Universidade Norte do Paraná

SISTEMA DE ENSINO PRESENCIAL CONECTADO
ESPECIALIZAÇÃO EM TECNOLOGIA PARA APLICAÇÕES WEB

MARCOS FERREIRA

MISSÃO CUMPRIDA:

Aplicativo Web com tecnologias Java para gerenciamento de tarefas

Maravilha
2013

MARCOS FERREIRA

MISSÃO CUMPRIDA:

Aplicativo Web com tecnologias Java para gerenciamento de tarefas

Monografia apresentada à Universidade Norte do Paraná - UNOPAR, como requisito parcial para a obtenção do titulo de especialista em tecnologia para aplicações web.

Orientador: Prof. Bruno Pereira Bodelão

Maravilha
2013

MARCOS FERREIRA

MISSÃO CUMPRIDA:

Aplicativo Web com tecnologias Java para gerenciamento de tarefas

Monografia aprovada, apresentada à Universidade Norte do Paraná - UNOPAR, como requisito parcial para a obtenção do titulo de especialista em tecnologia para aplicações web.

_____

Professor Membro 1
Universidade Norte do Paraná

_____

Professor Membro 2
Universidade Norte do Paraná

_____

Professor Membro 3
Universidade Norte do Paraná

Maravilha, _____ de _____ de 2013

À minha família pelo apoio e o amor recebido.

AGRADECIMENTOS

Agradeço a DEUS por mais uma oportunidade de aprendizagem e crescimento.

Agradeço a minha família pelo apoio e paciência dedicados durante o período desta especialização.

FERREIRA, Marcos. Missão Cumprida: Aplicativo Web com tecnologias Java para gerenciamento de tarefas. 2013. 33 f. Monografia (Especialização em Tecnologia para Aplicações Web) - Sistema de Ensino Presencial Conectado, Universidade Norte do Paraná, Maravilha, 2013.

RESUMO

Os processos de delegação de tarefas por parte do chefe da Secretaria Administrativa do 11º Batalhão de Polícia Militar da cidade de São Miguel do Oeste eram até então realizados utilizando a delegação direta verbal ou meramente informativa via e-mail, ocasionando uma gestão menos eficiente dos trabalhos administrativos realizados naquela secretaria, ocorrendo por vezes ruídos na dinâmica de comunicação da equipe. Foi proposta a atenuação deste transtorno com a centralização das tarefas em um aplicativo web, o qual o presente estudo tem por objetivo desenvolver. A aplicação fará uso de tecnologias relacionadas à linguagem de programação Java e centralizando a criação, delegação, edição e encerramento de tarefas por parte dos profissionais da referida secretaria.

Palavras-chave: Aplicativo. Web. Java. Gerenciamento.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AJAX - Asynchronous Javascript and XML (Javascript Assíncrono e XML)

BPM - Batalhão de Polícia Militar

CSS - Cascading Style Sheets

HIBERNATE - Framework para o mapeamento objeto-relacional

HTML - HyperText Markup Language (Linguagem de Marcação de Hipertexto)

J2EE - Java 2 Enterprise Edition

Java - Linguagem de programação orientada a objeto

Java Script - Linguagem de programação interpretada

JSF – Java Server Faces

MVC – Model-View-Controller (modelo de arquitetura de software)

MySQL - Sistema de Gerenciamento de Banco de Dados

OO – Orientação a Objeto

POO – Programação Orientada a Objeto

PRIMEFACES – Suíte de componentes JSF

SA - Secretaria Administrativa

SGBD - Sistema Gerenciador de Banco de Dados

SPRING - Framework open source para a plataforma Java

UML - Unified Modeling Language – Linguagem de Modelagem Unificada

XHTML - Extensible Hypertext Markup Language

XML - Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A característica polivalente dos trabalhos realizados atualmente nas seções dos órgãos públicos, especificamente na Secretaria Administrativa (SA) do 11º Batalhão de Policia Militar (11º BPM) do município de São Miguel do Oeste, foi o fator que motivou o desenvolvimento do aplicativo web descrito nesta monografia.

A realização deste trabalho leva em considerando a burocratização dos processos administrativos, como licitações e orçamentos, utilizados para que sejam possíveis ações como a manutenção física do quartel, aquisição e manutenção de equipamentos e viaturas, bem como a gestão dos convênios existentes. Em contrapartida, observa-se a constante defasagem no quadro de pessoal trabalhando na SA, fato que ocasiona o acumulo de funções e tarefas realizadas por cada auxiliar.

São dois os tipos de atores identificados para o desenvolvimento do sistema, sendo o primeiro o chefe da SA, que gerencia o desenvolvimento dos trabalhos realizados e autoriza, ou não, a realização de procedimentos da área administrativa e o segundo os auxiliares que são os profissionais que executam as tarefas (missões) delegadas pelo chefe.

O nome do aplicativo web, proposto nesta monografia, será MISSÃO CUMPRIDA o qual pretende auxiliar o gerenciamento das tarefas desenvolvidas na SA, unificando o local onde elas serão criadas, executadas e finalizadas. O chefe criará as tarefas e indicará o auxiliar que deverá cumpri-la, por sua vez o auxiliar visualizará suas tarefas e deverá executá-las, posteriormente o chefe revisará a tarefa, podendo concluí-la ou encaminhá-la ao auxiliar para solução de possíveis pendências.

2 DESENVOLVIMENTO

Ao final do presente trabalho é pretendido obter-se um aplicativo *web* para o gerenciamento de tarefas, o qual terá como tecnologias empregadas tecnologias como: *JSF*, *PRIMEFACES*, *MYSQL*, *UML*, *AJAX*, *JAVA*, *CSS*, *JAVA Script*, *XHTML*, *SPRING*, *HIBERNATE*, as quais quando usadas em conjunto permitirão o desenvolvimento de uma aplicação *JAVA Enterprise*, aos moldes da plataforma *J2EE*.

Embora o desenvolvimento do MISSÃO CUMPRIDA tenha feito uso das tecnologias acima mencionadas, a descrição da instalação e configuração dos softwares utilizados estão fora do escopo do presente trabalho, podendo em alguns momentos serem descritos de forma sucinta quando tal explicitação se fizer necessária.

2.1 TECNOLOGIA

As tecnologias utilizadas na construção do aplicativo web MISSÃO CUMPRIDA têm por base softwares baseados na tecnologia *JAVA*, os quais associados formam um sistema robusto e confiável.

2.1.1 Programação orientada a objetos (POO)

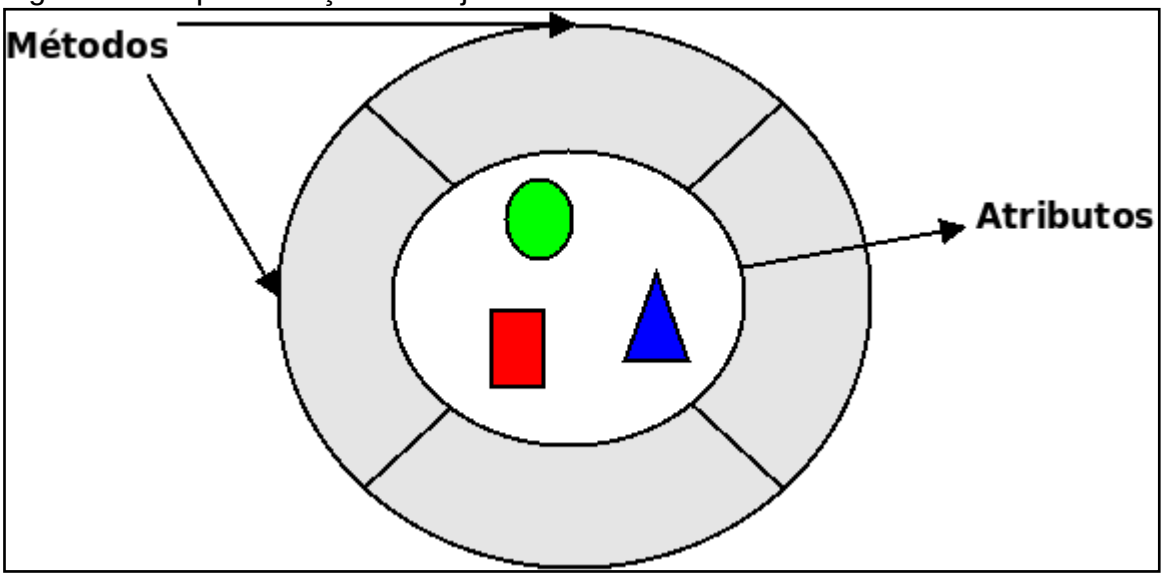
> Durante varias décadas, a construção de programas foi abordada sob um ponto de vista: decomposição algorítmica. Assim, grandes programas são divididos em módulos e estes, em funções. O paradigma orientado a objetos (OO) oferece uma alternativa diferente para particionar um problema, que não se baseia apenas em algoritmos (BOOCH, 1994).

A POO trabalha com a ideia de abstração, que consiste em abstrair o problema, fazendo uma analogia em paralelo com o mundo real, o que permite decompor um grande problema em problemas menores, mais pontuais e de melhor entendimento, os quais serão mais fáceis de resolver e após a resolução dos mesmos consequentemente ocorrerá à solução do problema maior.

Na POO todos os componentes do sistema são representados por

objetos, mostrado na figura 1. Essa representação é obtida por meio de classes e atributos, onde as classes representarão os objetos em si e os atributos suas características.

Figura 1 – Representação de objeto em POO



Fonte: o autor

Os objetos são instanciados (criados) a partir das classes. Classes são representações de um grupo de objetos, agrupados por compartilharem características e atributos, operações e relações semelhantes. A instanciação de objetos e a troca de mensagens entre si, com ou sem passagem de parâmetros, é que permite a execução e interação dos sistemas baseados em POO.

Além da abstração, é conteúdo basilar da POO o encapsulamento, que consiste em delegar o acesso direto aos atributos de um objeto através de métodos, ou seja, manter acesso privado aos atributos e acesso publico aos métodos.

A POO ainda revela os conceitos de herança e polimorfismo como pontos básicos para sua utilização. Herança consiste no compartilhamento de características comuns a diversas classes, que revelem comportamentos comuns ou semelhantes, os quais podem ser abstraídos e colocados em uma classe base (superclasse). Já o polimorfismo, que significa muitas formas, representa as diversas formas que objetos e métodos podem assumir a partir de uma superclasse, fazendo uso de objetos instanciados a partir de uma árvore de herança, através de referências do tipo de uma superclasse na hierarquia.

Para elaboração do sistema proposto nesta monografia será utilizado à linguagem de programação *JAVA*, a qual é orientada a objeto e atenderá completamente as necessidades no tocante a codificação.

2.1.2 Banco de dados

A informação acaba sendo o ponto central e de maior interesse quando se refere a sistemas de informação, para tanto, foi escolhido o MySQL para elaboração e manutenção do banco de dados, o qual persistirá as informações que ao aplicativo web serão fornecidas.

Segundo Câmara (2009, p. 12) banco de dados é um conjunto de dados armazenados de forma organizada que permita sua recuperação posterior.

O MySQL é conceituado como sendo um programa gerenciador de banco de dados (SGBD), relacional e de código-fonte aberto, sendo o mais popular no mundo, nesta categoria (LUCKOW E MELO, 2010, p.63).

O uso de uso de um SGBD possibilita algumas vantagens quando da sua utilização como: controle de redundância, compartilhamento de dados, restrição de acesso não autorizado, representação de relacionamentos complexos entre dados, tolerância a falhas, *backup* e restauração e múltiplas interfaces (CÂMARA, 2009, p.18).

O uso de banco de dados relacional com o método de orientação a objetos apresenta dificuldades em sua implementação, principalmente no desempenho, situação contornada com uso do *HIBERNATE*, que consiste em um *framework* desenvolvido em linguagem JAVA para mapeamento de banco de dados relacionais, como o MySQL, sendo indicado para aplicativos JAVA que utilizem o modelo MVC no desenvolvimento.

2.1.3 Java Server Faces (JSF) e MVC

Segundo Greco (2010):

> JSF (Java Server Faces) é uma tecnologia que incorpora características de um framework MVC (Model-View-Controller) para WEB e de um modelo de interfaces gráficas baseado em eventos. Por basear-se no padrão de projeto MVC, uma de suas melhores vantagens é a clara separação entre a visualização e regras de negócio (modelo).

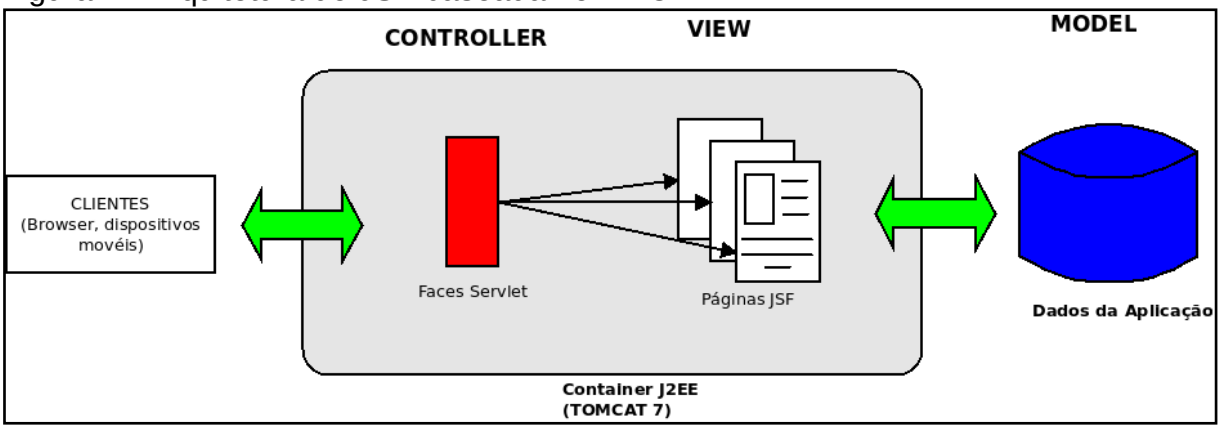
Considerando a qualidade como premissa para o desenvolvimento do aplicativo proposto, bem como as facilidades encontradas na linguagem de programação JAVA, o MISSÃO CUMPRIDA será desenvolvido utilizando-se o *framework web* JavaServer Faces (JSF), o qual comprova sua qualidade por ser o *framework web* eleito como padrão para uso na plataforma J2EE, a qual permite uma melhor componentização das aplicações, sendo utilizada no desenvolvimento de aplicações empresariais.

Sobre o MVC Greco (2010) afirma que:

> A ideia do padrão MVC é dividir uma aplicação em três camadas: modelo, visualização e controle.
> O modelo é responsável por representar os objetos de negócio, manter o estado da aplicação e fornecer ao controlador o acesso aos dados. A visualização representa a interface com o usuário, sendo responsável por definir a forma como os dados serão apresentados e encaminhar as ações dos usuários para o controlador. Já a camada de controle é responsável por fazer a ligação entre o modelo e a visualização, além de interpretar as ações do usuário e as traduzir para uma operação sobre o modelo, onde são realizadas mudanças e, então, gerar uma visualização apropriada.

Considerando que o JSF trabalha naturalmente com a utilização do padrão MVC, o mesmo será implementado no projeto do aplicativo, objeto deste trabalho e tem sua dinâmica de execução representada na figura 2.

Figura 2 - Arquitetura do JSF baseada no MVC



Fonte: o autor

A figura 2, acima, apresenta a arquitetura do padrão MVC, no uso do JSF, que separa a aplicação em três responsabilidades (*model*, *view* e *controller*), onde o *model* (modelo) fica responsável pela representação dos objetos de negócio,

e manter o estado do aplicativo e ainda fornecer ao *controller* (controlador) acesso aos dados. A *view* (visualização) é a camada que fica responsável pela interface do usuário, ou seja, a página *web* onde o usuário fará interação com a aplicação. Esta camada define a forma como os dados serão mostrados e encaminha as interações do usuário para o controlador. O *controller* (controlador) é a camada responsável pela ligação entre o modelo e a visualização, interpretando solicitações do usuário, traduzindo em operações no modelo e devolvendo a visualização adequada à referida solicitação.

Ainda na figura 2 é possível perceber a presença do *Faces Servlet* que é um *servlet* que realiza o controle através de arquivos XML para configuração e por manipuladores de ações e *listeners* (observadores de eventos). O *Faces Servlet* recebe a requisição do usuário, redireciona para o modelo e retorna uma resposta. Arquivos de configuração guardam informações sobre mapeamentos de ações e regras de navegações entre páginas web.

O *model* (modelo) é a camada que representa os objetos de negócio, os quais executam uma determinada lógica de negócio quando recebem dados provenientes da camada de visualização.

2.1.4 PRIMEFACES e AJAX

Para o desenvolvimento dos componentes de interação com o usuário será feito uso do PRIMEFACES que implementa a especificação dos componentes JSF, o qual pode ser associado a técnicas como AJAX e tendo como base de programação a linguagem JAVA.

O PRIMEFACES é uma biblioteca de componentes, com mais de 90 componentes para uso em projetos que implementem o JavaServer Faces. Sua instalação consiste em baixar seu arquivo binário, o qual contem nomenclatura e extensão semelhante à PRIMEFACES-2.x.jar e coloca-lo junto à pasta WEB-INF\lib do projeto para posterior configuração do arquivo web.xml e estará pronto para uso (LUCKOW e MELO, 2010, p.313).

O PRIMEFACES trabalha de forma padrão com a metodologia AJAX (*Asynchronous JavaScript and XML*) a qual possibilita os componentes realizarem chamadas assíncronas nas requisições ao servidor, tonando a experiência de navegação do usuário mais interativa, consequentemente mais interessante.

2.1.5 XHTML , CSS e JavaScript

O desenvolvimento usando JSF utiliza ainda outras tecnologias, como é o caso do XHTML o qual significa *extensible hypertext markup language* (Linguagem Extensível de Marcação de Hipertexto) e trata-se de um arquivo HTML que pode ser lido por qualquer navegador web e é também um arquivo XML (extensible markup language) válido, ou seja, é uma HTML (linguagem de marcação) modificado para suportar recursos do XML.

Segundo Luckow e Melo (2010, p.305) *Cascading Style Sheets*, ou CSS, é uma linguagem de estilos utilizada em páginas web para personalização da definição gráfica e como a informação será apresentada.

O uso do CSS em um arquivo próprio, separado do XHTML, permite padronização e uma melhor manutenção do código da aplicação.

JavaScript é uma linguagem de programação interpretada atuando no navegador do usuário, a qual permite ações dinâmicas, como validação de formulários e interações do usuário com a página web.

2.1.6 UML

UML significa *Unified Modeling Language* (Linguagem Unificada de Modelagem) é, segundo Santos (2009/2, p.58), um modelo de linguagem padronizado, baseada em diagramas, para a análise e projeto orientado a objetos.

São nove os tipos de diagramas principais: de use case, de classe, de objeto, de estado, de sequencia, de colaboração, de atividade, de componente e o execução.

2.1.7 SPRING FRAMEWORK

Para o desenvolvimento do aplicativo MISSÃO CUMPRIDA, na questão de segurança, será utilizado o *framework SPRING Security*, o qual é um

dos vários projetos disponíveis na suíte *SPRING Framework*, que é desenvolvida em JAVA, de código-fonte aberto e facilita o desenvolvimento J2EE (Luckow e Mello, 2010, p.232).

2.2 LEVANTAMENTO DE REQUISITOS

Segundo afirma Pressman (2009, p.117) a engenharia de requisitos estabelece uma base sólida para o projeto e a construção. Sem ela, o software resultante tem uma alta probabilidade de não satisfazer as necessidades dos clientes.

Porém deve-se considerar o fato de que a engenharia de requisitos não é linear, ao contrário, deve ser adaptada às necessidades do processo, do projeto e do pessoal envolvido no trabalho, podendo ocorrer uma abordagem abreviada, porém todo requisito deve ser entendido para que seja resolvido (PRESSMAN, 2009, p.117).

A concepção do aplicativo web MISSÃO CUMPRIDA, iniciou de forma casual, do questionamento do chefe da Secretaria Administrativa (SA) sobre a possibilidade do desenvolvimento de um software que possibilitasse o registro e acompanhamento das missões (tarefas) distribuídas aos seus auxiliares.

No modelo de trabalho atual o chefe da SA realiza o gerenciamento e as distribuições de tarefas aos colaboradores com o auxilio de agendas, bilhetes, envio de *e-mails*, pessoalmente ou via telefone, o que acaba fragmentando o processo e consequentemente prejudicando o acompanhamento das atividades e tornando muito difícil o gerenciamento das tarefas, sejam elas novas, em andamento ou concluídas.

A metodologia de levantamento de requisitos que melhor se enquadrou no desenvolvimento foi a de Coleta Colaborativa de Requisitos, onde ocorrem reuniões com as pessoas interessadas (*stakeholders*) e desenvolvedoras para trabalharem em conjunto a fim de especificar requisitos para a solução de problemas levantados (PRESSMAN, 2009, p.125).

Restou proposto um aplicativo web que possibilite o cadastro e acompanhamento das tarefas delegadas pelo chefe da SA aos colaborados.

Destas reuniões surgiram os requisitos funcionais e não funcionais,

conforme quadros 1 e 2, respectivamente.

2.2.1 Requisitos funcionais

Os requisitos funcionais indicam o que o sistema deve fazer, descrevendo suas funções detalhadamente, suas entradas e saídas, bem como as possíveis exceções (SOMMERVILLE, 2007, p.81).

O quadro 1 exibe os principais requisitos funcionais do MISSÃO CUMPRIDA os quais serão descritos e classificados em sua prioridade como: Essencial, Importante e Desejável.

Quadro 1 – Requisitos funcionais

Id	**Descrição**	**Prioridade**
RF01	O sistema deve cadastrar usuários, os quais usando seu login e senha poderão autenticar-se e utilizar a aplicação.	Essencial
RF02	O sistema deve editar informações sobre os usuários quando ocorrerem alterações.	Essencial
RF03	O sistema deve excluir usuários, quando a permanência de seu registro não for mais necessária para a aplicação.	Importante
RF04	O sistema deve cadastrar graduações, pois cada usuário tem uma graduação específica.	Essencial
RF05	O sistema deve editar graduações, pois podem ocorrer alterações nos detalhes de cadastro das graduações.	Importante
RF06	O sistema deve excluir graduações, pois uma graduação pode deixar de ser utilizada.	Desejável
RF07	O sistema deve criar missões que devem permanecer editáveis até seu encerramento.	Essencial
RF08	O sistema deve editar missões enquanto não for concluída e encerrada.	Essencial
RF09	O sistema deve encerrar missões tão logo ela seja considerada concluída.	Essencial

Fonte: o autor

2.2.2 Requisitos não funcionais

Requisitos não funcionais são os não estão diretamente relacionados às funções específicas fornecidas pelo sistema, mas se referem às propriedades do mesmo, como confiabilidade, tempo de resposta, espaço de armazenamento ou ainda trabalhar com restrições (SOMMERVILLE, 2007, p.82).

Os principais requisitos não funcionais do MISSÃO CUMPRIDA são relacionados no quadro 2 e também serão classificados em sua prioridade como: Essencial, Importante e Desejável.

Quadro 2 – Requisitos não funcionais

Id	**Descrição**	**Prioridade**
RNF01	O sistema deve usar algoritmo de HASH MD5 para criação da senha do usuário.	Essencial
RNF02	O sistema deve prover ao usuário uma interface simples e intuitiva que seja de fácil navegação para facilitar seu uso.	Essencial
RNF03	O sistema deve apresentar menus e mensagens em língua portuguesa.	Essencial
RNF04	A implementação do sistema deve empregar uma arquitetura baseada em MVC (Modelo-Visão-Controle).	Essencial
RNF05	Para implementação do sistema deve ser utilizado a linguagem Java com o padrão J2EE.	Importante
RNF06	O aplicativo deverá estar disponível aos usuários 24 horas por dia e 7 dias por semana.	Essencial
RNF07	O sistema deve criar missões que devem permanecer editáveis até seu encerramento.	Essencial

Fonte: o autor

2.2.3 Validação de Requisitos

A validação de requisitos demonstra sua importância frente aos elevados custos que de retrabalho quando da ocorrência de erros no documento de requisitos, descobertos no desenvolvimento ou logo que o sistema estiver em operação. Quando mais adiantado o processo de desenvolvimento, maior será o custo para realizar alterações nos requisitos.

O documento de requisitos permite o gerenciamento das mudanças dos requisitos que ocorram ocasionadas por uma melhor compreensão das necessidades do software ou mudanças no próprio ambiente organizacional do sistema.

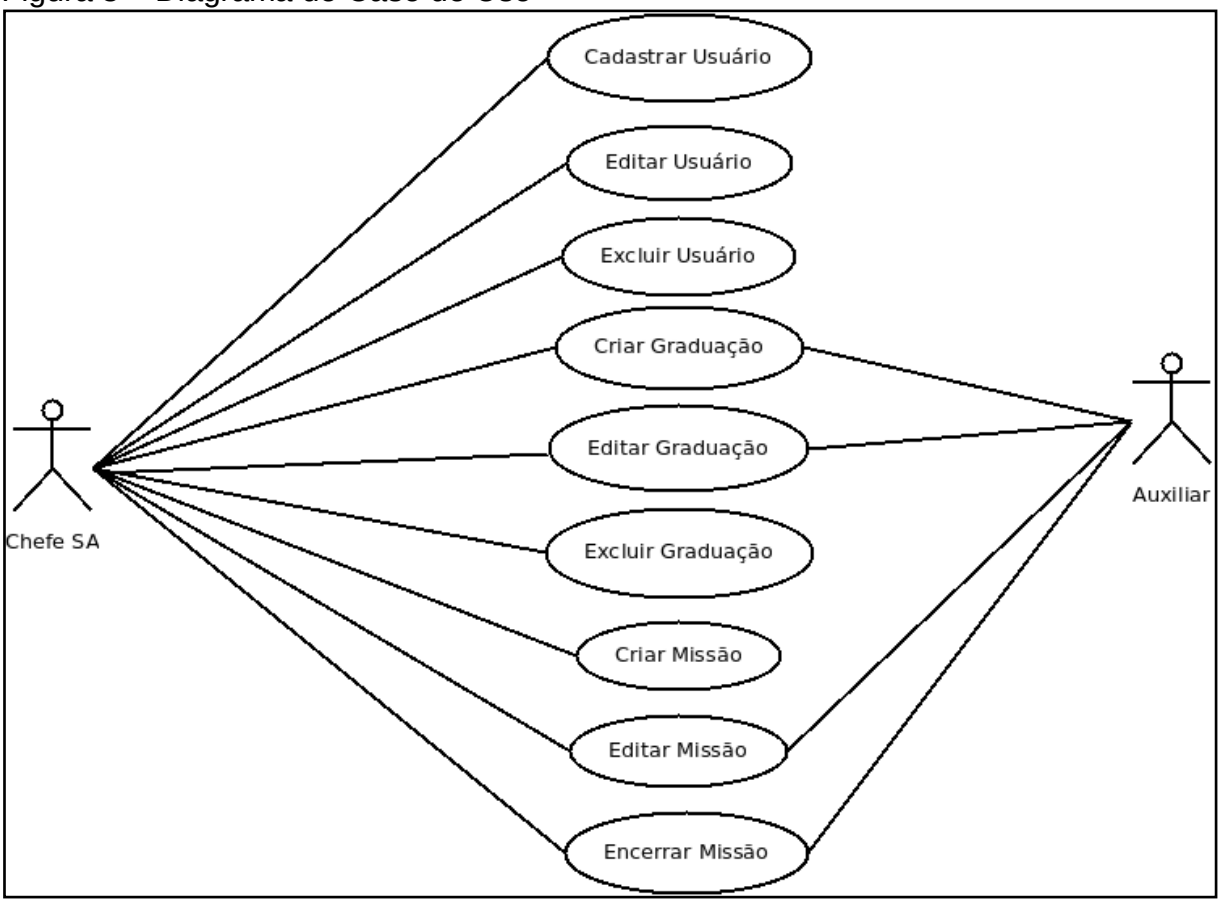
2.3 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso do MISSÃO CUMPRIDA foi desenvolvido usando a ferramenta de modelagem DIA a qual possibilita o desenho

de diagramas estruturados, sendo um software livre, podendo ser baixado no site https://wiki.gnome.org/Apps/Dia/Download.

Segundo Santos (2009, p. 59), os atores representam o papel de uma entidade externa ao sistema, podendo ser tanto um usuário como um hardware ou outro sistema. O diagrama de caso de uso do aplicativo proposto nesta monografia é representado pela figura 3.

Figura 3 – Diagrama de Caso de Uso



Fonte: o autor

O chefe da secretaria administrativa, representado pelo ator Chefe SA, tem acesso a todas as funcionalidades do sistema, podendo criar, editar e excluir usuários e graduações, bem como criar, editar e finalizar missões (tarefas).

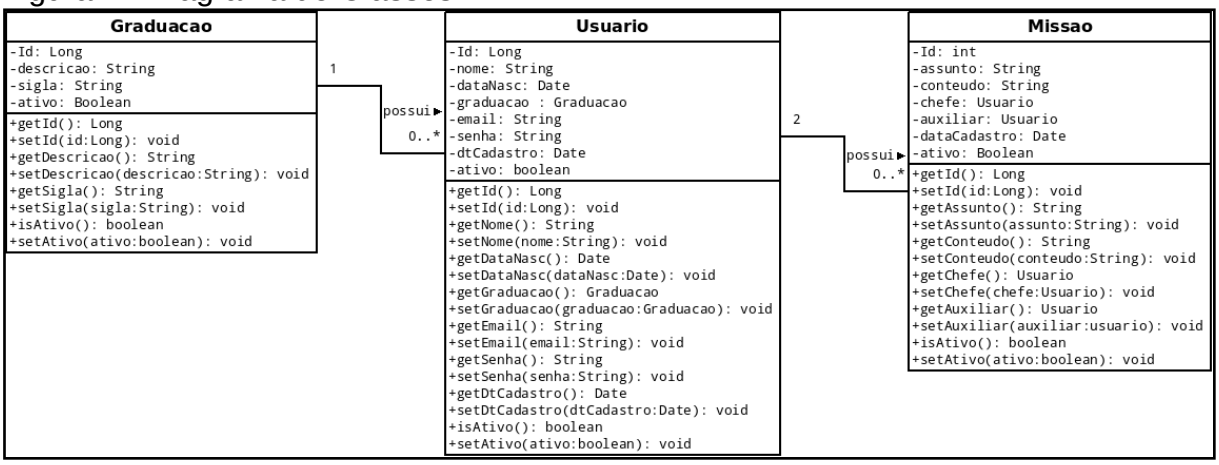
Já o auxiliar, subordinado ao chefe da SA e representado pelo ator Auxiliar, interage com o sistema apenas em requisitos funcionais específicos, como edição e encerramento de missões.

2.4 DIAGRAMA DE CLASSE

Segundo Santos (2009, p. 60), o diagrama de classes representa a estrutura estática das classes de um sistema e que serão gerenciadas pela aplicação modelada. As classes podem relacionar-se entre si através de associações quando se conectam, por dependência quando uma depende ou usa outra classe, por especialização quando uma classe é uma especialização de outra ou por pacotes quando as classes são agrupadas por características semelhantes.

Conforme a figura 4 o sistema MISSÃO CUMPRIDA apresenta com classes principais: Usuario, Graduacao e Missao, apesar de possuir diversas outras classes para possibilitar a operacionalização do sistema.

Figura 4 - Diagrama de Classes



Fonte: o autor

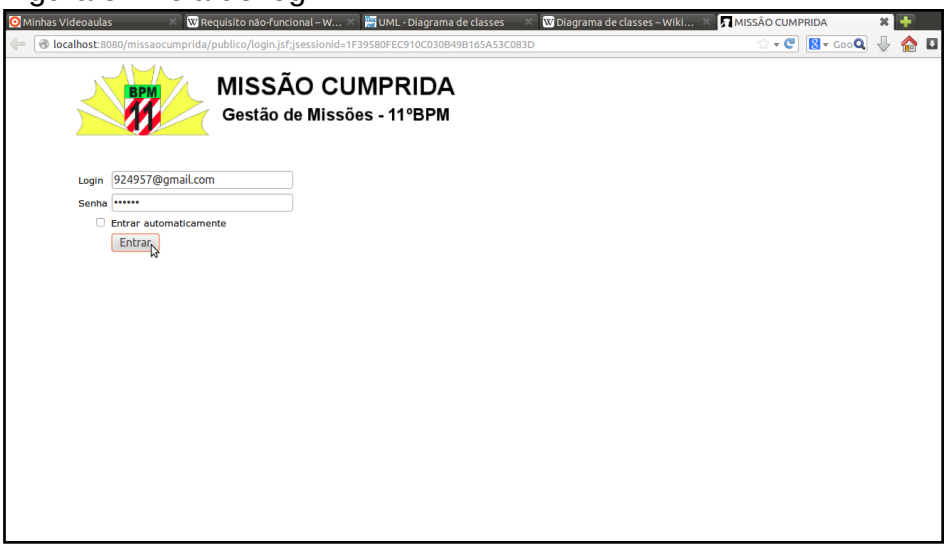
2.5 TELAS

A seguir serão apresentadas algumas telas do sistema MISSÃO CUMPRIDA.

2.5.1 Tela de login

A tela de login é a que fornece validação do usuário para acesso ao sistema e é representado pela figura 5.

Figura 5 - Tela de login



Fonte: o autor

2.5.2 Tela principal

A tela principal é o local onde o usuário será direcionado após o login e também quando não estiver realizando algum cadastro ou edição de registros. É representada na figura 6 e pode ser acessível clicando no botão que representa um boneco segurando um cartaz com o termo “HOME”.

Figura 6 – Tela principal

Fonte: o autor

2.5.3 Telas de cadastros, listagem e edição

Na figura 7 é possível a visualização da tela de cadastro de

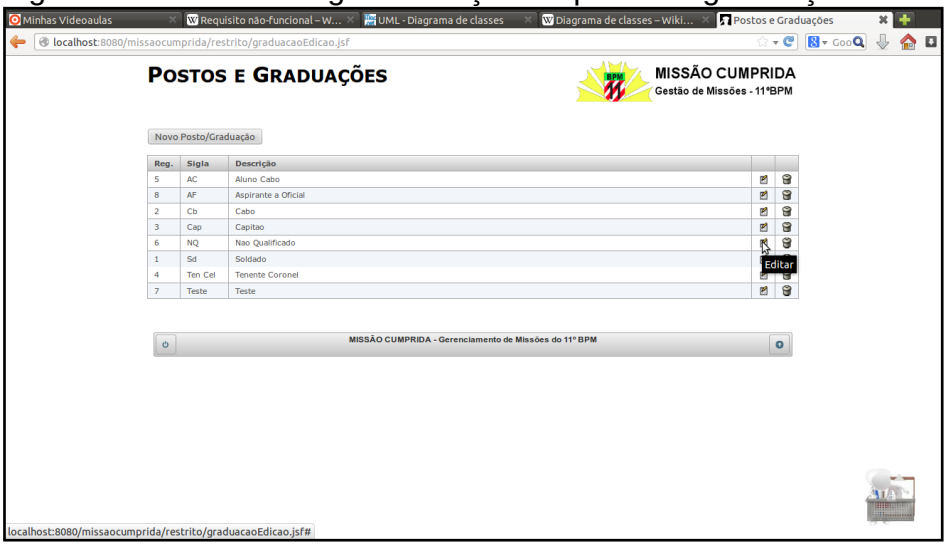
usuários, onde após a inserção das informações é possível a persistência destas informações na base de dados.

Figura 7 – Tela de Cadastro de usuários

Fonte: o autor

Já a figura 8 representa a tela de edição de postos e graduações, que representam o grau hierárquico dos usuários na instituição. É composta basicamente por uma tabela que lista todos os registros existentes de postos e graduações e permite sua edição ou exclusão.

Figura 8 – Tela de listagem e edição de postos e graduações



Fonte: o autor.

2.6 CÓDIGO FONTE

A figura 9 mostra o código fonte da classe GraduaçãoDaoImp, que realiza as atividades entre a aplicação e o banco de dados, possibilitando através de seus métodos persistência, atualização, exclusão, carregamento e listagem de registros.

É importante resaltar que foi optado pela exclusão lógica em todos os componentes da aplicação, ou seja, existe um campo booleano em cada classe representante de uma tabela do banco de dados o qual é marcado como *false*, linha 29 da figura 9, quando se pretende excluir um registro, porém não ocorre a exclusão real dele, pois não será mais listado como um registro ativo, conforme o código de listagem na linha 53 da figura 9. Essa forma de desenvolvimento é importante quando se pretende desativar registro que não são mais necessários no momento atual, porém são fundamentais para elaboração de relatórios anteriores e para que seja mantida a integridade do banco de dados.

Figura 9 – Arquivo GraduacaoDaoImp.java

Arquivo: /home/marcos/workspace_jsf/mi...graduacao/GraduacaoDaoImp.java Página 1 de 1

```
1   package missaocumprida.graduacao;
2
3   import java.util.List;
4
5   import missaocumprida.CadastroDao;
6   import missaocumprida.CadastroDaoImp;
7
8   import org.hibernate.Criteria;
9   import org.hibernate.Query;
10  import org.hibernate.criterion.Order;
11  import org.hibernate.criterion.Restrictions;
12
13  public class GraduacaoDaoImp extends CadastroDaoImp implements CadastroDao {
14
15          @Override
16          public void salvar(Object graduacao) {
17                  super.getSession().save(graduacao);
18          }
19
20          @Override
21          public void atualizar(Object graduacao) {
22                  super.getSession().update(graduacao);
23          }
24
25          // Exclusão Lógica
26          @Override
27          public void excluir(Object graduacao) {
28                  //this.session.delete(graduacao);
29                  ((Graduacao) graduacao).setAtivo(false);
30                  super.getSession().update(graduacao);
31          }
32
33          @Override
34          public Object carregar(Long id) {
35                  String hql = "select u from Graduacao u where u.id = :id";
36                  Query consulta = super.getSession().createQuery(hql);
37                  consulta.setLong("id", id);
38                  return (Graduacao) consulta.uniqueResult();
39          }
40
41          @Override
42          public Object buscarPorParametro(String descricao) {
43                  String hql = "select u from Graduacao u where u.descricao = :descricao";
44                  Query consulta = super.getSession().createQuery(hql);
45                  consulta.setString("descricao", descricao);
46                  return (Graduacao) consulta.uniqueResult();
47          }
48
49          @SuppressWarnings("unchecked")
50          @Override
51          public List<Object> listar() {
52                  Criteria filtro = super.getSession().createCriteria(Graduacao.class);
53                  filtro.add(Restrictions.eq("ativo", Boolean.TRUE) );
54                  filtro.addOrder(Order.asc("descricao"));
55                  return filtro.list();
56          }
57
58  }
```

Fonte: o autor

A figura 10 exibe o código fonte da classe GraduaçãoBean, que realiza as ações executadas pelo usuário, exibindo os resultados e redirecionando sua navegação para as páginas corretas.

Figura 10 – Arquivo GraduacaoBean.java

Arquivo: /home/marcos/workspace_jsf/mi...umprida/web/GraduacaoBean.java Página 1 de 1

```
1   package missaocumprida.web;
2
3   import java.io.Serializable;
4   import java.util.List;
5   import javax.faces.bean.ManagedBean;
6   import javax.faces.bean.RequestScoped;
7   import missaocumprida.graduacao.Graduacao;
8   import missaocumprida.graduacao.GraduacaoRN;
9
10  @ManagedBean(name="graduacaoBean")
11  @RequestScoped
12  public class GraduacaoBean implements Serializable {
13
14          private static final long serialVersionUID = 8009404742312327 26L;
15          private Graduacao graduacao;
16          private List<Object> lista;
17          private String destinoSalvar;
18
19          public GraduacaoBean() {
20                  graduacao = new Graduacao();
21          }
22
23          public String novo() {
24                  destinoSalvar = "graduacaoEdicao";
25                  graduacao = new Graduacao();
26                  return "graduacao";
27          }
28
29          public String editar(){
30                  return "/restrito/graduacao";
31          }
32
33          public String salvar() {
34                  GraduacaoRN graduacaoRN = new GraduacaoRN();
35                  this.graduacao.setAtivo(true);
36                  graduacaoRN.salvar(this.graduacao);
37                  return this.destinoSalvar;
38          }
39
40          public String excluir(){
41                  GraduacaoRN graduacaoRN = new GraduacaoRN();
42                  graduacaoRN.excluir(this.graduacao);
43                  this.lista = null;
44                  return null;
45          }
46
47          public Graduacao getGraduacao() { return graduacao; }
48          public void setGraduacao(Graduacao graduacao) { this.graduacao = graduacao; }
49
50          public List<Object> getLista() {
51                  if (this.lista == null) {
52                          GraduacaoRN graduacaoRN = new GraduacaoRN();
53                          this.lista = graduacaoRN.listar();
54                  }
55
56                  return this.lista;
57          }
58
59          public void setLista(List<Object> lista) {
60                  this.lista = lista;
61          }
62
63          public String getDestinoSalvar() { return destinoSalvar; }
64          public void setDestinoSalvar(String destinoSalvar) { this.destinoSalvar = destinoSalvar; }
65  }
```

Fonte: o autor

A figura 11 mostra o código fonte do arquivo graduacao.xhtml que é a página visível ao usuário para o cadastro de postos e graduações, muito semelhante ao cadastro de usuários visto na figura 7.

Apesar do uso de um arquico XHTML, as tags utilizadas são do JSF, o que fará que sejam exibidos componentes do JSF e não os componentes tradicionais do HTML

Figura 11 – Arquivo graduacao.xhtml

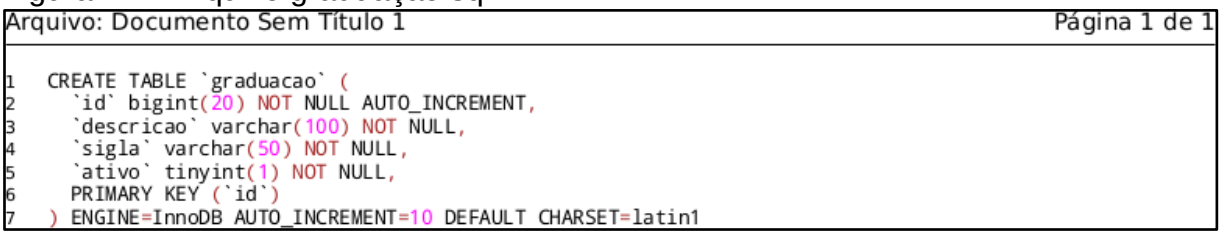
Arquivo: /home/marcos/workspace_jsf/mi...ntent/restrito/graduacao.xhtml Página 1 de 1

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
   xhtml1-transitional.dtd">
3  <html xmlns="http://www.w3.org/1999/xhtml"
4          xmlns:ui="http://java.sun.com/jsf/facelets"
5          xmlns:h="http://java.sun.com/jsf/html"
6          xmlns:f="http://java.sun.com/jsf/core">
7
8  <ui:composition template="/templates/interna.xhtml">
9
10         <ui:define name="titulo">
11         Cadastro de Postos e Graduações
12         </ui:define>
13
14         <ui:define name="corpo">
15
16                 <h:form id="cadastro">
17                         <h:messages />
18                     <h:inputHidden value="#{graduacaoBean.graduacao.id}" />
19                         <h:inputHidden value="#{graduacaoBean.destinoSalvar}" />
20
21
22                         <h:panelGrid columns="2">
23
24                         <h:outputLabel value="Descricao" for="descricao"/
   >
25                                 <h:inputText id="descricao" label="Descrição da graduacao"
26                                         value="#{graduacaoBean.graduacao.descricao}"
   required="true" size="50"
27                                         maxlength="100" requiredMessage="A descrição é
   obrigatória!">
28                                         <f:validateLength minimum="2" maximum="100" />
29                                 </h:inputText>
30
31                                 <h:outputLabel value="Sigla" for="sigla"/
   >
32                                 <h:inputText id="sigla" label="Sigla da graduacao"
33                                         value="#{graduacaoBean.graduacao.sigla}" required="true"
   size="25"
34                                         maxlength="100" requiredMessage="A sigla é obrigatória!">
35                                         <f:validateLength minimum="2" maximum="50" />
36                                 </h:inputText>
37
38                                 <!-- Usado apenas para ocupar a primeira celula da ultima linha -->
39                                 <h:outputText />
40                                 <h:commandButton action="#{graduacaoBean.salvar}" value="Salvar" />
41
42                         </h:panelGrid>
43                 </h:form>
44         </ui:define>
45 </ui:composition>
46 </html>
```

Fonte: o autor

Nas figuras 12 e 13 é possível a visualização de scripts SQL, representando a criação da tabela “graduacao” e “missao”, respectivamente, no banco de dados.

Figura 12 – Arquivo graduaçao.sql

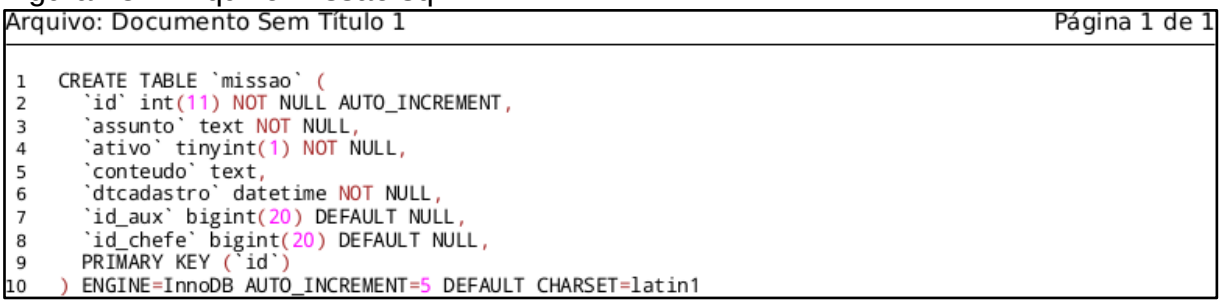


Arquivo: Documento Sem Título 1 Página 1 de 1

```
1   CREATE TABLE `graduacao` (
2     `id` bigint(20) NOT NULL AUTO_INCREMENT,
3     `descricao` varchar(100) NOT NULL,
4     `sigla` varchar(50) NOT NULL,
5     `ativo` tinyint(1) NOT NULL,
6     PRIMARY KEY (`id`)
7   ) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=latin1
```

Fonte: o autor

Figura 13 – Arquivo missao.sql



Arquivo: Documento Sem Título 1 Página 1 de 1

```
1   CREATE TABLE `missao` (
2     `id` int(11) NOT NULL AUTO_INCREMENT,
3     `assunto` text NOT NULL,
4     `ativo` tinyint(1) NOT NULL,
5     `conteudo` text,
6     `dtcadastro` datetime NOT NULL,
7     `id_aux` bigint(20) DEFAULT NULL,
8     `id_chefe` bigint(20) DEFAULT NULL,
9     PRIMARY KEY (`id`)
10  ) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=latin1
```

Fonte: o autor

3 CONCLUSÃO

Com a finalização deste trabalho algumas observações merecem registro, como o fato do desenvolvimento de um bom software demandar tempo e tendo como maior parcela seu planejamento, o levantamento de requisitos, estudos e pesquisas. Outra observação importante é o fato do desenvolvedor, além da habilidade no campo em que trabalha, deve também procurar entender o ramo do negócio que o projeto pretende contemplar, pesquisar fontes e realmente compreender os requisitos do projeto a ser desenvolvido.

De modo prático, entende-se que uma aplicação deve ser desenvolvida almejando a excelência, que atenda as necessidades reais do usuário, que seja bem projetado e que sobreponha à concorrência, em um setor que muda seus conceitos com muita rapidez.

Outra importante conclusão é que devido às muitas tarefas envolvidas no desenvolvido, o mesmo deve ser realizado em equipe, pois o desenvolvimento solitário acaba focando o desenvolvedor em uma visão geral do processo, deixando passar despercebidos detalhes que não ocorreria frente à multiplicidade de visões de uma equipe, pois ao dividir as tarefas também é reduzida a carga individual de trabalho e ampliado o tempo de observação dos detalhes de desenvolvimento naquela parte, o que evitaria possíveis retrabalhos.

Em resumo, um bom software é gerado da soma de vários fatores como o conhecimento específico na área de desenvolvimento, o estudo contínuo, o planejamento e organização do tempo, de contatos e parcerias com outros profissionais do ramo.

REFERÊNCIAS

BOOCH G. **Object Oriented Analysis and Design with Applications**. Addison-Wesley, 1994.

GRECO, Ingrid Chaves Carneiro. **Componentes visuais de especificação em JSF**. 2010. Artigo. Disponível em: <http://www.webartigos.com/artigos/componentes-visuais-de-especificacao-em-jsf/53689/>. Acesso em 06/09/2013.

LUCKOW, Décio H.; MELO, Alexandre A. **Programação JAVA Para Web.** São Paulo: Novatec, 2010.

PRESSMAN, Roger S. **Engenharia de Software**. Sexta Edição. Porto Alegre: Editora MCGrawHill, 2009.

SANTOS, Marco Aurélio Freitas. **Análise de Sistemas II.** Dourados: Unigran, 2009/2.

SOMMERVILLE, Ian. **Engenharia de Software 8ª ed**. São Paulo: Person, 2007.