CRIANDO APLICAÇÕES WEB RICAS (RIA) COM APARÊNCIA DE DESKTOP UTILIZANDO O *FRAMEWORK* AJAX EXT JS

CREATING RICH INTERNET APPLICATIONS WITH DESKTOP APPEARANCE USING THE FRAMEWORK AJAX EXT JS

Gleydson Lucínio[1]
Eder Carlos Salazar Sotto[2]

RESUMO

Este artigo tem como objetivo apresentar o *framework* Ext JS como alternativa para o desenvolvimento de RIAs[3] (aplicações ricas para a internet), aproveitando seus excelentes recursos, que permitem a criação de aplicações com usabilidade e aparência de *desktop*. O Ext JS é um *framework client-side* (executado no navegador do usuário) escrito em *JavaScript*. Sua execução é *cross-browser* (compatível com vários navegadores). Também possui suporte à orientação a objeto, e é compatível com o padrão MVC (modelo-visão-controle).

Palavras-chave: AJAX. EXT JS. JavaScript. RIA. Aplicações Web Ricas.

ABSTRACT

This article aims to present the Ext JS framework as an alternative for the developing rich Internet applications (RIA), taking advantage of its great features, which enable the creation of applications with desktop usability and appearance. The Ext JS is a client-side framework (executed in the user's browser) written in JavaScript, its execution is cross-browser (compatible with multiple browsers), it presents support for object orientation, and is compatible compatible with the MVC standard (model-vision-control).

1. Graduando do curso de Sistemas para Internet da Fatec Taquaritinga. E-mail: gleydsonlucinio@gmail.com.
2. Docente da Fatec Taquaritinga. E-mail: eder.sotto@fatectq.edu.br.
3. RIA: *Rich Internet Application*.

Keywords: AJAX. EXT JS. JavaScript. RIA. Rich Web Applications.

INTRODUÇÃO

O movimento de tornar a Web mais amigável ao usuário com a criação de RIAs foi nomeado “Web 2.0” e sua existência foi possível, em grande parte, graças a um conjunto de tecnologias denominado AJAX.[1]Uma aplicação avançada de Internet (RIA) combina a usabilidade de uma aplicação *desktop* com a flexibilidade de implantação baseada na Web. Existem duas abordagens principais de RIAs. A primeira utiliza *plug-in*[2] no navegador para criação do ambiente de execução, como o Flash, Silverlight e o Java. A segunda utiliza *frameworks* de extensão baseados no *JavaScript* como Dojo, Ext JS, jQuery, Prototype, entre outros. Cada uma das abordagens tem suas vantagens e desvantagens.

A utilização de *frameworks JavaScript* é uma opção muito utilizada para criação de RIAs, pois o *JavaScript* é suportado pelos principais navegadores *desktop* e *mobile*, não sendo necessária a instalação de *plug-ins* adicionais. Os *frameworks JavaScript* rodam diretamente no navegador do usuário e utilizam AJAX para comunicação com o servidor.

1. Apresentando o Ajax

O termo AJAX é um acrônimo em língua inglesa de ***A****synchronous* ***J****avaScript* ***a****nd* ***X****ML*,criado em 2005 por Jesse James Garrett para nomear o uso conjunto das tecnologias, as quais já eram utilizadas e passaram a ser referenciadas como AJAX a partir de então.

Para Garcia (2001), o AJAX surgiu da necessidade dos desenvolvedores em criar aplicações interativas que pudessem ser disponibilizadas na Web. Com o AJAX, é possível criar aplicações Web Ricas (RIAs), nas quais o conteúdo é carregado em segundo plano enquanto o usuário navega pela página, sem que seja necessário interromper a navegação para que uma nova página seja carregada.

O AJAX utiliza a linguagem *JavaScript* para modificar diretamente a UI,[3] utilizando o

1. AJAX: Asynchronous JavaScript and XML.
2. Plug-in: Programa instalado no navegador do usuário que permite a execução de código compatível.
3. UI: User Interface.

objeto XMLHttpRequest para se comunicar com o servidor em segundo plano, tornando-se transparente ao usuário. A aplicação, dessa forma, usa a informação retornada do servidor, normalmente em XML[1] ou JSON[2] para atualizar a UI (GARCIA, 2011).

As figuras 1 e 2 mostram a diferença entre uma aplicação Web típica e uma aplicação Web AJAX, na qual é possível verificar que, enquanto no primeiro modelo a navegação é interrompida durante o carregamento da página pelo navegador, no segundo (que utiliza AJAX) verificamos que isso não ocorre, visto que o carregamento das informações do servidor passa a ser assíncrono e em segundo plano, sem interromper a interação do usuário com a aplicação.

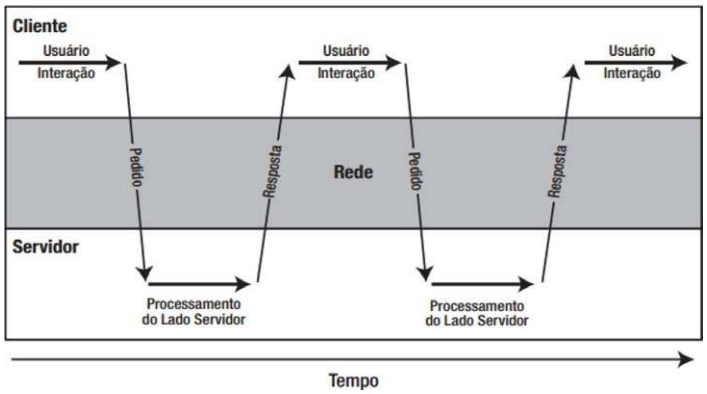


Figura 1. Modelo de interação de uma aplicação Web típica.
Fonte: Adaptado de Deitel (2008).

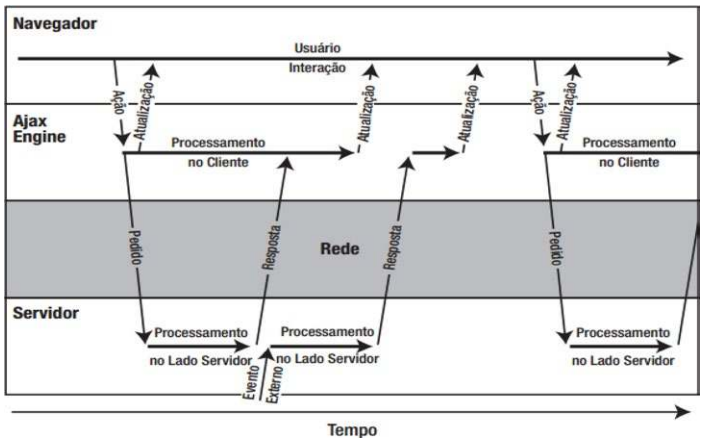


Figura 2. Modelo de interação de uma aplicação AJAX.
Fonte: Adaptado de Deitel (2008).

1. XML: Extensible Markup Language.
2. JSON: JavaScript Object Notation.

2. O framework Ext JS

Criado em 2006 por Jack Slocum para ser uma extensão do *framework* YUI[1] e, inicialmente batizado de YUI-ext, o Ext JS teve, com o passar do tempo, um grande crescimento de seus recursos e componentes de interface, o que permitiu ser reconhecido por muitos desenvolvedores e colaboradores pelo mundo, influenciando no surgimento de novas versões em um ciclo reduzido de tempo, até a chegada da versão atual (versão 5.0) (SUNDERARAMAN, 2013).

O Ext JS é um *framework cross-browser* (suporta múltiplos navegadores) escrito em *JavaScript* para o desenvolvimento em RIA, que permite desenvolver aplicações complexas para a Web com aparência e recursos comumente encontrados apenas em aplicações *desktop* (FREDERICK, 2010). Atualmente, o Ext JS é suportado oficialmente pelos navegadores Internet Explorer 6+, Firefox 1.5+, Safari 3+ e Opera 9+, além de possuir dois tipos de licença: comercial e *open source*.

A licença *open source* é distribuída através da GPL[2] v3, também conhecida como GNU, ou seja, esta licença é válida apenas se a aplicação desenvolvida com a tecnologia também for licenciada sobre os termos da GNU.

Segundo Groner (2013), por ser um *framework frontend*, o Ext JS se enquadra perfeitamente na camada de visão do padrão MVC,[3] na qual o desenvolvimento da aplicação Web é dividido em três camadas (visão, modelo e controle). Sua execução é *client-side* (roda no navegador do usuário), o que proporciona uma boa velocidade de execução e menor consumo de banda, uma vez que a aplicação, após carregada, passa a se comunicar com o servidor Web apenas para envio e recuperação de informações solicitadas pela aplicação.

O Ext JS permite ao desenvolvedor criar aplicações Web com múltiplas janelas de forma simplificada, sem a necessidade de um *Web designer*, já que toda estrutura visual é descrita por código, utilizando os componentes já existentes no *framework*, tornando a interface da aplicação muito semelhante à uma aplicação *desktop* (FREDERICK, 2010).

A aplicação escrita no *framework* Ext JS pode ser executada juntamente a qualquer linguagem de programação Web *server-side*, como por exemplo PHP,

1. YUI: Yahoo! User Interface.
2. GPL: General Public Licence.
3. MVC: Model-view-controller.

Python, Ruby, Java, ASP.NET, ou qualquer outra linguagem que tenha suporte para comunicação em JSON ou XML, formatos suportados pelo Ext JS (SUNDERARAMAN, 2013).

3. Frameworks para a camada de visão

Segundo Sunderaraman (2013), desenvolver uma aplicação RIA sem o uso de um *framework* pode ser um trabalho muito repetitivo e complexo, pois exige uma grande quantidade de código e validações para diferentes recursos como formulários, envio de requisições AJAX, manipulação do DOM,[1] *layout*, etc. Esta é a principal razão da criação da grande quantidade de *frameworks JavaScript* que se tem hoje.

Abaixo são apresentados outros *frameworks JavaScript* muito utilizados atualmente (SUNDERARAMAN, 2013).

- **Prototype:** um dos marcos iniciais da linguagem *JavaScript* que fornece um conjunto de utilitários para trabalhar com o DOM e AJAX. Não oferece componentes de interface prontos, como o Ext JS. Pode ser utilizado para tarefas simples como validações de formulários, envio de requisições AJAX e operações DOM.
- **jQuery:** biblioteca muito popular e que fornece uma API[2] fácil de usar. O jQuery UI, que é construído sobre o núcleo do jQuery, oferece vários controles de interface do usuário. É possível estendê-lo às necessidades através do uso de *plug-ins*.
- **DOJO:** Dojo Toolkit traz muitos recursos de orientação a objetos em *JavaScript*. É um conjunto de ferramentas completo com controles de interface de usuário e pode ser facilmente utilizado na construção de aplicativos empresariais. DOJO, no entanto, é criticado por sua falta de coerência na documentação e por problemas de desempenho.
- **GWT:** Google Web Toolkit é uma biblioteca Java compilada para *JavaScript*. A GWT API permite produzir código *JavaScript* altamente otimizado que pode ser implantado em um servidor Web. Naturalmente, esta biblioteca é voltada para desenvolvedores Java.

1. DOM: Document Object Model.
2. API: Application Programming Interface.

- **JS angular:** é uma biblioteca de código aberto do Google. É popular com o uso de MVC devido a esta capacidade de integração. Sua principal desvantagem é a dificuldade de aprendizagem, em grande parte devido a sua pobre documentação e exemplos de código.

4. Principais vantagens do uso do framework Ext JS no desenvolvimento de RIAs

Para Ashworth (2012), as principais vantagens do uso do *framework* Ext JS são:

- Grande quantidade de componentes visuais;
- Desenvolvimento *Cross-browser* (independente de navegador);
- Componentes prontos para integração com servidor através de AJAX;
- Suporte nativo à MVC e acesso nativo ao DOM;
- Suporte para criação de *layout responsivo*;
- Orientação a objetos;
- Por ser escrito em *JavaScript*, não necessita a instalação de *plug-in* adicional;

5. Exemplos de uso dos principais recursos do Ext JS

Abaixo serão apresentados exemplos de uso dos principais componentes do Ext JS. Os códigos utilizados foram escritos na versão 3.2 do *framework*, com o objetivo de mostrar a facilidade de sua utilização; então contemplam o código a ser utilizado no lado do servidor. Exemplos mais detalhados poderão ser encontrados na documentação disponibilizada no site do desenvolvedor, bem como nas obras referenciadas ao final deste artigo.

O *download* de todas as versões do *framework* Ext JS poderá ser realizado no site da SENCHA através do endereço http://www.sencha.com/products/extjs/download/.

5.1. Inicialização do frameworkExt JS

Para utilização do *framework* Ext JS é necessário importar os arquivos necessários para sua inicialização no documento HTML,[1] conforme apresentado

1. HTML: HyperText Markup Language.

na listagem 1. Para todos os demais exemplos, é necessário antes repetir este processo para que o *framework* seja inicializado.

```
<!-- IMPORTA OS ARQUIVOS CSS NECESSÁRIOS PARA FUNCIONAMENTO -->
<link rel="stylesheet" type="text/css" href="../extjs/resources/
css/ext-all.css">
<script src="../extjs/adapter/ext/ext-base.js"></script>
<script src="../extjs/ext-all.js"></script>
<script src="../extjs/src/locale/ext-lang-pt_BR.js" ></script>

<!-- INICIA O CÓDIGO JAVASCRIPT -->
<script language="text/javascript">
// MÉTODO onReady EXECUTARÁ O CÓDIGO APENAS QUANDO O FRAMEWORK
ESTIVER TOTALMENTE CARREGADO
Ext.onReady(function(){
      // NESTE PONTO DEVE SER INSERIDO O CÓDIGO
});
</script>
```

Listagem 1. Código-fonte para inicialização do *framework* Ext JS

5.2. Caixa de diálogo com botão e ícone utilizando o método MessageBox.Show

De acordo com Garcia (2011), o método *MessageBox.Show* fornece uma interface para exibir a mensagem usando qualquer combinação das opções disponíveis. Na janela é mostrada a mensagem, a qual pode conter formatação HTML e texto de múltiplas linhas, além de um ícone e botões com texto configurável, conforme apresentado na figura 3.

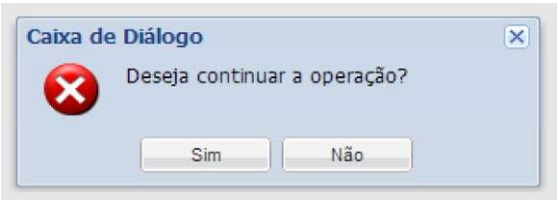


Figura 3. Caixa de diálogo com ícone.
Fonte: Elaborado pelo autor.

5.3. Formulário com validação utilizando componente *formPanel*

O *formPanel* permite criar um formulário em um local específico, o qual pode ser uma DIV, janela, painel, etc.

Segundo Ashoworth (2012), os campos do formulário são especificados nos itens do painel. Definir “defaultType: ‘textField’” nos poupa de ter que especificar em cada item o tipo que possui.

Em cada campo pode ser configurado o atributo “allowBlank: false”, que faz com que o preenchimento do campo seja obrigatório.

Na figura 4, é apresentado um formulário contendo tipos de campos diferentes com preenchimento definido como obrigatório.

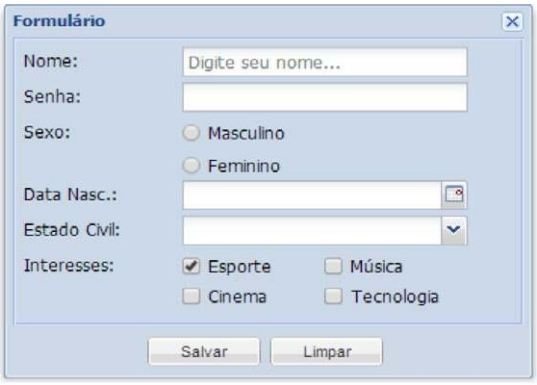


Figura 4. Formulário com validação
Fonte: Elaborado pelo autor

5.4. Janela com *menu* e *submenus*

No Ext JS, *menus* podem conter um ou mais *itens de menu*, e cada *item de menu* criado pode possuir uma nova hierarquia de *submenus*, e assim sucessivamente. Tal como no *Desktop*, *menus* EXT JS podem ser apresentados de várias maneiras.

Para cada *item de menu*, é possível associar o método *handler*, o qual permite definir a ação que será realizada quando o item for acionado com o clique.

A figura 5 apresenta uma *janela* contendo *itens de menu* e *submenu*.



Figura 5. Janela contendo menu e submenu
Fonte: Elaborado pelo autor

5.5. Janela com *GridPanel* com consulta AJAX ao servidor PHP

Segundo Garcia (2011), desde a primeira versão do Ext JS, o *GridPanel* tem sido seu item fundamental. Ele pode exibir dados como uma tabela, mas é muito mais robusto.

O *GridPanel* inclui recursos de gerenciamento de colunas como classificação, redimensionamento, reordenação, mostrar e esconder colunas. Também permite monitorar eventos do mouse, permitindo destacar uma linha ao passar com o mouse e até mesmo selecionar linhas existentes.

Cada *GridPanel* é associado a um item chamado *JsonStore*, o qual faz consultas AJAX a um servidor para buscar as informações que preencherão a *grid*.

A figura 6 apresenta um simples exemplo de uso dos componentes *GridPanel* e *JsonStore*.



Figura 6. Janela contendo grid com consulta à servidor PHP
Fonte: Elaborado pelo autor

5.6. *Layout* completo com janelas contendo *Grid*, *CRUD*, *paginação* e *filtro*

O exemplo apresentado na figura 7 utiliza todos os itens apresentados anteriormente em uma só página.

Uma das grandes vantagens do Ext JS é a facilidade em se encaixar itens diferentes em uma única tela, criando a aparência semelhante a um sistema *desktop*.

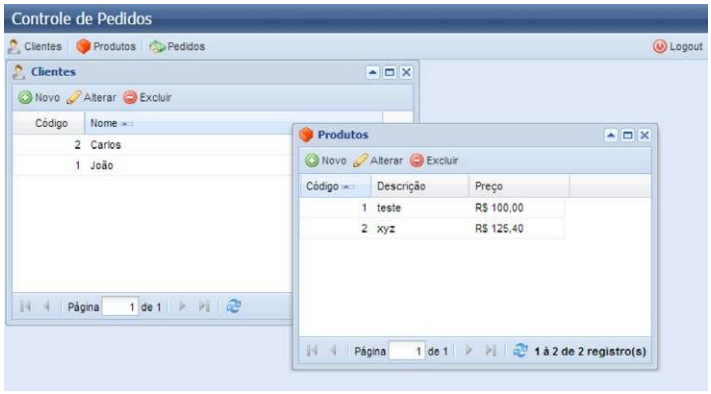


Figura 7. Layout completo com menu e múltiplas janelas
Fonte: Elaborado pelo autor

O código-fonte dos exemplos apresentados nas figuras de 3 a 7 poderão ser obtidos através do endereço http://dev.sencha.com/deploy/ext-4.0.1/examples/portal/portal.html.

CONCLUSÃO

As tecnologias de desenvolvimento para a Web estão em contínua evolução e a criação de *frameworks client-side* tem contribuído e reforçado ainda mais esta tendência, pois permite ao desenvolvedor utilizar componentes prontos e de fácil customização, que facilitam o trabalho de desenvolvimento.

A migração de aplicações anteriormente exclusivas de *desktop* para a Web é uma tendência impulsionada principalmente pela facilidade em executar a aplicação diretamente no navegador, sem que o usuário necessite instalar qualquer programa em seu computador ou mesmo que seu sistema operacional seja compatível com a aplicação.

O framework EXT JS é uma excelente alternativa para o desenvolvimento *client-side* de RIAs, pois, além de reduzir o tempo de desenvolvimento, oferece ferramentas poderosas para a criação de interfaces elegantes, eficientes e funcionais, além ser uma plataforma *cross-browser JavaScript*, o que permite que a mesma aplicação possa ser utilizada em diferentes navegadores e dispositivos.

REFERÊNCIAS

ASHWORTH, S.; DUNCAN, A. **Ext JS 4 Web Application Development Cookbook**.Birmingham: Packt Publishing, 2012.

DEITEL, P.; DEITEL, H. **AJAX, Rich Internet Applications and Web Development for Programmers**. Boston: Pearson, 2008.

EXT JS API DOCUMENTATION. Disponível em: <http://docs.sencha.com/ext-js/>. Acesso em: 11 jun. 2014.

FREDERICK, S. *et al*. **Learning Ext JS 3.2**. Birmingham: Packt Publishing, 2010.

GARCIA, J. **Ext JS in action**. Stanford: Manning, 2011.

GRONER, L. **Ext JS 4 First Look**. Birmingham: Packt Publishing, 2011.

GRONER, L. **Mastering Ext JS**. Birmingham: Packt Publishing, 2013.

RAMON, J. **Ext JS 3.0 Cookbook**. Birmingham: Packt Publishing, 2009.

SENCHA EXT JS. Disponível em: <http://www.sencha.com/products/extjs/>. Acesso em: 11 jun. 2014.

SUNDERARAMAN, P. **Practical Ext JS 4**. New York: Apress, 2013.