**Análise comparativa entre dois *Frameworks* MVC *web* para plataforma *JAVA*:**

***Spring* MVC e *VRaptor***

**Cezar Toshiaki Nakase**

[cezar.nk@gmail.com](mailto:cezar.nk@gmail.com)

**Prof. MSc. Claudinei Di Nuno**

[professorclaudinei@uol.com.br](mailto:professorclaudinei@uol.com.br)

Curso de Pós-Graduação *Lato Sensu*em Desenvolvimento Orientado a Objetos com Java

UNESA–Universidade Estácio de Sá

# Resumo

O objetivo deste artigo é de apresentar o comparativo de dois *frameworks Java* voltado ao desenvolvimento web, no qual serão o *Spring MVC* e *VRaptor 4.* Com intuito de auxiliar o desenvolvedor na escolha do *framework*, com base na estrutura e necessidade do seu projeto. Neste artigo será apresentado os requisitos dos frameworks, o fluxo do processamento da informação, a importância da utilização da arquitetura MVC e a divisão do código em camadas, a utilização da injeção de dependências, a inclusão de validadores e os principais plug-ins para facilitar ainda mais o desenvolvimento. Além disso, serão abordados os pontos fortes e fracos dos *frameworks*, a dificuldade em relação à curva de aprendizagem, analise e continuidade da plataforma e suporte do mantenedor. Com base nas informações apresentadas, o artigo servirá como disseminador dos *frameworks Java* para web e de material de apoio para futuras pesquisas e desenvolvimento.

- Neste cenário, este trabalho apresenta um estudo de caso sobre padrões de

desenvolvimento de software, abrangendo, suas características e funcionalidades.

Apresenta-se um protótipo desenvolvido utilizando os *frameworks* *Spring* MVC e *VRaptor*.

Palavras-chave: ***framework*, comparativo, padrão MVC, *Vraptor* e *Spring MVC*.**

# Introdução

Quando iniciamos o desenvolvimento de um projeto web na linguagem *Java*, uma das questões que muitos programadores têm de defini-la, é com qual ferramenta será utilizada para o desenvolvimento do projeto?

E desenvolver sistemas em *Java* para web, antes das criações dos *framework*s, que em resumo é um conjunto de bibliotecas que reúne inúmeras funcionalidades que ficam à disposição do programador, aumentando sua produtividade; era uma tarefa trabalhosa, pois mesmo o *Java* possuindo os *Servlets* e o JSP para auxiliar nos serviços específicos web, as requisições e tratamentos eram feitos de forma manual, onde cada programador criava sua metodologia de criação, bibliotecas e ferramentas com códigos massivos para sanar suas necessidades, assim não havendo uma padronização no desenvolvimento web com *Java*.

Atualmente, temos a disposição vários *frameworks* para facilitar a produtividade e a padronização do código. Porém, devemos analisar se vale à pena utilizar? Como escolher? Qual ganho e vantagem terão ao escolher uma?

Baseado nesses questionamentos, por boa prática, devemos seguir alguns critérios para a escolha do *framework*. Por exemplo, a sua estrutura, pois verificamos se ele poderá atender as necessidades; o quão reconhecido ele é, pois assim podemos ter mais informações através de comunidades e fóruns a respeito de novas ideias, nova funcionalidade e a qualidade do *framework*; segurança, verificar a capacidade de garantir o funcionamento do sistema, o gerenciamento de riscos e redução de vulnerabilidade; e a documentação, pois sendo bem formulada serão mais fáceis o seu entendimento e a sua utilização.

Seguindo as boas práticas citado anteriormente, este artigo vai apresentar os *frameworks Spring MVC*e o *VRaptor4*.

E a relevância da escolha dos frameworks encontra-se da seguinte forma: *Spring MVC –* atualmente, um dos *frameworks* de *Java* para web mais utilizada, além de ser mais completo em questões de bibliotecas e ferramentas e com muitos módulos que facilitam na configuração do projeto. *VRaptor* 4 – um *framework* desenvolvido por brasileiros, levando o conceito de praticidade na criação do projeto por seguir a convenção de *Convention over Configuration,* conceito de redução de arquivos adicionais de configuração, facilitando e agilizando o desenvolvimento do projeto e a manutenção do código; e por possuir total integração com *Java EE.*

Por conseguinte, o objetivo final em servir como um material de pesquisa para desenvolvedores que buscam alternativas de frameworks *Java* para o desenvolvimento do seu projeto web, apresentando comparativo de ambos os frameworks, visando os prós e contras. Tornando mais claro a diferença em seus pontos específicos, facilitando a visualização para escolha do framework de acordo com o projeto.

- Apresentar um estudo de caso sobre padrões de desenvolvimento de software, e

por fim desenvolver um protótipo utilizando o padrão arquitetural MVC.

Apresentar um estudo de caso e analisar funções e características de

padrões de desenvolvimento de software.

b) Desenvolver um protótipo de registros para controle de contratos de uma

empresa utilizando o padrão de projeto arquitetural MVC (*Model, View,*

*Control*).

1. **Fundamentação Teórica**

**2.1 Um pouco da história do Java no mundo web**

Desde o avanço da internet, em meados de 1996, quando o Java iniciou seus primeiros passos ao mundo da web através dos recursos do *Servlets*: Oracle (2013) “uma classe de linguagem de programação Java usada para estender os recursos de servidores que hospedam aplicativos acessados ​​por meio de um modelo de programação de solicitação-resposta”; foi um grande avanço, dando vida à comunicação da linguagem Java, Banco de Dados e página web. Porém, no seu início, não havia uma organização definida entre os códigos, se misturavam códigos Java com os de apresentação (*HTML, CSS e JavaScript*) em um único arquivo. Com o surgimento do JSP (*Java Server Page*), onde o intuito era de organizar e definir as especificações que uma página web deveria seguir. IBM - Noel J. Bergman e Abhinav Chopra  (2001) “O JSP combina HTML e XML com *servlet* Java TM (extensão de aplicativo de servidor) e tecnologias *JavaBeans* para criar um ambiente altamente produtivo para desenvolvimento e implantação de sites confiáveis, interativos e independentes de plataforma de alto desempenho”. Resultando em melhora na questão da separação dos códigos.

Porém, mesmo com essas tecnologias criadas, o desenvolvimento web com Java, ainda não seguia uma estrutura organizada, e a necessidade de padronização ou uma arquitetura eficiente era inevitável.

- Segundo Massoni et al. (2003), o desenvolvimento de software tem se tornado

mais complexo ao longo dos anos. As exigências por parte dos clientes são cada vez

maiores, principalmente em termos de produtividade, qualidade de software e prazos.

O surgimento de novas tecnologias e a necessidade da realização de mudanças

nos softwares desenvolvidos para atender às exigências dos clientes também dificulta a

tarefa de desenvolver software com qualidade.

Acompanhar as mudanças tecnológicas e atender às necessidades de mudança

pode ser uma tarefa bastante complicada se o software não estiver preparado para

suportar tais alterações.

**2.2 Padrões de projeto para ajudar o desenvolvedor**

-Segundo Laborde (2013), a utilização de padrões possibilita várias vantagens no

desenvolvimento de software, dentre elas, pode ser citada à diminuição do processo de

aprendizagem de um novo engenheiro de software dentro de um projeto, a reutilização

e customização em projetos de desenvolvimento.

O uso de padrões auxilia no desenvolvimento de um projeto com bom nível de

coesão e reusabilidade, o que facilita o processo de manutenção do software.

(FERREIRA, 2013).

Segundo Sommerville (2011), padrões arquiteturais permitem a construção de

uma arquitetura aderente a certas propriedades.

- *O que são padrões de projeto*

Segundo Negrão (2013), padrões de projetos são combinações de classes e

algoritmos associados que cumprem com propósitos comuns de projetos. São

normalmente soluções consagradas que se baseiam nas estruturas da orientação a

objeto em sua melhor forma.

A partir das citações acima fica claro que padrões de projeto são soluções para

problemas que alguém algum dia já teve e resolveu utilizando um modelo que foi

documentado e que pode ser adaptado totalmente ou de acordo com a necessidade de

cada aplicação.

Visando solucionar conflitos referentes à padronização no desenvolvimento de um projeto, foram criados conceitos onde reunia as melhores práticas formalizadas por programadores para solução de um problema em comum, técnicas adotadas dentro de um contexto para uma melhor organização, no qual segundo [Christopher Alexander](https://pt.wikipedia.org/wiki/Christopher_Alexander) (*A Times Way of Building,* 1979) devia ter as seguintes características: “Encapsulamento, Generalidade, Equilíbrio, Abstração, Abertura e Combinatoriedade”. Padrões de projetos não são específicos a uma empresa ou tecnologia, são soluções reutilizáveis aplicados pela comunidade de desenvolvimento como um todo. Onde define segundo [Christopher Alexander](https://pt.wikipedia.org/wiki/Christopher_Alexander),“ cada padrão descreve um problema no nosso ambiente e o núcleo da sua solução, de tal forma que você possa usar esta solução mais de um milhão de vezes, sem nunca fazê-lo da mesma maneira”.

Resumindo a sua importância na utilização de um padrão, o ganho na qualidade do código, a reutilização de soluções para um determinado problema e contribuição de especialistas e desenvolvedores para auxilio de manutenção, pois o padrão representa o conhecimento entre os envolvidos.

Diante desse cenário, o padrão de arquitetura MVC (*Model-View-Controller*) se destaca nas mais diversas aplicações, e principalmente em aplicações web, por sua organização, onde são divididas em camadas bem estruturadas em relação aos dados, regras de negócios e interface, além de ser um modelo utilizado por muitos *frameworks*.

O MVC criado em 1979 por Trygve Reenskaug, no qual ele justifica sua criação da seguinte forma: “Eu criei o padrão *Model-View-Controller* como uma solução óbvia para o problema geral de dar aos usuários controle sobre suas informações como visto de múltiplas perspectivas”, transcrevendo o modelo mental humano e o modelo digital do computador.

*Model (*Modelo*):* é a representação da estrutura de dados, camada responsável pela manipulação, consulta e persistência das informações no banco de dados.

*View* (Visão): é a camada responsável pela interface apresentado ao usuário, no qual deve gerir requisições e respostas, através de componentes visuais, por exemplo, formulários, tabelas, menus e botões para entrada e saída de dados e que a visão reflita o estado do modelo.

*Controller* (Controladora): responsável pela comunicação ou fluxo de informação entre a camada do modelo e a camada de visão, é nele onde dizemos quais regras de negócio o sistema deve seguir, quais operações devem ser executadas.

Assim concluindo, o padrão MVC nos proporciona as seguintes vantagens:

- O MVC por trabalhar em multi-camadas, tende a facilitar o gerenciamento do projeto e código de uma forma mais clara, pois mantém explícito o que cada camada deve executar, tornando o código mais limpo;

- É possível o desenvolvimento das camadas do projeto em paralelo ou escalável, além de maior integração da equipe em questão da divisão de tarefas;

- Possibilidade do reaproveitamento ou re-usabilidade do código de uma forma mais fácil, podendo incluir bibliotecas ou adicionando interfaces no projeto;

- Redução na dificuldade na manutenção do software, pois são realizadas as correções e alterações separadamente, não afetando outras camadas do sistema;

- Diversidade de frameworks ou tecnologias que estão utilizando essa metodologia de padrão de projeto.

Para Ladd(2006, p. 52), *Controllers* são responsáveis pelo processamento das requisições HTTP, pelo execução das regras de negócio, pela composição dos objetos de resposta e por passar o controle de volta ao fluxo de tratamento principal. O *Controller* não trata a rederização da visão, focando no tratamento de requisições e respostas e delegando ações à camada de serviço.

**2.3 *Spring*MVC, popular e com muitos recursos**

Após uma breve explicação sobre a história do Java para web, padrões de projetos e modelo MVC, o assunto abordado será sobre o framework *Spring,* mais especificamente o *Spring* MVC, no qual se refere justamente ao módulo web.

**Um pouco da história, estrutura e funcionamento do Spring**

Uma resposta a complexidade das primeiras especificações do J2EE (Java2 Platform Enterprise Edition), surgiu em 2003, criado por Rod Johnson, descrito em seu livro “Expert One-To-One J2EE Design and Development”.

*Spring* é organizado de forma modular, permitindo que você se preocupe apenas com os módulos necessários. Atualmente estão disponíveis 21 módulos, fornecendo tudo o que um desenvolvedor pode precisar para o uso no desenvolvimento de um projeto. WEISSMANN Henrique Lobo (2014, pg. 27) Ressalta a importância dos módulos “É importante termos esta visão panorâmica dos módulos que compõem o framework para que fique claro o quão abrangente ele é: basicamente o *Spring* abrange todas as necessidades de uma aplicação corporativa”.

*Spring* é baseado na estrutura de POJOS (*Plain Old Java Object*), um objeto Java não limitado por nenhuma restrição, sem a necessidade de implementar ou estender classes pré-especificadas na estrutura. Segue o padrão de IOC (*Inversion of Control –* Inversão de Controle), um fluxo do controle do sistema é invertido, permitindo que indique outro elemento o controle do método dizendo quando deve ser executado. Para KAYAL, Dhrubojyoti (2008, pg. 23) “O contêiner de Inversão de Controle de Mola (IOC) é o coração de todo o framework. Isso ajuda a unir as diferentes partes do aplicativo,formando assim uma arquitetura coerente.” Um dos métodos utilizados pelo IOC é a injeção de dependência, no qual separa um objeto de suas dependências, deixando o foco da classe apenas nos recursos para realizar as tarefas que precisa, e possamos decidir quais dependências serão injetadas durante o tempo de execução. Uma das características mais conhecidas quando programamos com *Spring*.

Descrito por HEMRAJANI, Anil da seguinte forma “O Spring Web MVC Framework é uma estrutura robusta, flexível e bem projetada para aplicativos da web em rápido desenvolvimento usando o padrão de design MVC, usando este módulo Spring são semelhantes àqueles que você obtém do resto do Spring Framework”.

Por ser um dos mais completos frameworks com diversos componentes de auxílio, além de ser um dos mais utilizado para desenvolvimento web com Java.   
É verdadeiramente abrangente e expansivo que é mais do que capaz de assumir qualquer tarefa ou projeto em potencial no qual você deseja trabalhar.

Devido à modularidade da própria ferramenta, isso permite que você escreva códigos muito limpos e acessíveis. Há uma enorme quantidade de documentação excelente e uma comunidade próspera que o ajudará se você tiver dúvidas ou preocupações sobre como fazer certas coisas ou como certas coisas funcionam ou qualquer coisa desse tipo.

Possui as funcionalidades necessárias para processar as requisições HTTP, gerenciar os componentes para o processamento de dados além de processar e apresentar a resposta da requisição, seguindo o padrão MVC.

Além de possuir uma documentação robusta, uma comunidade ativa onde possui uma página exclusiva no site do *StackOverflow,* apenas com perguntas relativas ao *Spring* MVC e um amplo pacote de ferramentas abrangente para qualquer tipo de projeto que você possa ter.

- O Spring Web MVC Framework é um framework robusto, flexível e bem projetado para o desenvolvimento rápido de aplicações Web usando o padrão de projeto MVC (HEMRAJANI, 2006, p.128). Ele é parte do Spring Framework, que provê diversas soluções para persistência de dados, programação orientada a aspectos, entre outras.

- O Spring MVC é projetado ao redor do DispatcherServlet, que é um Servlet que distribui as requisições para as classes configuradas para trata-las, com visão, localização e temas configuráveis (Spring Reference).



Figura 4 - Arquitetura do Spring MVC (Alto Nível)

Fonte: Spring Reference (2007)

**2.4 *VRaptor* para agilizar o desenvolvimento**

Diferente de outros frameworks, *VRaptor*, criado em 2004, pelos brasileiros Paulo Silveira e Guilherme Silveira, apresenta-se como uma alternativa eficiente e com a proposta de sua simplicidade e por trazer a ideologia de ser rápido e de fácil aplicação. E desde esse período, houve novas versões e melhorias, acompanhando as melhores práticas de desenvolvimento no mercado e desempenho; atualmente o *framework* encontra-se na versão *4,*trazendototal integração com *Java* EE 7, mas sempre mantendo sua proposta.*VRaptor.org (2018) “* O *VRaptor* 4 traz alta produtividade para um desenvolvimento Java Web rápido e fácil com CDI”.

O que seria o CDI e qual vantagem da utilização desse componente ao *VRaptor*?

CDI - *Contexts and Dependency Injection* (Injeção de Dependência e Contextos), em sua versão 1.0 introduzida junto a plataforma *Java* EE 6 e atualmente na versão 2.0 continuada no *Java* EE 8; possui um conjunto de serviços com intuito de melhorar a estrutura do código visando a produtividade, fornece uma arquitetura uniforme para injeção de dependência e o gerenciamento do ciclo de vida de beans. De acordo com o site oficial do CDI fica claro o objetivo e ganhos no caso de sua utilização. *Cdi-spec.org* (2018):

- Um ciclo de vida bem definido para objetos com estado ligados a contextos de ciclo de vida, em que o conjunto de contextos é extensível.

-Um mecanismo de injeção de dependência sofisticado e seguro de tipos, incluindo a capacidade de selecionar dependências no tempo de desenvolvimento ou de implantação, sem configuração detalhada.

- Suporte para modularidade *Java* EE e a arquitetura de componente *Java* EE - a estrutura modular de um aplicativo *Java* EE é levada em conta ao resolver dependências entre componente Java EE.

Além disso, o *VRaptor,* trabalha com o conceito de estrutura MVC e integra com as arquiteturas atuais como o REST *– Representational State Transfer* (Transferência de Estado Representacional) e *ActionBased,* trazendo consigo mais benefícios em sua utilização;

Um exemplo, ao utilizar a arquitetura REST, Arnon Rotem-Gal-Oz (2012, pg. 234), faz uma observação positiva em sua utilização: “O termo implica integração fácil e rápida, frequentemente usando APIs e fontes de dados para produzir resultados enriquecidos que não era necessariamente a razão original para produzir os dados da fonte bruta. As principais características do *mashup* são combinação, visualização e agregação. É importante tornar os dados existentes mais úteis, além disso, para uso pessoal e profissional. ”

Em questão da arquitetura *ActionBased*, que suportam as requisições de entrada através de controladores de ações. Paulo Silveira, Guilherme Silveira, Sergio Lopes, Guilherme Moreira, Nico Stepat, Fabio Kung (2011, pg. 152 e 153) orienta da seguinte forma “Podemos preferir trabalhar orientados a requisições e respostas com características stateless para melhor aproveitar a Web e outras ferramentas que giram em torno do HTTP, além de garantir escalabilidade e disponibilidade mais facilmente. Nesse tipo de situação, usar um framework action-based costuma se encaixar melhor como solução”.

O *framework VRaptor* possui a característica de flexibilidade, permitindo a possibilidade de sobrescrever praticamente quase todos os seus comportamentos, sem a necessidade das configurações em XML. Assim, o desenvolvedor ganha autonomia para fazer ajustes e configurações específicas de acordo com seu projeto. Essa facilidade é descrita por Lucas Cavalcanti (2014, pg. 3), mostrando a importância da seguinte forma “mesmo os problemas mais complexos e necessidades mais específicas dos projetos conseguiram ser resolvidos sobrescrevendo o comportamento do *VRaptor* usando os meios normais da sua API, ou sobrescrevendo um de seus componentes”.

Finalizando, o *VRaptor* tem se empenhado em atrair e conquistar desenvolvedores, através dos conceitos citado acima, mantendo sempre seu foco de extensibilidade e desenvolvimento de aplicações o mais fácil e produtivo.

1. **Materiais e Métodos**

Para avaliação e levantamento de critérios de comparação entre os frameworks foi desenvolvido um protótipo com cada ferramenta estudada. A camada de negócio da aplicação foi desenvolvida anteriormente e foi usada da mesma forma para todos os frameworks.

O protótipo é uma simplificação de uma aplicação de cálculo de contratos de empréstimo para fins de renegociação. A aplicação foi escolhida por conter alguns cadastros simples, sem regras de negócio complexas, e uma rotina de cálculo, específica da aplicação. Dessa forma foi possível avaliar o desempenho dos frameworks tanto nas tarefas simples, de simplesmente manter uma tabela num banco de dados, quanto para tarefas mais complexas, como uma rotina de cálculo.

Para a realização prática deste trabalho de conclusão de curso, foi desenvolvida uma aplicação que exemplifica a integração das linguagens Ruby e Java, que possibilitando agregar a facilidade do código Ruby com as importantes funcionalidades e bibliotecas do Java.

A aplicação escolhida consiste em um Web site de comercio eletrônico, totalmente escrito em linguagem Ruby, implantada em um servidor Glassfish, com chamadas de códigos Java. O intuito disso é mostrar a possibilidade de comunicação entre as duas linguagens utilizadas.

As principais integrações desenvolvidas, que serão mais detalhadas nas próximas seções, serviram não apenas para agregar maior funcionalidade e segurança ao sistema, como, principalmente, demonstrar que é possível utilizar qualquer biblioteca desenvolvida para a linguagem Java em código Ruby através do interpretador JRuby.

Neste capítulo apresenta-se o software utilizado e a modelagem do protótipo,

com o principal objetivo de esclarecer suas funcionalidades e estrutura do sistema a ser

desenvolvido.

Neste capitulo é apresentado a implementação do sistema proposto

anteriormente, o qual foi desenvolvido utilizando o padrão arquitetural MVC, as

tecnologias utilizadas para o desenvolvimento deste protótipo, suas telas e principais

códigos.

**Ferramentas utilizadas no desenvolvimento do sistema**

**Conclusão**

Neste capítulo foi apresentado, de forma detalhada, a implementação do

protótipo desenvolvido após o estudo de caso sobre padrões de desenvolvimento de

software, protótipo este, que utilizou o padrão MVC **Conclusão**

Neste capítulo foi apresentado, de forma detalhada, a implementação do

protótipo desenvolvido após o estudo de caso sobre padrões de desenvolvimento de

software, protótipo este, que utilizou o padrão MVC

**4 – Resultados ou Discussão**

### *Spring MVC*

O *Spring MVC* ajuda a construir aplicações web flexíveis e com baixo acoplamento. O padrão de design Modelo-visão-controlador ajuda na separação da lógica de negócio, lógica de apresentação e lógica de navegação.

Uma estrutura completa para a criação de aplicativos da web com enorme estabilidade, amplo alcance e usada por muitas pessoas, a simplicidade de configurar, pouca intrusão do framework, diminuição do acoplamento ao utilizá-lo, a modularização dos projetos, a ótima integração entre os projetos Spring e a fácil customização do *framework*.

*Spring MVC* é um dos módulos que compõem o *Spring Framework* utilizado para construir aplicações web. Ele conta com as boas práticas de projeto para desenvolvimento de software web utilizando a plataforma Java EE.

Pontos negativos a serem observados no *Spring MVC*, no qual embora simples, possui muitas camadas e abstrações que podem ser difíceis de depurar se surgirem problemas. Também é altamente dependente do núcleo do *Spring*. É uma estrutura antiga e madura que possui inúmeras maneiras de estendê-la e configurá-la - e isso, na verdade, tornando-a bastante complexa, ele ainda não fornece nenhuma estrutura rica para construir boas Interfaces.

Curva de aprendizado íngreme, mas quando você trabalha com o produto algumas vezes, é muito fácil se adaptar e melhorar, porém se deseja incluir outros módulos do Spring, pode exigir mais tempo de aprendizagem, pois requer mais tempo para adquiri conhecimento para customizar novos componentes.

A documentação oficial cobre praticamente tudo. O site oficial também tem uma série de ótimos tutoriais em formatos de vídeo e texto. Há links para os repositórios do *Github* para aplicativos de amostra do *Spring* e também há muitos tutoriais de terceiros para o fato de que o *Spring MVC* é amplamente utilizado por muitos desenvolvedores experientes. Entretanto, como o *framework MVC* é apenas uma parte do Spring, ele acaba tendo uma documentação menos detalhada tanto nos livros quanto na documentação oficial.

*Spring MVC* tem uma comunidade massiva de seguidores que são muito úteis e forneceram vários tutoriais e respostas sobre o SO. A Spring até realiza uma conferência anual chamada *SpringOne*. Os fóruns do *Spring* e SO são ótimos lugares para perguntar e obter ajuda sobre qualquer coisa relacionada à Primavera. O blog e o boletim informativo do site mantêm os desenvolvedores informados sobre todas as notícias relacionadas à estrutura.

Um levantamento em um dos maiores fóruns internacional o *StackOverflow*, o *framework* possui 49.722 interações relacionado ao Spring MVC, em uma matéria do site *JavaPipe* e *DailyRazor*, o *Spring MVC* aparece entre os dez *frameworks Java* mais utilizado, e um levantamento que foi realizado pelo *Rebellabs* em 2016 a *framework* apareceu em primeiro lugar em uso.

O *Spring MVC* é bem reconhecido no mercado trabalho tanto nacional como internacional, pois junto com o *Spring*, está constantemente mudando e melhorando. A questão é que seus desenvolvedores terão que acompanhar constantemente a tecnologia para melhorar o aplicativo à medida que o Java cresce, os navegadores da Web mudam e outras melhorias acontecem no espaço da Web.

### *VRaptor*

Se você precisa de uma estrutura fácil e funcional para criar programas de web com Java, o *VRaptor* é o caminho certo. De *e-commerces* a aplicações de grande escala. Sem dúvida, é fácil de usar e agradável criar um utilitário com este *framework*.

O *VRaptor* é um *Framework MVC* para desenvolvimento rápido de aplicações WEB que faz uso das anotações e conceitos de inversão de controles e injeção de dependência. Outros conceitos como o de Convenção do Invés de Configuração tornam o desenvolvimento bastante produtivo sem perder flexibilidade tornando a curva de aprendizado muito pequena.

Um ponto negativo ao *VRaptor*, talvez seja por não possuir bibliotecas ou componentes voltado a camada da visão, exigindo ao desenvolvedor o conhecimento voltado ao *front-end* como por exemplo de linguagens como CSS, HTML e *JavaScript*.

Em questão da documentação o *VRaptor* possui a documentação oficial centralizada em seu site oficial, no qual é possível verificar instruções de uso, exemplos de aplicações e tutoriais, além de possuir a documentação traduzida em português como um diferencial, porém poderia ser ainda melhor se tivesse mais explicações mais detalhadas sobre seu funcionamento do fluxo interno e sua estrutura, além de como resolver algumas exceções a serem tratadas, o *VRaptor* também possui documentações não oficiais através de *blogs*, fóruns, livros e artigos.

A comunidade do *VRaptor*, por sua vez, é um projeto brasileiro e não possui grande expressão no mercado exterior. Em consulta ao *StackOverflow*, possui somente cerca de 123 ocorrências de postagens que referenciam o *VRaptor*.

No mercado de trabalho o *VRaptor* ainda possui pouca representatividade, porém existe grandes empresas que utilizam o *VRaptor*, como o *Mamute*, *GUJ, Wine e Locaweb.*

**5 – Conclusão**

Na escolha do *framework*, há relevante considerações que devem ser analisadas para sua escolha, como técnica, segurança, documentação, licença, popularidade, filosofia, sustentabilidade e recurso no mercado.

Há uma grande variedade de *frameworks* para o desenvolvimento Web em Java, o que torna muito difícil a sua avaliação. O levantamento de critérios auxilia a escolha de um framework para uma determinada situação, pois permite a tabulação das características de cada artefato estudado, facilitando assim a análise.

No artigo foi possível visualizar a arquitetura MVC no qual ambos os *frameworks* trabalham, além do desacoplamento da camada visão e utilização de injeção de dependências em que são semelhantes, e a estrutura particular de cada um, podendo visualizar as vantagens e desvantagens.

Porém, é difícil levantar critérios objetivos na comparação de tecnologias. Critérios como velocidade de desenvolvimento ou linhas de código necessárias para desenvolver uma aplicação não seriam avaliadas adequadamente apenas com a construção de protótipos.

O estabelecimento de critérios, embora subjetivos, deve auxiliar futuras avaliações de frameworks, permitindo que o analista investigue diretamente a classificação do artefato nos critérios pré-estabelecidos.

No caso dos frameworks avaliado, não é possível apontar qual seria o ideal para qualquer situação. No entanto, o estudo feito deve auxiliar na análise numa situação específica, uma vez que os dados estão tabulados e seu embasamento está contido no trabalho.

- As vantagens de utilizar uma arquitetura que separe a lógica de apresentação e a manipulação de componentes de interface gráfica são desde uma melhor legibilidade e modularidade do código, o que facilita a manutenção, até uma melhora na testabilidade das classes

- O levantamento de critérios auxilia a escolha de um framework para uma determinada situação, pois permite a tabulação das características de cada artefato estudado, facilitando assim a análise.

A elaboração de um quadro comparativo facilita as análises por sintetizar boa parte do estudo elaborado. Ele pode ser facilmente atualizado, no caso de uma nova versão do framework, ou expandido, no caso da avaliação de um novo framework.

No caso dos frameworks avaliado, não é possível apontar qual seria o ideal para qualquer situação. No entanto, o estudo feito deve auxiliar na analise numa situação específica, uma vez que os dados estão tabulados e seu embasamento está contido no trabalho.

Além da agilidade, outros fatores como robustez, segurança e praticidade formam as palavras-chave no desenvolvimento de uma aplicação qualificada a exercer qualquer função essencial para estas organizações.

**Referências Bibliográficas**

BERGMAN Noel J, CHOPRA Abhinav. **Introdução às páginas do JavaServer**. Publicado em 28 de Ago. 2001. Disponível em <https://www.ibm.com/developerworks/java/tutorials/j-introjsp/j-introjsp.html>. Acesso em 02 de Set. 2018.

CAELUM, Vraptor 4. **Repositório de download e instruções de instalação da framework**. Disponível em <https://github.com/caelum/vraptor4>. *2018 Github, Inc.* Acesso em 27 de Ago. 2018.

CALVALCANTI Lucas, ***Vraptor*– Desenvolvimento Ágil para Web com Java**, publicado em 16 de Abr. 2014, Ed. Casa do Código.

CHRISTOPHER[Alexander, **An Introduction for Object-Oriented Designers**](http://g.oswego.edu/dl/ca/ca/ca.html). Disponível em <http://g.oswego.edu/dl/ca/ca/ca.html>, Acessado em 07 de Set. 2018

DAILYRAZOR, **The 10 Best Java Web Frameworks for 2018**. Disponível em [www.dailyrazor.com/blog/best-java-web-frameworks/](http://www.dailyrazor.com/blog/best-java-web-frameworks/) . Acesso em 20 de Nov. 2018

FRANZINI, Fernando, **O que aprendi com livro VRaptor – Desenvovlimento Ágil para Web com *Java***. Publicado em 11 de Dez. 2013. Disponível em<https://imasters.com.br/back-end/o-que-aprendi-com-o-livro-vraptor-desenvolvimento-agil-para-web-com-java>

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

HEMRAJANI, Anil, **Agile JAVA Development with Spring, Hibernate and Eclipse**. Publicado em 2006. Editora Paperback

JAVAPIPE **– 10 Best Java Web Frameworks to Use in 2018 (100% Future-Proof)**. Disponível em <https://javapipe.com/hosting/blog/best-java-web-frameworks>. Acesso em 10 de Nov. 2018

KAYAL, Dhrubojyoti, **Pro JAVA Spring Patterns – Best Practices and Design Strategies Implementing JAVA EE Patterns with the Sprign Framework**, Publicado em 2008. Editora Apress.

MATTEI, Marcelo. **Boas práticas no desenvolvimento de websites.** Publicado em 19 de Nov. 2007. Disponível em <https://webinsider.com.br/boas-praticas-no-desenvolvimento-de-websites/>. Acesso em 27 de Ago. 2018.

ORACLE, **O que é um *servlet*?** Disponível em <https://docs.oracle.com/javaee/6/tutorial/doc/bnafe.html>. Acesso em 02 de Set. 2018.

REENSKAUG, T., **Models - Views - Controllers**. Disponível em <https://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf>. Acessado em 07 de Set. 2018.

REBBELSLAB, **Java Tools and Technologies Landscape Report 2016**, Disponível em <https://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-2016/#web-frameworks>. Acesso em 18 de Nov. 2018

ROTEM GAL OZ, Arnon, ***Soa Patterns***. Publicado em 23 de Set. 2012. Ed. *Manning.*

# SILVEIRA Paulo e Guilherme, Lopes Sergio,Moreira Guilherme, Steppat Nico, Kung Fabio, Introdução À Arquitetura e Design de Software - Uma Visão Sobre a Plataforma Java, publicado em 9 de Dez. 2011,  Ed. *Elsevier Ltda.*

SPRING. **Documentação oficial do framework**. Disponível em <https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html>.*2018 Pivotal Software.* Acesso em 27 de Ago. 2018.

SPRING-PROJECT, Spring Framework. **Repositório de projetos, instruções e downloads referente a framework**. Disponível em <https://github.com/spring-projects/spring-framework>. *2018 Github, Inc*. Acesso em 27 de Ago. 2018.

VRAPTOR. **Documentação oficial do framework**. Disponível em <http://www.vraptor.org/pt/docs>. Acesso em 27 de Ago. 2018.

WEISSMANN Henrique Lobo, **Vire o jogo com *Spring Framework***. Publicado em 16 de Abr. 2014. Ed. Casa do Código.

Figura: 1 - Disponível em: <https://www.javatpoint.com/spring-mvc-tutorial>. Acesso em 20 de Nov. 2018.

LADD, S. **Expert Spring MVC and Web Flows**, Primeira Edição, Nova Iorque: Apress, 2006

NEGRÃO, Eduardo. **Padrões de Projetos (Designer patterns) -** O que são? Disponível em:

<http://portalengenhariadesoftware.blogspot.com.br/2010/04/padroes-de-projeto-designpatterns-

o.html>. Acesso em: 14 maio 2013.