UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

Dezembro 2012
ISSN 1413-9014
E-ISSN 2317-2193

CADERNOS DO IME
Série Informática
Volume 34

Especial de Projetos Finais

CADERNOS DO IME
Série Informática
Volume 34

Publicação do Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro. Periodicidade semestral, com circulação em junho e dezembro.

Normalização, divulgação e distribuição:
Biblioteca do Centro de Ciências de Tecnologia A (CTC/A) da rede Sirius de Biblioteca da UERJ – ctca@uerj.br

Organização e Edição do volume 34:
Alexandre Sztajnberg
Fabiano de Souza Oliveira

Correspondência:
Universidade do Estado do Rio de Janeiro
Instituto de Matemática e Estatística
Editores do Cadernos do IME - Série Informática
Rua São Francisco Xavier, 524 - Pavilhão Reitor João Lyra Filho,
6º andar, sala 6019 B
Maracanã - 20550-900 – Rio de Janeiro, RJ

Telefax: +55 21 2334-0144
e-mail: cadernos_inf @ime.uerj.br
site: http://www.ime.uerj.br/cadernos/cadinf/

Produção e editoração gráfica:
Alexandre Sztajnberg
Fabiano de Souza Oliveira

Organizadores:
Alexandre Sztajnberg
Fabiano de Souza Oliveira

Contato: **cadernos_inf@ime.uej.br**

Dezembro 2012
ISSN 1413-9014
E-ISSN 2317-2193

CADERNOS DO IME
Série Informática
Volume 34

Especial de Projetos Finais

Nota dos Editores

Este volume foi formatado como uma Edição Especial dedicada a publicação de resultados de trabalhos de fim de curso, Projeto Final, do Curso de Bacharelado em Ciência da Computação do IME/UERJ. Os trabalhos submetidos foram revisados pelos autores, e o conteúdo aprimorado e reestruturado para apresnetar a qualiadde esperado para um artigo. Os oriendadores e a banca de defesa de cada projeto fizeram um primeiro filtro. Além disso, cada trabalho submetido passou pelo processo de avaliação e revisão, com no mínimo duas avaliações. Com isso os alunos-co-autores ainda tirveam a oportunidade pedagógica de participar do processo de submissão/críticas/melhora dos artigos.

O Volume 34 ainda está sendo publicado com atraso, devido às atividades de migração do Casdernos do IME – Série Informática para o portal SEER da UERJ.

Reiteramos os agradecimentos aos revisores internos e externos, que fizeram suas avaliações em tempo. Agradecemos também a apcieência dos autores pela demora no processo.

Novembro de 2013

Cadernos do IME – Série Informática
ISSN 1413-9014
E-ISSN 2317-2193

Apresentação

O Volume 34 dos Cadernos do IME – Série Informática apresenta três artigos, oriundos de Projetos Finais selecionados.

O primeiro artigo, *Visualização da manipulação de dados em Point Quadtrees*, de Marcos P. G. de Lima, Guilherme L. A. Mota e Paulo E. D. Pinto, apresenta a implementação de uma ferramenta didática para o estudo da estrutura de dados Point Quadtree, um tipo específico de Quadtree. A ferramenta apresenta graficamente as manipulações de dados em Point Quadtrees, permitindo a visualização do que ocorre na inserção, exclusão ou busca de pontos nesta estrutura

O segundo artigo, *HHS Sticker: Aplicação Médica em Java para TV Digital*, de Claudia O. Neves, Thales V. G. da Silva, Roberto C. R. Machado, Alexandre Sztajnberg, está inserido no projeto SCIADS (tempo.ic.uff.br/sciads). O HHS Stiker é uma aplicação para a transmissão de dados fisiológicos e alerta de pacientes em suas residências, desenvolvido e embarcado num setop-box de TV Digital utilizando Ginga-Java. O artigo discute o desenvolvimento e a utilização da aplicação.

O último artigo, *Avaliação e Redução do Tempo de Resposta de Sistemas Web* de Victor Marconi Arouca Ribeiro, Valdenir Tavares, Alexandre Sztajnberg, apresenta uma avaliação de desempenho variações no tempo de resposta de um sistema web padrão. Nos expermentos são variadas as especificações do hardware da máquina hospedeira, bem como aplicandas as recomendações de Steve Souders, ex-Yahoo, para a configuração dos vários componentes de software da infraestrutura.

Renovamos o convite aos pesquisadores para enviarem suas contribuições para os próximos números dos Cadernos do IME – Série Informática, para mantermos o mesmo padrão de qualidade obtido na presente edição.

Novembro de 2013

Alexandre Sztajnberg e Fabiano de Souza Oliveira
Organizadores do Volume 34

cadernos_inf@ime.uerj.br

Visualização da manipulação de dados em Point Quadtrees

Marcos P. G. de Lima[1], Guilherme L. A. Mota[1], Paulo E. D. Pinto[1]

[1]Instituto de Matemática e Estatística
Universidade do Estado do Rio de Janeiro (UERJ)
Rua S. Francisco Xavier, 524 - Maracanã - Rio de Janeiro - RJ - CEP 20.550-013

{marcosgaldino, guimota, pauloedp}@ime.uerj.br

Abstract. *Quadtrees are data structures used in image processing in Geographic Information Systems and as generators of meshes. This paper presents the implementation of a teaching tool for the study of Quadtree Point data structure, a specific type of Quadtree. We developed an application that graphically displays data manipulations in Point Quadtrees, allowing visualization of what happens at insertion, exclusion or seek of points in this structure.*

Resumo. *As Quadtrees são estruturas de dados utilizadas no processamento de imagens, em Sistemas de Informações Geográficas e como geradores de malhas. Este artigo apresenta a implementação de uma ferramenta didática para o estudo da estrutura de dados Point Quadtree, um tipo específico de Quadtree. Foi desenvolvido um aplicativo que apresenta graficamente as manipulações de dados em Point Quadtrees, permitindo a visualização do que ocorre ao inserirmos, excluirmos ou buscarmos pontos nesta estrutura.*

1. Introdução

A estrutura de dados Quadtree [Finkel e Bentley 1974] é uma árvore de dados em que cada nó possui exatamente quatro nós filhos, sendo uma adaptação da árvore binária para dados bidimensionais ordenados. É utilizada no processamento de imagens [Strobach 1991], como geradora de malhas [Miranda e Martha 2002], em Sistemas de Informação Geográfica (SIGs) [Samet, Rosenfeld, Shaeffer e Webber 1984], entre outras áreas. Em duas dimensões, uma Point Quadtree é uma Quadtree tal que a decomposição do espaço se dá a partir das coordenadas dos pontos na árvore.

As Point Quadtrees permitem representar dados expressos por pontos no espaço bidimensional, sendo comumente usada para lidar com dados geográficos, bem como aplicações que envolvem pesquisa. O estudo desta estrutura demanda a associação entre a inserção de dados em uma árvore quaternária com um correspondente particionamento do espaço, havendo operações complexas de manipulação de dados, o que torna o seu estudo não trivial.

Em nossa pesquisa, verificamos a inexistência de ferramentas para a visualização de árvores em geral, com o detalhamento das operações de inserção, busca e deleção, bem como a ausência de trabalhos específicos relacionados a estrutura point quadtree, seja da estrutura em si ou do ponto de vista do apoio à aprendizagem da mesma.

De acordo com [Gardner, Chen e Moran 2010], os conceitos importantes devem ser ensinados de diversas formas, que devem atuar ativando diferentes inteligências ou combinações de inteligências, valorizando as diferenças entre os indivíduos. Essa afirmação mostra que o desenvolvimento de ferramentas de apoio ao ensino com o uso de recursos gráficos pode ser visto como elemento propiciador da diversificação de abordagens para o estudo em questão, respeitando a individualidade dos estudantes.

Os argumentos acima e a relevância das áreas citadas, que utilizam a estrutura de dados Quadtree, somada com a complexidade dos conceitos associados a esta estrutura mostra o quanto é importante o desenvovimento de ferramentas que possam facilitar o seu processo de aprendizagem, criando condições para a realização de exercícios pedagógicos e também para o autoaprendizado.

1.1. Objetivo

O presente artigo teve como objetivo desenvolver uma ferramenta computacional para fins didáticos do uso da Point Quadtree utilizando a linguagem C++, a IDE Qt e a biblioteca Graphviz. O didatismo desta ferramenta requer que a aplicação seja dotada de uma interface gráfica visual que representa graficamente as alterações da estrutura da árvore e do particionamento do espaço geográfico, ao longo da execução dos algoritmos de manuseio da estrutura de dados.

1.2. Organização do Trabalho

Este trabalho está dividido da seguinte maneira. No Capítulo 2 é apresentada a fundamentação teórica da Point Quadtree. O Capítulo 3 descreve as ferramentas computacionais usadas no trabalho: Graphviz e Qt. O Capítulo 4 descreve a ferramenta, mostrando as classes, os requisitos satisfeitos pelo software e a sua interface gráfica. O Capítulo 5 contém os resultados obtidos com o uso do aplicativo e no Capítulo 6 são apresentadas as conclusões do trabalho.

2. Point Quadtree

De acordo com [Samet 2006] e [Bentley 1974], Quadtree é uma estrutura de dados que traduz uma estrutura de árvore em um grid. A maneira como essa tradução ocorre depende do tipo de Quadtree. De maneira geral, a cada inserção no subespaço que já contiver um nó, este será dividido em quatro regiões retangulares. O presente trabalho se dedica ao estudo de um tipo especial de Quadtree: as Point Quadtrees (PQ).

A estrutura de dados Point Quadtree pode ser compreendida como uma análoga bidimensional das árvores binárias de busca. Criadas por Bentley e Finkel [Bentley e Finkel 1974], Point Quadtrees são árvores bidimensionais em que cada nó, que pode possuir até quatro filhos, representa as coordenadas 2D de um ponto.

Cada nó pai da árvore divide a área (ou subárea) definida por seus ascendentes em quatro quadrantes – situados respectivamente a nordeste (NE), sudeste (SE), noroeste (NO) e sudoeste (SO) das coordenadas 2D deste nó. Por outro lado, os nós-folha não indicam subdivisão do espaço, havendo a garantia de que se um nó não teve seus quadrantes subdivididos seus filhos são nós-folha ou nulos. Desta forma, há no máximo um ponto por quadrante. A cada inserção de um nó, se o quadrante não dividido em que o nó inserido estiver localizado já contiver algum ponto o mesmo é

desmembrado. Os quatro novos quadrantes são definidos a partir das coordenadas do ponto já anteriormente contido, como pode ser visto na Figura 5.

2.1. Estrutura do nó das Point Quadtrees

Cada ponto bidimensional é representado na forma de um nó contendo oito campos. Esses campos guardam quatro ponteiros dos quatro nós filhos que subdividem os quadrantes NE, NO, SO e SE, as duas coordenadas do ponto, seu rótulo e um ponteiro para uma lista de colisão.

A lista de colisão permite a inclusão de vários pontos com as mesmas coordenadas, porém com rótulos distintos. No presente trabalho não foi implementada a lista de colisão. Usou-se o artifício de representar os diferentes rótulos no mesmo campo, separados por vírgula. A Figura 1 representa essa estrutura.

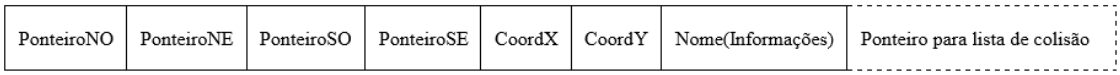


Figura 1. Representação da estrutura do nó de uma Point Quadtree.

2.2. Criação de uma Point Quadtree

O primeiro nó inserido é definido como raiz da árvore. Cada inserção subsequente é feita percorrendo a árvore, como em uma árvore de busca, até se chegar a um nó vazio ou a um nó com as mesmas coordenadas. Se esse nó não for vazio, o rótulo é modificado, agregando-se a nova descrição. Caso o nó seja vazio, é modificado com os novos dados e são criados quatro novos filhos, um para cada quadrante, todos nós vazios. O espaço é subdividido à medida que os nós deixam de ser nós folha. A ordem da inserção dos dados na PQ define a maneira como ocorrerá o particionamento do espaço.

Na inserção de pontos em linhas de divisão em quadrantes os limites inferiores e à esquerda de cada quadrante são fechados (fazem parte do quadrante) enquanto que os limites superiores e à direita de cada bloco estão abertos (não fazem parte do quadrante).

Tabela 1. Pontos com seus rótulos e suas coordenadas x e y.

Rótulo do Nó	Coordenada X	Coordenada Y
A	125	116
B	178	188
C	72	63
D	10	9
E	231	194
F	150	170
G	163	178
H	50	42

A Tabela 1 apresenta pontos com seus rótulos e suas coordenadas que exemplificarão a formação de uma Point Quadtree.

Na apresentação das árvores Point Quadtree, será adotada a seguinte convenção: nós pais serão representados por círculos brancos, nós folha por quadrados preenchidos

de preto, e nós nulos são representados por quadrados brancos. Na Figura 2(a) pode-se observar o particionamento do espaço de 250x250 para a PQ referente aos dados da Tabela 1 inseridos ordenadamente. A árvore correspondente é vista na Figura 2(b).

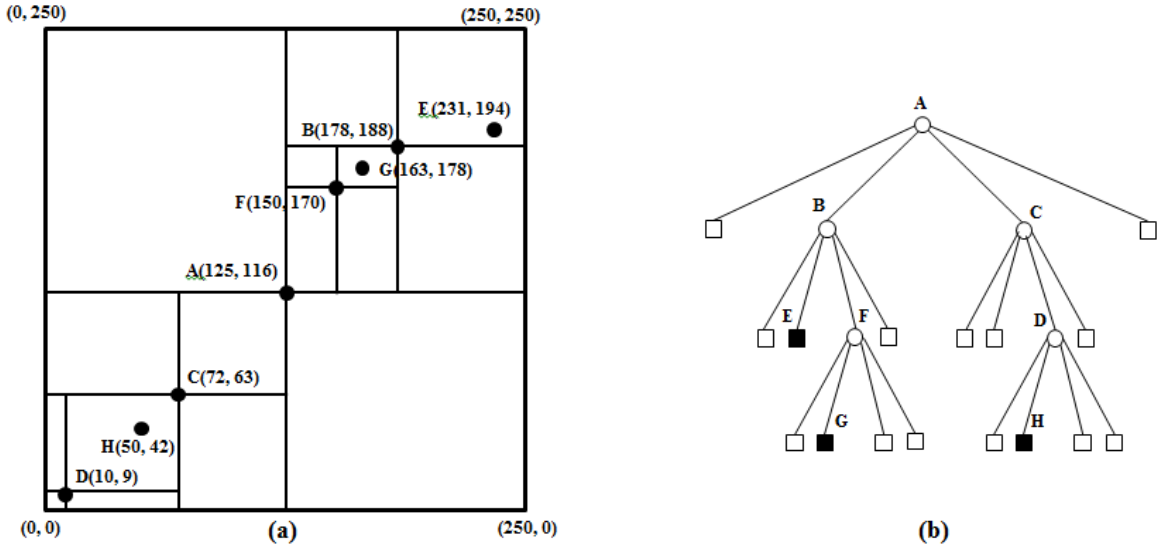


Figura 2. Formação de Point Quadtree a) Partição do Espaço b) Point Quadtree resultante.

2.3. Inserção em Point Quadtrees

Suponha que o espaço a ser dividido tenha 250 x 250 unidades. Na primeira inserção de um ponto, este espaço será o quadrante de destino do ponto.

A inserção do primeiro ponto (125, 116) não provoca a divisão do espaço. Este primeiro nó é um nó folha e este tipo de nó não divide quadrantes nas PQs. A Figura 3 ilustra esta inserção. Este nó, por ser o primeiro a ser inserido, torna-se a raiz da árvore.

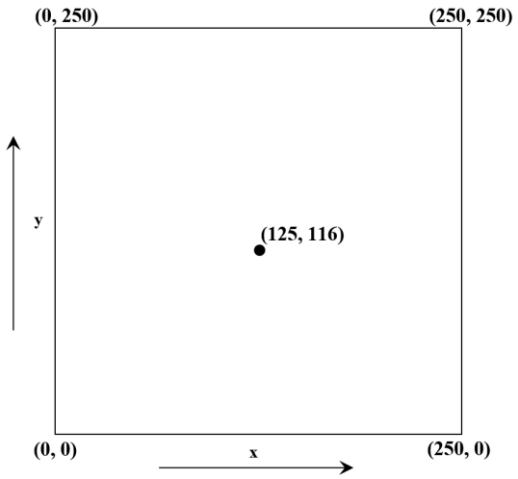


Figura 3. Inserção do ponto (45, 67) no espaço.

A representação em árvore correspondente é exibida na Figura 4.

Figura 4. Árvore com o nó raiz, (125, 116).

A divisão do espaço só ocorre a partir da inserção de um segundo ponto e se dá através da divisão do espaço não delimitado em quatro espaços resultantes a partir das coordenadas do ponto pai do nó a ser inserido. Assim, ao inserir o ponto (178, 188) a divisão se dá no ponto (125, 116), cujo resultado obtido é mostrado na Figura 5.

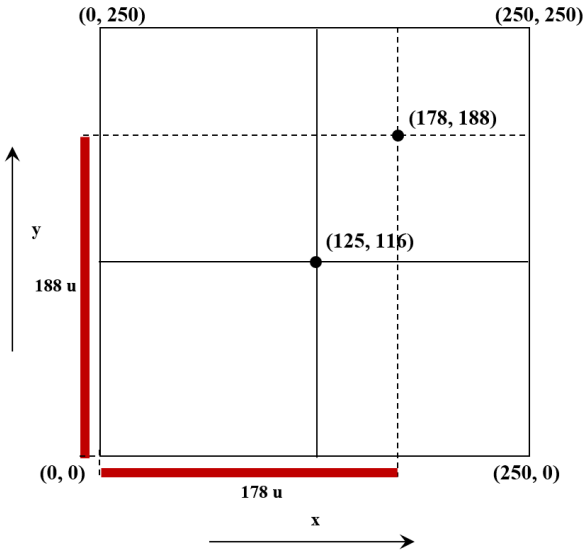


Figura 5. Particionamento após inserção do novo ponto, (178, 188), na PQ.

A visualização da árvore correspondente é mostrada na Figura 6.

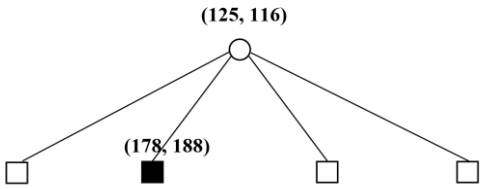


Figura 6. Árvore resultante da inserção de ponto, (178, 188), na PQ.

De acordo com [Samet 2006] e [Bentley 1975], a inserção, da mesma forma que a busca por um ponto, apresenta custo médio $O\left(\log_4^N\right)$, onde N é o número de pontos.

2.4. Exclusão em Point Quadtree

A exclusão de nós em Point Quadtrees pode ser feita de muitas maneiras. Uma delas, sugerida por Bentley e Finkel, consiste na reinserção de todos os nós da árvore que têm raiz no nó excluído. Embora essa solução seja bem simples, é um processo muito caro, a menos que o nó a ser excluído seja nó folha ou se seus filhos o são.

As próximas linhas descrevem um processo mais complexo e mais eficiente desenvolvido por [Samet 1990] para a exclusão em point quadtrees. O que se deseja é substituir um nó A com coordenadas (x_A, y_A) por um nó B, com coordenadas (x_B, y_B), com uso de retas auxiliares $x = x_A$, $x = x_B$, $y = y_A$ e $y = y_B$, de tal maneira que a união da região entre $x = x_A$ e $x = x_B$ com a região entre $y = y_A$ e $y = y_B$ esteja vazia. Essa situação é ilustrada na Figura 7.

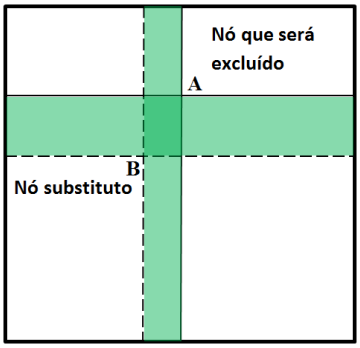


Figura 7. Região que favorece a exclusão, quando vazia (em verde).

Com essa condição satisfeita, teríamos apenas que substituir o nó A pelo nó B e a exclusão seria concluída. O problema é que encontrar um nó nessas condições envolve uma quantidade considerável de visitas a nós, podendo não haver a existência de tal nó. Para nós com apenas um filho ou nós folha a exclusão é feita de maneira bem simples, substituindo o nó excluído pelo filho no primeiro caso ou excluindo o nó, no segundo. Em caso contrário são determinados quatro candidatos para substituição, um para cada quadrante formado a partir do ponto a ser excluído. Após a escolha do melhor dos quatro candidatos, os quadrantes são reorganizados, finalizando o processo de exclusão do nó. A escolha dos candidatos à exclusão é feita percorrendo, em cada quadrante, a partir do nó filho do nó a ser excluído, no sentido oposto (a 180º) à posição do quadrante. Como resultado, encontramos o nó naquele quadrante que se encontra mais próximo do nó a ser excluído, como mostrado na Figura 8.

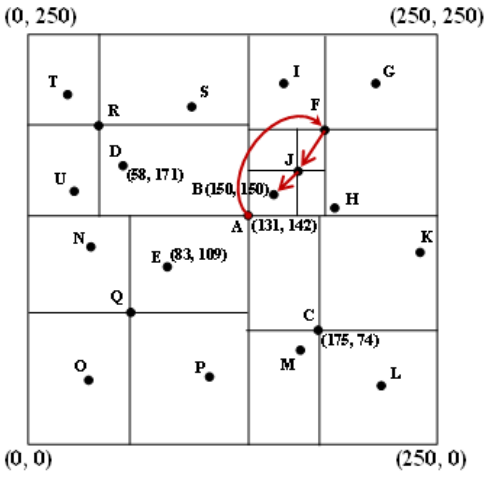


Figura 8. Eleição do melhor candidato para substituição do nó A (131, 142).

Uma vez que um candidato é encontrado para cada um dos quatro quadrantes é feito um teste para encontrar o "melhor" candidato, que é o candidato que provocaria

menos processamento no processo de reinserção de nós. Vale observar que não há garantia de que esse "melhor" candidato seja o candidato ótimo. Samet definiu dois critérios para escolha do candidato mais adequado:

Critério 1 - É escolhido o candidato que está mais próximo de cada um dos eixos (x ou y, a partir do nó raiz) do que outros candidatos do mesmo lado dos eixos.

Os eixos utilizados são as linhas de separação definidas a partir das coordenadas do ponto a ser excluído. O candidato no quadrante oposto não está do mesmo lado de nenhum dos eixos. Por isso, entre um nó e o seu oposto a comparação de proximidade não ocorre.

Sabe-se que, se um nó é mais próximo aos eixos que os nós que estão nos quadrantes adjacentes, o único nó que também pode apresentar essa característica é o nó que está no quadrante oposto a esse nó.

Quando houver mais de um candidato que se encaixe nas condições do critério 1 se faz necessário o uso do critério 2, que é apresentado a seguir.

Critério 2 ("City Block metric" ou "the Manhattan metric") – É escolhido o candidato que apresentar o valor mínimo para a métrica L_1. A métrica L_1 é igual à soma dos deslocamentos a partir das fronteiras dos eixos x e y com origem no ponto a ser excluído $\left(L_1 = d_x + d_y\right)$.

Uma vez definido o candidato mais adequado, o nó a ser excluído é substituído por este. Em seguida, é feita a reinserção de nós situados no mesmo quadrante do substituto e nos quadrantes adjacentes ao deste nó. Não ocorre reinserção no quadrante oposto ao quadrante que contém o nó de substituição.

Através de testes empíricos, Samet observou que o número de comparações necessárias para a exclusão em point quadtrees é proporcional a $\log_4^N$.

2.5. Busca em Point Quadtrees

Neste trabalho foram implementadas as buscas por nó (simples), por região e por vizinhança. Os três tipos de busca são descritos abaixo.

2.5.1. Busca Simples

A busca simples em PQs é implementada como parte do processo de inserção. É verificado o quadrante em que o nó pode ser posicionado, de acordo com os valores das suas coordenadas. Dessa maneira, a pesquisa busca atingir a posição exata do nó percorrendo subquadrantes da PQ, retornando o nó encontrado ou mensagem negativa.

2.5.2. Busca Por Região

A pesquisa por região define um retângulo no espaço de particionamento e busca por pontos nessa região. A partir da raiz, a árvore é percorrida nos quatro quadrantes, selecionando os pontos que estão na região de busca. Caso o ponto esteja na região retangular, é guardado em um vetor.

2.5.3. Busca Por Vizinhança

A busca por vizinhança realiza a pesquisa em um círculo definido por um ponto (centro da busca) e um raio que define a vizinhança a ser verificada. Caso o ponto esteja na região circular, é guardado em um vetor.

3. Infraestrutura do Software

O desenvolvimento do Software Visualizador de PQs acompanhou a infraestrutura utilizada no desenvolvimento da Interface Luana [Dacome 2012].

Para a representação gráfica da árvore PQ, foi escolhida a linguagem C++. Foi utilizado o framework multiplataforma Qt que suporta código em C++ e fornece uma grande quantidade de bibliotecas para implementar softwares com interface gráfica. Para representar os nós de forma hierárquica, foi utilizada a biblioteca para gerar grafos Graphviz, cujos gráficos a serem gerados são codificados utilizando a linguagem DOT.

3.1 Graphviz

O nome Graphviz [Ellson, Gansner, et al 1988] (Graph Visualization Software) é um pacote de ferramentas, de código aberto, desenvolvido nos laboratórios de pesquisa da AT&T para desenhar grafos especificados em DOT[Gansner, Koutsofios e North 2009]. O posicionamento dos nós e arestas do grafo é feito de forma automática. Para a implementação do software de interface, o Graphviz foi utilizado como uma biblioteca, adicionando ao projeto as bibliotecas libgvc.so e libgraph.so e acrescentando o header gvc.h através da seguinte linha de código:

```
#include <graphviz/gvc.h>
```

Os grafos são gerados em memória principal. É necessário que seja gerado um array de char (char*) contendo o código DOT do grafo a ser desenhado, esse array é passado para o Graphviz para que seja renderizado. O Graphviz, então, entrega a imagem codificada em outro array de char.

Desta forma, o software de interface consegue exibir a representação hierárquica dos nós de uma Point Quadtree, através de uma imagem gerada pelo Graphviz a cada evento ocorrido.

3.2. Qt

Qt [Blanchette e Summerfield 2008] é um framework multiplataforma que permite compilar aplicações para Windows, Mac, Linux ou outros sistemas Unix. Suporta código em C++, possui código aberto, desenvolvido pela empresa norueguesa Trolltech (posteriormente adquirida pela Nokia). O Qt fornece uma grande quantidade de bibliotecas para implementar softwares com interface gráfica. Um exemplo de sucesso da utilização do Qt é o projeto KDE [KDE... 2013].

Para exibir os gráficos do resultado do particionamento do espaço e da representação hierárquica dos nós de uma Point Quadtree no software de interface, foram utilizadas algumas classes do Qt. Para gerar o gráfico do resultado do particionamento do espaço, foram inseridos objetos da classe QGraphicsItem (QGraphicsRectItem para representar pontos e QGraphicsLineItem para representar

linhas de particionamento) em um QGraphicsScene, e posteriormente este QGraphicsScene é exibido em um QGraphicsView.

Para gerar o gráfico da representação hierárquica dos nós, é obtido um array de char contendo a imagem renderizada pelo Graphviz, convertido em um QByteArray e depois gerado um pixmap em um QPixmap. O QPixmap gerado é inserido em um QGraphicsScene, e postriormente este QGraphicsScene é exibido em um QGraphicsView.

4. Implementação

A implementação do software de interface foi realizada com o uso da linguagem de programação C++. As duas classes principais criadas, *noQuadtree* e *quadtree*, implementam os algoritmos necessários para a representação, inclusão, exclusão em busca em PQs. A classe *noQuadtree* trata das propriedades dos nós de uma PQ, com os métodos que manipulam os dados e as ações que são feitas sobre os nós da PQ. A classe *quadtree* trata da criação da estrutura de dados e da implementação dos métodos de manipulação dos dados, desde a criação da árvore, passando pelas rotinas de inserção, busca e exclusão, até a destruição da árvore.

Além disso, a classe *QMainWindow* cuida da interface gráfica, com a criação das funções que gerenciam a janela de interação com o usuário, para todos as ações e eventos que foram planejadas na construção do software presente neste trabalho.

4.1. Requisitos

Para demonstrar didaticamente uma PQ com suas operações e funcionalidades, foram elicitados os seguintes requisitos:

- A PQ deve ser gerada dinamicamente;
- Todas as imagens apresentadas em tela devem ser geradas em memória principal;
- Permitir inserir, excluir e buscar nós a partir de dados informados pelo usuário;
- Apagar completamente uma árvore;
- Apresentar em tela os resultados obtidos das buscas;
- Atualizar a tela a cada alteração na estrutura da árvore realizada através dos processos de inclusão ou exclusão;
- Permitir a qualquer momento interação com as imagens apresentadas em tela (zoom, rolagem de tela com o mouse e informações sobre os nós);
- Mostrar o caminho percorrido no processo de inclusão de um nó, permitindo interação com as imagens apresentadas em tela antes de finalizar o processo (zoom, rolagem de tela com o mouse e informações sobre os nós);
- Mostrar, passo a passo, o processo de exclusão de um nó, permitindo interação com as imagens apresentadas em tela durante o processo (zoom, rolagem de tela com o mouse e informações sobre os nós);
- Salvar o estado de uma PQ em um arquivo em memória secundária, em coordenadas cartesianas definidas pelo software;
- Carregar uma árvore a partir de um arquivo em memória secundária, podendo ele estar em coordenadas cartesianas conforme limites definidos pelo software ou em coordenadas UTM conforme limites definidos pelos fusos e zonas do território brasileiro, limites estes verificados no momento em que um arquivo é carregado.

4.2. Descrição da aplicação

4.2.1. Tela Inicial

A tela inicial do software Visualizador de PQ consiste de um menu para acesso às funções e de três janelas usadas para as operações e opções de visualização de Point Quadtrees, conforme a Figura 9.

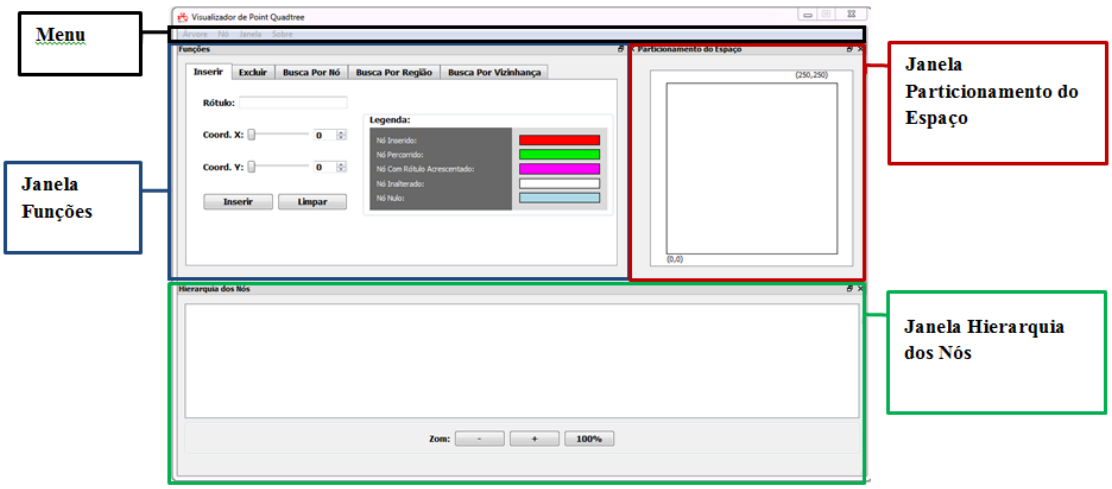


Figura 9. Tela inicial do Visualizador de PQ.

Cada uma das três janelas foi desenvolvida usando a classe QDockWidget do Qt, que permite o encaixe dentro da janela principal ou a flutuação de cada uma das janelas em um nível superior na área de trabalho.

4.2.2. Demais telas para entrada de opções e de dados

Há várias barras de menus para escolha dos modos de visualização e janelas de Funções para entrada de dados para atualização e consulta. Esses elementos da interface não serão aquí detalhados.

4.2.3. Janela de Particionamento do Espaço

A janela Particionamento do Espaço mostra a disposição geométrica dos pontos inseridos em uma PQ. O espaço usado é um quadrado que admite a representação de coordenadas inteiras, sendo o canto inferior esquerdo (0, 0) e o canto superior direito (250, 250). Todos os pontos com coordenadas inteiras com valor entre 0 e 250 podem ser representados neste espaço, como pode ser visto na Figura 10.

Optou-se por acompanhar o dimensionamento da tela de particionamento proposto na Interface Luana [Dacome 2012], uma vez que se gera uma padronização entre as duas ferramentas, possibilitando a integração futura entre os softwares desenvolvidos. A limitação das coordenadas a um quadrante de lado 250 não influencia na experiência com o software, uma vez que o seu objetivo é de ilustração das operações e algoritmos de uma PQ.

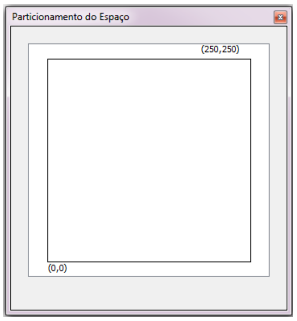


Figura 10. Janela de Particionamento do Espaço

Na inclusão de um novo ponto em uma PQ os nós percorridos até a inserção são realçados, através de mudança de sua cor. Na exclusão, a mudança de cor também é usada para destacar o nó que será substituído, o nó que o substituirá e todos os nós que serão reinseridos. Nos três processos de busca implementados são destacados o(s) ponto(s) encontrado(s) e a área delimitadora.

4.2.4. Janela de Hierarquia dos Nós

A janela de Hierarquia dos Nós, exibida na Figura 11, mostra a representação gráfica da árvore correspondente à PQ. Cada nó inserido é representado com seu rótulo e suas coordenadas, x e y, em uma elipse que é ligada por meio de quatro setas orientadas a cada um dos seus filhos. A orientação segue a direção "de pai para filho".

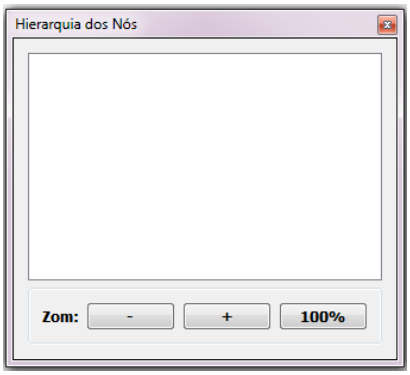


Figura 11. Janela de Hierarquia dos Nós

Assim como na janela de Particionamento do Espaço, a cada nó inserido, excluído ou buscado, temos o destaque de todos os nós relacionados através da mudança de cor.

Devido ao grande espaço ocupado pela representação gráfica de uma árvore, a janela de Hierarquia dos Nós pode ser redimensionada e pode haver a alteração do nível de zoom da imagem exibida. Além disso, é possível, através da barra de rolagem acessar áreas que não estejam visíveis na geração de árvores muito grandes.

5. Exemplos

Os exemplos a seguir mostram o comportamento das janelas do software mediante a inclusão, exclusão e buscas por nó em PQ, usando os dados apresentados na Tabela 1.

5.1. Exemplo de Inclusão em PQ

Usando os dados da Tabela 1, foi feita a inclusão em uma Point Quadtree. A Figura 12. Mostra a inserção dos cinco primeiros pontos desta tabela.

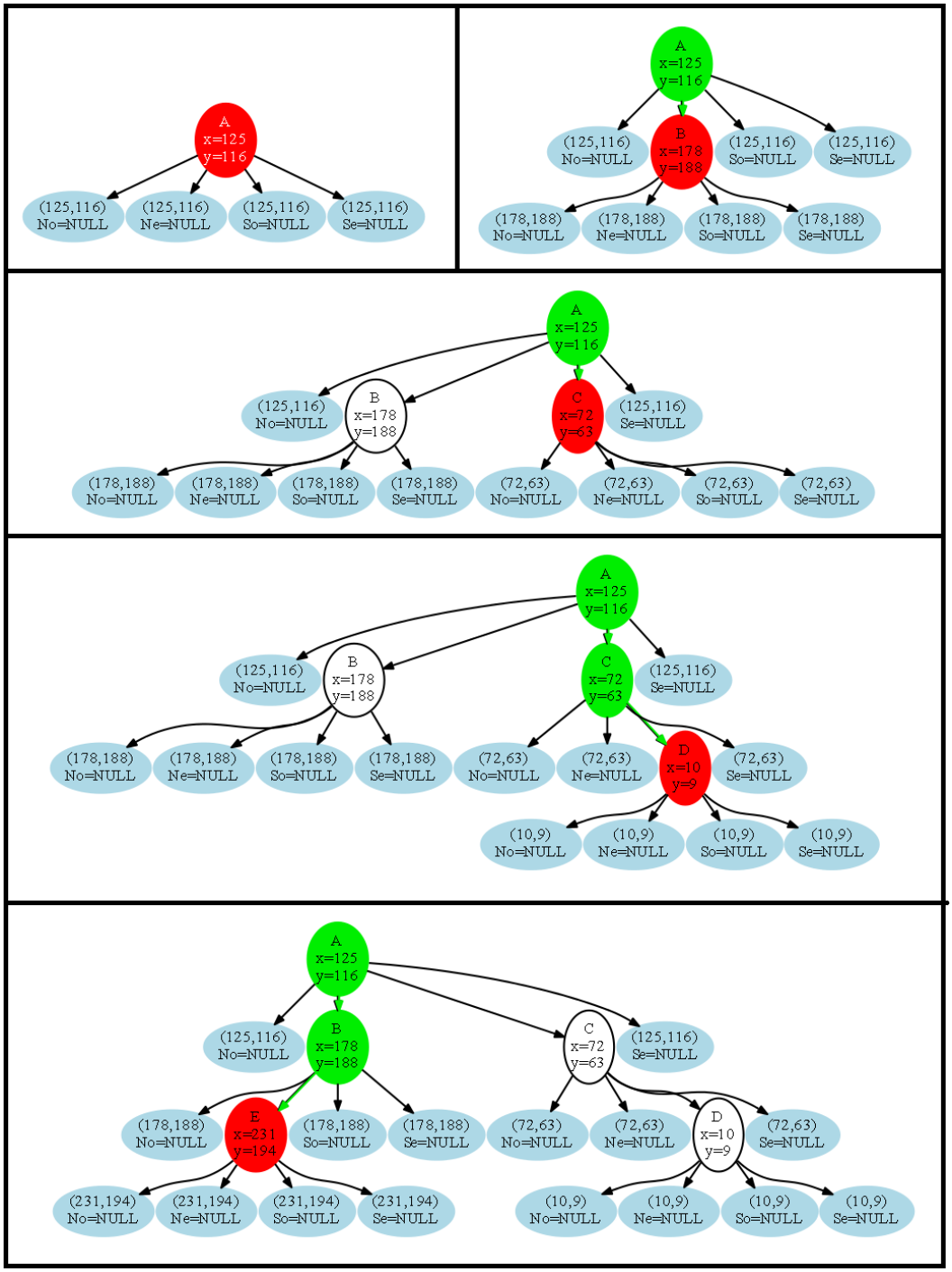


Figura 12. Inserção dos cinco primeiros pontos da Tabela 1.

5.2. Exemplo de Exclusão em Point Quadtree

A Figura 13 mostra a exclusão da raiz da árvore gerada com os dados da Tabela 1. No primeiro passo, o nó que será excluído e o seu nó substituto são destacados com

mudança de cor, de acordo com a legenda da aba de exclusão. No passo seguinte, além dos nós já destacados, os nós que serão reinseridos são também coloridos. Por fim, é mostrada a situação da árvore após a exclusão.

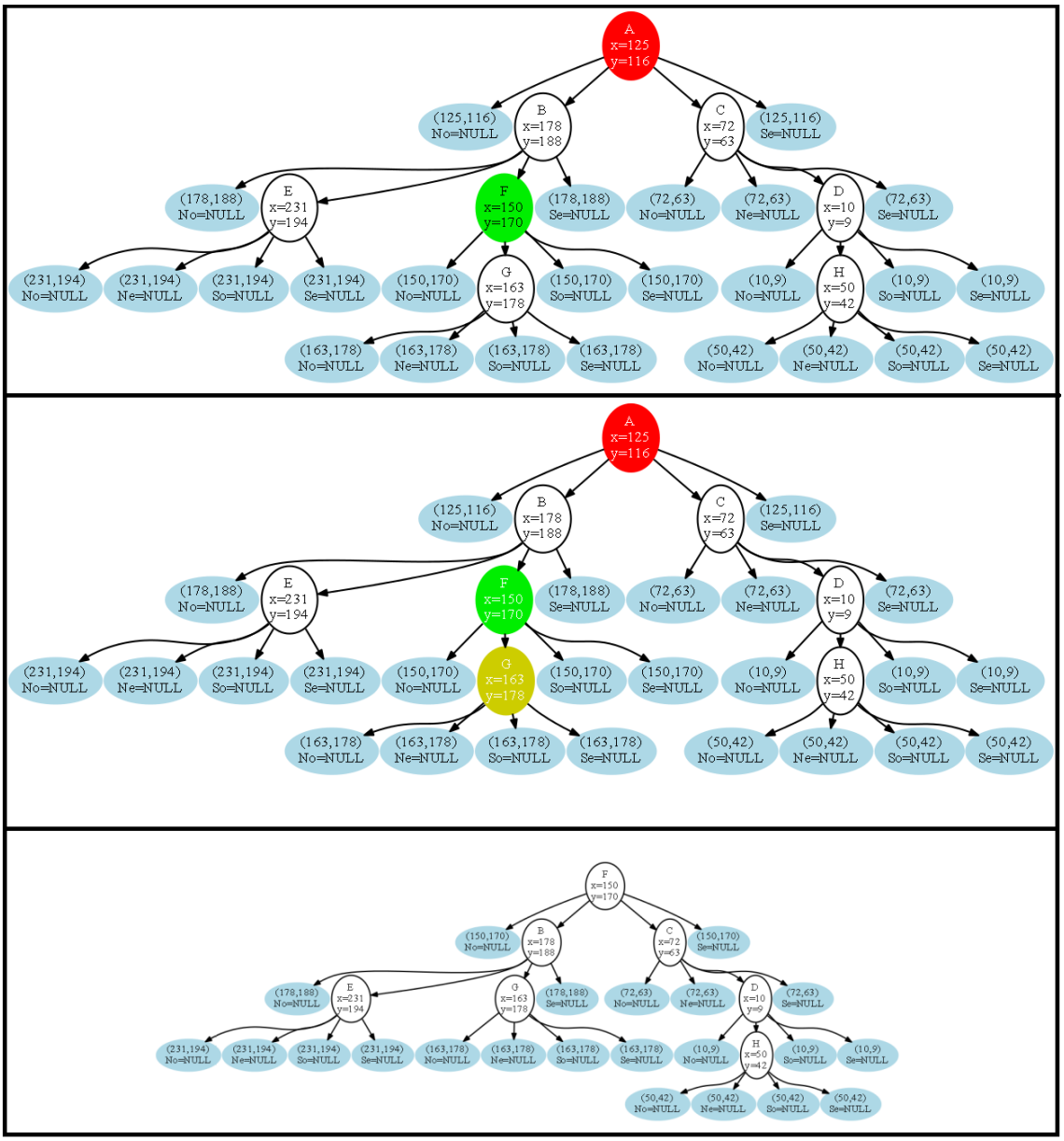


Figura 13. Exclusão do nó A (em vermelho). Substituição pelo no F (em verde). Reinserção do nó G (Em amarelo).

5.3. Exemplos de Busca em Point Quadtree

Seja a árvore criada no último passo do exemplo de inclusão em Point Quadtree. A Figura 14 mostra os três tipos de busca implementados no software.

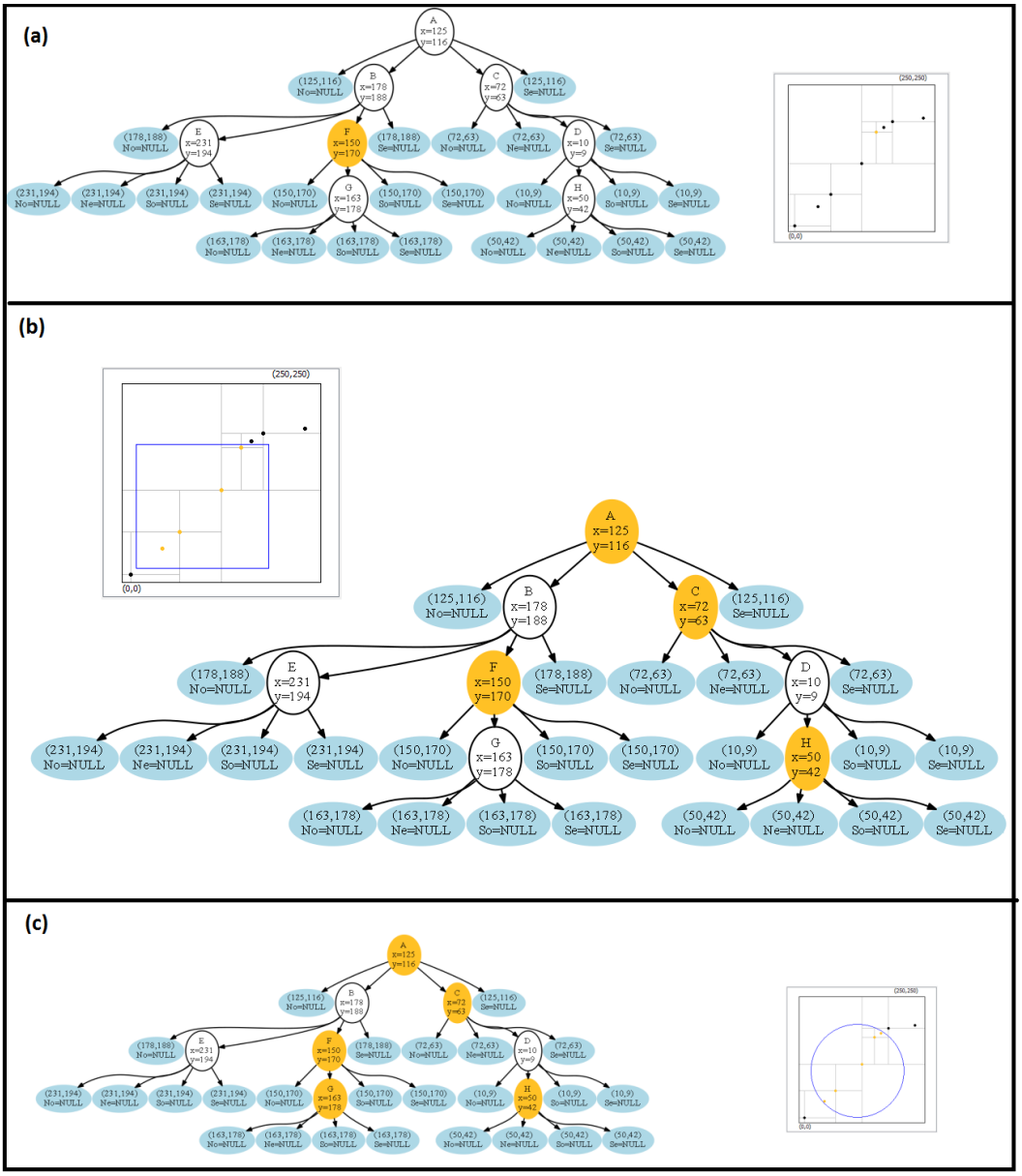


Figura 14. (a) Busca Simples. (b) Busca por Região. (c) Busca por Vizinhança.

6. Conclusões

O presente trabalho mostrou a definição, características e a manipulação de dados em Point Quadtrees. Foi feita uma descrição detalhada de cada conceito para as operações de inserção, exclusão e busca em Point Quadtrees.

Paralelamente, foi apresentado o software Visualizador de Point Quadtree desenvolvido, que propicia uma experiência visual no contato com essa estrutura de dados, sendo potencial facilitador no aprendizado da mesma. Com o software é possível a criação de uma PQ em um espaço 250 x 250. Também é possível a criação de uma árvore com dados aleatórios. Com isso, torna-se fácil o estudo de árvores dos mais diversos tamanhos e, assim, a busca, inclusão e exclusão podem ser experimentadas de diversas maneiras, de acordo com o interesse do usuário.

Referências

Bentley, J. L. (1975) "Multidimensional Binary Search Trees Used for Associative Searching", In Communications of the ACM 18, Volume 9, September, p. 509-517.

Bentley, J. L. and Finkel, R. A. (1974) "Quad Trees – A Data Structure for Retrieval on Composite Keys", In: Acta Informatica 4, p. 1–9.

Blanchette, J., Summerfield, M. (2008) "C++ GUI Programming with Qt 4", Prentice Hall; 2 edition.

Ellson, J., Gansner, E., Koutsofios, L., North, S. and Woodhull, G. (2003) "Graphviz and Dynagraph — Static and Dynamic Graph Drawing Tools", http://www.graphviz.org/Documentation/EGKNW03.pdf, acesso em 20 set 2013.

Ellson, J., Gansner, E., Koutsofios, L., North, S. and Woodhull, G. (1988), "Graphviz— Open Source Graph Drawing Tools". In. P. Mutzel, M. Jünger, Michael, S. Leipert, Graph Drawing, Isbn: 978-3-540-43309-5.

Gansner, E. R., Koutsofios, E. and North, S. (2009) "Drawing Graphs with Dot", http://www.graphviz.org/pdf/dotguide.pdf, acesso em 20 set 2013.

Gardner, H., Chen, J.-Q. and Moran S. (2010) "Inteligências Múltiplas ao Redor do Mundo", Artmed Editora; 1ª edição .

KDE Free Qt Foundation (2013), http://www.kde.org/ community/ whatiskde/ kdefree qtfoundation.php, acesso em 20 set 2013.

Lima, R. D., Mota, G. L. A. and Pinto, P. E. D. (2012) "Interface Luana: uma Aplicação Gráfica para o Ensino da Árvore Binária Kd-Tree" In SIBGRAPI - Conference on Graphics, Patterns and Images, 2012, Ouro Preto. Anais do Sibgrapi, 2012.

Miranda, A. C. O. and Martha, L. F. (2002) "Mesh Ggeneration on High-Curvature Surfaces Based on Background Quadtree Sstructure" In 11th International Meshing Roundtable , Springer-Verlag, p.333-342.

Samet, H. (1990) "The Design and Analysis of Spatial Data Structures", Addison_Wesley.

Samet, H. (2006) "Foundations of Multidimensional and Metric Data Structures", Morgan Kaufmann.

Samet, H., Rosenfeld, F. A., Shaeffer, C. A. and Webber, R. E. (1984) "A Geographic Information System Using Quadtrees" In Pattern Recognition Volume 17, Issue 6, p. 647–656.

Strobach, P. (1991) "Quadtree-structured Recursive Plane Decomposition Coding of Images", In IEEE Trans. Signal Processing, vol. 39, p. 1380-1397.

HHS Sticker: Aplicação Médica em Java para TV Digital

Claudia O. Neves[1], Thales V. G. da Silva[1], Roberto C. R. Machado[2], Alexandre Sztajnberg[1, 2, 3]

[1]Bacharelado em Ciência da Computação, DICC/UERJ
[2]Mestrado em Engenharia Eletrônica - PEL/FEN
[3]Mestrado em Ciências Computacionais - CComp/IME
Universidade do Estado do Rio de Janeiro - UERJ
CEP - 20550-900 – Maracanã, Rio de Janeiro – RJ – Brasil

claudiadasneves@gmail.com, thvela@gmail.com,
roberto.rodrigues@uerj.br, alexszt@ime.uerj.br

Abstract. *This paper presents an interactive DiTV application to support the remote monitoring of elderly patients in their homes The application is able to receive relevant information such as the Care Plan directed by medical personnel and real-time alerts to notify the patient of any routine to be executed. The application also allows sending physiological measurements performed by the patient or caregiver to a Monitoring Central. This requires the installation of a digital TV running the Ginga middleware, and Internet access. The application, developed in Java facilitates the patient interacting with the application with no need to learn a new technology, via the TV remote. This application is integrated with a broader patient monitoring system, called HHS (Home Health System).*

Resumo. *O presente artigo apresenta uma aplicação interativa para TVDi para o acompanhamento remoto de paciente idosos em suas residências A aplicação é capaz de receber dados relevantes, tais como o Plano de Cuidados indicado pela equipe médica e alertas em tempo real para notificar o paciente de alguma rotina que deve ser executada. A aplicação também permite o envio de medidas fisiológicas realizadas pelo paciente ou cuidador para uma Central de Monitoramento. Para isso é necessária a instalação de uma TV digital com o middleware Ginga, e acesso à Internet. A aplicação, desenvolvida em Java, cria facilidades, via controle remoto da TV, para o paciente interagir com a aplicação, sem a necessidade de aprender uma nova tecnologia. Esta aplicação é integrada a um sistema de monitoramento de pacientes, chamado de HHS (Home Health System).*

1. Introdução

A televisão, presente em quase todas as residências segundo IBGE [IBGE, 2011], é uma mídia de fácil acesso e simples manuseio. Com o encerramento das transmissões analógicas previsto para 2016 [Enc, 2012], televisores analógicos serão substituídos pelos digitais, que além de reproduzirem imagens com maior qualidade, possibilitam o desenvolvimento de aplicativos interativos.

Na arquitetura proposta para a TV Digital (TVDi) os canais de descida (*emissora-usuários*) podem transportar conteúdo de áudio e vídeo, e também dados e aplicações, que devem aderir ao padrão Ginga. Esses conteúdos devem ser produzidos, estruturados e transmitidos pela emissora, e serão recebidos, processados, exibidos e executados no receptor digital ou *set-top box* do usuário. Neste caso, a possibilidade de interatividade se dá através dos dados recebidos junto com o áudio e vídeo e também através dos programas, recebidos da emissora, que podem ser executados localmente no equipamento do usuário. Entretanto, não é possível, com estes mecanismos continuar a interação, pois seria necessário uma forma de retorno, um mecanismo de comunicação para troca de mensagens. Alguns *set-top boxes* disponíveis oferecem esta possibilidade, chamada de canal de retorno, provendo em seu sistema mecanismos de conectividade, através de interfaces de rede. Neste caso, se o usuário possui conexão à Internet, é possível configurar o *set-top box* como um ponto da rede. Com isso, abre-se a possibilidade para a execução de programas na televisão, que (i) não dependam de uma emissora para serem recebidos, isto é, podem ser carregados (download) de um repositório da Internet e (ii) que necessitem de troca de mensagens, com um banco de dados ou com um servidor para receber ou enviar dados, requisições e respostas.

A interatividade possibilitada pelo sistema de TVDi com canal de retorno, tem o potencial para levar aplicações sérias a uma parte da população que não tem intimidade com tecnologia ou acesso à Internet com facilidade. Entre estas aplicações podemos citar o acesso a instituições bancárias, aplicações de Governo Digital (e-Gov) ou TeleSaúde (e-Health). Por outro lado, os lares brasileiros com acesso à Internet ainda são em número reduzido, se comparado aos que possuem aparelhos de TV. Por isso, são necessários investimentos para popularizar aparelhos de TVDi e o desenvolvimento do suporte ao canal de retorno usando tecnologias mais acessíveis ou subsidiadas como WiMAX, 3G, UHF, etc; visto que as tecnologias atuais, como: ADSL ou *cable modems*, ainda têm custo restritivo.

Na reportagem veiculada pelo Banco Mundial sobre TV Digital Brasileira [The World Bank Group, 2013] são apresentados os serviços de e-GOV providos pela EBC, Empresa Brasileira de Comunicação, grupo de mídia estatal, que oferece acesso interativo a vários serviços para o cidadão, incluindo serviços de saúde. O relatório *Brasil 4D: Estudo de impacto socioeconômico sobre a TV digital pública interativa* [World Bank, 2013] oferece mais informações sobre as primeiras avaliações do SBTD (Sistema Brasileiro de TV Digital). Um dos pontos de avaliação são as aplicações na área de saúde que executam sobre o Ginga, o sistema de *middleware* proposto para o sistema de TV Digital do Brasil. Os benefícios para a população de baixa renda são indiscutíveis. Mas a maioria das aplicações avaliadas é limitada à oferta de informação direcionada por área. Não foram avaliadas nesse relatório aplicações estritamente interativas.

O presente trabalho apresenta uma aplicação para TVDi para o acompanhamento remoto de pacientes idosos em suas residências A aplicação é capaz de receber dados relevantes, tais como o Plano de Cuidados indicado pela equipe médica, e alertas em tempo real para notificar o paciente de alguma rotina que deve ser executada. A aplicação também permite o envio de medidas fisiológicas realizadas pelo paciente ou cuidador para uma Central de Monitoramento. Para isso é necessária a instalação de uma TV digital com o *middleware* Ginga, e acesso à Internet. A aplicação desenvolvida

cria facilidades, via controle remoto da TV, para o paciente interagir com a aplicação, sem a necessidade de aprender uma nova tecnologia. Esta aplicação é integrada a um sistema de monitoramento de pacientes, chamado de HHS (*Home Health System*) [Sztajnberg, 2009].

A arquitetura da aplicação é apresentada, junto com um protótipo desenvolvido em Java, utilizando a plataforma Sticker Center [StickerCenter, 2013], da TQTVD [TQTVD, 2012] aderente ao padrão Ginga. Por esta razão a aplicação é chamada *HHS Sticker*. Elementos que permitem sua integração ao HHS também foram desenvolvidos, incluindo o acesso ao banco de dados e o mecanismo para enviar notificações de alerta ao paciente, que podem aparecer no *HHS Sticker* durante sua novela favorita, para lembrá-lo de uma medicação, por exemplo.

Este artigo está estruturado da seguinte forma. Na Seção 2 apresentam-se alguns conceitos do padrão brasileiro de TV Digital e seus componentes. Na sequência, Seção 3, é apresentado o sistema de monitoramento remoto de pacientes e descrita a arquitetura do HHS *Sticker*. Na Seção 4 é apresentada a arquitetura do sistema. A Seção 5 apresenta o protótipo implementado e uma discussão sobre detalhes da comunicação entre as partes, bem como, uma descrição de sua interface. Na Seção 6 é feito um rápido comparativo com outros trabalhos relacionados e na Seção 7, são apresentadas as considerações finais e algumas sugestões de trabalhos futuros.

2. Padrão Brasileiro de TV Digital

Um sistema de TV Digital consiste na transmissão de fluxos de bits contendo áudio, vídeo e dados de uma emissora para os aparelhos de TV. Para exibir o conteúdo, os aparelhos de TV precisam estar preparados para receber o sinal digital das emissoras, seja usando um conversor digital integrado ou um conversor externo, chamado de *set-top box (STB)*.

No Brasil, conversores seguem o padrão Brasileiro – SBTVD [HIST 2013] de TV Digital chamado comercialmente de ISDB-TB, que se baseia no padrão japonês, acrescido de algumas novas tecnologias e melhorias, como maior capacidade por canal usando compressão de vídeo MPEG-4 AVC (H.264) e maior robustez através do *middleware* Nipo-Brasileiro de especificação aberta "Ginga". Esse, desenvolvido pelos laboratórios Telemídia (da PUC-Rio) e LAViD (da UFPB) seguindo recomendação ITU-T [ITU, 2013] para serviços IPTV [IPTV, 2013] e permitindo aplicações interativas complexas em um ambiente também procedural [Grupos de Trabalho na SBTVD, 2011].

Em sua especificação 1.0, o Ginga apresenta o ambiente Ginga-NCL como subsistema lógico obrigatório, responsável pela execução de aplicações na linguagem declarativa LUA/NCL (Nested Context Language), desenvolvida pela PUC-Rio (Figura 1). Sua arquitetura permite a adição de extensões opcionais, entre elas o subsistema e ambiente Ginga-J [Ginga, 2013], desenvolvido pela UFPB (Ginga 2.0) que é responsável pela execução de aplicações na linguagem procedural Java. Essas aplicações são entregues aos dois ambientes pelo terceiro subsistema Ginga-CC (Ginga Common-Core), que cuida da exibição de vários formatos de mídia, do controle do plano gráfico para a especificação do ISDB-TB e também do acesso ao Canal de

Interatividade ou Retorno (responsável por controlar o acesso à camada de rede), previsto no padrão.

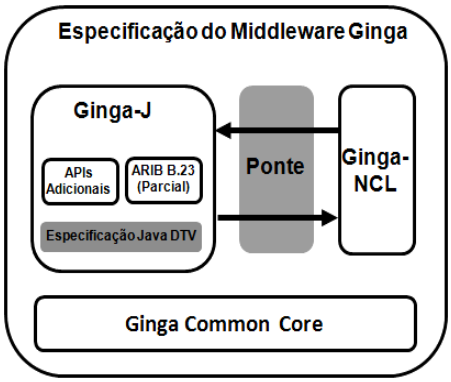


Figura 1. Especificação Ginga [Esp, 2009]

3. Ambiente de Desenvolvimento

Para o desenvolvimento da aplicação foi utilizada a plataforma de execução e desenvolvimento chamada AstroTV, aderente ao padrão Ginga e um *set-top box* com o AstroTV embarcado e com canal de retorno disponível.

3.1 AstroTV

O AstroTV [AstroTV, 2012] é uma implementação de *middleware* para TVDi no padrão Brasileiro, compatível com a especificação Ginga. O AstroTV é atualmente embarcado em *set-top boxes* e televisores em parceria com diversos fabricantes (Sony, LG, Panasonic, Philips, Toshiba, Sharp, D-Link) e possui recursos como: interatividade, portabilidade (interatividade em dispositivos móveis) e programação estendida em entretenimento e informação (acréscimo de conteúdo pela emissora).

A TQTVD também disponibiliza o AstroBox [AstroBox, 2011], uma ambiente virtualizado para simulação do ambiente AstroTV, distribuído com Linux Ubuntu 10.04 [Ubuntu, 2010] e executado através de uma máquina virtual, Virtual Box [V-Box, 2013]. Com o AstroTV e o AstroBox é possível executar *Stickers* (detalhes na Seção 3.3), que também são aplicações Ginga, e podem ser incorporados em qualquer receptor que possua o AstroTV e que dê suporte à solução de *Broadband* e *Broadcast* da TQTVD (o *StickerCenter*) [StickerCenter, 2013].

3.2 Java DTV

O Ginga-J, suporte Java ao Ginga, baseia-se nos pacotes do padrão JavaDTV [Kulesza, 2009]. Este padrão, por sua vez, usa como base a API LWUIT (*Lightweight User Interface Toolkit*), desenvolvida pela Sun. Diversos componentes da API JavaDTV foram utilizadas na aplicação desenvolvida, entre eles os componentes gráficos (*buttons*, *checkboxes*, *dropdown*, etc.), gerenciadores de *layout* (*menus*, *tabs,* etc.), estilos, temas, transição de tela animada e um modelo de tratamento de eventos bastante simples, similar a um *Applet*.

3.3 Stickers e a Plataforma StickerCenter

Para proporcionar um ambiente de programação produtivo, com suporte às aplicações Ginga Lua/NCL e Ginga-J, a TQTVD criou o conceito de *Sticker* [Ubuntu, 2010], com o suporte de um ambiente de execução e um repositório para download na web, chamado *StickerCenter* [StickerCenter, 2012].

Um Sticker (Figura 2) é uma aplicação interativa Ginga (Lua ou Java) similar em conceito a uma aplicação para celulares e *tablets* ou a um *Widget* - pequeno aplicativo com funcionalidade limitada, instalado e executado dentro de uma página web por um usuário final. Através dele é possível: acessar conteúdos complementares à programação da TV, participar de votações, acessar Internet, etc. *Stickers* podem ser adquiridos ou atualizados via *StickerShop*, baixados para o receptor via canal de interatividade (via Internet) ou através do canal de TV digital terrestre transmitido por emissoras que suportem a plataforma *StickerCenter*.

Figura 2. Sticker e suas áreas identificadas [UBU 2010]

Os Stickers seguem padrões de consistência visual, funcional e de navegação, com funções comuns para facilitar o aprendizado e a experiência de seu uso aos usuários. Elementos que constituem esse padrão são fixos e não podem ser customizados. Por exemplo, fontes, como ilustradas na Figura 2, não possuem variações de negrito e itálico e de tamanho, para proporcionar uma leitura confortável na TV. Abas são numeradas para serem acessadas diretamente pelo controle remoto.

Podemos interpretar o *Sticker* como uma padronização ou wrapping de aplicações TVDi. Para implementar a aplicação, propriamente dita, em Java, a classe principal deve implementar a interface *Xlet* [Xlet, JSR 217]. Para uma aplicação de

TVDi se caracterizar como um Sticker é necessário utilizar o pacote *com.tqtvd.stickercenter*.

Para que um Sticker esteja disponível para download e instalação em set-top boxes é necessário que o desenvolvedor faça *upload* para o site do *StickerCenter* de um arquivo compactado contendo os arquivos: imagens "ico.png" (96 x 84 pixels), "exibition.png" (66 x 55 pixel), "screenshot.png" (114 x 269 pixels) e uma pasta *source* contendo os códigos e imagens utilizadas na aplicação.

3.4 A Interface Xlet

A interface *Xlet,* para programação de aplicações de TVDi em Java, tem a estrutura similar à da classe Applet. Cada *Xlet* possui uma instância da classe *javax.microedition.xlet.XletContext*, que é um contexto associado, usado para prover um modo para a aplicação obter informações sobre o seu ambiente, além de comunicar mudanças de estados (Figura 3) em seu ciclo de vida.

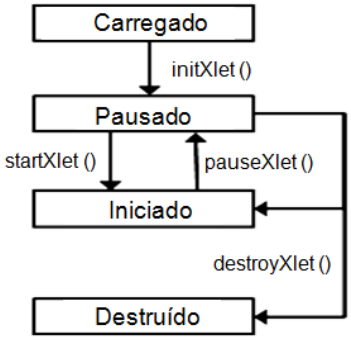


Figura 3. Ciclo de vida dos Xlets [Morris, 2012]

O *middleware* instancia a classe *Main* e identifica nela, o ponto de entrada da aplicação (classe *javax.microedition.xlet.Xlet*) e executa a aplicação utilizando a máquina virtual Java. A classe *Main* invoca os métodos responsáveis pela mudança de estados [Batista, 2007] e é responsável por gerenciar o ciclo de vida das *Xlets* conforme figura 4*;* controle este, semelhante ao dos *MIDlets* – aplicativos Java para dispositivos móveis – [Midlet, 2013]. Ambos oferecem a funcionalidade de pausar as aplicações em inatividade (não visíveis) com objetivo de liberar recursos em ambientes onde haja restrições de hardware [Kulesza, 2009].

```
public class Main implements Xlet{
   public void startXlet() throws XletStateChangeException { }
   public void pauseXlet() {  }
   public void destroyXlet(boolean arg0)
            throws XletStateChangeException { }
(...)
```

Figura 4. Assinatura da classe *Main* gerenciadora do ciclo de vida das *Xlets*

3.4 Set-top Box

O *set-top box* modelo DTB-331 [D-Link, 2013] foi utilizado para avaliar a aplicação. Assim como todo set-top box (STB) ele possibilita que televisões digitais ou analógicas que não possuam conversor digital integrado apresentem o conteúdo enviado pelas

emissoras via SBTVD. Ele possui o AstroTV embarcado, com suporte ao Ginga-J, e interfaces de rede Ethernet e WiFi. Com isso, foi possível contar com o canal de retorno ou de interatividade para retornar os dados selecionados ou incluídos pelo usuário ao banco de dados do HHS, hospedado em um servidor remoto. O usuário acessa e interage com o *HHS Sticker* através do controle remoto da TV.

4. HHS Sticker

O sistema *Home Health System* (HHS), no qual o HHS Sticker está inserido, tem o objetivo de monitorar, coletar, armazenar, avaliar e distribuir dados de contexto, compostos por informação de sensores (temperatura, luminosidade, umidade) dados de atividades do paciente, sua localização e medições fisiológicas (pressão arterial, ritmo cardíaco, etc.). Esses dados são enviados pela Internet para uma Central de Monitoramento, onde os mesmos são avaliados e persistidos em uma base de dados.

Os dados são disponibilizados para médicos e outros profissionais de saúde envolvidos no tratamento, que possuem históricos individualizados de cada paciente [Sztajnberg, 2009]. Os próprios pacientes, familiares e cuidadores também podem ter acesso a esses dados, porém de forma restrita, segundo orientação médica. O sistema pode enviar notificações aos pacientes de forma automática, segundo um Plano de Cuidados determinado pelo médico, ou manualmente (médicos e cuidadores).

No HHS é pressuposta uma infraestrutura instalada na residência do paciente (Figura 5) para coletar e monitorar os dados de contexto, composta por sensores de ambiente, pontos de acesso Wi-Fi para comunicação dos sensores e atuadores com um sistema de computação local [Macedo, 2011], que persiste as informações temporariamente e as transmite com segurança para a Central de Monitoramento.

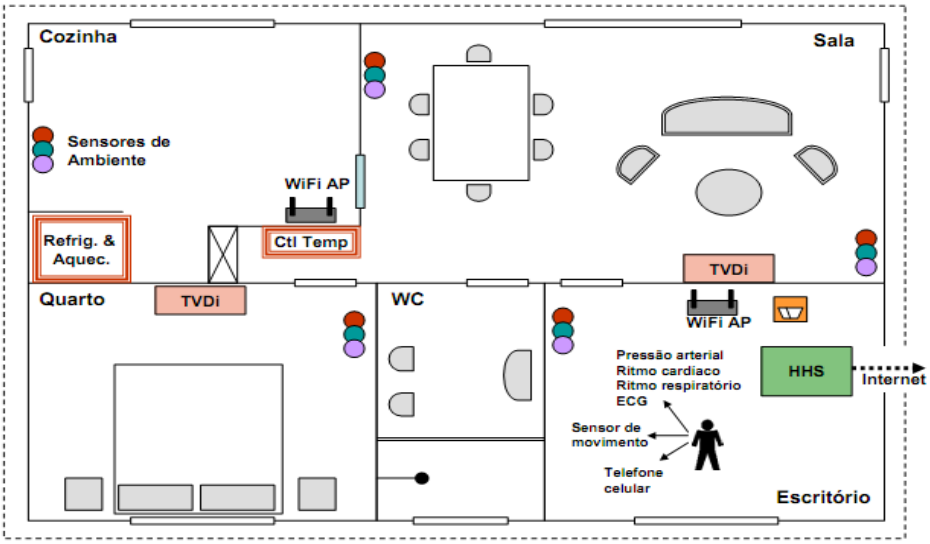


Figura 5. Sistema de Monitoramento Domiciliar da Saúde [Macedo, 2011]

A proposta do presente trabalho introduz o uso de TVs Digitais, capazes de receber e exibir mensagens, integrados à infraestrutura do HHS, permitindo a interação de pacientes, familiares e cuidadores com o sistema através de um equipamento que já

faz parte de seu cotidiano. Como preparação preliminar, cada *set-top box* da residência deve ser configurado para acessar o Sticker Center de onde o HHS Sticker será obtido.

4.1 Arquitetura

A arquitetura do sistema proposto neste trabalho é apresentada na Figura 6 e integra três elementos: (i) o cliente para apresentação de dados e interação paciente-médico, mais especificamente o próprio *HHS Sticker*, (ii) o servidor que recebe os dados coletados dos vários pacientes para persistência e realiza a comunicação com o *HHS Sticker* (uma extensão do sistema da Central de Monitoramento do HHS) e uma interface de suporte para os médicos e monitores enviarem mensagens e notificações ao HHS Sticker de forma simples. Essa função de envio de mensagens é integrada à interface de acesso do HHS.

O *HHS Sticker* foi desenvolvido em Java versão 1.3, por questões de compatibilidade com a JVM embarcada no *set-top box*. Deve ser carregado (*download)* a partir do site do Sticker Center através de conexão à Internet, configurada no *set-top box*. Os outros módulos de usuário e suporte, disponíveis no HHS, executam em computadores e notebooks que possuem mais recursos se comparados com o *set-top box*, e também foram desenvolvidos em Java (versão 1.6).

O *HHS Sticker* e os demais módulos comunicam-se com o servidor na Central de Monitoramento, que é responsável pela autenticação do usuário, a gerência de dados e notificações e o acesso ao banco de dados. Toda comunicação é realizada através de socket TCP.

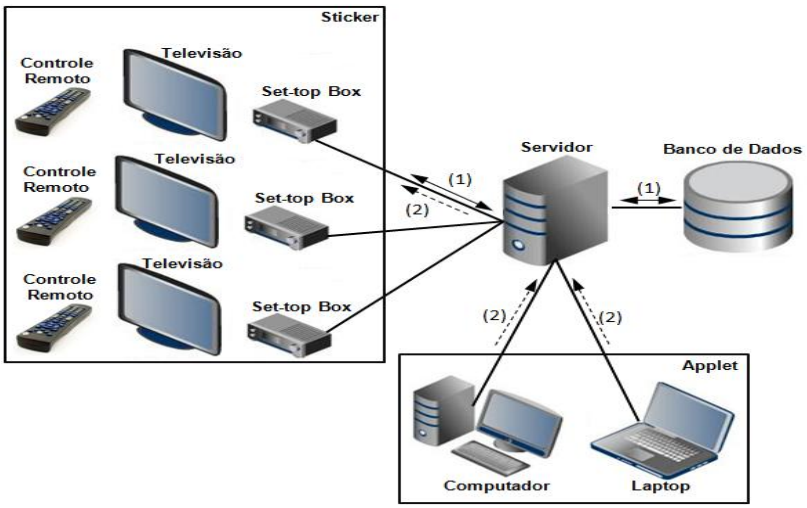


Figura 6. Arquitetura Geral da comunicação Cliente-Servidor

Após o *download* pelo canal de interação e a instalação no *set-top box*, o HHS *Sticker* pode ser acionado no aparelho pelo controle remoto. Através do endereço padrão da Central de Monitoramento embutido no *HHS Sticker*, ou introduzindo um endereço alternativo (diferentes provedores de cuidados médicos), qualquer usuário pode iniciar uma conexão com o servidor.

Estando o Sticker ativo e conectado, as interações de usuários com o sistema se dão conforme a Figura 6. O paciente pode navegar pelas abas da aplicação e decidir enviar medidas manuais e acessar as informações desejadas disponíveis no banco de

dados, como: plano de atividades, avisos e plano de cuidados (1). O médico, os monitores ou o próprio sistema podem enviar mensagens para o Sticker, notificando o paciente com alguma informação sobre o plano de cuidados (2).

4.2 Estrutura do Sistema

O *HHS Sticker*, módulo cliente do sistema, adere à estrutura básica de um *Xlet*, e faz uso de elementos adicionais para as funções de *Sticker* e para a comunicação com os demais módulos do sistema (Figura 7).

A classe *Main* é o *entryPoint*, que implementa a interface *Xlet*, instancia um objeto *Sticker*, realiza a configuração do mesmo seguindo seus padrões e caracterizações. A classe *RequestManager* é responsável por gerenciar requisições vindas do cliente, um objeto da classe *PageManager* (do pacote *StickerCenter*), que também implementa a interface *IResponseListener.* Ao receber uma requisição, salva referências do objeto requerente em uma lista de espera (método *addRequestReference* do *ResponseReceiver*) e em seguida chama o objeto *Connection* encarregado de estabelecer, manter ou retomar uma conexão com o Servidor, passando um objeto *Request* como parâmetro no método *sendRequest*. Essa referência do objeto, juntamente com seu código, também são salvos em uma lista pela classe *ResponseReceiver* que é acionada novamente assim que uma resposta for recebida, procurando o objeto requerente que espera pela resposta (um *IResponseListener*) e o notifica. Com esta abordagem, mesmo que o usuário troque a página ativa (*ActivePage*) a resposta é encaminhada ao elemento (*PageManager*) responsável pela requisição.

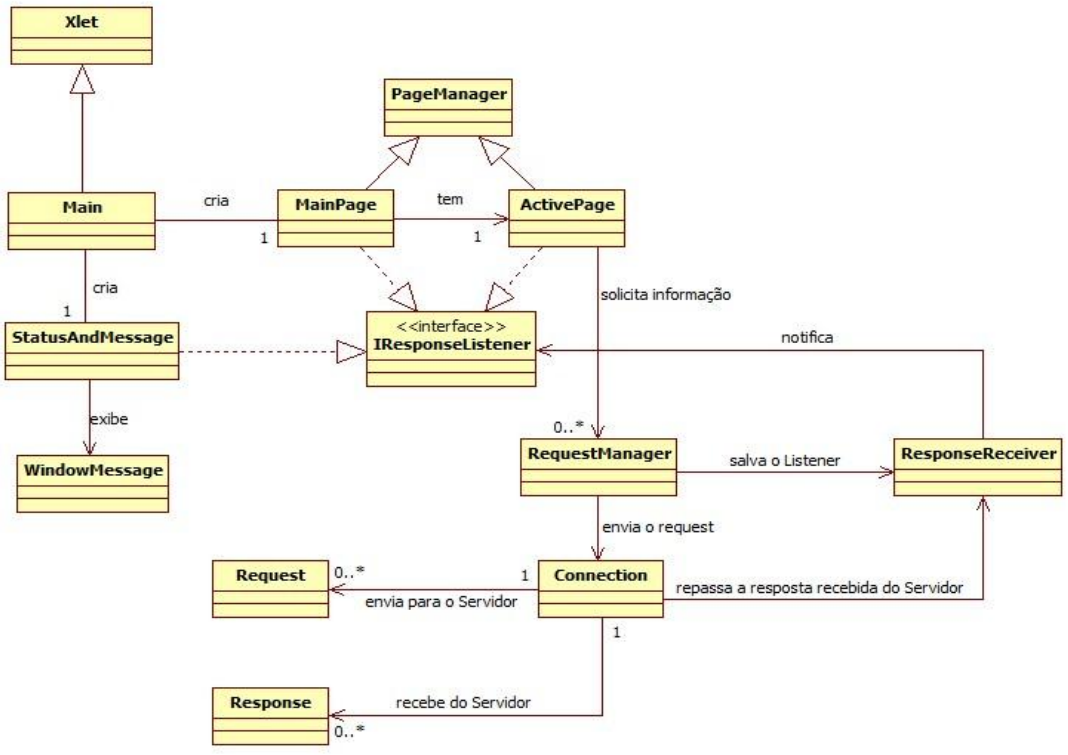


Figura 7. Estrutura do Sticker

Assim que *o HHS Sticker* inicia sua execução, uma conexão é estabelecida com o Servidor. As informações de endereço IP e número de porta para conexão com o Servidor são conhecidos e a classe *Connection* entra em execução para estabelecer a execução. Na primeira interação entre o *HHS Sticker* e o Servidor são informados os dados para *login* (usuário e senha cadastrados), obtidos na ação de enviar dados na *MainPage*. Uma vez estando o usuário conectado e autenticado, um canal persistente se forma e o *HHS Sticker* pode enviar requisições e dados, e receber informações e mensagens de notificação (Figura 8).

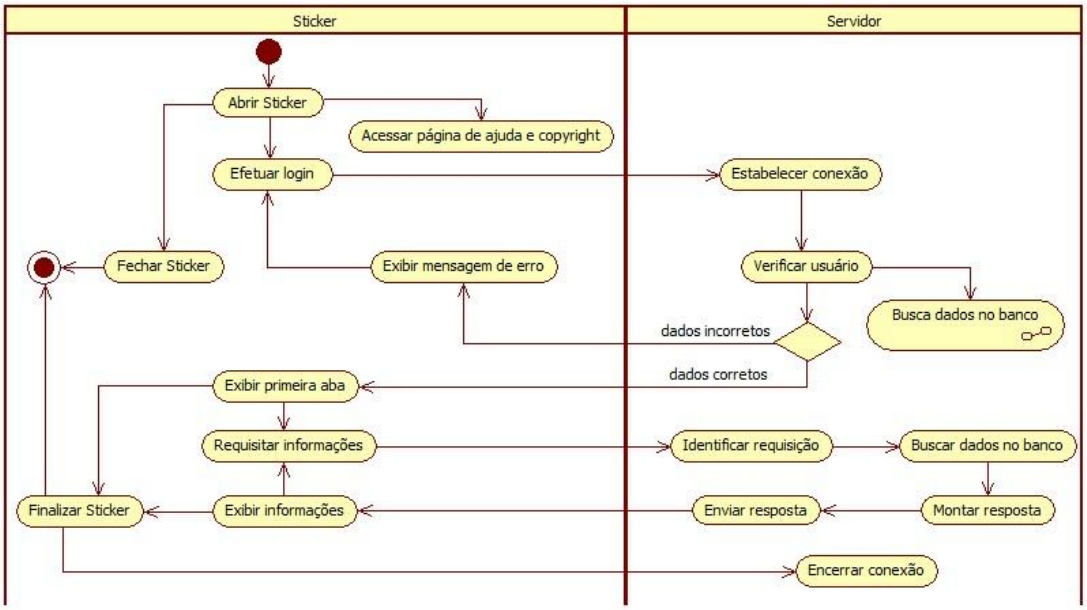


Figura 8. Fluxo de interação entre o HHS Sticker e o Servidor

Todas as mensagens recebidas pelo *HHS Sticker* chegam à classe *Connection*, que repassa as mesmas para a classe *ResponseReceiver*. Esta identifica qual objeto está esperando tal mensagem e repassa as informações para ele através de um evento de notificação. Cada mensagem tem um número de identificação que define se ela é uma resposta a algum pedido de um *PageManager* ou se é uma mensagem vinda direto do Servidor, como um pedido de Status ou de Aviso. Se a resposta for para um *ActivePage* (página *PageManager* que está ativa no momento) ela atualiza as informações na tela e se for uma mensagem ou verificação de status a classe *StatusAndMessage* será notificada e irá responder à verificação de status ou abrir uma janela na página ativa informando o conteúdo recebido.

A classe *Server* é a classe base do Servidor (Figura 9), responsável por aceitar conexões dos *Stickers*, e criar um *socket* TCP/IP ativo e passá-las adiante para um tratador *ServerConnection*, uma *thread* que por sua vez gerencia a troca de mensagens (requisições e respostas) desta conexão. Ao receber uma requisição o *ServerConnection* delega a execução da requisição ao objeto *ResponseManager*, chamando seu método *executeRequest* e aguarda a resposta para enviá-la ao cliente *Sticker*.

A classe *ResponseManager* valida o *token* de identificação do paciente e identifica o tipo de requisição recebida, passando-o para o método *ResponseMaker* que,

por sua vez, processa a requisição específica e devolve um *Response*. O *ResponseMaker* é um elemento importante na integração da aplicação HHS Sticker com o sistema HHS e o servidor de banco de dados. Para isso, ele utiliza a biblioteca DAO, comum a todos os módulos do sistema HHS. Em princípio, toda a funcionalidade do Servidor poderia estar embutida no sistema HHS. Entretanto, um servidor específico com finalidade de tratar os dados dos pacientes permitiu uma avaliação isolada do comportamento da aplicação sem dependência com o resto do sistema HHS que trata os dados dos sensores e, ainda assim, integrado ao mesmo.

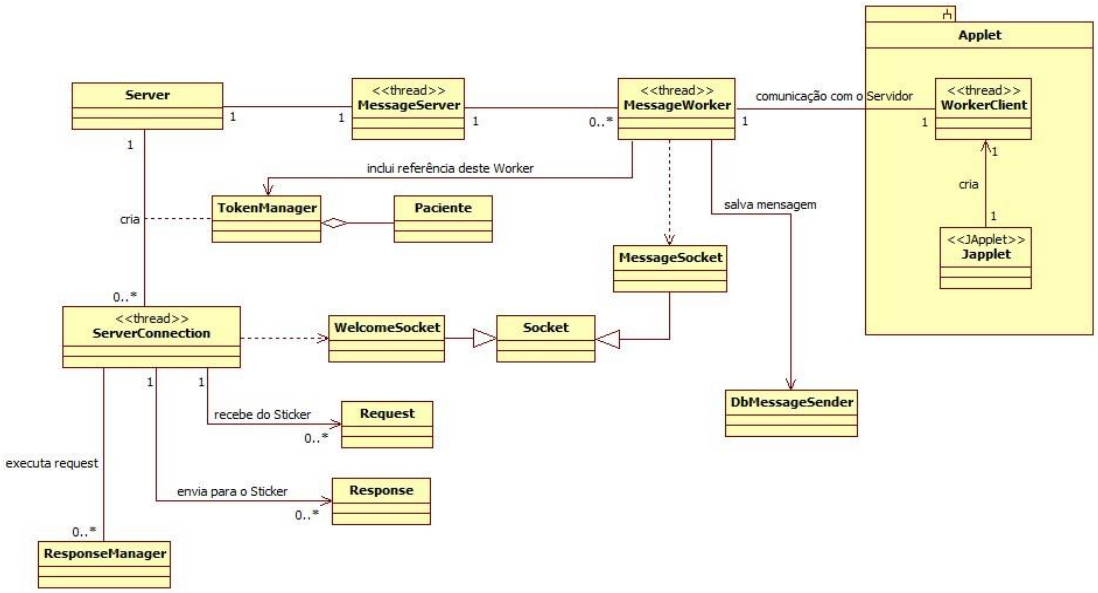


Figura 9. Estrutura do Servidor

A classe Server também é encarregada da criação do servidor de mensagens (uma *thread* especificamente utilizada para esta finalidade), aguardando o contato do Applet de Mensagens (também representado na Figura 9).

Um objeto da classe *TokenManager* é criado uma vez e armazena os dados de cada conexão com um cliente *Sticker*. Nele, ficam salvas as informações de identificação do paciente, a referência da conexão com o Sticker e informações da classe *MessageWorker* (encarregada de gerenciar a conexão com o Applet).

As classes *MessageWorker* e *MessageServer*, e as classes diretamente relacionadas a elas suportam a interação entre o Applet de Mensagens e o *HHS Sticker* (que, de certa forma passa a fazer papel de servidor). Quando o Applet de Mensagens estabelece uma conexão com a *MessageWorker* e o operador seleciona a opção de verificar o status do *Sticker*, esta classe primeiro verifica no *TokenManager* se existe algum registro de conexão na tabela referente ao paciente buscado. Se existir, o servidor irá enviar a mensagem para saber se o *Sticker* ainda está ativo ou não. Se o registro não existir na *TokenManager,* a mensagem de verificação de status não será enviada e uma exceção é lançada. Uma vez feita a verificação, o operador pode enviar uma mensagem de texto para o *HHS Sticker*.

Observa-se que esta parte do módulo Servidor também é importante na integração da aplicação com o sistema HHS. Primeiramente, apenas usuários cadastrados no banco de dados do sistema podem enviar mensagem para um *HHS Sticker* e, para isso, também precisam consultar a identificação do paciente no sistema. Além disso, as funções de agendamento e acionamento do mecanismo de envio de mensagens podem ser integradas ao sistema, sem a necessidade de um módulo separado para isso. A abordagem de se desenvolver um módulo dedicado a isso também se justifica pela independência na realização de testes de operação.

Comunicação entre os elementos

A Figura 10 detalha a comunicação entre o cliente e o servidor. O usuário (paciente) solicita uma informação através da interface do *Sticker* (1) e a página do *Sticker* que estiver ativa no momento realiza o pedido da informação desejada e se registra em uma lista de espera para aguardar a devida resposta. A informação, então, é encapsulada em um objeto *Request* e enviada ao servidor (2). Este recebe esta requisição, busca as informações no banco de dados existente (3), processa a resposta e a encapsula em um objeto *Response*, logo depois o envia ao *Sticker* (4), que sempre espera receber objetos deste tipo. Então, assim que a resposta chega até a entrada do *Sticker* (5), a página anteriormente registrada é notificada e mostra na tela o conteúdo solicitado (6).

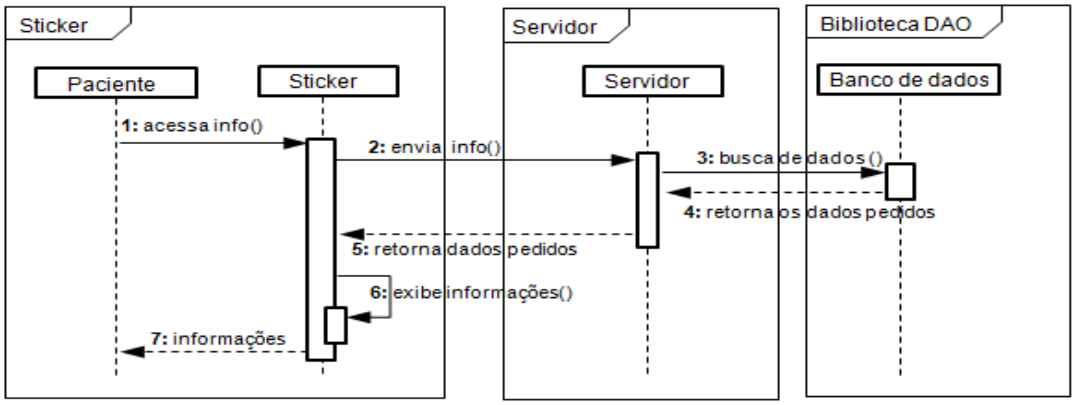


Figura 10. Comunicação Sticker-Servidor

No diagrama da Figura 11 é mostrada a comunicação da aplicação que permite o envio de mensagens e notificações para o *Sticker* (2), repassadas pelo servidor (3) invertendo os papéis cliente-servidor (4). Antes do envio da mensagem é possível verificar se o *Sticker* está em execução (8 a 12). Desta forma, é também esperado que o paciente receba a notificação e confirme que recebeu a mesma, indicando, por exemplo, que ele ou ela tem ciência de que deve realizar uma medida de pressão arterial ou ingerir um medicamento.

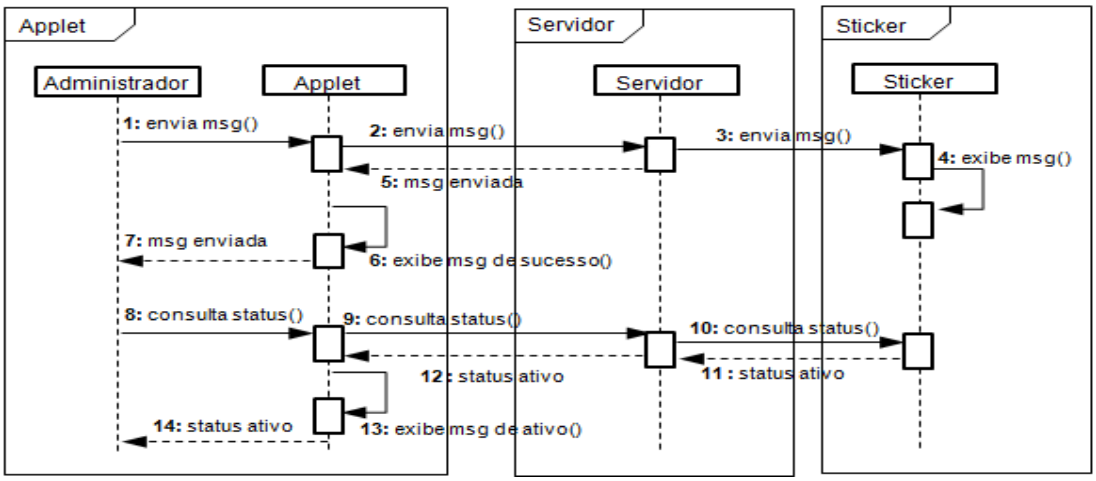


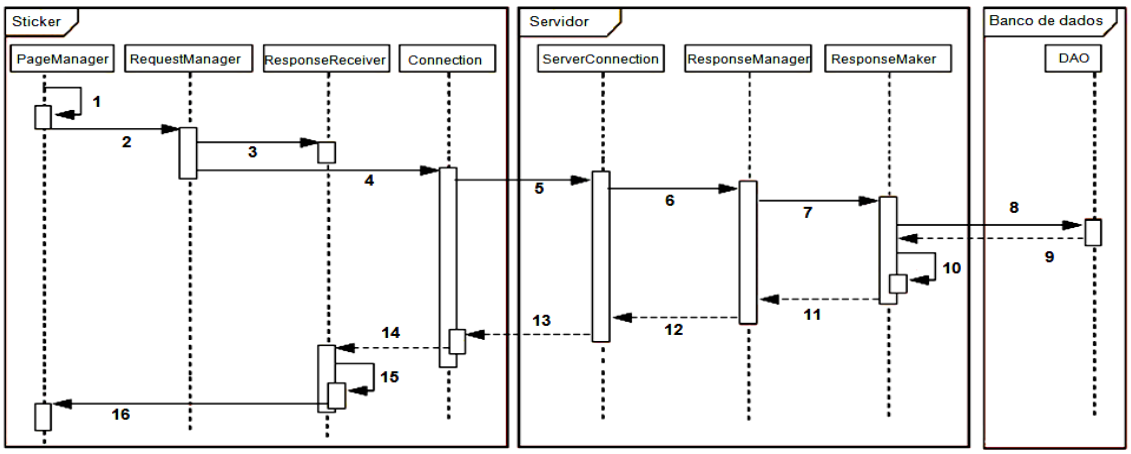
Figura 11. Comunicação Notificação e Status do Sticker

Etapas de requisição/resposta discutidas anteriormente são padronizadas, assim como, no diagrama da Figura 12, para requisição de *login*. Cada objeto que requisita uma informação ao Servidor é do tipo *PageManager* e deve implementar a interface *IResponseListener*, para que todos tenham o método *responseHasBeenReceived* (*Response response*). O Servidor está sempre à espera de um objeto do tipo *Request* e o *Sticker* está sempre à espera de um objeto do tipo *Response*. Os objetos dEsses tipos têm a indicação no próprio nome, por exemplo, um objeto de requisição de *login* se chama *LoginRequest*, assim como o objeto de resposta equivalente se chama *LoginResponse*.

Por exemplo, no primeiro contato é enviado ao Servidor uma requisição de *login* (contendo o nome de usuário e senha) e o cliente recebe como retorno um *LoginResponse*, contendo um *token* identificador daquele paciente e algumas informações sobre suas atividades, que vão preencher a primeira página do *Sticker*. Nos contatos seguintes o *token* é enviado para fazer a validação do paciente no sistema, sem a necessidade de reenvio de nome e senha, e a informação pedida só é respondida se a houver uma validação com sucesso. Seguindo a Figura 12, temos:

1. A classe *MainPage* solicita ao usuário seu nome e senha de acesso ao sistema e cria um objeto *LoginRequest* com as informações (1) e passa este para o *RequestManager* (2). Neste momento, é criado o socket de conexão com o Servidor.
2. O *RequestManager* chama o método *addRequestReference(int requestOp_code, IResponseListener listener)* do objeto *ResponseManager* para salvar a referência da classe que o chamou (3), no caso a *MainPage*, juntamente com um identificador do tipo de requisição. Depois ele chama o método *sendRequest(Request request)* da classe *Connection* para enviar a requisição ao servidor (4).
3. A classe *Connection* envia o objeto *request* pelo *socket* e aguarda a resposta (5).
4. No *Servidor*, a requisição é recebida e cria uma nova thread (*ServerConnection*) e passa o controle da conexão para a mesma, que assume o controle da comunicação com este *Sticker*.
5. Então a *ServerConnection* recebe a requisição, a transfere para o *ResponseManager* e aguarda a resposta (6).

6. *ResponseManager* identifica o tipo da requisição, delega a responsabilidade de processar a resposta à classe *ResponseMaker* e fica aguardando a resposta (7).
7. Como a requisição é de *login*, primeiramente o *ResponseMaker* usa a classe *TokenManager* para criar e armazenar um *token* de identificação do paciente, em seguida utiliza chamadas de métodos dos objetos do pacote *hhs.dao,* usado por todo o sistema HHS, para buscar informações no sistema existente (8) e com o retorno (9) ele cria um objeto de resposta, contendo o *token* (10), e retorna para quem o chamou, o *ResponseManager* (11). Quando não é uma requisição de *login*, a *TokenManager* é chamada (no *ResponseManager*) apenas para validar o usuário, buscando o objeto paciente através do *token* recebido, e na hora de retornar a resposta não é necessário enviar o *token* novamente, pois o cliente já sabe esta informação.
8. *ResponseManager* recebe o objeto de resposta e repassa para a classe *ServerConnection* (12), que envia direto para a classe Connection do *Sticker* que está esperando (13).
9. Ao receber a resposta, a Connection usa a *ResponseReceiver* (14) para notificar ao *listener* (no caso a *MainPage*) que a resposta foi recebida (15).
10. *MainPage* é notificada da resposta (16) e assim que isso acontece ela libera o acesso ao sistema para o paciente, que já pode ver as informações das suas atividades na primeira aba do *Sticker*.



1: cria um objeto de requisição( ); **9:** retorna dados;

2: faz requisição( ); **10:** cria um objeto de resposta();

3: salva referencia(); **11 e 12:** retorna resposta;

4: solicita envio da requisição(); **13:** envia resposta();

5 e 6: envia e transfere requisição(), respectivamente; **14:** repassa resposta();

7: [tokenValido=true]: solicita processamento da requisição(); **15:** busca a referência();

8: busca informações no sistema existente(); **16:** notifica e passa a resposta recebida();

Figura 12. Diagrama de interação request/response

5. Protótipo do *HHS Sticker*

A apresentação do *HHS Sticker* está dividida em três abas e uma janela (complemento das informações das abas), além das duas páginas de ajuda (auxílio das teclas do controle remoto) e copyright (autoria e versão do aplicativo). Esta apresentação segue o padrão visual recomendado para um *Sticker* (Seção 3.3). Cada aba possui conteúdo carregado dinamicamente com informações relativas ao paciente "logado".

Primeiramente, para ter acesso ao sistema, o paciente precisará fazer o *login* digitando seu nome de usuário e senha previamente cadastrados (Figura 13). Caso algum dado esteja errado, será exibida uma mensagem de erro pedindo para que ele tente novamente. Se os dados inseridos estiverem corretos, o *login* será feito com sucesso e a primeira aba aparecerá. Esta primeira aba contém as informações retiradas do *loginResponse* (a resposta de requisição de *login*), com as próximas atividades do paciente.

Figura 13. Realizando o *login*

Aba "Principal" (Figura 14). Contém informações retiradas do *loginResponse* (a resposta de requisição), aparece após a realização do *login* e contém uma caixa de notificação, além de uma lista em ordem cronológica das próximas atividades do paciente com seus respectivos detalhes: data, hora, duração e descrição.

Figura 14. Tela da Aba Principal

Aba **"Plano de Cuidados"** (Figura 15). Apresenta uma lista com todos os planos de cuidados, assim como seus respectivos detalhes: data, hora de início e término, descrição e atividades relativas.

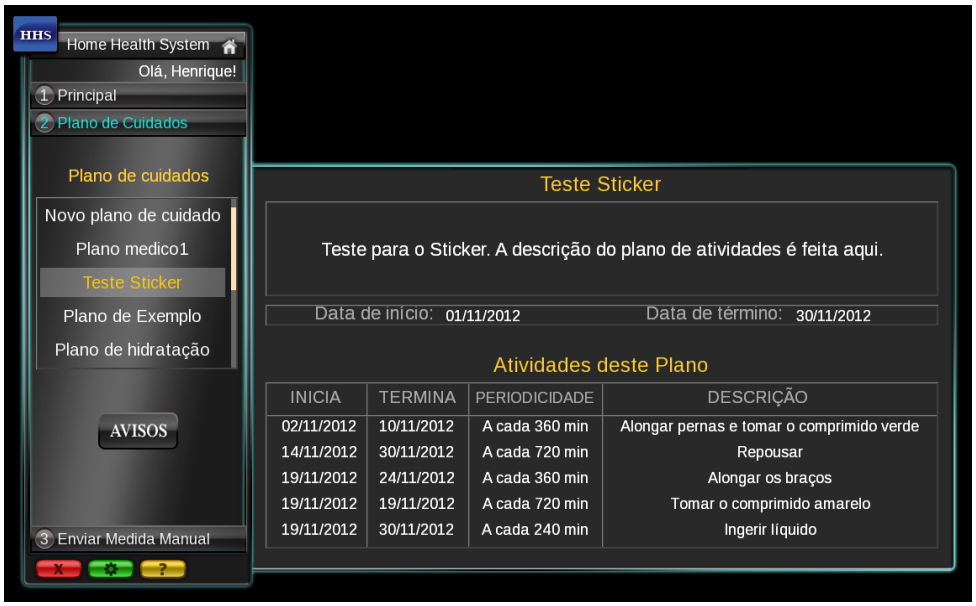


Figura 15. Tela da Aba Plano de Cuidados

Aba **"Enviar Medida Manual"**. Permite ao paciente enviar medidas aferidas manualmente. Existem duas opções de medida com janelas correspondentes conforme Figura 16: "Usar módulo de IA" – utilizada para envio de medida com análise do modo de inteligência artificial (apenas para pressão arterial) – e "Relacionar medida EVA" – utilizada para adicionar a escala de dor, juntamente com a medida. Ambas podem ser marcadas ao mesmo tempo, porém a janela ativa será equivalente a quando a opção "Usar módulo de IA" estiver marcada.

Figura 16. Opção "Usar módulo de IA" e "Relacionar medida EVA"

As "**Mensagens de Alerta"** (Figura 17) são enviadas por um administrador para informar o paciente sobre algum evento do plano de cuidados. Se o *Sticker* estiver aberto na hora do envio, a mensagem será mostrada na tela ativa, caso contrário, será armazenada com o *status*=0 e assim que o *Sticker* estiver ativo, serão apresentadas ao alterar automaticamente o *status* para 1.

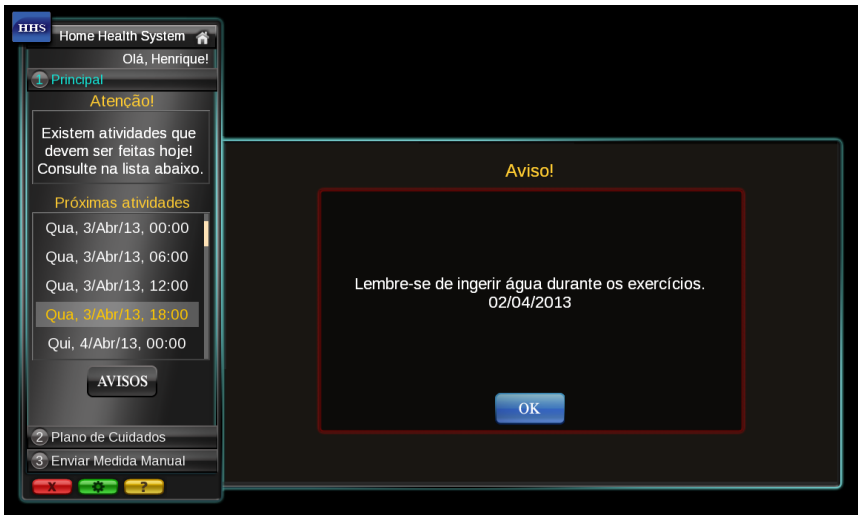


Figura 17. Sticker recebendo mensagem de alerta

5.2 O Applet de Envio de Mensagens

Para facilitar a rápida integração do sistema de envio de mensagens foi desenvolvido um módulo na forma de um Applet para que administradores, com conta criada no sistema HHS, pudessem enviar mensagens aos pacientes (Figura 18) para notificá-los de quaisquer informações inerentes ao tratamento.

A mesma interface também permite verificar se o Sticker está ativo (botão "STATUS DO STICKER") no momento. Este último serviço é disponibilizado pelo

HHS Sticker, para que a aplicação possa verificar se a TVDi está ligada, antes de enviar a notificação, ou buscar uma outra alternativa usando os serviços de contexto.

Para enviar a mensagem o administrador deverá localizar o nome do paciente, selecioná-lo em uma lista para o campo "Para", escrever a mensagem e clicar em "ENVIAR".

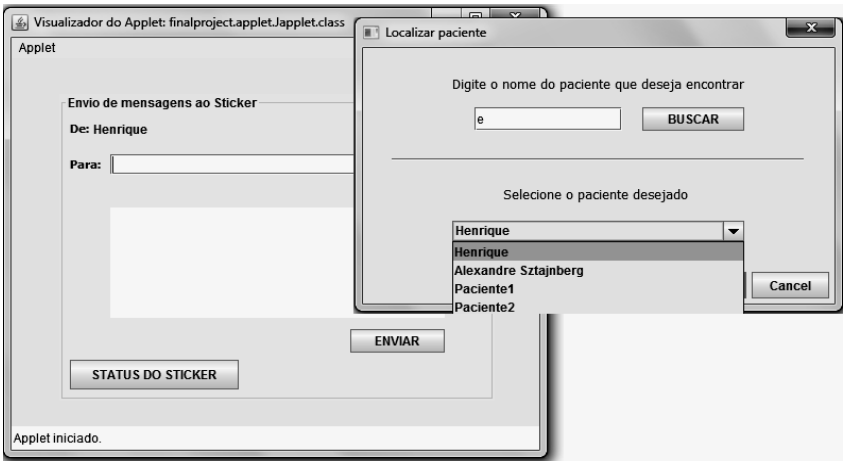


Figura 18. Tela de mensagens do applet

5.3 Detalhes de implementação

Todo desenvolvimento do *HHS Sticker* foi feito com a linguagem Java, segundo o padrão GingaJ. Conforme discutido na Seção 3.3 a classe *Main* implementa a interface *Xlet* e obtém e inicializa uma referência do objeto *Sticker,* no método *initSticker()*, utilizando o pacote *com.tqtvd.stickercenter* (Figura 19). Os métodos obrigatórios da interface *Xlet* são também implementados.

```
// Bibliotecas necessárias para o Xlet e para o Sticker
import javax.microedition.xlet.*;
import com.tqtvd.stickercenter.*;
[...]
public class Main implements Xlet{           // A implementação de Xlet
  public void initSticker(){                // A implementação do sticker
    stick = Sticker.getInstance();            // Instanciando o Sticker
    [...]
    stick.setWindowComponent(windowComponent);  //Definindo components

     mainPage = new MainPage();

     stick.setMainComponent(mainPage);

  }
  [...]
```

Figura 19. Trecho da classe *Main* (implementação do *Xlet)*

A estrutura do *Xlet*, seu ciclo de vida e a estrutura do *Sticker* obedecem a padrões de projeto estabelecidos. Entretanto, algumas soluções adotadas no código do *Sticker* e do Servidor merecem destaque, entre elas as estruturas de dados usadas para representar requisições de pacientes repassadas como padrão entre os objetos, bem como alguns aspectos de comunicação via *sockets* e seus empregos no *Xlet*.

Na Figura 20, destacam-se algumas partes da comunicação do *sticker* com o servidor, descrita na Seção 4.2, que exige a comunicação através de *sockets*. A iniciativa de requisição parte do objeto do tipo *PageManager*, ao criar um objeto de requisição (linha 1) e acionar o *RequestManager* enviando a requisição e a sua própria referência como parâmetros (linha 2). No método *requestReceiver* do *RequestManager* esta referência é registrada para encaminhar a futura resposta (linha 4) e a requisição é passada para a Connection, que a envia para o servidor (linha 5). Ao ser chamado, o método *addRequestReference* registra o código da requisição e a referência de quem espera a resposta (linha 8). O objeto da classe Connection, por sua vez recebe a requisição e a envia para o servidor através do *socket* (linha 11).

No lado do Servidor, a classe *ServerConnection* recebe a mensagem, também por um *socket*, já convertendo de Object para a classe *Request* esperado (linha 12), salva sua referência e em seguida encaminha a mesma para execução, chamando o método *executeRequest* (linha 14). No *ResponseManager*, o método *executeRequest* verifica a validade do token e o tipo da requisição (linhas 17 e 18), e aciona a rotina correspondente, que chama métodos do *ResponseMaker* para tratá-la e montar a resposta (linhas 19 a 21) A ação do *ResponseMaker* requer interação com o sistema HHS e será destacada adiante.

Uma vez obtida a resposta, o *ServerConnection* envia a resposta pelo socket e com isso, pelo lado do Servidor, a requisição está concluída (linha 24).

Novamente ao Sticker, *Connection* recebe a resposta pelo *socket* ativo (linha 25) e a passa para o *ResponseReceiver* (linha 26), que por sua vez notifica, chamando o método *notifyListener* (linha 27). No método *notifyListener* o *PageManager* que está aguardando a resposta é identificado (linha 29) e o mesmo é notificado chamando o método de *call-back listWillRecResponse.responseHasBeenReceived(rspns)* (linha 30), completando a interação.

```
// PageManager
1    Request myPlanRequest = new MyPlanRequest(3,CurrentLogin.getToken());
2    RequestManager.getReference().requestReceiver(myPlanRequest, reference);

// RequestManager
3   public void requestReceiver(Request request, IResponseListener listener){
4     ResponseReceiver.getReference().
                    addRequestReference(request.getOperationCode(),listener);
5     Connection.getReference().sendRequest(request);
6   }

// ResponseReceiver
7   public void addRequestReference(int reqOpCode, IResponseListener lstnr){
8     this.responseList.put(new Integer(reqOpCode), listener);
9   }

// Connection
10  public void sendRequest(Request request){
11    out.writeObject(request);

// ServerConnection
12   objectRequest = (Request) in.readObject();
13   objectResponse = ResponseManager.getReference().
14   executeRequest(objectRequest);
```

```
// ResponseManager
15  public Response executeRequest(Request request){
16    Response response = null;
17    if (TokenManager.getReference().checkUser(request.getToken()){
18       switch(request.getOperationCode()) {
      […]
19         case 3: {
20           response = ResponseMaker.getReference().doMyPlanSelect(
                    TokenManager.getReference().getUser(request.getToken()));
21           response.setOperationCode(request.getOperationCode());
22      }
        [...]
23    return response;}

// ServerConnection
24    if (objectResponse != null) out.writeObject(objectResponse);

// Connection
25    Response objectResponse = (Response) in.readObject();
26    ResponseReceiver.getReference().responseReceiver(objectResponse);

// ResponseReceiver
27  public void responseReceiver(Response rspns){ notifyListener(rspns);}

28  public void notifyListener(Response rspns){
29    IResponseListener listWillRecResponse = (IResponseListener)
                  responseList.get (new Integer(rspns.getOperationCode()));
30    listWillRecResponse.responseHasBeenReceived(rspns);
```

Figura 20. Trecho de código (cliente-servidor) do processo de requisição e resposta

Para discutir a interação do Servidor com o sistema HHS, no *ResponseMaker*, e seguindo o exemplo de requisição já utilizado, o acesso ao banco é requisitado para a obtenção do Plano de Cuidados (método *doMyPalnSelect*), conforme Figura 21. Como o paciente já está identificado no sistema, a integração de chamadas de métodos às bibliotecas DAO é direta. A referência ao paciente é passada ao método pela variável *currentPatient* (linha 1).

```
1   public MyPlanResponse doMyPlanSelect(Paciente currentPatient){

2     myTreatment = new TratamentoDao().
                                    pegarTratamentos(currentPatient);
3     myActivities = new AtividadeDao().
                       pegarAtividadesTratamento(myTreatment.get(j));
4     myActivities.set(i, TransformToActivity.makeTransformation(
                      myActivities.get(i)));
5     myPlanActivities.put(
                     new Integer(((Tratamento)myTreatment.get(j)).
                                    getIdTratamento()), myActivities);
6     myTreatment.set(i, TransformToTreatment.makeTransformation(
            (Tratamento)myTreatment.get(i)));
7     myPlanResponse = new MyPlanResponse
                                    (myTreatment, myPlanActivities);
8     return myPlanResponse;
    }
```

Figura 21. Trecho de código referente à execução da requisição do cliente

No caso da obtenção do Plano de Cuidados do HHS, contendo as rotinas prescritas pelo médico para o paciente, as estruturas de objetos são listas varridas iterativamente (os laços de iteração foram omitidos por simplicidade). Primeiro a referência ao tratamento é obtida (linha 2) e em seguida para cada tratamento as atividades são também obtidas (linha 3). Antes de retornar as informações, os objetos são "transformados" para simplificar os mesmos para atender a compatibilidade da versão 1.3 do Java executando no *set-top box* (linhas 4 e 6). Em seguida um resultado é criado com as listas de tratamento e atividades (linha 7), e finalmente este resultado é retornado ao chamador (linha 8).

6. Trabalhos relacionados

O livro *Digital Health Information for the Consumer: Evidence and Policy Implications* [Nicholas, 2007] avalia várias tecnologias para e-Health, dedicando o Capítulo 5 à TV Digital Interativa. Foram avaliados vários aspectos da tecnologia, incluindo o impacto para o paciente, para o médico e para os serviços de saúde já estabelecidos. Quatro consórcios apresentaram projetos piloto para esta avaliação: Communicopia, FlextechTelewest, Living Health e dktv (A Different Kind of Television). Basicamente o serviço oferecido nos quatro projetos era o acesso a informações específicas de áreas da saúde. A interatividade não foi usada como no *HHS Sticker*, onde existe interatividade do paciente com o médico e a equipe de monitoramento.

Uma proposta de uso da TV digital com interatividade para facilitar a comunicação paciente-médico é apresentada em [Niiranen, 2002], desenvolvido para o mercado finlandês. Nessa proposta, dados do paciente são salvos em arquivos XSL (*Extensible Stylesheet Language*) e apresentados na televisão em formato XML (*eXtensible Markup Language*). Adicionalmente existe um servidor de páginas WEB que recebe, via método POST, as medições realizadas e junto com outros dados do paciente através de uma comunicação cliente-servidor TCP/IP sobre PPP (Point-to-Point Protocol). A proposta ainda prevê a integração entre dispositivos de medição, usados nos tratamentos residenciais, e a TV digital, no envio dos dados. No *HHS Sticker*, a integração de dispositivos sem fio foi avaliada, mas dependeria de acionadores e código de terceiros que não estavam disponíveis.

O projeto PANACEIA-iTV [Prentza, 2005], na mesma linha adotada para o *HHS Sticker*, é um sistema que incentiva o paciente a monitorar sua saúde e acessar informações e suporte à saúde. O sistema foi testado com pacientes portadores de *Adult Congenital Heart Disease* (ACHD) grave no *Royal Brompton Hospital*, sendo os mesmos, monitorados via satélite, em suas residências, utilizando a tecnologia DVB-S. Um experimento com o sistema é relatado em [Karagiannis, 2006], envolvendo 9 pacientes hospitalizados e 12 pacientes não especializados. Após um rápido treinamento, os pacientes foram orientados a realizar medidas de pressão arterial, ECG, SpO2 e peso, e enviá-las através do sistema. A avaliação mostrou que o uso do sistema foi positivo para a maioria dos pacientes.

O projeto *MHPHomecare* [Angius, 2008] tem elementos comuns com o *HHS Sticker*. O sistema utiliza o padrão DVB-T (Digital Video Broadcasting – Terrestrial) para transporte de mídias no formato MPEG e executa uma aplicação Java, também estruturada a partir de um *Xlet* em um *set-top box* compatível. A aplicação desenvolvida

tem o objetivo de receber medidas fisiológicas através da interface de infravermelho do *set-top box*. Para isso foi desenvolvido um hardware externo microcontrolado que realiza a captura dos sinais biológicos de interesse e os transmite para o *set-top box* via interface de infravermelho. Em seguida o *Xlet* transmite os dados para um centro de cuidados remoto por uma conexão à Internet. O *Xlet* também provê acesso gráfico a outros elementos do sistema. Duas outras características devem ser mencionadas. O projeto utiliza um *smartcard* com informações personalizadas para o paciente, que facilita a configuração de parâmetros específicos, inserido na leitora do *set-top box*, lida pelo Xlet. Outra característica é o envio da aplicação para o *set-top box* pelo canal de difusão de *sofware* do padrão DVB-T. Com isso a atualização do software, quando necessária, é automática. Não há necessidade de download da aplicação de uma página web, como no *HHS Sticker*. Por outro lado, o envio de aplicações por um canal de difusão requer uma infraestrutura relativamente complexa e cara [RCA].

O projeto Med-Reminder [Stojmenova, 2013] utiliza a TVDigital para enviar notificações e lembretes. A preocupação com a interface gráfica, considerando que o público-alvo seria a população de 3ª idade, além da garantia de recebimento e leitura da notificação enviada pelo administrador (médico) é semelhante às funções do *Applet* de Mensagens do HHS *Sticker*. No entanto, o projeto [Stojmenova, 2013] apresenta problemas relativos à dificuldade em inserir lembretes usando o controle remoto, pela quantidade de informação necessária durante a criação. No *HHS Sticker* os lembretes são sempre enviados pelo médico, monitor ou pelo sistema de Plano de Cuidados e não pelo paciente. O paciente só precisa entrar com poucas informações, basicamente as medidas fisiológicas.

Em [Oliveira, 2009], os autores apresentam a proposta de uma arquitetura para monitoramento de pacientes utilizando o padrão Ginga, chamado DIGA Saúde TV, elencando características importantes para este tipo de sistema, similares ao HHS e ao *HHS Sticker*. Entretanto, não apresentam um protótipo consistente, nem realizam simulações com o sistema de comunicação ou de integração de elementos.

Entre as propostas relacionadas, Motiva da Philips é uma das mais abrangentes. Motiva é uma proposta da A Philips [Philips, 2013] para um sistema de telessaúde fortemente apoiado por TVDi. Entre os objetivos do Motiva estão um mecanismo de informação e lembretes baseados em regras, integrando suporte prático para o paciente aderir ao seu plano de cuidados. Através de aplicações para TV Digital e serviços de suporte, o sistema oferece informações educativas, sistema de mensagens personalizado, questionários de motivação e o envio de dados fisiológicos monitorados segundo recomendação médica. Os aspectos técnicos da implementação das aplicações de vídeo não são discutidos. Mas, diferentemente dos trabalhos com foco na implementação, como o *HHS Sticker*, a Philips oferece incentivo para a realização de testes clínicos. Destacam-se dois artigos do mesmo grupo, ambos [Domingo, 2013] e [Domigos, 2013], realizados como parte do CARME (*Catalan Remote Managenet Evaluation Study*), no Instituto Catalão de Saúde. Os dois testes realizados durante um ano dentro de padrões de rigor e ética, envolveram mais de 300 pacientes, com doença cardíaca. Os autores discutem todos os detalhes de seus resultados, como: a verificação da diminuição de reinternações, aspecto bastante positivo; a interação simples dos pacientes com o sistema; e até mesmo a diminuição da aderência às rotinas de envio de medidas fisiológicas através do sistema.

7. Conclusão

A arquitetura do *HHS Sticker* foi desenvolvida para ser integrada a um conjunto de módulos de clientes de contexto do sistema HHS (*Home Health Subsystem*). Além de permitir a exibição de conteúdo do acompanhamento médico dos pacientes e a transmissão de medidas fisiológicas, ele atua como sensor, fornecendo a informação de seu estado (ligado ou desligado), bem como qualquer outra informação que possa ser obtida dentro do ambiente GingaJ. Num cenário típico de uso do HHS é possível identificar se o paciente está próximo ou localizado no mesmo cômodo em que a TV se encontra e se a mesma está ligada (e o *HHS Sticker* devidamente conectado ao servidor). Assim, se o sistema ou o médico pretende enviar uma mensagem a este paciente neste instante, ele pode decidir fazer isso por meio da TV (através do Sticker), pois a probabilidade que ela seja vista é muito grande.

Um das características da aplicação é a simplicidade no manuseio e na integração com o *HHS Applet*, somados ao visual atrativo.

O fator custo também pode ser considerado. Segundo dados IBGE 2011, apenas 4,42% da população acima de 60 anos têm computador com qualquer tipo de acesso à Internet em sua residência. Por outro lado existem metas governamentais para desativação da televisão analógica, com planos de incentivo à aquisição de TV digital por parte da população. Ainda, entre as metas, a indústria está sendo incentivada a implantar canais de retorno e interagir através de infraestrutura que utilize o espectro de frequência VHF brasileiro (54 a 88Mhz e 174 a 216Mhz). O uso do canal de retorno via VHF tornará possível o uso de aplicações como o HHS Sticker até mesmo para áreas rurais, distantes e carentes da população.

O trabalho de integração *HHS Sticker* ao sistema HHS está em desenvolvimento. A interface para envio de notificações ao paciente, desenvolvida na forma do *Applet* de Mensagens será incorporada à interface com os médicos, bem como integrada ao sistema de gerenciamento do Plano de Cuidados. Outro ponto de aprimoramento é relacionado ao aspecto visual do HHS Sticker. Uma avaliação de usabilidade, com pacientes idosos, permitirá verificar que aspectos devam ser melhorados, desde a quantidade de informações apresentadas em cada aba, até o tamanho das letras exibidas.

Agradecimentos. Agradecemos à Faperj pelo apoio financeiro na aquisição dos aparelhos set-top box da TQTVD. Agradecemos à TQTVD pelo apoio técnico na elaboração deste projeto.

Referências

Angius G, Pani D, Raffo L, Randaccio P, Seruis S. "A tele-home care system exploiting the DVB-T technology and MHP", Methods Inf Med, Schattauer GmbH, Vol. 47, No. 3, pp. 223-228, 2008; doi:10.3414/ME9114

AstroBox. TOTVS, 2011. https://www.astrodevnet.com/AstroDevNet/ restrict/ astrobox_sobre.html, [Acesso em: Jan/2013].

AstroTV, TOTVS 2012. http://www.tqtvd.com/novo/br/astrotv-interna.html#tv1 [Acesso em: 07/2013]

Batista, Carlos Eduardo. “TV Digital - Java na sala de estar”. 2007. Revista Mundo Java, número 17, ano III – Editora Mundo.

Domingo M., Lupón J, González B, Crespo E, López R, Ramos A, et al. “Noninvasive remote telemonitoring for stable patients with heart failure: effect on number of hospitalizations, days in hospital, and quality of life”. CARME (CAtalan Remote Management Evaluation) Study. Rev Esp Cardiol, in press, doi:10.1016/j.recesp.2010.10.032.

Domingo M., Lupón J, González B, Crespo E, López R, Ramos A, et al, “Evaluation of a telemedicine system for heart failure patients: Feasibility, acceptance rate, satisfaction and changes in patient behavior”. Results from the CARME (CAtalan RemoteManagement Evaluation) study European Journal of Cardiovascular Nursing (2011), doi:10.1016/j.ejcnurse.2011.02.003

D-Link, RECEPTOR D-LINK (2011). http://www.dlink.com.br/produtos-detalhes/items/dtb-331.html [Acesso: 07/2013]

Enc, 2009. Encerramento TV Analógica. http://www.brasil.gov.br/sobre/ciencia-e-tecnologia/industria-eletronica-digital/tv-digital, [Acesso: Fev/2014]

Esp, 2014. Especificação Ginga. http://gingadf.com.br/blogGinga/ javaDtv Api/ javaDoc_V1.3, [Acesso: Fev/2014]

Ginga, 2013. Sobre o Ginga. http://www.ginga.org.br/pt-br [Acesso: 07/2013]

Grupos de Trabalho na SBTVD, 2011. http://sbtvdbcc.wordpress.com/2011/10/05/ grupos-de-trabalho-na-sbtvd/ [Acesso em: Abr/2013].

[HIST 2013]. ”História da TV Digital no Brasil”. http://www.dtv.org.br/informacoes-tecnicas/historia-da-tv-digital-no-brasil/ [Acesso: 07/2013]

IBGE, 2011. “Pesquisa Nacional por Amostra de Domicílios 21/09/12” http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/00000010135709212012572220530659.pdf [Acesso: 07/2013]

IPTV, 2013. http://www.ip.tv/iptv_site/ptb/html/client.html [Acesso: 07/2013]

ITU, 2013. ITU-T in brief. http://www.itu.int/en/ITU-T/about/Pages/default.aspx [Acesso: 07/2013]

Karagiannis, George E., Stamatopoulos, Vasileios G., George Roussos, Takis Kotis, Michael A Gatzoulis, "Health and lifestyle management via interactive TV in patients with severe chronic cardiovascular diseases" *J Telemed Telecare July 1, 2006 Vol 12*: pp. *17*-19,Telemed Telecare July 1, 2006 vol. 12 no. suppl 1 17-19 SAGE Journals, , UK. doi: 10.1258/135763306777978489J

Kulesza, R. “API JavaTV-Criando e Controlando Aplicações Java no Ginga”, 2009. http://www.tvdi.inf.br/site/artigos/Ginga-J/Desenvolvimento Ginga-J, JavaDTV, OpenGinga - Parte 2 – KULESZA, FERREIRA.pdf [Acesso: 07/2013]

Kulesza, R. e Ferreira, J. “Desenvolvimento Ginga-J – JavaDTV – OpenGinga”, 27/06/09. http://www.tvdi.inf.br/site/artigos/Ginga-J/Desenvolvimento Ginga-J, JavaDTV, OpenGinga - Parte 1 – KULESZA, FERREIRA.pdf [Acesso: 07/2013]

Macedo, E. L. C. ; Ferreira, D. B. ; Lemos, G. M. R. ; Sztajnberg, A. ; Loques, Orlando Gomes. “Suporte para Coleta e Persistência de Dados de Contexto em um Sistema de Monitoramento Domiciliar Remoto de Pacientes”. In: Computer on the Beach 2011, 2011, Florianópolis. Proceedings of the Computer on the Beach 2011, 2011.

Midlet Profile”. http://docs.oracle.com/javame/config/cldc/ref-impl/ midp2.0/jsr118/javax/microedition/midlet/MIDlet.html[Acesso: 07/2013]

Morris, S. “An Introduction To Xlets” [Figura: The state diagram for an Xlet] http://www.interactivetvweb.org/tutorials/javatv/xlet_intro [Acesso: 07/2013]

Nicholas, D., Huntington P., Jamali H., "Digital Health Information for the Consumer: Evidence and Policy Implications", Chapter 5 - Health Digital Interactive Television (DiTV), Ashgate Publishing, Ltd., England, 2007.

Niiranen, S., Lamminen, H., Mattila, H., Niemi, K., Kalli, S. Personal health care services through digital television – Computer Methods and Programs in Biomedicine 68 (2002) 249-259 - 25/01/2002

Oliveira, M.; Cunha, P.R.F.; da Silva Santos, M.E.; Bezerra, J.C.C. "Implementing home care application in Brazilian Digital TV", Global *Information Infrastructure Symposium, GIIS '09*, pp. 1 - 7, 2009. 10.1109/GIIS.2009.5307042

Philips HealthCare, "Philips Motiva". http://www.healthcare.philips.com/ br_pt/ products/ telehealth/ products/ motiva.wpd [Acesso: Fev/2014]

Prentza A., Stavroula Maglavera, George Stalidis, Eleni Sakka, Irini Lekka, Pantelis A. Angelidis, Lefteris Leondaridis, Nicos Maglaveras, and Dimitris Koutsouris, “Cost-effective health services for interactive lifestyle management: the PANACEIA-iTV and the e-Vital concepts” Journal of Telecommunications and Information Technology, April 2005: pp. 49-58. ISSN 1509-4553

RCA, "EITV Playout Professional", http://www.rcasoft.com.br/playout.php [Acesso em: Fev/2014]

StickerCenter da TOTVS. https://www.stickercenter.com.br/StickerWeb/pt_BR/ index.html [Acesso em: Abr/2013].

Stojmenova, E., Debevc, M., Zebec, L., Imperl, B., Cunha, P. Assisted living solutions for the elderly through interactive TV - Multimed Tools Appl (2013) 66:115–129 - http://scienceindex.com/

Sztajnberg, A., Rodrigues, A. L. B., Bezerra, L. N., Loques, O. G., Copetti, A., & Carvalho, S. (2009). “Applying context-aware techniques to design remote assisted living applications. International Journal of Functional Informatics and Personalized Medicine”, 2(4), 358-378.

The World Bank Group, "Brazil’s digital TV system delivers health, education and jobs at the touch of a remote control", FEATURE STORY, May 24, 2013. http://www.worldbank.org/en/news/feature/2013/05/22/Brazil-digital-TV-system-4D-benefits-poor, [Acesso: Jul/2013]

TQTVD. TQTVD (Sobre), 2012. http://www.tqtvd.com/novo/br/sobre.html [Acesso: 07/2013]

Ubuntu. “O que é o Ubuntu?”, 2010. http://www.ubuntu-br.org/ [Acesso: 07/2013]

V-Box, 2013. Virtual Box. https://www.virtualbox.org/ [Acesso: 07/2013]

World Bank. 2013. *Brasil 4D : Estudo de impacto socioeconômico sobre a TV digital pública interativa*. Washington DC; World Bank. http://documents. Worldbank .org/curated/en/2013/08/18203867/brazil-4d-study-socioeconomic-impact-digital-tv-interactive-public-brasil-4d-estudo-de-impacto-socioecon%C3%B4mico-sobre-tv-digital-p%C3%BAblica-interativa [Acesso: Out/2013]

Xlet, JSR 217 (Maintenance Release). http://docs.oracle.com/javame/config/cdc/ref-impl/pbp1.1.2/jsr217/javax/microedition/xlet/package-summary.html [Acesso em: Out/2012].

Avaliação e Redução do Tempo de Resposta de Sistemas Web

Victor Marconi Arouca Ribeiro[1], Valdenir Tavares[2], Alexandre Sztajnberg[1, 2, 3]

[1]Bacharelado em Ciência da Computação – (DICC/IME)
[2]Mestrado em Engenharia Eletrônica – (PEL/FEN)
[3]Mestrado em Ciências Computacionais – (CComp/IME)
Universidade do Estado do Rio de Janeiro(UERJ)

victormarconi@gmail.com, valdenir.tavares@oi.com.br, alexszt@uerj.br

Abstract. *The growing availability of Internet access and the intensive use of the web as the infrastructure for distributed business applications require constant attention of the providers to turn positive the user experience when accessing a given site. In this paper variations in the response time of a web system are evaluated, when typical elements and parameters of the supporting infrastructure are modified. The results obtained in the tests were compared and it could be verified which set of interventions, configuration of hardware and software, was more efficient, i.e., had the lowest response time for the submitted requests.*

Resumo. *A crescente disponibilidade de acesso à Internet e o uso intensivo da web como infraestrutura para aplicações comerciais distribuídas requer atenção constante dos provedores para tornar a experiência do usuário positiva quando este estiver acessando determinado* site*. No presente trabalho são avaliadas as variações no tempo de resposta de um sistema web típico quando elementos e parâmetros da infraestrutura que suporta esse sistema são modificados. Os resultados obtidos nos testes foram comparados e, assim pôde ser verificado qual o conjunto de intervenções, configuração de hardware e software, se mostrou mais eficiente, ou seja, obteve o menor tempo de resposta no atendimento às requisições submetidas.*

1. Introdução

A crescente disponibilidade de acesso à Internet e o uso intensivo da web como infraestrutura para aplicações comerciais distribuídas requer um trabalho constante dos provedores para tornar a experiência do usuário positiva quando este estiver acessando determinado *site*. Avaliar o quanto agradável é um site significa satisfazer critérios de usabilidade, aparência, conteúdo e, sobretudo, velocidade de resposta.

A demora para acessar ou recuperar conteúdos de *sites* web é o principal motivo de insatisfação de usuários. Assim, é importante identificar e compreender os fatores que contribuem para a variação do tempo de carga de um site, como também identificar e exercitar as melhores práticas que permitam reduzi-lo. É recorrente a prática da realização de ações de otimização em servidores e banco de dados, o uso de algoritmos mais eficientes, alteração da linguagem de programação ou, até mesmo, a migração de sistemas para equipamentos com maior capacidade de processamento com o objetivo melhorar a velocidade de acesso, ou o tempo de resposta, sites.

Entretanto, tecnicamente, é possível verificar com base no tempo de carga dos componentes de uma página web, que os maiores tempos ocorrem após a geração do documento HTML (no caso de um sistema de informações que gera uma página dinamicamente em resposta a uma requisição) e sua carga no cliente, isto é, após todo o processamento ter ocorrido no *backend*. Desse modo, o maior atraso, cerca de 80%, é creditado aos componentes de *frontend* [Souders 2007].

A partir da constatação de que a maior parte do tempo de uma requisição é gasta no *frontend*, Steve Souders, à época um dos engenheiros do *Yahoo!*, publicou 14 recomendações que consistem em alterar configurações associadas ao *frontend* sem, no entanto, afetar a arquitetura ou a lógica da aplicação [Souders 2007].

No presente trabalho são avaliadas as variações no tempo de resposta de um sistema web padrão quando elementos e parâmetros da infraestrutura que suporta esse sistema são modificados, seja alterando as especificações do hardware da máquina, ou aplicando as recomendações de Souders para a configuração dos vários componentes de software da infraestrutura, e como estas alterações afetam, por sua vez, o tempo de resposta.

Os resultados obtidos nos testes foram comparados e, assim pôde ser verificado qual o conjunto de intervenções, na configuração de hardware e software se mostrou mais eficiente, ou seja, obteve o menor tempo de resposta no atendimento das requisições submetidas. Foi possível, então, traçar um conjunto de recomendações gerais para um serviço web típico.

O restante do artigo está estruturado da seguinte forma: A Seção 2 apresenta a motivação para se preocupar em reduzir a o tempo de resposta de um site. A Seção 3 apresenta os conceitos de funcionamento de um site através dos seus elementos constituintes. A Seção 4 descreve os elementos que afetam o tempo de resposta de resposta e define as unidades de medida utilizadas na análise. A Seção 5 explica a metodologia utilizada, softwares e o ambiente de infraestrutura para os testes. A seção 6 mostra como as recomendações de Steve Souders foram empregadas no trabalho. A Seção 7 apresenta os resultados obtidos. A Seção 8 discute alguns trabalhos relacionados. A Seção 9 retrata as conclusões e considerações finais do trabalho e apresenta as perspectivas em torno da avaliação e redução do tempo de resposta dos sistemas web.

2. Percepção do tempo de resposta de um site

Em pesquisa conduzida pelo Centro de Estudos sobre as Tecnologias da Informação e Comunicação [CETIC 2010] a respeito do uso de tecnologias da informação e da comunicação no Brasil, revelou-se que a principal dificuldade encontrada no uso da Internet reside no fato de que os sites demoram a carregar. A percepção de demora pode ser padronizada de acordo com três limites de espera [Nielsen 1993].

- 0,1 segundo: É o limite do tempo de resposta para que os usuários sintam como se suas ações estivessem efeito direto sobre o que ocorre na tela. Para criar a ilusão de manipulação direta, a interface de usuário deve ser mais rápida do que 0,1 segundo.
- 1 segundo: Quando o computador leva entre 0,1 e 1 segundo para responder, a sensação é de que o computador está produzindo o resultado. Portanto, as páginas

devem ser exibidas dentro de 1 segundo para que os usuários sintam-se navegando livremente.

- 10 segundos: Após 1 segundo, usuários ficam impacientes e percebem que estão esperando o computador processar a informação. Se a demora for além de 10 segundos, o usuário, normalmente, ficará impaciente e deixará o site.

A especificação do tempo de resposta de um site está relacionada à finalidade para qual o mesmo foi desenvolvido. Em aplicações de correio eletrônico tais como Gmail, a expectativa é que as respostas sejam muito rápidas, abaixo de 0,1 segundo. Já para divulgação de informações corporativas onde se deseja anunciar produtos ou serviços são aceitáveis respostas em até 0,1 segundo. O ambiente de testes utilizado para esse trabalho adota o último caso como parâmetro de comparação para avaliar as variações do tempo de resposta.

3. *World Wide We*b

A *World Wide Web* – WWW, ou simplesmente Web, representa, do ponto de vista dos usuários, um conjunto de documentos conhecidos como páginas web que consistem da combinação de:

- uma linguagem da marcação, o *Hypertext Markup Language* (HTML), um sistema que permite a associação de documentos com suporte a múltiplos formatos, como imagens e sons, exibidos de forma padronizada em um visualizador, que utiliza o conceito de hipertexto, que admite a inclusão de referências a outros documentos (*links*);
- um sistema de referências que permite o acesso às páginas Web e seus componentes, chamada de *Uniform Resource Locator* (URL) e
- um protocolo, o *Hypertext Transfer Protocol* (HTTP), que permite o tráfego eficiente desses dados [Kozierok 2005].

3.1 HTML

O HTML [RFC_2854 2000] é uma linguagem de marcação utilizada para definir os documentos de hipertexto. Ela define elementos sintáticos, chamadas *tags*, que informam ao navegador como o documento deve ser exibido. Essas *tags* associam documentos entre si e descrevem a semântica do texto.

Diferente de texto simples, documentos HTML são estruturados logicamente e divididos em uma série de elementos organizados de acordo com as regras da linguagem. Essas regras informam a sintaxe dos elementos e combinações permitidas. O título, um parágrafo, uma tabela e um *link* são exemplos de elementos.

O *Uniform Resource Locator* (URL) [RFC_1738 1994] foi desenvolvido para permitir que recursos pudessem ser facilmente encontrados na Internet, tendo a sintaxe mostrada na Figura 1:

<protocol>://<user>:<password>@<host>:<port>/<path>?<query-string>

Figura 1. Sintaxe da URL

- <*protocol*>: Representa o protocolo da camada de aplicação utilizado na Web é o HTTP.
- <*user*> e <*password*>: São opcionais, com a finalidade de autenticar o usuário em servidores com recursos protegidos. São raramente utilizados.
- <*host*>: O endereço do servidor. Pode ser tanto o endereço IP quanto o endereço de domínio.
- <*port*>: Caso não seja especificado, é inferido o número 80 – padrão para servidores web.
- <*path*>: Indica qual recurso do servidor deve ser recuperado.
- <*query-string*>: Campo opcional utilizado para enviar parâmetros adicionais ao servidor.

3.2 HTTP

O HTTP [RFC_2616 1999] é o protocolo da camada de aplicação através do qual se comunicam clientes - normalmente navegadores - e servidores web. Ele foi concebido como um protocolo textual, projetado para permitir a um cliente enviar uma requisição para obter uma informação e recebê-la do servidor.

Em sua forma mais simples, a operação do HTTP envolve apenas um cliente e um servidor HTTP, mais conhecido como servidor web. Após a conexão TCP ser criada entre os processos cliente e servidor, os dois passos na comunicação são a requisição do cliente e a resposta do servidor.

O cliente envia uma mensagem de requisição formatada de acordo com as regras do HTTP (Figura 2). Esta requisição especifica basicamente o método e o recurso desejado.

```
GET / HTTP/1.1
Host: www.zdnet.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:9.0.1) Gecko/20100101
Firefox/9.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: pt-br,pt;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7Connection: keep-alive
```

Figura 2. Exemplo de requisição HTTP de uma página HTML

A primeira linha da requisição especifica qual recurso desejado. Em seguida, são especificados os cabeçalhos da requisição – um por linha. Separado dos cabeçalhos por uma linha em branco, segue o corpo da requisição, se houver. Todas as linhas terminam com a sequência não impressa de caracteres *carriage-return-line-feed* (CRLF).

O servidor web interpreta a requisição, executa a devida ação e cria uma mensagem de resposta HTTP, que é enviada de volta para o cliente. A mensagem de resposta indica se a requisição foi bem sucedida através de um código de status e pode incluir o conteúdo do recurso solicitado pelo cliente (Figura 3).

```
HTTP/1.1 200 OK
Date: Tue, 24 Jan 2012 19:34:29 GMT
Server: ApacheX-Ua-Compatible: IE=edge,chrome=1
Keep-Alive: timeout=15, max=975
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
Cache-Control: private
Content-Encoding: gzip
Transfer-Encoding: chunked

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" xlmns:og="http://ogp.me/ns#"
xlmns:fb="http://www.facebook.com/2008/fbml" lang="en" class="">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<meta http-equiv="X-Ua-Compatible" content="IE=edge,chrome=1" />
(...)
```

Figura 3. Exemplo de resposta HTTP de uma página HTML

Após receber a resposta, o navegador inicia a fase de interpretação do HTML. Neste processo, conforme encontra *tags* HTML, ele preenche uma estrutura de dados chama *Document Object Model* (DOM) [Flanagan 2002] [W3C 1998], que guardará todos os elementos da página, como formulários, parágrafos e imagens (Figura 4).

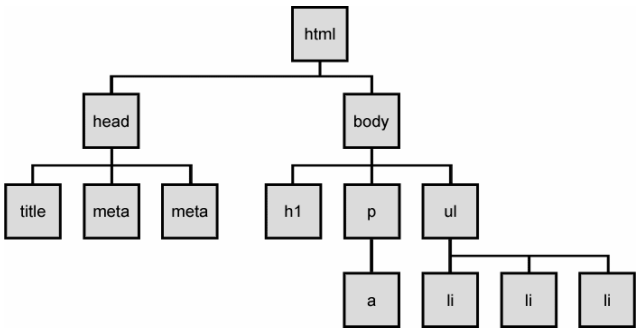


Figura 4. *Document Object Model* (DOM)

No processo de interpretação, ao identificar referências a componentes externos ao HTML como imagens, código JavaScript, Java Applets, CSS ou Flash, o navegador faz uma busca em seu *cache* para verificar se o recurso já foi transferido anteriormente. Caso não o tenha em *cache*, gera outras requisições HTTP para o servidor, uma para cada componente, de forma a obter os mesmos. Após todos os componentes serem transferidos, a versão final da página é exibida. Em alguns casos, interpretadores ou ambientes de execução são acionados, como, por exemplo, no caso de JavaScript, ou arquivos Flash.

O protocolo HTTP define oito métodos com os quais uma requisição pode ser feita [RFC_2616 1999], dentre os quais, os mais comuns são:

- GET: Tem como propósito requisitar um recurso do servidor. É o tipo mais básico de requisição e responsável pela maior parte do tráfego. Parâmetros adicionais são passados através da própria URL, no campo *query-string*.

- POST: Serve para enviar informações que devam ser processadas no servidor, o que pode resultar na inserção de um novo recurso ou na atualização de um já existente. Qualquer parâmetro é passado através do corpo da requisição.

Embora esses sejam os usos sugeridos, não são obrigatórios. Cabe ao servidor interpretar o que a requisição realmente executa [RFC_2616 1999].

O protocolo HTTP também define códigos de status, que fornecem ao navegador o estado da resposta em sua primeira linha (trecho de código da Figura 3). A Tabela 1 apresenta algumas classes de código padronizadas no HTTP.

Tabela 1. Significado dos códigos de status HTTP pelo primeiro dígito

Código	Significado	Descrição
1XX	Information	Informações genéricas. Não significa sucesso nem falha.
2XX	Success	Requisição foi recebida e aceita pelo servidor.
3XX	Redirect	Alguma ação adicional do cliente é necessária.
4XX	ClientError	A requisição é inválida ou não pôde ser completada por alguma razão que o servidor crê que seja do cliente.
5XX	Server Error	A requisição foi válida, mas por culpa do servidor, não pode ser completada.

De acordo com o código de status recebido, a máquina de estados do navegador é acionada diferentemente, levando à exibição da informação, à geração de novas requisições ou à exibição de uma mensagem de erro ao usuário.

3.3 Arquitetura web típica

Um sistema web típico é dividido logicamente em duas partes: frontend e backend.

No *frontend* são empregadas tecnologias relacionadas à interface de usuário, como HTML, CSS, JavaScript, imagens, áudio, vídeo, Flash e Silverlight. Esses componentes, inicialmente residentes no servidor, são transferidos para o navegador conforme solicitados.

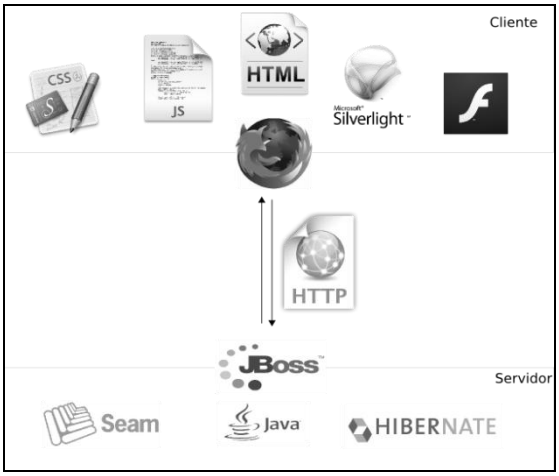


Figura 5. Tecnologias encontradas no Frontend e Backend

No *backend*, onde se localiza o servidor, é executada a lógica da aplicação. Nele, são utilizadas linguagens de programação como Java, Perl, PHP e Python, seus respectivos *frameworks* e é feito o acesso a bancos de dados e a sistemas de arquivos (Figura 5). Sistemas coorporativos podem usar servidores de aplicação (JBoss, por exemplo) e acesso à Sistemas Gerenciadores de Banco de Dados.

O papel do *backend* é processar a requisição e gerar uma resposta para o usuário. Em uma aplicação típica, a requisição chega ao servidor web, que verifica qual recurso o cliente deseja analisando a primeira linha da mensagem (Figura 6). Em seguida compara o recurso desejado com uma relação de rotas, que mapeia URLs para ações a serem executadas (Figura 6).

```
# Estáticos
GET /images/logo.png                  → $DOCUMENT_ROOT/images/logo.png
GET /images/books/$BOOK_ID.png   →
$DOCUMENT_ROOT/images/books/$BOOK_ID.png

# Dinâmicos
GET   /books                          → listAllBooks();
GET   /books?$BOOK_ID                       → showBook($BOOK_ID);
GET   /dvds?$DVD_ID                   → showDVD($DVD_ID);
POST  /checkout/                            → chargeCreditCard($CREDIT_CARD_NO);
```

Figura 6. Tabela de rotas do servidor

O recurso requisitado pode ser estático ou dinâmico. Recursos estáticos estão fisicamente salvos em disco, de onde são prontamente recuperados e prontamente enviados ao cliente. Recursos dinâmicos, por sua vez, são processados pelo servidor web ou por serviços externos acionados pelo servidor web, e criados conforme necessário pela aplicação.

Aplicações web, de uma forma geral, apresentam uma grande quantidade de arquivos HTML que descrevem conteúdos e estruturas semelhantes umas às outras, diferindo apenas no conteúdo específicos de alguns elementos. Uma técnica utilizada nos sistemas web, nestes casos, é usar uma versão inacabada do arquivo chamada *template* (Figura 7) ao invés de mantê-los completos em disco.

```
<imgsrc="/books/<%= image_path_of_book (@book_id) %>" alt="" />
<tr>
<td>List Price:</td>
<td>U$ <%= list_price_of_book(@book_id) %></td>
</tr>
<tr>
<td>Price:</td>
<td>U$ <%= price_of_book(@book_id) %></td>
</tr>
```

Figura 7. Trecho HTML de um *template*

Ao final do processo, o cliente não sabe se a página já estava completamente montada (estática) ou se foi criada após a requisição (dinâmica), nem quais tecnologias foram empregadas no servidor. Ele apenas recebe uma mensagem de texto como resposta seguindo as regras definidas pelo HTTP.

4. Fatores Determinantes da Velocidade de um Site

O tempo de resposta da página é o tempo decorrido entre o envio da primeira requisição pelo documento HTML, o processamento no *backend* e *frontend*, e o término de todas as transferências associadas, ou seja, dos demais componentes externos [Souders 2007]. Três fatores influenciam o tempo de resposta: a comunicação entre cliente e servidor, o tamanho do recurso e o tempo para o servidor processar a requisição.

Dois parâmetros de desempenho para avaliar redes de computadores são relevantes neste caso pois influenciam no tempo de resposta percebido pelo usuário:

- Largura de Banda: Indica a quantidade máxima de dados que pode passar de um ponto a outro de uma rede em uma unidade de tempo.
- Latência: É o tempo entre o envio e resposta a uma mensagem de solicitação. Quanto menor a Latência melhor é a performance de uma rede [Kozierok, Charles 2005].

Durante a comunicação entre cliente e servidor podem ser identificados pontos onde o processamento e o transporte de dados podem representar gargalos de tempo, dando indícios de onde podem ser aplicadas otimizações:

Roteadores Intermediários. Cliente e Servidor são duas entidades que na grande maioria dos casos estão separados geograficamente. Com isso, é necessário que a mensagens trocadas entre essas duas entidades, sejam retransmitidas por máquinas intermediárias, chamadas de roteadores. Cada roteador decide localmente para onde encaminhar a mensagem. Esse processo, chamado roteamento, contribui para o aumento da latência.

Banda da Comunicação. Cliente e servidor possuem conexões a rede IP com largura de banda diferenciadas, na grande maioria dos casos, servidores estão instalados em *datacenters* que proporcionam conexões com centenas de Mbps. Já o cliente, geralmente se conecta à Internet, através de provedores de acesso, com velocidades que variam entre dezenas de Kbps a dezenas de Mbps. Isso faz com que a taxa do cliente atue como um limitador da comunicação [CETIC 2010].

Tamanho do Recurso. Intimamente associado à banda disponível, o tamanho do recurso também é proporcional ao tempo de transferência dos dados. Quanto maior o recurso, maior o volume de dados a ser transmitido e maior será o tempo necessário para transferi-lo.

Tempo do Servidor. Para atender uma requisição, o servidor precisa interpretá-la e executar a ação pertinente. Recursos estáticos não necessitam de muito tempo de processamento e são prontamente atendidos. Recursos dinâmicos requerem a execução de atividades adicionais e podem contribuir para o aumento do tempo de respostas.

Ao longo do seu trabalho, Steve Souders, conduziu testes para avaliar e velocidade dos dez *sites* mais acessados nos Estados Unidos, realizando duas medidas de tempo:

- Tempo gasto no *backend*: Intervalo de tempo entre a solicitação e o recebimento dos primeiros pacotes da resposta do HTML .
- Tempo gasto no *frontend*: Intervalo de tempo entre a solicitação do primeiro componente externo e o recebimento do último.

A Tabela 2 mostra o resultado dos testes realizados a partir de dois cenários. O primeiro com o "*cache* do *frontend* vazio", isto é, as páginas foram acessadas pela primeira vez sendo necessário transferir todos os componentes das páginas para o *frontend*. Os acessos seguintes, relativos ao segundo cenário, foram realizados com o "*cache* cheio", isto é, alguns componentes da página foram armazenados no *frontend*, não sendo necessário transferi-los novamente, reduzindo assim o tempo total de carga das páginas.

Os testes mostraram que o tempo de carga do documento HTML, na maioria dos casos, variou entre 10 e 20% do tempo total necessário para carregar uma página completa, com ou sem a utilização de *cache* no *frontend*.

Tabela 2. Percentual do tempo gasto no backend de 10 grandes sites

Site	***Cache* vazio**	***Cache* cheio**
AOL	6%	14%
Amazon	18%	14%
CNN	19%	8%
EBay	2%	8%
Google	14%	36%
MSN	3%	5%
MySpace	4%	14%
Wikipedia	20%	12%
Yahoo!	5%	12%
YouTube	3%	5%

Dentre os fatores que influenciam o tempo de resposta, o desenvolvedor não tem controle sobre a comunicação entre clientes e servidor, isto é, não tem como interferir nos roteadores intermediários nem na banda do usuário. Com isso, para melhorar o desempenho do sistema o estudo deve ser dedicado a melhorar o tempo de resposta do servidor (*backend*) – escalamento vertical– e reduzir o tamanho das transferências (*frontend*) – otimizações de software.

Tomando a proporção aproximada de que 20% do tempo total de carga de uma página é gasto no *backend* (Tabela 2), é possível concluir que 80% do tempo total é consumido no *frontend*. Desse modo, os testes sugerem que qualquer esforço para se diminuir o tempo de resposta terá maior eficiência focando-se em otimizar o *frontend*.

5. Avaliação

Para realizar a avaliação uma sequência de testes foi executada, utilizando uma aplicação web (um *blog*) executando em um nó servidor. A esse servidor, foram encaminhadas requisições através de um gerador sintético de carga, e calculado o tempo de resposta para diversos cenários de configuração. Dois blocos de testes foram

desenvolvidos: uma para avaliar o escalamento vertical e outro para avaliar as diferentes configurações de software.

Nos cenários de avaliação do escalamento vertical, apenas as especificações do servidor sofreram alterações. Foram acrescentados, gradualmente, mais tempo de CPU e quantidade de memória. Nessa etapa todas as configurações padrão de softwares foram mantidas.

Para os cenários em que o objetivo era verificar como as configurações de software afetam o desempenho, as especificações do servidor foram mantidas e alguns ajustes nas configurações de software, sugeridas por Souders, foram aplicadas. Alguns dos testes foram planejados para verificar o efeito das recomendações publicadas pelo Yahoo quando aplicadas individualmente ou combinadas.

O ambiente de avaliação foi estruturado a partir de máquinas virtuais que permitem não só controlar os softwares instalados como também adicionar os recursos de hardware necessários as especificações dos testes. Foram criadas duas máquinas virtuais: Na primeira foi instalado um servidor web (Apache), um gerenciador de banco de dados (MySQL) e um gerenciador de conteúdo web (Wordpress). Na segunda, para simular o comportamento dos clientes, foi instalado o WWW::Mechanize, software que possibilitam gerar requisições web de forma sintética gerando carga para o servidor.

5.1 Geração de Requisições

O funcionamento da Web conceituado anteriormente, mostra que é relativamente simples gerar carga em um servidor, bastando enviar-lhe uma quantidade significativa de requisições HTTP. Em um cenário real, vários usuários geram requisições para o servidor (Figura 8).

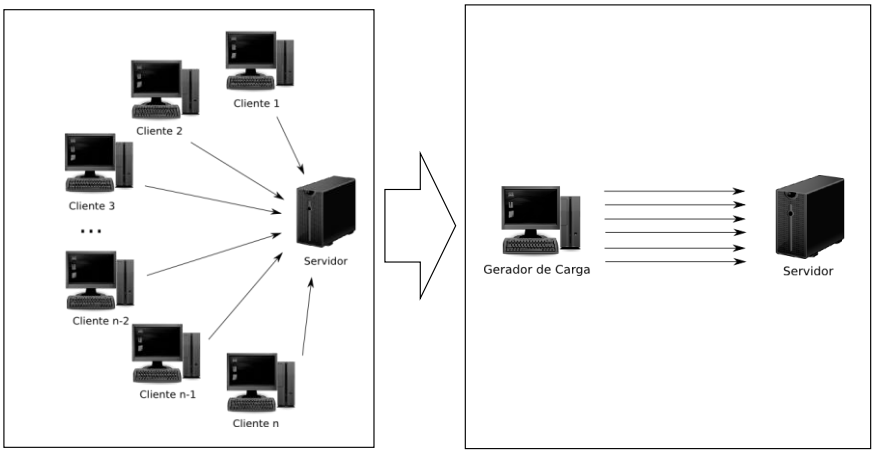


Figura 8. Carga gerada por clientes reais **Figura 9. Carga gerada artificialmente**

No entanto, em um ambiente de testes, para causar o mesmo efeito e gerar múltiplas requisições a ponto de estressar o servidor é necessária uma aplicação que simule o comportamento de diversos clientes (Figura 9).

Existem algumas ferramentas, como o ApacheBench [Apache.org 2013] e o HTTPerf [Mosberger e Jin 1998] que são amplamente usadas para a geração de tráfego HTTP sintético e para medir o desempenho de servidores web.

O WWW::Mechanize [Neumann 2005] é uma biblioteca disponível nas linguagens Perl, Python e Ruby que auxiliam a criação de scripts de visitação de *sites*. Seu uso comum é para *screenscraping* e criação de *spiders*. É constituído de dois módulos principais, que em conjunto executam as tarefas básicas de um navegador:

- *User-agent*: Implementa o protocolo HTTP. É responsável por formatar as requisições e interpretar a resposta.
- *HTML parser*: Permite a navegação pelo DOM da página por meio de seletores CSS. Através dele que é possível determinar com quais componentes da página interagir (ex.: clicar em *links* e preencher formulários).

No presente trabalho utilizamos o WWW::Mechanize porque é possível simular também o comportamento dos sistemas locais de *cache* no cliente, e manter o perfil dos casos de uso desejados para cada cliente simulado. Cada usuário é abstraído numa classe chamada *Navigator,* que desempenha o papel do navegador. O gerador de carga lê um arquivo que contém o *workload* (carga de trabalho), onde é determinado o número de clientes e seus respectivos casos de uso. Em seguida cria um *Navigator* para cada cliente. (Figura 10).

Cada *Navigator* envia requisições para o servidor de acordo com seus casos de uso. Caso a resposta da requisição seja HTML, o *Navigator* utiliza o *HTML parser* para interagir com componentes da página. Os dados estatísticos relativos a tempo de resposta são coletados pelos próprios clientes.

Figura 10. User-agents criados pelo gerador de carga

Cada cliente foi programado para realizar requisições em uma taxa constante de 15 requisições por minuto. O teste de carga, com o aumento progressivo da taxa de requisições foi simulado criando-se mais instâncias da classe *Navigator* no início do teste, disparando a execução das mesmas concomitantemente.

5.2 Infraestrutura

A infraestrutura contendo servidores, rede e cliente gerador de carga foi simulada em um ambiente virtualizado (Figura 11).

Em relação aos servidores, a ideia básica por trás de uma máquina virtual é a abstração do hardware de um computador (CPU, memória, acionadores de disco e demais componentes) em vários ambientes de execução diferentes, dando a sensação de que cada usuário tem seu próprio ambiente de execução privado [Silberschatz 2010].

Cada máquina tem a ilusão de ter processadores, memória e disco próprios. No contexto de hospedagem para a Web, a virtualização serviu para preencher a lacuna entre dois extremos: máquinas dedicadas e hospedagem compartilhada.

- Máquina dedicada: A máquina é cedida a apenas um usuário, que tem liberdade de instalar qualquer software. No entanto, nem sempre o usuário precisa de tantos recursos e acaba ocorrendo desperdício.
- Hospedagem compartilhada: Uma única instalação do ambiente de execução (sistema operacional, banco de dados, linguagens de programação, softwares utilitários) é compartilhada por vários usuários.
- Máquina Virtual ou *Virtual Private Server* (VPS): Uma máquina real, chamada de hospedeira, é fracionada em algumas máquinas virtuais, chamadas de convidadas. Cada máquina virtual é cedida a um usuário, que tem posse da conta de administrador e pode instalar e configurar quaisquer softwares. Normalmente a máquina hospedeira é dividida em uma quantidade pequena de máquinas convidadas a fim de manter o desempenho aceitável em todas as instâncias.

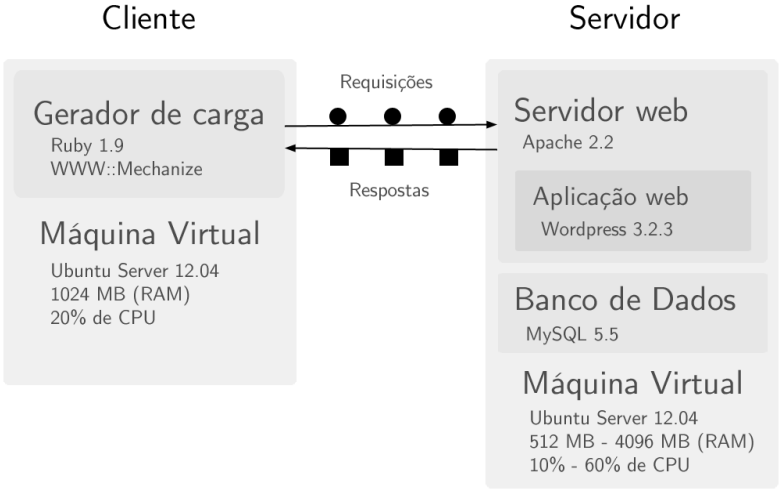


Figura 11. Estrutura usada nos testes

A proposta dos testes realizados foi a de aproximar as condições dos testes sintéticos ao de um cenário real. Para isso é importante que tanto softwares quanto infraestrutura sejam os mesmos utilizados em soluções reais. A Figura 11 apresenta a estrutura usada nos testes, que utiliza Linux, Apache, MySQL e PHP, que formam a plataforma mais comum para desenvolvimento web [Netcraft 2010], e também utiliza a Linode [Linode 2013] que oferece um ambiente virtualizado semelhante aos principais *datacenters* profissionais.

Para avaliar as recomendações, também presente na Figura 11, foi usado um Sistema de Gerenciamento de Conteúdo (CMS - *Content Management System*) popular, o Wordpress [Wordpress.org 2013]. O Wordpress é sistema de blogs personalizável, *open source*, codificado em PHP e normalmente utilizado em conjunto com a plataforma LAMP (Linux, Apache, MySQL, PHP). Com uso de determinados *plugins*, é possível estender suas funcionalidades para fóruns de discussão, lojas virtuais e e-zines.

É o CMS mais utilizado atualmente, com 15% da fatia de mercado [W3Tech 2012]. O perfil de uso desse sistema é de muitas leituras (para recursos estáticos ou dinâmicos) e poucas escritas.

Essa plataforma foi instalada em servidores com diversas configurações e teve os resultados comparados. A base para a comparação foi uma instalação padrão do Wordpress 3.2.3 na máquina servidor (Figura11).

O último componente, o usuário, foi simulado por meio de casos de uso, que seguiram a proporção de 10% de escritas e 90% de leituras a fim de refletir o comportamento típico dos usuários [Nielsen 2006]. Os cálculos do tempo de resposta foram baseados numa conexão de 1Mbps por ser a velocidade mais comum dos usuários brasileiros [CETIC.br 2010].

6. Técnicas de Redução do Tempo de Resposta

Na avaliação do efeito de configurações de software do frontend, as recomendações sugeridas em [Souders 2007] foram aplicadas separadamente e em conjunto e o desempenho do sistema avaliado. Estas configurações podem ser aplicadas sem a necessidade de alterar a lógica da aplicação e são realizadas modificando-se os arquivos de configuração (httpd.conf e header.php) e arquivos estáticos (JavaScript, CSS e imagens). Os aspectos técnicos e o resultado esperado para cada uma das configurações são discutidas nas próximas subseções.

6.1 Redução do número de requisições HTTP

Parte do tempo de resposta está associada à transferência de componentes externos da página, tais como: imagens, folhas de estilo e códigos JavaScript. Reduzir o número de componentes tem como resultado a redução no número de requisições HTTP necessárias a exibição da página.

Cada requisição contém cabeçalhos cujo tamanho deve ser considerado no consumo da banda disponível para o usuário. Em transferências de grandes volumes de dados, o cabeçalho se torna proporcionalmente pequeno. Analogamente, para transferências de pequenas volumes, o cabeçalho se torna proporcionalmente grande (Figura 12).

Figura 12. Sobrecarga de cabeçalhos HTTP

Ao combinar o conteúdo de vários componentes em um único arquivo, economiza-se a banda que seria utilizada na transmissão de múltiplos cabeçalhos. Além disso, o tamanho do arquivo combinado é menor que a soma dos tamanhos dos arquivos individuais, uma vez que um arquivo grande proporciona maior taxa de compressão do que muitos arquivos pequenos.

Para combinar arquivos de texto, como CSS e JavaScript, é necessário copiar o conteúdo dos arquivos para um único documento. Já a combinação de imagens é feita através de um *sprite* de imagens, isto é, cria-se um novo arquivo com as figuras posicionadas uma ao lado da outra. A Tabela 4 apresenta alguns resultados numéricos para alguns casos.

Tabela 4. Comparação entre arquivos individuais e combinados

Formato do arquivo	Tamanho não combinados	Tamanho combinados	Redução
JavaScript	449 kB	294 kB	34 %
CSS	43 kB	16 kB	62 %
PNG (image)	51 kB	38 kB	25 %

6.2 Adicionar Cabeçalho Expires

Na primeira visita do usuário à página, o navegador pode ter que fazer várias requisições por componentes externos. Usando o cabeçalho *expires* nas respostas, o servidor instrui o navegador a manter o componente em *cache* durante um determinado período, evitando futuras requisições para o mesmo componente (considerando o prazo de validade). A instalação padrão do Apache inclui o cabeçalho *expires* ativado. Em um dos testes o cabeçalho expires foi desativado para analisar sua contribuição para o tempo de resposta.

6.3 Configurar *ETags*

EntityTags (ETags) é um mecanismo que servidores e navegadores usam para determinar se o componente no *cache* do navegador combina com o residente no servidor. Junto da primeira reposta à requisição por um determinado componente, o servidor envia um *hash* (Etag) daquele componente.

Diferente do cabeçalho *expires*, o ETag (Figura13) não fornece meios do navegador decidir localmente se o componente é válido ou não. O navegador precisa fazer a requisição, no entanto adicionando o cabeçalho *If-None-Match* na mensagem.

```
ETag: "686897696a7c876b7e"
```

Figura 13. Exemplo de Etag na resposta do servidor

Ao receber a requisição, o servidor compara o Etag enviado pelo navegador com o Etag de sua versão do componente. Se as Etags combinarem, a versão do cliente ainda é valida, bastando notificá-lo a usar a versão em cache (através do código de controle 304 – NotModified). Caso não combinem (Figura 14), o componente é transferido normalmente.

```
If-None-Match: "686897696a7c876b7e"
```

Figura 14. Exemplo de Etag na requisição do cliente

ETags e *expires* são dois mecanismos distintos para implementar a mesma política – de evitar transferências desnecessárias. A diferença é que com *expires*, pelo fato de ter um prazo de validade já determinado, o navegador decide localmente por usar sua versão em *cache*, evitando a necessidade da requisição. Por outro lado, com *Etags*, o navegador sempre faz a requisição, no entanto esperando evitar a transferência do componente na resposta.

6.4 Comprimir os Componentes

A compressão reduz o tamanho da resposta HTTP. A partir do HTTP/1.1, clientes web podem indicar o suporte para compressão através do cabeçalho *accept-encoding*. Gzip é o método de compressão mais popular, reduzindo o tamanho da resposta em cerca de 70% [Yahoo 2012].

A instalação padrão do Apache já inclui a compressão habilitada. Para efeitos de comparação, ela foi desabilitada em um dos testes.

6.5 Colocar CSS e JavaScript em Arquivos Externos

Tanto CSS quanto JavaScript podem ser incluídos nas páginas de duas formas: *inline* e em arquivos externos. *Inline* significa que o código residirá no corpo do próprio HTML página. Quando colocados em arquivos externos, apenas uma referência é inserida no HTML. Pelo fato de estarem em seus próprios arquivos, CSS e JavaScript externos podem ser reutilizados por muitas páginas.

Colocando CSS e JavaScript em arquivos externos permite que sejam salvos no *cache* do navegador. Em oposição, quando são mesclados diretamente no código HTML (*inline*), o código é baixado novamente cada solicitação.

6.6 Gerenciar o Número de Consultas ao DNS

O navegador não pode começar a baixar os componentes da página até saber o endereço IP do servidor. Para isso, deve consultar o DNS. Cada consulta leva de 20 a 120 ms. Evitar consultas ao DNS pode reduzir o tempo de resposta.

Por outro lado, devido a uma recomendação da W3C para que os navegadores transfiram apenas 2 componentes em paralelo de cada domínio, alguns sites optam por dividir os componentes por mais de um domínio, o que aumenta o paralelismo das transferências. A divisão em n domínios permite $2*n$ transferências em paralelo, porém requer n consultas ao DNS.

Navegadores modernos desprezam a recomendação da W3C e baixam até 8 componentes de cada vez, de forma que não faz mais sentido servir os componentes através de vários domínios.

6.7 Minificar JavaScript e CSS

Minificar é o processo de remover caracteres desnecessários do código para reduzir o tamanho e melhorar o tempo de carregamento. Quando o código é minificado, todos os comentários e espaços em branco são removidos.

Ao minificar os arquivos JavaScript da aplicação, houve uma redução de 757kB para 238kB, 31% do tamanho original. Os arquivos CSS foram reduzidos de 67kB para 33kB, 49% do tamanho original.

6.8 Otimizar as Imagens

Câmeras fotográficas modernas são capazes de salvar imagens muito grandes. Uma câmera de 5MP, por exemplo, salva imagens em dimensões de 2560px X 1920px, com baixa taxa de compressão. Quanto maior a imagem, maior será o espaço ocupado em disco, o que aumenta o tempo de transferência.

No entanto, monitores são dispositivos de baixa resolução quando comparado à mídia impressa. Cerca de 75% dos usuários utiliza resoluções inferiores a 1440px x 900px [W3Schools 2012]. Desta forma, devem-se reduzir ao máximo as imagens sem perder qualidade, diminuindo suas dimensões para se adequar à página.

Além de ter suas dimensões reduzidas, a imagem também pode ser comprimida. O JPEG, formato preferido para fotografias, permite selecionar o nível de compressão. Pelo fato de ser um formato *lossy* (com perdas), a qualidade da imagem comprimida é inferior à imagem original. Com isso é importante selecionar um nível de compressão que não piore substancialmente a qualidade da imagem.

Como última observação, atualmente as aplicações podem selecionar dinamicamente a qualidade da imagem e até mesmo o conteúdo da página a ser enviada para o navegador a partir de obtidas do próprio, como por exemplo, sua memória, resolução de exibição ou a capacidade do processador e placa gráfica.

A aplicação contém 8 fotografias que são carregadas aleatoriamente na parte superior de cada página. Um usuário que navegar por muitas páginas eventualmente terá requisitado todas as imagens. Elas, em conjunto, têm 488 kB. Quando otimizadas, salvando as mesmas com formato "Jpeg / High", ocupam 330 kB – 67% do tamanho original.

7. Resultados

Como cerca de 80% do tempo de resposta está associado à transferência dos componentes, as recomendações de Souders estão relacionadas à diminuição do total de dados transferido. Algumas delas reduzem o tamanho dos componentes a serem transferidos (recomendações 6.1, 6.4, 6.7 e 6.8). Outras evitam completamente a necessidade de uma requisição (6.2, 6.3 e 6.5).

Nas próximas subseções são apresentados os resultados obtidos durante as várias execuções dos testes, discutindo-se como os mesmos influenciaram o tempo de resposta do sistema.

7.1 Uso de CPU / memória X o número de clientes

Após a definição dos casos de uso (Seção 6) para o gerador de carga sintético, foi verificada a carga de trabalho máxima que a aplicação suporta, em sua configuração padrão, *1024*, sem degradar o desempenho. O número de usuários, e suas respectivas requisições, foi aumentado em saltos de 15 usuários para cada cenário, começando com 15 e terminando com 90 usuários.

A duração de cada teste é de 5 minutos. Cada usuário instanciado realiza 15 requisições em cada teste (3 requisições por minuto). Isso representa uma taxa de 45 requisições por minuto no primeiro cenário, terminando com 270 requisições por minuto no último cenário. Os resultados são apresentados na Figura 15, com a média e desvio padrão do total das requisições realizadas em cada cenário.

Pode ser observado que de 15 até 45 clientes (gerando de 45 a 135 requisições por minuto, respectivamente), o servidor respondeu de maneira previsível – em torno de 1 segundo, o que é aceitável para a aplicação. No entanto, a partir de 60 clientes (180 requisições por minuto), houve um grande aumento do tempo de resposta (valores maiores que1 seg, superando a marca de 2 seg) e a variação se apresenta ligeiramente superior.

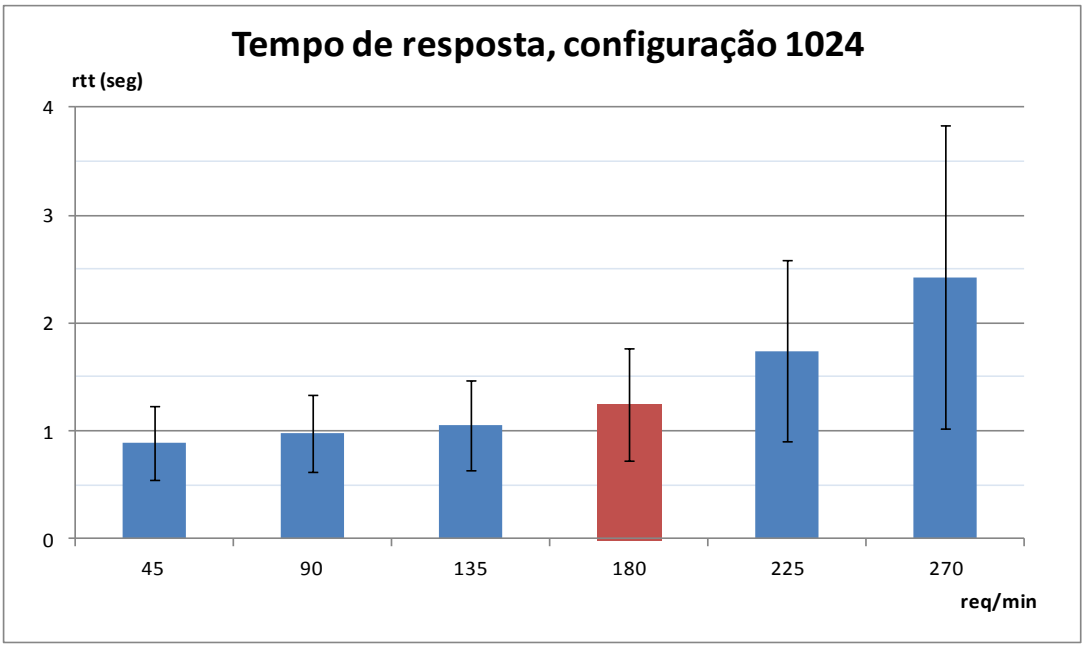
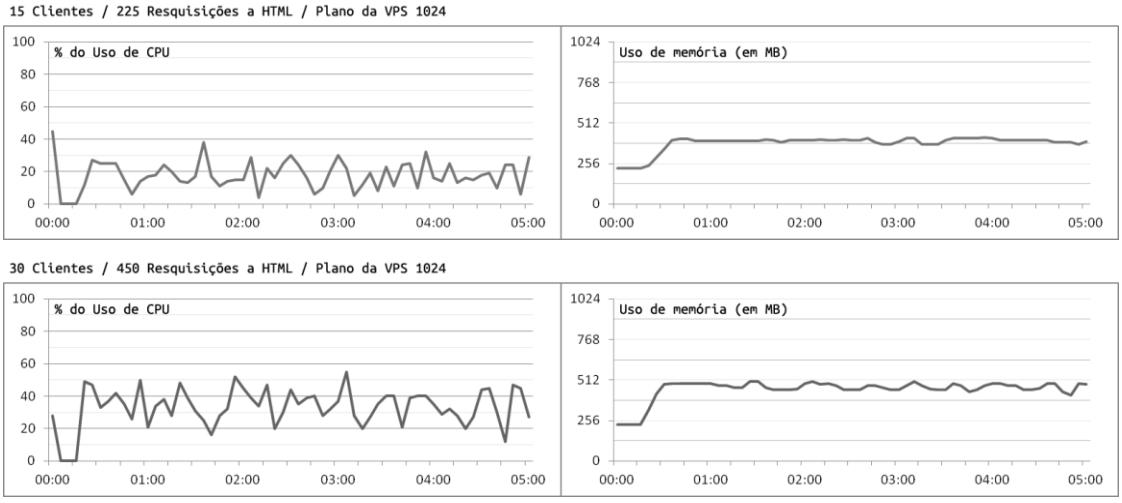


Figura 15. Tempo de resposta da aplicação para diversos *workloads*. Plano1024

O uso de recursos de CPU e memória são exibidos nos gráficos da Figura 16 para cada um dos cenários de carga.



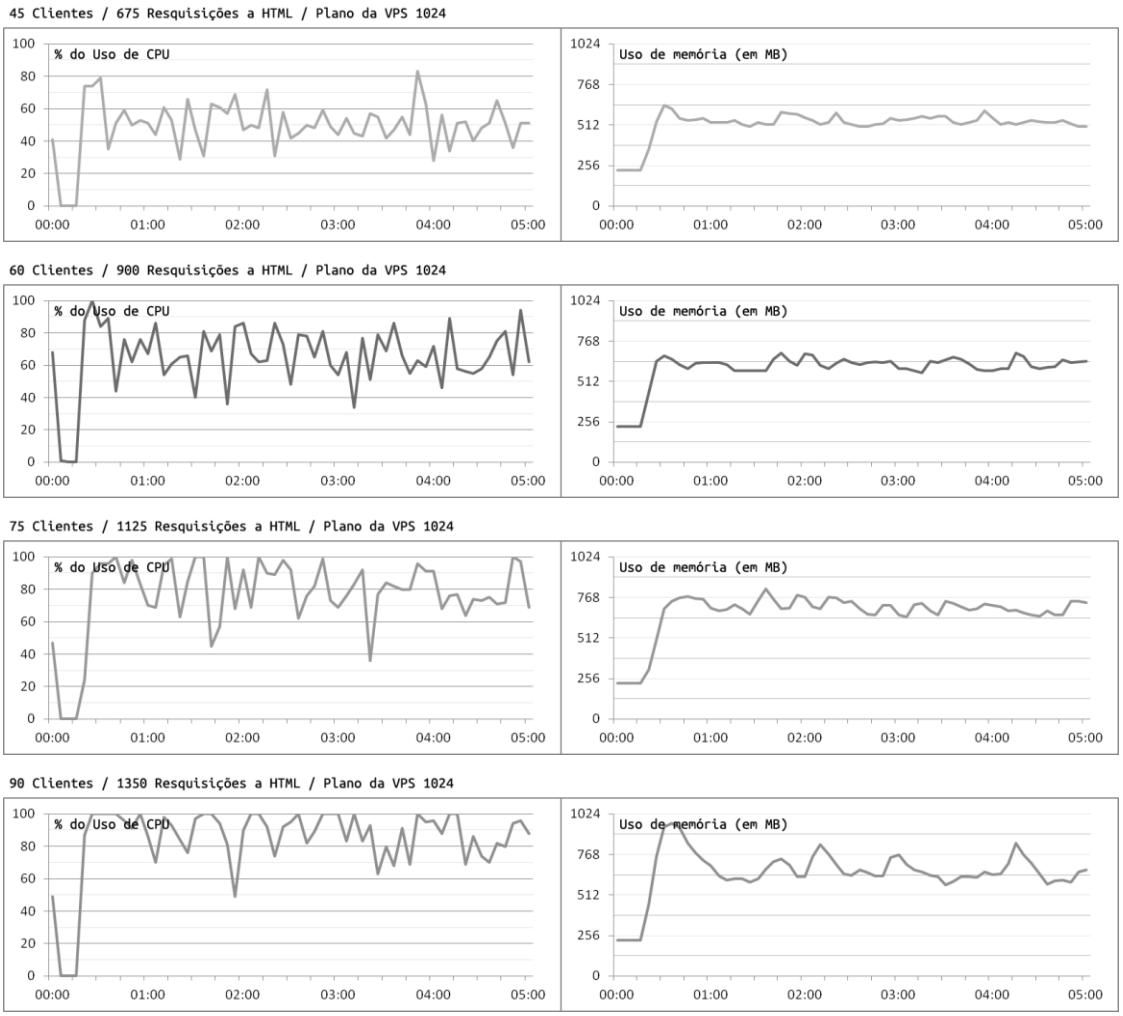


Figura 16. Uso de CPU e memória para as diversas cargas de clientes. Plano de hospedagem "1024"

Para os três primeiras cenários - 15, 30 e 45 clientes -, o uso de CPU permaneceu inferior a 60% durante a maior parte do tempo, com poucos picos, que não ultrapassaram 85%. O uso de memória aumentou linearmente.

A partir de 75 clientes, o aumento do uso de CPU e memória condizem com o aumento de tempo de resposta observados. Em diversos momentos, o uso de CPU atingiu 100% e o uso de memória aproximou-se do total disponível – 1024 MB. Com o esgotamento de recursos, algumas requisições sofreram atrasos, resultando no aumento da média do tempo de resposta e do desvio padrão. Foi decidido fixar a carga aceitável para esta configuração de servidor em 60 clientes como parâmetro de comparação entre as abordagens para redução do tempo de resposta.

7.2 Escalamento Vertical

Um dos métodos para se melhorar o desempenho de um sistema, é aumentar a disponibilidade dos recursos. No caso de serviços de hospedagens de máquina virtuais, como Linode e Amazon Web Services [Amazon 2014], isso significa migrar para planos com mais tempo de CPU e quantidade de memória. Os planos avaliados neste trabalho

foram: *1024*, *1536*, *2048* e *4096*, onde o número de recursos vai aumentando para 5%, 6%, 10% e 20% de CPU correspondente a um processador e 1024, 1536, 2048, 4096 MB de memória, respectivamente.

Nos testes de escalamento vertical, a seqüência de testes para 60 usuários repetida para cada uma das quatro configurações no servidor. O cliente foi mantido com uma configuração de 5% de CPU e 1024 MB de memória. Lembrando que a taxa de requisições é de 180 requisições por segundo e cada teste durou 5 min.

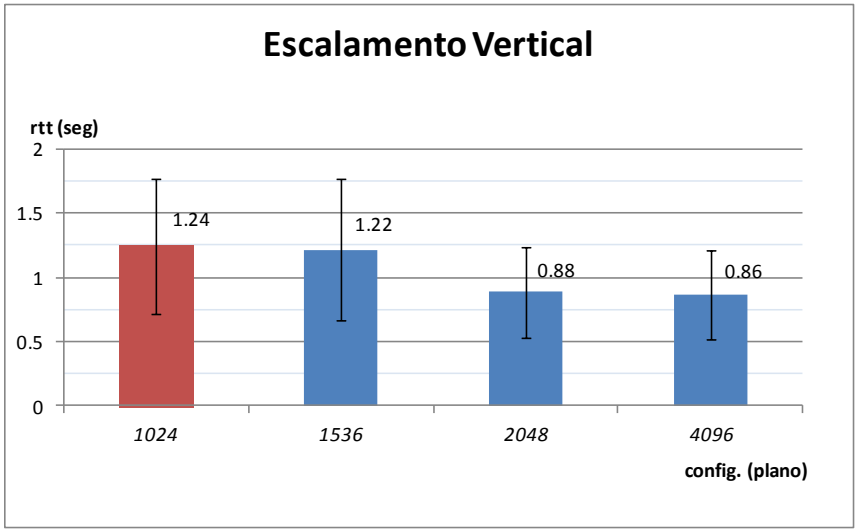
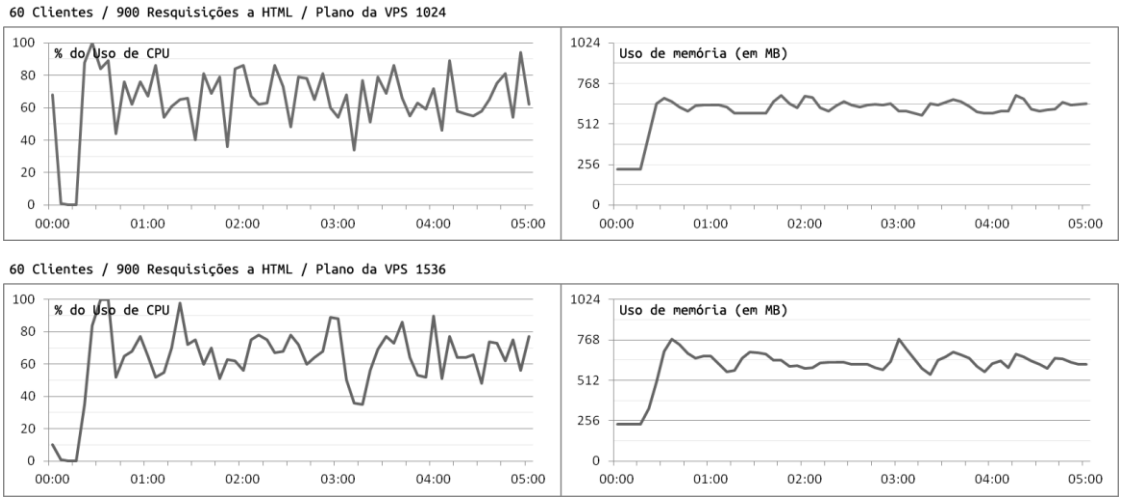


Figura 17. Comparação do tempo de resposta da aplicação para um *workload* de 60 clientes entre 4 planos de VPS

Como pode ser visto, a migração para planos com maior disponibilidade de recursos surte algum efeito na redução do tempo de resposta, trazendo-o para um patamar abaixo de 1 segundo (Figura 17), em torno de 30% mais rápido na configuração *4096*, que conta com 4 vezes mais recursos de CPU e memória que a configuração *1024*. Os tempos no plano *4096* parecem também mais estáveis.

Os gráficos da Figura 18 apresentam detalhes do uso de CPU e memória nos servidores nos 4 planos testados.



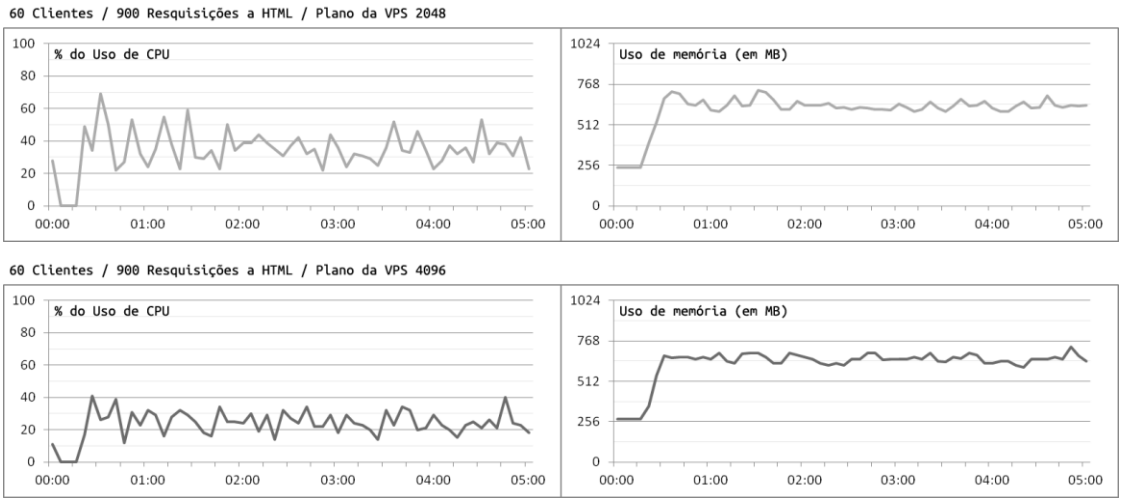


Figura 18. Uso de CPU e memória para 60 clientes nos diversos planos de hospedagem: "1024", "1536", "2048" e "4096"

A comparação entre os gráficos desses planos e dos planos *1024* e *1536* sugerem que a disponibilidade de mais tempo de CPU dos planos maiores, suficiente para não causar picos de 100%, contribuiu para o sistema responder mais rapidamente e consistentemente.

Nos planos *2048* e *4096*, embora o uso de memória tenha permanecido inalterado, o percentual do uso de CPU diminuiu, ficando em média abaixo de 40%. Por outro lado, a permanência do tempo de resposta na casa de 0,9 segundos após a migração do plano *2048* para o plano *4096* demonstra que, para a carga do teste, a fatia de CPU do plano 2048 já é suficiente para atender eficientemente as requisições. Assim, o aumento de recursos físicos pode não ser o fator de maior peso em todos os cenário de uso.

7.3 Influência das diversas configurações de software

Desde 2007, quando se tornaram públicas no *Yahoo! Developers Network*, algumas recomendações passaram a ser ativadas na instalação padrão dos servidores, como o uso do cabeçalho *expires*, *etags* e compressão.

Para testar a contribuição dessas recomendações, que já vem ativadas, foi executado um teste com cada uma delas individualmente ativadas, e as demais desativadas. Também foi realizado um teste sem nenhuma configuração ativa e outro com todas elas ativadas.

Todos os testes foram feitos tomando como base o do plano de *1024*, com a carga de 60 usuários sintéticos que geraram 900 requisições em 5 min, numa taxa média de 180 requisições por min. No gráfico da Figura 19 são apresentadas a média e o desvio padrão de todas as requisições para cada teste.

Para cada um dos resultados algumas interpretações podem ser discutidas, comparadas com a configuração padrão (*Default*):

Menos Requisições. A média do tempo de resposta não sofreu alteração significativa com a redução do número de requisições.

Cabeçalho Expires Desabilitado. Ao desabilitar o cabeçalho *expires*, os clientes precisaram requisitar componentes externos a cada página HTML, aumentando o volume de dados transferido. Com isso, o tempo de resposta aumentou expressivamente.

Compressão Desabilitada. Desabilitar a compressão aumentou timidamente o tempo de resposta. O aumento na variação foi alto devido à combinação de componentes grandes (não comprimidos) e a inexistência desses componentes em *cache*. Ou seja, a primeira requisição ao componente foi demorada, pois ele não estava comprimido, necessitando maior tempo de transferência. No entanto, requisições seguintes não foram necessárias uma vez que os componentes já estavam em *cache*.

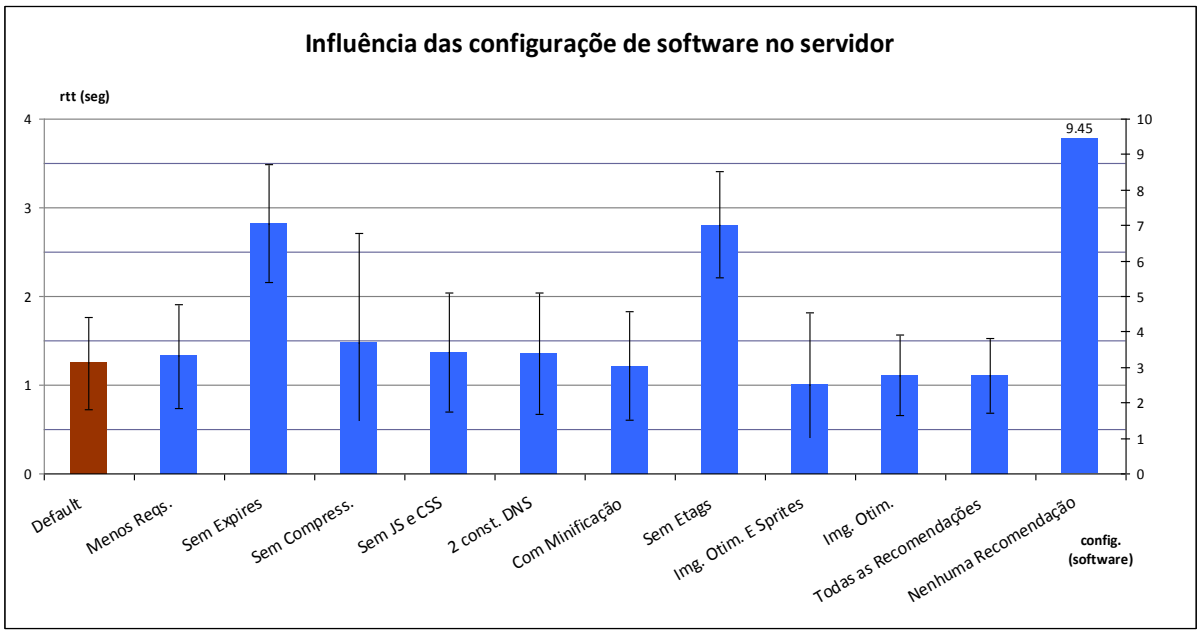


Figura 19. Comparação entre os tempos de resposta da aplicação implementando as recomendações de Souders

JavaScript e CSS *Inline.* A presença do JS e CSS *inline* fez com que os documentos HTML se tornassem maiores. No entanto o aumento no tempo de resposta não foi substancial.

Duas Consultas ao DNS. Ao aumentar o número de domínios com o qual a aplicação interage, foi necessário realizar uma segunda consulta ao DNS. Após realizar a resolução do endereço de domínio em endereço IP na primeira requisição, as solicitações seguintes não sofreram atrasos. O impacto foi pequeno, mas não desprezível. Atualmente não há motivos para servir componentes a partir de múltiplos domínios.

JavaScript e CSS Minificado. Minificar o JS e CSS surtiu pouco efeito. Como, por padrão, o mecanismo de compressão esteve habilitado, a diferença entre o arquivo minificado e original tornou-se pequena, quando comparado ao total de tamanho da página e seus demais componentes. A diferença se tornou ainda mais desprezível pelo fato dos arquivos serem mantidos em *cache* após a primeira solicitação.

Etags Desabilitadas. Assim como no caso do cabeçalho expires, desabilitar as *etags*, implicou na realização de transferências para componentes que já haviam sido baixados anteriormente.

Imagens Otimizadas e Sprites. Através da otimização das imagens e enviando-as combinadas em um único arquivo (*sprite*), foi possível atingir a menor média de tempo de resposta. No entanto, a redução veio ao custo do aumento da variação.

Como todas as fotografias do *site* foram combinadas em um único arquivo, a requisição pela página inicial sofreu um grande atraso. Já as páginas seguintes não tiveram que realizar qualquer solicitação pelas fotografias, pois o *sprite* já se encontrava em *cache*.

Imagens Otimizadas. Otimizar as imagens, mas enviá-las individualmente, resultou na redução do tempo de resposta e de sua variação, quando comparados com a configuração padrão.

Todas as Recomendações. Ao implementar todas as sugestões de Souders, foi obtido o menor tempo de resposta e variação possíveis.

Nenhuma Recomendação. Quando nenhuma recomendação foi implementada, os componentes das páginas não só mantiveram seus tamanhos originais (não comprimidos) como também precisaram ser transferidos novamente a cada solicitação, resultando em tempos de resposta acima de 6 segundos (valor no eixo secundário).

Observa-se, adicionalmente, que algumas configurações têm pouca influência no desempenho médio, mas aumentam consideravelmente a variação dos tempos de resposta: compressão, consultas ao DNS e otimização de imagens e *sprites*.

Também é possível observar que a aplicação de todas as configurações de software recomendadas implica em uma redução do tempo de resposta, mas esta redução não é, na média, muito significativa. Por outro lado, o desvio padrão resultou discretamente menor em relação à configuração padrão, o que pode significar uma experiência melhor para o usuário.

8. Trabalhos Relacionados

Além do trabalho de Souders, usado como base na avaliação de desempenho realizada, outros trabalhos perseguem eficiência nos sistemas Web e para isso investigam os possíveis gargalos em elementos do sistema que possam ser otimizados.

Uma abordagem recorrente para melhorar o desempenho de um sistema Web, mantendo o tempo de resposta das aplicações é aumentar o número de réplicas do servidor, controlando ativos de rede, nós de processamento e acesso a armazenamento de massa (*storage*) e banco de dados. O servidor Apache oferece suporte à manutenção de réplicas do servidor, permitindo que uma das instâncias atue como *frontend*, encaminhando as requisições para os servidores replicados, de acordo com uma política também configurável (Balanceador).

Além de aplicar técnicas de replicação, inclusive com o suporte de hardware (por exemplo, *switches* nível 7), equipes de grandes provedores de acesso e aplicações como o Facebook, se mobilizam constantemente para tornar o acesso de usuários mais

eficiente. Em [Nishtala 2013] a equipe avalia o desempenho do mecanismo de *memcache* em várias configurações e propõe uma arquitetura para otimizar este mecanismo.

Outro exemplo é o tratamento específico dado a diversos níveis de otimização de *cache*, compressão de dados e mecanismos de acesso para mídias distintas na Globo.com. Conhecendo a natureza da aplicação, do tipo de mídia armazenada (texto, aplicações dinâmicas, figuras ou fluxos de vídeo) a equipe desenvolveu sistemas específicos de *cache*, com vários níveis de ação, para permitir a entrega dos diversos tipos de dados com a maior eficiência possível.

A avaliação de sistemas Web, ainda que com foco no servidor, oferece a possibilidade de visões diferentes. Em [Galeote 2006], por exemplo, sistemas Web são avaliados sob a visão de Engenharia de Software e um *checklist* é elaborado. Neste *checklist*, diferente das características avaliadas no presente trabalho, são pesquisados aspectos como o acoplamento, uso de recursos compartilhados e caminho crítico da aplicação que executa no servidor.

[Liu, Niclausse and Villanueva 2001] apresentam um modelo que caracteriza o tráfego entre clientes e servidores Web e realizam uma avaliação de desempenho usando o servidor Apache e um gerador de tráfego desenvolvido pelos autores, o WAGON. O foco da análise é o protocolo HTTP e a condição da rede. Como resultado, verificam os benefícios do HTTP 1.1 comparado à versão 1.0, e a influência de parâmetros como a janela TCP e tempos de timeout.

O trabalho de Abbas e Kumar [Abbas and Kumar 2011], mais próximo do presente trabalho, avalia a percepção do cliente Web em relação ao tempo de resposta, latência e taxa de requisições. Porém diferentemente do presente trabalho a avaliação observa parâmetros da rede que influenciam no tempo de resposta e não os parâmetros de configuração do servidor. Para isso usam aplicações do tipo *ping* com protocolos ICMP, UDP e TCP.

9. Conclusão

Os resultados obtidos nas avaliações comprovam a influência das diversas configurações de software disponíveis na infraestrutura web atual, clientes e servidores, na redução do tempo de resposta das aplicações. Verificou-se que quando nenhuma recomendação esteve habilitada, o tempo de resposta aumentou em média 4 vezes. Os mecanismos de *cache* desempenham papel determinante na redução do tempo de respostas, seguidos pela compressão.

Mecanismos de compressão, como Gzip e DEFLATE, por utilizarem algoritmos sem perdas, podem ser sempre ativados. No entanto, *caches*, dependendo da aplicação, necessitam de maior cautela. Quando um componente for recebido junto de um cabeçalho *expires*, o navegador não hesitará em usar sua versão nas requisições seguintes – mesmo que a versão no servidor tenha sido alterada. Para usar a versão em *cache* com a certeza de que combina com a versão do servidor, é necessário usar *etags*. No entanto, *etags* também devem ser usados com cuidado, uma vez que um dos parâmetros padrão para sua criação é o *inode* do arquivo. Em sistemas distribuídos,

onde um componente pode residir simultaneamente em várias máquinas para balancear a carga, o uso do *inode* no cálculo do *etag* deve ser explicitamente desativado.

O tempo de resposta também foi reduzido através de escalamento vertical, demonstrando que são abordagens complementares, pois incidem em diferentes fases do ciclo de uma requisição. Por outro lado verificou-se que para uma determinada carga, o aumento de recursos tem um limite de influência.

O escalamento vertical permite que mais clientes sejam atendidos simultaneamente sem degradar o desempenho do servidor. Já as configurações de software visando otimizar o uso de recursos reduzem o tempo de transferência em função da redução na quantidade de dados trafegados. Desse modo, usuários com velocidade de acesso limitada, são os mais beneficiados com a aplicação destas otimizações. Para usuários com velocidade de acesso maiores, o gargalo residirá no tempo de processamento do servidor – para tais usuários, o escalamento vertical surtirá mais efeito.

Seguindo a política de redução do volume de dados transferidos, técnicas como AJAX, por permitirem que apenas fragmentos da página sejam requisitados e transferidos, o tempo de resposta pode ser melhorado. No entanto, é mais difícil de ser implementada porque requer alteração na camada de visão da aplicação.

Embora o WWW::Mechanize facilite a interação com web *sites*, seu estado atual fornece pouco suporte a JavaScript, tornando aplicações que façam muito uso de JavaScript difíceis de serem avaliadas. No entanto, se revelou uma boa ferramenta para gerar carga programada.

O protocolo SPDY em desenvolvimento pelo Google é um protocolo da camada de aplicação similar ao HTTP. Ele modifica a maneira como requisições e respostas são trocadas entre cliente e servidor para que sejam mais eficientes, melhorando o tempo de carregamento de uma página web em cerca de 23% [Google 2009]. Embora ainda esteja em desenvolvimento, alguns sistemas em produção, como o Gmail e o mecanismo de busca do Google, já fazem uso desse protocolo. A avaliação sistemática deste protocolo pode ser avaliada em trabalhos futuros.

Outros cenários de teste poderiam ser avaliados. Por exemplo, em um cenário com um balanceador de carga, disponível no servidor Apache, por exemplo, e sistema de *cache* independente, como o *squid*, podem apresentar resultados que nos permitam obter informações complementares às obtidas neste trabalho.

Referências

Amazon Web Service. “Produtos e Soluções de AWS e computação em nuvem” http://aws.amazon.com/pt/products-solutions/, Fevereiro 2014

ANATEL. Resolução No 575, 2011.

Apache – Apachebench, “Apache HTTP server benchmarking tool”, http://httpd.apache.org/docs/2.2/programs/ab.html#synopsis, Janeiro 2013

Berners-lee, Tim. “Biography”, http://www.w3.org/People/Berners-Lee/, Junho 2012

CETIC.br (2010), “Pesquisa Sobre Uso das Tecnologias da Informação e Comunicação no Brasil 2010”, http://www.cetic.br/tic/2010/index.htm, Março 2012.

Flanagan, David. JavaScript O Guia Definitivo. 4a Edição. O’Reilly (2002).

Frisch, Ællen. Essential System Administration. 3rd Edition. O'Reilly (2002).

D. Mosberger and T. Jin.- “httperf: A Tool for Measuring Web Server Performance-Overview Paper”, http://www.hpl.hp.com/research/linux/httperf/docs.php, Março 1998.

Developler, Google (2009), “SPDY perfomance on mobile networks”, http://googledevelopers .blogspot.ca/2012/05/spdy-performance-on-mobile-networks.html, Junho 2012.

Iana (2012), “Service Name and Transport Protocol Port Number Registry”, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt , Junho 2012.

ITU (2010), “Internet Users”, http://www.itu.int/ITU-D/ict/statistics/material/ excel/ 2010/IndividualsUsingInternet_00-10.xls, Abril 2012.

Kozierok, Charles M.. The TCP/IP Guide. 1st Edition. No Starch Press (2005).

Kurose, James & ROSS, Keith. Computer networking: a top-down approach featuring the internet. 5th Edition. Pearson Education (2010).

Linden, Greg. “Marissa Mayer at Web 2.0”, http://glinden.blogspot.com.br/2006/ 11/marissa-mayer-at-web-20.html, Abril 2012.

Linode (2013). “Deploy and Manage Linux Virtual Servers in the Linode Cloud”, https://www.linode.com/, Setembto 2013.

Neumann, Michael (2005). “Mechanize” , http://mechanize.rubyforge.org/HTTP.html

Lester, Andy. “WWW-Mechanize-1.73” http://search.cpan.org/dist/WWW-Mechanize/, Agosto 2013.

Nielsen, Jakob (1993). “Response Times: The 3 Important Limits”, http://www.useit.com/papers/responsetime.html, Março 2012.

Nielsen, Jakob (1997). “The Need for Speed”, http://www.useit.com/alertbox/ 9703a.html, Março 2012.

Nielsen, Jakob (2006). “Participation Inequality: Encouraging More Users to Contribute”, http://www.useit.com/alertbox/participation_inequality.html, Março 2012.

Nielsen, Jakob (2009). “Powers of 10: Time Scales in User Experience”, http://www.useit.com/alertbox/timeframes.html, Março 2012.

Nielsen, Jakob (2010). “Website Response Times”, http://www.useit.com/ alertbox/response-times.html, Março 2012.

Bekman, Stas e Cholet,Eric. Practical mod_perl. 1st Edition (2009).

Raymond, Eric S. The Art of Unix Programming. 1ª Edição. Addison-Wesley (2003), http://catb.org/~esr/writings/taoup/, Novembro,2012

RFC 2616, "HyperText Transfer Protocol 1.1", R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, http://www.ietf.org/, Junho1999.

RFC 2854, "The text/html Media Type", D. Connolly & L. Masinter , http://www.ietf.org, Junho 2000.

RFC 1738, "Uniform Resource Locators (URL)", T. Berners-Lee, L. Masinter, M. McCahill, http://www.ietf.org, Dezembro 1994.

Silberschatz, Abraham; GALVIN, Peter Baer; GAGNE, Greg. Fundamentos de Sistemas Operacionais. 8ª Edição. LTC (2010).

Singhal, Amit; Cutts, Matt Cutts. "Using site speed in web search ranking".

http://googlewebmastercentral.blogspot.com.br/2010/04/using-site-speed-in-web-search-ranking.html, Abril 2012.

Souders, Steve. (2009) "Even Faster Web Sites: Performance Best Practices for Web Developers". 1st Edition. O'Reilly.

Souders, Steve (2007) "High Performance Web Sites: Essential Knowledge for Front-End Engineers". 1st Edition. O'Reilly.

Tanenbaum, Andrew. Redes de Computadores. 4ª Edição. Campus, 2003.

W3C (1998) – "Document Object Model – DOM Level 1 Specification", http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/ , Outubro 1998.

W3C (1999). "RFC 2616 –HyperText Transfer Protocol 1.1", http://www.w3.org/Protocols/rfc2616/rfc2616.html, Maio 2012.

W3Tech. "Historical trends in the usage of content management systems for websites", http://w3techs.com/technologies/history_overview/ content_management/all, Maio 2012.

WORDPRESS. "Web software you can use to create a beautiful website or blog", http://wordpress.org/, Setembro 2013.

Yahoo Developer Network. "Best Practices for Speeding Up Your Web Site", http://developer.yahoo.com/performance/rules.html, Mar 2012.

Nishtala, Rajesh et al (2013) "Scaling Memcache at Facebook", USENIX Association, 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI '13), pp. 385-398.

Galeote, Sidney (2006) "Desempenho em sistemas Web: uma proposta de método para sua avaliação", Dissertação de Mestrado, Instituto de Pesquisas Tecnológicas do Estado de São Pualo – IPT, São Paulo, Novembro.

Abbas, Ash Mohammad and Kumar, Ravindra (2011). "A Client Perceived Performance Evaluation of Web Servers", First International Conference Advances in Computing and Communications, ACC 2011, pp 307-316, Kochi, India.

Liu, Zhen, Niclausse, Nicolas e Jalpa-Villanueva (2001). César. "Traffic model and performance evaluation of Web servers". *Perform. Eval.* 46, 2-3, pp. 77-100.

CADERNOS DO IME
Série Informática

Normas para publicação de trabalhos

A revista Cadernos do IME é o veículo de divulgação dos trabalhos acadêmicos do corpo docente do Instituto de Matemática e Estatística da Universidade do Estado do Rio de Janeiro.

A revista é semestral, sendo publicada em junho e dezembro. São aceitos trabalhos inéditos e que apresentem elementos de originalidade, em português, inglês ou espanhol.
Os Cadernos do IME aceitam os seguintes tipos de contribuições:

- Artigo técnico, trabalho de pesquisa com elementos de originalidade;
- Tutorial, texto introdutório a áreas específicas da ciência da computação, com revisão de literatura e organização dos conceitos relacionados;
- Resenha / Revisão, revisão de literatura e análise crítica de produtos da área de ciência da computação e informática;
- Resumo, texto destinado à informação sobre trabalhos de graduação e de pós-graduação;
- Comunicação, texto destinado à divulgação de opiniões, pesquisas em andamento, lançamentos e divulgação de eventos.

Os artigos e tutoriais não devem ultrapassar 20 páginas e as demais formas de publicação devem se restringir a no máximo 10 páginas.

Todos os trabalhos enviados para publicação nos Cadernos do IME – Série Informática, independente de tipo de contribuição, devem seguir o modelo para publicação de artigos da SBC – Sociedade Brasileira de Computação.

Os arquivos digitais no formato PDF deverão ser encaminhados aos editores da série, juntamente com uma cópia impressa de boa qualidade.

A submissão é eletrônica e em fluxo contínuo no endereço http://www.e-publicacoes.uerj.br/index.php/cadinf. As chamadas de trabalho para volumes especiais estão disponíveis http://www.ime.uerj.br/cadernos/cadinf/.

Os conteúdos e pontos de vista expressos nos trabalhos publicados são de inteira responsabilidade dos autores.