Visualização de Dados em Dispositivos Móveis: Um Estudo de Caso com Dados Pluviométricos

Wallyson Nunes Alves Lima[1], Milton Hirokazu Shimabukuro[2]

[1]Departamento de Matemática e Computação (DMC) – Faculdade de Ciências e Tecnologia da Unesp (FCT/Unesp)

wallyson.n.a.lima@gmail.com, milton.h.shimabukuro@unesp.br

Abstract.

Mobile devices have become popular over the years, earning more and more resources. One area that has been growing and being used on mobile devices is the visualization of data, with several scientific works being produced. In this context, it was proposed to analyze different visualization techniques to verify the suitability to mobile devices, to identify the inherent limitations of these devices and to compare with a web version.

It was verified that there was some difficulty in replicating some features of the web version in the mobile version, since there were differences in the use of the interface between the mobile device and the traditional computers (Interface touch x keyboard and mouse), and it was seen that the web version obtained a better performance than the mobile version.

Resumo.

Os dispositivos móveis têm se popularizado ao longo dos anos, ganhando cada vez mais recursos. Uma área que tem crescido e sendo utilizada nos dispositivos móveis é a visualização de dados, com diversos trabalhos científicos sendo produzidos. Neste contexto, foi proposto analisar diferentes técnicas de visualização para verificar a adequação aos dispositivos móveis, identificar as limitações inerentes a esses dispositivos e comparar com uma versão web.

Verficou-se que houve certa dificuldade em replicar alguns recursos da versão web na versão mobile, pois haviam diferenças no uso da interface entre o dispositivo móvel e os computadores tradicionais (Interface touch x teclado e mouse), e foi visto que a versão web obteve um desempenho melhor do que a versão mobile.

1. Introdução

Neste artigo serão abordadas técnicas de visualização em dispositivos móveis aplicadas aos dados de índices pluviométricos, coletados por um longo período de tempo em centenas de postos do estado de São Paulo, com o objetivo de suportar a interpretação da grande quantidade de dados. Nesta linha, foi proposto criar representações visuais explorando algumas técnicas para facilitar a análise dos dados em dispositivos móveis, considerando as limitações inerentes tais como tamanho pequeno da tela, o limitado poder de processamento, pouca memória e duração da bateria (Weng et al., 2012). Por isso, ao serem criadas visualizações em dispositivos móveis é preciso pensar cuidadosamente se a visualização é adequada à aquele tipo de tela, se a interface touch (mão) é adequada ao invés de mouse e teclado como nos computadores tradicionais, as visualizações precisam ser dinâmicas, os usuários precisam interagir e não apenas visualizar. (Ward; Grinstein; Keim; 2010).

A principal biblioteca utilizada para gerar as visualizações foi a Data-Driven Document (D3) que é uma biblioteca Javascript que manipula documentos baseados em dados. Ela ajuda a visualizar dados usando Hypertext Markup Language (HTML), Scalable Vector Graphics (SVG) e Cascading Style Sheets (CSS) com ênfase em padrões web e rodando nos principais navegadores modernos utilizando o padrão Document Object Model (DOM), que é uma interface que fornece uma representação estruturada do documento como uma árvore e permite manipular os objetos. Outra biblioteca utilizada neste estudo foi a NVD3 que é uma framework que reutiliza as visualizações e componentes do D3.js, as iniciais NV possuem relação com a empresa Novus Partners que criou a biblioteca.

Duas aplicações foram definidas e implementadas para a análise dos dados por meio das visualizações. A primeira, foi chamada de Mobile Visualization Tool (MobiViTool), é uma webapp que usa visualizações no Android por meio de Webviews. A segunda, foi construída utilizando somente linguagens web, guardando similaridade com a primeira, chamada Web Visualization Tool (WebViTool). Em ambas foi utilizado o D3 e o NVD3 e serão comparadas como solução ao problema de interpretação e análise dos dados pluviométricos.

Ao longo deste trabalho serão apresentados na Seção 2 a definição da visualização de dados e a sua importância para a sociedade, o seu crescimento e o processo para se criar uma visualização. Na Seção 3 será visto o referencial teórico sobre a plataforma Android e a implementação de um Webapp. Em seguida na Seção 4 será visto a arquitetura e implementação deste trabalho, serão vistos as tecnologias utilizadas, a arquitetura das aplicações bem como seu fluxo de funcionamento. Posteriormente na Seção 5 será feito a análise de desempenho entre as aplicações, as suas limitações, diferenças, as dificuldades encontradas no desenvolvimento de ambas e os resultados obtidos. Por fim na Seção 6 será visto a conclusão deste trabalho bem como o que pode ser feito futuramente.

2. Visualização de Dados

A visualização é definida como a representação da informação por meio de representações gráficas, pois uma imagem pode sintetizar uma grande quantidade de informação e ela é processada de forma mais eficiente pelo ser humano do que observar apenas dados textuais. Com isso, a visualização da informação pode ampliar a capacidade cognitiva armazenando uma grande quantidade de informação em rápidas e acessíveis formas usando representações visuais (Neugebauer et al., 2015).

As representações visuais possuem a vantagem de ser independentes de linguagem, qualquer pessoa que saiba interpretar um gráfico é capaz de extrair informações da representação visual. E em nosso cotidiano é comum ser encontrado visualizações da informação que facilitam o seu entendimento, tais como uma tabela em um jornal, o mapa do metrô, um mapa de uma cidade, o gráfico do tempo, dentre outros. Fica claro com esses exemplos o poder da visualização de dados, pois torna mais intuitiva a interpretação da informação através das representações visuais.

A área da visualização de dados tem crescido rapidamente desde a sua criação, com a disponibilização de uma grande quantidade de diferentes técnicas e ferramentas e a maior parte das visualizações são inteiramente construídas em

tecnologias para Web, utilizando principalmente Javascript, HTML e CSS. (Kerren et al., 2017).

As representações visuais seguem um processo ao serem criadas. Os dados podem estar armazenados de maneira simples ou estruturados e podem vir de uma variedade de locais, tais como de um servidor web, banco de dados e até mesmo arquivos de dados textuais. A visualização visa transformar dados brutos, difíceis de interpretar, em representações visuais a fim de tornar mais intuitiva a extração da informação pelo usuário. O pipeline de Visualização, apresentada na Figura 1, mostra as etapas do processo de criação ou geração de uma representação visual: (Ward; Grinstein; Keim; 2010)

- **Modelagem de dados.** Nesta etapa, os dados que podem estar em um arquivo ou banco de dados, devem ser estruturados e é preciso conhecer o nome, tipo, tamanho e atributos dos dados;
- **Seleção de Dados.** Nesta etapa, são identificados o subconjunto de dados que serão utilizados para a visualização. Isto pode ser feito por meio de algortimos;
- **Mapeamento Visual dos Dados.** Esta é uma etapa importante, citado frequentemente como o coração da visualização, na qual é realizado o mapeamento dos dados em entidades gráficas. Nesta etapa são controlados os atributos da visualização como tamanho, cor e posição do objeto;
- **Configuração de Parâmetros de Cena.** As visualizações podem ter atributos que são independentes dos dados que incluem a cor da seleção do mapa, sons do mapa e especificações de iluminação (visualizações em 3D).

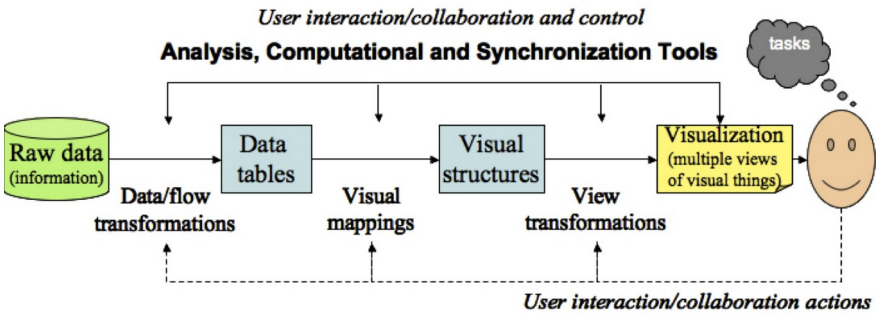


Figura 1: Processo de Visualization Pipeline

Fonte: Ward, Grinstein e Keim (2010).

3. Plataforma Android e Implementação de um Webapp

O Android é um sistema operacional para dispositivos móveis que foi baseado em uma versão modificada do kernel do Linux e outros projetos open source. O sistema foi desenvolvido pela Android Inc para máquinas digitais; entretanto, foi visto que era um mercado muito restrito e mudaram o foco para os smartphones. A empresa Android Inc. foi adqurida pela Google e em 2007 foi lançada a primeira versão, o Android 1.5, chamada Cupcake.

O Android foi desenvolvido baseado no linux e ele foi projetado como uma pilha de software, na qual primeiramente tem-se o kernel do linux (núcleo), responsável por gerenciar o hardware do dispositivo. Em seguida, a Hardware Abstraction Layer (HAL) fornece abstrações pela Application Programming Interface (API) Java sejam acessados os componentes de hardware, tais como, câmera, bluetooth, acelêrometro, dentre outros. Na sequência, a camada Android Runtime (ART) provê recursos para a compilação “Ahead-of-Time” (AOT) e “Just in Time” (JIT), que permitem que a compilação de um aplicação que já foi compilada anteriormente seja mais rápida. As bibliotecas C/C++ nativas permitem criar programas utilizando estas linguagens. Na camada superior está a estrutura JAVA API que são interfaces que simplificam e reutilizam componentes e serviços de sistemas modulares. E por fim na camada mais alta estão as aplicações que são criadas para a plataforma Android. (Arquitetura da Plataforma; 2018)

A criação das visualizações no Android foram utilizadas as tecnologias para web e para conseguir rodá-las no Android foi utilizada Webviews, que é uma View (Componente de Interface de Usuário) que exibe páginas web. Assim, é utilizada

uma Webview para importar e executar a biblioteca e o código web que utiliza uma Webkit para renderizar a página web e ela se comunica diretamente com a camada Java API. Dessa maneira, pode-se dizer que a aplicação criada utilizando Webview é um webapp, pois utiliza tecnologias para web e recursos nativos do Android e sua arquitetura é dividida em componentes que se comunicam. Um dos componentes contém o HTML, CSS, Javascript e os arquivos, e essa camada se comunica com a Webkit que renderiza o HTML e por sua vez se comunica com os recursos nativos do Android, como pode-se ver na Figura 2.

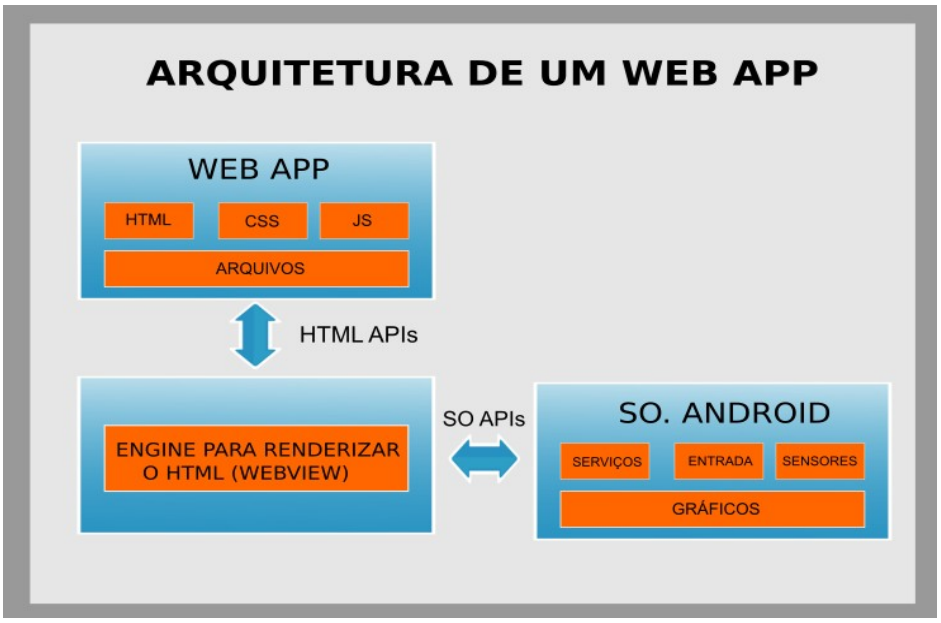
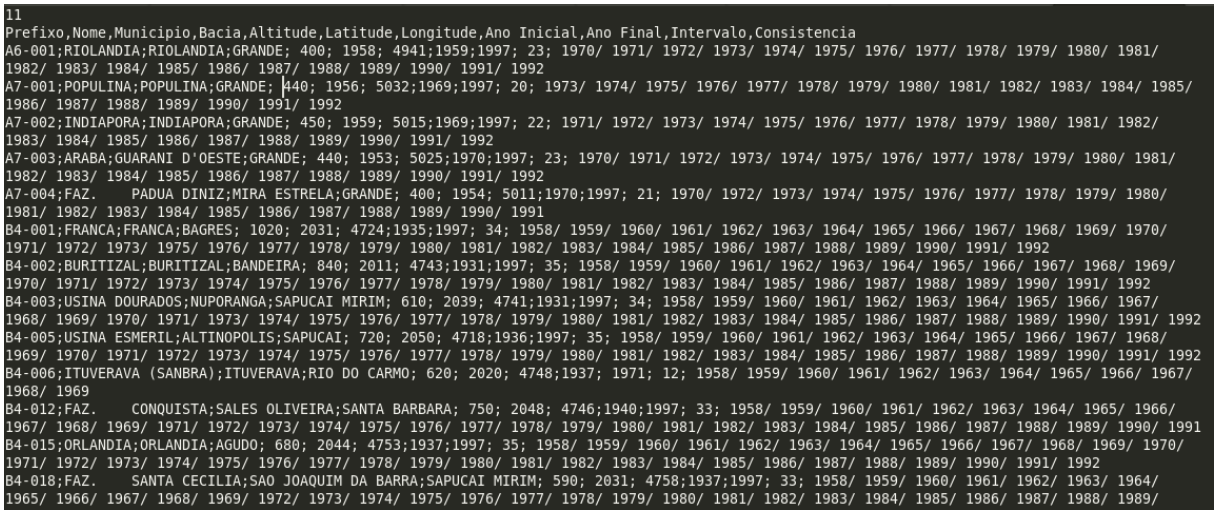


Figura 2: Arquitetura de um Web App.

Fonte: Autor

Além disso os arquivos que contém os dados do índice pluviométrico do estado de São Paulo inicialmente foram armazenados em diversos arquivos textuais. Os dados estavam separados por vírgula/ponto e vírgula, mas foi verificado que seria ineficiente utilizar esses arquivos textuais diretamente no dispositivo móvel já que existem uma grande quantidade de dados e os dispositivos móveis possuem limitações de armazenamento e processamento (Weng et al., 2012) e seria custoso computacionalmente para lidar diretamente com os arquivos. Tendo isso em mente uma solução encontrada foi armazenar esses dados em um servidor de banco de dados (Mariadb) e o dispositivo móvel fazer a requisição desses dados remotamente

conforme a necessidade, buscando somente aquilo que ele necessitasse ao longo do uso, não necessitando ter todos os dados baixados no dispositivo o que deixou a aplicação mais eficiente. Um exemplo dos dados armazenado em arquivo textual como pode-se ver na Figura 3.



```
11
Prefixo,Nome,Municipio,Bacia,Altitude,Latitude,Longitude,Ano Inicial,Ano Final,Intervalo,Consistencia
A6-001;RIOLANDIA;RIOLANDIA;GRANDE; 400; 1958; 4941;1959;1997; 23; 1970/ 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/
1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
A7-001;POPULINA;POPULINA;GRANDE; 440; 1956; 5032;1969;1997; 20; 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/
1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
A7-002;INDIAPORA;INDIAPORA;GRANDE; 450; 1959; 5015;1969;1997; 22; 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/
1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
A7-003;ARABA;GUARANI D'OESTE;GRANDE; 440; 1953; 5025;1970;1997; 23; 1970/ 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/
1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
A7-004;FAZ.    PADUA DINIZ;MIRA ESTRELA;GRANDE; 400; 1954; 5011;1970;1997; 21; 1970/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/
1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991
B4-001;FRANCA;FRANCA;BAGRES; 1020; 2031; 4724;1935;1997; 34; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/ 1968/ 1969/ 1970/
1971/ 1972/ 1973/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
B4-002;BURITIZAL;BURITIZAL;BANDEIRA; 840; 2011; 4743;1931;1997; 35; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/ 1968/ 1969/
1970/ 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
B4-003;USINA DOURADOS;NUPORANGA;SAPUCAI MIRIM; 610; 2039; 4741;1931;1997; 34; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/
1968/ 1969/ 1970/ 1971/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
B4-005;USINA ESMERIL;ALTINOPOLIS;SAPUCAI; 720; 2050; 4718;1936;1997; 35; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/ 1968/
1969/ 1970/ 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
B4-006;ITUVERAVA (SANBRA);ITUVERAVA;RIO DO CARMO; 620; 2020; 4748;1937; 1971; 12; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/
1968/ 1969
B4-012;FAZ.    CONQUISTA;SALES OLIVEIRA;SANTA BARBARA; 750; 2048; 4746;1940;1997; 33; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/
1967/ 1968/ 1969/ 1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991
B4-015;ORLANDIA;ORLANDIA;AGUDO; 680; 2044; 4753;1937;1997; 35; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/ 1965/ 1966/ 1967/ 1968/ 1969/ 1970/
1971/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/ 1990/ 1991/ 1992
B4-018;FAZ.    SANTA CECILIA;SAO JOAQUIM DA BARRA;SAPUCAI MIRIM; 590; 2031; 4758;1937;1997; 33; 1958/ 1959/ 1960/ 1961/ 1962/ 1963/ 1964/
1965/ 1966/ 1967/ 1968/ 1969/ 1972/ 1973/ 1974/ 1975/ 1976/ 1977/ 1978/ 1979/ 1980/ 1981/ 1982/ 1983/ 1984/ 1985/ 1986/ 1987/ 1988/ 1989/
```

Figura 3: Arquivo textual com os dados

Fonte: Autor

4. Arquitetura e Implementação

O principal propósito deste estudo foi desenvolver um Webapp para a plataforma Android para explorar técnicas de visualizações utilizando como dados os índices pluviométricos do Estado de São Paulo. Foi desenvolvido, também, uma versão exclusivamente web do mesmo webapp para comparar as suas limitações e diferenças com a versão mobile.

As visualizações foram construídas utilizando tecnologias para web e as bibliotecas para visualização D3 e NVD3. Dessa maneira, a versão mobile chamada de MobiViTool é um aplicativo Android desenvolvido em Java e XML e para o back-end existe um banco de dados MariaDB que faz conexão remota com o aplicativo. Visto que o aplicativo mobile é muito diferente de uma aplicação desktop e que possui limitações e uma interface de usuário diferente (Cobas; Iglesias; Seoane; 2015), foi criada também uma versão Web, chamada WebViTool, que foi desenvolvida utilizando a linguagem de programação PHP, o servidor Web de

software livre Apache, a biblioteca Javascript JQuery e para gerar as visualizações o D3 e o NVD3, além de HTML, CSS, e Javascript. A seguir, na Tabela 1, é exibido o ambiente de desenvolvimento utilizado nesse estudo para o MobiViTool e WebViTool.

Tabela 1. Estrutura do ambiente de desenvolvimento

	Ambiente de Desenvolvimento	**Plataforma**
Lado Servidor	Sistema Operacional	Fedora 27
	Servidor de Banco de Dados	MariaDB (10.1.29)
	Web Server	Apache (2.4.29)
Lado Cliente	Biblioteca para o ambiente	jQuery (1.9.1)
	Bibliotecas para os gráficos	D3 (v3 e v4), NVD3 (1.1.15)
	Sistema Operacional	Android 6.0 Marshmallow

4.1. Funcionamento da Arquitetura

O MobiViTool é um webapp que utiliza um servidor de banco de dados que armazena os dados de pluviometria. Primeiramente, o usuário seleciona através do aplicativo o local dos postos e o período para ser analisado e após isso o aplicativo faz uma conexão remota com o banco de dados, que retorna as médias dos dados escolhidos. Em seguida, o aplicativo cria um arquivo texto no formato CSV ou TSV, formatado para uso no aplicativo, a webview é gerada e lê esses dados armazenados em arquivos textos e com as bibliotecas de visualização (D3, NVD3) e código Javascript são geradas as visualizações. Já no caso do WebViTool existe um servidor Apache e um banco de dados, o usuário se conecta a esse servidor através de um navegador, os dados são processados no lado servidor, o processamento utiliza a linguagem PHP no lado servidor e após o processamento é gerado a visualização no dispositivo móvel. Diferentemente do MobiViTool o processamento do WebViTool é feito totalmente no lado servidor e apenas exibido no lado cliente como pode-ser visto no diagrama da Figura 4 ele possui um elemento a menos na arquitetura se comparado ao MobiViTool como pode-ser visto na Figura 5.

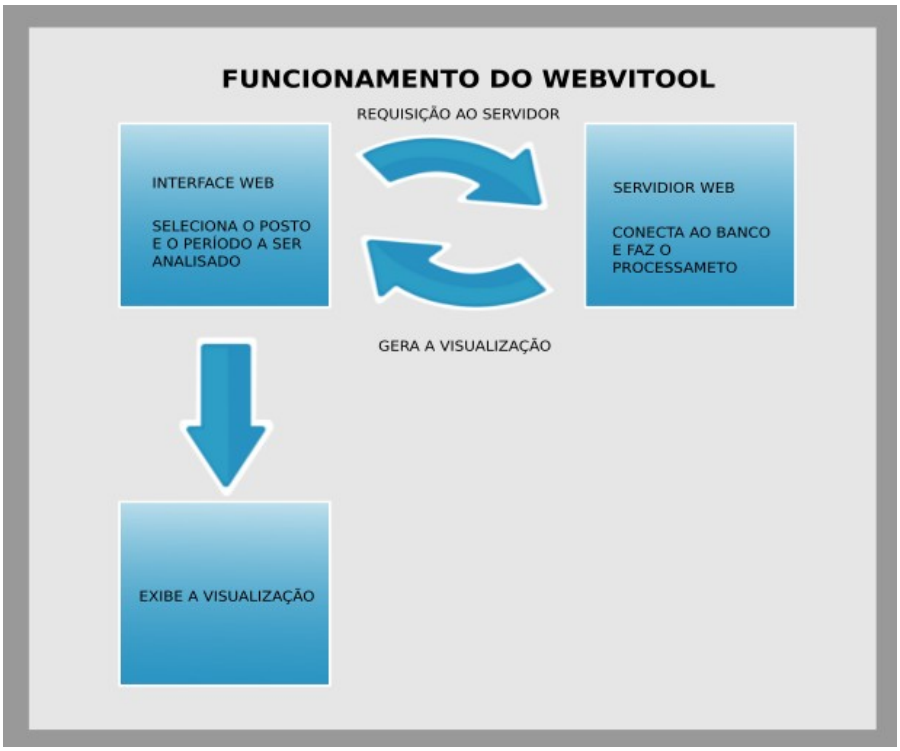


Figura 4: Arquitetura de Funcionamento do Webvitool

Fonte: Autor

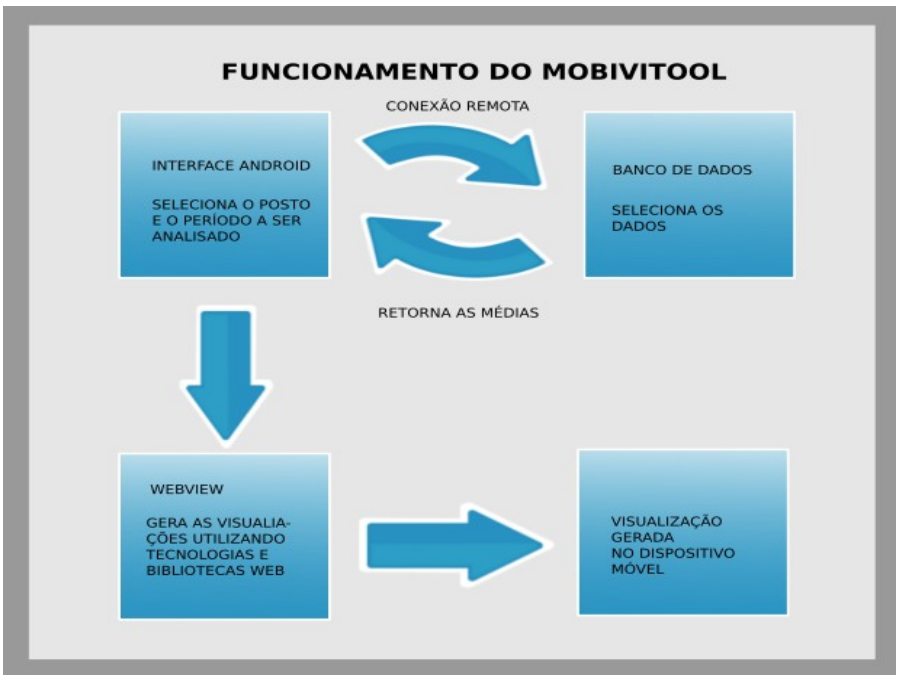


Figura 5: Arquitetura de uma Webview em uma aplicação Android

Fonte: Autor

5. Resultados e Discussão

O estudo das diferentes técnicas de visualização para dispositivos móveis envolveu avaliar o design e a implementação. É apresentada uma discussão sobre a escolha das técnicas e as limitações encontradas ao trabalhar com visualizações em dispositivos móveis, bem como a comparação entre a versão mobile (MobiViTool) com sua versão web (WebViTool), com acesso pelo smartphone para ambos os casos.

Inicialmente para fazer a comparação entre as duas versões foi pensado criar uma aplicação híbrida para a versão web utilizando frameworks como Apache Cordova ou Xamarin, entretanto verificou-se que não haveria uma grande diferença entre as duas aplicações, pois as duas utilizariam a mesma arquitetura e bibliotecas web para gerar as visualizações, a única diferença que a interface das telas seria feita com tecnologias para web no caso do Cordova ou Xamarin, e a outra com Java e Xml. Pensando nisso foi pensado construir a versão web totalmente como uma aplicação no lado servidor pois sua comparação faria mais sentido já que a execução da aplicação é feita no lado servidor, sendo o smartphone um elemento de exibição.

Uma diretriz para a escolha das técnicas de visualização foi considerar as limitações que os smartphones possuíam, além das, que seriam ideais para representar os dados pluviométricos do estado de São Paulo. Com esta definição, primeiramente, foi escolhida uma técnica “Bar Chart” por ela ser simples, fácil de entender, por exemplo, ordenado com base na média mensal de pluviometria, tornando o entendimento dos dados mais intuitivo, permitindo um destaque, em cor diferente e valor numérico por um tooltip, ao click sobre um determinado mês. Veja Figura 6. Estendendo esta visualização, foi construída outra visualização baseada em bar chart para analisar mais de uma média por estação simultaneamente, veja Figura 7.

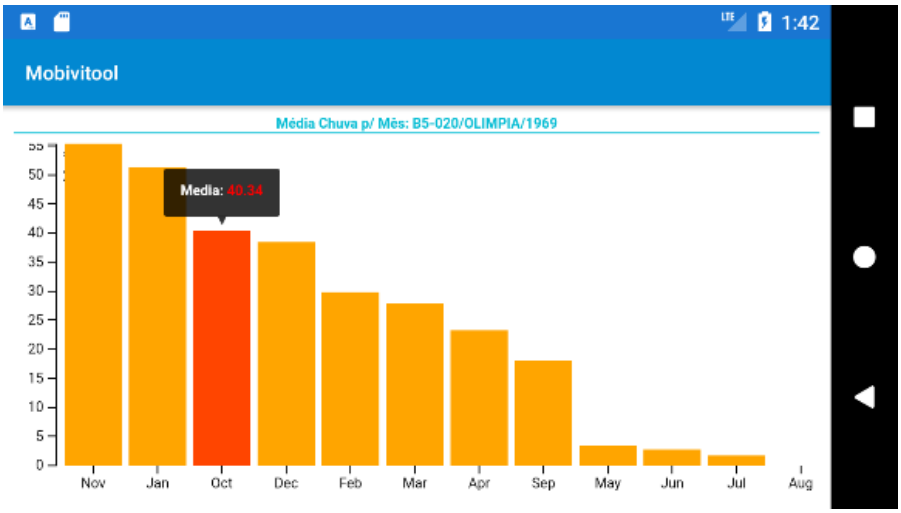


Figura 6: Visualização no Mobivitool Simple chart utilizando a técnica “Bar Chart” organizada descrescente pela média do volume pluviométrico de cada mês

Fonte: Autor

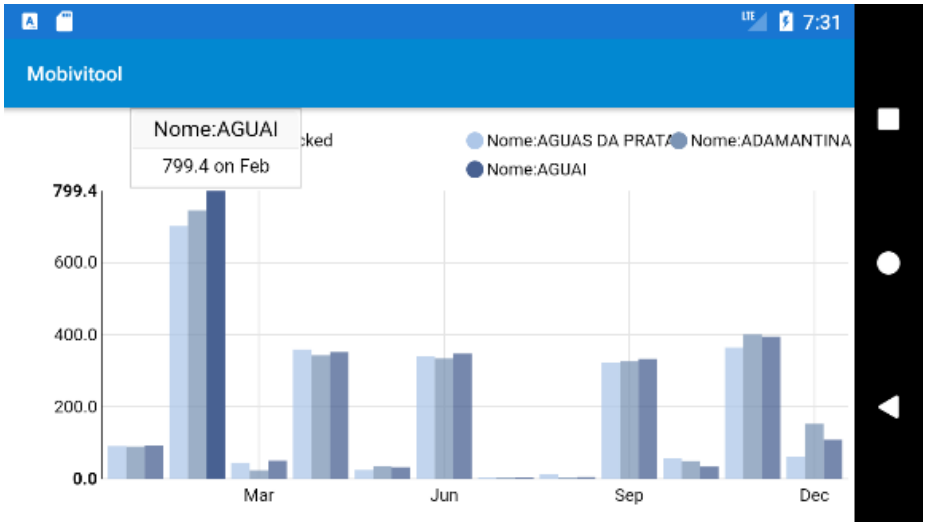


Figura 7: Visualização no Mobivitool baseada na técnica bar chart, mútiplas barras

Fonte: Autor

Utilizando a técnica “Brush and Zoom”, que permite ao usuário selecionar uma região de interesse para ampliar, concentrando-se em uma região específica, o ideal para a pequena tela dos dispositivos móveis. Assim, uma visualização que tem o tamanho reduzido, pela interação com o usuário pode ser usada para representar uma grande quantidade de dados em um espaço pequeno. Essa técnica é adequada para separar os dados em séries de tempo e, nesse estudo, conseguiu-se utilizar 10 anos de dados de índice pluviométrico em uma única visualização sem que o uso e a visualização fossem prejudicadas como pode-se ver na Figura 8.

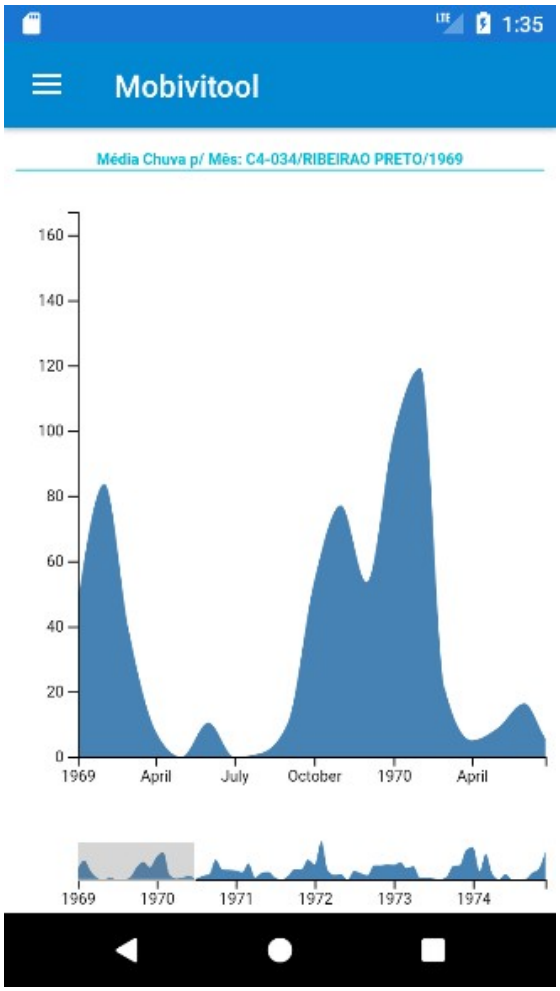


Figura 8: Visualização no Mobivitool baseada na técnica “Brush and Zoom”

Fonte: Autor

Outra técnica escolhida foi uma baseada em localização geoespacial para dividir os dados de cada posto baseado em sua localização no mapa do estado de São Paulo. A visualização das informações de cada posto foi realizada utilizando o zoom para ampliar a região de interesse e um tooltip para mostrar a informação do posto. A seguir na Figura 9, Figura 10, vê-se as visualizações no aplicativo Mobivitool e uma visualização no Webvitool respectivamente.

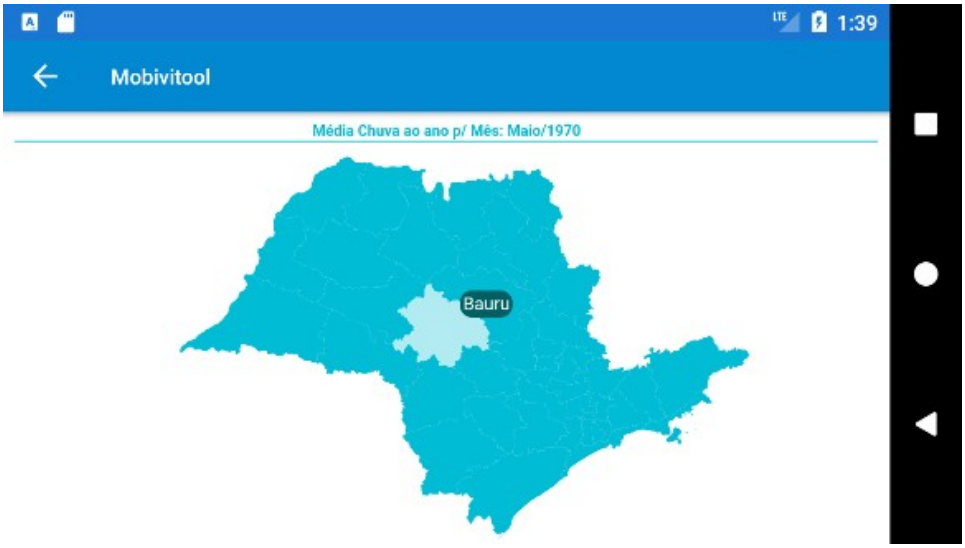


Figura 9: Visualização do Mobivitool baseada na técnica geoespacial no mapa do Estado de São Paulo

Fonte: Autor

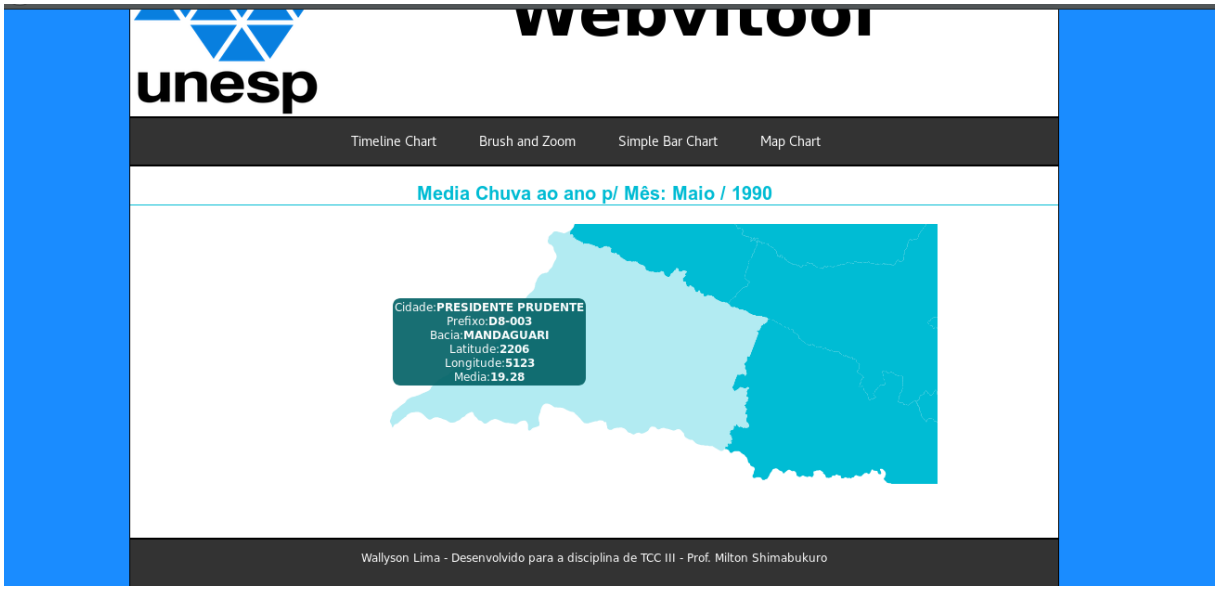


Figura 10: Visualização do Webvitool baseada na técnica geoespacial com o zoom na região de interesse

Fonte: Autor

A seguir, é descrito o processo adotado para o desenvolvimento e apresentadas as dificuldades percebidas. Primeiramente, foi desenvolvido o MobiViTool e foram encontrados obstáculos para fazer rodar as bibliotecas de visualização bem como as tecnologias para web através de uma Webview no Android. As bibliotecas D3 e NVD3 não estavam rodando localmente, foi necessário utilizá-las através de links, requerendo da aplicação baixá-las via internet. As visualizações foram feitas e testadas no navegador Web, e assim que elas eram concluídas, eram adaptadas para o Android. Esta estratégia, foi adotada pela facilidade em testar e encontrar erros utilizando as ferramentas do navegador; tarefa não trivial no Android.

Assim que o MobiViTool foi concluído, foi iniciado o desenvolvimento da versão com somente tecnologias para web, utilizando basicamente, linguagem de programação PHP. O desenvolvimento foi muito mais rápido, já que as tecnologias para construir as visualizações foram criadas para serem utilizadas para web; a única dificuldade encontrada foi portar o código Java que foi utilizado para construir as interfaces para o PHP.

A diferença na interface entre uma aplicação Android e web/desktop – touch e mouse/teclado, respectivamente – é um fator de impacto no desenvolvimento. Na versão Web, algumas funcionalidades como o uso de tooltip quando o usuário passa o mouse (mouseover) em uma região de interesse, como por exemplo, uma barra na técnica bar chart ou a cidade na técnica geoespacial, não foi possível replicar no Android.

O desempenho das duas aplicações em dispositivo móvel também foi analisado. Para os testes, foram utilizados um smartphone virtual utilizado na própria IDE Android Studio, configuração 2 núcleos de processamento, 1 GB de memória e tela de 720 x 1280, representando smartphones de entrada/intermediários na época de realização dos testes. O outro dispositivo utilizado foi um smartphone físico: um Motorola Moto E 2º Geração, com 4 núcleos, 1 GB de memória e tela 960 x 540

representando um smartphone de entrada com baixo poder de processamento. Para realizar os testes foram utilizados as ferramentas de análise de desempenho do IDE Android Studio, analisando a memória utilizada, a porcentagem de uso da CPU e a quantidade de threads utilizadas. O teste considerou o tempo entre a escolha da visualização até a carga na webview.

O primeiro teste realizado foi da visulização bar chart com múltiplas barras como na Figura 7. No teste realizado com o smartphone virtual para carregar a visualização ela exigiu um uso que variou entre 45% - 60% do CPU em média e deu um pico de 68%, variou de 60 MB – 90 MB em média a memória e teve um pico de 92 MB, ela utilizou 52 threads. Já a mesma visualização rodando no smartphone real, precisou de uma variação de 10%-30% do CPU em média e deu um pico de 41%, variou de 20 MB-60 MB em média e um pico de 80 MB de memória utilizando 51 threads. Outra váriavel analisada foi o tempo para carregar a visualização o smartphone virtual demorou 58 segundos aproximadamente, enquanto que o smartphone físico utilizou 53 segundos como vemos na Tabela 2. Os resultados do smartphone virtual e físico guardam uma coerência, indicando que testes com um smartphone virtual corretamente configurado fornecem métricas para a avaliação de desempenho.

Tabela 2. Teste visualização bar chart com múltiplas barras no MobiViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Virtual	45-60	68	60-90	92	52	58
Smartphone Físico	10-30	41	20-60	80	51	53

A visualização com a técnica “Brush and Zoom” foi utilizada para exibir informações de um período de até 10 anos. No smartphone físico variou de 5%-35% de uso da CPU com pico de 40%, variou de 35 MB-80 MB de memória com pico de 87 MB e utilizou 51 threads. No quesito tempo o smartphone físico levou 10,68 segundos aproximadamente. Esta técnica foi construída utilizando a biblioteca D3, enquanto a anterior foi utilizada a biblioteca NVD3, mesmo que a técnica “Brush and

Zoom” exigisse um maior uso de memória devido a maior quantidade de dados, ela foi ligeiramente mais econômica no uso da CPU, isso se deve que a NVD3 é uma biblioteca construída em cima da D3, ela exige mais do CPU, o uso da memória é maior na “Brush and Zoom” o que era esperado, utiliza uma maior quantidade de dados como podemos ver na tabela 3.

Tabela 3. Teste visualização “Brush and Zoom” no MobiViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Físico	5-35	40	35-80	87	51	10,68

Outra técnica analisada foi a técnica geoespacial, que exige um alto poder de processamento devido ao fato de carregar dados de todos os postos somado ao fato que a visualização tem que carregar o mapa do estado de São Paulo que exige bastante o uso do CPU. No smartphone físico ele variou de 20%-70% do uso da CPU com pico de 78%, e utilizou de 60 MB-100 MB de memória com pico de 100 MB e utilizou 53 threads. Em relação ao tempo levou 72 segundos para carregar no smartphone físico aproximadamente, vê-se aqui que ele demorou mais para carregar que as outras visualizações, muito disso se deve ao fato da limitação do smartphone físico em carregar o mapa do estado de São Paulo, que possue muitas coordenadas cartesianas o que exige um poder de processamento alto.

Tabela 4. Teste visualização geoespacial no MobiViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Físico	20-70	78	60-100	100	53	72

Ao analisar as três técnicas percebeu-se que as visualizações tiveram desempenhos diferentes. A visualização que carregou mais rápido em tempo foi a “Brush and Zoom”, carregou em apenas 10,68 segundos mesmo ela utilizando mais uso da CPU e memória que a técnica baseada em bar chart. Isso se deve ao fato da bar chart utilizar a biblioteca NVD3, que é uma tecnologia desenvolvida em cima da D3 que consome menos recursos. A que utilizou mais recursos como memória e CPU foi a geoespacial devido ser necessário carregar as coordenadas cartesianas e gerar o mapa de São Paulo.

Posteriormente foi testado o desempenho do WebViTool, é uma aplicação Web que roda diretamente no lado servidor, então por isso para analisar o seu desempenho no Android foi preciso criar uma Webview que chamasse o endereço e então gera-se a aplicação dentro da Webview no Android, eu precisei utilizar as ferramentas de análise de desempenho da IDE Android Studio. Foram testados as mesmas visualizações.

O primeiro teste também foi a técnica bar chart, no smartphone físico variou de 5%-55% do uso de CPU com pico de 57%, ele utilizou de 73 MB – 90 MB de memória com pico de 94 MB e 47 threads. Quanto ao tempo gasto para gerar a visualização levou apenas 4 segundos como vê-se na Tabela 5.

Tabela 5. Teste visualização bar chart no WebViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Físico	5-55	57	73-90	94	47	4

A visualização “brush and zoom” teve o seguinte desempenho no smartphone físico variou entre 5%-40% de uso da CPU com pico de 40%, e a memória foi de 50 MB-80 MB de memória com pico de 81 MB e 49 threads. Na medida de tempo levou 2 segundos com vê-se na Tabela 6.

Tabela 6. Teste visualização “Brush and Zoom” no WebViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Físico	5-40	40	50-80	81	49	2

E por fim foi visto a técnica de geoespacial no webvitool, no smartphone físico variou de 5%-70% de uso da CPU com pico de 79%, e o uso de memória foi de 75 MB-100 MB com pico de 101 MB e 49 threads. Quanto ao tempo para carregar a visualização levou 14 segundos com vê-se na Tabela 7.

Tabela 7. Teste visualização geoespacial no WebViTool

	CPU %	Pico CPU %	Memória (MB)	Pico Memória (MB)	Qtde Threads	Tempo (segundos)
Smartphone Físico	5-70	79	75-100	101	49	14

Ao analisar os testes do webvitool ele obedeceu o mesmo padrão de processamento que tinha sido feito no mobivitool com as visualizações que foram executadas no smartphone físico. Agora comparando o desempenho das duas aplicações (MobiViTool x WebViTool), vemos que o WebViTool carregou as visualizações em um menor tempo, utilizou uma porcentagem inferior da CPU, mas no quesito memória se manteve estável com pequenas variações. A aplicação WebViTool pode ser considerada como mais adequada para a implementação, considerando que ambas fazem acesso a internet. E ficou claro como no caso da visualização geoespacial que demorou muito tempo para carregar a visualização, o que seria necessário uma otimização.

6. Conclusão

Relembrando alguns pontos. Houve uma certa dificuldade para criar visualizações para dispositivos móveis uma vez que foram utilizadas tecnologias para Web, as quais não rodam nativamente no Android, precisando utilizar o recurso de Webview. Ainda, uma vez que as visualizações eram criadas e validadas no navegador, elas precisavam ser adaptadas para os dispositivos móveis. Percebeu-se que alguns recursos que rodavam na versão web não foram possíveis replicar na versão mobile, pois haviam diferenças no uso da interface entre o dispositivo móvel e os computadores tradicionais (Interface touch x teclado e mouse), além das limitações dos dispositivos móveis como pequena tela, limitado poder de processamento e memória. (Weng et al., 2012)

Ao realizar os testes ficou claro que as visualizações rodaram melhor em um smartphone real do que no smartphone virtual, mas os testes neste último pode ser um bom indicador de desempenho e consumo de recursos. Ao analisar o desempenho das duas aplicações MobiViTool e WebViTool pôde-se ver que a aplicação rodando totalmente na Web no lado servidor possibilitou um grande ganho de desempenho, mesmo considerando que existe um gap, já que os dados são processados no lado servidor e apenas visualizados no smartphone, o WebViTool se mostrou vantajoso.

Como trabalhos futuros, sugere-se implementar visualizações mais complexas, como em 3D, construir uma versão utilizando programação nativa para analisar o desempenho e testar formas de interação com a interface touch.

Referências

Ward, M., Grinstein, G., Keim, D.. **Interactive Data Visualization: Foundations, Techniques, and Applications.** 1º Edição. Boca Raton: CRC Press, 2010. 507pg.

Kerren, A., Kucher, K., Li, Y., Schreiber, F., **BioVis Explorer: A visual guide for biological data visualization techniques.** Plos One, 2017.

Cobas, C., Iglesias, I., Seonane, F., **NMR data visualization, processing, and analysis on mobile devices.** Wiley Online Library, 2015.

Neugebauer, T., Bordeleau, E., Burrus, V., Brzezinski, R.. **DNA Data Visualization (DDV): Software for Generating Web-Based Interfaces Supporting Navigation and Analysis of DNA Sequence Data of Entire Genomes.** Plos One, 2015.

Weng, Y., Sun, F., Grigsby, J., **GeoTools: An android phone application in geology.** Comput. Geosci, 2012.

[1]- WebView. Disponível em: <https://developer.android.com/reference/android/webkit/WebView.html>. Acesso em: 20 de maio. 2018.

[2]- D3 Data-Driven Documents. Disponível em: <https://d3js.org/>. Acesso em: 20 de maio. 2018.

[3]- D3 Brush. Disponível em: <https://github.com/d3/d3-brush/>. Acesso em: 20 de maio. 2018.

[4]- D3 Zoom. Disponível em: <https://github.com/d3/d3-zoom/>. Acesso em: 20 de maio. 2018.

[5]- Arquitetura da Plataforma. Disponível em: <https://developer.android.com/guide/platform/index.html?hl=pt-br#api-framework>. Acesso em: 01 de junho. 2018.

[6]- Building Web Apps in WebView. Disponível em: <https://developer.android.com/guide/webapps/webview.html>. Acesso em: 01 de junho. 2018.