IVAN CAMOLESI

GERENCIADOR DE PROJETO UML PARA FÁBRICA DE SOFTWARE

ASSIS
2009

GERENCIADOR DE PROJETO UML PARA FÁBRICA DE SOFTWARE

IVAN CAMOLESI

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador:_____

Analisador (1):_____

Analisador (2):_____

ASSIS
2009

IVAN CAMOLESI

GERENCIADOR DE PROJETO UML PARA FÁBRICA DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis, como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador:_____

Área de concentração:_____

_____

ASSIS
2009

DEDICATORIA

Dedico esse trabalho a minha família, a minha namorada ao meu orientador e a todos que me ajudaram nessa caminhada de cinco anos.

AGRACECIMENTOS

Agradeço a todos os professores e todos meus amigos e especialmente a minha namorada que sempre me apoiou muito.

RESUMO

Este trabalho aborda uma pesquisa, para o desenvolvimento de uma aplicação para gerenciar os documentos de análise orientado objeto, com o intuito de organizar e arquivar toda a documentação de análise de uma fábrica de software, através de interface web para facilitar o acesso e a busca de documentos.

Palavras-chave: Fábrica de Software. Gerenciar documentos. Web.

ABSTRACT

This paper addresses a study for the development of an application to manage the documents of object oriented analysis, in order to organize and archive all the documents of analysis of a factory of software, through of the web interface to facilitate access and search for documents.

Keywords: Factory of Software. Manage the documents. Web.

LISTA DE ILUSTRAÇÔES

LISTA DE ABREVIATURAS E SIGLAS

OMG	Object Management Group
OMT	Object Modeling Technique
OOSE	Objected Oriented Software Engineering
TI	Tecnologia da Informação
UML	Unified Modeling Language

SUMÁRIO

INTRODUÇÃO

Nos últimos anos o mercado de TI passou por grande mudança. Diante da necessidade de redução de custos e otimização no desenvolvimento de software, empresas adotaram a prática de terceirizar serviços que não são a sua especialidade ou não fazem parte do seu negócio.

Este trabalho tem por objetivo principal o desenvolvimento de uma aplicação para arquivar e organizar o material gerado por um processo de análise de sistema em uma fábrica de software.

O termo software factory (fábrica de software em inglês) foi empregado pela primeira vez em 1969, pela japonesa Hitachi®, mas só começou a ficar popular no início dos anos 90. A idéia era aplicar conceitos da indústria em geral em ambientes de desenvolvimento de software, de forma a aumentar a produtividade e diminuir prazos e custos. Utiliza em sua operação indicadores de qualidade e produtividade em cada etapa do ciclo de desenvolvimento de software, bem como busca maximizar a re-utilização de componentes anteriormente desenvolvidos (Fernandes, 2004).

O processo de desenvolvimento abrange a gerência de requisitos, o planejamento e acompanhamento do processo, a gerência de configuração, a gerência de qualidade, a codificação, os testes e a implantação.

Tendo em vista que as fábricas de software trabalham com grandes projetos ao mesmo tempo, que envolvem uma equipe grande de profissionais da área, que por decorrência geram uma documentação de analise enorme, julgou-se necessário desenvolver uma aplicação para arquivar e organizar essa documentação visando facilitar o acesso a esse tipo de documentação e com isso dar a fábrica ainda mais produtividade e organização. Este projeto será desenvolvido com base na tecnologia Java com a persistência no banco feita com o auxilio de Hibernate, e sua estrutura com base nos diagramas da Uml.

1. DESCRIÇÃO DOS CONCEITOS

Para compreender a aplicação temos que ter alguns conceitos básicos inseridos como a modelagem de sistemas utilizando a diagramação da UML, a forma de desenvolvimento de uma fábrica de software, à forma de como é feito o gerenciamento de requisitos, rastreabilidade entre outros conceitos.

1.1. CONCEITOS DE UML

A UML (Unified Modeling Language) é uma linguagem para especificação, documentação, visualização e construção de sistemas orientados a objetos, sendo considerada uma das linguagens para modelagem de sistemas orientados a objetos, mais expressiva de todos os tempos. Através dos diagramas oferecidos por essa linguagem, é possível representar sistemas de softwares sob diversas perspectivas de visualização de modo a fornecer a toda equipe envolvida (cliente, analista, programador, etc.) uma compreensão única do projeto.

A UML tem origem na compilação das "melhores práticas de engenharia" que provaram ter sucesso na modelagem de sistemas grandes e complexos. Sucedeu aos conceitos de BOOCH, OMT(Object Modeling Technique) e OOSE (Objected Oriented Software Engineering) fundindo-os numa única linguagem de modelagem comum e largamente utilizada. Em 1997, a UML v1.1 foi adotada pela OMG (Object Management Group) e desde então tornou-se o padrão da indústria de software para a modelagem de objetos e componentes.

A UML é um modelo de linguagem, não um método. Um método pressupõe um modelo de linguagem e um processo. O modelo de linguagem é a notação que o método usa para descrever o projeto. O processo são os passos que devem ser seguidos para se construir o projeto.

O modelo de linguagem é uma parte muito importante do método. Corresponde ao ponto principal da comunicação. Se uma pessoa quer conversar sobre o projeto,

como outra pessoa, é através do modelo de linguagem que elas se entendem. Nessa hora, o processo não é utilizado.

A UML define uma notação e um meta-modelo. A notação são todos os elementos de representação gráfica vistos no modelo (retângulo, setas, o texto, etc.), é a sintaxe do modelo de linguagem. A notação do diagrama de classe define a representação de itens e conceitos tais como: classe, associação e multiplicidade. Um meta-modelo é um diagrama de classe que define de maneira mais rigorosa a notação.

A UML permite avaliar a aderência e a qualidade da arquitetura através de iterações precoces com o usuário quando os defeitos podem ser corrigidos antes de comprometer o sucesso do projeto. Utilizando uma linguagem de modelagem padrão como a UML, os diferentes membros da equipe tanto de desenvolvimento como business podem comunicar suas decisões sem que haja ambigüidades ou diferenças de interpretação.

A modelagem visual permite que os detalhes do processo sejam expostos ou escondidos conforme a necessidade, auxiliando o desenvolvimento de projetos complexos e extensos. Além disto, a UML ajuda a manter a consistência entre a especificação e a implementação através do desenvolvimento iterativo e do planejamento de testes em cada iteração. Com o desenvolvimento focado no usuário e no business, o resultado final é diminuir o ciclo de vida e garantir a qualidade do sistema.

1.2. VISÕES E DIAGRAMAS DA UML

A arquitetura de um sistema pode ser descrita através de cinco visões interligadas. Cada visão constitui uma projeção na organização e estrutura do sistema, cujo foco está voltado para determinado aspecto desse sistema. A UML é uma linguagem muito expressiva, abrangendo todas as visões necessárias ao desenvolvimento e implantação de sistemas:

- **Visão de caso de uso:** focaliza os comportamentos de um sistema devendo ser transparente a todos os envolvidos: gerentes, analistas, programadores e usuários finais.
- **Visão de Projeto:** focaliza a estrutura de um sistema através da definição de classes, colaborações e as interfaces do sistema.
- **Visão de Processo:** focaliza as questões de desempenho e escalabilidade do sistema.
- **Visão de Implementação:** focaliza os artefatos físicos (programas, bibliotecas, banco de dados) para a efetiva montagem do sistema.
- **Visão de Implantação:** focaliza a topologia do hardware, liberação e instalação do sistema.

Um diagrama é a apresentação gráfica de um conjunto de elementos e são desenhados para permitir a visualização de um sistema sob diferentes perspectivas. A UML disponibiliza diagramas específicos para a modelagem visual das cinco visões:

- **Diagrama de casos de uso** para ilustrar as interações do usuário com o sistema, esse diagrama representa um conjunto de atores, casos de usos e os relacionamentos entre eles.
- **Diagrama de classe** para ilustrar a estrutura lógica. Encontrado na maioria dos sistemas orientados a objetos. São exibidos classes e seus respectivos relacionamentos.
- **Diagrama de objetos** para ilustrar os objetos e suas interações. O diagrama de objeto pode der considerada uma variação do diagrama de classes, pois utiliza quase todas suas notações. O objetivo do diagrama de objetos é mostrar as instâncias das classes de um sistema em determinado momento de execução desse sistema.

- **Diagrama de estados** para ilustrar comportamentos. O diagrama gráfico de estados possíveis de um objeto em particular. São exibidos os estados de um objeto, eventos, transição e atividades. Podem ser usados principalmente para a modelagem de estados de classes e colaborações.
- **Diagrama de componentes** para ilustrar a estrutura física do software. O diagrama de componentes representa um conjunto de componentes e suas respectivas dependências, podendo ter como base de sua construção os diagramas de classes.
- **Diagrama de Interações**: composto de diagrama de seqüência e diagrama de colaboração. Utilizado para ilustrar comportamentos.
- **O diagrama de seqüência** dá ênfase à ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Podemos entender por mensagens, os serviços solicitados por um objeto a outro, e as respostas envolvidas na suas solicitações.
- **Diagrama de Colaboração** da ênfase a ordenação estrutural em que as mensagens são trocadas entre os objetos de um sistema.
- **Diagrama de Atividades** para ilustrar o fluxo dos eventos. O diagrama de atividades representa a modelagem do fluxo de controle de uma atividade para uma outra no sistema. São exibidos os estados das atividades e ações, transições e objetos.
- **Diagrama de implantação** representa a configuração e a arquitetura de um sistema em que estarão ligados seus respectivos componentes, podendo ser representado também à arquitetura físicas de hardwares, processadores, etc. O diagrama de implantação abrange a visão estática de implantação de um sistema, envolvendo a modelagem da topologia de hardware em que o sistema, envolvendo a modelagem da topologia de hardware em que o sistema será executado.

A modelagem visual permite que os detalhes do processo sejam expostos ou escondidos conforme a necessidade, auxiliando o desenvolvimento de projetos complexos e extensos. Além disto, a UML ajuda a manter a consistência entre a especificação e a implementação através do desenvolvimento iterativo e do planejamento de testes em cada iteração. Com o desenvolvimento focado no

usuário e na regra de negócio, o resultado final é diminuir o ciclo de vida e garantir a qualidade do sistema.

1.3. FÁBRICA DE SOFTWARE

Nos últimos anos pôde-se perceber uma movimentação crescente no mercado de desenvolvimento de sistemas em direção ao modelo de denominado fábrica de software. O termo Fábrica de Software surgiu no mercado, como uma solução para alcançar maior produtividade e menor custo na produção de sistemas de software.

Muitos sistemas são desenvolvidos sem um projeto concreto, sem passar por todas as etapas existentes de uma Fábrica de Software; a conseqüência disso é que os sistemas podem ficar lentos, com bugs, sem possibilidade de realizar alterações ou atualizações futuras e, na maioria das vezes, é necessária até a reconstrução do projeto.

Um dos objetivos da Fábrica de Software é o desenvolvimento de projetos documentados para serem discutidos com os clientes e repassados a todos os colaboradores que vierem a construir softwares com qualidade. Apesar de todo software ser único na fábrica de software, recorre-se a reutilização de documentação, projetos e códigos de programação, tornando o trabalho mais produtivo e organizado. Com a constante evolução dos softwares e das tecnologias envolvidas no desenvolvimento de sistemas, a tendência é que a industrialização do sistema seja cada vez mais eficiente na produção de softwares com qualidade, em menos tempo e por baixo custo. Por este motivo, verifica-se o crescimento na adoção deste modelo para o desenvolvimento de sistemas.

Dentre as várias definições encontradas durante a pesquisa esta, me pareceu a mais completa e coerente:” Processo estruturado, controlado e melhorado de forma contínua, considerando abordagens de engenharia industrial, orientado para o atendimento a múltiplas demandas de natureza e escopo distintas, visando à geração de produtos de software, conforme os requerimentos documentados dos usuários e/ou clientes, da forma mais produtiva e econômica” (Fernandes, 2004).

1.4. GERENCIAMENTO DE REQUISITOS

O gerenciamento de requisitos é um modelo sistemático para encontrar, documentar, organizar e rastrear os requisitos variáveis de um sistema. Um requisito é definido como: Uma condição ou uma capacidade com a qual o sistema deve estar de acordo trata-se de um modelo sistemático para:

- Identificar, organizar e documentar os requisitos do sistema;
- Estabelecer e manter acordo entre o cliente e a equipe do projeto nos requisitos variáveis do sistema.

Os principais itens para o gerenciamento eficiente de requisitos incluem manter uma declaração clara dos requisitos, juntamente com atributos aplicáveis para cada tipo de requisito e rastreabilidade para outros requisitos e outros artefatos do projeto.

A coleta de requisitos pode parecer uma tarefa bem precisa. Nos projetos reais, contudo, você encontrará dificuldades porque:

- Nem sempre os requisitos são óbvios e podem vir de várias fontes.
- Nem sempre é fácil expressar os requisitos claramente em palavras.
- Existem diversos tipos de requisitos em diferentes níveis de detalhe.
- O número de requisitos poderá impossibilitar a gerência se não for controlado.
- Os requisitos estão relacionados uns com os outros, e também com o produto liberado do processo de engenharia do software.
- Os requisitos têm propriedades exclusivas ou valores de propriedade. Por exemplo, eles não são igualmente importantes nem igualmente fáceis de cumprir.
- Há várias partes interessadas, o que significa que os requisitos precisam ser gerenciados por grupos de pessoas de diferentes funções.
- Os requisitos são alterados.

O gerenciamento de requisitos corresponde ao conjunto de atividades que auxilia a equipe do projeto a identificar, controlar e rastrear os requisitos, bem como as alterações nos requisitos em muitos momentos do projeto. Em outras palavras, é o processo que gerencia mudanças nos requisitos de um sistema. Estas mudanças ocorrem conforme os clientes desenvolvem um melhor entendimento de suas reais

necessidades. Novos requisitos surgem e há alterações nos requisitos em todos os estágios do processo de desenvolvimento do sistema. São comuns os casos em que mais de 50% dos requisitos são alterados antes que o sistema seja posto em operação, o que causa sérios problemas para os desenvolvedores. Para minimizar dificuldades, os requisitos devem ser documentados e controlados, segundo *Rational Software – Conceitos: Gerenciamento de Requisitos* (2001).

1.5 RASTREABILIDADE

A rastreabilidade é a capacidade de rastrear um elemento do projeto a outros elementos correlatos, especialmente aqueles relacionados a requisitos. Os elementos do projeto envolvidos em rastreabilidade são chamados de itens de rastreabilidade. Os itens típicos de rastreabilidade incluem diferentes tipos de requisitos, elementos de modelo de design e de análise, artefatos de testes (conjuntos de testes, casos de teste, etc.) e material de treinamento e documentação de suporte a usuário final, como é mostrado na figura abaixo.

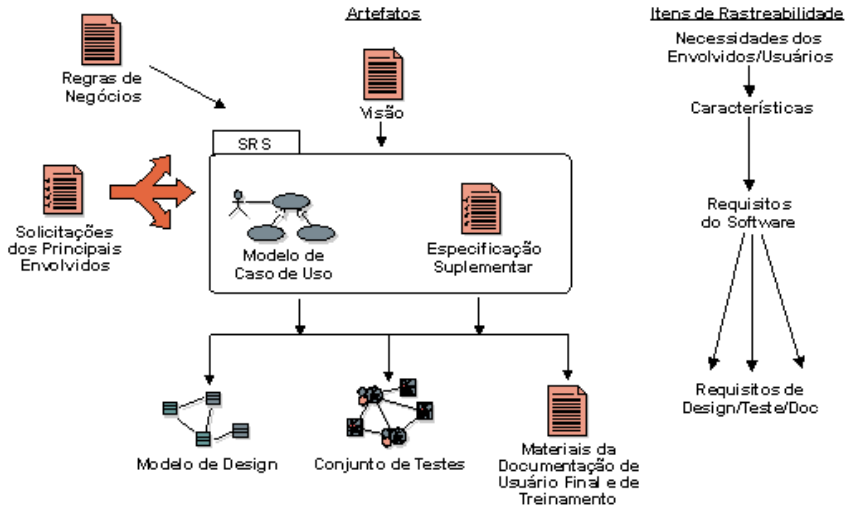

Figura 1 - A hierarquia da rastreabilidade.
(Rational Software Corporation)

A finalidade de estabelecer rastreabilidade é ajudar a:

- Compreender a origem dos requisitos
- Gerenciar o escopo do projeto
- Gerenciar mudanças nos requisitos
- Avaliar o impacto no projeto da mudança em um requisito
- Avaliar o impacto da falha de um teste nos requisitos (isto é, se o teste falhar, talvez o requisito não seja atendido);

- Verificar se todos os requisitos do sistema são desempenhados pela implementação;
- Verificar se o aplicativo faz apenas o que era esperado que ele fizesse.

Os requisitos não podem ser gerenciados de forma efetiva sem rastreabilidade. Um requisito é rastreável se for possível identificar quem solicitou o requisito, porque o requisito existe, quais os requisitos relacionados e como os requisitos se relacionam as outras informações como design de sistemas, implementações e documentos do usuário. Estas informações são utilizadas para identificar todos os requisitos afetados por mudanças propostas. Um conceito-chave para ajudar a gerenciar mudanças nos requisitos é o de um vínculo de rastreabilidade "suspeito"**.** Quando um requisito (ou outro item de rastreabilidade) muda em qualquer extremidade do vínculo de rastreabilidade, todos os vínculos associados àquele requisito são marcados como "suspeitos". Isso é uma marca para que o papel responsável analise a mudança e determine se os itens associados precisarão mudar também. Esse conceito também ajuda a analisar o impacto de mudanças potenciais.

De um modo mais simples, rastrear é manter os registros necessários para identificar e informar os dados relativos à origem e ao destino de um produto, segundo *Rational Software – Conceito de Rastreabilidade* (2001).

2. ANÁLISE DO PROJETO

Com base na pesquisa realizada viu-se a necessidade de uma aplicação para arquivar e organizar a documentação gerada pelo processo de análise de uma fábrica de software. Pois pelo fato de uma fábrica trabalhar com um grande número de projetos ao mesmo tempo, cada qual gerando uma grande documentação de análise a aplicação vem ser desenvolvida para armazenar e organizar essa documentação vindo assim a facilitar o acesso a esses documentos por toda equipe envolvida em determinado projeto, ganhando com isso muito mais velocidade em acesso a documentação e garantindo que diferentes membros da equipe tanto de desenvolvimento como de análise possam comunicar suas decisões sem que haja ambiguidades ou diferenças de interpretação.

Após levantamento de requisitos chegamos aos seguintes diagramas.

2.1. DIAGRAMA DE CASO DE USO

Após levantamento de requisitos chegamos ao seguinte diagrama de caso de uso, ilustrado pela figura 2, onde demonstramos todas as funcionalidades que poderão ser exercidas por um usuário no sistema.

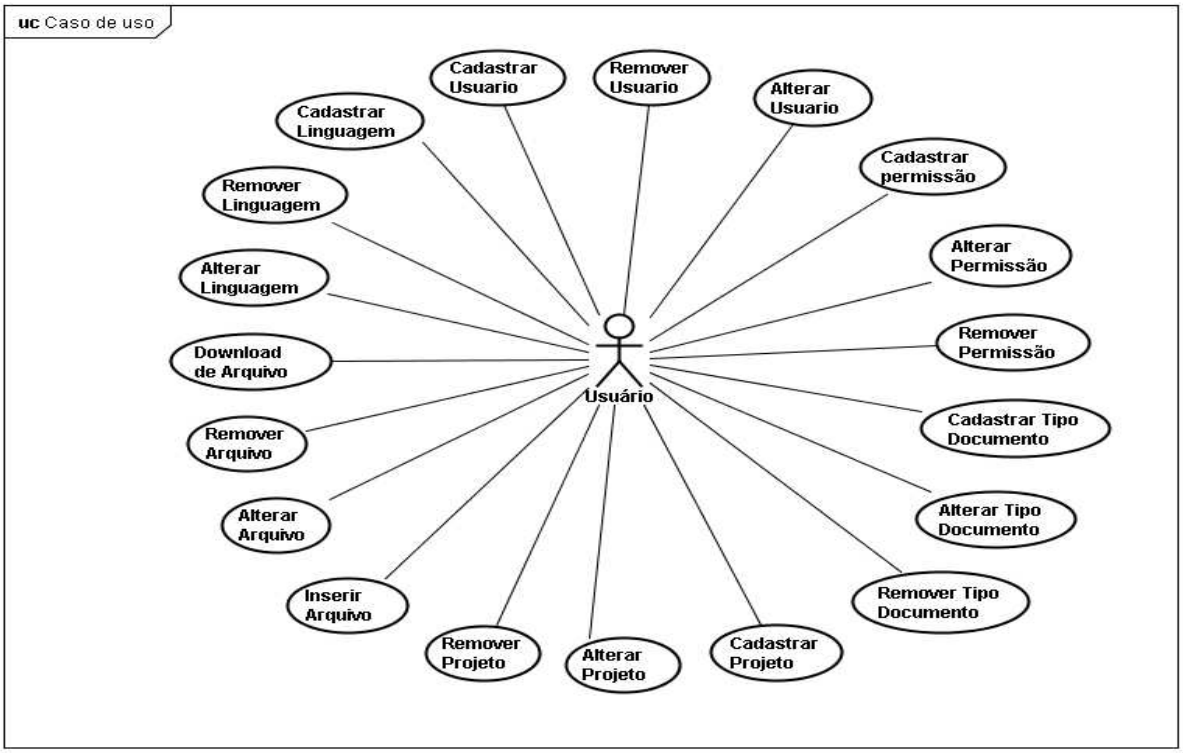

Figura 2. Diagrama de Caso de Uso.

O sistema proporcionará ao usuário, neste caso, funcionalidades para o gerenciamento de toda a aplicação. É possível que o usuário, com perfil administrador, gerencie as permissões que serão atribuídas a todos os usuários.

2.2. DIAGRAMA DE CLASSE

Observando o diagrama de caso de uso, chegamos ao seguinte diagrama de classe ilustrado pela figura 3.

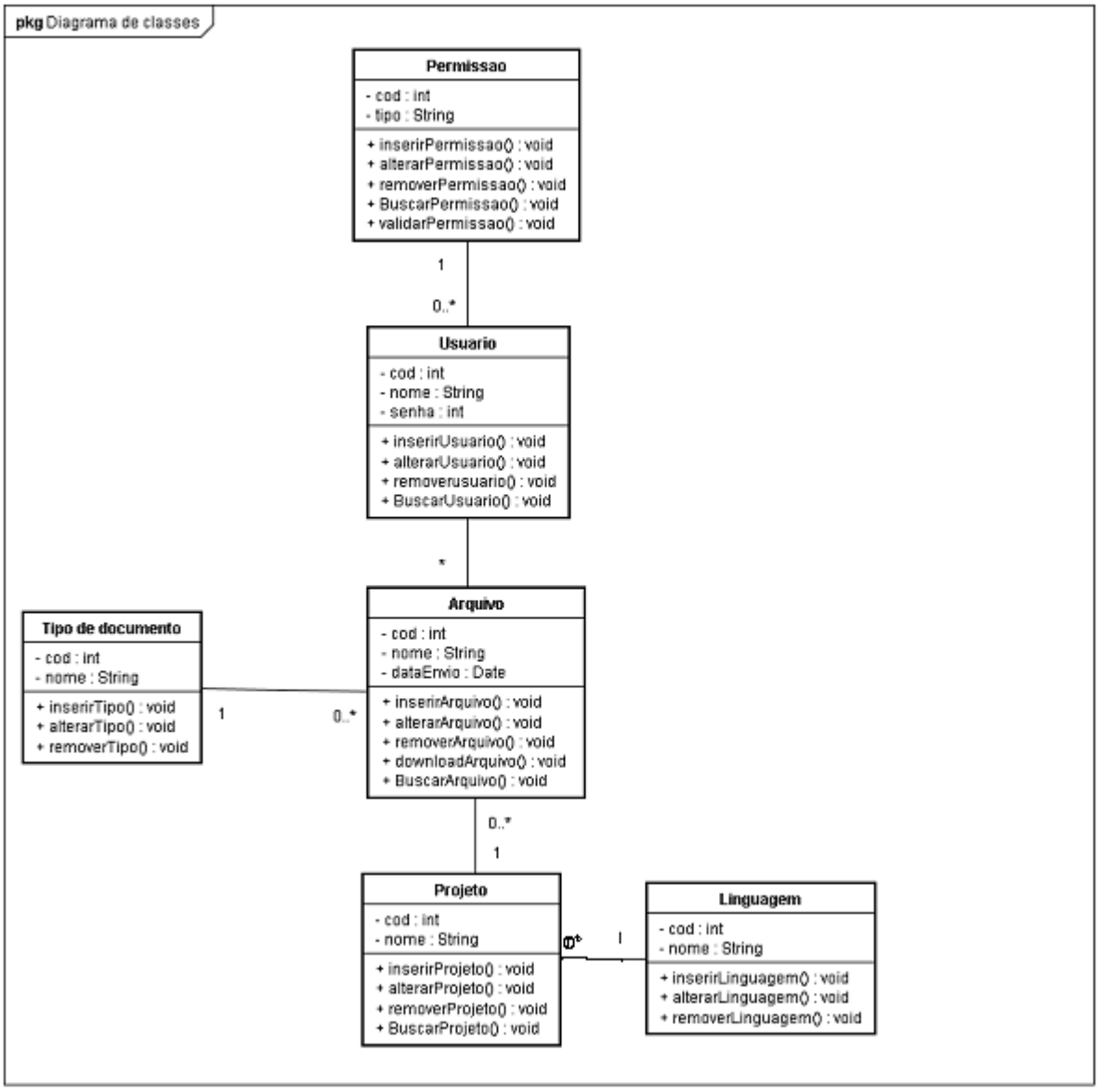

Figura 3. Diagrama de Classe.

Definido nosso diagrama de classe, passamos para as ações do sistema, através do diagrama de seqüência, onde é definido cada passo do sistema diante dos casos de uso existentes.

2.3. DIAGRAMA DE SEQUÊNCIA

O diagrama de seqüência nos mostra o comportamento do sistema perante solicitação de ações feita pelos usuários. As figuras abaixo demonstram os principais fluxos do sistema respectivamente de usuário, permissão, linguagem, projeto, tipo de documento e arquivo.

2.3.1. Usuário

O sistema de manipulação de usuários contará com as funções básicas de um cadastro, no nosso caso, podemos inserir, alterar e remover possíveis usuários do sistema, de acordo com a permissão do usuário.

Segue figura 4 ilustrando o cadastro de usuário no sistema.

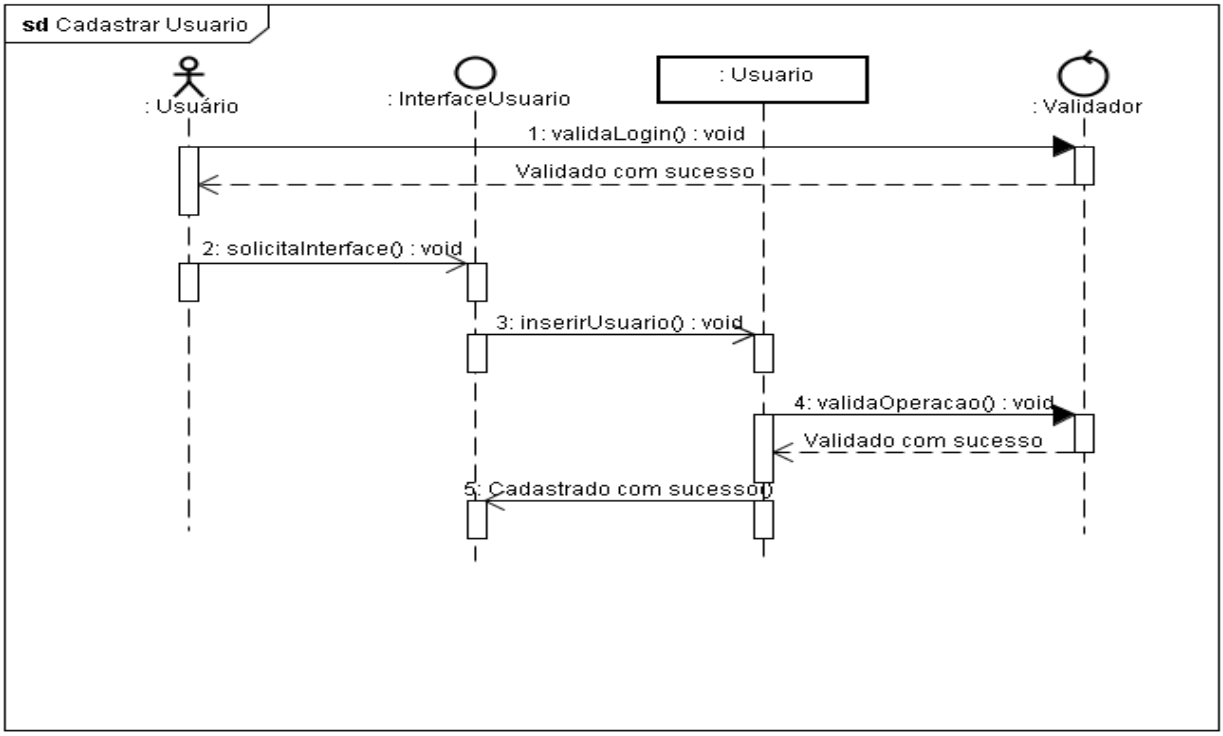

Figura 4. Diagrama de seqüência - Inserir usuário Fluxo normal.

Segue figura 5 ilustrando a alteração de usuário no sistema.

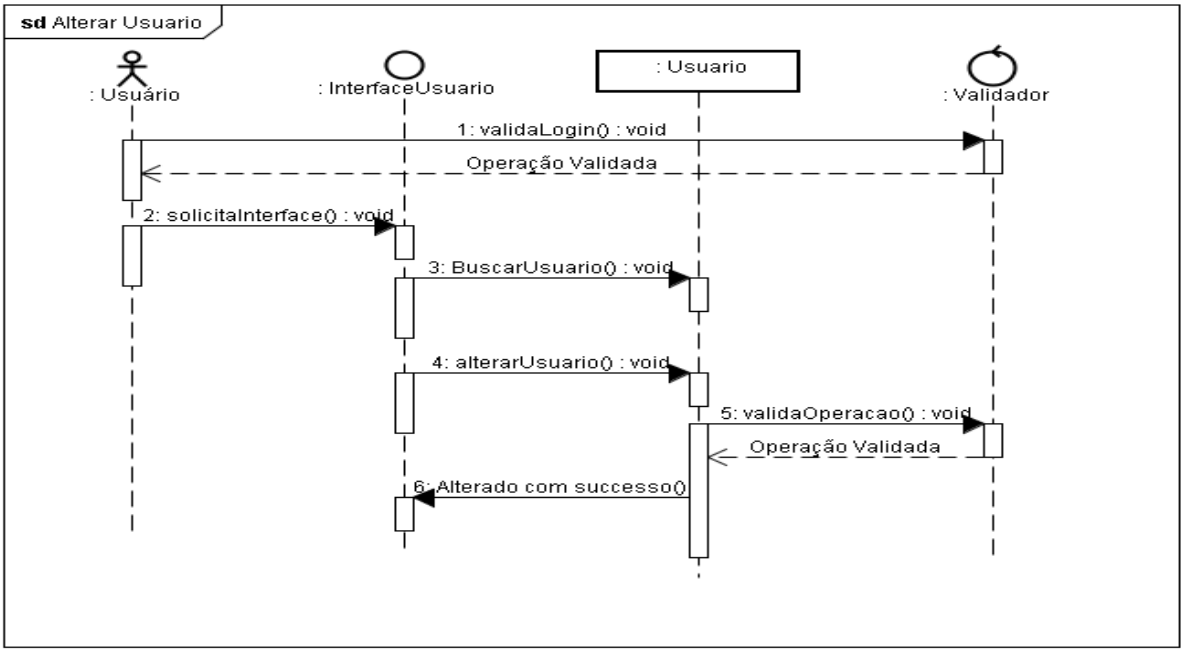

Figura 5. Diagrama de seqüência - Alterar usuário Fluxo normal.

Segue figura 6 ilustrando a remoção de usuário no sistema.

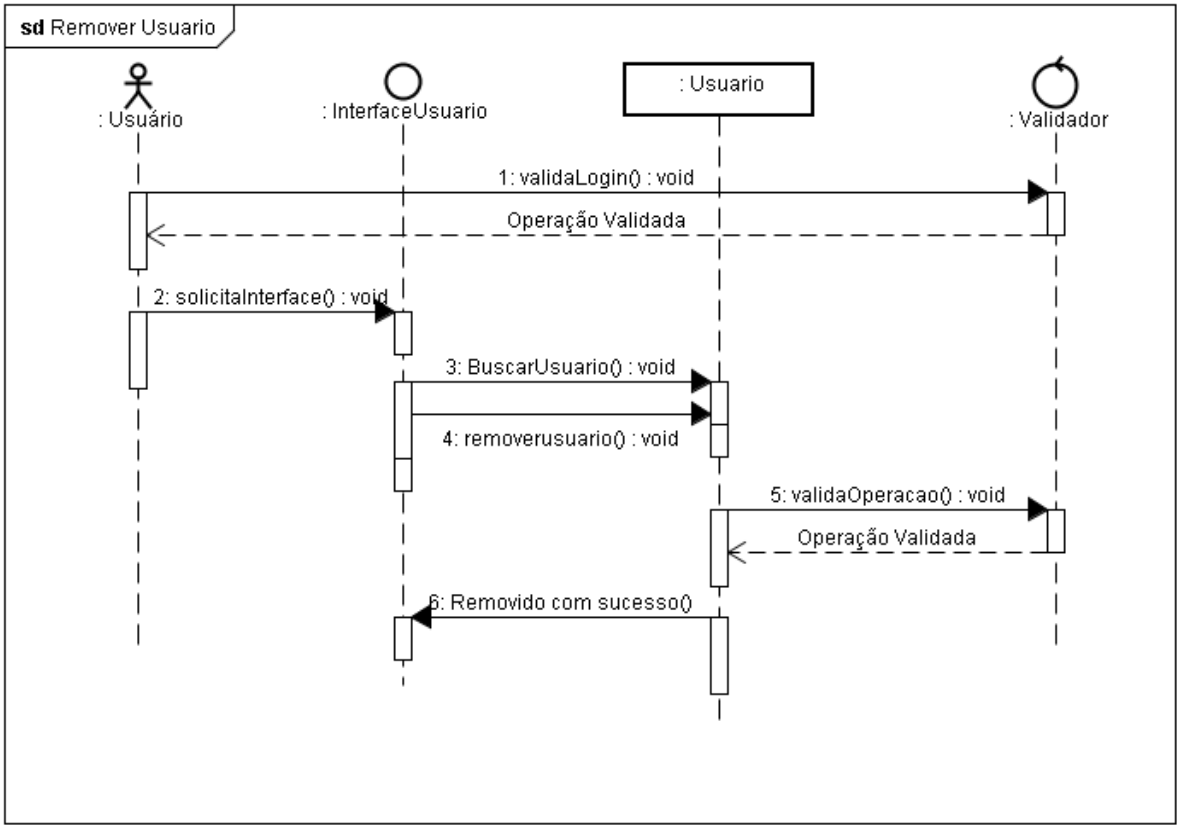

Figura 6. Diagrama de seqüência - Remover usuário Fluxo normal.

2.3.2. Permissão

O sistema de manipulação de permissões contará com as funções básicas de um cadastro, no nosso caso, o único que poderá inserir alterar e remover possíveis permissões do sistema será o administrador.
Segue figura 7 ilustrando o cadastro de permissão no sistema.

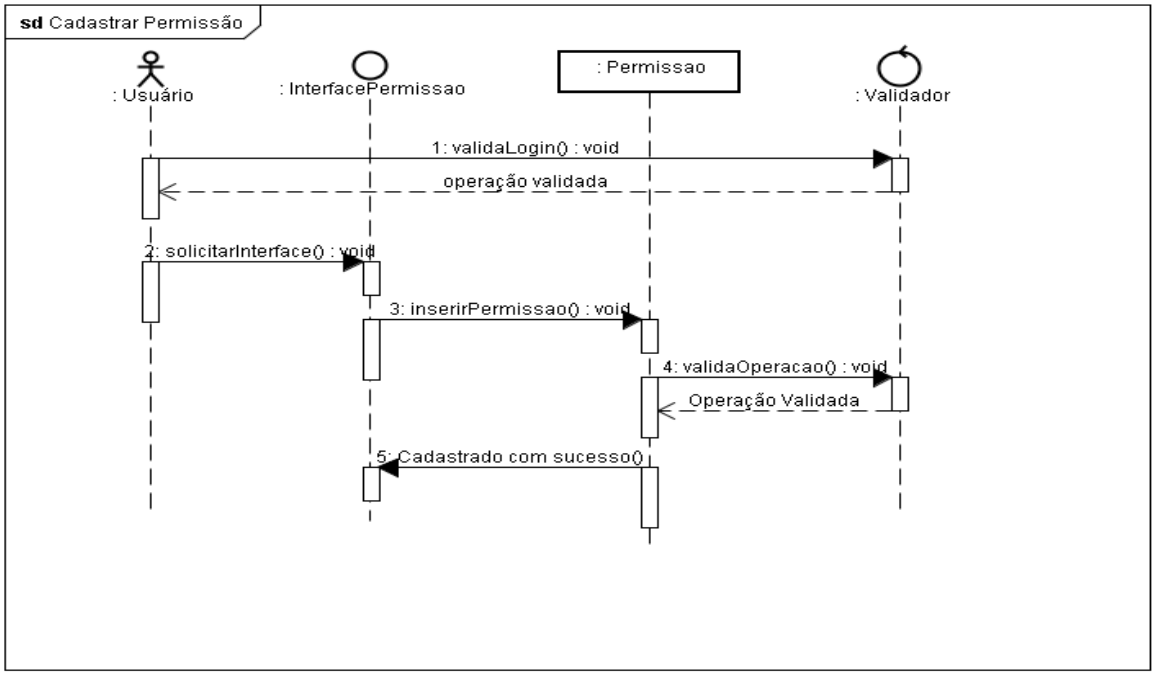

Figura 7. Diagrama de seqüência - Inserir Permissão Fluxo normal.

Segue figura 8 ilustrando a alteração de permissão no sistema.

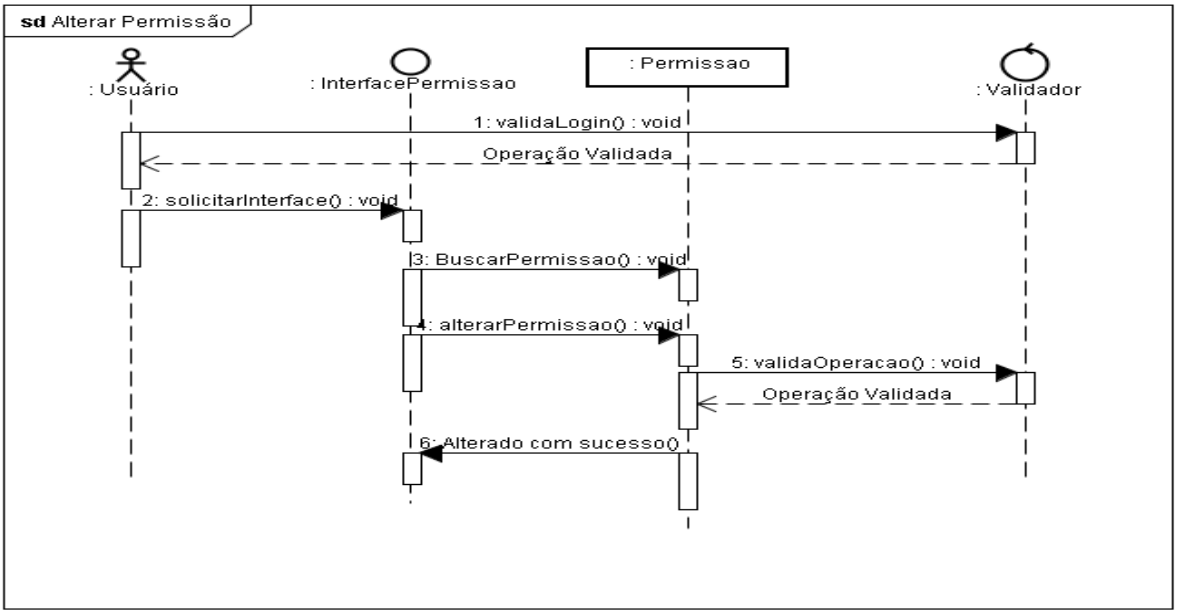

Figura 8. Diagrama de seqüência - Alterar Permissão Fluxo normal.

Segue figura 9 ilustrando a remoção de permissão no sistema.

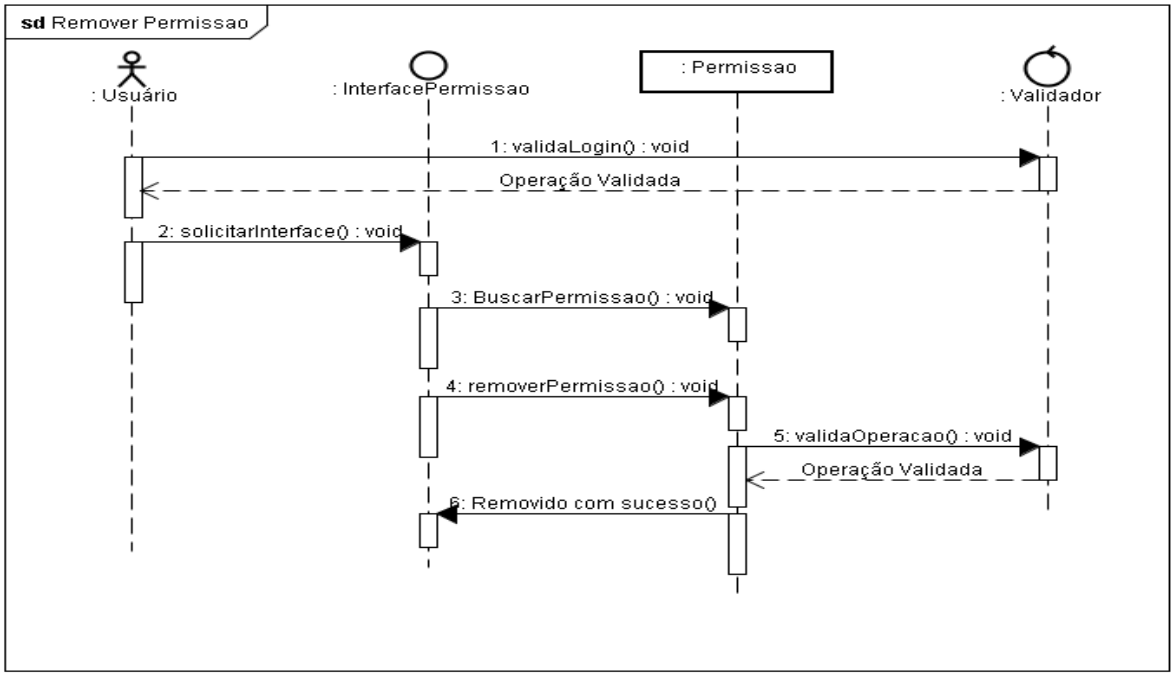

Figura 9. Diagrama de seqüência – Remover Permissão Fluxo normal.

2.3.3. Linguagem

O sistema de manipulação de Linguagem contará com as funções básicas de um cadastro, no nosso caso, podemos inserir, alterar e remover possíveis linguagens do sistema, de acordo com a permissão do usuário.

Segue figura 10 ilustrando o cadastro de Linguagem no sistema.

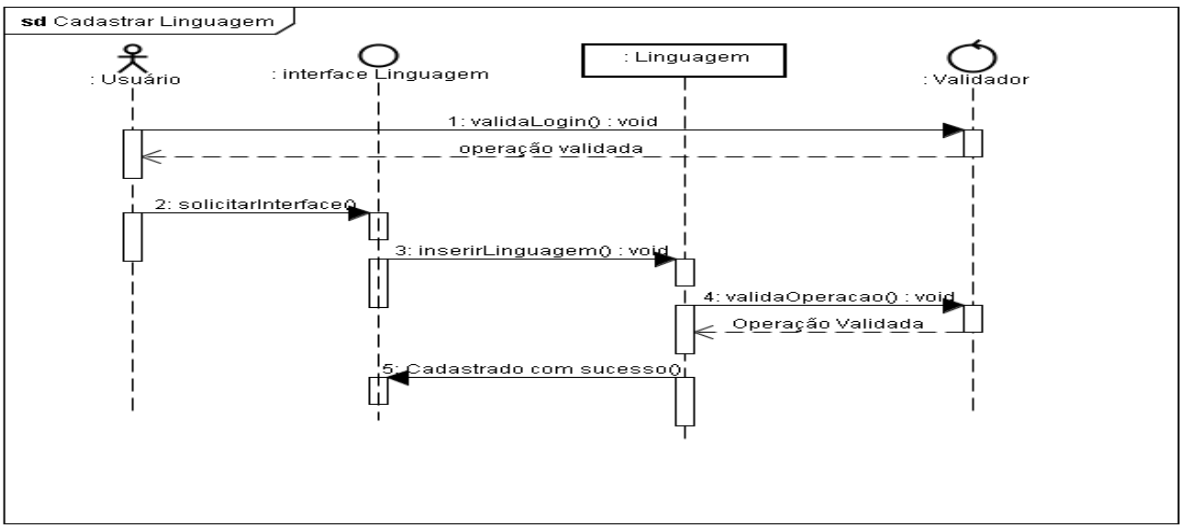

Figura 10. Diagrama de seqüência – Inserir Linguagem Fluxo normal.

Segue figura 11 ilustrando a alteração de Linguagem no sistema.

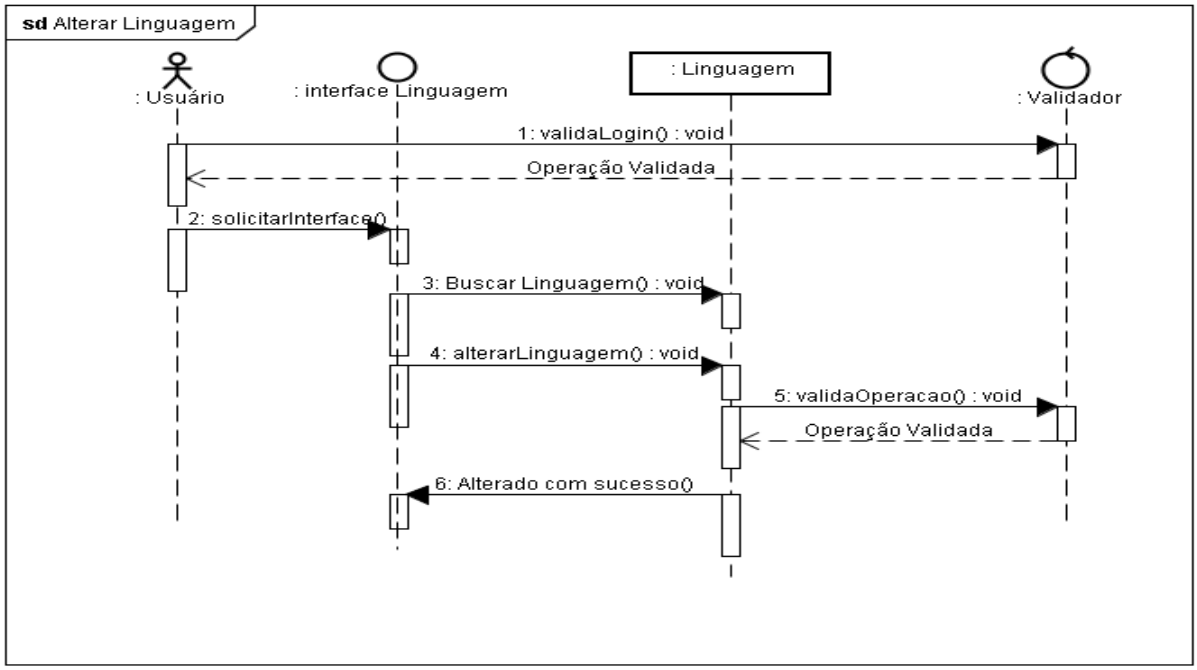

Figura 11. Diagrama de seqüência – Alterar Linguagem Fluxo normal.

Segue figura 12 ilustrando a Remoção de Linguagem no sistema.

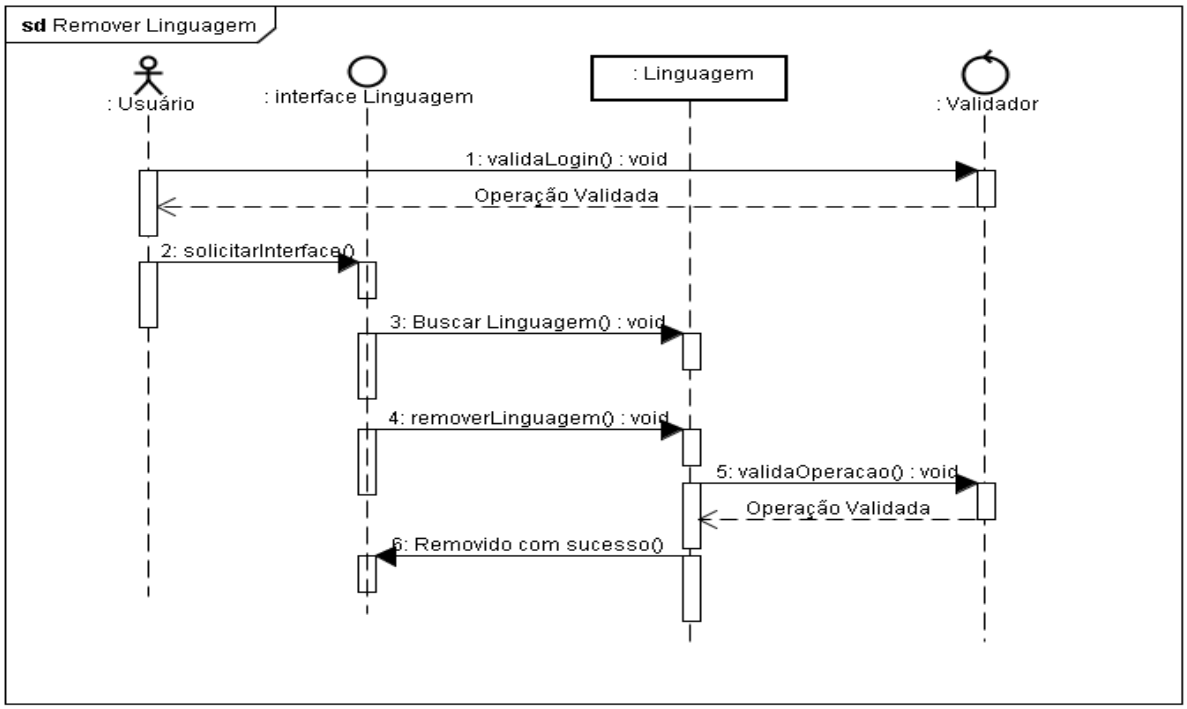

Figura 12. Diagrama de seqüência – Remover Linguagem Fluxo normal.

2.3.4. Projeto

O sistema de manipulação de Projeto contará com as funções básicas de um cadastro, no nosso caso, podemos inserir, alterar e remover possíveis projetos do sistema, de acordo com a permissão do usuário.

Segue figura 13 ilustrando o cadastro de Projeto no sistema.

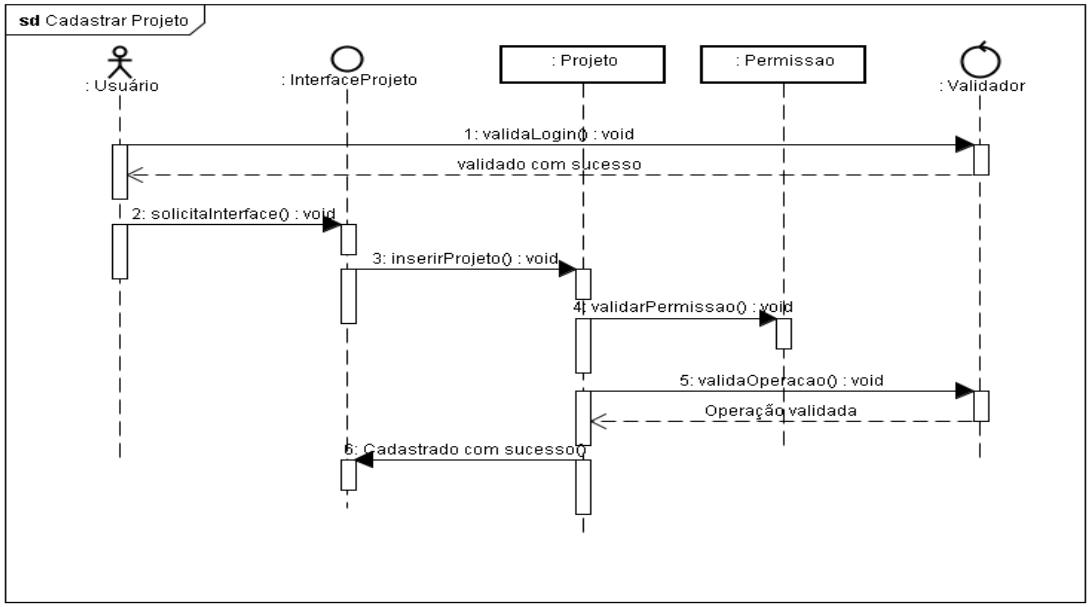

Figura 13. Diagrama de seqüência - inserir Projeto Fluxo normal

Segue figura 14 ilustrando a alteração de Projeto no sistema.

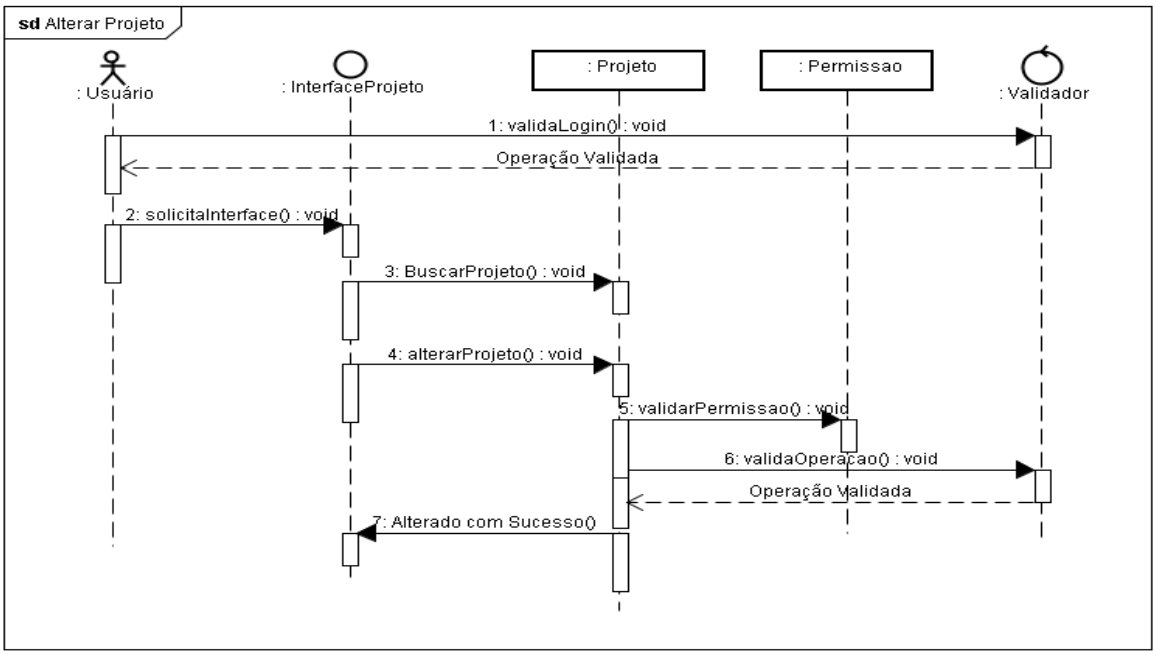

Figura 14. Diagrama de seqüência – Alterar Projeto Fluxo normal.

Segue figura 15 ilustrando a remoção de Projeto no sistema.

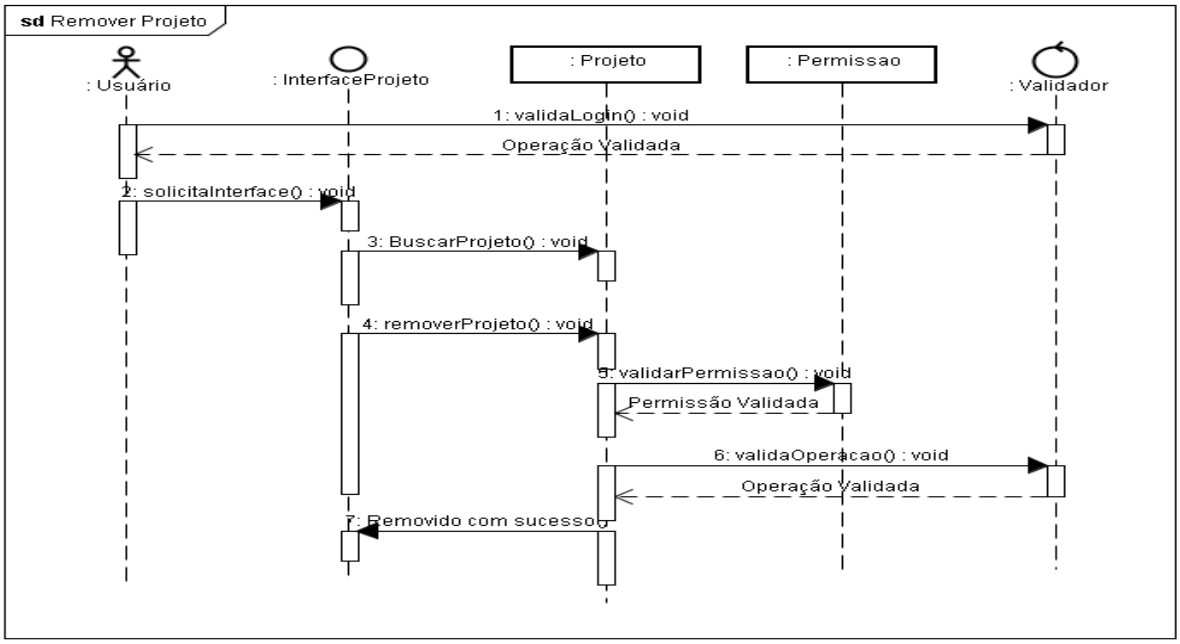

Figura 15. Diagrama de seqüência – Remover Projeto Fluxo normal.

2.3.5. Tipo de Documento

O sistema de manipulação de Tipo de Documento contará com as funções básicas de um cadastro, no nosso caso, podemos inserir, alterar e remover possíveis tipo de documentos do sistema, de acordo com a permissão do usuário.

Segue figura 16 ilustrando o cadastro de Tipo Documento no sistema.

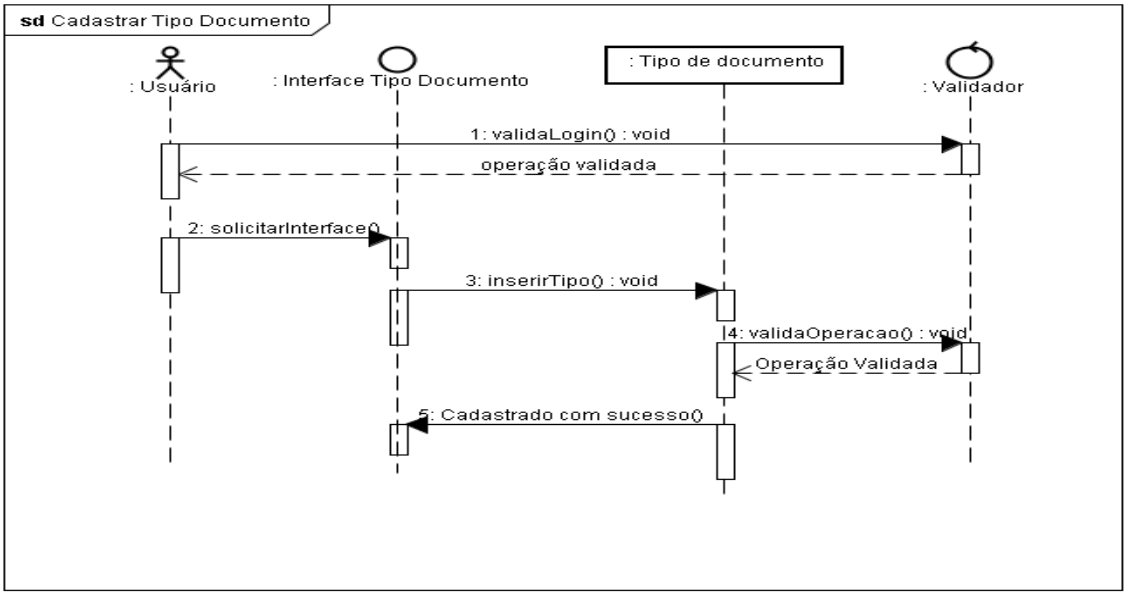

Figura 16. Diagrama de seqüência – Inserir Tipo Documento Fluxo normal.

Segue figura 17 ilustrando a alteração de Tipo Documento no sistema.

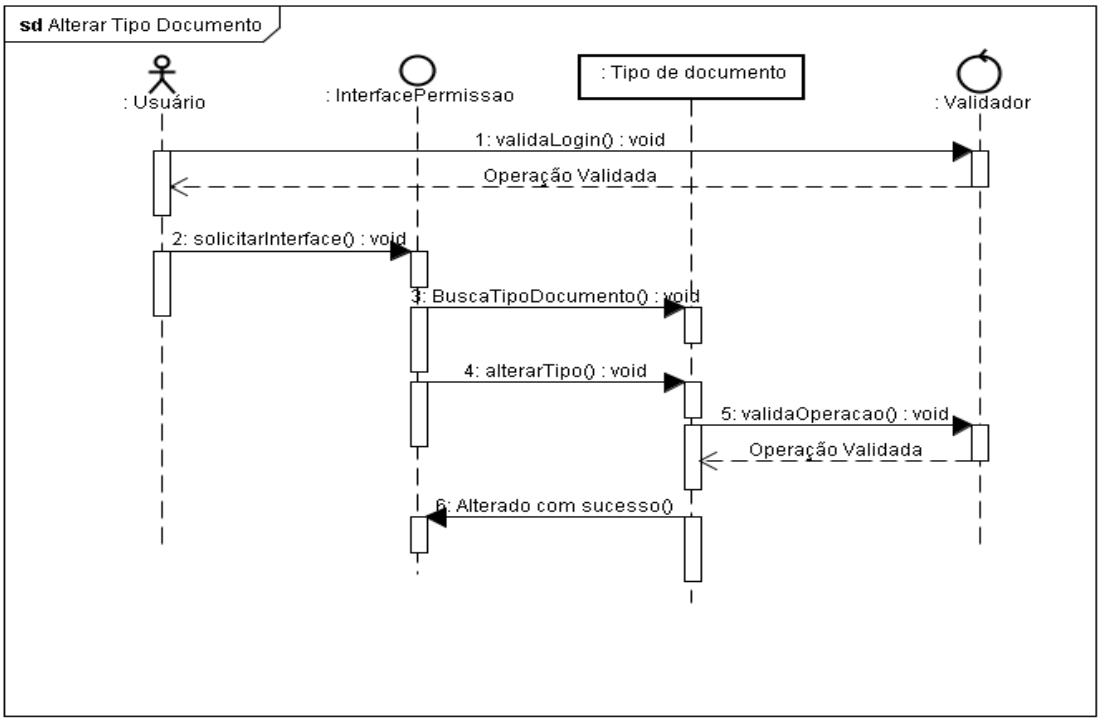

Figura 17. Diagrama de seqüência – Alterar Tipo Documento Fluxo normal.

Segue figura 18 ilustrando a remoção de Tipo Documento no sistema.

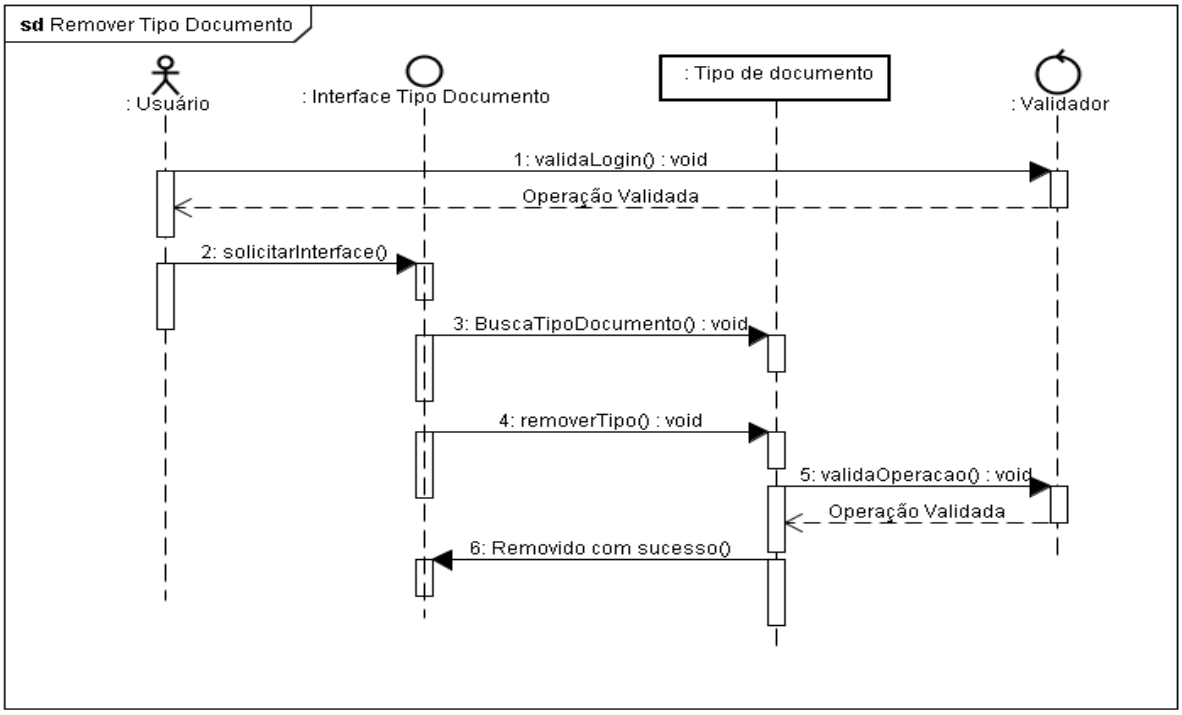

Figura 18. Diagrama de seqüência – Remover Tipo Documento Fluxo normal.

2.3.6. Arquivo

O sistema de manipulação de Arquivo contará com as funções básicas de um cadastro, no nosso caso, podemos inserir, alterar e remover e também fazer download de possíveis arquivos do sistema, de acordo com a permissão do usuário.

Segue figura 19 ilustrando a inserção de Arquivo no sistema.

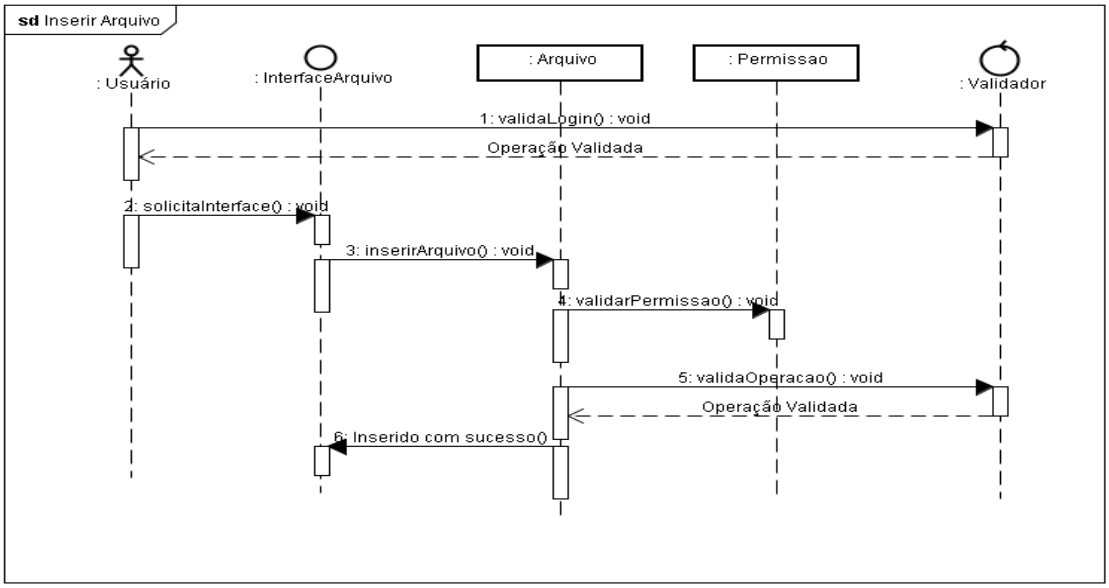

Figura 19. Diagrama de seqüência - inserir Arquivo Fluxo normal

Segue figura 20 ilustrando a alteração de Arquivo no sistema.

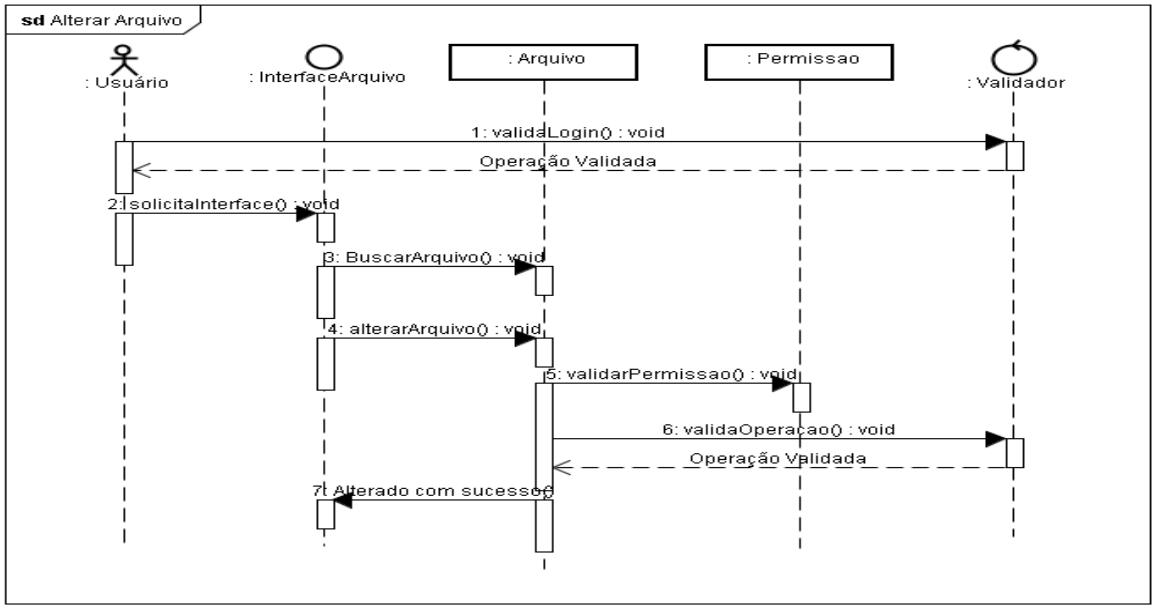

Figura 20. Diagrama de seqüência - Alterar Arquivo Fluxo normal

Segue figura 21 ilustrando a remoção de Arquivo no sistema.

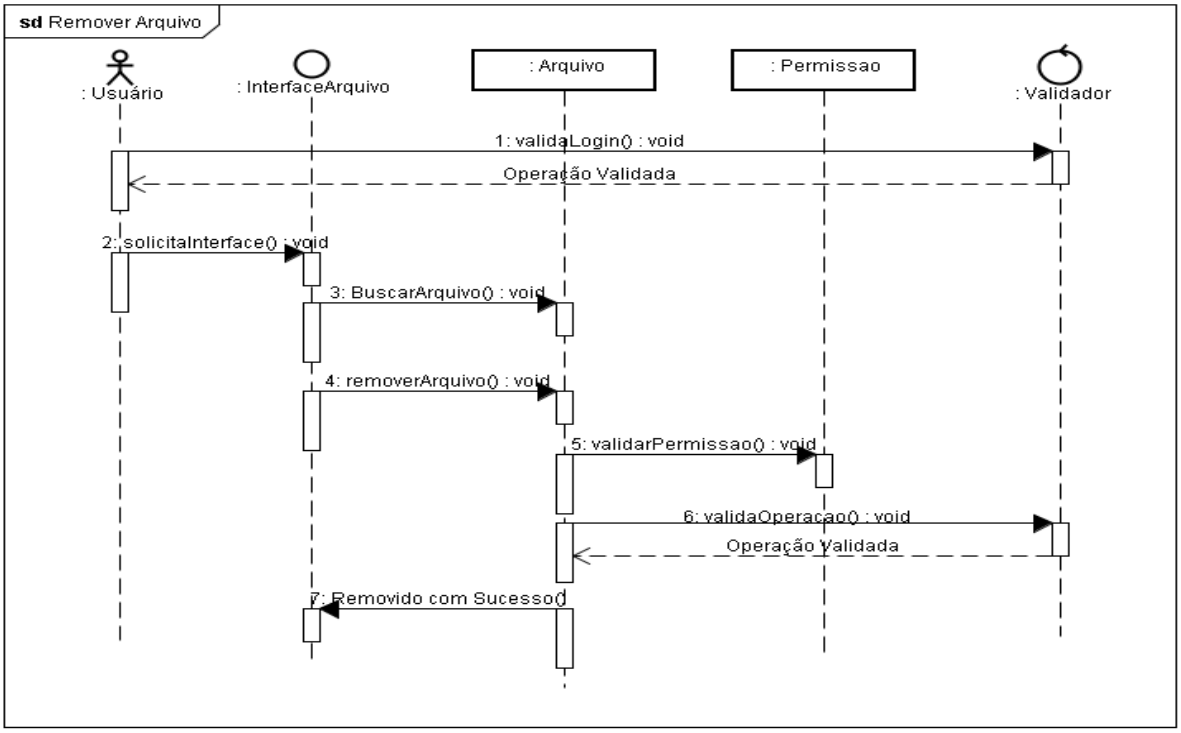

Figura 21. Diagrama de seqüência - Remoção Arquivo Fluxo normal

Segue figura 22 ilustrando o Download de Arquivo no sistema.

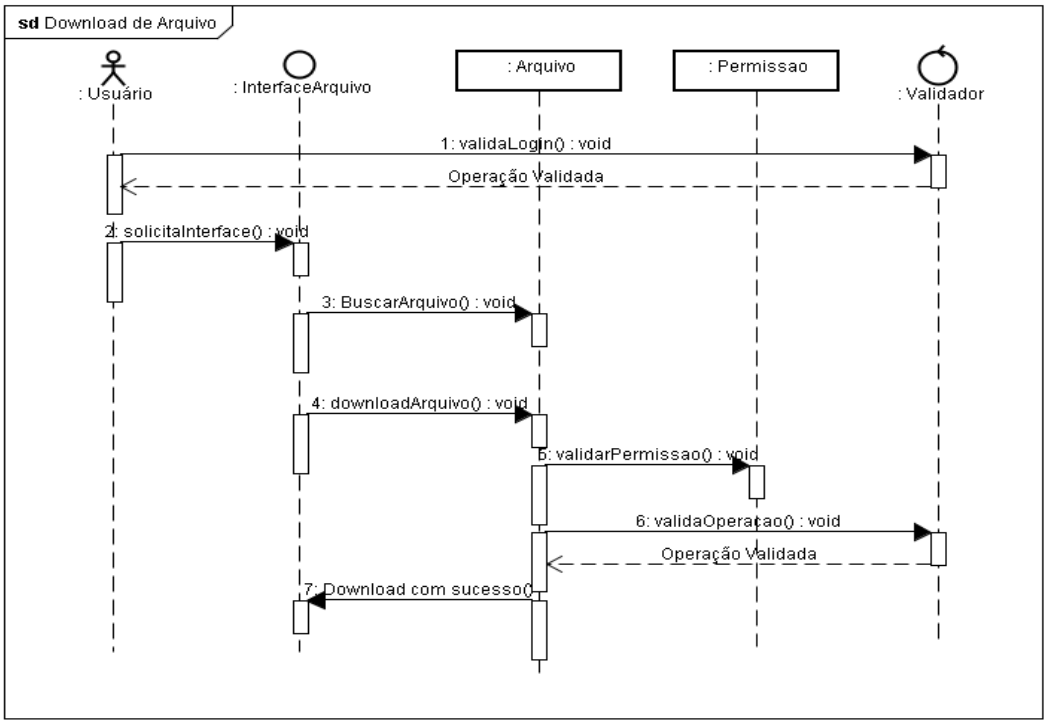

Figura 22. Diagrama de seqüência - Download Arquivo Fluxo normal

3. DESENVOLVIMENTO E IMPLEMENTAÇÃO

Com base na análise realizada viu-se a necessidade de uma aplicação com interface web para integrar todas as equipes envolvidas nos projetos de uma determinada fabrica de software, onde cada qual possa acessar de acordo com sua funcionalidade.

3.1. ARMAZENAMENTO E ORGANIZAÇÃO

Sendo o processo de análise um dos mais importantes na produção de software, a aplicação vai possuir níveis de acesso para garantir a integridade da documentação. Para o armazenamento ou qualquer alteração em documentos já existentes na aplicação, o usuário necessita de permissão, ou seja, a aplicação é organizada por diferentes níveis de acesso, cada um de acordo com seu nível de funcionalidade no projeto. Os documentos serão armazenados em único banco de dados, onde serão classificados pelo seu projeto, tipo e linguagem.

A aplicação irá possuir uma rastreabilidade entre os documentos, para quando houver a alteração do mesmo este fique marcado como suspeito, para que todos os outros documentos relacionados tenham uma noção do impacto de determinada modificação.

3.2. OBJETIVO

A aplicação tem a perspectiva de otimizar o tempo no acesso e organização do material de análise contribuindo para aumento da produtividade da fábrica de software. Na forma de armazenar toda a documentação de uma determinada fábrica de software em uma única base que será acessada com interface web, facilitando o acesso à documentação de análise de todos os projetos da fábrica de acordo com a permissão do usuário.

O acesso por interface web pode facilitar o serviço do usuário que faz a documentação de análise junto ao cliente, podendo acessar a documentação junto ao cliente e podendo fazer modificações em tempo real, facilitando o serviço de usuários que aguardam a documentação.

Assegurar que todos da equipe permaneçam informados sobre as informações de requisitos mais atuais para garantir consistência. Fornece as equipes à capacidade de entender o impacto das alterações.
Cientes que a aplicação não gera a documentação de análise, ela apenas arquiva os diagramas UML gerados por determinada ferramenta e os guarda de forma organizada. Para entender melhor a aplicação vamos analisar um pouco de sua estrutura:

3.3. ORGANIZAÇÃO

A aplicação será organizada no padrão Model View Controller ou simplesmente MVC é um conceito de desenvolvimento de aplicação, que separa a aplicação em três partes distintas: model, view e control.
Model (Modelo): é a camada que representa os dados do programa, ou seja, o modelo são classes que trabalham no armazenamento e busca de dados. Por exemplo, a modelagem de usuário na aplicação.
View (Apresentação): é a apresentação visual dos dados representados no Model, para o usuário. Por exemplo, a pagina de login onde os usuários logam na aplicação.
Controller (Controlador): é a camada que responde pelos dados executados pelo usuário, atuando nos dados apresentados no Model Por exemplo, o Controller recebe um pedido para exibir uma lista de documentos interagindo com o Modelo e entrega uma Apresentação onde esta lista poderá ser exibida.
O Modelo MVC é uma forma de desenvolvimento que auxilia na manutenção do sistema, um padrão muito utilizado no desenvolvimento de aplicações Java, principalmente Web.

3.3.1. Organização dos Pacotes

Seguindo o padrão MVC as classes do projeto foram estruturadas nos seguintes pacotes (figura 23).

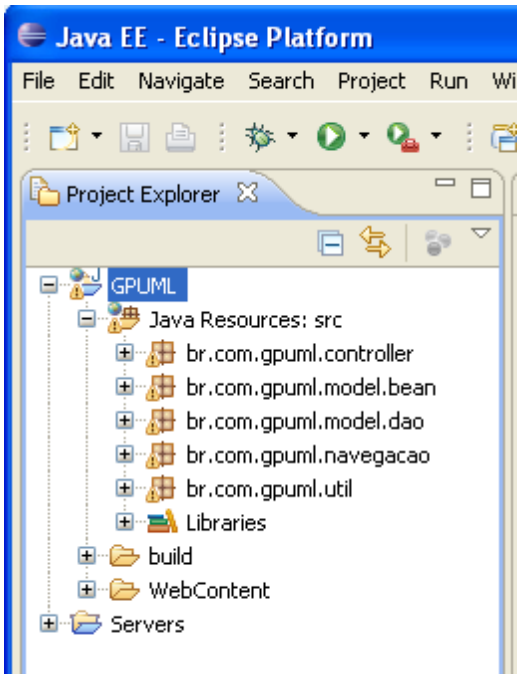

Figura 23. Organização dos Pacotes.

Ressaltando que o projeto esta em desenvolvimento e o conteúdo dos pacotes podem ser alterados, segue a descrição dos mesmos:

O pacote br.com.gpuml.bean: pacote que contém as classes do projeto essas que modela toda a regra de negócio do projeto, além das classes de mapeamento do hibernate.

O pacote br.com.gpuml.util: pacote que contém as classes de manipulação do banco de dados, sendo essas responsáveis pela criação e conexão com o banco, portanto sempre que a aplicação necessitar de conexão com o banco de dados vai ser necessário instanciar um objeto da classe conexão.

O pacote br.com.gpuml.dao: nesse pacote ficam as classes que realizam a persistência no banco de dados contendo classes genéricas que recebem como parâmetro o bean que será persistido e os beans que necessitarem de métodos

além dos da classe genérica possuem seu próprio dao com apenas os métodos que não possui na classe genérica.

O pacote br.com.gpuml.navegacao: pacote onde se encontra as classes que fazem a integração do modelo ou seja dos beans da aplicação e as views que são as páginas da aplicação.

O pacote br.com.gpuml.controller: pacote onde se encontra as classes de manipulação de dados, essas responsáveis por converter e validar dados mas a principal classe deste pacote é a classe ListenerFasesJSF que é responsável por não permitir a entrada de usuários que não estão cadastrados na base de dados da aplicação além de redirecionar quando a sessão é expirada.

3.3.2. Organização das Páginas

As páginas estão distribuídas no diretório WebContent do Eclipse como mostra a figura 24. Ressaltando que o projeto está em desenvolvimento e essas podem sofrer modificações.

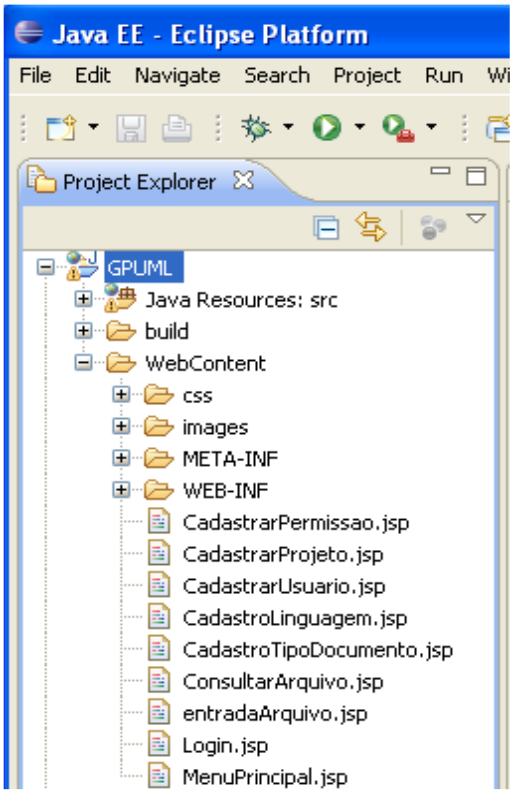

Figura 24. Organização das Páginas.

3.4. FUNCIONAMENTO DA APLICAÇÃO

3.4.1. Login

O usuário que desejar acessar alguma entidade do sistema primeiramente terá de acessar a aplicação por um navegador de internet qualquer e em seguida vai ter de efetuar o login da aplicação que vai lhe permitir o acesso na aplicação de acordo com a sua permissão. Agora se o usuário ainda não possuir acesso vai ter que solicitar um login ao usuário que tenha permissão igual a administrador que é o único que pode criar um novo usuário na aplicação.

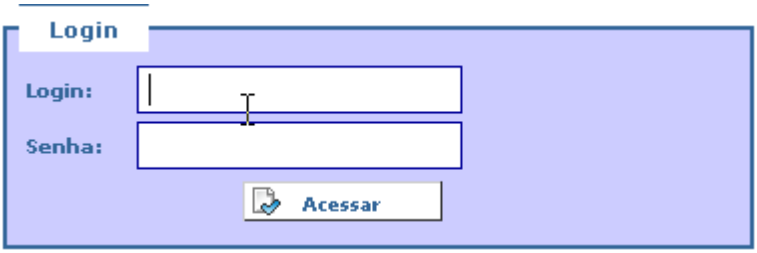

Figura 25. Tela de Login

Para verificar e autenticar os usuários teremos uma autenticação dos usuários através das PhaseListener do JSF; por que isso garante que o usuário não acesse a aplicação sem passar pela página de login. Então antes da fase beforephase, ou seja, renderização da página colocamos um validador de login para testar se o usuário esta logado, então para toda requisição da nossa aplicação será validado o login do usuário não sendo possível acessar nenhuma das páginas da aplicação sem passar pela página de login.

3.4.2. Movimentações

O usuário que deseja cadastrar, atualizar, excluir qualquer que seja a movimentação em alguma entidade do sistema, primeiramente acessa a aplicação, em seguida efetua o ‘login’ no sistema. Depois do ‘login’ tem acesso ao módulo administrativo, em seguida esse usuário tem a opção de acessar qualquer entidade do sistema conforme sua permissão.

3.4.3. Rastreabilidade

A rastrebilidade do sistema será uma maneira de informar os usuários da aplicação de atualizações na documentação dos projetos que venha a interferir em usos futuros que se não informados possam causar danos, a aplicação vem informar aos usuários sobre as mudanças de forma a marcar o projeto como suspeito e informar as mudanças ocorridas no seu conteúdo, para que quem o acesse fique informado das devidas alterações e possa se previr dos possíveis danos.

4. COMPONENTES E TECNOLOGIAS

Este capítulo descreve as tecnologias utilizadas no desenvolvimento da aplicação GPUML e também tecnologias utilizadas no Módulo de Configuração e análise.

4.1. ESTRUTURA TECNOLÓGICA

A aplicação foi desenvolvida utilizando-se das seguintes tecnologias:

- Linguagem JAVA;
- Eclipse;
- Java Server Pages;
- Java Server Faces;
- CSS;
- Banco de dados HSQLDB;
- Hibernate;
- Servidor de aplicação Tomcat;
- Apache MyFaces;
- Padrão de arquitetura de software MVC (Model View Control);
- UML.

4.1.1. Linguagem Java

Criada pela Sun Microsystems foi inicialmente destinada a pequenos dispositivos eletrônicos e quase fracassou. Comercialmente só se tornou um bom negócio quando se voltou para a internet. Isto porque, graças a sua característica multiplataforma, possibilitou a inclusão de pequenas aplicações diretamente nas páginas web, independente do sistema usado no computador do cliente. Java é totalmente orientada a objetos e sua parenta mais próxima é a linguagem C, C++. Java é uma linguagem de programação totalmente orientada a objetos. Seu funcionamento, no que diz respeito ao desenvolvimento, é semelhante a outras linguagens: DIGITAR, COMPILAR, EXECUTAR, DEPURAR. Mas como já vimos, é uma das poucas, senão a única linguagem de programação que permite ao programador criar um único código que roda sem alteração em qualquer computador que possua uma Java Virtual Machine instalada. É justamente a JVM ou Máquina

Virtual Java o segredo por trás da característica de multiplataforma, apresentada pelo ambiente de desenvolvimento Java. Um programa fonte escrito em linguagem Java é traduzido pelo compilador para os *bytecodes*, isto é, o código de máquina de um processador virtual, chamado *Java Virtual Machine (JVM).* A JVM é um programa capaz de interpretar os bytecodes produzidos pelo compilador, executando o programa cerca de 20 vezes mais lento do que C. Pode parecer ruim, mas é perfeitamente adequado para a maioria das aplicações. Com isto, um programa Java pode ser executado em qualquer plataforma, desde que esteja dotada de uma JVM. É o caso dos programas navegadores mais populares, como o Netscape Navigator e o Internet Explorer, que já vêm com uma JVM. A vantagem desta técnica é evidente: garantir uma maior portabilidade para os programas Java em código-fonte e compilados. Porém, as JVM tendem a ser programas extensos que consomem muitos recursos, restringindo assim o tamanho das aplicações escritas em Java. (SANTOS, 2004).

4.1.2. Eclipse

É a *Integrated Development Environmente (IDE)* Java mais utilizada no mundo, focada no desenvolvimento em aplicações de *software* e ferramentas. Possui como principais características a forte orientação ao desenvolvimento baseado em *plug-ins* e o amplo suporte ao desenvolvedor que garantem atender todas as necessidades dos programadores. Consiste em um software que contém um conjunto de funcionalidades embutidas, cuja finalidade é prover um modo mais fácil e interativo de construir e manipular seus programas. Entre estas ferramentas geralmente figuram:

- Um editor de texto com facilidades especialmente desenhadas para a linguagem;
- Um compilador (e um interpretador, no caso de Java e outras linguagens interpretadas) ;
- Um editor gráfico, com facilidades para criação e edição da interface gráfica do programa a ser desenvolvido;
- Um *debugger*, uma ferramenta especialmente feita para se tirar os *bug*s do código. Ela possibilita um monitoramento mais elegante do funcionamento do seu programa, facilitando a detecção e remoção dos erros. Perceba que

> não estamos falando em erros de sintaxe, mas erros na própria lógica do programa, que fazem seu programa gerar resultados indesejados ou travar (apesar de ele compilar), e que geralmente são difíceis de encontrar simplesmente analisando o código. (JAVA FREE, 2009)

4.1.3. Java Server Pages (JSP)

O JSP é uma tecnologia utilizada para desenvolvimento de aplicações para web, o diferencial do JSP é a capacidade de gerar páginas dinâmicas através de uma sintática familiar o HTML que é utilizado para desenvolvimento de conteúdo estático, o JSP mescla suas “tags” com as “tags” do HTML inserindo o conteúdo dinâmico em meio ao conteúdo estático. O JSP nada mais é que um documento de texto contendo HTML e Java. Por ser baseado em linguagem de programação JAVA, tem a vantagem da portabilidade de plataforma que permite sua execução em diversos sistemas operacionais. Esta tecnologia permite ao desenvolvedor produzir aplicações que acessem o banco de dados, manipulem arquivos no formato de texto dentre outros recursos. O JSP nada mais é que um documento de texto contendo HTML e Java (TODD, 2003)

4.1.4. Java Server Faces (JSF)

Java Server Faces faz com que o desenvolvimento de aplicações web se torne fácil, trazendo suporte para ricos e poderosos componentes de interface, como caixa de texto, lista, painel de abas e rede de dados, para o mundo do desenvolvimento web.
Facilidade de uso é a principal característica da arquitetura JSF, tornando simples a ligação entre a camada de apresentação e o código aplicado. Com essa tecnologia é possível que cada desenvolvedor web foque sua parte do processo de desenvolvimento, pois a JSF nos fornece um modo simples para podermos conectar todas as partes do projeto.
Desenvolvida para ser flexível, a JSF estabelece a ligação entre o desenvolvimento do código e a interação do usuário por meio de aplicações web.
JSF é uma tecnologia que incorpora características de um framework MVC (Model-View-Controller) para WEB e de um modelo de interfaces gráficas baseado em

eventos. Por basear-se no padrão de projeto MVC, uma de suas melhores vantagens é a clara separação entre a visualização e regras de negócio (modelo).
Java Server Faces oferece ganhos no desenvolvimento de aplicações WEB por diversos motivos:

- Permite que o desenvolvedor crie Uis através de um conjunto de componentes predefinidos;
- Fornece um conjunto de tags JSP para acessar os componentes;
- Reusa componentes da página;
- Associa os eventos do lado cliente com os manipuladores dos eventos do lado (os componentes de entrada possuem um valor local representando o estado servidor);
- Fornece separação de funções que envolvem a construção de aplicações WEB.

Embora Java Server Faces forneça tags JSP para representar os componentes em uma página, ele foi projetado para ser flexível, sem limitar-se a nenhuma linguagem markup em particular, nem a protocolos ou tipo de clientes. Ele também permite a criação de componentes próprios a partir de classes de componentes, conforme mencionado anteriormente. (MANN, 2005).

4.1.5. CSS

Folhas de estilos em cascata — Cascading Style Sheets (CSS) — é uma ferramenta fantástica para construção do layout dos seus websites. Permite que você projete websites com uma técnica completamente diferente da convencional e possibilita uma considerável redução de tempo de trabalho. Conhecer CSS é uma necessidade para qualquer um envolvido com o projeto web.
Uma folha de estilos é um conjunto de regras que informa a um programa, responsável pela formatação de um documento, como organizar a página, como posicionar e expor o texto e, dependendo de onde é aplicada, como organizar uma coleção de documentos. (CSS, 2009)

4.1.6. HSQLDB

O *Hyperthreaded Structured Query Language Database* (HSQLDB) é um servidor de banco de dados, de código aberto e escrito em Java que permite a manipulação de banco de dados em uma arquitetura cliente-servidor, ou standalone. Uma grande vantagem de utilização do HSQLDB é a possibilidade de agregarmos o banco de dados ao pacote de nossas aplicações. Outra característica do banco é a possibilidade de manipularmos bancos de dados em disco, memória ou em formato texto. Trata-se de uma tecnologia flexível e muito útil na construção de aplicações que manipulem banco de dados.

Geralmente usado para aplicações desktop que necessitam de persistência através de linguagem *SQL* em suas aplicações. A portabilidade, a simplicidade e o bom desempenho dispondo da necessidade de poucos recursos garantem sua utilização.

4.1.7. Hibernate

Hibernate é um projeto ambicioso que visa ser a solução para os problemas de persistência em Java. Ele age mediando à interação da aplicação com o banco de dados relacional, deixando o desenvolvedor livre para se concentrar no problema em si. *Hibernate* é uma solução não-intrusiva, ou seja, não é necessário que você siga todas as regras específicas e padrões de projeto quando escreve sua lógica empresarial e classes de persistência, deste modo o *Hibernate* se integra suavemente com a maioria das novas e já existentes aplicações e não requerem drásticas modificações nas mesmas O Hibernate é uma ferramenta de mapeamento objeto/relacional para Java. Ela transforma os dados tabulares de um banco de dados em um grafo de objetos definido pelo desenvolvedor. Usando o Hibernate, o desenvolvedor se livra de escrever muito do código de acesso a banco de dados e de SQL que ele escreveria não usando a ferramenta, acelerando a velocidade do seu desenvolvimento de uma forma fantástica. Mas o framework não é uma boa opção para todos os tipos de aplicação. Sistemas que fazem uso extensivo de stored procedures, triggers ou que implementam a maior parte da lógica da aplicação no banco de dados, contando com um modelo de objetos “pobre” não vai se beneficiar com o uso do Hibernate. Ele é mais indicado para sistemas que contam com um modelo rico, onde a maior parte da lógica de negócios fica na

própria aplicação Java, dependendo pouco de funções específicas do banco de dados.

Framework para fazer mapeamentos entre objeto e a parte relacional da aplicação mantendo a independência entre os bancos de dados. Possui sua própria linguagem de *Structure Query Language (SQL)* chamada *Hibernate Query Language (HQL)*, que é convertida para a *SQL* específica de cada banco de dados. (BAUER; KING, 2005).

4.1.8. Tomcat

Servidor Web Java, mais especificamente um *container* de *servlets*. Implementação das tecnologias de *Java Servlet* e *JavaServer Pages.* Desenvolvido em ambiente aberto e participativo, tem a colaboração de todos os desenvolvedores do mundo, mas não pode ser considerado um servidor de aplicação completo, por não preencher todos os requisitos necessários.

O Tomcat é robusto e eficiente o suficiente para ser utilizado mesmo num ambiente de produção. Ele tem a capacidade de atuar também como servidor web, ou pode funcionar integrado a um servidor web dedicado como o Apache ou o IIS. Como servidor web, ele provê um servidor web HTTP puramente em Java. O servidor inclui ferramentas para configuração e gerenciamento, o que também pode ser feito editando-se manualmente arquivos de configuração formatados em XML.

A disponibilização de aplicações nesse servidor deve seguir um padrão rígido e bem estabelecido, garantido que uma mesma aplicação possa ser também montada em outro servidor de aplicação de forma transparente. (TOMCAT, 2009)

4.1.9. Apache Myfaces

O projeto *Apache MyFaces* ou simplesmente *MyFaces,* foi criado para dar sustentação ao padrão JSF e é um *framework* para o desenvolvimento de aplicações web.

Uma das implementações do JavaServer Faces mais utilizada atualmente é o MyFa ces, um projeto da Apache Software Foundation que vem crescendo rapidamente e hoje vai muito além da especificação JSF. Além do MyFaces Core, que é a impleme ntação do JSF em si, há vários subprojetos, que fornecem componentes adicionais, t

razendo suporte a Ajax, vinculação com dados, entre outras funcionalidades importa ntes no desenvolvimento web.

5. CONSIDERAÇÕES FINAIS

Os conceitos básicos para o desenvolvimento deste trabalho foram vistos, tais como as visões e diagramas da uml, fábrica de software, gerenciamento de requisitos e rastreabilidade. Todo esse conhecimento descrito, aliado a conhecimento técnico adquirido ao longo do curso e em atividades extracurriculares, tornaram possível a implementação de uma nova ferramenta chamada Gerenciamento de Projetos uml para Fábrica de Software. Com ela torna-se possível manipular informações contidas na documentação de análise de uma fábrica de software que antes não tinham uma maneira organizada de serem armazenadas e consultadas. Além de dar toda uma segurança que todos os usuários que acessam essa documentação ficarem informados de atualizações e novos documentos sem que haja ambigüidades ou diferenças de interpretação.

Os resultados encontrados permitem a continuação deste trabalho, seja por melhoramentos ou pela extensão do escopo do projeto. Assim algumas propostas para trabalhos futuros devem ser citados:

- Quanto à construção dos diagramas de análise criar uma ferramenta para criação desta documentação na própria aplicação de forma a integrar todo o escopo da fase de análise em uma única aplicação;
- Quanto à rastrebilidade de alterações na documentação de análise dos projetos, criar um mecanismo de informação via email aos envolvidos no projeto.

Cabe aqui ressaltar que este trabalho poderia ser expandido, englobando as etapas de um projeto de implantação de um sistema de GPUML em uma fábrica de software para possíveis testes, tais tópicos não foram abordados devido ao tempo disponível para a realização e à falta de uma fábrica de software disponível para a implantação.

De qualquer forma, este trabalho teve seu mérito no que diz respeito à aquisição de conhecimentos com relação ao Gerenciamento de requisitos e o aprendizado nas

tecnologias utilizadas para o desenvolvimento da aplicação e à possível disponibilização de tais conhecimentos a outrem.

REFERÊNCIAS BIBLIOGRÁFICAS

SANTOS, Rui Rossi dos. **Programando em Java 2 - Teoria e Aplicação.** 1. ed. Rio de Janeiro: Axcel Books do Brasil Editora, 2004.

FREEMAN, Eric; FREEMAN, Elisabeth. **Use a Cabeça! Padrões de Projetos.** 1. ed. Tradução de Andreza Gonçalves, Marcelo Soares e Pedro César de Conti. Castelo Rio de Janeiro: Editora Alta Books, 2005.

BAUER, Christian; KING Gravin. **Hibernate in Action.** Grennwich: Manning Publication Co., 2005.

MANN, Kito D. **JavaServer Faces in Action.** Grennwich: Manning Publication Co., 2005.

TODD, NICK; SZOLKOWSKI, MARK. JavaServer Pages: Guia do Desenvolvedor. **Campus, 2003.**

LEE, Richard C. TEPFENHART, William M. **UML e C++ - Guia Prático de Desenvolvimento Orientado Objeto.** 1. ed. São Paulo: Makron Books, 2001.

FERNANDES, Agnaldo A. TEIXEIRA Descartes de Souza. **Fabrica de Software-Implantação e Gestão de Operações.** 1. ed. Atlas, 2004.

Rational Software Corporation - Conceitos: Gerenciamento de Requisitos
Disponível em: http://www.wthreex.com/rup/process/workflow/requirem/co_rm.htm
Acessado em: 05/2009.

JAVA FREE - Eclipse
Disponível em http://javafree.uol.com.br/wiki/Eclipse
Acessado em: 05/2009.

Rational Software Corporation - Conceitos: Rastreabilidade
Disponível em: http://www.wthreex.com/rup/process/workflow/requirem/co_trace.htm
Acessado em: 05/2009.

TALITA, Pitanga. **GUJ - JavaServer Faces: A mais nova tecnologia Java para desenvolvimento WEB**
Disponível em: http://www.guj.com.br/JSF/tutotial/javaServerFaces.htm
Acessado em: 06/2009.

CSS - **Introdução a CSS**
Disponível em: **http://www.pt-br.html.net/tutorials/css/lesson3.asp**
Acessado em: 08/2009.

TOMCAT- Disponível em: http://tomcat.apache.org
Acessado em: 08/2009.