UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ADRIANA ARIATI

SISTEMA WEB PARA ARMAZENAMENTO E RECUPERAÇÃO DE ARTEFATOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2012

ADRIANA ARIATI

SISTEMA WEB PARA ARMAZENAMENTO E RECUPERAÇÃO DE ARTEFATOS 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.

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2012

ATA Nº: **196**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DA ALUNA **ADRIANA ARIATI**.

Às 09:30 hrs do dia 9 de outubro de 2012, Bloco S da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Fábio Favarim (Convidado) e Marco Antonio de Castro Barbosa (Convidado), para avaliar o Trabalho de Diplomação da aluna Adriana Ariati, matrícula 1116592, sob o título **Sistema Web para Armazenamento e Recuperação de Artefatos 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 a candidata foi entrevistada pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 10:25 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Prof. Fábio Favarim, Dr.
Convidado

Prof. Marco Antonio de Castro Barbosa, Dr.
Convidado

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

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

ARIATI, Adriana. Sistema web para armazenamento e recuperação de artefatos de software. 2012. 111 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2012.

Um dos objetivos do reuso de artefatos de software é reduzir o tempo de desenvolvimento de um sistema pelo uso de artefatos (partes do sistema) já prontos. Reuso também está relacionado à qualidade porque os artefatos sendo reusados já foram amplamente testados. Esses aspectos, dentre outros, fazem de reuso um assunto estudado e discutido. Contudo, a realização efetiva de reuso é, ainda, um desafio. Os artefatos devem ser desenvolvidos visando reuso e pode haver necessidade de mudanças na cultura organizacional para que a prática de reuso ocorra. Visando facilitar a adoção da prática de reuso, por meio da realização deste trabalho um repositório de artefatos é desenvolvido com o objetivo de facilitar a recuperação dos artefatos e consequentemente promover reuso. Entende-se como artefato tudo o que é produzido no ciclo de vida de software e que possa ser armazenado, incluindo documentos de análise e projeto, modelos desses documentos, componentes de código, procedimentos e experiências da equipe. A definição conceitual do repositório desenvolvido tem como base técnicas de classificação como vocabulário controlado, com classificação por palavras-chave e descrição textual. O repositório desenvolvido é um aplicativo para a web.

Palavras-chave: Repositório de artefatos. Reuso de artefatos. Artefatos de software.

ABSTRACT

ARIATI, Adriana. Web system to store and retrieve software artifact. 2012. 111 f. Monografia de Trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2012.

The reuse of software artifacts is an issue that has been widely discussed and studied because it aims to reduce the development time of a system by the use of artifacts (parts of software) developed early. Reuse promotes quality through the use of artifacts most widely tested. However, the actual realization of reuse is still a challenge. It is necessary that the artifacts are developed to reuse and, usually, it is also necessary to change the organizational culture to the practice of reuse occurs. As a contribution to the reuse, it is proposed a repository of artifacts in order to facilitate the artifact recovery and thus the reuse. It is understood as an artifact all produced in the software life cycle and that can be stored, including analysis and design documents, models of these documents, code components, procedures and experiences. The conceptual definition of the repository is based on techniques of classification such as controlled vocabulary, classification by keywords and textual description.

Keywords: Software artifact repository. Artifact reuse. Software artifact.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

ACID	Atomicidade, Consistência, Isolamento, Durabilidade
API	*Application Programming Interface*
ASP	*Active Server Pages*
BPMN	*Business Process Modeling Notation*
CSS	*Cascading Style Sheets*
DB	*Data Base*
DOM	*Document Object Model*
FGR	Forma Gráfica de Representação
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
IIS	*Internet Information Services*
JDBC	*Java Database Connectivity*
ODBC	*Open Data Base Connectivity*
OLEDB	*Object Linking and Embedding Data Base*
PSQL	PostgreSQL
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
W3C	*World Wide Web Consortium*
XHTML	*eXtensible* HTML
XML	*Extensible Markup Language*
XSTL	*Extensible Style Sheet Language*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais que inserem a proposta deste trabalho no assunto de pesquisa, que é reuso e repositório de artefatos de software. Em seguida são apresentados os objetivos e a justificativa do trabalho. E, por fim, a organização do texto por meio da apresentação dos seus capítulos é apresentada.

1.1 CONSIDERAÇÕES INICIAIS

De maneira geral, reuso está vinculado a código. Isso porque a orientação a objetos provê reuso de código, inclusive pelo seu mecanismo de herança. E o desenvolvimento baseado em componentes também está centrado no reuso de elementos de código. Porém, há outras possibilidades de realizar reuso, como, por exemplo, documentos de análise e projeto de sistemas, planos de testes, padrões de projeto, experiências e conhecimento adquirido com a realização das atividades relacionadas à modelagem, à implementação e à manutenção de software.

Esse contexto mostra que o conceito de reuso pode ser aplicado a todo o ciclo de vida de software. Embora reuso seja mais evidente na fase desenvolvimento, Justo (1996) argumentou que não há razões teóricas que impossibilitem sua aplicação nas fases iniciais de desenvolvimento de software. O reuso pode ocorrer no início do ciclo de vida seja pelo aproveitamento de documentos de análise e projeto, seja pelo uso de modelos para produzir esses documentos ou pelo registro de experiências (melhores práticas). A especificação dos requisitos é uma fase crucial no ciclo de vida de um sistema e aplicar reuso efetivo nessa fase auxiliaria a desenvolver sistemas que atendessem mais adequadamente os requisitos requeridos pelo usuário (JUSTO, 1996).

Os padrões de projeto, tendo como referência os vinte e três padrões propostos por Gamma et al. (1997), podem ser definidos como componentes conceituais reutilizáveis. Eles se aplicam à produção de código, mas o que é reutilizado são os conceitos envolvidos. Os conceitos se referem à estrutura conceitual de um padrão que resolve um problema bem estabelecido.

Este trabalho se refere a reuso em escopo mais amplo, abrangendo todos os produtos resultantes no ciclo de vida de software. Esses produtos são denominados artefatos, e representam componentes de código (software), documentos (como diagramas de classes,

planos de testes, modelos para elaboração de documentos) e procedimentos (métodos, técnicas, orientações para realizar as atividades) que podem ser resultantes da experiência (prática) da equipe ou de seus membros na realização das atividades.

O desenvolvimento baseado em reuso requer tempo e esforço consideráveis para que bons resultados sejam obtidos, o que tem sido uma barreira para pequenas organizações (SHIVA; SHALA, 2007). O aplicativo desenvolvido por meio deste trabalho é um repositório para o armazenamento de artefatos de software voltado para fábricas de software de pequeno porte. Uma maneira de recuperação dos artefatos, visando localizar artefatos que atendam os requisitos de busca, também é definida.

O repositório foi implementado com o uso de tecnologias para desenvolvimento para ambiente *web*. Facilitando, assim, o compartilhamento de artefatos por fábricas de software ou unidades distintas de uma mesma fábrica. Cada artefato armazenado no repositório terá definidas permissões de acesso.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um sistema computacional que possibilite armazenar artefatos produzidos e utilizados durante o ciclo de vida de software e recuperá-los visando reuso.

1.2.2 Objetivos Específicos

- Definir os metadados para os artefatos a serem armazenados no repositório visando facilitar a busca desses artefatos a partir dos critérios de pesquisa, incluindo palavras-chave e descrição textual do artefato;
- Auxiliar, por meio do sistema desenvolvido, fábricas de software a compartilhar artefatos de software;
- Disponibilizar um repositório de artefatos de software implementado como um aplicativo *web* facilitando o acesso e a manutenção do sistema.

1.3 JUSTIFICATIVA

O reuso de artefatos de software pode auxiliar fábricas de software a desenvolver software em menos tempo, com custo menor e com mais qualidade. A redução do tempo e de custo ocorre pelo uso de partes (código e documentos de modelagem, dentre outros) já prontas por terem sido implementadas em outros projetos. A qualidade é decorrente do uso de artefatos mais amplamente testados porque eles foram usados satisfatoriamente em outros projetos.

Fábrica de software, no contexto deste trabalho, é considerada como o ambiente de desenvolvimento de software. Esse ambiente abrange as atividades relacionadas ao ciclo de vida de software e que se inserem e interagem com as demais atividades de uma organização, quer tenha ela o desenvolvimento de software como atividade meio ou fim.

Os benefícios proporcionados pelo reuso como redução de tempo e de custos e melhoria de qualidade estão relacionados a dois aspectos básicos que são a redução de retrabalho e ao uso de artefatos já testados em outros projetos. A prática de reuso não necessariamente precisa estar relacionada a uma infraestrutura de grande porte ou fábricas de software com muitos funcionários. É, também, uma questão de cultura organizacional.

Desenvolver artefatos visando reuso requer seguir padronização, modelos e procedimentos. E para isso pode ser necessário mudanças nas formas de trabalho que estão incorporadas nas práticas organizacionais. Outro aspecto relevante no reuso é a maneira que os artefatos são armazenados visando facilitar a sua recuperação e, especialmente, que a busca seja otimizada. Otimização se refere a que seja recuperado o artefato que melhor atende aos critérios definidos para a busca.

As mudanças culturais e mesmo de procedimentos que podem ser requeridas na organização para que a prática de reuso se torne efetiva não fazem parte do escopo deste trabalho. Contudo, tem-se o interesse de contribuir para o reuso de artefatos de software pelo desenvolvimento de um repositório que visa facilitar o armazenamento e a recuperação desses artefatos. Esse interesse se justifica pela oportunidade de oferecer uma forma de fomentar reuso em fábricas de software de pequeno porte. Essas fábricas podem ter dificuldade de desenvolver os seus próprios repositórios ou de adquirir software pronto para essa finalidade. Além disso, o repositório desenvolvido poderá ser utilizado por fábricas de software que façam parte de iniciativas de desenvolvimento regional, incluindo incubadoras de projetos de software, por exemplo. Elas possam compartilhar modelos, experiências e mesmo produtos.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos. Este é o primeiro e apresenta a introdução com a visão geral do problema e da solução proposta. Incluindo os objetivos e a justificativa do trabalho.

No Capítulo 2 está o referencial teórico, centrado em reuso e repositório de artefatos e formas de classificação desses artefatos, visando armazená-los de maneira a facilitar a sua recuperação. E, ainda, trabalhos relacionados que são publicações sobre reuso que se assemelham à proposta deste trabalho. Esses trabalhos foram tomados como base, ainda que o repositório proposto não tenha como fundamentação exclusivamente o constante nesses textos.

O Capítulo 3 apresenta os materiais e o método utilizados no desenvolvimento do sistema, desde a definição dos requisitos até a implementação. Os materiais se referem às ferramentas de modelagem tanto dos requisitos como da base de dados, o ambiente de desenvolvimento e para a geração de relatórios. E, ainda, as tecnologias para que o sistema pudesse ser executado, como o banco de dados e o servidor *web*.

No Capítulo 4 é apresentado o resultado obtido com a realização deste trabalho que é o sistema desenvolvido. Os resultados incluem a modelagem do sistema, a apresentação das suas funcionalidades por meio das telas (interface) e exemplos da implementação por meio de listagens de código.

No Capítulo 5 estão as considerações finais e as perspectivas futuras para continuidade deste trabalho.

2 REUSO E REPOSITÓRIO DE ARTEFATOS

O reuso de artefatos está vinculado à forma de produzir, armazenar e recuperar esses artefatos. Portanto, esses são assuntos relacionados quando tratado sobre reuso de software. O conceito de artefato utilizado neste trabalho vai ao encontro com o apresentado por Tavares e Santa'Anna (2006), segundo esses autores artefato se refere ao que é produzido, utilizado e modificado por um processo que pode ser informação, documento, modelo, código fonte, dentre outros.

Neste capítulo é apresentado o referencial teórico do trabalho que inclui trabalhos relacionados à proposta apresentada e implementada, bem como sobre reuso, repositórios e formas de classificação de artefatos.

2.1 REUSO DE ARTEFATOS

Reuso está relacionado ao uso em situações novas de conceitos ou produtos previamente adquiridos ou construídos. Sendo que os mesmos devem ser definidos e/ou produzidos visando o seu reuso, estarem armazenados de forma a facilitar sua recuperação e prover a identificação de similaridades entre situações novas e antigas, permitindo sua adaptação (PRIETO-DIAZ, 1989).

Kotonya, Lock e Mariani (2011) fazem um apanhado histórico sobre reuso, destacando as primeiras associações com o tema, iniciando com o trabalho de McIlroy (1968) que sugeriu que a indústria de software deveria ser baseada em componentes reusáveis. Na sequência, Parnas (1976) propôs a noção de família de programas e Neighbours (1984) definiu os conceitos de domínio e análise de domínio. Essas ideias iniciaram a sistematização de engenharia de linha de produtos de software, que é tratada por Greenfield e Short (2004) como fábrica de software. Mais recentemente destacam-se Frakes e Kang (2005) e Shiva e Shala (2007) que identificaram mudanças na forma de localizar artefatos em repositórios. A popularidade do reuso aumentou recentemente devido à possibilidade de desenvolvimento distribuído de software (SHIVA; SHALA, 2007), ocasionado, principalmente, pela distribuição geográfica de filiais de uma mesma empresa e pelo desenvolvimento compartilhado de produtos por empresas distintas.

Reuso baseado em componentes, que se refere à composição de aplicações a partir de componentes existentes tem como base o conceito de componente de software como um

elemento de software com funcionalidades explícitas e interfaces definidas de forma que permitam identificar seus mecanismos de interação e o seu comportamento.

Shiva e Shala (2007) ressaltam que o maior desafio no reuso de componentes de software está no gerenciamento eficiente desses componentes de forma a prover localização e busca rápidas. A localização e a busca rápidas podem estar associadas aos metadados que caracterizam o artefato. Os metadados definem os atributos que caracterizam o artefato e quanto melhor caracterizam o artefato mais efetiva será a busca do ponto de vista do usuário e mais rápida em termos de processamento computacional. Isso porque, com metadados adequadamente definidos pode não ser necessário buscar no conteúdo do artefato, por exemplo. Para Shiva e Shala (2007), uma melhor forma de estruturar esses repositórios e de implementar interfaces de busca não tem sido proposta. Sendo este um assunto de pesquisa ainda em aberto.

Embora o conceito de reuso seja geralmente aplicado na fase de implementação de software com o objetivo de reutilizar partes de código, de forma geral qualquer item gerado no processo de desenvolvimento de software pode ser reusado. Dentre esses itens estão os documentos de modelagem do sistema e os resultados de experiências e do aprendizado na realização das atividades, incluindo procedimentos e maneiras de resolver problemas.

A aplicação de reuso nas fases iniciais de desenvolvimento de software foram investigadas desde o início da década de 1990, como atestam as publicações de Grosz (1992), Maiden e Sutcliffe (1992) e Sutcliffe e Maiden (1994). Melo (2007) também afirma que a reutilização, ou reuso, não é só aplicável a fragmentos de código fonte, mas a todo o trabalho gerado durante o processo de desenvolvimento de software, como dados, arquitetura e projeto.

A utilização sistemática de reuso no processo de desenvolvimento de software reduz o tempo de desenvolvimento, de testes e a possibilidade de inclusão de novos erros no processo de desenvolvimento (PREE, 1994). O reuso de qualquer item no processo de desenvolvimento de software faz com que a produtividade tenha crescimento na mesma proporção em que se aplica o reuso em projetos (BARNES, 1991).

O desenvolvimento de software voltado para reuso, de acordo com Sommerville (2007) e Melo (2007), provê vantagens competitivas em relação à forma tradicional de desenvolvimento. Dentre as vantagens citadas por esses dois autores estão:

- A confiabilidade nos artefatos reusados é sempre maior do que a confiabilidade em artefatos novos. É o aumento da qualidade.
- Risco reduzido no processo, devido ao menor custo para desenvolvimento, gerando previsibilidade e confiabilidade.

- Profissionais especialistas podem encapsular seu conhecimento em artefatos reusáveis que serão utilizados em outros projetos.
- Utilização de padrões existentes ao longo dos projetos.
- Desenvolvimento acelerado devido à redução no tempo de desenvolvimento e de realização de testes, gerando redução dos custos, no tempo de entrega e aumento da produtividade.

Portanto, o reuso, visto no processo de desenvolvimento de software de forma sistemática, visa trazer resultados relacionados à produtividade e à qualidade. Situações já enfrentadas em algum sistema anterior podem ser utilizadas no processo de desenvolvimento um novo sistema. (SOMMERVILLE, 2007).

2.2 REPOSITÓRIO DE ARTEFATOS

É importante que os artefatos de um repositório estejam devidamente catalogados para que a busca possa ser mais efetiva. Isso no sentido de que a pesquisa possa identificar os artefatos que atendam aos critérios de busca definidos pelo usuário. Esse catálogo se refere à maneira de identificação dos artefatos visando localizá-los de forma facilitada e eficiente. E é importante que todos os envolvidos tenham conhecimento do repositório, devendo ser capazes usá-lo com facilidade. Assim, como função básica os repositórios permitem o armazenamento e a recuperação de artefatos.

Para que artefatos sejam armazenados visando facilitar a sua localização, a documentação dos mesmos deve definir as características de identificação do artefato, tais como nome, descrição textual, data de criação e observações que se fizerem necessárias. Essas informações são denominadas metadados (PRESSMAN, 2002). Os metadados podem ser utilizados por mecanismos de busca. Contudo, o aspecto relevante de um repositório de artefatos está em como organizar esses dados e definir os atributos do artefato de maneira que seja possível localizar o artefato que melhor atende aos interesses do usuário. Esses interesses são representados pelos critérios de busca.

A estrutura do repositório é um fator essencial nos resultados da busca. Mesmo que algoritmos possam prover efetividade na busca com um esforço mínimo de estrutura e indexação, repositórios fracamente estruturados com indexação inadequada não terão um bom desempenho independentemente dos algoritmos de busca utilizados (HENNINGER, 1997).

2.3 INDEXAÇÃO, CLASSIFICAÇÃO E RECUPERAÇÃO DE ARTEFATOS

A recuperação efetiva de artefatos de software em repositórios está vinculada à combinação dos métodos de classificação adotados. Um método de classificação é um meio de produzir uma organização sistemática com base em um vocabulário, seja controlado ou não (PRIETO-DIAZ, 1991). Um vocabulário controlado constitui-se dos termos utilizados para especificar artefatos, extraídos de um conjunto predefinido de atributos. E em um vocabulário não-controlado, os termos são automaticamente extraídos a partir dos textos que descrevem os artefatos (VANDERLEI, 2006). Um método de classificação pode se basear em um ou em ambos os tipos de vocabulários, podendo ser, basicamente, classificação por palavras-chave, por enumeração e por facetas.

De acordo com Prieto-Diaz e Freeman (1987), um método de classificação é um meio de produzir uma organização sistemática com base em um vocabulário controlado e indexado estruturalmente. Desta forma, é possível dizer que um vocabulário controlado constitui-se dos termos utilizados para especificar os artefatos, extraídos de um conjunto predefinido de atributos que representa um determinado domínio. Em um vocabulário não-controlado, os termos são automaticamente extraídos de textos que descrevem um componente por meio de linguagem natural.

Um método de classificação baseada em vocabulário controlado pode ser definido de duas formas: classificação enumerativa e classificação em facetas. Para o vocabulário não controlado é utilizada a indexação de texto livre, resultando em uma indexação automática a partir de termos mais relevantes recuperados. Esse método é conhecido como classificação por palavras-chave (PRIETO-DIAZ; FREEMAN, 1987; VANDERLEI, 2006).

Esses métodos de classificação são apresentados a seguir.

a) **Classificação por enumeração ou classificação orientada a objetos**

Esse método propõe que os artefatos sejam atribuídos a classes hierárquicas mutuamente exclusivas. De acordo com Vanderlei (2006), a classificação enumerativa se baseia na submissão de informações para uma estrutura hierárquica pré-definida. Os problemas envolvidos em utilizar esse método incluem inflexibilidade inerente e dificuldade em entender grandes hierarquias, já que a estrutura pré-definida das categorias apenas permite a introdução de novos elementos de forma sequencial.

Métodos de estruturação (indexação) de repositórios podem ser divididos em duas categorias principais de indexação (SHIVA; SHALA, 2007): manual (por exemplo, classificação por enumeração e facetas) e automática, cujos métodos mais comuns são os

relacionados à indexação em texto livre.

Enumeração envolve definir uma estrutura hierárquica de categorias e subcategorias definindo uma árvore hierárquica. Esse método é fácil de usar, mas é inflexível e de mudanças difíceis à medida que o domínio de classificação evolui. E, além disso, requer conhecimento do domínio para que possa ser usado (SHIVA; SHALA, 2007).

b) Classificação baseada em palavras-chave

Trata-se de um método baseado em um vocabulário não-controlado, no qual os artefatos de software são classificados por palavras-chave extraídas automaticamente de textos contidos nos próprios artefatos ou mesmo fornecidas pelo usuário.

Entre as técnicas usadas para recuperação de texto Salton (1989) sugere: pesos e frequências para distinguir a importância de palavras-chave; listas de parada para eliminar palavras sem grande importância, mas que geralmente são utilizadas (preposições, por exemplo); dicionários de palavras (palavras semelhantes); truncamento de termos para extrair as raízes e padronizar o uso de palavras; indexação estatística de texto para automatizar a classificação de documentos; e aglomeração de documentos para melhorar a recuperação de documentos relacionados.

Métodos de texto livre são simples para construir e recuperar, mas precisam de grande quantidade de texto disponível para conseguir uma boa precisão nas buscas, sendo melhor aplicáveis para domínios com extensa documentação (MAAREK; SMADJA, 1989; FRAKES; POLE, 1994). A classificação por palavras-chave é menos dispendiosa, pois não requer intervenção humana no processo de indexação de documentos (FRAKES; POLE, 1994).

Uma dificuldade que pode ocorrer com classificação por palavras-chave, são os sinônimos que podem ser utilizados tanto por quem classifica como por quem busca um componente. Isso porque nem sempre quem classificou o componente é quem irá recuperá-lo, podendo dificultar a recuperação de componentes.

c) Classificação por facetas

Segundo Prieto-Diaz (1991), uma faceta pode ser vista como uma propriedade predefinida do componente que deve aceitar valores predefinidos. Uma faceta é um par atributo-valor, no qual há atributos definidos e vários valores associados a cada atributo. A classificação baseada em facetas define um conjunto mutuamente exclusivo de combinação de facetas que descrevem os componentes do domínio. Cada faceta pode ser descrita por um número de termos diferentes. Os usuários buscam por componentes escolhendo um termo que descreve cada uma das facetas. Para ser usada, os usuários precisam estar familiarizados com os termos. E, também, pode não estar claro qual é a combinação de termos a serem usados em

uma pesquisa (SHIVA; SHALA, 2007). Essa técnica de classificação consiste em categorizar os artefatos pela síntese de valores das facetas.

De acordo com Vanderlei (2006), na classificação por facetas os usuários classificam os componentes, especificando um ou mais valores para cada um dos atributos (facetas), características que um componente pode possuir. Facetas são mais flexíveis, precisas e se enquadram melhor para repositórios em constante expansão, se comparadas com esquemas enumerativos. Entretanto, este tipo de classificação é ideal somente para repositórios que armazenam um conjunto homogêneo de componentes. Conforme novos componentes são armazenados eles devem ser classificados em função dos valores predefinidos, sendo um método adequado apenas para componentes de um domínio restrito.

Caldiera e Basili (1991) destacam que enquanto facetas facilitam a sintetização e a combinação de valores para representar componentes, é difícil para os usuários encontrar a combinação correta de valores que descrevem a informação procurada, especialmente em espaços de informação grandes e/ou complexos.

2.4 TRABALHOS RELACIONADOS

Um dos pontos de maior importância deste trabalho é a recuperação de artefatos de software para que possam ser reconfigurados e/ou reutilizados em outros projetos. Contudo, para que esse processo seja viável, é essencial a utilização de maneiras eficientes para classificar e recuperar os artefatos.

Os primeiros trabalhos voltados para busca e recuperação de artefatos de software reutilizáveis tiveram seu foco voltado para os métodos de classificação desses artefatos (PRIETO-DIAZ, 1991; PODGURSKI; PIERCE, 1993; FRAKES; POLE, 1994). Em seguida outros aspectos começaram a ser analisados e mesmo requisitados, especialmente em relação à recuperação.

Muitos trabalhos têm sido realizados propondo a mescla de métodos já consolidados, como a busca por palavras-chave (pré-definidas ou em texto livre), por enumeração ou hierárquica (entre categorias pré-definidas de classificação) e baseada em facetas.

Esta seção apresenta alguns trabalhos que utilizam esses métodos e que se relacionam ao proposto neste trabalho.

a) Henninger (1994) propôs o sistema denominado CodeFinder, em que a recuperação é sustentada pela reformulação interativa de perguntas e por uma aproximação da

recuperação usando uma rede neural, com um algoritmo capaz de recuperar os componentes de software relacionados a uma determinada pergunta, feita por palavras-chave, frases ou afinidades léxicas.

b) Redolfi et al. (2005) identificaram um conjunto de funcionalidades para um repositório de componentes de software visando oferecer suporte às atividades de desenvolvimento baseado em componentes. E propõem um repositório que atende a essas funcionalidades, incluindo busca e recuperação, controle de versão e gerenciamento de controle de versão.

c) Garcia et al. (2006) propuseram um mecanismo denominado Maracatu que permite o reuso de componentes de software, por meio de buscas em repositórios de código-fonte de projetos que são realizadas pela combinação de métodos de recuperação por palavras-chave e facetas. Os autores mostraram que a combinação de métodos utilizados por eles apresenta taxa de recuperação e precisão melhores em relação ao uso dos métodos separadamente.

d) Vanderlei et al. (2007) combinam técnicas originais de busca por palavras-chave e facetas com *folksonomia*. *Folksonomia* é uma forma relacional de categorizar e classificar informações disponíveis na *web* sejam elas representadas por meio de textos, imagens, áudio, vídeo ou qualquer outro formato (RUFINO, 2009). O uso de *folksonomia* permite maior aproximação da realidade do usuário, seja no cadastro de artefatos ou na sua recuperação.

e) Petrov e Buchmann (2008) adotaram o uso de palavras-chave, permitindo melhor representação dos requisitos do componente. Para auxiliar o usuário na classificação e recuperação dos componentes, são apresentadas questões, cujas respostas serão as palavras-chave pré-definidas para ambas as situações. As palavras-chave poderiam também ser diretamente informadas pelo usuário, indicando atributos desejáveis e não desejáveis para o componente de software.

f) Batista Junior e Domingues (2010) definiram um método que gera automaticamente uma Forma Gráfica de Representação (FGR) para descrições de componentes de software em linguagem natural ou por questões de usuários. A busca de componentes na biblioteca é suportada pela avaliação automática da combinação entre a FGR de uma pergunta do usuário e as FGRs das descrições dos componentes da biblioteca. A análise da pergunta é realizada com a extração das suas palavras-chave.

A partir destes trabalhos é possível perceber que o foco das pesquisas está bastante voltado para a eficácia na recuperação de componentes de software a partir de um repositório, de forma a tornar essa tarefa mais prática, flexível e efetiva, retornando o artefato desejado.

Os trabalhos apresentados se referem ao armazenamento e recuperação de componentes de código a fim de oferecer suporte ao desenvolvimento de software baseado em componentes, o que reforça a relevância desta proposta, que se refere aos artefatos de software e não apenas componentes de código.

Neste trabalho adotou-se o uso de palavras-chave e descrição textual porque elas poderiam melhor representar os requisitos do artefato seja para inclusão no repositório ou para sua recuperação. Como, os artefatos são classificados e descritos por palavras-chave, a busca seria mais precisa se também fosse realizada dessa forma. A busca pode depender de conhecimento do usuário sobre tecnologias ou conceitos envolvidos no artefato requerido. Além disso, o resultado da busca poderá ser mais preciso se o usuário informar exatamente os atributos ou requisitos que ele espera que o artefato tenha e os que não deve ter.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os recursos utilizados para a modelagem e a implementação do sistema resultante do desenvolvimento deste trabalho. Neste capítulo também é apresentado o método, que é a sequência dos passos principais para o desenvolvimento do trabalho.

3.1 MATERIAIS

Materiais são as ferramentas e as tecnologias, que incluem ambientes de desenvolvimento, linguagens de programação e de marcação e os utilizados para realizar as atividades do ciclo de vida: da definição dos requisitos à implementação do sistema. Para a modelagem e a implementação do sistema foram utilizados os seguintes materiais:

a) A ferramenta Astah Community (ASTAH, 2012) para a modelagem do sistema;

b) A ferramenta CaseStudio (CASESTUDIO, 2012) para a modelagem do banco de dados;

c) A ferramenta Visual Studio 2010 como IDE (*Integrated Development Environment*) de desenvolvimento;

d) A tecnologia Asp.Net (ASP.NET, 2012) como linguagem de programação, agregada por C# (SANT'ANNA, 2012), HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheets*), JavaScript, Ajax (GARRETT, 2005) para validações de formulários e jQuery User Interface (JQUERY, 2012);

e) A ferramenta IbExpert (IBEXPERT, 2012) para administração do banco de dados;

f) A tecnologia Firebird (FIREBIRD, 2012) para o banco de dados;

g) A ferramenta Crystal Report (CRYSTAL, 2012) para o desenvolvimento dos relatórios;

h) A ferramenta IIS (IIS, 2012) para servidor *web*.

Essas tecnologias e ferramentas estão a seguir descritas.

1) Astah Community

Astah Community (ASTAH, 2012) é uma ferramenta de modelagem gratuita para modelagem de sistemas orientados a objeto. Essa ferramenta é baseada na elaboração de diagramas em notação UML 2.0 (*Unified Modeling Language*) e pode gerar código em Java,

C# e C++. O código na linguagem Java é gerado apenas para a definição da classe e de seus atributos e métodos. Para que a geração desse código possa ser feita, o diagrama de classes deve conter as classes com a especificação dos seus atributos e os métodos devem ter os seus parâmetros definidos. Essa ferramenta também permite engenharia reserva, ou seja, obter o diagrama a partir das classes implementadas em Java.

2) CaseStudio

Case Studio 2 (CASESTUDIO, 2012) é uma ferramenta de modelagem de banco de dados. CASE que significa *Computer Aided Software Engineering* é uma técnica que visa facilitar o desenvolvimento de sistemas de software por meio do uso de ferramentas.

3) Visual Studio 2010

O Microsoft Visual Studio 2010 é um ambiente integrado de desenvolvimento que inclui um pacote de programas da Microsoft para desenvolvimento de *software*, permitindo o uso de diversas linguagens como, por exemplo, C#, C++, C, Java e Visual Basic (MICROSOFT, 2012).

4) ASP.NET

ASP.NET é um *framework web* para desenvolvimento de aplicações *web* usando HTML, CSS e JavaScript (ASP.NET, 2012). ASP.NET é baseado no *framework* .NET herdando todas as suas características. As aplicações para esse *framework*, que é livre, podem ser escritas em diversas linguagens, como C# e Visual Basic .NET.

5) C#

A linguagem C# foi desenvolvida pela Microsoft em conjunto com a arquitetura .NET. Sant'Anna (2012) refere-se a C# como a linguagem de referência do .NET, porque a maior parte das classes do .NET Framework e até mesmo o compilador JScript foram desenvolvidos em C#. C# usa como base a sintaxe de C++. Isto significa que os seus elementos como declaração de variáveis, métodos e estruturas de controle (decisão e repetição) são muito semelhantes à linguagem C++. Além da origem na linguagem C++, o C# possui várias características em comum com a linguagem Java e de acordo com esse autor implementa melhorias em relação à linguagem Java. Essas características são (SANT'ANNA, 2012):

a) Modelo de orientação a objetos baseado em herança simples de classes, isto é, não é permitida herança múltipla.

b) Gerenciamento de memória automático com desalocação de memória que não está mais em uso.

c) Tipagem forte, as variáveis precisam ter um tipo explícito quando declaradas.

d) Executa em um ambiente gerenciado, no qual a segurança e a integridade das operações efetuadas pelos programas podem ser asseguradas.

e) Suporte a *reflections*, que fornece informação de tipos em tempo de execução.

f) Sobrecarga de operadores, que é útil na realização de cálculos científicos, por permitir tratar números complexos, vetores e matrizes com a notação dos operadores aritméticos tradicionais como "+" e "*".

g) Operadores de conversão, para converter valores de um tipo para outro. Em C# existem operadores de conversão implícitos e explícitos, que exigem o operador de *cast*.

h) Unificação do sistema de tipos. Todos os valores podem ser atribuídos a uma variável do tipo *object* em um processo chamado *boxing*.

i) Declaração de um tipo ponteiro para método, chamado *delegate*. Um *delegate* contém o endereço da função e também do método que a implementa. Os *delegates* permitem que uma classe chame métodos em outras classes sem que a mesma seja derivada de um ancestral conhecido.

j) Tipo decimal para representar valores monetários, minimizando erros de arredondamento e representação.

k) Os métodos devem ser explicitamente declarados com a palavra reservada "virtual". Existe um protocolo específico para indicar se um método de classe derivada reimplementa um método virtual (*override*) ou o torna não-virtual (*new*).

6) jQuery User Interface

jQuery User Interface prove um conjunto de *plugins* de interação, *widgets* de interface e efeitos visuais que usam o estilo jQuery. jQuery provê uma arquitetura baseada em eventos e foco em padrões *web* e acessibilidade (JQUERY, 2012).

jQuery é uma biblioteca JavaScript que simplifica a identificação de *tags* em documentos HTML, a manipulação de eventos, o desenvolvimento de animações e de interações Ajax facilitando a implementação de sistemas *web*. Essa biblioteca possui as seguintes características (SILVA, 2010):

a) Uso de seletores CSS para localizar elementos componentes da estrutura de marcação HTML da página.

b) Arquitetura compatível com a instalação de *plugins* e extensões em geral.

c) Indiferença às inconsistências de renderização entre navegadores.

d) oferece interação implícita, isto é, não há necessidade de construção de estruturas de repetição para localização de elementos no documento.

e) Admite programação encadeada, ou seja, cada método retorna um objeto.

f) É extensível, pois admite criação e inserção de novas funcionalidades em bibliotecas existentes.

jQuery se destina a adicionar interatividade e dinamismo às páginas *web*, facilitando (SILVA, 2010): adicionar efeitos visuais e animações; acessar e manipular o DOM (*Document Object Model*); buscar informações no servidor sem necessidade de recarregar a página; alterar conteúdos; modificar apresentação e estilo e simplificar tarefas específicas de JavaScript.

jQuery foi criada com a preocupação de ser uma biblioteca em conformidade com os padrões *web*, ou seja, compatível com qualquer sistema operacional, navegador e com suporte total para CSS3. Essa biblioteca está em acordo com as diretrizes do W3C (*World Wide Web Consortium*). Contudo, cabe ao desenvolvedor escrever os *scripts* em conformidade com essas diretrizes (SILVA, 2010).

7) Ajax

Ajax é um conjunto de tecnologias e incorpora (GARRETT, 2005):

a) Apresentação baseada em padrões usando XHTML (*eXtensible* HTML) e CSS.

b) Visualização dinâmica e interação usando DOM.

c) Troca e manipulação de dados usando XML (*Extensible Markup Language*) e XSTL (*Extensible Style Sheet Language*).

d) Recuperação assíncrona de dados usando *XMLHttpRequest* que possibilita comunicação com o servidor sem necessidade de redesenhar a tela inteira a cada interação do usuário.

e) JavaScript utilizada para modificar diretamente no navegador a interface com o usuário.

8) HTML

HTML é uma linguagem de marcação para a criação de páginas *web*. Essa linguagem provê um meio para descrever a estrutura de informação baseada em texto em um documento. Isso é feito pela definição de tipos de formatação de texto para cabeçalho, parágrafos, listas e outros; e pela inclusão de imagens e outros objetos. Mikkonen e Taivalsaari (2008)

consideram HTML como um dos componentes fundamentais das páginas *web*.

9) CSS

CSS é uma linguagem de estilo que é usada para descrever aspectos de apresentação de um documento escrito em uma linguagem de marcação. Um estilo permite estabelecer regras de formatação de uma página *web*, como, por exemplo, cores e fontes, que são definidos independentemente do conteúdo da página *web*. Para Mikkonen e Taivalsaari (2008) CSS também é um dos componentes fundamentais das páginas *web*.

10) JavaScript

JavaScript possibilita a execução de código em um documento *web*. JavaScript é uma linguagem de *script*, caracterizada por suas instruções serem interpretadas e com execução vinculada a outras linguagens. As linguagens de *script* como JavaScript são um dos componentes fundamentais das páginas *web* (MIKKONEN; TAIVALSAARI, 2008).

11) IBExpert

IBExpert (IBEXPERT, 2012) é uma ferramenta para administração de bancos de dados Interbase e Firebird que permite criar e gerenciar usuários e tabelas. A interface do aplicativo é simples, consistindo, basicamente, de uma barra de ferramentas e do DB (*Data Base*) Explorer.

11) Firebird

O Firebird é um Sistema de Gerenciamento de Banco de dados (SGBD) que possui suporte nativo para diversos sistemas operacionais (incluindo Windows, Linux, Solaris e MacOS), *triggers* de conexão e transação (CANTU, 2012).

Dentre os recursos do Firebird destacam-se: transações compatíveis com ACID (Atomicidade, Consistência, Isolamento, Durabilidade); integridade referencial; linguagem nativa para *stored procedures* e *triggers* (PSQL (PostgreSQL)); suporte para funções externas; diversas formas de acesso ao banco de dados, como: nativo/API (*Application Programming Interface*), dbExpress, ODBC (*Open Data Base Connectivity)*, OLEDB (*Object Linking and Embedding Data Base)*, .Net provider, JDBC (*Java Database Connectivity)* nativo tipo 4, Python *module*, PHP, Perl; cópias de segurança incrementais e tabelas temporárias.

12) Crystal Reports

Crystal Reports é uma ferramenta para a criação de relatórios que faz parte do Visual Studio desde 1993 e atualmente está integrada diretamente ao ambiente de desenvolvimento (MSDN, 2012). Essa ferramenta permite criar relatórios com integração com Flash, gráficos interativos, e opções de exportação. Os relatórios podem ser visualizados em aplicativos no Microsoft Office, *web*, *email* ou incorporados na aplicação (CRYSTAL, 2012). O Crystal Reports permite criar relatórios mais elaborados, como os que possuem a inserção de fórmulas, por exemplo.

13) Internet Information Services

O *Internet Information Services* (IIS) é um servidor *web* criado pela Microsoft. IIS é flexível, seguro e fornece gerenciamento fácil para servidores *web* (IIS, 2012). Uma de suas características mais utilizadas é a geração de páginas HTML dinâmicas, que diferentemente de outros servidores *web*, usa tecnologia proprietária, o ASP (*Active Server Pages*), mas também pode usar outras tecnologias com adição de módulos de terceiros.

3.2 MÉTODO

O processo de modelagem e de implementação do aplicativo desenvolvido tem como base o modelo sequencial linear como descrito em Pressman (2002), complementado pelo processo unificado (BLAHA et al., 2006).

O processo unificado auxiliou a definir ciclos interativos de modelagem e de implementação. Essa forma de procedimento foi adotada porque os requisitos do sistema não estavam completamente definidos no início do projeto e era necessário estudar as tecnologias utilizadas. Além disso, os requisitos relacionados à busca foram definidos após terem sido implementados os cadastros para inserção dos artefatos no repositório. Como havia necessidade de estudo da tecnologia optou-se pela implementação de cadastros básicos na primeira iteração. Assim, foi possível também experimentar modelos para a interface. O Quadro 1 apresenta as principais realizações de cada iteração.

Processos	Iteração 1	Iteração 2	Iteração 3	Iteração 4
Requisitos do sistema	Visão geral do sistema do ponto de vista do usuário.	Definição dos requisitos relacionados à inserção dos artefatos no repositório.	Definição dos requisitos relacionados à busca de artefatos.	
Análise e projeto	Diagramas com a visão geral do sistema (processos de reuso, armazenamento e recuperação de artefatos).	Modelagem dos requisitos, casos de uso, campos de formulários, classes, banco de dados e representação da ideia para a interface de busca.	Modelagem dos requisitos para a busca de artefatos.	Revisão de classes e banco de dados.
Implementação	De alguns cadastros básicos para estudo da tecnologia.	Dos cadastros simples. Dos cadastros complexos (com referências).	Dos demais requisitos relacionados à inserção de dados no banco de dados.	Da busca por artefatos.
Testes	De verificação se o modelo da interface é adequado, realizados pela orientadora.	De código e realizados pela autora deste trabalho, visando verificar se o código implementado atende os respectivos requisitos.	De código e de verificação de atendimento aos requisitos.	De usuário visando verificar os requisitos e da forma de interação realizados pela orientadora.

Quadro 1 – Processos e iterações

A seguir estão descritos os processos definidos para a modelagem e a implementação do sistema. Foram definidos apenas processos técnicos, embora o processo unificado determine a existência de processos gerenciais.

a) Levantamento de requisitos

O levantamento dos requisitos iniciou com a definição da visão geral do sistema que foi elaborada a partir dos interesses da orientadora deste trabalho. Essa visão foi representada por meio dos principais conceitos envolvidos, utilizando a ferramenta Astah Community. Esses conceitos relacionados entre si forneciam uma ideia das funcionalidades básicas pretendidas para o sistema.

Em seguida foi realizada uma busca por publicações relacionadas aos interesses representados por essa visão geral. Essa visão geral auxiliou a limitar os trabalhos relacionados que permitiu definir claramente o interesse em repositório de artefatos de todo o ciclo de vida de software e não somente código. Assim, evidenciou-se a importância dos metadados a serem utilizados no cadastro dos artefatos para que os artefatos ficassem bem

caracterizados e a sua busca pudesse ser mais efetiva. Efetiva no sentido de melhor atender aos interesses de busca do usuário. Esses interesses são representados pelos critérios informados na busca.

Muitos dos trabalhos relacionados se referem aos repositórios e mecanismos de buscas para componentes de código, que o código fonte é armazenado. Esses componentes podem permitir buscas no conteúdo do artefato. Contudo, esse tipo de busca é dificultado se o artefato armazenado é um documento em formato não editável que representa um diagrama com a visão do sistema do ponto de vista do usuário. Por isso evidenciou-se a importância dos metadados para o armazenamento de artefatos no escopo e contexto pretendido para esse trabalho.

Esses trabalhos relacionados foram complementados com publicações que pudessem fornecer a fundamentação conceitual da ideia definida como escopo do repositório e do mecanismo de busca. Assim, outros materiais bibliográficos foram localizados e estudados.

Com base no diagrama que representa a visão geral do sistema, nos conceitos provenientes do referencial teórico e nos trabalhos considerados relacionados foram extraídos os principais requisitos funcionais e não funcionais. Esses requisitos foram complementados e revisados em iterações posteriores.

b) Análise e projeto do sistema

Os requisitos serviram de base para a definição dos casos de uso do sistema. Esses casos de uso foram documentados gerando informações para a definição do banco de dados e das classes. Uma listagem dos cadastros foi elaborada, juntamente com a definição de relatórios para o sistema. Essa listagem apresenta os campos dos formulários indicando o significado dos mesmos, por meio de uma descrição textual, o tipo de dado a ser armazenado e se o preenchimento do campo é obrigatório. Essas informações auxiliaram no momento de definir a interface dos formulários e também na realização dos testes unitários, como, por exemplo, a obrigatoriedade de preenchimento e a validação do tipo de dado.

Uma representação gráfica dos campos das telas de busca também foi elaborada, visando descrever a distribuição dos campos para a inserção dos dados e a apresentação dos resultados de busca para o usuário.

Em seguida, o digrama de classes e o diagrama de entidades e relacionamentos do banco de dados foram definidos. O diagrama de classes foi definido com a representação das classes básicas. O diagrama de entidades e relacionamentos foi complementado com quadros de descrição das tabelas visando indicar os campos das tabelas, o tipo de dado a ser armazenado em cada campo, se o campo é chave primária e/ou estrangeira e observações. Em

observações foi descrito se o campo possui um valor padrão e a origem do mesmo, por exemplo. A origem indica a tabela que possui os dados a serem utilizados no preenchimento do campo, indicando, assim, que possivelmente um componente como, por exemplo, uma caixa de combinação, seria utilizada no respectivo formulário. Os dados para esse componente seriam provenientes da tabela indicada.

Para a modelagem dos casos de uso e do diagrama de classes foi utilizada a ferramenta Astah Community e para a representação das entidades e relacionamentos do banco de dados foi utilizada a ferramenta Case Studio.

c) Implementação

A implementação foi realizada utilizando a ferramenta Visual Studio 2010. Inicialmente foram implementados os cadastros simples visando o aprendizado das tecnologias. Nas iterações posteriores, como apresentado no Quadro 1, requisitos mais complexos foram implementados.

Durante a implementação testes informais (a referência a informais decorre do fato que não foram elaborados planos de testes) foram realizados visando identificar erros de codificação.

d) Testes

Os testes foram realizados de maneira informal e se centraram em localizar erros de codificação e na verificação do atendimento dos requisitos definidos para o sistema. Os testes relacionados à codificação foram realizados pela autora deste trabalho. Os testes de requisitos e de uso do sistema foram realizados pela autora deste trabalho e a orientadora. Esses testes consistiram em verificar se as funcionalidades pretendidas e definidas para o sistema estavam sendo atendidas e validar campos de formulários. E, ainda, se a forma de distribuição dos dados nos formulários estava adequada, ou seja, o leiaute dos formulários.

Vários testes de uso do sistema foram realizados, pela orientadora, pela autora desse trabalho e por dois alunos da área de informática do Câmpus. Esses testes indicaram ajustes e pequenos complementos nos requisitos e na forma de implementação.

4 RESULTADO E DISCUSSÃO

Este capítulo apresenta o que foi obtido como resultado do trabalho. Na Seção 4.1 está a descrição do sistema. A Seção 4.2 contém os principais diagramas e as descrições que modelaram o problema e a respectiva solução computacional implementada é apresentada na Seção 4.3 por meio da forma de uso do sistema e na Seção 4.4 estão exemplos da codificação realizada. A Seção 4.5 apresenta uma breve discussão sobre o trabalho desenvolvido.

4.1 APRESENTAÇÃO DO SISTEMA

Os artefatos armazenados no repositório são caracterizados por atributos que representam as informações consideradas necessárias para descrevê-los com o objetivo de facilitar a sua recuperação, visando promover reuso. O modelo proposto utiliza técnicas de indexação baseadas em vocabulário controlado, como a classificação por palavras-chave e por descrição textual.

A busca dos artefatos é realizada por palavras-chave informadas ou pré-definidas nos atributos de identificação do artefato e na sua descrição (neste caso a busca é feita por palavras-chave em texto livre). O usuário indica palavras-chave que o artefato deve ou não conter. O uso combinado de técnicas distintas visa prover uma forma de localizar, além dos artefatos que melhor atendem aos critérios de busca, os artefatos com características próximas a esses critérios.

Brito et al. (2009) citam que alguns sistemas de repositório de componentes adotam técnicas de indexação que geram automaticamente informações de representação dos componentes, mas, mesmo assim, os componentes são localizados por busca exaustiva na coleção de informações (GIRARDI; IBRAHIM, 1994). Por exemplo, em abordagens baseadas em ontologias (SIVASHANMUGAM et al., 2003) a inferência sobre a coleção de conceitos é bastante custosa (DING et al., 2004). Brito et al. (2009) ressaltam que sistemas que adotam métodos de recuperação clássicos, baseados na indexação de termos, se sobressaem em eficiência quando comparados aos sistemas que empregam mecanismos de inferência.

Para minimizar os problemas decorrentes de uso de palavras-chave diferentes para cadastrar e buscar um mesmo artefato, na proposta deste trabalho, critérios para inserção e orientações para busca foram definidos. Os critérios se referem às padronizações recomendadas e a existência de um dicionário de sinônimos para ser consultado quando da

inserção do artefato. As orientações explicam o contexto e exemplificam o conteúdo de cada campo de cadastro do artefato. Além disso, sugere-se que a inserção seja cooperativa, ou seja, os dados de cadastro de um novo artefato sejam verificados por usuários distintos. Ainda assim, para que o reuso seja efetivo o desenvolvimento de software deve estar inserido em um contexto de desenvolvimento de software para e com reuso. Para reuso significa que os artefatos são produzidos visando reuso e com reuso está relacionado ao desenvolvimento de software utilizando artefatos existentes.

Desta forma, deve haver integração entre as atividades de produção de software e de componentes e o armazenamento e a disponibilização dos artefatos para reuso. A Figura 1 expressa essa integração por meio de processos. Esses processos são representados em notação *Business Process Modeling Notation* (BPMN) (WHITE, 2004). Nessa notação, retângulo com cantos arredondados representa atividade ou subprocesso, círculo com borda delgada indica o início do processo e círculo com borda espessa o seu final. A interligação entre processos é representada por seta pontilhada direcionada e entre as atividades de um processo por seta com traço contínuo.

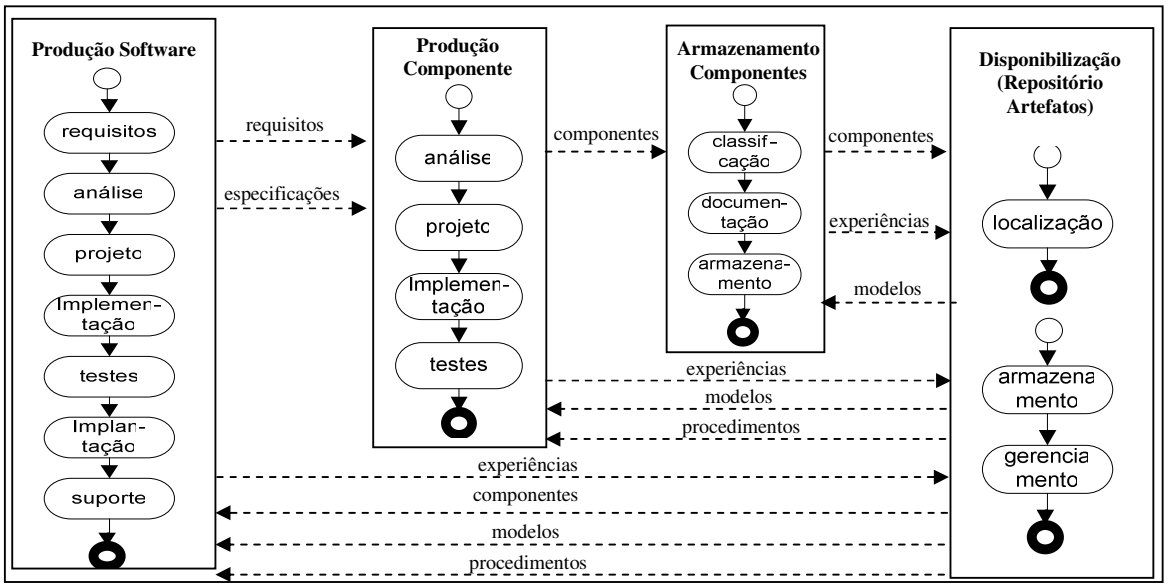

Figura 1 – Processos para produção de software com e para reuso

Fonte: Borsoi et al. (2011, p. 12).

Na Figura 1 está representado que a fábrica de software possui um processo de implementação que é suportado por um processo de produção de componentes. O processo de implementação envia requisitos e especificações para o processo de produção de componentes e este os desenvolve com o auxílio de modelos e procedimentos provenientes do repositório. Os componentes produzidos são disponibilizados no repositório por meio de um processo de

armazenamento. O processo de produção implementa software reusando componentes, modelos e procedimentos. O conhecimento gerado na realização das atividades (denominado experiência) pode ser documentado e armazenado no repositório e disponibilizado para reuso com o nome de procedimento. No processo de implementação, tanto os modelos quanto os componentes de software podem ser adaptados ao contexto do projeto em desenvolvimento.

4.2 MODELAGEM DO SISTEMA

Para serem recuperados de forma a melhor atender aos critérios de busca, os componentes precisam ser devidamente cadastrados. Redolfi et al. (2005) classificaram, com base em diversos autores, as informações sobre componentes de software em sete grupos: identificação, uso, maturidade, documentação, tecnologia, alteração e controle de qualidade. Considerando esses grupos, no Quadro 2 são apresentados os grupos de requisitos propostos neste trabalho para o armazenamento de artefatos. Esses grupos têm como base o apresentado em Redolfi et al. (2005), mas outras denominações e elementos são definidos.

Grupo	Elementos
Identificação	Identidade; Nome; Versão; Local de disponibilização (repositório para componentes de código, *link web*).
Descrição	Descrição; Tipo (componente, documento, procedimento); Classificação (baseada em palavras-chave); Requisitos para uso (o que é necessário para usar o artefato); Histórico de alterações.
Acesso	Disponibilidade (público ou privado); Leitura e escrita; Descrição da interface (se componente); Tecnologia (linguagem, ferramenta de desenvolvimento).
Qualidade	Testes de qualidade realizados e respectivos resultados; aspectos de qualidade, incluindo problemas identificados e restrições apresentadas.
Relacionamentos	Pode ser usado com (quais artefatos podem ser utilizados junto); Composto por (artefatos que o compõem); Depende de (artefatos que necessariamente precisam estar vinculados para uso); Projetos que o utilizam.

Quadro 2 – Agrupamentos de informações sobre artefatos

Os grupos apresentados no Quadro 2 representam os agrupamentos de requisitos para identificar os artefatos. Os elementos que compõem esses grupos são exemplos de requisitos que poderiam ser utilizados como metadados dos artefatos armazenados.

Na Figura 5 está o modelo conceitual proposto para o repositório de artefatos tendo como base o Quadro 2. Essa figura contém um conjunto de conceitos relacionados que explicitam a estrutura e a organização do repositório. A notação utilizada é de

relacionamentos (generalização, associação e agregação) entre classes da UML. Contudo, ressalta-se que na Figura 2 o que é representado como classes são apenas conceitos relacionados ao domínio do repositório de artefatos proposto. Portanto, esses conceitos representados como classes não necessariamente serão classes e/ou tabelas de banco de dados do sistema.

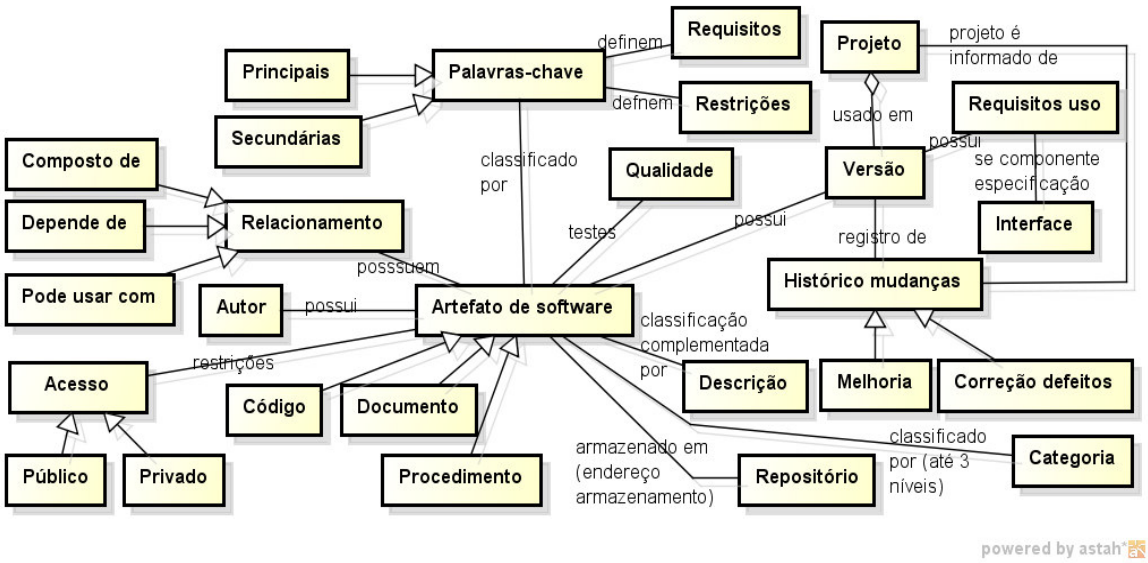

Figura 2 – Visão geral do repositório de artefatos
Fonte: Borsoi et al. (2011, p. 13).

A visão geral do repositório apresentada na Figura 2 indica que o repositório provê o armazenamento de componentes de código, documentos produzidos durante o ciclo de vida (exemplo: plano de testes, diagrama de casos de uso), modelos de documentos (padrões e orientações) e procedimentos (o conhecimento gerado com a realização de atividades). As palavras-chave são utilizadas para definir requisitos do artefato e restrições (requisitos que o artefato não atende), podendo ser principais e secundárias. A descrição é utilizada para complementar a classificação baseada em palavras-chave. As palavras-chave secundárias permitem localizar artefatos que não atendem os requisitos ou critérios de busca completamente, mas que se aproximam deles.

Os relacionamentos definem as conexões entre os artefatos e eles são estabelecidos quando da sua inserção no repositório. Em uma busca, os relacionamentos vinculados a um artefato são indicados.

O histórico de mudanças contém as alterações realizadas em um artefato e o seu tipo, que podem ser correção de defeitos ou melhoria. Correção de defeitos indica que todos os projetos que utilizam o referido artefato deveriam atualizá-lo. E melhorias podem agregar ou

não funcionalidades e gerar ou não inconsistências em projetos que utilizam o referido artefato. O registro dos artefatos utilizados nos respectivos projetos é mantido para, por exemplo, informar os projetos cujos artefatos passaram por alterações de melhoria e de correção de erros.

Na Figura 3 está representada a definição conceitual para a busca de artefatos no repositório. A forma de representação dessa figura é semelhante à Figura 5. O usuário escolhe entre palavras-chave existentes ou informa as palavras desejadas utilizadas para buscas no texto da descrição do artefato e nos atributos que o categorizam.

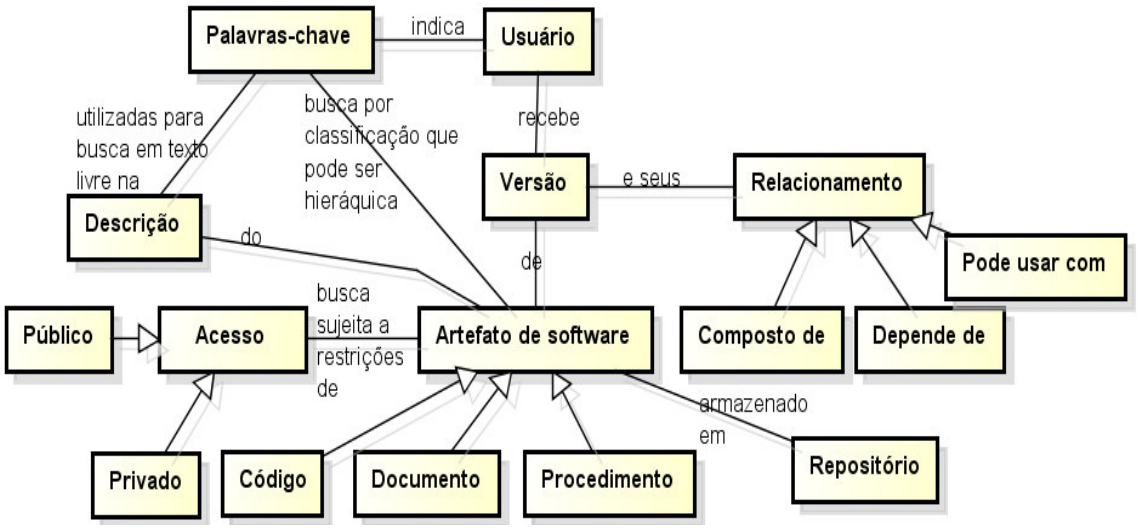

Figura 3 – Visão geral da busca no repositório de artefatos

Fonte: Borsoi et al. (2011, p. 14).

Os artefatos resultantes de uma busca são apresentados em classificação descendente ao atendimento dos critérios de busca. E juntamente com cada artefato são informados os artefatos relacionados. Os artefatos de acesso público podem ser localizados em buscas por qualquer usuário. Os artefatos de acesso privado são disponibilizados para o autor do artefato.

A partir das representações das Figuras 2 e 3 que definem a ideia geral do armazenamento e da busca dos artefatos foram identificados os requisitos do sistema. O Quadro 3 apresenta os requisitos funcionais identificados.

(continua)

Identificação	Nome	Descrição
RF01	Cadastro de usuários	Usuários que podem ter acesso ao sistema. Atributos identificados: IdUsuario Nome Login Senha E-mail Categoria. As três categorias para usuário são: Administrador – acesso a todas as funcionalidades do sistema. Usuário com *login* – permissão de consulta a todos os artefatos públicos, de inclusão de novos artefatos ou de inclusão resultado de alteração aos seus artefatos aos quais ele tem acesso. Usuário sem login – permissões somente para realizar buscas no sistema para artefatos com tipo de acesso público.
RF02	Cadastro de grupos de usuários	Grupos de usuários. Esses grupos podem ser utilizados para atribuir permissões de acesso em versões de artefatos. Pode ser utilizado para cadastrar uma empresa que utiliza o repositório, por exemplo. A esse grupo estarão vinculados usuários cadastrados no sistema. Atributos identificados: IdGrupo Nome Descrição
RF03	Associar usuários a grupos	Um usuário pode pertencer a mais de um grupo. Atributos identificados: IdGrupo IdUsuario
RF04	Vincular usuários a versão de artefatos	Atribuição de permissão de acesso de usuários a artefatos. Define as versões dos artefatos que o usuário pode ter acesso. Atributos identificados: IdUsuario (sendo que usuário pode ter sido selecionado por meio do grupo ao qual ele pertence) IdVersaoArtefato
RF05	Cadastro dos artefatos	Artefatos armazenados no repositório. Atributos identificados: IdArtefato Nome Descrição IdTipo (corresponde ao tipo do artefato) IdSubTipo (corresponde ao subtipo do artefato) IdSubsubTipo (corresponde ao subsubtipo do artefato) IdFerramenta (o que foi usado para gerar o artefato) Observação (informações complementares)

(continua)

Identificação	Nome	Descrição
RF06	Cadastro de tipos de artefatos	São os agrupamentos de tipos dos artefatos: por exemplo, componente de código, documento e outros. Atributos identificados: IdTipo Nome. Identifica o tipo. Exemplos: Componente de código (exemplos: classe, tabela, DLL, componente, módulo, script de BD, gerador de relatórios). –Modelagem (documentos produzidos durante o ciclo de vida. Exemplos de requisitos, de análise, de projeto, testes). Documentos de apoio (documentos utilizados para produzir os artefatos. Exemplos *checklist*, instrumentos de coleta, técnicas, procedimento). Descrição do tipo (complemento ao nome)
RF07	Cadastro de versões de artefatos	As versões dos artefatos, identificando as mudanças, melhorias e correções realizadas. Atributos identificados: IdArtefato IdVersao NomeVersão IdTipoVersao Autor Descrição versão Data da versão IdTecnologia (para artefatos de implementação, como, componente de código na linguagem de programação Java, script de banco de dados Firebird, banco de dados em MySQL) Disponibilidade (pode ser público ou privado, neste caso o acesso é definido pelo cadastro de relacionamento entre artefatos e usuários) Local de armazenamento (o artefato pode não estar disponibilizado no repositório) Motivo da versão. Exemplos: a) Correção de defeitos/erros b) Melhorias, que podem ser b.1) agregam funcionalidades b.1.1) que não geram inconsistências com usos na versão anterior. Um projeto que utiliza a versão anterior pode substituir pela atual sem necessidade de alteração de código, por exemplo. b.2) que geram inconsistências com usos na versão anterior. Um projeto que utiliza a versão anterior, se substituir pela atual há necessidade de alterações na codificação do sistema. b.2.1) Não agregam funcionalidades, mas melhoram aspectos de uso e não geram inconsistências com usos na versão anterior. Um projeto que utiliza a versão anterior pode substituir pela atual. b.2.2) Não agregam funcionalidades, mas melhoram aspectos de uso e geram inconsistências com usos na versão anterior. Um projeto que utiliza a versão anterior não pode substituir pela atual sem que haja alterações na codificação.

(continua)

Identificação	Nome	Descrição
RF08	Cadastro de componente de código	Se o artefato é um componente de código são armazenadas informações complementares ao definido para o cadastro de versão de artefato. Atributos identificados: Objetivo (o que efetivamente o componente faz). Funcionalidades (a abrangência de uso do componente e funcionalidades providas). CaracteristicasNaoFuncionais (requisitos não funcionais atendidos pelo componente). Restrições (aspectos relevantes não atendidos). Interfaces (forma de acesso ao componente, descrição dos dados e seus tipos e de retorno, se for o caso). AcessoAoCodigoFonte (se permite acesso ao código fonte). RequisitosUso (especificações para uso do componente). LocalDocumentação (local de armazenamento de documentação do componente. Pode ser um artefato armazenado no próprio repositório. Nesse caso colocar o nome do mesmo).
RF09	Cadastro de palavras-chave	Palavras utilizadas para classificar o artefato. Atributos identificados: IdPalavra Descrição
RF010	Atribuir palavras-chave a artefatos	Relacionamento entre palavras-chave e artefatos. Atributos identificados: IdPalavra IdArtefato TipoPalavra (Primárias são mais específicas ou Secundárias são menos específicas).
RF11	Cadastro de sinônimos	Palavras que são sinônimos de palavras-chave. É o vínculo entre palavras-chave que são consideradas sinônimo. Atributos identificados: IdPalavraChave IdPalavraChaveSinônimo
RF12	Cadastro de tipos de relacionamentos de artefatos	Relacionamentos entre artefatos armazenados no repositório. Atributos identificados: IdArtefato IdArtefatoRelacionado TipoRelacionamento (exemplos: Composto de (o artefato composto por um conjunto de outros); Depende de (os artefatos necessários para uso ou aplicação); Pode ser usado com (indicação de artefatos que podem ser utilizados em conjunto).
RF13	Cadastro de tipos de versão de artefato	Para o tipo de versão de um artefato. Exemplos: demonstração, *shareware*, *freeware*, paga. Atributos identificados: IdTipoVersao Nome Descrição

(conclusão)

Identificação	Nome	Descrição
RF14	Cadastro de ferramentas ou tecnologias	Cadastro de tecnologias e ferramentas utilizados na definição do artefato ou necessárias para o seu uso. Atributos identificados: IdTecnologiaFerramenta Nome Descrição Versão (no caso de ferramenta)
RF15	Buscar artefatos	Busca de artefatos armazenados no repositório. O usuário fornece os critérios de consulta e a busca é realizada nos artefatos que o usuário tem acesso.
RF16	Cadastro de tipo de relacionamentos	Tipos de relacionamentos entre artefatos (identificados: composto de; depende de; pode ser usado com). Atributos identificados: IdTipoRelacionamento NomeRelacionamento Descrição (observações)
RF17	Cadastro de relacionamentos entre artefatos	Relacionar os artefatos de acordo com um tipo de relacionamento. Atributos identificados: IdTipoRelacionamento IdArtefato IdArtefatoRelacionado
RF18	Cadastro de atributos de qualidade do artefato	Comprovação de qualidade do artefato, incluindo problemas e resultados de testes. Atributos identificados: IdArtefato IdVersaoArtefato TipoAtributo (problema, aspeto positivo, observação) Descrição (descrição da ocorrência do tipo de atributo)
RF19	Cadastro de tipos atributos de qualidade	Características não funcionais que o componente pode ter. São atributos de qualidade. IdCaracteristicaNaoFuncional Nome Descrição
RF20	Cadastro de projetos	Projetos que utilizam artefatos do repositório. Atributos identificados: IdProjeto Nome Descrição Data início Data fim IdArea. Área que o projeto pertence.
RF21	Cadastro de áreas de projetos	Cadastro de áreas dos projetos. Atributos identificados: IdArea Descrição
RF22	Relacionamento de versão do artefato e projeto que o utiliza	As versões de artefatos utilizadas pelos projetos. Atributos identificados: IdArtefato IdVersao IdProjeto

Quadro 3 – Requisitos funcionais

O Quadro 4 apresenta a listagem dos requisitos não-funcionais identificados para o sistema. Os requisitos não funcionais podem explicitar regras de negócio, restrições de acesso e requisitos de qualidade e segurança dentre outros.

(continua)

Identificação	Nome	Descrição
RN01	Cadastro de versão de artefato	Uma versão de artefato pode herdar as permissões da versão anterior. É possível editar uma versão de um artefato e fazer um salvar como para gerar uma nova versão. Assim, serão salvas as permissões da versão anterior. Quando um artefato é consultado, listar as versões que o mesmo possui.
RNF02	Acesso aos artefatos	No cadastro do artefato é necessário informar se o mesmo é de acesso público ou privado. Quando privados podem ser acessado por um ou mais usuários e/ou um ou mais grupos. Nesse caso os usuários devem estar vinculados à versão do artefato. O vínculo é realizado pelo usuário Administrador.
RNF03	Relacionamento entre palavras-chave e artefatos	Um artefato terá uma ou mais palavras-chave relacionadas.
RNF04	Relacionamento entre versão de artefato e projeto	Permitir o vínculo de artefatos e projetos que os utilizam.
RNF05	Busca de artefatos	A busca no repositório pode ser realizada por usuários sem *login* ou usuários cujo *login* ainda não foi validado e por usuários com *login*. A busca dos usuários com *login* ou *login* não validados é realizada apenas nos artefatos públicos. Os tipos de busca realizados são: a) Nos campos de cadastro do artefato b) No campo descrição do artefato (busca em texto livre) c) Nas palavras-chave O resultado da busca é apresentado em uma listagem.
RNF06	Relacionamentos entre artefatos	Um artefato pode ter mais de um tipo de relacionamento.
RNF07	Cadastro de projetos	Artefatos distintos podem estar vinculados a projetos, mas de uma única versão para cada tipo de artefato. Portanto, não é possível vincular em um projeto um mesmo artefato com versões diferentes. No vínculo da versão do artefato ao projeto, verificar o campo que define o identificador do artefato. No momento de vincular uma versão de artefato a um projeto verificar se o referido artefato já não está vinculado ao projeto.
RNF08	Cadastro de usuários	Um usuário faz o seu próprio cadastro que deve ser validado pelo administrador. O usuário somente pode incluir artefatos no repositório após ter o seu cadastro validado.

(conclusão)

Identificação	Nome	Descrição
RNF09	Validação de usuário	Um usuário está validado se o campo tipo de usuário do seu cadastro está com valor diferente de usuário sem login, que pode ser nulo, por exemplo.
RNF10	Tipo de usuário	Ao cadastrar-se um usuário é do tipo sem *login*. Um usuário administrador valida o cadastro, atribuindo permissões de usuário com *login* ou administrador.
RNF11	Tipo de usuário	Um usuário do tipo Usuário com login tem acesso à inclusão de artefatos, consulta de artefatos públicos e aos artefatos aos quais ele está relacionado (tem permissão de acesso) e alteração dos seus próprios artefatos. Um usuário do tipo Administrador possui acesso a todas as funcionalidades do sistema. Um usuário do tipo Usuário sem login ou consulta somente pode consultar artefatos públicos e não possui cadastro no sistema.
RNF12	Exclusão de dono de artefato	O autor (dono) de artefatos pode ser excluído mesmo com artefatos vinculados. No momento da exclusão de um usuário com artefatos solicitar se os artefatos serão excluídos, se somente os de acesso público, somente os de acesso privado ou ambos. Esses artefatos, se não excluídos, podem ser atribuídos acesso a outros usuários ou tornados públicos. O usuário Administrador possui acesso a todos os artefatos.
RNF13	Cadastro de usuário	No cadastro de usuário solicitar a redigitação da senha.

Quadro 4 – Requisitos não funcionais

A Figura 4 apresenta o diagrama de casos de uso do sistema. Esses casos de uso foram identificados a partir dos requisitos definidos para o sistema. Eles representam funcionalidades que são realizadas a partir dos dados obtidos pelos cadastros.

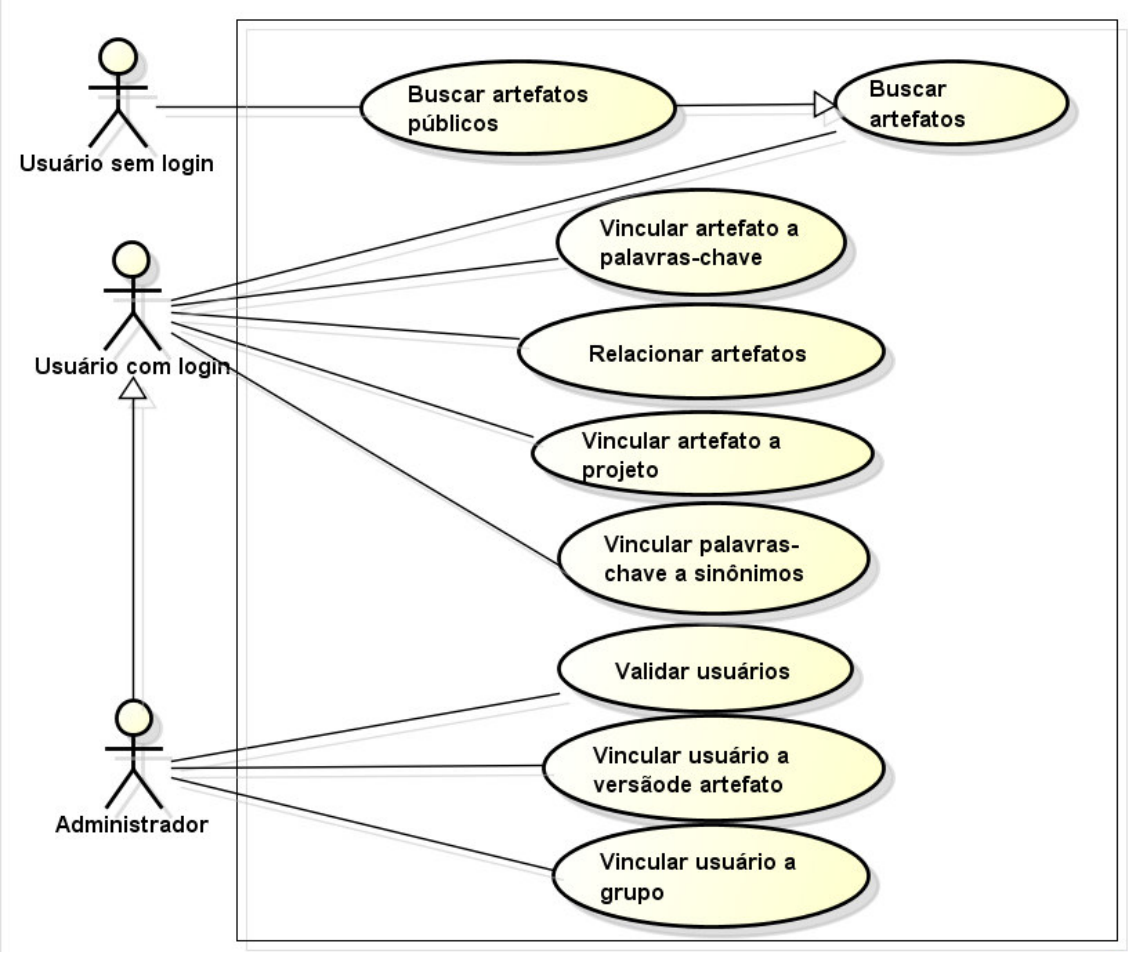

Figura 4 – Diagrama de casos de uso do sistema

De acordo com a representação da Figura 4, o sistema possui três tipos de usuários, que são caracterizados pelos privilégios de acesso que os mesmos possuem ao sistema. Esses três tipos de usuário são:

a) Administrador – com acesso a todas as funcionalidades do sistema e exclusivamente a validar cadastro de usuários, inclusive para atribuir acesso de administrador a outros usuários, atribuir acesso a artefatos e manter grupos de usuários.

b) Usuário com *login* – tem permissão para cadastrar, exceto grupos, e alterar os seus artefatos.

c) Usuário sem *login* – somente consulta artefatos públicos. É o usuário sem cadastro no sistema, com cadastro incompleto (sem indicação de login e senha) ou com cadastro ainda não validado.

No Quadro 5 está a documentação do caso de uso identificado como “Vincular palavras-chave a sinônimos”. Esse caso de uso exemplifica os casos de uso identificados

como “vincular” na Figura 4.

Caso de uso: Vincular palavras-chave a sinônimos **Descrição**: Cadastro de sinônimos entre palavras-chave que serão utilizadas para categorizar os artefatos. **Evento Iniciador**: O usuário solicita a inclusão de sinônimos para palavras-chave. **Atores**: Usuário com *login* ou Administrador. **Pré-condição**: As palavras associadas como sinônimos devem estar incluídas no cadastro de palavras-chave. **Sequência de Eventos**: 1. Ator acessa a tela para cadastro de sinônimos para palavra-chave incluindo as informações necessárias. As palavras incluídas como sinônimos devem estar cadastradas como palavras-chave. 2. O sistema insere os dados no banco de dados e informa o usuário que o referido conjunto de sinônimos para a respectiva palavra foi incluído. **Pós-Condição**: Sinônimos de uma palavra inseridos no banco de dados. **Extensões**: Palavra a ser usada como sinônimo não está cadastrada como palavra-chave.

Nome do fluxo alternativo (extensão)	Descrição
1 Cadastro de palavras-chave.	1.1 O ator acessa a tela do sistema para cadastrar palavras-chave e inclui as informações solicitadas. 1.2 Sistema inclui informações no banco de dados.

Quadro 5 – Caso de uso “vincular palavras-chave a sinônimo”

O Quadro 6 apresenta a descrição do caso de “relacionar artefatos” que está na Figura 4

Caso de uso: Relacionar artefatos **Descrição**: Relacionar artefatos. **Evento Iniciador**: O usuário acessa tela para buscar relacionar artefatos **Atores**: Usuário com *login*, Administrador. **Pré-condição**: Os artefatos que serão relacionados devem estar cadastrados. O tipo de relacionamento a ser usado deve estar cadastrado. **Sequência de Eventos**: 1. Ator acessa a tela para relacionar artefatos. 2. Ator escolhe o tipo de relacionamento a ser utilizado. 3. Ator escolhe os artefatos que serão relacionados para o tipo escolhido. 4. Ator solicita inclusão do relacionamento. 3. Sistema apresenta tela informando que o relacionamento foi efetuado. **Pós-Condição**: Relacionamento entre artefatos armazenado no banco de dados.

Quadro 6 – Caso de uso “relacionar artefatos”

O Quadro 7 apresenta a descrição do caso de "buscar artefatos" constante na Figura 4.

Caso de uso:
Buscar artefatos públicos.
Descrição:
Buscar artefatos no repositório.
Evento Iniciador:
O usuário acessa tela para buscar artefatos.
Atores:
Usuário com login, Usuário sem login, Administrador.
Pré-condição:
Não há
Sequência de Eventos:
1. Ator acessa a tela para buscar artefatos.
2. Ator escolhe tipo de busca e indica os critérios de busca solicitados.
3. Sistema apresenta resultados de acordo com os critérios de busca solicitados.
Pós-Condição:
Resposta da consulta do autor apresentada como uma listagem dos componentes que atendem aos critérios de busca especificados.

Quadro 7 – Caso de uso "buscar artefatos"

O Quadro 8 apresenta a descrição do caso de "vincular usuário a grupo" constante na Figura 4.

Caso de uso:
Vincular usuário a grupo
Descrição:
Vincular usuários a grupos cadastrados.
Evento Iniciador:
O usuário solicita o relacionamento de usuários a grupos.
Atores:
Administrador.
Pré-condição:
Os usuários e os grupos devem estar cadastrados para que possam ser relacionados.
Sequência de Eventos:
1. Ator Administrador acessa a tela para relacionar usuários a grupos, informando o grupo e os usuários que pretende vincular ao mesmo.
2. O sistema insere os dados no banco de dados e informa o usuário que os usuários foram vinculados ao respectivo grupo.
Pós-Condição: usuários vinculados ao grupo.
Extensões:
Grupo não cadastrado.

Nome do fluxo alternativo (extensão)	Descrição
1 Cadastro de grupo	1.1 O ator acessa a tela do sistema para cadastrar grupos e inclui as informações solicitadas. 1.2 Sistema inclui informações no banco de dados.

Quadro 8 – Caso de uso "vincular usuário a grupos"

A descrição do caso de uso "validar usuário" é apresentada no Quadro 9. Esse caso de

uso permite que o usuário Administrador atribua permissões de acesso com *login* ou de administrador a um usuário pré-cadastrado. Essa ação do administrador permite o acesso a inclusão de artefatos no repositório, vínculo do usuário a outros artefatos e operações de cadastros permitidas pelo sistema.

Caso de uso:
Validar usuário
Descrição:
Validar cadastro de usuário, atribuindo acesso ao sistema como Usuário com *login* ou Administrador.
Evento Iniciador:
O administrador acessa tela para validar usuários.
Atores:
Administrador.
Pré-condição:
Usuário cadastrado com *login* e senha informados.
Sequência de Eventos:
1. Ator Administrador acessa a tela para validar usuários.
2. O sistema lista os usuários que possuem cadastro não validado.
3. Ator Administrador valida cadastro.
4. Sistema envia *email* informado que o respectivo cadastro foi validado e o tipo de acesso atribuído.
Pós-Condição: usuário validado.

Quadro 9 – Caso de uso "validar usuário"

Os casos de uso relacionados a cadastro estão apresentados na Figura 5. Esses casos de uso permitem que as funcionalidades representadas nos casos de uso apresentados na Figura 4 possam ser realizados.

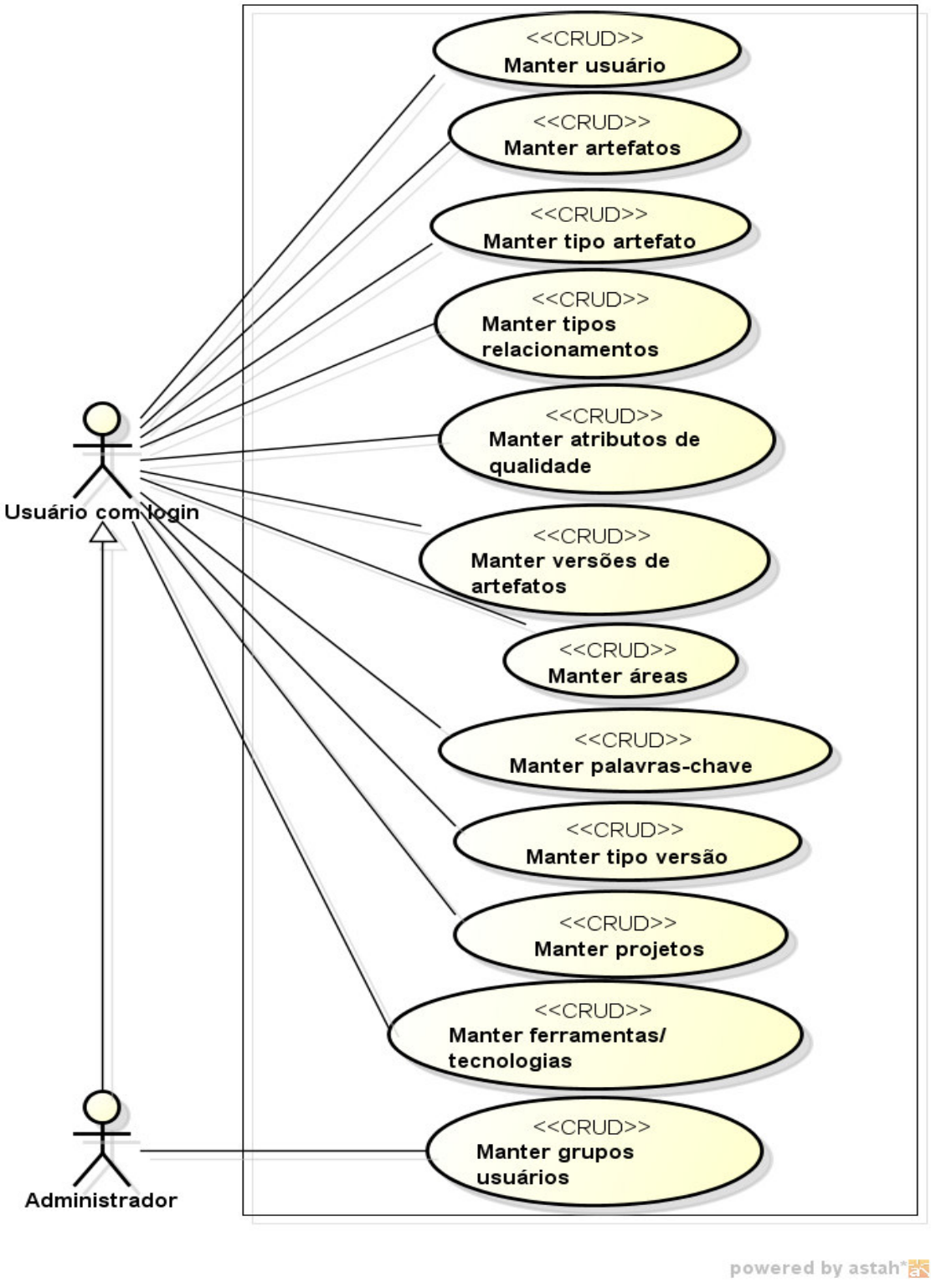

Figura 5 – Diagrama de casos de uso de cadastro

O Quadro 10 apresenta os relacionamentos entre os casos de uso constantes nas Figuras 4 e 5 com os requisitos funcionais listados no Quadro 3.

Caso de uso	Requisitos relacionados
Buscar artefatos públicos	RF15 - Buscar artefatos
Buscar artefatos	RF15 - Buscar artefatos
Vincular artefatos e palavras-chave	RF10 - Atribuir palavras-chave a artefatos
Relacionar artefatos	RF17 - Cadastro de relacionamentos entre artefatos
Vincular artefato a projeto	RF22 - Relacionamento de versão do artefato e projeto que o utiliza
Vincular palavras-chave a sinônimos	RF11 - Cadastro de sinônimos
Vincular usuário a versão de artefato	RF04 - Vincular usuários a versão de artefatos
Vincular usuário a grupo	RF03 - Associar usuários a grupos
Manter usuário	RF01 - Cadastro de usuários
Manter artefatos	RF05 - Cadastro dos artefatos
Manter tipo artefatos	RF06 - Cadastro de tipos de artefatos
Manter tipos de relacionamentos	RF16 - Cadastro de tipo de relacionamentos
Manter atributos de qualidade	RF18 - Cadastro de atributos de qualidade do artefato RF19 - Cadastro de tipos atributos de qualidade
Manter versões de artefatos	RF07 - Cadastro de versões de artefatos RF08 - Cadastro de componente de código
Manter tipos de relacionamentos de artefatos	RF12 - Cadastro de tipos de relacionamentos de artefatos
Manter áreas	RF21 - Cadastro de áreas de projetos
Manter palavras-chave	RF09 - Cadastro de palavras-chave
Manter tipo de versão	RF13 - Cadastro de tipos de versão de artefato
Manter projetos	RF20 - Cadastro de projetos
Manter ferramentas /tecnologias	RF13 - Cadastro de ferramentas ou tecnologias
Manter usuários e grupos de usuários Validar usuários	RF02 - Cadastro de grupos de usuários

Quadro 10 – Relacionamento entre casos de uso e requisitos funcionais

No Quadro 11 está documentado um caso de uso de cadastro (identificado com o estereótipo CRUD na Figura 5). Esse caso de uso é o de "Manter sinônimos" e é utilizado para exemplificar como são descritos os casos de uso desse tipo, ou seja, do tipo "manter". Todos os casos de uso seguem o mesmo padrão, mesmo considerando que pode ou não haver necessidade de dados provenientes de outros cadastros.

Caso de uso:
Manter sinônimos
Descrição:
Cadastro de sinônimos de palavras-chave que serão utilizadas para categorizar os artefatos.
Evento Iniciador:
O usuário solicita a inclusão de sinônimos para palavras-chave.

Atores:
Usuário com login ou Administrador.
Pré-condição:
As palavras associadas como sinônimos devem estar incluídas no cadastro de palavras-chave.
Sequência de Eventos:
1. Ator acessa a tela para cadastro de um conjunto de sinônimos para uma palavra incluindo as informações necessárias. As palavras incluídas como sinônimos devem estar cadastradas como palavras-chave.
2. O sistema insere os dados no banco de dados e informa o usuário que o referido conjunto de sinônimos para a respectiva palavra foi incluído.
Pós-Condição:
Sinônimos de uma palavra inseridos no banco de dados.
Extensões:
Palavra a ser usada como sinônimo não está cadastrada como palavra-chave.

Nome do fluxo alternativo (extensão)	Descrição
1 Cadastro de palavras-chave.	1.1 O ator acessa a tela do sistema para cadastrar palavras-chave e inclui as informações solicitadas. 1.2 Sistema inclui informações no banco de dados.

Quadro 11 – Caso de uso "manter sinônimos"

A seguir são apresentadas as listagens de campos para os cadastros visando atender os requisitos identificados. Esses cadastros representam formulários no sistema e a identificação dos campos de entrada se refere aos campos que esses formulários devem conter. Esses cadastros se referem os casos de uso identificados como o estereótipo CRUD apresentados na Figura 8 e os da Figura 7 para os quais é necessário ter um formulário. O Quadro 12 apresenta a listagem dos campos de entrada do cadastro de usuários.

Dado	**Descrição**	**Tipo**	**Obrigatório**
Nome do usuário	Identificação do usuário	Texto	Sim
Login	*Login* de acesso ao sistema	Texto	Sim
Senha	Senha para acesso ao sistema	Texto	Sim
Email	Endereço de *email*	Texto	Não
Tipo de usuário	Identificação do tipo de usuário. Tipos: 0 – usuário sem aprovação. Cadastro não validado. É o tipo padrão ao ser incluído um usuário novo, que deve ser aprovado pelo administrador para ter acesso ao sistema. 1 – usuário com aprovação. Cadastro aprovado. O usuário pode incluir artefatos no sistema. 2 – administrador. Acesso a todas as funcionalidades do sistema.	Numérico	Sim

Quadro 12 – Listagem dos campos do cadastro de usuários

A listagem dos campos de entrada do cadastro de grupos de usuários é apresentada no Quadro 13. Um grupo de usuários pode ser utilizado para atribuir permissões de acesso a usuários distintos a um mesmo artefato. Um grupo de usuários também pode representar uma

fábrica de software, por exemplo. Assim, esse usuário pode ser utilizado pela fábrica para acesso aos seus artefatos ou para agrupar todos os usuários cadastrados no sistema que pertencem à referida fábrica.

Dado	Descrição	Tipo	Obrigatório
Nome do grupo	Identificação do grupo	Texto	Sim
Descrição	Descrição do grupo	Texto	Não

Quadro 13 – Listagem dos campos do cadastro de grupos de usuários

No Quadro 14 está a listagem dos campos do formulário para atribuir permissões de acesso (usuário com *login* ou administrador) de usuários a artefatos.

Dado	Descrição	Tipo	Obrigatório
Usuário	Identificação do usuário	Numérico	Sim
Artefato	Identificação do artefato	Numérico	Sim
Versão do Artefato	Identificação da versão do artefato	Numérico	Sim

Quadro 14 – Listagem dos campos de permissões de acesso aos sistema

A listagem dos campos de entrada do cadastro para vincular usuários a grupos de usuários está no Quadro 15. Esse é o cadastro para composição de grupos. Um grupo de usuários pode ser composto por usuários ou mesmo por grupos de usuários.

Dado	Descrição	Tipo	Obrigatório
Grupo	Campo chave. Vem da tabela de grupo.	Numérico	Sim
Usuário	Campo chave. Vem da tabela de usuários ou da tabela de grupos.	Numérico	Sim

Quadro 15 – Listagem dos campos de associação de usuários a grupos

A listagem dos campos de entrada do cadastro de artefatos é apresentada no Quadro 16. Os campos “Tipo do artefato” são utilizados para categorizar um artefato. O usuário pode indicar até três categorizações para o artefato. Por exemplo: componente de código para validar CPF implementado em Java. Possui como tipo: componente, subtipo validação de CPF e subsubtipo linguagem Java.

Dado	Descrição	Tipo	Obrigatório
Nome do artefato	Identificação do artefato.	Texto	Sim
Descrição	Descrição do artefato.	Texto	Não
Observação	Informações complementares.	Texto	Não

Quadro 16 – Listagem dos campos de cadastro de artefatos

A listagem dos campos de entrada do cadastro de versões de artefatos (Quadro 17) apresenta os dados informados para cadastrar uma versão de artefato. Se o artefato sendo armazenado é um componente de software outros atributos adicionais são definidos. Esses

campos são apresentados em uma aba do cadastro de artefatos.

Dado	Descrição	Tipo	Obrigatório
Artefato	Artefato a que a versão pertence	Numérico	Sim
Nome da versão	Identificação da versão.	Texto	Sim
Autor	Autor do artefato.	Texto	Não
Descrição	Descrição da versão	Texto	Não
Data da versão	Data da criação da versão.	Data	Não
Tipo da versão	Tipo da versão, selecionado a partir de um cadastro de tipos. Vem do cadastro de tipos de versões.	Numérico	Não
Ferramenta	Ferramenta utilizada para produzir o artefato. Vem da tabelas de ferramentas e tecnologias.		
Motivo da versão	Motivo que gerou a versão Escolhida de opções predefinidas: Melhoria, Correção de erros/defeitos, agregação de funcionalidades. O Formulário pode apresentar botões de rádio ou outro componente para a escolha.	Numérico	Sim
Disponibilidade	Um dos dois tipos: público ou privado. Público todos os usuários cadastrados acessam. Privado é necessário atribuir acesso. Padrão do campo é privado. O Formulário pode apresentar botões de rádio ou outro componente para a escolha.	Numérico	Sim
Local de armazenamento	Endereço ou localização se o artefato não está no mesmo repositório	Texto	Não
Observação	Complemento às informações	Texto	Não

Quadro 17 – Listagem dos campos de cadastro de versão de artefatos

Os artefatos que são componentes de código possuem informações complementares. O cadastro de artefato possui uma aba complementar que é habilitada se o usuário indicar que está sendo cadastrado um artefato. Os campos complementares para o cadastro de componentes de código são apresentados no Quadro 18.

(continua)

Dado	Descrição	Tipo	Obrigatório
Objetivos	Objetivos, finalidades, do componente	Texto	Não
Funcionalidades	Serviços providos pelo componente	Texto	Não
Restrições	limitações em termos da funcionalidade do componente	Texto	Não
Interfaces	Forma de interação com o componente (parâmetros de entrada). Interfaces.	Texto	Não

(conclusão)

Dado	Descrição	Tipo	Obrigatório
Acesso ao código fonte	Sim, se o código do componente é editável. Não se o código do componente não é editável, se não há acesso ao mesmo. O formulário pode apresentar botões de rádio ou outro componente para a escolha.	Numérico	Sim
Requisitos de uso	O que é necessário para que o componente possa ser usado.	Texto	Não
Local de documentação	Endereço ou outro artefato armazenado no repositório que seja a documentação de uso do componente. Se for uma descrição curta pode estar contida nesse campo.	Texto	Não

Quadro 18 – Listagem dos campos de cadastro de componente de código

Listagem dos campos de entrada do cadastro de palavras-chave (Quadro 19). As palavras-chave são utilizadas para definir o artefato e essas palavras-chave são utilizadas na busca de artefatos no repositório.

Dado	Descrição	Tipo	Obrigatório
Nome	Nome da palavra-chave	Texto	Sim

Quadro 19 – Listagem dos campos de cadastro de palavras-chave

No Quadro 20 está a listagem dos campos de entrada do cadastro de atribuição de palavras-chave aos artefatos.

Dado	Descrição	Tipo	Obrigatório
Palavra-chave	Palavra-chave. Vem do cadastro de palavras-chave.	Numérico	Sim
Artefato	Identificação do artefato. Vem do cadastro de artefato.	Numérico	Sim
Tipo de palavra-chave	Palavra-chave primária ou secundária. Escolha entre um desses dois tipos. O Formulário pode apresentar botões de rádio ou outro componente para a escolha.	Numérico	Sim

Quadro 20 – Listagem dos campos de atribuição de palavras-chave a artefato

Os artefatos possuem tipo. O tipo é utilizado para classificação hierárquica do artefato, por exemplo: componente de código (tipo) desenvolvido na linguagem Java (subtipo) para validação de campo numérico (subsubtipo). A forma de definição da hierarquia depende do usuário. A listagem dos campos de entrada do cadastro de tipos de artefatos é apresentada no Quadro 21.

Dado	Descrição	Tipo	Obrigatório
Nome do tipo do artefato	Identificação do tipo de artefato.	Texto	Sim
Descrição	Descrição do tipo de artefato	Texto	Não

Quadro 21 – Listagem dos campos de cadastro de tipos de artefatos

No Quadro 22 está a listagem dos campos de entrada do cadastro de ferramentas e tecnologias. As ferramentas e tecnologias são utilizadas para preenchimento de dois campos no cadastro de artefatos.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação da ferramenta ou tecnologia.	Texto	Sim
Versão	Versão da ferramenta ou tecnologia.	Texto	Não
Descrição	Descrição da ferramenta ou tecnologia.	Texto	Não

Quadro 22 – Listagem dos campos de cadastro de ferramentas e tecnologias

A listagem dos campos de entrada do cadastro de sinônimos é apresentada no Quadro 23 e visa reduzir o uso de palavras com significados semelhantes na definição de artefatos, e também ser utilizada na busca.

Dado	Descrição	Tipo	Obrigatório
Palavra-chave	Palavra-chave. Vem do cadastro de palavras-chave.	Texto	Sim
Sinônimo	Palavra-chave que é sinônimo da palavra-chave no outro campo. Vem do cadastro de palavras-chave.	Texto	Sim

Quadro 23 – Listagem dos campos para associação de sinônimos entre palavras-chave

A listagem do Quadro 24 apresenta os campos de entrada do cadastro de tipos de versão de artefato. Os tipos de versão são utilizados no cadastro de artefatos.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação da versão do artefato. Exemplo: *shareware*, *freeware*, paga	Texto	Sim
Descrição	Descrição da versão do artefato. Complemento ao nome da versão.	Texto	Não

Quadro 24 – Listagem dos campos de tipos de versões de artefatos

No Quadro 25 está a listagem dos campos de entrada do cadastro de tipos de relacionamentos entre os artefatos. Os tipos de relacionamentos são utilizados para definir associações entre os artefatos. Esses relacionamentos são utilizados para definir artefatos que obrigatoriamente devem ser utilizados juntos e artefatos que poderiam ser utilizados juntos.

Dado	Descrição	Tipo/ Tamanho	Obrigatório
Nome	Identificação do tipo de relacionamento	Texto	Sim
Descrição	Descrição ou observações sobre o tipo de palavra-chave cadastrado	Texto	Não

Quadro 25 – Listagem dos campos de cadastro de tipos de relacionamentos entre artefatos

A listagem dos campos de entrada do cadastro de relacionamentos entre artefatos está no Quadro 26.

Dado	Descrição	Tipo	Obrigatório
Artefato	Artefato. Vem do cadastro de artefatos.	Numérico	Sim
Artefato relacionado	Artefato. Vem do cadastro de artefatos.	Numérico	Sim
Tipo de relacionamento	Um dos três tipos: composto de, depende de ou pode ser usado com. Vem do cadastro de relacionamentos.	Numérico	Sim

Quadro 26 – Listagem dos campos de relacionamento entre artefatos

A listagem dos campos de entrada do cadastro de projetos é apresentada no Quadro 27. Esse cadastro visa determinar quais projetos utilizam quais artefatos. Um dos objetivos desse cadastro é o "aviso" aos projetos que utilizam artefatos que foram modificados para correção de erros.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação do projeto	Texto	Sim
Descrição	Descrição do projeto	Texto	Não
Data de início	Data de início do projeto	Data	Não
Data de término	Data de fim do projeto	Data	Não
Área	Área do projeto. Vem do cadastro de áreas.	Numérico	Não

Quadro 27 – Listagem dos campos do cadastro de projetos

Os projetos são categorizados por áreas. As áreas podem auxiliar na busca de artefatos armazenados, visando atendimento aos critérios de pesquisa. A listagem dos campos de entrada do cadastro de áreas é apresentada no Quadro 28.

Dado	Descrição	Tipo	Obrigatório
Nome	Identificação da área do projeto	Texto	Sim
Descrição	Descrição da área do projeto	Texto	Não

Quadro 28 – Listagem dos campos do cadastro de áreas de projetos

Um projeto de software poderá ter vários artefatos vinculados ou mais especificamente versão desses artefatos. O Quadro 29 apresenta a listagem dos campos de entrada do cadastro

de relacionamento de versão de artefatos e projetos que os utilizam. Indica quais artefatos (versões) são usadas pelos projetos cadastrados. Inicialmente é selecionado o artefato e em seguida são apresentadas as versões do respectivo artefato. A listagem dos artefatos é apresentada após a seleção do artefato.

Dado	Descrição	Tipo	Obrigatório
Artefato	Identificação do artefato. Vem do cadastro de artefatos.	Numérico	Sim
Versão do artefato	Identificação da versão do artefato. Vem do cadastro de versão artefato.	Numérico	Sim
Projeto	Identificação do projeto. Vem do cadastro de projeto.	Numérico	Sim

Quadro 29 – Listagem dos campos para associação entre versões de artefatos e projetos

A listagem dos campos de permissões para acesso aos artefatos. Os campos para cadastro de relacionamento entre versões de artefatos e usuários são apresentados no Quadro 30. É o cadastro para atribuir acesso a versões de artefatos para usuários. São os artefatos que o usuário tem acesso.

Dado	Descrição	Tipo	Obrigatório
Artefato	Identificação do artefato. Vem do cadastro de artefato.	Numérico	Sim
Versão do artefato	Identificação da versão do artefato. Vem do cadastro versões do artefato.	Numérico	Sim
Usuário	Identificação do usuário. Vem da tabela de usuários.	Numérico	Sim

Quadro 30 – Listagem dos campos para atribuir acesso de usuários a artefatos

Atributos de qualidade são utilizados para definir requisitos não funcionais associados aos artefatos. A listagem dos campos de cadastro de atributos de qualidade é apresentada no Quadro 31. Uma versão de artefato pode possuir vários atributos de qualidade. A listagem das versões do artefato é apresentada após a seleção do respectivo artefato.

Dado	Descrição	Tipo	Obrigatório
Artefato	Artefato. Vem do cadastro de artefatos.	Numérico	Sim
Versão do artefato	Versão do artefato. Vem do cadastro de versões do artefato.	Numérico	Sim
Atributo de qualidade	Característica não funcional, aspecto ou atributo de qualidade. Vem do cadastro de tipos de atributos de qualidade.	Numérico	Sim
Descrição	Descrição da característica qualidade	Texto	Não
Observações	Resultado de testes, descrição de problema, restrições	Texto	Não
Justificativa	Comprovação que o artefato possui a referida característica.	Texto	Não

Quadro 31 – Listagem dos campos de cadastro de atributos de qualidade de artefatos

Os atributos de qualidade são categorizados por tipos. No Quadro 32 está a listagem de tipos de atributos de qualidade.

Dado	Descrição	Tipo	Obrigatório
Nome	Nome do atributo, aspecto de qualidade. Por exemplo: testes, certificação.	Texto	Sim
Descrição	Descrição do atributo de qualidade.	Texto	Não

Quadro 32 – Listagem dos campos de cadastro de atributos de qualidade

Na fase de levantamento e registro dos requisitos também foram definidos esboços de interfaces para a busca de artefatos no repositório. Esses modelos têm o objetivo de apresentar a ideia para busca, sem qualquer interesse de expressar ergonomia ou padrão de interface. É uma forma de registro para representar os dados a serem considerados nos formulários de busca.

A busca dos artefatos no repositório poderá ser realizada de três maneiras básicas:

a) busca baseada em filtros;

b) busca por palavras-chave;

c) busca baseada no campo descrição. Essas formas de busca podem ser combinadas.

Os campos apresentados nos formulários de pesquisa não são obrigatórios, a filtragem é feita pelos campos preenchidos.

A Figura 6 apresenta a ideia da tela para busca baseada em filtros. Neste tipo de busca as caixas de combinação são preenchidas com os dados provenientes das respectivas tabelas no banco de dados e o usuário os seleciona. O usuário escolhe entre esses campos. Os campos de tipo, subtipo e subsubtipo podem formar uma pesquisa hierárquica.

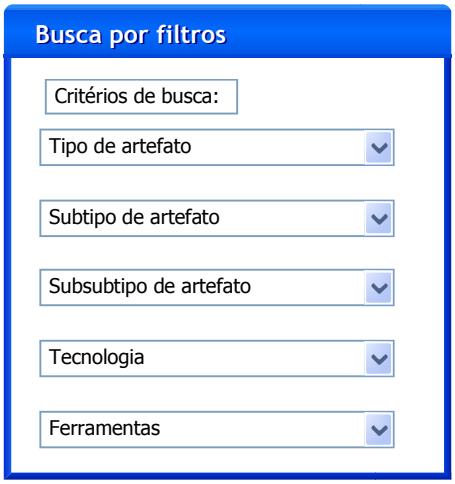

Figura 6 – Busca baseada em filtros

A Figura 7 apresenta a tela para busca baseada no campo de descrição do artefato. É

uma forma de busca em texto livre.

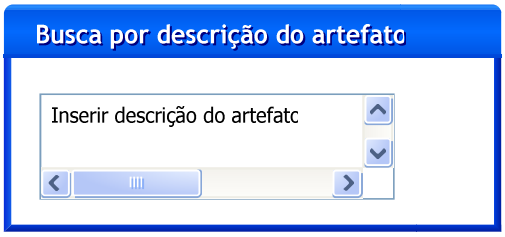

Figura 7 – Busca baseada em filtros e descrição

A Figura 8 apresenta a tela para busca baseada palavras-chave. As caixas de combinação são preenchidas com dados provenientes das respectivas tabelas. O usuário seleciona dentre essas palavras-chave cadastradas as que ele quer que o artefato contenha e as que não contenha.

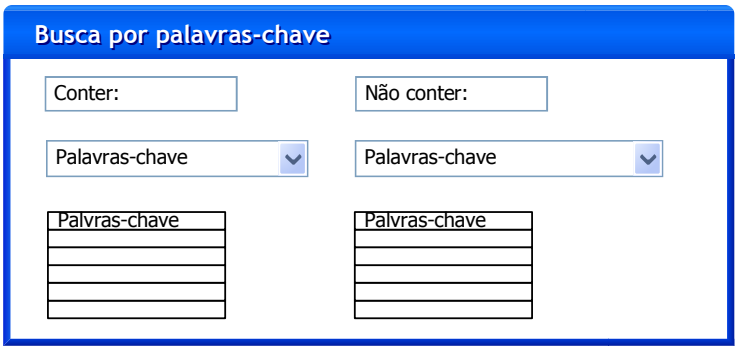

Figura 8 – Busca baseada em filtros e palavras-chave

O resultado da busca é apresentado como uma listagem contendo o nome do artefato, a versão e informações complementares (Figura 9). Nessa figura o nome do artefato é um *link* para *download* ou visualização do artefato e em "Informações" são apresentadas as informações obtidas no banco de dados sobre o artefato. O ícone representado por uma lupa apresenta todas as propriedades do artefato. É uma espécie de ficha com sua descrição, versões relacionadas e atributos de qualidade. Visualizar traz o cadastro com todos os dados.

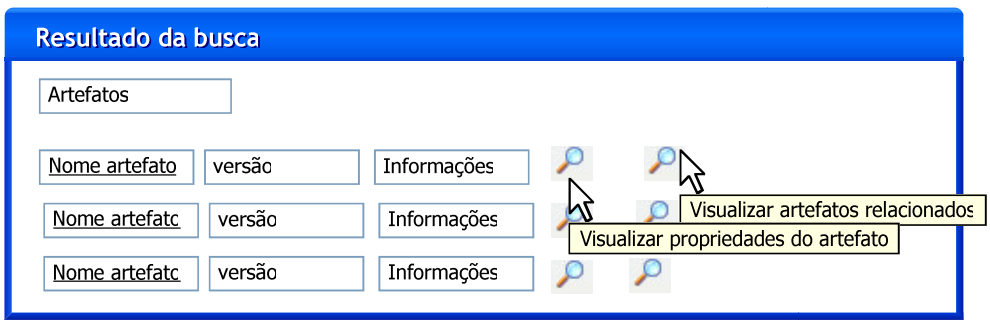

Figura 9 – Resultado da busca baseada em filtros e descrição

A Figura 10 apresenta o diagrama de classes do sistema. São apresentados apenas alguns atributos das classes, quando existentes. Esse diagrama apresenta apenas uma visão geral das principais classes definidas, na implementação outras classes foram definidas.

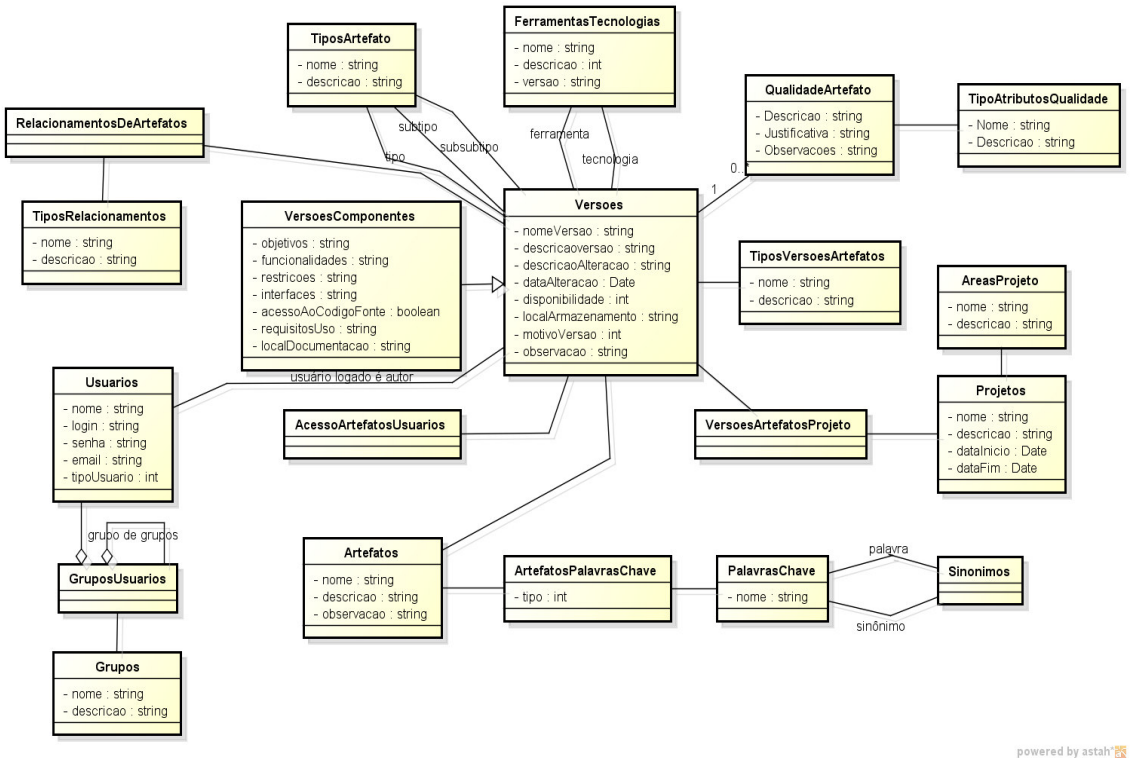

Figura 10 – Diagrama de classes

O diagrama de entidades e relacionamentos do banco de dados é apresentado na Figura 11.

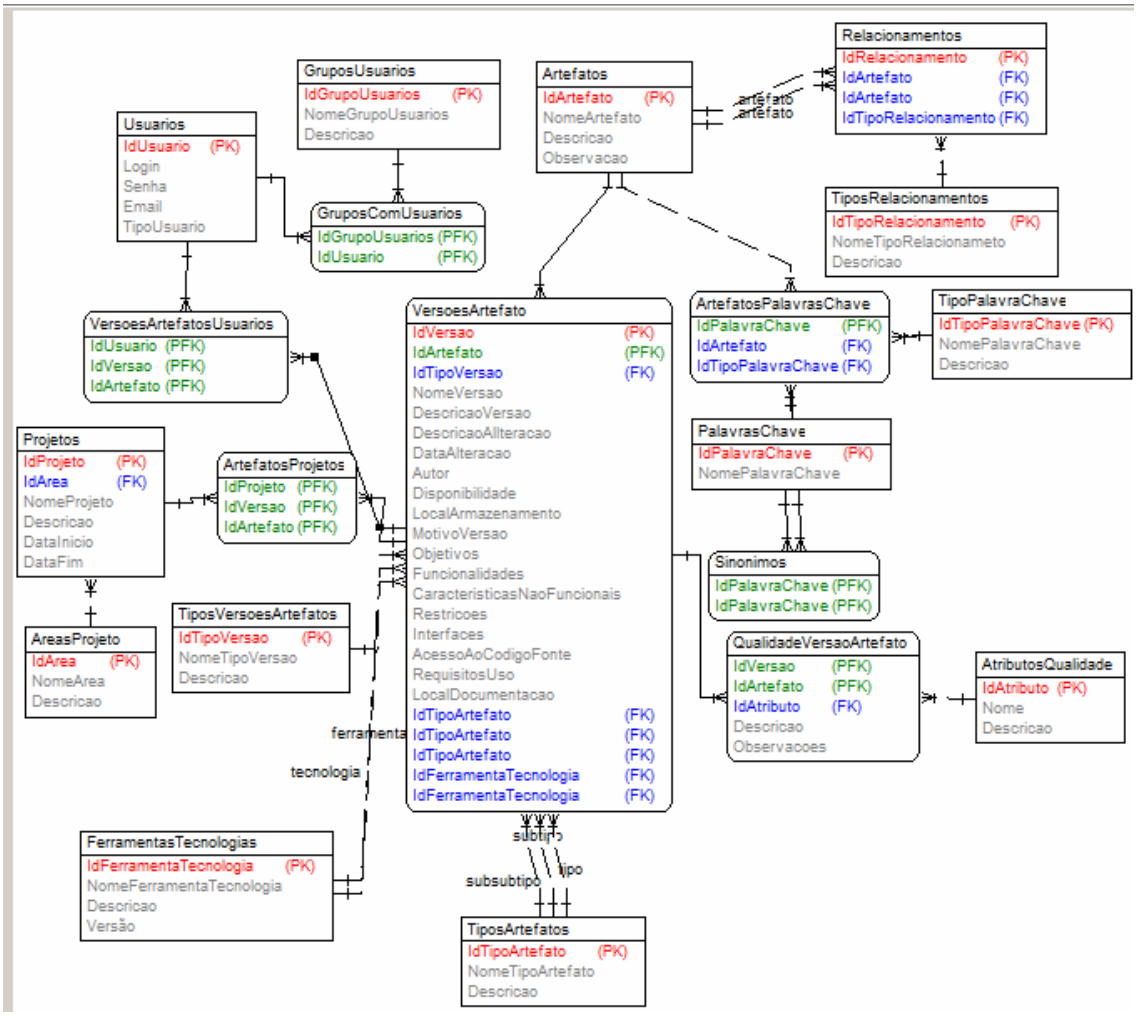

Figura 11 – Diagrama de entidades e relacionamentos do banco de dados

Se o artefato incluído não é um componente de software, os campos identificados no requisito funcional denominado “RF08 Cadastro de componente de código” permanecerão sem preenchimento. Isso porque eles foram definidos especificamente para o cadastro de artefatos que são do tipo componentes de código.

Na Figura 11 a tabela Artefatos possui três campos denominados “IdTipoArtefato” porque podem ser utilizados três nomes de tipos de artefatos distintos, provenientes da tabela “TiposArtefatos” para classificar ou categorizar o artefato. Esses atributos são definidos como tipo, subtipo e subsubtipo como forma de identificação, mas todos são provenientes da mesma tabela.

Na tabela Relacionamentos, da Figura 11, há dois campos IdArtefato, isso ocorre porque os artefatos são relacionados aos pares. Por exemplo: Artefato1 pode ser utilizado com (que é um tipo de relacionamento) Artefato2 e ambos têm origem na mesma tabela de

Artefatos.

A seguir a descrição das tabelas que compõem o banco de dados, conforme exposto na Figura 11.

A tabela usuários (Quadro 33) contém os campos de cadastro de usuários. Ao solicitar o cadastro o usuário preenche os seus dados, exceto tipo de usuário. Esse campo é preenchido pelo Administrador no momento da validação do cadastro.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**	**Observações**
IDUSUARIO	Numérico	Não	Sim	Não		
NOME	Texto	Não	Não	Não		
LOGIN	Texto	Sim	Não	Não		
SENHA	Texto	Sim	Não	Não		Obrigatório se *login* estiver preenchido
EMAIL	Texto	Sim	Não	Não		
TIPOUSUARIO	Numérico	Não	Não	Não	Usuário sem aprovação	Alterado pelo administrador ao validar o cadastro.

Quadro 33 – Campos da tabela Usuários

A tabela grupos de usuários permite agrupar usuários. Isso facilita a atribuição de acesso a artefatos, mais especificamente às suas versões. O Quadro 34 apresenta os campos da tabela para armazenamento dos grupos de usuários.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**	**Observações**
IDGRUPO_USUARIO	Numérico	Não	Sim	Não		
NOME	Texto	Não	Não	Não		
DESCRICAO	Texto	Não	Não	Não		

Quadro 34 – Campos da tabela Grupos

Na tabela grupos e usuários (Quadro 35) estão os usuários relacionados aos grupos. Essa tabela armazena os grupos com os seus usuários. Um usuário pode estar associado a vários grupos ou a nenhum.

Campo	**Tipo**	**Nulo**	**Chave primária**	**Chave estrangeira**	**Valor padrão**	**Observações**
IDGRUPO_USUARIO	Numérico	Não	Sim	Sim		Da tabela Grupos
IDUSUARIO	Numérico	Não	Sim	Sim		Da tabela Usuários Grupos.

Quadro 35 – Campos da tabela GruposUsuarios

Os campos da tabela de artefatos são apresentados no Quadro 36. Esses campos se referem aos dados básicos ou principais de um artefato e são complementados pela tabela que armazena os dados das versões do artefato.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDARTEFATO	Numérico	Não	Sim	Não		
NOME_ARTEFATO	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		
OBSERVACAO	Texto	Sim	Não	Não		

Quadro 36 – Campos da tabela Artefatos

Na tabela versões de artefatos estão os campos que complementam os dados do artefato, seja um componente de código, um documento ou outro. O Quadro 37 apresenta os campos dessa tabela.

(continua)

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
IDVERSAO	Numérico	Não	Sim	Sim	
IDARTEATO	Numérico	Não	Sim	Sim	Da tabela Artefato
IDTIPOVERSAO	Numérico	Sim	Não	Sim	Da tabela TipoVersao
NOMEVERSAO	Texto	Não	Não	Não	
DESCRICAOVERSAO	Texto	Sim	Não	Não	
DESCRICAOALTERACAO	Texto	Sim	Não	Não	
DATAALTERACAO	Data	Sim	Não	Não	
IDUSUARIO	Texto	Sim	Não	Não	Autor. É o usuário logado.
DISPONIBILIDADE	Numérico	Sim	Não	Não	Público ou privado
LOCALARMAZENAMENTO	Texto	Sim	Não	Não	
MOTIVOVERSAO	Texto	Sim	Não	Não	
IDTECNOLOGIA	Numérico	Sim	Não	Sim	Da tabela FerramentasTecnologias
IDFERRAMENTA	Numérico	Sim	Não	Sim	Da tabela FerramentasTecnologias
IDTIPOARTEFATO	Numérico	Sim	Não	Sim	Da tabela TipoArtefato
IDSUBTIPOARTEFATO	Numérico	Sim	Não	Sim	Da tabela TipoArtefato
IDSUBSUBTIPOARTEFATO	Numérico	Sim	Não	Sim	Da tabela TipoArtefato
OBSERVACAO	Texto	Sim	Não	Não	
OBJETIVOS	Texto	Sim	Não	Não	Para componente
FUNCIONALIDADES	Texto	Sim	Não	Não	Para componente

(conclusão)

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Observações
RESTRIÇÕES	Texto	Sim	Não	Não	Para componente
INTERFACES	Texto	Sim	Não	Não	Para componente
ACESSOAOCODIGOFONTE	Booleano	Sim	Não	Não	Para componente
REQUISITOSUSO	Texto	Sim	Não	Não	Para componente
LOCALDOCUMENTACAO	Texto	Sim	Não	Não	Para componente
DATA_INCLUSAO	Data	Não			Data obtida do sistema

Quadro 37 – Campos da tabela VersoesArtefatos

A tabela tipos de artefatos (campos apresentados no Quadro 38) armazena os tipos de artefatos utilizados no cadastro de versões de artefatos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDTIPO_ARTEFATO	Numérico	Não	Sim	Não		
NOME_TIPO	Texto	Não	Não	Não		
DESCRICAO	Texto	Não	Não	Não		

Quadro 38 – Campos da tabela TiposArtefatos

A tabela versões de artefatos e usuários armazena as permissões de acesso dos usuários às versões dos artefatos. Os campos dessa tabela estão no Quadro 39.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDARTEFATO	Numérico	Não	Sim	Sim		Da tabela de Artefatos
IDVERSAO_ARTEFATO	Numérico	Não	Sim	Sim		Da tabela de VersaoArtefato
IDUSUARIO	Numérico	Não	Sim	Sim		Da tabela de Usuários

Quadro 39 – Campos da tabela VersoesArtefatosUsuarios

O Quadro 40 apresenta os campos da tabela que armazena a atribuição de palavras-chave a artefatos. Um artefato pode possuir diversas palavras-chave vinculadas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDARTEFATO	Numérico	Não	Sim	Sim		Da tabela de Artefatos
IDPALAVRA_CHAVE	Numérico	Não	Sim	Sim		Da tabela PalavrasChave
TIPOPALAVRACHAVE	Numérico	Não	Não	Não	Primária	Primária e secundária

Quadro 40 – Campos da tabela ArtefatosPalavrasChave

A tabela Projetos contém os dados de cadastro de projetos que utilizam artefatos do repositório. Os campos dessa tabela estão no Quadro 41.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
ID_PROJETO	Numérico	Não	Sim	Não		
NOME_PROJETO	Texto	Não	Não	Não		
DESCRICAO	Texto	Não	Não	Não		
DATA_INICIO	Data	Sim	Não	Não		
DATA_FIM	Data	Sim	Não	Não		
ID_AREA	Numérico	Sim	Não	Sim		Da tabela Areas

Quadro 41 – Campos da tabela Projetos

Na tabela relacionamentos (campos no Quadro 42) estão os dados de relacionamentos entre artefatos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDRELACIONAMENTO	Numérico	Não	Sim	Não		
IDARTEFATO	Numérico	Não	Não	Sim		Da tabela Artefato
IDARTEFATO_RELAC	Numérico	Não	Não	Sim		Da tabela Artefato
IDTIPO_RELAC	Numérico	Não	Não	Não		Da tabela TipoRelacionamento

Quadro 42 – Campos da tabela Relacionamentos

Os dados da tabela que armazena os tipos de relacionamentos entre versão de artefatos são apresentados no Quadro 43.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDTIPO_RELAC	Numérico	Não	Sim	Não		
NOME_RELAC	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		

Quadro 43 – Campos da tabela TiposRelacionamentos

Os campos da tabela de tipos de versões de artefatos estão no Quadro 44.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDTIPO_VERSAO	Numérico	Não	Sim	Não		
NOME_TIPO	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		

Quadro 44 – Campos da tabela TiposVersoesArtefatos

Na tabela palavras-chave (Quadro 45) são armazenados as palavras-chave utilizadas no cadastro de artefatos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDPALAVRA_CHAVE	Numérico	Não	Sim	Não		
NOME_PALAVRA	Texto	Não	Não	Não		

Quadro 45 – Campos da tabela PalavrasChave

O Quadro 46 contém os dados da tabela sinônimos. Sinônimos identificados entre as palavras-chave cadastradas.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDPALAVRA_CHAVE	Numérico	Não	Sim	Sim		Da tabela PalavrasChave
IDPALAVRA_CHAVE_SINO	Numérico	Não	Sim	Sim		Da tabela PalavrasChave

Quadro 46 – Campos da tabela Sinônimos

Áreas permitem agrupar projetos. Os dados da tabela de áreas projetos são apresentados no Quadro 47.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDAREA	Numérico	Não	Sim	Não		
NOME_AREA	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		

Quadro 47 – Campos da tabela AreaProjetos

No Quadro 48 estão os campos para armazenar os dados de vínculo entre projeto e versões de artefatos. São as versões de artefatos utilizadas em um projeto.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDPROJETO	Numérico	Não	Sim	Sim		Da tabela Projetos
IDARTEFATO	Numérico	Não	Sim	Sim		Da tabela Artefatos
IDVERSAO	Numérico	Não	Sim	Sim		Da tabela VersaoArtefatos

Quadro 48 – Campos da tabela VersoesArtefatosProjetos

A tabela ferramentas e tecnologias contém as informações para as ferramentas e as tecnologias utilizadas na elaboração do artefato. Os campos dessa tabela estão no Quadro 49.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDFERRAMENTA_TEC	Numérico	Não	Sim	Não		
NOME_FERRAMENTA	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		
VERSÃO	Texto	Sim	Não	Não		

Quadro 49 – Campos da tabela Ferramentas/Tecnologias

No Quadro 50 estão os campos para armazenar atributos de qualidade de versões de artefatos.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDARTEFATO	Numérico	Não	Sim	Sim		
IDVERSAO	Numérico	Não	Sim	Sim		Da tabela VersaoArtefato
IDTIPO_ATRIBUTO	Numérico	Não	Sim	Sim		Da tabela TipoAtributo
DESCRICAO	Texto	Sim	Não	Não		Situação em que o problema ocorre
JUSTIFICATIVA	Texto	Sim	Não	Não		Comprovação do atributo
OBSERVACAO	Texto	Sim	Não	Não		

Quadro 50 – Campos da tabela QualidadeVersaoArtefato

A tabela tipos de atributos de qualidade (descrita no Quadro 51) possui os dados para armazenar os tipos de atributos de qualidade que um artefato pode ter.

Campo	Tipo	Nulo	Chave primária	Chave estrangeira	Valor padrão	Observações
IDTIPO_ATRIBUTO	Numérico	Não	Sim	Não		
NOME	Texto	Não	Não	Não		
DESCRICAO	Texto	Sim	Não	Não		

Quadro 51 – Campos da tabela TiposAtributosQualidade

4.3 DESCRIÇÃO DO SISTEMA

Na Figura 12 está a tela principal que é apresentada para o usuário quando o sistema é iniciado. A partir dessa tela o usuário poderá escolher se deseja somente fazer uma pesquisa dos artefatos cadastrados, se deseja realizar seu cadastro no repositório ou, se já possuir

cadastro, efetuar o *login* e ter acesso às demais funcionalidades do sistema de acordo com o seu tipo de usuário.

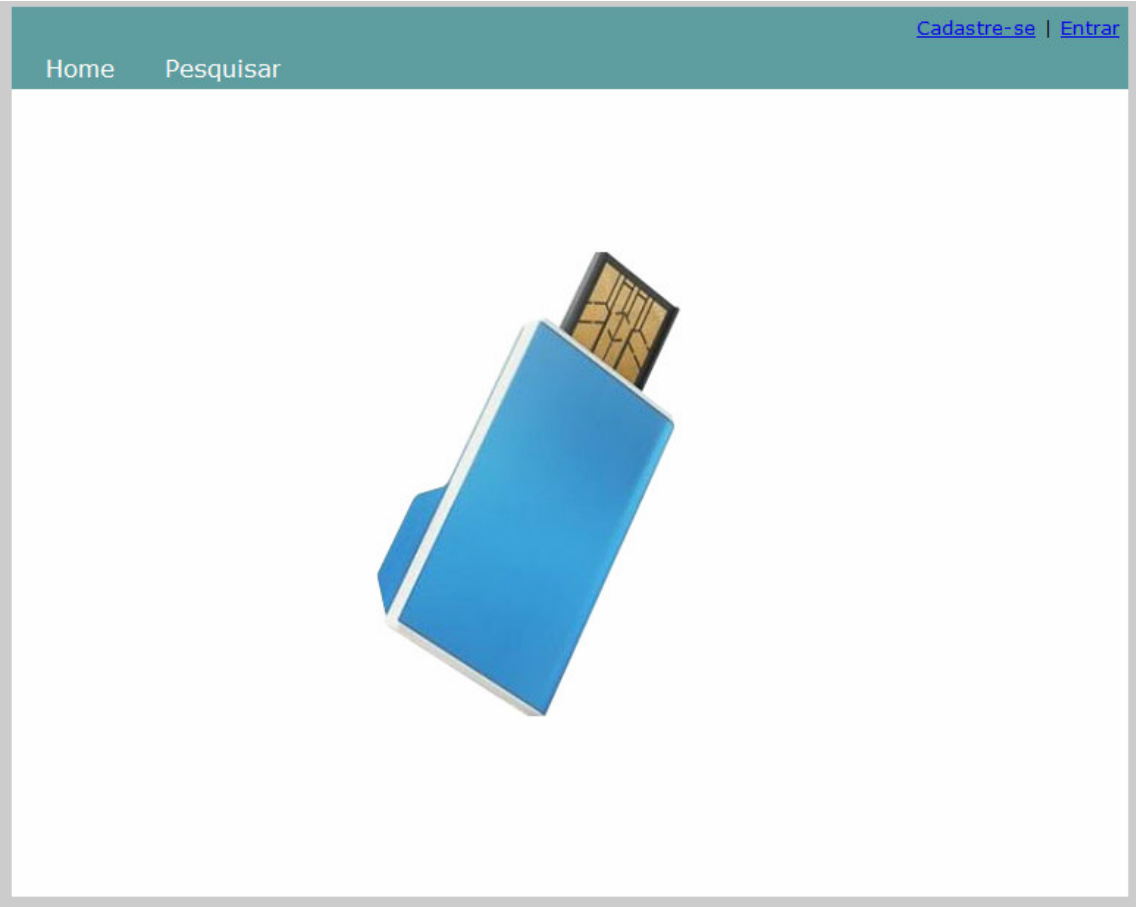

Figura 12 – Tela inicial do sistema

Um usuário sem *login* pode fazer buscas no repositório e os resultados das buscas são realizados somente nos artefatos com atributo de acesso público. Na parte superior da Figura 15 estão os menus que são apresentados ao usuário sem *login*: "Home" apresenta a tela principal do sistema e "Pesquisar" apresenta a tela de busca de artefatos no repositório. Esses artefatos são os disponibilizados para acesso a todos os usuários do sistema, com ou sem *login* definido. Para um usuário realizar operações como a inclusão de artefatos é necessário que o mesmo esteja logado no sistema.

A Figura 13 mostra a tela de cadastro de novo usuário do sistema. Nessa tela o usuário preencherá os campos *login*, senha e confirmação de senha que são obrigatórios, os demais campos são opcionais. Os botões Cancelar e o ícone "X" no canto superior direito da tela retornam para a tela principal.

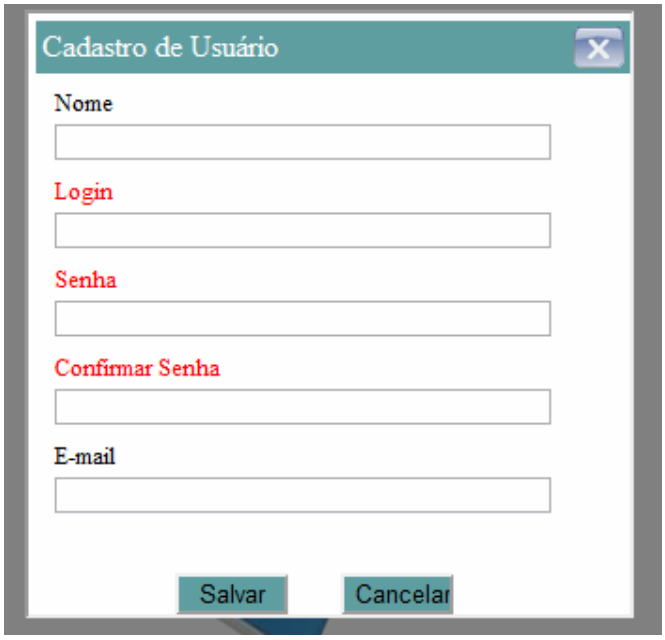

Figura 13 – Tela de cadastro de usuário

O cadastro do usuário precisa ser validado pelo administrador para que o acesso ao sistema possa ser feito por meio de *login*. Se o cadastro do usuário já foi aprovado pelo administrador, após o *login* ele será encaminhado para a tela mostrada pela Figura 14. Caso contrário, uma mensagem informativa será mostrada para usuário alertando-o que o seu cadastro ainda não foi aprovado pelo administrador.

Figura 14 – Tela de *login*

Os menus apresentados na Figura 15 se referem a um usuário definido como “usuário com login”, como explicitado no diagrama de casos de uso. Esse usuário pode fazer inclusão de artefatos no repositório e a ele podem ser atribuídas permissões de acesso a artefatos. Isso é feito pelo vínculo de usuários a artefatos. Essa operação permite que o usuário tenha acesso ao referido artefato. Com isso, um grupo de fábricas de software ou de usuários (pessoa física) pode compartilhar artefatos. Assim, o repositório pode ser utilizado para que cada fábrica de software armazene artefatos de uso exclusivo e artefatos de uso compartilhado. Com essa funcionalidade o repositório pode ser utilizado por grupos de fábricas de software que participem de iniciativas de desenvolvimento regional, por exemplo. Os artefatos de uso comum são compartilhados e cada fábrica poderá manter artefatos de uso exclusivo.

Figura 15 – Tela após *login* para usuário aprovado

Se um usuário fez cadastro e logar-se ao sistema antes que o referido cadastro tenha sido validado pelo administrador, uma mensagem é apresentada. A mensagem informativa é mostrada pela Figura 16.

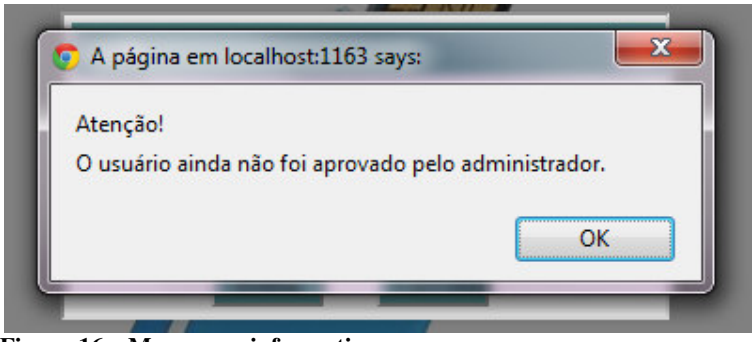

Figura 16 – Mensagem informativa

Se o usuário que efetuou o *login* é do tipo Administrador, então todas as funcionalidades do sistema são disponibilizadas. Os menus da tela inicial são alterados em relação a um usuário com e sem *login*, como apresentado na Figura 17.

Figura 17 – Tela após *login* para usuário administrador

As Figuras 18, 19 e 20 apresentam os dados de cada aba mostrada no menu para um usuário do tipo Administrador. A Figura 18 mostra as opções da aba pesquisa.

Figura 18 – Tela de opções da aba pesquisa

Na Figura 19 são apresentadas as opções da aba artefato.

Figura 19 - Tela de opções da aba artefato

Na Figura 20 são apresentadas as opções da aba projeto

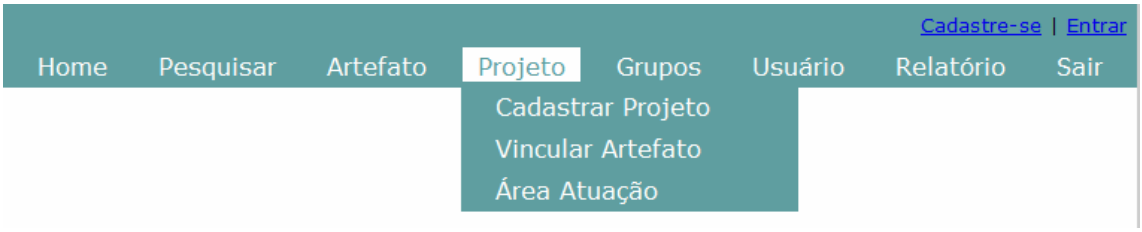

Figura 20 – Tela de opções da aba projeto

As opções da aba grupos são apresentadas na Figura 21.

Figura 21 – Tela de opções da aba grupos

Na Figura 22 são apresentadas as opções da aba usuário

Figura 22 – Tela de opções da aba usuário

A Figura 23 mostra as opções da aba de relatórios

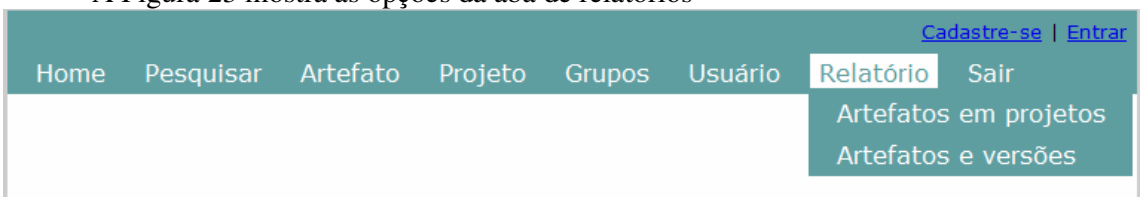

Figura 23 – Tela de opções da aba relatórios

Quando o usuário selecionar uma das opções do menu, será apresentada uma tela como a da Figura 24. Nela existe uma listagem de todos os dados cadastrados no banco de dados relacionados àquele cadastro. No lado direito, existe um menu que permite ao usuário escolher que operação ele deseja realizar. Se o usuário selecionar a opção “novo”, uma tela de novo cadastro será mostrada. Se o usuário selecionar a opção “alterar”, será aberta a tela de cadastro com os dados selecionados para a alteração. Por fim, o usuário poderá excluir um artefato cadastrado, para isso ele deverá selecionar na listagem qual artefato deseja excluir. Após uma verificação no banco de dados, o usuário será informado se o artefato pode ser excluído ou se ele possui vínculos com outras tabelas. Caso o artefato não possua relacionamentos, será mostrada uma tela de confirmação de exclusão, representada na Figura 25.

Figura 24 – Listagem de artefatos cadastrados no banco de dados

A tela de confirmação de exclusão de um arquivo do banco de dados é apresentada na Figura 25.

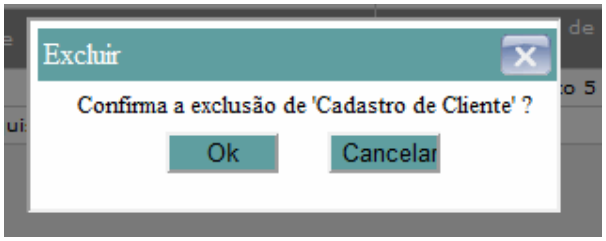

Figura 25 – Tela de confirmação de exclusão

O cadastro representado na Figura 26 é o de artefato. Nele o usuário informa o nome do artefato, uma descrição e uma observação. Os campos com o nome em vermelho são de preenchimento obrigatório. Diversos outros cadastros são apresentados e preenchidos da mesma forma que esse, por exemplo: o cadastro de tipo de artefato, de tipo de versão, de ferramenta e tecnologia e de projetos.

Figura 26 – Cadastro de artefato

A Figura 27 mostra a primeira aba do cadastro, nela o usuário informará dados da versão e selecionará o artefato relacionado a essa versão, se a versão será visível somente para ele ou para todos, ou seja, publico, dentre outros dados. Somente os campos em vermelho são de preenchimento obrigatório. As duas primeiras abas desse cadastro contêm dados relacionados a qualquer versão que seja cadastrada. Porém, a terceira aba somente será habilitada para preenchimento, caso a versão do artefato a ser inserida seja um componente de código, ou seja, se a versão não diz respeito a um componente do código o preenchimento dos dados das duas primeiras abas é suficiente. Lembrando que é obrigatório somente o preenchimento dos dados que possuem o nome em cor vermelha. E isso é válido para todos os cadastros do sistema.

O cadastro da versão de artefatos possui relacionamentos com diversos outros cadastros e uma série de dados. Os dados que vem de outros cadastros alimentam caixas de combinação e podem ser escolhidos pelo usuário. Uma versão pertence a um artefato. Desta forma, a seleção de um artefato para ser vinculado à versão é um campo obrigatório. E os artefatos vêm do cadastro de artefato. Assim, o preenchimento desse campo é realizado a partir de uma caixa de combinação cujos dados vem do cadastro de artefatos (apresentado na Figura 27).

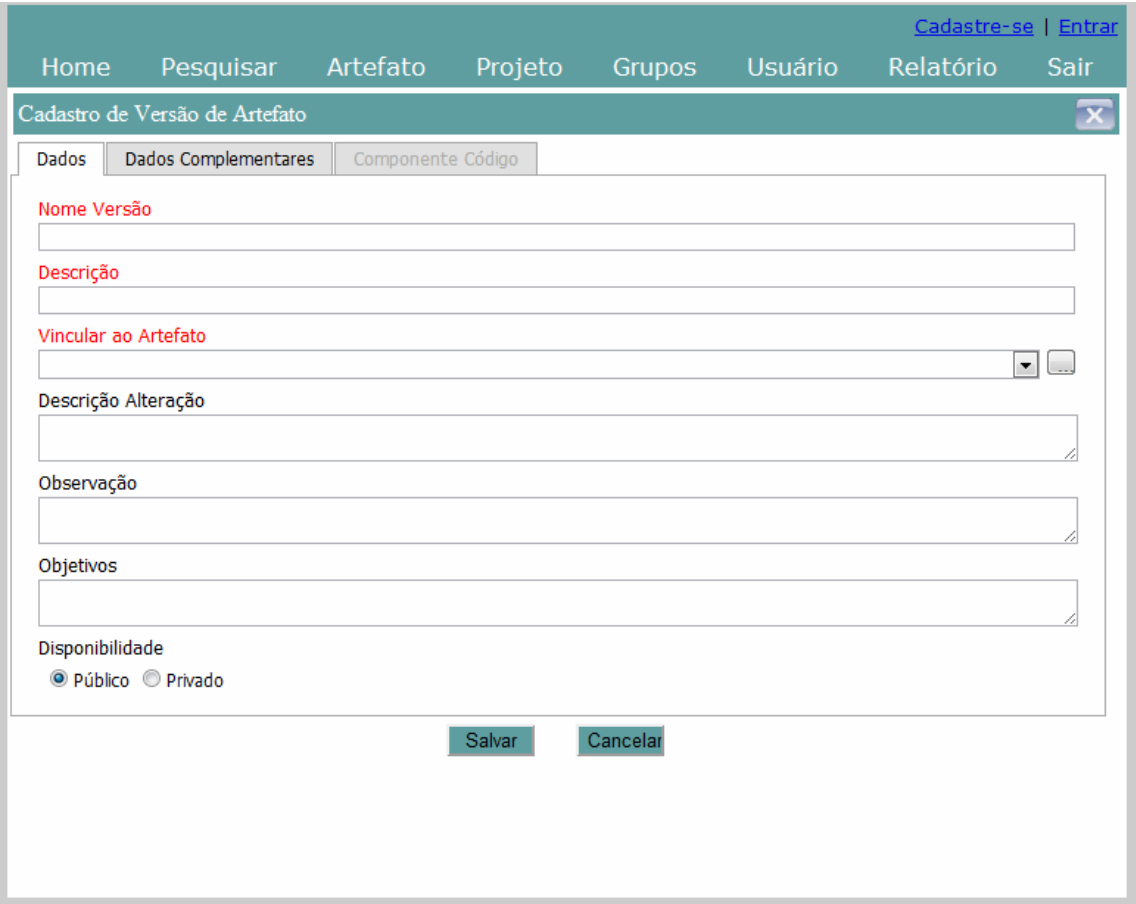

Figura 27 – Cadastro de versão de artefato – aba Dados

Na Figura 28 é apresentada a segunda aba desse cadastro, nela o usuário marcará os tipos de artefatos que fazem parte dessa versão, os motivos que levaram a criação dessa versão e quais ferramentas foram utilizadas no desenvolvimento dentre outros. Na área inferior do cadastro, no botão escolher arquivo, o usuário selecionará o artefato para ser salvo.

Logo acima do botão escolher arquivo, o usuário marcará se o artefato cadastrado é um objeto de código ou não, caso seja, a terceira aba do cadastro será habilitada. Ainda na segunda aba, no lado de cada opção há um botão. Por meio desses botões é possível acrescentar um novo registro no banco correspondente ao cadastro dos dados apresentados na respectiva caixa de combinação. Esses botões permitem acesso aos respectivos cadastros.

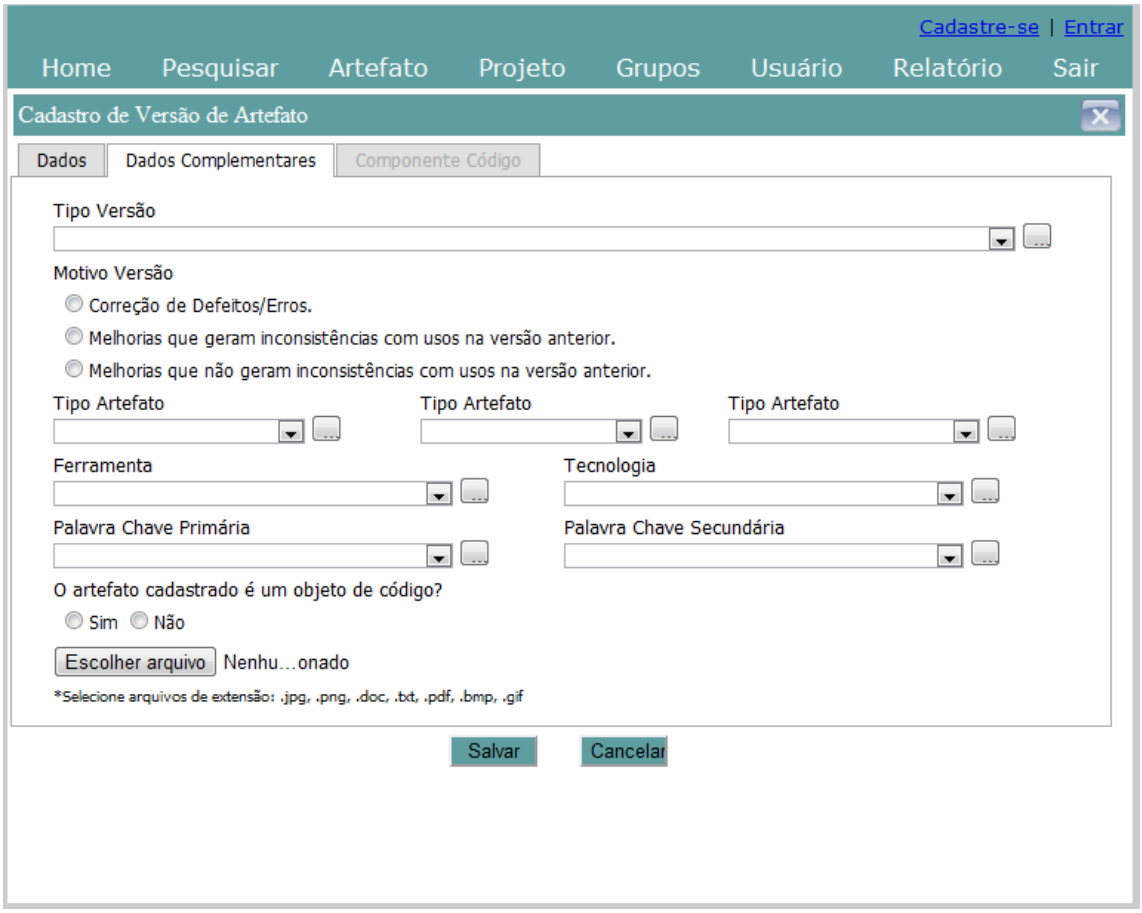

Figura 28 – Tela da segunda aba do cadastro de versão de artefatos

Na Figura 29 está a terceira aba do cadastro de versão. Essa aba se tornará disponível somente se indicado que o artefato é componente de código.

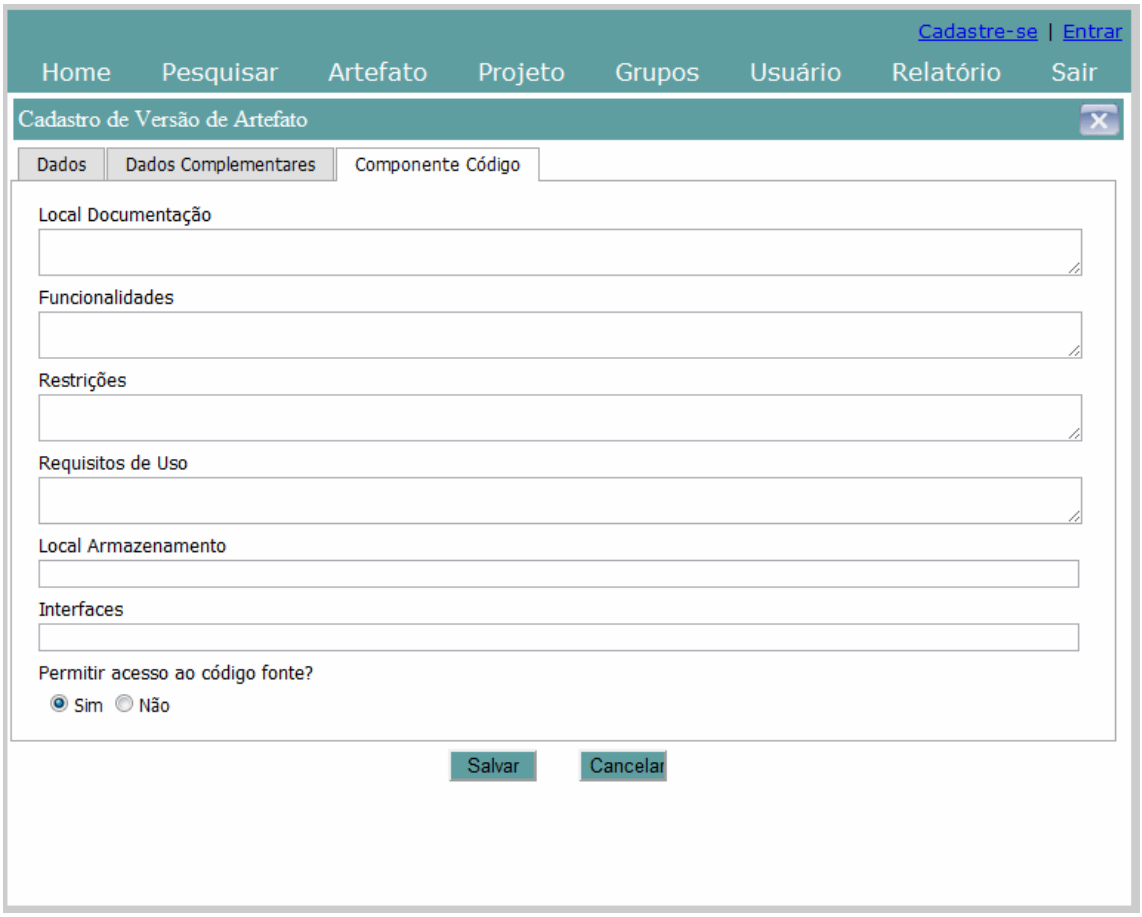

Figura 29 – Tela da terceira aba do cadastro de versão de artefatos

A Figura 30 mostra a tela usada para cadastrar vínculo entre artefatos e projetos.. O usuário selecionará qual versão de artefato deseja vincular a um projeto já cadastrado e realiza o cadastro. Da mesma forma que esse, os outros cadastros de vínculos são realizados, como: vincular atributos de qualidade e vincular usuários a grupos.

Figura 30 – Cadastro de vinculo de artefatos e projetos

O repositório disponibiliza aos seus usuários três formas de pesquisa dos artefatos cadastrados. As Figuras 31, 32 e 33 apresentam essas formas de busca.

A Figura 31 apresenta a busca realizada através de filtros predeterminados. O usuário deve escolher pelo menos um dos filtros e clicar no símbolo de lupa para que a pesquisa seja realizada. Se não existir nenhuma versão de artefato relacionado ao(s) filtro(s) selecionado(s) uma mensagem informará o usuário que a pesquisa não obteve itens. Caso contrário, as versões de artefatos que possuam relação com o(s) filtro(s) selecionado(s) serão listadas. No lado direito da lista há dois ícones, se o usuário clicar no ícone da lupa de uma das versões de artefatos listados, será mostrada uma ficha com a descrição dessa versão e se clicado no outro ícone o usuário fará o *download* da referida versão do artefato.

Nessa pesquisa com filtro, o usuário poderá selecionar até três valores para o campo tipos de artefato, que foram cadastradas por meio do cadastro de tipos de artefatos na aba artefato do menu. E dois tipos de ferramenta ou tecnologia, que foram cadastradas pelo cadastro de ferramenta/tecnologia na aba artefato.

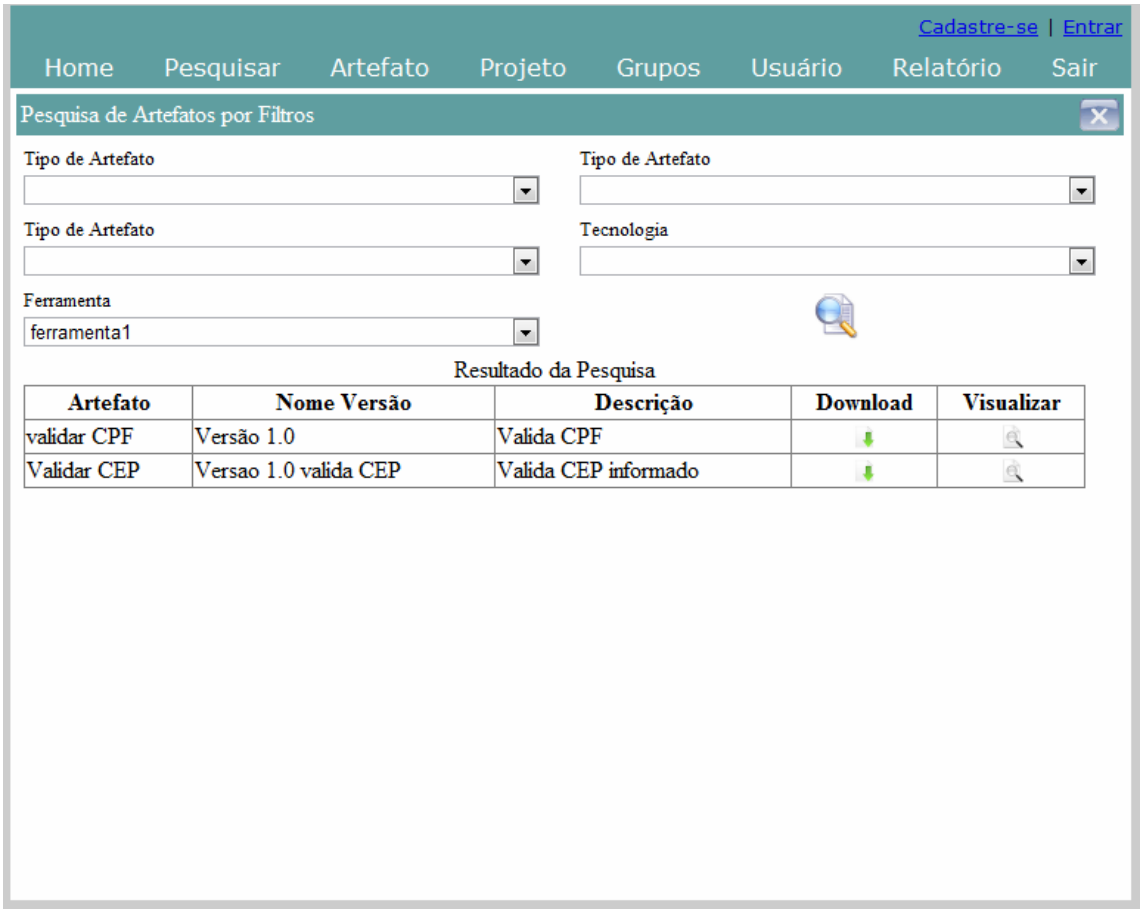

Figura 31 – Tela de pesquisa por filtro

A Figura 32 mostra a busca de artefato usando filtros e um campo de descrição. Esse campo de descrição permite ao usuário preencher com quais palavras ele deseja que o artefato possua em sua descrição. Como na busca por filtros, a busca por descrição pode ser realizada com o preenchimento de todos os campos ou somente com o preenchimento de um deles. A lista de artefatos filtrados, de acordo com os filtros e descrição informados, é mostrada na mesma tela e logo abaixo dos campos.

Nessa forma de pesquisa, o usuário pode informar até dois valores para tipos de artefato, uma ferramenta e uma tecnologia utilizada no desenvolvimento do artefato. Além de palavras que ele queira que esteja no campo descrição da versão do artefato.

A busca por descrição utiliza o mesmo método de acesso aos resultados que a pesquisa por filtro oferece, ou seja, o usuário pode ter acesso a uma ficha com a descrição do artefato e/ou ao *download* do mesmo.

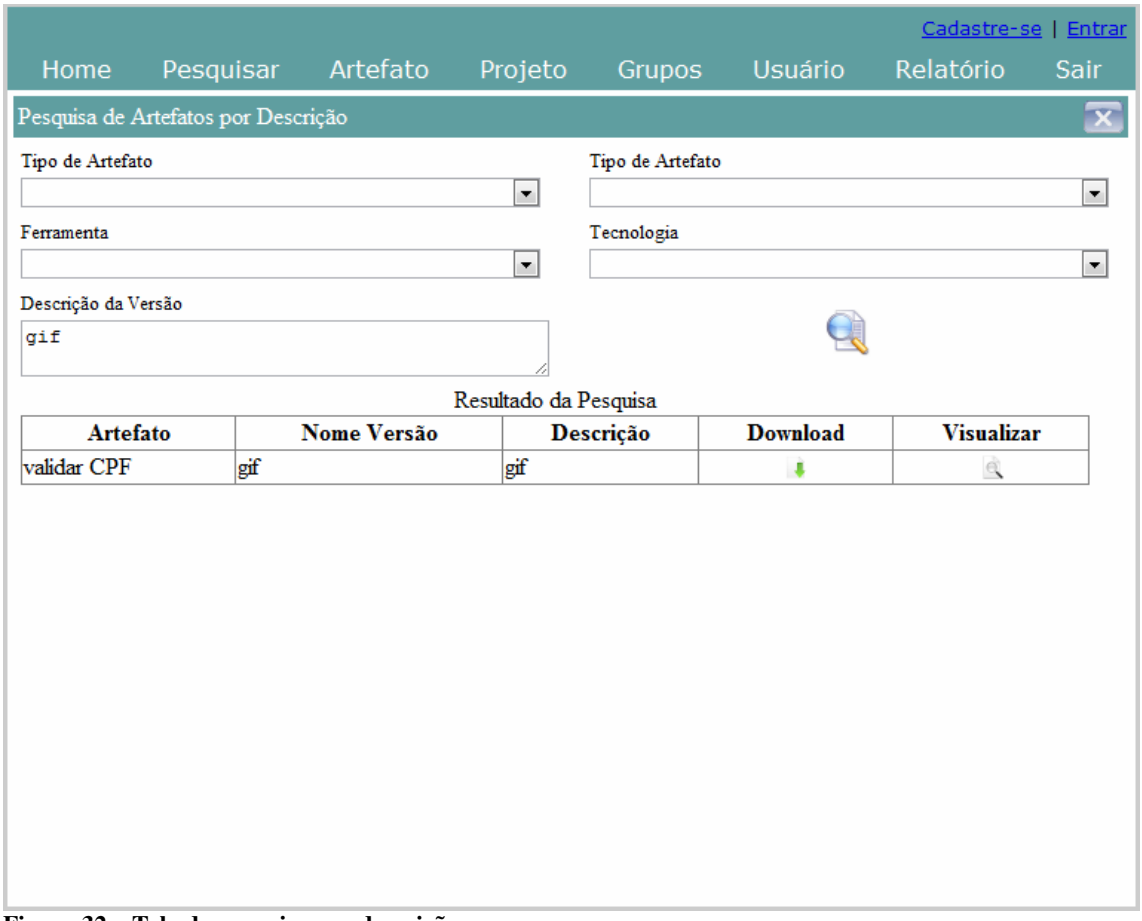

Figura 32 – Tela de pesquisa por descrição

A terceira forma de busca que o repositório disponibiliza é a por palavras-chave. A Figura 33 mostra a tela de pesquisa. Nessa tela o usuário poderá selecionar até dois valores

para o tipo de artefato, uma ferramenta e uma tecnologia utilizada no desenvolvimento. O diferencial dessa filtragem é a possibilidade do usuário escolher quais palavras-chave fazem parte da versão de artefato e quais palavras não fazem parte.

O usuário deve selecionar qual palavra-chave deseja que faça parte ou não da versão e clicar na seta para que a palavra seja adicionada na pesquisa. O método de visualização é o mesmo das outras duas formas de busca apresentadas nas Figuras 31 e 32.

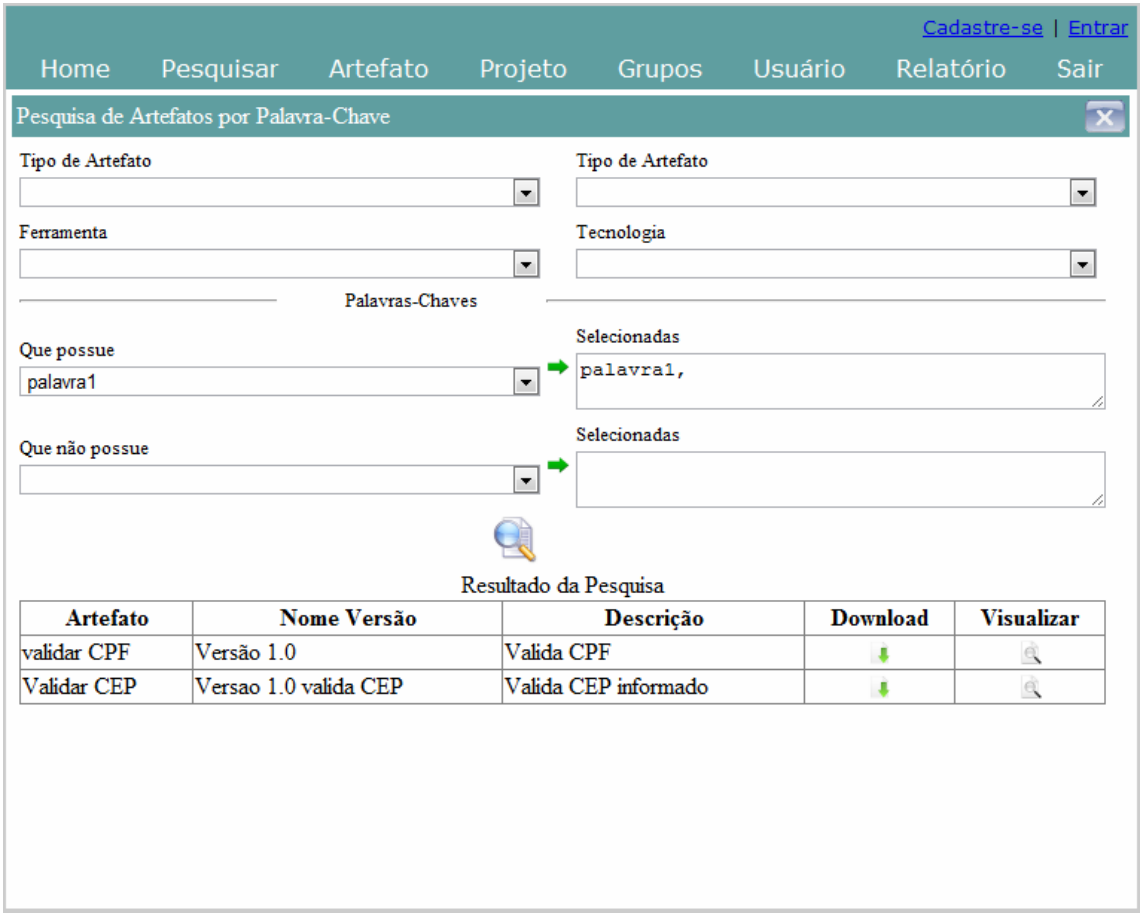

Figura 33 – Tela de pesquisa por palavras-chave

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção são apresentadas partes dos principais códigos desenvolvidos na implementação do sistema.

A Listagem 1 apresenta o código que recupera os dados no banco de dados e insere no AspxGridView para visualização do usuário. Uma variável do tipo FbCommand foi criada para armazenar a instrução SQL (*Structured Query Language*) que recupera os dados

relacionados aos artefatos no banco de dados. Um FbDataReader foi criado para armazenar os dados recuperados. Na linha 29, a conexão é aberta e os dados são recuperados do banco de dados através do *ExecuteReader*() e armazenados no *FbDataReader*. Depois disso, os dados são carregado no *DataSource* do *AspxGridView* e mostrados através do *DataBind*(). Nas linhas seguintes a conexão é fechada, o *DataReader* e o *FbCommand* são liberados.

```
22 private void carregarGridArtefato()
23         {
24             FbCommand cmd = new FbCommand("SELECT IDARTEFATO, NOME_ARTEFATO,
               DESCRICAO FROM ARTEFATO", conn);
25             FbDataReader dr;
26 
27             try
28             {
29                 conn.Open();
30                 dr = cmd.ExecuteReader();
31                 gridArtefato.DataSource = dr;
32                 gridArtefato.DataBind();
33                 conn.Close();
34                 dr.Dispose();
35                 cmd.Dispose();
36             }
37             catch (Exception Ex)
38             {
39                 ScriptManager.RegisterStartupScript(this, typeof(string),
                     "alertScript", "alert('Erro!!!\\n" + Ex.Message + "');", true);
40             }
41         }
```

Listagem 1 – Código que carrega os dados no *AspxGridView*

Na Listagem 2 está o código responsável por chamar uma tela para um novo cadastro, uma tela para alteração dos dados ou para exclusão. O componente usado para permitir ao usuário a inserção, alteração ou exclusão de dados foi o *AspxNavBar*.

O case 0, representado entre as linhas 47 e 50, é executado quando o usuário selecionar a opção inserir novo, dessa forma uma nova janela de cadastro será aberta através da função ***JanelaModal()*** que recebe por parâmetro o caminho da janela a ser aberta, a altura e o comprimento da janela.

O case 1, representado entre as linhas 51 e 59, verifica, no momento em que o usuário clicar em alterar, se existe alguma linha para ser alterada no *AspxGridView*. Se tiver, a sessão armazenará o identificador do campo selecionado para alteração e chamará, por meio da função **JanelaModal()**, a tela para alteração com os dados. Os dados relativos ao registro selecionado estarão carregados em seus respectivos campos para alteração.

O case 2 mostrado entre as linhas 61 e 69, tratará do clique relacionado à exclusão de um campo selecionado no *AspxGridView*. Ele recupera o identificador da linha selecionada e envia para uma tela de validação da exclusão.

```
protected void ASPxNavBar1_ItemClick(object source, DevExpress.Web.ASPxNavBar.NavBarItemEventArgs e)
        {
            switch (e.Item.VisibleIndex)
            {
                case 0:
                    Session["ArtefatoId"] = null;
                    ScriptManager.RegisterStartupScript(this, typeof(Page),
"javascript", "window.top.JanelaModal('Cadastros/CadArtefato.aspx', '280', '450')",
true);
                    break;
                case 1:
                    if (gridArtefato.VisibleRowCount != 0)
                    {
                        Session["ArtefatoId"] = Convert.ToInt32
                         (gridArtefato.GetRowValues(gridArtefato.FocusedRowIndex,
                         "IDARTEFATO"));
                        ScriptManager.RegisterStartupScript(this, typeof(Page),
                         "javascript", "window.top.JanelaModal
                         ('Cadastros/CadArtefato.aspx', '280', '450')", true);
                    }
                    else
                        ScriptManager.RegisterStartupScript(this, typeof(string),
                         "alertScript", "alert('Atenção!!!\\nSelecione um artefato
                         para alteração');", true);
                    break;
                case 2:
                    if (gridArtefato.VisibleRowCount != 0)
                    {
                        Session["ArtefatoId"] = Convert.ToInt32
                         (gridArtefato.GetRowValues(gridArtefato.FocusedRowIndex,
                         "IDARTEFATO"));
                        string texto = gridArtefato.GetRowValues
                         (gridArtefato.FocusedRowIndex, "NOME_ARTEFATO").ToString();
                         ValidarExclusao();
                    }
                    else
                        ScriptManager.RegisterStartupScript(this, typeof(string),
                         "alertScript", "alert('Atenção!!!\\nSelecione um artefato
                         para alteração');", true);
                    break;
            }
        }
```

Listagem 2 – Código para chamada de inserção, alteração e exclusão

A validação é mostrada na Listagem 3. Uma variável do tipo FbDataAdapter que vai receber a instrução SQL é criada. Essa instrução fará uma pesquisa em todas as tabelas que possuam ligação com ela, para verificar se esse registro a ser excluído não está sendo usado em outra tabela.

O resultado da consulta é armazenado em uma variável no tipo DataTable e é verificado se existem linhas nele. Se existirem linhas, é porque existe ligação entre as tabelas, portanto a exclusão não poderá ser feita, caso contrário o registro poderá ser excluído.

```
93 protected void ValidarExclusao()
94         {
95             FbDataAdapter daExc = new FbDataAdapter("SELECT IDARTEFATO FROM
96             VERSAO_ARTEFATO_USUARIO WHERE IDARTEFATO = " + Session["ArtefatoId"] + "
97             UNION " + "SELECT IDARTEFATO FROM VERSAO_ARTEFATO WHERE IDARTEFATO = " +
               Session["ArtefatoId"] + " UNION " + "SELECT IDARTEFATO FROM
98             QUALIDADE_VERSAO_ART WHERE IDARTEFATO = " + Session["ArtefatoId"] + "
               UNION " + "SELECT IDARTEFATO FROM ARTEFATO_PALAVRA_CHAVE WHERE IDARTEFATO
99             = " + Session["ArtefatoId"] + " UNION " + "SELECT IDARTEFATO FROM
               RELACIONAMENTO WHERE IDARTEFATO = " + Session["ArtefatoId"] + " UNION " +
100            "SELECT IDARTEFATO FROM RELACIONAMENTO WHERE IDARTEFATO_RELAC = " +
               Session["ArtefatoId"] + " UNION " + "SELECT IDARTEFATO FROM
101            ARTEFATO_PROJETO WHERE IDARTEFATO = " + Session["ArtefatoId"], conn);
102              DataTable dtExc = new DataTable();
103             daExc.Fill(dtExc);
104             if (dtExc.Rows.Count == 0)
105             {
106                 daExc.Dispose();
107                 dtExc.Dispose();
108                  string texto = gridArtefato.GetRowValues
109                  (gridArtefato.FocusedRowIndex, "NOME_ARTEFATO").ToString();
                     ScriptManager.RegisterStartupScript(this, typeof(Page),
                     "Menu.aspx", "window.top.JanelaModal
                     ('../Fichas/ValidarExclusao.aspx?cod=" + texto + "', '100',
                     '280')", true);
110             }
111             else
112             {
113                 ScriptManager.RegisterStartupScript(this, typeof(string),
                     "alertScript", "alert('Atenção!\\nO Artefato não pode ser
                     Excluído!\\nEsta sendo utilizado em outra Tabela.');", true);
114             }
115
116         }
```

Listagem 3 – Código de validação de exclusão de artefato

O código representado na Listagem 4 mostra a exclusão de um dado no banco de dados. A linha 76 cria uma variável do tipo FbCommand que recebe a instrução SQL responsável por excluir o registro. A *Session["ArtefatoId"]* possui o valor do registro que foi selecionado para exclusão. Nas linhas 79 e 80 a conexão é aberta e a instrução SQL é executada. Se algum erro ocorrer, o *catch* mostrará o erro. Nas linhas 88 e 89 a variável *cmd* é liberada e a conexão é fechada.

```
73 public void Deletar()
74         {
75
76             FbCommand cmd = new FbCommand("DELETE FROM ARTEFATO WHERE IDARTEFATO = "
               + Session["ArtefatoId"] + " ", conn);
77             try
78             {
79                 conn.Open();
80                 cmd.ExecuteNonQuery();
81             }
82             catch (Exception Ex)
83             {
```

```
84                  ScriptManager.RegisterStartupScript(this, typeof(string),
                    "alertScript", "alert('Erro!!\\n" + Ex.Message + "');", true);
85              }
86              finally
87              {
88                  cmd.Dispose();
89                  conn.Close();
90              }
91          }
```

Listagem 4 – Código de exclusão de um registro do banco

A Listagem 5 apresenta a validação de um artefato antes dele ser cadastrado no banco. Esse código será executado no momento em que o cliente clicar no botão salvar. O código mostrado faz uma pesquisa no banco de dados e retorna se já existe ou não um artefato com o mesmo nome cadastrado.

A linha 143 do código mostrado na figura verifica se o cadastro se trata da inserção de um novo artefato ou da alteração de um artefato já cadastrado. Caso seja um novo artefato, a função responsável pela inserção no banco será chamada, caso contrário, a função responsável pela alteração será executada.

A linha 147 chama a função **FecharModal** responsável por fechar a tela de cadastro e retornar para a tela a ficha, o parâmetro 1 passado para a função representa que a tela da ficha deverá ser atualizada.

```
123 public void ValidarArtefato()
124{
125     FbDataAdapter daInc = new FbDataAdapter("SELECT NOME_ARTEFATO FROM ARTEFATO " +
126     "WHERE (UPPER(NOME_ARTEFATO) = @NOME_ARTEFATO)", conn);
127
128     aInc.SelectCommand.Parameters.Add("@NOME_ARTEFATO", FbDbType.VarChar).Value = 1
        txtNomeArtefato.Text.ToUpper();
129             DataTable dtInc = new DataTable();
130             daInc.Fill(dtInc);
131
132             if (dtInc.Rows.Count != 0)
133             {
134                 daInc.Dispose();
135                 dtInc.Dispose();
136     criptManager.RegisterStartupScript(this,      typeof(string),      "alertScript",
        alert('Atenção!\\nJá existe um Artefato com esse nome cadastrado.');", true);
137                 return;
138             }
139             else
140             {
141                 daInc.Dispose();
142                 dtInc.Dispose();
143                 if (Session["ArtefatoId"] == null)
144                     Insert();
145                 else
146                     Update();
147                 ScriptManager.RegisterStartupScript(this, typeof(Page),
                    "javascript", indow.top.FecharModal(1)", true);
148             }
149
```

```
150 }
```

Listagem 5 – Código de validação de inserção de artefato

A Listagem 6 apresenta a função responsável por recuperar o identificador do novo item que será inserido na tabela do banco de dados. Uma sessão é usada para armazenar o valor desse identificador. Essa função sempre será chamada dentro do método responsável pela inserção de um registro no banco de dados.

```
public void GeraID(string nome)
 {
      try
      {
       FbDataAdapter da = new FbDataAdapter("SELECT IDTABELA FROM
       GERADORCODIGO(@NOMETABELA)", conn);
         DataTable dt = new DataTable();
        da.SelectCommand.Parameters.Add("@NOMETABELA", FbDbType.VarChar).Value = nome;
        da.Fill(dt);
        Session["ArtefatoId"] = Convert.ToInt32(dt.Rows[0].ItemArray[0]);
        da.Dispose();
        dt.Dispose();
        conn.Close();
      }
      catch (Exception Ex)
      {
       ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
       "alert('Erro\\n" + Ex.Message + "');", true);
            }
        }
```

Listagem 6 – Código de geração de identificador

Na Listagem 7 está o código responsável pela inserção dos dados no banco de dados. Na primeira linha do código foi criada uma variável do tipo *FbCommand* que receberá a instrução SQL. Depois, será chamada a função que gera um identificador dentro da tabela e os dados são passados para serem armazenados. A conexão é aberta e a instrução é realizada, caso ocorra alguma exceção, a execução cairá no *catch* e uma mensagem de erro será mostrada, caso contrário, a conexão será finalizada. Nas linhas 86 e 87, a conexão é fechada e a variável *cmd* é liberada.

```
64 private void Insert()
65 {
66     FbCommand cmd = new FbCommand("INSERT INTO ARTEFATO(IDARTEFATO, NOME_ARTEFATO,
67     DESCRICAO, OBSERVACAO) "+ "VALUES(@IDARTEFATO, @NOME_ARTEFATO, @DESCRICAO,
       @OBSERVACAO)", conn);
68     GeraID("ARTEFATO");
69     cmd.Parameters.Add("@IDARTEFATO", FbDbType.Integer).Value =
       Convert.ToInt32(Session["ArtefatoId"]);
70     cmd.Parameters.Add("@NOME_ARTEFATO", FbDbType.VarChar).Value =
       txtNomeArtefato.Text;
71     cmd.Parameters.Add("@DESCRICAO", FbDbType.VarChar).Value = txtDescricao.Text;
72     cmd.Parameters.Add("@OBSERVACAO", FbDbType.VarChar).Value = (txtObs.Text.Trim()
       != "") ? txtObs.Text : null;
73
74      try
```

```
75          {
76               conn.Open();
77               cmd.ExecuteNonQuery();
78
79           }
80           catch (Exception Ex)
81           {
82               ScriptManager.RegisterStartupScript(this, typeof(string),
                   "alertScript", "alert('Erro!!!\\n " + Ex.Message + "');", true);
83           }
84           finally
85           {
86               conn.Close();
87               cmd.Dispose();
88           }
89 }
```

Listagem 7 – Código de inserção de dados no banco de dados

O código responsável pela alteração de dados de um cadastro no banco é apresentado na Listagem 8. As linhas 93 e 94 trazem os dados do banco em que o identificador da tabela seja igual ao identificador do cadastro que foi selecionado para alteração. O valor selecionado vem armazenado na *Session[“ArtefatoId”]*.

Os novos dados para alteração são recuperados, a conexão é aberta e a instrução realizada. Caso não ocorra nenhuma exceção, a alteração será realizada com sucesso e essa tela será fechada, caso contrário, uma mensagem de erro será mostrada através do *catch*. Nas linhas 112 e 113 a conexão é fechada e a variável do tipo *FbCommand* é liberada.

```
91 private void Update()
92{
93     FbCommand cmd = new FbCommand("UPDATE ARTEFATO SET NOME_ARTEFATO =
       @NOME_ARTEFATO,      DESCRICAO = @DESCRICAO, OBSERVACAO = @OBSERVACAO "+
94       "WHERE IDARTEFATO = " + Session["ArtefatoId"], conn);
95
96     cmd.Parameters.Add("@NOME_ARTEFATO", FbDbType.VarChar).Value =
txtNomeArtefato.Text;
97     cmd.Parameters.Add("@DESCRICAO", FbDbType.VarChar).Value = txtDescricao.Text;
98     cmd.Parameters.Add("@OBSERVACAO", FbDbType.VarChar).Value = (txtObs.Text.Trim()
       != "") ? txtObs.Text : null;
99
100           try
101           {
102               conn.Open();
103               cmd.ExecuteNonQuery();
104
105           }
106           catch (Exception Ex)
107           {
108           ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
             "alert('Erro!!!\\n " + Ex.Message + "');", true);
109           }
110           finally
111           {
112               conn.Close();
113               cmd.Dispose();
114           }
```

```
115         }
```

Listagem 8 - Código de alteração de dados no banco de dados

A Listagem 9 contém o código responsável por recuperar os dados do banco e carregá-los nos respectivos campos para alteração. Nas primeiras linhas, é realizada uma pesquisa no banco de dados para recuperar o registro com identificador igual ao selecionado para alteração. Na linha 33, os dados são carregados no *DataTable* e nas linhas seguintes cada componente recebe o valor relacionado a linha do *DataTable*.

```
26 public void CarregaDados()
27 {
28     FbDataAdapter da = new FbDataAdapter("SELECT NOME_ARTEFATO, DESCRICAO,
       OBSERVACAO FROM ARTEFATO WHERE IDARTEFATO = " + Session["ArtefatoId"], conn);
29      DataTable dt = new DataTable();
30
31      try
32      {
33           da.Fill(dt);
34           txtNomeArtefato.Text = dt.Rows[0].ItemArray[0].ToString();
35           txtDescricao.Text = dt.Rows[0].ItemArray[1].ToString();
36           txtObs.Text = dt.Rows[0].ItemArray[2].ToString();
37
38      }
39      catch(Exception Ex)
40      {
41     ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript", `
       "alert('Erro!!!\\n " + Ex.Message + "');", true);
42       }
43 }
```

Listagem 9 – Código de carrega os dados para alteração

A Listagem 10 mostra o código usado para criar uma janela no estilo modal que utilizada para alguns cadastros que possuem poucos campos. A função **JanelaModal** recebe por parâmetro o caminho da tela a ser aberta, a altura e a largura da janela.

As linhas 45 e 46 recuperam a altura e a largura da tela e dividem por 2 para posicionar a tela que será criada no centro do monitor. A linha 48 cria uma *div* com os atributos da classe **divModal_**, mostrado na Listagem 11 e essa *div* é criada sobre o *body*, ou seja o corpo do menu que é a tela principal. A linha 49 cria outra *div* sobre o corpo da tela principal, com os atributos da classe **divJanela_** usando os tamanhos que foram passados por parâmetros e centralizada de acordo com os valores que foram recuperados e armazenados nas variáveis anteriormente. A linha 56 cria um *iframe* dentro da *div* centralizada e esse *iframe* irá carregar o cadastro que foi passado por parâmetro.

A função **FecharModal** removerá as *divs* que foram criadas. Essa função recebe por parâmetro um valor que se for igual a 1, direcionará a execução para o evento do clique do

botão atualiza que, por sua vez chamará a função que carrega o *AspxGridView* atualizando os dados da página.

```
43 function JanelaModal(url, altura, largura) {
44
45                 var left_ = ($('body')[0].clientWidth - largura) / 2;
46                 var top_ = ($('body')[0].clientHeight - altura) / 2;
47
48                 $('<div id="c">').attr('class', 'divModal_').appendTo('body');
49                 $('<d iv id="j">').attr('class', 'divJanela_').css({
50                     height: altura + 'px',
51                     width: largura + 'px',
52                     left: left_ + 'px',
53                     top: top_ + 'px'
54                 }).appendTo('body');
55
56             $('<iframe name="iframePrimeiro" scrolling="no" src="' + url +
               '">').attr('class', 'ifr_').appendTo('#j');
57
58         }
59
60       function FecharModal(atualiza) {
61           $('#j').remove();
62           $('#c').remove();
63           if (atualiza == 1) {
64            document.getElementById("frame1").contentWindow.document.
              getElementById("btAtualiza").click();
65           }
66       }
```

Listagem 10 – Código responsável por criar janela de efeito modal

Os códigos mostrados na Listagem 11 são CSS (*Cascading Style Sheets*) usados para determinar as classes que criaram as *divs* mostradas na Listagem 10.

A linha 18 mostra a classe **.divModal_**. Ela possui posição absoluta, ou seja, ela será posicionado de acordo com o corpo da tela principal, *body*. Nas linhas 21 a 24 a *div* será posicionada no início da tela e com tamanho igual a tela principal. A linha 25 atribui a cor de preenchimento como preta para a div. As linhas 26 e 27 atribuem uma opacidade de 0,5 para a cor preta e a linha 28 atribui um valor para o eixo z da tela, seria como uma sobreposição nos demais elementos da tela.

A segunda classe **.divJanela_,** recebe posição absoluta e ela é colocada sobre a **.divModal_** por meio do **z-index(eixo z)**. A linha 35 mostra a classe **.ifr_** que possui o tamanho da **.divJanela_**, pois ele ficará dentro dela.

```
17 <style type="text/css">
18         .divModal_
19         {
20             position: absolute;
21             top: 0;
22             left: 0;
23             width: 100%;
24             height: 100%;
25             background-color: #000;
```

```
26              opacity: 0.5;
27              filter: alpha(opacity=65);
28              z-index: 10;
29          }
30          .divJanela_
31          {
32              position: absolute;
33              z-index: 20;
34          }
35          .ifr_
36          {
37              width: 100%;
38              height: 100%;
39          }
40      </style>
41      <script type = “text/javascript” >
```

Listagem 11 – Código CSS

A Listagem 12 apresenta o cadastro de artefatos no banco de dados. Para que o artefato pudesse ser carregado foi usado um componente chamado *FileUpload.* Ele possibilita que o usuário carregue qualquer documento que esteja no computador. Na linha 339 é verificado se o usuário selecionou um artefato para ser cadastrado, caso contrário uma mensagem de erro será mostrada. Entre as linhas 343 e 347 o artefato selecionado é transformado em binário e armazenado na variável *imageBytes* e seu nome, caminho que o artefato estava armazenado e extensão são recuperados. Na linha 348 é verificado se o artefato selecionado para inserção atende às restrições de extensão. Se a extensão não for válida, uma mensagem de erro será mostrada, caso contrário o artefato será cadastrado.

Na linha 444 é chamada a função **VincularArtefato** responsável pela versão que está sendo cadastrada, ao artefato que ela diz respeito e ao usuário que está cadastrando o referido artefato.

```
337 private void Inserir()
338 {
339 if (FileUpload1.HasFile)
340 {
341 byte[] imageBytes;
342
343 imageBytes = new byte[FileUpload1.PostedFile.InputStream.Length + 1];
344 FileUpload1.PostedFile.InputStream.Read(imageBytes, 0, imageBytes.Length);
345 string nome = FileUpload1.PostedFile.FileName;
346 string caminho = MapPathSecure(nome);
347 string extensao = Path.GetExtension(caminho);
348 if ((extensao != ".jpg") || (extensao != ".png") || (extensao != ".doc") || (extensao
!= ".txt") || (extensao != ".pdf") || (extensao != ".bmp") || (extensao != ".gif"))
349 {
```

```
350
351 FbCommand cmd = new FbCommand("INSERT INTO VERSAO_ARTEFATO(ARTEFATO, EXTENSAO,
IDVERSAO, IDARTEFATO, IDTIPO_VERSAO, NOME_VERSAO, DESCRICAO_VERSAO, DESCRICAO_ALTERACAO, "
+
352 "DATA_ALTERACAO, IDUSUARIO, DISPONIBILIDADE, LOCAL_ARMAZ, MOTIVO_VERSAO, IDTECNOLOGIA,
IDFERRAMENTA, IDTIPOARTEFATO, IDSUBTIPOARTEFATO, IDSUBSUBTIPOARTEFATO, " +
353 "OBSERVACAO, OBJETIVOS, FUNCIONALIDADES, RESTRICOES, INTERFACES, REQUISITO_USO,
LOCAL_DOCUMENTACAO, ACESSO_COD_FONTE, IDPALAVRA_PRIMARIA,
IDPALAVRA_SECUNDARIA)VALUES(@ARTEFATO, @EXTENSAO, @IDVERSAO, " +
354 "@IDARTEFATO, @IDTIPO_VERSAO, @NOME_VERSAO, @DESCRICAO_VERSAO,
@DESCRICAO_ALTERACAO,@DATA_ALTERACAO, @IDUSUARIO, @DISPONIBILIDADE, @LOCAL_ARMAZ,
@MOTIVO_VERSAO, @IDTECNOLOGIA," +
355 "@IDFERRAMENTA,@IDTIPOARTEFATO,@IDSUBTIPOARTEFATO,@IDSUBSUBTIPOARTEFATO,
@OBSERVACAO,@OBJETIVOS, @FUNCIONALIDADES, @RESTRICOES, @INTERFACES,@REQUISITO_USO, "+
356 "@LOCAL_DOCUMENTACAO, @ACESSO_COD_FONTE, @IDPALAVRA_PRIMARIA, @IDPALAVRA_SECUNDARIA)",
conn);
357
358 GeraID("VERSAO_ARTEFATO");
359
360 cmd.Parameters.Add("@ARTEFATO", FbDbType.Binary,
Convert.ToString(imageBytes).Length).Value = imageBytes;
361 cmd.Parameters.Add("@EXTENSAO", FbDbType.VarChar).Value = extensao;
362 cmd.Parameters.Add("@IDVERSAO", FbDbType.Integer).Value =
Convert.ToInt32(Session["IDVERSAO_ARTEFATO"]);
363 cmd.Parameters.Add("@IDARTEFATO", FbDbType.Integer).Value = ddlArtefato.SelectedValue;
364 if (ddlTipoVersao.SelectedValue == "")
cmd.Parameters.Add("@IDTIPO_VERSAO", FbDbType.Integer).Value = null;
365 else cmd.Parameters.Add("@IDTIPO_VERSAO", FbDbType.Integer).Value =
Convert.ToInt32(ddlTipoVersao.SelectedValue);
366 if (txtNome.Text == "") cmd.Parameters.Add("@NOME_VERSAO", FbDbType.VarChar).Value =
"";
367 else cmd.Parameters.Add("@NOME_VERSAO", FbDbType.VarChar).Value = txtNome.Text;
368 if (txtDescricao.Text == "") cmd.Parameters.Add("@DESCRICAO_VERSAO",
FbDbType.VarChar).Value = "";
369 else cmd.Parameters.Add("@DESCRICAO_VERSAO", FbDbType.VarChar).Value =
txtDescricao.Text;
370 if (txtDescricaoAlteracao.Text == "") cmd.Parameters.Add("@DESCRICAO_ALTERACAO",
FbDbType.VarChar).Value = "";
371 else cmd.Parameters.Add("@DESCRICAO_ALTERACAO", FbDbType.VarChar).Value =
txtDescricaoAlteracao.Text;
372
373 string dataAtual = string.Format("{0:d}", Convert.ToDateTime(DateTime.Now));
374 cmd.Parameters.Add("@DATA_ALTERACAO", FbDbType.VarChar).Value = dataAtual;
375 cmd.Parameters.Add("@IDUSUARIO", FbDbType.Integer).Value =
Convert.ToInt32(Session["UsuarioId"]);
376
377 cmd.Parameters.Add("@DISPONIBILIDADE", FbDbType.Integer).Value =
Convert.ToInt32(rbDisponibilidade.SelectedValue);
378
379 if (txtArmazenamento.Text == "") cmd.Parameters.Add("@LOCAL_ARMAZ",
FbDbType.VarChar).Value = "";
```

```
380 else cmd.Parameters.Add("@LOCAL_ARMAZ", FbDbType.VarChar).Value =
txtArmazenamento.Text;
381
382 if (rbVersao.SelectedValue == "") cmd.Parameters.Add("@MOTIVO_VERSAO",
FbDbType.Integer).Value = null;
383 else cmd.Parameters.Add("@MOTIVO_VERSAO", FbDbType.VarChar).Value =
Convert.ToInt32(rbVersao.SelectedValue);
384
385 if (ddlTecnologia.SelectedValue == "") cmd.Parameters.Add("@IDTECNOLOGIA",
FbDbType.Integer).Value = null;
386 else cmd.Parameters.Add("@IDTECNOLOGIA", FbDbType.Integer).Value =
Convert.ToInt32(ddlTecnologia.SelectedValue);
387
388 if (ddlFerramenta.SelectedValue == "") cmd.Parameters.Add("@IDFERRAMENTA",
FbDbType.Integer).Value = null;
389 else cmd.Parameters.Add("@IDFERRAMENTA", FbDbType.Integer).Value =
Convert.ToInt32(ddlFerramenta.SelectedValue);
390
391 if (ddlTipoArtefato.SelectedValue == "") cmd.Parameters.Add("@IDTIPOARTEFATO",
FbDbType.Integer).Value = null;
392 else cmd.Parameters.Add("@IDTIPOARTEFATO", FbDbType.Integer).Value =
Convert.ToInt32(ddlTipoArtefato.SelectedValue);
393
394 if (ddlTipoArtefato1.SelectedValue == "") cmd.Parameters.Add("@IDSUBTIPOARTEFATO",
FbDbType.Integer).Value = null;
395 else cmd.Parameters.Add("@IDSUBTIPOARTEFATO", FbDbType.Integer).Value =
Convert.ToInt32(ddlTipoArtefato1.SelectedValue);
396
397 if (ddlTipoArtefato2.SelectedValue == "") cmd.Parameters.Add("@IDSUBSUBTIPOARTEFATO",
FbDbType.Integer).Value = null;
398 else cmd.Parameters.Add("@IDSUBSUBTIPOARTEFATO", FbDbType.Integer).Value =
Convert.ToInt32(ddlTipoArtefato2.SelectedValue);
399
400 if (txtObs.Text == "") cmd.Parameters.Add("@OBSERVACAO", FbDbType.VarChar).Value = "";
401 else cmd.Parameters.Add("@OBSERVACAO", FbDbType.VarChar).Value = txtObs.Text;
402
403 if (txtObjetivos.Text == "") cmd.Parameters.Add("@OBJETIVOS", FbDbType.VarChar).Value =
"";
404 else cmd.Parameters.Add("@OBJETIVOS", FbDbType.VarChar).Value = txtObjetivos.Text;
405
406 if (txtFuncionalidades.Text == "") cmd.Parameters.Add("@FUNCIONALIDADES",
FbDbType.VarChar).Value = "";
407 else cmd.Parameters.Add("@FUNCIONALIDADES", FbDbType.VarChar).Value =
xtFuncionalidades.Text;
408
409 if (txtRestricoes.Text == "") cmd.Parameters.Add("@RESTRICOES", FbDbType.VarChar).Value
= "";
410 else cmd.Parameters.Add("@RESTRICOES", FbDbType.VarChar).Value = txtRestricoes.Text;
411
412 if (txtInterfaces.Text == "") cmd.Parameters.Add("@INTERFACES", FbDbType.VarChar).Value
= "";
```

```
413 else cmd.Parameters.Add("@INTERFACES", FbDbType.VarChar).Value = txtInterfaces.Text;
414
415 if (txtRequisitos.Text == "") cmd.Parameters.Add("@REQUISITO_USO",
FbDbType.VarChar).Value = "";
416 else cmd.Parameters.Add("@REQUISITO_USO", FbDbType.VarChar).Value = txtRequisitos.Text;
417
418 if (txtDocumentacao.Text == "") cmd.Parameters.Add("@LOCAL_DOCUMENTACAO",
FbDbType.VarChar).Value = "";
419 else cmd.Parameters.Add("@LOCAL_DOCUMENTACAO", FbDbType.VarChar).Value =
txtDocumentacao.Text;
420
421 cmd.Parameters.Add("@ACESSO_COD_FONTE", FbDbType.Integer).Value =
Convert.ToInt32(rbAcesso.SelectedValue);
422
423 if (ddlPalavraChave1.SelectedValue == "") cmd.Parameters.Add("@IDPALAVRA_PRIMARIA",
FbDbType.Integer).Value = null;
424 else cmd.Parameters.Add("@IDPALAVRA_PRIMARIA", FbDbType.Integer).Value =
Convert.ToInt32(ddlPalavraChave1.SelectedValue);
425
426 if (ddlPalavraChave2.SelectedValue == "") cmd.Parameters.Add("@IDPALAVRA_SECUNDARIA",
FbDbType.Integer).Value = null;
427 else cmd.Parameters.Add("@IDPALAVRA_SECUNDARIA", FbDbType.Integer).Value =
Convert.ToInt32(ddlPalavraChave2.SelectedValue);
428
429 try
430 {
431 conn.Open();
432 cmd.ExecuteNonQuery();
433 }
434 catch (Exception ex)
435 {
436 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!'" + ex.Message + ");", true);
437 return;
438 }
439 finally
440 {
441 conn.Close();
442 cmd.Dispose();
443 }
444 VincularArtefato();
445 Response.Redirect("../Fichas/FichaVersaoArtefato.aspx");
446 }
447 else
448 {
449 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Selecione um artefato para inclusão');", true);
450 return;
```

```
451 }
452 }
453 else
454 {
455 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Selecione um arquivo com outra extensão');", true);
456 return;
457 }
458 }
```

Listagem 12 – Inserção de artefatos no banco de dados

A Listagem 13 apresenta a função usada para realizar a pesquisa através de filtros. A linha 116 verifica se algum filtro foi selecionado, caso contrário uma mensagem informando o usuário que filtros não foram selecionados será mostrada. Uma variável chamada *sql* do tipo *string* foi usada para armazenar a instrução SQL responsável por recuperar os dados da pesquisa. Nas linhas seguintes, a variável *sql* recebrá as instruções de acordo com cada filtro selecionado. Entre as linhas 152 e 170, a pesquisa será executada de acordo com a instrução montada.

```
113 protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
114 {
115 string sql = "";
116 if ((ddlTipoArtefato.SelectedValue != "") || (ddlTipoArtefato1.SelectedValue != "") ||
(ddlTipoArtefato2.SelectedValue != "")
117 || (ddlFerramenta.SelectedValue != "") || (ddlTecnologia.SelectedValue != ""))
118 {
119 sql = " SELECT VERSAO_ARTEFATO.IDARTEFATO, VERSAO_ARTEFATO.IDVERSAO,
VERSAO_ARTEFATO.DESCRICAO_VERSAO, VERSAO_ARTEFATO.NOME_VERSAO, "+
120 " ARTEFATO.IDARTEFATO, ARTEFATO.NOME_ARTEFATO FROM VERSAO_ARTEFATO "+
121 " INNER JOIN ARTEFATO ON ARTEFATO.IDARTEFATO = VERSAO_ARTEFATO.IDARTEFATO WHERE 1 = 1
";
122
123 if (ddlTipoArtefato.SelectedValue != "")
124 {
125 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue +
" OR " +
126 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue + " OR " +
127 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue;
128 }
129 if (ddlTipoArtefato1.SelectedValue != "")
130 {
131 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue +
" OR " +
132 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue + " OR " +
```

```
133 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue;
134 }
135 if (ddlTipoArtefato2.SelectedValue != "")
136 {
137 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato2.SelectedValue +
" OR " +
138 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato2.SelectedValue + " OR " +
139 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato2.SelectedValue;
140 }
141 if (ddlFerramenta.SelectedValue != "")
142 {
143 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlFerramenta.SelectedValue + "OR
"+
144 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlFerramenta.SelectedValue;
145 }
146 if (ddlTecnologia.SelectedValue != "")
147 {
148 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlTecnologia.SelectedValue + "OR
" +
149 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlTecnologia.SelectedValue;
150 }
151
152 FbCommand cmd = new FbCommand(sql.ToString(), conn);
153 FbDataReader dr = null;
154 try
155 {
156 conn.Open();
157 dr = cmd.ExecuteReader();
158 gridInsercao.DataSource = dr;
159 gridInsercao.DataBind();
160 }
161 catch (Exception Ex)
162 {
163 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!\\n " + Ex.Message + "');", true);
164 }
165 finally
166 {
167 conn.Close();
168 dr.Dispose();
169 cmd.Dispose();
170 }
171 }
172 else
173 {
```

```
174 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!\\n Selecione pelo menos um filtro');", true);
175 }
176
177 }
```

Listagem 13 – Pesquisa por filtros

A Listagem 14 apresenta a função responsável por efetuar o *download* do artefato selecionado pelo usuário ou apresentar os dados do mesmo. O código mostrado na linha 189, recupera o index da linha clicada do *AspxGridView* pelo usuário. Na linha 190 é recuperado o identificador da versão de artefato que esta carregado na linha que foi clicada. Se o usuário clicou no ícone da coluna *Download* do *AspxGridView* a função **FazerDownload** será executada, caso tenha sido clicada na coluna Visualizar, uma tela com os dados da versão será mostrada.

```
185 protected void gridInsercao_RowCommand1(object sender, GridViewCommandEventArgs e)
186 {
187 if (e.CommandName == "Download")
188 {
189 int index = int.Parse((string)e.CommandArgument);
190 int idVersao = int.Parse(gridInsercao.DataKeys[index] ["IDVERSAO"].ToString());
191 FazerDownload(idVersao);
192 }
193 if (e.CommandName == "Visualizar")
194 {
195 int index = int.Parse((string)e.CommandArgument);
196 Session["IDVERSAO_ARTEFATO"] = int.Parse(gridInsercao.DataKeys[index]
["IDVERSAO"].ToString());
197 ScriptManager.RegisterStartupScript(this, typeof(Page), "Menu.aspx",
"window.top.JanelaModal('Pesquisar/VisualizarVersaoArtefato.aspx', 480, 740)", true);
198 }
199 }
```

Listagem 14 – Código para download ou visualização

A Listagem 15 apresenta o código responsável por fazer o *download* do arquivo que está no banco de dados para a pasta *Download* do computador do usuário. Na linha 203 é realizada uma pesquisa no banco de dados para recuperar os dados da versão do artefato que foi selecionado para d*ownload.* Na linha 207 esses dados são lidos, nas linhas seguintes a extensão deles é verificada para determinar o tipo do artefato que será baixado e nas linhas 225 e 226 o artefato é baixado.

```
201 private void FazerDownload(int idVersao)
```

```
202 {
203 string SQL = "SELECT IDVERSAO, ARTEFATO, EXTENSAO, NOME_VERSAO, DISPONIBILIDADE,
IDUSUARIO FROM VERSAO_ARTEFATO WHERE IDVERSAO = " + idVersao;
204 FbCommand cmd = new FbCommand(SQL, conn);
205 conn.Open();
206 FbDataReader dr = cmd.ExecuteReader();
207 if (dr.Read())
208 {
209 Response.AddHeader("content-disposition", "attachment; filename=" +
dr["NOME_VERSAO"].ToString() + dr["EXTENSAO"].ToString() + "");
210 if (dr["EXTENSAO"].ToString() == ".jpg")
211 Response.ContentType = "Image/jpeg";
212 else if (dr["EXTENSAO"].ToString() == ".png")
213 Response.ContentType = "Image/png";
214 else if (dr["EXTENSAO"].ToString() == ".doc")
215 Response.ContentType = "Application/.doc";
216 else if (dr["EXTENSAO"].ToString() == ".txt")
217 Response.ContentType = "Application/txt";
218 else if (dr["EXTENSAO"].ToString() == ".pdf")
219 Response.ContentType = "Application/pdf";
220 else if (dr["EXTENSAO"].ToString() == ".bmp")
221 Response.ContentType = "Image/bmp";
222 else if (dr["EXTENSAO"].ToString() == ".gif")
223 Response.ContentType = "Image/gif";
224
225 Response.BinaryWrite((byte[])dr["ARTEFATO"]);
226 Response.End();
227 }
228 conn.Close();
229 dr.Dispose();
230 cmd.Dispose();
231 }
```

Listagem 15 – Código para download do artefato do banco

A Listagem 16 apresenta a função usada na para realizar a pesquisa por descrição dos artefatos. A linha 106 verifica se algum filtro foi selecionado ou se o campo descrição foi preenchido, caso contrário uma mensagem informando o usuário será mostrada. Uma variável chamada *sql* do tipo *string* foi usada para armazenar a instrução SQL responsável por recuperar os dados da pesquisa. Nas linhas seguintes, a variável *sql* irá receber as instruções de acordo com cada filtro selecionado. Na linha 140, é criado um *array* de *string* com os palavras que foram informadas no campo descrição. Na linha 141 é feita uma pesquisa no banco de dados para recuperar a descrição de todas as versões de artefatos cadastradas no

banco. A partir dessa consulta serão comparados todos as palavras dos resultados obtidos na pesquisa com as palavras que foram informadas no campo de descrição da pesquisa. Caso as palavras sejam iguais, o identificador dessa versão será utilizado na montagem da SQL da pesquisa. O código para isso está entre as linhas 147 e 167.

```
103 protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
104 {
105 string sql = "";
106 if ((ddlTipoArtefato.SelectedValue != "") || (ddlTipoArtefato1.SelectedValue ! = "")
107 || (ddlFerramenta.SelectedValue != "") || (ddlTecnologia.SelectedValue != "") ||
(txtDescricao.Text != ""))
108 {
109 sql = " SELECT VERSAO_ARTEFATO.IDARTEFATO, VERSAO_ARTEFATO.IDVERSAO,
VERSAO_ARTEFATO.DESCRICAO_VERSAO, VERSAO_ARTEFATO.NOME_VERSAO, " +
110 " ARTEFATO.IDARTEFATO, ARTEFATO.NOME_ARTEFATO FROM VERSAO_ARTEFATO " +
111 " INNER JOIN ARTEFATO ON ARTEFATO.IDARTEFATO = VERSAO_ARTEFATO.IDARTEFATO WHERE 1 = 1
";
112
113 if (ddlTipoArtefato.SelectedValue != "")
114 {
115 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue +
" OR " +
116 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue + " OR " +
117 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue;
118 }
119 if (ddlTipoArtefato1.SelectedValue != "")
120 {
121 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue +
" OR " +
122 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue + " OR " +
123 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue;
124 }
125 if (ddlFerramenta.SelectedValue != "")
126 {
127 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlFerramenta.SelectedValue + "OR
" +
128 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlFerramenta.SelectedValue;
129 }
130 if (ddlTecnologia.SelectedValue != "")
131 {
132 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlTecnologia.SelectedValue + "OR
" +
133 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlTecnologia.SelectedValue;
134 }
135
136 if (txtDescricao.Text != "")
```

```
137 {
138 string ultimo = "";
139 string descricao = txtDescricao.Text;
140 string[] descPartes = descricao.Split(' ');
141 FbCommand cmd1 = new FbCommand("SELECT IDVERSAO, DESCRICAO_VERSAO FROM
VERSAO_ARTEFATO", conn);
142 FbDataReader dr1 = null;
143 try
144 {
145 conn.Open();
146 dr1 = cmd1.ExecuteReader();
147 while (dr1.Read())
148 {
149 string[] descBanco = dr1.GetString(1).Split(' ');
150 for (int i = 0; i < descBanco.Length; i++)
151 {
152 for (int j = 0; j < descPartes.Length; j++)
153 {
154 if (descBanco[i].ToUpper() == descPartes[j].ToUpper())
155 {
156 if (ultimo != dr1.GetString(0))
157 {
158 if(ultimo == "")
159 sql = sql + " AND VERSAO_ARTEFATO.IDVERSAO = " + dr1.GetString(0);
160 else
161 sql = sql + " AND VERSAO_ARTEFATO.IDVERSAO = " + dr1.GetString(0);
162 ultimo = dr1.GetString(0);
163 }
164 }
165 }
166 }
167 }
168 }
169 catch (Exception Ex)
170 {
171 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!\\n " + Ex.Message + "');", true);
172 }
173 finally
174 {
175 conn.Close();
176 dr1.Dispose();
177 cmd1.Dispose();
178 }
179 }
```

Listagem 16 – Código para pesquisa por descrição

A Listagem 17 apresenta a função usada para realizar a pesquisa por palavra-chave. A linha 142 verifica se algum filtro foi selecionado ou se palavras-chave foram selecionadas, caso contrário uma mensagem informando o usuário será mostrada. Uma variável chamada *sql* do tipo *string* foi usada para armazenar a instrução SQL responsável por recuperar os dados da pesquisa. Nas linhas seguintes, a variável *sql* receberá as instruções de acordo com cada filtro selecionado. Na linha 174 é realiza uma pesquisa de todas as palavras-chave cadastrada no banco. Um *array* de *string* é formado com as palavras que foram selecionadas para a pesquisa e é recuperado o identificador de todas as palavras selecionadas para a pesquisa para montar a instrução SQL. O mesmo procedimento é realizado para recuperar o identificador das palavras selecionadas que não devem possuir vínculo com a versão do artefato. As linhas seguintes do código efetuam a pesquisa.

```
139 protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
140 {
141 string sql = "";
142 if ((ddlTipoArtefato.SelectedValue != "") || (ddlTipoArtefato1.SelectedValue != "")
143 || (ddlFerramenta.SelectedValue != "") || (ddlTecnologia.SelectedValue != "")
144 ||(txtPalavraPossue.Text != "") || (txtPalavraNao.Text != ""))
145 {
146 sql = " SELECT VERSAO_ARTEFATO.IDARTEFATO, VERSAO_ARTEFATO.IDVERSAO,
VERSAO_ARTEFATO.DESCRICAO_VERSAO, VERSAO_ARTEFATO.NOME_VERSAO, " +
147 " ARTEFATO.IDARTEFATO, ARTEFATO.NOME_ARTEFATO FROM VERSAO_ARTEFATO "+
148 " INNER JOIN ARTEFATO ON ARTEFATO.IDARTEFATO = VERSAO_ARTEFATO.IDARTEFATO WHERE 1 = 1
";
149
150 if (ddlTipoArtefato.SelectedValue != "")
151 {
152 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue +
" OR " +
153 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue + " OR " +
154 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato.SelectedValue;
155 }
156 if (ddlTipoArtefato1.SelectedValue != "")
157 {
158 sql = sql + " AND VERSAO_ARTEFATO.IDTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue +
" OR " +
159 "VERSAO_ARTEFATO.IDSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue + " OR " +
160 "VERSAO_ARTEFATO.IDSUBSUBTIPOARTEFATO = " + ddlTipoArtefato1.SelectedValue;
161 }
162 if (ddlFerramenta.SelectedValue != "")
163 {
```

```
164 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlFerramenta.SelectedValue + "OR
" +
165 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlFerramenta.SelectedValue;
166 }
167 if (ddlTecnologia.SelectedValue != "")
168 {
169 sql = sql + " AND VERSAO_ARTEFATO.IDFERRAMENTA = " + ddlTecnologia.SelectedValue + "OR
" +
170 "VERSAO_ARTEFATO.IDTECNOLOGIA = " + ddlTecnologia.SelectedValue;
171 }
172 if ((txtPalavraNao.Text != "") || (txtPalavraPossue.Text != ""))
173 {
174 FbCommand cmd1 = new FbCommand("SELECT IDPALAVRA_CHAVE, NOME_PALAVRA FROM
PALAVRA_CHAVE", conn);
175 FbDataReader dr1 = null;
176 try
177 {
178 conn.Open();
179 dr1 = cmd1.ExecuteReader();
180 while (dr1.Read())
181 {
182 if (txtPalavraNao.Text != "")
183 {
184 string[] Npossue = txtPalavraNao.Text.Split(',');
185 for (int i = 0; i < Npossue.Length - 1; i++)
186 {
187 if (Npossue[i].ToString() == dr1.GetString(1))
188 {
189 sql = sql + " AND IDPALAVRA_PRIMARIA <> " + Convert.ToInt32(dr1.GetString(0)) + " OR
IDPALAVRA_SECUNDARIA <> " + Convert.ToInt32(dr1.GetString(0));
190 }
191 }
192 }
193 if (txtPalavraPossue.Text != "")
194 {
195 string[] possue = txtPalavraPossue.Text.Split(',');
196 for (int i = 0; i < possue.Length - 1; i++)
197 {
198 if (possue[i].ToString() == dr1.GetString(1))
199 {
200 sql = sql + " AND IDPALAVRA_PRIMARIA = " + Convert.ToInt32(dr1.GetString(0)) + " OR
IDPALAVRA_SECUNDARIA = " + Convert.ToInt32(dr1.GetString(0));
201 }
202 }
203 }
204 }
```

```
205 conn.Close();
206 cmd1.Dispose();
207 dr1.Dispose();
208 }
209 catch (Exception Ex)
210 {
211 ScriptManager.RegisterStartupScript(this, typeof(string), "javascript", "alert('Erro!"
+ Ex.Message + "');", true);
212 }
213 }
214
215
216 FbCommand cmd = new FbCommand(sql, conn);
217 FbDataReader dr = null;
218 try
219 {
220 conn.Open();
221 dr = cmd.ExecuteReader();
222 gridInsercao.DataSource = dr;
223 gridInsercao.DataBind();
224 }
225 catch (Exception Ex)
226 {
227 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!\\n " + Ex.Message + "');", true);
228 }
229 finally
230 {
231 conn.Close();
232 dr.Dispose();
233 cmd.Dispose();
234 }
235 }
236 else
237 {
238 ScriptManager.RegisterStartupScript(this, typeof(string), "alertScript",
"alert('Erro!!!\\n Preencha pelo menos um dos campos');", true);
239 }
240
241 }
```

Listagem 17 – Código para pesquisa por palavra-chave

A Figura 34 mostra o DataSet que foi criado para a geração de relatórios. Quatro tabelas foram usadas para compor o relatório de versões de artefatos que são usadas em cada projeto. A relação entre essas tabelas é apresentada na Figura 37.

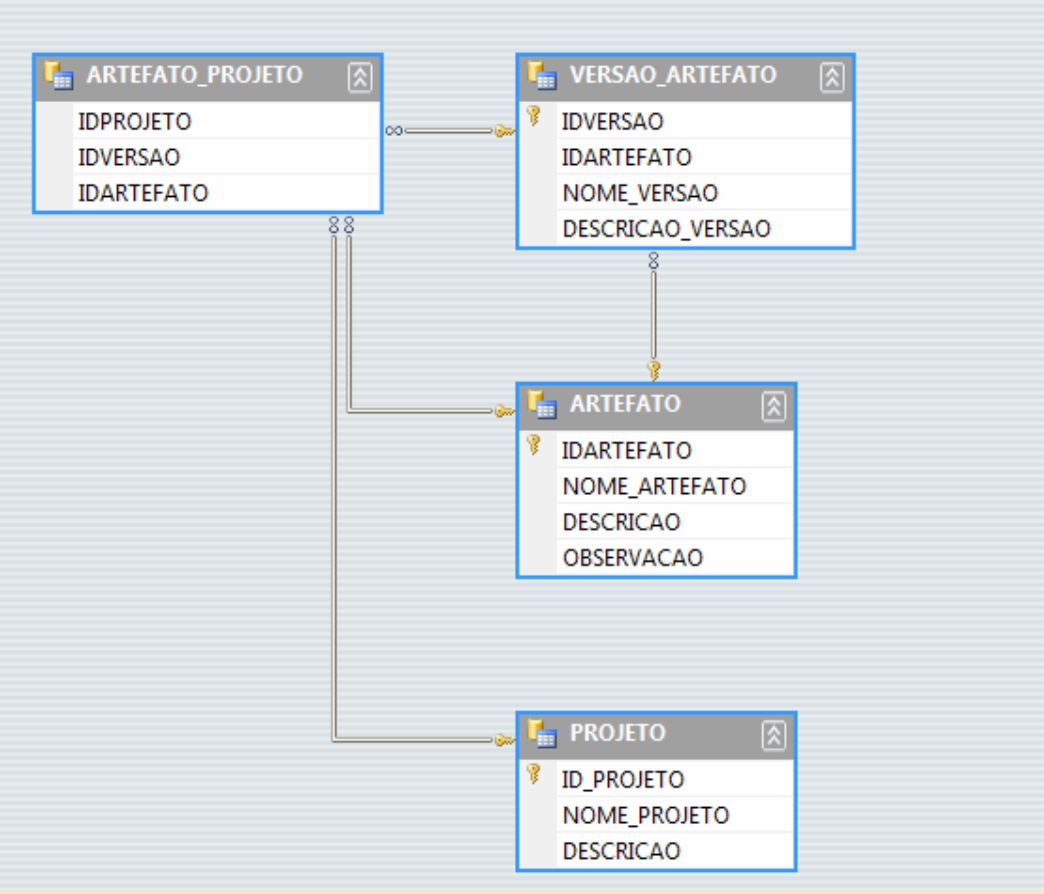

Figura 34 – Tabelas para o relatório de projeto, artefatos e versão de artefatos

A Listagem 18 mostra o código usado para carregar o DataSet com os dados do banco e mostrar no relatório. Na linha 24 é criado um novo objeto do tipo *ReportDocument,* depois é instanciado um DataSet do tipo que foi criado e mostrado na figura anterior. Linha 26 desabilita as restrições do DataTable. A linha 30 recebe o caminho do relatório que deve ser mostrado. O código das linhas 32 e 33, carregam os dados relativos a tabela de artefatos e projetos, nas linhas 35 e 36 a tabela projeto será carregada, as linhas 38 e 39 carregam a tabela de artefato e as linhas 41 e 42 carregam a tabela de versão de artefatos. Todas essas tabelas são carregadas e adicionadas ao DataSet. A linha 44 desse código é que carrega os dados para relatório que será mostrado.

```
24 ReportDocument crReportDocument = new ReportDocument();
25 DataSetArtefatoUsuario oDataset = new DataSetArtefatoUsuario();
26 oDataset.EnforceConstraints = false;
27 try
28 {
29     string strPathreport = Server.MapPath("RelArtefatoProjeto.rpt");
30     crReportDocument.Load(strPathreport);
31     conn.Open();
```

```
32      FbDataAdapter daArtVersaoProjeto = new FbDataAdapter("SELECT      IDPROJETO,
IDVERSAO, IDARTEFATO FROM ARTEFATO_PROJETO", conn);
33      daArtVersaoProjeto.Fill(oDataset, "ARTEFATO_PROJETO");
34
35      FbDataAdapter daProjeto = new FbDataAdapter("SELECT ID_PROJETO, NOME_PROJETO,
DESCRICAO FROM PROJETO", conn);
36      daProjeto.Fill(oDataset, "PROJETO");
37
38            FbDataAdapter  daArtefato  =  new  FbDataAdapter("SELECT  IDARTEFATO,
NOME_ARTEFATO, DESCRICAO, OBSERVACAO FROM ARTEFATO", conn);
39      daArtefato.Fill(oDataset, "ARTEFATO");
40
41        FbDataAdapter daVersao = new FbDataAdapter("SELECT IDVERSAO, IDARTEFATO,
NOME_VERSAO, DESCRICAO_VERSAO FROM VERSAO_ARTEFATO", conn);
42      daVersao.Fill(oDataset, "VERSAO_ARTEFATO");
43
44      crReportDocument.SetDataSource(oDataset);
45
46      CrystalReportViewer1.ReportSource = crReportDocument;
47  }
48  catch (Exception ex)
49  {
50      System.Diagnostics.Debug.WriteLine(ex.Message);
51  }
52  finally
53  {
54      conn.Dispose();
55      oDataset.Dispose();
56      Response.Clear();
57
58  }
```

Listagem 18 – Código para criação dos relatórios

4.5 DISCUSSÃO

O uso de palavras-chave facilita a implementação de mecanismos de busca, mas pode dificultar o cadastramento de artefatos no repositório. Se o conjunto dessas palavras é restrito, o artefato pode não ficar bem caracterizado. E um conjunto muito amplo pode conter palavras distintas para um mesmo conceito e dificultar a recuperação do artefato desejado ou que melhor atenda aos critérios de busca. Uma forma de minimizar é problema é pela associação de sinônimos entre palavras-chave.

No repositório implementado, os artefatos de acesso público podem ser localizados em buscas por qualquer usuário. Os artefatos de acesso privado são disponibilizados para o autor e para outros usuários ou grupos de usuários. Esse controle de acesso possibilita o uso do repositório por conjuntos de fábricas de software, como, por exemplo, uma incubadora de empresas ou as que pertençam a um arranjo produtivo local. E, também, para o desenvolvimento cooperativo ou distribuído de software.

O modelo proposto não determina como obrigatório o preenchimento de todos os campos para cadastramento de artefatos. Isso torna o sistema flexível e facilita o seu uso, inclusive por fábricas de software de pequeno porte, porque uma solução mais simples pode ser implementada. Contudo, ressalta-se que para maior efetividade na busca é necessário que os artefatos estejam bem caracterizados, com informações consistentes e completas.

5 CONCLUSÃO

O objetivo desse trabalho foi implementar um repositório de artefatos visando facilitar reuso e definir um mecanismo de busca baseado em palavras-chave e na descrição do artefato.

Considerando o levantamento bibliográfico realizado, percebe-se que existe uma carência de repositórios voltados para o reuso de artefatos de software como definido neste trabalho, pois os repositórios existentes são, normalmente, para componentes de software. A proposta deste trabalho fundamenta-se na necessidade verificada em fábricas de software, principalmente as de pequeno porte, em reaproveitar conhecimento, experiências, documentos e componentes de forma a agilizar os processos executados durante o ciclo de vida do software. Além de prover qualidade pelo uso de artefatos já utilizados em outros projetos e assim mais amplamente testados e possibilitar o compartilhamento de artefatos por fábricas de software.

No levantamento bibliográfico realizado trabalhos e propostas foram lidos no sentido de identificar quais seriam os atributos (ou metadados) a serem utilizados no armazenamento dos artefatos. Redolfi et al. (2005) apresentam uma revisão bastante interessante de diversos outros autores e agrupam os requisitos definidos por esses diversos autores em categorias. Tendo como base outros autores, inclusive mais recentes que o trabalho de Redolfi et al. (2005), e o interesse de atender fábricas de software de pequeno porte (e desta forma que o repositório fosse prático e simples de uso tanto para cadastramento dos artefatos quanto para a busca dos mesmos) outros agrupamentos e mesmo requisitos foram definidos como metadados para o armazenamento dos artefatos no repositório desenvolvido.

Assim, o objetivo de definir uma ferramenta que propiciasse reuso de artefatos por fábricas de software como pretendido por esse trabalho foi alcançado. O armazenamento dos diversos tipos de artefatos produzidos durante o ciclo de vida, como documentos de modelagem, pode contribuir para a profissionalização das atividades das fábricas de software e auxiliar na sua consolidação no mercado. A profissionalização está relacionada ao desenvolvimento de artefatos de acordo com padrões e modelos de qualidade. Para isso pode ser necessário que a fábrica de software tenha modelos de qualidade implementados ou ter seus processos definidos de acordo com esses modelos.

Um repositório e formas de busca como os apresentados neste trabalho podem ser vistos como um diferencial para fábricas de software de pequeno porte. Elas podem trabalhar cooperativamente e assim compartilhar artefatos. O compartilhamento é facilitado pelo repositório desenvolvido ser um sistema para *web*.

Uma das vantagens oferecidas por uma aplicação *web* é a redução do trabalho de atualização da aplicação, pois basta atualizá-la no servidor para que os usuários tenham acesso às mudanças. Dentre as desvantagens existentes em uma aplicação *web*, pode ser citada a falta de padronização nos navegadores, fazendo com que a aplicação tenha comportamentos diferentes dependendo do navegador usado. Em termos de interface, uma aplicação *web* tendo como base HTML para o desenvolvimento da interface pode não ser tão "visualmente interessante" como uma aplicação *desktop*, devido ao fato de não existirem componentes gráficos em HTML em diversidade tão ampla quanto para ambiente *desktop*.

Essa dificuldade tem sido resolvida ou pelo menos minimizada pelas aplicações denominadas *Rich Internet Applications* que visam prover interface para aplicações *web* semelhante às *desktop*, superando as limitações apresentadas pela HTML, tanto em termos de interface como de otimização de processamento. Com uso de Ajax, por exemplo, a página não precisa ser completamente recarregada a cada interação do usuário. Apenas as partes da tela que são alteradas serão atualizadas. O tráfego de rede é minimizado. Esse tipo de aplicação também pode compartilhar o processamento entre o cliente e o servidor. Melhorando, assim, o desempenho da aplicação.

A forma de classificação e de recuperação utilizada no repositório implementado pode representar uma solução para otimizar o reuso efetivo em repositórios de artefatos. Contudo, um mecanismo (algoritmo) de busca mais efetivo será implementado como trabalho complementar a esse. Esse mecanismo utilizará lógica *fuzzy*, dentre outros recursos. O usuário determinará o "grau" de importância que ele atribui a cada um dos atributos a serem utilizados no relacionamento de palavras-chave aos artefatos e na busca. É a pertinência da palavra-chave em relação ao referido artefato. Uma combinação de pertinência entre as palavras-chaves definidas no cadastro e na busca definirá será estabelecida.

E, assim, será possibilitada a recuperação de artefatos armazenados no repositório que não atendem a todos os critérios de busca estabelecidos em mesmo grau de importância no cadastro e na busca. São os artefatos que atendem parcialmente aos critérios de busca.

Além da implementação de requisitos, como trabalhos futuros, destaca-se a realização de testes comparativos entre as formas de busca implementadas. Para isso um volume considerável de artefatos deverá ser cadastrado na base de dados e os resultados da busca registrados para que possam ser analisados comparativamente.

REFERÊNCIAS

ASP.NET. **Asp.net.** Disponível em <http://www.asp.net/get-started>. Acesso em: 07 fev. 2012.

ASTAH. **Astah community**. Disponível em <http://astah.change-vision.com/en/product/astah-community.html>. Acesso em: 08 ago. 2012.

BARNES, Bruce H.; BOLLINGER Terry B. Making software reuse cost effective. **IEEE Software Magazine**, v. 8, p. 13-24, 1991.

BATISTA JUNIOR, Joinvile; DOMINGUES, Paulo E. Method for searching software components. **Natural Language. IEEE Latin America Transactions**, v. 8, n. 3, june 2010, p. 296-303.

BLAHA, Michael, JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. **Modelagem e projetos baseados em objetos com UML 2**. 2ª ed. Rio de Janeiro: Elsevier, 2006.

BORSOI, Beatriz T.; FÁVERO, Eliane M. de B.; ASCARI, Rúbia E. de O. S.; ARIATI, Adriana; BELAZI, Renato S. **Uma proposta de modelo conceitual para repositório de artefatos de software visando promover o reuso**. In: XX SEMINCO - Seminário de Computação, 2011, p. 6-17.

BRITO, Talles; RIBEIRO, Thiago; NÓBREGA Hugo; ELIAS, Glêdson. **Uma técnica de indexação de artefatos de software baseada em dados semi-estruturados**. In: III Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software (SBCARS 2009), 2009, p. 166-179.

CALDIERA, Gianluigi; BASILI, Victor R. Identifying and qualifying reusable software components. **The IEEE Computer**, v. 24, n. 2, 1991, p. 61–71.

CANTU, Carlos H. **Get to know Firebird in 2 minutes (Conheça o Firebird em dois minutos)**. Disponível em: <http://www.firebirdnews.org/docs/f>. Acesso em: 26 jan. 2012.

CRYSTAL. **Crystal Reports**. Disponível em: < http://www.crystalreports.com/products/software-products.asp>. Acesso em: 20 jun. 2012.

CASESTUDIO. **Toad data model**er. Disponível em: <http://www.casestudio.com/enu/default.aspx>. Acesso em: 29 jun. 2012.

DING, Li; FININ, Tim; JOSHI, Anupam, PAN, Rong; COST, R. Scott; PENG, Yun; Reddivari, Pavan; DOSHI, Vishal; SACHS, Joel. **Swoogle: a semantic web search and metadata engine**. In: 13th ACM Conference on Information and Knowledge Management, 2004, p. 652-659.

FIREBIRD. **Firebird**. Disponível em: <http://www.firebirdsql.org/>. Acesso em: 18 jul. 2012.

FRAKES, William B.; KANG, Kyo. Software reuse research: status and future. **IEEE Transaction Software Engineering**, v. 31, n. 7, p. 529-536, 2005.

FRAKES, William B.; POLE, Thomas P. An empirical study of representation methods for reusable software component. **IEEE Transactions on Software Engineering**, v. 20, n. 8, p. 617-630, 1994.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Design patterns. Elements of reusable object-oriented software**. Addison-Wesley, 1997.

GARCIA, Vinicius C.; LUCRÉDIO, Danile; DURÃO, Frederico A.; SANTOS, Eduardo C. R.; ALMEIDA, Eduardo S.; FORTES, Renata P. M.; MEIRA, Silvio R. L. **From specification to experimentation: a software component search engine architecture.** In: 9th International Symposium on Component-Based Software Engineering (CBSE 2006), 2006, p. 82-97.

GARRETT, James J. **Ajax: a new approach to web applications**, 2005. Disponível em: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>. Acesso em: 11 abr. 2012.

GIRARDI, Maria del Rosario; IBRAHIM, Bertrand. **Automatic indexing of software artifacts.** In: 3rd International Conference on Software Reuse, IEEE Computer Society Press, 1994, p. 24-32.

GREENFIELD, Jack; SHORT, Keith. **Software factories: assembling applications with patterns, models, frameworks, and tools**. Indianapolis Indiana: Wiley Publications Inc, 2004.

GROSZ, Georges. **Building information system requirements using generic structures**. In: 16th Computer Software & Applications Conference (COMPSAC 92), 1992, p. 200-205.

HENNINGER, Scott. Using iterative refinement to find reusable software. **IEEE Software**, v. 11, n. 5, p. 48-59, 1994.

HENNINGER, Scott. An evolutionary approach to constructing effective software reuse repositories. **ACM Transactions on Software Engineering and Methodology**, v. 6, n. 2, 1997, p. 111-140.

IBEXPERT. **IBExpert developer sudio**. Disponível em: <http://www.ibexpert.com/>. Acesso em: 14 ago. 2011.

IIS. **Internet information services**. Disponível em <http://www.iis.net/>. Acesso em: 07 fev. 2012.

JQUERY. **jQuery**. Disponível em: <http://jqueryui.com>. Acesso em: 22 ago. 2012.

JUSTO, José Luis B. **A repository to support requirement specifications reuse**. In: Information Systems Congress of New Zealand (ISCNZ'96), p. 53-62, 1996.

KOTONYA, Gerald; LOCK, Simon; MARIANI, John. Scrapheap software development: lessons from an experiment on opportunistic reuse. **IEEE Software**, v. 28, n. 2, p. 68-74, 2011.

MAAREK, Yoëlle S.; SMADJA, Frank A. **Full text indexing based on lexical relations, an application: software libraries**. In: Conference on Research and development in information retrieval (ACMSIGIR ’89), 1989, p. 198-206.

MAIDEN, Neil; SUTCLIFFE, Alistair. Exploiting reusable specifications through analogy. **Communications of the ACM**, v. 35, n. 4, p. 55-64, 1992.

MCILROY, Malcolm D. **Mass produced software components**. In: NATO Software Engineering Conference, Springer, p. 138-155, 1968.

MELO, Cláudia de Oliveira. **Classificação semi-automática de componentes Java**, Dissertação de Mestrado, Instituto de Matemática e Estatística da Universidade de São Paulo: São Paulo, 2007.

MICROSOFT. **Visual Studio**. Disponível em: <http://www.microsoft.com/visualstudio>. Acesso em: 20 ago. 2012.

MIKKONEN, Tommi; TAIVALSAARI, Antero. **Web applications – spaghetti code for the 21st century.** Sixth International Conference on Software Engineering Research, Management and Applications, 2008, p. 319-328.

MSDN. **What is Crystal Reports?.** Disponivel em: < http://msdn.microsoft.com/en-us/library/ms225593%28v=vs.80%29.aspx>. Acesso em: 23 de jul. 2012

NEIGHBOURS, James M. The draco approach to constructing software from reusable components. **IEEE Transactions on Software Engineering**, v. 10, n. 5, p. 564-574, 1984.

PARNAS, David L. On the design and development of program families. **IEEE Transaction On Software Engineering**, v. 2, n. 1, p. 1-9, 1976.

PETROV, Ilia; BUCHMANN, Alejandro. **Architecture of OMG MOF-based repository systems**. In: International Organization for Information Integration and Web-based Application and Services (iiWAS2008), 2008, p. 193-200.

PODGURSKI, Andy; PIERCE, Lynn. Retrieving reusable software by sampling behavior. **ACM Transaction on Software Engineering and Methodology**, v. 2, n. 3, p. 286-303, 1993.

PREE, Wolfgang. **Design Patterns for oriented object software development**. New York: Addison-Wesley, 1994.

PRESSMAN, Roger. **Engenharia de software.** 5ª ed. 2002. Rio de Janeiro: McGrawHill.

PRIETO-DIAZ, Ruben. **Classification of reusable modules in software reusability: concepts and models**. In: Biggerstaff, T. J. and Perlis, A. J. (Editors). Addison-Wesley Pub. Co.: New York, NY, p. 99-123, 1989.

PRIETO-DIAZ, Ruben. Implementing faceted classification for software reuse.

Communications of the ACM, v. 34, n. 5, p. 89-97, 1991.

PRIETO-DÍAZ, Ruben; FREEMAN, Peter. Classifying Software for Reusability. **IEEE Software**, v. 4, n. 1, p. 6-16, 1987.

PRIETO-DIAZ, Ruben. **Classification of reusable modules in software reusability: concepts and models**. In: Biggerstaff, T. J. and Perlis, A. J. (Editors). Addison-Wesley Pub. Co.: New York, NY, 1989, p. 99-123

REDOLFI, Giliane; SPAGNOLI, Luciana de A.; HEMESATH, Peter; BASTOS, Ricardo M.; RIBEIRO, Marcelo B.; CRISTAL, Mauricio; ESPINDOLA, Anete P. **A reference model for reusable components description**. In 38th Hawaii International Conference on System Sciences, p. 282-291, 2005.

RUFINO, Airtiane. **Folksonomia: novos desafios do profissional da informação frente às novas possilibidades de organização de conteúdos**. XXXII Encontro Nacional de Estudantes de Biblioteconomia, Documentação, Ciência e Gestão da Informação, 2009, p. 1-12.

SALTON, Gerard. **Automatic text processing: the transformation, analysis, and retrieval of information by computer**. Readings, Massachusetts: Addison-Wesley Pub, 1989.

SANT'ANNA, Mauro **C#: A nova linguagem da arquitetura .NET**. Disponível em: <http://www.portaldaprogramacao.com/artigos2.asp?n=103>. Acesso em: 20 ago. 2012.

SHIVA, Sajjan G.; SHALA, Lubna A. **Software reuse: research and practice**. In: 4th IEEE Conference Information Technology (ITNG 07), IEEE CS Press, p. 603-609, 2007.

SILVA, Mauricio S. **JQuery. A biblioteca do programador javascript**. Novatec, 2010.

SOMMERVILLE, Ian. **Engenharia de software**. 8ª ed. São Paulo: Pearson Addison Wesley. 2007.

SIVASHANMUGAM, Kaarthik; VERMA, Kunal; SHETH, Amit; MILLER, John. **Adding semantics to web services standards**. In: International Conference on Web Services, 2003, p. 395-401.

SUTCLIFFE, Alistair; MAIDEN, Neil. **Domain modeling for reuse**. In: 3rd International Conference on Software Reuse, p. 169-177, 1994.

TAVARES, Roberta P.; SANT'ANNA, Nilson. Uma abordagem para definição de um repositório de artefatos dinâmico utilizando DOM (Dynamic Object Model) em um ambiente de engenharia de software para controle de satélites. **Revista Científica da FAI.** Santa Rita do Sapucaí, MG, v.6, n.1, p. 25-36, 2006.

VANDERLEI, Taciana A.; DURÃO, Frederico A.; GARCIA, Vinicius C.; ALMEIDA, Eduardo S.; MEIRA, Silvio R. L. **A Cooperative classification mechanism for search and retrieval software components**. In: 22th Annual ACM Symposium on Applied Computing (ACM SAC), 2007, p. 866-871.

VANDERLEI, Taciana A.; GARCIA, Vinicius C.; ALMEIDA, Eduardo S.; MEIRA, Silvio R. L. **Folksonomy in a software component search engine – cooperative classification through shared metadata**. In: XX Simpósio Brasileiro de Engenharia de Software (SBES), 2006, p. 1-13.

WHITE, Stephen I. **Business process modeling notation**. v. 1.0, Business Process Management Initiative, 2004.