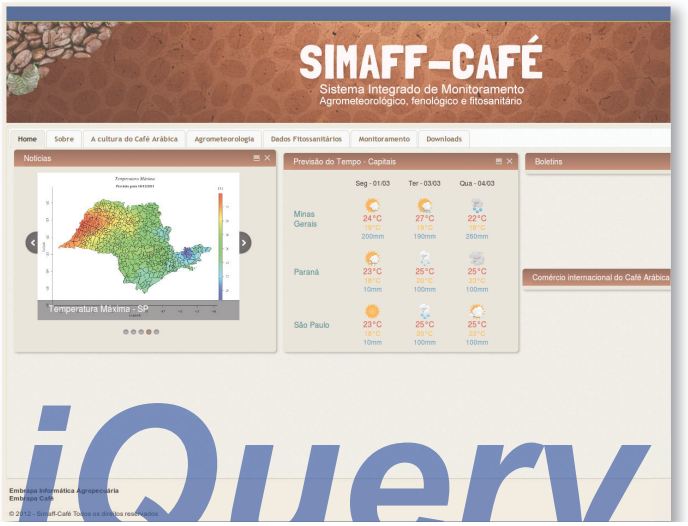
Comunicado Técnico 113

ISSN 1677-8464
Dezembro, 2012
Campinas, SP



Explorando recursos do *jQuery* para tornar interfaces web adaptáveis

Luciana Alvim Santos Romani[1]
Juliana Pereira de Santana[2]
João Paulo Silva[3]

Com a evolução tecnológica e o desenvolvimento de diferentes dispositivos para acesso web como *tablets* e *smartphones* que possuem tamanhos de telas menores do que um computador padrão, as interfaces de usuário devem ser projetadas de forma a possibilitar seu uso adequado em diferentes navegadores e aplicativos, além de serem adaptadas às necessidades e preferências do usuário. Um sistema adaptável permite ao usuário a adaptação da interface, possibilitando não apenas mudar o visual, *design* da interface com alteração de cores e estilos do sistema, mas também alteração do conteúdo visualizado e o modo em que será apresentado, podendo alterar posicionamentos, e visibilidades (NERIS; BARANAUSKAS, 2011). Essas interfaces tem o propósito de facilitar o trabalho do usuário, melhorando a eficiência e qualidade do sistema, possibilitando uma interação mais agradável e com maior usabilidade (SILVA; SILVA, 2007).

A biblioteca *JavaScript jQuery* possui diversos componentes e *plugins* para facilitar a criação de aplicações web e torná-las interativas. Com ela é possível desenhar uma interface com elementos interativos e compatíveis com os diversos tipos de dispositivos móveis e navegadores padrões. Além disso, é possível projetar a interface de modo a torná-la flexível e adaptável tornando-se adequada às necessidades dos diferentes tipos de usuários.

Este trabalho apresenta componentes da biblioteca *jQuery* que possam ser usados para tornar a interface web mais flexível para os usuários. Neste documento são apresentados métodos em que a biblioteca *jQuery* pode ser utilizada para criar interfaces interativas e adaptáveis. A construção de uma interface web com os componentes enumerados neste trabalho foi feita no escopo do projeto SIMAFF-Café desenvolvido em parceria por instituições que integram o consórcio café. O objetivo do projeto é o monitoramento agrometeorológico, fenológico e fitossanitário da cultura do café em São Paulo, Minas Gerais e Paraná.

O restante deste documento está organizado da seguinte forma: a próxima seção apresenta a biblioteca *jQuery*, com seus eventos e funções e como utilizá-los, na página 5 são apresentados os componentes e *plugins* do *jQuery* que podem ser usados para tornar a interface interativa, os resultados e discussões do uso dos componentes do *jQuery* no sistema SIMAFF-Café

[1] Doutora em Ciência da Computação, Pesquisadora da Embrapa Informática Agropecuária, Campinas, SP, luciana.romani@embrapa.br
[2] Bolsista da Embrapa Informática Agropecuária, julips.hp@gmail.com
[3] Estagiário da Embrapa Informática Agropecuária, jp_e@hotmail.com

estão na página 7, e finalmente são apresentadas as conclusões na página 10.

Sobre o *jQuery*

Atualmente existe uma tendência de migração de aplicações para a plataforma web, em que o navegador é o meio de interação com o usuário (BOZZON, 2006). Para que essa migração seja bem aceita pelos usuários, é imprescindível que existam ferramentas atraentes visualmente e simples de serem manipuladas pelos desenvolvedores e usuários finais. O usuário espera uma ferramenta web que alie um visual agradável com usabilidade, ao passo que o desenvolvedor deseja que esta tenha uma sintaxe simples, porém com a maior gama de funcionalidades e compatibilidade disponíveis, para que os projetos desenvolvidos com essa ferramenta sejam ricos e interativos. Outro fator a ser considerado é o custo das ferramentas de desenvolvimento a serem utilizadas no projeto do sistema, visto que isso afeta diretamente no orçamento final do projeto, e, nesse caso, as ferramentas de código aberto levam vantagem, pois estas não têm custo de obtenção, podendo ser utilizadas gratuitamente. Além disso, é importante utilizar bibliotecas que tenham uma comunidade ativa e que é mantida para garantir que novas versões compatíveis com novas versões de navegadores serão desenvolvidas.

O *jQuery* é uma dessas bibliotecas *JavaScript*, com funções, componentes, *plugins*, manipulação de eventos, *Cascade Style Sheet* (CSS) e elementos do *Document Object Model* (DOM) e interação com *Asynchronous Javascript and XML* (AJAX).

Apesar de não ser a pioneira, a *jQuery*, lançada por John Resig, em 2006, é uma das bibliotecas *javascript* mais utilizada atualmente. Uma de suas funcionalidades mais notáveis é a simplicidade na criação de efeitos visuais, o que aumenta o dinamismo na utilização de sites onde o *jQuery* é utilizado.

Em relação ao *JavaScript*, a *jQuery* herda muitas das funcionalidades, como declaração de variáveis, funções e classes, operadores e outros. A diferença, portanto, está na sintaxe utilizada na implementação de algoritmos em *jQuery*, que é mais simples, facilitando o entendimento e consequente uso. Outra vantagem é o custo de armazenamento que é baixo, pois a biblioteca ocupa apenas 15 kB do disco rígido, valor irrelevante para a capacidade dos equipamentos atuais.

O *jQuery* é compatível com os navegadores mais utilizados atualmente, como Google Chrome, Firefox, Internet Explorer, Safari e Opera. Essa biblioteca é *OpenSource* e trabalha sobre a licença *General Public License* (GPL), permitindo assim que o *jQuery* seja utilizado em sites comerciais. Todas estas facilidades, tanto de utilização quando de acesso, tornam cada vez mais interessante considerar o uso da biblioteca *jQuery*.

Para começar a usar a biblioteca *jQuery* é necessário criar um *link* da aplicação com o arquivo .js da biblioteca (Exemplo 1), que está disponível para *download* na página oficial do *jQuery*: http://jquery.com.

Exemplo 1

```
<html>
  <head>
     <script src="jquery.js"> </ script>
    <title> Criando uma Aplicação com jQuery </
title>
  </ Head>
  <body>
  </ Body>
</ Html>
```

Os seletores são utilizados para buscar elementos do DOM, para facilitar a manipulação do documento. O *jQuery* permite que sejam selecionados elementos a partir do seu id, classe, por um atributo específico ou pelo tipo do elemento.

- Selecionar por id: `$("#minhaDiv")`
- Selecionar pela classe: `$(".minhaClasse")`
- Selecionar por um atributo: `$('inpu[name="Nome"]')`
- Selecionar pelo tipo: `$("input:text")`
  `$(":button")`

Com os manipuladores de eventos pode-se controlar as ações do usuário. A Tabela 1 apresenta uma lista de exemplos de eventos da biblioteca *jQuery*.

Com o *jQuery* é possível manipular o CSS, obtendo, aplicando e modificando propriedades do CSS. A Tabela 2 apresenta uma lista de exemplos de tipos de manipulação do CSS da biblioteca jQuery.

Manipulação do DOM. A Tabela 3 apresenta uma lista com exemplos de tipos de manipulação do DOM da biblioteca *jQuery*.

A biblioteca *jQuery* possui uma séria de efeitos para serem adicionadas à aplicação. A Tabela 4 apresenta uma lista de exemplos de efeitos da biblioteca *jQuery*.

A biblioteca *jQuery* possui um conjunto completo de recursos AJAX que permitem recuperar e enviar dados

Tabela 1. Lista de eventos da biblioteca *jQuery*.

Evento	Descrição	Exemplo
.ready()	Executar uma função quando o DOM estiver completamente carregado.	`$(document).ready(function(){ alert("Página carregada"); });`
.click()	Executar uma função quando um elemento for clicado.	`$("#meuId").click(function(){ alert("Evento Click"); });`
.submit()	Enviar um formulário. Só pode ser ligados a elementos `<form>`.	`$("#idBotao").submit(function(){ alert("Formulário enviado"); return true; });`
.hover()	Duas funções para serem executadas quando o mouse entrar e sair do elemento respectivamente.	`$("#idElemento").hover( function(){ //Função quando o mouse entrar }, function(){ //Função quando o mouse sair } );`
.bind()	Associar mais de um evento a um elemento.	`$("#meuId").bind({ click: function(){ //Evento click }, error: function(){ //Manipulador de erros } });`
event.preventDefault()	Impede a ação padrão de um evento.	`$("#meuId").click(function(){ event.preventDefault(); });`

Tabela 2. Lista de métodos CSS da biblioteca *jQuery*.

Método	Descrição	Exemplo
.css()	Definir um valor ou mais, de uma propriedade *css* para um elemento.	`$("#meuId").css("background-color", "#FFFFFF");`
		`$("#meuId").css("background-color": "#FFFFFF", "color":"#000000");`
	Obter o valor de uma propriedade css.	`var valor = $("meuId").css("background-color");`
.addClass()	Adicionar uma classe a um determinado elemento.	`$("#meuId").addClass("minhaClasse");`
.removeClass()	Remover uma classe de um determinado elemento.	`$("#meuId").removeClass("minhaClasse");`
.offSet()	Obter as coordenadas de um elemento.	`var offSet = $("elemento").offset();`
		`$("#coordenadas").html("Left:" + offSet.left + "Top:" + offSet.top);`
	Definir as coordenadas de um elemento.	`$("elemento").offset({top: 25, left: 25});`

Tabela 3. Lista de métodos para manipulação do DOM com *jQuery*.

Método	Descrição	Exemplo
.append()	Inserir um conteúdo no fim do elemento correspondente.	`$("#meuId").append("<p>Novo parágrafo</p>");`
.text()	Obter o conteúdo do elemento correspondente.	`var p = $("p").text();`
.remove()	Remover um elemento do DOM.	`$("#meuId").remove();`
.replaceAll()	Substituir um elemento pelo elemento correspondente.	`$("Novo conteúdo").replaceAll("p");`
.attr()	Obter o valor de um atributo do elemento correspondente.	`$("a").attr("title");`
	Definir o valor de um atributo do elemento correspondente.	`$("a").attr("title", "Título do link");`
.removeAttr()	Remover o atributo do elemento correspondente.	`$("a").removeAttr("title");`
.val()	Obter o valor do elemento correspondente.	`var valor = $("input").val();`
	Definir o valor do elemento correspondente.	`$("input").val("Valor");`

Tabela 4. Lista de efeitos do *jQuery*.

Efeito	Descrição	Exemplo
.hide()	Ocultar um elemento	`$("elemento").click(){ $("#meuId").hide("slow"); }`
.show()	Mostrar um elemento	`$("elemento").click(){ $("#meuId").show("slow"); }`
.animate()	Executar uma animação por um conjunto de propriedades CSS.	`$("elemento").animate({ left:150, opacity: 4 },{ duration: 1000 });`
.fadeOut()	Ocultar um elemento com opacidade.	`$("#meuId").click({ $("elemento").fadeOut(500); });`
.fadeIn()	Mostrar um elemento, surgindo com opacidade.	`$("#meuId").click({ $("elemento").fadeIn(200); });`

ao servidor sem recarregar a página. A Tabela 5 apresenta uma lista de métodos para manipulação do AJAX a partir do *jQuery*.

Caso não seja especificado nenhum tipo de dado a ser retornado, o servidor retornará um MIME XML. Abaixo é apresentada uma lista de tipos de dados disponíveis que podem ser retornados pelo servidor:

- XML – retorna um documento xml
- HTML – retorna um código html
- SCRIPT – retorna um tag script
- JSON – retorna um objeto JavaScript
- JSONP – retorna blocos de dados JSON
- TEXT – retorna um sequência de texto

O uso paralelo de outras bibliotecas, juntamente com o *jQuery*, poderá causar conflitos, pois o caracter $ é usado no *jQuery* como um apelido para *jQuery*; só que em outras bibliotecas *JavaScript*, o caracter $ é usado para identificar seus elementos. Portanto deve-se usar o método noConflict() que permite ser criado um novo apelido para o *jQuery*, ou continuar usando o caracter $ sem conflitos.

Exemplos:

```
<script type="text/javaScript">
  var $j = jQuery.noConflict(); // $j como novo
apelido para jQuery
</script>

<script type="text/javaScript">
  jQuery.noConflict(function($){
    //Código jQuery usando como apelido $
  })(jQuery);
</script>
```

Tabela 5. Lista de métodos AJAX do *jQuery*.

Método	Descrição	Exemplo
$.ajax()	Realiza uma solicitação HTTP(AJAX)	`$.ajax({ url: "url", type: "POST", data: "dados", dataType: "json" }).done(function(){ alert("Sucesso"); }).fail(function(){ alert("Erro"); });`
$.post	Carrega dados do servidor por uma requisição POST.	`$.post("url",function(data){ });`
$.get	Carreda dados do servidor por uma requisição GET.	`$.get("url",function(data){ });`
$.getJSON()	Carrega dados do servidor usando um solicitação HTTP GET e retorna um JSON.	`$.getJSON("url", function(data){ alert(data[0].nome); });`
.load()	Carrega dados do servidor e coloca o html retornado no elemento correspondente.	`$("#meuId").load("url");`
.serializeArray()	Codifica os elementos de um formulário com uma matriz de nome e valor.	`Var fields = $("#idFormulario").serializeArray();`

Aplicações web interativas com *jQuery*

O *jQuery* possui uma grande quantidade de componentes e *plugins* para deixar a interface mais interativa, além das suas plataformas de extensão *jQuery* UI que possui um conjunto de elementos widgets simples de configurar e personalizar, facilitando a criação de aplicações web. Possui também o *jQuery Mobile* que permite criar aplicações compatíveis e adequadas aos principais dispositivos móveis (Android, BlackBerry, Windows Phone, entre outros) e navegadores modernos. Combinando funcionalidades e design com o *jQuery*, é possível aumentar a interatividade das aplicações web com o usuário, permitindo interfaces adaptáveis às necessidades e preferências do usuário.

jQuery UI

O *jQuery* UI é uma plataforma de extensão do *jQuery* que possui diversos elementos de interação, *widgets* e efeitos que proporcionam animações de alto nível para construir aplicações web interativas.

Os elementos de interação possuem funcionalidades que dependem da ação do usuário, permitindo aos usuário modificar posições e tamanhos de elementos da página, deixando a interface adaptável. É apresentada a seguir uma lista dos elementos de interação do *jQuery* UI:

- *Draggable* (Arrastar): permite ao usuário arrastar qualquer elemento da página e posicioná-lo no local desejado, clicando sobre ele com o mouse.
- *Droppable* (Soltar): permite ao usuário arrastar elementos da página e soltá-los dentro de outros elementos.
- *Resizable* (Redimensionar): permite ao usuário controlar as dimensões de um elemento, podendo redimensioná-lo, deixando-o com altura e largura desejadas.
- *Selectable* (Selecionar): permite ao usuário selecionar um elemento da página ou um conjunto de elementos.
- *Sortable* (Ordenar): permite ao usuário ordenar elementos da página da forma desejada arrastando-os e soltando-os.
- Os elementos widgets são componentes visuais prontos, com efeitos e visuais que aumentam a interatividade e despertam o interesse do usuário, além de facilitar a criação das aplicações web. Segue abaixo uma lista de elementos widgets do *jQuery* UI:

- *Acordion*: no elemento acordion, apresentado na Figura 1, o conteúdo é dividido em seções com opções de abrir e fechar para ocultar e mostrar conteúdos, economizando espaço da tela e permitindo ao usuário escolher o conteúdo que deseja visualizar.

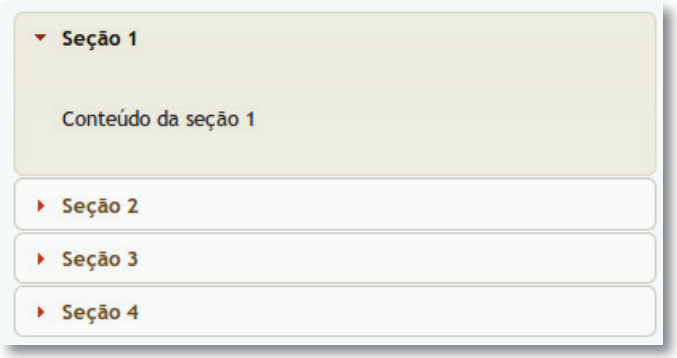


Figura 1. Elemento *Acordion* do *jQuery*.

- *Autocomplete*: no elemento autocomplete (Figura 2), palavras são sugeridas enquanto o usuário digita, facilitando o uso para o usuário.

Figura 2. Elemento AutoComplete do *jQuery*.

- *Datapicker*: o elemento *datapicker* (Figura 3) é uma janela popup para selecionar a data a ser digitada em um campo por um calendário interativo.

Figura 3. Elemento *Datapicker* do *jQuery*.

- *Dialog*: o elemento *dialog* (Figura 4) é uma caixa de diálogo que sobrepõe o conteúdo da janela, protegendo-o.

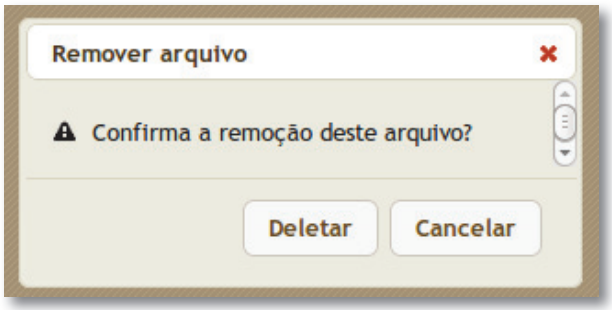


Figura 4. Elemento *Dialog* do *jQuery*.

- *Tabs*: o elemento *Tab* (Figura 5), possui guias que dividem o conteúdo em seções, organizando o conteúdo da página.

Figura 5. Elemento *Tab* do *jQuery*.

Plugins

Os *plugins* podem ser criados pelos próprios usuários da biblioteca, e, por ter uma comunidade muito extensa, o *jQuery* possui vários plugins de diferentes temas, entre eles slides, tabelas, áudio, vídeo, menus, calendários para ajudar na criação das aplicações. São apresentados a seguir alguns exemplos.

- *Slides*: existe uma série de plugins de slides com exibição de imagens de diversas formas e funcionalidades que se adequam a diferentes tipos de interface e necessidades. A Figura 6 apresenta um tipo de slide simples e básico para exibição de imagens feito com a biblioteca *jQuery*.

Figura 6. *Plugin Slide Orbit*.

- *DataTable*: o *plugin DataTable*, apresentado na Figura 7, adiciona controles de interação a tabelas HTML, dentre eles paginação, filtragem, manipulação de colunas, entrada de dados a partir de fontes de dados AJAX, DOM, matriz *javaScript*, e processamentos do lado do servidor (Hypertext Preprocessor - PHP, C-Sharp - C# etc).

- *Plugins MB.JQUERY*: o mb.jquery é um repositório de *plugins* do *jQuery*.
  - Mb.*Container*: o *Container* é um *plugin* formado por caixas flutuantes que possibilitam ao usuário

Exibir 10 registros por página

Nome	Codigo IBGE	Latitude	Longitude	Altitude
Adamantina	3500105	-21.685	-51.073	401
Adolfo	3500204	-21.235	-49.644	443
Aguaí	3500303	-22.059	-46.979	660
Agudos	3500709	-22.469	-48.988	580
Alambari	3500758	-23.551	-47.899	585
Alfredo Marcondes	3500808	-21.955	-51.413	448
Altair	3500907	-20.524	-49.059	557
Altinópolis	3501004	-21.026	-47.374	889
Alto Alegre	3501103	-21.581	-50.164	521
Alumínio	3501152	-23.535	-47.262	790

Mostrando 1 a 10 de 645 regristros

Figura 7. *Plugin DataTable*.

a modificação da interface de acordo com o seu interesse, podendo minimizar, maximizar, fechar, redimensionar e posicionar cada caixa, escolhendo os assuntos de sua preferência para deixar visível. O *Container* permite que o sistema possua diferentes possibilidades de uso, abrangendo as diferentes necessidades dos diversos usuários. A Figura 8 apresenta as caixas flutuantes de acordo com a apresentação padrão do sistema e a Figura 9 apresenta uma disposição diferente para as mesmas caixas apresentadas na Figura 8.

jQuery Mobile

O *jQuery Mobile* é uma estrutura móvel do *jQuery* que permite a criação de uma aplicação web combatível com as plataformas móveis mais populares (Android, BlackBerry, Windows Phone, entre outros) possibilitando o uso da aplicação pelos diferentes dispositivos móveis e juntamente com o *jQuery*UI é possível criar interfaces adequadas ao uso nesses diferentes dispositivos, aumentando as possibilidades de acesso da aplicação e a interatividade com o usuário.

Resultados e discussões

Para o usuário final, é importante que um sistema web tenha as principais informações disponíveis de forma simples e interativa, e essa necessidade aumenta quando a pessoa usa tal sistema para fins de trabalho. Não seria útil um sistema muito burocrático de se usar, tampouco um sistema simples demais e pobre de conteúdo. Um sistema pode ser rico em conteúdo e ao mesmo tempo ter rápida utilização, tudo vai depender das funcionalidades e ferramentas disponíveis para alcançar uma interatividade satisfatória para o usuário.

Existem pessoas que não dominam profundamente a área da informática, mas que apesar disso dependem de alguma informação que deve ficar disponível em meio eletrônico, como por exemplo um produtor rural que deve ter à disposição a cotação diária dos preços no mercado agropecuário ou um criador de gado de leite que deve ter acesso à distância às informações geradas na ordenha, principalmente a produção diária de leite pelas vacas em lactação. Esse é um exemplo de um grupo de pessoas que, dependendo do tama-

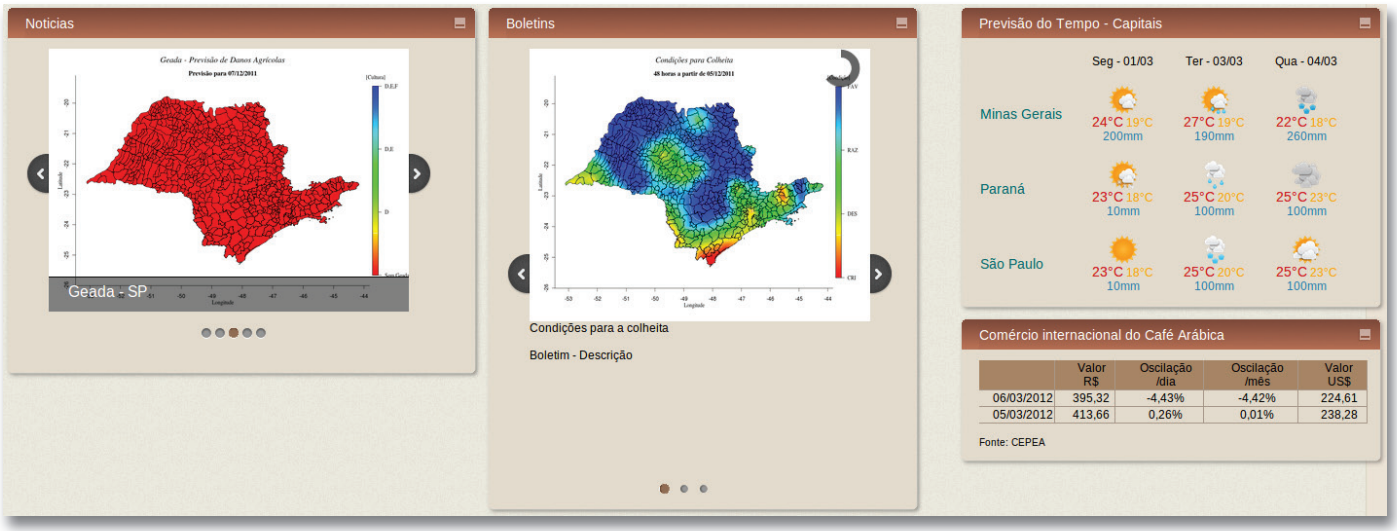


Figura 8. *Plugin Container* - Interface padrão do sistema SIMAFF-Café.

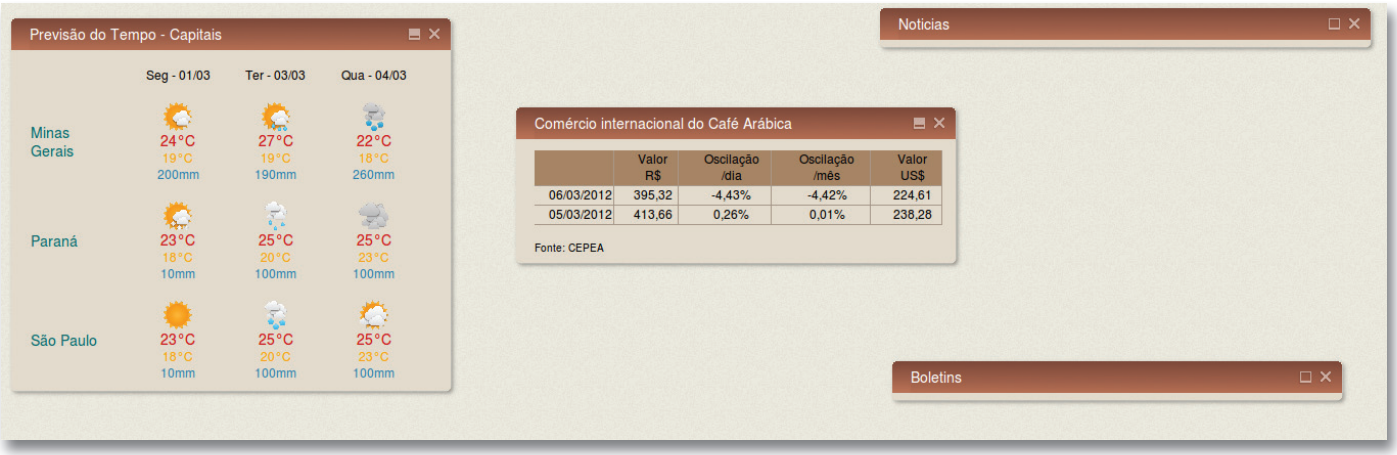


Figura 9. *Plugin Container* - Interface do sistema SIMAFF-Café modificada pelo usuário.

nho da propriedade, movimentam muito dinheiro, mas que geralmente não são grandes conhecedores da informática. Para essas pessoas é indispensável que existam ferramentas rápidas e simples para obterem informações, caso contrário terão atraso nas atividades de gerenciamento da produção e isso resultará em prejuízo.

Nessa linha de trabalho, o estudo detalhado da biblioteca *jQuery* que deu respaldo para elaboração deste trabalho auxiliou na decisão sobre quais ferramentas utilizar no desenvolvimento da interface do sistema web que está em desenvolvimento no projeto SIMAFF-Café. A fim de se obter uma interface interativa e adaptável para o sistema de monitoramento de pragas e doenças do café nos estados de São Paulo, de Minas Gerais e do Paraná, foram desenvolvidos protótipos de interfaces utilizando diversos componentes e plugins da biblioteca *jQuery*.

A Figura 10 apresenta a página inicial do sistema onde foi utilizado o componente *Tab* e o *plugin Container*, visando a possibilidade do usuário escolher as informações de sua preferência para serem visualizadas, podendo adaptar a interface de acordo com as suas necessidades. Também foi utilizado um plugin de slide para exibição de notícias.

As Figuras 11 e 12 apresentam respectivamente o uso do *plugin DataTable* e do componente *Dialog* para o monitoramento de dados do sistema pelo administrador.

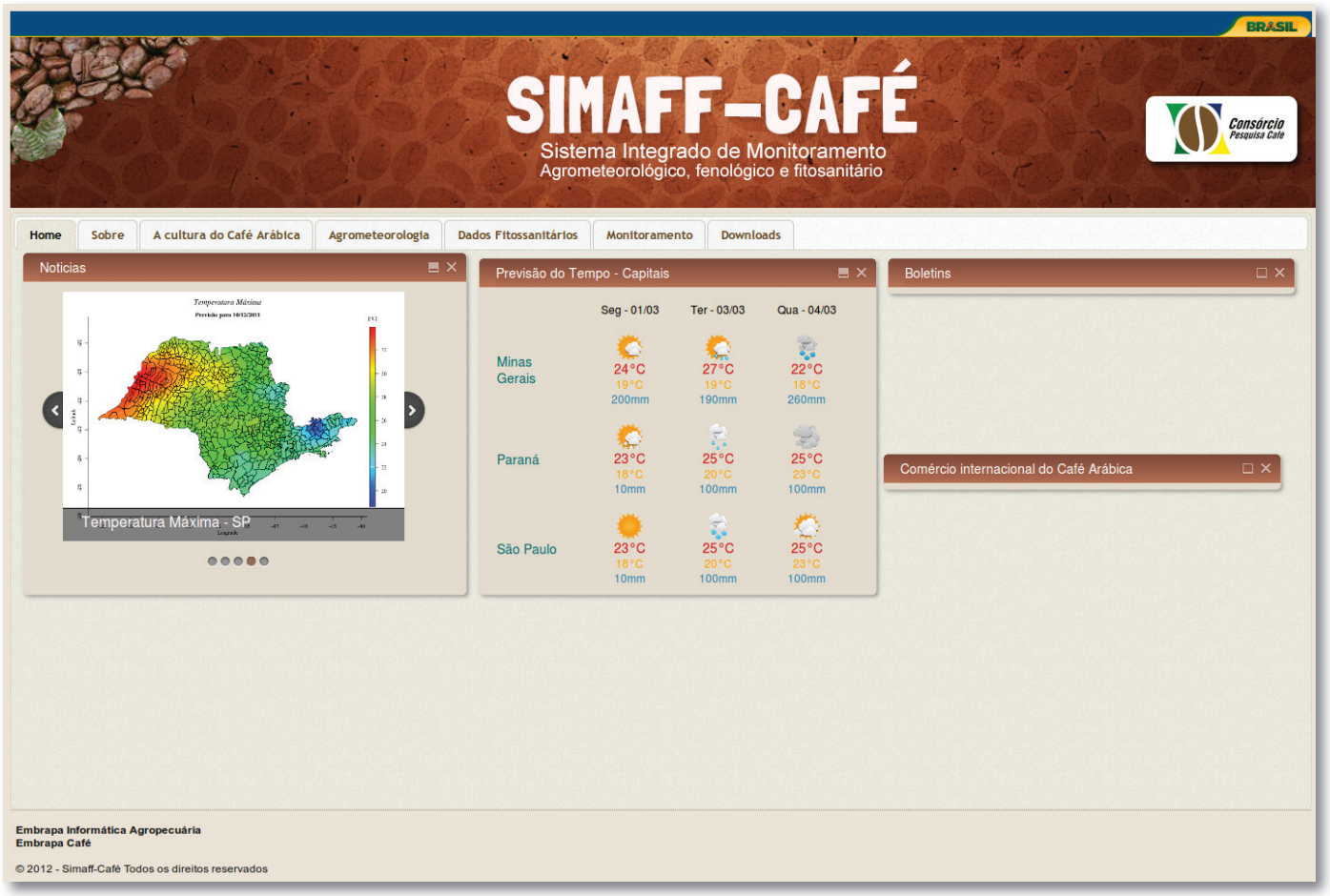


Figura 10. Página Inicial do sistema SIMAFF-Café.

Figura 11. Utilização do *plugin DataTable* na interface do sistema SIMAFF-Café.

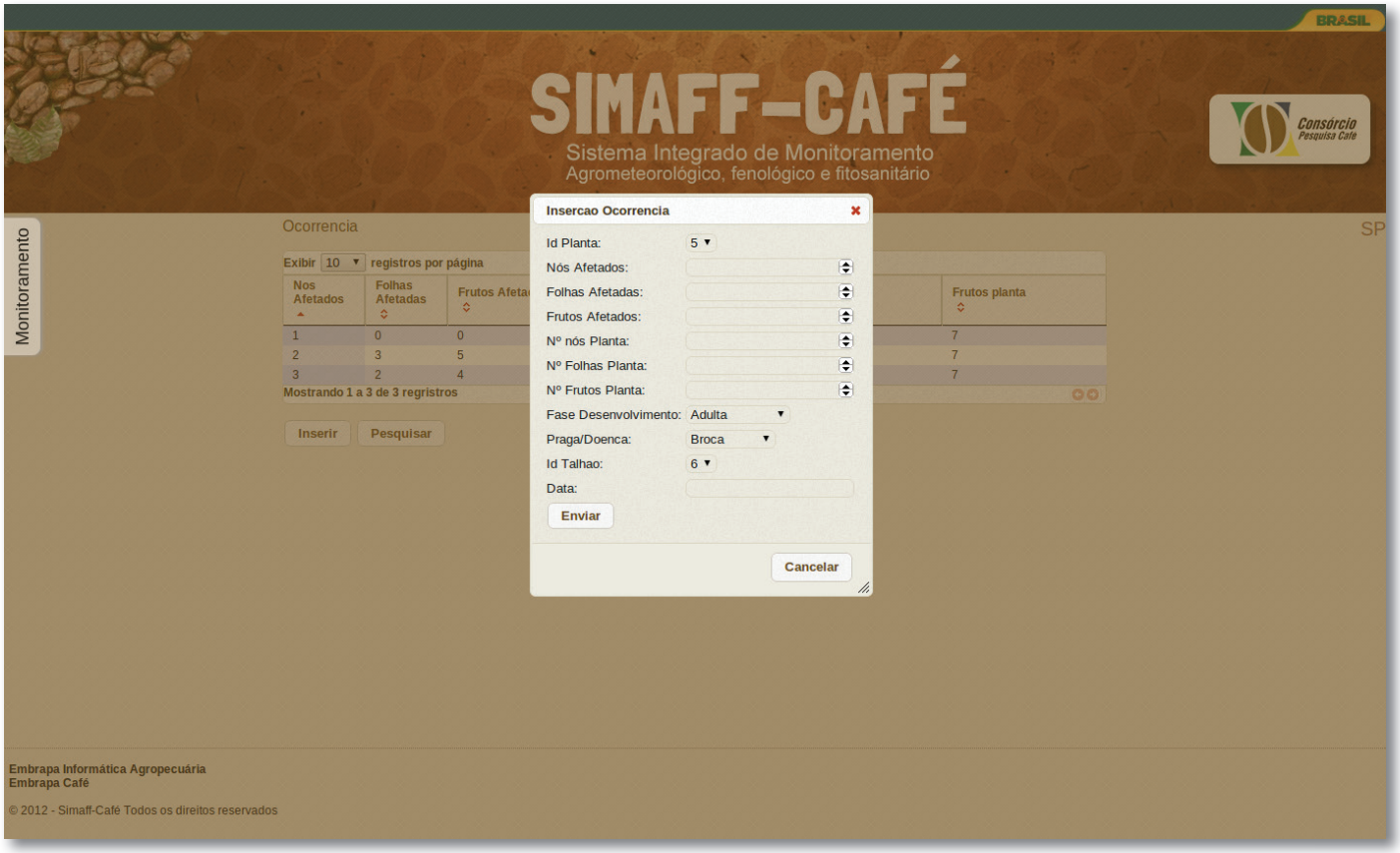


Figura 12. Utilização do componente *Dialog* na interface do sistema SIMAFF-Café.

Conclusão

A biblioteca *jQuery* possui uma vasta documentação, com uma sintaxe simples que ajuda no desenvolvimento de aplicações em poucas linhas de código, e com os seus componentes e *plugins* é possível criar aplicações web interativas e adaptáveis, possibilitando interfaces simples e usáveis, contribuindo para atender melhor as necessidades e preferências do usuário.

O processo de validação do uso da biblioteca *jQuery* para criação de interfaces interativas está sendo realizado por meio da criação da interface do sistema SIMAFF-Café com o objetivo de permitir aos usuários administradores a possibilidade de configurar a interface de acordo com sua preferência e seu modo de trabalho.

Referências

BOZZON, A.; COMAI, S.; FRATERNALI, P.; CARUGHI, G. T. Conceptual modeling and code generation for rich internet applications. In: INTERNATIONAL CONFERENCE ON WEB ENGINEERING, 6., 2006, Palo Alto. **Proceedings**... New York: ACM., 2006. ICWE 06.

NERIS, V. P. A.; BARANAUSKA, M. C. C. A Framework for designing flexible systems. In: IEEE INTERNATIONAL CONFERENCE SYSTEMS, MAN, AND CYBERNETICS, 2011, Anchorage, Alaska. **Proceedings**... [S.l.]: IEEE, 2011. p. 2600-2607. SMC 2011.

SILVA, P. E. C.; SILVA, P. F. P. da. Interfaces adaptativas aplicadas a Sistemas de Informação - características desejáveis. **Revista Abstração**, Florianópolis, Ano 4. n. 2, nov. 2007.

Literatura recomendada

DATATABLES. Disponível em: <http://datatables.net/>. Acesso em: 10 out. 2012.

INTRODUÇÃO à jQuery: caminho, verdade e vida. Disponível em: <http://algoritmizando.com/desenvolvimento/introducao-a-jquery-caminho-verdade-e-vida/>. Acesso em: 10 out. 2012.

JQUERY. **Poder e simplicidade**. Disponível em: <http://blog.andrefaria.com/jquery-poder-e-simplicidade>. Acesso em: 10 out. 2012.

JQUERY: the write less, do more, javascript library. Disponível em: <http://jquery.com/>. Acesso em: 10 out. 2012.

JQUERY: user interface. Disponível em: <http://jqueryui.com/>. Acesso em: 10 out. 2012.

JQUERY Mobilie. Disponível em: <http://jquerymobile.com/>. Acesso em: 10 out. 2012.

MB.IDEAS: repository. Disponível em: <http://pupunzi.com/>. Acesso em: 10 out. 2012.

ZURB. Disponível em: <http://www.zurb.com/playground/orbit-jquery-image-slider>. Acesso em: 10 out. 2012

Comunicado Técnico, 113

Embrapa Informática Agropecuária
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-886 - Campinas, SP
Fone: (19) 3211-5700
Fax: (19) 3211-5754
http://www.cnptia.embrapa.br
e-mail: cnptia.sac@embrapa.com.br

1ª edição on-line - 2012

CGPE 10222