Recebido em: 18/07/08
Aprovado em: 20/06/10

Usando *Ajax* na apresentação de tabelas com grandes volumes de dados em aplicações *web*

David Luiz Ramiris (FATEC, SCS – SP/Brasil) - david_ramiris@yahoo.com.br
• R. Bell Aliance, 225, Jd. São Caetano, São Caetano do Sul-SP, CEP 09581-420
Jardel Capelasso Amante (FATEC, SCS – SP/Brasil) - jardelamante@yahoo.com.br
Alan Henrique Pardo de Carvalho (FATEC, SCS – SP/Brasil) - alancarv@uol.com.br

Resumo

A abordagem *Ajax* promete solucionar os problemas da comunicação assíncrona inerente ao protocolo HTTP, no qual a *Web* é baseada. Este estudo tem como objetivo demonstrar como uma aplicação *Web* utilizando *Ajax*, pode, no que se refere às métricas externas de qualidade de *software* propostas pela norma ISO/IEC 9126, auxiliar no sentido de atingir o mesmo tempo de resposta e nível de experiência dos *softwares* aplicativos tradicionais, desenvolvidos em *Clipper* e *Visual Basic*, nas consultas a tabelas com dezenas de milhares de linhas. Foram utilizadas as linguagens *JavaScript* e PHP para usufruir dos recursos *Ajax*, no desenvolvimento da aplicação. Após utilizar algoritmos de consulta e apresentação baseados em conceitos e práticas avançadas, utilizadas por especialistas nas outras linguagens, foram feitas comparações entre os resultados obtidos pela aplicação, utilizando *Ajax* e os programas escritos em *Clipper* e *Visual Basic*, segundo a metodologia apresentada pela norma. Pode-se constatar que a aplicação *Ajax* foi capaz de atingir as métricas de qualidade de *software* propostas pela norma ISO/IEC 9126 e, em alguns casos, seu desempenho foi superior aos das aplicações escritas em *Clipper* e *Visual Basic*, abrindo promissoras possibilidades para o desenvolvimento de soluções multiplataforma, que necessitem acessar grandes bases de dados rapidamente.
Palavras-chave: *Ajax*; tabela; *grid*; métricas; XML; *XmlHttpRequest*.

Abstract

The Ajax approach promises to solve the asynchronous communication problems related to the HTTP protocol, on which the Web is based. This article aims to demonstrate how a web application that uses Ajax can, with a reference to the external software quality metrics proposed by the ISO/IEC 9126 standard, assist in achieving the same response time and experience level from those traditional applications developed with Clipper and Visual Basic, when accessing data tables containing tens of thousands of rows. JavaScript and PHP languages were used in order to take advantage of the Ajax resources in the application development. After using search and presentation algorithms based on advanced routines and concepts used by specialists in other languages, comparisons were made between the application results using Ajax and the software developed in Clipper and Visual Basic, according to the ISO/IEC 9126 standard. It was possible to see that the Ajax application was able to achieve the software quality required by ISO/IEC 9126 standards and in some cases the performance was even better than Clipper and Visual Basic applications, opening up possibilities for multitasking solution development which require rapid access to huge databases.
Keywords: Ajax; table; grid; metrics; XML; XmlHttpRequest.

1. INTRODUÇÃO

Uma das métricas em qualidade de *software* é o tempo de resposta, ou seja, "o tempo despendido até que o aplicativo complete uma tarefa específica" (ISO, 2002). Uma *interface* com *grid* de muitas linhas eficiente é aquela que permite ao usuário, na maior parte do tempo, visualizar e manipular dados, ao invés de ficar aguardando que eles estejam disponíveis.

Em *softwares* aplicativos síncronos, diversas metodologias são empregadas, a fim de diminuir o tempo de resposta: trabalhos em segundo plano, multitarefa preemptiva e estruturas de dados com ponteiros, entre outras, como expõem Machado e Maia (1997).

Uma das formas de minimizar o problema, é dividir esta grande quantidade de dados em parcelas menores, priorizando o carregamento e a apresentação das parcelas que realmente estão sendo manipuladas ou visualizadas pelo usuário em cada momento. Esse é, inclusive, o método utilizado por muitos *softwares* aplicativos, desenvolvidos nas mais variadas linguagens.

Com o uso da abordagem *Ajax*, surge a possibilidade de se utilizar esta metodologia em *softwares* aplicativos, baseados na *Web* 2.0. Assim, o objetivo deste trabalho é verificar se é possível, com *Ajax*, aproximar o tempo de resposta nas visualizações de tabelas com grande quantidade de linhas aos obtidos em programas escritos em *Clipper* e *Visual Basic*. Essas duas linguagens foram escolhidas, neste estudo, por serem comumente utilizadas em aplicativos comerciais de pequeno e médio porte.

Para tanto, foram desenvolvidos códigos para medição do tempo de resposta, usando *Ajax* e nas duas linguagens citadas, método este em conformidade com o item 8.4.1.1 (*Time behavior metrics*) item "a" (*response time*) da norma *ISO/IEC TR 9126-2 Software engineering - Product quality*, que define:

a) Recomenda-se levar em consideração a análise estatística de várias execuções.
b) O tempo de resposta requerido pode ser derivado de necessidades de execução em tempo real, expectativas do usuário, necessidades de negócio ou observações das reações dos usuários. O reconhecimento dos aspectos da ergonomia humana podem ser considerados.

Partindo do princípio de que a formatação, estilos e apelos visuais utilizados na apresentação dos dados são muito específicos para cada propósito e fortemente dependentes de fatores fora do escopo do trabalho, as medições foram feitas utilizando o menor número possível de código específico de formatação. Assim, tanto no *Visual Basic* quanto no *Clipper*, não foi aplicado qualquer tipo de apelo visual, que não fosse padrão ou básico na linguagem. No caso do código, utilizando *Ajax*, os estilos ou formatação aplicados foram os mínimos necessários para se conseguir os mesmos resultados visuais e organização dos dados obtidos com as outras duas linguagens.

No escopo deste trabalho, tabela refere-se a uma quantidade x de linhas (ou registros) e y de colunas (ou campos), com numeração sequencial iniciada em 1. Uma página de dados refere-se a uma área de visualização de n linhas e m colunas, que mostram uma parcela de x e y em cada momento. Avançar uma página significa avançar n linhas em x, e, por consequência, retroceder uma página, significa retroceder n linhas em x.

Por exemplo: apresenta-se na aplicação uma janela com 20 linhas e 2 colunas, visualizando uma tabela com 10.000 linhas e 10 colunas. A primeira página é a apresentação dos 20 primeiros registros da tabela. A segunda página é a apresentação dos 20 registros seguintes. A última página (500) é a apresentação dos 20 últimos registros da tabela. Avançar uma página, é apresentar os 20 próximos registros da tabela e retroceder uma página, é apresentar os 20 registros anteriores. Ir para a página 141, significa apresentar os registros de 2801 a 2820.

O tempo de resposta medido é a parcela de tempo entre um comando do usuário e o momento em que o programa processou a solicitação e está pronto para receber novos comandos do usuário. No comando, pode-se iniciar a apresentação dos dados, avançar ou retroceder uma página, ir para a primeira página, ir para a última página e localizar um registro.

2. A ABORDAGEM *AJAX*

Ajax (*Asynchronous JavaScript* + XML) não é uma nova tecnologia para desenvolvimento de aplicações *Web*, mas uma abordagem baseada na combinação de algumas tecnologias definida por Garret (2005), quais sejam:

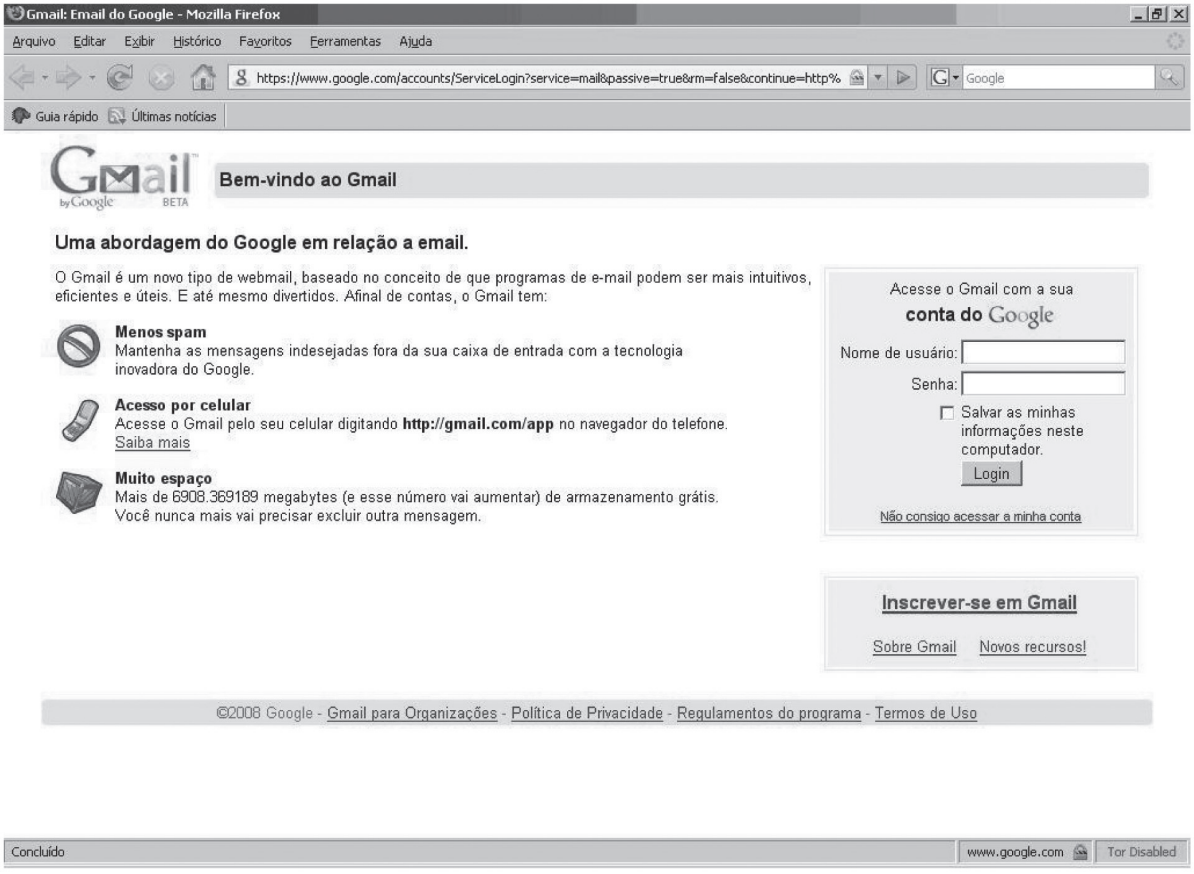
a) XHTML (*eXtensible HyperText Markup Language*) e CSS (*Cascading Style Sheets*), linguagens utilizadas na definição de estruturação e apresentação das páginas da *Web*;
b) DOM (*Document Object Model*), por meio do qual é possível maior interação e dinamismo da aplicação;
c) XML (*eXtensible Markup Language*) e XSLT (*eXtensible Stylesheet Language Transformations*), linguagens para manipulação e intercâmbio de dados;
d) *XMLHttpRequest*, o objeto que será utilizado na recuperação de dados e
e) *JavaScript*, a linguagem de programação que permitirá unir as tecnologias listadas.

Normalmente, em uma aplicação *Web*, determinadas ações do usuário disparam um pedido, via protocolo HTTP, para o servidor. O servidor efetua algum processamento, que pode envolver acesso a outros servidores (como um de banco de dados, por exemplo) e então, retorna uma página HTML para o cliente (*browser* ou navegador).

Esse mecanismo acaba prejudicando a interação do usuário com a aplicação, uma vez que ela se dará em "ciclos", que serão interrompidos a cada nova página que deva ser carregada. O intuito de *Ajax* é minimizar a necessidade de carregamento de páginas, diminuindo a necessidade de espera por parte do usuário e com isso, melhorando a experiência que terá frente à aplicação que fizer uso dessa abordagem.

Um exemplo de aplicação que utiliza *Ajax* é o *GMail*, conhecida aplicação de *webmail*. Em sua página inicial, pode-se observar o formulário para inserção do nome de usuário e senha. Fosse uma aplicação "não-*Ajax*", uma vez que esses dados fossem inseridos e o botão "*Login*" acionado, a consulta ao banco de dados de usuários para autenticação, implicaria na necessidade de recarregar a página toda, no caso de erro de nome de usuário ou senha. No entanto, qualquer usuário do *GMail* observa que somente a parte da página relativa ao formulário é atualizada, caso os dados de *login* sejam fornecidos incorretamente.

Figura 1 – Página de entrada no serviço *GMail*, que utiliza extensivamente a abordagem *Ajax*.



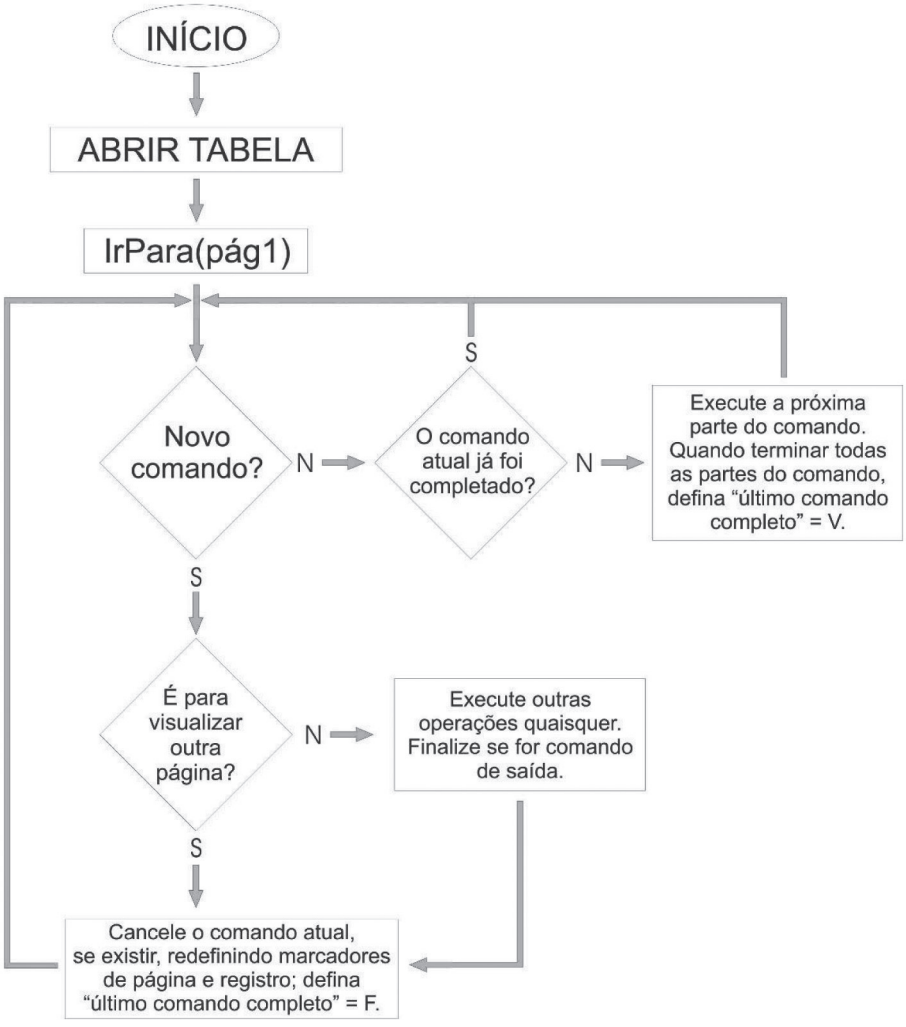
Fonte: Captura de tela de acesso ao site *Gmail*, realizada pelos autores.

Apesar de haver desvantagens no uso dessa abordagem apontadas por *Wikipédia* (2008), como a dependência de *JavaScript* e mesmo na catalogação por mecanismos de busca, *Ajax* vem sendo adotada em aplicações desenvolvidas por companhias, como *Yahoo!* e *Google*, para citar dois exemplos de *players* de grande importância na *World Wide Web*.

3. A APLICAÇÃO *CLIPPER*

Para a navegação em tabelas de muitas linhas, o algoritmo proposto por Spence (1994), é interessante do ponto de vista do tempo de resposta. No diagrama em blocos, fica evidente como o programa se comporta, a fim de entregar, o mais rapidamente possível, o controle do programa ao usuário:

Figura 2 – Diagrama em blocos do algoritmo adotado por Rick Spence.



Fonte: Adaptado de Spence (1994).

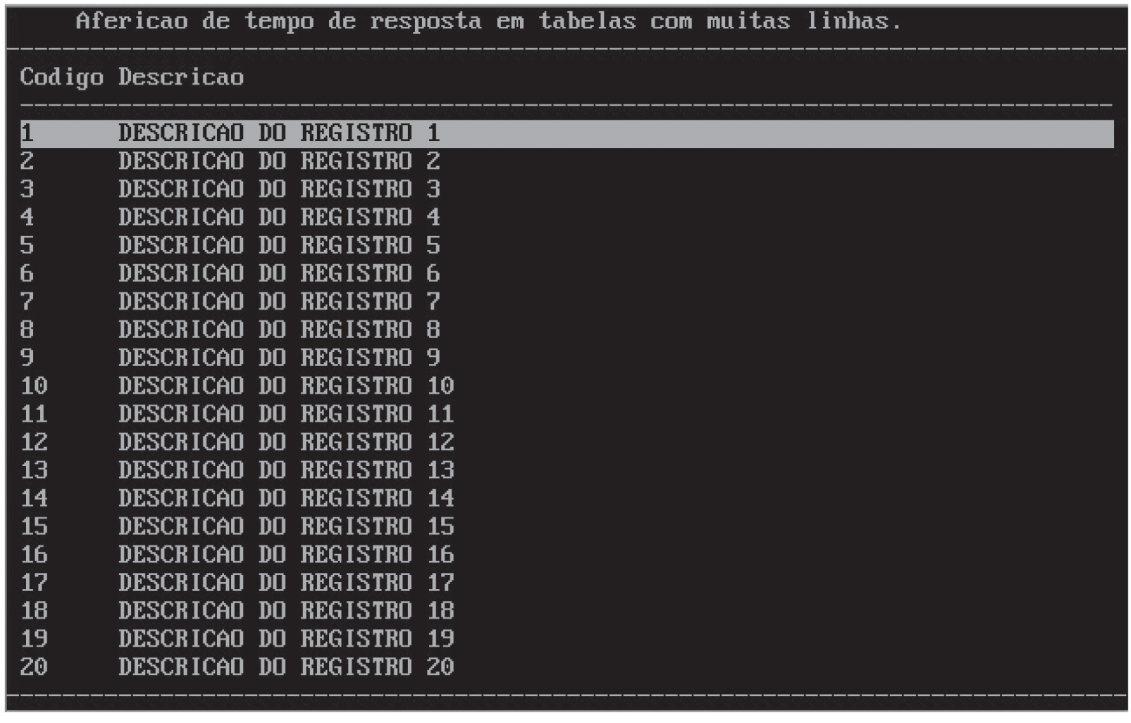
Neste algoritmo, entende-se por "parte" do comando, uma parcela do processamento completo, necessário para responder adequadamente ao comando do usuário. No caso específico, apresentar uma nova página de dados, implica em posicionar o ponteiro do banco de dados no registro correto, carregar e apresentar, na tela, os dados de cada registro, sequencialmente, até que o último registro visível seja apresentado. Neste momento, define-se uma *flag* "último comando completo" como "verdadeiro".

Nota-se, neste caso, o forte apelo à assincronia entre execução do último comando e permissão ao usuário de efetuar novos comandos. Na prática, reduzimos o tempo de resposta do aplicativo, melhorando a experiência no uso do programa.

É certo que o algoritmo proposto deverá ser modificado ao ser implementado como uma aplicação *Web*, visto que ele foi aplicado a uma linguagem estruturada e não em uma orientada a objetos. Mesmo assim, a ideia principal deve ser mantida, que é a de devolver o controle ao usuário, o mais rápido possível e garantir que o processamento da última solicitação seja efetuado.

O código foi escrito na linguagem *Clipper* 5.2, utilizando o algoritmo de Spence e os objetos *TBrowseDB* e *TBColumn*, que têm a finalidade específica de apresentação de dados em formato de tabela. Salvo algumas alterações, no sentido de aferir o tempo de resposta e certo refinamento aplicado ao código de modo que ele se enquadre na metodologia deste trabalho, trata-se de uma reprodução fiel do código, proposto por Spence.

Figura 3 – Instantâneo de tela do programa em *Clipper* 5.2.

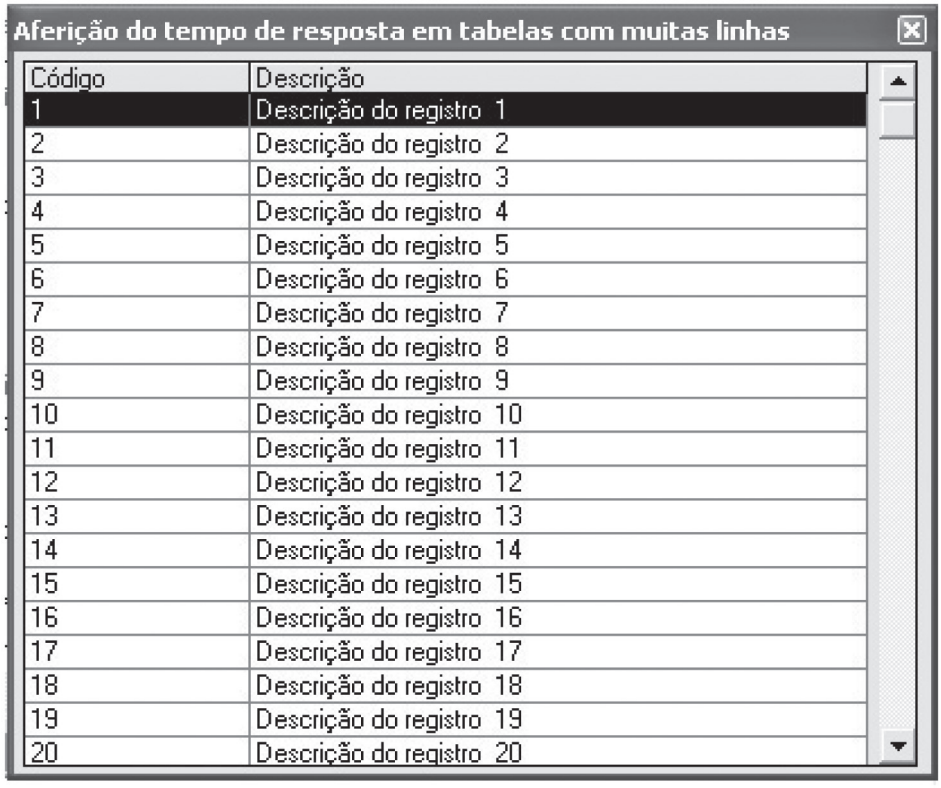


Fonte: Captura de tela do aplicativo em *Clipper* 5.2, realizada pelos autores.

4. O CÓDIGO EM *VISUAL BASIC*

O código a ser utilizado será uma estrutura de apresentação comum em *Visual Basic*, que é a utilização do controle *MS Data Bound Grid* 5.0 SP3. Como mostra *The VB Programmer* (s.d.), esse controle já fornece todas as funções necessárias para a correta apresentação dos dados, restando apenas algumas tarefas a serem executadas pela aplicação em si: a recuperação e formatação dos dados.

Figura 4 – Instantâneo de tela do programa em *Visual Basic*.



Fonte: Captura de tela de aplicativo em *Visual Basic*, realizada pelos autores.

5. A APLICAÇÃO *WEB* BASEADA EM *AJAX*

O foco principal no desenvolvimento do algoritmo foi reduzir, ao máximo, o tempo de resposta. Para isso, ele deve se comportar conforme as seguintes características:

a) Aceitar novas solicitações o mais rápido possível: antes de um comando ou pedido entrar em execução, ele deve ser reconhecido e classificado pela aplicação. O algoritmo desenvolvido trabalha com processamento incremental, ou seja, as rotinas de apresentação têm um tempo máximo de execução permitido. Ao atingir o limite de tempo, a apresentação é paralisada, verificam-se e classificam-se novos comandos e, se o fluxo não foi alterado, retoma o processamento interrompido.

b) Priorizar processamentos: do ponto de vista do usuário, o processamento mais importante é aquele que devolve a ele o controle do programa mais rapidamente. O algoritmo desenvolvido dá prioridade ao que o usuário está vendo, ou seja, o processamento e a apresentação dos dados realmente utilizados pelo usuário, em cada parcela de tempo.

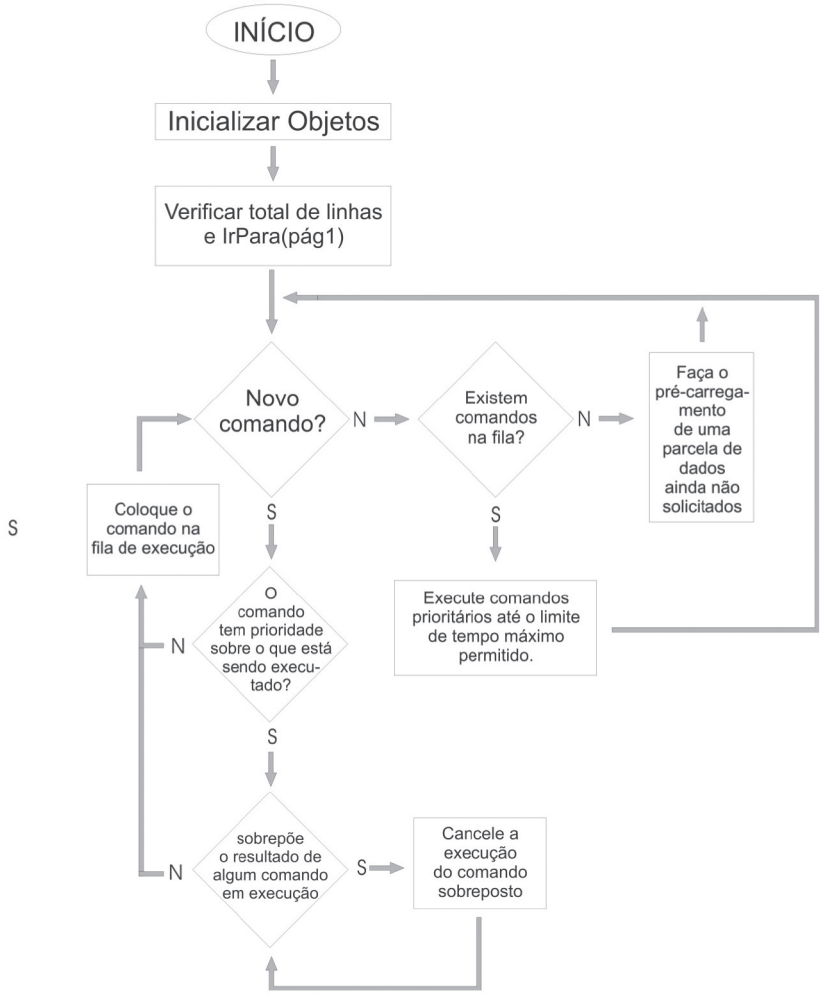
c) Descartar processamentos desnecessários: uma vez que um processamento se inicia, outros eventos podem ocorrer – e normalmente ocorrem – durante sua execução, que afetam diretamente o resultado final esperado. Neste caso, não é justificável continuar o processamento "ultrapassado"; deve-se descartar a solicitação e orientar os esforços, a fim de se obter um resultado válido. É o caso, por exemplo, do avanço de páginas. Se o usuário avança certa quantidade de páginas, não há necessidade de atualizar a apresentação das páginas intermediárias, apenas da última ou das últimas páginas.

d) Utilizar ao máximo o tempo livre, a fim de adiantar processamentos futuros: quando o usuário não está solicitando novos comandos ou aguardando a retomada do controle do programa, os recursos de processamento se tornam disponíveis para outras execuções, em segundo plano, que possam, de uma maneira ou de outra, diminuir o tempo necessário para os próximos comandos do usuário.

Claramente, o programa não pode "adivinhar" quais serão as novas solicitações, como, por exemplo, "finalizar programa", mas deve oferecer mecanismos que pré-executem ações futuras comuns. No caso da navegação por tabela de muitas linhas, em ambiente *Web*, o maior consumo de tempo se dá na carga dos dados. Assim, o algoritmo utiliza o tempo "ocioso" para pré-carregar o conteúdo da tabela, mesmo que ainda não tenha sido solicitado pelo usuário.

No diagrama em blocos, podem-se verificar os mecanismos utilizados, a fim de diminuir o tempo de resposta do algoritmo.

Figura 5 – Diagrama em blocos do algoritmo em *JavaScript*, usando *Ajax*.

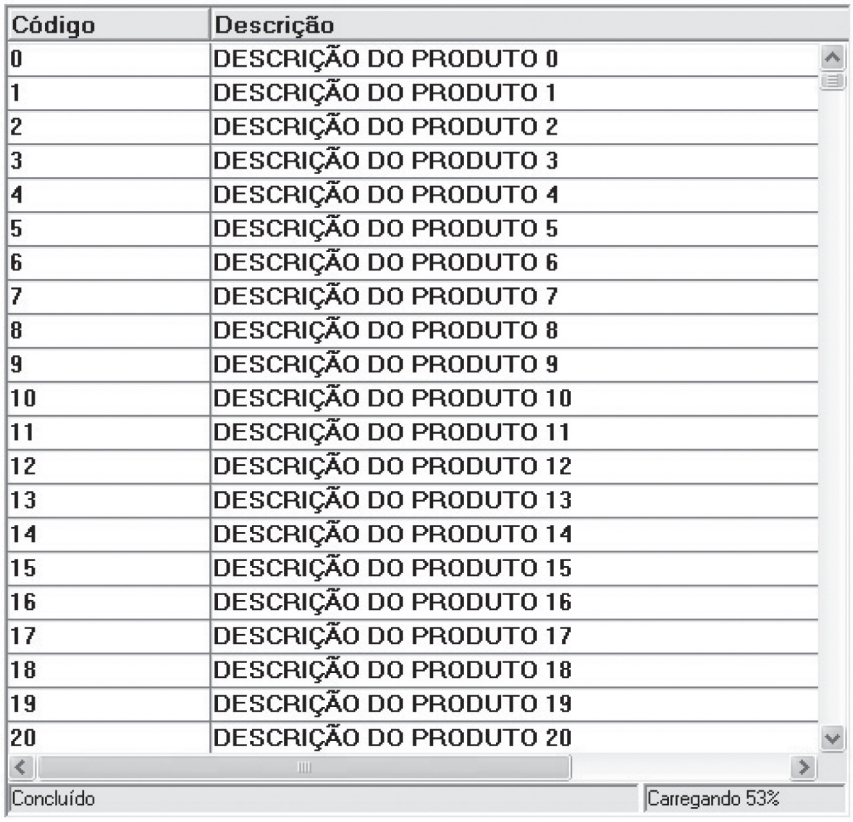


Fonte: Elaborado pelos autores.

É importante salientar que o pré-carregamento não deve consumir a memória de processamento em níveis que prejudiquem o desempenho geral da aplicação. Tabelas com milhares de linhas podem facilmente exigir centenas de milhares de *megabytes* de memória RAM, para serem pré-carregadas. Desta maneira, o algoritmo só ativará o recurso de pré-carregamento, quando a tabela contiver um máximo de 10.000 linhas. Porém, este valor deve ser ajustado, conforme os requisitos e objetivos de cada aplicativo. Além disso, deve-se levar em consideração o aumento da "janela virtual", ou seja, da margem que se considera "dados úteis", quando o recurso é desabilitado.

A seguir, pode-se observar um instantâneo da tela da aplicação *Ajax*, com uma janela de 20 linhas de dados.

Figura 6 – Instantâneo de tela da aplicação *Web Ajax*.



Fonte: Tela da aplicação *Web Ajax*, capturada pelos autores.

6. TESTES E RESULTADOS

Foram utilizadas tabelas de 1.000, 10.000 e 100.000 registros, com duas colunas, COD e DESCRIÇÃO e o mesmo conteúdo em todos os casos.

No caso do *Clipper*, o banco de dados estava em formato DBF. No *Visual Basic*, foi utilizado um banco de dados do *Access* e, finalmente, para a aplicação *Web Ajax*, foi utilizado um banco de dados baseado no servidor de banco de dados relacional MySQL, sendo a página elaborada com a linguagem PHP.

Seguindo a orientação da norma ISO/IEC 6.192, foram efetuados 20 testes em cada quesito. Os cálculos para a média utilizaram a fórmula proposta pela norma, como segue:

T = tempo, em milésimos de segundos, entre a solicitação do comando e o fim da execução
N = Quantidade de execuções
Tm = tempo médio, dado por Tm = Σ(T)/N
Td = tempo médio desejado
R = resultado, ou seja, o percentual atingido da meta, dado por Td/Tm*100

As métricas foram obtidas para os seguintes comandos:

a) Tempo para inicialização;
b) Tempo para avanço e retrocesso de uma página;
c) Tempo para ir para a última página, a partir do início e a partir de 30% e 60% do volume total de dados;
d) Tempo para ir para a primeira página, a partir do início e a partir de 30% e 60% do volume total de dados.

Os testes apresentaram os seguintes resultados:

Tabela 1 – Resultados obtidos na comparação com *Clipper* e *Visual Basic*.

Métrica	Comando			
	a	b	c	d
Tm *Clipper*	1237	89	93	101
Tm *Visual Basic*	2250	211	198	232
Tm *Ajax*	1750	186	212	207
R *Ajax->Clipper*	54,98%	42,18%	46,97%	43,53%
R *Ajax -> Visual Basic*	128,57%	113,44%	93,40%	112,08%

Fonte: Elaboração dos autores.

7. CONCLUSÃO

Utilizando corretamente os recursos da nova tecnologia, ficou claro ser possível atingir as métricas de qualidade já obtidas com as linguagens comuns e, em alguns casos, o desempenho da aplicação *Web Ajax* chegou a ser superior ao obtido pelos programas escritos em *Clipper* e *Visual Basic*.

Assim, abrem-se possibilidades de unir grandes vantagens da *Web* 2.0, entre elas, a computação em nuvem, com o desempenho e a usabilidade dos aplicativos executados localmente, desde que, para isso, o projeto de *software* leve em consideração não apenas as características dessa nova metodologia, bem como faça uso de todas as ferramentas e recursos disponibilizados pelo *Ajax*.

Com o *Ajax*, estamos mais próximos dos *softwares* realmente multiplataforma: basta ter um navegador compatível para poder usufruir do *software* em sua totalidade, e com a mesma qualidade, usabilidade, padronização e objetivos para qual a ferramenta foi desenvolvida. E a realidade não é distante – a maioria dos navegadores, atualmente utilizados oferece suporte ao protocolo HTTP assíncrono e cada vez mais o respeito à padronização torna-se característica importante das tecnologias que envolvem a *Web*.

8. REFERÊNCIAS BIBLIOGRÁFICAS

GARRET, J. J. **Ajax: A New Approach to Web Applications**. San Francisco: Adaptative Path, 2005. Disponível em <http://www.adaptivepath.com/ideas/essays/archives/000385.php>. Acesso em 09/07/2008.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. ISO/IEC TR 9126-2 **Software engineering – Product quality – Part 2: External metrics**. Genebra: ISO SC7/WG6 Editors, 2002.

MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais**. 2ª ed. Rio de Janeiro: Livros Técnicos e Científicos Editora, 1997.

SPENCE, R. **CA-Clipper 5.2**. São Paulo: Makron Books, 1994.

THE VB PROGRAMMER. **Database access with the data control**. New Jersey: The VB Programmer. Disponível em <http://www.thevbprogrammer.com/Ch11/11-02-DAODC.htm>. Acesso em: 09/07/2008.

WIKIPEDIA, **Ajax** (programming). San Francisco: Wikimedia Foundation, 2008. Disponível em <http://en.wikipedia.org/wiki/AJAX>. Acesso em 09/07/2008.