

**FELIPE RODRIGUES DO PRADO  
JOÃO PAULO NAKAJIMA PEREIRA**

**DESENVOLVIMENTO MODULAR DE SOFTWARE  
UTILIZANDO OSGI**

**UNIVERSIDADE DO VALE DO SAPUCAÍ  
POUSO ALEGRE  
2015**

## LISTA DE FIGURAS

Figura 1 - Módulo API, <i>Core</i> e <i>View</i> do cliente instalados e ativos.....	6
Figura 2 - Módulo de cadastro de clientes ativo no sistema.....	6
Figura 3 - Módulos <i>Core</i> e <i>View</i> do cliente parados.....	7
Figura 4 - Módulo de cadastro de clientes parado no sistema.....	7
Figura 5 - Módulo <i>View</i> do cadastro de clientes parado no sistema.....	8
Figura 6 - Diagrama representando a granularidade do <i>software</i> .....	10

## **LISTA DE TABELAS**

## LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
BSD	<i>Berkeley Software Distribution</i>
CSS	<i>Cascading Style Sheet</i>
EJB	<i>Enterprise JavaBeans</i>
HTML	<i>HyperText Markup Language</i>
IDE	<i>Integrated Development Environment</i>
IoT	<i>Internet of Things</i>
JDBC	<i>Java Database Connectivity</i>
JEE	<i>Java Platform, Enterprise Edition.</i>
JMS	<i>Java Message Service</i>
JVM	<i>Java Virtual Machine</i>
NTT	<i>Nippon Telegraph and Telephone</i>
OSGi	<i>Open Services Gateway initiative</i>
SQL	<i>Structured Query Language</i>
SRA	<i>Systems Research and Applications Corporation</i>
TDC	<i>The Developers Conference</i>
UNIVAS	<i>Universidade do Vale do Sapucaí</i>
XHTML	<i>Extensible HyperText Markup Language</i>
XML	<i>Extensible Markup Language</i>
W3C	<i>World Wide Web Consortium</i>

## SUMÁRIO

1 DISCUSSÃO DE RESULTADOS.....	5
REFERÊNCIAS.....	12

# 1 DISCUSSÃO DE RESULTADOS

Neste capítulo serão apresentados e discutidos os resultados obtidos conforme a pesquisa realizada sobre modularização de *softwares* utilizando a tecnologia OSGi. Com o objetivo de destacar suas características e os pontos principais.

O objetivo da pesquisa foi demonstrar um modelo de desenvolvimento modular utilizando a tecnologia OSGi, a qual fornece suporte para o mesmo. Dessa maneira decidiu-se desenvolver uma aplicação simples que exemplifique a utilização dessa tecnologia.

Em uma visão geral, a aplicação desenvolvida está composta pelos módulos de usuário, clientes, financeiro, *data source* e log. Porém, esses módulos por sua vez, são compostos por módulos menores, que proporcionam flexibilidade e desacoplamento dos mesmos no *software*.

Conforme afirma Knoernschild (2012), uma aplicação modular é aquela que seus módulos podem ser instaláveis, parados, reiniciados e desinstalados sem interromper o restante da aplicação, além de serem reutilizáveis, combináveis e oferecerem uma interface clara.

Com base nessa teoria e utilizando a tecnologia OSGi que oferece suporte a essas características, desenvolveu-se uma aplicação que permite que seus módulos possam ser instalados, desinstalados, parados, atualizados sem interromper o restante da aplicação. Os mesmos também podem ser reutilizáveis e combinados com outros módulos. É importante destacar que existem módulos que não devem ser parados ou desinstalados por serem partes principais do sistema.

Com exceção dos módulos principais, que são denominados como *Main Util API*, *Main Web Resources View*, e todos os outros módulos do tipo API, os demais módulos podem ser desinstalados, instalados, reiniciados e parados sem comprometer o restante do sistema, apenas parando de executar suas responsabilidades específicas.

A Figura 1 demonstra, através da ferramenta *Apache Felix Web Console Bundles* que possibilita o gerenciamento dos módulos, o módulo de clientes instalado. O mesmo é composto por três módulos nomeados de *Customer API*, *Customer Core* e *Customer Register View*.

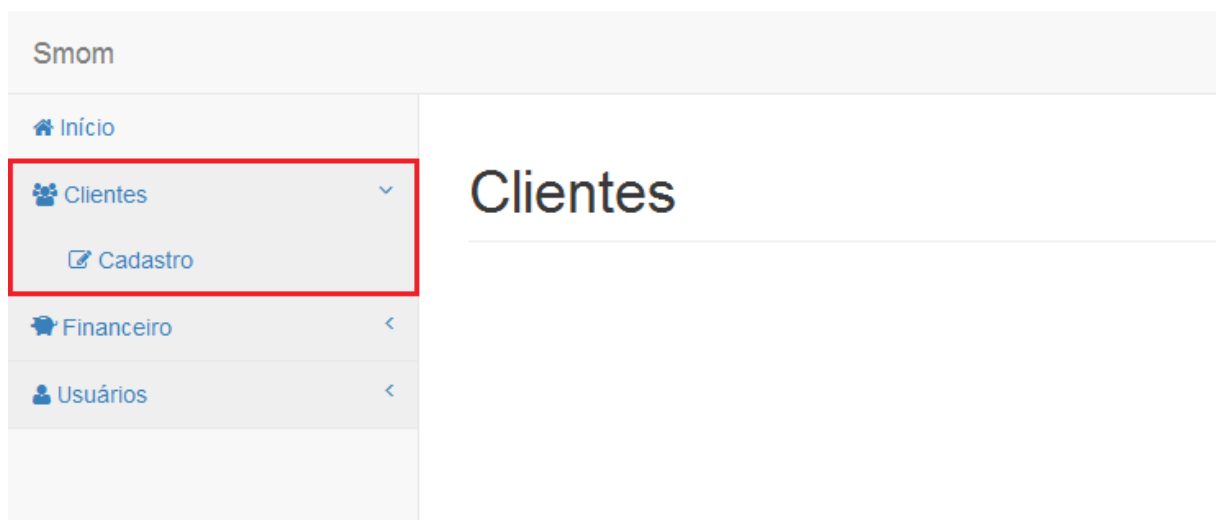
## Apache Felix Web Console Bundles



OSGi Status Web Console						Log out
Bundle information: 313 bundles in total, 109 bundles active, 8 active fragments, 100 bundles resolved, 96 bundles installed						
<input type="text"/> Apply Filter Filter All <span>Reload</span> <span>Install/Update...</span> <span>Refresh Packages</span>						
Id	Name	Version	Category	Status	Actions	
327	Customer Register View ( <i>br.com.smom.customer.register.view</i> )	1.0.0		Active		
326	Customer Core ( <i>br.com.smom.customer.core-v1</i> )	1.0.0		Active		
325	Customer API ( <i>br.com.smom.customer.api-v1</i> )	1.0.0		Active		
324	Main Web Resources View ( <i>br.com.smom.main.webresources.view</i> )	1.0.0		Active		
323	Main Util API ( <i>br.com.smom.main.util.api-v1</i> )	1.0.0		Active		
317	Apache Log4j ( <i>log4j</i> )	1.2.17		Active		
310	Apache Commons Pool ( <i>org.apache.commons.pool2</i> )	2.4.2		Active		
309	Apache Commons Logging ( <i>org.apache.commons.logging</i> )	1.2.0		Active		
308	Apache Commons DBCP ( <i>org.apache.commons.dbcp2</i> )	2.1.1		Active		
303	PostgreSQL JDBC Driver JDBC41 ( <i>org.postgresql.jdbc41</i> )	9.4.0.build-1201		Active		

**Figura 1** - Módulo API, Core e View do cliente instalados e ativos. **Fonte:** Elaborado pelos autores.

A Figura 2 demonstra que o módulo de clientes está disponível no menu do sistema. Dos três módulos que compõe o módulo de clientes, é possível parar ou desinstalar o módulo *Customer Register View* e *Customer Core* sem afetar o restante do sistema. A Figura 3 demonstra agora esses dois módulos parados e a Figura 4 demonstra o *software* ainda em funcionamento porém com o módulo de clientes indisponível para acesso.



**Figura 2** - Módulo de cadastro de clientes ativo no sistema. **Fonte:** Elaborado pelos autores.

## Apache Felix Web Console Bundles



OSGi Status Web Console Log out

Bundle information: 313 bundles in total, 109 bundles active, 8 active fragments, 100 bundles resolved, 96 bundles installed

Apply Filter

Reload Install/Update... Refresh Packages









































<div>Id</div>	<div>Name</div>	<div>Version</div>	<div>Category</div>	<div>Status</div>	<div>Actions</div>
327	Customer Register View (br.com.smom.customer.register.view)	1.0.0		Resolved	   
326	Customer Core (br.com.smom.customer.core-v1)	1.0.0		Resolved	   
325	Customer API (br.com.smom.customer.api-v1)	1.0.0		Active	   
324	Main Web Resources View (br.com.smom.main.webresources.view)	1.0.0		Active	   
323	Main Util API (br.com.smom.main.util.api-v1)	1.0.0		Active	   
317	Apache Log4j (log4j)	1.2.17		Active	   
310	Apache Commons Pool (org.apache.commons.pool2)	2.4.2		Active	   
309	Apache Commons Logging (org.apache.commons.logging)	1.2.0		Active	   
308	Apache Commons DBCP (org.apache.commons.dbcp2)	2.1.1		Active	   
303	PostgreSQL JDBC Driver JDBC41 (org.postgresql.jdbc41)	9.4.0.build-1201		Active	   

Figura 3 - Módulos Core e View do cliente parados. Fonte: Elaborado pelos autores.

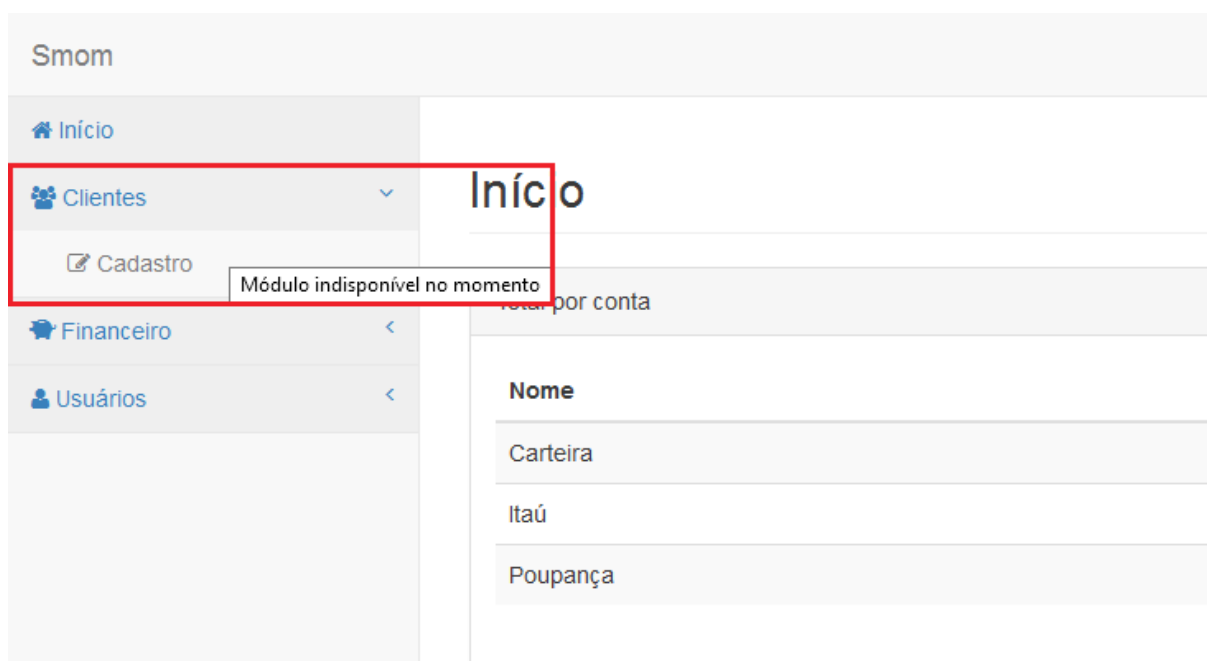


Figura 4 - Módulo de cadastro de clientes parado no sistema. Fonte: Elaborado pelos autores.

Nesse exemplo os módulos *Customer Core* e *Customer Register View* foram parados e com isso o módulo de cadastro de cliente fica desativado no menu do sistema conforme demonstrado na Figura 4. Porém se somente o módulo *Customer Register View* for parado o módulo de cadastro de clientes já fica desativado, com isso as funcionalidades do módulo *Customer Core* continuam funcionando. A Figura 5, mostra o que acontece ao acessar a página do cadastro de clientes com seu respectivo módulo parado.



## Apache Felix Web Console Bundles



OSGi Status Web Console

Log out

Bundle information: 314 bundles in total, 113 bundles active, 8 active fragments, 105 bundles resolved, 88 bundles installed

x

Apply Filter

Filter All

Reload

Install/Update...

Refresh Packages

Id	Name	Version	Category	Status	Actions
327	Customer Register View (br.com.smom.customer.register.view)	1.0.0		Resolved	<div><div></div><div></div><div></div><div></div></div>
326	Customer Core (br.com.smom.customer.core-v1)	1.0.0		Active	<div><div></div><div></div><div></div><div></div></div>
325	Customer API (br.com.smom.customer.api-v1)	1.0.0		Active	<div><div></div><div></div><div></div><div></div></div>
324	Main Web Resources View (br.com.smom.main.webresources.view)	1.0.0		Active	<div><div></div><div></div><div></div><div></div></div>
323	Main Util API (br.com.smom.main.util.api-v1)	1.0.0		Active	<div><div></div><div></div><div></div><div></div></div>

localhost:8080/modules/customer/register/ Search

⚠ Módulo indisponível no momento

Módulo indisponível no momento, contate o administrador do sistema ou tente mais tarde.

**Figura 5** - Módulo *View* do cadastro de clientes parado no sistema. **Fonte:** Elaborado pelos autores.

Utilizando a tecnologia OSGi e estruturando os módulos através de uma arquitetura que forneça flexibilidade e o desacoplamento dos módulos, confirma-se então a teoria de Knoernschild, que um sistema modular é aquele onde é possível gerenciar seus módulos sem comprometer o restante da aplicação.

Umas das vantagens da modularização é a reutilização das funcionalidades que uma vez implementadas, podem ser reutilizadas em outros sistemas ou partes internas do mesmo sistema e, para se conseguir fazer isso foi necessário definir bem a estrutura do sistema, organizando as funcionalidades e responsabilidades de cada módulo, de forma em que sua arquitetura permitisse uma alta reutilização de funcionalidades. Porém essa organização da arquitetura exige que nos momentos iniciais do desenvolvimento do *software* seja criado uma granularidade adequada para o mesmo.

A granularidade é formada pelo tamanho ou complexidade dos módulos, quanto menor a granularidade do *software*, menores são os módulos, assim possuem menos inteligência de negócio dentro deles, isso facilita a reutilização dos módulos, porém dificulta o uso deles, pois o sistema ao todo terá um grande número de módulos, cada um representando uma funcionalidade, o que no caso, talvez se torne mais complicado e difícil de usar e expandir o sistema do que reutilizar. Mas em contrapartida temos a alta granularidade, onde um módulo possui uma granularidade maior, sendo relativamente grande e com mais inteligência de

negócio, usar ele vai ser fácil de usar, pois para executar uma rotina é necessário realizar a chamada a um serviço de determinado módulo, porém no momento em que alguma rotina ou código que está presente dentro daquele módulo tiver de ser reutilizada em outro local da aplicação, não vai ser possível, pois essa funcionalidade está encapsulada junto às outras dentro do módulo, impedindo assim de reutilizar ela (KNOERNSCHILD, 2012).

Com base na teoria citada pelo autor Knoernschild, foram desenvolvidos vários protótipos até que encontrou-se a granularidade ideal para podermos desenvolver um *software* que exemplifica o modelo de desenvolvimento modular. Conforme mostra a Figura 6, essa foi a granularidade criada para o *software*. É extremamente importante salientar que cada projeto vai possuir uma granularidade própria, ou seja, dependendo do *software*, e do objetivo a ser atingido, essa granularidade estará se ajustando ao mesmo. Com isso essa granularidade por nós autores encontrada pode ser profundamente alterada dependendo do projeto a ser desenvolvido utilizando OSGi.



Nesta granularidade dividiu-se cada um dos módulos em outros menores, em que cada módulo principal é composto por outros três módulos menores:

- A parte das interfaces dos serviços disponibilizadas aos outros módulos, ou seja, um módulo que contém a API;
- A parte da implementação dessas interfaces, nessa parte se teria as classes e códigos desenvolvidos em Java que implementam as interfaces e teriam consigo a inteligência e regras de negócio da aplicação, essa parte é o módulo *Core* da aplicação;
- A parte da visualização e interação com o usuário, o *front-end* deste módulo. Nesta parte estaria toda o código *web* que realiza a interação com usuário.

Alguns módulos como o de *Log* e *Data Source* não são compostos por essa estrutura, os mesmos não possuem a parte de interação com o usuário, pois apenas servem como funcionalidades utilizadas pelos outros módulos.

A granularidade resultante do desenvolvimento deste *software*, possibilita a reutilização de funcionalidades, tanto por outros módulos como por outros *softwares*.

Além disso, com essa divisão entre as funcionalidades e códigos do sistema, foi possível obter uma alta coesão e desacoplamento, uma vez que com essa granularidade cada módulo executa apenas a sua funcionalidade, além de não depender diretamente da implementação de outros módulos.

Com base nas funcionalidades dos módulos do sistema, caso dividísse-mos uma funcionalidade para cada módulo, iríamos obter uma alta granularidade, assim o reaproveitamento de alguma funcionalidade presente dentro do módulo de financeiro, por exemplo, estaria encapsulada, pois para reaproveitar em outra parte do sistema seria necessário levar junto o restante de códigos e funcionalidades do módulo, o que não é aceitável, pois teríamos no sistema código duplicado e que afetaria a coesão da implementação.

## REFERÊNCIAS

APACHE. **OSGi Frequently Asked Questions**. 2015. Disponível em <http://felix.apache.org/documentation/tutorials-examples-and-presentations/apache-felix-osgi-faq.html>. Acesso em 16 de junho, 2015.

APACHE. **What is maven**. 2015. Disponível em <http://maven.apache.org/what-is-maven.html>. Acesso em 16 de junho, 2015.

APPOLINÁRIO, Fábio. **Dicionário de metodologia: um guia para a produção do conhecimento científico**. São Paulo: Atlas, 2004.

BORBA, Paulo. **Aspectos de Modularização**. 2015. Disponível em <http://www.di.ufpe.br/~java/graduacao961/aulas/aula4/aula4.html>. Acesso em 21 de junho, 2015.

BARTLETT, Neil. **OSGi In Practice**. 2009.

BOSSCHAERT, David. **OSGi in Depth**. Shelter Island: Manning Publications Co, 2012

CAELUM. **Apostila Desenvolvimento Web com HTML, CSS e JavaScript**. 2015. Disponível em <https://www.caelum.com.br/apostila-html-css-javascript/javascript-e-interatividade-na-web/>. Acesso em 08 de março, 2015.

CAELUM. **Apostila Java e Orientação a Objetos**. 2015. Disponível em <http://www.caelum.com.br/apostila-java-orientacao-objetos/>. Acesso em 08 de março, 2015.

CLARO, Daniela Barreiro; SOBRAL, João Bosco Manguiera. **Programação em Java**. Santa Catarina: Cengage Learning, 2008.

COOK, Stuart; SELTZER, Claire; WRIGHTSMAN, Lawrence Samuel. **Métodos de pesquisa nas relações sociais**. São Paulo: EPU, 1987.

COSTA, Gabriel. **O que é bootstrap?** 2014. Disponível em <http://www.tutorialwebdesign.com.br/o-que-e-bootstrap/>. Acesso em 09 de agosto, 2015.

DEITEL, Harvey Matt; DEITEL, Paul John. **Java How to Program**. 8. ed. Edson Furmankiewicz. São Paulo: Pearson Prentice Hall, 2010.

DEVMEDIA. **Introdução ao Spring Framework**. 2015. Disponível em <http://www.devmedia.com.br/introducao-ao-spring-framework/26212>. Acesso em 10 de março, 2015.

DEVMEDIA. **Novidades do GlassFish 3.1**. 2011. Disponível em: <http://www.devmedia.com.br/novidades-do-glassfish-3-1-artigo-java-magazine-91/21124>.

Acesso em 19 de junho, 2015.

DURHAM, Alan; JOHNSON, Ralph. A Framework for Run-time Systems and its Visual Programming Language. In: **OBJECT-ORIENTED PROGRAMMING SYSTEMS, LANGUAGES, AND APPLICATIONS**. San Jose, CA, 1996, p. 20-25.

FERNANDES, Leonardo. **OSGi e os benefícios de uma Arquitetura Modular**. 37.ed. 2009. p. 27-35.

GAMA, Kiev. **Uma visão geral sobre a plataforma OSGi**. 2008. Disponível em <https://kievgama.wordpress.com/2008/11/24/um-pouco-de-osgi-em-portugues/>. Acesso em 09 de março, 2015.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa**. São Paulo: Atlas, 2002.

GONÇALVES, Antonio. **Beginning Java EE 6 Platform with GlassFish 3**. Nova York: Springer Science+Business Media, LCC. 2010.

JERSEY. **Jersey: RESTful Web Services in Java**. 2015. Disponível em <https://jersey.java.net/>. Acesso em 10 de agosto, 2015.

KIOSKEA. **Condução de reunião**. 2014. Disponível em <http://pt.kioskea.net/contents/579-conducao-de-reuniao>. Acesso em 16 de abril, 2015.

KNOERNSCHILD, Kirk. **Java Application Architecture: Modularity Patterns with Examples Using OSGi**. Crawfordsville: Pearson Education, 2012.

LIRA, Douglas; SCHMITZ, Daniel. **AngularJS na prática**. São Paulo: Leanpub, 2014.

LUCENA, Fábio Nogueira de. **Introdução ao Equinox**. 2010. Disponível em <https://code.google.com/p/exemplos/wiki/equinox>. Acesso em 09 de março, 2015.

MADEIRA, Marcelo. **OSGi – Modularizando sua aplicação**. 2009. Disponível em <https://celodemelo.wordpress.com/2009/11/12/osgi-modularizando-sua-aplicacao/>. Acesso em 09 de março, 2015.

MALCHER, Marcelo Andrade da Gama. **OSGi Distribuído: deployment local e execução remota**. Monografia de Seminários de Sistemas Distribuídos. Pontifícia Universidade Católica do Rio de Janeiro, 2008.

MARIE, Victor. **Bootsrapt e formulários HTML5**. 2015. Disponível em <http://www.caelum.com.br/apostila-html-css-javascript/bootstrap-e-formularios-html5/>. Acesso em 09 de agosto, 2015.

MAUJOR. **Site sobre CSS e Padrões Web: Por que CSS?**. 2015. Disponível em <http://www.maujor.com/index.php>. Acesso em 08 de março, 2015.

MAYWORM, Marcelo. **OSGi Distribuída: Uma Visão Geral**. 42.ed. 2010. p. 60-67.

MILANI, André. **PostgreSQL: Guia do Programador**. São Paulo: Novatec Editora, 2008.

Mozilla Developer Network. **CSS**. 2015. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em 08 de março, 2015.

Mozilla Developer Network. **HTML**. 2014. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em 07 de março, 2015.

Mozilla Developer Network. **JavaScript**. 2015. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em 08 de março, 2015.

Mozilla Developer Network. **JavaScript language resources**. 2014. Disponível em [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language\\_Resources](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources). Acesso em 08 de março, 2015.

OLIVEIRA, Eric. **Aprenda AngularJS com estes 5 Exemplos Práticos**. 2013. Disponível em <http://javascriptbrasil.com/2013/10/23/aprenda-angularjs-com-estes-5-exemplos-praticos/>. Acesso em 09 de agosto, 2015.

OSGI ALLIANCE. **OSGi**. 2015. Disponível em <http://www.osgi.org/Main/HomePage>. Acesso em 08 de março, 2015.

ORACLE. **Difference between GlassFish Open Source and Commercial Editions**. 2011. Disponível em [https://blogs.oracle.com/GlassFishForBusiness/entry/difference\\_between\\_glassfish\\_open\\_source](https://blogs.oracle.com/GlassFishForBusiness/entry/difference_between_glassfish_open_source). Acesso em 20 de junho, 2015.

SAUDATE, Alexandre. **REST: Construa API's inteligentes de maneira simples**. São Paulo: Casa do Código, 2013.

SANTOS FILHO, Walter dos. **Introdução ao Apache Maven**. Belo Horizonte: Eteg Tecnologia da Informação Ltda, 2008.

SANTOS, Wagner Roberto dos. **RESTful Web Services e a API JAX-RS**. 2009. Disponível em <http://www.univale.com.br/unisite/mundo-j/artigos/35RESTful.pdf>. Acesso em 08 de agosto, 2015.

SILVA, Maurício Samy. **CSS3: Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3**. São Paulo: Novatec Editora, 2012.

SILVA, Maurício Samy. **HTML5: A linguagem de marcação que revolucionou a web**. São Paulo: Novatec Editora, 2011.

SILVA, Maurício Samy. **JavaScript: Guia do Programador**. São Paulo: Novatec Editora, 2010.

SOUZA, Arthur Câmara; AMARAL, Hugo Richard; LIZARDO, Luis Eduardo O.

**PostgreSQL:** uma alternativa para sistemas gerenciadores de banco de dados de código aberto. In: Anais do Congresso Nacional Universidade, EAD e Software Livre, 2012.

STERN, Eduardo Hoelz. **PostgreSQL** - Introducao e Conceitos. 2003. Disponível em <http://www.devmedia.com.br/artigo-sql-magazine-6-postgresql-introducao-e-conceitos/7185>. Acesso em 10 de agosto, 2015.

USP. **Fundamentos do projeto de software**. 2015. Disponível em <http://www.pcs.usp.br/~pcs722/98/Objetos/bases.html>. Acesso em 21 de junho, 2015.

W3C. **About W3C**. 2015. Disponível em <http://www.w3.org/Consortium/>. Acesso em 07 de março, 2015.

W3C. **What is CSS?**. 2015. Disponível em <http://www.w3.org/Style/CSS/>. Acesso em 08 de março, 2015. COOK, Stuart; SELLTIZ, Claire; WRIGHTSMAN, Lawrence Samuel. Métodos de pesquisa nas relações sociais. São Paulo: EPU, 1987.