UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS PATO BRANCO
CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ADRIANE DE COL
FABIO ADRIANO NESELLO

APLICATIVO WEB COM JSF 2.0 E PRIMEFACES PARA GERENCIAMENTO DE REQUISITOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2013

ADRIANE DE COL
FABIO ADRIANO NESELLO

APLICATIVO WEB COM JSF 2.0 E PRIMEFACES PARA GERENCIAMENTO DE REQUISITOS DE SOFTWARE

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: profa. Beatriz T. Borsoi

PATO BRANCO
2013

ATA Nº: **212**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DOS ALUNOS **FáBIO ADRIANO NESELLO e ADRIANE DE COL.**

Às 17:10 hrs do dia 19 de abril de 2013, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Robison Cris Brito (Convidado) e Maico Fernando Wilges Carneiro (Convidado), para avaliar o Trabalho de Diplomação do aluno Fábio Adriano Nesello, matrícula 1030728 e da aluna Adriane De Col, matrícula 610321, sob o título **Aplicativo Web com JSF 2.0 e Primefaces para gerenciamento de requisitos de software**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação os candidatos foram entrevistados pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 18:00 hrs foi encerrada a sessão.

Beatriz Borsoi
Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Prof. Robison Cris Brito, M.Sc.
Convidado

Prof. Maico Fernando Wilges Carneiro, Esp.
Convidado

Prof. Omero Francisco Bertol, M.Sc.
Coordenador do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

DE COL, Adriane; NESELLO, Fabio A. Aplicativo web JSF 2.0 e PrimeFaces para gerenciamento de requisitos de software. 2013. 44 f. Trabalho de conclusão de curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

O desenvolvimento de páginas *web* (os sites *web*) teve início com o uso de uma linguagem de marcação de hipertexto, a *HyperText Markup Language* (*HTML*). Essa linguagem estabeleceu Uma forma de apresentar conteúdos que poderiam ser vinculados permitindo a ligação entre partes distintas de um texto ou de textos em arquivos distintos. Além das ligações, por meio de HTML, é possível definir formatação por meio de especificações (*tags*) predefinidas. Com o uso da Internet, especificamente da *web*, a apresentação das aplicações com formulários simples e baseados em ligações precisou evoluir. Os usuários de aplicativos como os de suporte para negócios estavam acostumados a recursos não fornecidos pelo HTML, como botões e formulários diferenciados, por exemplo. Surgem, assim, recursos que permitem ao usuário uma interação mais próxima às aplicações *desktop*. É a denominada *Rich Internet Application* (aplicações ricas para Internet). Atualmente existem, também, *frameworks*, a exemplo do JavaServer Faces, juntamente com bibliotecas como a PrimeFaces, que facilitam o desenvolvimento de aplicativos com esse tipo de interface. Neste trabalho é apresentado o uso desse *framework* por meio do desenvolvimento de um aplicativo para *web* que tem o objetivo de auxiliar no gerenciamento de requisitos de software.

Palavras-chave: PrimeFaces. JSF 2. Rich Internet Application. Web frameworks.

ABSTRACT

DE COL, Adriane; NESELLO, Fabio A. Web application wih JSF 2.0 and PrimeFaces to manager software requirements. 2013. 44 f. Trabalho de conclusão de curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2013.

The development of web sites began with the use of the HyperText Markup Language (HTML). This language has established a way to introduce content that could be linked allowing the connection between different parts of a text or texts in separated files. In addition to the links through HTML, it is possible to set formatting using specifications (tags) preset. With the use of the Internet, specifically the web presentation of applications based on simple forms and links needed to evolve. Users of applications such as support for businesses were accustomed to features not provided by HTML like buttons and forms differentiated, for example. Thus appear to resources that allow the user closer interaction with desktop applications. It is called Rich Internet Application (Rich Internet Applications). Currently there are also frameworks, such as JavaServer Faces, along with libraries like PrimeFaces, which facilitates the development of applications with this type of interface. This work presents the use of this framework by developing a web application that aims to assist in managing software requirements.

Palavras-chave: PrimeFaces. JSF 2. Rich Internet Application. Web frameworks.

LISTAGENS DE CÓDIGOS

LISTA DE FIGURAS

LISTA DE SIGLAS

B/S	*Browser/Servidor*
CGI	*Common Gateway Interface*
DAO	*Data Access Object*
DHTML	*Dynamic HTML*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
HQL	Hibernate *Query Language*
IDE	*Integrated Development Environment*
Java EE	Java *Enterprise Edition*
JDBC	*Java Database Connectivity*
JDK	*Java Development Kit*
JMS	*Java Message Service*
JSF	*JavaServer Face*
LGPL	*Lesser General Public License*
MVC	*Model-View-Controller*
RIA	*Rich Internet Application*
RMI	*Remote Method Invocation*
SOA	*Service Oriented Architecture*
UML	*Unified Modeling Language*
XHTML	*Extensible HyperText Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais do trabalho, seus objetivos (geral e específicos) e sua justificativa.

1.1 CONSIDERAÇÕES INICIAIS

A abrangência da Internet e a relativamente pouca exigência de recursos de hardware e software para acesso à *web* contribuíram para o seu uso para realizar negócios. Esses negócios se referem às operações internas aos diversos segmentos como empresas de comércio, prestação de serviços, indústrias, órgãos governamentais e entidades do terceiro setor. E, ainda, as operações com o consumidor final que ocorrem por meio das compras e transações comerciais.

Com o uso da Internet muitas modalidades de negócio se redimensionaram e outras cresceram em dimensões exponenciais. O comércio eletrônico, por exemplo, cresceu. No Brasil foram R$ 10,2 bilhões no primeiro semestre de 2012, um aumento de 21% em comparação ao mesmo período em 2011 (CARNETI, 2012). E de certa forma, redimensionou operações como, por exemplo, os sites que permitem leilões, as vendas de consumidor para consumidor e as compras coletivas.

A expansão do uso dos recursos da Internet para fins comerciais, dentre outros fatores, trouxe a necessidade de que os aplicativos *web* apresentassem recursos existentes nas aplicações *desktop*. Os usuários de aplicativos do ambiente *desktop* estavam acostumados a recursos muito mais ricos (em termos de funcionalidades, praticidade e facilidade de realização) que os proporcionados pelas páginas desenvolvidas com *HyperText Markup Language* (HTML) e seus formulários bastante simples baseados em *Common Gateway Interface* (CGI).

Essa necessidade contribuiu para que houvesse o desenvolvimento de tecnologias e recursos para o desenvolvimento de aplicativos *web*. Dentre esses recursos estão os *frameworks* voltados para desenvolvimento *web*.

Assim, como forma de estudar o *framework* JavaServer Faces (JSF) com a biblioteca PrimeFaces foi desenvolvido um sistema para gerenciamento de requisitos de software

utilizando essas tecnologias.

1.2 OBJETIVOS

A seguir serão apresentados os objetivos gerais e específicos do trabalho.

1.2.1 Objetivo Geral

Desenvolver um aplicativo *web* utilizando JSF e PrimeFaces para o gerenciamento de requisitos de software.

1.2.2 Objetivos Específicos

Os objetivos específicos do trabalho os seguintes:

- Desenvolver um aplicativo computacional para *web* que facilite o gerenciamento de requisitos de sistemas;
- Exemplificar a forma de uso JSF e PrimeFaces no desenvolvimento de um aplicativo *web*.

1.3 JUSTIFICATIVA

A expansão de uso da Internet, especificamente dos seus recursos como a *web*, na realização de operações de negócio (de todos os setores da economia, governamentais e outros), traz a necessidade do desenvolvimento de aplicativos que atendam aos novos interesses e anseios dos usuários. Um desses interesses ou mesmo necessidade é pelo desenvolvimento de aplicativos *web* com recursos de interação semelhantes aos aplicativos *desktop*. Essas aplicações são denominadas de interface rica, as *Rich Internet Applications* (RIA).

Como forma de mostrar o uso de tecnologias que permitam o desenvolvimento de aplicativos caracterizados como RIA, será implementado um aplicativo para gestão de requisitos com JSF e PrimeFaces.

A gestão de requisitos é um aspecto de suma importância no desenvolvimento de sistemas porque auxilia a verificar se os interesses do usuário estão sendo atendidos e se o que foi modelado para o sistema está sendo efetivamente implementado. A rastreabilidade dos requisitos durante todo o ciclo de vida do processo de software é dependente de gerenciamento dos requisitos.

1.4 ESTRUTURA DO TRABALHO

Este relatório está organizado em capítulos, dos quais este é o primeiro e apresenta as considerações iniciais, o objetivo e a justificativa do trabalho.

O Capítulo 2 apresenta o referencial teórico que é baseado em conceitos de aplicações para *web* e *frameworks*, especificamente JavaServer Faces e a biblioteca PrimeFaces e o padrão de projetos MVC (*Mode-View-Controller*). Esse padrão de projetos é utilizado no desenvolvimento do aplicativo obtido como resultado deste trabalho.

O Capitulo 3 apresenta os materiais utilizados no desenvolvimento do trabalho e o método utilizado.

No Capítulo 4 está o resultado obtido da realização deste trabalho que é o desenvolvimento de um aplicativo simples visando apresentar o uso das tecnologias envolvidas no desenvolvimento do trabalho.

Por fim, está o Capítulo 5 com a conclusão, seguido das referências bibliográficas.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta a fundamentação teórica de desenvolvimento de interfaces em JSF para sistemas *web*, conceituados como de interface rica.

2.1 APLICATIVOS WEB

O crescimento dinâmico da Internet nos últimos anos tem contribuído para a construção de um novo tipo de aplicações, as aplicações baseadas em páginas *web* (JUSZKIEWICZ et al., 2011). Em aplicações desse tipo todos os dados e as operações são realizados em um mesmo local (servidor) o que permite reduzir significativamente os custos de atualização e modernização (FARRELL, NEZLEK, 2007).

Contudo, essa solução mostrou-se não ser muito ideal em termos de interação com a aplicação. A razão para isso são as limitações de interface com o usuário, decorrentes da HTML. Mesmo com o surgimento de elementos DHTML (*Dynamic HTML*), essa solução ainda apresenta problemas pela incompatibilidade desse tipo de aplicação em diferentes *browsers*, forçando os desenvolvedores a criar múltiplas versões de uma aplicação para distintos navegadores *web* que executam nos diferentes sistemas operacionais (JUSZKIEWICZ et al., 2011).

A solução para o problema de interface de interação com o usuário de aplicações de negócio está nas RIAs (DĘBIŃSKI, SAKOWICZ, KAMIŃSKI, 2010). Um dos principais objetivos das RIAs é propor uma solução para além das aplicações *web* baseadas em páginas HTML, reduzir a quantidade de dados necessários de serem transferidos, prover uma interface com os recursos conhecidos das aplicações *desktop* (JUSZKIEWICZ et al., 2011).

A complexidade das arquiteturas e tecnologias Internet tem crescido amplamente nos últimos anos com a chegada de novos tipos de aplicação *web*, as denominadas *Rich Internet Application*, ou aplicações Internet com interface rica (MELIÁ, PARDILLO, CACHERO, 2010).

O conceito de RIA foi inicialmente proposto pela Macromedia (atualmente Adobe). RIA integra as vantagens das funcionalidades com o usuário das aplicações *desktop*, a adoção geral e os baixos custos de distribuição das aplicações *web* e a interatividade da comunicação multimídia (TAN, WANG, 2010).

Tan e Wang (2010) se referem às aplicações *web* como *Browser*/Servidor (B/S).

Contudo, essas aplicações quando desenvolvidas com HTML apresentam os dados sem efeitos e funcionalidades que podem ser interessantes para o usuário. As aplicações B/S têm que lidar com muitas mudanças nos aspectos de processamento. Aplicações complexas podem requerer diversas páginas para completar o processamento de um evento e vários passos podem ser necessários para atualizar os dados de uma página. As aplicações RIA vem suprir essas demandas (TAN, WANG, 2010).

As RIAs possuem uma arquitetura distribuída em n-camadas cujo projeto, a forma da sua organização, influencia amplamente na qualidade final da aplicação (MELIÁ, PARDILLO, CACHERO, 2010). Dado ao fato que baixos níveis de qualidade estão diretamente relacionados à baixa satisfação do usuário (CHUNG, LEITE, 2009), as RIAs devem ser projetadas de forma que contribuam para otimizar a qualidade dos requisitos relacionados à interação do usuário com o sistema computacional (MELIÁ, PARDILLO, CACHERO, 2010).

Para Meliá, Pardillo e Cachero (2010) para prover qualidade de interação é necessário que a aplicação seja projetada para essa qualidade e não somente para atender aos requisitos do sistema. Para melhorar a portabilidade e a facilidade de mudança é importante usar na implementação das aplicações um estilo arquitetural em camadas (FOLMER, BOSCH, 2004).

O modelo das aplicações RIA combina as vantagens da arquitetura Cliente/Servidor e da arquitetura Browser/Servidor, assim elas não apresentam custos de distribuição ao mesmo tempo em que melhoram a experiência de uso do cliente. As principais vantagens das RIAs residem no fato que o cliente não possui somente o papel de mostrar páginas, mas também implementa uma maneira assíncrona de realizar cálculos, entregar e solicitar dados (TAN, WANG, 2010).

As RIAs possuem uma plataforma tecnológica que possibilita recuperar a capacidades dos computadores *desktop* ou ser similares as capacidades dos sistemas Cliente/Servidor tradicional (TAN, WANG, 2010).

Como forma de resolver muitos problemas surgidos no desenvolvimento de aplicativos para Internet, várias tecnologias têm sido criadas. Uma delas é SOA (*Servide Oriented Architecture*) que prove métodos para o desenvolvimento e integração de sistemas nos quais as funcionalidades são empacotadas como serviços interoperáveis. A outra é RIA. As RIAs combinam a capacidade de resposta e riqueza de interface das aplicações *desktop* com o amplo alcance das aplicações *web* para permitir uma experiência mais efetiva com o usuário (WANG, 2009). As RIAs incorporam em aplicações *web* o comportamento e as características das aplicações *desktop*, como animações, conteúdo multimídia, computação no

lado cliente. Além disso, as RIAs minimizam as desvantagens presentes nas aplicações *web* tradicionais como a continua atualização das páginas *web* e a sobrecarga de processamento no servidor (MARTÍNEZ-NIEVES, HERNÁNDEZ-CARRILLO, ALOR-HERNÁNDEZ, 2010).

Atualmente há várias tecnologias para o desenvolvimento de RIAs, tais como Flex, Silverlight, Google Web Toolkit (MARTÍNEZ-NIEVES, HERNÁNDEZ-CARRILLO, ALOR-HERNÁNDEZ, 2010), Flash/Flex, Ajax e JavaFX (PANG, WEN, PAN, 2010).

RIAs são aplicações *web* que transferem parte da carga de processamento da interface com o usuário para o cliente *web*, enquanto a parte predominante dos dados (controle e manutenção dos dados de negócio) permanecem na aplicação servidor (MARTINEZ-RUIZ, et al., 2006). Uma arquitetura padrão para as RIAs é apresentada na Figura 1.

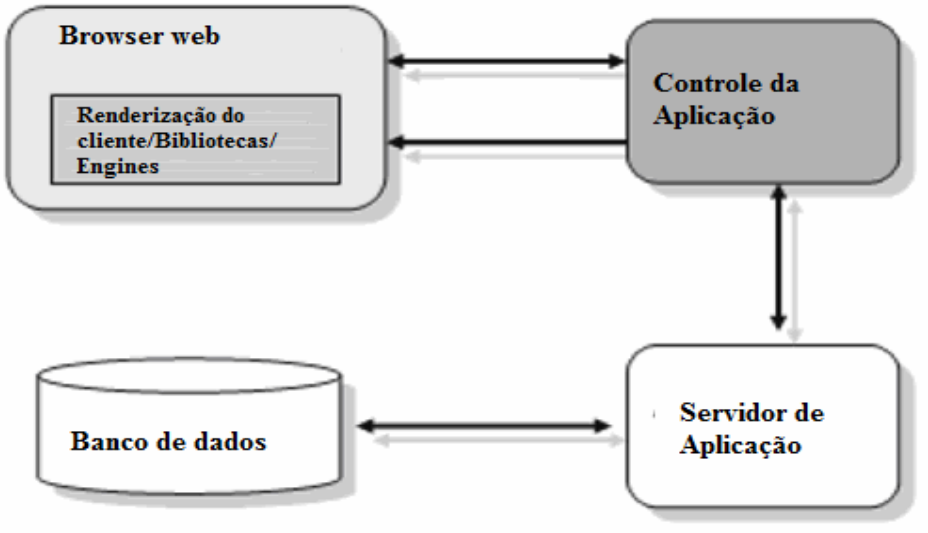

Figura 1 – Arquitetura típica de uma aplicação web

Fonte: traduzido de Martínez-Ruiz (2010, p. 345).

De acordo com a representação da Figura 1, o acesso à aplicação é realizado por um navegador (*browser*) *web*. O navegador é responsável por apresentar a interface da aplicação. A composição (renderização) da página pode ser realizada por meio do uso de bibliotecas (como PrimeFaces, RichFaces e IceFaces, para citar algumas) e *engines*, como a Flash Player, por exemplo. O navegador interage com uma camada que faz o controle da aplicação e a intermediação com o servidor da aplicação. E esse se faz a interação com o banco de dados que pode estar no mesmo servidor ou em servidor distinto.

2.2 Model-View-Controller

No desenvolvimento de software modelos e notações padronizadas possibilitam expressar ideias complexas e também representar o problema e a solução de formas distintas visando facilitar a comunicação entre os envolvidos no projeto e atender aos interesses das distintos desses envolvidos. Os diagramas da UML (*Unified Modeling Language*) exemplos desses modelos e a própria notação dessa linguagem expressa essa padronização.

Modelos auxiliam a entender e representar o problema que está sendo resolvido, bem como a solução desse problema (SELFA, CARRILO, BOONE, 2006). Esses autores ressaltam que existem abordagens diferentes para modelar um problema e expressar os requisitos e as restrições do sistema e que podem ser aplicadas soluções já utilizadas em outros. Padrões provem um meio de obter o conhecimento de soluções bem-sucedidas no desenvolvimento de software (SELFA, CARRILO, BOONE, 2006). Gamma et al. (1997) definiram vinte e três padrões de projeto de software.

Um desses padrões é o MVC, também denominado como arquitetura. Nesse padrão um sistema é segmentado em três tipos de módulos ou componentes (SELFA, CARRILO, BOONE, 2006, MCHEICK, QI, 2011):

a) Modelo (*Model*) que expressa o domínio de conhecimento. O modelo lida com a lógica e os dados da aplicação. E é responsável pela atualização das informações das visões e receber os comandos do controle.

b) Visão (*View*) que apresenta a interface do sistema e é responsável pela apresentação. É importante que a visão esteja separada da estrutura de dados. As visões podem ser distintas para um mesmo conjunto de dados.

c) Controle (*Controll*) que gerencia as atualizações nas visões. O controle pode ser implementado separadamente ou como combinado com as visões. O controle é responsável pelas entradas fornecidas ao sistema incluindo eventos de mouse e teclado. O controle notifica o modelo dessas entradas por meio de eventos.

Uma arquitetura clássica desse padrão é apresentada na Figura 2.

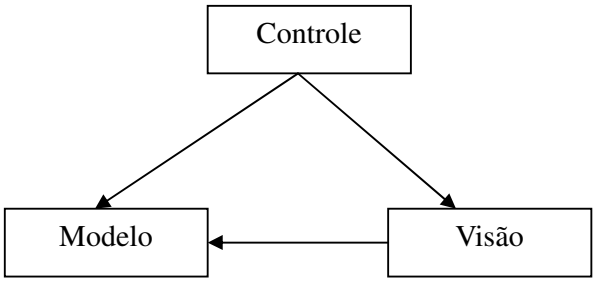

Figura 2 – Arquitetura típica do padrão MVC

Fonte: traduzido e adaptado de Microsfot (2013).

O padrão de projetos MVC tem apresentado benefícios para aplicações interativas possibilitando que permitem múltiplas representações para a mesma informação, promovendo a reutilização de código e auxiliando os desenvolvedores a concentrar-se nos aspectos essenciais e fundamentais da aplicação (MOOCK, 2004).

3 MATERIAIS E MÉTODO

A seguir são descritos os materiais e o método utilizados no desenvolvimento deste trabalho.

3.1 MATERIAIS

Os materiais se referem às tecnologias, ferramentas e recursos utilizados no desenvolvimento do aplicativo:

a) NetBeans - IDE (*Integrated Development Environment*) para desenvolvimento da aplicação. A versão utilizada é a 7.2.1.

b) Glass Fish Server – como servidor de aplicação. Utilizada a versão 3.1.2.

c) PrimeFaces – biblioteca de componentes para JSF. A versão utilizada foi a 3.5.

d) Hibernate – para o mapeamento entre os objetos Java e as tabelas do banco de dados que é relacional. Foi utilizada a versão 3.2.5 do Hibernate.

e) MySQL – para o banco de dados e gerenciador do banco de dados.

f) JasperReports – para o desenvolvimento de relato´rios. A versão utilizada foi 5.0.1.

3.1.1 IDE NetBeans

O NetBeans é uma IDE *open source* completa para o desenvolvimento de projetos *web*, *desktop* e para dispositivos móveis em linguagem Java, PHP e C/C++(NETBEANS, 2013).. O projeto NetBeans foi fundado pela Sun Microsystem em 2000. É livre e de código fonte aberto, além de possuir uma ampla comunidade de usuários e desenvolvedores

Várias informações sobre a IDE e seu *download* podem ser obtidas na página *web* http://netbeans.org.

3.1.2 Glass Fish Server

Servidor de aplicação para a plataforma Java EE (*Enterprise Edition*) com suporte à JDBC (*Java Database Connectivity*), RMI (*Remote Method Invocation*), JMS (*Java Message Service*), entre outros. Permite o desenvolvimento de sistemas corporativos portáteis, escaláveis e facilmente integrados com códigos legados (GLASSFISH, 2013).

3.1.3 PrimeFaces

O PrimeFaces é um *framework* para JSF 2.0 de código aberto, gratuito para utilização e distribuído sob a licença Apache v2. Além dos componentes *web*, PrimeFaces possui uma biblioteca exclusiva para sistemas para dispositivos móveis. Sua utilização e possível em projetos de código aberto ou comerciais, desde que respeitados os termos da licença (PRIMEFACES, 2013).

O PrimeFaces é uma biblioteca de apenas um arquivo, não exige configurações nem possui dependências. Para utilizá-lo basta adicionar o arquivo às bibliotecas do projeto e importar o *namespace*[1] para o local que será feita a utilização dos componentes.

Conforme Junior (2011), as análises elaboradas pelo Google Trends[2] mostram que a popularidade do PrimeFaces teve crescimento exponencial nos últimos anos. As pesquisas envolvendo o *framework* nos últimos anos podem ser resumidas no gráfico apresentado na Figura 3.

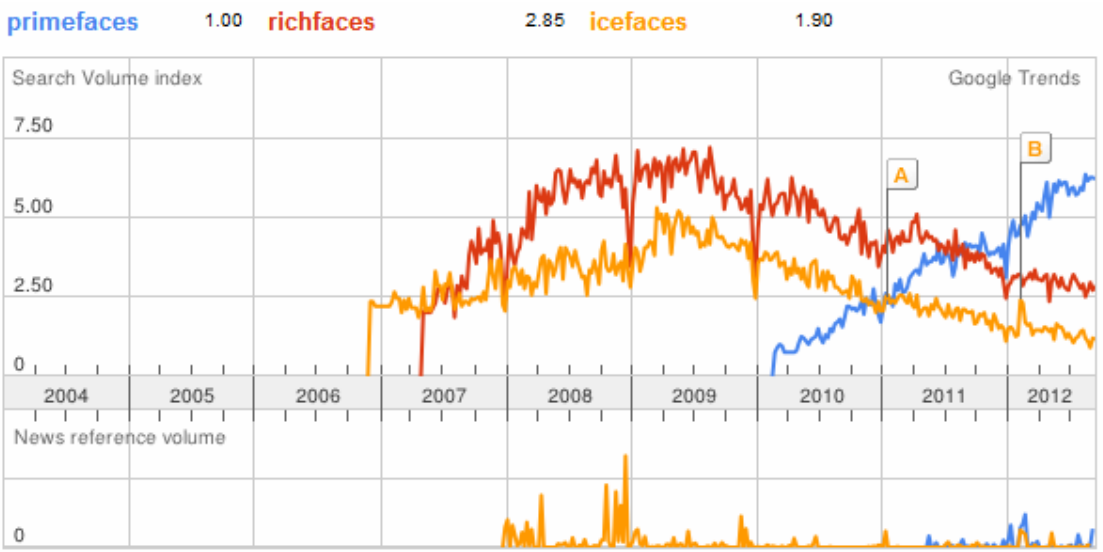

Figura 3 – Gráfico da popularidade dos frameworks

Fonte:http://www.patternizando.com.br/2011/05/primefaces-supera-richfaces-e-icefaces-segundo-google-trends/

O gráfico da Figura 3 apresenta a comparação de uso (popularidade) de três bibliotecas de componentes para JSF que são PrimeFaces, RichFaces e IceFaces. Pela representação do gráfico é possível verificar que IceFaces surge no final de 2006, RichFaces no primeiro semestre de 2007 e PrimeFaces somente no início de 2010. E apresenta uma

[1] Declaração utilizada para designar os componentes que serão utilizadas no desenvolvimento da página JSF.
[2] Ferramenta da Google responsável por analisar os termos mais pesquisados nos últimos tempos.

curva de crescimento bastante acentuada. Ressalta-se que essa é a tendência do momento, mesmo porque novas tecnologias podem surgir e apresentar a mesma expressividade de crescimento.

O sítio oficial do PrimeFaces é http://primefaces.org/. Nessa página é possível fazer o *download* do *framework*, além de consultar sua documentação e sua licença, entre outras informações.

3.1.4 Hibernate

Hibernate é um *framework* de mapeamento objeto-relacional em linguagem Java que facilita o mapeamento de entidades de base de dados e o modelo objeto de uma aplicação. É um software livre distribuído sobre a licença LGPL (*Lesser General Public License*) (HIBERNATE, 2013).

O Hibernate tem como principal característica a transformação das classes Java em tabelas de banco de dados, gerando automaticamente as instruções SQL. Dessa forma é possível manter a aplicação portátil para qualquer base de dados SQL. Contudo, pode haver perda de desempenho na execução da aplicação.

O dialeto utilizado nas consultas do hibernate é o HQL (Hibernate *Query Language*), que é uma linguagem de consulta semelhante ao SQL, porém é orientada a objetos.

3.1.5 MySQL

O MySQL é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL (*Structured Query Language*) como interface. É um banco de dados *open source* mais utilizado atualmente para aplicativos *web* em aplicações como *Facebook*, *Twitter*, *Wikipedia*, *YouTube* utilizam o MySQL (MYSQL, 2013).

Atualmente pertencente à *Oracle*, o MySQL se popularizou por aspectos como a facilidade de integração, portabilidade, compatibilidade, pouca exigência de *hardware*, desempenho, estabilidade e facilidade de uso.

3.1.6 JasperReports Library

Engine para relatórios Java mais popular no mundo, possibilita a exportação em vários formatos, como HTML, PDF entre outros (JASPERREPORTS, 2013).

É possível montar relatórios com texos estáticos, imagens, linhas, formas geométricas, além dos campos que serão preenchidos a partir da consulta de dados. O relatório é gravado em um arquivo XML que, após ser compilado passa a ter a extensão ".jasper". Nesse arquivo as expressões Java serão verificadas em tempo de execução. As consultas dos relatórios podem ser montadas em SQL ou por meio de classes Java.

3.2 MÉTODO

Um dos principais objetivos da realização deste trabalho é o aprendizado do desenvolvimento de aplicações para Internet utilizando JSF e PrimeFaces. Um sistema simples para gerenciamento de requisitos foi implementado como resultado deste estudo. Como os autores deste trabalho conheciam o contexto do sistema implementado foi utilizado o modelo sequencial linear (PRESSMAN, 2002). As fases desse de modelo são: análise, projeto, codificação e teste.

Como trabalho de estágio de Fabio Adriano Nesello, um dos autores, deste trabalho de conclusão de curso foi realizado o estudo das tecnologias e parte do referencial teórico foi elaborado. Para que o estudo das tecnologias pudesse ocorrer inicialmente foi definido que seria utilizada a IDE Netbeans com os *framewoks* Hibernate e JSF e a biblioteca PrimeFaces. Também foi definido que o aplicativo atenderia ao conceito do padrão de projetos MVC. As requisições realizadas em tela (*view*) são enviadas para o *controller* e este, por sua vez, repassa as solicitações ao *model* que é responsável por realizar a inclusão do registro na base de dados MySQL.

A seguir uma descrição sucinta das etapas realizadas:

a) Análise

A análise iniciou com a definição dos requisitos do aplicativo que teve como base o conhecimento dos autores deste trabalho pela experiência em empresas de desenvolvimento de software.

O levantamento dos requisitos foi realizado por meio de uma listagem que se referia às

funcionalidades que o sistema deveria oferecer ao usuário e aos aspectos de qualidade (os requisitos não funcionais).

Na fase de análise também foi definido que o sistema deveria fornecer relatórios, preferencialmente sob a forma gráfica, para facilitar o gerenciamento dos requisitos controlados pelo sistema.

b) Projeto

Com base na listagem dos requisitos foram definidos o diagrama de entidades e relacionamentos com as tabelas do banco de dados e o diagrama de classes do sistema.

c) Codificação

Na fase de codificação o banco de dados foi construído e o sistema foi implementado utilizando as tecnologias apresentadas na Seção 3.1 dos materiais.

d) Testes

Os testes foram realizados de maneira informal e pelos autores deste trabalho. Esses testes visaram localizar erros de codificação e se os requisitos estavam sendo atendidos. A usabilidade, no sentido da interação do usuário com o sistema, também foi verificada.

4 RESULTADOS

Neste capítulo serão apresentados os resultados da realização deste trabalho, que é a codificação do sistema para gerenciamento de requisitos de software.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema implementado tem como finalidade realizar o gerenciamento de requisitos. Para que esse gerenciamento possa ser realizado é necessário fazer o cadastro de situações, tipos e dos próprios requisitos, bem como as compilações do sistema.

Cada compilação poderá ter uma ou mais iterações, conforme a necessidade e as regras de negócio da empresa. Os requisitos serão incluídos nas iterações. Uma vez que o requisito faz parte de uma iteração, é possível definir sua situação e o responsável pela execução dos testes. Cada iteração possui uma situação, data de disponibilização, início e finalização dos testes.

4.2 MODELAGEM DO SISTEMA

O sistema de gerenciamento de requisitos terá como base os requisitos funcionais representados no Quadro 1. Os requisitos funcionais representam a visão do usuário sobre o que o sistema precisa atender.

Identificação	Nome	Descrição
RF001	Cadastrar Situação	Cadastrar situação de requisito. Uma situação possui: código e descrição. A geração do código deverá ser feita automaticamente pelo sistema. Permitir exclusão caso não tenha nenhum registro vinculado. Editar a descrição. Consultar lista de situações.
RF002	Cadastrar Tipo	Cadastrar tipo de requisito. Um tipo possui: código e descrição. A geração do código deverá ser feita automaticamente pelo sistema. Permitir exclusão caso não tenha nenhum registro vinculado. Editar a descrição. Consultar lista de tipos.
RF003	Cadastrar Requisito	Cadastrar requisito. Um requisito possui: código, título, descrição, desenvolvedor e tipo. A geração do código deverá ser feita automaticamente pelo sistema. Permitir exclusão caso não tenha nenhum registro vinculado. Editar a descrição, desenvolvedor e tipo.
RF004	Cadastrar Compilação	Cadastrar compilação. Uma compilação possui: código, descrição, situação, data prevista e data de liberação. A geração do código deverá ser feita automaticamente pelo

		sistema. Permitir exclusão caso não tenha nenhum registro vinculado. Editar a descrição, situação e data de liberação. Consultar lista de compilações.
RF005	Cadastrar Iteração	Cadastrar iteração. Uma iteração possui: código, descrição, data de disponibilização, data de início, data de finalização dos testes, situação, compilação e requisitos. A geração do código deverá ser feita automaticamente pelo sistema. Permitir exclusão caso não haja nenhum registro vinculado. Editar a descrição, data de finalização dos testes e situação. Consultar lista de iterações.
RF006	Vincular o requisito à iteração	É necessário realizar o vínculo entre o requisito e iteração, sendo que o requisito pode estar em nenhuma, uma ou várias iterações.

Quadro 1 – Requisitos funcionais

A listagem do Quadro 2 representa os requisitos não funcionais do sistema, que são restrições de acesso, regras de negócio, entre outros.

Identificação	**Nome**	**Descrição**
RNF001	O requisito deve possuir desenvolvedor e tipo vinculado.	Deve haver um tipo de requisito vinculado ao mesmo, bem como um desenvolvedor.
RNF002	Quanto vinculado à iteração, o requisito deve possuir um responsável pelos testes e também uma situação.	Ao vincular o requisito a uma iteração é necessário vincular o mesmo com uma situação e também informar o responsável pela execução dos testes.
RNF003	A iteração deve possuir uma compilação.	Ao realizar o cadastro da iteração é necessário realizar o vinculo da compilação a qual a iteração pertence.
RNF004	Apresentar gráfico de iterações por situação.	O sistema deve possibilitar a visualização em gráfico dos requisitos de uma iteração por situação, de forma que seja possível visualizar, por exemplo, o percentual de reprovação dos requisitos.

Quadro 2 – Requisitos não-funcionais

Com referência aos requisitos funcionais apresentados no Quadro 1, foi desenvolvido o DER (Diagrama Entidade Relacionamento), conforme a Figura 4.

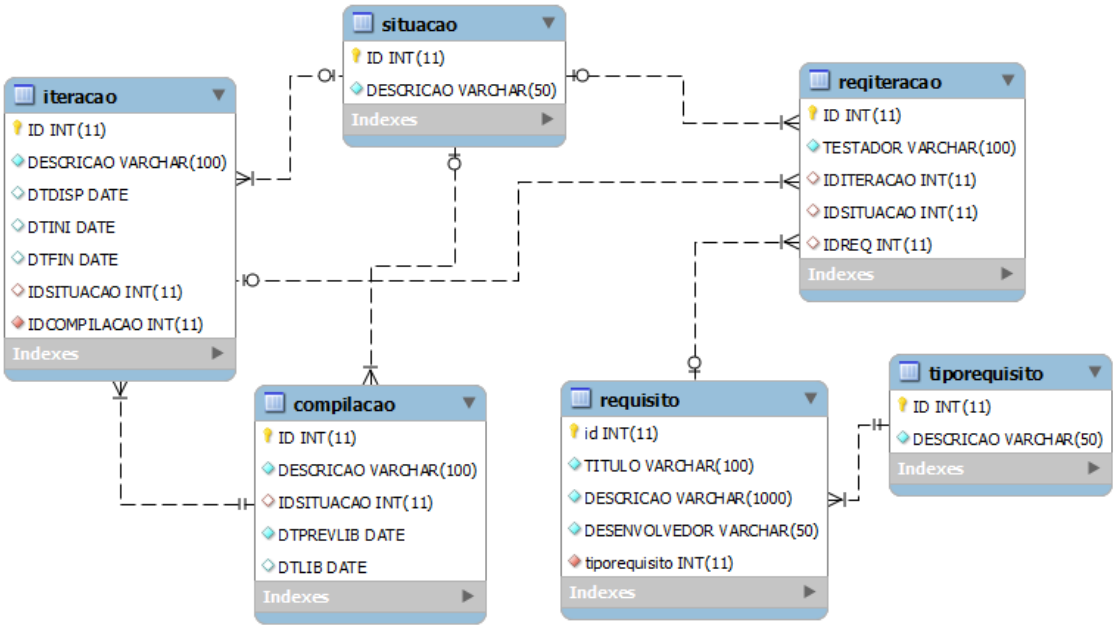

Figura 4 – Diagrama Entidade Relacionamento

Os Quadros 4 a 9 apresentam a descrição das tabelas que compõem o banco de dados, conforme expõe a Figura 4.

No Quadro 4 está a descrição da tabela “requisitos”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não
Titulo	Caractere	Não	Não	Não
Descrição	Caractere	Sim	Não	Não
Desenvolvedor	Caractere	Sim	Não	Não
Tiporequisito	Numérico	Não	Não	sim

Quadro 4 – Tabela requisitos

No Quadro 5 está a descrição da tabela “tiporequisito”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não
Descrição	Caractere	Não	Não	Não

Quadro 5 – Tabela tipos de requisito

O Quadro 6 apresenta a descrição da tabela “compilacao”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não

Descrição	Caractere	Sim	Não	Não
IdSituacao	Numérico	Não	Não	Sim
DtPrevLib	Data	Sim	Não	Não
DtLib	Data	Sim	Não	Não

Quadro 6 – Tabela compilação

O Quadro 7 apresenta a descrição da tabela “situacao”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não
Descrição	Caractere	Sim	Não	Não

Quadro 7 – Tabela situação

O Quadro 8 apresenta a descrição da tabela “iteracao”.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não
Descrição	Caractere	Sim	Não	Não
DtDisp	Data	Sim	Não	Não
DtIni	Data	Sim	Não	Não
DtFin	Data	Sim	Não	Não
IdSituacao	Numérico	Não	Não	Sim
IdCompilacao	Numérico	Não	Não	Sim

Quadro 8 – Tabela iteração

Os campos da tabela “reiteracao” são apresentados no Quadro 9.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
Id	Numérico	Não	Sim	Não
Testador	Caractere	Sim	Não	Não
IdIteracao	Numérico	Não	Não	Sim
DtDisp	Data	Sim	Não	Não
IdSituacao	Numérico	Não	Não	Sim
IdReq	Numérico	Não	Não	Sim

Quadro 9 – Tabela requisição iteração

A Figura 5 apresenta o diagrama de classes do sistema. As classes representam as entidades persistentes do banco de dados.

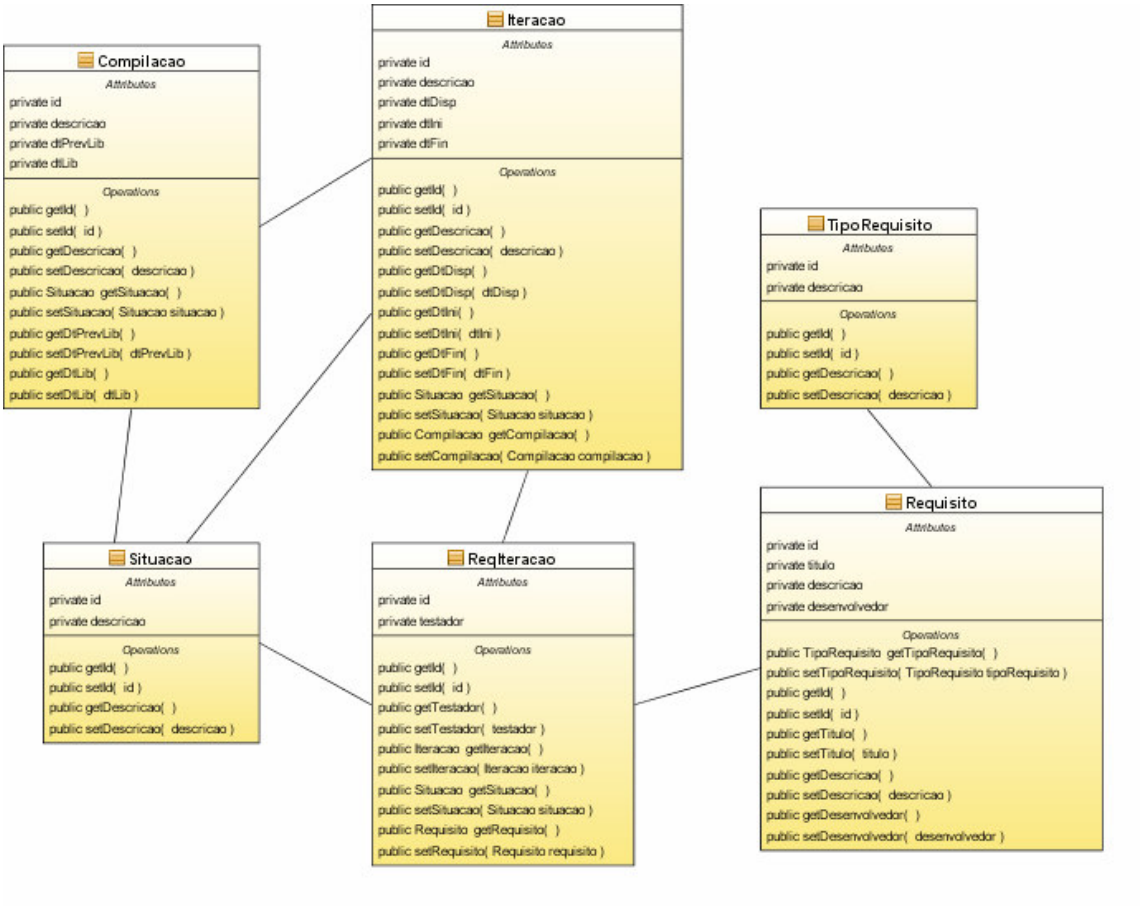

Figura 5 – Diagrama de classes

4.3 DESCRIÇÃO DO SISTEMA

A tela inicial do sistema de gerenciamento de requisitos apresenta o menu na parte superior, o qual ficará disponível em todas as telas e possibilita acesso aos cadastros, como de situação, tipo, requisitos, entre outros, bem como aos relatórios e demais funcionalidades disponibilizadas. A Figura 6 apresenta a interface inicial do sistema.

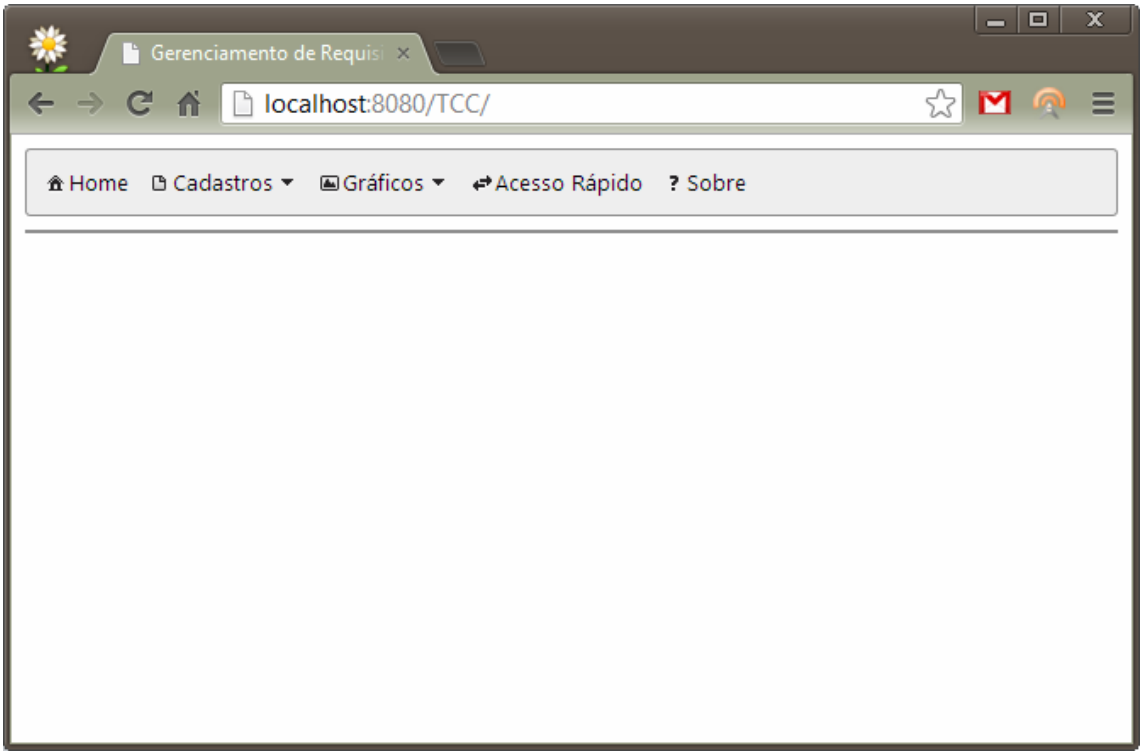

Figura 6 – Tela inicial do sistema de gerenciamento de requisitos

Ao selecionar um item do menu Cadastros será carregada uma tela com um painel superior contendo as ações do cadastro (incluir, alterar, entre outros) e abaixo um *grid* que apresenta os registros com a possibilidade de seleção, paginação e em alguns casos filtros (dispostos na própria coluna). Para filtrar os registros basta digitar parte da informação do campo na coluna respectiva e aguardar que o filtro seja realizado. A remoção do filtro é realizada ao apagar o conteúdo contido nos campos. No cabeçalho do *grid* é apresentado o nome do cadastro e no rodapé o total de registros. Após a realização de alguma ação, como a inclusão, alteração ou exclusão de um registro, será mostrada uma mensagem de *feedback* logo abaixo do menu do sistema. A Figura 7 demonstra a tela de manutenção dos requisitos.

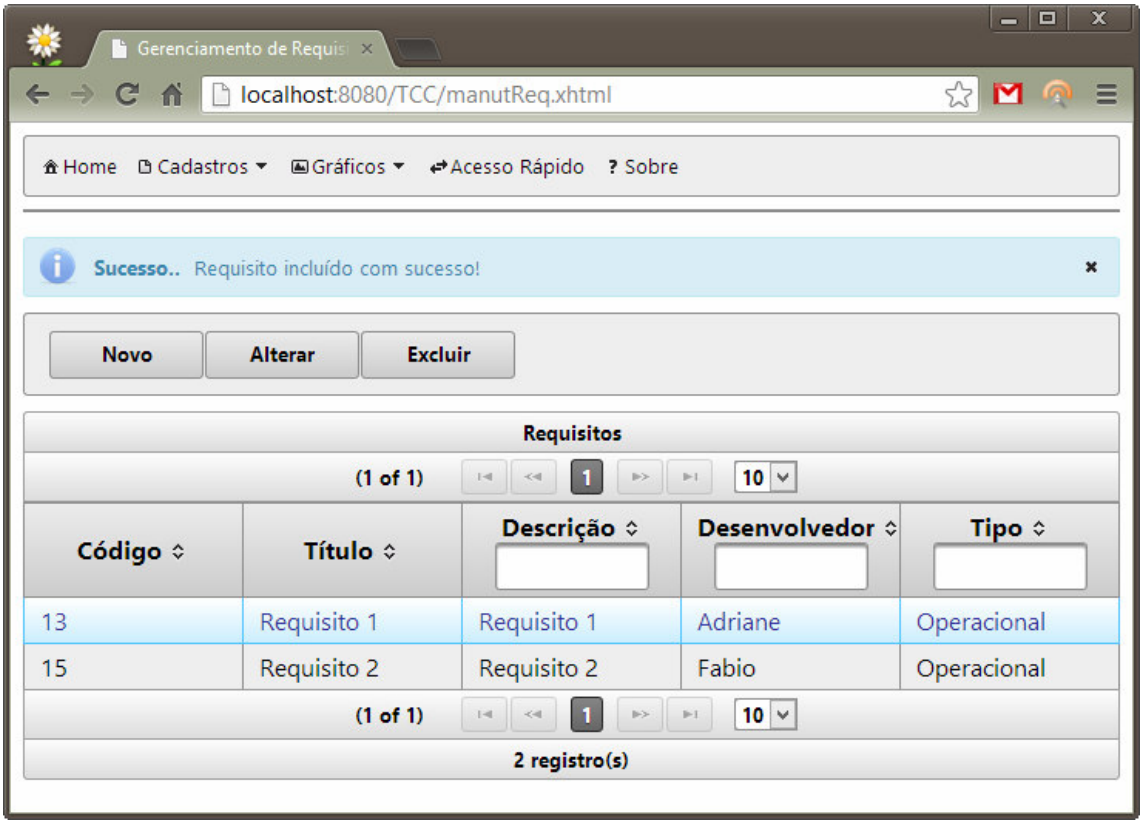

Figura 7 – Tela de cadastro de requisitos

Em todas as telas do sistema de gerenciamento de requisitos que existir componentes de seleção (*combobox*), como a seleção de situação e tipo de requisito, por exemplo, ao digitar parte da descrição o sistema exibirá os registros que iniciarem com a informação digitada. Para visualizar todos os registros basta clicar no botão ao lado do campo. A Figura 8 apresenta a forma de utilização de um campo de seleção na tela de inclusão de requisitos.

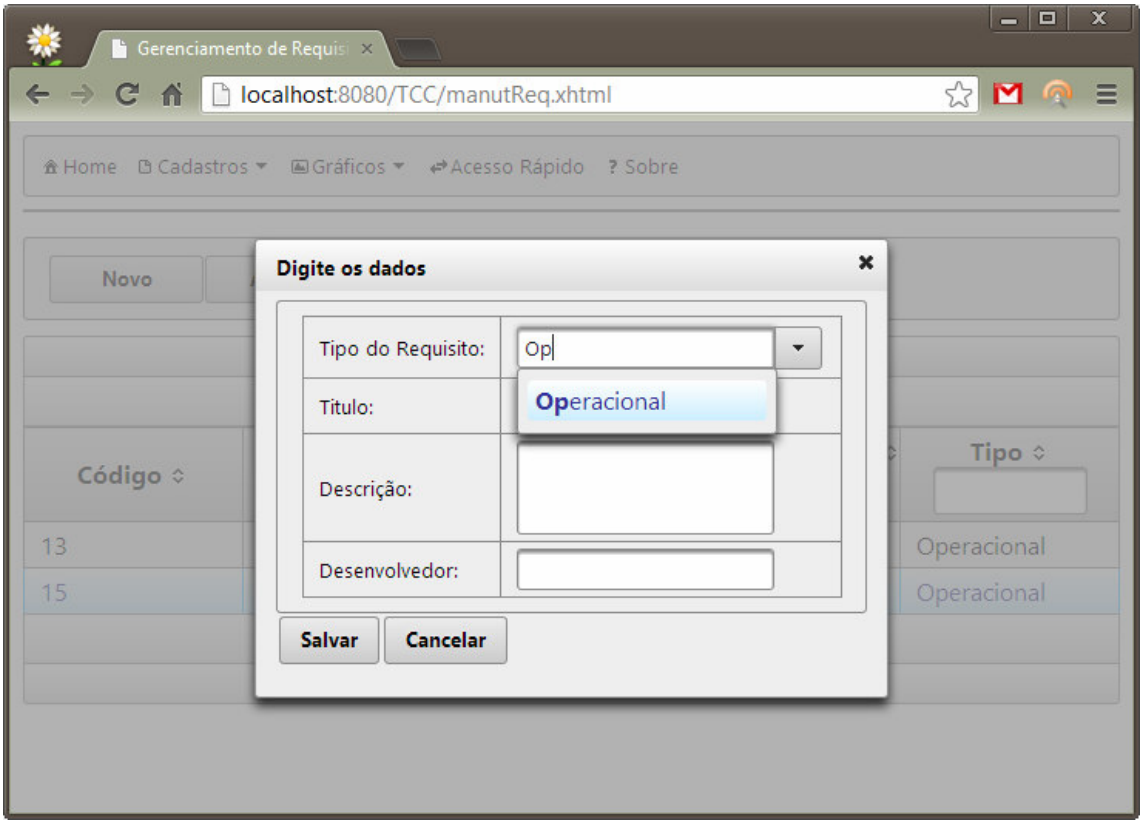

Figura 8 – Componente de seleção da tela de inclusão de requisitos

Na inclusão e alteração de um registro será exibida uma janela flutuante modal contendo os campos de dados. É possível mover a janela para qualquer lugar da tela, bem como salvar ou cancelar o procedimento. A Figura 9 mostra a edição de um requisito.

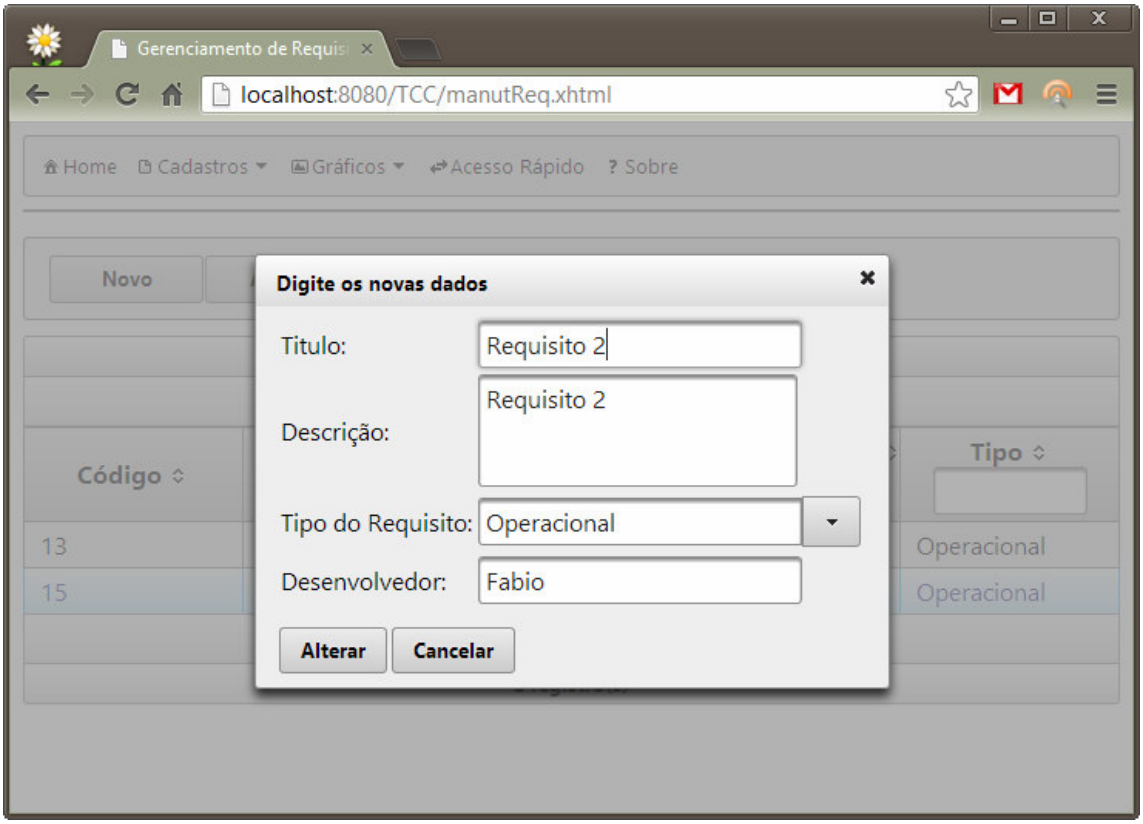

Figura 9 – Janela flutuante para edição de requisito

Em casos de telas que possibilitam a emissão de relatórios, o mesmo será exibido em uma nova aba do navegador, contendo a data e hora de emissão, o número da página e um totalizador de registros. O relatório de tipos de requisitos está apresentado na Figura 10.

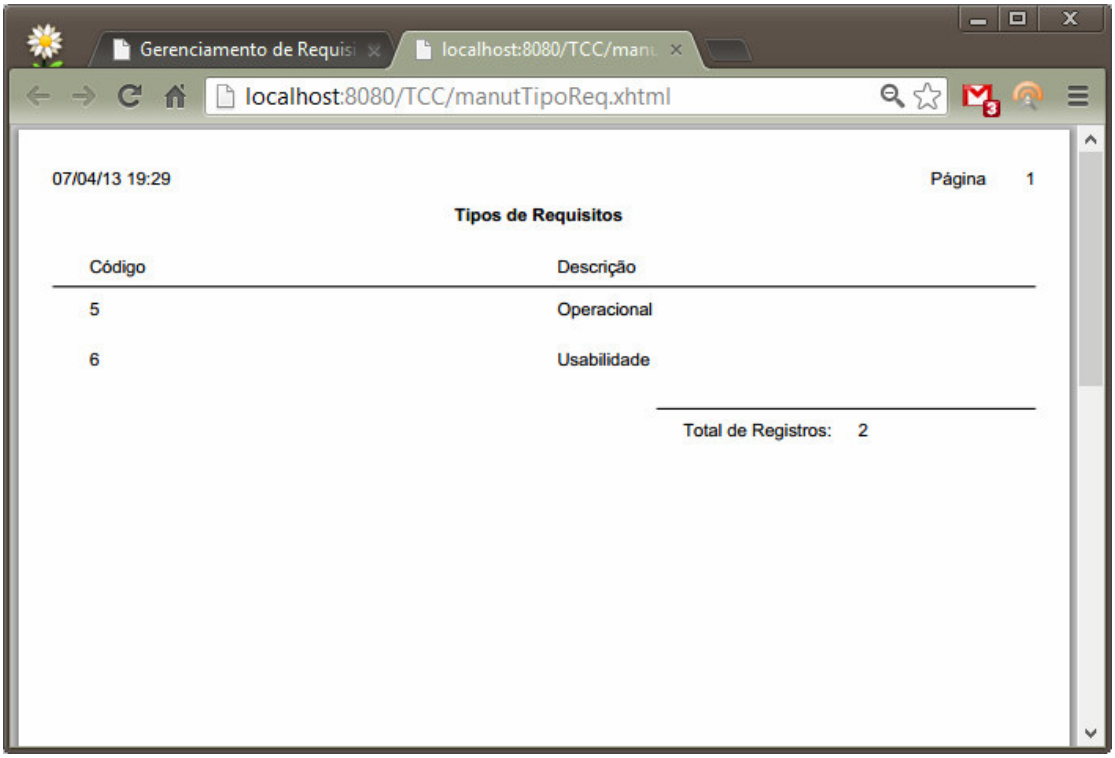

Figura 10 – Emissão do relatório de tipos de requisitos

Todas as telas de cadastros possuem as funcionalidades explicadas anteriormente. Contudo, a tela de iterações possui diversos tratamentos específicos para atender os processos de inclusão, alteração e exclusão de requisitos vinculados a esta. Para cada registro de iteração há botão de expansão, conforme demonstra a Figura 11.

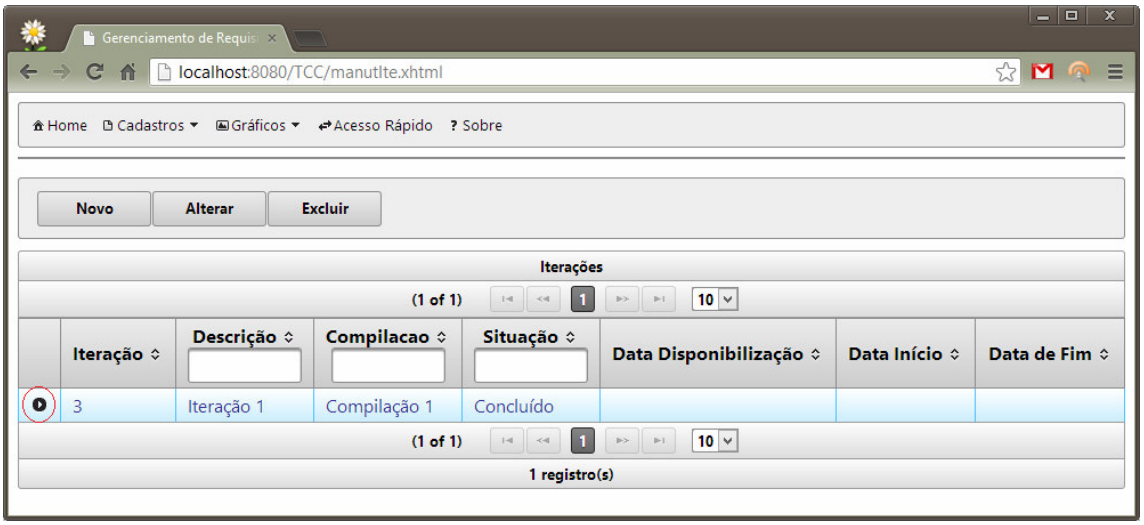

Figura 11 – Botão de expansão da tela de iterações

Ao clicar no botão de expansão são listados abaixo do registro da iteração um painel com campos para a inclusão de um requisito na iteração e um *grid* contendo os registros já inclusos. Para cada requisito da iteração é possível realizar a edição de dados e exclusão clicando no botão da respectiva coluna. Ao solicitar a alteração ou exclusão será mostrada uma tela flutuante modal seguindo o comportamento anteriormente explicado. A expansão de informações da iteração está listada na Figura 12.

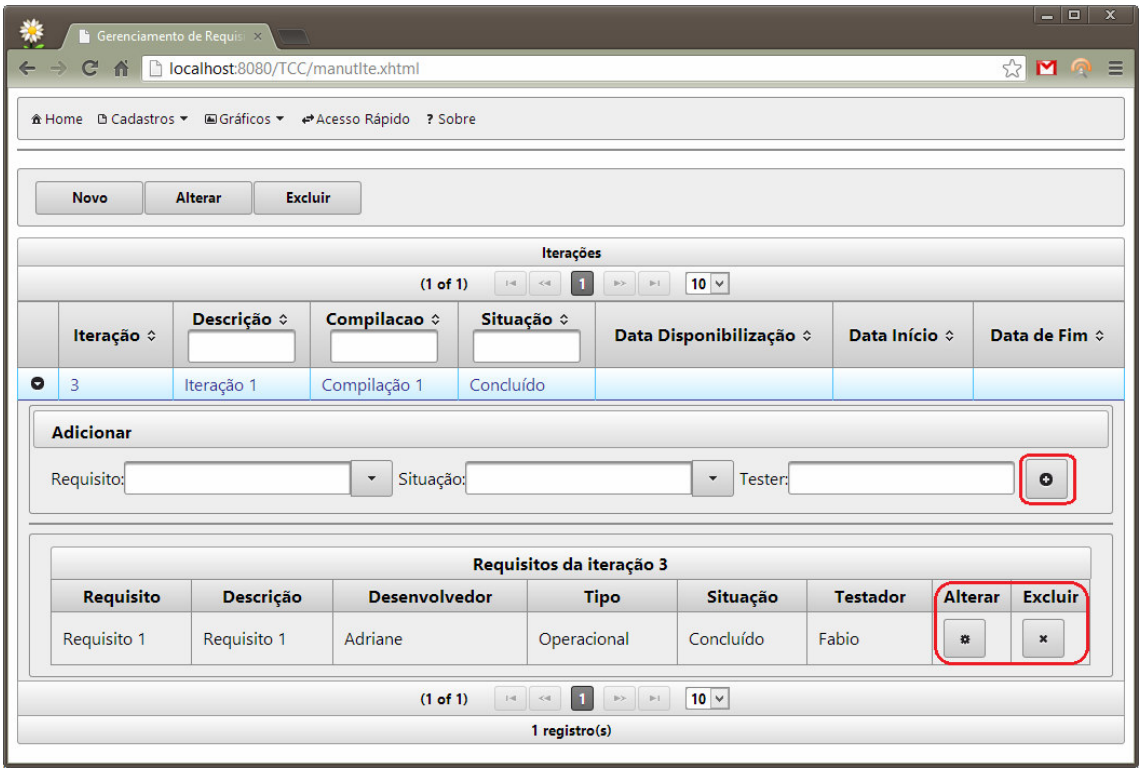

Figura 12 – Expansão de informações da iteração

O sistema de gerenciamento de requisitos possibilita a emissão de gráfico com as informações de Iteração e Situações (Figura 13). A informação do gráfico é mostrada em percentual e apresenta os resultados da iteração selecionada. Para visualizar o gráfico é necessário acessar o menu Gráficos > Iteração por Situações, selecionar a iteração desejada e clicar no botão de visualização.

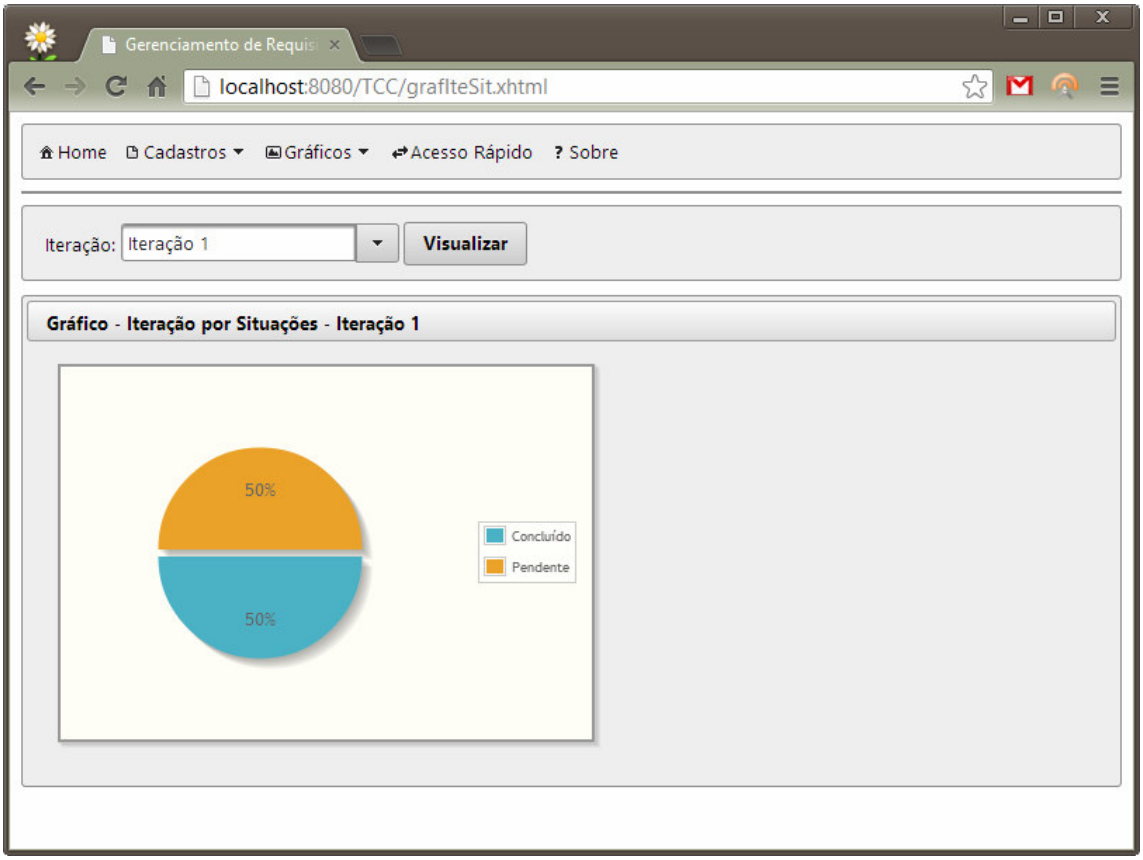

Figura 13 – Visualização do gráfico de iteração por situações

4.4 IMPLEMENTAÇÃO DO SISTEMA

A seguir serão descritos os aspectos mais relevantes da implementação do sistema. Para facilitar a organização do projeto foram criados vários pacotes, conforme apresenta a Figura 14.

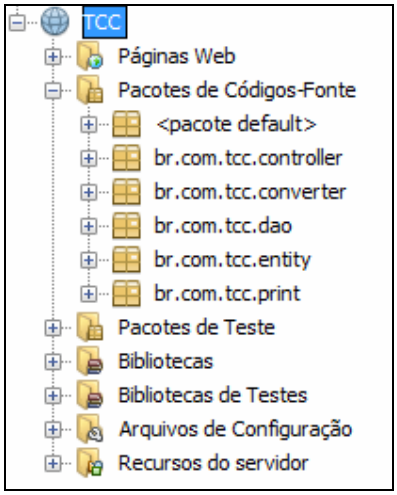

Figura 14 – pacotes do projeto

4.4.1 Mapeando as entidades e configurando o Hibernate

Para configuração do Hibernate são necessários os seguintes passos:

a) Inclusão das *annotations*[3] nas classes de mapeamento das tabelas do banco de dados;

b) Criação e configuração do arquivo de configuração "hibernate.cfg.xml".

Ao fazer a utilização do Hibernate, a manipulação das tabelas do banco de dados é realizada por meio de objetos, os quais são mapeados em classes. Para isso é necessário incluir as *annotations* de forma que o Hibernate identifique que a classe é uma entidade do tipo relacional (tabela), bem como, qual(is) o(s) campo(s) chave(s) da mesma, quais relacionamentos possui, restrições de campos, dentre outros. A Listagem 1 apresenta como exemplo o mapeamento da tabela situação.

```
@Entity
@Table(name="SITUACAO")
public class Situacao implements Serializable {

   @Id
   @Column(name="ID")
   @GeneratedValue(strategy=GenerationType.AUTO)
   private int id;

   @Column(name="DESCRICAO", nullable=false, length=50)
   private String descricao;

   public int getId() {
      return id;
```

[3] Informações que serão interpretadas pelo compilador para realizar uma tarefa pré-definida.

```
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }
}
```

Listagem 1 – Mapeamento de tabela por annotations

As *annotations* comumente utilizadas são as representadas na Listagem 1. Outras informações sobre *annotations* podem ser encontradas em Hibernate (2013).

O arquivo "hibernate.cfg.xml" (Figura 15) possui a configuração de conexão com o banco de dados e também as classes mapeadas e o que pode ser criado no projeto por meio de um assistente gráfico. Para isso basta criar um novo arquivo no projeto na categoria "Hibernate" e tipo de arquivo "Assistente de configuração do Hibernate". Caso o arquivo seja salvo em uma pasta diferente do padrão (preenchida automaticamente no momento da criação do mesmo) será necessário informar o caminho do mesmo ao solicitar uma conexão com o banco de dados.

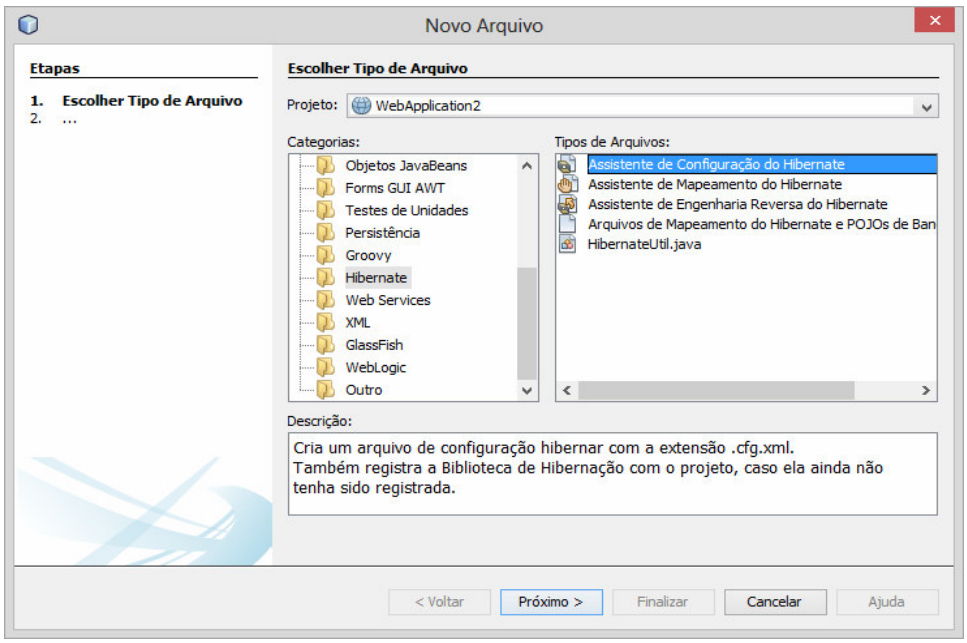

Figura 15 – Criando o arquivo "hibernate.cfg.xml"

Para o correto funcionamento da conexão com a base de dados e utilização dos objetos mapeados do mesmo, é necessário incluir as informações das tabelas que tiveram mapeamento, conforme apresenta a Listagem 2.

```
<mapping class="br.com.tcc.entity.Situacao"/>
<mapping class="br.com.tcc.entity.Requisito"/>
<mapping class="br.com.tcc.entity.TipoRequisito"/>
<mapping class="br.com.tcc.entity.Compilacao"/>
<mapping class="br.com.tcc.entity.Iteracao"/>
<mapping class="br.com.tcc.entity.ReqIteracao"/>
```

Listagem 2 – Inclusão do mapeamento das entidades no arquivo "hibernate.cfg.xml"

A conexão com a base de dados será realizada com base na configuração do arquivo "hibernate.cfg.xml". A classe HibernateUtil é a responsável pela conexão, conforme descrito na Listagem 3.

```
public class HibernateUtil {

   private static final SessionFactory sessionFactory;

   static {
      try {
         // cria a SessionFactory a partir do arquivo hibernate.cfg.xml
         sessionFactory = new AnnotationConfiguration().configure().buildSessionFactory();
      } catch (Throwable ex) {
         System.err.println("Início da SessionFactory falhou." + ex);
         throw new ExceptionInInitializerError(ex);
      }
   }

   public static SessionFactory getSessionFactory() {
      return sessionFactory;
   }
}
```

Listagem 3 – Classe HibernateUtil

4.3.2 Desenvolvendo as classes de DAO

As classes de acesso à base de dados possuem uma interface padrão que é definida na *interface* InterfaceDao (Listagem 4).

```
public interface InterfaceDao<T> {

   boolean incluir(T instance);

   boolean alterar(T instance);
```

```
    boolean excluir(T instance);

}
```

Listagem 4 – Interface padrão do DAO

As classes de DAO (*Data Access Object*) são responsáveis por intermediar a base de dados com a aplicação. A interface definida possui apenas três métodos, são eles: “incluir”, “alterar” e “excluir”, porém como apresentado posteriormente algumas classes necessitarão de outros métodos para rotinas específicas.

Na Listagem 5 é mostrado como exemplo a função “incluir” da classe SituacaoDAO, que além dos métodos herdados da interface possui também métodos específicos.

```
    @Override
    public boolean incluir(Situacao instance) {
        try {
            sf = HibernateUtil.getSessionFactory();
            session = sf.getCurrentSession();
            tx = session.beginTransaction();
            session.save(instance);
            tx.commit();
            System.out.println("Record Inserted");
            return true;
        } catch (Exception e) {
            System.err.println("Erro: " + e.getMessage());
            tx.rollback();
            return false;
        }
    }
```

Listagem 5 – Método “incluir” da classe SituacaoDAO

Como mostrado na Listagem 5, as três primeiras linhas do método são responsáveis por iniciar uma sessão do Hibernate, bem como uma sessão de banco de dados e também uma transação, que possibilita que os comandos sejam persistidos ou retornados. Após iniciar a transação é realizado o procedimento, que no caso da inclusão é salvar um registro na base de dados pelo comando “session.save(instance)”. Pode-se perceber que toda a manipulação de dados via aplicação é realizada por classes que foram previamente mapeadas (conforme explicado na Seção 4.3.1). Nos demais métodos, como alteração e exclusão, a lógica permanece a mesma. É alterado somente o comando utilizado, “session.update(instance)” no caso de alteração e “session.delete(instance)” no caso de exclusão.

Muitas vezes é necessário realizar consultas específicas na base de dados, como, por exemplo, uma pesquisa por id do campo ou ainda por descrição. Nessas ocasiões também

existem métodos de pesquisa que abrangem as situações mais comuns. Na Listagem 6 é apresentado o método “findByName” da classe SituacaoDAO.

```
public Situacao findByName(String descricao) {
      try {
         sf = HibernateUtil.getSessionFactory();
         session = sf.getCurrentSession();
         tx = session.beginTransaction();
         Situacao situacao = new Situacao();
         if (!session.createCriteria(Situacao.class).add(Restrictions.eq("descricao", descri-
cao)).list().isEmpty()) {
            situacao = (Situacao) session.createCriteria(Situacao.class).add(Restrictions.eq("descricao",
descricao)).list().get(0);
         }
         tx.commit();
         return situacao;
      } catch (Exception e) {
         System.err.println("Erro: " + e.getMessage());
         tx.rollback();
         return null;
      }
   }
```

Listagem 6 – Método “findByName” da classe SituacaoDAO

O método “findByName” utiliza a mesma estrutura de todas as outras transações com a base de dados. Porém o seu método de pesquisa é realizado via critérios, tornando possível definir filtros. Neste caso, foi utilizada a pesquisa por descrição, ou seja, é aplicada uma restrição em todas as situações cadastradas conforme determinada descrição solicitada.

4.3.3 Desenvolvendo as classes de controle e as páginas XHTML

As classes de controle possuem todas as informações necessárias para a inclusão/alteração/exclusão de um registro da base de dados. Para seu correto funcionamento é necessário a utilização das anotações de escopo(@SessionScoped, por exemplo) e iteração com as páginas JSF (@ManagedBean), bem como possibilitar a serialização do *controller* implementando a interface java.io.Serializable. Além disso, é necessário realizar a implementação dos eventos das páginas, como, por exemplo, a inclusão. O *controller* da página de CRUD de situações está disposto na Listagem 7.

```
@ViewScoped
@ManagedBean(name = "manutSitReqController")
public class ManutSitReqController implements Serializable {

   private SituacaoDao situacaoDao;
```

```
    private DataModel situacoes;
    private Situacao situacao;
    private String descricao;

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public DataModel getSituacoes() {
        return situacoes;
    }

    public void setSituacoes(DataModel situacoes) {
        this.situacoes = situacoes;
    }

    public Situacao getSituacao() {
        return situacao;
    }

    public void setSituacao(Situacao situacao) {
        this.situacao = situacao;
    }

    @PostConstruct
    public void init() {
        situacaoDao = new SituacaoDao();
        List<Situacao> lista = situacaoDao.findAll();
        this.situacoes = new ListDataModel(lista);
        if (lista.isEmpty()) {
            this.situacao = new Situacao();
        } else {
            this.situacao = lista.get(0);
        }
    }

    public void preparaAlteracao(ActionEvent actionEvent) {
        //recupera o registro que está sendo alterado
        this.situacao = (Situacao) (situacoes.getRowData());
    }

    public void incluir(ActionEvent actionEvent) {
        Situacao sit = new Situacao();
        sit.setDescricao(this.descricao);
        this.situacaoDao = new SituacaoDao();
        if (situacaoDao.incluir(sit)) {
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO, "Sucesso.. ", "Situação incluída com sucesso!"));
        } else {
```

```
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_WARN, "Atenção...", "Cadastro não realizado, tente
novamente!"));
        }
        atualizaSituacoes();
    }

    public void alterar(ActionEvent actionEvent) {
        if (this.situacao == null) {
            FacesContext.getCurrentInstance().addMessage(null, new FacesMes-
sage(FacesMessage.SEVERITY_WARN, "Atenção...", "Nenhum registro selecionado!"));
        } else {
            situacaoDao = new SituacaoDao();
            situacao.setDescricao(this.descricao);
            if (situacaoDao.alterar(this.situacao)) {
                FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO, "Sucesso.. ", "Registro alterado com sucesso!"));
            } else {
                FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_WARN, "Atenção...", "Registro não alterado, tente
novamente!"));
            }
            atualizaSituacoes();
        }

    }

    public void excluir(ActionEvent actionEvent) {
        if (this.situacao == null) {
            FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_WARN, "Atenção...", "Nenhum registro selecionado!"));
        } else {
            this.situacaoDao = new SituacaoDao();
            //this.situacao = (Situacao) (situacoes.getRowData());
            if (situacaoDao.excluir(this.situacao)) {
                FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_INFO, "Sucesso.. ", "Registro excluído com sucesso!"));
            } else {
                FacesContext.getCurrentInstance().addMessage(null, new
FacesMessage(FacesMessage.SEVERITY_WARN, "Atenção...", "Registro não excluído, tente
novamente!"));
            }
            atualizaSituacoes();
        }
    }

    public void atualizaSituacoes() {
        this.situacaoDao = new SituacaoDao();
        List<Situacao> lista = situacaoDao.findAll();
        this.situacoes = new ListDataModel(lista);
        if (lista.isEmpty()) {
            this.situacao = new Situacao();
        } else {
```

```
            this.situacao = lista.get(0);
        }
    }

    public void imprimir(ActionEvent actionEvent) {
        this.situacaoDao = new SituacaoDao();
        List<Situacao> lista = situacaoDao.findAll();
        new ImprimirRelatorio(lista, "RelSituacao");
    }
}
```

Listagem 7 – Controle da inclusão de situações

Nas classes de controle será necessário criar todas as variáveis para executar o processo proposto (e seus *getters* e *setters*), bem como o(s) evento(s) de ação do formulário XHTML. Esse formulário ficará responsável por solicitar ao DAO o processamento das informações e também dar um *feedback* ao usuário.

Para criar uma página XHTML é necessário adicionar o *namespace* do PrimeFaces, conforme demonstrado na Listagem 8.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui" lang="pt">
```

Listagem 8 –*Namespace* das páginas XHTML

Em seguida basta adicionar os componentes necessários e a chamada aos eventos do controle. A Listagem 9 apresenta o código do menu do sistema.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui" lang="pt">
    <h:head>
        <title>Gerenciamento de Requisitos</title>
        <h:form>
            <p:menubar>
                <p:menuitem value="Home" icon="ui-icon-home" url="index.xhtml" />

                <p:submenu label="Cadastros" icon="ui-icon-document">
                    <p:menuitem value="Tipo de Requisito" url="manutTipoReq.xhtml"/>
                    <p:menuitem value="Situação" url="manutSitReq.xhtml"/>
                    <p:menuitem value="Requisito" url="manutReq.xhtml"/>
                    <p:menuitem value="Compilação" url="manutComp.xhtml"/>
                    <p:menuitem value="Iteração" url="manutIte.xhtml"/>
                </p:submenu>
```

```
            <p:submenu label="Gráficos" icon="ui-icon-image">
               <p:menuitem value="Iteração por Situações" url="grafIteSit.xhtml"/>
            </p:submenu>

            <p:menuitem value="Sobre" icon="ui-icon-help" onclick="dlgSobre.show();"/>

         </p:menubar>

         <p:separator/>

         <p:dialog header="Sobre" widgetVar="dlgSobre" resizable="false" modal="true">
            <h:form id="formSobre">
               <p:panel>
                  <p:outputLabel>
                     SGR - Sistema de Gerenciamento de Requisitos.
                  </p:outputLabel>
                  <p:separator/>
                  <p:outputLabel>
                     Desenvolvido por
                     Adriane de Col e Fabio Adriano Nesello.
                  </p:outputLabel>
               </p:panel>

            </h:form>
         </p:dialog>

      </h:form>
   </h:head>
</html>
```

Listagem 9 – Menu do sistema

5 CONCLUSÃO

O desenvolvimento de aplicativos *web* eficientes e com boa usabilidade tornou-se muito mais fácil a partir do momento em que *frameworks* como o Hibernate e JavaServer Faces, com apoio de bibliotecas, como PrimeFaces, foram disponibilizados. Consequentemente, a agilidade e a praticidade proporcionadas por essas tecnologias reduzem o tempo de implementação de sistemas, além de garantir confiabilidade que antes dependiam unicamente do desenvolvedor.

A utilização de *frameworks* no desenvolvimento de sistemas é de grande valia. O Hibernate, por exemplo, trabalha com objetos, o que torna o projeto dinâmico quanto à base de dados escolhida, podendo ser trocada a qualquer momento sem necessidade de refazer todas as consultas, processos de manipulação de dados, entre outros. Já o PrimeFaces possui componentes altamente configuráveis e de rápido processamento e mesmo comportamento em vários navegadores. Contudo, também pode-se citar desvantagens, como lentidão para iniciar a conexão com a base de dados, dificuldades com os eventos e comportamentos dos componentes PrimeFaces, dentre outros.

REFERÊNCIAS

CARNET, Karen. **Crescimento do e-commerce fica abaixo do esperado no primeiro semestre**, disponível em: <http://idgnow.uol.com.br/internet/2012/08/22/crescimento-do-e-commerce-fica-abaixo-do-esperado-no-primeiro-semestre/>. Acesso em: 29 de jan. 2013.

CHUNG, Lawrence; PRADO LEITE, Julio Cesar S. do. **On non-functional requirements in software engineering.** Conceptual Modeling: Foundations and Applications, 2009, p. 363–379.

DĘBIŃSKI Adam; SAKOWICZ Bartosz; KAMIŃSKI M Marek. **Methods of creating graphical interfaces of web applications based on the example of Flex framework**, TCSET'2010 (International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science), 2010, p. 170-173.

FARRELL, Jeason; NEZLEK, George S. **Rich internet applications the next stage of application development**. 29th International Conference on Information Technology Interfaces, 2007, p. 413-418.

FOLMER, Eelke; Bosch, Jan. **Architecting for usability: a survey**. Journal of Systems and Software, v. 70, p. 61–78, March 2004.

GAMMA, Erich; JOHNSON, Ralph; HELM, Richard; VLISSIDES, John. Padrões de projeto: soluções reutilizáveis de software orientado a objetos . Porto Alegre: Bookman, 2004.

GLASSFISH. **Glassfish Server**. Disponível em: <http://glassfish.java.net/>. Acesso em: 05 jan. 2013.

HIBERNATE. **Hibernate community documentation.** Disponível em: http://docs.jboss.org/hibernate/annotations/3.5/reference/en/html/entity.html. Acesso em: 15 jan. 2013.

JASPERREPORTS. **JasperReports Library**. Disponível em: <http://community.jaspersoft.com/project/jasperreports-library>. Acesso em: 13 jan. 2013.

JUNIOR, Sylvio B. **PrimeFaces supera RichFaces e IceFaces segundo Google Trends**. Disponível em: <http://www.patternizando.com.br/2011/05/primefaces-supera-richfaces-e-icefaces-segundo-google-trends/>. Acesso em 17 set. 2012.

JUSZKIEWICZ, Przemyslaw; SAKOWICZ, Bartosz; MAZUR, P., NAPIERALSKI, Andrzej **The use of Adobe Flex in combination with Java EE technology on the example of ticket booking system**. 11th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 2011, p. 317-320.

MARTÍNEZ-NIEVES, Luis Alfredo; HERNÁNDEZ-CARRILLO, Víctor Manuel; ALOR-HERNÁNDEZ, Giner. **An ADV-UWE based phases process for rich internet applications development.** IEEE Electronics, Robotics and Automotive Mechanics Conference, 2010, p. 45-50.

MARTÍNEZ-RUIZ, Francisco Javier. **The triad-based design of rich user interfaces for**

internet applications. ACM SIGCHI Symposium on Engineering Interactive Computing Systems, 2010, p. 345-348.

MCHEICK, Hamid, QI, Yan. **Dependency of components in MVC distributed architecture.** Canadian Conference on Electrical and Computer Engineering (CCECE, 2011), 2011, p. 691-694.

MELIÁ, Santiago; PARDILLO, Jesús; CACHERO, Cristina. **Automatic selection of RIA software architectures using quality models**. Seventh International Conference on the Quality of Information and Communications Technology, 2010, p. 505-510.

MICROSOFT, MSDN, **Model-View-Controller**, version 1.0.1.Disponível em: <http://msdn.microsoft.com/en-us/library/ff649643.aspx>. Acesso em 04 de abr. 2013.

MOOCK, Colin. **Object-Oriented development with ActionScript 2.0,** O,Reilly, 2004.

MOURA, Maria F. **Como desenvolver interfaces para web**. Disponível em: <http://www.cnptia.embrapa.br/node/138.html>. Acesso em 10 jun. 2012.

MYSQL. **MySQL. The world's most popular open source database**. Disponível em: <http://www.mysql.com/>. Acesso em: 13 jan. 2013.

NETBEANS. **NetBeans IDE features**. Disponível em: <http://netbeans.org/index_pt_BR.html>. Acesso em: 28 jan. 2013.

PANG, Zhen; WEN, Fuan; PAN, Xiwei; LU, Cen. **Migration model for rich internet applications based on PureMVC framework**. International Conference On Computer Design And Applications (ICCDA 2010), v. 5, 2010, p. 343.

PRIMEFACES. **PrimeFaces ultimate JSF comonent suite**. Disponível em: <http://www.primefaces.org/>. Acesso em 05 jan. 2013.

SELFA, Diana M., CARRILLO, Maya, BOONE, Ma. del Rocío. **A database and web application based on MVC architecture**. 16th International Conference on Electronics, Communications and Computers (CONIELECOMP '06), p. 48-53, 2006.

TAN, Baohua; WANG, Juntao. **A DeviceNet Fieldbus data acquisition system based on Flex technology and RIA model**. 2010 IEEE International Conference on Progress in Informatics and Computing (PIC), v. 2, 2010, p. 1167-1169.

WANG Fei. **The development of Rich Internet Application based on current technologies**. International Conference on Web Information Systems and Mining. 2009, p. 815-818.