenas um tipo de categoria de forma rápida sem que seja preciso marcar e desmarcar categorias no filtro de categorias. Para sair do foco em grupo basta clicar no botão DESFOCAR. Veja Figura 23.

Figura 23 – Exemplo de Foco em Grupo por categoria segurança



4.2.4 Geração e Compartilhamento automático de Mapas

O framework Searchlight permite a geração automática de mapas, através de fontes de dados externas, sem a necessidade de programar o mapa. Para que isso seja possível, convencionou-se que todo objeto de informação terá que conter 3 propriedades essenciais: latitude, longitude e texto.

As propriedades **latitude** e **longitude** são responsáveis por informar a posição em que o marcador deve aparecer no mapa. A propriedade **texto** é responsável por informar o texto que será exibido no balão do marcador. O conteúdo da propriedade texto pode ser qualquer tipo de texto, inclusive código html.

A [Figura 24](#) exibe um exemplo de fonte de dados que utiliza a convenção adota pelo framework Searchlight. Neste caso, foi utilizado uma planilha pública do Google Drive e na coluna texto foi colocado códigos html e texto simples.

De modo geral, é necessário criar uma planilha de dados para gerar um mapa automaticamente. Além disso, é preciso acessar o gerador de mapas no website do projeto e colar o endereço da planilha no campo “endereço da planilha”. Por ultimo, deve-se clicar no botão “compartilhar” para visualizar o mapa gerado.

É importante perceber, que a fonte de dados não precisa ser necessariamente uma

Figura 24 – Exemplo de fonte de dados utilizada para gerar mapas automaticamente

searchlight: exemplo1 : Página1		
latitude	longitude	texto
-20,277233	-40,303752	UFES
-20,27353	-40,305448	CT
-20,279483	-40,30269	CEMUNI
		BIBLIOTECA <iframe width="320" height="240" src="http://www.youtube.com/embed/JweTAhyR4o0?rel=0" frameborder="0" allowfullscreen></iframe>
-20,276519	-40,304503	

Publicado por [Google Drive](#) – [Denunciar abuso](#) – Atualizado automaticamente a cada 5 minutos

planilha do Google Drive. O framework Searchlight também suporta o formato de dados JSON. E por meio do protocolo JSONP pode usar estes arquivos como fonte dados na geração automática de mapas.

5 Considerações Finais

Este capítulo apresenta as conclusões do trabalho realizado, mostrando suas contribuições. Por fim, são apresentadas suas limitações e perspectivas de trabalhos futuros.

5.1 Conclusões

Com a popularização dos computadores e da internet, está acontecendo um aumento da interação social entre as pessoas no meio virtual. Essa interação permite o cultivo de novos comportamentos e interações sociais como o crowdsourcing. Além disso, tem crescido a necessidade de mecanismos virtuais que permitam as pessoas exercer sua cidadania de forma conectada e contribuir para a cidade com críticas e sugestões. Entretanto, tais mecanismos precisam lidar com uma enorme quantidade de informação e isso dificulta a visualização dos dados coletados.

Neste contexto, este trabalho apresentou a criação do framework Searchlight com o intuito de facilitar a visualização de informações crowdsourcing em Mapas Web. Pesquisou-se por estratégias para resolver os problemas de sobreposição de informações, zoom arbitrário e informações arbitrárias. E para solucionar esses problemas, foi necessário utilizar as estratégias de redução de número de marcadores e agrupamento de marcadores.

Para a aplicar essas estratégias, o framework Searchlight implementou alguns recursos como o filtro por categorias, o agrupamento de marcadores, Balões de Resumo e foco em grupo. Esses recursos, ajudaram a melhorar a visualização do mapa e a deixá-lo mais limpo e compreensível.

Durante a fase de pesquisa, percebeu-se, que além de melhorar a visualização do mapa, era necessário melhorar também o acesso ao mapa. Para isso, foi feito a implementação do recurso de gerar e compartilhar mapas automaticamente. Com esse recurso, usuários que não possuem conhecimento de programação também podem criar e compartilhar os seus próprios mapas.

5.2 Dificuldades Encontradas

No desenvolvimento do framework Searchlight foram encontradas dificuldades que podem ser resumidas em 3 áreas distintas: Design, Compatibilidade e Detecção de Erros.

Na área de Design a maior dificuldade foi decidir qual seria os objetos de interface

para interagir com o usuário. Por exemplo, para exibir os balões de resumo inicialmente pensou-se em utilizar o botão direito do mouse. Porém, isso gera um empecilho para a utilização do framework em tablets e smartphones, pois estes aparelhos não possuem a opção de click com botão direito do mouse.

Para resolver esse problema, adotou-se um click para abrir o balão de resumo e duplo click para realizar zoom em grupo. Isso aumentou a complexidade do controle de interface, pois foi necessário criar algumas variáveis de controle para tratar o toque e duplo toque no contexto de tablets e smartphones.

Na área de Compatibilidade, a maior dificuldade foi com as diferentes regras e políticas de segurança adotadas pelos navegadores. Por exemplo, o navegador Firefox não permite acesso direto aos arquivos JSON quando a página HTML é acessada localmente via protocolo file. Porém, se a página é acessada localmente mas via protocolo http o Firefox libera o acesso.

Também ocorreram dificuldades para exibir os balões de resumo nos navegadores Firefox e Internet Explorer. Para solucionar esse problema, foi preciso definir uma largura fixa para o balão de resumo que é exibido nesses navegadores.

No entanto, de todos os problemas encontrados, o que mais atrapalhou o desenvolvimento do projeto foi a detecção de erros. JavaScript é uma linguagem de script e por isso boa parte dos erros acontecem durante a execução do código. Isso dificulta o desenvolvimento pois é difícil localizar o que provocou o erro. Em muitos casos, as mensagens de erro são consequências de um erro que não é exibido pelo navegador. Por causa disso, pode-se perder dias procurando um erro que em uma linguagem compilada seria detectado no momento da compilação.

5.3 Limitações e Perspectivas

No cenário alcançado, algumas limitações são observadas, o que dá margem para a realização de trabalhos futuros. Dentre elas destaca-se o fato da geração automática de mapas aceitar apenas planilhas do Google Docs e não ser possível utilizar planilhas de aplicativos como Excel, Calc e Numbers. Atualmente, o usuário pode adicionar seus dados nesses outros formatos de planilhas, mas para usá-las no mapa é necessário que elas sejam convertidas para o formato de planilha do Google Docs.

Além disso, na [Seção 2.2](#) foi demonstrado que uma das estratégias para se reduzir o número de marcadores é a Otimização visual. Um exemplo de otimização visual genérica que poderia ser implementada no framework é a conexão de marcadores de uma mesma categoria. Por exemplo, se considerarmos um mapa que exibe todos os pontos de ônibus de uma cidade e conectarmos os pontos pertencentes a uma mesma linha de ônibus, podemos

substituir todos os marcadores do trajeto por um desenho de uma linha passando pela localização dos pontos de ônibus.

Ainda assim, há espaço para acréscimos de novas funcionalidades como: filtros por data em mapas temporais; campos de busca em mapas com muita informação textual; utilização de mais de uma planilha como fonte de dados e muitas outras funcionalidades.

Portanto, vislumbra-se a possibilidade de trabalhos futuros buscando o oferecimento de mais serviços por parte do framework Searchlight e uma melhoria continua dos recursos de visualização de mapas de crowdsourcing.

Referências

LEAFLET. *An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps*. maio 2013. [Http://leafletjs.com](http://leafletjs.com). Disponível em: <<http://leafletjs.com>>. Citado na página 26.

LEAFLET.MARKERCLUSTER. *Provides Beautiful Animated Marker Clustering functionality for Leaflet, a JS library for interactive maps*. maio 2013. [Http://github.com/Leaflet/Leaflet.markercluster](http://github.com/Leaflet/Leaflet.markercluster). Disponível em: <<http://github.com/Leaflet/Leaflet.markercluster>>. Citado 2 vezes nas páginas 26 e 27.

MAPATL. *MapATL - Visualizing Atlanta's Crime*. mar. 2013. [Http://crime.mapatl.com/](http://crime.mapatl.com/). Disponível em: <<http://crime.mapatl.com/>>. Citado 2 vezes nas páginas 9 e 10.

PORTOALEGRE.CC. *PortoAlegre.cc é um espaço de colaboração cidadã, onde você pode conhecer, debater, inspirar e transformar a própria cidade. Participe*. mar. 2013. [Http://www.portoalegre.cc/](http://www.portoalegre.cc/). Disponível em: <<http://portoalegre.cc/>>. Citado 2 vezes nas páginas 8 e 9.

QUIRINO, W. S. *Searchlight - Facilitando a visualização de informações em mapas crowdsourcing*. maio 2013. [Http://wancharle.github.com/Searchlight/](http://wancharle.github.com/Searchlight/). Disponível em: <<http://wancharle.github.com/Searchlight/>>. Citado na página 12.

SHEIKHOLESLAMI, G.; CHATTERJEE, S.; ZHANG, A. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal—The International Journal on Very Large Data Bases*, 2000. Springer-Verlag New York, Inc., v. 8, n. 3-4, p. 289–304, 2000. Citado 2 vezes nas páginas 10 e 23.

SILVA, R. F. Solap+. 2010. Faculdade de Ciências e Tecnologia, 2010. Citado 4 vezes nas páginas 10, 22, 23 e 26.

SOMA, J. *Tabletop.js takes a Google Spreadsheet and makes it easily accessible through JavaScript*. maio 2013. [Https://github.com/jsoma/tabletop](https://github.com/jsoma/tabletop). Disponível em: <<https://github.com/jsoma/tabletop>>. Citado na página 29.

SVENNERBERG, G. *Beginning Google Maps API 3*. [S.l.]: Apress, 2010. Citado 6 vezes nas páginas 13, 14, 15, 19, 20 e 26.

THIAGARAJAN, A. et al. Cooperative transit tracking using smart-phones. In: *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. New York, NY, USA: ACM, 2010. (SenSys '10), p. 85–98. ISBN 978-1-4503-0344-6. Disponível em: <<http://doi.acm.org/10.1145/1869983.1869993>>. Citado na página 8.

THIAGARAJAN, A. et al. VTrack: accurate, energy-aware road traffic delay estimation using mobile phones. In: *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*. [s.n.], 2009. p. 85–98. Disponível em: <<http://doi.acm.org/10.1145/1644038.1644048>>. Citado na página 8.

TSEPKOV, A. *RapydScript is a pre-compiler for JavaScript, similar to CoffeeScript, but with cleaner, more readable syntax.* maio 2013. [Https://bitbucket.org/pyjeon/rapydscript](https://bitbucket.org/pyjeon/rapydscript). Disponível em: <<https://bitbucket.org/pyjeon/rapydscript>>. Citado na página 29.