FELIPE RODRIGUES DO PRADO
JOÃO PAULO NAKAJIMA PEREIRA

MODULARIZAÇÃO DE SOFTWARE

UNIVERSIDADE DO VALE DO SAPUCAÍ
POUSO ALEGRE
2015

FELIPE RODRIGUES DO PRADO
JOÃO PAULO NAKAJIMA PEREIRA

MODULARIZAÇÃO DE SOFTWARE

Pesquisa para desenvolvimento do projeto apresentado à disciplina de TCC 1 do curso de Sistemas de Informação com requisito parcial para obtenção de créditos.

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	*HiperText Markup Language*
IoT	*Internet of Things*
JDBC	Java *Database Connectivity*
JMS	Java *Message Service*
JPA	Java *Persistence* API
JVM	Java *Virtual Machine*
NTT	*Nippon Telegraph and Telephone*
OSGi	*Open Services Gateway initiative*
SQL	*Structured Query Language*
SRA	Systems Research and Applications Corporation
XHTML	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de um *software* não envolve apenas métodos de programação. Para desenvolvê-lo é necessário preocupar-se com planejamento, engenharia, metodologia e tecnologia a ser utilizada. Esses fatores influenciam na manutenção, atualização e expansão do mesmo. Dessa forma, definir tais fatores é fundamental para que o *software* seja flexível a mudanças. Pensando dessa maneira, demonstraremos um modelo de desenvolvimento modular, utilizando a especificação OSGi.

OSGi, abreviação para *Open Services Gateway Initiative*, possibilita o desenvolvimento de software em módulos, disponibilizando o gerenciamento dos mesmos e facilidades na manutenção e expansão da aplicação.

Segundo Fernandes (2009), um sistema modular possui algumas propriedades. Deve ser autocontido, os módulos podem ser incluídos, retirados, instalados ou desinstalados. Outra propriedade é ter alta coesão, um módulo deve cumprir apenas sua finalidade, ou seja, deve fazer somente as funções que lhe foram atribuídas. O baixo acoplamento é outra propriedade muito importante, um módulo não precisa se preocupar com implementações de outros módulos que interagem com ele, além de permitir alterá-lo sem a necessidade de atualizar os outros.

Muitos *softwares* são desenvolvidos de maneira semelhante à modularização, divididos em partes, tendo cada parte uma responsabilidade. Porém não são realmente modularizados, ou seja, não atendem ao conceito de modularização como descrito acima. Podemos citar dois exemplos de projetos de *softwares* que foram desenvolvidos utilizando a especificação OSGi e atendendo a definição de modularização, que são, IDE Eclipse, ferramenta de desenvolvimento de softwares e o GlassFish, servidor de aplicações JEE.

O uso da modularização, com certeza, traz grandes benefícios para o desenvolvimento e manutenção de um *software*. Poder parar parte de uma aplicação para fazer uma manutenção ou poder instalar novas funcionalidades, garantindo que todas as outras partes restantes continuem funcionando normalmente, seria uma característica notável da aplicação.

Fernandes (2009) forneceu uma visão geral sobre o *framework*, mostrando seus benefícios e salientando a importância da plataforma Java possuir um melhor suporte à

modularidade, até demonstrando com um exemplo simples as premissas e vantagens do OSGi.

Mayworm (2010) demonstra a tecnologia OSGi[1] no contexto de aplicações distribuídas, permitindo a disponibilização de seus serviços remotamente, integrando com diferentes *frameworks* de *middleware* para o desenvolvimento de aplicações empresariais.

Malcher (descobrir ano) apresenta um modelo de componentes adaptável para se usar em ambientes distribuídos. Também analisa alguns aspectos como modelo de distribuição, transparência, descoberta de novos módulos disponíveis, desempenho e performance dos mesmos. Afirma ainda que para se escolher entre um modelo de distribuição – *deployment* local ou execução remota – é necessário analisar o contexto e o objetivo da aplicação, pois cada um se adapta melhor a determinadas situações e ambientes de execução.

Portanto, com a especificação OSGi sendo um padrão de desenvolvimento de aplicativos modulares em Java, que oferece benefícios no desenvolvimento, manutenção e atualização do software, nos despertou um grande interesse em aprofundarmos nossos estudos nessa especificação, decidindo desenvolver uma aplicação que exemplifique seu funcionamento em módulos.

1 OSGi – Abreviação para *Open Services Gateway initiative.*

2 OBJETIVOS

Os objetivos desta pesquisa serão demonstrados a seguir.

2.1 Objetivo geral

- Demonstrar o modelo de desenvolvimento modular utilizando a especificação OSGi[2] para aplicações empresariais.

2.2 Objetivo específico

Para atingir o objetivo geral apresentamos os seguintes objetivos específicos:

- Pesquisar as melhores práticas e *frameworks* que contribuem para a produtividade no desenvolvimento modularizado;
- Desenvolver uma aplicação que exemplifique a modularização de *softwares*;
- Obter através dos resultados uma conclusão sobre a viabilidade de desenvolvimento modularizado para empresas.

2 OSGi – Abreviação para *Open Services Gateway initiative.*

3 JUSTIFICATIVA

Diante dos vários segmentos e constantes mudanças empresariais, novos *softwares* são desenvolvidos e precisam estar preparados para acompanhar as modificações que ocorrem nos processos de uma empresa. Dessa forma o desenvolvimento separado em módulos seria uma solução para atender empresas de diferentes setores em vez de criar um *software* para cada segmento. Além disso a utilização de módulos em um sistema torna mais flexível sua manutenção, pois é realizada nos módulos, não afetando o restante da aplicação.

O trabalho terá relevância na visão acadêmica diante da aprendizagem da tecnologia OSGi[3]. Agregar conceitos sobre o desenvolvimento de *software* de forma modularizada. E ainda, por ser pouco utilizada no mercado devido à sua complexidade de aprendizagem, reunir informações e disponibilizar uma documentação com práticas de programação e vantagens que essa tecnologia pode oferecer.

Desenvolvedores de *softwares* procuram sempre a utilização de novas práticas e tecnologias produtivas. O modelo de desenvolvimento modular através da tecnologia OSGi oferece diversos benefícios que ajudam na expansão e manutenção de uma aplicação. Isto também se torna útil para as empresas, pois para novas funcionalidades seriam criados módulos, e a manutenção do sistema seria realizada somente no módulo específico, sem a necessidade de parar o restante do sistema.

3 OSGi – Abreviação para *Open Services Gateway initiative.*

4 QUADRO TEÓRICO

Para que se possa desenvolver qualquer solução, é necessário o uso de algumas ferramentas, teorias e tecnologias. Abaixo serão apresentadas algumas delas usadas no desenvolvimento desta pesquisa, bem como sua utilidade.

4.1 Java

A linguagem de programação Java é composta por coleções de classes existentes nas bibliotecas de classe Java, conhecidas como Java APIs. Classes são partes que consistem uma aplicação Java, compostas por outras partes chamadas métodos, que são responsáveis por realizar tarefas e retornar informações. É possível criar classes para formar uma aplicação Java e também utilizar as coleções de classes do Java API. Portanto, para se desenvolver uma aplicação utilizando essa linguagem, é preciso entender como criar classes próprias que irão compor a aplicação e como trabalhar utilizando as classes do Java API (DEITEL, 2010).

Java foi lançada em 1996 pela Sun Microsystems[4] com a finalidade de desenvolver aplicativos para diferentes plataformas (Windows – Linux – Web – Mobile). Qualquer dispositivo que possua a JVM[5] é capaz de executar um software desenvolvido em Java (CLARO; SOBRAL, 2008).

O objetivo da linguagem Java é uma plataforma para o desenvolvimento de sistemas de médio a grande porte. Uma das muitas vantagens de se utilizar a plataforma Java é a grande quantidade de bibliotecas gratuitas que podem ser utilizadas em diversos tipos de projetos. Como cada linguagem é apropriada para uma determinada área, a utilização de Java é melhor para o desenvolvimento de aplicações que tenham tendência a crescer (CAELUM, 2015).

Para desenvolver aplicações em Java é necessário instalar um Kit de desenvolvimento, o Java *Development Kit* – JDK, o qual pode ser obtido no próprio site da Oracle – empresa

4 Fabricante de computadores, semicondutores e *software* adquirida pela Oracle Corporation em 2009.
5 JVM – Abreviação para Java *Virtual Machine.*

mantenedora da plataforma. Ele é composto de compilador, máquina virtual, bibliotecas e utilitários.(CAELUM, 2015).

Essa linguagem demonstra ser de imprescindível utilidade para nosso projeto, pois além das vantagens citadas anteriormente, oferece a especificação OSGi[6], tecnologia utilizada para o desenvolvimento de *software* modular.

4.2 OSGi

Grandes aplicações enfrentam um problema conhecido no seu desenvolvimento, a complexidade. Uma maneira de resolver esse problema é quebrar o sistema em partes menores, ou módulos. Devido a sua flexibilidade, a linguagem de programação Java permitiu construir sobre sua plataforma um poderoso sistema de módulos, o OSGi, que oferece suporte à construção de sistemas modulares (Fernandes, 2009).

Segundo a OSGi Alliance (2015), a tecnologia Open Services Gateway initiative – OSGi é um conjunto de especificações, o qual segue um modelo de desenvolvimento em que as aplicações são dinamicamente formadas por componentes distintos e reutilizáveis.

A especificação teve início em março de 1999 pela própria OSGi Alliance. Seus principais desafios na época não era desenvolver uma solução para a execução de diferentes versões de um mesmo projeto na mesma aplicação, mas sim de elaborar uma maneira que diferentes componentes que não se conhecem possam ser agrupados dinamicamente sob o mesmo projeto (OSGI ALLIANCE, 2015).

Segundo Knoernschild (2012), para que uma aplicação seja modularizada, seus módulos devem ser instaláveis, gerenciáveis (parar, reiniciar e ser desinstalado sem interromper o restante da aplicação), reutilizáveis (utilizar em outros sistemas), combináveis (combinar com outros módulos), não guardar estado e oferecer uma interface clara.

OSGi Alliance (2015) diz que a API[7] disponibilizada permite o fácil gerenciamento (instalação, inicialização, parada e atualização) dos pacotes, bem como possibilita enumerar os pacotes e utilizar de seus serviços.

De acordo com OSGi Alliance (2015), os utilizadores dessa tecnologia obtém como

6 OSGi – Abreviação para *Open Services Gateway initiative.*
7 API – Abreviação para *Application Programming Interface.*

benefício a redução da complexidade do desenvolvimento de modo em geral, como por exemplo, aumento da reutilização de módulos, incremento da facilidade de codificação e teste, gerenciamento total dos módulos, sem a necessidade de reiniciar a aplicação, aumento do gerenciamento da implantação e detecção antecipada de bugs. Possui como uma de suas aplicações mais populares a IDE Eclipse e o *framework* Spring.

Os *bundles* – como os módulos são chamados no contexto da OSGi[8] – consomem ou disponibilizam serviços. Eles estão organizados de uma maneira que formam a tríade consumidor, fornecedor e registro de serviços, na qual pode-se consumir serviços ou disponibilizá-los. Para a liberação de um novo serviço, deve-se registrá-lo em um catálogo, onde este teria visibilidade por parte de *bundles* externos que consumiriam esses serviços disponibilizados (OSGI ALLIANCE, 2015).

Bartlett (2009) afirma que um *bundle* pode passar por vários estados em seu ciclo de vida. Na Figura 1 é demonstrado cada um desses estados e suas funções.

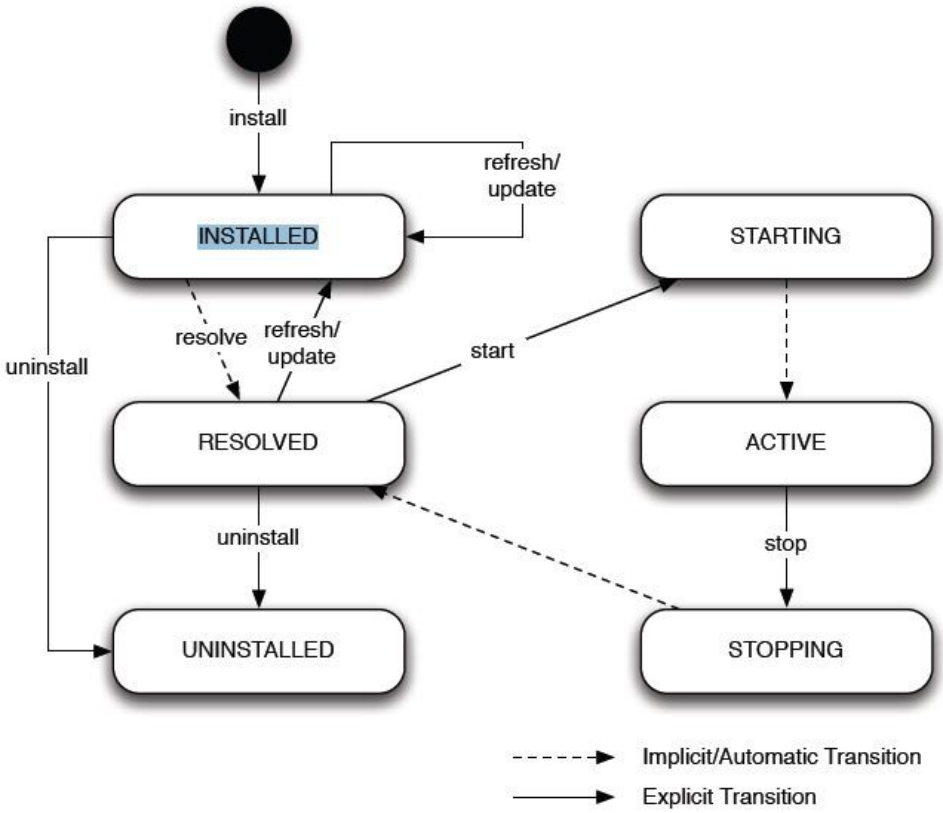

Figura 1 - Estados do *bundle* dentro de um contêiner. Fonte: OSGi In Practice (2009)

8 OSGi – Abreviação para *Open Services Gateway initiative.*

- **Installed:** o *bundle* é instalado com êxito;
- **Resolved:** avalia se o *bundle* está pronto para ser iniciado ou parado, validando informações como *bundles*, pacotes e versões destes, tais como até a versão do java necessária para executar o *bundle*;
- **Starting:** estado de transição, em que nenhum *bundle* vai se encontrar nele, apenas é executado o ativador e logo em seguida, o *bundle* é passado para o estado descrito a seguir;
- **Active:** nesse estado o *bundle* está validado e ativo, somente esperando alguma requisição para ser utilizado;
- **Stopping:** parecido com o estado *Starting*, porém nele é executado o interruptor do *bundle* e este é transferido para o estado *Resolved* novamente;
- **Uninstalled:** quando se ocorre a desinstalação do *bundle*, não se pode transitá-lo para nenhum outro estado e este não é mais representado no *framework*. Se este for reinstalado, ele assume o papel de um novo *bundle*.

Quando um *bundle* é instalado, ele fica armazenado em um meio persistente do *framework* e continua lá até ser explicitamente desinstalado (Fernandes, 2009).

Um aspecto importante é que um *bundle* pode ser executado em qualquer implementação de OSGi. Ou seja, um *bundle* desenvolvido e testado em uma implementação pode ser executado em qualquer outra implementação de OSGi (LUCENA, 2010).

O *framework* é o centro da especificação da plataforma OSGi, fornecendo um ambiente seguro para distribuição e gerenciamento dos *bundles* (Fernandes, 2009).

Fernandes (2009) afirma que o *framework* possui a arquitetura dividida nas camadas conforme demonstrado na Figura 2.

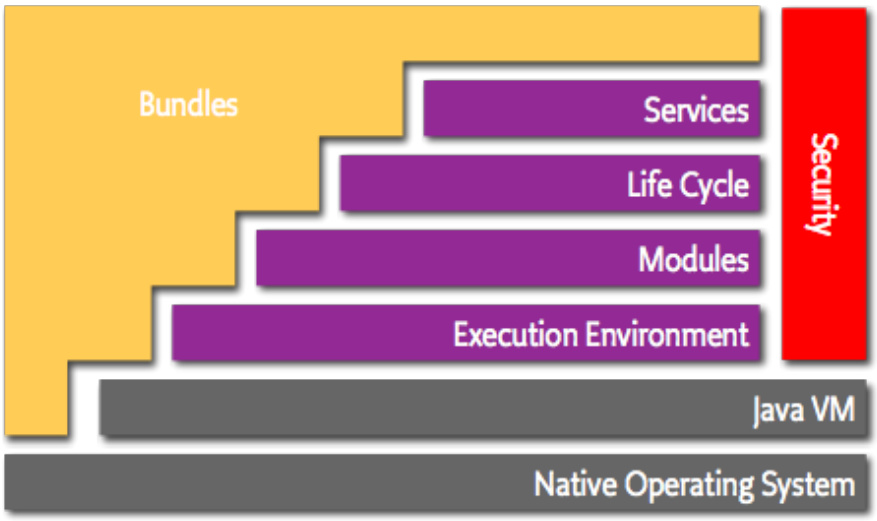

Figura 2 - Estrutura de Camadas. Fonte: OSGi Alliance (2015)

- **Module Layer:** define o conceito de módulo, esclarecendo como um *bundle* pode importar e exportar código;
- **Life Cycle Layer:** define como os *bundles* são dinamicamente instalados e gerenciados no *framework*.
- **Service Layer:** oferece um modelo de desenvolvimento flexível que utiliza o conceito de *Service-Oriented Architecture* (SOA), *publish-find-bind*. Responsável por conectar os *bundles* de maneira dinâmica.

Segundo Bartlett (2009), a OSGi Alliance apenas define uma especificação do framework, por isso existem várias implementações do mesmo nos dias atuais, mas as quatro a seguir merecem destaque por serem as principais e terem seu código livre.

- **Equinox:** é o *framework* mais usado atualmente, sua grande popularidade provém de fazer parte do gerenciador de *plug ins* do Eclipse. Pode ser encontrado em muitos outros softwares, como os da IBM *Lotus Notes* e *Websphere Application Server*. Encontra-se sob a licença Eclipse *Public License* (EPL) e implementa a versão 4.1 das especificações OSGi.
- **Knopflerfish:** é uma implementação bem madura e popular da versão 4.1 da especificação. É desenvolvido e mantido pela empresa sueca Makewave AB, a qual

oferece seu produto tanto em uma versão gratuita sob a licença BSD[9] quanto uma versão comercial com suporte oferecido pela empresa.

- **Felix:** é a implementação mantida pela *Apache Software Foundation*, se encontra na versão 4.1 e possui foco na compactação e facilidade de incorporação do *bundle* na aplicação. Está sob a licença Apache 2.0.
- **Concierge:** é uma implementação bem compacta e altamente otimizada da especificação versão 3. É mais usado para plataformas com recursos limitados, como por exemplo, aplicações móveis e embarcadas. Se encontra sob a licença BSD.

O OSGi[10] especifica um conceito de versionamento de componentes, em que os módulos podem trabalhar com versões diferentes, seguindo uma consistente estrutura de numeração, utilizando três segmentos de números e um segmento alfanumérico, são eles, *major*, *minor*, *micro* e *qualifier*, respectivamente. Exemplo, "1.2.3.beta_1". Esses segmentos também podem ser omitidos, formando por exemplo a versão "1.2". Isso se torna necessário para resolver o problema de incompatibilidade de versões entre módulos (FERNANDES, 2009).

A OSGi Alliance, responsável pela plataforma OSGi trabalha na especificação de uma infraestrutura, que possa distribuir e realizar a comunicação de aplicações e serviços baseados em componentes de forma transparente. Dessa forma, explorar a tecnologia OSGi no ambiente de IoT[11].

9 BSD – Abreviação para *Berkeley Software Distribution.*
10 OSGi – Abreviação para *Open Services Gateway initiative.*
11 IoT – Abreviação para *Internet of Things* (Internet das coisas).

Figura 3 - OSGi IoT. Fonte: OSGi Alliance (2015)

Segundo Gama (2008) “existe uma curva de aprendizado que não vale a pena e nem faz sentido se você está desenvolvendo aplicações que não precisam das vantagens do OSGi”.

4.3 Spring Framework

Spring Framework disponibiliza um modelo de programação e configuração para o desenvolvimento de aplicações corporativas para a plataforma Java, com o objetivo de estruturar a aplicação de maneira que o desenvolvedor tenha foco na lógica de negócios em nível de aplicativo, enquanto o framework gerencia implementações específicas (PIVOTAL SOFTWARE, 2015).

De acordo com DevMedia (2015), o *Spring* é um *framework Open Source* que foi

criado por Rod Johnson no ano de 2002, com o objetivo de simplificar o desenvolvimento utilizando a plataforma Java, possibilitando a criação de softwares que só poderiam ser criados anteriormente através de EJB's[12].

Referente às funcionalidades, podemos destacar como principais a injeção de dependências, programação orientada a aspectos, aplicação web *Spring* MVC, *framework* de serviços web RESTFul e suporte para JDBC[13], JPA[14] e JMS[15] (PIVOTAL SOFTWARE, 2015).

O *Spring Framework* é integrado com a plataforma OSGi[16] através do *Spring Dynamic Modules* que gerencia o ciclo de vida, controla e permite exportar e importar serviços OSGi de forma transparente (PIVOTAL SOFTWARE, 2015).

4.4 Hibernate

Segundo Bauer e King (2005), *Hibernate* é um *framework* de persistência que possui como propósito fornecer uma visão orientada a objetos sobre banco de dados relacionais, ou seja, permite persistir e gerenciar objetos da programação Java para tabelas existentes no banco de dados relacional, de forma simplificada. Para realizar essa conversão de dados, é necessário utilizar arquivos de extensão XML[17], que contém configurações necessárias para fazer o mapeamento de classes em Java para as colunas de uma tabela do Sistema Gerenciador de Banco de Dados ou SGBD. Além dos arquivos XMLs, as classes são mapeadas através de anotações definidas e entendidas pelo *framework*.

Utilizar-se dos benefícios oferecidos pelo Hibernate para o desenvolvimento de uma aplicação que utilize banco de dados relacional e linguagem orientada a objetos é de grande estima, pois segundo Durham e Johnson (1996) no desenvolvimento de *software*, um *framework* é uma estrutura de suporte definida em que um outro projeto de *software* pode ser organizado e desenvolvido. Tipicamente, o *framework* pode incluir programas de apoio, bibliotecas de código, linguagens de *script* e outros *softwares* para ajudar a desenvolver e juntar diferentes componentes do seu projeto.

12 EJB – Abreviação para *Enterprise JavaBeans.*
13 JDBC – Abreviação para Java *Database Connectivity.*
14 JPA – Abreviação para Java *Persistence* API.
15 JMS – Abreviação para Java *Message Service.*
16 OSGi – Abreviação para *Open Services Gateway initiative.*
17 XML – Abreviação para *Extensible Markup Language.*

4.5 PostgreSQL

Segundo Neto (2003 apud Souza et al., 2012, p. 2), o PostgreSQL é um Sistema Gerenciador de Banco de Dados Relacional (SGBDR) que está baseado nos padrões SQL[18] ANSI-92, 96 e 99, possui alta performance, de fácil administração e utilização em projetos por especialistas *Database Administrators* (DBAs) e Projetistas de Sistemas.

PostgreSQL teve origem em um projeto chamado de POSTGRES na Universidade Berkeley, na Califórnia (EUA), em 1986. Sua equipe fundadora foi orientada pelo professor Michael Stonebraker com o apoio de diversos órgãos, entre eles o *Army Research Office* (ARO) e o *National Science Foundation* (NSF). Atualmente, o SGBD encontra-se em sua versão 9.4 estável, contendo todas as principais características que um SGBD pode disponibilizar (MILANI, 2008).

Seu código é livre e há um grupo responsável pela sua validação. Grandes empresas como a Fujitsu, NTT[19] *Group*, Skype, Hub.org, Red Hat e SRA[20] são financiadoras do PostgreSQL que, além disso, recebe doações. É utilizado por multinacionais, órgãos governamentais e universidades. Recebeu vários prêmios como melhor sistema de banco de dados *Open Source* (Souza et al., 2012).

4.6 HyperText Markup Language (HTML)

HyperText Markup Language, é o significado da sigla HTML, que, em português, significa linguagem para marcação de hipertexto. Foi criada por Tim Berners-Lee na década de noventa tornando-se um padrão internacional. De modo geral, o hipertexto é todo o conteúdo de um documento para web, com a característica de se interligar a outros documentos da web através de links presentes nele próprio(Silva, 2011).

Conforme Mozilla Developer Network (2014), a HTML é uma linguagem de marcação que estrutura o conteúdo de um documento da *web*. O conteúdo visto ao acessar uma página através do navegador é estruturado e descrito utilizando a HTML, tornando-se a

18 SQL – Abreviação para *Structured Query Language*.
19 NTT – Abreviação para *Nippon Telegraph and Telephone.*
20 SRA – Abreviação para Systems Research and Applications Corporation.

principal linguagem para conteúdo da web mantida pelo *World Wide Web Consortium* (W3C).

O W3C é uma comunidade internacional liderada pelo criador da web Tim Berners-Lee, é formada por organizações, profissionais e público em geral com o objetivo de conduzir a web ao seu potencial máximo, através do desenvolvimento de padrões e especificações (W3C, 2015).

Segundo Silva (2011), desde a criação da web em 1992, a HTML passou pelas versões HTML, HTML+, HTML 2.0, HTML 3.0, HTML 3.2, HTML 4.0, HTML 4.01 e atualmente se encontra no nível HTML 5. Entre essas versões, o W3C, considera somente as versões HTML 2.0, HTML 3.2, HTML 4.0, HTML 4.01 e HTML 5 oficialmente pois as outras versões são anteriores à criação da W3C.

Mesmo sendo uma linguagem destinada à criação de documentos, a HTML não tem como objetivo definir estilos de formatação, como por exemplo, nomes e tamanhos de fontes, margens, espaçamentos e efeitos visuais. Também não possibilita adicionar funcionalidades de interatividade avançada à página. A linguagem HTML destina-se somente a definir a estrutura dos documentos web, fundamentando dessa maneira os princípios básicos do desenvolvimento seguindo os padrões web (Silva, 2011).

4.7 JavaScript

JavaScript é uma linguagem de programação criada pela Netscape em parceria com a Sun Microsystems[21]. Sua primeira versão, definida como JavaScript 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador Netscape *Navigator* 2.0 (Silva, 2010).

Segundo Silva (2010), o JavaScript tem como finalidade fornecer funcionalidades para adicionar interatividades avançadas a uma página *web*. É desenvolvido para ser executada no lado do cliente, ou seja, é interpretada pelo navegador do usuário. Os navegadores possuem funcionalidades integradas para realizar a interpretação e o funcionamento da linguagem JavaScript.

Conforme Mozilla Developer Network (2015), JavaScript é baseado na linguagem de programação ECMAScript, o qual é padronizado pela Ecma *International* na especificação ECMA-262 e ECMA-402.

21 Fabricante de computadores, semicondutores e *software* adquirida pela Oracle Corporation em 2009.

Entre suas características está ser uma linguagem leve para a execução, interpretada, assíncrona e baseada em objetos com funções de primeira classe. Funções de primeira classe são funções que podem ser passadas como argumentos, retornadas de outras funções, atribuídas a variáveis ou armazenadas em estrutura de dados. Além de ser executada nos navegadores, o JavaScript é utilizada em outros ambientes, como por exemplo, node.js ou Apache CouchDB (Mozilla Developer Network, 2015).

De acordo com Caelum (2015), JavaScript é responsável por aplicar qualquer tipo de funcionalidade dinâmica em páginas *web*. É uma linguagem poderosa, que possibilita ótimos resultados. Podemos citar o Gmail, Google *Maps* e Google *Docs* como exemplos de aplicações *web* desenvolvidas utilizando JavaScript.

4.8 Cascading Style Sheet (CSS)

Cascading Style Sheet, que traduzido para o português significa folhas de estilo em cascata com a abreviação CSS, tem a finalidade de definir estilos de apresentação para um documento HTML[22] (Silva, 2012).

De acordo com a W3C (2015), "*Cascading Style Sheets (CSS) is a simple mechanism for adding style (e.g., fonts, colors, spacing) to Web documents*[23]".

Tim Berners-Lee considerava que os navegadores eram responsáveis pela estilização de uma página web, até que em setembro de 1994, surge como proposta a implementação do CSS. Em dezembro de 1996, foi lançada oficialmente pela W3C as CSS1. O W3C utiliza o termo nível em vez de versão, tendo dessa maneira CSS nível 1, CSS nível 2, CSS nível 2.1 e atualmente CSS nível 3, conhecida também como CSS3 (Silva, 2012).

De acordo com Mozilla Developer Network (2015), a CSS descreve a apresentação de documentos HTML, XML[24] ou XHTML[25]. É padronizada pelas especificações da W3C e já contém seus primeiros rascunhos do nível CSS4.

Enquanto o HTML é uma linguagem destinada para estruturar e marcar o conteúdo de documentos na web, o CSS fica responsável por definir cores, fontes, espaçamentos e outros

22 HTML – Abreviação para *HiperText Markup Language.*
23 Folha de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) para documentos *web* (tradução nossa).
24 XML – Abreviação para *Extensible Markup Language.*
25 XHTML – Abreviação para *Extensible HyperText Markup Language.*

atributos relacionados a apresentação visual da página usando a marcação fornecida pelo HTML como fundamento para aplicação da estilização. Essa separação da marcação e estrutura de um documento do seu estilo de apresentação torna o uso do CSS uma grande vantagem no desenvolvimento de documento para a web (Maujor, 2015).

4.9 Test-Driven Development (TDD)

Test-Driven Development ou em português desenvolvimento guiado por testes é o significado da abreviação TDD. É sem dúvida uma das práticas mais populares utilizadas no desenvolvimento de software, que traz uma ideia bem simples: escrever os testes antes mesmo de escrever o código de produção (ANICHE, 2014).

Automatizar os testes de um software é a atividade de submeter o código desenvolvido para outro software testá-lo, e assim obter o apontamento de erros no código e indicação das falhas para futura correção.

Qualidade do software, fácil manutenção e evolução são uma das vantagens que se conquista testando um software (VAZ, 2003).

Teste é a atividade do ciclo de desenvolvimento de software na qual pode ser observada maior distância entre a teoria (técnicas de teste propostas) e a prática (aplicação destas técnicas) (MYERS, 2004).

Conforme ANICHE (2012), quando a produtividade é medida através do número de linhas de código escrito por dia, o rendimento será menos produtivo. Todavia, se produtividade for a quantidade de linhas de código final sem defeitos escritos por dia, provavelmente o rendimento será mais produtivo ao usar testes automatizados.

Segundo VAZ (2003), para facilitar a execução das rotinas dos testes automatizados foram criados *frameworks* como o JUnit, que através da instalação de seu *plug in* na ferramenta de desenvolvimento Eclipse, disponibiliza uma infraestrutura para se realizar os testes do código.

Dentre as atividades de um processo de desenvolvimento, as atividades de testes têm uma importância fundamental para a garantia de qualidade do software que está sendo desenvolvido, a qual é aplicada no decorrer de todo o projeto.

5 QUADRO METODOLÓGICO

Neste capítulo serão abordados os caminhos definidos para conduzir a pesquisa. Serão apresentados o tipo de pesquisa, seu contexto, bem como os participantes, o orçamento, o cronograma, os instrumentos e os procedimentos para o desenvolvimento da pesquisa.

5.1 Tipo de pesquisa

Pesquisa é um processo desenvolvido com o objetivo de 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 desta pesquisa, será desenvolvida uma pesquisa de abordagem aplicada, a qual é utilizada quando se desenvolve um produto real, com uma finalidade prática, que pode ser aplicado 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 mostrar melhor os resultados obtidos, será desenvolvido um software modularizado com o objetivo de facilitar sua manutenção e aplicação em empresas de diferentes ramos.

5.2 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 torna o seu desenvolvimento e manutenção um processo desgastante e trabalhoso.

Esta pesquisa demonstrará a utilização do *framework* OSGi[26], que possibilita o desenvolvimento do software em módulos, oferecendo melhor organização na sua estrutura, vantagens na manutenção e maior facilidade na expansão do mesmo, já que é construído em módulos.

Softwares modularizados oferecem maior flexibilidade para atender empresas de diferentes áreas. Através do OSGi, os módulos podem ser desenvolvidos de forma independente, sendo assim, o software pode disponibilizar módulos específicos para cada empresa. Dessa maneira será desenvolvido um software que pode ser adotado 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 trazer as metodologias, engenharias, técnicas e tecnologias para o âmbito de softwares comerciais e prover uma base para a facilitação na manutenção e expansão do mesmo.

5.3 Participantes

Participam deste projeto os acadêmicos do curso de bacharelado em Sistemas de Informação da Universidade do Vale do Sapucaí – UNIVAS, os alunos Felipe Rodrigues do Prado e João Paulo Nakajima Pereira sob a orientação do professor Márcio Emílio Cruz Vono de Azevedo.

Felipe Rodrigues do Prado, no ano de 2010, formou-se em técnico em informática pelo Serviço Nacional de Aprendizagem Comercial (SENAC) e atualmente é estagiário no Inatel *Competence Center* (ICC), atuando como desenvolvedor de software para soluções empresariais.

João Paulo Nakajima Pereira exerce a profissão de analista de suporte na empresa Automação e Cia, a qual presta serviços de atendimento ao cliente e suporte em *softwares* de automação de empresas.

Márcio Emílio Cruz Vono de Azevedo é bacharel em Engenharia Elétrica com ênfase

26 OSGi – Abreviação para *Open Services Gateway initiative.*

em Eletrônica pelo Instituto Nacional de Telecomunicações – INATEL, Mestre em Ciência da Computação pela Universidade Federal de Itajubá – UNIFEI, é professor nas instituições de ensino INATEL na disciplina de Orientação a Objetos e na UNIVAS nas disciplinas de Engenharia de Software, Linguagem de Programação e Sistemas Distribuídos. É diretor de desenvolvimento de *software* na empresa TM Tecnologia da Informação Ltda.

O interesse dos participantes nesta pesquisa é conhecer e demonstrar a potencialidade de um sistema modularizado de atingir novas áreas empresariais, analisando os benefícios e desafios que poderão ser encontrados, além de obter mais conhecimentos sobre essa nova prática de modularizar o software, permitindo assim, como consequência final o alcance de maior vantagem competitiva para as empresas, pois estas usufruiriam de menor custo(tanto financeiro quanto de tempo) de manutenção do software, propiciando maior confiabilidade e disponibilidade do mesmo.

5.4 Instrumentos

Para desenvolver esta pesquisa será necessária 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 colectivamente. Além disso, decisões tomadas coletivamente, com representantes das diferentes entidades interessadas, serão mais facilmente aceitas por todos.

Os dados serão obtidos a partir de livros, artigos, documentações e reuniões online com pessoas que possuem conhecimento e experiência nesse tema. Após aplicados estes instrumentos será possível analisar e organizar as informações.

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

5.5 Procedimentos

A ideia deste trabalho é elaborar um *software* modularizado que permita a fácil manutenção e expansão, auxiliando no suporte e futura ampliação do mesmo. Desta forma, para o desenvolvimento desta pesquisa, tornou-se necessário a distribuição deste em vários procedimentos citados a seguir.

- Definir o *software* a ser desenvolvido para demonstrar as funcionalidades e vantagens de um software modularizado;
- Levantar dos requisitos;
- Modelar e elaborar o banco de dados;
- Mapear o banco de dados seguindo a metodologia objeto relacional;
- Fazer o *design* e criar as *interfaces* de interação com o usuário;
- Implementar a lógica do negócio;
- Estruturar e ligar funcionalmente os módulos do sistema;
- Documentar o projeto.

Após a conclusão desses procedimentos deverá ser feita a análise para a verificação do funcionamento e desempenho tecnologia no âmbito de aplicações empresariais, averiguando os benefícios e o ponderando a utilização dessa nova forma de se desenvolver *softwares*.

5.6 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			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
3. RESERVA TÉCNICA/ Despesas Operacionais (10% no total do dispêndio)			
Reserva	1	50,21	50,21
Subtotal			50,21
TOTAL	Valor previsto R$	Reserva de gastos	Total
	502,11	50,21	552,32

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.

5.7 Cronograma

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

5.7.1 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			
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

5.7.2 Segundo Semestre de 2015

Mês / Tarefas	JUN	JUL	AGO	SET	OUT	NOV	DEZ
Estudo da tecnologia OSGi e desenvolvimento de protótipos	X						
Estudo do framework Spring	X						
Desenvolvimento do software modularizado	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 têm em vista serem seguidas minuciosamente, sabendo que poderão sofrer alterações.

6 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.

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

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

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

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.

DURHAM, Alam; 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.

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.

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.

MYERS, Glenford John. **The Art of Software Testing**. Hobokey: Word Association, 2004.

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

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.