ISSN 2316-2872
T.I.S. São Carlos, v. 4, n. 3, p. 249-259, set-dez 2015

Análise comparativa entre dois Frameworks MVC para a Plataforma Java EE: JSF e VRaptor

Rodolpho Sbaraglini Couto, Ivan João Foschini

Resumo. Com o notável avanço de tecnologias para a Internet, aliado ao crescimento de dispositivos conectados a ela, observa-se um grande volume de aplicações inovadoras e escaláveis sendo disponibilizadas a cada dia. Para atender a essa demanda, novas tecnologias, plataformas e frameworks estão sendo evoluídos, com o objetivo de tornar o desenvolvimento dessas aplicações cada vez mais ágil e com maior valor de entrega. Este artigo apresenta um estudo comparativo entre o JSF (JavaServer Faces), framework MVC baseado em componentes definido na especificação Java EE e o VRaptor, framework MVC baseado em ações que, apesar de ser uma iniciativa externa, também é totalmente integrado à plataforma Java EE.

Palavras-chave: MVC, JavaServer Faces, VRaptor

Comparative analysis between two MVC Frameworks for Java EE Platform: JSF and VRaptor

***Abstract**. With the remarkable breakthrough of technologies for the Internet, coupled with the increase of devices connected to it, it is possible to observe a great deal of innovative and scalable applications becoming available every day. To meet this demand, new technologies, platforms and frameworks are being developed in order to make the development of these applications increasingly faster and with higher delivery value. This article presents a comparative study between JSF (JavaServer Faces), MVC framework based on components defined in Java EE specification and VRaptor, MVC framework based on actions that, despite being a foreign initiative, is also fully integrated into Java EE platform.*

***Keywords:** MVC, JavaServer Faces, VRaptor*

I. Introdução

Com o avanço contínuo da Internet e dos dispositivos a ela conectados, cada vez mais é possível notar que aplicações estão sendo implantadas com maior agilidade e qualidade. Isso deve-se ao fato de que a evolução das tecnologias, plataformas e frameworks de desenvolvimento estão crescendo em mesma escala, permitindo maior valor de entrega ao usuário final.

Nos últimos anos, a tecnologia da informação adentrou ao cotidiano das pessoas e transformou modelos de negócios, tornando-se cada vez mais necessária e importante em praticamente todos os nichos de mercado. Aplicações que antes só poderiam ser utilizadas em computadores conectados à rede interna da empresa, hoje evoluíram e estão sendo migradas para outras plataformas que possibilitam o acesso de diversos dispositivos de qualquer lugar do mundo, trazendo acessibilidade e praticidade ao usuário.

Diante disto, padrões de arquitetura de software foram definidos para facilitar a implementação de funcionalidades e aumentar a manutenibilidade das aplicações. Um dos padrões de projetos mais difundidos na comunidade de desenvolvimento é o MVC (Model-View-Controller), que separa a saída da representação de dados das lógicas e regras de negócio, intermediados através de uma camada controladora.

Com a ascensão do MVC na comunidade, muitos frameworks foram desenvolvidos baseados neste padrão. Surgiram duas vertentes de frameworks MVC que influenciam diretamente no seu comportamento: framework MVC baseado em ações e framework MVC baseado em componentes.

A plataforma Java EE (Enterprise Edition) define um conjunto de especificações de frameworks para facilitar o desenvolvimento para a Web. A especificação responsável por implementar o padrão MVC é o JSF (JavaServer Faces), que possibilita desenvolver aplicações baseadas em componentes de forma ágil, abstraindo os detalhes do protocolo HTTP.

Além das especificações oficiais da plataforma Java EE, muitos frameworks são desenvolvidos pela comunidade através de iniciativa própria, sendo que alguns deles podem até servir de base para a definição de novas especificações. O VRaptor é um framework MVC baseado em ações integrado ao Java EE, que permite desenvolver aplicações de forma ágil sem precisar abstrair ou inibir os detalhes do protocolo HTTP.

Departamento de Computação - Universidade Federal de São Carlos (UFSCar)
Caixa Postal 676 – 13.565-905 – São Carlos – SP – Brasil
Autor para correspondência: rodolpho.sbaraglini@gmail.com, ivan@dc.ufscar.br

Dado o contexto, a proposta deste trabalho é apresentar as principais características, vantagens e desvantagens através de um estudo comparativo entre os frameworks MVC JavaServer Faces e VRaptor.

Na próxima seção são apresentados os conceitos, padrões e tecnologias nos quais esse estudo foi embasado. Nas seções 3 e 4 são apresentadas as principais características de cada tecnologia e na seção 5 são apresentados os critérios escolhidos e a comparação realizada nestas duas tecnologias. Posteriormente, a seção 6 apresenta os trabalhos relacionados e trabalhos futuros e, finalmente, as considerações finais deste trabalho são apresentadas na seção 7.

II. Padrões de Projetos e Frameworks

Padrões de projetos são técnicas, práticas e soluções reutilizáveis para resolver problemas recorrentes no contexto de desenvolvimento de software. Esses padrões não são específicos para determinada tecnologia ou empresa, mas sim especificados e aplicados pela comunidade de desenvolvimento como um todo.

Segundo Guerra (2012), padrões de projetos não representam novas soluções, mas soluções já existentes que foram implementadas com êxito em diferentes casos. Destaca também que, ao utilizar padrões, é possível absorver conhecimento e experiência de outros desenvolvedores que passaram por situações similares.

O MVC (Model-View-Controller) é um padrão de arquitetura de software que tem por objetivo separar a representação dos dados em três camadas: Modelo (Model), Visão (View) e Controlador (Controller), visando organizar o desenvolvimento. Foi descrito originalmente por Trygve Reenskaug, em 1979.

Reenskaug (1979) explica que o principal objetivo do MVC é fazer a intermediação entre o modelo mental humano e o modelo digital do computador. Idealmente esta solução deve trazer uma ilusão ao usuário de ver e manipular diretamente as informações do domínio.

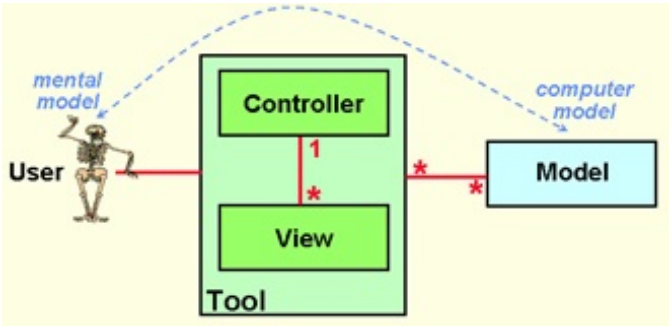

Figura 1. Padrão MVC (REENSKAUG, 1979)

Seguindo a definição de Reenskaug (1979), a camada modelo (Model) é o conhecimento, que pode ser representado por um único objeto ou uma estrutura de objetos, onde cada nó do modelo deve representar uma parte identificável do problema. A visão (View) é uma representação do modelo, podendo ou não destacar e inibir determinados atributos, agindo como um filtro de apresentação. É de responsabilidade da visão utilizar a mesma terminologia do modelo e saber exatamente a semântica dos atributos que o modelo representa. O controlador (Controller), por sua vez, é o elo entre o usuário e o sistema, garantindo a comunicação entre a camada de visão e a camada de modelo.

A característica de maior valor do padrão MVC é a separação entre os componentes de apresentação do resto da aplicação, ou seja, a separação entre a camada visual da aplicação do domínio e do negócio. (SILVEIRA et al., 2012)

O MVC foi amplamente adotado pela comunidade de desenvolvimento web. Os principais frameworks do mercado utilizam esse padrão, que pode ser dividido em dois modelos: Baseado em Componentes (Component-Based) e Baseado em Ações (Action-Based).

Os frameworks baseados em componentes trazem uma abstração maior no desenvolvimento Web, ocultando os detalhes e o fluxo do protocolo HTTP. O objetivo é trabalhar com componentes visuais ricos, reaproveitáveis e de alto nível, com um modelo de desenvolvimento parecido com o paradigma Desktop. Entretanto, o protocolo HTTP é orientado à requisições e não é formado por componentes e eventos. Para inibir esse conceito, frameworks baseados em componentes necessitam criar abstrações complexas sobre o modelo tradicional da web. (SILVEIRA at al., 2012)

De acordo com Almeida (2012), os frameworks MVC baseados em componentes (também conhecidos como MVC Pull) iniciam o processamento sempre na camada de visão, que é responsável por obter os dados do controlador, que por sua vez, executa as regras de negócio contidas nas classes da camada modelo. O JSF (JavaServer Faces) faz parte desta categoria de frameworks. Ao chegar uma requisição no servidor, o JSF inicia o processamento pela visão, que obtém os dados através dos Managed Beans, responsável por invocar as regras de negócio nas classes do Modelo e, por fim, disponibilizar os dados necessários para a renderização da visão ao usuário.

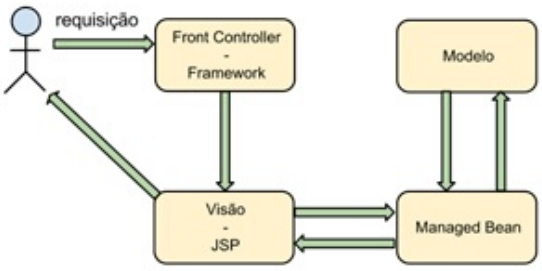

Figura 2. Fluxo da requisição no framework MVC baseado em componentes (ALMEIDA, 2012)

A abordagem MVC baseada em ações permite um controle mais fino da aplicação. É possível fazer otimizações para buscas (SEO – Search Engine Optimization), controlar o HTML final para aperfeiçoar a acessibilidade ou suportar diferentes navegadores e plataformas, além de maior flexibilidade em requisições AJAX e código Javascript. Utilizando esse modelo, é possível aproveitar melhor a Web e

outras ferramentas que trabalham com o protocolo HTTP, além de garantir disponibilidade e escalabilidade mais facilmente. (SILVEIRA at al., 2012)

O modelo de framework baseado em ações, que também é conhecido como MVC Push, inicia o processamento sempre na camada Controller, que é responsável por executar as regras de negócio da camada Model e devolver a resposta ao usuário através das regras de renderização da View. Ao chegar uma requisição no servidor, o framework identifica qual é o Controller responsável e delega a responsabilidade para ele. (ALMEIDA, 2012)

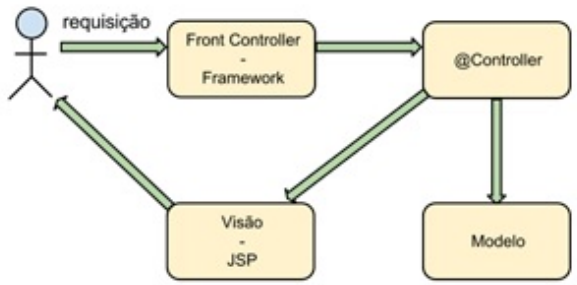

Figura 3. Fluxo da requisição no framework MVC baseado em ações (ALMEIDA, 2012)

III. JavaServer Faces (JSF)

JavaServer Faces (JSF) é a especificação de um framework MVC baseado em componentes presente na plataforma Java EE que facilita o desenvolvimento de aplicações Web. A primeira versão (1.0) foi lançada em 2004, mas somente em 2006, na versão 1.2, o framework foi incorporado ao Java EE como uma especificação oficial da plataforma.

A tecnologia JavaServer Faces representa um conjunto de APIs para manipular componentes de interface com o usuário, eventos, validações de entrada, acessibilidade e internacionalização. Sua arquitetura define claramente uma separação entre a lógica da aplicação e apresentação, permitindo com que cada membro da equipe de desenvolvimento de aplicações Web se concentre em sua parte do processo de desenvolvimento. (ORACLE, 2015).

O framework JSF possui diversos componentes na camada de visão que proporcionam praticidade ao desenvolvedor, abstraindo as dificuldades de implementação, como é o caso de requisições AJAX. Esses componentes são acoplados a um controlador, que por sua vez, pode executar lógicas de validação, conversão, redirecionamento de páginas, além de integração com outras camadas da aplicação, como o modelo (Model).

A) Ciclo de Vida

Segundo Silveira at. al (2012), o JSF possui um ciclo de vida complexo e poderoso para os componentes e requisições. Este ciclo de vida é dividido em fases que podem ser acompanhadas pelo desenvolvedor.

Cordeiro (2012) define as 6 fases do ciclo de vida do JSF da seguinte forma:

- *Restore View*: A primeira fase é responsável por construir ou restaurar a árvore de componentes correspondente à tela.
- *Apply Request Values*: Esta fase é onde o JSF obtém os valores informados pelo usuário e insere nos respectivos componentes.
- *Validate*: A terceira fase é responsável por converter os valores vindos da requisição para seus respectivos tipos para, posteriormente, validá-los de acordo com as restrições.
- *Update Model*: Uma vez que os dados já foram convertidos e validados, o JSF então aplica esses valores dentro do modelo.
- *Invoke Application*: É nesta fase que as lógicas e regras de negócio da aplicação são executadas.
- *Render Response*: A última fase do ciclo de vida é quando o JSF renderiza a resposta da requisição na tela do usuário.

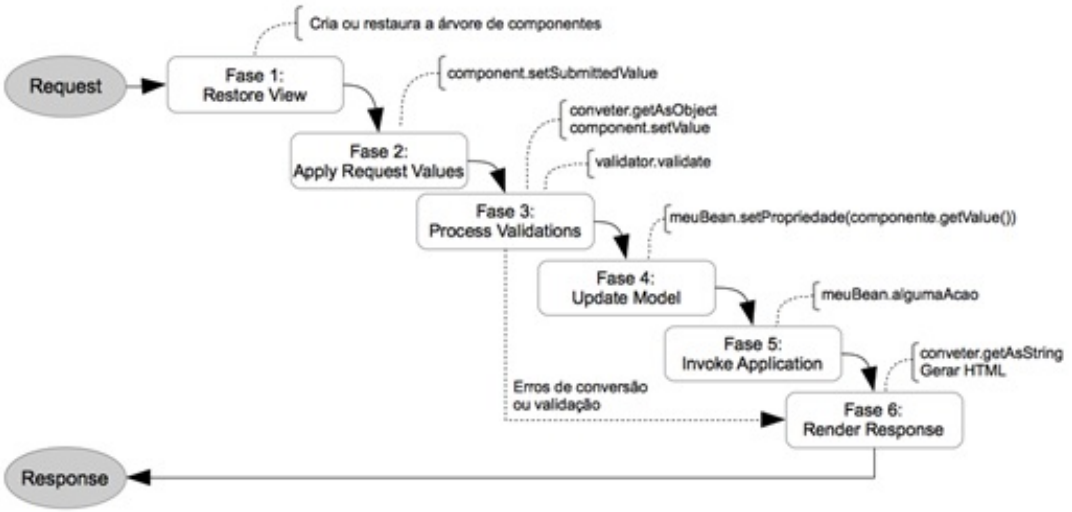

Figura 4. Ciclo de vida do JSF (CORDEIRO, 2012)

B) Tecnologias da View

No início da especificação do JavaServer Faces, uma das principais promessas era a possibilidade de escrever a interface de usuário em uma só linguagem e poder executá-la em um navegador comum (HTML), em um celular (WML) ou em Flash. Para fazer isso, bastaria substituir o componente que renderiza a resposta, chamado Render-kit. No entanto, essa característica pouco se acentuou no mercado, pois, na maioria das vezes, a maior necessidade das empresas era apenas gerar páginas HTML para navegadores comuns. (SILVEIRA at al., 2012)

De acordo com Coelho (2013), nas primeiras versões do JavaServer Faces, a principal tecnologia para exibir informações para o usuário era o JSP (JavaServer Pages). A partir da versão 2.0, porém, surgiu um novo template para gerar páginas dinâmicas denominado Facelets.

A característica de escrever páginas HTML utilizando os componentes do JSF permite que o framework cuide de alguns detalhes trabalhosos da Web, permitindo, por exemplo, que o valor de um campo seja automaticamente convertido para o formato correto ou que, ao clicar em um botão, um evento seja disparado para executar código Java em uma classe. (SILVEIRA at. al 2012)

Silveira at. al (2012) salienta que o código HTML gerado pelo JSF pode tornar qualquer otimização ou modificação complexa e custosa. Ao invés de uma abordagem mais livre no design e experiência de usuário, o front-end (onde ocorre a interação entre o usuário e os componentes da tela) é, no geral, adaptado para as limitações do framework.

C) Árvore de Componentes

Silveira at. al (2012) explica que, para que o JSF possa controlar seus componentes, eles devem ser organizados em uma estrutura de dados do tipo árvore. O estado de cada componente da árvore é armazenado e, a cada requisição, o framework se encarrega de atualizar as informações da árvore. De acordo com Cordeiro (2012), o JSF armazena na memória a árvore utilizada para renderizar determinada tela. Uma vez que o framework tenha em mãos a árvore de componentes, cada componente será comparado com os atributos da requisição. Com essas informações, o JSF inicia a fase de validação dos dados enviados para verificar se são compatíveis com os disponíveis, caso contrário, a requisição se torna inválida. Esse comportamento define o JavaServer Faces como um framework stateful.

D) Managed Beans

Em JSF, a comunicação entre o cliente e o servidor se dá através de classes Java denominadas Managed Beans, que podem ser consideradas como a camada controladora (Controller) da aplicação, seguindo o modelo MVC. Essas classes são responsáveis por responder aos eventos disparados pelos componentes da View. De acordo com Cordeiro (2012), um Managed Bean não passa de uma simples classe, cuja função é ter uma relação dos componentes contidos na tela.

Cordeiro (2012) afirma que para acessar um método no Managed Bean, alguns componentes do JSF definem actions (ações), que devem ser passadas através de EL (Expression Language), indicando qual método de qual Managed Bean deve ser executado.

```
@ManagedBean
public class AutomovelBean {

    public void salva() {
    }
}
```

Quadro 1. Exemplo de um Managed Bean (CORDEIRO, 2012)

```
<h:commandButton value="Salvar" action="#{automovelBean.salva}"/>
```

Quadro 2. Exemplo de um botão invocando uma ação no Managed Bean (CORDEIRO, 2012)

E) Injeção de Dependências

Para controlar o ciclo de vida dos objetos e diminuir o acoplamento entre as classes Java, o JSF utiliza um framework de injeção de dependências e inversão de controle. Silveira at. al (2012) explica que padrões como Inversão de Controle e Injeção de Dependências ajudam a manter os dois princípios básicos da orientação a objetos: alta coesão e baixo acoplamento. Com o objetivo de facilitar esse trabalho, sugiram diversos frameworks de injeção de dependências, como por exemplo o Spring, o PicoContainer e o Google Guice, além da especificação CDI (Context and Dependency Injection).

O lançamento da especificação CDI para a plataforma Java EE 6, cuja implementação de referência é o Weld, trouxe um poderoso container de injeção de dependências totalmente integrado ao JSF e às outras especificações da plataforma Java EE. Silveira at al. (2012) destaca que o CDI é uma ferramenta completa, trazendo novas funcionalidades avançadas de injeção que antes haviam sido pouco exploradas em outras especificações da plataforma, como o EJB (Enterprise JavaBeans).

F) Conversores

Segundo Cordeiro (2012), para que o JSF possa converter

as Strings recebidas na requisição em objetos do tipo correto, são definidos conversores. O framework oferece conversores para todos os tipos básicos do Java, como por exemplo Integer, Float, Double, Date e Boolean. Para alguns tipos de dados, o JSF já utiliza o conversor por padrão, porém quando a conversão é mais complexa, é possível utilizar o conversor explicitamente de acordo com a necessidade, como é o caso dos conversores de número e data.

```
<h:inputText value="#{managedBean.objeto.data}">
    <f:convertDateTime pattern="dd/MM/yyyy"/>
</h:inputText>

<h:inputText value="#{automovel.preco}">
    <f:convertNumber type="currency"/>
</h:inputText>
```

Quadro 3. Exemplo de conversores de datas e números. (CORDEIRO, 2012)

Além dos conversores padrões do JSF, é possível criar conversores customizados para cada tipo de dado. De acordo com Coelho (2013), a necessidade de criar conversores específicos é uma tarefa bastante trivial em projetos JSF.

G) Validadores

O JSF disponibiliza um conjunto de componentes para efetuar validações de dados inseridos pelo usuário. É possível validar, por exemplo, um campo obrigatório, o tamanho mínimo e máximo, formatações inválidas e até expressões regulares. Cordeiro (2012) destaca alguns validadores nativos, conforme o quadro 4.

```
<h:inputText value="#{auto.anoFabricacao}">
    <f:validateLongRange minimum="1950" maximum="2012"/>
</h:inputText>

<h:inputText value="#{auto.preco}">
    <f:validateRequired/>
</h:inputText>
```

Quadro 4. Exemplo de validadores nativos do JSF (CORDEIRO, 2012)

Assim como os conversores, JSF também permite que sejam criados validadores customizados, de acordo com a necessidade do domínio da aplicação. Além dos validadores contidos no framework, o JSF também pode ser integrado à especificação de validação do JavaEE, o Bean Validation. De acordo com Codeiro (2012), usando o Bean Validation, as regras de validação ficam especificadas no modelo através de anotações e não na apresentação ou controle do padrão MVC.

H) Bibliotecas Externas de Componentes

Além dos componentes pré-definidos pela especificação do JavaServer Faces, outras bibliotecas externas de componentes gráficos foram desenvolvidas através da iniciativa de terceiros. Um dos projetos mais conhecidos é o PrimeFaces, que encapsula o uso de frameworks front-end consolidados como o JQuery UI em componentes JSF, trazendo maior facilidade ao desenvolvedor.

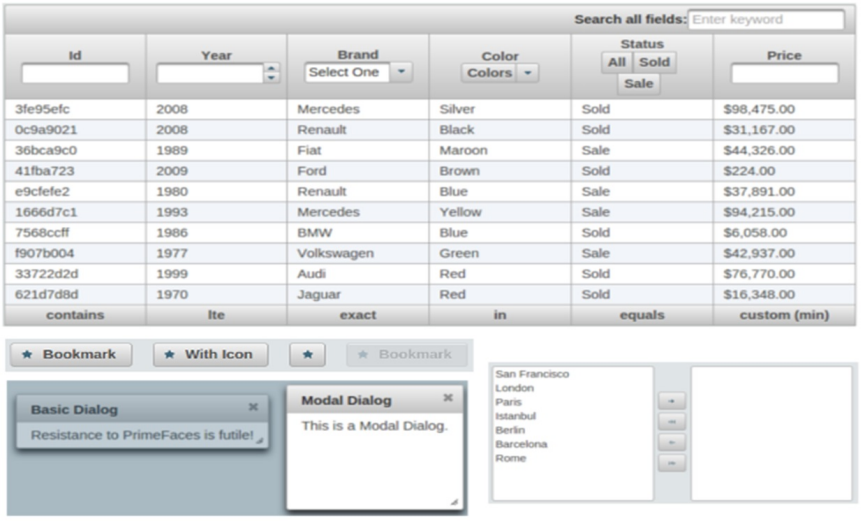

Figura 5. Exemplos de componentes JSF do PrimeFaces (PRIMEFACES, 2015)

IV. VRaptor

VRaptor é um framework MVC baseado em ações que facilita o desenvolvimento de aplicações Web. Segundo Cavalcanti (2013), surgiu em 2004 na Universidade de São Paulo, através de uma iniciativa dos irmãos Paulo Silveira e Guilherme Silveira, sendo mantido atualmente pela Caelum.

Ao contrário do JSF, o VRaptor não é uma especificação presente no Java EE, mas, por outro lado, possui integração com as outras especificações contidas na plataforma, como por exemplo o EJB ou o Bean Validation. Isso deve-se ao fato de que, a partir da versão 4, o VRaptor foi totalmente refatorado e passou a utilizar a especificação CDI para injeção de dependências. Isso possibilitou total integração com as demais especificações da plataforma Java EE.

Cavalcanti (2013) explica que o VRaptor é totalmente flexível, sendo um dos frameworks mais extensíveis da plataforma Java. Isso permite com que o desenvolvedor sobrescreva praticamente todos os seus comportamentos, sem a necessidade de configurações em XML. O framework foi modelado para desenvolver aplicações web e também APIs HTTP/REST para troca de informações entre sistemas.

A) Tecnologias da View

De acordo com Cavalcanti (2013), o VRaptor possui grande flexibilidade na escolha de tecnologias da camada de visão, devido ao baixo acoplamento com o controlador proporcionado pelo modelo MVC baseado em ações. É possível utilizar vários templates para geração de páginas dinâmicas, como por exemplo JavaServer Pages (JSP), Velocity, Freemarker, entre outros. Entretanto, os componentes visuais devem ser desenvolvidos manualmente, utilizando bibliotecas externas como o JQuery UI, Bootstrap, ExtJS, AngularJS e etc.

Silveira at. al (2012) destaca que a camada de visão não precisa ser, necessariamente, gerada dinamicamente no servidor através de uma ferramenta de template. No caso do cliente ser outra aplicação ou simplesmente uma página HTML estática, a comunicação pode ser feita através da serialização de objetos em determinado formato, como XML ou JSON. Este tipo de solução é comum ao disponibilizar serviços web para integração entre sistemas.

No quadro 5, é possível visualizar uma página utilizando JSP como template.

```
<body>
    Cliente:
    <br/>

    <label>ID</label>
    <input type="text" name="cliente.id" value="${cliente.id}"/>

    <br/>

    <label>Nome</label>
    <input type="text" name="cliente.nome" value="${cliente.nome}"/>
</body>
```

Quadro 5. Exemplo de um objeto representado em formato HTML, através de JSP

B) Controllers

Um Controller do VRaptor representa uma classe Java responsável por receber as requisições HTTP. Como o próprio nome já diz, é a camada controladora do modelo MVC baseado em ações, responsável por intermediar a comunicação entre a camada de visão e a camada modelo. É responsabilidade do Controller definir qual visão será utilizada na resposta da requisição HTTP.

Segundo Cavalcanti (2013), os Controllers do VRaptor trabalham com o conceito de convenção sobre configuração, onde o comportamento padrão do framework já vem pronto para ser utilizado sem precisar de nenhuma configuração, poupando algumas linhas de código do desenvolvedor. De acordo com a convenção adotada pelo VRaptor, todos os métodos públicos do Controller já estão disponíveis para receberem requisições HTTP. A visão padrão é uma JSP que fica localizada na pasta WEB-INF/jsp/{NOME_DO_CONTROLLER}/{NOME_DO_METODO}.jsp. Todas as JSPs ficam dentro de WEB-INF para que não sejam diretamente acessadas, somente através de métodos publicados nos Controllers.

C) REST

É cada vez mais raro encontrar uma aplicação isolada, sem nenhum tipo de integração com outros aplicativos e plataformas, portanto ao desenvolver uma aplicação, é importante pensar como ficará a integração dela com outras no futuro. Existem diversas opções para integração entre sistemas distribuídos, sendo que a abordagem REST sobre o protocolo HTTP têm ganhado destaque. (SILVEIRA at. al, 2012)

De acordo com Saudate (2013), REST é o acrônimo de REpresentational State Transfer (Transferência de Estado Representativo) e pode ser definido como um conjunto de boas práticas de uso do protocolo HTTP para expor web services, ou seja, serviços responsáveis por executar determinada lógica ou regra de negócio, geralmente utilizados para integração entre aplicações.

Saudate (2013) define algumas boas práticas do REST, como o uso adequado de métodos HTTP (GET para recuperar dados, POST para criar um recurso, PUT para atualizar um recurso e DELETE para remover um recurso), uso adequado de URLs e o uso de códigos de status padronizados para

representação de sucessos ou falhas.

Os Controllers do VRaptor permitem expor recursos utilizando o padrão REST de forma simples, conforme exemplificado no quadro 6.

```
@Controller
public class ClienteController {

    @Get("/cliente")
    public Cliente visualiza(Cliente cliente) {...}
    @Post("/cliente")
    public void adiciona(Cliente cliente) {...}
    @Put("/cliente")
    public void atualiza(Cliente cliente) {...}
    @Delete("/cliente")
    public void remove(Cliente cliente) {...}
}
```

Quadro 6. Exemplo de Controller REST (VRAPTOR, 2015)

D) Injeção de Dependências

Assim como o JavaServer Faces, o VRaptor também utiliza um framework de injeção de dependências para controlar o ciclo de vida de seus componentes. Até a versão 3, era possível escolher um container provedor, como Spring, Google Guice e PicoContainer. A partir da versão 4, o VRaptor passou a utilizar o CDI (Context and Dependency Injection), que é a especificação padrão de um framework de injeção de dependências para a plataforma Java EE.

Cavalcanti (2013) explica que o VRaptor tira proveito das funcionalidades do CDI para manter boas práticas de desenvolvimento, torna o framework totalmente extensível, além da possibilidade de integração com os recursos nativos do servidor de aplicação e com as demais especificações contidas na plataforma Java EE.

E) Conversores

Ao chegar uma requisição no servidor, ela deve passar por um ou mais conversores, responsáveis por transformar as Strings recebidas em objetos dos tipos esperados pelo Controller.

Cavalcanti (2013) explica que o VRaptor já possui um conjunto de conversores implementados para os tipos simples do Java, como números, datas, strings, enum e boolean, mas também é possível criar conversores customizados para tipos específicos. Todos os conversores são gerenciados pelo CDI, permitindo que sejam estendidos facilmente, de acordo com as necessidades da aplicação.

F) Validadores

O principal método de validação do VRaptor é baseado na especificação Bean Validation, presente na plataforma Java EE. Entretanto, é possível utilizar o validador próprio do framework. (VRAPTOR, 2015).

Segundo Cavalcanti (2013), o VRaptor possui uma API fluente de validações, dando suporte para internacionalização de forma simples. Permite ao desenvolvedor tomar determinada ação quando uma restrição de validação é violada, como por exemplo redirecionar para outro método de outro Controller ou simplesmente retornar um status de erro.

O quadro 7 apresenta um método com regra de redirecionamento no caso de uma restrição de validação.

```
public void salva(@Valid Livro livro) {
    validator.onErrorRedirectTo(this).formulario();
    estante.guarda(livro);
    result.redirectTo(this).lista();
}
```

Quadro 7. Exemplo de validação no Controller do VRaptor (CAVALCANTI, 2013)

G) Interceptadores

Um dos principais componentes que o VRaptor oferece é o interceptador, que é análogo ao clássico Filter da Servlet, porém integrado ao contexto de injeção de dependências.

De acordo com Cavalcanti (2013), existem tarefas ou funcionalidades que impactam boa parte da aplicação, como por exemplo o controle de acessos, controle de transações, logs de erros e etc. Geralmente, quase todas as funcionalidades da aplicação passam por esse tipo de controle, que deve ser implementado em um único ponto do código, facilitando a manutenibilidade. O interceptador permite que o desenvolvedor registre funções de callback antes e depois da execução de cada Controller.

H) Plugins

Ao atingir certo nível de maturidade e conquistar espaço na comunidade de desenvolvimento, diversos plugins surgiram através de iniciativas de terceiros. Cavalcanti (2013) explica que a arquitetura do VRaptor facilita a criação de componentes reusáveis que tem por objetivo resolver problemas em comum e podem ser facilmente adicionados em qualquer aplicação. Existe um catálogo de plugins disponíveis para o VRaptor, alguns criados pela Caelum e outros pela própria comunidade.

V. JSF e VRaptor

Dadas as características técnicas de cada framework, o

estudo comparativo entre JSF e VRaptor aborda critérios como curva de aprendizagem, tamanho da comunidade, aceitação pelo mercado, documentação e arquitetura REST.

A) Curva de Aprendizagem

Cada vez mais, aplicações devem ser desenvolvidas e entregues rapidamente em produção, agregando valor aos seus usuários. Para tanto, é fundamental que a tecnologia ou framework escolhido ofereça uma baixa curva de aprendizagem, garantindo que os desenvolvedores não percam muito tempo entendendo o funcionamento interno do framework e foquem no desenvolvimento das funcionalidades de negócio da aplicação.

O paradigma de desenvolvimento Web envolve diversos padrões e tecnologias, como HTTP, HTML, CSS, Javascript, JSON, XML e etc, além de exigir do desenvolvedor noções em redes, infraestrutura e ambientes de concorrência. Grande parte dos desenvolvedores de software, principalmente corporativos, ainda estão acostumados com o paradigma Desktop e, portanto, possuem dificuldade em lidar com todos esses termos e tecnologias do mundo Web.

Visando recepcionar melhor esses desenvolvedores, frameworks baseados em componentes como o JavaServer Faces foram concebidos. Abstraindo muitas das dificuldades do desenvolvimento Web, o JSF traz ao desenvolvedor um modelo de desenvolvimento análogo ao paradigma Desktop, diminuindo a curva de aprendizagem. No entanto, para que o JSF cuide de todos esses detalhes para o desenvolvedor, o framework possui uma estrutura de abstração complexa, que demora algum tempo para ser totalmente absorvida. Na parte da view, o JSF dispõe de componentes orientados a eventos que aumentam a produtividade do desenvolvimento, porém o uso indiscriminado pode acarretar problemas difíceis de rastrear e resolver, principalmente relacionados à performance da aplicação. A utilização de bibliotecas externas de componentes, como por exemplo o PrimeFaces, permite ao desenvolvedor criar interfaces ricas para o usuário mesmo sem o domínio das tecnologias de front-end, como HTML, CSS e Javascript.

Por outro lado, para desenvolvedores que já estão acostumados com o paradigma Web, entender o ciclo de vida e todos os componentes do JSF pode levar um tempo considerável. A grande maioria dos frameworks MVC para Web são baseados em ações, assim como o VRaptor, e possuem uma estrutura mais simples e mais próxima dos padrões e tecnologias presentes na Web. Portanto, para esses desenvolvedores a adaptação ao VRaptor torna-se mais confortável, diminuindo a curva de aprendizagem. Ao contrário do JSF, o VRaptor não disponibiliza um conjunto de componentes gráficos para construir páginas HTML, exigindo que o desenvolvedor domine as tecnologias de front-end. Entretanto, vale ressaltar que o framework oferece maior flexibilidade para o desenvolvimento da view, permitindo que o desenvolvedor escolha quaisquer bibliotecas de CSS e Javascript que desejar.

B) Tamanho da Comunidade

A comunidade de desenvolvedores é algo primordial para manter a evolução e a qualidade de qualquer tecnologia ou framework. O feedback dos desenvolvedores permite que o framework possa evoluir alinhado às expectativas de quem o utiliza, priorizando os itens mais importantes destacados na comunidade.

O JSF, por se tratar da única especificação de um framework MVC contida na plataforma Java EE, possui uma vasta comunidade de desenvolvedores e é amplamente adotado em projetos Java. O fórum de discussões internacional Stack Overflow possui aproximadamente 75.000 postagens com referências ao JSF. O mesmo fórum na versão em português possui pouco mais de 260 perguntas, enquanto o GUJ (Grupo de Usuários Java), um dos principais fóruns brasileiros de discussões, possui uma média de 33.000 resultados que referenciam JSF em uma pesquisa interna. Outros fóruns brasileiros, com o Javafree e o DevMedia possuem aproximadamente 7.800 e 590 ocorrências, respectivamente.

O VRaptor, por sua vez, é um projeto brasileiro e não possui grande expressão no mercado exterior. Em uma consulta no Stack Overflow, somente cerca de 100 ocorrências de postagens que referenciam o VRaptor podem ser encontradas, enquanto na versão em português do fórum, 29 questões. Já nos fóruns brasileiros GUJ, Javafree e DevMedia podem ser encontrados por volta de 18.000, 900 e 13 ocorrências, respectivamente.

O gráfico 1 representa a expressão dos dois frameworks em números de postagens em fóruns de discussão.

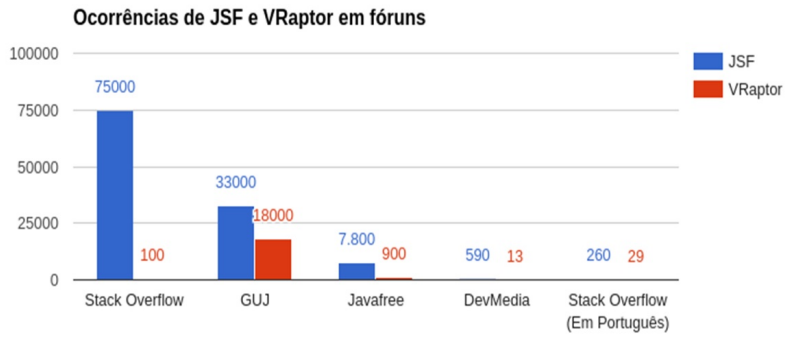

Gráfico 1. Ocorrências de JSF e VRaptor em fóruns de desenvolvimento

C) Aceitação pelo Mercado

A aceitação pelo mercado é um fator decisivo para a adoção e continuidade de qualquer framework. Empresas e desenvolvedores sempre buscam tecnologias consolidadas, confiáveis, seguras e que trazem produtividade e qualidade para o desenvolvimento de software. Esses quesitos sem dúvida são primordiais para que qualquer framework se consolide no mercado.

O JSF possui todas essas qualificações e está consolidado no mercado de software. Por se tratar de uma especificação dentro da plataforma Java EE, o JSF ganhou adeptos rapidamente, desde as primeiras versões. De acordo com Silveira at al. (2012), para uma empresa é bastante vantajoso trabalhar com especificações, tendo em vista que isso se torna uma garantia da continuidade da tecnologia. No entanto, este não deve ser o único parâmetro para a escolha de um framework. JavaServer Faces, mesmo com tantas vantagens, não possui unanimidade entre os desenvolvedores Java. Isso deve-se ao fato de que o modelo baseado em componentes traz algumas dificuldades para o paradigma Web, dentre elas, a falta de controle sobre o HTML gerado pelo framework.

O VRaptor é um framework brasileiro que sempre possuiu reconhecimento no mercado nacional. Segundo Cavalcanti (2013), desde a primeira versão em 2004, o VRaptor passou por duas grandes refatorações, sempre buscando aperfeiçoar o framework e atender a demanda do mercado. A última grande refatoração, que ocorreu em meados de 2013, tornou o framework totalmente baseado em CDI. Esta decisão estratégica tornou o framework uma grande alternativa para quem não se sente confortável em utilizar o JSF e ao mesmo tempo não quer abrir mão dos outros recursos da plataforma Java EE, tendo em vista que o VRaptor se integra muito bem com eles.

D) Documentação

Para que cada vez mais desenvolvedores passem a utilizar determinada tecnologia ou framework, é necessário que a documentação disponibilizada seja a mais clara e completa possível. Através de uma boa documentação, o desenvolvedor consegue entender com mais profundidade o comportamento do framework bem como suas características, além de resolver possíveis problemas sem a necessidade de solicitar ajuda em fóruns de discussões, o que leva tempo antes da obtenção de uma resposta. Clareza na documentação é fundamental para não perder novos desenvolvedores que estão em processo de aprendizagem e descoberta do framework.

A documentação do JSF pode ser encontrada no site oficial da Oracle, da JSR (Java Specification Request) 314 ou no site oficial da especificação. É possível obter informações gerais do framework, além de instruções de utilização e a visão de toda a hierarquia de pacotes, classes e métodos da API, inclusive de versões mais antigas. Por se tratar de uma especificação muito utilizada pela comunidade, alternativamente é possível encontrar documentações não oficiais através de blogs, fóruns, livros e artigos.

O VRaptor possui toda a documentação oficial centralizada em seu próprio site. É possível receber instruções de uso, exemplos de implementações, tutoriais para migração de versões antigas, além das metas de entrega de funcionalidades e de toda hierarquia de pacotes, classes e métodos da API. Um diferencial é a documentação oficial traduzida em português, além do inglês. Assim como o JSF, o VRaptor também possui documentações não oficiais através de blogs, fóruns, livros e artigos, porém em menor quantidade.

E) Arquitetura REST

Nos últimos anos, arquiteturas de software baseadas em front-end e back-end totalmente separados têm ganhado expressão na comunidade de desenvolvimento. Nestas arquiteturas, geralmente o back-end é constituído por uma API de serviços REST que contém todas as regras de negócio da aplicação, enquanto o front-end fica incumbido de executar lógicas de renderização e apresentar de forma amigável para o usuário os dados enviados e recebidos do back-end. O baixo acoplamento oferecido por estas arquiteturas permitem que o back-end seja escrito uma única vez e seja consumido de vários front-ends, como aplicações web e desktop, dispositivos móveis ou até mesmo a imensidão de dispositivos que surgirão com o movimento da Internet das Coisas (IoT). Outra grande vantagem deste modelo é o aumento da manutenibilidade do código, tendo em vista que times especializados podem dar manutenção em partes ou aplicações específicas do projeto, sem impactar no restante.

O JavaServer Faces não oferece suporte para a criação de serviços REST, portanto não atende estas arquiteturas de software. Uma boa alternativa neste caso é a especificação oficial da plataforma Java EE para a construção de serviços REST, o JAX-RS, cuja implementação de referência é o Jersey.

O VRaptor, por outro lado, possui suporte para a criação de serviços REST nativamente através de seus Controllers, atendendo muito bem estes tipos de arquiteturas de software.

VI. Trabalhos Relacionados e Trabalhos Futuros

No trabalho de Murgo e Foschini (2014), foram abordadas as características dos frameworks JSF e JavaFX direcionadas para o desenvolvimento de interfaces ricas para o usuário. O JavaFX é apontado como uma tecnologia promissora, no entanto, necessita de um plugin instalado no browser para ser executado. Já o JSF é tido como a tecnologia mais consolidada dentro do desenvolvimento Java para Web e, portanto, é mais recomendado para uma aplicação completa que será utilizada em produção, oferecendo mais segurança para o projeto.

Turini (2014) apresenta as novidades do framework VRaptor 4, enfatizando as vantagens adquiridas com o uso da convenção sobre configuração, permitindo que o desenvolvedor dê foco no desenvolvimento das funcionalidades da aplicação e não perca tempo com configurações complexas. Também são abordadas as formas de extensão dos componentes do framework através de especializações do CDI, além de integração com especificações da plataforma Java EE, como a JPA (Java Persistence API).

O JCP (Java Community Process), comitê responsável pela definição de novas especificações para a plataforma Java, anunciou em 2014 a criação da JSR (Java Specification Request) 371, que é a especificação de um framework MVC baseado em ações para a plataforma Java EE. Geralmente, uma JSR é formada por desenvolvedores especialistas conhecidos pela comunidade Java juntamente com profissionais experientes de grandes empresas que colaboram com o comitê, como Oracle, IBM e RedHat. Dentre os especialistas da JSR 371, estão alguns desenvolvedores da Caelum, empresa responsável por manter o projeto VRaptor. Nesse sentido, é possível que o VRaptor se torne uma implementação da especificação MVC 1.0 nas próximas versões. Como possibilidade de um trabalho futuro, poderiam ser exploradas, através de um estudo comparativo, as características do MVC 1.0 e do JSF, ambos como especificações oficiais da plataforma Java EE.

VII. Considerações Finais

Foram apresentadas as principais características técnicas das tecnologias JavaServer Faces e VRaptor, além de um estudo comparativo abordando critérios importantes sobre cada framework. Ambos estão consolidados no mercado e atendem, na maioria dos casos, todas as necessidades de uma aplicação Web, fornecendo um conjunto de recursos poderosos e abstraindo implementações trabalhosas do desenvolvedor, trazendo produtividade e manutenibilidade para o código.

O JSF, por se tratar de uma especificação padrão contida na plataforma Java EE, é um dos frameworks mais utilizados em Java e traz segurança ao desenvolvedor. JavaServer Faces utiliza a abordagem MVC baseada em componentes, portanto abstrai os detalhes de infraestrutura da Web criando uma camada, onde cada requisição recebida no servidor passa por um complexo ciclo de vida formado por 6 fases. Um dos principais recursos do JSF é sua biblioteca de componentes visuais orientadas a eventos, que permite uma interação fluente com o servidor, trazendo um paradigma de desenvolvimento análogo ao Desktop. Visando aproveitar melhor esses recursos, diversas extensões de componentes foram concebidas por iniciativa de terceiros, como por exemplo o PrimeFaces, que permite ao desenvolvedor construir interfaces ricas para o usuário sem precisar dominar as tecnologias de front-end, como CSS e Javascript.

O VRaptor, por outro lado, utiliza a abordagem MVC baseada em ações, como a maioria dos frameworks MVC do mercado. Essa abordagem permite que o desenvolvedor tenha mais controle sobre o HTML gerado, permitindo otimizações e customizações difíceis de se obter com JSF, por exemplo. Entretanto, o VRaptor não abstrai tanto os detalhes do protocolo HTTP e da Web, exigindo que o desenvolvedor domine os padrões e tecnologias deste paradigma. Apesar de não ser uma especificação oficial do Java EE, o VRaptor têm se esforçado para conquistar os desenvolvedores da plataforma, tanto que na versão 4 o framework foi totalmente refatorado para utilizar o CDI como container de injeção de dependências, permitindo maior integração com os recursos do servidor e com as outras especificações do Java EE. Embora tenha maior expressão no mercado nacional, o VRaptor também tem ganhado foco no exterior, principalmente depois que alguns desenvolvedores do projeto entraram no grupo de especialistas da JSR 371, que define um novo framework MVC baseado em ações para a plataforma Java EE. Uma das grandes vantagens do VRaptor é o desacoplamento entre a view e o controller, garantindo que o desenvolvedor tenha total liberdade de escolher as tecnologias de front-end, além da possibilidade de publicar recursos REST para serem consumidos de qualquer dispositivo.

Escolher entre um ou outro é uma decisão cautelosa que deve levar em consideração muitos fatores, como maturidade do time, prazo do projeto, usabilidade e experiência do usuário, dispositivos que acessarão a aplicação e integrações com outros softwares e parceiros. O JSF, de acordo com as características apresentadas, é uma escolha mais confortável para times de desenvolvedores que não possuem tanta experiência com os padrões e tecnologias do paradigma Web ou que não possuem um profissional focado em design e experiência do usuário. O VRaptor, por sua vez, é uma boa opção para aplicações que necessitam de maior liberdade para a view, exigindo que o time tenha experiência com o protocolo HTTP e os demais detalhes de infraestrutura presentes no paradigma Web.

Referências

ALMEIDA, A., Entenda os MVCs e os frameworks Action e Component Based. Disponível em: <http://blog.caelum.com.br/entenda-os-mvcs-e-os-frameworks-action-e-component-based/>. Acessado em: 17 de Setembro de 2015.

CAVALCANTI, C., VRaptor: Desenvolvimento ágil para web com Java. Ed. Casa do Código, 2013.

COELHO, H., JSF Eficaz: As melhores práticas para o desenvolvedor web Java. Ed. Casa do Código, 2013.

CORDEIRO, G., Aplicações Java para a web com JSF e JPA. Ed. Casa do Código, 2012.

Fórum de discussões DevMedia. Disponível em: <http://www.devmedia.com.br/forum>. Acessado em: 17 de Setembro de 2015.

Fórum de discussões GUJ. Disponível em: <http://www.guj.com.br>. Acessado em: 17 de Setembro de 2015.

Fórum de discussões Javafree. Disponível em: <http://javafree.uol.com.br>. Acessado em: 17 de Setembro de 2015.

Fórum de discussões StackOverflow. Disponível em: <http://stackoverflow.com>. Acessado em: 17 de Setembro de 2015.

Fórum de discussões StackOverflow em Português. Disponível em: <http://pt.stackoverflow.com>. Acessado em: 17 de Setembro de 2015.

GUERRA, E., Design Patterns com Java: Projeto orientado a objetos guiado por padrões. Ed. Casa do Código, 2012.

MURGO, L.; FOSCHINI, I., Análise das Tecnologias JSF2/PrimeFaces e JavaFX para a Criação de Interfaces

Ricas para Internet. Disponível em: <http://revistatis.dc.ufscar.br/index.php/revista/article/view/90>. Acessado em: 17 de Setembro de 2015.

ORACLE, JavaServer Faces Technology Overview. Disponível em: <http://www.oracle.com/technetwork/java/javaee/overview-140548.html>. Acessado em: 17 de Setembro de 2015.

Página oficial da JSR 371: Model-View-Controller (MVC 1.0) Specification. Disponível em: <https://www.jcp.org/en/jsr/detail?id=371>. Acessado em: 17 de Setembro de 2015.

PRIMEFACES. Página oficial da biblioteca de componentes PrimeFaces. Disponível em: <http://www.primefaces.org/showcase/>. Acessado em: 17 de Setembro de 2015.

REENSKAUG, T., Models-Views-Controllers. Disponível em: <https://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>. Acessado em: 17 de Setembro de 2015.

SAUDATE, A., REST: Construa API's inteligentes de maneira simples. Ed. Casa do Código, 2013.

SILVEIRA, P.; SILVEIRA, G.; LOPES, S.; MOREIRA, G.; STEPPAT, N.; KUNG, F., Introdução à Arquitetura e Design de Software: Uma Visão Sobre a Plataforma Java. Ed. Elsevier, 2012.

TURINI R., Java na Web com VRaptor 4. Disponível em: <http://www.infoq.com/br/articles/vraptor4-java-web>. Acessado em: 17 de Setembro de 2015.

VRAPTOR, Documentação oficial do framework VRaptor. Disponível em: <http://www.vraptor.org/pt/docs>. Acessado em: 17 de Setembro de 2015.