# SIMPLEUI: FRAMEWORK PARA INTERAÇÕES NA INTERFACE DO USUÁRIO

*Lucas Ribeiro Maia*

lucas@lucasmaia.com

*Instituto de Educação Continuada*

*Pontifícia Universidade Católica de Minas Gerais (PUC MG) – Belo Horizonte, MG – Brasil*

*Trabalho de Conclusão de Curso – Orientador Sandro Jerônimo*

**Resumo**

Aplicações *web* devem ser intuitivas para que sistemas e usuários se comuniquem de maneira eficiente. O sucesso desta operação depende de uma interface capaz de simplificar o processo de transmissão da informação e da maneira como o usuário compreende os elementos interativos para executar tarefas. Essa inteligibilidade pode ser obtida através de uma área de interação familiar, dotada de recursos que enriquecem a experiência do usuário e que a torne mais dinâmica e recíproca, reagindo aos estímulos recebidos. Executar essas soluções, por outro lado, consome tempo e recursos no projeto de desenvolvimento de uma aplicação *web*. Essa tarefa, muitas vezes, exige exclusividade dos profissionais responsáveis. A proposta deste trabalho é desenvolver um arcabouço de funcionalidades de fácil implementação para aumentar a produtividade e eficiência no desenvolvimento de interfaces para aplicações *web,* não importando o seu cenário de atuação.

**Palavras-chave**: Framework. Javascript. Interface. Interação.

## 1. Introdução

No princípio da internet, no início dos anos 90, as páginas da *web*, também conhecidas pelo equivalente em inglês *webpages*, eram apenas conjuntos de hipertextos conectadas entre si, acessíveis através de um navegador, com recursos visuais limitados e sem interações relevantes com os seus usuários.

Com o passar do tempo, outras ferramentas e tecnologias de desenvolvimento foram criadas para aprimorar o *Hypertext Markup Language* (HTML), como PHP, JAVA, ASP e etc. A partir disso, as páginas, até então estáticas, também puderam armazenar e recuperar informações dinâmicas que seriam geradas por ela e, até mesmo, por outras aplicações.

As interfaces, por sua vez, deveriam ser amigáveis sob o ponto de vista do usuário, sem expor a complexidade dos algoritmos executados pelo servidor da aplicação. Assim, os sistemas buscaram, muitas vezes, metáforas visuais para se tornarem mais intuitivos. — Esse cenário exemplifica a linha de pensamento denominada “Modelo Mental”, introduzido pelo filósofo Kenneth Craik (1943) para justificar a apropriação de conhecimentos prévios de

mecanismos externos para tornar uma representação da realidade, como interfaces gráficas, familiar para os seus utilizadores. Sob a ótica da *web*, em 2002 a empresa *Macromedia* reinterpretou a teoria original e criou o termo "Aplicações de Internet Rica" (da sigla em inglês *RIA - Rich Internet Application*) para designar as aplicações *web* que se apossam de experiências de sistemas tradicionais já conhecidos, como os *desktops*.

Em 2004, a empresa americana *O'Reilly Media* classificou a nova era de aplicações *web* que utilizariam algumas dessas técnicas de interação para estabelecer um novo paradigma, onde os usuários passaram de meros leitores e espectadores para colaboradores e editores do conteúdo, é o início da "*Web* 2.0". Destaca-se, então, o *JavaScript*, uma linguagem criada em 1995, desenvolvida por *Brendan Eich*, responsável pelas implementações dos recursos *frontend* que irão enriquecer a interface do usuário (da sigla em inglês *UI - User Interface*), tornando-se o protagonista dos avanços que as *webpages* precisavam, como métodos de uso que simplificam a experiência e a interação entre os usuários e as aplicações *web*.

Em 2009, Roger S. Pressman e David Lowe, cunharam o termo *webapp* para reunir e representar todos os sistemas e as aplicações baseadas na *web, que* residem na *Internet*, *Intranet* ou *Extranet*, desde as páginas mais simples, como aquelas encontradas no início da internet, até *websites* completos como portais de notícias e comércios eletrônicos.

## 1.1 Problema

A linguagem de marcação universal para a internet, o HTML, mostrou-se não ser suficiente para criar interfaces interativas para as *webapps*. O *JavaScript*, como mencionado, é a linguagem que complementa o HTML oferecendo tais possibilidades. Por outro lado, nenhuma solução estará disponível até que o desenvolvedor crie *scripts*, baseados em *JavaScript*, a fim de simular as funções visuais desejadas. Essa etapa consome muito tempo e recursos no projeto de desenvolvimento de *webapp*, seja durante o próprio desenvolvimento ou nas fases posteriores, uma vez que linguagens do lado cliente pressupõem testes exaustivos em todos as plataformas e versões de sistemas operacionais e navegadores que irão executá-las.

Para encurtar esse tempo e diminuir a probabilidade de erros, algumas equipes adotam outros caminhos, como a adoção de *plug-ins* de *JavaScript* — como são conhecidas as soluções modularizadas e isoladas que resolvem problemas específicos de interface.

No entanto, a adoção de *plug-ins* de terceiros pode deixar a *webapp* suscetível à falhas, pois existe a possibilidade de conflitos de versões ou duplicidade de nomes reservados entre dois ou mais *plug-ins*. Por esse mesmo motivo, também não se estinguem os prejuízos de produtividade, já que optar por alternativas alheias sugere maior dedicação às etapas de testes de regressão a fim de não expor o projeto à conflitos a cada atualização de *plug-in*, uma vez que as equipes estrangeiras de desenvolvimento não garantem compatibilidade entre eles.

Além disso, a maior parte dos *plug-ins* estrangeiros não exclui a escrita de *scripts*. Ainda se faz necessário o desenvolvimento de novas linhas de códigos, seja na própria linguagem *JavaScript* ou pela utilização da notação de objeto JSON, um acrônimo para *JavaScript Object Notation*, para chamar ou customizar as respectivas funcionalidades em que eles, os *plug-ins*, se propõe em resolver.

Como consequência, os códigos de uma aplicação ficam mais "inchados" à medida que novos *scripts* são adicionados. Tudo isso contribui para o aumento do número de bytes e de dados que serão trafegados entre navegador e servidor. Também há o risco de se incluir comandos que não são necessariamente marcações de conteúdo dentro da HTML, fato que pode conflitar com os padrões *web* designados pela *World Wide Web Consortium* (*W3C*), um consórcio internacional, fundado por Tim Berners-Lee, que rege o desenvolvimento *web* e, dentre outras atribuições, garante que as aplicações feitas sob seus preceitos sejam acessadas e visualizadas por qualquer pessoa, tecnologia, dispositivo ou navegador.

Mesmo após a publicação do projeto esse ciclo extenso não se eximirá completamente, uma vez que os mesmos esforços serão requisitados caso haja manutenções dos códigos, tanto para uma simples correção ou para ajustes complexos dos *scripts*. Da mesma forma, a cada novo projeto, soluções genéricas são ignoradas, dando lugar para soluções locais, como a replicação de métodos comuns, que já foram admitidas em outras oportunidades, mas que são reescritas ou descartadas em novos contextos. Além disso, soluções locais desconsideram o uso do sistema CDN (*Content Delivery Network*), já que exclui a chance de uma funcionalidade requisitada em outro projeto estar armazenada em cache. Ou seja, a cada criação ou alteração de um projeto, uma nova rodada de desenvolvimento e de testes será requisitada à equipe, onerando o projeto como um todo. Por fim, todo esse processo contribui para a redução da performance e de produtividade de uma *webapp*, devido a baixa coesão e reaproveitamento dos *scripts*.

## 1.2 Objetivo

O objetivo do trabalho é centralizar e desenvolver um arcabouço de soluções de interface para auxiliar o desenvolvimento de *scripts* para aumentar a produtividade na construção de aplicações *web*, não importando o seu cenário de atuação. Essa proposta pode ser resumida pelo termo "*framework"*, introduzido por Fayad et al (1999b) e Johnson & Foote (1988), que representa um conjunto de classes que constitui um projeto abstrato para solucionar uma família de problemas.

Para atingir esse objetivo, o trabalho foi divido nas seguintes etapas:

1. Levantamento de projetos similares;
1.1. Pesquisar *frameworks* similares;
1.2. Pesquisar *webapps* e principais recursos de interação;
1.3. Identificar funcionalidades que irão compor framework;

2. Desenvolver *framework*;
3. Desenvolver estudo de caso (guia);

### 1.3 Justificativa

SimpleUi reúne uma série de funcionalidades pré-definidas para a interface do usuário que resolvem questões práticas de interação entre utilizador e *webapp*. Ele se baseia e amplia o uso do *Javascript* e da *JQuery*, um *framework* que estende o próprio *JavaScript*. A nova proposta tem como meta agilizar e simplificar o desenvolvimento de interfaces ricas. Seu objetivo principal é universalizar o uso dos recursos oferecidos pelo *JavaScript* sem que seja necessário escrever *scripts* para implementá-los. Dessa forma, a proposta amplia o próprio conceito da *jQuery*: "Escreva menos, faça mais", já que, diferentemente do framework original, não são exigidos conhecimentos de nenhuma outra sintaxe além da marcação universal para a internet, o HTML. Essa mesma simplicidade mantém o código-fonte limpo, facilitando a validação de uma aplicação *web* pela W3C, além de diminuir o tamanho dos arquivos de uma *webapp*.

O *framework* está disponível no modelo de *CDN*, através do link ui.lucasmaia.com/ui/ui.js.php. Dessa forma, as aplicações que utilizam SimpleUi podem compartilhar a possibilidade dele estar armazenado no *cache* do navegador através de outras *webapps* que já o carregaram, otimizando a taxa de bytes trafegados. Para isso, o *framework* está hospedado em um servidor disponível no formato “24/7/365”, ou seja, acessível em qualquer ocasião. Além disso, uma coleção compartilhada por diversos projetos é constantemente submetida e testada pelas aplicações que a utiliza. É provável que ajustes e aprimoramentos sejam lançados sem a necessidade de testá-los da forma como se testaria soluções locais. Nesse caso, o conceito de “teste” e “testadores” pode ser ampliado entre equipes e aplicações que não se conhecem, mas que tem anseios comuns para a Interface do Usuário. Por fim, as aplicações poupam tempo de desenvolvimento e de projeto, uma vez que o uso da *CDN* proporcionará sempre a versão mais atual do *framework* sem a necessidade de novos testes de regressão por essas *webapps*.

As próximas seções estão organizadas das seguintes maneiras: Referencial Teórico; Framework; Guia de Uso; Comentários e Conclusões.

## 2. Referencial Teórico

Nessa seção são abordados os principais conceitos para a compreensão de um *framework* de interação de interface do usuário.

### 2.2 “RIA” – *Rich Internet Application*

Introduzido em 2002 pela empresa americana *Macromedia*, RIA é um acrônimo em inglês de Aplicações de Internet Rica (*Rich Internet Application*). As RIA são aplicações *web*

que se apropriam de símbolos, mecanismos e funcionalidades de interface de sistemas notáveis, como os efeitos "arrastar e soltar" e as janelas dos *desktops*.

O objetivo das RIA é aproveitar o conhecimento prévio e a experiência de outras plataformas, diminuindo o tempo de aprendizagem, além de torná-lo familiar para o usuário, obedecendo o preceito de usabilidade que, segundo Loranger Nielsen (2009), representa um atributo de qualidade relacionado à facilidade do uso de algo. Mais especificamente, refere-se à rapidez com que o usuários podem aprender a usar alguma coisa (curva de aprendizado), a eficiência deles ao usá-la, o quanto lembram daquilo, seu grau de propensão a erros e o quanto gostam de utilizá-la. Se as pessoas não puderem ou não utilizarem um recurso, ele pode muito bem não existir.

Em uma Aplicação de Internet Rica, cabe ao *JavaScript* implementar os recursos de interação e ao navegador interpretá-los. As RIA, portanto, endossam a divisão das tarefas de processamento de uma aplicação *web*: enquanto o servidor se responsabiliza pelo armazenamento e pela recuperação dos dados de uma aplicação, o navegador e a linguagem *frontend* se especializam nas interações e manipulações da interface das *webapps*.

## 2.3 Engenharia de *Software*

Engenharia de software é a criação e a utilização de sólidos princípios de engenharia, a fim de obter softwares econômicos que sejam confiáveis e que trabalhem de forma eficientemente em máquinas reais. (PRESSMAN, 2006).

A engenharia de software é uma tecnologia em camadas. Qualquer abordagem de engenharia (inclusive a engenharia de software) deve-se, apoiar num compromisso organizacional com a qualidade, gestão de qualidade total e Seis Sigmas(Six Sigmas)[1] (PRESSMAN, 2006).


**Figura 1** – Engenharia de *Softwares* em camadas

Todo *software* pode, então, ser modificado e aperfeiçoado de acordo com as inovações tecnológicas, mudanças no ambiente ou pelas demandas dos utilizadores. A exemplo disso, destaca-se o *plug-in*, um complemento ou uma extensão para sistemas, ferramentas ou programas modulares. Após instalado, o *plug-in* executará funções especiais que não são

[1] **Seis Sigmas:** é um conjunto de práticas para melhorar sistematicamente os processos ao eliminar defeitos. Um defeito é definido como a não conformidade de um produto ou serviço com suas especificações. Seis Sigma também é definido como uma estratégia gerencial para promover mudanças nas organizações, fazendo com que se chegue a melhorias nos processos, produtos e serviços para a satisfação dos clientes.

encontradas no *software* original. Ou seja, trata-se de um meio para aperfeiçoar ou personalizar um programa. No desenvolvimento de interface para a *web*, *plug-ins* são *scripts* pré-concebidos que resolvem problemas específicos de interação com o usuário e, por isso, visam aumentar a produtividade na construção de uma aplicação por meio de soluções localizadas.

Assim como os *plug-ins*, as bibliotecas são extensões para sistemas e aplicações *web*. No entanto, uma biblioteca reúne várias funcionalidades, ou seja, são conjuntos de métodos e funções focadas em resolver problemas de interação e de interface para aplicações *web*.

Já o *framework*, segundo definição de Fayad et al (1999b) e Johnson & Foote (1988), é um conjunto de classes que constitui um projeto abstrato para solucionar uma família de problemas. Por esse motivo, sua proposta é oferecer uma série de recursos para facilitar a implementação dos problemas que os utilizadores se propõe a resolver.

## 2.4 Trabalhos Relacionados

Já existem bibliotecas baseadas em *JavaScript* que têm como propósito solucionar problemas genéricos de interface, como *jQueryUi*, *Scriptaculous*, *EasyUi* e etc. No entanto, essas bibliotecas resolvem apenas parte do problema, uma vez que simplificam a escrita dos *scripts*, mas sendo ainda necessário escrevê-los, seja dentro da própria HTML ou em arquivos *JavaScripts* importados pela HTML. Esse mesmo empecilho se estende nas bibliotecas que utilizam a notação de objeto JSON, uma outra sintaxe exigida para customização dos *scripts* oferecidos pelas bibliotecas.

Essas características de implementação ainda representam as mesmas barreiras e prejuízos já mencionados. Um *framework* mais coeso e intuitivo pode encurtar a lacuna entre as etapas de criação e de desenvolvimento, ponto de omissão de outras bibliotecas. Como dito anteriormente, o que diferencia SimpleUi dos demais *framework* e bibliotecas é o modo como os seus métodos são implementados: os *triggers* (termo em inglês para "gatilho" — como são conhecidos as chamadas de um método) são as próprias marcações HTML, parte da estrutura de elementos que compõe o HTML, ou DOM – acrônimo em inglês para ***D**ocument **O**bject **M**odel*. Dessa forma, não são necessárias experiências com *JavaScript* ou JSON, adotados pela maioria das bibliotecas e *frameworks* já citados.

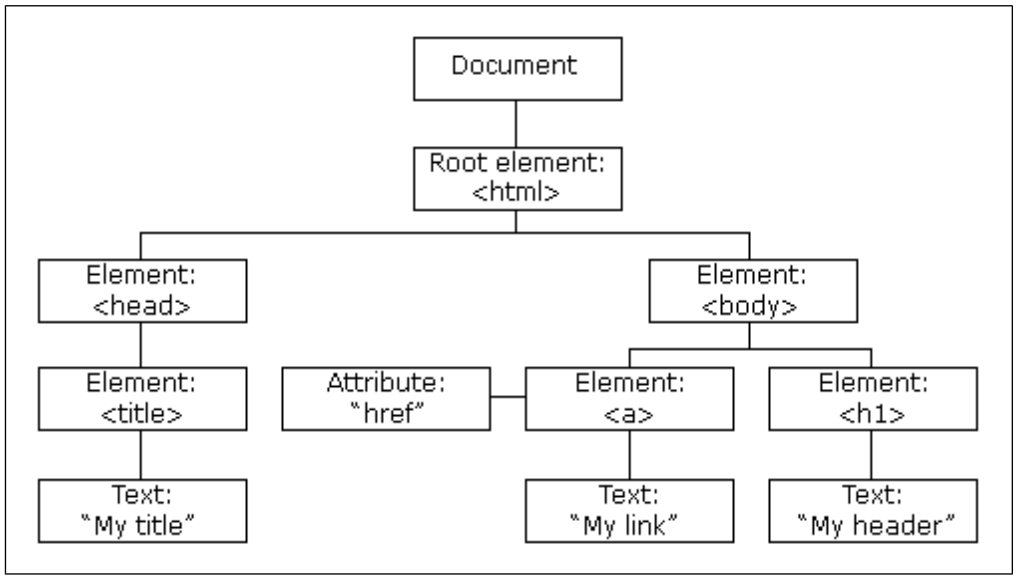


**Figura 2** – Árvore DOM – Objetos tangíveis pelo *JavaScript*
**Fonte** – http://www.w3schools.com/js/js_htmldom.asp

A universalização do uso de SimpleUi também se aplica para *layouts* e *wireframes* — como são conhecidos os protótipos estáticos criados durante a fase de planejamento de uma aplicação *web*. Destaca-se, então, a possibilidade da implementação pelas equipes que antecedem a fase de construção de uma aplicação *web*, geralmente sem experiência em programação *web*, como arquitetos da informação e *designers*, público que, muitas vezes, não está entre os destinatários das bibliotecas e *frameworks* de interface do usuário.

A proposta de enriquecimento dos protótipos estáticos foi concebida em 2008 pela empresa americana *37signals*, onde o objetico é minimizar o tempo gasto entre as fases de criação e de desenvolvimento, geralmente inflacionado pela abstração e pelos ruídos de comunicação causados pelos *layouts* estáticos. Como consequência, os protótipos interativos também facilitam o entendimento entre os *stakeholders* (termo em inglês para designar a equipe que participa de um projeto de *webapp*). Isso se aplica, inclusive, para aqueles envolvidos que não têm conhecimentos de desenvolvimento *web*, como os clientes que aprovam as aplicações a partir de *layouts* ou *wireframes*, poupando tempo e recursos para a viabilizar projetos.

## 3. Framework

A escolha das funcionalidades que compõe o *framework* SimpleUi iniciou-se a partir de uma pesquisa realizada em vinte e um de junho de 2013 pela *Alexa Analytics*, um serviço gerenciado pela *Amazon*, que analisa e elabora rankings dos sites mais acessados. Na oportunidade, a *Alexa Analytics* informou os dez sites mais visitados no Brasil, são eles (em ordem):

1. *Facebook* (http://www.facebook.com)
2. *Google* Brasil (http://www.google.com.br)

3. *YouTube* (http://www.youtube.com)
4. *Google* (http://www.google.com)
5. Universo *Online* (http://www.uol.com.br)
6. *Windows Live* (http://www.live.com)
7. Globo (http://www.globo.com)
8. *Amazon* (http://www.amazon.com)
9. *Yahoo* (http://www.yahoo.com)
10. Mercado Livre (http://www.mercadolivre.com.br)

Foram analisados os dez principais recursos de interface do usuário nos dez *sites* mais acessados no Brasil. Todos os *sites* mostraram recursos de máscara e validação dos campos dos seu formulários, como autopreenchimento de parênteses nos campos para digitar telefones, confirmação de e-mails ou validação de campos em branco. Observou-se que seis deles usaram o recurso de *accordion* para organizar alguma informação da aplicação. Da mesma forma, cinco dos dez *sites* mais acessados no Brasil utilizam abas para ordenar o seu conteúdo. Três desses *sites* usaram caixas de diálogos do tipo *modal* para se comunicar com o usuário. Seis *sites* apresentaram galeria de imagens do tipo *banner* rotativos, seja para propagandas ou destaques na página inicial. Cinco *sites* tinham controles do tamanho da fonte. Quatro apresentavam galerias (ou *slideshow*), onde o usuário tem autonomia sobre a exibição das imagens. Quatro *sites*, dentre os dez mais acessados, tinham algum tipo de *tooltip*. Todos os *sites* apresentavam *thumbnails* e paginação que, apesar de ser uma solução do lado do servidor, poderia contar com uma alternativa do lado cliente para evitar requisições desnecessárias a cada página solicitada pelo usuário.

Portanto, compõe SimpleUi aquelas funcionalidades genéricas que ficaram em evidência nos dez sites mais acessados no Brasil. A escolha se baseia nos principais recursos encontrados em aplicações notáveis e, por isso, potencialmente úteis para qualquer aplicação. A seguir são listados:

- **Abas**: A aba (do inglês *tab*) tem como objetivo tratar, organizar e classificar os conteúdos de uma *webapp*, sem que seja necessário requisitá-los novamente para o servidor da aplicação a cada solicitação, uma vez que a totalidade dos dados já se encontra no navegador de internet;


**Figura 3** – Exemplo de navegação em abas

- **Modais:** São alternativas elegantes e compatíveis com os padrões web às caixas de diálogos invasivas, como os *pop-ups* — como são conhecidas as janelas extras que, muitas vezes, surgem sem que sejam solicitadas pelos usuários. Além disso, diferente de outros modelos limitados, como a caixa de alerta do *JavaScript*, o modal permite exibir qualquer conteúdo ou ferramentas interativas;

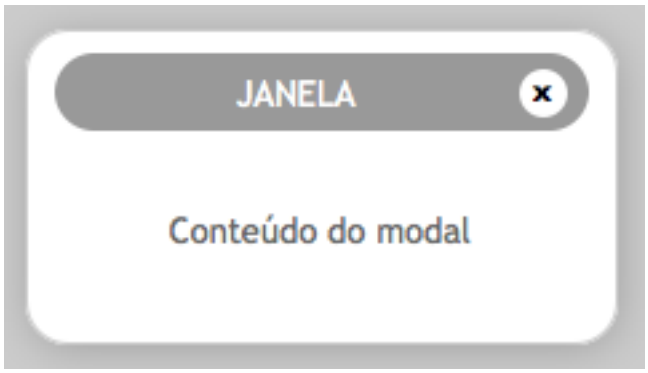


**Figura 4** – Exemplo de modal

- **Accordion:** Um *accordion* (termo inglês para sanfona) servem para aproveitar o espaço útil da tela onde a aplicação será exibida. Como o nome sugere, o *accordion* omite parte da informação e a exibe somente quando o usuário requisitá-la através de um evento, como o clique. Da mesma forma que as abas, o *accordion* também reduz o número de requisições ao servidor, já que todo o conteúdo é carregado uma única vez pela aplicação;

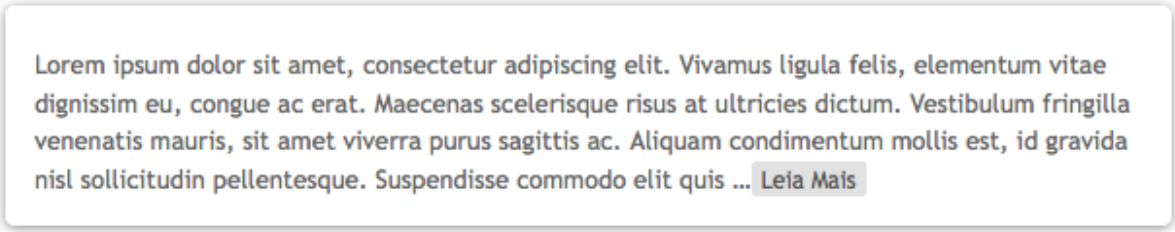


**Figura 5** – Exemplo de *accordion*

- ***Tooltip*:** Junção das palavras inglesas *tool* (ferramenta) e *tip* (dica), o *tooltip* é um recurso de interface para estender o significado de um elemento HTML quando o usuário assim requisitar, ou seja, quando o cursor do mouse apontar para o respectivo elemento. Diferente da marcação HTML, o *title*, o *tooltip* permite a exibição ilimitada de conteúdo, incluindo recursos interativos, como *links* ou imagens;

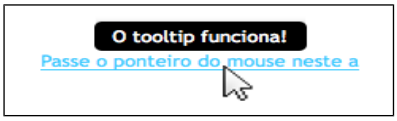


**Figura 6** – Exemplo de tooltip

- ***Thumbnail****:* O termo em inglês *thumbnail* designa as imagens em miniaturas usadas nas *webapps*. O tratamento dessas miniaturas proposto por SimpleUi, organiza as miniaturas em função de sua moldura, ou seja, diferente dos atributos *width* e *height* da HTML, ele é responsável por realinhar a imagem de acordo com as dimensões e proporções entre ela e o seu enquadramento;

**Figura 7** – Exemplos de *thumbnails*

- **Máscaras para campos e validações de formulários**: As máscaras e as validações de formulários servem para auxiliar e notificar erros de preenchimento, evitando requisições desnecessárias para o servidor;

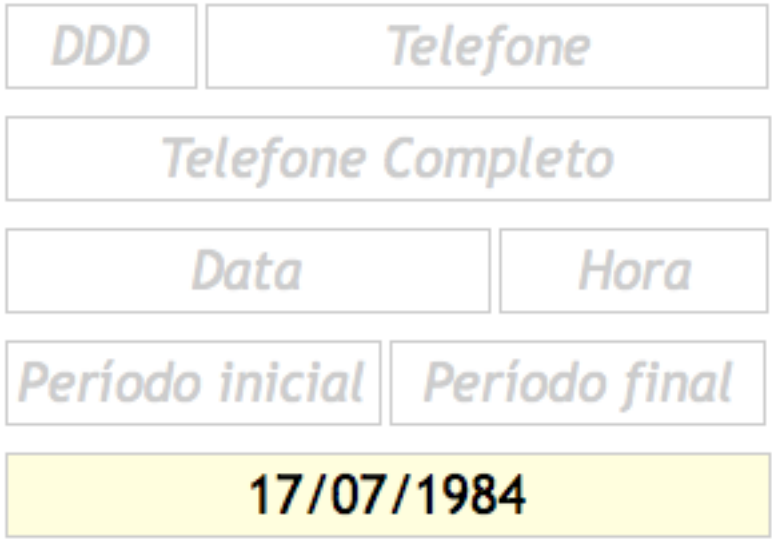


**Figura 8** – Exemplo de máscara em um formulário

- **Controle de fonte**: ferramenta de acessibilidade para aumentar ou diminuir o tamanho da letra;

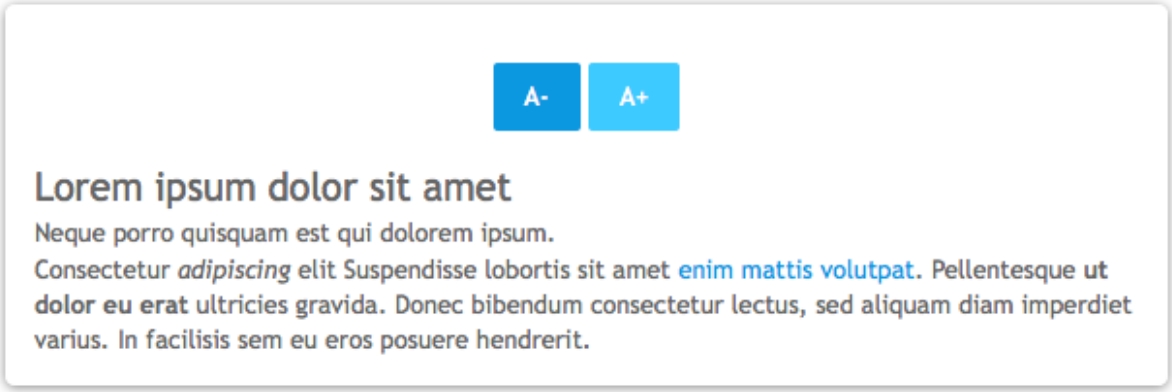


**Figura 9** – Controle de fonte

- **Paginação *frontend***: recurso para organizar informações de uma página em blocos de páginas;

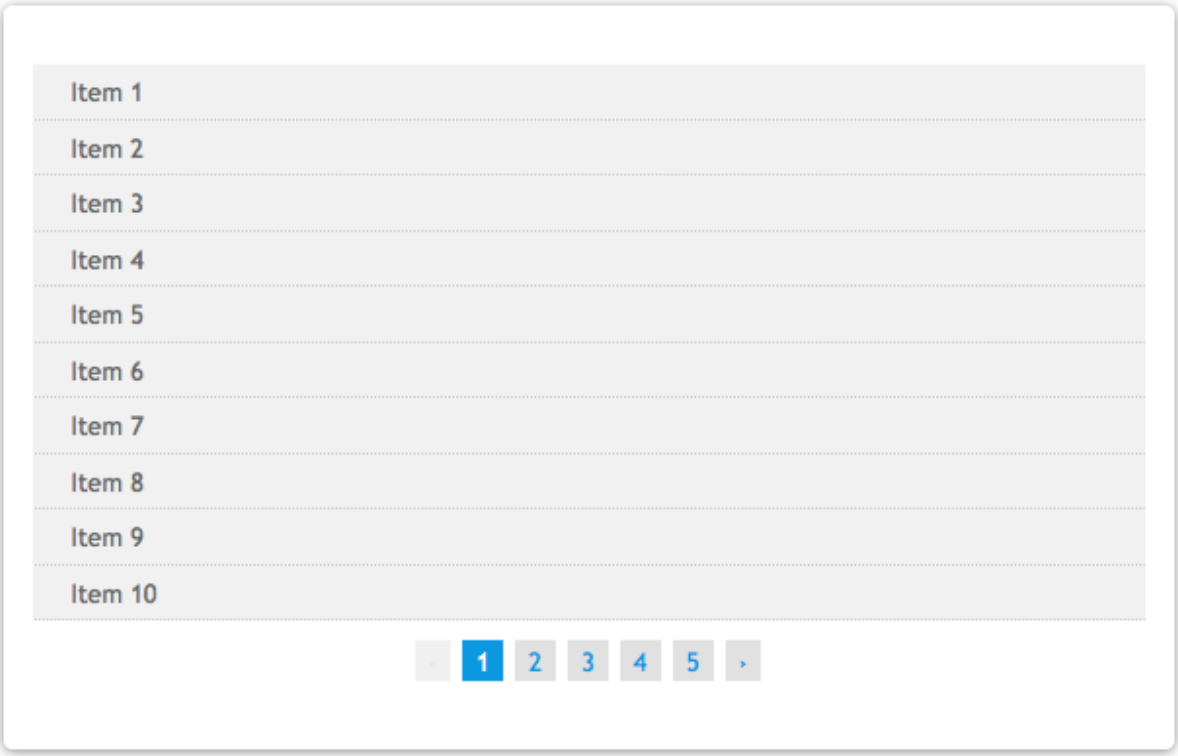


**Figura 10** – Paginação

- **Galeria de imagens**: ferramenta para navegação e transição entre imagens de um álbum;

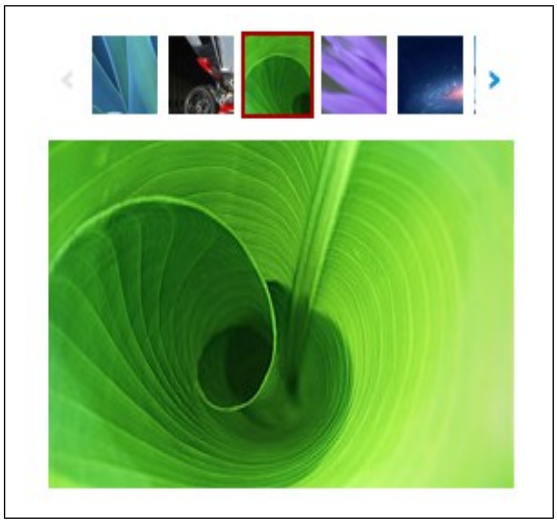

**Figura 11** – Galeria de imagem

- **Apresentação de imagens e infográficos**: Interface de navegação para imagens grandes ou infográficos. Possibilita zoom e seleção de trechos de uma imagem para visualização detalhada;


**Figura 12** – Navegação em um infográfico

Além das funcionalidades citadas, SimpleUi possui um *Core* – termo em inglês para designar a base e o centro de operações de um *framework* – do qual destacam-se os métodos globais que servem de suporte para as demais funções, como as ações para detectar o tamanho da tela do usuário, tratar o conteúdo (acentuação, espaços, numéricos, alpha-numéricos e etc),

recuperar *URL* e seus parâmetros, salvar e recuperar dados locais (cookies), detectar navegador, detectar dispositivos móveis, detectar flash player, imprimir documento personalizado, etc.

Como dito anteriormente, os métodos de SimpleUi serão implementados apenas através da identificação dos elementos que sofrerão os efeitos, ou seja, apenas por meio das *Tags* de HTML – como são conhecidas as marcações da HTML – ou através de seletores de CSS – sigla em inglês para *Cascading Style Sheets* – presentes nos atributos *id* e *class* da HTML.

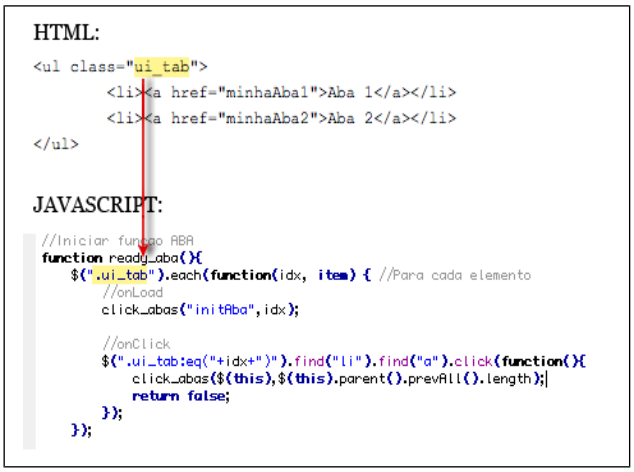


**Figura 13** – Gatilho HTML*:* class acionando uma função *Javascript*

Todas as funcionalidades são compostas por valores padrões. No entanto, se for do interesse do desenvolvedor, ele pode customizar um método também através de atributos da HTML. Nesse caso, cada funcionalidade pode possuir atributos proprietários da HTML5, o *data*-*, para editar itens como como cores, taxa de velocidade de determinado evento, número de itens selecionados por um método, etc.

## 4. Guia de Uso

Nesta seção, será apresentado a ferramenta implementada usando o simpleUi e que serviu como suporte de estudo do *framework*.

Este guia está disponível em ui.lucasmaia.com. Ele foi concebido em plataforma cruzada – do inglês: *crossbrowser*, ou seja, ele pode ser executado tendo o mesmo desempenho em diversos navegadores, como Internet Explorer, Safari, Chrome, Opera e Firefox; dispositivos, como *PC*, *notebook*, *tablet* ou celular e sistemas operacionais, como Windows, MacOS, Linux, Android ou iOS. O guia foi submetido e avaliado com êxito pela W3C como um documento HTML5 válido, seguindo todos as recomendações e padrões de desenvolvimento *web*.

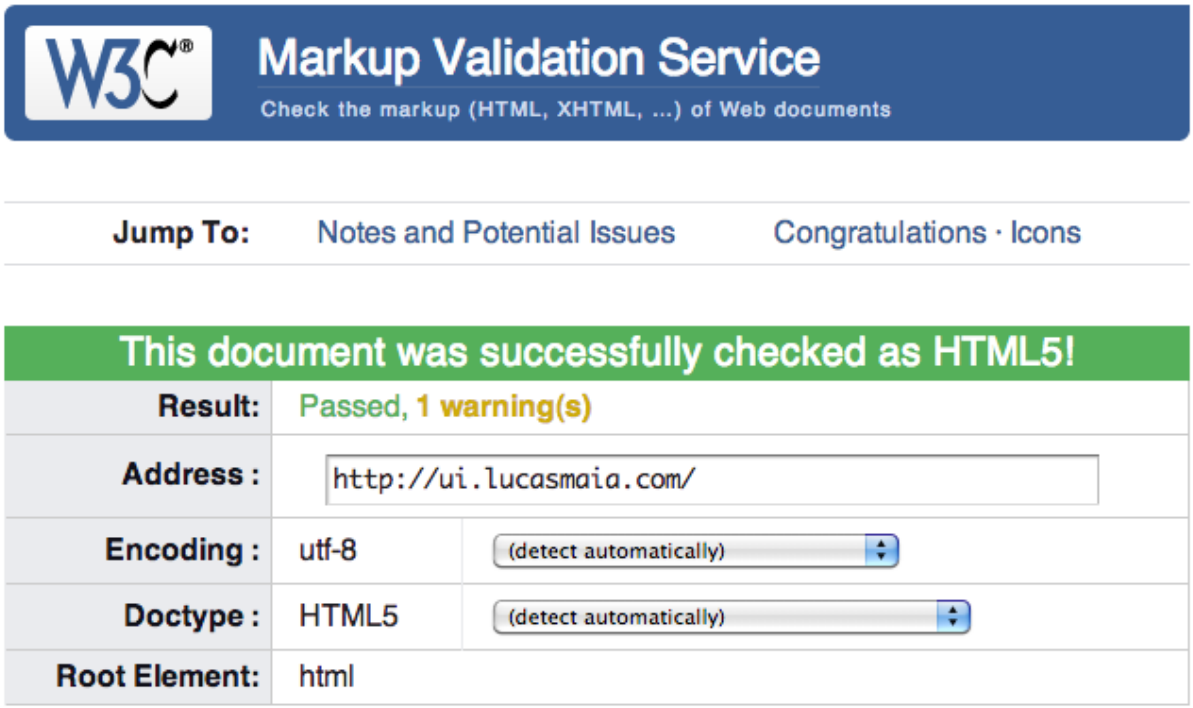


**Figura 14** – Marcação HTML5 válida
**Fonte:** http://validator.w3.org/check?uri=http%3A%2F%2Fui.lucasmaia.com%2F

Os padrões de *web* semântica também foram estudados e obedecidos durante a construção das páginas que compõe o guia de uso. Os conteúdos tornam-se, então, inteligíveis para homens e máquinas, podendo ser interpretados, de forma otimizada, por buscadores, compartilhadores de conteúdo, leitores para deficientes visuais e etc. Na prática, cada elemento de seção cria uma entrada ordenada no esboço do documento HTML (do inglês HTML *outliner*), como um sumário lógico de navegação.

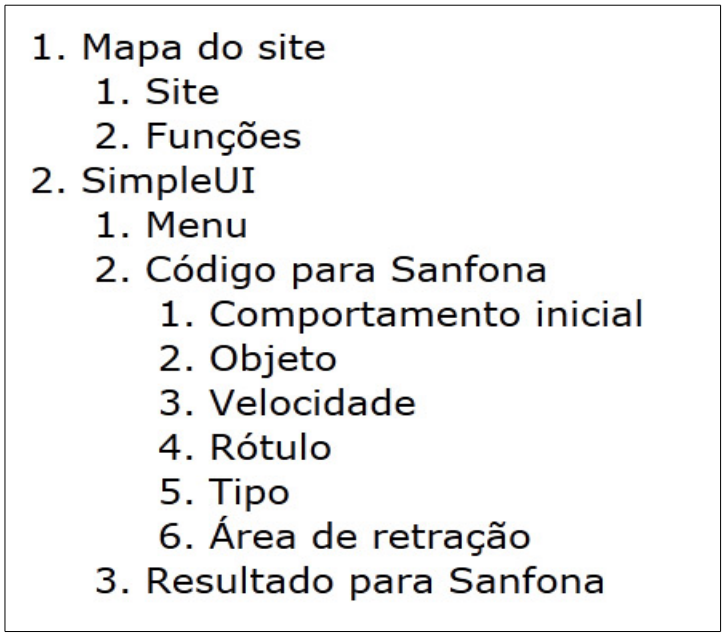


**Figura 15** – Sumário do documento HTML para a função *accordion* (sanfona)

**Fonte:** http://gsnedders.html5.org/outliner/

Os usuários podem testar e editar qualquer um dos recursos do *framework* através do endereço ui.lucasmaia.com/functions.php. Cada exemplo de função é um arquivo incluído no lado do servidor da aplicação e exibido em dois momentos distintos: como texto do código da função requisitada pelo usuário e como resultado renderizado pelo navegador.

A partir do exemplo inicial, é possível customizar o código-fonte original por meio de editores WYSIWYG – acrônimo da expressão em inglês "***W**hat **Y**ou **S**ee **I**s **W**hat **Y**ou **G**et*", cuja tradução é "O que você vê é o que você obtém". Isso significa que os usuários podem explorar o potencial das funcionalidades em tempo de execução. por meio do método de programação conhecido como AJAX – acrônimo em inglês de ***A**synchronous **J**avascript **a**nd **X**ML*, onde o tráfego de dados entre o servidor e a aplicação ocorre de maneira assíncrona, ou seja, as requisições são feitas por blocos de informação, melhorando a experiência do usuário, já que visa agilizar a interação com ferramenta. Por fim, após selecionar e customizar uma função, o usuário pode capturar o código gerado como produto final, sem precisar passar por outros editores de códigos.

**Figura 16** – Exemplo do guia de uso: inclusão, customização e saída da função *accordion* como código e como resultado renderizado pelo navegador.

O *download* de SimpleUi pode ser feito em ui.lucasmaia.com/download.php e também é personalizado, ou seja, o usuário pode optar pelo pacote completo ou apenas pelos métodos do seu interesse. Dessa forma, o tamanho em *bytes* ajusta-se à necessidade do projeto, otimizando o desempenho e o tráfego entre o servidor e a aplicação que a utiliza. Além do download, o guia oferece o *framework* por meio da *CDN*, seja o pacote completo ou personalizado.

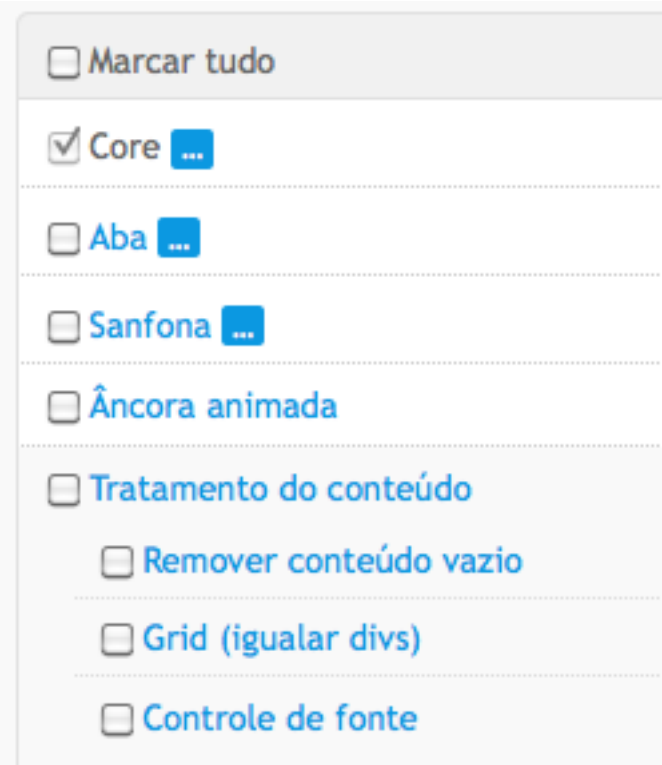


**Figura 17** – Exemplo do *download* personalizado

```
<?php if(in_array("all", $functions) || in_array("equal", $functions)){ ?>
/* ---------------------- */
/* --- Equalizar divs --- */
/* ---------------------- */

function equalizarGrid() {
    if($(".ui_gridDiv").length>0){ //Teste necessario para ready via ajax
        //Estilos iniciais
        $(".ui_gridDiv:visible").css("float","left");

        ..
```

**Figura 18** – Trecho de código do *framework*.
Em destaque uma condição PHP para particionar o download do pacote

O guia oferece um manual completo para cada uma das funções que compõe o *framework*. Ele pode ser acessado através do endereço: ui.lucasmaia.com/manual.php. Os

usuários e visitantes também podem entrar em contato através do endereço ui.lucasmaia.com/contact.php, seja para registrar uma falha, colaborar ou comentar sobre o projeto.

## 5. Comentários e Conclusões

O trabalho gerou dois produtos: o *framework* SimpleUi, que implementa interações na interface do usuário, e um guia de uso, que exemplifica e contextualiza os seus métodos. A versão de lançamento de SimpleUi possui algumas limitações, das quais poderão ser trabalhadas em abordagens futuras. A seguir, serão comentadas as principais limitações do *framework.*

### 5.1 Limitações do trabalho

A proposta de um *framework* que implementa *JavaScript* sem *scripts* expõe a ausência previsível e calculada dos seletores dinâmicos, tangíveis após a renderização da página, como são os elementos de *JavaScript*. Por esse motivo, algumas funcionalidades do *framework* tratam apenas alguns seletores notáveis, como os elementos "próximos" (*next*) e "anteriores" (*prev*) a um objeto. No entanto, outras relações mais complexas, a exemplo dos filhos, pais ou netos de um objeto, só serão obtidas através de marcadores estáticos, como classes e identificadores de CSS. Apesar disso, o *framework* é extensível, ou seja, pode ser atualizado de forma a ampliar o leque dos seletores dinâmicos mais requisitados, seguindo a mesma solução criada para os elementos "próximos" (*next*) e "anteriores" (*prev*).

O *framework* e o guia foram implementados em português e não têm suporte para outras línguas, mas, independente da limitação, SimpleUi foi desenvolvido em código aberto. Isso significa que usuários avançados podem alterar e implementar novas entradas, conforme a necessidade do seu projeto, seja para mudar um termo ou um comportamento de uma ação.

### 5.2 Trabalhos futuros

Tratando-se de uma versão inicial, o *framework* é modular, ou seja, capaz de evoluir de acordo com a demanda e está preparado para receber novos recursos ou, ainda, ampliar e melhorar as funcionalidades mais requisitadas, evidenciando novamente a extensibilidade como característica notável de SimpleUi. Essas atualizações serão automáticas para aquelas aplicações que incorporarem o *framework* por meio da *CDN*. Nesse caso, nenhum movimento será necessário das *webapps* que utilizarem SimpleUi por este método.

A ferramenta tem potencial para ser colaborativa, ou seja, mantida por pessoas ou comunidades de desenvolvedores. Na prática, o sistema deverá contar com uma base de dados de usuários e um *Bug-tracker* – termo em inglês para rastreamento de defeitos – para que qualquer interessado possa sugerir, indicar e acompanhar as correções de falhas, atualizações e novidades do *framework*.

Além do guia de uso, SimpleUi também poderá ser exemplificado em ambientes que simulam uma aplicação real. Para isso, existe a possibilidade de criar sistemas que implementam e contextualizam o uso do *framework*, como lojas virtuais ou portais de notícias.

**Referências**


Alexa. *The top 500 sites in Brazil*. Disponível em <http://www.alexa.com/topsites/countries/BR>. Acessado em 21 de junho de 2013.

Fried, Jason. *Why we skip Photoshop*. Disponível em <http://37signals.com/svn/posts/1061-why-we-skip-photoshop>. Acessado em 5 de agosto de 2013.

NIELSEN, Jakob e LORANGER, Hoa. *Usabilidade na web: projetando websites com qualidade*. Rio de Janeiro. Elsevier/Campus, 2007.

O'Reilly, Tim. *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*. Disponível em: <http://oreilly.com/web2/archive/what-is-web-20.html>. Acessado em 19 de junho de 2013.

PRESSMAN, Roger S.; LOWE, David. *Engenharia Web*. LTC, 2009.

SCHMIDT, D. C. e BUSCHMANN, F. *Patterns, Frameworks, and middleware: their synergistic relationships*. ICSE, 2003

W3C. *The HTML DOM (Document Object Model)*. Disponível em <http://www.w3schools.com/js/js_htmldom.asp>. Acessado em 7 de setembro de 2013.