EXPLORANDO O HTML5 PARA VISUALIZAÇÃO DE DADOS GEOGRÁFICOS

Marcel Mendonça Grilo[1], Jéferson José Ribeiro[2], Sérgio Souza Costa[3]

[1]Cientista da Computação, UNIFEI, Itajubá-MG, marcelgrilo86@gmail.com
[2]Cientista da Computação, UNIFEI, Itajubá-MG, jefersonjribeiro@gmail.com
[3]Doutor em Computação Aplicada, Professor Adjunto, UFMA, São Luís-MA, prof.sergio.costa@gmail.com

RESUMO: Os novos recursos do HTML5 estão trazendo novas possibilidades para o desenvolvimento de aplicações web, como por exemplo, as aplicações para visualização de dados geográficos. Agora é possível visualizar dados vetoriais diretamente nos navegadores web. Este recurso pode ter um grande impacto na forma de desenvolver e projetar essas aplicações, pois não requer que estes dados sejam convertidos em imagens. Considerando este novo cenário, este trabalho tem como objetivo investigar estes novos recursos, identificando suas vantagens e restrições para visualização de dados vetoriais através da web.

PALAVRAS-CHAVE: SIG, HTML5, Servidor de Mapas, Dados Geográficos

INTRODUÇÃO:

A utilização de dados geográficos é de extrema importância em diversas aplicações, como por exemplo nas áreas sociais, acadêmicas ou de políticas, por se tratar de dados que representam com precisão objetos e fenômenos em localizações geográficas (CÂMARA; QUEIROZ, 2001). Estes dados são coletados por diversas fontes e armazenados em bancos de dados geográficos para serem visualizados posteriormente. Para que esses dados geográficos possam ser melhor visualizados surge a necessidade da criação de servidores que possam suportar esse grande volume de dados armazenados. Os servidores de mapas tem o objetivo de permitir a visualização de dados geográficos através da *web*. Hoje em dia esse tipo de serviço ajuda as pessoas nas suas atividades do dia a dia, como buscar endereços, os melhores caminhos para chegar a um determinado destino. Os locais inseridos nos mapas podem ser acompanhados de informações, como por exemplo, estabelecimentos comerciais e a orientação para o motorista encontrá-los. Estas informações são armazenadas em banco de dados espaciais, utilizando representações vetoriais, como pontos, linhas e polígonos. Antes da versão cinco do HTML, não existia um mecanismo que permitisse os navegadores *web* apresentar os dados vetoriais ao usuário. Isso exigia que os servidores de mapas convertessem os dados vetoriais para imagens, para que essas pudessem ser visualizas nos navegadores. Este sistema baseado em tiles (blocos em português) funciona processando os vetores dos mapas em um servidor, gerando uma imagem para um determinado retângulo envolvente, em seguida, envia esta imagem ou conjunto de imagens para a máquina do usuário (KROPLA, 2005). A função do navegador neste caso é apenas apresentar as imagens na tela. Esta alternativa é problemática, por diversos motivos, em primeiro lugar, porque a transmissão de imagens é em geral demorada, e realizada de forma repetitiva tende a sobrecarregar os recursos de rede. E segundo, existe o problema de sobrecarga no servidor, que precisa construir o mapa em formato imagem, geralmente a partir de um banco de dados vetorial, e transmiti-lo para o cliente. Note-se que qualquer operação simples, como *zoom* ou *pan*, exige a formação de uma nova imagem daquele dado e sua retransmissão. Contudo, esta abordagem ainda é amplamente utilizada pelos servidores de mapas. Deste modo, este trabalho tem como objetivo explorar os novos recursos do HTML 5 para visualização de dados geográficos.

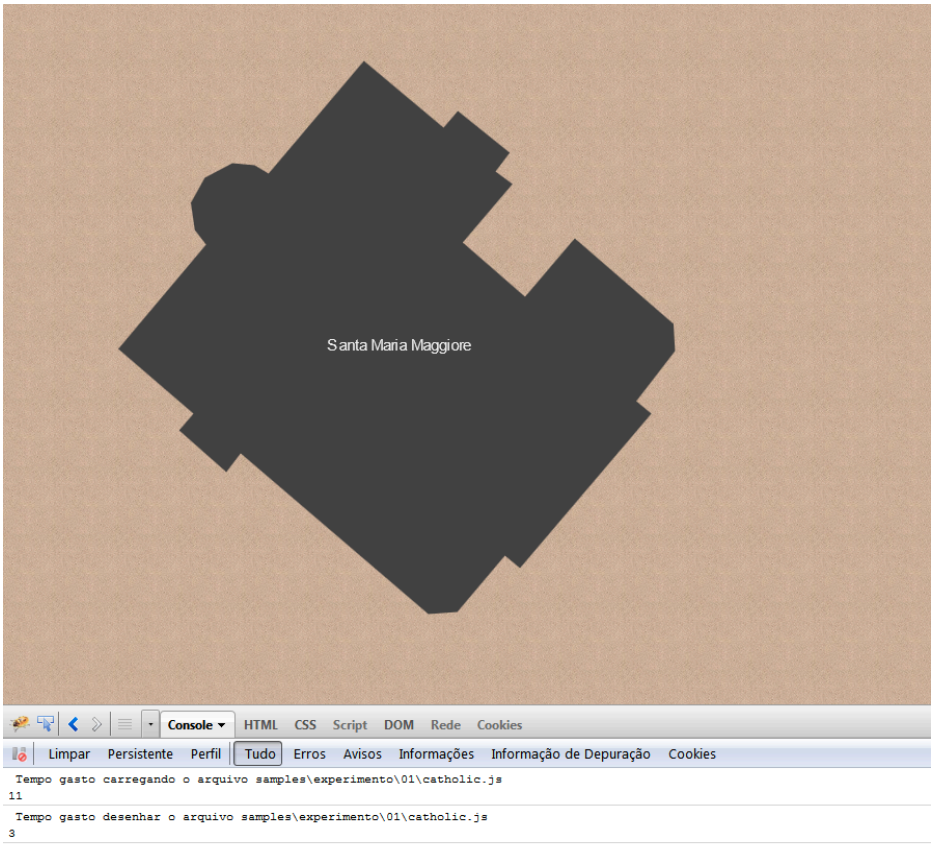
MATERIAL E MÉTODOS: Neste trabalho, comparamos fatores de tempo de transferência de dados juntamente com a capacidade de armazenamento, tempo de processamento e capacidade de personalização. Para todos os fatores, sempre que possível consideramos aspectos tanto do lado do servidor quanto do lado cliente. Utilizamos para os experimentos um computador com o sistema operacional Windows 7, processador Core i3 e 4 GB de memória RAM. O Cartagen e OpenStreetMap (OSM) foram os principais softwares que utilizamos, como detalharemos a seguir.

O OpenStreetMap é um serviço que disponibiliza de forma livre mapas do mundo todo. Os mapas são editáveis por pessoas comuns, que utilizam dados de receptores GPS ou informações manuais (HAKLAY; WEBER, 2008). Este serviço é dedicado a incentivar o crescimento, desenvolvimento e distribuição de dados geoespaciais de maneira livre, para que qualquer um use e compartilhe. As pessoas podem escrever mensagens nos mapas, indicando algum erro ou acrescentando informações. As informações atribuídas ao OSM são acumuladas sob a forma de pares que são constituídos de chaves e valores, sendo estes conjuntos flexíveis a possíveis mudanças.

O Cartagen provê visualização de dados geográficos em um navegador web utilizando os novos elementos e recurso do HTML 5, mas especificamente o Canvas (BOULOS *et al.*, 2010). Este elemento, unido com uma API em JavaScript, permite desenhar geometrias como pontos, linhas e polígonos no navegador. Contudo, ela não é específica para dados geográficos, tornando complexo o seu uso para estes dados. Deste modo, utilizaremos o Cartagen que já provê uma API específica para dados geográficos. O Cartagen estiliza os mapas a serem desenhados através de Geographic Style Sheet (GSS), que é uma especificação na forma de folha de estilo em cascata para representação de informação geográficas, baseada em CSS e JSON. O Cartagen renderiza dados no formato JSON (Java Script Object Notation), mas especificamento no formato GeoJSON. Este é um formato de troca de dados geoespaciais que se baseiam em JSON. Um objeto GeoJSON pode apresentar uma forma geométrica, uma função ou diversas características. Nas formas geométricas, o GeoJSON suporta os tipos Point, LineString Polygon, MultiPoint, MultiLineStrings, MultiPolygon e GeometryCollection.

O carregamento de mapas da aplicação Cartagen é feito de duas maneiras - estaticamente e dinamicamente - porém ambas lidam diretamente com vetores. O carregamento estático é feito através de um único arquivo presente em um diretório ou servidor. Quando o Cartagen carrega um mapa estático ele já carrega todos os vetores daquele mapa e os mantém na memória, independentemente da porção de vetores que estejam sendo visualizado na tela do navegador. Diferentemente, quando os mapas são carregados dinamicamente apenas estará na memória o que estiver sendo visualizado na tela do canvas. Para isto, existe a necessidade de uma conexão com um servidor de mapas que forneça estas porções de vetores. Estas porções de vetores nos servidores de mapas atuais (como o OSM) são calculadas e armazenadas ordenadamente nos servidores de tiles que geram as imagens a partir de uma dada porção de vetores e envia estas imagens para o navegador do cliente. O que o Cartagen propõe ao usar mapas carregados dinamicamente é remover a necessidade de um serviço que gere as imagens a partir dessas porções de vetores, transferindo esta tarefa direto para o navegador do cliente e diminuindo o tráfego de dados na rede. Neste trabalho usaremos apenas o carregamento estático.

RESULTADOS E DISCUSSÃO: Para este experimento foi utilizado um dado vetorial (23880 Bytes) do contorno da área ocupada pela Basílica de Santa Maria Maior em Roma. Em nossos experimentos, o Cartagen teve um tempo médio de 11 milissegundos para preparar e carregar este dado para memória e 3 milissegundos para o desenho na tela. Para a medição de tempo, foram incluídos códigos que verificava o tempo antes e depois de um determinado processo. A diferença destas medidas era impressa no console do navegador, como observado na Figura 1.



No Cartagen os níveis de zoom são obtidos através de uma transformação vetorial de escala no vetor dado. Analogamente, um servidor de mapas também executa esta operação, porém este servidor deve produzir e armazenar a imagem ou as imagens para cada nível de zoom dado. Por exemplo, a Tabela 1 mostra os tamanhos das imagens geradas pelo dado apresentado nos formatos bitmap, GIF, JPEG e PNG de dimensão 1920x978 pixels.

Tipo	Vetorial	Bitmap	GIF	JPEG	PNG
Tamanho em bytes	23880	5633334	278831	622867	184460

Considerando 10 níveis de zoom com o mesmo tamanho da tela, verifica-se que o servidor de mapas deveria gerar ou enviar para a máquina do cliente um conjunto de imagens para cada um desses níveis de zoom. Levando em conta que as imagens são armazenadas no formato PNG, tem-se que o total, em pixels, para os 10 níveis de zoom seria 1.844.600 Bytes. Observando a razão entre o mapa no formato GeoJSON e o total das imagens em bytes, resultando em:

$$\frac{23880}{1844600} = 0{,}01294$$

Assim observa-se que o tamanho do vetor representa aproximadamente 1,3% do tamanho da imagem. Esta economia possibilitaria dados vetoriais mais ricos em informações. Além disso, se focarmos no envio de imagens dos servidores de mapas para o cliente tem-se um caso muito semelhante. A mesma lógica e os mesmos números se aplicariam também neste caso. Uma vez que o cliente recebe um dado vetorial, este não precisa mais requisitar ao servidor por atualizações caso faça operações de escala (zoom), rotação ou translação. Isso não ocorre com as imagens, ou seja, qualquer interação feita pelo usuário resulta em uma requisição ao servidor, o que faz com que o servidor processe muitas imagens, e ainda lide com o monitoramento de requisições vindas do lado do cliente.

Para calcular o tempo de transferência, assumiremos uma velocidade de 2,1 Megabits por segundo (Mbps). Segundo Rodrigues (2012) essa seria a taxa média de velocidade de conexão de banda larga no Brasil. Com essa velocidade de conexão, tem-se: (2,1*1000*1000bits) = 2100000 bits por segundo, transformando para byte, temos 262500Bytes por segundo. Assim, a taxa de transferência para download de uma conexão de 2,1Mbps em bytes por segundo é de 262500Bps (262500 Bytes por segundo). Como os tamanhos discutidos aqui estão em bytes, calcula-se:

Tempo de transferência para o vetor: 23880Bytes / 262500Bps = 0,09097 segundos.
Tempo de transferência para 1 imagem: 184460Bytes / 262500Bps = 0,70270 segundos

Quando se lida apenas com a transferência, temos uma razão de 0,09097(s) / 0,70270(s) = 0,12945. Ou seja, o tempo para descarga do vetor é aproximadamente 13% do tempo gasto para a descarga de uma imagem. No caso onde ocorrem transformações no mapa, podemos fazer este cálculo considerando 10 níveis de zoom. Para cada um dos dez níveis de zoom, tem-se uma imagem, logo calcula-se a velocidade total de carregamento das 10 imagens obtendo:

1844600Bytes / 262500Bps = 7,02704 segundos

Observa-se que o tempo de download das 10 imagens quase 100 vezes maior, em comparação com a transmissão do vetor. Essa observação enfatiza o potencial do uso do sistema vetorial sobre o sistema de imagens. Com relação ao tempo de processamento não se pode afirmar com exatidão, pois, cada máquina pode ter um tempo diferente devido a diversos aspectos, como processadores, memória, sistema operacional e navegadores. Porém, o que se enfatiza aqui é a velocidade geral do processamento vetorial que em nossos experimentos pareceu ser um período de tempo aceitável, como mostrado na Figura 1.

O Cartagen apresenta uma ideia excelente em relação à diminuição da carga de processamento dos servidores de mapas, porém ainda não possui muitas funcionalidades esperadas. Neste trabalho o Cartagen foi estendido a fim de permitir o usuário gerenciar mapas localmente, ou seja, sem precisar estar conectado na internet. Para isso foi utilizada uma nova API *Local Storage* do HTML5 para o armazenamento de dados locais. Este recurso torna possível o desenvolvimento de aplicações que podem ser executadas sem conexão com a internet. Neste trabalho, exploramos este recurso com o objetivo de permitir ao usuário selecionar e visualizar mapas armazenados localmente. Os mapas armazenados localmente podem ser carregados de um servidor de dados vetoriais. Uma alternativa interessante seria o uso da API Rest (*Representational State Transfer*) fornecida pelo servidor OSM. Deste modo seria possível carregar os mapas usando apenas requisições HTTP. Contudo, neste trabalho não utilizamos esta API, ficando assim para trabalhos futuros. Para prova de conceito, exportamos o mapa diretamente na interface *web* do OSM para um arquivo XML que é o formato do OSM. Como o Cartagen suporta apenas arquivos no formato de arquivo JSON foi necessário codificar um algoritmo que converte os dados do formato XML para o formato JSON. Para salvar o arquivo do lado cliente foi utilizada a API FileSaver do HTML5 (The FileSaver interface, 2012).

CONCLUSÕES: Pelas experiências realizadas, foi possível notar que trabalhar com processamento vetorial usando o Cartagen é uma alternativa interessante. O seu uso reduz a quantidade de requisições ao servidor de mapas para processamento das imagens, dado que o Cartagen utiliza o mesmo dado vetorial para os diversos níveis da imagem. Com o uso da linguagem GSS, O Cartagen consegue separar a apresentação dos dados. Essa característica também contribui para diminuição de carga do servidor, dado que o mesmo dado pode ser visualizado de distintas maneiras, sem a necessidade de novo carregamento. Neste trabalho foi ainda desenvolvida uma extensão a fim de permitir o usuário gerenciar mapas localmente, obtido de um servidor de mapas. Contudo existem ainda diversas outras funcionalidades:

- **Cálculo de rotas**, não existe atualmente, porém pode ser desenvolvido, desde que esteja integrado com um servidor com este recurso.

- **Detalhes geográficos e topográficos**, já é suportado através da inclusão de novas camadas de dados geográficos.
- **Informações Periféricas** (Clima, Trânsito, Imagens, etc.), ainda não existe, porém seria possível estender adaptando o formato de dados ou gerar um formato para este tipo de informação. Neste caso seria um tipo de dado dinâmico no sentido de que são necessárias atualizações frequentes nos dados.
- **Configurações de usuário** (marcação de locais, armazenamento de rotas, imagens, vídeos, webcam, etc.), assim como em muitos sistemas, as seções de usuários são gerenciadas separadamente, portanto é possível que o Cartagen faça uso destas funcionalidades.

O HTML5 ainda é uma tecnologia recente, porem é possível observar que ela já está influenciando no desenvolvimento de aplicações poderosas, no lado cliente. Estas aplicações são mais rápidas e dinâmicas, fornecendo muito mais liberdade e interação aos usuários. Aos desenvolvedores ela oferece a capacidade de explorar muitos dos recursos que antes eram impossíveis ou muito complexos de se desenvolver. Um exemplo disso é equilíbrio da carga de processamento entre cliente e servidor, diminuindo o tráfego de informações e a carga de processamento computacional. Entretanto, ainda é cedo para dizer que este trará um impacto que quebre diversos paradigmas dos sistemas de informação geográfica. Porém, já se pode identificar e analisar diversas aplicações que trazem muitas ideias inovadoras as quais possuem muitas chances de prosperar, e que talvez com investimento, possam trazer grandes mudanças aos sistemas de informação geográficas voltados para a web.

REFERÊNCIAS:

BOULOS, M. N. K.; WARREN, J.; GONG, J.; YUE, P. Web GIS in practice VIII: HTML5 and the canvas element for interactive online mapping. **International journal of health geographics**, v. 9, p. 14, jan 2010.

CÂMARA, G.; QUEIROZ, G. R. Arquitetura de sistemas de informação geográfica. **Introdução à Ciência da Geoinformação**. [S.l: s.n.], 2001. p. 1–12.

HAKLAY, M.; WEBER, P. OpenStreet map: User-generated street maps. **IEEE Pervasive Computing**, v. 7, n. 4, p. 12–18, 2008.

KROPLA, B. **Beginning MapServer**. [S.l.]: Apress, 2005. p. 427