FELIPE RODRIGUES DO PRADO
JOÃO PAULO NAKAJIMA PEREIRA

ARQUITETURA MODULAR DE SOFTWARES UTILIZANDO OSGI

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE
2015

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
CSS	*Cascading Style Sheet*
EJB	*Enterprise JavaBeans*
HTML	*HyperText Markup Language*
ICC	Inatel *Competence Center*
IDE	*Integrated Development Environment*
IoT	*Internet of Things*
JDBC	Java *Database Connectivity*
JEE	Java *Platform, Enterprise Edition.*
JMS	Java *Message Service*
JPA	Java *Persistence* API
JVM	Java *Virtual Machine*
NTT	*Nippon Telegraph and Telephone*
OSGi	*Open Services Gateway initiative*
SENAC	Serviço Nacional de Aprendizagem Comercial
SQL	*Structured Query Language*
SRA	*Systems Research and Applications Corporation*
UNIFEI	Universidade Federal de Itajubá
UNIVAS	Universidade do Vale do Sapucaí
XHTML	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*

SUMÁRIO

1 QUADRO METODOLÓGICO

Neste capítulo serão abordados e descritos os caminhos definidos e utilizados para conduzir o projeto até seu fim. Serão apresentados o tipo de pesquisa, seu contexto, bem como os instrumentos, os procedimentos, o orçamento e o cronograma para o desenvolvimento deste projeto.

1.1 Tipo de pesquisa

Pesquisa é um processo desenvolvido com o objetivo de se obter respostas para indagações propostas, através de conhecimentos existentes e a utilização de métodos, técnicas e procedimentos científicos. Uma pesquisa se faz necessária quando não existem repostas suficientes que satisfaçam a resolução de problemas (GIL, 2002).

Para atingir os objetivos deste projeto, desenvolveu-se uma pesquisa de abordagem aplicada, a qual é utilizada quando se desenvolve um produto real, com uma finalidade prática, que pode ser aplicada em determinado contexto. Conforme aponta Appolinário (2004), pesquisas aplicadas têm o objetivo de resolver problemas ou necessidades concretas e imediatas.

Aplicando esses conceitos e utilizando a pesquisa de forma aplicada por agregar maiores vantagens e demonstrar melhor os resultados obtidos, desenvolveu-se um *software* modularizado com o objetivo de apresentar o paradigma da modularização e sua arquitetura, apresentando suas vantagens como facilidade de manutenção e aplicação em empresas de diferentes ramos, utilizando a tecnologia OSGi.

1.2 Contexto de pesquisa

Cada vez mais *softwares* são utilizados em empresas, indústrias, computadores

pessoais, web e dispositivos móveis. Estes são desenvolvidos por meio de práticas e tecnologias existentes que auxiliam na sua criação, porém a não utilização de tais ferramentas tornam o seu desenvolvimento e manutenção um processo desgastante e trabalhoso.

Esta pesquisa demonstra a arquitetura e desenvolvimento modularizado utilizando o *framework* OSGi, oferecendo melhor organização na estrutura do *software*, vantagens na manutenção e maior facilidade na expansão do mesmo, isso pelo fato de o *framework* permitir a adição e integração de novas funcionalidades em modo *on-the-fly*[1].

Softwares modularizados oferecem maior flexibilidade para atender empresas de diferentes áreas. Através do OSGi, os módulos podem ser desenvolvidos de forma independente, disponibilizando módulos específicos para cada empresa. Dessa maneira foi-se desenvolvida uma aplicação que pode ser adotada por empresas de diferentes segmentos no mercado, bastando apenas que exista um módulo que atenda a necessidade específica da área.

O objetivo desse projeto é apresentar o paradigma do desenvolvimento modular, sua arquitetura, metodologias, engenharias, técnicas e tecnologias para o âmbito de softwares comerciais e provendo uma base para a facilitação na manutenção e expansão do mesmo. Esta pesquisa, por apresentar uma análise e demonstração do desenvolvimento modular, servirá de base de conhecimento para estudantes, profissionais da área de tecnologia da informação e empresas que pretendem empregar essa forma de desenvolver seus produtos de *software*.

1.3 Instrumentos

Durante o desenvolvimento deste projeto foi necessário a realização de reuniões entre os participantes, para a obtenção e organização das informações, divisão das tarefas e desenvolvimento do *software*, pois segundo Kioskea (2014), as reuniões são um meio para partilhar, num grupo de pessoas, um mesmo nível de conhecimento sobre um assunto ou um problema e para tomar decisões coletivamente. Além disso, decisões tomadas coletivamente, com representantes das diferentes entidades interessadas, serão facilmente aceitas por todos.

Os dados foram obtidos a partir de livros, artigos, documentações e reuniões online com pessoas que possuem conhecimento e experiência a respeito do tema do projeto. Após

1 *On-the-fly:* Termo referente ao funcionamento do sistema, onde a nova funcionalidade é adicionada sem que seja necessário reiniciar toda a aplicação, com ela em pleno funcionamento.

aplicados estes instrumentos foi possível analisar e organizar as informações e experiências coletadas.

As reuniões entre os participantes possuem o objetivo de planejar a execução do projeto, tomar decisões quanto ao desenvolvimento do *software*, bem como as tecnologias e metodologias que foram utilizadas.

Para obtermos mais conhecimentos acerca do tema, decidiu-se conversar com uma pessoa que possuísse grandes conhecimentos teóricos e práticos sobre modularização e OSGi. Pesquisando, percebeu-se que a pessoa de maior competência e domínio no Brasil sobre o tema era Filipe Portes, graduado em Ciências da Computação pela Universidade Paulista, com mais de oito anos de experiência em Desenvolvimento e Arquitetura de Sistemas Java para *web*. Este ministrou palestras sobre o tema no TDC[2] em 2012 e 2014, onde apresentou uma visão ampla do OSGi explicando desde o paradigma da modularização e suas características até os detalhes e *design patterns* utilizados na implementação Apache Felix.

Foi realizada uma entrevista com Filipe Portes em 1 de agosto de 2015, utilizando o *software* Skype[3]. Através desta, foram sanadas dúvidas sobre a arquitetura modular e sua melhor organização, de forma que facilite ainda mais a compreensão e desenvolvimento utilizando o modelo de arquitetura modular.

Houve o compartilhamento de experiências, projetos e padrões de desenvolvimento, que foram utilizados para obtenção de mais conhecimentos práticos acerca do tema do projeto, além de termos percebido que existiam algumas falhas na arquitetura do *software*.

1.4 Procedimentos

Neste projeto desenvolveu-se uma aplicação que exemplifica os conceitos de uma arquitetura modular, utilizando como principais tecnologias, a linguagem de programação Java e a especificação OSGi, que oferece suporte a modularização de softwares.

Para o desenvolvimento deste projeto, foi definido um cronograma com os principais procedimentos, que foram executados de forma progressiva conforme os resultados obtidos através dos estudos realizados em cada tecnologia.

2 TDC – Abreviação para *The Developers Conference*

3 Skype – software gratuito que permite fazer de chamadas de voz e vídeo, chat de mensagens e o compartilhamento de arquivos. Fonte: https://support.**skype**.com/pt/faq/FA6/o-que-e-o-**skype**. 2015

1.4.1 Prototipação

Através dos resultados obtidos na realização de estudos sobre a especificação OSGi, desenvolveu-se protótipos a fim de verificar seu funcionamento. Foram desenvolvidos diferentes tipos de protótipos, com o objetivo de definir como seria desenvolvido uma aplicação que demonstrasse a modularização de softwares.

Desenvolveu-se protótipos que podem ser executados no ambiente do tipo desktop e web. De acordo com os resultados, definimos que a aplicação desenvolvida para exemplificar a modularização de software seria do tipo web.

A especificação OSGi é implementada por vários frameworks, dentre eles, os protótipos foram criados utilizando o Equinox e Apache Felix. Desta forma, definiu-se como implementação a ser utilizada para o desenvolvimento o framework Apache Felix, devido a sua excelente integração com o GlassFish. Sendo assim, definiu-se também, a utilização do GlassFish como servidor de aplicação, completando assim o escopo back-end para o desenvolvimento de uma aplicação web.

Com o desenvolvimento dos protótipos, baseado nos estudos realizados, observou-se que a tecnologia OSGi oferece suporte a projetos do tipo Java Application, Web Application e Enterprise Java Bean. Com isso, constatou-se a possibilidade de desenvolver módulos de diferentes tipos, que interagem entre si através de interfaces bem definidas, que são disponibilizadas como serviços ou exportadas dentro do contexto OSGi.

Foram desenvolvidos também protótipos de módulos responsáveis por disponibilizar a interface gráfica com o usuário, utilizando as tecnologias HTML, JavaScript, Angular JS, CSS e Bootstrap.

A prototipação foi um procedimento muito importante para o desenvolvimento deste projeto, pois além de se comprovar o funcionamento prático das teorias estudadas, observou-se que o desenvolvimento modular requer um planejamento complexo de sua arquitetura, proporcionando desacoplamento dos módulos e alta coesão, ou seja, os módulos podem ser desinstalados, parados e atualizados em tempo de execução sem afetar outros módulos, além de definir responsabilidades específicas para cada módulo.

1.4.2 Definição do *software*

Após estudos e criação de protótipos, definiu-se uma aplicação básica contendo cinco módulos, para demonstrar uma arquitetura modular utilizando a tecnologia OSGi através do *framework* Apache Felix. O objetivo do *software* é apenas demonstrar o funcionamento da arquitetura assim como características da tecnologia OSGi.

- **Módulo Usuário:** responsável por controlar os cadastros do usuário e autenticação dos mesmos no sistema;
- **Módulo Clientes:** responsável por controlar os cadastros de clientes, com integração ao módulo financeiro;
- **Módulo Financeiro:** responsável por controlar os lançamentos de contas a pagar e a receber, com integração ao módulo de clientes;
- **Módulo Logs:** responsável por registrar logs do sistema em arquivos, com integração entre todos os outros módulos.
- **Módulo Data Source:** responsável por fornecer e controlar acesso ao banco de dados, com integração entre todos os outros módulos.

A aplicação foi definida para execução no ambiente web, sendo implantada no servidor de aplicação Glassfish. Isso se justifica devido a grande flexibilidade que os sistemas web oferecem, podendo ser executado em qualquer sistema operacional, assim como qualquer dispositivo que tenha suporte ao browser.

1.4.3 Modelagem da arquitetura dos módulos

A arquitetura de um *software* é uma das principais partes no desenvolvimento de uma aplicação. Dessa maneira, criou-se uma arquitetura modular que fornece aos módulos flexibilidade para que possam ser desinstalados, parados e atualizados enquanto o restante do sistema está em funcionamento.

Modelou-se uma arquitetura em que os módulos do sistema são formados por outros módulos menores. Normalmente, cada módulo do software está formado por outros três módulos, são eles, módulo de Visão, que é responsável por conter as telas de interação com o usuário, o módulo API, que contém funcionalidades e interfaces e, que são compartilhadas e expostas como serviços, respectivamente. E por fim o módulo Core, que contém a implementação das interfaces do módulo API e a lógica de negócio da aplicação.

Com os módulos dispostos dessa maneira, o único módulo que gera dependência para outros módulos é o módulo API, sendo assim o módulo de Visão e Core podem ser desinstalados, parados e atualizados sem comprometer o restante do software. Define-se então, que o módulo API é a base principal para os módulos de Visão e Core, desta maneira, o mesmo não deve ser desinstalado do sistema.

Essa arquitetura pode ser entendida melhor de acordo com a figura xxx que demonstra a estrutura definida para os módulos.

*****Colocar uma figura da estrutura padrão de um módulo*****

Essa arquitetura dos módulos foi desenvolvida com base nos estudos, práticas pesquisadas, e de acordo com necessidade da aplicação. Porém isso pode mudar dependendo da aplicação que se deseja desenvolver.

A figura yyy demonstra como foi definida a arquitetura completa de todos os módulos que compõem o software desenvolvido.

*****Colocar uma figura da arquitetura de todos os módulos*****

Através da modelagem dessa arquitetura, foi possível desenvolver um software em que os módulos não dependam diretamente da implementação de outros módulos. Isso se fez devido a criação de interfaces bem definidas que estão nos módulos APIs. Resultando em um software desacoplado e também de alta coesão devido a cada módulo ter sua responsabilidade bem definida.

1.4.4 Desenvolvimento

O *software* desenvolvido está separado por duas partes que possuem responsabilidades específicas, o *back-end* e *front-end*. Os módulos foram divididos de forma que estas partes fiquem bem definidas. O *back-end* é composto pelos módulos que contém regras de negócio, recursos e serviços. Enquanto o *front-end* é responsável por realizar uma interface entre o sistema e o usuário consumindo as funcionalidades do *back-end*.

1.4.4.1 *Back-end*

Basicamente cada módulo do sistema é composto por três módulos, o módulo de Visão, API e Core. No módulo de visão, estão os recursos RESTful e controle de fluxo, no módulo API estão definidos as interfaces que são expostas como serviços e funcionalidades, ambas estão disponíveis para qualquer outro módulo. E por fim, no módulo Core está a implementação das interfaces, assim como toda lógica de negócio e controle da aplicação. Todos esses fatores compõem a parte denominada *back-end* da aplicação.

1.4.4.2 *Front-end*

O *front-end* da aplicação é composto pelas telas que realizam a interface entre o sistema e o usuário. As mesmas foram desenvolvidas utilizando as tecnologias HTML, CSS e JavaScript. Essa parte do sistema está disposta nos módulos de Visão que são projetos do tipo *Web Application*.

1.5 Orçamento

A Tabela 1 demonstra o planejamento dos gastos com o projeto.

ORÇAMENTO DETALHADO DO PROJETO			
1. RECURSOS MATERIAIS			
1.1 Material Permanente: (equipamentos, livros, máquina fotográfica e gravadores, *softwares*, equipamentos de informática, etc.)			
Descrição do Material	Quantidade	Valor (unidade – em reais)	Total R$
Livros	1	52,11	52,11
Subtotal			52,11
1.2 Material de Consumo: (Papéis necessários para impressões, cartuchos de tinta para impressora, pastas, etc.)			
Descrição do Material	Quantidade	Valor (unidade – em reais)	Total R$
Material de papelaria	200	0,10	20,00
Subtotal			20,00
2. SERVIÇOS: (cópias, encadernações, impressos gráficos, despesas de locomoção e estadia, etc.)			
Descrição do Material	Quantidade	Valor (unidade – em reais)	Total R$
Capa dura	2	50,00	100,00
Cópias	50	0,10	5,00
Locomoção	100	2,75	275,00
Impressões	500	0,10	50,00
Subtotal			430,00
2.1 Serviços de equipamentos: (Aluguel de servidores *web* para hospedagem do sistema e compra de domínio)			
Descrição do Material	Quantidade	Valor (unidade – em reais)	Total R$
Servidor de hospedagem	1	140,00	140,00
Compra de domínio	1	30,00	30,00
Subtotal			170,00
3. RESERVA TÉCNICA/ Despesas Operacionais (10% no total do dispêndio)			
Reserva	1	51,71	51,71
Subtotal			51,71
TOTAL	Valor previsto R$	Reserva de gastos	Total
	517,11	51,71	568,82

Tabela 1 - Orçamento do projeto

Após a análise dos objetivos do projeto, foi visto que serão necessários livros para estudo, passagens de ônibus para reuniões, incluso também gastos com material de papelaria, impressões, xerox e encadernações, para assim possibilitar um melhor projeto de pesquisa.

1.6 Cronograma

Os cronogramas demonstrados na Tabela 2 e 3 mostram como serão realizadas as etapas do desenvolvimento da pesquisa.

1.6.1 Primeiro Semestre de 2015

Mês / Tarefas	DEZ	JAN	FEV	MAR	ABR	MAI
Orientação do Pré-projeto	X					
Formulação do Pré-projeto		X				
Pesquisas dos itens do pré – projeto		X				
Fechamento do pré-projeto			X			
Entrega do pré-projeto			X			
Orientação da Introdução, Objetivos e Justificativa			X			
Formulação da Introdução, Objetivos e Justificativas			X			
Fechamento da Introdução, Objetivos e Justificativas			X			
Entrega da Introdução, Objetivos e Justificativas			X			
Estudo da tecnologia OSGi				X		
Orientação do Quadro Teórico				X		
Formulação do Quadro Teórico				X		

Entrega do Quadro Teórico				X		
Orientação do Quadro Metodológico				X	X	
Formulação do Quadro Metodológico					X	
Entrega do Quadro Metodológico					X	
Revisão do projeto para entrega					X	
Entrega dos projetos para qualificação						X
Bancas de qualificação de Projetos						X
Orientações finais dos projetos						X

Tabela 2 - Cronograma do Primeiro Semestre de 2015

1.6.2 Segundo Semestre de 2015

Mês **Tarefas**	**JUN**	**JUL**	**AGO**	**SET**	**OUT**	**NOV**	**DEZ**
Desenvolvimento de protótipos	X						
Estudo de *frameworks* e ferramentas	X						
Desenvolvimento do *software* modularizado		X	X	X	X		
Atualização da pesquisa			X	X			
Análise e discussão de resultados				X			
Pré-banca					X		
Redação final do TCC					X	X	
Defesa pública						X	
Acertos finais para capa dura						X	X
Entrega da capa dura							X

Tabela 3 - Cronograma do Segundo Semestre de 2015

Estas etapas foram e têm em vista serem seguidas minuciosamente, sabendo que poderão sofrer alterações de eventos futuros a data de entrega dessa pesquisa.

REFERÊNCIAS

ANICHE, Maurício. **Test-Driven Development:** Teste e Design no Mundo Real. São Paulo: Casa do Código, 2012.

ANICHE, Maurício. **TDD**. 2014. Disponível em http://tdd.caelum.com.br/. Acesso em 11 de março, 2015.

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.

BAUER, Christian; KING, Gavin. **Hibernate in action**. Greenwich: Manning Publications Co., 2005.

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-einteratividade-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 Mangueira. **Programação em Java**. Santa Catarina: Copyleft Pearson Education, 2008.

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.

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

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.

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

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.

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.

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. Acesso em 08 de março, 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. Acesso em 20 de junho, 2015.

Pivotal Software. **Preface**. 2015. Disponível em http://docs.spring.io/osgi/docs/current/reference/html/preface.html. Acesso em 10 de março, 2015.

Pivotal Software. **Spring Framework**. 2015. Disponível em http://projects.spring.io/springframework/. Acesso em 10 de março, 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.

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.

VAZ, Rodrigo Cardoso. **JUnit – Framework para testes em Java**. Palmas: Centro Universitário de Palmas, 2003.

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.