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

MARCOS ANTÔNIO TANCON
RAFAEL LUCINI

SISTEMA PARA CONTROLE DE DESENVOLVIMENTO DE REQUISITOS DE SOFTWARE

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2014

MARCOS ANTÔNIO TANCON
RAFAEL LUCINI

SISTEMA PARA CONTROLE DE DESENVOLVIMENTO DE REQUISITOS DE SOFTWARE

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

Orientador: Profa. Beatriz Terezinha Borsoi

PATO BRANCO
2014

ATA Nº: **260**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DOS ALUNOS **RAFAEL LUCINI e MARCOS ANTONIO TANCON.**

Às 14:30 hrs do dia 19 de dezembro de 2014, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Eliane Maria de Bortoli Fávero (Convidada) e Robison Cris Brito (Convidado), para avaliar o Trabalho de Diplomação do aluno Rafael Lucini, matrícula 1030809 e do aluno Marcos Antonio Tancon, matrícula 1066900, sob o título **Sistema para Controle de Desenvolvimento de Requisitos de Software**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação os candidatos foram entrevistados pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 15:05 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Convidada

Prof. Robison Cris Brito, M.Sc.
Convidado

Profa. Eliane Maria de Bortoli Fávero, M.Sc
Coordenadora do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

LUCINI, Rafael; TANCON, Marcos Antônio. Sistema para controle de desenvolvimento de requisitos de software. 2014. 68f. Trabalho de Conclusão de curso do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014.

O processo de desenvolvimento e controle de requisitos de software é de extrema importância no desenvolvimento geral de um sistema e dos próprios requisitos de forma individual. Para garantir confiabilidade é necessário um sistema completo para gerenciar a maneira que os requisitos são desenvolvidos, controlando suas diversas iterações e testes, e também padrões de tempo estimados para sua conclusão. Sistemas como estes podem ser aplicados para auxiliar no desenvolvimento desde software simples, até os mais complexos e extensos, dos quais a utilização de controles é essencial. Assim, viu-se oportuno nesse trabalho realizar o desenvolvimento de um software para auxiliar no controle da implementação de requisitos de software.

Palavras-chave: Requisitos de software. Rastreabilidade de requisitos. Gerenciamento de requisitos de software.

ABSTRACT

LUCINI, Rafael; TANCON, Marcos Antônio. System to manager software development requirements. 2014. 68f. Trabalho de Conclusão de curso do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. Pato Branco, 2014.

The development process and control software requirements is of utmost importance in the overall development of a system and own individually requirements. To ensure reliability of the complete system is required to manage the way the requirements are developed by controlling its various iterations and testing, and also estimated time standards for its completion. Systems like these can be applied to assist in developing software from simple to the most complex and extensive, including the use of controls is essential. Thus, it is appropriate make the development of software to help control the implementation of software requirements.

Palavras-chave: Requisitos de software. Rastreabilidade de requisitos. Gerenciamento de requisitos de software.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

DER	Diagrama de Entidade e Relacionamento
MER	Modelo Entidade e Relacionamento
OMG	*Object Management Group*
OMT	*Object Modeling Techniquee*
OOSE	*Objected Oriented Software Engineering*
RF	Requisito Funcional
RNF	Requisito Não Funcional
UML	*Unified Modeling Language*
UP	*Unified Process*

SUMÁRIO

1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais com uma visão geral do assunto que envolve o sistema que é implementado como resultado deste trabalho, os objetivos e a justificativa. Por fim está a organização do texto por meio da apresentação dos seus capítulos.

1.1 CONSIDERAÇÕES INICIAIS

Uma empresa de desenvolvimento de software tem como principal finalidade implementar aplicativos e sistemas computacionais. O processo de desenvolvimento é composto por uma série de etapas que são necessárias para que o resultado pretendido seja alcançado. Independentemente do modelo utilizado para organizar essas etapas, como, por exemplo, o processo unificado (KRUCHTEN, 2004) ou o modelo sequencial linear (PRESSMAN, 2008), as atividades envolvidas no desenvolvimento de software estão relacionadas a requisitos, análise, implementação e testes.

De maneira geral, o principal produto obtido com a realização das atividades de desenvolvimento de software é o sistema ou aplicativo em si. Um sistema é, normalmente, composto por diversos requisitos que precisam ser atendidos para que o produto final esteja de acordo com os interesses e as necessidades dos usuários e que o sistema possua a qualidade esperada, além de o projeto atender o orçamento e o prazo estimados.

Uma versão de um software é obtida por meio do desenvolvimento de requisitos, que são implementados e testados individualmente e em conjunto. Seja para implementar funcionalidades novas ou ajustes e correções realizados pela empresa ou solicitados por clientes, o processo é semelhante: os requisitos a serem implementados (novos, ajustes ou correções) são modelados por meio de atividades de análise e após implementados passam por verificações e testes.

Os requisitos a serem implementados são registrados e servem de base para calcular o tempo e os recursos necessários para realizar determinada atividade do processo de desenvolvimento, seja ela referente a ajustes em software existente ou

implementação de funcionalidades novas. Para calcular o tempo e os recursos necessários para execução de cada atividade até sua conclusão, leva-se em conta o processo de desenvolvimento e os testes envolvidos. Esse tempo calculado pode ser utilizado para realização de estimativas de prazos para realização de atividades semelhantes e determinadas modalidades de requisitos. Também por meio do registro (cadastro) é possível realizar um controle dos requisitos que se encontram em fila para implementação. Essa fila pode ser ordenada utilizando o tipo do requisitos como critério, priorizando correções de sistema ou a quantidade de clientes associados ao requisito, por exemplo. Esse controle também pode ser utilizado para consulta das prioridades associadas a cada requisito, que definem a ordem de implementação considerando todas as solicitações existentes no momento.

Controlar e acompanhar as solicitações para ajustes ou melhorias em um software demanda bastante cuidado e atenção e para isso a utilização de um software específico pode ser bastante proveitosa. O sistema desenvolvido como resultado da realização deste trabalho possibilitará o acompanhamento de cada requisito associado a um sistema em desenvolvimento. Sendo, assim, possível gerenciar a implementação do mesmo, mantendo o cliente informado e fornecendo dados para a gestão do processo de implementação e das atividades associadas.

1.2 OBJETIVOS

O objetivo geral apresenta o resultado principal da realização deste trabalho e os objetivos específicos complementam o objetivo geral.

1.2.1 Objetivo Geral

Desenvolver um sistema para gerenciamento de implementação de requisitos de software.

1.2.2 Objetivos Específicos

- Gerenciar requisitos no processo de implementação de sistemas que é realizado em uma empresa de desenvolvimento de software.
- Auxiliar no controle de prazos para desenvolvimento de software, fornecendo dados para o cálculo de estimativas de prazo na realização de cada etapa do processo de software.

1.3 JUSTIFICATIVA

Com a expansão do mercado de desenvolvimento de software e clientes cada vez mais exigentes, surge a necessidade de se trabalhar com prazos e metas bem estabelecidos nas empresas de software, tendo, assim, maior controle de qualidade do produto desenvolvido e maior satisfação por parte dos clientes. O gerenciamento dos requisitos de um software em implementação ou de ajuste de requisitos já implementos tem um papel relevante na execução do processo e na qualidade do produto. O cliente espera que as suas necessidades sejam prontamente atendidas. A empresa, por outro lado, precisa realizar a implementação de forma a otimizar a sua equipe e os recursos envolvidos e gerenciar as outras demandas.

A opção por implementar um aplicativo de controle e gerenciamento de requisitos ocorreu devido a real necessidade das empresas do ramo de desenvolvimento de software. Essas empresas precisam aplicar métodos de controle durante o processo de desenvolvimento de sistemas. Assim, um software que auxilie no controle e gerenciamento de requisitos pode auxiliar em definir prioridades de requisitos e manter o histórico das implementações. Esse histórico pode ser utilizado em estimativas futuras e para levantamento do trabalho realizado para cada produto e/ou cliente.

1.4 ESTRUTURA DO TRABALHO

No Capítulo 2 é apresentado o referencial teórico sobre análise e projeto de sistemas orientados a objetos, *Unified Modeling Language,* análise estruturada e processo de desenvolvimento de software.

No Capítulo 3 são apresentados os materiais, que se referem às tecnologias e ferramentas utilizadas no desenvolvimento do trabalho e também o método utilizado.

No Capítulo 4 é apresentado o resultado final da análise desenvolvida, descrevendo os requisitos identificados e a modelagem desenvolvida.

No último Capitulo é apresentada a conclusão e suas considerações, finalizando com as referências bibliográficas que fundamentaram este trabalho de estágio supervisionado.

2 FUNDAMENTAÇÃO TEÓRICA

Esse trabalho foi desenvolvido com base nos conceitos de análise e modelagem de sistemas e exemplifica-os por meio dos diagramas da *Unified Modeling Language* (UML), a Linguagem de Modelagem Unificada.

Foi realizado um estudo com base na área de atuação do sistema, sobre o funcionamento de todo o processo de gerência de requisitos e também baseado em ferramentas e métodos utilizados para esses controles. Com base nisso, buscou-se o referencial teórico sobre conceitos e informações necessários para auxiliar no desenvolvimento deste trabalho.

2.1 ANÁLISE E PROJETO DE SISTEMAS ORIENTADOS A OBJETOS

A análise de sistemas é um conjunto de métodos utilizados para a solução de diversificados problemas. Por meio dela, utilizando levantamento de requisitos e informações úteis, se dá o molde para a criação e implementação de um software.

Especificando, Wazlawick (2011) explica que nas atividades de análise e projeto, a fase de concepção vai exigir do analista uma visão inicial e geral do sistema a ser desenvolvido. Essa visão pode ser obtida a partir de entrevistas, documentos e sistemas. Para apoiar a modelagem dessa visão geral pode-se usar diagramas de máquina de estados ou diagramas de atividades da UML, que correspondem, nessa fase, à modelagem de negócios. A partir dessa compreensão do negócio pode-se analisar de forma mais aprofundada cada uma das atividades ou estados para obter os requisitos funcionais e não funcionais do sistema.

O mesmo autor também cita que a elaboração dos modelos pode ser refinada e mais informações agregadas a partir das descobertas feitas durante a expansão dos casos de uso.

2.1.1 Processo Unificado de Desenvolvimento de Software

O desenvolvimento desse trabalho tomou como base a obra do autor Raul Sidnei Wazlawick (WAZLAWICK, 2011), que aborda o Processo Unificado (UP) de desenvolvimento de software e também as fases de concepção e elaboração do sistema.

"A fase de concepção incorpora o estudo de viabilidade, o levantamento dos requisitos e uma parte de sua análise. A fase de elaboração incorpora o detalhamento da análise de requisitos, a modelagem de domínio e o projeto." (WAZLAWICK, 2011, p. 5)

De acordo com Wazlawick (2011) a fase de concepção é a primeira fase do processo unificado, na qual se procura levantar os requisitos e compreender o sistema de forma abrangente.

Como já destacado, a concepção consiste nas atividades de levantamento e organização de requisitos. Na etapa de levantamento, ocorre a busca de informações possíveis sobre as funções que o sistema deve executar, e suas restrições de operação. Por meio dessa etapa da concepção será extraído o documento de requisitos.

Na etapa de análise de requisitos, eles serão estruturados e detalhados de forma que possam ser utilizados na fase futura de elaboração, e desenvolvimento de casos de uso e classes.

2.2 UML - *UNIFIED MODELING LANGUAGE*

A UML, segundo Booch (2000), é uma linguagem padrão para elaboração da estrutura de projetos de software, podendo ser empregada para visualização, especificação, construção e documentação dos artefatos de sistemas de software. É o resultado da unificação da linguagem de modelagem de objetos de 3 métodos líderes do mercado: Booch, *Object Modeling Technique* (OMT) e *Objected Oriented Software Engineering* (OOSE). Em 1997, a UML v 1.1 foi adotada pela OMG *(Object*

Management Group) e desde então tornou-se o padrão da indústria de software para a modelagem de objetos e componentes (IWEB, 2003).

De acordo com a definição de Wazlawick (2011), a UML é constantemente revisada, e correntemente, possui três famílias de diagramas:

- Diagramas Estruturais: compreendendo diagramas de pacotes, classes, objetos, estrutura composta, componentes e distribuição.
- Diagramas Comportamentais: compreendendo os diagramas de casos de uso, atividades e máquina de estados.
- Diagramas de Interação: compreendendo os diagramas de comunicação, sequência, tempo e visão geral de interação.

2.2.1 Diagrama de Casos de Uso

Booch ressalta a importância dos diagramas de casos de uso:

> "Os diagramas de casos de uso são importantes para visualizar, especificar, e documentar o comportamento de um elemento. Esses diagramas fazem com que sistemas, subsistemas e classes fiquem acessíveis e compreensíveis, por apresentarem uma visão externa sobre como esses elementos podem ser utilizados no contexto". Booch (2000, p.241).

O diagrama de casos de uso, exemplificado na Figura 1, é a modelagem do conjunto de funcionalidades básicas de um software. O diagrama tem como objetivo a visualização das funcionalidades sem detalhá-las. Essa visão proporciona suporte, principalmente, para o comportamento de um sistema – os serviços externamente visíveis que o sistema fornece no contexto do seu ambiente (BOOCH, 2000, p. 244).

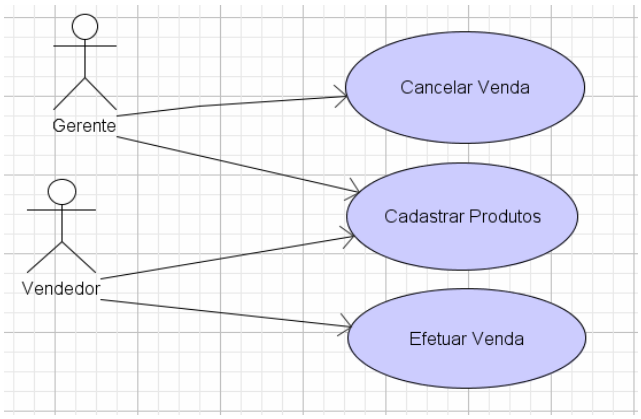

Figura 1 - Exemplo de diagrama de casos de uso

Os casos de uso e os atores são as peças fundamentais de um diagrama de caso de uso. O primeiro especifica de forma genérica, o comportamento básico do sistema, enquanto o segundo define os agentes externos que interagem com o mesmo.

A seguir as possíveis associações entre os elementos de um diagrama de caso de uso:

a) Entre Ator e Caso de Uso

Associações entre Atores e Casos de Uso representam a interação entre pessoas ou entidades externas com uma ou mais funcionalidades do sistema. Os dois elementos devem possuir ao menos uma relação, não deve haver ator ou caso de uso isolado no diagrama. Mais de um ator pode estar associado a um mesmo caso de uso, conforme a Figura 2.

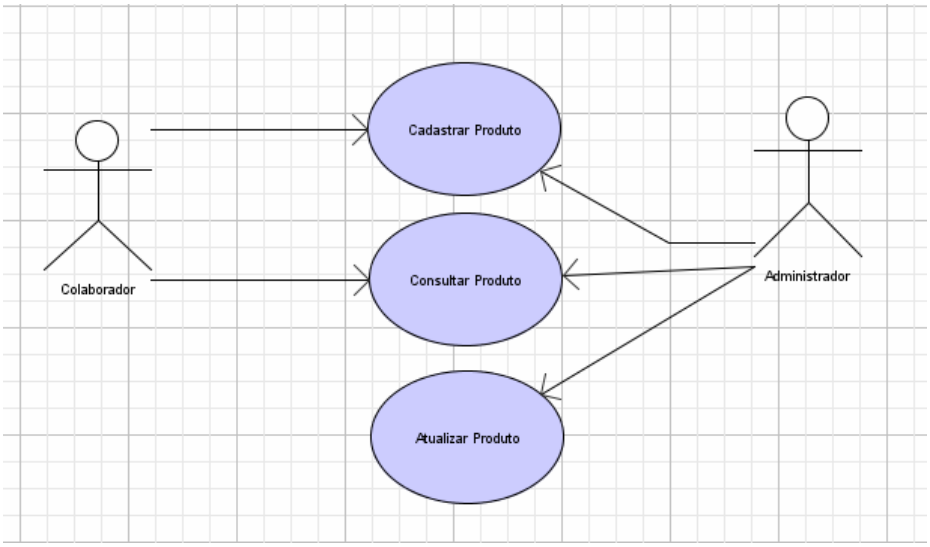

Figura 2 - Mais de um ator associado a um caso de uso

b) Entre Atores

Um Ator A está ligado a um Ator B que possui seus próprios casos de uso, assim os casos de uso de B são também casos de uso de A e A pode ter seus próprios casos de uso.

c) Entre Casos de Uso

Booch se referem ao relacionamento de generalização entre casos de uso.

> "Os casos de uso também podem ser organizados pela especificação de relacionamentos de generalização, inclusão e extensão, existentes entre eles. Você aplica esses relacionamentos com a finalidade de fatorar o comportamento comum (obtendo esse comportamento a partir de outros casos de uso que ele incluiu) e de fatorar variantes (obtendo esse comportamento em outros casos de uso que o estendem)", Booch (2000, p.35).

A seguir são exemplificados os três tipos de associação entre casos de uso:

- Associação de Extensão:

Passos destaca que:

> "Os casos de uso de extensão (*extends*) são usados para modelar o comportamento condicional e opcional dos casos de uso. Dito de outra forma, os casos de uso de extensão são usados para modelar serviços assíncronos ou de interrupção que o ator pode utilizar. Também são usados quando você está escrevendo adendos a documentos de requisitos fechados. Ou seja, quando, por algum motivo, é indesejável modificar o caso de uso base" (PASSOS, 2008, p.1).

A Figura 3 apresenta um exemplo de caso de uso estendido.

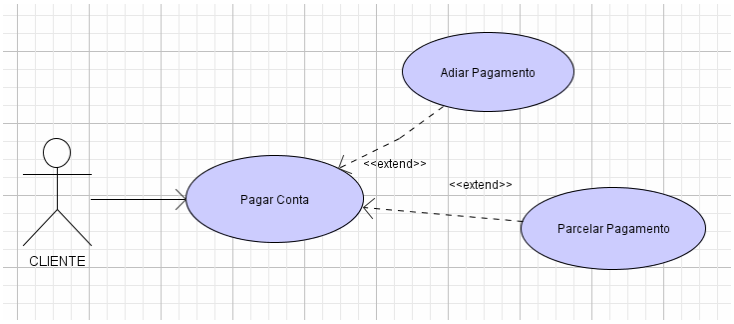

Figura 3 - Casos de Uso Estendidos

- Associação de Inclusão:

São usadas para melhor compreensão dos casos de uso, evitar repetições ou fragmentar casos de uso demasiado extensos. Ao contrário da extensão o caso de uso é contínuo e sempre será executado.

A Figura 4 apresenta um exemplo de inclusão.

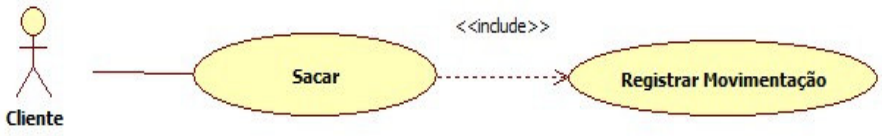

Figura 4 – Associação de inclusão

- Associação de Generalização:

A generalização é usada quando um caso de uso de base possui novos cenários que incluem ou sobrescrevem-no. A Figura 5 apresenta um exemplo de generalização entre casos de uso.

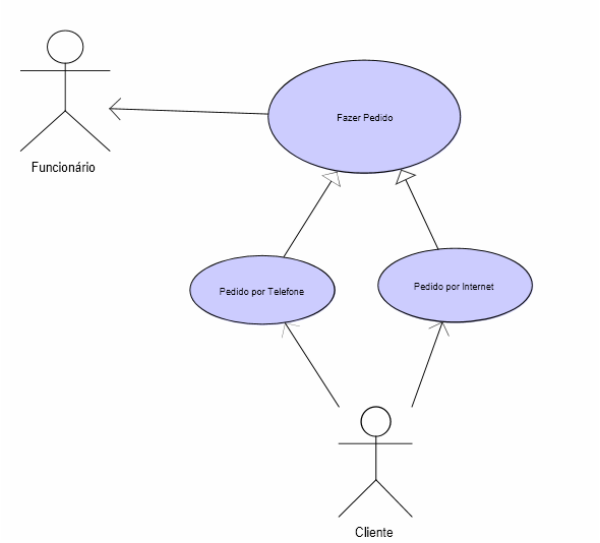

Figura 5 - Exemplo de generalização de casos de uso

2.2.2 Diagrama de Sequência

A UML possui um diagrama que pode ser útil para representar a sequência dos eventos do sistema em um cenário de um caso de uso. O diagrama de sequência tem como elementos, *instâncias de atores,* representados por figuras humanas esquematizadas, e instâncias que representam elementos do sistema (WAZLAWICK, 2011).

As mensagens retornadas pelo sistema são tracejadas porque o sistema apenas reage aos atores, e a mensagem tracejada representa então esse retorno de informação a partir de um estímulo provocado por um dos atores. Da mesma forma, a numeração das mensagens pode ser diferente da numeração dos casos de uso, visto que os retornos são subordinados à mensagem original. Assim, se o caso de uso numera os elementos como 1, 2, 3, 4..., o diagrama de sequência poderá ter os passos equivalentes numerados como 1, 1.1, 2, 2.2... (Figura 6).

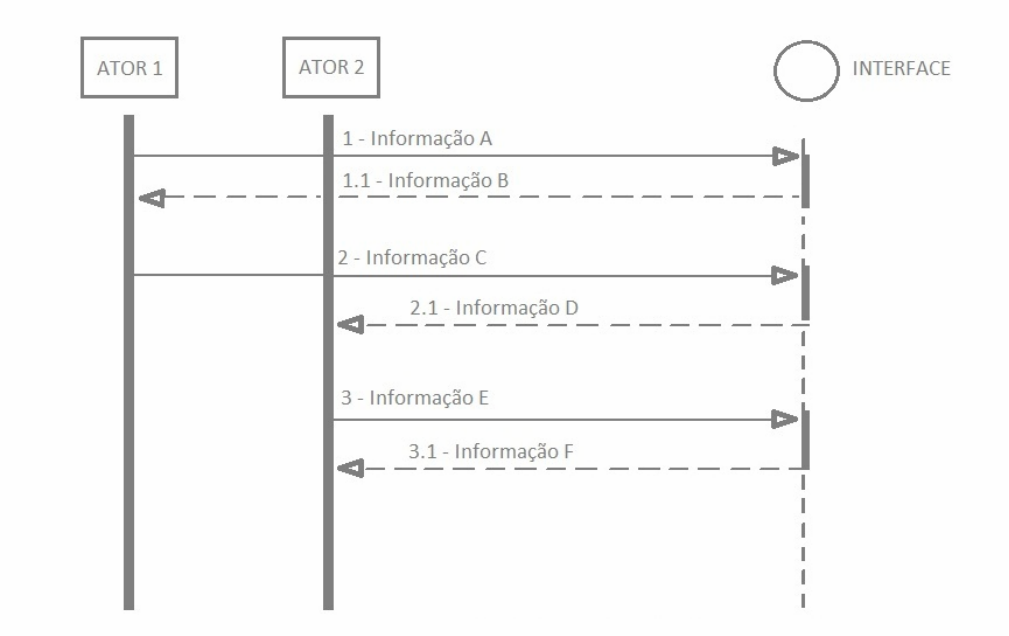

Figura 6 - Diagrama de Sequência de Sistema

O diagrama de sequência de sistema é uma forma de sistematizar o caso de uso expandido e, assim, refiná-lo para obter mais detalhes sobre o funcionamento do sistema.

A representação do caso de uso em um diagrama de sequência de sistema é feita em duas etapas:

a) Representação dos passos do caso de uso como troca de informações entre os atores e a interface do sistema;
b) Representação de operações e consultas de sistema como troca de mensagens entre a interface e a controladora-fechada da camada de domínio do sistema.

A primeira etapa é simples: a cada passo identificado com [IN] equivale a um envio de informação a de um ator para a interface do sistema, e a cada passo [OUT] equivale a um envio de informação do sistema para um ator (Figura 7).

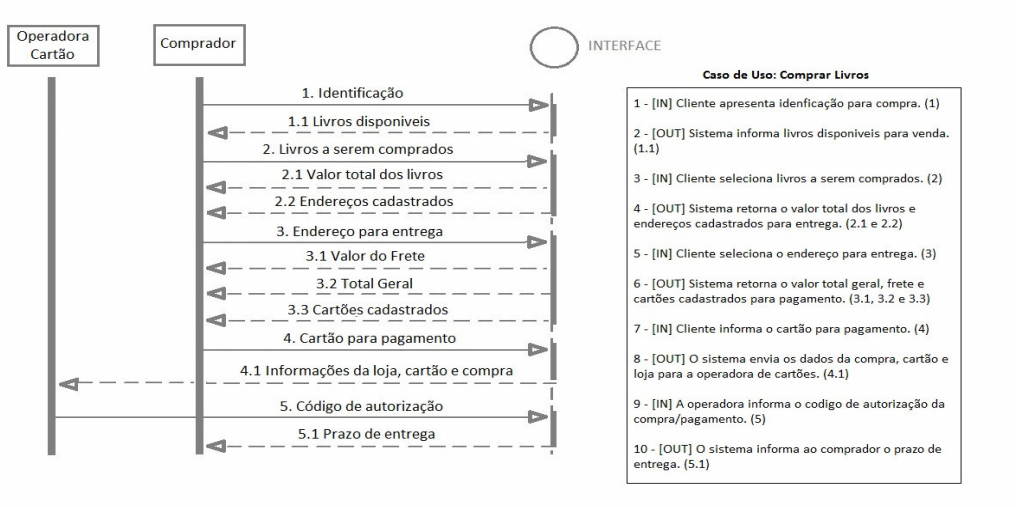

Figura 7 - Caso de uso representado no diagrama de sequência

Ao sistematizar os passos do caso de uso como envios de informação de atores para o sistema e vice-versa, o analista poderá, entre outras coisas, se dar conta de informações faltantes no caso de uso.

2.2.3 Diagrama de Classes

Os diagramas de classes são os diagramas encontrados com maior frequência na modelagem de sistemas orientados a objetos. Um diagrama de classes mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. Graficamente, são uma coleção de vértices e arcos (Booch, 2000).

A utilização desse modelo de diagrama tem ligação direta com a visão definida para o sistema no projeto, definindo vocabulário, colaborações e esquemas. Também é um diagrama essencial para visualização, especificação e documentação de modelos estruturais. A utilização dos diagramas de classe se expande também como base para outros diagramas relacionados, como componentes e implantação.

Pode-se identificar facilmente nos diagramas as classes, contendo seus devidos atributos, variados tipos e quantidades de acordo com a classe tratada e operações realizadas através da mesma. Também é possível a observação de ligações variadas entre classes, com as definições de suas cardinalidades (Figura 8).

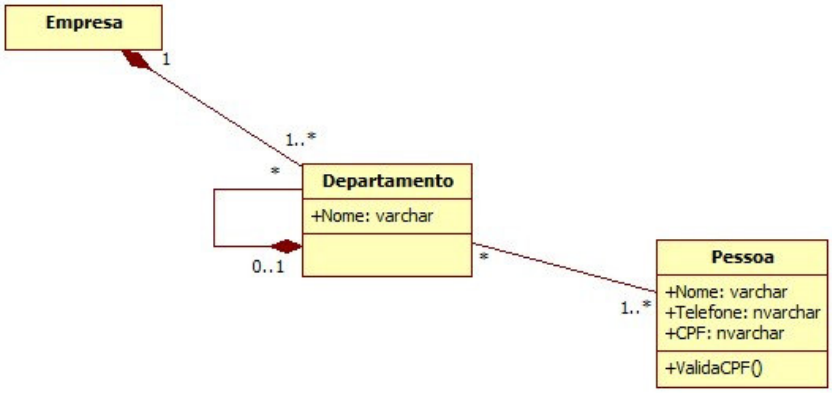

Figura 8 - Diagrama de classes

Para representar o fato de que objetos podem se relacionar uns com os outros, utiliza-se associações. Uma associação representa relacionamentos (ligações) que são formados entre objetos durante a execução do sistema. Embora as associações sejam representadas entre classes do diagrama, tais associações representam ligações possíveis entre os objetos das classes envolvidas (BEZERRA, 2006).

Na UML, associações são representadas por uma linha que liga as classes cujos objetos se relacionam:

Figura 9 - Exemplo de associação cliente e produto

Geralmente, as multiplicidades são definidas com um limite inferior e um limite superior, como 2..4 para jogadores de canastra. O limite inferior pode ser qualquer número positivo ou zero; o limite superior é qualquer número positivo ou “*” (para ilimitado). Se os limites inferior e superior forem os mesmos, pode-se usar um único número; assim 1 é equivalente a 1..1. Como se trata de um caso comum, * é a abreviatura de 0..* (FOWLER, 2005).

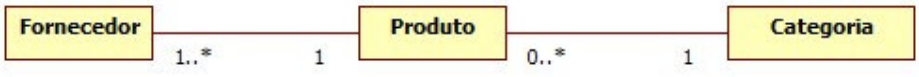

Figura 10 - Exemplo de cardinalidades nos diagrama de classes

No diagrama da Figura 10, está exemplificada a multiplicidade por meio de fornecedor, produto e categoria. Na representação, um ou vários fornecedores podem fornecer um produto. A categoria exibida nesse caso pode ser atribuída para um produto ou vários, ou não ser atribuída caso não exista um produto para associação. O Quadro 1 apresenta a simbologia de cardinalidades da UML.

Nome	**Simbologia na UML**
Apenas Um	1..1 (ou 1)
Zero ou Muitos	0..* (ou *)
Um ou Muitos	1..*
Zero ou Um	0..1
Intervalo Específico	$1_i..1_s$

Quadro 1 - Simbologia de cardinalidades UML

Um diagrama de classes compartilha as mesmas propriedades dos outros diagramas – um nome e um conteúdo gráfico que são uma projeção em um modelo. O que diferencia os diagramas de classe dos outros tipos de diagramas é o seu conteúdo particular.

Os diagramas de classe costumam conter os seguintes itens (BOOCH, 2000, p.109):

- Classes: descrições de conjuntos de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. Uma classe implementa uma ou mais interfaces.
- Interfaces: coleção de operações utilizadas para especificar um serviço de uma classe ou de um componente. As interfaces são empregadas na visualização, especificação, construção e documentação da coesão interna do sistema.
- Relacionamentos de dependência, generalização e associação.

2.3 ANÁLISE ESTRUTURADA

A análise é uma fase crítica no desenvolvimento de sistemas e programas de software porque afeta as fases de desenvolvimento seguintes. Ela é difícil em função dos problemas de comunicação, das mudanças nos requisitos dos sistemas, e das técnicas inadequadas de avaliação. Não é fácil descrever os requisitos do sistema em uma forma precisa. A linguagem do usuário e a linguagem do responsável pelo desenvolvimento são tão diferentes que torna complicada uma comunicação eficaz. Os requisitos, no entanto, apresentam um alvo móvel que continua a modificar-se por todo o desenvolvimento do sistema e por todo seu ciclo de vida.

A análise estruturada tem como objetivo resolver essas dificuldades fornecendo uma abordagem sistemática, etapa por etapa, para desenvolver a análise e produzir uma especificação de sistema nova e melhorada. Para conseguir este objetivo, a análise estruturada centraliza-se em uma comunicação clara e concisa (SME, 2014). Dentre os modelos utilizados pela análise estruturada, está o modelo de entidade e relacionamento, empregado nesse trabalho.

2.3.1 Diagrama e Modelo de Entidade e Relacionamento (DER e MER)

Segundo Martins (2007), em março de 1976, Peter P. Chen publicou um trabalho intitulado "*The Entity-Relationship Model*" no qual definia uma possível abordagem para o processo de modelagem de dados. Esse trabalho, após sua divulgação e ampla aceitação, passou a ser considerado como um referencial definitivo para o processo de modelagem de dados evoluído com o passar dos anos para uma abordagem mais próxima do ambiente de orientação de objeto, abordagem esse que torna a técnica mais rica e, portanto, aplicável a novas finalidades.

Apesar de ter recebido algumas outras representações e abordagens diferentes por alguns outros estudiosos, o modelo entidade e relacionamento acabou se tornando bastante usado. O modelo relacional estabeleceu-se como o primeiro

modelo de dados para aplicações comerciais, desenvolvido para facilitar o projeto de banco de dados relacionais permitindo a representação de todas as estruturas dos mesmos e um melhor diálogo entre os profissionais envolvidos no projeto.

Os diagramas de entidade e relacionamento representam os relacionamentos entre objetos de dados e conduzem a modelagem de dados.

2.4 PROCESSO DE DESENVOLVIMENTO DE SOFTWARE

O processo de desenvolvimento de software é um conjunto de atividades que tem como objetivo a organização, definição, desenvolvimento, teste e manutenção de um software. Há diversos processos de desenvolvimento de software, a seguir algumas atividades em comum entre esses processos que buscam a padronização das atividades em prol da qualidade do software:

a) Diagnóstico das Solicitações

Esta etapa tem como finalidade o registro das solicitações feitas e o detalhamento da viabilidade e/ou inviabilidade das mesmas. Funciona como um “pente fino” para a construção e análise dos requisitos, nela são eliminadas possíveis funcionalidades já existentes ou que não suprem a necessidade do usuário.

b) Análise de Requisitos

Nessa etapa é realizado o estudo detalhado da solicitação pelo desenvolvedor e/ou pessoa responsável pela tarefa, usando-se das necessidades do cliente para descrever uma solução para o problema. Valida-se as reais necessidades do cliente e verifica-se se os modelos construídos estão de acordo com os requisitos.

c) Projeto

Um conjunto de requisitos pré-selecionados, com prioridades definidas a partir da complexidade e/ou viabilidade dos mesmos, são priorizados requisitos com maior urgência (exemplo: erro que impeça o uso) e também a quantidade de clientes que estão no aguardo da implementação.

d) Implementação

Fase de codificação do sistema, fazendo o uso de linguagens de programação, definição de classes e objetos, desenvolvimento dos requisitos definidos no projeto.

d) Testes.

Para Arilo Claudio Dias Neto,

> "Teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, dizemos que sua natureza é "destrutiva", e não "construtiva", pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final." (DEVMEDIA, 2014)

Nessa etapa são executadas várias rotinas de testes a fim de validar o produto de software, testando as funcionalidades e analisando seu comportamento em vários aspectos, resultando em um relatório de testes.

2.4 GERÊNCIA DE REQUISITOS

Sommerville (2011) destaca que os requisitos são definidos nas fases iniciais de um projeto e especificam o que deve ser implementado. Os requisitos são as descrições das funcionalidades, propriedades e atributo do sistema. Requisitos podem descrever facilidades em termos de usuário, propriedades e restrições do sistema e restrições no desenvolvimento do sistema.

Diversos fatores contribuem para que mudanças, sejam para realizar ajustes, corrigir erros, incluir ou melhorar funcionalidades. Além dos aspectos inerentes ao processo de levantamento de requisitos e entendimento dos interesses e necessidades dos usuários, estão as mudanças nas regras de negócio que podem ocorrer durante o processo de desenvolvimento do software. Devido às inevitáveis alterações nos requisitos, é necessário gerenciar como essas modificações serão realizadas. Além disso, é responsabilidade deste processo identificar e documentar os requisitos do software a ser desenvolvido (SANTOS, 2014).

No MPS.BR, o processo de gerência de requisitos encontra-se no nível G, definindo um dos primeiros processos a serem implementados por uma organização de software. Para o MPS.BR (SOFTEX, 2011) o propósito do processo Gerência de Requisitos é gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto.

De acordo com o MPS.BR (SOFTEX, 2011), os resultados esperados do processo gerência de requisitos envolvem o entendimento dos requisitos é obtido junto aos fornecedores de requisitos e esse entendimento deve ser comprovado.

Para esse entendimento e respectiva comprovação, inicialmente, é necessário identificar os responsáveis por definir e alterar os requisitos para que os requisitos não sejam definidos de forma equivocada. A identificação deve ser acompanhada de documentação que contenha os requisitos avaliados. Esse documento, embora seja desenvolvido de acordo com os padrões da organização, deve conter a definição dos requisitos funcionais e não funcionais do sistema, além de outras informações relevantes para o desenvolvimento do sistema.

Ë necessário, ainda, apresentar uma forma de avaliação dos requisitos. Essa avaliação pode ser realizada por meio de uma lista de critérios objetivos que guiem a análise a cerca do entendimento dos requisitos. Esses critérios devem ser voltados para a visão dos fornecedores de requisitos, ou seja, não devem ser considerados detalhes técnicos voltados para o desenvolvimento dos requisitos (SANTOS, 2014).

2.5 GERÊNCIA DE CONFIGURAÇÃO

Gestão de configuração de software define um conjunto de atividades desenvolvidas para administrar modificações ao longo do ciclo de vida de software. Para Pressman (2008) essa gestão pode ser vista como uma atividade de garantia de qualidade aplicada ao longo de todo o processo de software. E sua principal responsabilidade é o controle de modificação.

O propósito do processo gerência de configuração é estabelecer e manter a integridade de todos os produtos de trabalho de um processo ou projeto e disponibilizá-los a todos os envolvidos (SOFTEX, 2012).

Os resultados esperados com a gerência de configuração são (SOFTEX, 2012, p. 31):

a) Um sistema de gerência de configuração é estabelecido e mantido;

b) Os itens de configuração são identificados com base em critérios estabelecidos;

c) Os itens de configuração sujeitos a um controle formal são colocados sob *baseline*;

d) A situação dos itens de configuração e das *baselines* é registrada ao longo do tempo e disponibilizada;

e) Modificações em itens de configuração são controladas;

f) O armazenamento, o manuseio e a liberação de itens de configuração e *baselines* são controlados;

g) Auditorias de configuração são realizadas objetivamente para assegurar que as *baselines* e os itens de configuração estejam íntegros, completos e consistentes.

3 MATERIAIS E MÉTODO

Este capítulo está dividido em duas seções, sendo a primeira voltada para descrever os materiais e a segunda para apresentar o método utilizado para a elaboração deste trabalho.

3.1 MATERIAIS

Tendo como base os requisitos identificados, foram desenvolvidos alguns diagramas da UML utilizando o software de modelagem *StarUML.*

No Quadro 2 estão representadas as ferramentas utilizadas juntamente com as suas finalidades.

Ferramenta / Tecnologia	**Versão**	**Referência**	**Finalidade**
StarUML	5.0.2.1570	http://staruml.sourceforge.net	Documentação da modelagem baseada na UML.
Microsoft SQL Server Management Studio	11.0.3128.0	http://www.microsoft.com/pt-br/download/details.aspx?id=7593	Diagramação do DER.
.Net	4.5	http://www.microsoft.com/net	Linguagem de programação.
Visual Studio	Ultimate 2013	http://www.visualstudio.com/	Ambiente de desenvolvimento.
JavaScript		http://www.javascriptsource.com/	Linguagem de programação no lado cliente.
jQuery		http://jquery.com/	Biblioteca para JavaScript.
Bootstrap		http://getbootstrap.com/	Biblioteca CSS.
SQL Server	2011	http://www.microsoft.com/pt-br/server-cloud/products/sql-server/	Administrador do banco de dados.
IIS		http://www.iis.net/	Servidor.
Entity framework		http://www.iis.net/	Persistência.

Quadro 2 - Ferramentas e tecnologias utilizadas

3.2 MÉTODO

A modelagem do sistema foi realizada como estágio, sob a forma de estudo dirigido, pelos autores deste trabalho. Além da modelagem, também foi desenvolvido o referencial teórico. Neste trabalho de conclusão de curso a ênfase esteve no estudo das tecnologias e na implementação do sistema.

Inicialmente recursos das tecnologias foram estudados e testados visando identificar a melhor opção para implementar o sistema. Em seguida as operações de inclusão, exclusão, consulta e alteração de um cadastro foram implementadas. Permitindo, assim, em seguida reproduzir essas operações para os demais cadastros simples.

As funcionalidades específicas e mais complexas do sistema foram desenvolvidas posteriormente. Testes unitários e de atendimento das funcionalidades foram realizados visando identificar erros de código e a adequada implementação dos requisitos.

4 RESULTADO

No presente capítulo é apresenta a modelagem e a implementação realizadas.

4.1 ESCOPO DO SISTEMA

O sistema desenvolvido tem como objetivo registrar, organizar e controlar os requisitos de desenvolvimento de software, sendo voltado para empresas de software de pequeno e médio porte.

Inicialmente são cadastrados os produtos (o software em desenvolvimento) assim como suas respectivas versões e clientes. Em seguida, à medida que os requisitos são identificados, eles são cadastrados e posteriormente é realizado o vínculo com “ocorrências” e “requisitos”. Assim que os produtos estiverem cadastrados, os usuários poderão registrar ocorrências que servirão para documentação e síntese do requisito a ser desenvolvido.

Os usuários poderão cadastrar requisitos que estarão descritos em um texto para o acompanhamento do seu desenvolvimento. Clientes poderão ser vinculados a esses requisitos.

4.2 MODELAGEM DESENVOLVIDA

A seguir estão os requisitos e diagramas modelados a fim de representar os processos e toda a estrutura que servirá como base para o software que será desenvolvido posteriormente.

O levantamento de requisitos está dividido em requisitos funcionais, que representam funcionalidades essenciais para o sistema e requisitos não funcionais, ou seja, regras de negócio, segurança, qualidade do software, etc.

Nos quadros a seguir estão representados com a sigla “RF” (Requisitos Funcionais) e “RNF” requisitos não funcionais, juntamente com o nome e descrição

breve dos mesmos.

A coluna “Implementar” representa que o requisito em questão será implementado e não será reaproveitado de outras estruturas, já a coluna “Prioridade” indica que o requisito é essencial para o funcionamento do sistema.

O Quadro 3 apresenta a listagem dos requisitos funcionais do sistema.

Identificação	Nome	Descrição	Implementar	Prioridade
RF1	*Cadastrar Usuário*	O Administrador do sistema cadastra os usuários informando o nome, *login* e senha.	(X)	(X)
RF2	*Cadastrar Produto*	O Administrador do sistema cadastra o produto a ser desenvolvido.	(X)	(X)
RF3	*Cadastrar Versão*	O Administrador do sistema cadastra a versão do produto a ser desenvolvido.	( )	( )
RF4	*Cadastrar Tipo do Requisito*	O Administrador do sistema cadastra os tipos dos requisitos.	( )	( )
RF5	*Cadastrar Situação do Requisito*	O Administrador do sistema cadastra as situações possíveis para os requisitos. (Como “concluído” e “em desenvolvimento”)	( )	(X)
RF6	*Cadastrar Tipo de Ocorrência*	O Administrador do sistema cadastra os tipos de ocorrência. (Como “Correção relatada” e “Solicitação de Migração de Dados”)	( )	(X)
RF7	*Cadastrar Cliente*	O Administrador do sistema cadastra os clientes.	(X)	( )
RF8	*Cadastrar Requisitos*	O usuário do sistema cadastra os requisitos que serão registrados para o desenvolvimento do produto.	(X)	(X)

Quadro 3 - Requisitos Funcionais

O Quadro 4 apresenta os requisitos não funcionais definidos para o sistema.

Identificação	Nome	Descrição	Implementar	Prioridade
RNF1	*Efetuar Login*	O sistema autenticará os dados de *login* e senha para conceder acesso ao sistema.	(X)	(X)
RNF2	*Interface*	O sistema deverá operar em uma interface *web*.	(X)	(X)

Quadro 4 - Requisitos Não Funcionais

O detalhamento dos requisitos é apresentado nos Quadros de 5 a 16. O Quadro 5 apresenta o detalhamento do requisito funcional Cadastrar Usuário.

RF1 – Cadastrar Usuário
Descrição: O administrador do sistema cadastrará os usuários que terão acesso ao sistema. **Fontes:** Usuário passará as informações ao administrador. **Usuário:** Administrador. **Informações de entrada:** Nome, *login* e senha de acesso do usuário. **Informações de saída:** Lista de usuários do sistema. **Restrições lógicas:** 1. Não permitir que sejam cadastrados usuários de mesmo *login*. **Restrições tecnológicas:** 1. O cadastro será feito a partir de uma tela modal.

Quadro 5 - Detalhamento do requisito funcional Cadastrar Usuário

O Quadro 6 apresenta o detalhamento do requisito funcional Cadastrar Produto.

RF2 – Cadastrar Produto
Descrição: O administrador do sistema cadastrará os produtos que são desenvolvidos pela empresa. **Fontes:** O administrador. **Usuário:** Administrador. **Informações de entrada:** Somente o nome do produto. **Informações de saída:** Lista de produtos da empresa (nome). **Restrições lógicas:** 1. Não permitir que sejam cadastrados produtos de mesmo nome. **Restrições tecnológicas:** 1. O cadastro será feito a partir de uma caixa de texto dentro de um *grid*.

Quadro 6 - Detalhamento do requisito funcional Cadastrar Produto

O Quadro 7 apresenta o detalhamento do requisito funcional Cadastrar Versão.

RF3 – Cadastrar Versão
Descrição: O administrador do sistema cadastrará as versões dos produtos desenvolvidos pela empresa. **Fontes:** O administrador. **Usuário:** Administrador. **Informações de entrada:** Nome/número da versão. **Informações de saída:** Lista de versões (nome). **Restrições lógicas:** 1. Não permitir que sejam cadastradas versões de nomes iguais para um mesmo produto. **Restrições tecnológicas:** 1. A listagem de produtos deve conter um botão de atalho para o cadastro de suas respectivas versões. 2. O cadastro de versões será feito a partir de uma tela modal com a listagem das versões. O nome será informado em uma caixa de texto dentro de um *grid*.

Quadro 7 - Detalhamento do requisito funcional Cadastrar Versão

Quadro 8 apresenta o detalhamento do requisito funcional Cadastrar Tipo do Requisito.

RF4 – Cadastrar Tipo do Requisito
Descrição: O administrador do sistema cadastrará os tipos dos requisitos. **Fontes:** O administrador. **Usuário:** Administrador. **Informações de entrada:** Somente o nome do tipo do requisito. **Informações de saída:** Lista de tipos de requisitos (nome). **Restrições lógicas:** 1. Não permitir que sejam cadastrados tipos de requisitos de mesmo nome. **Restrições tecnológicas:** 1. O cadastro será feito a partir de uma caixa de texto dentro de um *grid*.

Quadro 8 - Detalhamento do requisito funcional Cadastrar Tipo do Requisito

O Quadro 9 apresenta o detalhamento do requisito funcional Cadastrar Situação do Requisito.

RF5 – Cadastrar Situação do Requisito
Descrição: O administrador do sistema cadastrará as situações do requisito. **Fontes:** O administrador. **Usuário:** Administrador. **Informações de entrada:** Nome da situação do requisito. **Informações de saída:** Lista de situações do requisito (nome). **Restrições lógicas:** 1. Não permitir que sejam cadastradas situações do requisito de mesmo nome. **Restrições tecnológicas:** 1. O cadastro será feito a partir de uma caixa de texto dentro de um *grid*.

Quadro 9 - Detalhamento do requisito funcional Cadastrar Situação do Requisito

O Quadro 10 apresenta o detalhamento do requisito funcional Cadastrar Tipo de Ocorrência.

RF6 – Cadastrar Tipo de Ocorrência
Descrição: O administrador do sistema cadastrará o tipo de ocorrência. **Fontes:** O administrador. **Usuário:** Administrador. **Informações de entrada:** Nome do tipo de ocorrência. **Informações de saída:** Lista de tipos de ocorrência (nome). **Restrições lógicas:** 1. Não permitir que sejam cadastrados tipos de ocorrência de mesmo nome. **Restrições tecnológicas:** 1. O cadastro será feito a partir de uma caixa de texto dentro de um *grid*.

Quadro 10 - Detalhamento do requisito funcional Cadastrar Tipo de Ocorrência

O Quadro 11 apresenta o detalhamento do requisito funcional Cadastrar Cliente.

RF7 – Cadastrar Cliente
Descrição: O administrador do sistema poderá cadastrar os clientes da empresa. **Fontes:** O administrador, cliente. **Usuário:** Administrador. **Informações de entrada:** Nome, CPF/CNPJ, informações de endereço e contato.

Informações de saída:
Lista de clientes.
Restrições lógicas:
1. Não permitir que sejam cadastrados clientes com o mesmo CPF/CNPJ.

Restrições tecnológicas:
1. O cadastro será feito a partir de uma tela modal.

Quadro 11 - Detalhamento do requisito funcional Cadastrar Cliente

O Quadro 12 apresenta o detalhamento do requisito funcional Cadastrar Requisito.

RF8 – Cadastrar Requisito
Descrição: O usuário do sistema poderá cadastrar os requisitos do produto. **Fontes:** Todos os usuários do sistema, cliente. **Usuário:** Administrador, Usuário. **Informações de entrada:** Produto, tipo e situação do requisito e a descrição do mesmo por meio de um texto livre. **Informações de saída:** Lista de requisitos. **Restrições lógicas:** Não há. **Restrições tecnológicas:** 1. A descrição do requisito possuirá um limite de 500 caracteres.

Quadro 12 - Detalhamento do requisito funcional Cadastrar Requisitos

O Quadro 13 apresenta o detalhamento do requisito funcional Cadastrar Ocorrência.

RF9 – Cadastrar Ocorrência
Descrição: O usuário do sistema poderá cadastrar as ocorrências. **Fontes:** Todos os usuários do sistema, cliente. **Usuário:** Administrador, Usuário. **Informações de entrada:** Produto e tipo da ocorrência. **Informações de saída:** Lista de ocorrências. **Restrições lógicas:** Não há. **Restrições tecnológicas:** 1. A descrição da ocorrência possuirá um limite de 500 caracteres.

Quadro 13 - Detalhamento do requisito funcional Cadastrar Ocorrência

O Quadro 14 apresenta o detalhamento do requisito funcional Cadastrar Feedback.

RF10 – Cadastrar Feedback
Descrição: O usuário do sistema poderá cadastrar *feedback* nos requisitos e ocorrências. **Fontes:** Todos os usuários do sistema. **Usuário:** Administrador, Usuário. **Informações de entrada:** Descrição, Data de Início e Fim do *feedback*. **Informações de saída:** Lista de *feedbacks* do requisito/ocorrência em questão. **Restrições lógicas:** Não há. **Restrições tecnológicas:** 1. A listagem de *feedbacks* aparecerá em um *grid* no cadastro do requisito/ocorrência.

Quadro 14 - Detalhamento do requisito funcional Cadastrar Feedback

O Quadro 15 apresenta o detalhamento do requisito não funcional Efetuar *login*.

RNF1 – Efetuar login
Descrição: O sistema autenticará os dados de *login* e senha do usuário. **Fontes:** Todos os usuários do sistema. **Usuário:** Administrador, Usuário. **Informações de entrada:** Login e senha. **Informações de saída:** Não há. **Restrições lógicas:** Não há. **Restrições tecnológicas:** Não há.

Quadro 15 - Detalhamento do requisito não funcional Efetuar login

O Quadro 16 apresenta o detalhamento do requisito não funcional Interface.

RNF2 – Interface
Descrição: O sistema operará em uma interface *web*. **Fontes:** Não há. **Usuário:**

Administrador, Usuário. **Informações de entrada:** Não há. **Informações de saída:** Não há. **Restrições lógicas:** Não há. **Restrições tecnológicas:** Não há.

Quadro 16 - Detalhamento do requisito não funcional Interface

A expansão criada para o caso de uso “Cadastrar Ocorrência” é apresentada no Quadro de 17.

Caso de Uso: Cadastrar Ocorrência Permitir que um Administrador ou Usuário do sistema cadastre ocorrências
Atores: Administrador, Usuário
Pré-condições: Deve haver uma ocorrência em edição
Pós-condições: 1**-** O usuário poderá cadastrar *feedbacks* para ocorrência. 2 - O usuário poderá cadastrar um requisito a partir da ocorrência. 3 - O usuário poderá associar um cliente à ocorrência.
Fluxo Principal: 1 – [IN] O usuário informa seu *login* e senha; 2 – [OUT] O sistema lista os menus de acesso aos módulos. 3 – [IN] O usuário acessa o módulo de ocorrências. 4 – [OUT] O sistema lista as ocorrências. 5 – [IN] O usuário edita ou cria uma nova ocorrência. 6 – [OUT] O sistema abre a tela de edição de ocorrência. 7 – [IN] O usuário informa os campos obrigatórios e salva a ocorrência.
Tratamento de Exceções: 1a – O *login* ou senha estão errados 1a1 – [OUT] O sistema informa o usuário; 1a2 – [OUT[Retorna ao fluxo principal no passo 1; 7a – Os campos obrigatórios não foram preenchidos. 7a1 – [OUT] Sistema informa o usuário; 7a2 – [OUT] Retorna ao fluxo principal no passo 5;

Quadro 17 - Expansão do UC ”Cadastrar Ocorrência”

A expansão criada para o caso de uso “Cadastrar Requisito” é apresentada no Quadro 18.

Caso de Uso: Cadastrar Requisito Permitir que um Administrador ou Usuário do sistema cadastre requisitos
Atores: Administrador, Usuário
Pré-condições: 1-Deve haver um produto cadastrado. 2-Deve haver uma versão cadastrada.

3-Deve haver um tipo de requisito cadastrado. 4-Deve haver uma situação de requisito cadastrada. 5-Deve haver um requisito em edição.
Pós-condições: 1**-** O usuário poderá cadastrar *feedbacks* para o requisito. 2 - O usuário poderá associar um cliente ao requisito.
Fluxo Principal: 1 – [IN] O usuário informa seu *login* e senha; 2 – [OUT] O sistema lista os menus de acesso aos módulos. 3 – [IN] O usuário acessa o módulo de requisitos. 4 – [OUT] O sistema lista os requisitos. 5 – [IN] O usuário edita ou cria um novo requisito. 6 – [OUT] O sistema abre a tela de edição de requisito. 7 – [IN] O usuário informa os campos obrigatórios e salva a ocorrência.
Tratamento de Exceções: 1a – O *login* ou senha estão errados 1a1 – [OUT] O sistema informa o usuário; 1a2 – [OUT[Retorna ao fluxo principal no passo 1; 7a – Os campos obrigatórios não foram preenchidos. 7a1 – [OUT] Sistema informa o usuário; 7a2 – [OUT] Retorna ao fluxo principal no passo 5;

Quadro 18 – Expansão do UC "Cadastrar Requisito"

A Figura 11 representa o diagrama de Casos de Uso, o qual foi implementado utilizando o software *Open Source* de modelagem *StarUML*. No diagrama estão representados os atores “Administrador” e “Colaborador”, conectados às suas respectivas funcionalidades (Casos de Uso).

Os “extends” na Figura 11 indicam que os casos de uso base incorporam condicionalmente o comportamento do caso de uso especificado indiretamente pelo caso de uso estendido. De acordo com Booch, Rumbaugh e Jacobsen (2000), o caso de uso base poderá permanecer isolado, mas, sob certas condições, o seu comportamento poderá se estendido pelo comportamento do outro caso de uso. O caso de uso poderá ser estendido somente nos pontos definidos como de extensão. É nesse contexto que são representados os relacionamentos de extensão na Figura 11.

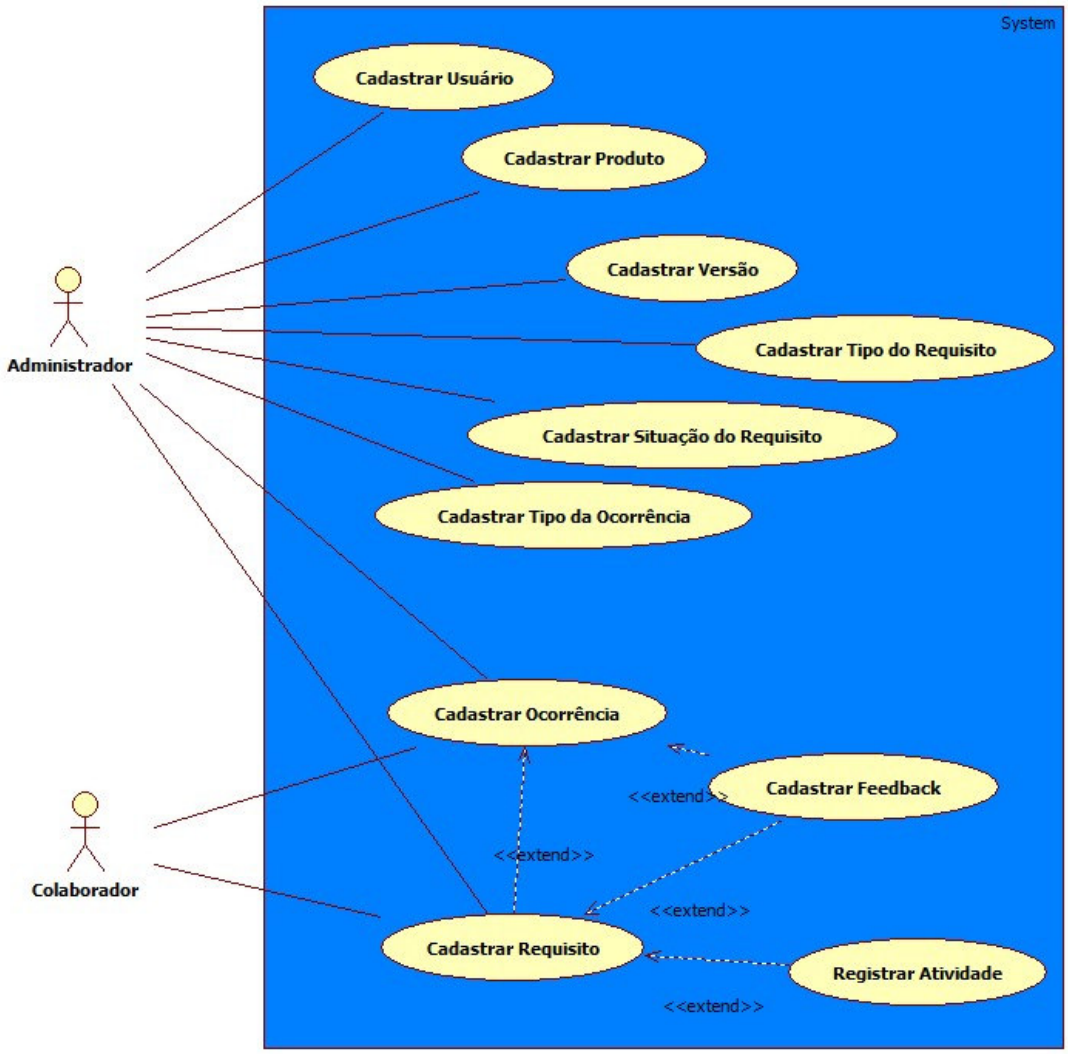

Figura 11 – Diagrama de Casos de Uso

A Figura 11 apresenta o diagrama de classes desenvolvido para representar as classes do sistema. Estão representados os atributos de cada classe e seus respectivos tipos e o relacionamento entre as classes.

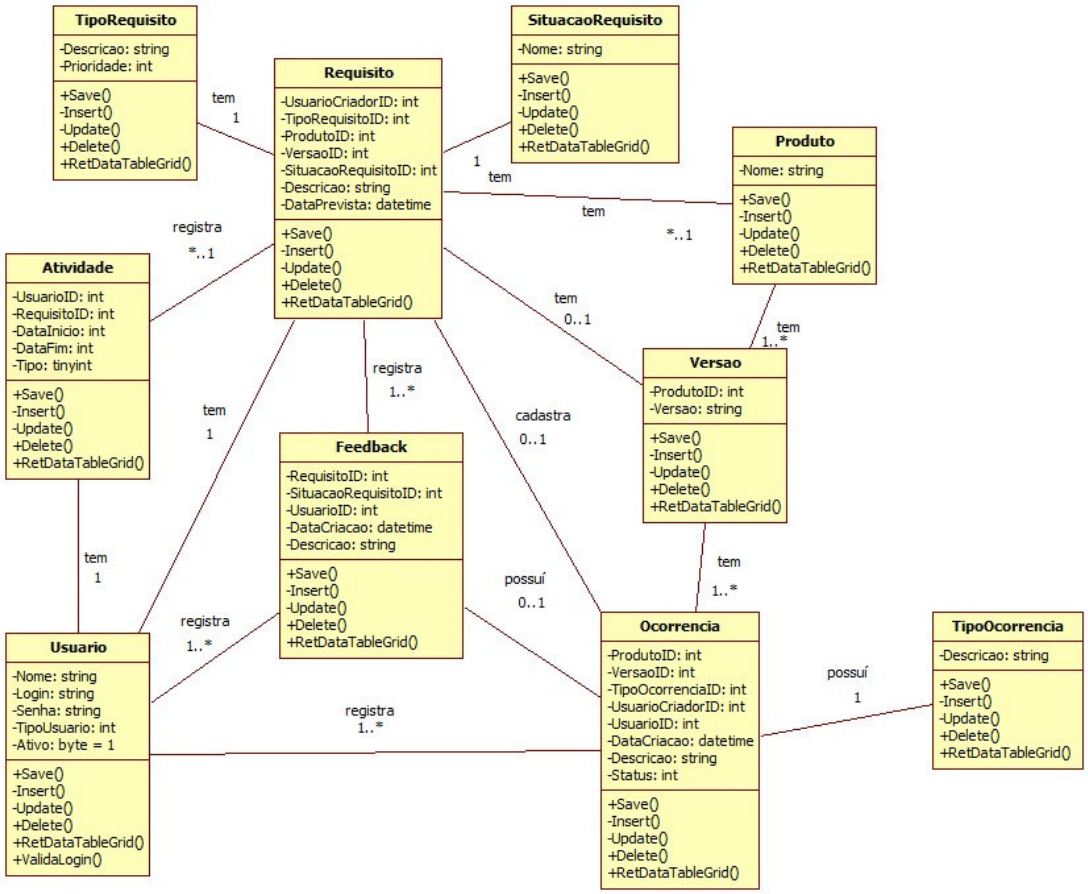

Figura 12 - Diagrama de classes

Conforme apresentado na Figura 12, será permitido o registro de *feedback* tanto para requisitos quanto para ocorrências, para fins de registro dos trabalhos realizados.

O usuário poderá cadastrar um requisito a partir de uma ocorrência cadastrada e também poderá associar os clientes interessados no requisito.

A classe LogsRequisitos tem a função de registro das situações em que o requisito passou (Ex.: Desenvolvido, Passou, Falhou).

Na Figura 13 está representado o modelo relacional do banco de dados, com os campos e seus respectivos tipos de dados. Esse modelo facilitará a criação do banco de dados.

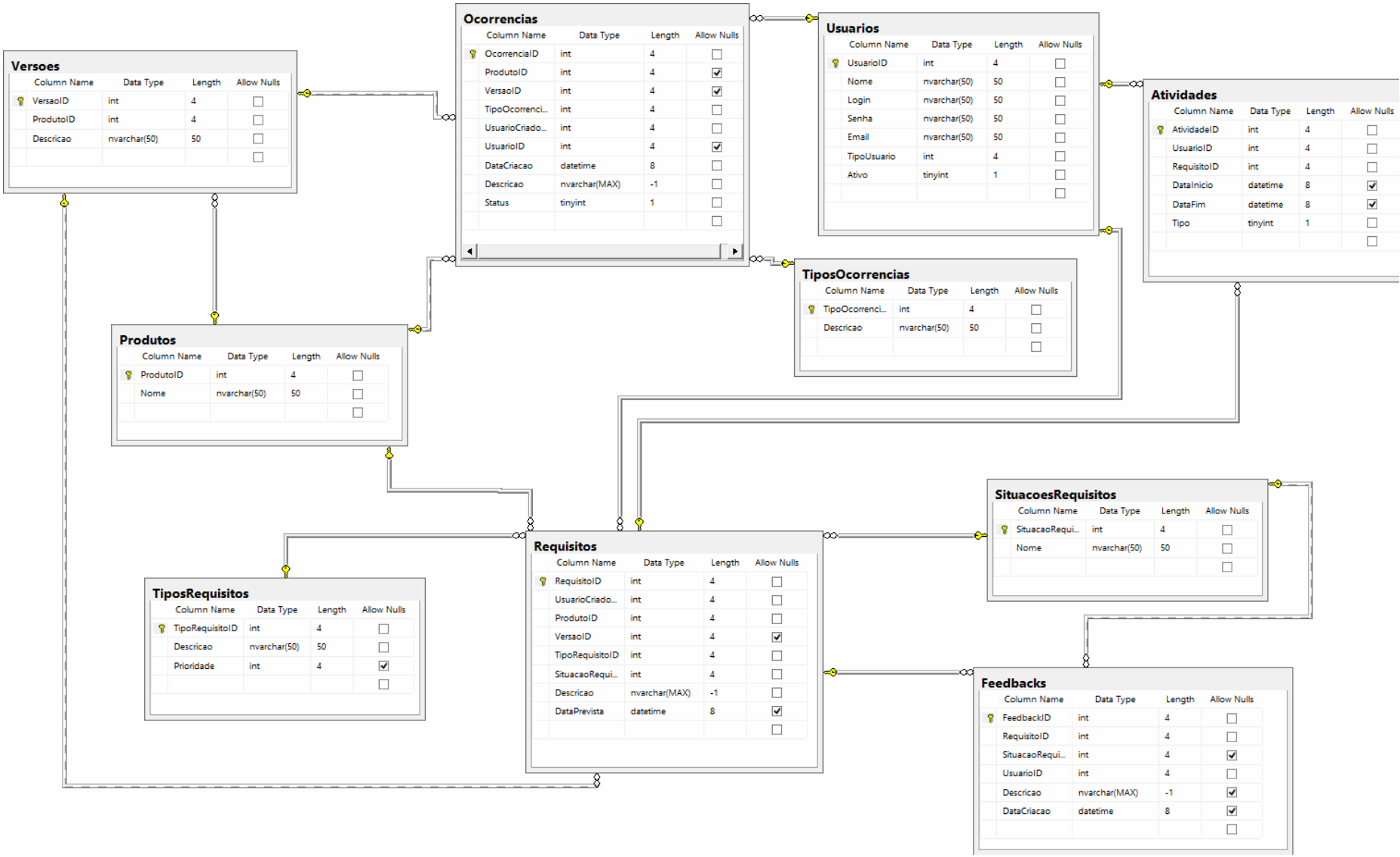

Figura 4 - Modelo relacional do banco de dados

4.3 APRESENTAÇÃO DO SISTEMA

Ao acessar o sistema é apresentada a tela de *login* (Figura 5). Por meio dessa tela o usuário irá inserir seus dados e caso estejam corretos de acordo com a validação realizada no banco de dados, será apresentada a tela inicial do sistema.

Figura 5 – Tela de *login*

Na tela inicial do sistema (Figura 6) está a barra de menus para navegação geral no sistema. As telas que possuem as chamadas para os cadastros como, por exemplo, o cadastro de usuários, serão acessadas por meio da barra de menus que se encontra no topo da tela.

Figura 6 – Barra de menus

A barra de menus estará presente em todas as telas e é utilizada para acessar todos os cadastros e funcionalidade presentes no sistema.

As telas de cadastros apresentam vários componentes utilizados para inserção de novas informações na base de dados do sistema. A região destacada da Figura 7 apresenta as funcionalidades vinculadas aos cadastros. Quando acessadas (por meio da barra de menus), a tela apresentará um *grid* com a listagem de dados contidos na base. O *grid* apresentará paginação quando constarem mais de 10 itens no cadastro específico. O *grid* está na parte central da Figura 7.

Figura 7 – Tela de cadastro de usuários

O menu de ações apresenta as opções de inserir, editar e excluir registros de determinado cadastro. As opções de Inserir e Editar abrirão uma tela do formato *modal na qual* são apresentados os campos para inserção ou edição de dados. A opção de Excluir removerá os registros com a *checkbox* marcada (apresentada no final da listagem), após uma mensagem de confirmação de exclusão.

Ao utilizar as opções de Incluir e Editar no menu de ações é apresentado o *modal* de Cadastros. Esse *modal* é apresentado para todas as telas de cadastro. Os campos estarão de acordo com as funcionalidades do cadastro, mas o *modal* apresentado terá sempre a mesma forma.

Na Figura 8 é apresentado o *modal* no qual são inseridas as informações referentes ao Cadastro de Usuários. Constam os campos de Nome, Login, Senha, E-mail e opções para definir o tipo de usuário e se ele será ativo ou inativo. No caso da opção “Ativo”, a *box* será exibida no *modal* marcada por padrão, para casos de inserção de registros. No *grid* de apresentação dos dados será exibida uma *box*, porém somente para visualização da situação dos dados. E somente poderão ser inativados ou reativados cadastros acessando a opção de Edição, no menu de ações.

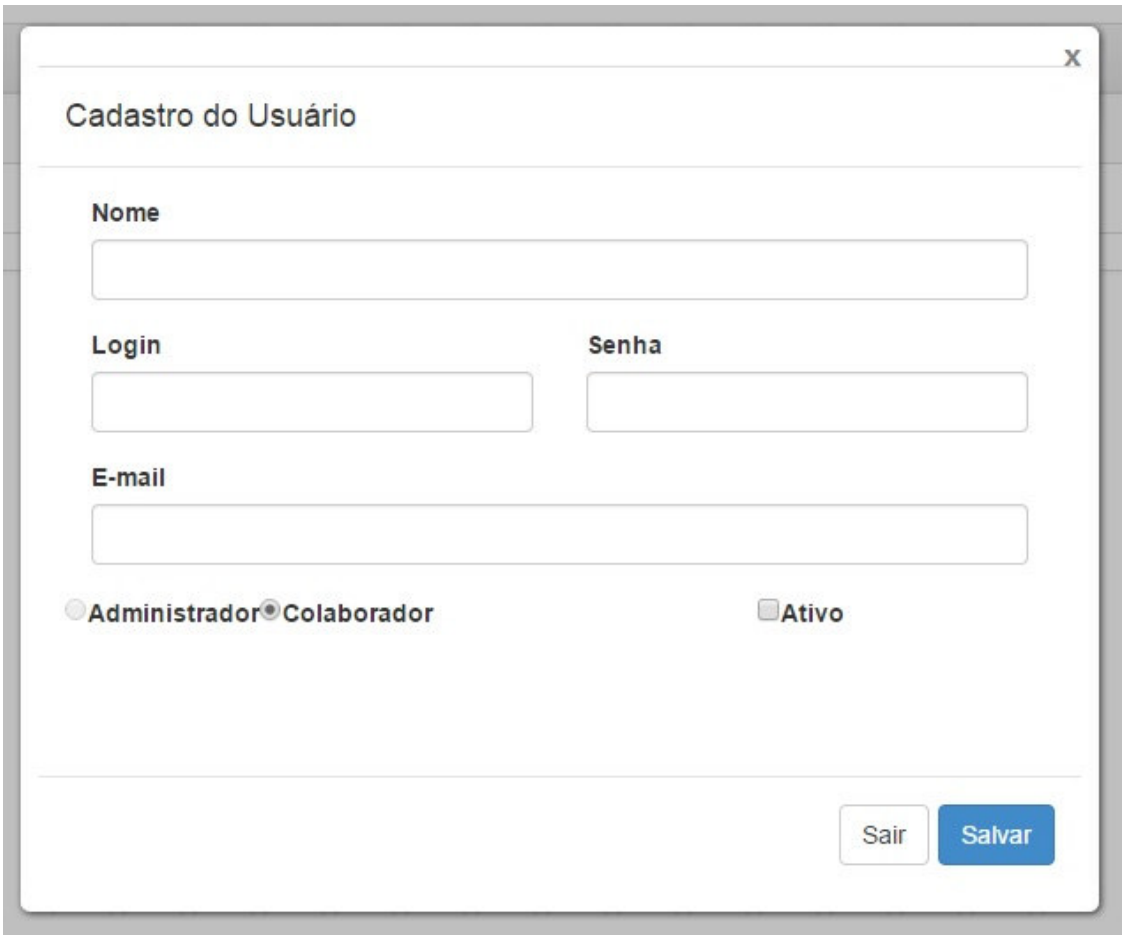

Figura 8 – Tela de modal cadastro

Ao inserir as informações e salvar o cadastro, os dados serão armazenados no banco de dados. O *modal* permanecerá aberto e apresentará uma mensagem informando que o registro atual, tanto para inserção quando edição foi salvo corretamente. Após a inserção de um registro o mesmo será automaticamente listado no *grid*, juntamente com os demais registros que já se encontrem na base de dados. O formato de inserção e listagem dos dados segue o mesmo modelo dos Usuários para os demais cadastros.

Para cadastrar novos requisitos de software no sistema, deve-se utilizar o botão “+ Novo Requisito”, que direcionará o sistema para a tela de cadastro de requisitos. Esse botão (Figura 9) também está presente na barra de menus.

Figura 9 – Tela de cadastro de usuários

Diferente dos demais cadastros do sistema, para inserir novos requisitos não é utilizado um cadastro *modal.* O usuário será redirecionado para uma página de cadastro (Figura 10).

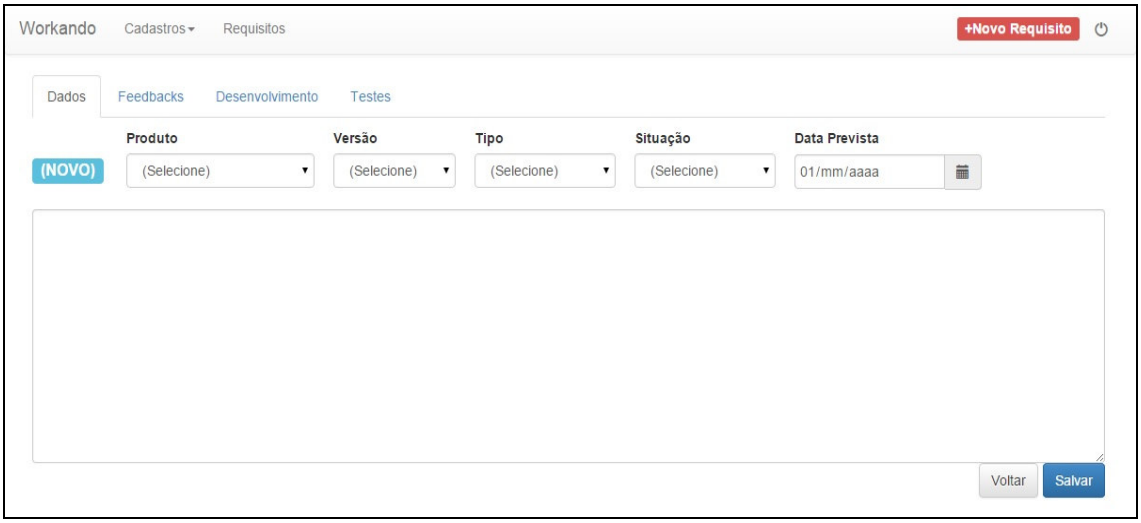

Figura 10 – Tela de cadastro de requisitos

Na tela de cadastro de requisitos são solicitados o preenchimento das informações de produto, versão, tipo, situação e data prevista. Essas informações devem ser cadastradas em suas devidas páginas para que sejam listadas e selecionadas nesse momento. Ao selecionar um produto são listadas somente as versões que estejam vinculadas ao mesmo. Em tipo são listados os tipos de atividades realizadas e a que será definida para o novo registro, como uma correção ou desenvolvimento. O campo situação exibirá as situações em que o requisito pode se encontrar e qual a situação que o atual se encontra. Para requisitos que não foram trabalhados, provavelmente serão cadastrados inicialmente na situação “proposto”. Para o campo de data prevista, a mesma pode ser inserida clicando no campo, para o dia, mês ou ano, e inserindo a informação manualmente, ou clicando no campo para inclusão da data é apresentado um calendário que pode ser utilizado

para consultar as datas e selecionando o dia, mês e ano que deverá ocupar o campo de data. Após a inserção desses dados, deve ser definida a descrição do requisito. Campo no qual pode ser inserido todo o detalhamento necessário a respeito da atividade que será realizada e após isso o cadastro já pode ser salvo. Se todos os dados estiverem corretos, uma caixa informará que o cadastro foi realizado com sucesso. A Figura 11 apresenta a tela de cadastro de requisitos com a listagem de requisitos cadastrados.

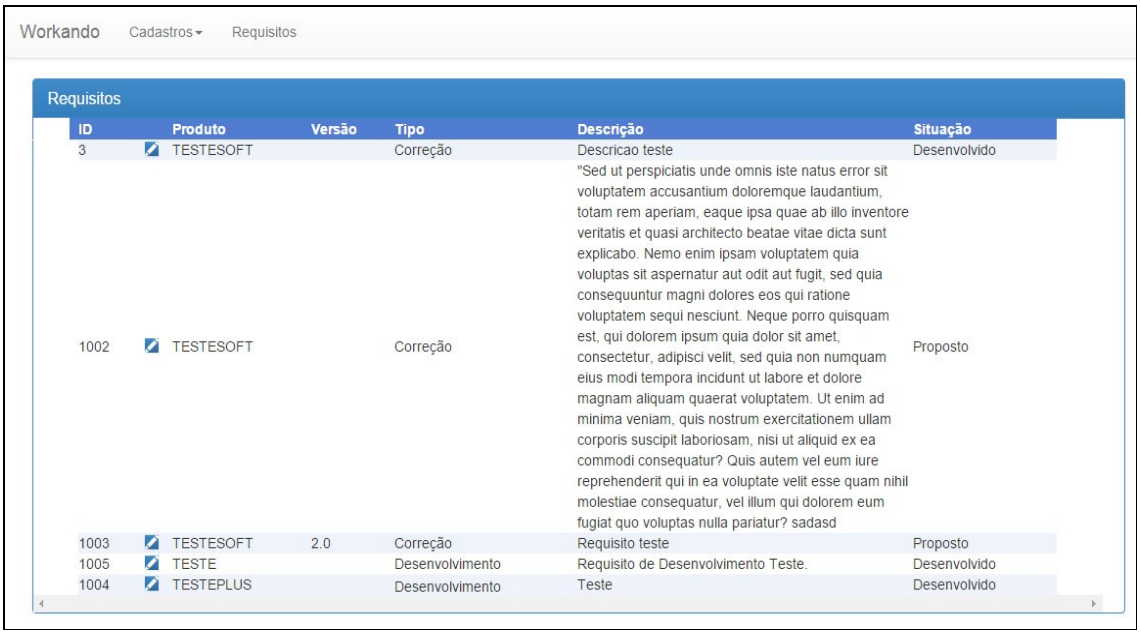

Figura 11 – Tela de listagem de requisitos cadastrados

Na tela de requisitos, que tem seu acesso por meio da opção “Requisitos” na barra de menus, é possível visualizar a listagem de todos os requisitos do sistema. No *grid* de exibição são listados os campos de ID, produto, versão, tipo, descrição, e situação. Os requisitos serão ordenados por prioridade do tipo de requisito do qual ele está associado. Essa prioridade é definida no momento em que as situações são cadastradas. Para editar e trabalhar com um dos requisitos listados, ao lado da coluna do ID do requisito há um botão que abrirá o requisito do qual o usuário escolher no *grid*. A Figura 12 apresenta um requisito em edição.

Figura 12 – Editar requisitos

Ao lado do *grid* no qual são apresentados os requisitos cadastrados, existem os filtros que permitem definir o que será apresentado na listagem. Para os filtros, de acordo com a informação selecionada, somente serão exibidos os registros no *grid* que estiverem de acordo com as informações combinadas no filtro. Para um produto selecionado, poderá ser definida a sua versão caso seja o objetivo do filtro. Se não houver sistema selecionado, não poderá ser selecionada uma versão. Também consta um campo de ID para inserir um ID de requisito para busca no *grid*. Para essa busca basta inserir a numeração e pressionar Enter e a busca será realizada. A Figura 13 apresenta a tela de listagem dos filtros possíveis.

Figura 13 – Filtros da grid de requisitos

Ao Editar um requisito, os dados serão carregados na mesma tela de cadastro inicial de requisitos. Na aba “Dados” podem ser alterados de acordo com a necessidade. Também constam as abas de “Feedbacks”, “Desenvolvimento” e “Testes”. A Figura 14 apresenta a aba Feedbacks.

Figura 14 – Aba feedbacks

Na aba de Feedbacks, toda a atualização necessária por parte dos usuários, deverá ser inserida para controle da situação do requisito, para evitar que informações referentes ao processo realizado sejam perdidas. Para inserir as informações, todos os dados devem ser inseridos no campo de texto exibido ao acessar a aba. Após concluir as inserções de informações, o usuário deverá também selecionar a situação referente aos dados inseridos. Ao adicionar o *feedback*, o conteúdo do mesmo será exibido abaixo do campo de texto, em uma listagem que exibirá o usuário que inseriu as informações, a data e hora do *feedback*, os dados que foram inseridos no campo de texto e também a situação. Todos os *feedbacks* inseridos ficarão nesse formato para consulta. Quando um novo registro é inserido, a listagem é atualizada, e o novo e mais atual registro ficará no topo da lista. (Conforme Figura 14 – Aba de *feedbacks*)

Nas abas Desenvolvimento (Figura 15) e Testes (Figura 16), serão calculados os tempos de desenvolvimento do ID em questão, por meio de Check In e Check Out. A contagem é iniciada quando é realizado Check In e concluída com o Check Out. Quando é iniciada a atividade, o botão de Check In fica inativo, o mesmo ocorre para o Check Out quando a atividade é concluída. O tempo de início e término são gravados na listagem abaixo dos botões na página, também como o tempo que foi trabalhado em determinado momento.

Figura 15 – Aba de desenvolvimento

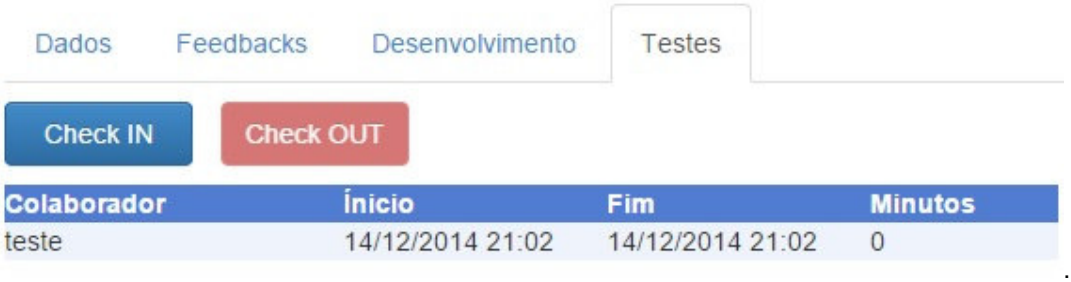

Figura 16 – Aba de testes

4.3 DESENVOLVIMENTO DO SISTEMA

O sistema foi desenvolvido utilizando as tecnologias de desenvolvimento do .NET, em conjunto com JavaScript na qual foi utilizada a biblioteca JQuery e também CSS com a utilização do Bootstrap Framework. As classes orientadas a objeto foram feitas no .NET exibindo a classe em si e seus atributos e o tipo de dados para cada um deles.

Para que a tela de *login* seja exibida ao acessar o sistema, antes do próprio carregamento da página, é feita uma verificação se existe um *login* ativo, chamando a Sub OnInit. A Session dentro dela é responsável por armazenar dados em um *cookie* referente ao *login* ativo e caso nenhum usuário esteja logado no momento do acesso é realizado o redirecionamento para a página de *login*, conforme mostrado na Listagem 1.

```
    Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
        MyBase.OnInit(e)

        If Session("UsuarioID") Is Nothing OrElse Val(Session("UsuarioID")) = 0 Then
            Response.Redirect("~/Login.aspx")
        End If
    End Sub
```

Listagem 1 – Verificação de login ativo

Caso o acesso seja redirecionado para a página de *login* serão solicitados os dados para acesso que devem ser inseridos nas textbox exibidas em tela e serão validadas através do botão Entrar. Ao pressionar o botão, dentro da classe de Login

será chamado o evento btnEntrar_click (Listagem 2), que contém as ações que serão realizadas.

```
    Private Sub btnEntrar_Click(sender As Object, e As EventArgs) Handles
btnEntrar.Click
        Dim nUsuarioID As Integer = Usuario.ValidaLogin(txtLogin.Text, txtSenha.Text)

        If nUsuarioID > 0 Then
            Session("UsuarioID") = nUsuarioID
            Response.Redirect("Default.aspx")
        Else
            ClientScript.RegisterStartupScript(Me.GetType(), "logininvalido",
"alert('Usuário ou senha inválidos!');", True)
        End If
    End Sub
```

Listagem 2 – Evento do clique do botão entrar na tela de login

A função do clique do botão Entrar armazenará o valor retornado da função ValidaLogin da classe de usuários, na variável nUsuarioID do tipo integer. A função receberá os valores inseridos nos campos da tela de *login* para usuário e senha, utilizando-os para realizar um *select* na tabela de usuários na base de dados com essas informações na clausula *where*. Realizada a comparação das informações a *session* será iniciada para o usuário em questão e direcionado para a página inicial do sistema. Caso as informações sejam invalidas será apresentada a mensagem de erro que foi inserida nas ações do botão Entrar (Listagem 3).

```
   Public Shared Function ValidaLogin(ByVal sLogin As String, ByVal sSenha As String)
As Integer
        Dim dt As DataTable
        Dim cmd As New SqlCommand()
        cmd.CommandText = "SELECT TOP 1 UsuarioID FROM Usuarios "
        cmd.CommandText &= "WHERE Login=@Login AND Senha=@Senha"

        cmd.Parameters.Add(New SqlParameter("Login", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Senha", SqlDbType.NVarChar, 50))

        cmd.Parameters("Login").Value = sLogin
        cmd.Parameters("Senha").Value = sSenha

        dt = SQLServer.RetDataTable(cmd)

        If Not dt Is Nothing AndAlso dt.Rows.Count > 0 Then
            Return Val(dt.Rows(0)("UsuarioID"))
        Else
            Return 0
        End If
    End Function
```

Listagem 3 – Função de validação de logins na classe Usuário

Para exemplificar a divisão do sistema em classes será utilizada a classe de Usuários para apresentar a estrutura das classes em geral.

A Listagem 4 (Campos da Classe de Usuários) apresenta os campos da classe de usuários do sistema e define todos os atributos que pertencerão a ela. O campo UsuarioID será a chave primária na base de dados e o campo de identificação dos usuários cadastrados. Será inteiro (*integer*) justamente pela função de identificação na classe e na base. Nome, *login*, senha e *email* serão campos de texto (*string*), no qual serão armazenadas as informações de titulo de cada campo. Para campos como a senha, as informações não devem ser exibidas e será tratada no *modal* de Cadastros para o campo não exibir os caracteres inseridos e salvos. Validações de *email* também serão atribuídas, porém os campos na classe permanecem como *string*. O campo para definir se o aluno é ativo na classe estará em Public Enum Status, uma constante que possui dois valores, um para cada situação. O tipo de usuário estará na constante Public Enum TiposUsuario e cada um dos tipos também receberá um valor diferente.

```
Imports System.Data.SqlClient

Public Class Usuario

#Region "Fields"

    Protected nUsuarioID As Integer
    Protected sNome As String
    Protected sLogin As String
    Protected sSenha As String
    Protected sEmail As String
    Protected nTipoUsuario As Integer
    Protected nAtivo As Integer

    Public Enum TiposUsuario
        Admin = 1
        Colaborador = 2
    End Enum

    Public Enum Status
        Ativo = 1
        Inativo = 0
    End Enum


#End Region
```

Listagem 4 – Campos da classe de usuários

A Listagem 5 (*getter* e *setters*) apresenta os métodos assessores públicos *get* para acesso as informações e os métodos de modificação *set*, para caso a classe possa sofrer alterações de dados, para que assim se definam novos valores.

```
#Region "Properties"

    Public Property UsuarioID() As Integer
        Get
            Return nUsuarioID
        End Get
        Set(ByVal value As Integer)
            nUsuarioID = value
        End Set
    End Property
```

Listagem 5 – Getters e Setters da Classe de Usuários

A Listagem 6 (construtores) apresenta os métodos construtores da Classe de usuários, onde no New() os campos são inicializados, passando valores 0 para o UsuarioID por exemplo, que recebe valores Integer e para os campos String é passado String.Empty, uma cadeia de caracteres vazia, como no caso do Nome por exemplo. Na Sub é passado um ID de usuário é carregado um DataTable com as informações referentes a esse ao registro na tabela de Usuários da base de dados. Após isso é verificado se o DataTable retornou algum registro da base, para assim atribuir as informações do DataRow, ou seja, da linha retornada da base de dados, para os campos específicos.

```
#Region "Construtores"

    Public Sub New()
        Me.UsuarioID = 0
        Me.Nome = String.Empty
        Me.Login = String.Empty
        Me.Senha = String.Empty
        Me.Email = String.Empty
        Me.TipoUsuario = TiposUsuario.Colaborador
        Me.Ativo = Status.Ativo
    End Sub

    Public Sub New(ByVal ID As Integer)
        Me.New()

        Dim dt As DataTable = SQLServer.RetDataTable("Select * From Usuarios WHERE
UsuarioID=" & ID)
        If Not dt Is Nothing AndAlso dt.Rows.Count > 0 Then
            Dim dr As DataRow = dt.Rows(0)
            Me.UsuarioID = ID
            Me.Nome = dr("Nome")
            Me.Login = dr("Login")
```

```
            Me.Senha = dr("Senha")
            Me.Email = dr("Email")
            Me.TipoUsuario = dr("TipoUsuario")
            Me.Ativo = dr("Ativo")
        End If
    End Sub

#End Region
```

Listagem 6 – Construtores da classe de usuários

No *insert* da classe de Usuários (Listagem 7) ocorre o procedimento de adicionar novos usuários. O CommandText receberá o comando SQL de *insert* que define a tabela no caso Usuários e os campos que receberão valores, que para essa tabela são nome, login, senha, email, tipousuario e ativo. Após informados os campos que receberão informações na base são inseridos em *values* os dados que serão inseridos por campo. Os *values* atribuídos são parâmetros criados para cada campo e se apresentam respectivamente na mesma ordem dos campos do *insert*, no formato @nome, @login, @senha, @email, @tipousuario e @ativo. Os valores desses parâmetros são obtidos do objeto Usuário, na página do modal de cadastro de usuários. Por fim o SQLServer.ExecuteScalar executará o comando de cmd na base.

```
    Private Function Insert() As Boolean
        Dim cmd As New SqlCommand()
        cmd.CommandText = "Insert Into Usuarios (Nome, Login, Senha, Email,
TipoUsuario, Ativo) "
        cmd.CommandText &= "Values(@Nome, @Login, @Senha, @Email, @TipoUsuario,
@Ativo)"

        cmd.Parameters.Add(New SqlParameter("Nome", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Login", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Senha", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Email", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("TipoUsuario", SqlDbType.Int))
        cmd.Parameters.Add(New SqlParameter("Ativo", SqlDbType.SmallInt))

        cmd.Parameters("Nome").Value = Me.Nome
        cmd.Parameters("Login").Value = Me.Login
        cmd.Parameters("Senha").Value = Me.Senha
        cmd.Parameters("Email").Value = Me.Email
        cmd.Parameters("TipoUsuario").Value = Me.TipoUsuario
        cmd.Parameters("Ativo").Value = Me.Ativo

        Me.UsuarioID = SQLServer.ExecuteScalar(cmd)

        Return Me.UsuarioID > 0
    End Function
```

Listagem 7 – Inserção de usuários

O objeto usuário recebe os dados dos campos do *modal* de cadastros, informações essas que serão utilizadas pela classe como descritas anteriormente. Para os valores informados em Textbox é utilizado o texto desses campos para atribuir essa informação ao usuário. Para o tipo de usuário e *status* são passados como colaborador e usuário ativo. Ao salvar o objeto ele irá fazer a verificação do registro no *save* da classe de usuários. Também após o *save*, o *grid* é atualizado para exibir a ultima informação cadastrada juntamente com as demais que já poderiam estar na base.

```
    Private Sub Salvar()
        Dim oUsuario As New Usuario(Val(Request("id")))

        With oUsuario
            .Nome = txtNome.Text
            .Login = txtLogin.Text
            .Senha = txtSenha.Text
            .Email = txtEmail.Text
            .TipoUsuario = Usuario.TiposUsuario.Colaborador
            .Ativo = Usuario.Status.Ativo
        End With

        Try
            oUsuario.Save()

            ShowAlert("Dados salvos com sucesso!")
            ScriptManager.RegisterStartupScript(Me.Page, Me.GetType(),
"sRefreshGrid", "window.opener ? window.opener : window.parent.refreshGrid();", True)
        Catch ex As Exception
            ShowAlert(ex.Message)
        End Try

    End Sub
```

Listagem 8 – Objeto Usuário na pagina de cadastro de usuários

O *save* (Listagem 9) verificará por meio do UsuarioID se será inserido um novo registro pelo Insert() da classe ou se retornado um ID mais que zero ocorrerá a alteração do registro. Caso o registro seja editado, o Update() fará a atualização dos dados.

```
    Public Function Save() As Boolean
        If Me.UsuarioID > 0 Then
            Return Update()
        Else
            Return Insert()
        End If
    End Function
```

Listagem 9 – Save da classe de usuários

Da mesma forma que o Insert(), o Update() contém um CommandText, porém dessa vez ele receberá um comando de Update do SQL, definindo para os campos novos valores, que substituirão os dados anteriores na base. Esses novos valores serão os que constarem nos parâmetros do objeto. No retu*r*n o SQLSever.ExecuteNonQuery fará a execução do *update* na base. A Listagem 10 apresenta o código da função Update.

```
    Private Function Update() As Boolean
        Dim cmd As New SqlCommand()
        cmd.CommandText = "UPDATE Usuarios SET "
        cmd.CommandText &= "Nome=@Nome, Login=@Login, Senha=@Senha, Email=@Email,
TipoUsuario=@TipoUsuario, Ativo=@Ativo "
        cmd.CommandText &= "WHERE UsuarioID=@UsuarioID"

        cmd.Parameters.Add(New SqlParameter("UsuarioID", SqlDbType.Int))

        cmd.Parameters.Add(New SqlParameter("Nome", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Login", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Senha", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("Email", SqlDbType.NVarChar, 50))
        cmd.Parameters.Add(New SqlParameter("TipoUsuario", SqlDbType.Int))
        cmd.Parameters.Add(New SqlParameter("Ativo", SqlDbType.SmallInt))

        cmd.Parameters("UsuarioID").Value = Me.UsuarioID

        cmd.Parameters("Nome").Value = Me.Nome
        cmd.Parameters("Login").Value = Me.Login
        cmd.Parameters("Senha").Value = Me.Senha
        cmd.Parameters("Email").Value = Me.Email
        cmd.Parameters("TipoUsuario").Value = Me.TipoUsuario
        cmd.Parameters("Ativo").Value = Me.Ativo


        Return SQLServer.ExecuteNonQuery(cmd) > 0
    End Function
```

Listagem 10 – Edição usuários

Para realizar a exclusão de registros da base é necessário selecionar o registro em questão, clicando do respectivo registro no *grid* e deixando o mesmo selecionado (apresenta marcação em uma cor diferenciada dos demais registros). Após o registro estar marcado, por meio do menu de ações na lateral direita da tela, a opção de “Excluir” fará a chamada do procedimento de exclusão (Listagem 11).

```
    Private Sub btnExcluir_Click(sender As Object, e As EventArgs) Handles
btnExcluir.Click
        Dim oUsuario As New Usuario(hdfUsuarioID.Value)

        Try
            oUsuario.Delete()
```

```
            CarregaGrid()
        Catch ex As Exception
            ShowAlert(ex.Message)
        End Try
    End Sub
```

Listagem 11 – Ação do botão de Excluir na tela de usuários

A Listagem 12 (Exclusão de Usuários) a função de remover o usuário selecionado utilizará o ID do registro, no caso o campo UsuarioID, por meio dos parâmetros do objeto. Dessa forma, dentro do CommandText consta o comando SQL para remoção de um registro específico na base, na cláusula WHERE será passado o valor do parâmetro. No *return* será executado o comando, com o ID selecionado, removendo-o definitivamente.

```
    Public Function Delete() As Boolean
        Dim cmd As New SqlCommand()

        cmd.CommandText = "DELETE FROM Usuarios WHERE UsuarioID=@UsuarioID"
        cmd.Parameters.Add(New SqlParameter("UsuarioID", SqlDbType.Int))
        cmd.Parameters("UsuarioID").Value = Me.UsuarioID

        Return SQLServer.ExecuteNonQuery(cmd) > 0
    End Function
```

Listagem 12 – Exclusão de usuários

As informações cadastradas na base de dados são exibidas nos cadastros através de um *grid*. No exemplo dos usuários, o *grid* é apresentado na pagina de usuários. Para que o *grid* seja mostrado vazio é necessário que sejam carregadas as informações que ele deverá exibir. Na Listagem 13 (Carregar dados no *grid* de usuários) é apresentado o processo de atualização das informações que serão carregadas.

```
    Private Sub CarregaGrid()
        grdUsuarios.DataSource = Usuario.RetDataTableGrid()
        grdUsuarios.DataBind()
        grdUsuarios.SelectedIndex = -1

        hdfUsuarioID.Value = String.Empty

        updGrid.Update()
    End Sub
```

Listagem 13 – Carregar dados no grid de usuários

Os dados que serão carregados no *grid* de usuários são buscados pela função RetDataTableGrid() (Listagem 14). Essa função retornará um DataTable com os dados que o *grid* utilizará. Na função consta um comando SQL armazenado na string sSQL que apresenta todos os campos que serão apresentados no *grid*. Nem todos os campos que constam na tabela necessariamente precisam ser exibidos no *grid*. Após isso, a função retorna o *select* com as informações existentes na base, na tabela de usuário, passando as informações para o DataSource do *grid*. O grdUsuarios.DataBind() fará a ligação das informações carregadas com o *grid*, e o updGrid.Update() irá atualizar os dados do *grid* na tela.

```
    Public Shared Function RetDataTableGrid() As DataTable
        Dim sSQL As String = "" & _
        "SELECT UsuarioID, Nome, Login, Senha, Email, CASE WHEN TipoUsuario = 2 THEN
'Colaborador' ELSE 'Admin' END  AS TipoUsuario, Ativo " & _
        "FROM Usuarios "
        Return SQLServer.RetDataTable(sSQL)
    End Function
```

Listagem 14 – Função para carregar dados no grid de usuários

Para a tela de requisitos, na qual existe a possibilidade de filtrar as informações que serão exibidas no *grid*, existem os combos de filtro com as informações de produto, versão, tipo e situação. Para que essas informações sejam carregadas de acordo com o que está cadastrado na base de dados no momento, dentro da *sub* CarregaCombos (Listagem 15) na classe de requisitos existe a *sub* CarregaCombo, que terá a função de retornar dados na base e devido a isso receberá como parâmetros informações estáticas da tabela que deverá ser selecionada, campo de identificação da tabela e campo de nome ou descrição de acordo com a tabela informada. Com essas informações, dentro de CarregaCombo será montada a string SQL que retornará as informações através do DataTable, para que os dados sejam carregados nas combos dos filtros.

```
    Private Sub CarregaCombos()
        CarregaCombo(cmbProduto, "Produtos", "ProdutoID", "Nome")
        CarregaCombo(cmbVersao, "Versoes", "VersaoID", "Descricao", "ProdutoID=" &
cmbProduto.SelectedValue)
        CarregaCombo(cmbTipo, "TiposRequisitos", "TipoRequisitoID", "Descricao")
        CarregaCombo(cmbSituacao, "SituacoesRequisitos", "SituacaoRequisitoID",
"Nome")
    End Sub
```

```
  Public Sub CarregaCombo(ByRef cmb As DropDownList, ByVal sTabela As String, ByVal
```

```
sCampoID As String, ByVal sCampoTexto As String, Optional ByVal sFiltro As String =
"")
        Dim sSQL As String
        Dim dt As System.Data.DataTable
        Dim dr As System.Data.DataRow

        cmb.DataTextField = sCampoTexto
        cmb.DataValueField = sCampoID

        sSQL = String.Empty
        sSQL &= " SELECT " & sTabela & "." & sCampoID & ", " & sTabela & "." &
sCampoTexto
        sSQL &= " FROM " & sTabela

        If Not String.IsNullOrEmpty(sFiltro) Then
            sSQL &= " WHERE " & sFiltro
        End If

        sSQL &= " ORDER BY " & sTabela & "." & sCampoTexto

        dt = SQLServer.RetDataTable(sSQL)
        dt.TableName = sTabela
        dr = dt.NewRow()
        dr(sCampoID) = 0
        dr(sCampoTexto) = "(Selecione)"
        dt.Rows.InsertAt(dr, 0)
        cmb.DataSource = dt
        cmb.DataBind()
    End Sub
```

Listagem 15 –Sub CarregaCombo da classe requisitos

No caso do *grid* da tela de requisitos, o procedimento é o mesmo da tela de usuários. Por meio de um DataTable as informações são buscadas para serem exibidas. Porém, para o CarregaGrid da classe de Requisitos (Listagem 16 - CarregaGrid), constam as cláusulas *where* referentes aos filtros, por meio dos quais são obtidas as informações das combos para inserir no *where* do *select*.

```
    Private Sub CarregaGrid()
        Dim sWhere As String = String.Empty

        If cmbProduto.SelectedValue <> 0 Then
            sWhere &= " AND Requisitos.ProdutoID=" & cmbProduto.SelectedValue
        End If
        If cmbVersao.SelectedValue <> 0 Then
            sWhere &= " AND Requisitos.VersaoID=" & cmbVersao.SelectedValue
        End If
        If cmbTipo.SelectedValue <> 0 Then
            sWhere &= " AND Requisitos.TipoRequisitoID=" & cmbTipo.SelectedValue
        End If
        If cmbSituacao.SelectedValue <> 0 Then
            sWhere &= " AND Requisitos.SituacaoRequisitoID=" &
cmbSituacao.SelectedValue
        End If
```

```
        If Not String.IsNullOrEmpty(txtID.Text) Then
            sWhere &= " AND Requisitos.RequisitoID=" & txtID.Text
        End If

        grdRequisitos.DataSource = Requisito.RetDataTableGrid(sWhere)
        grdRequisitos.DataBind()

        updGrid.Update()
    End Sub
```

Listagem 16 –Sub CarregaGrid da classe requisitos

Quando a string está definida, ela é passada por parâmetro para o RetDataTable (Listagem 17), que também possui o *select* com os campos que serão exibidos no grid. Se a *string* do filtro não esteja vazia, ela será concatenada com o select, retornando somente as informações que estão de acordo com o filtro. Também nos filtros há um campo para inserir o ID do requisito para busca, sozinho ou juntamente com outras informações do filtro. Nesse caso, como a informação não será apanhada de nenhum *combo*, se o campo não estiver vazio, as informações serão concatenadas junto com o *select* e os filtros, para buscar e exibir os resultados.

```
    Public Shared Function RetDataTableGrid(Optional ByVal sFiltro As String = "") As
DataTable
        Dim sSQL As String = "" & _
        "SELECT RequisitoID, Produtos.Nome AS Produto, Versoes.Descricao AS Versao,
TiposRequisitos.Descricao AS Tipo, Requisitos.Descricao, SituacoesRequisitos.Nome AS
Situacao, FORMAT(DataPrevista, 'dd/MM/yyyy') AS DataPrevista " & _
        "FROM Requisitos " & _
        "INNER JOIN Produtos ON Requisitos.ProdutoID = Produtos.ProdutoID " & _
        "LEFT JOIN Versoes ON Requisitos.VersaoID = Versoes.VersaoID " & _
        "INNER JOIN TiposRequisitos ON Requisitos.TipoRequisitoID =
TiposRequisitos.TipoRequisitoID " & _
        "INNER JOIN SituacoesRequisitos ON Requisitos.SituacaoRequisitoID =
SituacoesRequisitos.SituacaoRequisitoID " & _
        IIf(String.IsNullOrEmpty(sFiltro), "", "WHERE 1=1 " & sFiltro) & _
        "ORDER BY TiposRequisitos.Prioridade DESC, Requisitos.ProdutoID,
Requisitos.VersaoID, RequisitoID"

        Return SQLServer.RetDataTable(sSQL)
    End Function
```

Listagem 17 – Função para carregar dados no grid de usuários requisitos

Para a inserção de *feedbacks* vinculados aos requisitos, as informações de descrição devem ser inseridas no devido campo e ao pressionar o botão de

adicionar (Listagem 18), elas passarão por uma validação para ver se existe alguma informação no campo de texto. Caso exista, a validação é completada, caso contrário, uma mensagem é exibida solicitando que sejam inseridas as informações.

```
    Private Sub btnAddFeedback_Click(sender As Object, e As EventArgs) Handles
btnAddFeedback.Click
        If ValidaFeedback() Then
            AdicionarFeedback()
        End If
    End Sub
```

Listagem 18 –Função do botão adicionar feedback

Após o *feedback* ser validado (código apresentado na Listagem 19), é chamada a sub AdicionarFeedback (Listagem 20), que irá criar o objeto feedback, que receberá as informações do ID, situação, usuário, data atual, e as informações contidas no campo de texto do *feedback*.

```
    Private Function ValidaFeedback() As Boolean
        If String.IsNullOrEmpty(txtDescFeedback.Text) Then
            ShowAlert("Informe uma descrição para o feedback.")
            Return False
        End If
        Return True
    End Function
```

Listagem 19 – Validação dos feedbacks

Após o objeto receber as informações e ser salvo, o campo de texto recebe uma *string* vazia, para que sejam possíveis inserir novas informações de forma instantânea. Por meio da sub CarregGridFeedback(), constante na Listagem 20) que é chamada após isso, o *grid* de *feedback* já é atualizado exibindo o registro salvo em seu topo.

```
    Private Sub AdicionarFeedback()
        Dim oFeedback As New FeedBack

        With oFeedback
            .RequisitoID = ViewState("RequisitoID")
            .SituacaoRequisitoID = cmbSituacaoFeedback.SelectedValue
            .UsuarioID = Session("UsuarioID")
            .Descricao = Trim(txtDescFeedback.Text)
            .DataCriacao = DateTime.Now
        End With

        Try
            If oFeedback.Save() Then
                txtDescFeedback.Text = String.Empty
                CarregaGridFeedback()
```

```
            End If
        Catch ex As Exception
            ShowAlert(ex.Message)
        End Try
    End Sub
```

Listagem 20 – Sub AdicionarFeedbacks

Para o controle dos inícios e términos de atividade, utilizando os botões de Check In e Check Out, é criado um objeto para a atividade. Para o Check In, o UsuarioID que está acessando o sistema no momento é passado para o objeto por meio da Session, o UsuarioID é passado pela ViewState, referente ao registro de requisito que está sendo atualizado no momento. A DataInicio recebe a data atual por meio do DateTime.Now e o tipo desenvolvimento é passado estático para essa aba. Para a aba de Testes o procedimento é idêntico, a diferença é o tipo de atividade estática, passando Atividade.TipoAtividade.Teste. A Listagem 21 apresenta o código para essas funcionalidades.

```
    Private Sub btnCheckIn_Click(sender As Object, e As EventArgs) Handles
btnCheckIn.Click
        Dim oAtivade As New Atividade

        With oAtivade
            .UsuarioID = Session("UsuarioID")
            .RequisitoID = ViewState("RequisitoID")
            .DataInicio = DateTime.Now
            .Tipo = Atividade.TipoAtividade.Desenvolvimento
        End With

        Try
            If oAtivade.Save() Then
                HabilitaDesabilitaBotoes()
                CarregaGridDesenv()
            End If
        Catch ex As Exception
            ShowAlert(ex.Message)
        End Try
    End Sub
```

Listagem 21 – Sub AdicionarFeedback

Após a atividade estar iniciada, em seu momento de conclusão é dado um Check Out. Por meio desse botão, o objeto Atividade já recebe as informações de UsuarioID, RequisitoID, o tipo desenvolvimento e a DataFim recebe a data atual como no inicio de atividade. Após o objeto ser salvo é desabilitado o botão de Check

Out. Sempre após a atividade ser iniciada ou finalizada, o botão referente a essa última ação é desabilitado e também é realizada a atualização do *grid* com as últimas informações.

```
    Private Sub btnCheckOut_Click(sender As Object, e As EventArgs) Handles
btnCheckOut.Click
        Dim oAtivade As New
Atividade(Atividade.VerificaAtividade(Session("UsuarioID"), ViewState("RequisitoID"),
Atividade.TipoAtividade.Desenvolvimento))

        With oAtivade
            .DataFim = DateTime.Now
        End With

        Try
            If oAtivade.Save() Then
                HabilitaDesabilitaBotoes()
                CarregaGridDesenv()
            End If
        Catch ex As Exception
            ShowAlert(ex.Message)
        End Try
    End Sub
```

Listagem 22 – Sub para check out da atividade

5 CONCLUSÃO

Com a realização deste projeto, utilizando conceitos de análise e modelagem de sistemas, foi possível conhecer melhor as ferramentas e metodologias utilizadas, e com os resultados obtidos torna-se possível visualizar prévias de como tudo se comportará no projeto do sistema que será desenvolvido.

Por meio da utilização dos processos e ferramentas destinadas a modelagem UML, torna-se facilmente possível a realização de atualizações, alterações e ajustes, utilizando como base de auxilio tudo que foi modelado anteriormente. Dessa forma, se fez possível a elaboração da documentação do sistema no formato teórico e gráfico, e verificação de quais pontos poderiam apresentar falhas em um processo futuro de desenvolvimento, ou baseado em uma visão inicial do sistema.

O processo de modelagem traz para a equipe uma visualização antecipada do sistema e também a possibilidade de se verificar quais são os pontos principais do sistema e assim dedicar maior atenção para esses pontos base. Dá-se a oportunidade de modificar itens que não ficariam corretos ou pensar outras formas mais eficientes de dar andamento ao projeto. O ponto principal do processo em geral é a visão do produto concluído e suas funcionalidades, levando em consideração quais propósitos e necessidades de usuários ele irá atender.

Com a modelagem prática realizada nesse trabalho, foi possível identificar as principais dificuldades no levantamento de requisitos e processo de modelagem em geral do sistema. Conclui-se que o tempo dedicado ao processo é totalmente necessário, pois o tempo aplicado a essa prática facilita o andamento geral do projeto, e também no decorrer da análise se adquire variados conhecimentos técnicos sobre a modelagem, e específicos sobre o projeto em si.

O aplicativo desenvolvido pode trazer diversos benefícios para a empresa, os desenvolvedores e clientes. O sistema auxilia no gerenciamento do atendimento das solicitações de mudanças solicitadas pelos clientes, quando é manutenção de sistema. Contudo, independentemente se manutenção ou produto novo, o controle do estado de cada solicitação de mudança ou requisitos sendo implementado é importante para o planejamento dos programadores, o acompanhamento do gerente e negociações com clientes.

A análise do *feedback*, que o sistema permite registro, possibilita a obtenção de métricas e na definição de estimativas de prazo e custos.

Durante o processo de desenvolvimento, as principais dificuldades encontradas foram relacionadas às tecnologias utilizadas que foram as versões mais recentes de todos os componentes, interface de desenvolvimento e frameworks. Apesar de os autores deste trabalho possuírem conhecimento e experiência em .NET, nunca haviam trabalhado com a versão 4.5 do *framework* .Net. Também foi optado por utilizar o Bootstrap Framework do qual não haviam conhecimento. Para algumas situações ocorreram problemas que levaram algum tempo para serem resolvidos, como o caso das janelas do tipo *modal*. Em determinados momentos ocorreram problemas de exibição da mesma em tela e de realizar a inserção de dados na base através dos cadastros exibidos em *modal*. Durante o desenvolvimento do sistema foi utilizado, também, o *Entity Framework* para manipular os dados no banco. Por meio do *Entity* os registros são inseridos e trabalhados. Durante o processo de utilização, mesmo com as pesquisas realizadas, foram encontradas situações que não foi possível resolver em tempo como, por exemplo, os vínculos entre tabelas da base com o *framework*. Dificuldades também foram encontradas com a edição e a exclusão de registros, devido ao não conhecimento do *framework*. Assim, optou-se por não utilizar um *framework* externo para trabalhar com a base de dados, removendo o *Entity* do projeto.

Como trabalhos futuros, complementares ao que foi desenvolvido, está a possibilidade de cada cliente acompanhar o andamento da sua solicitação. Com um *login* específico cada cliente, ao acessar o sistema, visualiza somente as suas solicitações e pode acompanhar a evolução de cada uma das suas solicitações. Para isso é necessário criar uma espécie de *workflow* que permita identificar as etapas do processo de atendimento de cada solicitação.

REFERÊNCIAS

BEZERRA, Eduardo. **Princípios de analise e projeto de sistemas UML: um guia pratico para modelagem de sistemas**. 2. ed. Rio de Janeiro: Campus, 2006.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. 7. ed. **UML - guia do usuário.** Rio de Janeiro: Campus, 2000.

DEVMEDIA. **Artigo engenharia de software: introdução a teste de software.** (2014). Disponível em: <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035> Acesso em: 06 fev. 2014.

FOWLER, Martin. **UML essencial: um breve guia para linguagem padrão.** Porto Alegre,Bookman, 3. ed. 2005.

IWEB. **UML.** 2003. Disponível em: <http://www.iweb.com.br/iweb/pdfs/20031008-uml-01.pdf> Acesso em: 03 out. 2014.

KRUCHTEN, Philippe. **Introdução ao RUP:** Rational Unified Process. 2. ed. Rio de Janeiro, RJ: Ciência Moderna, 2004.

MARTINS, José Carlos Cordeiro, **Gerenciando projetos de desenvolvimento de software com PMI, RUP e UML.** 3. ed. Rio de Janeiro: Brasport, 2007.

PASSOS, Antônio. **Sobre casos de uso e suas realizações – parte 3**. 2008. Disponível em: <http://antoniopassos.com/blog/?p=175> Acesso em: 01 nov. 2014.

PRESSMAN, Roger S. **Engenharia de software:** uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

SANTOS. Davi Viana dos. Abordando a Engenharia de Requisitos no MPS.BR. **Revista Engenharia de Software Magazine**, n. 47. Disponível em: <http://www.devmedia.com.br/abordando-a-engenharia-de-requisitos-no-mps-br-revista-engenharia-de-software-magazine-47/24099#ixzz3MZRr3DAq>. Acesso em: 19 dez. 2014.

SME. **Analise estruturada – diagrama de fluxo de dados.** Disponível em: <http://smeduquedecaxias.rj.gov.br/nead/Biblioteca/Forma%C3%A7%C3%A3o%20Continuada/Tecnologia/cursos/programacao/analise%20e%20logica/metodo%20integrado/DFD.pdf> Acesso em: 14 fev. 2014.

SOFTEX. **MPS.BR - Melhoria de Processo do Software Brasileiro. Guia de Implementação – Parte 2: Fundamentação para Implementação do Nível F do MR-MPS**, Brasília: Sociedade Softex, 2011.

SOFTEX**. MPS.BR - Melhoria de Processo do Software Brasileiro. Guia Geral MPS de Software**. Brasilia: Sociedade Softex, 2012.

SOMMERVILLE, Ian. **Engenharia de software**. 9 ed. Pearson Education – BR. 2011.

WAZLAWICK, Raul Sidnei. **Análise e projeto de sistemas de informação orientados a objetos.** 2. ed. Rio de Janeiro: Elsevier, 2011.