FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE - FANESE

NÚCLEO DE PÓS-GRADUAÇÃO E EXTENSÃO – NPGE

CURSO DE PÓS-GRADUAÇÃO “LATO SENSU”

ESPECIALIZAÇÃO EM SISTEMAS DE INFORMAÇÃO PARA WEB

JAVASERVER FACES (JSF): UM ESTUDO COMPARATIVO ENTRE BIBLIOTECAS DE COMPOENENTES

Rodrigo Menezes da Conceição

Resumo

Este artigo versa sobre a tecnologia de desenvolvimento Java para web JavaServer Faces (JSF), falando sobre sua arquitetura e comentando todos os benefícios em relação as tecnologias Java anteriores. Entre umas das características que trouxeram grande valor para o JSF, foi seu sistema de componentes de interface de usuário e a partir dos mesmos surgiram bibliotecas de componentes. Neste artigo foram escolhidas três das principais bibliotecas JSF e foi realizado um estudo comparativo entre elas.

Palavras-chave: Java. JavaServer Faces. Programação. Internet.

ABSTRACT

This article focuses on Java development technology for Web JavaServer Faces (JSF), talking about its architecture and commenting on all the benefits over the previous Java technologies. Among some of the characteristic that brought great value to the JSF, was its system of user interface components and libraries, which have emerged from the same components. This article were chosen from three main JSF component libraries and was conducted a comparative study between them.
Keywords: Java. JavaServer Faces. Programing. Internet.

1 INTRODUÇÃO

A Desde seu início a Internet mostrou ser uma ferramenta muito poderosa para compartilhamento de informação, fornecendo em seu inicio um conjunto de sítios com textos estáticos ligados entre si por meios de hiperlinks.

Com o passar do tempo surgiram as primeiras aplicações Web, utilizando tecnologias que permitiam o desenvolvimento de sítios com conteúdo dinâmico. Para atender esse avanço, diversas tecnologias surgiram, e, entre elas, podemos destacar o Java, que teve muito sucesso em atender esta demanda de tecnológica.

A tecnologia de aplicações web com Java se iniciou com os Servlets, que apesar de serem um avanço para época, logo foram se identificando as primeiras limitações desta tecnologia.

Como outro passo da evolução do Java, surgiu o JavaServe Pages (JSP), arquitetado para resolver as limitações dos Servlets, e com o tempo, outras tecnologias foram surgindo no mundo Java.

Atualmente a última tecnologia Java para o desenvolvimento de aplicações Web é o JavaServer Faces, que foi criada a pedido da comunidade mundial Java, que queria uma tecnologia oficial Java que seguisse o padrão MVC (Model – Control – View) de separação de camadas.

O JSF foi construído utilizando como base as tecnologias Java anteriores, seguindo a premissa de construir uma tecnologia que fosse robusta, escalável, confiável, e que ainda permitisse um desenvolvimento mais produtivo que as tecnologias anteriores.

Entre estes conceitos, um dos principais foi o de componentes de interface de usuário (UI), que trouxe uma vantagem muito grande em termos de reutilização de código e padronização, permitindo que equipes de desenvolvedores criassem bibliotecas de componentes, que poderiam ser reaproveitadas em muitos projetos. Este reaproveitamento motivou equipes a se especializarem na criação de bibliotecas poderosas e versáteis, tais como RichFaces, ICEFaces e a MyFaces Tomahawk.

Outro grande beneficio do componentes JSF foi facilitar a criação de aplicações denominadas de RIA (Rich Internet Aplications), que são definidas por [Bush e Koch 2009] como um novo tipo de aplicação web que veio para superar as limitações das aplicações web tradicionais no que diz respeito a usabilidade e interatividade.

Este artigo tem como principais objetivos: (i) explanar sobre o JavaServer Faces mostrando de onde esta tecnologia evoluiu, bem como suas características e vantagens; (ii) falar sobre a importância das bibliotecas de componentes JSF e realizar um estudo comparativo entre três das mais utilizadas pela comunidade Java: RichFaces, ICEFaces e a MyFaces Tomahawk.

2 JAVASERVER FACES (JSF)

Em um momento da evolução das tecnologias Java para aplicações web surgiu o JSP Model 2 que introduziu o uso do padrão MVC (Model - Control - View), que possui uma forte separação de camadas [Java BluePrints: Model-View-Controller]:

1. Model: Lógica de negócios e persistência de dados. Representa os dados e as regras de negócio de acesso e modificação destes dados.

2. View: Interface com o usuário. Acessa os dados através do model e especifica como os dados devem ser apresentados. É de sua responsabilidade manter a consistência nessa apresentação quando há mudanças no model, sendo obrigado, quando necessário, a se comunicar com o model para garantir que esteja sempre com os dados mais atualizados.

3. Controller: Controla os detalhes que envolvem a comunicação das ações do usuário, como pressionamento de teclas e movimento do mouse, à camada Model. Baseado nas interações com usuário e nas ações do model, o controller responde selecionando o view apropriado.

Esta separação de camadas trás inúmeros benefícios como: reaproveitamento de componentes e uma divisão muito melhor de responsabilidades.

Ficou muito claro que a arquitetura MVC trouxe ganhos muito grandes e que era necessária a criação de uma especificação oficial MVC para Java. A comunidade Java através do JCP (Java Community Process), criou o pedido de especificação [JSR 127: JavaServer Faces] para atender esta necessidade.

A especificação JSF foi desenvolvida por uma comunidade de grandes mentes no campo de desenvolvimento de interfaces do usuário (UI) de aplicações Web. Tentou-se colher as melhores idéias e abordagens dos vários frameworks, e uni-las de maneira coerente em uma especificação sólida.

2.1 As peças do quebra cabeças

Como a maioria das tecnologias, o JSF possui um conjunto próprio de termos, que formam a base conceitual para os recursos que proporciona [JSR 127: JavaServer Faces]. São muitos termos como: componentes de interfaces (UI), validadores, renderizadores, backing beans, entre outros, que serão encontrados por quem começar a desenvolver utilizando JSF.

Podemos ter uma idéia geral da função de cada um observando a Tabela 1.

Tabela 1: Termos importantes do JSF.

Termo	Descrição
Componente UI (também chamado de control ou simplesmente componente)	Um objeto sem estado de conversação associado, mantido no servidor, que fornece uma funcionalidade específica para interagir com um usuário final. Componentes UI são JavaBeans com propriedades, métodos e eventos. Elas são organizadas em uma estrutura, que é uma árvore de componentes geralmente apresentados como uma página.
Renderizador	Responsável por exibir um componente da UI e transformar uma entrada do usuário em um valor do componente. Renderizadores podem ser projetados para funcionar com um ou mais componentes UI, um componente UI pode ser associado a diversos renderizadores.
Validador	Responsável por garantir que o valor digitado por um usuário é aceitável. Um ou mais validadores podem ser associados com um componente da UI.
Backing beans	Uma especialização do JavaBeans que coleta valores a partir de componentes da UI e implementam métodos dos eventos dos ouvintes. Eles também podem possuir referências a elementos da UI.
Conversor	Converte o valor de um componente para String e também executa a operação reversa. Um componente UI pode ser associado com apenas um conversor.
Eventos e ouvintes(listeners)	JSF usa o modelo JavaBean de evento/ouvinte(também usado pelo Swing). Componentes UI (e outros objetos) geram eventos e ouvintes podem ser registrados para lidar com esses eventos.
Mensagens	Informações que são exibidas de volta para o usuário. Em qualquer parte da requisição (backing beans, validadores, conversores, e assim por diante) mensagens de erro podem ser geradas ou podem ser exibidas informações para o usuário.
Navegação	A capacidade de passar de uma página para a próxima. JSF possui um poderoso sistema de navegação que é integrado com os ouvintes dos eventos.

Fonte: Própria.

2.2 Ciclo de Vida

O JSF possui um ciclo de vida de requisição baseado em fases, que são executadas seguindo uma ordem. Para um desenvolvedor não é obrigatório que ele conheça a fundo o funcionamento interno do JSF, mas é um diferencial muito importante, ter um conhecimento sobre o ciclo de vida, sabendo a ordem e o que acontece em cada uma destas fases. Este conhecimento permite que sejam construídas aplicações melhores, pois o desenvolvedor sabe com detalhes que operações estão acontecendo em cada fase, tornando mais fácil inclusive detectar algum possível erro, ao se identificar em qual fase ele ocorreu. Podemos entender melhor este ciclo de vida observando a Erro: Origem da referência não encontrada1, pois ela mostra o que ocorre em cada uma das fases, ficando clara a responsabilidade de cada uma.

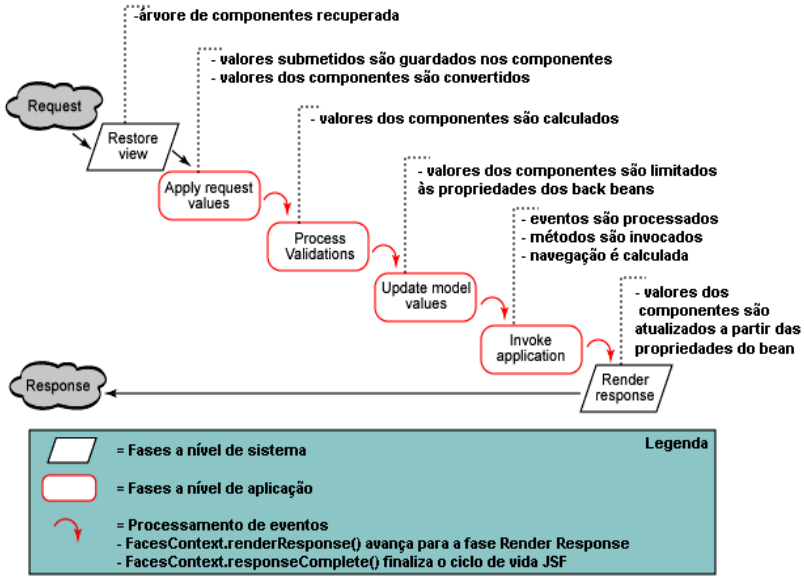

Figura 1: *Componentes de calendário no modo popup [**HIGHTOWER, 2008.**]*

3 BIBLIOTECAS DE COMPONENTES JSF

3.1 Introdução

Como foi visto durante este artigo, a especificação JSF foi criada para prover um *framework* de programação web poderoso, confiável e escalável, construído em cima da tecnologia Java, que tem se mostrado ao longo dos anos ser bastante confiável, o que explica sua maciça adoção.

Mesmo os requisitos acima sendo fundamentais, era claro que o JSF precisava ter ainda uma outra característica, a de ser bastante produtivo. E entre os pilares que fornecem esta produtividade, podemos destacar os componentes UI.

Como foi visto na evolução da tecnologia Java, criar a interface com o usuário, sempre foi uma tarefa complicada e isto fica claro pelo número de evoluções que foram necessárias neste sentido.

O JSF definiu um modelo de componentes, visando facilidade de uso e produtividade, e na sua implementação padrão trouxe implementação para todos os componentes básicos HTML. Este conjunto básico de componentes serve como pontapé inicial no desenvolvimento das aplicações e em muitos casos são suficientes, mas também é permitindo ao desenvolvedor criar seus próprios componentes, que depois de criados poderão ser utilizados em muitos outros projetos.

Em uma fábrica de software, é comum que alguns desenvolvedores fiquem responsáveis por alguma tarefa específica, e em muitos casos há desenvolvedores responsáveis pela criação e manutenção de componentes. A partir desta grande necessidade por componentes sofisticados, foi esperado o surgimento de projetos que direcionassem seus esforços para a construção de bibliotecas de componentes, o que levou ao surgimento de várias alternativas bastante maduras e com muita qualidade.

3.2 Escolha das bibliotecas estudadas

E o objetivo principal deste artigo foi selecionar e fazer um estudo comparativo das principais bibliotecas de componentes JSF do mercado. Entre os critérios utilizados para a escolha das bibliotecas, ficam destacados:

1. Aceitação pela comunidade Java.
2. Acabamento visual.
3. Documentação.
4. Número de componentes.

Seguindo estes critérios foram escolhidas três bibliotecas:

1. RichFaces
2. ICEFaces
3. Apache Tomahawk

3.3 Comparativo proposto

No nosso comparativo foram utilizados três projetos de demonstração acessados pela internet:

1. Demonstrativo não oficial de componentes do MyFaces Tomahawk.
2. Demonstrativo oficial de componentes do Jboss RichFaces.
3. Demonstrativo oficial de componentes do ICEFaces.

Neste início houve uma decepção quando ao Tomahawk, que não possuía um demonstrativo oficial de seus componentes, então foi usado um demonstrativo encontrado na jsfmatrix.com, que é um sítio que traz uma tabela comparativa entre bibliotecas de componentes JSF.

Neste comparativo foram usadas as seguintes versões das bibliotecas: RichFaces 3.2, ICEFaces 1.7 e Tomahawk 1.1.7.

Na análise dos componentes vários critérios serão avaliados tais como:

1. Usabilidade.
2. Uso de AJAX.
3. Compatibilidade com os navegadores: Microsoft Internet Explorer 7 e Firefox 3.
4. Acabamento e apelo visual.
5. Diferenciais perante os concorrentes.

Foi feita uma qualificação dos componentes. O critério de pontuação foi elaborado seguindo os itens citados anteriormente, escolhendo uma classificação para cada um dos itens, que pode ser observada na Tabela 2.

Tabela 2: Classificação proposta para quantificar a qualidade dos atributos de cada componente estudado.

Pontuação	Significado
3	Representa o nível desejado de qualidade. Esta classificação é dada quando todas as características necessárias são plenamente atendidas e com um nível bom de qualidade.
2	Esta classificação é dada quando as características necessárias são atendidas de maneira satisfatória.
1	Esta classificação é dada quando as características necessárias não são totalmente atendidas, ou são atendidas sem um nível bom de qualidade.

Fonte: Própria.

Com base na nota atribuída a cada critério, foi calculada uma nota para cada componente analisado e no final do estudo estas notas serão somadas para se obter as notas gerais das três bibliotecas.

As notas de cada biblioteca serão calculadas pela porcentagem da sua pontuação em relação à pontuação máxima possível, que é de cento e trinta e cinco pontos. Deste modo, teremos uma pontuação que irá de de zero a cem.

Foram escolhidos e comparados os seguintes componentes:

1. Calendário: Componente que serve para facilitar a escolha de uma data. Ele ainda pode ser configurado para marcar feriados e e datas especiais. Figura 2.
2. Envio de Arquivo: Componente que permite que o usuário envie arquivos para a aplicação. Figura 3, Figura 4 e Figura 5.
3. Abas: Componente que permite organizar melhor o layout de uma pagina, agrupando elementos comuns em abas. Figura 6, Figura 7 e Figura 8.

4. Editor avançado de texto: Editor que permite, que o usuário formate um texto html de maneira gráfica semelhante ao softwares Dreanweaver e Frontpage. Figura 9 e Figura 10.

5. Árvores: Componente que mostra um arvore de elementos, muito útil para objetos que possuem relacionamento hierárquico. Figura 11, Figura 12 e Figura 13.

6. Menu: Componente para montagem de menu de navegação. Figura 14, Figura 15 e Figura 16.

7. Tabelas dinâmicas: Componente que permite comtagem de tabelas dinâmicas a partir de estruturas de dados como listas e vetores. Figura 17, Figura 18 e Figura 19.

8. Google Maps: Componente que permite interagir com o sistema de mapas e coordenadas globais do google. Figura 20 e Figura 21.

9. Menu de contexto: Componente que permite adicionar menu de contexto a outros componentes. Figura 22 e Figura 23.

Figura 2: *Componentes de calendário no modo popup*

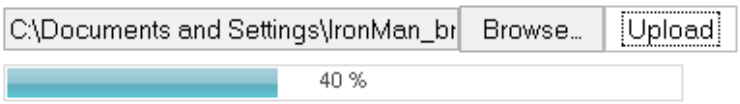

Figura 3: *Componente de envio de arquivo do ICEFaces*

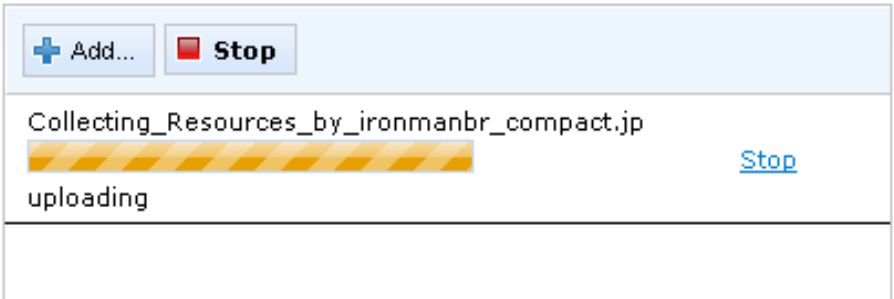

Figura 4: *Componente de envio de arquivo do RichFaces*

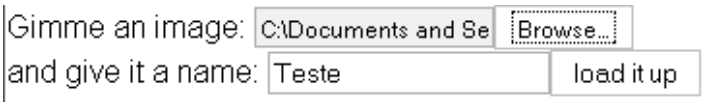

Figura 5: *Componente de envio de arquivo do Tomahawk*

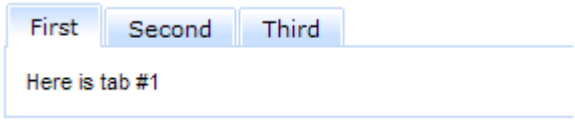

Figura 6: *Componente de Abas do RichFaces*

Figura 7: *Componente de Abas do ICEFaces*

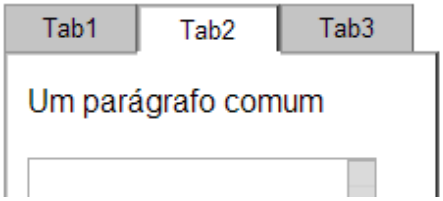

Figura 8: *Compoente de Abas do Tomahawk*

Figura 9: *Componente editor WYSIWYG do Tomahawk*

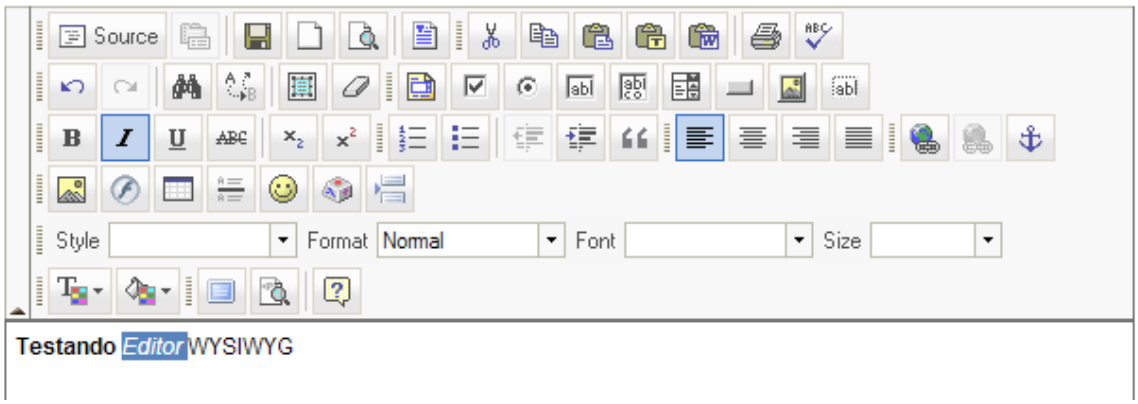

Figura 10: *Componente editor WYSIWYG do ICEFaces*

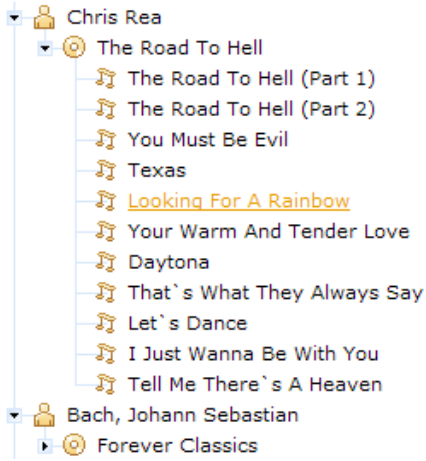

Figura 11: *Componente de Árvore do Richfaces representando uma coleção de música*

Figura 12: *Componente de árvore do Tomahawk*

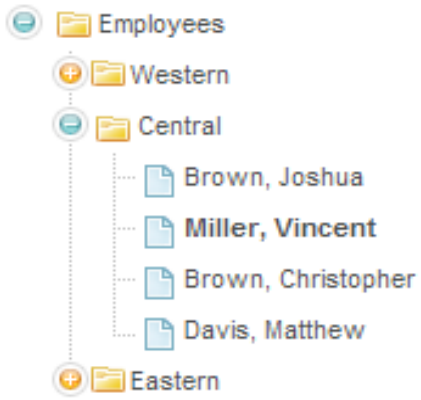

Figura 13: *Componente de árvore do ICEFaces*

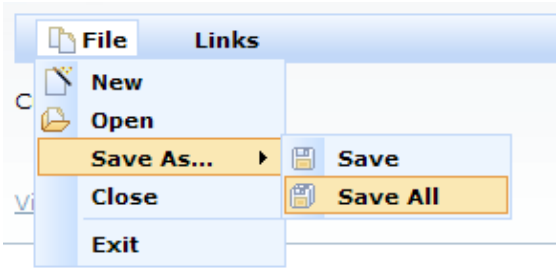

Figura 14: *Componente de menu do RichFaces*

ID	Region	Office	Contact Info		
			First Name	Last Name	Phone
10	Western	Calgary	Ethan	Smith	555-4562
15			Jacob	Smith	555-4563
20			Logan	Smith	555-4564
25			Benjamin	Smith	555-4565
30			Jack	Smith	555-4566
35			Noah	Johnson	555-4567
40			William	Johnson	555-4568
45			Andrew	Johnson	555-4569
46			Samuel	Johnson	555-4570
47		Victoria	Joseph	Johnson	555-4571
50			Daniel	Williams	555-4572
16			Anthony	Williams	555-4573
17			Angel	Williams	555-4574
18			Jacob	Williams	555-4575
100			David	Williams	555-4576

Figura 15: *Componente de tabela dinâmica do ICEFaces mostrando a possibilidade de agrupamento*

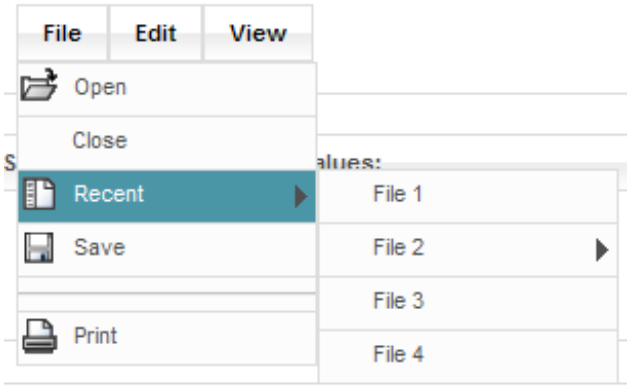

Figura 16: *Componente de menu do ICEFaces*

ID	Car type	Car color
1	car B	blue
6	car J	blue
11	car L	dark blue
7	car I	gray
4	car C	green
8	car M	lightGray
9	car N	magenta
5	car E	orange
2	car A	red
10	car K	unknown
3	car D	yellow

Figura 17: *Componente de tabela dinâmica do Tomahawk*

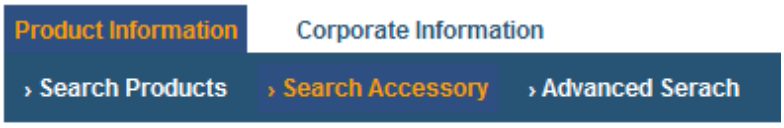

Figura 18: *Componente de menu do Tomahawk*

Sorting Example		
State Name ⇕	State Capital ▲	Time Zone
New York	Albany	GMT-5
Maryland	Annapolis	GMT-5
Georgia	Atlanta	GMT-5
Maine	Augusta	GMT-5
Texas	Austin	GMT-6
Louisiana	Baton Rouge	GMT-6
North Dakota	Bismarck	GMT-6
Idaho	Boise	GMT-8
Massachusetts	Boston	GMT-5
Nevada	Carson City	GMT-8
West Virginia	Charleston	GMT-5
Wyoming	Cheyenne	GMT-7
South Carolina	Columbia	GMT-5
Ohio	Columbus	GMT-5
New Hampshire	Concord	GMT-5
«« « 1 2 3 4 » »»		

Figura 19: *Componente de tabela dinâmica do RichFaces*

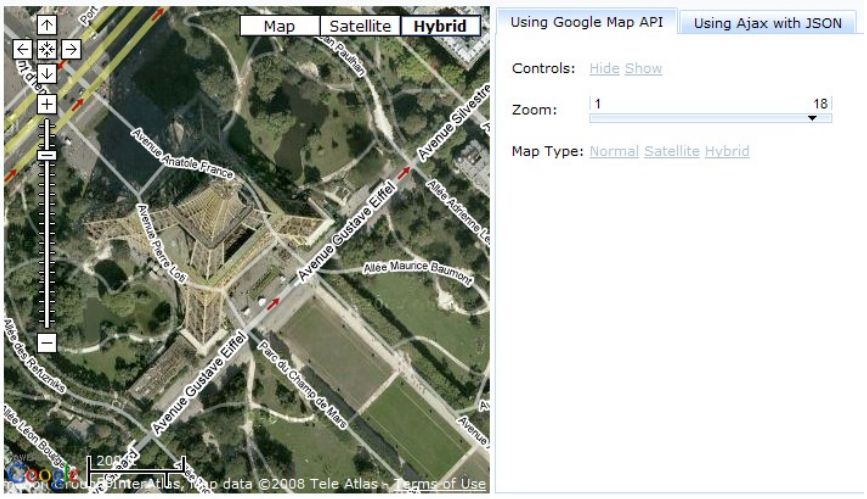

Figura 20: *Componente do Google Maps do RichFaces*

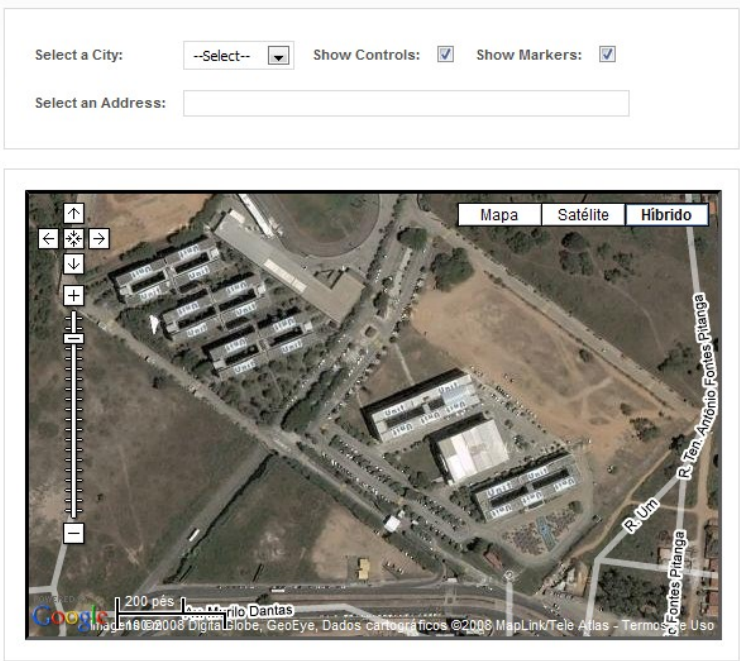

Figura 21: *Componente do Google Maps do ICEFaces*

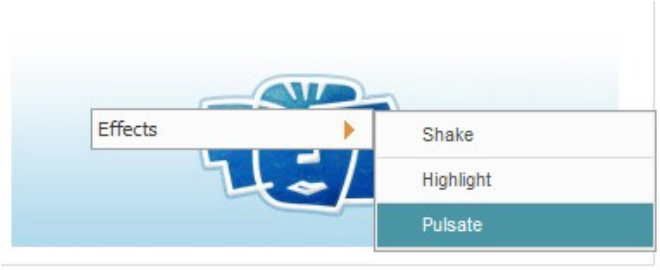

Figura 22: *Componente de menu de contexto do ICEFaces*

Make	Model	Price
Toyota	4-Runner	36582
Toyota	Camry	33508
GMC	Sierra	50459
Ford	T	
Nissan	Maxima	24631
Ford	Explorer	45102
Chevrolet	S-10	37314
Toyota	Avalon	17366
Ford	Explorer	29376
Toyota	Camry	45192

Last Menu Action

Toyota Camry details

GMC Sierra details

Actions

Put **GMC Sierra** To Basket

Read Comments

Go to **GMC** site

Figura 23: *Componente de menu de contexto do RichFaces*

Com base em nossa proposta para o comparativo entre as bibliotecas, chegamos a Tabela 3: Pontuação dos componentes das bibliotecas.

Tabela 3: Pontuação dos componentes das bibliotecas

Componente	Pontuação		
	RichFaces	ICEFaces	Tomahawk
Calendário	14	13	9
Envio de arquivo	15	15	9
Abas	13	14	9
Editor WYSIWYG	-	12	6
Árvore	15	15	7
Menu	14	14	10
Tabelas dinâmicas	14	14	10
Google Maps	14	13	-
Menu de contexto	12	12	-
Total em pontos	**111**	**122**	**60**
Nota da biblioteca	**8,22**	**9,03**	**4,44**

Fonte: Própria.

4 CONSIDERAÇÕES FINAIS

Nosso estudo atribuiu uma nota para cada atributo dos componentes estudados, obtendo assim uma pontuação para cada componente, quantificando sua qualidade perante aos concorrentes. De posse das notas dos componentes, foi criada a tabela 3, que mostra as notas realizando um somatório, obtendo as notas referentes a cada biblioteca estudada. As notas obtidas vieram a corroborar com tudo que foi exposto neste trabalho, mostrando que o RichFaces e ICEFaces possuem uma qualidade melhor que o Tomawank.

Outro ponto reforçado por nosso estudo e corroborado por [Busch e Koch 2009] é que as aplicações RIA são muito importantes para aumentar a qualidade das aplicações Web e nesse quesito as bibliotecas de componentes JSF dão um suporte muito grande,

facilitando o trabalho do desenvolvedor na construção de uma aplicação bastante interativa e com uma boa usabilidade.

REFERÊNCIAS

BUSCH, Marianne e KOCH, Nora. Rich Internet Applications: State-of-the-Art. 2009

GEARY, David e HORSTMANN, Cay. Core JavaServer Faces. Addison Wesley, 2004. 637p.

JSR 127: JavaServer Faces. Disponivel em: <http://jcp.org/en/jsr/detail?id=127>. Acessado em 18/09/2010

MAHMOUD, Qusay. Developing Web Applications with JavaServer Faces. Disponível em: <http://java.sun.com/developer/technicalArticles/GUI/JavaServerFaces/>. Acessado em: 17/06/2010

KURNIAWAN, Budi. Java para Web com Servlets, JSP e EJB. Editora Ciência Moderna, 2002. 807p.

Java BluePrints - Model-View-Controller. Disponível em: <http://java.sun.com/blueprints/patterns/MVC-detailed.html>. Acessado em 18/05/2010

GOYAL, Deepak, and VARMA, Vikas, Introduction do Java Server Faces (JSF), Sun Microsystems presetation.

MANN, Kito D. JavaServer Faces In Action. Editora Manning, 2005. 1038 pg.

JSR 127: JavaServer Faces - Java Specification Requests. Disponível em: <http://www.jcp.org/en/jsr/detail?id=127>. Acessado em 20/03/2010

HIGHTOWER, Richard. JSF for nonbelievers: The JSF application lifecycle. Disponível em: <http://www.ibm.com/developerworks/library/j-jsf2/>. Acessado em 03/05/2010.

Maryka, Stephen. Enterprise Ajax Security with ICEfaces. ICEsoft, 2007.Disponível em: <http://www.icefaces.org/main/resources/whitepapers.iface>. Acessado em 08/05/2010.

Sítio MyFaces Tomahawk. Disponível em: <http://myfaces.apache.org/tomahawk/index.html>. Acessado em 08/07/2010.

Demonstrativo não oficial de componentes do MyFaces Tomahawk. Disponível em: <http://www.irian.at/myfacesexamples/home.jsf>. Acessado em 07/07/2010.

Demonstrativo oficial de componentes do Jboss RichFaces. Disponível em: <http://livedemo.exadel.com/richfaces-demo/index.jsp>. Acessado em 01/07/2010.

Demonstrativo oficial de componentes do ICEFaces. Disponível em: <http://component-showcase.icefaces.org/component-showcase/showcase.iface>. Acessado em 01/07/2010.