Resumo

Este trabalho aborda o desenvolvimento de um Gerenciador Eletrônico de Documentos e informações, utilizando um processo de desenvolvimento baseado no *Unified Process*, práticas de *eXtreme Programming*, prevalência de dados e outras tecnologias de desenvolvimento alternativas. O sistema foi desenvolvido utilizando o *.NET Framework* e a linguagem de programação C#. Os diagramas gerados durante o desenvolvimento, assim como a descrição das tecnologias utilizadas, são apresentadas no trabalho. O Gerenciardor Eletrônico de Documentos e informações tem como objetivo principal proporcionar uma gerência simples dos documentos eletrônicos de uma empresa, e possibilitar que cada documento receba novas informações, identificadas aqui como metadados.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

GEDi	Gerenciador Eletrônico de Documentos e Informações
SAGC	Sistema de Apoio à Gestão do Conhecimento
CEFET-PR	Centro Federal de Educação Tecnológica do Paraná
UTFPR	Universidade Tecnológica Federal do Paraná
SGBD	Sistema Gerenciador de Banco de Dados
XP	eXtreme Programming
CLR	Common Language Runtime
VB.NET	Visual Basic .NET
ASP	Active Server Pages
UP	Unified Process

Sumário

1 Introdução

Um problema comum em empresas e instituições atualmente é a dificuldade de organizar a informação. Todos os funcionários estão habituados a lidar com documentos eletrônicos diariamente, documentos de texto, planilhas eletrônicas, fotos, apresentações multimídia, entre outros.

Quando não existe um controle eficiente dos documentos eletrônicos, esses ficam armazenados nos mais diversos lugares, os usuários perdem documentos dentro da próprio computador, e a empresa perde documentos na rede interna.

Este trabalho descreve a construção de um sistema para Gerência Eletrônica de Documentos e informações, utilizando tecnologias Web. Busca oferecer às empresas que o utilizarem a capacidade de gerenciar seus documentos de forma simples, assim como as informações que podem ser adicionadas a cada documento através do sistema.

Para a realização do trabalho várias novas tecnologias foram utilizadas com o objetivo gerar conhecimento em novas áreas e incentivar a pesquisa. A ForLogic Softwares apoiou o trabalho oferecendo infra-estrutura, acreditando na iniciativa por apostar em novas tecnologias.

Na seção 2 é possível ter uma breve descrição sobre gestão do conhecimento, em que alguns conceitos são explicados. Na seção 3 o contexto que envolve o trabalho é demonstrado, com uma breve visão do Gerenciador Eletrônico de Documentos e Informações (GEDi) e todas as partes envolvidas no seu desenvolvimento. Uma explanação sobre as tecnologias utilizadas durante o desenvolvimento do GEDi pode ser obtida na seção 4.

Na seção 5 a metodologia utilizada no trabalho é apresentada, e os artefatos gerados durante o desenvolvimento da metodologia são apresentados na seção 6. O resultado do desenvolvimento, o produto GEDi, é explicado na seção 7, junto com alguns diagramas de atividade, de sequência e de casos de uso.

Por fim na seção 8 a conclusão é apresentada juntamente com os possíveis trabalhos futuros e objetivos alcançados.

2 Gestão do Conhecimento

Segundo CRUZ (2000), gestão do conhecimento é "Um conjunto formado por metodologias e tecnologias que têm por finalidade criar condições para identificar, integrar, capturar, recuperar e compartilhar conhecimento existente em qualquer tipo de organização."

Logo é possível notar a gestão do conhecimento como uma coleção de processos que governam a criação, a disseminação e a utilização do conhecimento.

2.1 Dados, Informação e Conhecimento

Para entender a gestão do conhecimento é importante frisar a diferença entre dados, informação e conhecimento. Dados podem ser definidos como um conjunto de fatos distintos e objetivos relativos a eventos. Dados são unicamente um registro de um valor pra um determinado item (**??**).

A informação tem por finalidade mudar o modo como o destinatário vê algo, exercer algum impacto sobre seu julgamento e comportamento (**??**). A informação traz o contexto para os dados. Em termos práticos com a informação é possível analisar os dados no contexto em que eles estão descritos.

Uma definição de conhecimento amplamente aceita e utilizada é :

> Conhecimento é uma mistura fluída de experiência condensada, valores, informação contextual e insight experimentado, a qual proporciona uma estrutura para a avaliação e incorporação de novas experiências e informações. Ele tem origem e é aplicado na mente dos conhecedores. Nas Organizações, ele costuma estar embutido não só em documentos ou repositórios, mas também em rotinas, processos, práticas e normas organizacionais (**??**).

Ou seja, o conhecimento se produz em mentes que trabalham, utilizando como combustível a informação (**??**).

2.2 Memória Organizacional

O interesse das organizações sobre a maneira que elas crescem levou-as a buscar uma forma de administrar seus conhecimentos e experiências. Essa busca gerou um meio de acesso e manutenção à parte intelectual da organização, aqui denominado memória organizacional.

Pode-se entender por memória organizacional o local onde as informações da empresa são armazenadas para posteriormente serem obtidas. Memória organizacional não é apenas um servidor de documentos ou um banco de dados, mas sim um sistema que consegue demonstrar o que os funcionários de uma organização sabem.

"Se uma empresa desenvolver sua memória organizacional, poderá utilizar aquelas experiências que tiveram excelentes resultados no passado e, desta forma, evitar que os empregados do presente e do futuro cometam os mesmos erros"(**??**).

3 Contexto do Trabalho

3.1 Sistema de Apoio a Gestão do Conhecimento

O GEDi é parte de um Sistema de Apoio a Gestão do Conhecimento (SAGC), que tem por objetivo fortalecer a troca de conhecimento dentro das empresas, estabelecendo para as mesmas uma memória organizacional, onde as informações e os dados possam ser guardados e recuperados facilmente.

Um sistema com a complexidade do SAGC exige um trabalho exaustivo de uma grande equipe por vários meses. Por essa razão o projeto SAGC foi dividido em vários módulos que serão desenvolvidos separadamente, porém, sempre estabelecendo uma comunicação direta entre eles.

O módulo GEDi é imprescindível para que o auxílio a gestão do conhecimento possa ocorrer, pois é através dele que documentos são inseridos e acessados. Por essa razão ele foi o primeiro módulo do projeto SAGC a ser desenvolvido.

O GEDi foi desenvolvido para ser um sistema independente que possa ser utilizado separadamente, porém, que em um futuro próximo possa tornar-se parte do SAGC.

3.2 ForLogic Softwares

O GEDi é um sistema de propriedade da ForLogic Softwares, empresa que nasceu no Universidade Tecnológica Federal do Paraná (UTFPR), hospedada no Hotel Tecnológico e que após alguns meses tornou-se a primeira empresa legalmente constituída a surgir no Hotel Tecnológico.

A ForLogic Softwares se enquadra no item 7229-0 da Classificação Nacional de Atividades Econômicas, ou seja, Desenvolvimento de Software Sob Encomenda e outras Consultorias em Software, e é uma sociedade limitada por cotas de participação, sendo classificada como uma micro-empresa e com todas as suas obrigações fiscais em dia.

A ForLogic Softwares conta com uma equipe de seis pessoas, composta por dois sócios e quatro colaboradores. Todos trabalham com tecnologia em desenvolvimento de software e buscam

criar soluções inteligentes para os clientes.

3.3 Visão geral do GEDi

O GEDi foi desenvolvido para gerenciar documentos eletrônicos que uma empresa possui e possibilitar que os usuários possam inserir informações adicionais para cada documento.

O objetivo do sistema é permitir a troca de documentos entre os usuários que o utilizam, criando um repositório central de documentos. Com isso é possível centralizar a informação, facilitando backup de arquivos e o acesso de todos os usuários à informação (respeitando as permissões configuradas).

No GEDi todo usuário deve ser autenticado através de login e senha. O acesso ao sistema é feito através de um *browser*, onde o usuário pode inserir e recuperar documentos, assim como registrar informações adicionais para cada documento e navegar entre os grupos a que tem permissão de acesso.

Nas seções 3.3.1, 3.3.2 e 3.3.3, as principais funcionalidades do GEDi serão explicadas com maiores detalhes.

3.3.1 Gerenciamento Eletrônico de Documentos

Através do GEDi, documentos podem ser inseridos e recuperados do sistema por usuários através de um *browser*.Para o usuário enviar um documento ao sistema, deverá fazer *upload* do documento através do sistema Web, indicar a que grupo esse documento pertence e a sua classificação.

O grupo do documento indica a que grupo de usuários o arquivo pertence. Muitas vezes não é interessante que pessoas do grupo “produção” tenham acesso a documentos do grupo “financeiro”. O mesmo também é possível que ocorra na direção oposta, quando o grupo “produção” pode querer privacidade em seus documentos.

A classificação do documento é uma informação que identifica seu conteúdo do documento, como por exemplo, um documento de texto que é inserido e classificado como um “orçamento”. Da mesma forma, uma planilha eletrônica pode ser classificada como “orçamento”, ou mesmo um arquivo .pdf (*Portable Document Format*). Todos podem ser orçamentos enviados por um fornecedor. Dessa forma, a classificação une os arquivos por conteúdo e não pela extensão ou grupo a que pertencem.

Assim que o documento é inserido, este automaticamente ganha uma identificação que será única no sistema. Esta identificação possui o seguinte formato: “GGG-CCC-NNN.RRR.EEE”,

onde GGG representa a sigla do Grupo a que o documento pertence, CCC representa a Sigla de sua Classificação, NNN indica o número sequencial único do sistema, RRR é um número que indica qual a revisão deste documento e EEE indica a extensão convencional do documento. O nome que o documento possuía antes de ser inserido no sistema será armazenado como título deste documento e poderá ser utilizado em pesquisas.

Caso um usuário do grupo "Administrativo" insira um documento denominado "Logotipo.png" no sistema e indique que sua classificação é de "Marketing", o nome deste documento no sistema poderá ser: ADM-MKT-0034.000.png[1].

Para baixar o arquivo, o usuário encontrará o documento desejado através do *browser* e realizará um *download* do mesmo. Com isso, poderá editar o documento localmente e inserí-lo novamente no sistema, como uma nova revisão do documento anterior.

Sempre que um usuário baixar um arquivo, este já estará codificado pelo sistema no modelo "GGG-CCC-NNN.RRR.EEE", como explicado acima. O usuário poderá então enviá-lo[2] novamente para o sistema, e este automaticamente indentificará o arquivo como uma revisão de um arquivo já existente.

3.3.2 Gerência de Metadados

Os metadados são utilizados no GEDi para dar a possibilidade de agregar informações adicionais aos documentos. Com isso é possível inserir informações nos documentos para que os usuários possam realizar buscas mais eficientes e se informar sobre cada documento antes de o utilizar.

Uma figura, por exemplo, pode ser mais que apenas uma imagem. Essa figura pode ser um documento importante para a empresa, como uma planta da sua fundação ou uma foto da primeira filial. Com os metadados a figura pode ter, além do nome e tamanho, informações adicionais, como título, descrição, tipo de imagem (foto, diagrama, planta, gráfico), entre outros que podem ser definidos dentro da empresa. Ninguém melhor que os funcionários da empresa para saber quais metadados são importantes no seu dia a dia.

Assim que um usuário inserir documentos no sistema, poderá preencher os seus metadados. Cada tipo de documento poderá ter metadados diferentes. Por exemplo, uma apresentação multimídia pode ter um metadado "tempo médio de duração" enquanto um "documento de texto" pode ter um metadado "número de páginas".

[1] As siglas podem ser cadastradas e escolhidas pelo administrador do sistema, que supostamente neste caso escolheu ADM para o grupo "Administrativo" e MKT para a classificação "Marketing".

[2] Para isso o usuário deve possuir permissão de editor ou revisor, explicadas na seção 3.3.3

3.3.3 Controle de Usuários e Permissões

Para organizar os usuários e suas permissões existe uma Gerência de Grupos, onde um grupo pode ter vários subgrupos organizados em modelo de árvore.

Foram criados também papéis para definir as permissões dos usuários. Existem quatro papéis que um usuário pode assumir, são eles: Visitante, Editor, Revisor e Gerente.

O papel de "Visitante" proporciona ao usuário a capacidade de recuperar arquivos do grupo ao qual ele possui este papel. O "Editor" pode recuperar os arquivos e inserir novos documentos e revisões. O "Revisor" pode apenas recuperar documentos do sistema e inserir revisões[3], e o "Gerente" tem permissão para recuperar, inserir documentos e revisões e remover arquivos do sistema.

Os usuários podem pertencer a vários grupos diferentes, com diferentes papéis em cada grupo. Por exemplo, um usuário pode ser visitante no grupo "administrativo" e ao mesmo tempo ser editor no grupo "desenvolvimento". Isso agrega uma grande flexibilidade para que usuários possam exercer diferentes papéis em vários grupos.

Quando um usuário não tem permissão nenhuma em um grupo, ele não poderá visualizá-lo no sistema, e consequentemente, não poderá acessar os arquivos deste grupo.

[3]Uma revisão é um arquivo obtido do sistema que foi alterado e novamente inserido.

4 Tecnologias Utilizadas

O GEDi foi construído com algumas novas tecnologias de desenvolvimento de software, estas são explicadas em 4.1, 4.2 e 4.3.

4.1 Prevalência de Objetos

A prevalência de objetos é uma alternativa aos sistemas Gerenciadores de Banco de Dados relacionais, que possibilita que objetos serializáveis possam ser utilizados de forma persistente.

Com a utilização do conceito de orientação a objetos, trabalhar com um de banco de dados relacional acarreta um trabalho extra, pois todo o modelo de objetos definido deve ser mapeado para campos nas tabelas do banco de dados relacional.

A prevalência de objetos escrita pelo brasileiro Klaus Wuestelfield está baseada em um padrão de projeto denominado *"Command"* (**??**), que possibilita executar transações e impede que uma falha de software ou hardware deixe o sistema em um estado inconsistente. O *Prevayler*, como foi denominado por seu criador, foi desenvolvido inicialmente para plataforma Java e demonstrou em testes realizados(**??**), uma velocidade maior que bancos de dados como Oracle e MySQL, que estão atualmente entre os bancos de dados relacionais mais rápidos e utilizados do mercado.

A justificativa para a não utilização de um Sistema Gerenciador de Banco de Dados (SGBD), pode ser encontrada na dificuldade em se desenvolver software utilizando o conceito de orientação a objetos em conjunto com o modelo de banco de dados relacional. Na orientação a objetos, os dados são organizados de forma diferente de dados armazenados no modelo relacional. Esta situação exige grande esforço dos desenvolvedores para trabalhar com os dois modelos e ainda fazer o mapeamento, ou seja, levar os dados do paradigma orientado a objetos para o modelo relacional e vice-versa.

Em um banco de dados relacional, para obter informações o sistema realiza uma consulta em tabelas que estão no disco rígido para trazê-las para a memória. Utilizando a prevalência de objetos isso não acontece, pois as informações estão disponíveis o tempo todo em objetos na memória. Uma grande vantagem desse modelo é a velocidade de acesso. Existe o risco, porém, de o sistema

perder performance por uso excessivo de memória, já que o volume de informações na memória é substancialmente maior.

A ferramenta utilizada para prevalência de objetos no .NET é o Bamboo.Prevalence, desenvolvido na linguagem de programação C# pelo também brasileiro Rodrigo Bamboo e muito semelhante ao Prevayler (**??**).

4.2 eXtreme Programming

A eXtreme Programming[1] (XP) é uma metodologia de desenvolvimento ágil *(Agile Development)* que demonstra resultados muito bons em projetos onde o tamanho da equipe é reduzido (entre 1 e 15 pessoas) (**??**). Por essa razão a XP foi escolhida para auxiliar o desenvolvimento deste projeto, já que a equipe de desenvolvimento do projeto é pequena. Nas subseções de número 4.2.1 a 4.2.12 serão apresentadas de forma concisa as 12 práticas utilizadas pela XP, ressaltando a importância da sinergia que conecta estas práticas e demonstrando por que a XP é uma metodologia interessante, que busca construir softwares de alta qualidade em pequenos projetos, com requisitos que podem mudar (e geralmente mudam) constantemente.

4.2.1 Jogo do Planejamento (*Planning Game*)

A Prática do Jogo do Planejamento é uma negociação entre o time do negócio (geralmente clientes) e o time do desenvolvimento (os desenvolvedores) onde o objetivo é que ambos os times ganhem. No Jogo do Planejamento é definida basicamente uma *release* (o que vai compor essa *release* e a sua data de entrega).

O time do negócio define o que é importante ser feito, ou seja, quais requisitos devem ser cumpridos. Cabe ao desenvolvimento explicar o impacto de cada decisão do negócio, e mostrar caminhos alternativos que possam ser trilhados. Por exemplo, é importante mostrar ao time do negócio que uma lista de aniversariantes do mês só faz sentido após o cadastro dos funcionários.

4.2.2 Entregas Frequentes (*Small Releases*)

Diferente do que é feito normalmente, a XP define que, quanto mais rápido o sistema entrar em produção (funcionamento no cliente) melhor. “Coloque um sistema simples rapidamente em produção e depois libere novas versões em ciclos curtos” (**??**).

[1] O termo “*eXtreme Programming*” é escrito dessa forma, com as letras “X” e “P” maiúsculas para representar a sigla “XP”, seguindo a nomenclatura usada por livros técnicos da área.

Uma *release* é composta por um conjunto de Story Cards[2]. Uma boa *release* deve ser significativa, ou seja, fazer diferença para o cliente, agregando novas funcionalidades ou corrigindo problemas.

4.2.3 Metáfora (*Metaphor*)

A Metáfora é uma visão do sistema que deve ser compartilhada por todos os envolvidos no projeto incluindo os clientes. Por esse motivo, não deve ser abarrotada de termos técnicos. "(A metáfora) Ajuda os participantes a entender os elementos básicos (do sistema) e seus relacionamentos" (**??**). A metáfora escolhida no início do sistema pode evoluir com o amadurecimento do projeto.

4.2.4 Projeto Simples (*Simple Design*)

Um projeto simples sempre leva menos tempo para ser concluído que um projeto complexo (**??**). A prática do Projeto Simples diz que devemos desenvolver o que precisamos hoje, deixando de lado todo tipo de flexibilidade adicional que não é exigida pelo sistema no momento.

4.2.5 Refatoração (*Refactoring*)

Ao adicionar uma nova funcionalidade no sistema, os desenvolvedores devem se perguntar se alguma parte do código já existente no sistema pode ser melhorada para que a adição dessa nova funcionalidade seja mais simples. Ao terminarem uma *release*, os desenvolvedores se voltam novamente para o código para tentar simplificá-lo, e fazem isso repetidamente. "São possíveis melhorias, incluindo simplificações, otimizações, aumento de clareza, flexibilidade adicional, etc" (**??**).

4.2.6 Testes (*Testing*)

Toda funcionalidade do sistema deve possuir um teste automatizado que possa ser executado a qualquer momento. Isso possibilita que os desenvolvedores tenham confiança para simplificar o código e realizar a Refatoração. Os clientes também devem escrever seus testes de aceitação, para as funcionalidades que eles esperam do sistema. Desta forma, sempre que todos os testes estiverem executando corretamente o sistema estará estável.

[2]Story Cards: Mais detalhes são apresentados na seção 5.3.1

4.2.7 Programação em Pares (*Pair Programming*)

Todo o código é criado por duas pessoas ao mesmo tempo utilizando o mesmo computador. A prática da Programação em Pares adiciona qualidade ao código gerado e tem a característica de aumentar a comunicação entre as pessoas envolvidas. Enquanto o desenvolvedor que mantém o controle (teclado e mouse) está focado de um modo mais específico na implementação, o seu par pode estar pensando que problemas poderão resultar das decisões que forem tomadas, auxiliando na correção dos erros simples de implementação, tirando dúvidas do seu par quando solicitado e dando suas opiniões quando for conveniente.

4.2.8 Propriedade Coletiva (*Collective Code Ownership*)

Todos os desenvolvedores podem alterar o código a qualquer momento, sem a necessidade de pedir permissão. Dessa forma, sempre que alguém sentir que pode melhorar o código (*refactoring*), ele tem a obrigação de fazê-lo (**??**). É comum que algum desenvolvedor tenha maior domínio sobre uma parte do sistema que outros, mas todos sabem o suficiente para poder alterar qualquer parte. Isso elimina os “donos do código”, diminuindo os riscos para a empresa (caso algum funcionário se desligue da organização) e aumentando a capacidade de desenvolvimento de toda equipe.

4.2.9 Integração Contínua (*Continuous Integration*)

Cada nova funcionalidade desenvolvida é integrada ao sistema no momento em que fica pronta. Quando um software é desenvolvido por uma equipe, existe a necessidade de se ter uma “versão principal” do sistema. Essa versão principal é atualizada sempre que as equipes terminam alguma funcionalidade, com seus respectivos testes.

4.2.10 Semana de 40 Horas (*Sustainable Pace*)

Esta prática consiste em, não trabalhar em excesso. "Ninguém faz seu melhor trabalho quando está estressado, sob pressão ou cansado"(**??**). Por uma semana, trabalhar algumas horas extras pode ser aceitável, mas se isso se repete nas próximas semanas, algo no projeto está errado.

4.2.11 Cliente Presente (*On-site Customer*)

Esta é uma prática difícil mas de grande valia. “Um usuário do sistema deve ser incluído como parte do time e estar disponível a qualquer momento para responder perguntas” (**??**).

4.2.12 Padrões de Codifiação (*Coding Standards*)

Seu principal objetivo é apoiar a Propriedade Coletiva através de um conjunto de padrões de código. Todos os membros da equipe devem concordar voluntariamente em seguir estes padrões (**??**).

4.3 .NET Framework - Projeto Mono

O projeto GEDi foi desenvolvido utilizando a tecnologia .NET, que é uma plataforma de desenvolvimento de sistemas orientada a objetos que beneficia a reutilização de código e é altamente produtiva. Mais detalhes do desenvolvimento são apresentados nas seções 4.3.1 e 4.3.2.

4.3.1 .NET Framework

O .NET Framework é uma plataforma criada pela Microsoft em 2000 para desenvolver e executar aplicações. Utiliza o conceito de "máquina-virtual", que traz muitas vantagens aos desenvolvedores, tais como o código gerenciado, coletor de lixo, biblioteca de classes, e entre outras (**??**).

É possível entender que o .NET Framework consiste basicamente de um ambiente de execução chamado *Common Language Runtime* (CLR) e uma biblioteca de classes que oferece uma plataforma de desenvolvimento rica que pode ser explorada por uma variedade de linguagens e ferramentas (**??**).

Uma grande vantagem do .NET Framework é a possibilidade do desenvolvimento multi-linguagens, o que significa que uma classe escrita em Visual Basic .NET (VB.NET) pode ser utilizada em um projeto que utiliza a linguagem C# [3], C++, Boo, ou outras. Essa vantagem possibilita que todas estas linguagens, por compartilharem o .NET Framework, possam utilizar a mesma biblioteca de classes, e que componentes escritos em uma linguagem possam ser utilizados normalmente por um aplicativo escrito em outra.

4.3.2 Projeto Mono

O projeto Mono tem por objetivo criar uma versão multi-plataforma *Open Source* do .NET Framework da Microsoft. Criado por Miguel de Icaza, criador do Gnome[4] e fundador da Ximian[5], o Mono atualmente é desenvolvido por uma vasta comunidade de desenvolvedores ao redor do

[3] C# lê-se: Cê Sharp

[4] Gnome: Suite Desktop para Linux e Unix (http://www.gnome.org/)

[5] Ximian: Empresa especializada em desenvolvimento de software para Linux incorporada recentemente pela Novell.

mundo, e ainda é apoiado financeiramente pela Novell[6], que financia programadores para implementar as bibliotecas de classe do .NET para o Mono mantendo a compatibilidade.

É possível no Mono, assim como no .NET, desenvolver softwares utilizando várias linguagens de programação, como C#, Python, Java, Vb.Net, Boo, entre outras (**??**). Grande parte do sucesso do Mono é atribuído ao número de desenvolvedores e comunidades espalhadas pelo mundo que estão colaborando com o projeto. No Brasil podemos citar algumas comunidades como MonoBrasil e a Simios.org, que reunem pessoas ligadas ao Mono de alguma forma.

4.4 ASP.NET

O ASP.NET é uma arquitetura que pode ser utilizada nas duas plataformas de desenvolvimento, tanto no .NET da Microsoft, quanto no Mono. A plataforma de desenvolvimento fornece ao ASP.NET uma biblioteca de classes específica para criar aplicações Web, com componentes visuais e outros facilitadores. O ASP.NET introduz inovações que permitem a separação mais fácil da regra da aplicação de sua forma de apresentação (**??**).

Derivada do *Active Server Pages* (ASP), traz algumas modificações interessantes. O ASP.NET é parte do .NET Framework e proporciona muitos benefícios para o desenvolvimento Web, tais como, coletor de lixo, a integração de componentes escritos em diferentes linguagens e o tratamento de exceções (**??**).

4.5 Testes Unitários

Um teste unitário é um pedaço de código escrito por um desenvolvedor para testar uma área muito pequena e específica de uma funcionalidade que esteja sendo implementada (**??**). É importante que todas as pequenas partes do sistemas sejam testadas separadamente. Estes testes são automatizados e podem ser escritos na mesma linguagem em que o sistema está sendo desenvolvido.

Existem ferramentas que automatizam estes testes, e possibilitam que as verificações de cada funcionalidade possam ser feitas de forma clara e simples, visualmente, através de uma interface amigável. Este é o caso do NUnit [7], utilizado neste projeto e escrito para testar funcionalidades nos trechos de código .NET.

[6]Novell: http://www.novell.com/pt-br

[7]NUnit: ferramenta para facilitar os testes unitários automatizados, derivado do JUnit (http://www.nunit.org).

É possível visualizar na Figura 1 uma tela do NUnit, cada círculo representa um teste ou uma classe de testes. Os círculos em verde (cinza claro) representam os testes que executaram corretamente e os círculos em vermelho (cinza mais escuro) representam os testes que apresentaram problemas na sua execução.

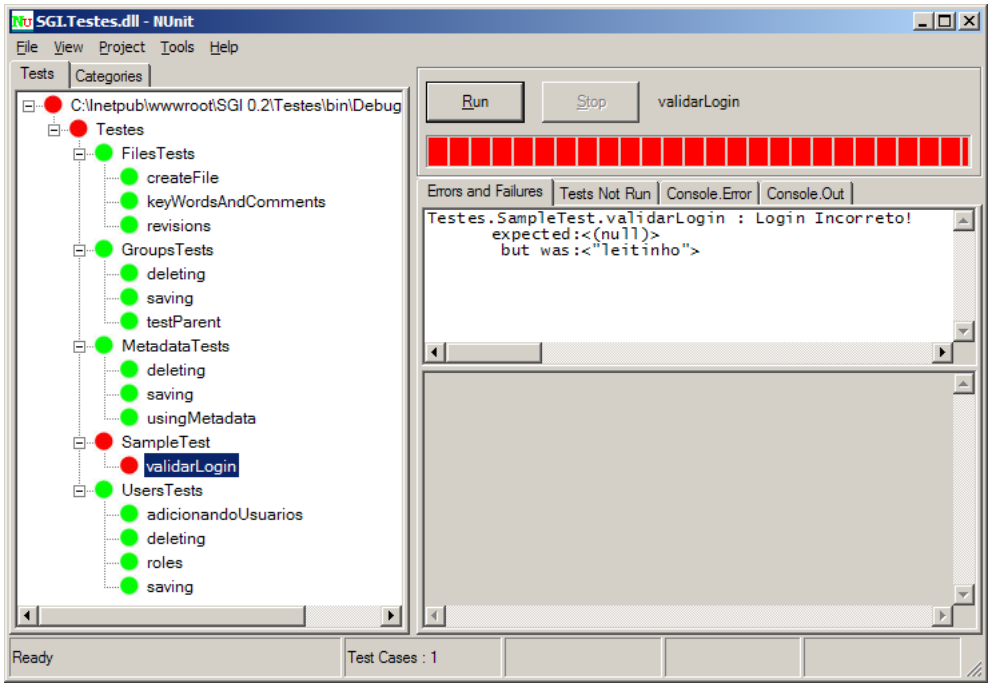

Figura 1: NUnit - Captura de Tela

Para utilizar o NUnit é necessário utilizar o seu framework de testes o NUnit.Framework. Porém os testes são tecnicamente simples de serem escritos e o NUnit.Framework proporciona um conjunto de funcionalidades que facilitam a relaização de testes e possibilitam testar um grande número de situações.

Na figura 2 é possível observar que alguns atributos são adicionados antes de cada método, sempre entre colchetes. O atributo [Test] indica que o método que o segue é um teste, e este será executado pelo NUnit.Framework, o atributo [TestFixtureSetUp] sinaliza que o método será executado antes que o primeiro teste da classe for chamado.

Ao colocar o atributo [SetUp] antes de um método, este será chamado sempre antes de cada novo método marcado com [Test].

A Tabela 1 demonstra a função dos principais atributos utilizados para realizar testes com o NUnit.

```
using System;
using NUnit.Framework;

namespace Testes
{
    [TestFixture]
    public class SampleTest
    {
        [TestFixtureSetUp]
        public void testFixtureSetup()
        {
            // Todo código aqui é executado na primeira vez
            // que algum teste da classe é realizado.
        }

        [SetUp]
        public void setup()
        {
            // Todo código aqui é executado antes de cada
            // teste específico.
        }

        [Test]
        public void saving()
        {
            // Aqui vem o código que irá realizar
            //o teste especificamente
        }
    }
}
```

Figura 2: Codificação com NUnit - Exemplos

Tabela 1: Principais atributos do NUnit Framework.

Atributo	**Função**
[TestFixture]	Marca uma classe que contenha testes
[Test]	Marca um método (que esteja em uma classe marcada por [TestFixture]) como um método de teste.
[SetUp]	Indica que método será chamado sempre antes de um teste ser iniciado
[TearDown]	Indica que método será chamado sempre antes de um teste ser terminado
[TestFixtureSetUp]	Marca um método que será chamado antes dos testes da classe (marcada por [TestFixture]) serem executados.
[TestFixtureTearDown]	Marca um método que será chamado após todos os testes da classe (marcada por [TestFixture]) serem executados.

5 *Descrição do Desenvolvimento da Metodologia*

O processo de desenvolvimento que foi utilizado é similar ao UP (Unified Process), utilizando quatro fases: Concepção; Elaboração; Construção; Transição. As atividades também semelhantes as do UP são: Requisitos; Projeto; Implementação e Testes.

É importante lembrar que o processo utilizado foi reduzido e simplificado para corresponder a uma equipe de desenvolvimento pequena. Um amostra disso foi a junção no processo utilizado das atividades de Projeto e Análise do UP em uma única atividade denominada Projeto.

Algumas adaptações foram feitas no processo durante seu desenvolvimento. Por isso existem algumas diferenças sutis entre o processo apresentado na proposta e o utilizado. Mais sobre a alteração do processo poderá ser visto na seção 5.1.

Um ciclo de vida iterativo e incremental foi utilizado, onde cada fase teve o número de iterações necessárias para que o produto (software e documentação) estivesse maduro o suficiente para que a próxima fase pudesse ser iniciada. Essas iterações sempre aconteceram incrementando os produtos já criados anteriormente.

A equipe de desenvolvimento reduzida do projeto foi uma razão para serem utilizadas práticas de eXtreme Programming, procurando aumentar a qualidade e a eficiência no desenvolvimento do software (**??**).

5.1 Adaptações do Processo

O processo utilizado é o processo que a ForLogic Softwares utiliza (até o momento) para o desenvolvimento de sistemas na empresa. A cada novo projeto, o processo sofre pequenas adaptações e passa por alterações que buscam sua melhoria.

Durante o desenvolvimento do GEDi, não foi diferente: o processo passou por pequenas alterações que em algumas situações uniu duas atividades em uma, ou em outros casos novas atividades foram criadas e como consequências, alguns novos artefatos surgiram.

As Figuras 3 e 4 podem ser observadas para comparar as alterações sofridas pelo processo

durante o desenvolvimento do projeto.

	Concepção	Elaboração	Construção	Transição
Requisitos	-Esp. de Requisitos -Avaliar Riscos -Glossário Comum -Projetar Casos de Uso -Projetar Interfaces de Usuário	- Definição de Release(cronograma)	- Mudança ou Novos Requisitos	-Validação da Proposta Inicial
Análise e Projeto	- Atacar Riscos Funcionais -Especif. Casos de Uso	- Projetar Diagr. de Classe - Projetar Arquitetura do SW - Modelagem do Banco de Dados	- Revisão do Diagrama de Classes - Revisão da Modelagem do Banco de Dados	
Implementação	- Atacar Riscos Técnicos	- Geração de Código	- Testes Unitários - Implementação - Refactoring	- Refactoring - Implementação -Gerar Instalação
Testes	- Avaliação dos Requisitos	- Validação do Código Gerado - Correções na Análise	-Teste Funcional -Testar Interface	- Teste de Integração - Implantação do Sistema

Figura 3: Processo apresentado na proposta

	Concepção	Elaboração	Construção	Transição
Requisitos	-Esp. de Requisitos -Avaliar Riscos -Projetar Casos de Uso -Projetar Interfaces de Usuário	- Definição das User Stories - Definição de Release	- Atualização de Requisitos	-Validação da Proposta Inicial
Análise e Projeto	- Tratar Riscos -Especif. Casos de Uso	- Projetar Diagr. de Classe - Projetar Arquitetura do SW - Modelagem do Banco de Dados	- Revisão do Diagrama de Classes - Revisão da Modelagem do Banco de Dados	
Implementação		- Geração de Código	- Testes Unitários - Implementação - Refactoring	- Refactoring - Implementação -Gerar Instalação
Testes	- Avaliação dos Requisitos	- Correções na Análise	-Teste Funcional	- Teste de Integração - Implantação do Sistema

Figura 4: Processo utilizado durante o trabalho

5.2 Concepção

5.2.1 Especificar Requisitos

"Requisitos são (ou deveriam ser) a base para todos os sistemas. São essencialmente uma indicação de o que o sistema deve fazer" (**??**).

A especificação de requisitos inicial foi feita baseada no conhecimento do problema, mais as informações coletadas de possíveis usuários e pessoas que já conheciam um gerenciador eletrônico de documentos.

Um documento foi criado contendo os requisitos e suas especificações. Este foi utilizado posteriormente para revisão dos propósitos do desenvolvimento e entendimento do sistema como um todo. Neste documento, além dos requisitos o escopo do sistema também foi definido, procurando ilustrar claramente qual seria o objetivo do GEDi, seu publico alvo, características e funcionalidades. Os requisitos estão listados na seção 6.1.

5.2.2 Avaliar Riscos

Vários riscos para o desenvolvimento do projeto foram identificados e avaliados. Sua grande maioria foi estudada no início do projeto para viabilizar todo o desenvolvimento.

Uma importante tarefa na gerência do projeto é antecipar os riscos que podem afetar o cronograma do projeto ou a qualidade do software que está sendo desenvolvido, e assim tomar uma atitude para evitar esses riscos (**??**).

5.2.3 Projetar Casos de Uso

Os casos de uso são utilizados durante todo o desenvolvimento do sistema. O Diagrama de Casos de Uso tem por objetivo descrever um modelo funcional de alto nível do sistema, procurando identificar os usuários e representar o sistema segundo a sua visão (**??**).

Os casos de uso projetados nesta fase de concepção foram criados com base no levantamento inicial de requisitos, porém, sofreram alterações durante todo o desenvolvimento do projeto. Os diagramas de caso de uso podem ser vistos durante toda a seção 7.

5.2.4 Projetar Interfaces de Usuário

O objetivo de projetar as interfaces no início do projeto foi encontrar novos requisitos e riscos. Foi possível notar claramente que muitas características e funcionalidades do sistema surgiram logo após o protótipo das interfaces.

As interfaces foram modeladas inicialmente para serem descartadas, e isso foi o que realmente ocorreu. O principal objetivo da prototipação das interfaces foi demonstrar quais seriam as funcionalidades de cada uma das partes do sistema, e como cada uma deveria funcionar.

Mesmo com os protótipos iniciais descartados, as interfaces seguintes tiveram diversas alterações desde sua criação até o resultado final.

5.2.5 Tratar Riscos

É possível pensar nos riscos como uma probabilidade de que alguma circunstância adversa possa ocorrer. Os riscos podem ameaçar o projeto e o software que está sendo construído (**??**). Identificar e minimizar os riscos é uma atividade complexa que foi tratada desde o inicio do projeto. Os maiores riscos avaliados estavam relacionados à falta de conhecimento nas novas tecnologias a serem utilizadas.

No início do projeto, nenhum dos colaboradores da ForLogic Softwares tinha experiência com desenvolvimento em ASP.NET. Havia então a necessidade de desenvolver aplicações que servissem de modelo para que a tecnologia fosse entendida pela equipe.

Outra tecnologia pouco conhecida até o início do projeto é a prevalência de dados. Existia o risco de algo inesperado inviabilizar o desenvolvimento do sistema, principalmente pelo fato da equipe ainda não ter testado a prevalência de dados juntamente com o ASP.NET. A solução foi criar aplicações que utilizassem tais tecnologias.

Uma das aplicações criadas foi o website da ForLogic Softwares, que foi re-desenvolvido como experimento para que o ASP.NET e a prevalência de dados fossem testados em conjunto. Toda a gerência de logs e downloads foi criada utilizando prevalência, e isso forneceu a credibilidade necessária para que estas tecnologias fossem utilizadas no projeto. Atualmente, o website ainda possui esse sistema de logs funcionando sobre prevalência de dados sem problemas.

5.2.6 Especificar Casos de Uso

Cada caso de uso pode ser especificado usando uma linguagem simples e natural. Isto ajuda os projetistas a identificar objetos no sistema e dá a eles uma compreensão do que o sistema pretende fazer (**??**).

Os casos de uso foram especificados para que em caso de dúvida a respeito de sua finalidade, essa dúvida pudesse ser esclarecida pela sua especificação. Essa especificação ajuda no entendimento do caso de uso por parte do desenvolvedor, explicando melhor a funcionalidade do sistema.

5.2.7 Avaliar Requisitos

Após a especificação dos casos de uso, os requisitos foram avaliados e estudados para verificar se realmente estavam condizentes com o objetivo do sistema.

A lista de requisitos foi atualizada, e as suas prioridades, riscos e descrições também, segundo a percepção da equipe de desenvolvimento.

5.3 Elaboração

5.3.1 Definir *Story Cards*

Os *Story Cards* são uma forma de documentar e estimar as funcionalidades do sistema (**??**). Os usuários do sistema escrevem e priorizam os *Story Cards*, e os desenvolvedores estimam quanto tempo cada funcionalidade levará para ser implementada, o que possibilita que seja feito um planejamento do desenvolvimento do sistema.

Cada *Story Card* possui um título, uma descrição, um número de identificação, o nível de prioridade, o tempo estimado, e pode conter um ou mais requisitos para o sistema a ser desenvolvido (ver a seção 6.1).

No desenvolvimento do GEDi, os *Story Cards* foram separados de acordo com as suas prioridades, formando grupos que representaram *releases* do sistema. Foi possível acompanhar o progresso do desenvolvimento conforme os *Story Cards* foram sendo concluídos (implementados). Um exemplo de *Story Card* pode ser visto na Figura 5.

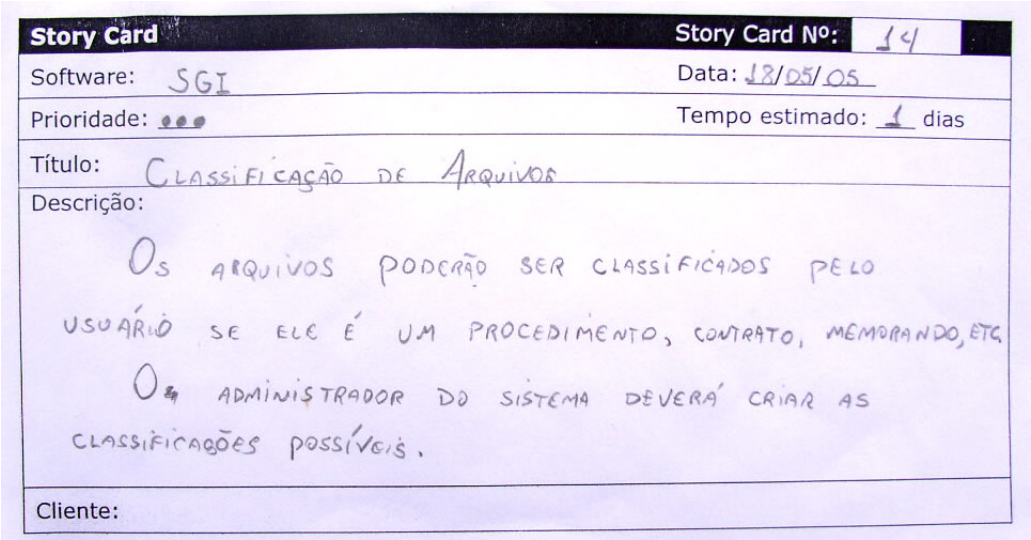
Story Card — Story Card Nº: 14
Software: SGI — Data: 18/05/05
Prioridade: ●●● — Tempo estimado: 1 dias
Título: CLASSIFICAÇÃO DE ARQUIVOS
Descrição:
OS ARQUIVOS PODERÃO SER CLASSIFICADOS PELO USUÁRIO SE ELE É UM PROCEDIMENTO, CONTRATO, MEMORANDO, ETC.
O ADMINISTRADOR DO SISTEMA DEVERÁ CRIAR AS CLASSIFICAÇÕES POSSÍVEIS.
Cliente:

Figura 5: Exemplo de Story Card

5.3.2 Definir Release

Uma *release* é uma versão do sistema com novas funcionalidades suficientes para ser entregue aos usuários pelo grupo de desenvolvimento (**??**).

Durante o projeto, várias pequenas *releases* foram geradas. Com isso, o sistema pôde ser testado e a *release* seguinte sempre implementava o que propunha, mais as alterações necessárias identificadas na *release* anterior.

5.3.3 Projetar Diagramas de Classes

O diagrama de classes demonstra a estrutura estática de um sistema. As classes representam objetos manipulados pelo sistema (**??**).

Os diagramas de classes começaram a ser projetados nesta fase de Elaboração, para representar graficamente o que seria desenvolvido, facilitando a visualização do sistema.

Durante todo o desenvolvimento estes diagramas de classes foram modificados, acompanhando as necessidades de implementação encontradas e os novos requisitos identificados no decorrer do projeto.

5.3.4 Projetar Arquitetura do Software

O software foi projetado em camadas com ênfase na aplicação da orientação a objetos, o que tornará simples a utilização de diversas interfaces de usuário e possibilitará customizações rápidas ao sistema (**??**).

A utilização de "sub-sistemas" (camadas) é recomendável para gerenciar dependências dentro de um projeto, trazendo benefícios à equipe de desenvolvimento (**??**).

Criado em paralelo com o diagrama de classes, a arquitetura do software representa as atribuições (responsabilidades) dos componentes de software que compõem o GEDi. Cada camada do sistema possui uma interface de acesso bem definida e não deve interferir no funcionamento de outra camada, o que garantirá maior integração e independência às camadas do sistema desenvolvido.

5.3.5 Geração de Código

A geração de código no projeto aconteceu através da ferramenta de modelagem Microsoft Visio, que possibilita, além da modelagem dos diagramas, a geração do código em linguagem de programação.

Isso facilitou muito o trabalho inicial, pois o no começo o trabalho é basicamente codificar as classes representadas pelo diagrama de classes, o que não foi necessário devido ao Microsoft Visio possuir essa funcionalidade.

Porém muitos pequenos problemas tiveram de ser corrigidos, já que a geração de códigos possui algumas limitações e não existia experiência da equipe para configurar o Microsoft Visio.

5.3.6 Correções na Análise

Após o código gerado e os diagramas já projetados, uma revisão foi realizada nos diagramas gerados até então na análise e seus pontos falhos foram corrigidos. Muita coisa foi corrigida nesta fase, porém a grande maioria das alterações ocorreu durante a fase de construção.

Isso aconteceu pela pouca experiência da equipe que não foi capaz de modelar exatamente o que o sistema deveria fazer, e também devido aos requisitos que surgiram no decorrer do projeto.

5.4 Construção

5.4.1 Atualização de Requisitos

Durante a fase de construção, muitos requisitos novos surgiram e muitos deles foram deixados para versões futuras do GEDi, por estarem fora do escopo definido inicialmente para o sistema. No entanto, alguns requisitos identificados possuíam grande relevância e não puderam ser descartados nem adiados. Com isso, houve a necessidade de documentar esses novos requisitos.

O documento de requisitos foi alterado, refletindo as modificações encontradas, e alguns *Story Cards* foram alterados e adicionados ao grupo da release pertinente.

5.4.2 Revisão dos Diagramas de Classes

Com o surgimento ou correção de requisitos, os diagramas de classex consequentemente sofreram alterações. As revisões dos diagramas de classes ocorreram diversas vezes no desenvolvimento, e não foram encaradas como um problema, mas como o curso natural de evolução do projeto.

Sempre que uma inconsistência era encontrada ou alguma funcionalidade era alterada, o diagrama de classes correspondente era revisado, refletindo as alterações.

5.4.3 Testes Unitários

Toda a fase de construção foi acompanhada pela implementação de testes unitários [1].

Individualmente, os componentes são testados para garantir que eles operem corretamente. Cada componente é testado independentemente, sem outros componentes do sistema (**??**).

Com isso, cada funcionalidade pôde ser testada separadamente de forma automatizada, aumentando muito a segurança dos desenvolvedores que poderiam executar os testes a qualquer

[1] Mais informações sobre Testes Unitários na seção 4.5

momento, verificando o resultado de suas alterações sobre cada funcionalidade já implementada anteriormente.

5.4.4 Implementação

A implementação do sistema foi feita utilizando várias práticas de eXtreme Programming, procurando aumentar a qualidade do código gerado. Além de testes unitários, práticas da XP como Projeto Simples, Propriedade Coletiva, Refatoração, Programação em Pares e Entregas Freqüentes foram aplicadas.

Por utilizar orientação a objetos, muito do que foi codificado pôde ser reaproveitado em outras partes do sistema.

Durante todo o projeto, muitos componentes ASP.NET foram criados, permitindo a utilização dos mesmos em outras partes do sistema e também em projetos futuros, ligados ou não ao GEDi, aumentando o reaproveitamento de código.

5.4.5 Refatoração *(Refactoring)*

Dentre todas as práticas de XP aplicadas neste projeto, a refatoração foi uma das mais utilizadas, obtendo resultados muito satisfatórios. Porém, a refatoração constante é uma prática difícil de ser aplicada por exigir disciplina e disposição.

Com a refatoração, o código esteve sempre limpo e fácil de ser entendido, e as novas alterações realizadas se tornaram mais simples.

5.4.6 Teste Funcional

Mesmo com o sistema utilizando testes unitários, testes funcionais com usuários ainda foram necessários para identificar problemas e apontar alterações.

Um teste funcional foi realizado para coletar sugestões e identificar problemas. Este foi feito com documentos utilizados no dia a dia da ForLogic Softwares, tais como: orçamentos, propostas comerciais, documentos de padrões de codificações, logotipos da empresa, entre outros.

Algumas pequenas alterações foram realizadas por razão deste teste funcional, porém muitas sugestões foram armazenadas para as futuras versões do GEDi.

5.5 Transição

O GEDi é um produto da ForLogic Softwares, não um software desenvolvido sob encomenda para alguma empresa, portanto, até o momento nenhuma unidade deste produto foi comercializada. Por esse motivo não existe ainda um cliente que possua esse produto implantado na sua empresa, o que deixou a fase de transição com um baixo nível de tarefas executadas.

Uma versão de testes foi utilizada na própria empresa desenvolvedora (ForLogic Softwares), e uma versão piloto deverá ser implantada em outras organizações.

A tarefa de validação da proposta inicial foi realizada pela ForLogic Softwares, e foi possível notar que as funcionalidades levantadas nos requisitos foram todas atendidas pelo sistema, e algumas novas idéias foram identificadas e deixadas para novas versões.

6 *Artefatos Gerados durante o desenvolvimento*

Esta seção expõe os artefatos gerados durante o desenvolvimento da metodologia, vale porém ressaltar que diagramas de atividade, seqüência e caso de uso, também gerados durante o desenvolvimento da metodologia, são apresentados na seção 7 para facilitar sua compreensão.

6.1 Requisitos

Os requisitos do sistema estão listados na Tabela 2. A coluna "ID" é uma identificação única para cada requisito; a coluna "Descrição" é um resumo do requisito; a coluna "*Story Card*" é uma ligação dos requisitos com os números de identificação dos *Story Cards*; a coluna "Prioridade" representa a ordem que os requisitos devem ser implementados[1]; e a coluna "Risco", é o nível de risco levantado pela equipe para implementar este requisito.

Tabela 2: Tabela de Requisitos

ID	**Descrição**	**Story Card**	**Prioridade**	**Risco**
R1	O sistema deve possibilitar aos usuários com permissão inserir arquivos no sistema.	2	***	Alto
R2	Através do sistema os usuários com permissão devem conseguir recuperar um arquivo específico.	2	***	Alto
R3	Os arquivos armazenados poderão ser exibidos por diferentes visões do sistema. Dando opção para o usuário alterar entre as maneiras de visualizar os arquivos.	3	***	Médio
continua na pagina seguinte...				

[1] "*" representa baixa prioridade, "**" representa prioridade média, e "***" representa prioridade alta.

ID	Descrição	Story Card	Prioridade	Risco
R4	Sempre que um documento for inserido no sistema ele deverá receber um código de identificação único.	4	***	Baixo
R5	Os usuários terão permissão de acesso ao sistema seguindo sempre a permissão do seu grupo, ou ainda cada usuário poderá ter permissões específicas para ele.	5	***	Alto
R6	Todo usuário deverá ter login e senha para acessar ao sistema.	5	***	Baixo
R7	Através de uma gerência o administrador poderá gerenciar grupos, usuários, tipos de arquivo, classificações, extensões e metadados.	7	***	Baixo
R8	Os usuários poderão ser separados por Grupo, e todo usuário obrigatoriamente deverá pertencer a um grupo.	5	***	Médio
R9	O sistema deverá disponibilizar buscas de arquivo pelo nome do arquivo, grupo, criador, código entre outros.	8	**	Médio
R10	Deverá existir a possibilidade do usuário encontrar arquivos realizando buscas por dados contidos nos metadados.	8	***	Alto
R11	Sempre que uma nova versão do documento for salva esta deverá ser considerada uma nova revisão do arquivo já existente.	9	**	Alto
R12	Os arquivos deverão possuir um item para classificação, cada arquivo inserido poderá ser classificado pelo usuário. (Procedimento, Contrato, Orçamento ...)	14	***	Baixo
R13	Cada arquivo inserido poderá receber informações adicionais (metadados) que serão sempre inseridos por funcionários com permissão.	16	***	Alto
R14	Os usuários poderão ter papéis definidos no sistema para acessar aos arquivos. (Editor, Revisor, Visitante..)	15	**	Baixo

continua na pagina seguinte...

ID	Descrição	Story Card	Prioridade	Risco
R15	O usuário poderá definir permissões de acesso para cada arquivo inserido		***	Baixo
R16	Usuários poderão preencher os metadados de um arquivo ao inseri-los no sistema		***	Médio
R17	Usuários com permissões especiais poderão cadastrar novos metadados para serem preenchidos no sistema.	16	**	Médio

6.2 Arquitetura do Sistema

Nesta seção será possível observar quatro camadas principais: Interface Web, Regra de Negócios, Prevalência de Dados e Testes.

A camada Interface Web é a camada onde a linguagem ASP.NET é utilizada e é a interface de relacionamento entre os usuários e o sistema.

A Regra de Negócios é a camada principal do sistema, que é independente das camadas de Testes e Interface Web, o que significa que outra camada de interface pode ser criada, como por exemplo uma camada Windows.Forms que possibilite o acesso ao sistema de um software instalado na máquina do usuário.

A camada de testes é utilizada para realizar os testes unitários, e é independente, podendo ser alterada ou substituída a qualquer momento.

A camada de Prevalência de Dados é a camada que garante o armazenamento das informações e a sua recuperação.

Uma representação gráfica das camadas é exibida na Figura 6.

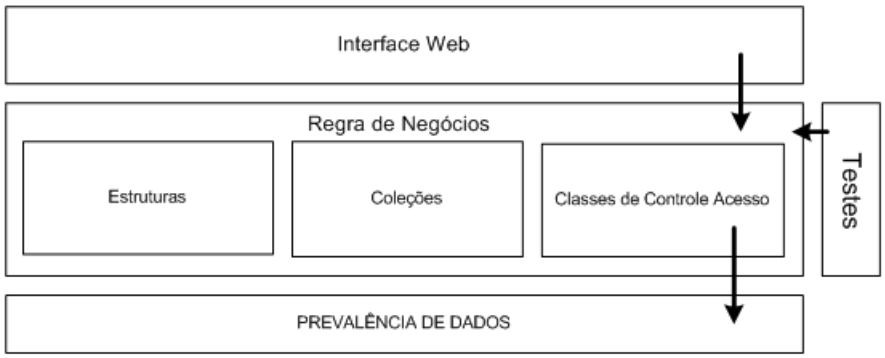

Figura 6: Arquitetura do Sistema

6.3 Diagrama de Classes

Nesta seção o diagrama de classes do GEDi será apresentado. Para facilitar o entendimento e facilitar a visualização ele foi dividido em várias partes.

Todas as classes apresentadas nesta seção, assim como os métodos e atributos estão em inglês, para seguir padrões de codificação da empresa ForLogic Softwares.

A Figura 7 demonstra o relacionamento entre as classes que possibilitam a gerência de grupos, as Figuras 8, 9, 10, 11 e 12 representam outras atividades de cadastro.

A Figura 13 representa a gerência de arquivos e revisões, As Classes representadas com cor diferente tiveram seus atributos retirados para se adequar ao tamanho do digrama, porém, serão exibidas por completo na Figura 14.

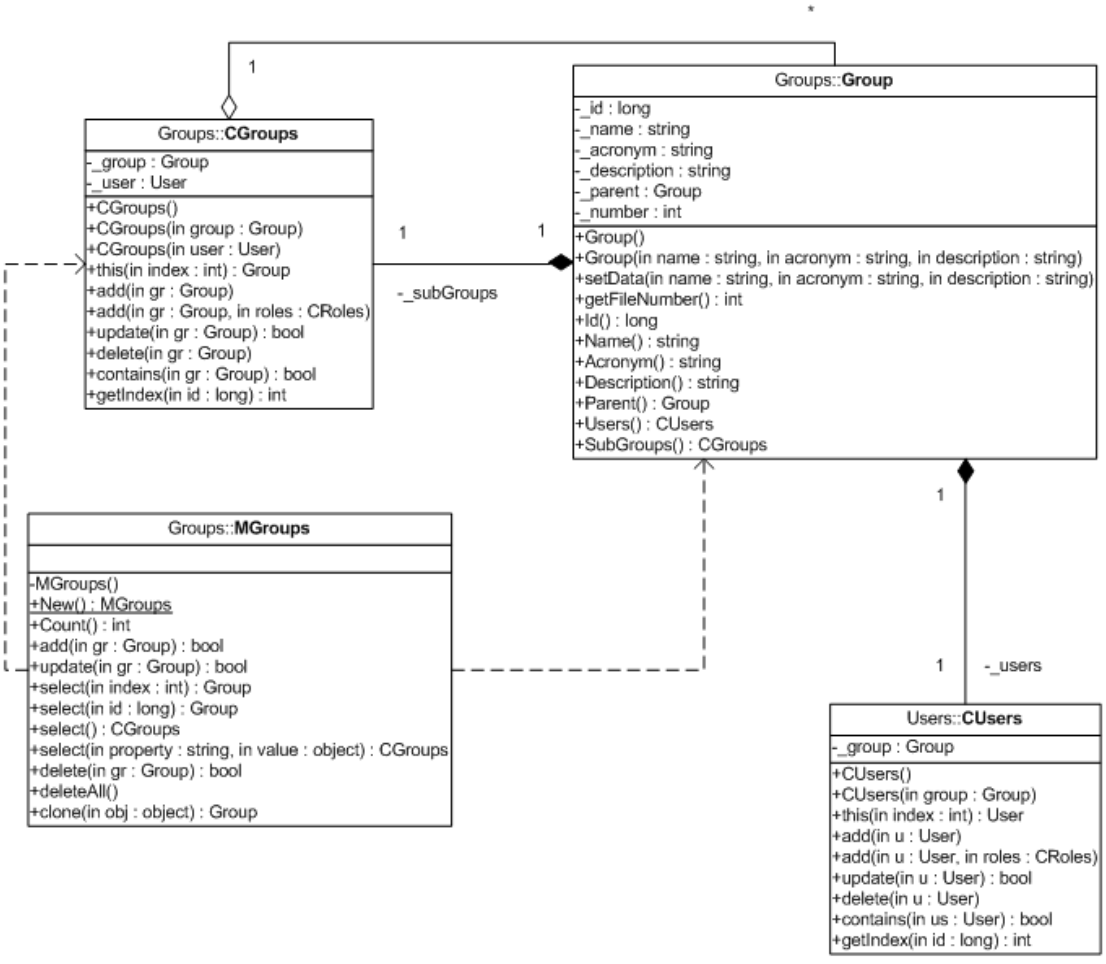

Figura 7: Gerência de Grupos - Diagrama de Classes

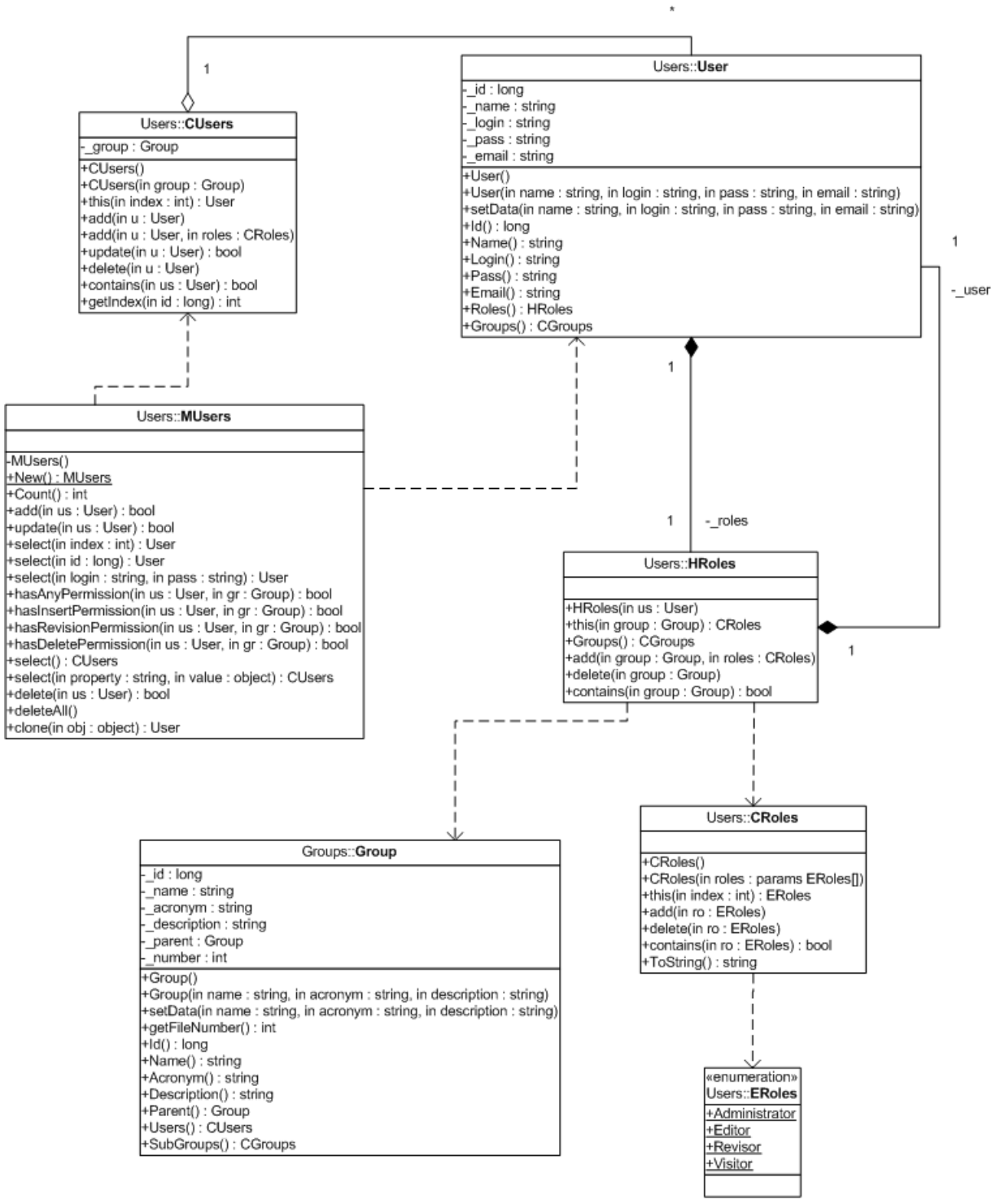

Figura 8: Gerência de Usuários - Diagrama de Classes

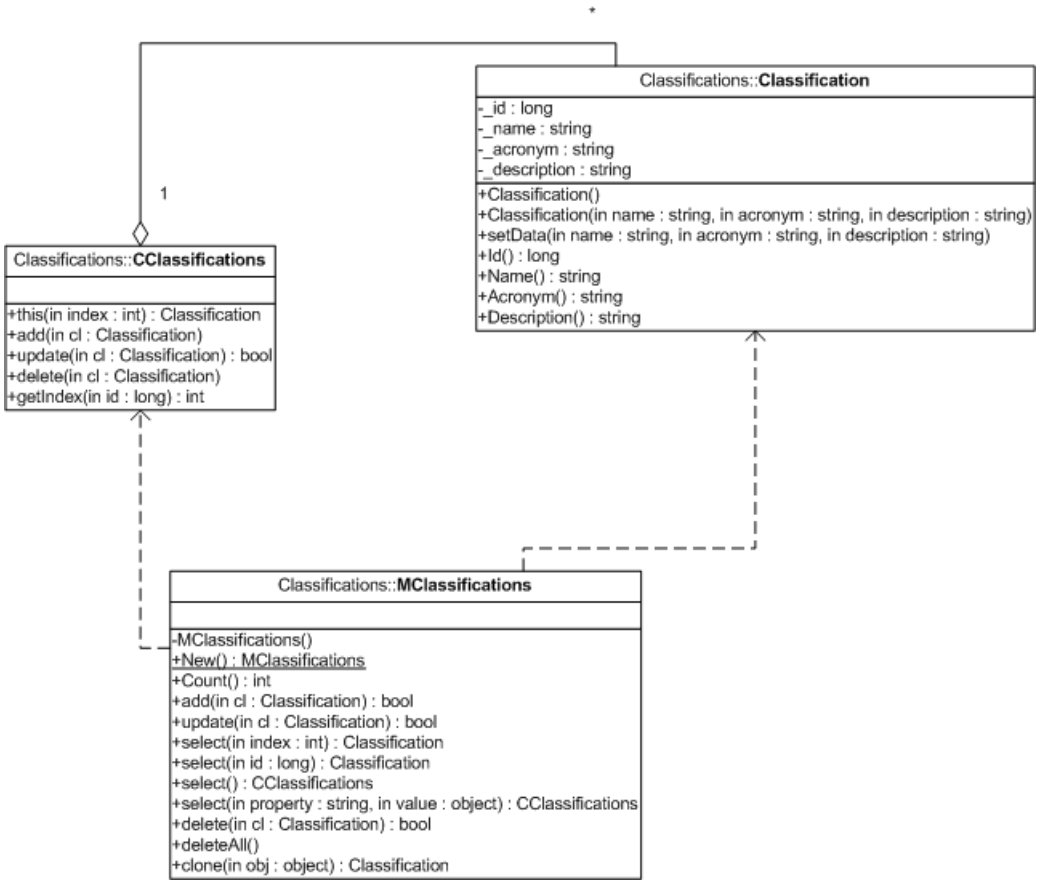

Figura 9: Gerência de Classificações - Diagrama de Classes

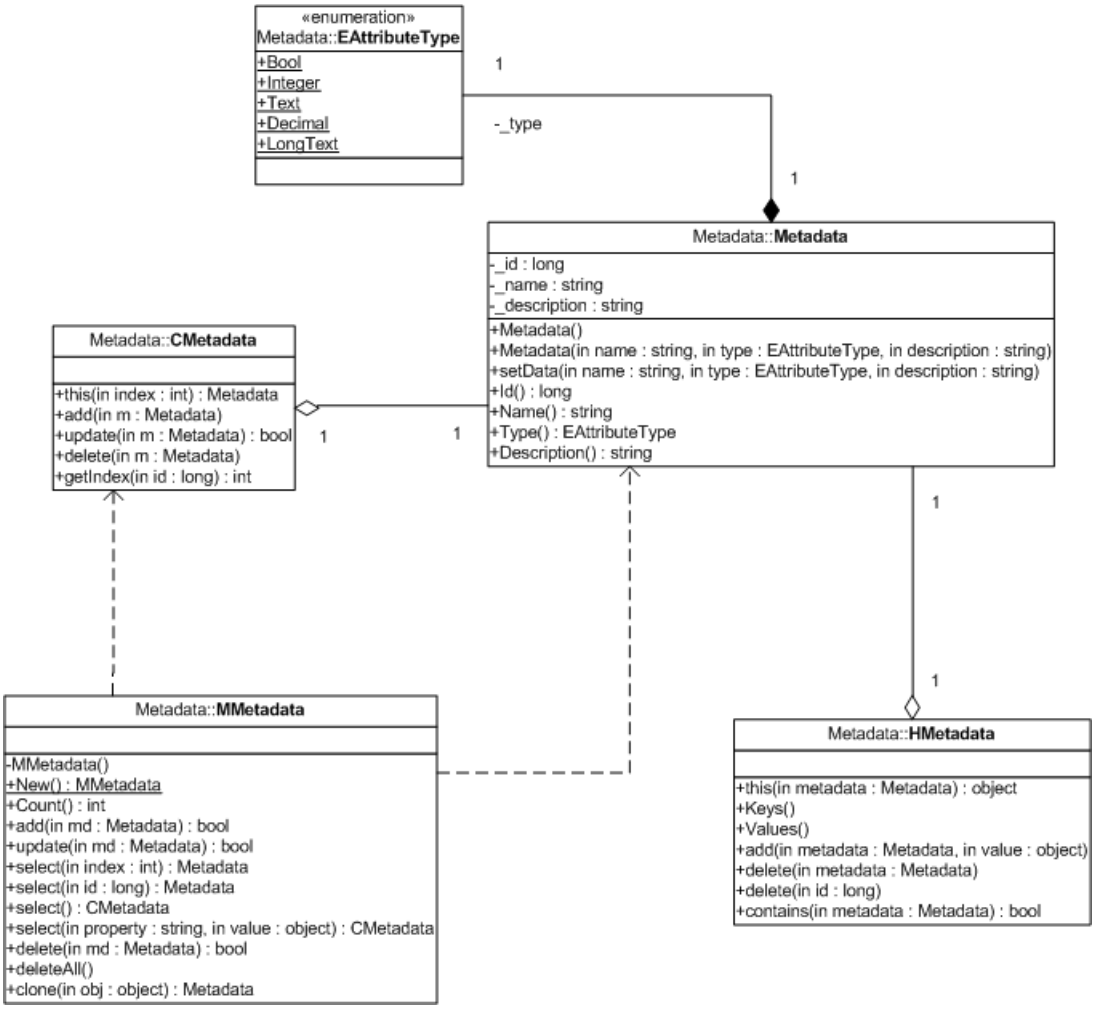

Figura 10: Gerência de Metadados - Diagrama de Classes

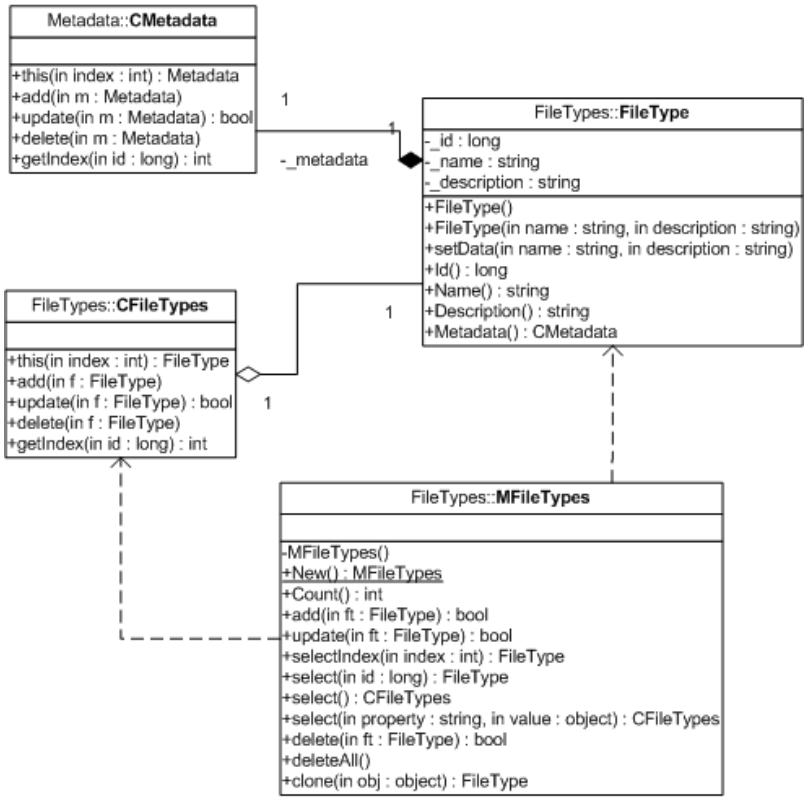

Figura 11: Gerência de Tipos de Arquivo - Diagrama de Classes

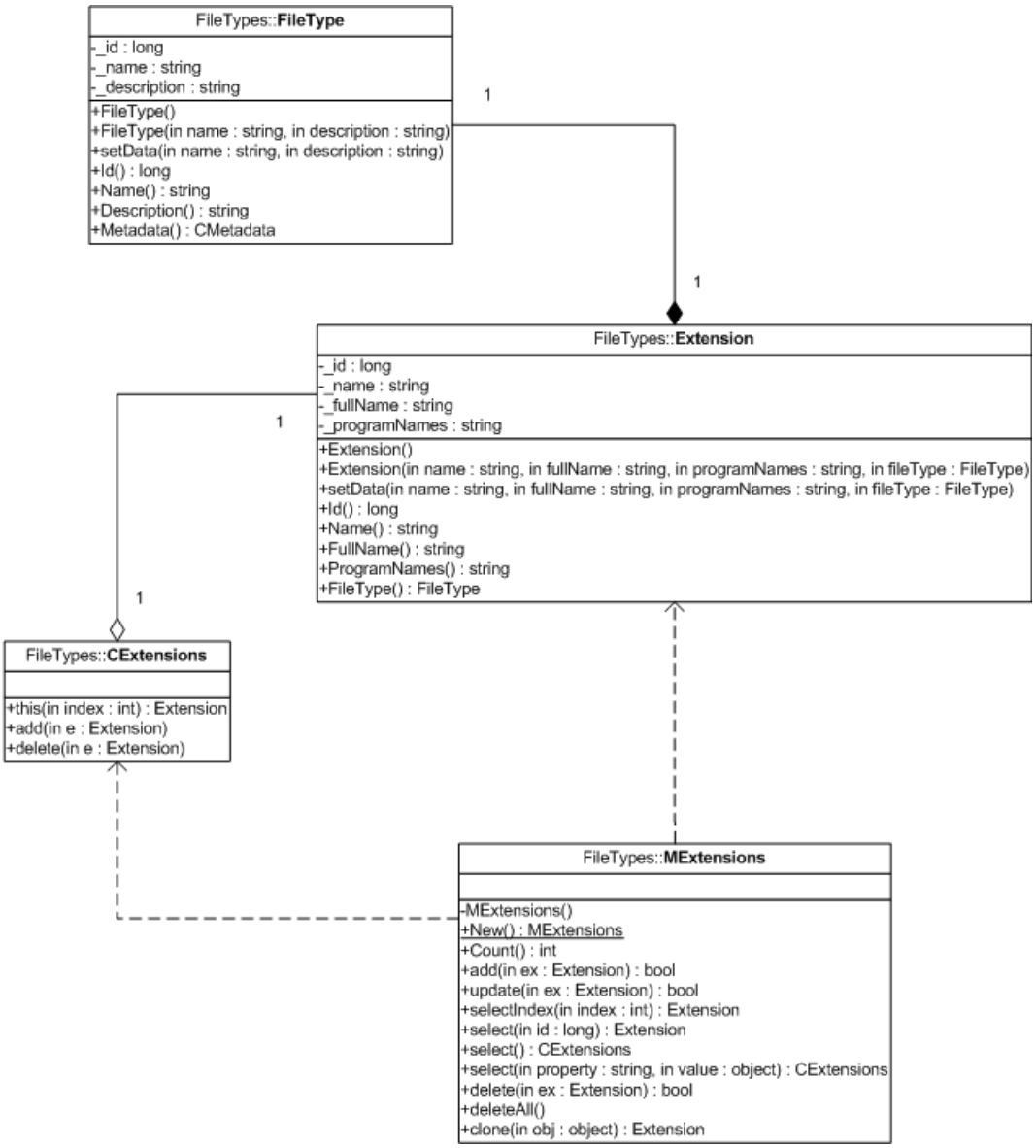

Figura 12: Gerência de Extensões - Diagrama de Classes

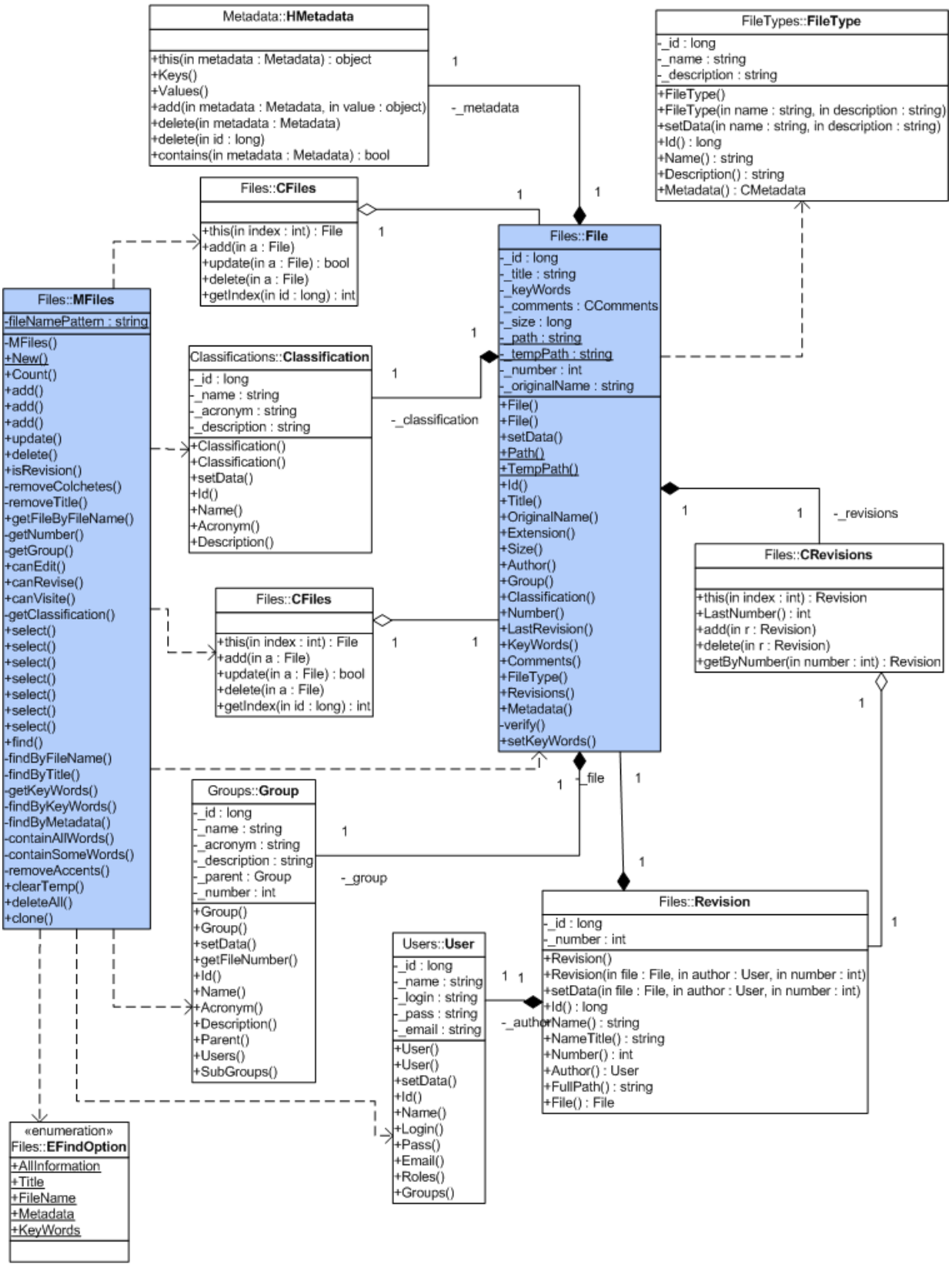

Figura 13: Gerência de Arquivos e Revisões - Diagrama de Classes

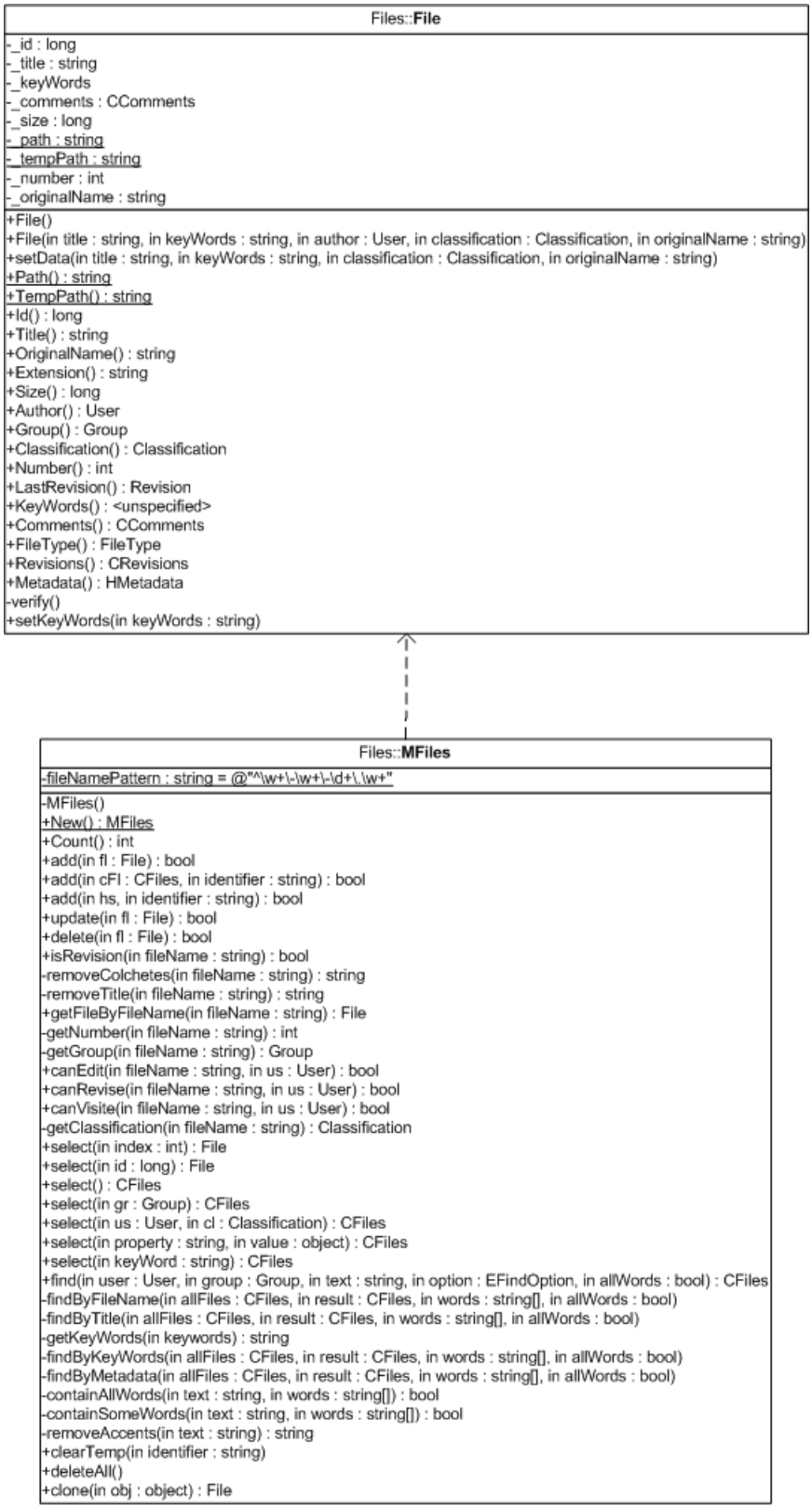

Figura 14: Classes MFiles e Files - Diagrama de Classes

7 Funcionamento da ferramenta GEDi

Como apresentado na seção 3.3, o GEDi possui como objetivo geral proporcionar a gerência dos documentos eletrônicos em uma organização, possibilitando ainda que cada documento receba informações adicionais.

Para facilitar o entendimento das funcionalidades do sistema, estas foram divididas em duas seções específicas. Os casos de uso com quem apenas o usuário administrador interage estão identificados na seção 7.1. Todos os outros casos de uso onde os outros usuários interagem estão dispostos na seção 7.2.

Uma visão geral dos casos de uso e dos atores do sistema pode ser obtida através da Figura 15.

7.1 Funcionalidades de Administrador

Um número grande de casos de uso foi gerado durante o decorrer do projeto. O desenvolvimento procurou seguir uma linha semelhante durante todo o projeto, e o mesmo aconteceu na modelagem dos diagramas.

Alguns casos de uso com funcionalidades semelhantes acarretaram na criação de diagramas semelhantes, este foi um caminho encontrado para facilitar o entendimento do sistema e para simplificar a manutenção.

Os diagramas similares geralmente possuem atividades de cadastro[1], e o que muda geralmente é o objeto de tais cadastros. Em alguns casos, usuários são gerenciados, em outros grupos, metadados, e assim por diante.

Um administrador do sistema terá permissão para acessar a várias atividades de cadastro, as quais os usuários comuns não tem acesso. Isso possível graças a uma barra de administração visível somente para o usuário administrador. Esta barra de administração do sistema é demonstrada na Figura 16.

Para garantir que usuários normais (que não sejam o administrador do sistema) não acessem as

[1] Atividades de cadastro - neste caso são identificadas por funcionalidades como cadastro, alteração e remoção de "itens".

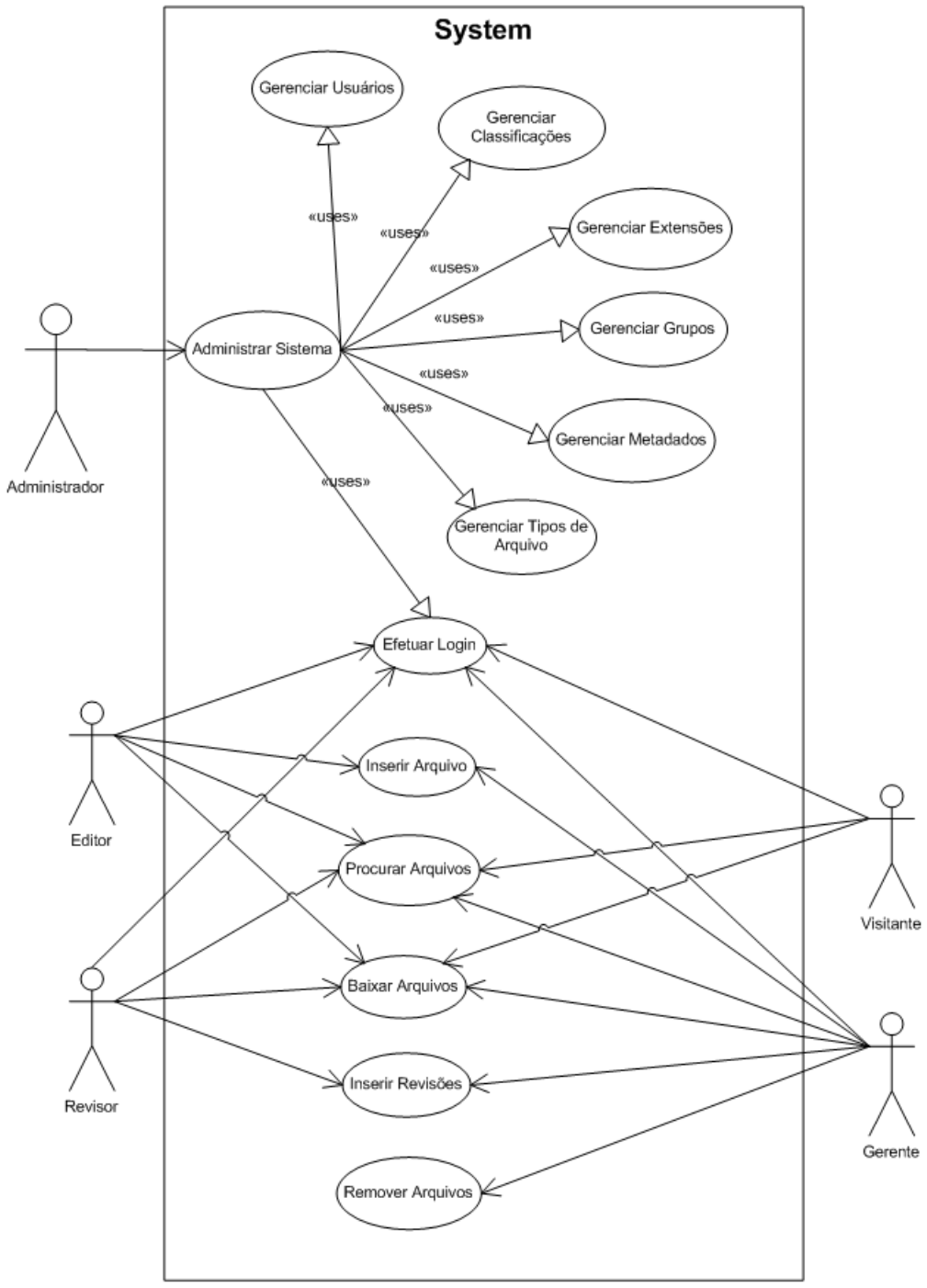

Figura 15: Visão Geral dos Casos de Uso

páginas de cadastro apenas digitando seu caminho na barra de endereços, é utilizado um controle através de sessões[2].

[2]Com o controle de Sessão, é possível identificar unicamente um usuário e guardar suas informações no servidor(**??**) e também disponibilizar acesso de acordo com o usuário identificado.

As funcionalidades do administrador serão explicadas e identificadas nos itens 7.1.1, 7.1.2 e 7.1.3.

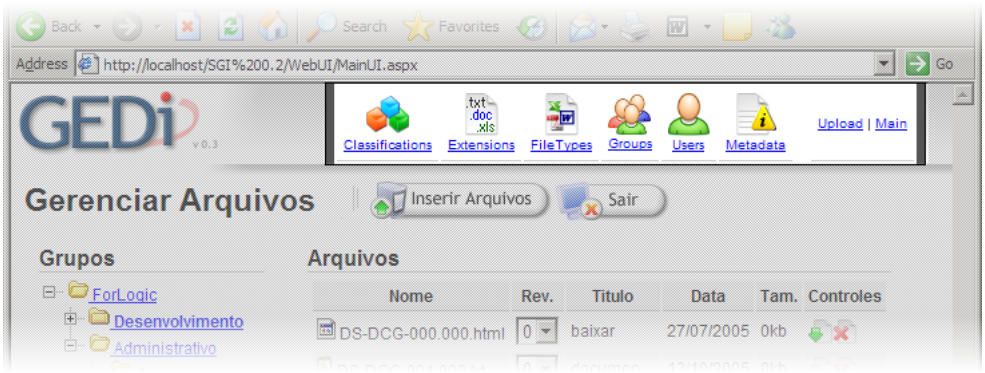

Figura 16: Barra de Administração do Sistema

7.1.1 Gerência de Usuários

A Gerência de Usuários é uma funcionalidade relativamente simples, pois proporciona ao administrador a capacidade de adicionar usuários, alterar suas informações, ou removê-los.

O diagrama de Gerência de Usuários é demonstrado na Figura 17 e representa a gerência dos usuários por parte do administrador.

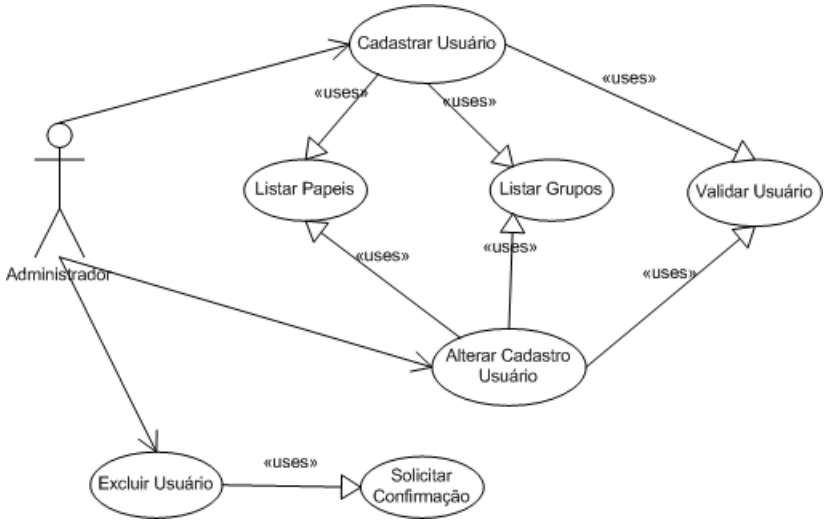

Figura 17: Gerenciar Usuários - UC3

A especificação do diagrama de caso de uso Gerenciar Usuários na Tabela 3 é sucinta, pois o detalhamento maior acontece na especificação dos diagramas seguintes demonstrados nas tabelas 4, 5 e 6.

Tabela 3: Especificação de Caso de Uso - Gerenciar Usuários

Caso de Uso: Gerenciar Usuários
ID: UC3
Descrição: Através deste caso de uso o administrador do sistema poderá gerenciar os usuários, cadastrando, alterando e removendo os usuários.
Ator: Administrador
Fluxo Normal: 1. Efetuar login no sistema 2. Acessar link de gerência de usuários 3. Gerenciar usuários
Fluxo Alternativo: 1. Login Incorreto

Tabela 4: Especificação de Caso de Uso - Cadastrar Usuários

Caso de Uso: Cadastrar Usuários
ID: UC3.1
Descrição: Através deste caso de uso o administrador do sistema poderá cadastrar um novo usuário, selecionando um ou mais grupos para este.
Ator: Administrador
Fluxo Normal: 1. Clicar em Novo 2. Preencher os dados do usuário 3. Selecionar o grupos e permissões e adicioná-los ao usuário. 4. Gravar
Fluxos Alternativos 1. No momento do cadastro pode já existir um usuário com este nome. Uma mensagem de erro será exibida informando o problema. 2. Os dados podem estar incompletos. Uma mensagem solicitará que os dados necessários sejam adicionados.

Tabela 5: Especificação de Caso de Uso - Alterar Usuário

Caso de Uso: Alterar Usuário
ID: UC3.2
Descrição: Através deste caso de uso o administrador do sistema poderá alterar os dados de um usuário, inclusive o grupo a que este usuário pertence.
Ator: Administrador
Fluxo Normal: 1. Clicar em Alterar 2. Preencher os novos dados do usuário 3. Selecionar os grupos necessários e adicioná-los ao usuário ou removê-los. 4. Gravar
Fluxos Alternativos 1. No momento do cadastro pode já existir um usuário com este nome (caso o nome do usuário tenha sido alterado). Uma mensagem de erro será exibida informando o problema. 2. Os dados podem estar incompletos. Uma mensagem solicitará que os dados necessários sejam adicionados.

Tabela 6: Especificação de Caso de Uso - Remover Usuário

Caso de Uso: Remover Usuário
ID: UC3.3
Descrição: Através deste caso de uso o administrador do sistema poderá remover um usuário específico.
Ator: Administrador
Fluxo Normal: 1. Selecionar usuário em uma lista 2. Clicar em remover 3. Confirmar exclusão do usuário
Fluxos Alternativos 1. O administrador pode não confirmar a exclusão. Neste caso o usuário não é removido.

Como demonstrado na Figura 18, o usuário administrador pode selecionar um usuário em uma lista do lado esquerdo na interface (em destaque) e alterar seus dados através do botão “Alterar”. O usuário administrador pode também simplesmente clicar no botão “Novo” e cadastrar um novo usuário. Em ambos os casos, o quadro do lado direito na interface será habilitado para edição.

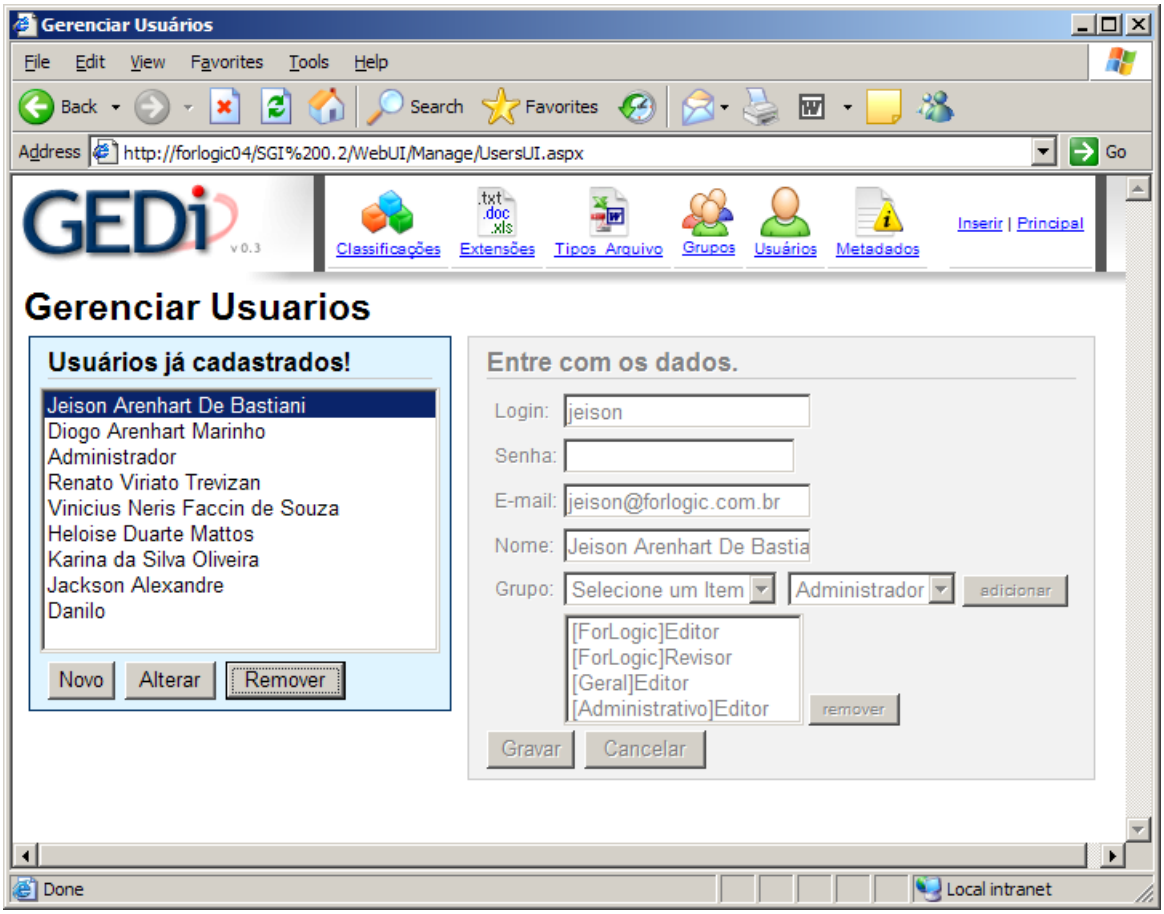

Figura 18: Lista de Usuários - Captura de Tela

É possível notar através da observação da Figura 18 e Figura 19, que foram utilizadas técnicas para orientar o usuário, buscando com isso que ele entenda com facilidade o funcionamento das gerências e o significado de sua interface. Sempre que um conjunto de controles é ativado, ele recebe outra cor, e o conjunto desativado se torna cinza. Isso possibilita que o usuário perceba visualmente quais controles ele pode utilizar e aprende a utilizá-los rapidamente(**??**).

Após pressionar o botão “Novo” o administrador pode editar o quadro disposto a direita que fica em destaque ao mesmo tempo que o quadro esquerdo é desabilitado e ganha uma aparência cinzenta. Como demonstrado na Figura 19, após preencher os dados, o usuário poderá selecionar um grupo e então selecionar o papel que este usuário poderá desempenhar no grupo[3].

[3]Mais sobre a organização dos papéis e suas permissões pode ser lido nas seções 3.3.3 e 7.2.2.

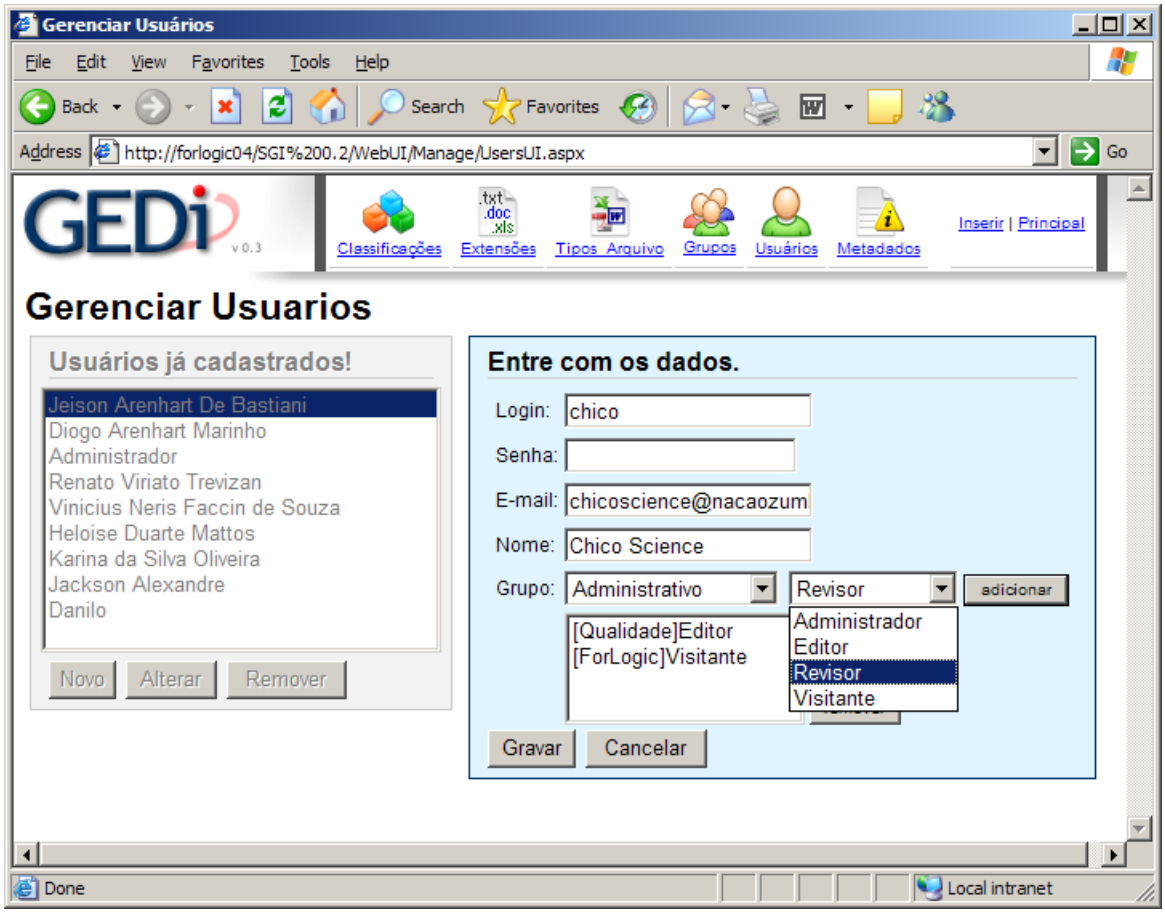

Figura 19: Adicionando Papéis aos Grupos - Captura de Tela

Existe a possibilidade de um usuário estar em vários grupos diferentes, e para cada grupo que ele for adicionado obrigatoriamente o administrador deverá selecionar um papel para este usuário dentro deste grupo.

Após preencher todos os dados que são necessários, o usuário poderá ser gravado quando o administrador clicar em “Gravar”. Uma mensagem de erro pode ser exibida caso algum dado obrigatório não esteja preenchido, ou estejam preenchidos de forma incorreta.

É possível obter uma representação diferente das atividades de cadastro, alteração e exclusão de usuários, observando o diagrama de atividades da Figura 20.

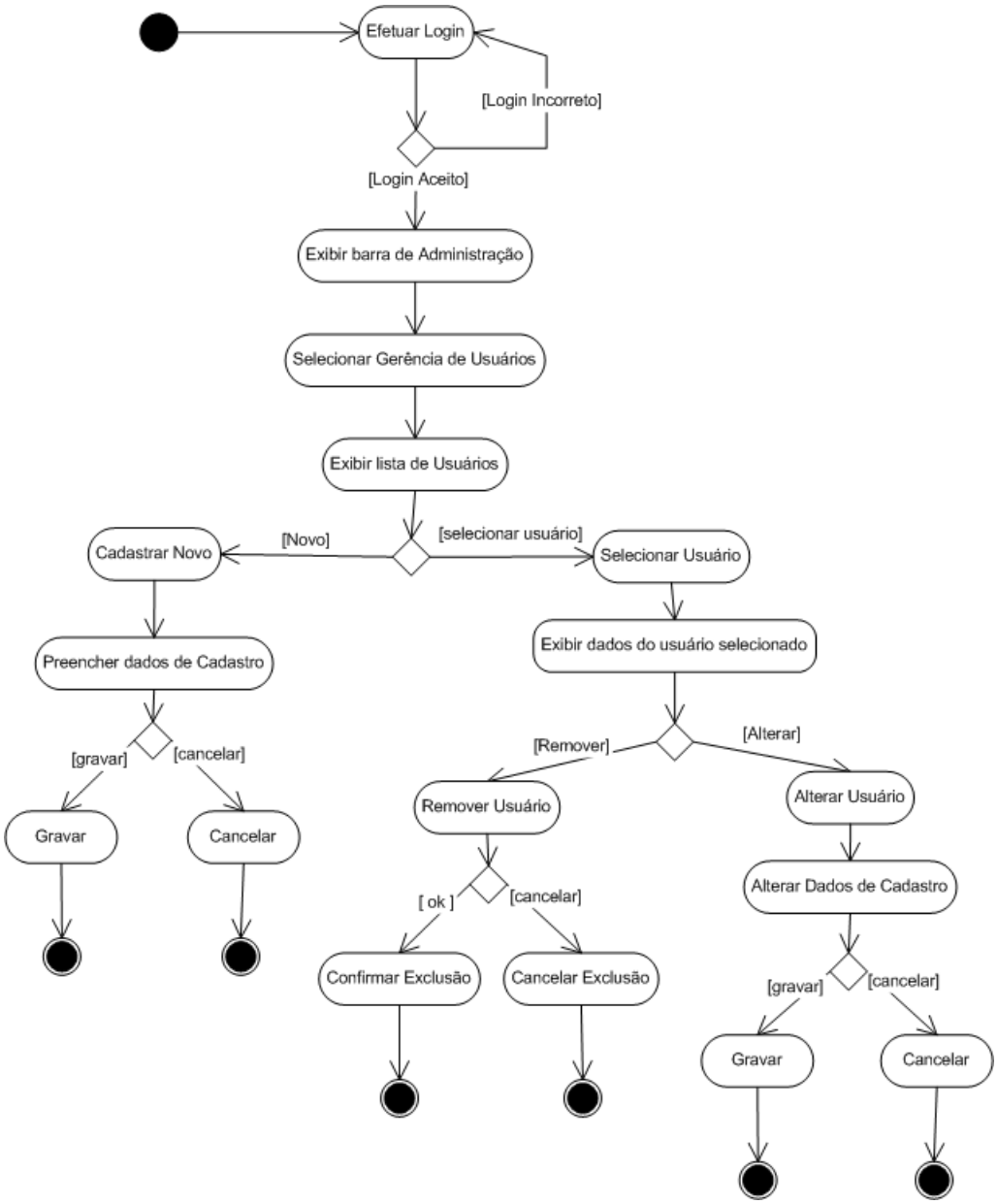

Figura 20: Gerência de Usuários - Diagrama de Atividades

Os diagramas de sequência representados nas figuras 21, 22 e 23 representam as três atividades de cadastro da gerência de usuário (cadastrar, alterar e remover).

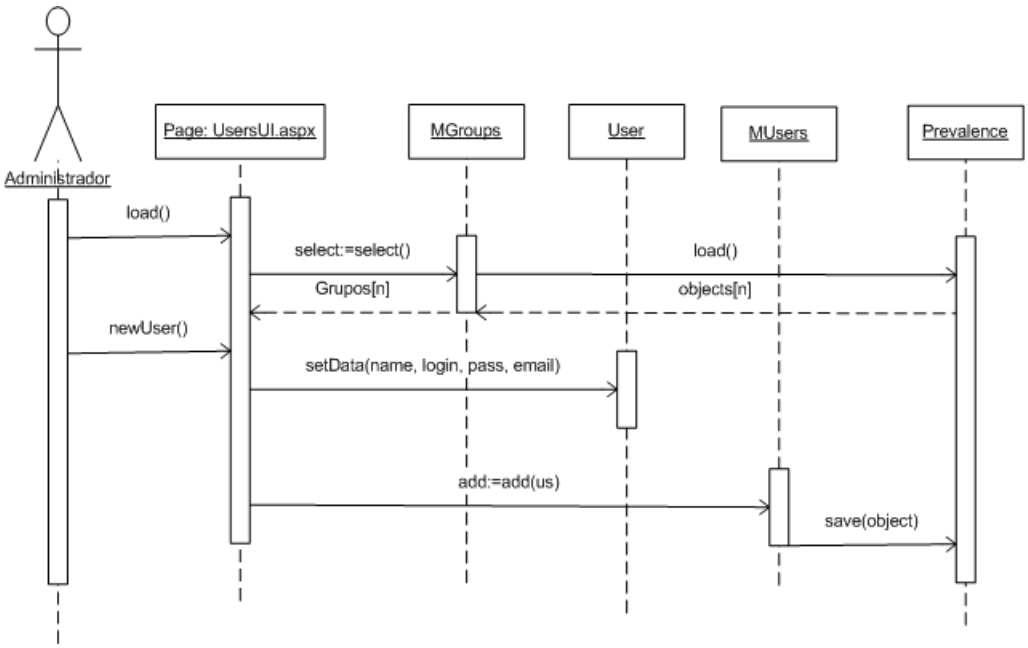

Figura 21: Cadastrar Usuário - Diagrama de Sequência

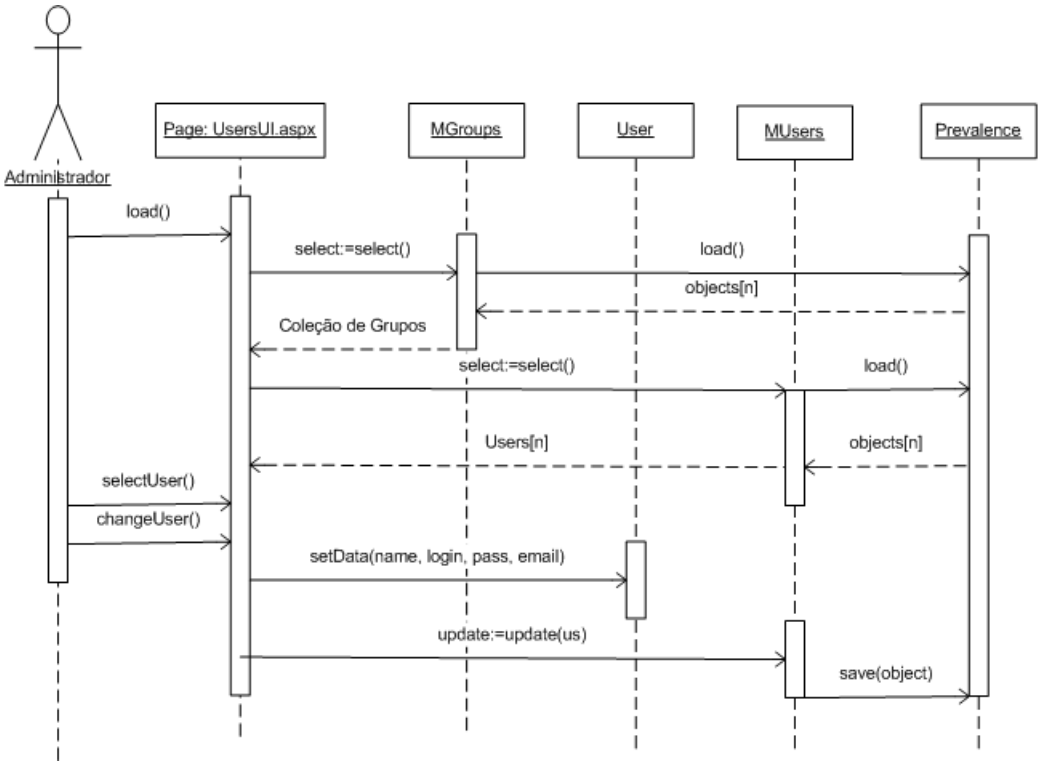

Figura 22: Alterar Usuário - Diagrama de Sequência

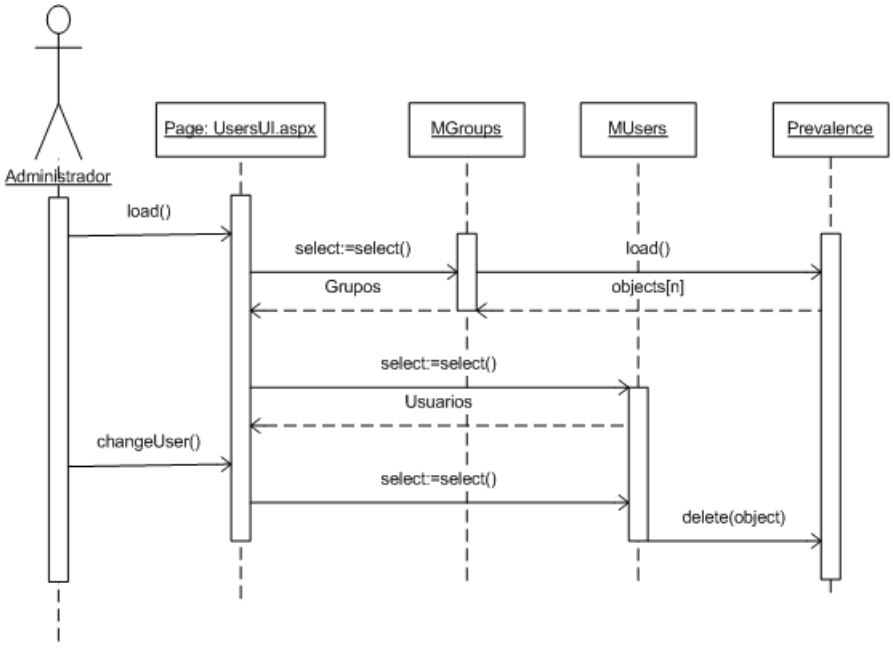

Figura 23: Remover Usuário - Diagrama de Sequência

7.1.2 Gerência de Grupos

A Gerência de Grupos, assim como a Gerência de Usuários na seção 7.1.1 é uma funcionalidade simples de ser entendida. A Gerência de Grupos proporciona ao administrador a possibilidade de adicionar grupos, alterar suas informações, ou removê-los.

O diagrama de Gerência de Grupos demonstrado na figura 24 representa gerência dos grupos por parte do administrador.

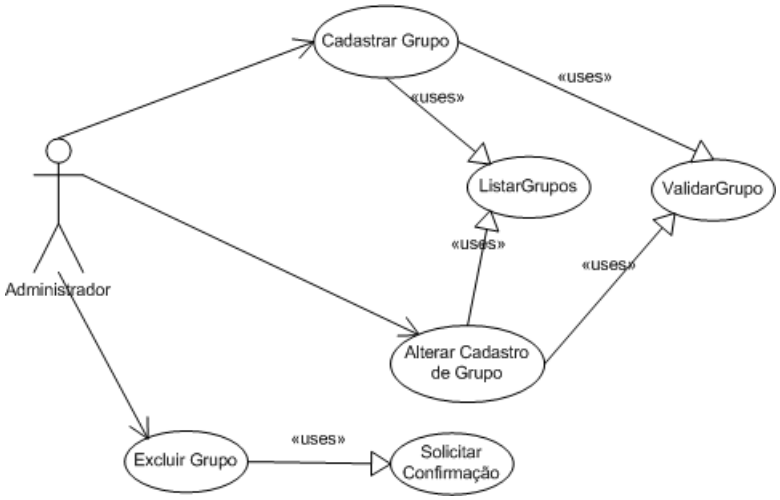

Figura 24: Gerenciar Grupos - UC2

A especificação do diagrama de caso de uso Gerenciar Grupos na Tabela 7 é sucinta, pois um detalhamento maior acontece na especificação dos diagramas seguintes, demonstrados nas tabelas 8, 9 e 10.

Tabela 7: Especificação de Caso de Uso - Gerenciar Grupos

Caso de Uso: Gerenciar Grupos
ID: UC2
Descrição: Através deste caso de uso o administrador do sistema poderá gerenciar os grupos, cadastrando, alterando e removendo os grupos.
Ator: Administrador
Fluxo Normal: 1. Efetuar login no sistema 2. Acessar link de gerência de grupos 3. Gerenciar grupos
Fluxo Alternativo: 1. Login Incorreto

Tabela 8: Especificação de Caso de Uso - Cadastrar Grupo

Caso de Uso: Cadastrar Grupo
ID: UC2.1
Descrição: Através deste caso de uso o administrador do sistema poderá cadastrar um novo grupo, selecionando obrigatoriamente um "Parent"[4] para este grupo.
Ator: Administrador
Fluxo Normal: 1. Clicar em Novo 2. Preencher os dados do grupo 3. Selecionar o grupo “Parent”. 4. Gravar
Fluxos Alternativos 1. No momento do cadastro pode já existir um grupo com este nome. Uma mensagem de erro será exibida informando o problema. 2. Os dados podem estar incompletos. Uma mensagem solicitará que os dados necessários sejam adicionados.

[4]Parent é o grupo Pai, o grupo a qual o grupo atual pertence, no GEDi cada grupo deve possuir ao menos um Parent (com excessão do grupo raiz).

Tabela 9: Especificação de Caso de Uso - Alterar Grupo

Caso de Uso: Alterar Grupo
ID: UC2.2
Descrição: Através deste caso de uso o administrador do sistema poderá alterar os dados de um grupo, podendo alterar inclusive a que grupo este pertence.
Ator: Administrador
Fluxo Normal: 1. Clicar em Alterar 2. Preencher os novos dados do grupo 3. Selecionar o grupo “Parent”. 4. Gravar
Fluxos Alternativos 1. No momento do cadastro pode já existir um grupo com este nome. Uma mensagem de erro será exibida informando o problema. 2. Os dados podem estar incompletos. Uma mensagem solicitará que os dados necessários sejam adicionados.

Tabela 10: Especificação de Caso de Uso - Remover Grupo

Caso de Uso: Remover Grupo
ID: UC2.3
Descrição: Através deste caso de uso o administrador do sistema poderá remover um grupo específico.
Ator: Administrador
Fluxo Normal: 1. Selecionar grupo em uma lista 2. Clicar em remover 3. Confirmar exclusão do grupo
Fluxos Alternativos 1. O administrador pode não confirmar a exclusão. Neste caso nada acontece.

É possível observar na Figura 25, que o administrador pode selecionar os grupos em uma lista a esquerda (em destaque) e alterar seus dados através do botão "Alterar", ou simplesmente clicar em "Novo". Em ambos os casos o quadro a direita será habilitado para edição.

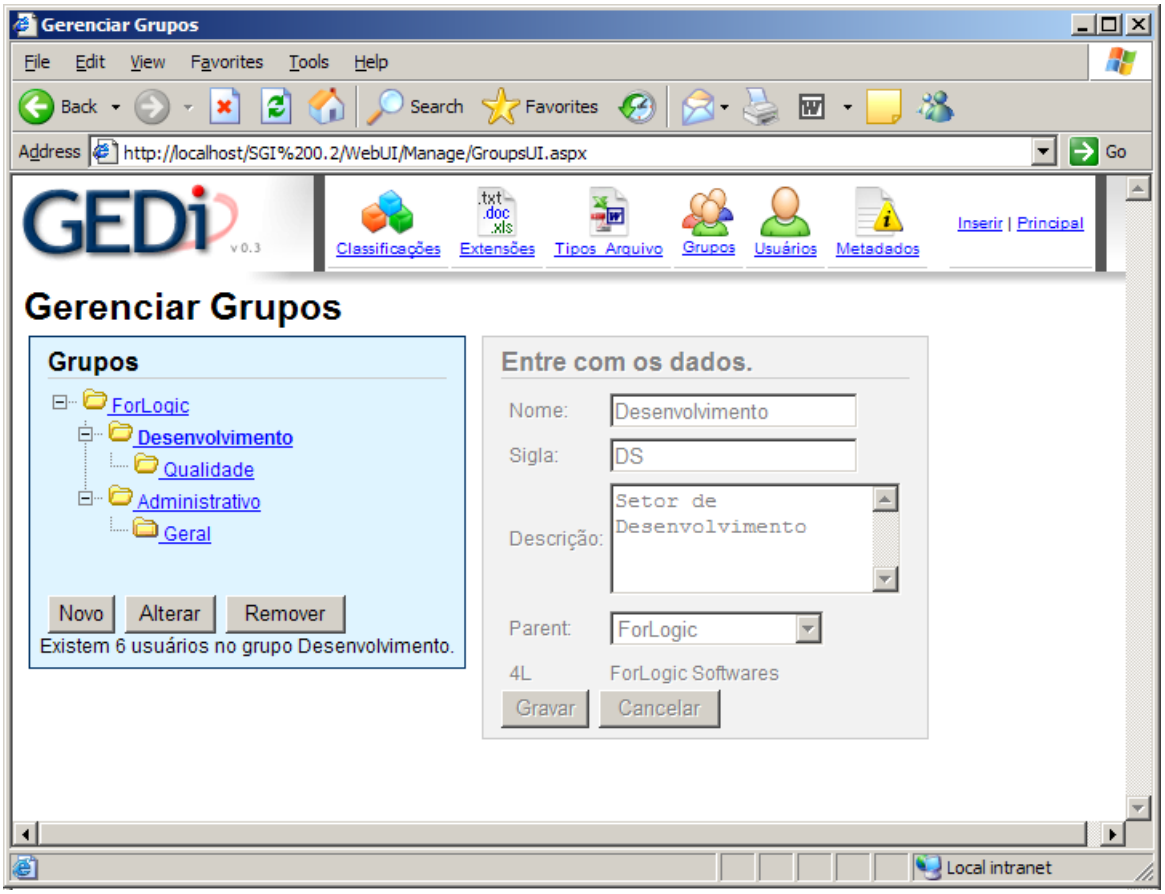

Figura 25: Selecionando Grupos - Captura de Tela

Após pressionar o botão "Novo", o administrador poderá editar o quadro disposto ao lado direito, que fica em destaque ao mesmo tempo que o quadro esquerdo é desabilitado e ganha uma aparência cinzenta. Como demonstrado na Figura 26, após preencher os dados o administrador deve selecionar um *Parent* para o grupo.

Figura 26: Alterando Dados - Captura de Tela

É possível notar as semelhanças existentes entre a utilização dos controles entre as Gerências de Usuários (7.1.1) e a Gerência de Grupos. Ambas as gerências utilizam os mesmos padrões na interface com usuário, simplificando ao máximo a utilização do sistema.

Observando a Figura 27, é possível obter uma representação diferente das atividades de cadastro, alteração e exclusão de grupos, através de uma diagrama de atividades.

Os diagramas de sequência representados nas figuras 28, 29 e 30 representam as três atividades de cadastro da gerência de grupos.

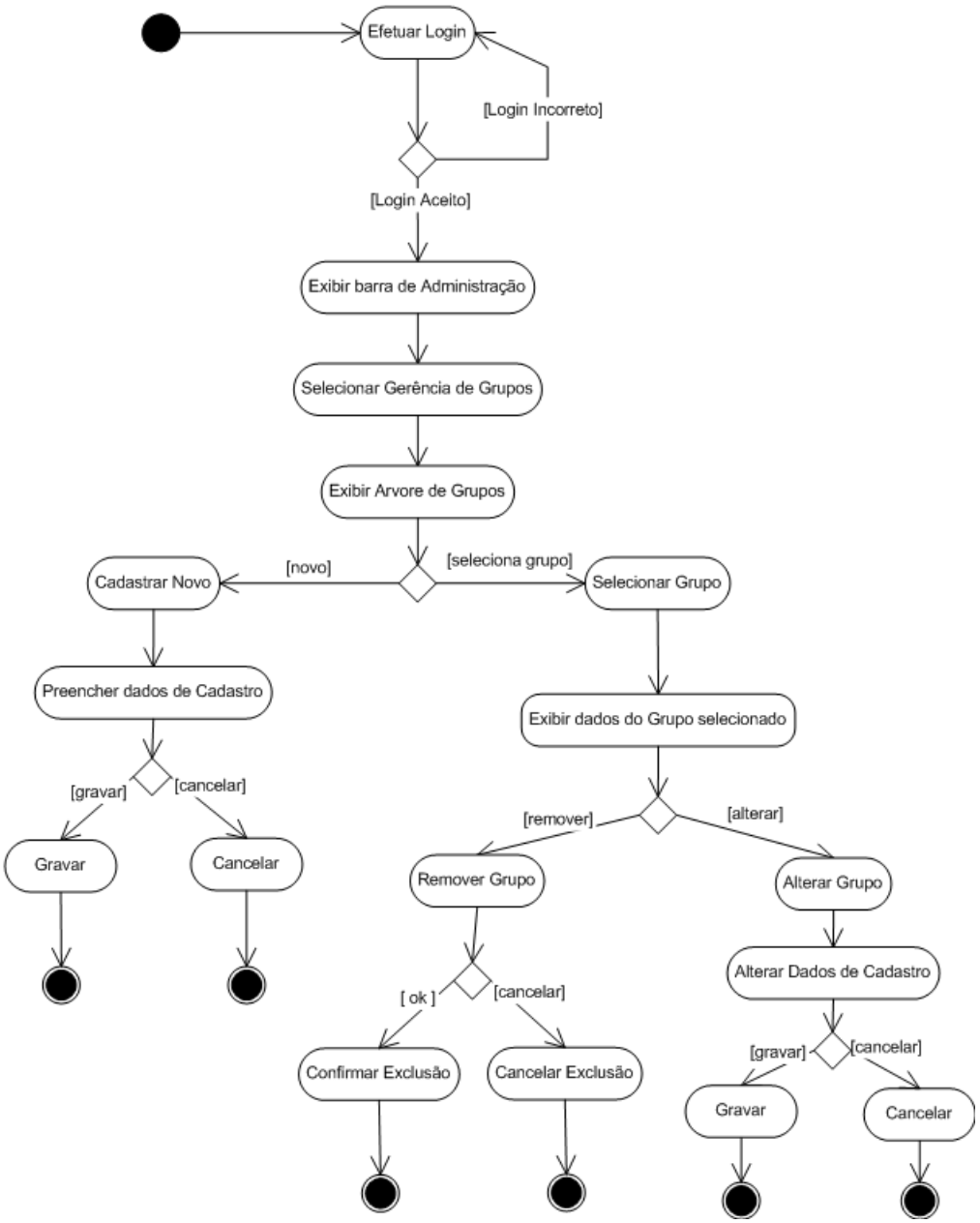

Figura 27: Gerência de Grupos - Diagrama de Atividades

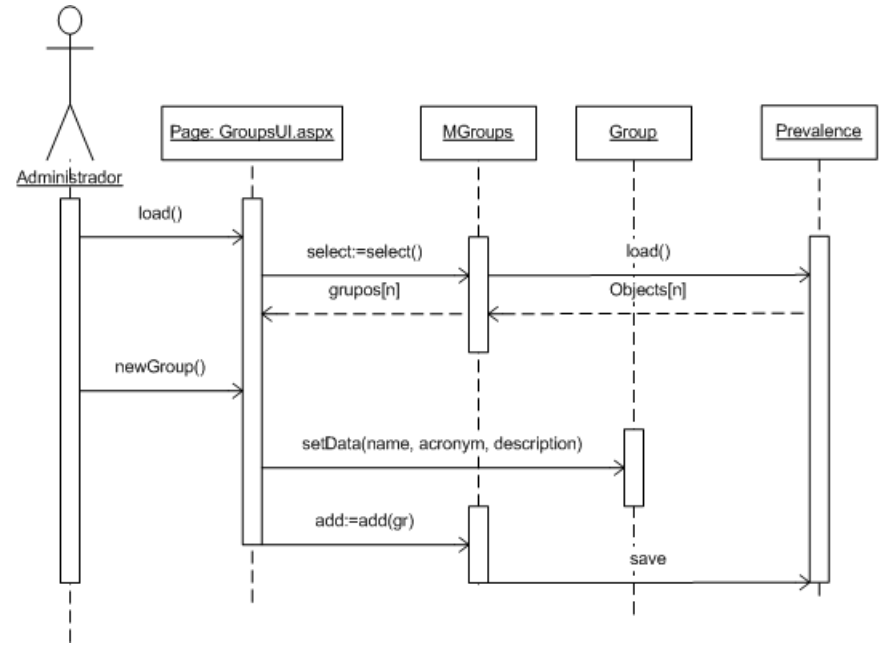

Figura 28: Cadastrar Grupo - Diagrama de Sequência

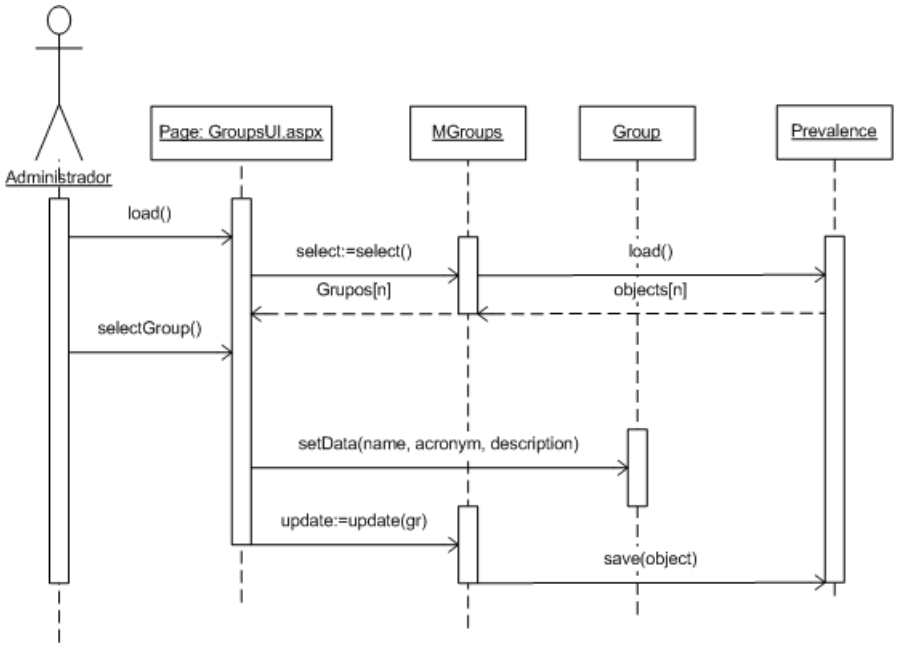

Figura 29: Alterar Grupo - Diagrama de Sequência

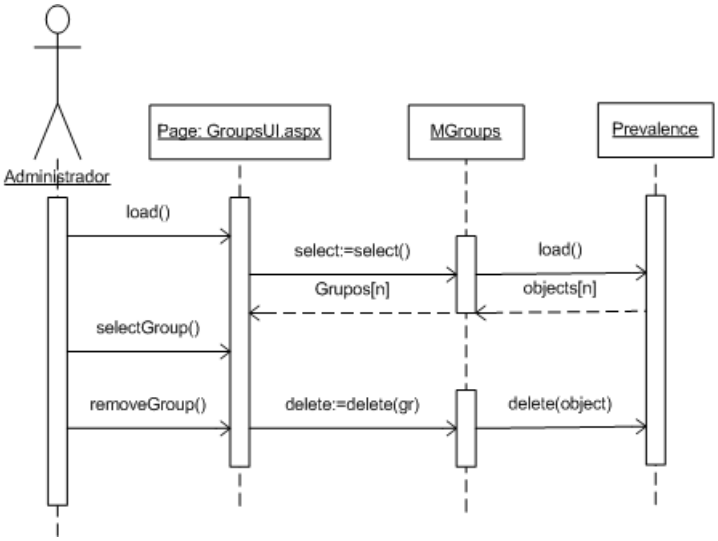

Figura 30: Remover Grupo - Diagrama de Sequência

7.1.3 Outras Gerências: Metadados, Tipos de Arquivo e Extensões

Como foi possível observar nas seções anteriores o GEDi foi construído de uma forma que, ao compreender o funcionamento de uma gerência básica seja possível compreender todas as outras. Sempre existem algumas peculiaridades que mudam de uma gerência para outra, porém, as principais funcionalidades são as mesmas.

Para simplificar e facilitar a leitura do trabalho, as descrições de Caso de Uso e os Diagramas de sequência e Caso de Uso foram retiradas desta seção, já que são muito semelhantes as seções 7.1.1 e 7.1.2. O objetivo nesta seção é entender o relacionamento entre as gerências de Metadados, Tipos de Arquivo e Extensões.

Nas seções 7.1.1 e 7.1.2, fica claro que um usuário pertencerá a um grupo, e que da mesma forma, os grupos conterão usuários. O relacionamento pode ser facilmente entendido. Nesta seção será demonstrado como os Metadados, os Tipos de Arquivo e as Extensões se relacionam, e qual a importância desse relacionamento.

É bom lembrar que o sentido de existirem metadados no sistema é para poder agregar informações adicionais aos arquivos, como visto na seção 3.3.2.

Caso o responsável de uma empresa ache necessário que todas as apresentações multimídia tenham metadados, o primeiro contendo o tempo médio de duração da apresentação, e o segundo com o número de slides, bastará ele adicionar esses metadados a este tipo de arquivo.

Para fazer isso, o administrador inicialmente deverá acessar a gerência de metadados através da barra de administração (Figura 16). Neste momento ele irá preencher os dados para o primeiro metadado correspondente ao número de slides, conforme a Figura 31.

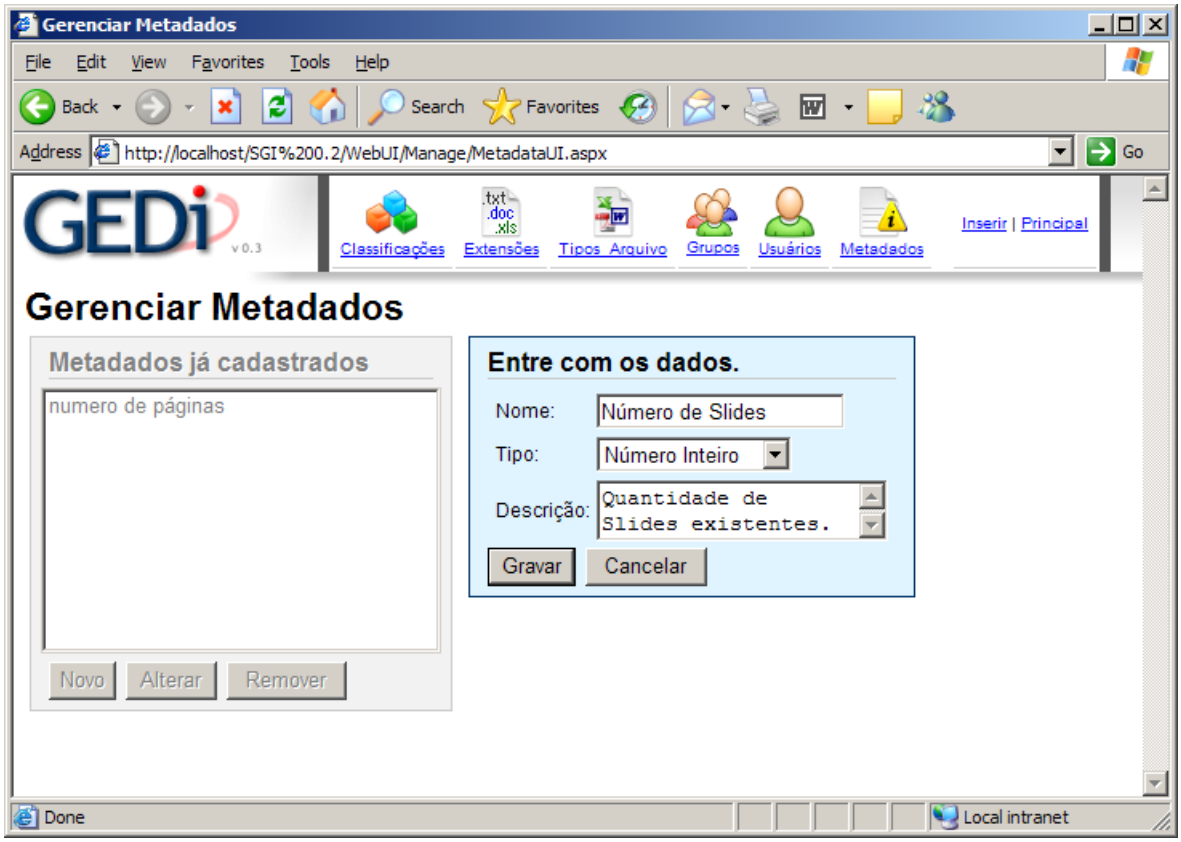

Figura 31: Gerenciar Metadados - Captura de Tela

Após isso o administrador poderá cadastrar o segundo metadado que diz respeito ao tempo médio de duração da apresentação. É importante lembrar que é possível escolher o tipo do metadado que deseja cadastrar, ou seja, número inteiro, decimal, verdadeiro ou falso, texto ou texto longo[5].

Após cadastrar os metadados solicitados pela empresa, o administrador poderá cadastrar um novo Tipo de Arquivo, através da gerência de Tipo de Arquivo.

Para cadastrar o Tipo de Arquivo, basta preencher os dados com o nome do Tipo do Arquivo "Apresentação Multimídia" e a descrição desejada. Após isso, é possível adicionar a este tipo de arquivo que está sendo cadastrado os metadados criados anteriormente (Número de Slides e Tempo Médio de Duração). Para inserir metadados ao Tipo de Arquivo, basta selecionar o nome do metadado e clicar em adicionar, como pode ser observado na Figura 32.

[5]Os trabalhos futuros irão acrescentar mais tipos a essa lista, porém é necessário implantar um projeto piloto e acompanhar as necessidades dos usuários.

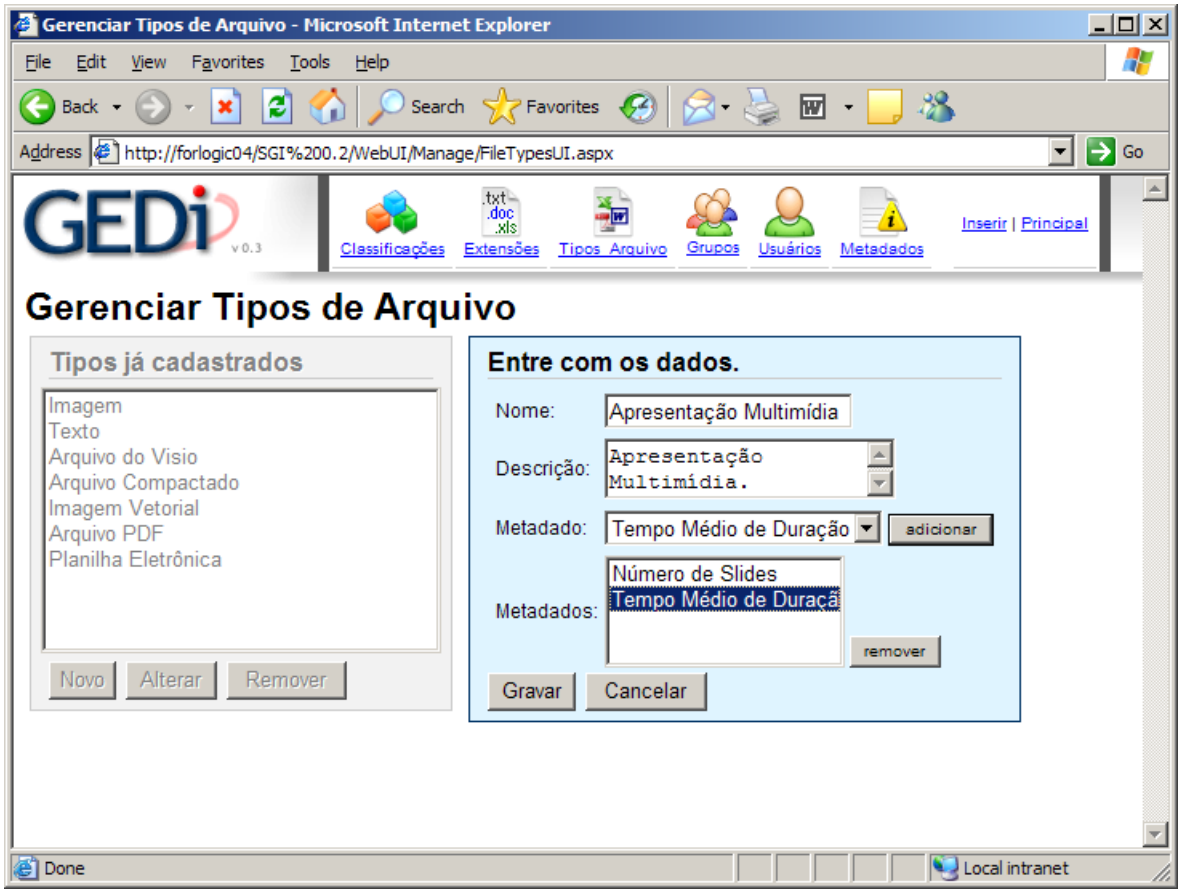

Figura 32: Gerenciar Tipos de Arquivo - Captura de Tela

Agora já existe um Tipo de Arquivo denominado "Apresentação Multimídia"e metadados relacionados a este Tipo de Arquivo ("Número de Slides"e "Tempo Médio de Duração"). Porém, o administrador ainda não cadastrou as extensões, e serão estas que irão indicar ao sistema quais arquivos serão compreendidos como uma apresentação multimídia. Para realizar este cadastro, o administrador deverá utilizar a gerência de extensões.

Como pode ser visto na Figura 33, para cadastrar uma nova extensão bastará preencher os dados como nas demais gerências, e após o preenchimento dos dados um tipo de arquivo deverá ser associado a esta extensão.

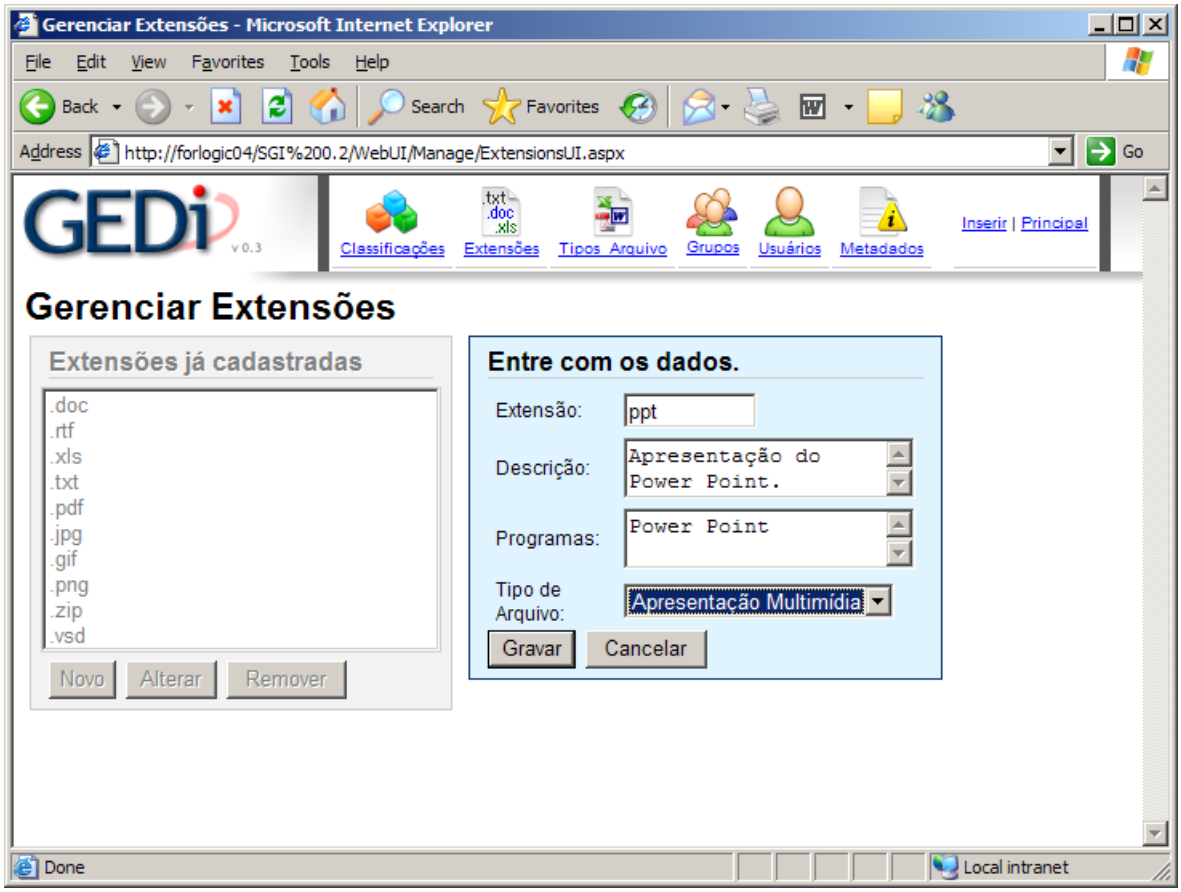

Figura 33: Gerenciar Extensões - Captura de Tela

Podem surgir questionamentos sobre essa maneira de ligar os metadados com um tipo de arquivo, e não diretamente com uma extensão, como seria mais comum. Porém, existem alguns pontos que devem ser observados.

Seguindo o exemplo citado, quando um responsável na empresa solicitou que as apresentações multimídia tivessem metadados, é importante lembrar que o Microsoft Power Point possui arquivos .ppt que são apresentações multimídia do Microsoft Office. Porém o Open Office.org também possui um software de apresentações multimídia, cuja extensão é .sxi. Ao criar um tipo de arquivo, é possível associar duas extensões a este, e isso possibilita que várias extensões com mesma finalidade sejam agrupadas.

Caso seja solicitado que todo documento de texto tenha um metadado "Número de Páginas", é possível criar esse metadado, criar um tipo de arquivo "Documento de Texto"e associá-lo ao metadado, após isso é só associar extensões como .doc, .rtf, .mcw[6], entre outras, ao tipo de arquivo "Documento de Texto".

Uma nova visão do relacionamento das gerências pode ser observado através do diagrama de atividades representado na Figura 34.

[6]A extensão .mcw é um extensão de texto muito utilizada para Macintosh.

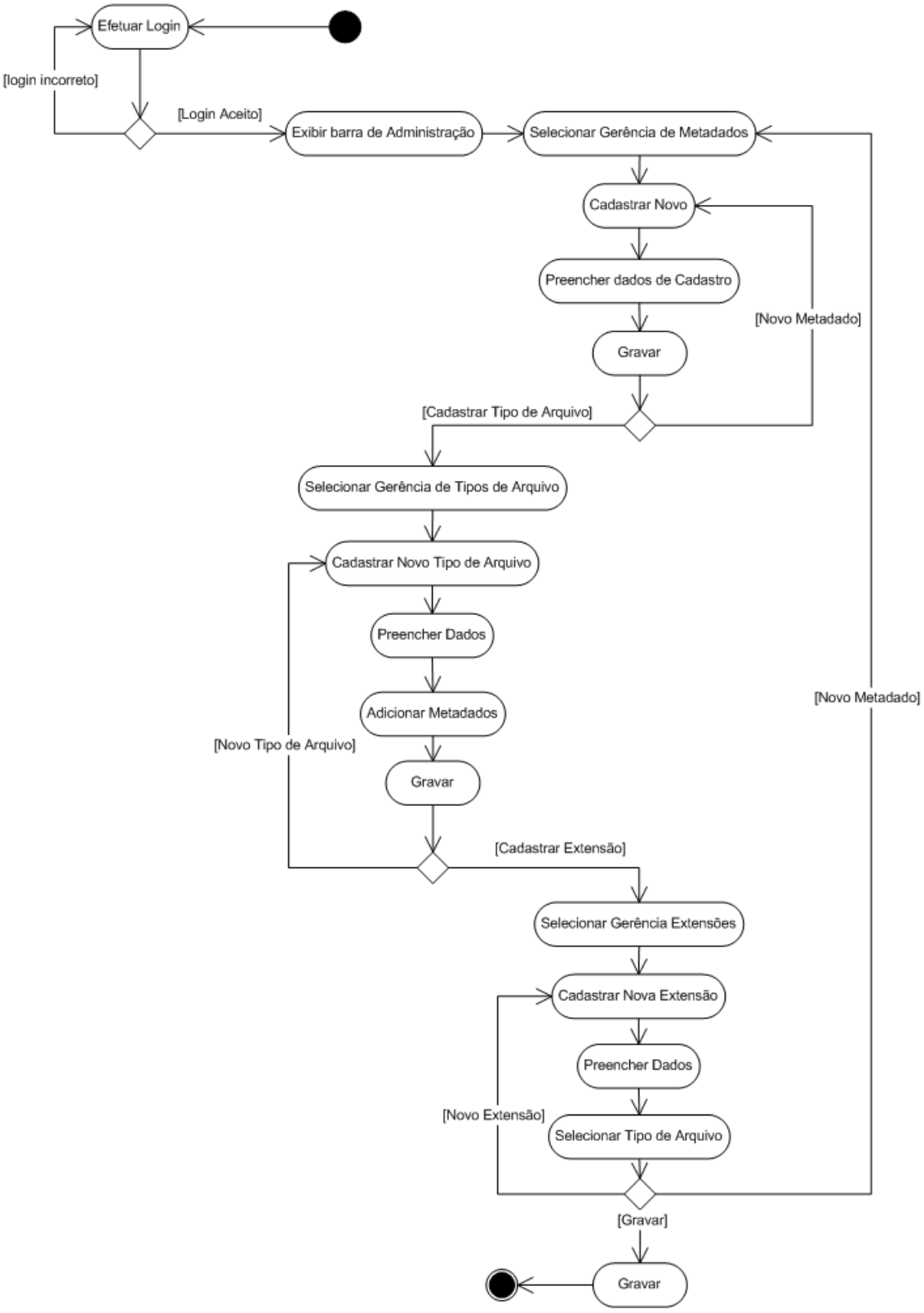

Figura 34: Adicionando Metadados ao Sistema - Diagrama de Atividades

É possível visualizar a mesma situação, acompanhando os diagramas de sequência, onde primeiramente é cadastrado o metadado (Figura 35), um tipo de arquivo é criado e associado a esse metadado (Figura 36), e uma nova extensão é cadastrada e associada ao tipo de arquivo escolhido (Figura 37).

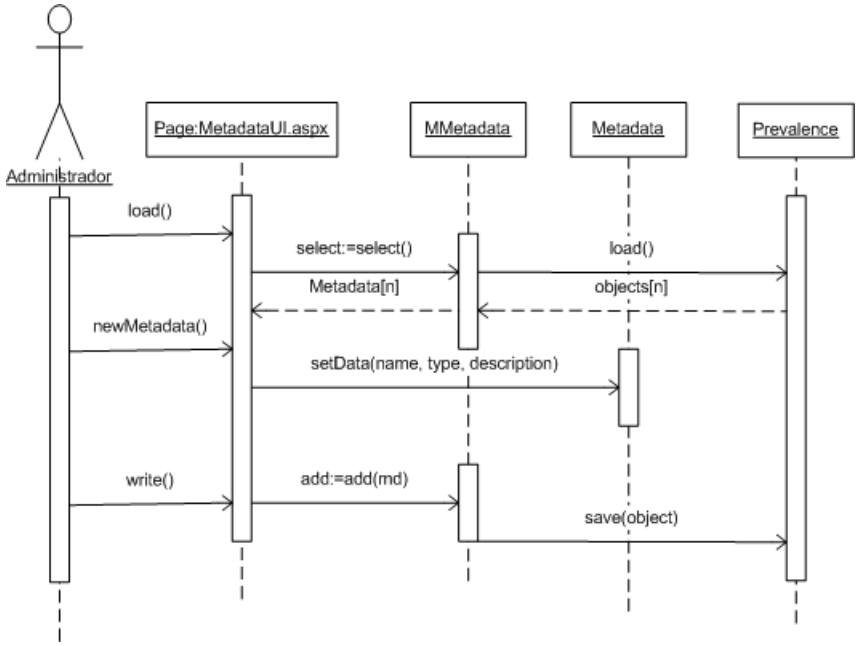

Figura 35: Cadastrar Metadados - Diagrama de sequência

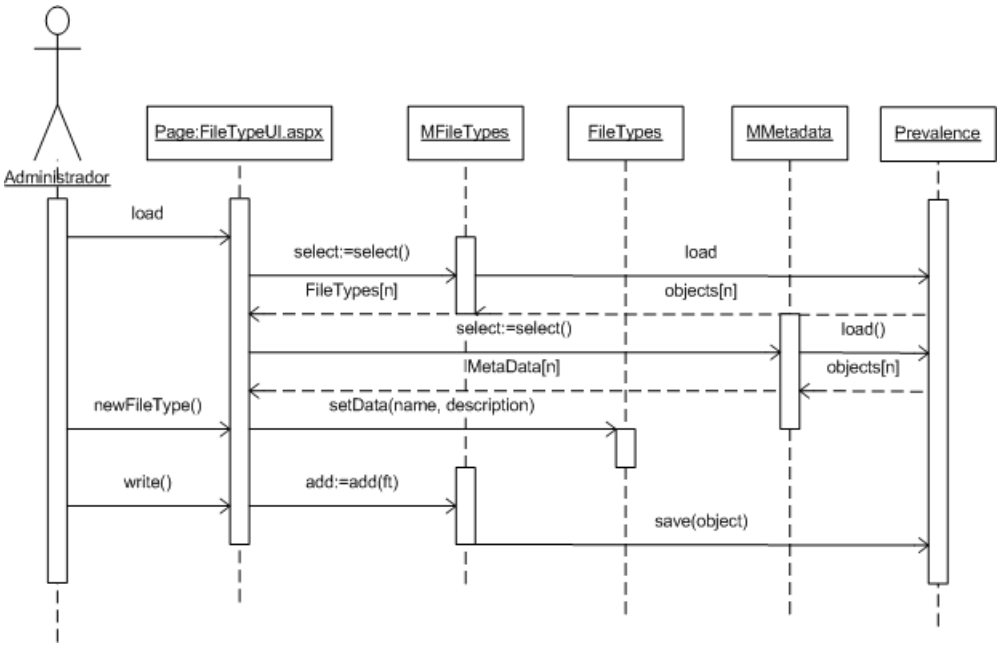

Figura 36: Cadastrar Tipos de Arquivo - Diagrama de sequência

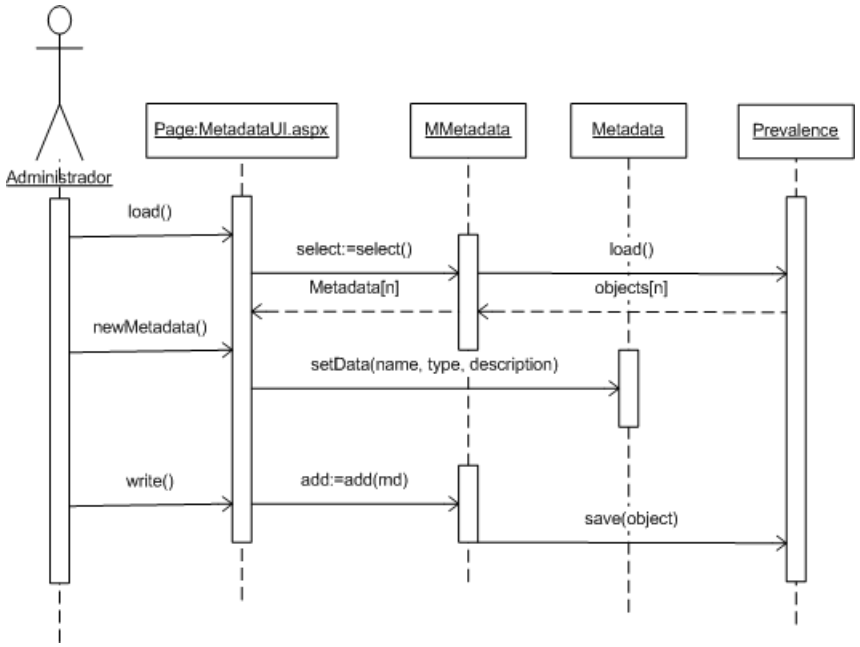

Figura 37: Cadastrar Extensões - Diagrama de sequência

7.2 Funcionalidades de Usuário

7.2.1 Listar, Remover e Baixar Arquivos

Nesta seção serão abordadas as funcionalidades de Listar Arquivos, Remover Arquivos e Baixar Arquivos, por possuírem uma proximidade muito grande e estarem envolvidas com as mesmas interfaces.

Ao Listar Arquivos, o usuário dependendo de seu papel no grupo, tem acesso a controles que possibilitam baixar ou remover estes.

Quando um usuário deseja baixar ou mesmo remover um arquivo, ele obrigatoriamente deve conseguir acessá-lo. Isso ocorre quando o usuário lista os arquivos de um determinado grupo. Para listar os arquivos, além de efetuar login no sistema, é preciso ter alguma permissão neste grupo, caso contrário o usuário não poderá listar os arquivos, nem mesmo visualizar o grupo.

O diagrama de casos de uso Baixar Arquivos é demonstrado na Figura 38, as especificações de caso de uso Baixar Arquivos e Listar Arquivos podem ser vistas nas Tabelas 11 e 12.

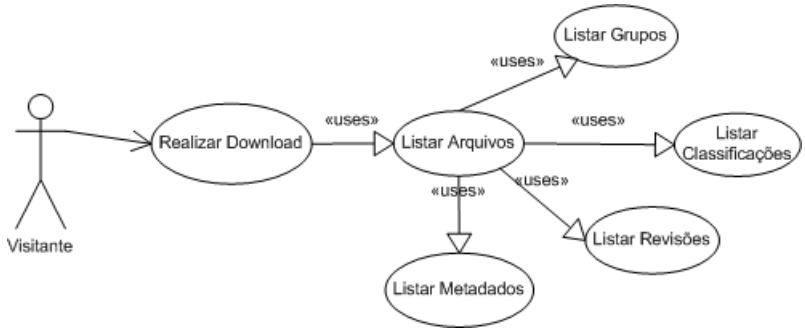

Figura 38: Baixar Arquivos - UC11

Tabela 11: Especificação de Caso de Uso - Baixar Arquivos

Caso de Uso: Baixar Arquivos
ID: UC11
Descrição: Através deste caso de uso o usuário poderá baixar arquivos do sistema, podendo ainda selecionar que revisão deseja baixar.
Ator: Visitante, Editor, Revisor, Gerente.
Fluxo Normal: 1. Efetuar login no sistema 2. Acessar grupo ou classificação 3. Listar Arquivos 4. Selecionar Revisão 5. Baixar Arquivo
Fluxo Alternativo: 1. Login Incorreto

Tabela 12: Especificação de Caso de Uso - Listar Arquivos

Caso de Uso: Listar Arquivos
ID: UC11.2
Descrição: Através deste caso de uso serão exibidos os arquivos do sistema, com as devidas permissões para cada usuário.
Ator: Visitante, Editor, Revisor, Gerente.
Fluxo Normal: 1. Verificar Usuário e seus Papéis 2. Exibir Arquivos segundo visualização (Grupos ou Classificações) 3. Listar Arquivos segundo os papeis do usuário 4. Habilitar controles para usuário
Fluxo Alternativo:

Como foi possível observar na Tabela 11 e no diagrama da Figura 38, Listar Arquivos e Baixar Arquivos são fundamentais para o funcionamento do GEDi, e simples de serem utilizadas pelos usuários. A funcionalidade Remover Arquivos é tão simples de ser utilizada quanto estas. O diagrama de caso de uso Remover Arquivos pode ser visto na Figura 39 e sua especificação na Tabela 13

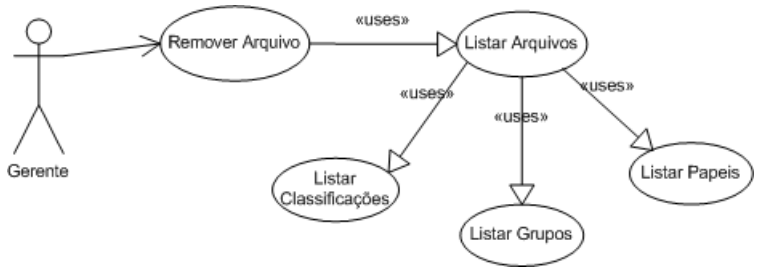

Figura 39: Remover Arquivos - UC13

Tabela 13: Especificação de Caso de Uso - Remover Arquivos

Caso de Uso: Remover Arquivos
ID: UC11.2
Descrição: Através deste caso de uso os arquivos poderão ser removidos do sistema, levando em conta o papel do usuário.
Ator: Gerente.
Fluxo Normal: 1. Efetuar login no sistema. 2. Acessar grupo ou classificação. 3. Listar Arquivos. 4. Remover Arquivo. 5. Confirmar Remoção.
Fluxo Alternativo: 1. Login Incorreto. 2. O Gerente pode não confirmar a exclusão. Neste casa o arquivo não é removido.

Para exemplificar as funcionalidades Remover Arquivo e Baixar Arquivo, um usuário recebeu o papel de Visitante no Grupo “Desenvolvimento” e o papel de Gerente, no grupo “Administrativo”. É possível notar uma pequena diferença entre a interface da Figura 40 que representa o grupo onde o usuário é apenas visitante e a Figura 41, onde ele é Gerente.

O papel de Gerente é o único que possibilita ao usuário remover[7] arquivos do sistema, pois os demais papéis (Visitante, Revisor e Editor) habilitam apenas o controle para baixar arquivo(Figura 40).

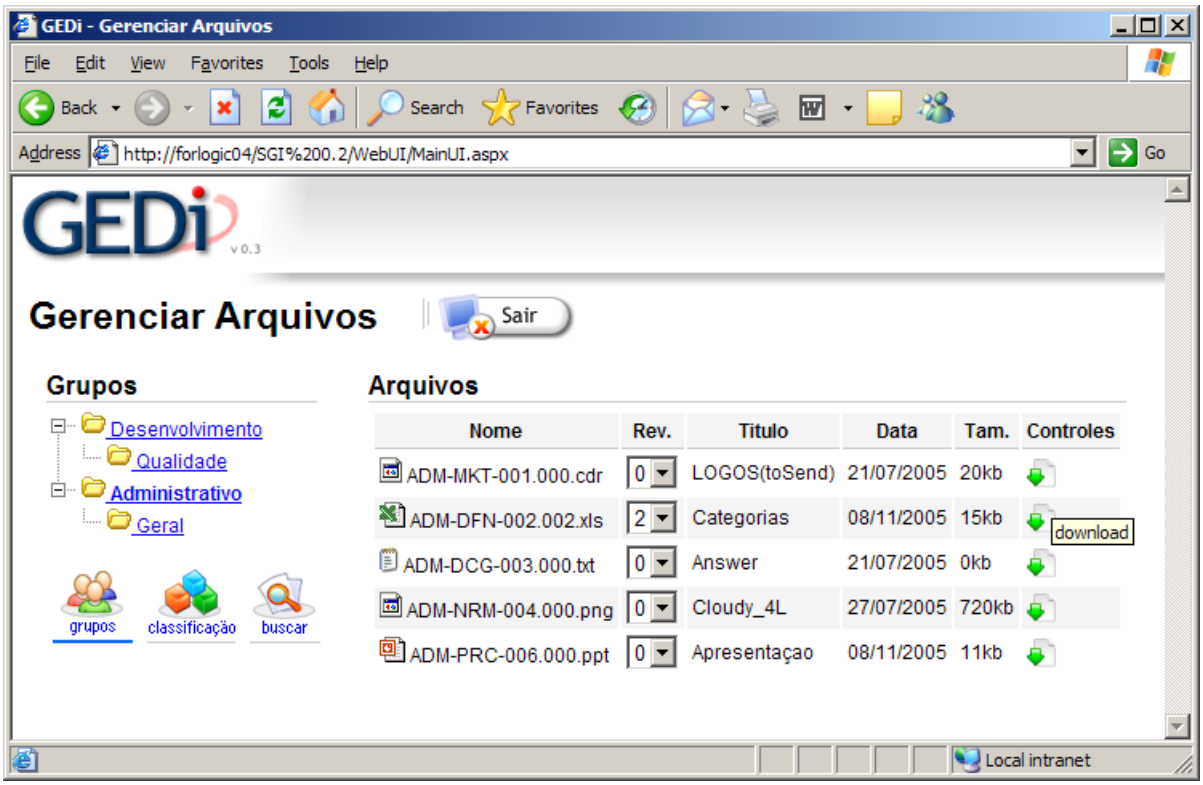

Figura 40: Habilitado apenas o controle para baixar arquivos - Captura de Tela

[7] É possível apagar apenas arquivos do grupo onde esse usuário tem o papel de gerente.

O usuário com papel de Gerente possui um controle representado por uma folha de papel com um “X” vermelho na frente de cada documento do grupo. Esse controle é utilizado para remover arquivos permanentemente do sistema (Figura 41).

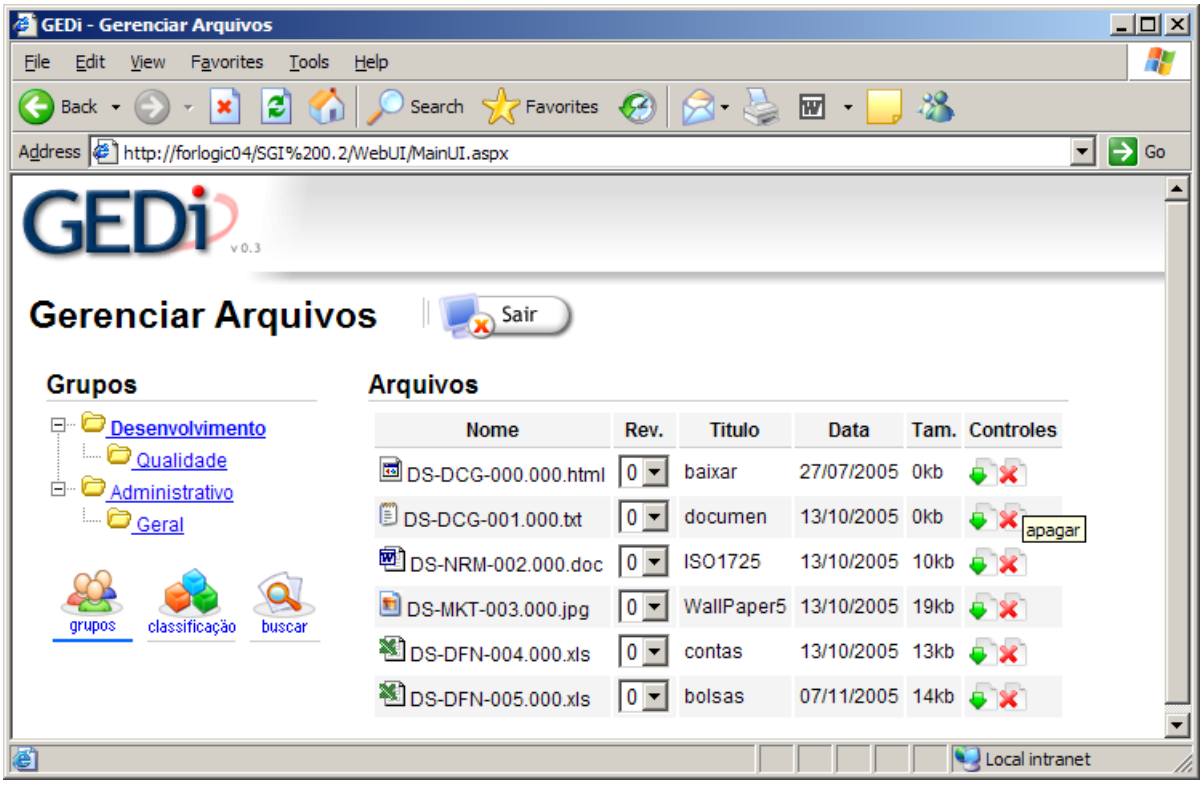

Figura 41: Habilitado também o controle para apagar arquivos - Captura de Tela

Todos os papéis do sistema possibilitam que o usuário baixe arquivos. Para issoum controle com uma folha de papel e uma seta em verde apontando para baixo é exibido em frente a todo o documento. Esse controle é utilizado para realizar o download do mesmo.

Em ambos os casos demonstrados nas figuras 40 e 41, tanto para Baixar Arquivos quanto para Remover Arquivos, um único clique no controle é necessário para realizar a operação, porém, uma confirmação é exigida no caso do Remover arquivo, como pode ser visualizado na Figura 42.

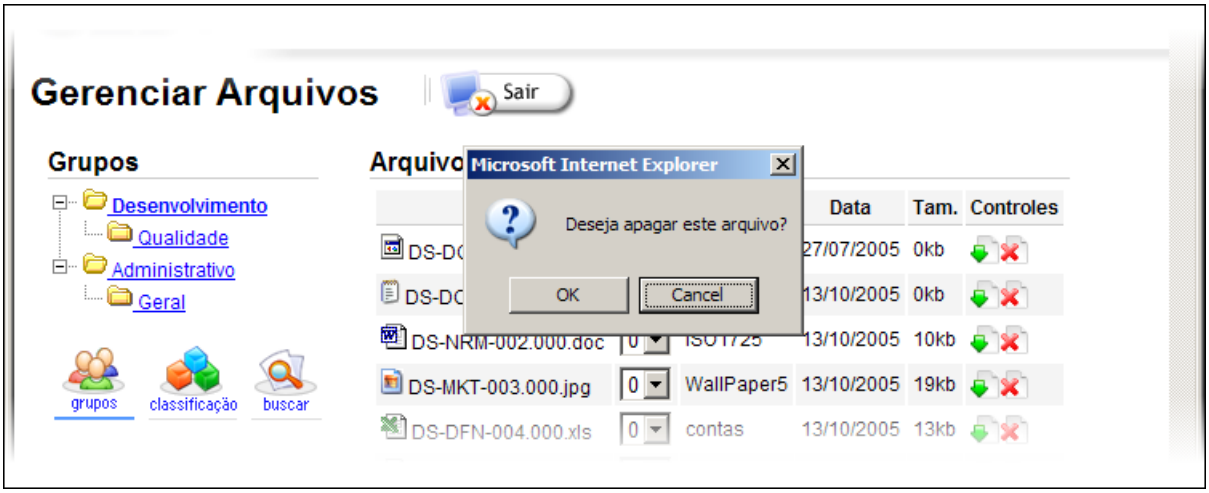

Figura 42: Confirmação de exclusão de arquivo - Captura de Tela

Uma outra visão sobre as funcionalidades de Baixar Arquivos e Remover Arquivos pode ser obtida observando os diagramas de sequência representados nas Figuras 43 e 44.

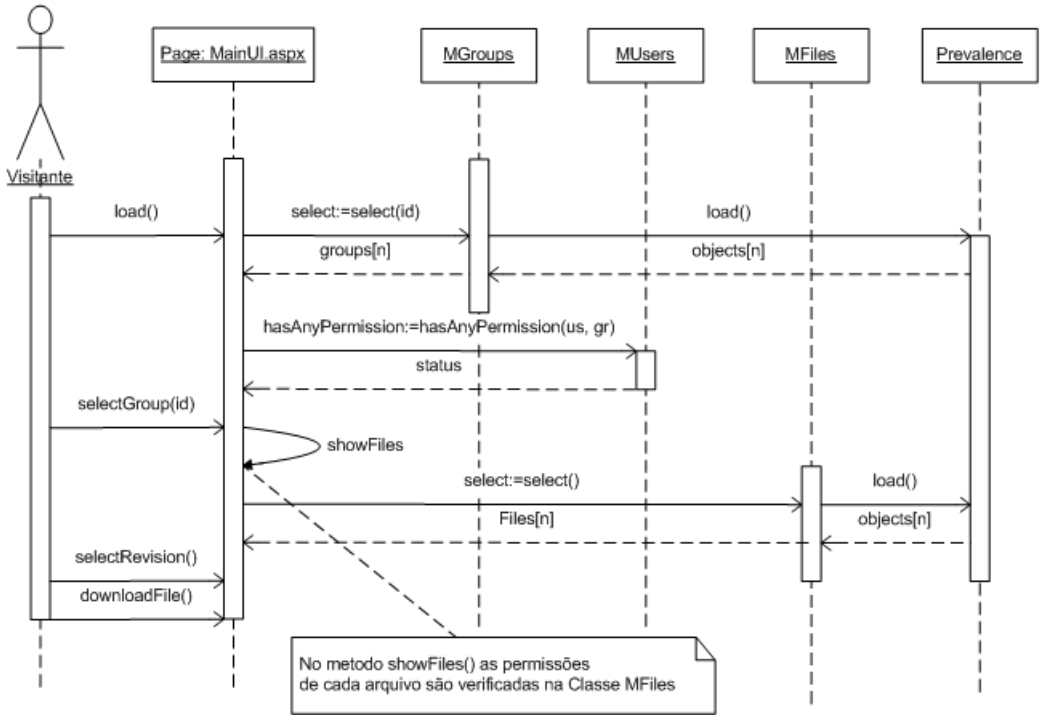

Figura 43: Baixar Arquivos - Diagrama de Sequência

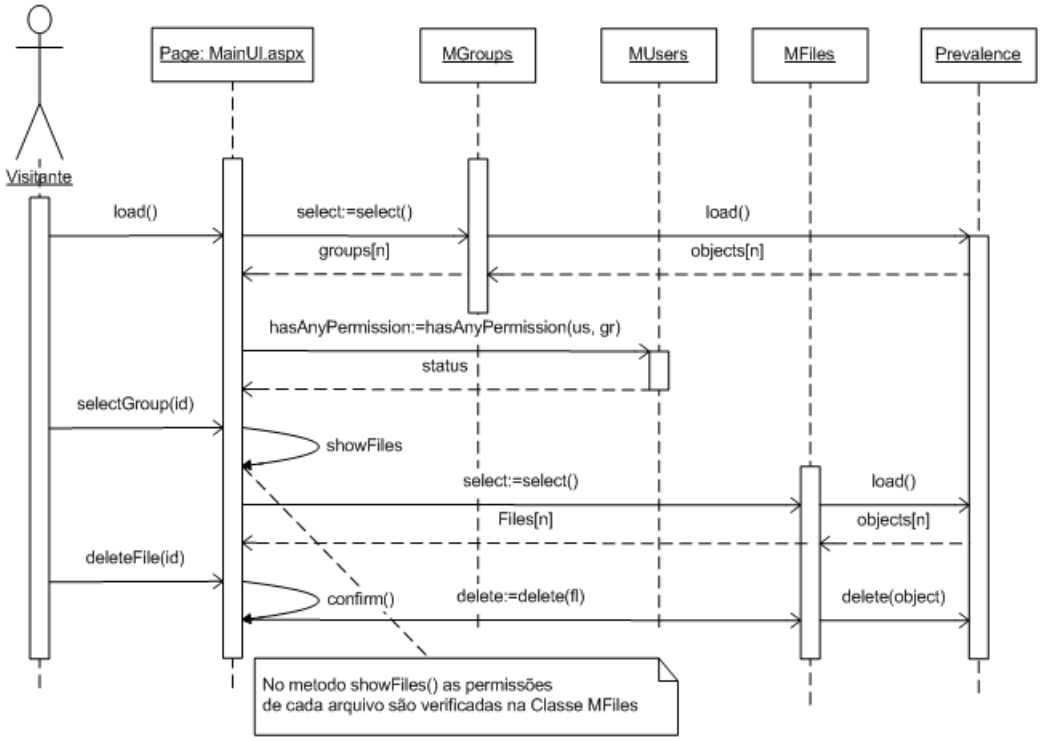

Figura 44: Remover Arquivos - Diagrama de Sequência

Na Figura 45 é possível observar como o usuário pode escolher a revisão que deseja baixar e após isso basta baixar o arquivo como demonstrado anteriormente.

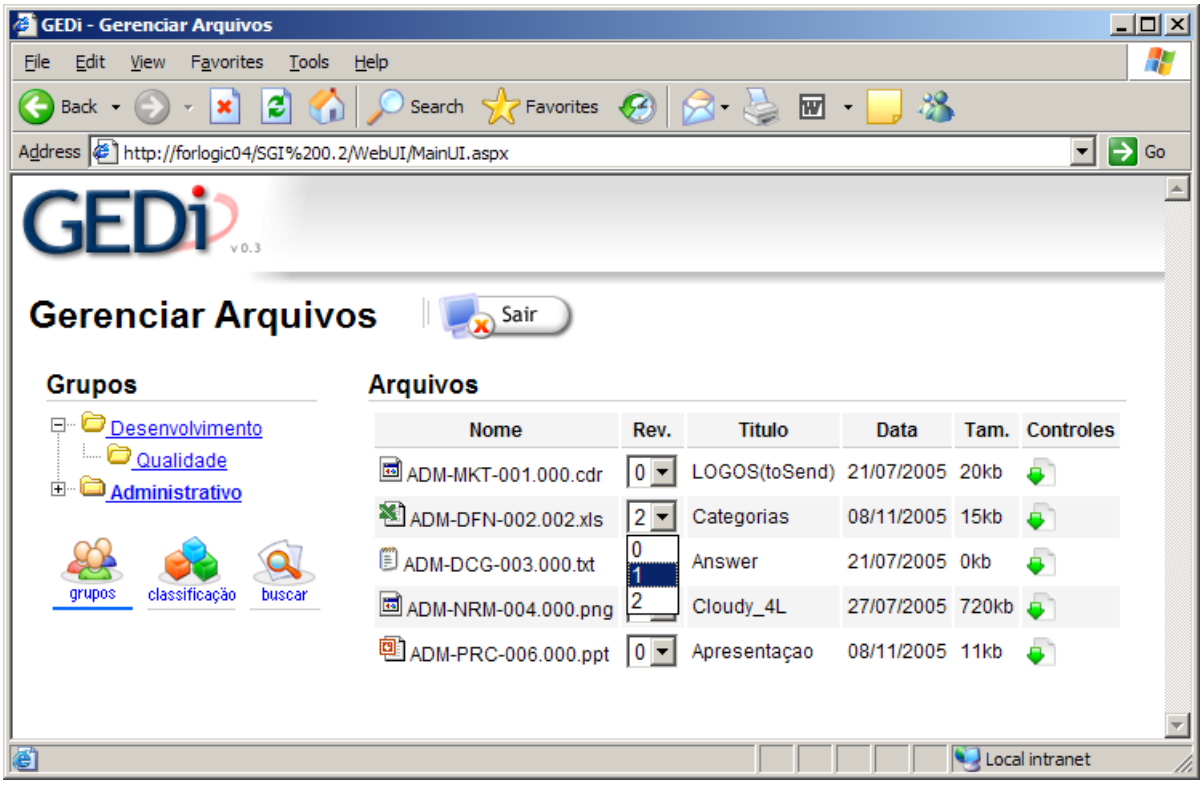

Figura 45: Selecionando Revisão - Captura de Tela

O diagrama de atividades representado na Figura 46 demonstra como um arquivo pode ser baixado pelo usuário, dando a possibilidade de escolher a revisão que deseja baixar.

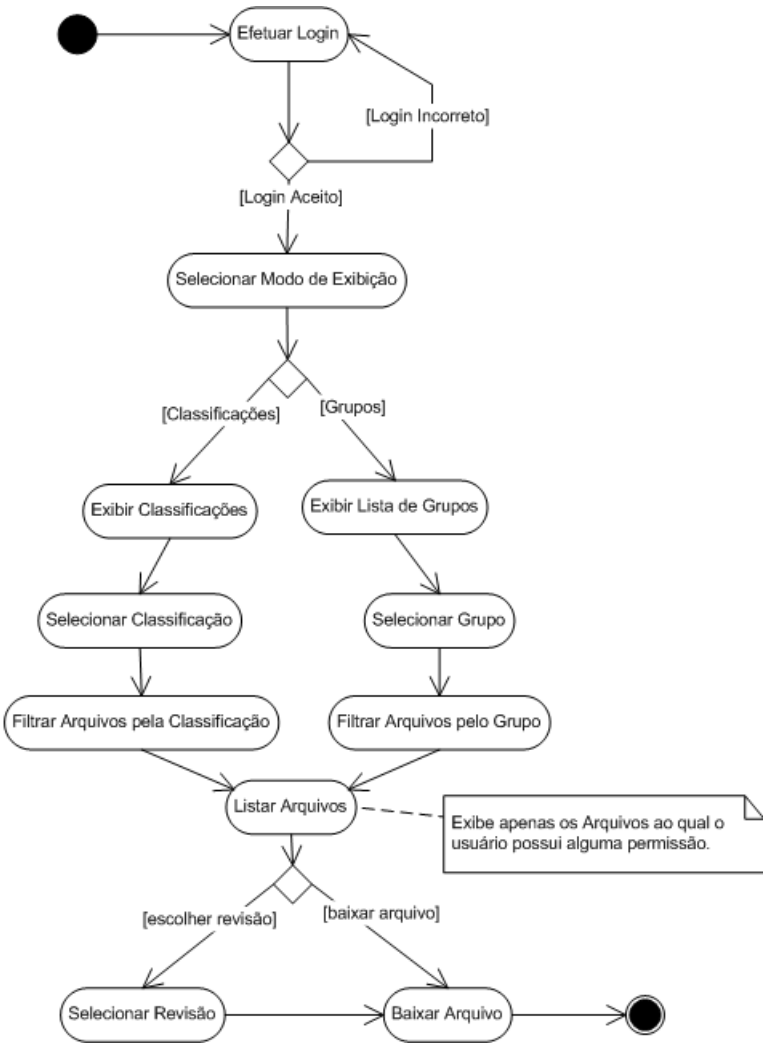

Figura 46: Baixando Arquivos - Diagrama de Atividades

7.2.2 Inserir Arquivos e Inserir Revisões

As duas funcionalidades “Inserir Arquivos” e “Inserir Revisões” serão demonstradas na mesma seção por possuírem diagramas em comum.

Para inserir arquivos novos ao sistema, o usuário que estiver utilizando o GEDi deverá ser obrigatoriamente, Editor ou Gerente no grupo a que ele deseja enviar os arquivos. No momento que o usuário acessar um grupo ao qual ele possui um dos dois papéis mencionados, um botão “inserir arquivos” é exibido na tela principal de utilização do sistema. Como no detalhe da Figura 47.

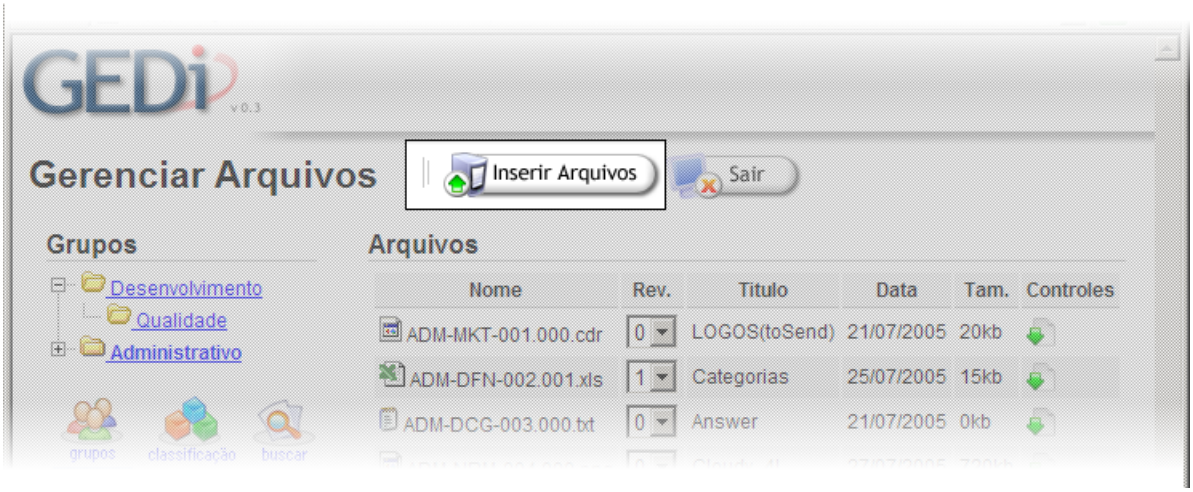

Figura 47: Inserir Arquivos - Captura de Tela

Da mesma forma, se o usuário tiver o papel de revisor, uma opção “Inserir Revisão” será exibida na pela principal do sistema, como pode ser observado na Figura 48.

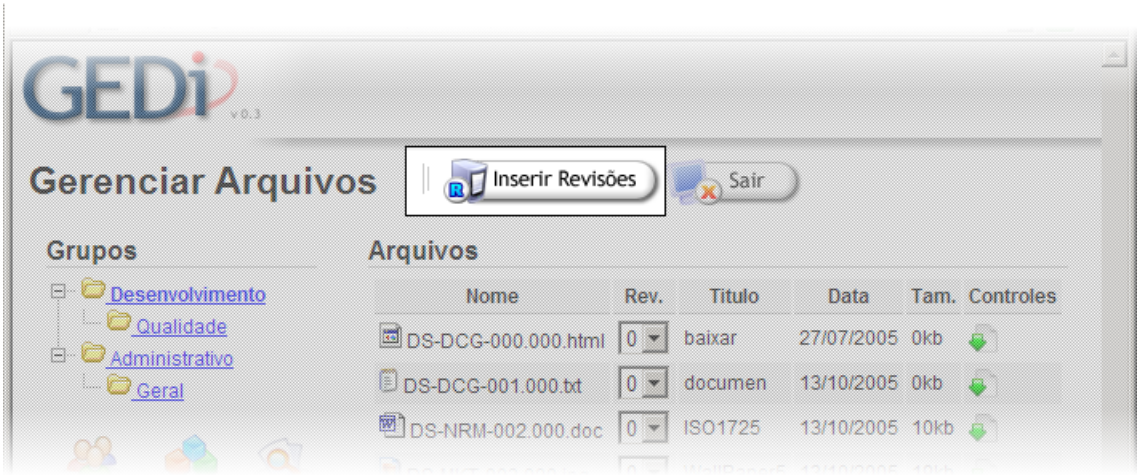

Figura 48: Inserir Revisões - Captura de Tela

A Figura 49 demonstra a interface “Enviar Arquivos” que é utilizada para inserir novos documentos e revisões no sistema. Sempre que um usuário clicar nos botões “Inserir Arquivos” ou “Inserir Revisões” demonstrados anteriormente ele será levado para a interface onde pode enviar arquivos.

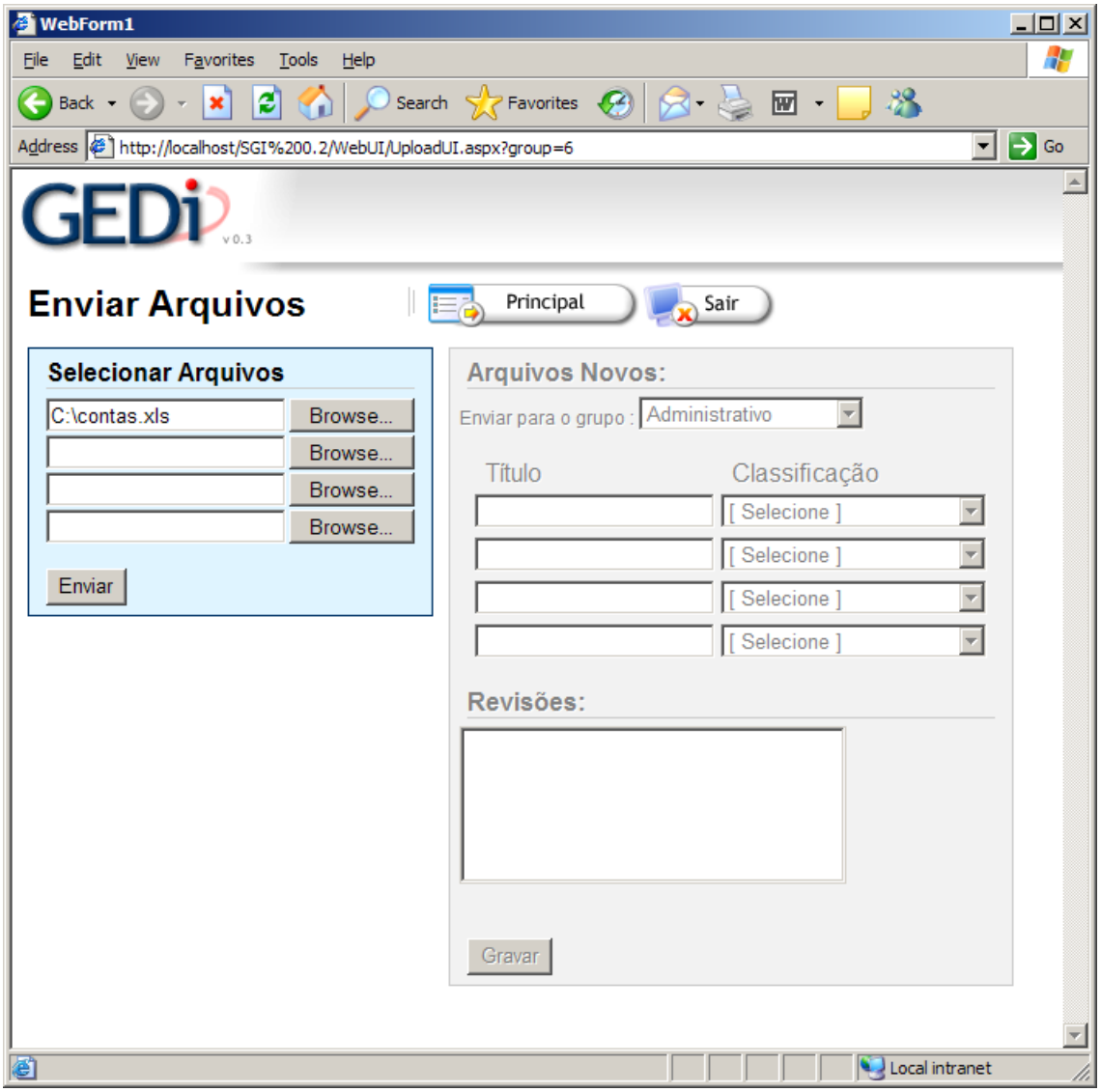

Figura 49: Interface Enviar Arquivos utilizada para inserir novos arquivos e revisões

A Figura 50 demonstrar um usuário com papel de gerente enviando dois arquivos, um deles uma revisão e outro um novo arquivo. É possível observar que após submetidos ao sistema (clicando enviar) o quadro a direita é habilitado, o novo arquivo fica na parte superior e o arquivo que é revisão fica na lista de revisões.

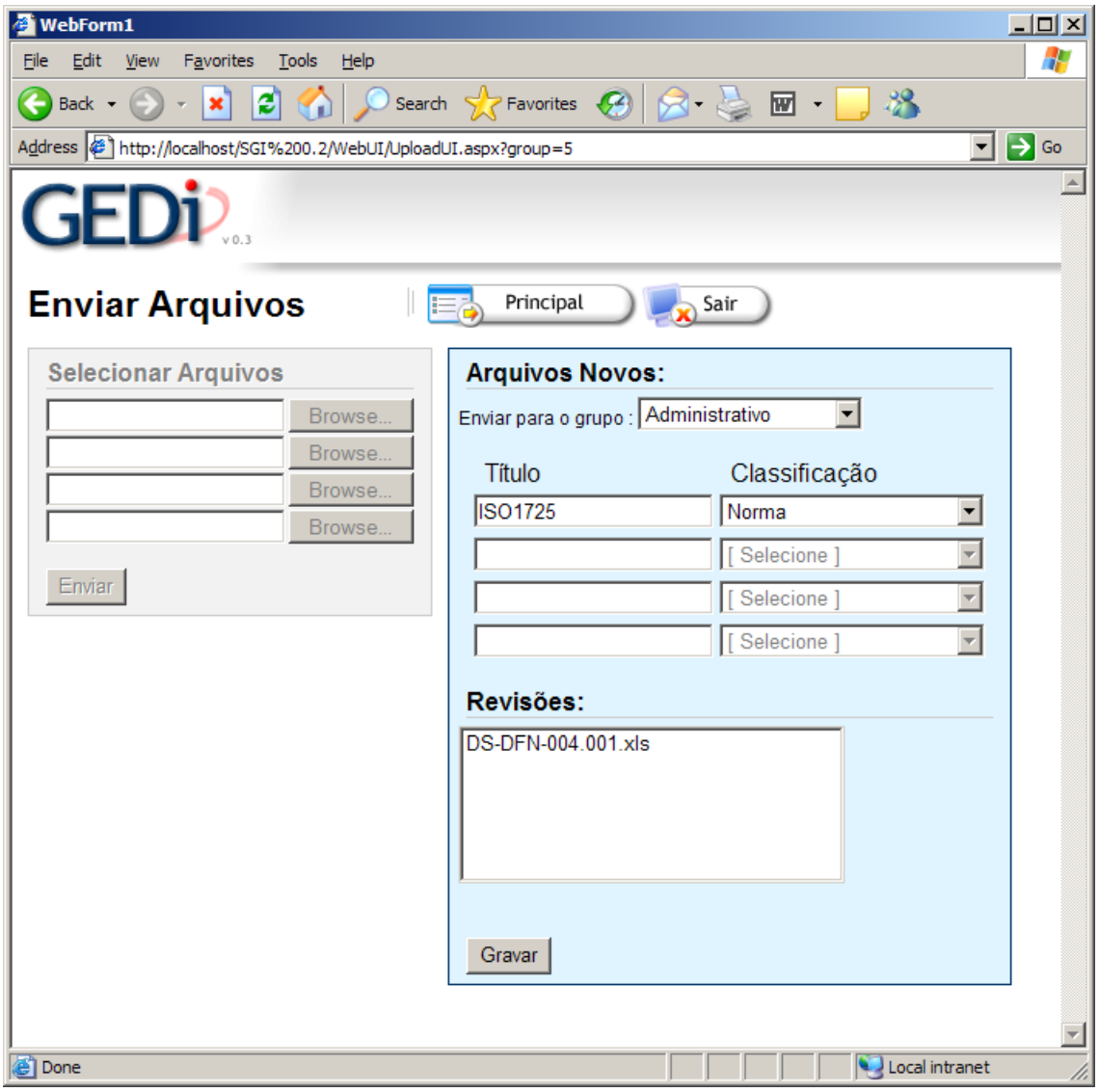

Figura 50: Interface Enviar Arquivos - inserindo arquivo e revisão

Os Casos de Uso referentes a inserção de arquivos e revisões estão demonstrados nas Figuras 51 e 52 respectivamente.

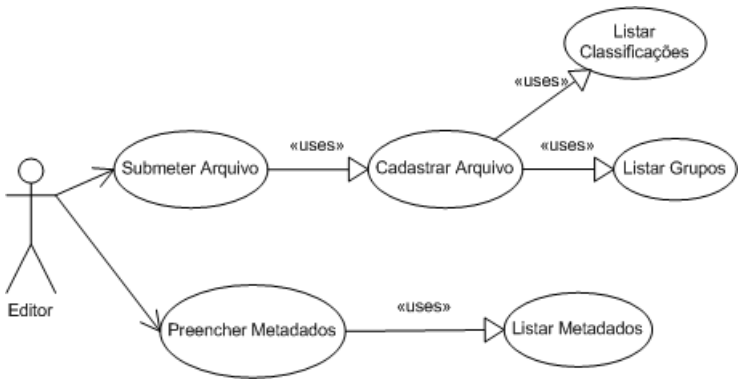

Figura 51: Inserir Arquivos - UC9

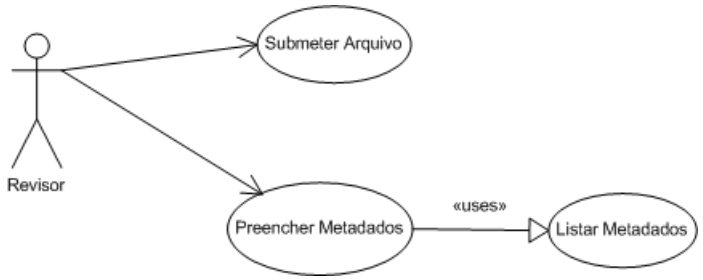

Figura 52: Inserir Revisões - UC10

As especificações dos diagramas de caso de uso Inserir Arquivos (Tabela 14) e Inserir Revisões (Tabela 15), são sucintos, pois um detalhamento maior acontece nas especificações seguintes, que podem ser observados nas tabelas 16, 17 e 18.

Tabela 14: Especificação de Caso de Uso - Inserir Arquivos

Caso de Uso: Inserir Arquivos
ID: UC9
Descrição: Através deste caso de uso o usuário poderá inserir arquivos novos no sistema, selecionando sua classificação, o grupo a que pertencem e preenchedo seus metadados.
Ator: Editor, Gerente.
Fluxo Normal: 1. Efetuar login no sistema 2. Acessar grupo 3. Inserir Arquivos
Fluxo Alternativo: 1. Login Incorreto

Tabela 15: Especificação de Caso de Uso - Inserir Revisões

Caso de Uso: Inserir Revisões
ID: UC10
Descrição: Através deste caso de uso o usuário poderá inserir revisões no sistema e preencher seus metadados.
Ator: Revisor, Gerente.
Fluxo Normal: 1. Efetuar login no sistema 2. Acessar grupo 3. Inserir Revisões
Fluxo Alternativo: 1. Login Incorreto

Tabela 16: Especificação de Caso de Uso - Submeter Arquivo

Caso de Uso: Submeter Arquivo
ID: UC9.1
Descrição: Através deste caso de uso o usuário poderá selecionar o arquivo que deseja inserir no sistema e realizar o upload do arquivo.
Ator: Editor, Gerente, Revisor
Fluxo Normal: 1. Clicar no botão "Browse..."[8] 2. Selecionar o arquivo. 3. Clicar em "Abrir". 4. Submeter arquivo (upload).
Fluxos Alternativos 1. O arquivo selecionado pode ter sido apagado antes de ser submetido, neste caso uma mensagem informará que o arquivo não foi encontrado.

Tabela 17: Especificação de Caso de Uso - Cadastrar Arquivo

Caso de Uso: Cadastrar Arquivo
ID: UC9.2
Descrição: Através deste caso de uso o usuário poderá preencher os dados do arquivo submetido ao servidor para inserí-lo no Sistema.
Ator: Editor, Gerente
Fluxo Normal: 1. Selecionar o grupo do arquivo. 2. Preencher os dados do arquivo (Título). 3. Selecionar Classificação do arquivo. 4. Gravar
Fluxos Alternativos 1. Os dados podem estar incompletos. Uma mensagem solicitará que os dados necessários sejam adicionados.

[8] "Procurar..." ou "Browse..." dependendo do idioma do *browser*

Tabela 18: Especificação de Caso de Uso - Preencher Metadados

Caso de Uso: Preencher Metadados
ID: UC9.3
Descrição: Através deste caso de uso o usuário poderá preencher os metadados de um arquivo.
Ator: Editor, Gerente, Revisor
Fluxo Normal: 1. Cadastrar Arquivo. 2. Selecionar arquivo. 3. Preencher dados do metadado. 4. Gravar
Fluxos Alternativos 1. Os dados podem estar preenchidos de forma errada. Uma mensagem solicitará que os dados necessários sejam preenchidos corretamente.

É possível ter uma outra visão das funcionalidades Inserir Arquivos e Inserir Revisões observando os digramas representados nas Figuras 53 e 54.

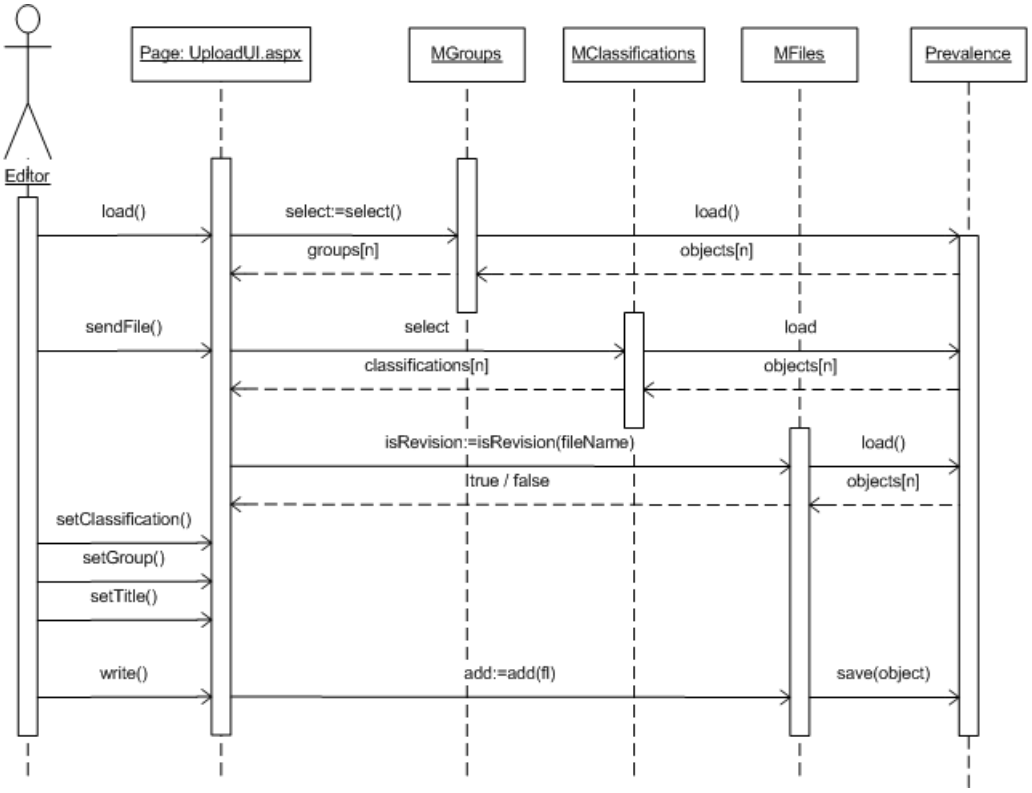

Figura 53: Inserir Arquivos - Diagrama de Sequência

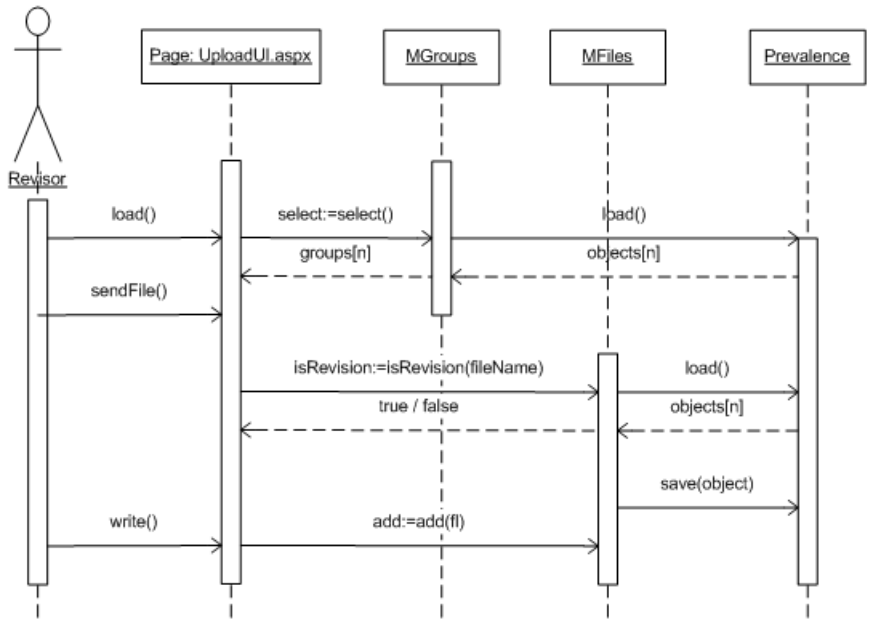

Figura 54: Inserir Revisões - Diagrama de Sequência

8 *Conclusão*

Este trabalho descreveu o desenvolvimento de uma ferramenta para gerenciamento eletrônico de documentos, utilizando tecnologias alternativas, com o intuíto de incentivar a pesquisa e a busca da inovação. Isso realmente ocorreu, e hoje a prevalência de dados não é mais uma incógnita para a equipe da ForLogic Softwares, os testes unitários já estão sendo utilizados na maioria dos projetos em desenvolvimento, o Mono é visto como uma alternativa muito mais viável e não apenas como uma promessa.

Algumas práticas de eXtreme Programming, estão se tornando práticas obrigatórias a serem utilizadas pelos desenvolvedores da ForLogic Softwares, assim como o ASP.NET é hoje uma tecnologia conhecida pela equipe de desenvolvimento.

As empresas que utilizarem o GEDi poderão compartilhar seus documentos e informações entre todos os colaboradores. Além de armazenar os documentos, o GEDi proporciona a possibilidade de funcionários agregarem informações a um documento através dos metadados, que podem ser preenchidos para cada novo documento que for inserido no sistema.

O GEDi cumpre o papel de Gerenciador Eletrônico de Documentos e informações, possibilitando que os documentos sejam organizados em grupos e localizados através de buscas dinâmicas. Estas buscas procuram nos títulos dos documentos, nomes de arquivos, palavras-chave, e ainda, nos metadados cada documento.

O objetivo de gerar conhecimento sobre novas tecnologias foi alcançado durante o desenvolvimento desse projeto, e uma ferramenta importante para o acervo de produtos da empresa, o GEDi, foi iniciada.

8.1 Resultados Atingidos

Algumas práticas da XP foram aplicadas e compreendidas pela a ForLogic Softwares, foi possível observar que ela pode ser utilizada como um processo de apoio muito interessante no desenvolvimento de software.

Uma camada para acesso a prevalência foi desenvolvida pela ForLogic Softwares para funcio-

nar como facilitador para sistemas que utilizam o Bamboo.Prevalence.[1]

A primeira versão funcional o GEDi foi construída e está sendo testada na empresa, muitas modificações e melhorias já foram identificadas e deverão ser implementadas nas próximas versões do sistema.

8.2 Trabalhos Futuros

Nesta seção, uma breve descrição das futuras funcionalidades que podem ser implementadas no GEDi são apresentadas.

Realizar *Check-in* e *Check-out* é muito uma funcionalidade importante, onde assim que um usuário ralizar *Check-out* em um documento, este fica bloquedo até o momento em que o usuário ralizar o *Check-in*. Essa funcionalidade irá evitar que um usuário pegue um documento e o altere, no mesmo momento que outro usuário já o está alterando.

Uma camada desktop de acesso também será interessante para facilitar a utilização do sistema, agregando funcionalidades que o navegador web não disponibiliza, como por exemplo, clicar o botão direito sobre um arquivo na área de trabalho e enviá-lo ao sistema.

Uma gerência de modelos é interessante, ao passo assim como o administrador cadastra atualmente, metadados, extensões, entre outros, ele poderá cadastrar modelos, e os usuários terão acesso a tais modelos.

Atualmente o sistema não possui um controle de logs, que será importante para detectar usuários mal intencionados e atividades em alta no sistema, essa é uma funcionalidade útil para a geração de gráficos estatísticos e para a segurança do GEDi, poderá ser implementada no futuro.

Um controle e acompanhamento de *Workflow* deverá ser implementado para que o GEDi ganhe essa nova utilização em breve, mas muita coisa deve ser feita para que isso seja alcançado. O acompanhamento de *Workflow* servirá para que um fluxo de trabalho possa ser determinado e seguido dentro da instituição para a criação de documentos.

Outra utilização que o GEDi pode assumir, com mudanças menores, é que o GEDi pode ser usado como ferramenta para submissão de trabalhos e documentos, em que prazos possam ser estipulados para aceitar e as datas das submissões sejam arquivadas.

[1] Essa camada está disponível para dowload em : ....

Referências

AHMED, M. et al. ***ASP.NET***: Web Developer's Guide. 1st. ed. [S.l.]: Syngress, 2002.

ALBAHARI, B.; DRAYTON, P.; MERRILL, B. ***C# Essentials***. 2nd. ed. [S.l.]: O'Reilly, 2001.

ARLOW, J.; NEUSTADT, I. ***UML and the Unified Process***: Pratical Oriented-Object Analysis and Design. 1st. ed. Great Britain: Addison-Wesley, 2002.

ASTELS, G. M. D.; NOVAK, M. *Extreme Programming*: Guia prático. 1st. ed. [S.l.]: Campus, 2002.

BAMBOO.PREVALENCE. 2005. Disponível em: <http://bbooprevalence.sourceforge.net/>. Acesso em: agosto, 2005.

BECK, K. ***Programação Extrema (XP) Explicada***. 1. ed. Porto Alegre: Bookman, 2000.

DAVENPORT, T.; PRUSAK, L. ***conhecimento empresarial***. 1. ed. Rio de Janeiro: Campus, 1998.

DEBONI, J. E. Z. ***Modelagem orientada a objetos com a UML***: the Unified Modeling Language. 1. ed. São Paulo: Futura, 2003.

DEITEL, H. et al. ***C# How to Program***. 1st. ed. [S.l.]: Prentice Hall, 2001.

EXTREMEPROGRAMMING.ORG. 2005. Disponível em: <http://www.extremeprogramming.org>. Acesso em: julho, 2005.

FURLAN, J. D. ***Modelagem de Objetos através da UML***: the Unified Modeling Language. 1. ed. São Paulo: Makron Books, 1998.

GAMMA, E. et al. ***Padrões de Projeto***: Soluções Reutilizáveis de Software Orientado a Objetos. 1st. ed. [S.l.]: Bookman, 2000.

GOLDMAN, A. et al. Electronic states of V-shaped semiconductor quantum wires in electric fields. *SBC Journal*, v. 58, p. 15381–15384, 1999.

HOHMANN, L. ***Beyond Software Architecture***: Creating and Sustaining Winning Solutions. 1st. ed. [S.l.]: Addison-Wesley, 2003.

HUNT, A.; THOMAS, D. ***Pragmatic Unit Testing***: in C# with NUnit. 1. ed. Dallas: The Pragmatic Programmers, 2004.

KRUGLIANSKAS, I.; TERRA, J. C. C. ***Gestão do conhecimento em pequenas e médias empresas***. 2. ed. Rio de Janeiro: Negócio, 2003.

MARTINS, A. Arquitetura de sistemas para gestão do conhecimento. *CONGRESSO ANUAL DA SOCIEDADE BRASILEIRA DE GESTÃO DO CONHECIMENTO*, São Paulo, v. 3, 2002.

MONO PROJECT - LANGUAGES. 2005. Disponível em: <http://www.mono-project.com/Languages>. Acesso em: agosto, 2005.

PRESS, M. *Microsoft .NET Framework e Aplicativos Web*. 1st. ed. [S.l.]: Campus, 2001.

PREVAYLER. 2005. Disponível em: <http://www.prevayler.org>. Acesso em: julho, 2005.

REILLY, D. J. ***Designing Microsoft(r) ASP.NET Applications***. 1st. ed. [S.l.]: Microsoft Press, 2002.

ROCHA, H. V. da; BARANAUSKAS, M. C. C. ***Design e Avaliação de Interfaces Humano-Computador***. 1st. ed. [S.l.]: Unicamp, 2003.

SOMMERVILLE, I. ***Software Engineering***. 6st. ed. England: Addison-Wesley, 2002.

WAKE, W. C. ***Extreme Programming Explored***. 1. ed. Boston: Addison-Wesley Professional, 2000.