Universidade Federal de Santa Catarina
Departamento de Informática e Estatística
Curso de Sistemas de Informação

ALOHA
Um Ambiente Virtual de Aprendizado

ADRIANO MANOEL DEMETRIO
ARTHUR MARTINS PEREIRA

Florianópolis – SC
Ano 2007 / 2

ADRIANO MANOEL DEMETRIO
ARTHUR MARTINS PEREIRA

ALOHA
Um Ambiente Virtual de Aprendizado

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientadora: Profa. Maria Marta Leite

Banca Examinadora
Prof. Leandro José Komosinski, Dr
Prof. João Bosco da Mota Alves, Dr

Resumo

Em um mundo altamente competitivo, o aprendizado torna-se o grande diferencial. Ferramentas que tornam este processo mais ágil assumem um papel cada vez mais importante. Sendo assim, a utilização de Ambientes Virtuais de Aprendizado (AVA) torna-se uma grande alternativa.

Este trabalho apresenta o desenvolvimento do projeto Aloha, um Ambiente Virtual de Aprendizado desenvolvido com foco para a Universidade Federal de Santa Catarina e que suporta as interações existentes entre alunos e alunos, entre alunos e professores, assim como entre professores e professores. Estas interações se dão através de três ambientes: ambiente Equipe, ambiente Turma e ambiente GPDA. No desenvolvimento do sistema de informação foram utilizadas tecnologias *open-source* e a metodologia de desenvolvimento de software *EasyProcess*.

Palavras-chave: Ambiente Virtual de Aprendizado, *EasyProcess*, Aloha, *Open Source*, Desenvolvimento de Software.

Abstract

In a highly competitive world, the learning becomes the most differential. A tool that makes it more agile takes a role each time more important. So, the use of the Virtual Learning Environment (VLE) becomes a great alternative.

This study presents the development of the Aloha project, an Virtual Learning Environment developed with focus on the Federal University of Santa Catarina and that support the interation existing between student and student, between student and teacher, and between teacher and teacher. Therefore, the system place three interation environment, the Group environment, the Class environment and the GPDA environment. In the development of the project, it was used open source technologies and the EasyProcess software development methodology.

Keywords: Virtual Learning Environment., EasyProcess, Aloha, Open Source.

Sumário

Índice de Figuras

Índice de Gráficos e Tabelas

Lista de Abreviaturas e Siglas

VLE – *Virtual Learning Environment*CMS – *Content Management System*

MIS – *Management Information System*

LibMS – *Library Management System*

AVA – Ambiente Virtual de Aprendizado

YP - *EasyProcess*

UFCG - Universidade Federal de Campina Grande

XP - *Extreme Programming*

RUP - *Rational Unified Process*

TAA - Tabela de Alocação de Atividades

CVS - *Current Version System*

SVN - *Subversion*

MLE - *Managed Learning Environment*

UCISA - *Universities and Colleges Information System Association*

JISC - *Joint Information Systems Committee*

CAGR - Sistema Acadêmico da Graduação

JME - *Java Platform Micro Edition*

JSE - *Java Platform Standard Edition*

JEE - *Java Platform Enterprise Edition*

JCP - *Java Community Process*

HTML - *HyperText Markup Language*

CSS - *Cascading Style Sheets*

JSR - *Java Specification Request*

JSP - *JavaServer Page*

JSF - *JavaServer Faces*

AJAX - *Asynchronous Javascript And XML*

XHTML - *Extensible HyperText Markup Language*

DOM - *Document Object Model*

XML - *eXtensible Markup Language*

XSLT - *Extendible Stylesheet Language Transformations*

W3C - *World Wide Web Consortium*

1 INTRODUÇÃO

Em uma visão muito superficial de uma organização, percebem-se pessoas trabalhando tendo o objetivo de realizar alguma tarefa que lhe foi designada. Refinada essa visão, as pessoas são separadas em departamentos. Cada um define seus processos, criando tarefas que serão associadas ao perfil do departamento. No nível de organização, têm-se processos que envolvem vários departamentos. Mas como é conduzido o fluxo dos processos? Como uma tarefa é encerrada e a próxima é avisada para ser iniciada?

O conceito fundamental que explica a resposta para essas perguntas é a comunicação. Para que a comunicação seja eficaz é necessário o estudo dos processos e das tarefas, identificando os perfis de pessoas que serão responsáveis por cada tarefa deste fluxo. Através deste estudo define-se o relacionamento entre os perfis identificados na empresa. Este relacionamento indica quem avisa que existe uma nova tarefa, e quem deve ser avisado quando a tarefa é concluída. Mas de que maneira as pessoas devem comunicar-se? O gerente de um departamento deve se deslocar de um departamento até o outro para avisar que sua parte do processo foi finalizada?

A resposta para as perguntas acima é tecnologia. As organizações investem em softwares de gerenciamento de tarefas, nos quais os usuários informam todos os passos. Assim, quando a parte de um processo organizacional é finalizada em um departamento, é iniciado automaticamente o processo definido para o próximo departamento. Os softwares de gestão devem fornecer as informações de forma rápida e precisa. A entrada das informações também deve ser simples e segura, para evitar dados inconsistentes, o que pode levar a ocorrência de erros no processo.

Pode ocorrer que organizações utilizem ferramentas apenas de comunicação, e não de gestão de tarefas, como por exemplo, o e-mail. O e-mail permite apenas a comunicação e não a organização e garantia da continuidade do fluxo do processo. Um e-mail com uma informação relevante pode ser movido para

uma pasta e ser esquecido, ou ficar tão misturado entre os outros muitos e-mails e passar despercebido.

Os fluxos de processos de empresas, como os descritos acima, podem ser generalizados para um ambiente de ensino. Em uma universidade, por exemplo, têm-se os perfis de aluno, professor, coordenador de curso, entre outros. Da mesma forma, existe o relacionamento entre estes perfis para realizar os processos inerentes ao funcionamento do ambiente de ensino. Dessa forma, pode-se utilizar um software para a gestão desses relacionamentos.

1.1 APRESENTAÇÃO

A UFSC, até o momento, não possui uma ferramenta de gestão de relacionamentos no ambiente de ensino, envolvendo professores, alunos e turmas, que enfoque o conteúdo das disciplinas e outros aspectos que envolvem o cotidiano de ensino e não somente a parte administrativa, como o cadastro de aluno, pedido de matricula, entre outros.

Algumas iniciativas foram tomadas, como a utilização do software Moodle em algumas disciplinas do Departamento de Informática e Estatística. Percebeu-se que o mesmo apresentou aceitação perante os usuários e colaborou para melhorar a eficiência da comunicação entre alunos e professores.

Com o uso do Moodle foram detectadas a deficiência ou inexistência de algumas funcionalidades que poderiam se adequar melhor ao ambiente específico da UFSC.

1.2 FORMULAÇÃO DO PROBLEMA

No curso de Sistemas de Informação da UFSC os professores disponibilizam algumas informações em páginas pessoais estáticas. Informações importantes como tirar dúvidas, divulgação de notas, agendamento de provas e trabalhos, montagem de equipes, entre outras atividades, são realizadas através de e-mail ou por comunicação verbal em sala de aula. São poucos os professores que

formalizam estas informações nas páginas pessoais ou de disciplinas. Esta forma de comunicação é precária e muitas vezes ineficiente. As principais deficiências detectadas relativas a comunicação no curso de Sistema de Informação, através de simples observações, foram:

- Informações de datas de provas e trabalhos são disponibilizadas nos sites dos professores. Alguns professores não disponibilizam estas informações em site, sendo necessário envio de e-mail ou comunicação pessoal para obter a informação;
- Os grupos para realizar trabalhos em uma disciplina são montados anotando os nomes dos integrantes em papel e entregando ao professor, ou enviando um e-mail com os nomes do integrante ao professor;
- As formas de comunicação entre professor e aluno, ou aluno e aluno, são apenas pessoalmente ou via e-mail;
- O único acesso à quantidade de faltas registradas em uma disciplina é pessoalmente com o professor ou perguntado via e-mail;
- As notas e trabalhos já realizados são disponibilizados nos sites dos professores. Alguns professores não disponibilizam estas informações em site, sendo necessário envio de e-mail ou comunicação pessoal para obter a informação;

1.3 JUSTIFICATIVAS

Como justificativa para a elaboração deste tem-se:

- Maior disponibilidade e rapidez no acesso a informação, uma vez que é possível ter conhecimento do material didático e de outros recursos a qualquer momento do dia.
- Dinamismo no processo de aprendizagem, incentivando o aluno na busca do conhecimento, contando com a ajuda de avaliações *on-line*, disponibilidade de material didático e trocas de informações.

- Este projeto tende a tornar o processo de aprendizado mais flexível à distância, pois os alunos poderão acessar os materiais didáticos de onde estiverem e ainda poderão acordar questões relativas às suas equipes sem precisarem encontrar-se pessoalmente, bastando somente os mesmos se relacionarem utilizando as facilidades do sistema.
- Aumento do controle das atividades existentes em um ambiente de aprendizado por parte dos professores.
- As informações do desempenho dos alunos serão transparentes, portanto, será possível que alunos se auto-avaliem e que professores monitorem o desempenho das suas disciplinas.
- Integração das informações em um único repositório.
- O professor poderá dar suporte a diferentes alunos de diferentes disciplinas ao mesmo tempo.

1.4 OBJETIVOS

1.4.1 OBJETIVO GERAL

Criar um ambiente de aprendizado virtual (AVA) visando o relacionamento de professores e alunos baseado nas necessidades do curso de Sistemas de Informações da Universidade Federal de Santa Catarina e no conceito de Ambiente Virtual de Aprendizagem utilizando-se de tecnologias *open-source* e da metodologia de desenvolvimento de processos ágil *EasyProcess*.

1.4.2 OBJETIVOS ESPECÍFICOS

I – Estudar os diversos tipos de relacionamento existentes entre os usuários de um ambiente de aprendizado virtual estudados e as demais funcionalidades inerentes ao processo de aprendizagem.

II – Desenvolver um sistema de informação em ambiente web utilizando tecnologias *open-source*.

III – Utilizar uma metodologia ágil para gerenciar o processo do desenvolvimento do sistema de informação.

1.5 DELIMITAÇÃO DO ESCOPO

Não é intenção do Aloha o desenvolvimento de infra-estrutura necessária para a realização de vídeo aulas ou vídeo conferências. Não é intenção realizar uma discussão profunda sobre o conceito de Ambiente Virtual de Aprendizagem (AVA).

1.6 METODOLOGIA DE DESENVOLVIMENTO

1.6.1 A PESQUISA E O MÉTODO

A pesquisa é um conjunto de ações propostas a fim de encontrar a solução para um problema proposto, utilizando-se de métodos para isto.

Segundo Ferreira (2002), método é "o caminho pelo qual se atinge um objetivo". O método de pesquisa utilizado foi o método indutivo, uma vez que o mesmo busca observar os fenômenos a fim de identificar alguma relação entre os mesmos para por fim haver uma generalização destas relações.

1.6.2 CARACTERIZAÇÃO DA PESQUISA

Esse trabalho, sob o ponto de vista de sua natureza, pode ser caracterizado como uma pesquisa aplicada, uma vez que todo conhecimento gerado será utilizado para um objetivo específico, o desenvolvimento do Projeto Aloha. Analisando do ponto de vista de abordagem do problema, a pesquisa em questão pode ser enquadrada como uma pesquisa qualitativa, visto que os dados pesquisados são indutivos, ou seja, não requerem técnicas de estatísticas para analisá-los, uma vez que há uma relação dinâmica entre o mundo real e o objetivo de pesquisa.

Analisando sob a ótica dos objetivos da pesquisa, a mesma enquadra-se como uma pesquisa exploratória, porque envolve levantamento bibliográfico e entrevistas com pessoas que podem colaborar com a pesquisa em virtude de suas experiências e necessidades. Ainda por interferências destas entrevistas, a pesquisa em questão pode ser classificada, segundo Gil (1991), como uma pesquisa de levantamento, a qual envolve a interrogação de pessoas com o objetivo de conhecer seus comportamentos.

1.7 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado como definido a seguir. O capítulo 1 apresenta os aspectos introdutórios do trabalho, seus objetivos e as justificativas. O capitulo dois apresenta a parte teórica da metodologia de gerencia de projetos *EasyProcess*. O capitulo três trata sobre os Ambientes Virtuais de Aprendizado. No capitulo quatro são explicitadas as tecnologias e ferramentas utilizadas no desenvolvimento do projeto. E finalmente o capitulo cinco apresenta os resultados obtidos com este trabalho.

2 EASY PROCESS – YP

O *EasyProcess (YP)* é uma metodologia de desenvolvimento de software criada com o intuito de atender o desenvolvimento de projetos acadêmicos. Esta metodologia foi idealizada pela Professora Drª Francilene Procópio Garcia da Universidade Federal de Campina Grande (UFCG) e foi concebida no ambiente do grupo PET Computação desta mesma universidade, no ano de 2003.

O *YP* foi moldado com base nas características de um projeto de desenvolvimento de software acadêmico, como, por exemplo, o escopo pequeno, em virtude da duração de um semestre, e a utilização de tecnologias consideradas estado da arte. Além disto, esta metodologia prima pela boa produtividade do projeto, amparado pelo uso de boas ferramentas de apoio e pela geração mínima de artefatos de software. O *EasyProcess* é uma metodologia simples e de fácil entendimento pelos integrantes da equipe, porém é completo e robusto a ponto de gerar produtos de qualidade.

Esta metodologia está apoiada em práticas do XP, RUP e *Agile Modeling*, utilizando o que há de mais adequado em cada uma delas em relação ao ambiente acadêmico.

Mas por que estudar uma metodologia no qual o foco são projetos acadêmicos? Quando se fala em projetos acadêmicos pode se entender também pequenos projetos, ou seja, o YP pode ser utilizado como a metodologia para pequenos projetos. Por exemplo, empresas de desenvolvimento de software podem desenvolver pequenos softwares, ou módulos, para utilização pela própria empresa. Para estes projetos pode utilizar o YP.

As fases do *EasyProcess*, conforme Garcia (GARCIA et al, 2007), são descritas nos itens deste capitulo a seguir.

2.1 IDENTIFICAÇÃO DO ESCOPO DO PROBLEMA

Consiste no estudo inicial do escopo do problema, a fim de obter conhecimento prévio que dê base ao entendimento e argumentação para a fase de conversa com o cliente, a qual será descrita posteriormente.

2.2 ESPECIFICAÇÃO DE PAPÉIS

É esperado que a equipe que participa do desenvolvimento do software seja dividida em papéis, os quais têm um conjunto de responsabilidades que determinada pessoa deverá desempenhar durante partes específicas do processo.

O *EasyProcess* recomenda a utilização de cinco papéis:

- Cliente: É a pessoa que solicitou o desenvolvimento do software. Suas principais atividades são: definir os requisitos do sistema, priorizar as funcionalidades, ajudar na elaboração do plano de *releases*, explicitar os testes de aceitação, identificar o perfil do usuário, identificar os objetivos de usabilidade, validar o protótipo de interface, validar o projeto arquitetural e estar ativo no processo de desenvolvimento do sistema. O ideal seria que o cliente estive presente o máximo de tempo possível com os desenvolvedores, seja na definição do sistema, no refinamento do protótipo ou na implementação dos testes de aceitação.
- Usuário: Esta é a pessoa que irá utilizar o sistema produzido, portanto o desenvolvimento do software deve atentar para as características desta pessoa. Logo, a presença do mesmo no desenvolvimento do software seria de grande valia quando possível. Suas principais responsabilidades são: ajudar na definição dos testes de aceitação, ajudar na identificação dos objetivos de usabilidade, validar o protótipo de interface e avaliar continuamente a interface do sistema.
- Gerente: É aquele que toma decisões referentes aos riscos e rumos do projeto, sendo responsável por coordenar as atividades de todos os outros membros do projeto. Suas competências abrangem: conduzir os

planejamentos e as ações dos desenvolvedores, elaborar o plano de desenvolvimento, avaliar sistematicamente os riscos descobertos, gerenciar configurações, coletar e analisar métricas, alocar testadores, resolver conflitos internos, tornar a documentação do projeto sempre atualizada e acessível.

- Desenvolvedor: É responsável por modelar os requisitos do sistema e produzir a codificação eficiente e correta para tais requisitos. O desenvolvedor deve utilizar-se de testes unitários a fim de encontrar erros em seu código e corrigir. As principais responsabilidades da pessoa que assume este papel são: levantar os requisitos funcionais e não funcionais junto com o cliente, auxiliar o gerente na elaboração de um plano de desenvolvimento, analisar e modelar a tarefa, gerar um protótipo da interface, identificar os objetivos de usabilidade, gerar testes de unidade, elaborar o projeto arquitetural – o qual deve ser validado pelo cliente –, construir o modelo lógico de dados, manter a integração contínua de código.
- Testador: É responsável por realizar testes de integração do sistema, por revisar os códigos dos desenvolvedores, refatorar se possível àqueles códigos a fim de torná-lo mais simples, legível, flexível e claro, porém caso haja um refatoramento de código, é necessário que o código seja testado tanto antes quanto depois do refatoramento, garantindo assim que as funcionalidades permanecem intactas. Além disso, o testador deve gerar os testes de aceitação e elaborar o material para realização dos testes de usabilidade.

2.3 CONVERSA COM O CLIENTE

Trata-se da primeira conversa entre o cliente e os desenvolvedores do sistema, tendo como objetivo fazer com que ambas as partes tenham uma idéia comum a respeito do sistema em questão. Ao final desta fase o escopo do problema deve estar bem definido, e deve-se conhecer o perfil dos usuários e suas necessidades, além saber quais os requisitos funcionais e não funcionais do sistema

e deve-se também listar os riscos do projeto. Para tal, o *EasyProcess* utiliza-se de elementos tais como os enumerados a seguir.

2.3.1 DOCUMENTO DE VISÃO

Após a conversa inicial com o cliente o documento de visão deve ser criado utilizando uma linguagem de fácil entendimento para o cliente, visto que este será um elo de comunicação entre o cliente e o desenvolver. O documento de visão deverá conter as idéias gerais sobre o que o sistema se propõe a fazer. Depois de concluído este documento de visão pelos desenvolvedores, o mesmo deve ser validado pelo cliente, a fim de verificar se aqueles compreenderam a visão do sistema, uma vez que este documento é de suma importância pois serve como uma espécie de contrato entre as partes envolvidas.

2.3.2 REQUISITOS

Dentro do Documento de Visão deve haver uma sessão em que são descritos os requisitos funcionais e não funcionais do projeto. Os requisitos são características e funções de um sistema. Os requisitos funcionais descrevem as funções que o software deve possuir. Já os requisitos não funcionais descrevem as propriedades e restrições do sistema. Os requisitos não funcionais podem ser, por exemplo: de usabilidade, manutenabilidade e desempenho. Estes requisitos devem ser descritos pelos desenvolvedores em conjunto com o cliente.

2.3.3 PERFIL DO USUÁRIO

São as descrições das características dos usuários reais do sistema que são relevantes para o desenvolvimento do software. Estas descrições devem revelar habilidades, preferências, limitações, os interesses dos usuários e o conhecimento prévio dos mesmos em relação às atividades que o sistema irá desempenhar e ao uso do computador. A descrição do perfil dos usuários pode aparecer no Documento de Visão. A seguir é mostrada uma lista de algumas características relevantes a serem observadas na primeira conversa com cliente, segundo a documentação do *EasyProcess*:

- Sexo;
- Canhoto, destro ou ambidestro;
- Uso de lentes corretivas;
- Faixa etária;
- Experiência prévia no uso de sistemas computacionais;
- Tempo de uso de sistemas computacionais;
- Freqüência de uso de sistemas computacionais;
- Plataforma computacional que utiliza com maior freqüência;
- Conhecimento prévio de outra versão do sistema que está sendo desenvolvido (melhorado);
- Freqüência de uso de tal versão;
- Familiaridade com a língua inglesa.

2.3.4 OBJETIVOS DE USABILIDADE

Define quais metas de usabilidade devem ser atingidas pelo software desenvolvido. Servem para avaliar a usabilidade do sistema com base no desempenho do usuário e deve ser desenvolvido pelo cliente e pelo usuário.

Geralmente estas metas referem-se à eficácia, eficiência, capacidade do sistema de ser memorável e aprendido pelos usuários, e pela a segurança. A eficácia diz respeito à forma com que as tarefas são realizadas e a disposição das informações necessárias para que a tarefa seja realizada. A eficiência trata do auxílio prestado ao usuário, pelo sistema, na realização de suas atividades.

2.4 INICIALIZAÇÃO

Após a fase de conversa com o cliente e de criação do Documento de Visão, dá-se inicio a fase de inicialização, a qual é composta por cinco atividades, descritas a seguir.

2.4.1 MODELAGEM DA TAREFA

Sua função é esclarecer os detalhes de cada tarefa do sistema. A modelagem da tarefa auxilia na geração da interface do sistema, visto que a mesma identifica as relações de hierarquias e dependência entre as sub-tarefas. Este modelo pode ser realizado com a utilização da ferramenta iTAOS (MEDEIROS 2003). Um exemplo de modelagem pode ser observado na Figura 1 a seguir.

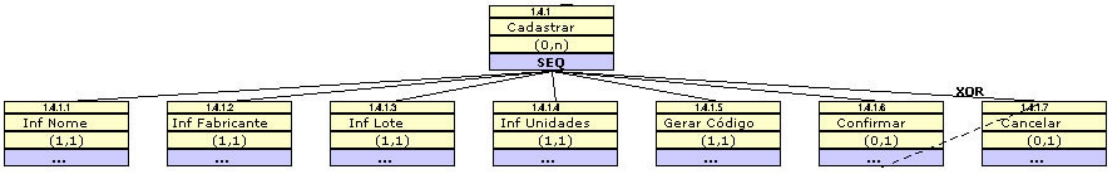


Figura 2.1 - Modelo da Tarefa

A software iTAOS é uma ferramenta gráfica para análise e modelagem de tarefa que utiliza o formalismo TAOS. Por sua vez, o TAOS é um formalismo que inicialmente visou a construção de um Sistema Baseado em Conhecimento e foi validado no ramo da biologia molecular. Porém este formalismo também é usado para a modelagem de tarefas, uma vez que o TAOS abrange todos os requisitos que a modelagem da tarefa exige.

2.4.2 DEFINIÇÃO DAS *USER STORIES*

Esta é a fase do projeto em que se definem as *User Stories* e estima-se o tempo de desenvolvimento para cada uma delas. *User Stories* são funções que o sistema deve desempenhar, as quais são definidas pelo cliente e pelo desenvolvedor. Após a definição das *User Stories* envolvidas no projeto, e que pode sofrer alterações no decorrer do tempo, é possível construir uma tabela com as *User Stories* do sistema e suas respectivas estimativas, como observado na Tabela 2.1 a seguir, retirada do site oficial da metodologia YP.

Tabela 2.1 - Tabela de User Stories

User Stories	Tempo Estimado (Horas)
US1 – Cadastro dos clientes, incluindo dados profissionais e pessoais.	8
US2 – Cadastro dos produtos que serão comercializados, descrevendo as particularidades de cada produto.	6
US3 – Cadastro de categorias. Estas categorias dizem respeito às possíveis classificações a que um cliente pode pertencer.	4
US4 – Fazer registro das vendas efetuadas, incluindo informações dos produtos vendidos e do cliente.	8
US5 – Implementar mecanismo de busca para produtos (por nome, por código, por preço) e clientes (por nome, por categoria, por CPF).	4

2.4.3 PROTÓTIPO DE INTERFACE

Permite visualizar como se dá a interação do usuário com o sistema. O protótipo de interface pode ser construído com base no modelo de tarefas criado anteriormente.

2.4.4 MODELO LÓGICO DE DADOS

Trata-se da modelagem das entidades de domínio em um banco de dados se houver necessidade.

2.4.5 PROJETO ARQUITETURAL

O Projeto Arquitetural é um artefato que permite explicitar como as partes do sistema interagem entre si ou com outros sistemas. Ele descreve o software com um alto nível de abstração. Com este diagrama, o qual pode também apresentar uma descrição do mesmo, é possível identificar os pontos críticos do sistema, as dependências e os requisitos não funcionais levantados junto com o cliente.

A fim de tornar ciente a arquitetura geral do sistema é importante a presença de todos os desenvolvedores na elaboração do mesmo, e para que haja a validação do que foi elaborado, faz-se necessário a presença do cliente.

2.5 PLANEJAMENTO

Esta fase do processo é marcada pelo planejamento de como o projeto será desenvolvido no tempo. Considerando que o *YP* foi elaborado com base em projetos acadêmicos, tal metodologia aconselha a duração de três meses para o desenvolvimento do software, sendo este espaço de tempo dividido em três partes, denominados *releases*, sendo estes *releases* divididos em duas partes, chamadas de iterações. Logo, esta fase de planejamento divide-se em planejamento de *release* e de iteração.

O planejamento de *release* diz respeito à alocação das *User Stories* nos *releases* existentes, atentando sempre para as exigências do cliente em relação a qual *User Story* deve ser realizada primeiramente, e para o fato de que se deve primeiro alocar as *User Stories* que representam um maior risco para o projeto, pois o risco do projeto ser abortado depois de muito tempo é minimizado. É notável lembrar que todas as *User Stories* devem ter pelo menos um teste de aceitação. Portanto, uma *User Story* somente poderá ser considerada finalizada após a execução dos testes de aceitação.

Após a fase de planejamento de *release* dá-se a fase de planejamento de iteração. O planejamento de iteração é a divisão das *User Stories*, as quais estavam anteriormente englobadas em *releases*, em iterações. Cada *User Story* pode, se preciso, ser dividida em atividades com o intuito de facilitar a implementação de tal *User Story* e o gerenciamento da mesma. Portanto, cada iteração será um conjunto de atividades a serem desenvolvidas, e por sua vez, cada *release* será um conjunto de iterações.

Para auxiliar no planejamento das *releases* e das iterações é possível construir uma matriz de competência, mapeando as habilidades de cada membro da equipe e suas respectivas horas de dedicação semanal ao projeto.

As atividades oriundas das divisões das *User Stories* devem estar explicitadas na Tabela de Alocação de Atividades (TAA) contendo, para cada atividade, o tempo de desenvolvimento e o responsável pela mesma. Um detalhe interessante a ressaltar é que o tempo de desenvolvimento de cada atividade é

estimado pelo próprio desenvolver responsável, respeitando o tempo estabelecido pela iteração.

Por fim, o *EasyProcess* define que em hipótese nenhuma deve haver alteração no tempo dos *releases* ou iterações, e que, caso haja uma necessidade de reajuste, o escopo é que deve ser alterado. Além disso, é de extrema importância a disponibilização, por parte do gerente, de todo o planejamento realizado em um local onde todos os membros da equipe possam acessar.

2.6 IMPLEMENTAÇÃO

As atividades definidas no plano de iteração são realizadas nesta fase. O principal artefato construído é o código do sistema. Com o intuito de gerar uma boa codificação, o *EasyProcess* apresenta algumas práticas a serem seguidas. Estas práticas são descritas a seguir.

2.6.1 PROPRIEDADE COLETIVA DE CÓDIGO

Toda a equipe é responsável por cada arquivo de código do sistema, não necessitando a solicitação de autorização para o criador do arquivo para fazer modificações no mesmo. Portanto, o código é propriedade coletiva e conseqüentemente, toda a equipe é responsável por ele.

Com esta prática é possível atingir uma melhoria contínua do software e uma melhor recuperação de falhas, uma vez que trechos problemáticos de código podem ser identificados e resolvidos com mais facilidade por determinados membros da equipe, enquanto que outros não teriam a capacidade suficiente para realizar tal tarefa.

Porém, para que a propriedade coletiva do código seja efetuada com sucesso, é preciso que os códigos desenvolvidos sejam limpos, claros e de fácil entendimento, além de haver um controle de versão, a fim de saber quem está alterando cada parte de código em determinado momento, evitando conflitos.

2.6.2 BOAS PRÁTICAS DE CODIFICAÇÃO

Com o intuito de gerar um código limpo, o *EasyProcess* recomenda a aplicação de práticas como: Design Simples, Refatoramento, Padrões de Projeto e Padrões de Codificação. Essas práticas são descritas a seguir.

2.6.2.1 Design Simples

O objetivo desta prática é atingir a geração do melhor código possível, ou seja, um código de fácil entendimento, auto-explicativo e de bom desempenho. Portanto, faz-se necessário atentar para a utilização correta da flexibilização de código, uma vez que esta prática, quando em excesso, pode levar a produção de código desnecessário.

2.6.2.2 Padrões de Codificação

Com o objetivo de facilitar o entendimento do código gerado por todos os membros da equipe, os mesmos devem antes de começar o desenvolvimento do projeto, acordar questões relativas ao padrão do código a ser desenvolvido. Nomenclatura de variáveis e métodos, tamanho da identação utilizada, posição dos parênteses e chaves, são exemplos de questões a serem acordadas.

2.6.2.3 Padrões de Projeto

A fim de ganhar tempo, garantindo eficiência, e fazer com que o produto final se aproxime de um produto de qualidade, faz-se importante a utilização de padrões de projeto, os quais são soluções previamente pensadas e testadas, de qualidade assegurada, para a solução de diversos problemas comuns ao desenvolvimento de software.

2.6.2.4 Refatoramento

São pequenas modificações sobre o código considerado testado completamente e aceito de estar funcionando 100%, que visam melhorar algumas qualidades não funcionais, como por exemplo: simplicidade, flexibilidade, desempenho e clareza do código. Para tanto, é preciso que o tempo necessário para a execução do refatoramento seja previsto no planejamento da iteração. Além disso,

o *EasyProcess* recomenda utilizar o controle de versão para dar suporte a está prática.

2.6.3 INTEGRAÇÃO CONTÍNUA

Como forma de melhorar o gerenciamento do projeto e o trabalho dos desenvolvedores, caso estes não possuam horários em comum, o *EasyProcess* utiliza-se da integração contínua de código.

Na medida em que o código é produzido e testado, é importante que o mesmo já seja integrado continuamente ao sistema como um todo. Para isso faz-se necessário a utilização de ferramentas com tal funcionalidade como, por exemplo, o CVS (*Current Version System*) e o SVN (*Subversion*).

Com a utilização desta prática, a coleta de métricas por parte do gerente do projeto fica mais consistente, facilitando a análise real do desenvolvimento, uma vez que o produto do desenvolvimento está todo centralizado.

2.6.4 TESTES

Para assegurar, na medida do possível, a qualidade e corretude do sistema, e conseqüentemente a satisfação do cliente e do usuário, o *YP* recomenda principalmente três tipos de teste:

2.6.4.1 Teste de Unidade

Testa a estrutura interna do código, ou seja, a lógica e o fluxo de dados. Recomenda-se que os testes de unidade sejam realizados antes da codificação, pois esta é uma conduta que ajuda na solução do problema e melhora a qualidade do código desenvolvido. Devem ser validados aspectos como: condições de limite, tratamento de erros, manipulação de dados inconsistentes e impróprios.

2.6.4.2 Teste de Aceitação

São situações definidas pelo cliente sobre como medir o sucesso do projeto. Usuários finais podem ser ouvidos também na definição destes testes para reforçar aspectos levantados pelo cliente. A elaboração destes testes é feita durante

o levantamento das *User Stories*, porém, a realização dos mesmos é feita após a finalização da *User Story.*

2.6.4.3 Teste de Usabilidade

Serve para verificar se os testes de usabilidade definidos pelo cliente e pelos usuários foram satisfeitos, servindo também como forma de verificar o grau de iteração dos usuários com o sistema. Para tal, o *EasyProcess* utiliza-se do roteiro de atividades e do questionário pós-teste. Este último tem por objetivo identificar o grau de satisfação do usuário ao utilizar o sistema e deve ser realizado após a execução do roteiro de atividades. Durante a execução do roteiro de atividades deve-se atentar ao levantamento de indicadores quantitativos, como por exemplo: número de erros repetidos, número de acessos à área de ajuda do software e tempo para realização das tarefas.

Portanto, o *YP* recomenda a seguinte seqüência de passos para a execução dos testes de usabilidade: elaborar o roteiro de atividades; recrutar os usuários para teste; observar os usuários executando as atividades do roteiro; colher os indicativos quantitativos para análise; aplicar questionários pós-teste; analisar os resultados obtidos e sanar as falhas detectadas.

2.7 FINALIZAÇÃO DA ITERAÇÃO

Ao final da implementação e teste de uma iteração, deve-se preencher os campos de *status* e tempo real de desenvolvimento de cada atividade da iteração na Tabela de Alocação de Atividades (TAA). Caso haja mais alguma iteração ou *release* a ser desenvolvida, deve-se voltar para a fase de planejamento de *release* e seguir toda a seqüência de passos seguintes.

2.7.1 REUNIÕES DE ACOMPANHAMENTO

Esta fase do processo deve ocorrer semanalmente em reuniões que visam recolher e analisar métricas, utilizando-se do *Big Chart* – explicado a seguir – e da Tabela de Alocação de Atividades. Nesta fase deve-se atentar para os riscos do projeto e para a necessidade explícita de refatoramento de código.

Com as reuniões de acompanhamento, o gerente do projeto tem uma visão dos resultados obtidos pela equipe em uma semana de trabalho. A partir daí o gerente é capaz de identificar os pontos fortes e fracos do projeto, podendo então tomar as atitudes cabíveis a fim de melhorar o andamento e o sucesso do mesmo.

O *Big Chart* pode ser visto como a análise quantitativa do andamento do projeto. As métricas são definidas pelo gerente do projeto de acordo com os pontos a serem analisados, como por exemplo, número de classes existentes, número de linhas de código, número de testes de aceitação prontos, testes de unidades que estão rodando perfeitamente, entre outros. Com este gráfico é possível identificar se o projeto não apresentou mudanças no decorrer da semana em questão, ou se ocorreu algo inesperado, como exemplo, o aumento do número de classes e a manutenção no número de testes de unidade.

Durantes estas reuniões os desenvolvedores podem solicitar que os seus códigos sejam refatorados quando os mesmos não foram gerados da melhor forma possível, esta é a chamada necessidade de refatoramento explícito. Porém esta prática deve acontecer com pouca freqüência, visto que é responsabilidade do desenvolvedor criar o seu melhor código.

Além disso, a gerência de risco deve estar sempre em pauta nestas reuniões uma vez que essa metodologia foi proposta para diminuir os riscos através da verificação constante da situação do projeto. Riscos são situações indesejáveis que ocorrem durante o processo de desenvolvimento. Alguns exemplos comuns são: custo do projeto, prazo determinado para o desenvolvimento do projeto e uso de tecnologias desconhecidas por parte da equipe. Para realizar a gerência de riscos, o *EasyProcess* utiliza-se da tabela de riscos, uma tabela contendo o risco identificado, a data de identificação, o responsável pela solução, o grau de prioridade e a solução encontrada. Com a tabela de risco é possível manter um histórico dos riscos encontrados e suas respectivas soluções a fim de agilizar a solução dos riscos futuramente identificados.

Outro fator que deve ser considerado em reuniões de acompanhamento é a necessidade de mudanças. Existem vários tipos de mudanças e cada uma tem o seu grau de impacto no andamento do projeto. Algumas implicam apenas em

modificações de implementação, outras em alocações de tarefas, outras em alterações no planejamento ou ainda do modelo lógico e projeto da arquitetura. Logo, cada mudança identificada deve ser analisada cuidadosamente pelo gerente do projeto, o qual tomará as atitudes cabíveis em resposta ao acontecimento. Cabe lembrar que o tempo determinado para as iterações e *releases* não deve ser alterado, mesmo em decorrência destas mudanças, restando a alteração do escopo do prometido, uma vez que acordado com o cliente anteriormente.

2.8 VERSÃO FINAL DO PRODUTO

Após a finalização do sistema, realizam-se os testes de usabilidades sobre o mesmo, feito pelos usuários, e caso o resultado seja positivo, gera-se a versão final do produto.

3 AMBIENTES VIRTUAIS DE APRENDIZADO

3.1 INTRODUÇÃO

Este capítulo tem como finalidade apresentar uma breve descrição a respeito do que é o *e-learning*. Além disso, apresenta-se um estudo mais detalhado a respeito dos Ambientes Virtuais de Aprendizado e a importância dos mesmos.

3.2 E-LEARNING

E-Learning refere-se à utilização das tecnologias da Internet para fornecer um amplo conjunto de soluções que melhoram o conhecimento e o desempenho (ROSEMBERG 2002). Sendo assim, o *e-learning* caracteriza-se como o aprendizado por meio da estrutura da Internet.

3.3 AMBIENTE VIRTUAL DE APRENDIZADO

3.3.1 INTRODUÇÃO

Em um mundo altamente competitivo, o aprendizado torna-se o grande diferencial. Ferramentas que tornam este processo mais ágil assumem um papel cada vez mais importante. Sendo assim, os Ambientes Virtuais de Aprendizado (AVA) vêm para suprir esta necessidade e, segundo JISC (2000), a impressão principal é que eles terão um impacto significativo no processo de aprendizagem e ensino no futuro.

Segundo Morgan (2003), os Ambientes Virtuais de Aprendizado (AVA), ou *Virtual Learning Environment* (VLE) estão tendo um grande papel na infra-estrutura de tecnologia da educação superior na atualidade. O AVA, segundo definição da mesma autora, pode ser definido como um software especificamente projetado para instituições de ensino e estudantes utilizarem no ensino e no aprendizado.

Os Ambientes Virtuais de Aprendizado são sistemas que se baseiam na utilização de tecnologias para auxiliar os alunos no processo de aprendizagem. Eles não mudam a forma como as pessoas aprendem, mas buscam novas alternativas de como estas podem ser ensinadas aproveitando seu potencial cognitivo e suas habilidades naturais, e, principalmente, mostrando que a responsabilidade de produzir a experiência de aprendizado pode ser dividida (Horton apud PEDROSO, 2002).

Assim com os ambientes de aprendizado, como por exemplo, as salas de aulas, os Ambientes Virtuais de Aprendizado dispõem de uma estrutura que suporta o processo de ensino e aprendizagem.

Um AVA dispõe uma estrutura na qual se pode praticar o *e-learning*. Porém *e-learning* e AVA não são sinônimos. Morgan (2003) afirma que a utilização do AVA não se restringe apenas ao ensino a distância, apesar do mesmo ter sido a principal ferramenta para a realização destes cursos nas universidades da América do Norte. O AVA é também um sistema de apoio para aulas presenciais, como ocorre na *University of Wisconsin System*, em que 80% da utilização do AVA assume tal finalidade.

3.3.2 PRINCIPAIS AVAS

Três AVAs parecem se destacar no mercado mundial:

- Moodle: O Moodle é um Ambiente Virtual de Aprendizado de código livre. Desenvolvido em PHP e de fácil instalação. Atualmente a UFSC está iniciando a sua utilização.
- Blackboard: O Blackboard é uma AVA proprietário e que recentemente comprou o AVA WebCT, tornando assim a sua participação do mercado ainda mais expressiva.
- Sakai: O Sakai é um AVA de código aberto, grátis e desenvolvido em Java. O Sakai suporta o ensino e aprendizado, colaboração em grupo e portfólios e colaborações em pesquisa.

3.3.3 PRINCIPAIS FUNCIONALIDADES

As funcionalidades dos AVAs auxiliam no processo de ensino e aprendizagem e, segundo Morgan (2003), as principais delas dizem respeita as atividades descritas a seguir:

- Gerenciar as atividades e os materiais da turma;
- Organizar e apresentar o conteúdo;
- Comunicação (síncrona e assíncrona);
- Avaliação da performance do aluno;
- Armazenar e disponibilizar as notas.

Um ambiente de aprendizado virtual propicia controle das atividades acadêmicas visto que algumas funções inerentes ao cotidiano das salas de aulas são gerenciadas pelo software. É possível, portanto, efetuar a chamada *on-line* dos alunos presentes, agendar trabalhos e avaliações, divulgar o plano de ensino da disciplina, divulgar dados de eventos e notícias relevantes à disciplina e outras funcionalidades. Além disso, pode-se armazenar e divulgar o material da disciplina em questão, servindo como repositório único de todas as disciplinas da universidade, tornando a pesquisa mais acessível aos seus alunos. Esta última foi apontada por Jenkins, Browne e Walker (JENKINS & BROWNE & WALKER 2005), como o principal uso do AVA.

Professores podem prover a seus alunos os documentos utilizados por suas aulas e também os conteúdos adicionais, como referências a outras leituras, por meio de um Ambiente Virtual de Aprendizado. Assim, alunos podem estudar o conteúdo disposto no AVA, seja à distância ou presencialmente em sala de aula.

Outro fator importante presente nos AVAs é a possibilidade de comunicação entre os seus participantes. Suleman (2003) define esta comunicação como síncrona e assíncrona. Comunicação assíncrona é todo tipo de mensagem postada para ser lida posteriormente, como por exemplo, fóruns, e-mails e *noticeboards*, os quais são informações em que a permissão de criação da mensagem é dada somente para usuários que têm acesso a criação. Por exemplo, um professor poderia utilizar um *noticeboard* para comunicar informações

administrativas de suas disciplinas, como a data da próxima prova, e os demais alunos poderiam somente visualizar tais mensagens, diferentemente de um fórum. Por sua vez, comunicação síncrona é a comunicação em tempo real, obtida, por exemplo, através de chats ou vídeo-conferências, e que permitem conectar pessoas que por alguma razão não poderiam estar na sala de aula.

A comunicação entre os membros do AVA dá suporte para o trabalho colaborativo, ou seja, propicia que os integrantes de uma turma acordem questões relativas aos seus trabalhos de aula, utilizando-se das funcionalidades de comunicação do AVA. Estas funcionalidades quebram as barreiras impostas pela distância, pelo tempo e pela timidez. As dificuldades inerentes a distâncias são eliminadas uma vez que única limitação para o acesso ao processo de aprendizagem é uma conexão a internet. As mensagens assíncronas permitem que pessoas troquem mensagens nas suas horas mais convenientes e impedem que incompatibilidades nos horários dos integrantes atrapalhem o rendimento do trabalho, eliminando assim as barreiras do tempo. E, conforme descrito por Nancy Chick da Universidade de *Wisconsin*, em Morgan (2003), a utilização do AVA encoraja a participação dos alunos tímidos, portanto a timidez torna-se menos prejudicial ao aprendizado. A comunicação presente no AVA também dá base para o suporte dos professores e monitores para com os alunos.

Porém, além de possibilitar o controle das atividades acadêmicas, garantindo o suporte para a comunicação entre os integrantes, o AVA, conforme descrito por Catley (2004), tem a capacidade de propiciar um maior engajamento dos alunos com as suas disciplinas. Tal qualidade é obtida devido às funcionalidades de avaliação, exercício e *quizzes on-line*. Por serem automáticas, tais funcionalidades permitem o retorno imediato, ou seja, é possível obter a correção dos exercícios e avaliações on-line em questão de poucos segundos. Com isso o aluno obtém um *feedback* do seu desempenho mais rapidamente, induzindo o mesmo a focar nos seus erros e procurar o suporte dos professores, seja on-line ou presencialmente na sala de aula, a fim de suprir suas carências o mais rápido possível. Nota-se então que tal sistema é diferente do tradicional sistema de aprendizado em que, após a realização da prova, espera-se, em grande parte das vezes, um considerável tempo até obter o resultado das mesmas. Por conseqüência,

o atraso da busca da correção dos erros por parte dos estudantes, resulta em notas baixas e reprovação. Além disso, estas funcionalidades são mais econômicas e rápidas, visto que reduzem o custo com fotocópias e agilizam a elaboração e correção das mesmas.

Outro fator motivante é que depois de corrigidas automaticamente as avaliações on-line, o cálculo das notas e da média final é automaticamente efetuado, assim como a divulgação automática das mesmas.

3.3.4 INTEROPERABILIDADE

Em um ambiente real o Ambiente Virtual de Aprendizado pode interagir com alguns outros sistemas. Cada um destes sistemas assume um papel especifico no ambiente acadêmico e podem colaborar entre si.

O *Management Information System* (MIS) é um tipo de sistema gerenciador de informações e, segundo Jarvis (JARVIS et al 2005), é um sistema que permite que faculdades ou escolas gerenciem os elementos essenciais do seu negócio, incluindo os números dos alunos, construções, quadro de horários e orçamento, e que permite tomar decisões, planejar e responder aos pedidos de informações dos gerentes e demais solicitantes. Portanto, o MIS armazena os dados pessoais dos alunos, assim como as disciplinas nas quais ele está matriculado, sua grade de horário, suas notas e outros cadastros afins. Logo, torna-se plausível uma integração entre o AVA e o MIS no tocante à troca de informações, com o intuito de evitar entrada manual de dados excessiva e duplicação de dados. Sendo assim, é possível que as notas resultantes de uma avaliação *on-line* interativa alimentem o sistema gerenciador de informações da universidade, assim como o MIS poderia fazer o cadastro automático de todos os alunos e suas respectivas matérias no ambiente virtual de aprendizado, além de outros exemplos de interação.

Além da interação com o MIS, o AVA pode comunicar-se com o sistema gerenciador da biblioteca da universidade, mais conhecido como *LibMS* (*Library Management System*), a fim de fazer consultas a recursos e reservas.

Segundo Wilson (2002), esta integração entre Ambiente Virtual de Aprendizado, MIS e LibMS compõe o *Managed Learning Environment* (MLE), como ilustrado na Figura 2 a seguir.

Figura 3.1- Partes do MLE (WILSON 2002)

3.3.5 PESQUISAS RELEVANTES

A UCISA (*Universities and Colleges Information System Association*) e a JISC (*Joint Information Systems Committee*), ambas entidades inglesas que visam a melhoria da educação e da pesquisa pela utilização de inovações tecnológicas, publicaram em 2005 uma pesquisa sobre a utilização dos Ambientes Virtuais de Aprendizado no Reino Unido (JENKINS & BROWNE & WALKER 2005). Esta pesquisa já havia sido realizada em 2001 e 2003 e o objetivo principal foi permitir um estudo longitudinal dos Ambientes Virtuais de Aprendizado. Em 2001 a pesquisa foi realizada com base nas respostas dos questionários de setenta e cinco instituições de ensino, em 2003, cento e duas instituições responderam os questionários e em 2005 este número foi de oitenta e cinco. Alguns fatores interessantes são apresentados a seguir.

Na Tabela 3.1 observa-se que em 2005, 95% das instituições de ensino do Reino Unido já utilizavam um Ambiente Virtual de Aprendizagem e que este número cresceu deste o ano de 2001. Um número desta magnitude assegura a sua importância no meio acadêmico.

Tabela 3.1 - Se é utilizado um Ambiente Virtual de Aprendizado

	2001	2003	2005
Sim	81%	86%	95%
Não	19%	14%	5%

Na Tabela 3.2 percebe-se que, nestes três anos da pesquisa, muitas instituições de ensino possuíam mais de um AVA. As informações nestes casos podem estar descentralizadas e duplicadas. Observa-se também que, com o passar do tempo, o número de instituição utilizando o AVA aumentou, assim como o número de instituições que estão centralizando suas informações em um único Ambiente Virtual de Aprendizado.

Tabela 3.2 - Número de Ambientes Virtuais de Aprendizado

	2001	2003	2005
Não usa AVA	19%	14%	5%
Usa 1 AVA	29%	36%	52%
Usa 2 AVAs	24%	32%	21%
Usa 3 AVAs	25%	10%	14%
Usa 4 AVAs	3%	5%	4%
Usa 5 AVAs	0%	1%	5%
Usa 6 AVAs	0%	2%	0%

Na Tabela 3.3 verificou-se que no Reino Unido o Ambiente Virtual de Aprendizado comercial mais utilizado era o Blackboard, seguido do WebCT. Porém, após a fusão das duas empresas, conforme citado por Chasen (2005), a tendência é que o Blackboard esteja com uma parcela ainda maior do mercado desta área na atualidade.

Observa-se também que o número de instituições de ensino daquela região que utilizam Ambientes Virtuais de Aprendizado desenvolvidos na própria instituição está crescendo consideravelmente e que o Moodle passou a ser utilizado em 2005 por, repentinamente, 8% das instituições.

Tabela 3.3 - Qual Ambiente Virtual de Aprendizado é utilizado

	2001	2003	2005
Blackboard	34%	43%	43%
Colloquia	0%	1%	2%
FD Learning`s le	0%	1%	0%
FirstClass	29%	19%	8%
Lotus Domino	8%	7%	5%
Lotus Learning Space	16%	5%	3%
Merlin	0%	1%	2%
TopClass	0%	0%	2%
WebCT	60%	34%	37%
Granada Learningwise	7%	7%	3%
MS Sharepoint (só 2005)	0%	0%	4%
Outro AVA comercial (só 2005)	0%	0%	0%
Produto Comercial Baseado em Intranet	0%	5%	0%
Bodington	0%	3%	8%
Moodle (só 2005)	0%	0%	8%
AVA desenvolvido na própria instituição	11%	23%	38%
AVA na Intranet desenvolvido pela instituição	0%	26%	17%
Outros	0%	18%	3%
Não respondido	0%	0%	0%

A Tabela 3.4 ilustra os principais usos dos AVAs, especificamente no ano de 2005. Destaca-se o Suporte em Par, a Trabalho Colaborativo, a Submissão de Tarefas, as Avaliações Formativas e o Acesso ao Material do Curso e aos Recursos Externos.

O Suporte em Par é a situação em que usuários se reúnem através do AVA para ajudar uns aos outros no aprendizado de algo. O Trabalho Colaborativo diz respeito ao trabalho que envolve duas ou mais pessoas colaborativamente. Para isso, elas precisam compartilhar informações, sem barreiras e com sinergia. Portanto, boas funcionalidades de comunicação fazem-se necessárias em um Ambiente Virtual de Aprendizado. A Submissão de Tarefa representa a criação e divulgação de tarefas por parte dos professores que devem ser realizadas pelos alunos. As Avaliações Formativas são as avaliações com o objetivo de identificar o

que os alunos sabem e o que os alunos não sabem, com o intuito de melhorar o processo de ensino e aprendizagem.

Tabela 3.4 - Uso feito do Ambiente Virtual de Aprendizado

	2005
Avaliação online	65%
e-Portfolio	27%
Suporte em Par (Peer Support)	70%
Aprendizado baseado em problemas (Learning Based Problem)	54%
Trabalho Colaborativo	81%
Apresentações online dos Alunos	57%
Submissão de Tarefas	75%
Avaliação Formativa	75%
Acesso ao material do curso	98%
Acesso a recursos multimidia	57%
Acesso a recursos web	90%
Planejamento do Aprendizado (Learning Design)	21%
Outros	5%
Não respondido	2%

Observa-se na Tabela 3.5, os diversos tipos de integração do Ambiente Virtual de Aprendizado com os demais sistemas das instituições de ensino em 2005. As integrações automáticas de maior destaque foram: a entrada dos dados dos estudantes no AVA e a integração com o LibMS e com e-mail. A avaliação com a ajuda do computador diz respeito a qualquer tipo de avaliação que tenha a ajuda do computador para sua efetivação, seja via provas on-line ou via correção automática impostas pelas leitoras de cartão resposta. Tal modalidade de avaliação é utilizada com integração automática ao Ambiente Virtual de Aprendizado em 38% dos casos.

Tabela 3.5 - Integração entre AVA e outros sistemas

	2005		
	Automáticamente	Manualmente	Não Respondido
Entrada dos dados dos estudantes	63%	22%	14%
Entrada das escolhas dos módulos dos alunos	51%	29%	21%
LibMS	30%	23%	47%
Outros sistemas de biblioteca	27%	22%	51%
Portal	29%	6%	65%
E-mail	48%	8%	44%
E-Portfolio	15%	5%	81%
Avaliação Assistida por Computador	38%	10%	52%
Outros	14%	0%	86%

3.3.6 AVA NO CONTEXTO DA UFSC

A Universidade Federal de Santa Catarina atualmente já conta com o seu *Managed Information System* (MIS), representado pelo Sistema Acadêmico da Graduação (CAGR), e com o *Library Management System* (LibMS) que é desempenhado pelo Sistema Integrado de Bibliotecas Pergamum. Recentemente, a UFSC iniciou o processo de utilização do Ambiente Virtual de Aprendizado Moodle.

3.4 CONCLUSÕES

Os Ambientes Virtuais de Aprendizado são importantes ferramentas para o bom andamento e aperfeiçoamento do processo de ensino e aprendizagem no meio acadêmico. É importante, portanto, um estudo a respeito do assunto, e se possível, a implantação deste ambiente virtual. Detalhes relativos ao processo de implantação dos VLEs podem ser encontrados em Morgan (2003) e em Jenkins, Browne e Walker (JENKINS & BROWNE & WALKER 2005).

É importante que a instituição de ensino tenha a cultura de utilizar um Ambiente Virtual de Aprendizado para que o processo de utilização se torne normal e proveitoso ao invés de ser algo que seja desconfortável para os usuários. Para isso, é importante que sejam realizados treinamentos sobre a utilização dos AVAs com os alunos e professores, dentro de suas necessidades a fim de que os mesmos percebam a importância real de se utilizar uma ferramenta como esta.

4 TECNOLOGIAS E INFRA-ESTRUTURA

Para o desenvolvimento de um sistema de informação se faz necessária de definição das tecnologias que serão utilizadas e da infra-estrutura base para possibilitar a codificação por parte do programador.

Com o objetivo de obter o máximo de reutilização de código foram selecionados vários *frameworks* e bibliotecas para integrar o sistema. Estes *frameworks* e componentes serão integrados na infra-estrutura do sistema para o projeto Aloha.

Neste capítulo serão apresentadas as tecnologias, os *frameworks* e as bibliotecas que serão utilizadas no desenvolvimento do projeto.

4.1 ANÁLISE, PROJETO E PROGRAMAÇÃO ORIENTADA A OBJETOS

Na análise e projeto, a análise é responsável pela investigação do problema descrito, fazendo o levantamento de requisitos, entre outros. O projeto enfatiza a solução lógica, ou seja, como o sistema atende aos requisitos. Assim a análise e projeto orientado a objetos visam enfatizar a consideração de um domínio de problema e uma solução lógica, segundo a perspectiva de objetos (coisas, conceitos ou entidades). Na análise é feita a descoberta e descrição dos objetos de domínio. No projeto é feita a definição de elementos lógicos de software, que serão implementados em uma linguagem de programação orientada a objetos na fase de programação (LARMAN, 2000). A programação orientada a objetos é a metodologia utilizada para programar classes com base em objetos definidos e cooperativos (ALLEN, 2003).

4.2 LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação utilizada no projeto será Java, lançada oficialmente em maio de 1995 pela Sun. Nos seus 12 anos de vida a linguagem de

programação Java amadureceu, ganhou uma vasta biblioteca, e incorporou muitas tecnologias. O Java possui implementação para a arquitetura de pequenos e médios dispositivos móveis (JME), uma versão padrão (JSE) e uma versão *enterprise* (JEE) (DEITEL, 2003).

Todas as tecnologias são especificadas por uma comunidade, a *Java Community Process (JCP)*, formada por pessoas e empresas. As tecnologias Java são especificadas, passando inclusive por avaliação pública, e é construída uma implementação de referência da tecnologia (JCP 2: *Process Document*, 2004).

4.3 ORGANIZAÇÃO EM CAMADAS

A programação de sistemas orientados a objetos pode utilizar o conceito de camadas para separar determinadas atividades do sistema. Um exemplo de uso de camadas é o modelo de rede OSI, apresentado na Figura 4.1. Este modelo é dividido em camadas hierárquicas, ou seja, cada camada usa as funções da própria camada ou da camada anterior para esconder a complexidade e transparecer as operações para o usuário, seja ele um programa ou uma outra camada (WIKIPEDIA: Modelo OSI, 2007).

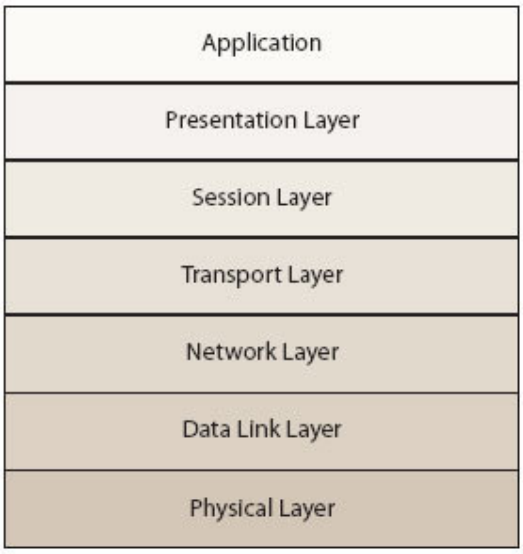


Figura 4.1 - Camadas de rede do modelo OSI

Neste formato o modelo OSI tornou-se um padrão, e sua divisão de responsabilidade em camadas serviu como exemplo para outros ramos da tecnologia.

Os sistemas orientados a objetos podem ser construídos também em camadas, sendo estas unidirecionais e bidirecionais, cada uma implementando e encapsulando sua complexidade. Apresentação, Serviço, Lógica de Negócio e Integração são exemplos de camadas utilizadas em sistemas (ALLEN, 2003), como mostra a Figura 4.2.

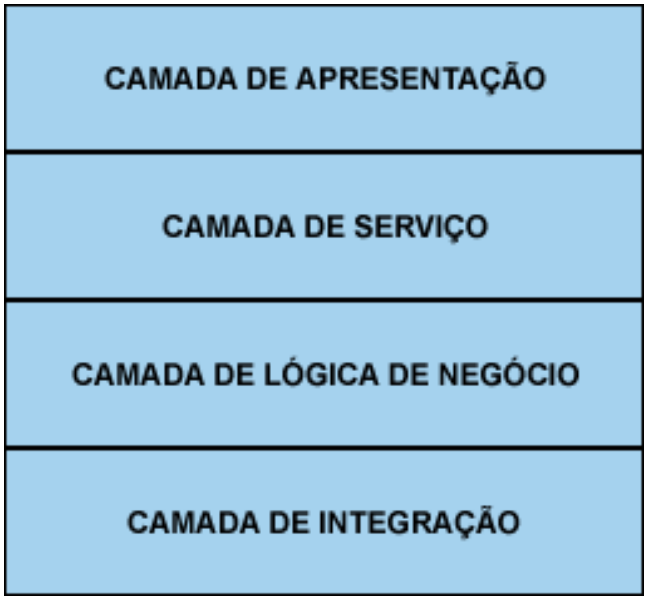


Figura 4.2 - Exemplo de camadas de sistema

A camada de apresentação implementa a lógica de apresentação, como telas, a implementação das ações geradas pelo usuário e os fluxos entre as telas do sistemas conforme as ações realizadas. A camada de Serviço implementa a integração entre a camada de Apresentação e Lógica de Negócio, realizando, por exemplo, o início e finalização de transações. A camada de Lógica de Negócio implementa os processos de negócios conforme os objetivos do sistema. A camada de Integração fornece acesso a Camada de Negócio a sistemas legados, como banco de dados, ou outros sistemas (ALLEN, 2003).

A seguir será mostrado o que será utilizado para realizar o objetivo deste projeto e quais tecnologias serão utilizadas em cada camada.

4.4 CAMADA DE APRESENTAÇÃO

A camada de apresentação tem como função apresentar os dados para o usuário, fornecendo um local para inserir e atualizar dados (ALLEN, 2003). No projeto será utilizado o ambiente web para apresentar os dados ao usuário através de código *HyperText Markup Language* (HTML). Será utilizado o Dynamic HTML (DHTML) para realizar interatividade nas interfaces gráficas. DHTML é a união das tecnologias HTML, Javascript e uma linguagem de apresentação, como folhas de estilo CSS aliada a um Modelo de Objeto de Documentos, para permitir que uma página Web seja modificada dinamicamente na própria máquina cliente, sem necessidade de novos acessos ao servidor web (WIKIPEDIA: DHTML, 2007). Também será utilizado o conceito de WEB 2.0 para que a interface se assemelhe com uma interface de *desktop*.

A programação da camada de apresentação seguirá o padrão *Modelo Visão Controle* (MVC). A abstração do MVC ajuda o processo de dividir a aplicação em componentes lógicos que podem ser construídos mais facilmente. O Modelo representa os dados da aplicação e as regras de negócio que definem o acesso e a modificação dos dados. A *Visualização* é a apresentação do conteúdo de uma parte particular do modelo, expondo os dados ao usuário. A visualização encaminha ações do usuário para o controlador. O *Controlador* define o comportamento da aplicação. Ele recebe a ação do usuário e realiza a chamada de um procedimento no modelo. Com base na ação do usuário e na resposta do modelo, o controlador seleciona uma visualização para ser exibida como resposta ao usuário (ALLEN, 2003).

Algumas tecnologias da camada de apresentação são executadas no lado do servidor (*server-side*) e outras tecnologias são executadas no lado do cliente (*client-side*). Por ser uma aplicação web o servidor é responsável por montar todo o código necessário, utilizando as tecnologias *server-side* e encaminhar este código gerado para o cliente, no caso um navegador web.

A seguir serão descritos os *frameworks* e tecnologias que serão utilizados nesta camada, que são executados no servidor.

4.4.1 APACHE TOMCAT

O Apache Tomcat implementa a especificação das tecnologias *Java Servlet* e *JavaServer Pages*. Ele é um *container* de *servlets*. (APACHE: Tomcat, 2007).

Um *Container Servlet* é uma parte de um servidor web, ou um servidor de aplicação, que oferece serviços que serão acessados através de requisições/respostas em uma rede.

O Apache Tomcat é desenvolvido pelo projeto Tomcat incubado pela Apache Software Fundation. É um projeto *open-source* e sua versão atual é 6.0.13 para a especificação 2.5 do Java Servlet. Sua logo é apresentada na Figura 4.3.

Este será o servidor utilizado para publicação do sistema Aloha.

Figura 4.3 - Logo do Apache Tomcat

4.4.2 JAVA SERVLET

Java Servlet (*servlet*) é uma tecnologia da especificação Java do grupo de componentes web. É especificada e mantida pela *Java Specification Request* (JSR) 154 e atualmente está na versão 2.5. Um *Servlet* é acessado por um cliente Web através do paradigma *request/response* implementado por um *Container Servlet*. As principais características dos servlets são:

- Uma classe *servlet* estende a classe *HttpServlet*;
- Geralmente estende os métodos *doGet (HttpServletRequest req, HttpServletResponse resp)* e/ou *doPost (HttpServletRequest req, HttpServletResponse resp)* para interceptar a ação do cliente Web.

A responsabilidade do *servlet* é receber a requisição do usuário, realizar o processamento definido e apresentar a resposta ao usuário, normalmente utilizando o formato HTML.

4.4.3 JAVASERVER PAGES

JavaServer Page (JSP) é uma tecnologia da especificação Java do grupo de componentes web. É especificada e mantida pela JSR 245 e atualmente está na versão 2.1. O JSP é uma extensão da tecnologia *Java Servlet* e permite aos desenvolvedores e *designers* um rápido desenvolvimento e fácil manutenção.

As principais características do JSP são:

- Programação com linguagem descritiva, como HTML ou XML;
- Não é necessário implementar classes, nem saber programação Java para fazer um JSP;
- Utilização de bibliotecas reutilizáveis, através de TAGS (JSTL).

A principal responsabilidade do JSP apresentar a resposta ao usuário, normalmente utilizando o formato HTML, e deixar o processamento da lógica de negócio ser realizada por um *servlet*. Um JSP também pode realizar o processamento da lógica de negócio, mas isso não é uma boa prática. O ideal é seguir o padrão Cliente-Magro no qual a responsabilidade do JSP é apenas a apresentação da resposta ao cliente.

4.4.4 JAVASERVER FACES

JavaServer Faces (JSF) é uma tecnologia da especificação Java do grupo de componentes web. É especificada e mantida pela JSR 252 e atualmente está na versão 1.2. É um *framework* para **interface com o usuário** (UI). Foi desenhado para facilitar o desenvolvimento e manutenção de aplicação que rodem em um servidor de aplicação Java.

As principais características do JSF são:

- Facilitar o desenvolvimento de interfaces com o usuário a partir de um conjunto de componentes reutilizáveis;

- Simplificar a transferência de dados da aplicação entre a camada de apresentação e a camada lógica;
- Manter o estado da interface do usuário entre requisições do servidor;
- Modelo simples para programar eventos gerados pelo cliente para o código do servidor de aplicação;
- Permitir a criação de componentes customizados.

A principal responsabilidade do JSF é realizar o ciclo de vida de uma requisição iniciada no cliente. O ciclo de vida corresponde a recuperar a visão, que é a última interface mostrada ao usuário, realizar a conversão dos valores enviados pela interface, processar as validações dos dados, atualizar os valores no objeto de modelo, invocar o método requisitado e montar a resposta para o cliente. Durante todo o processo o JSF faz o gerenciamento de erros que podem ocorrer.

O JSF também possui configurações para o gerenciamento da navegação entre as telas do sistema.

A Figura 4.4 apresenta o ciclo de uma requisição JSF.

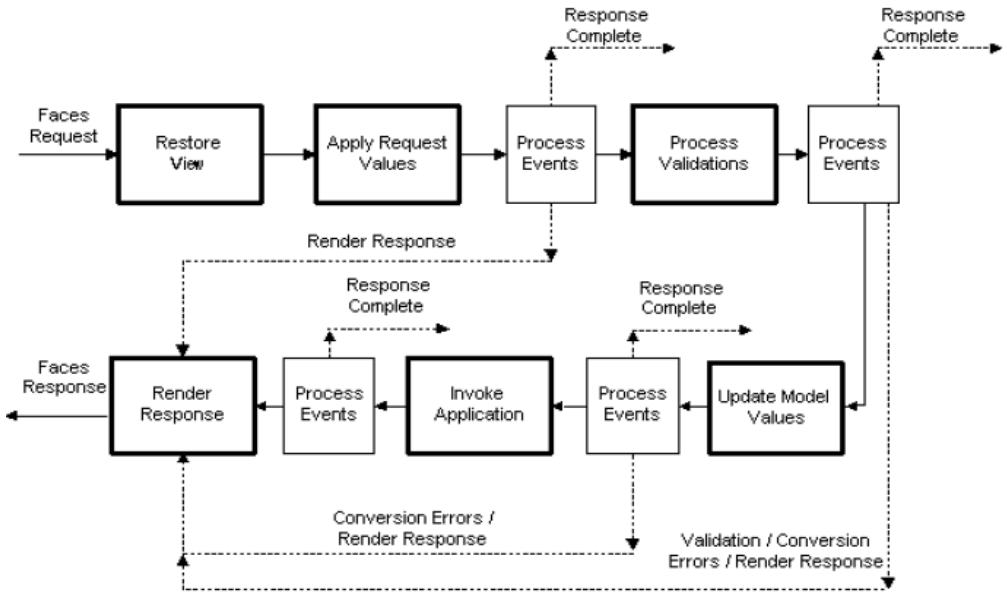


Figura 4.4 - Ciclo de uma requisição JSF

Fonte: (JCP: JSF, 2007)

4.4.5 FACELETS

Facelets é um *framework* de *templates* para ser utilizado com a tecnologia JSF. Um *framework* de *templates* é utilizado para reutilizar em várias interfaces um mesmo conjunto de código, como cabeçalho, rodapés, caixas de textos personalizadas, entre outros.

As principais características do *Facelets* são:

- Codificação rápida para componentes e páginas;
- Especifica componentes em arquivos separados, para reutilização;
- Gerenciador de exceção que informa a linha, *tag* ou atributo onde ocorreu o erro precisamente.

O projeto do *Facelets* é de software livre e é mantido pelo *java.net* através do site https://facelets.dev.java.net/. Atualmente encontra-se na versão 1.1.12.

4.4.6 AJAX

Asynchronous Javascript And XML (AJAX) é o uso sistemático de tecnologias providas por navegadores, como Javascript e XML, para tornar páginas mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. AJAX é uma iniciativa na construção de aplicações web mais dinâmicas e criativas. AJAX não é uma tecnologia, são realmente várias tecnologias trabalhando juntas, cada uma fazendo sua parte, oferecendo novas funcionalidades. AJAX incorpora em seu modelo:

- Apresentação baseada em padrões, usando XHTML e CSS;
- Exposição e interação dinâmica usando o DOM;
- Intercâmbio e manipulação de dados usando XML e XSLT;
- Recuperação assíncrona de dados usando o objeto *XMLHttpRequest*;
- *JavaScript*.

O principal benefício do AJAX é a possibilidade de alterar apenas uma parte da tela sem a necessidade de fazer uma submissão de toda a tela para o servidor (GARRETT, 2005).

Com o modelo clássico de uma aplicação web as requisições e respostas entre cliente e servidor são síncronas, conforme ilustrado na Figura 4.5.

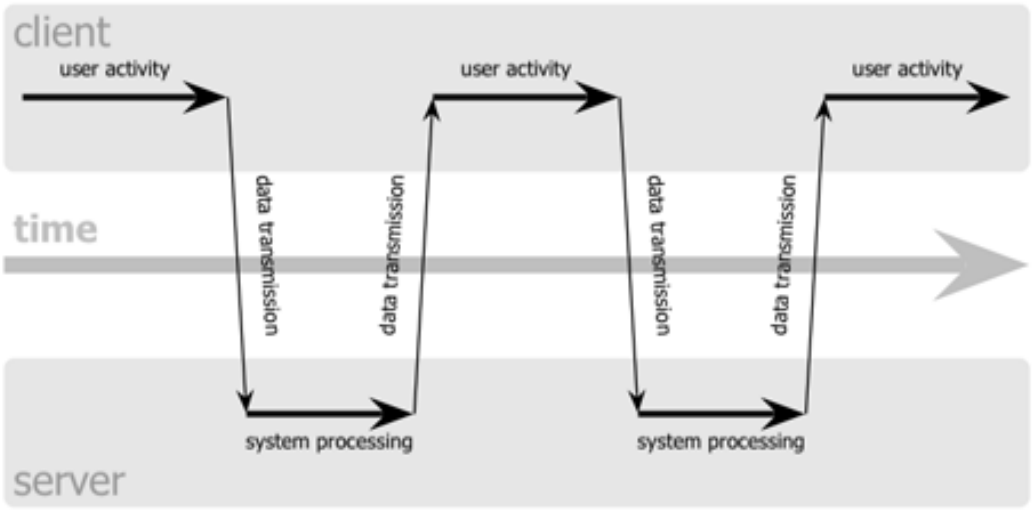


Figura 4.5 – Modelo clássico de uma aplicação web

A AJAX permite a utilização de chamadas assíncronas, conforme ilustrado na Figura 4.6.

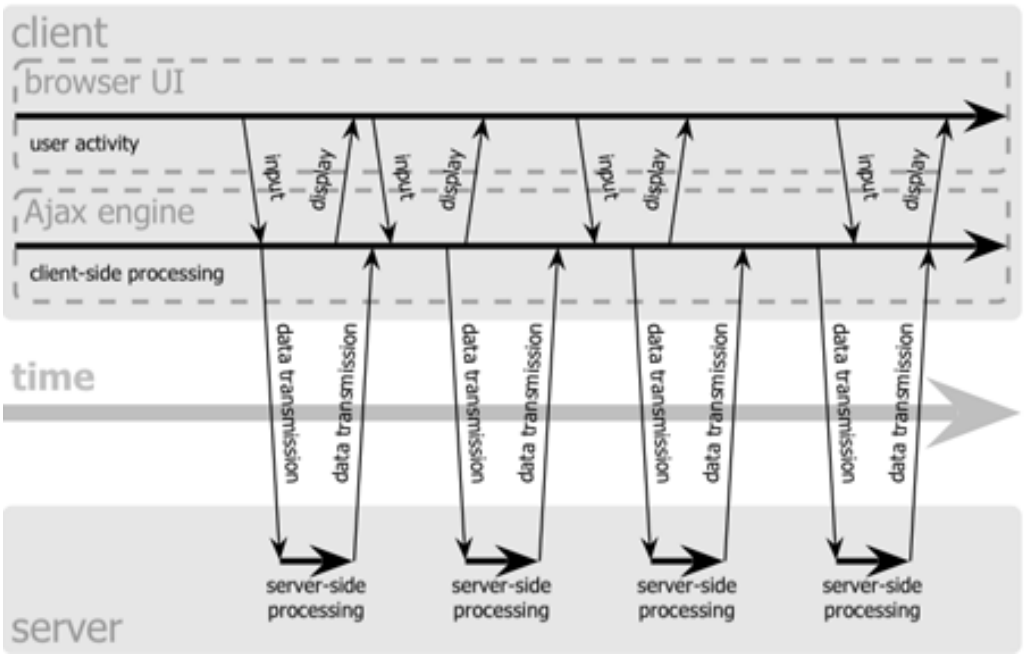


Figura 4.6 - Modelo AJAX de aplicação web

4.4.7 AJAX4JSF

O *framework* Ajax4jsf estende o JSF adicionando a capacidade de uso de AJAX em aplicações JSF sem a necessidade de codificar *JavaScrips*. O Ajax4JSF é *open-source*, está atualmente na versão 1.1.0 e é mantido e desenvolvido pela parceria realizada entre a Exadel e JBoss.

As principais características do Ajax4JSF são:

- Adicionar funcionalidades AJAX em aplicações JSF que já existem;
- Capacidade de fazer componentes customizados incluídos funcionalidades AJAX.

A Figura 4.7 ilustra como é simples adicionar AJAX a um componente JSF já existente.

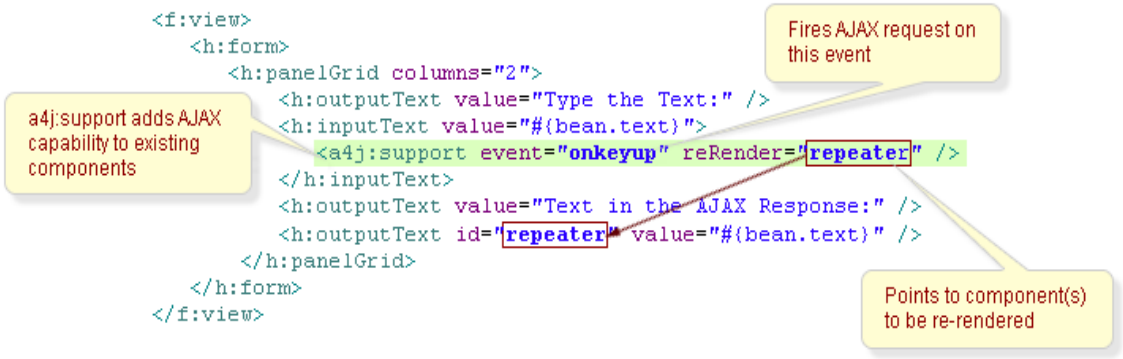


Figura 4.7 - Exemplo de uso do Ajax4JSF

A *tag <a4j:support>* é a responsável por capturar os eventos nos componentes JSF e realizar a ação especificada.

A Figura 4.8 mostra como o *framework* Ajax4JSF trabalha dentro do ciclo de vida do JSF.

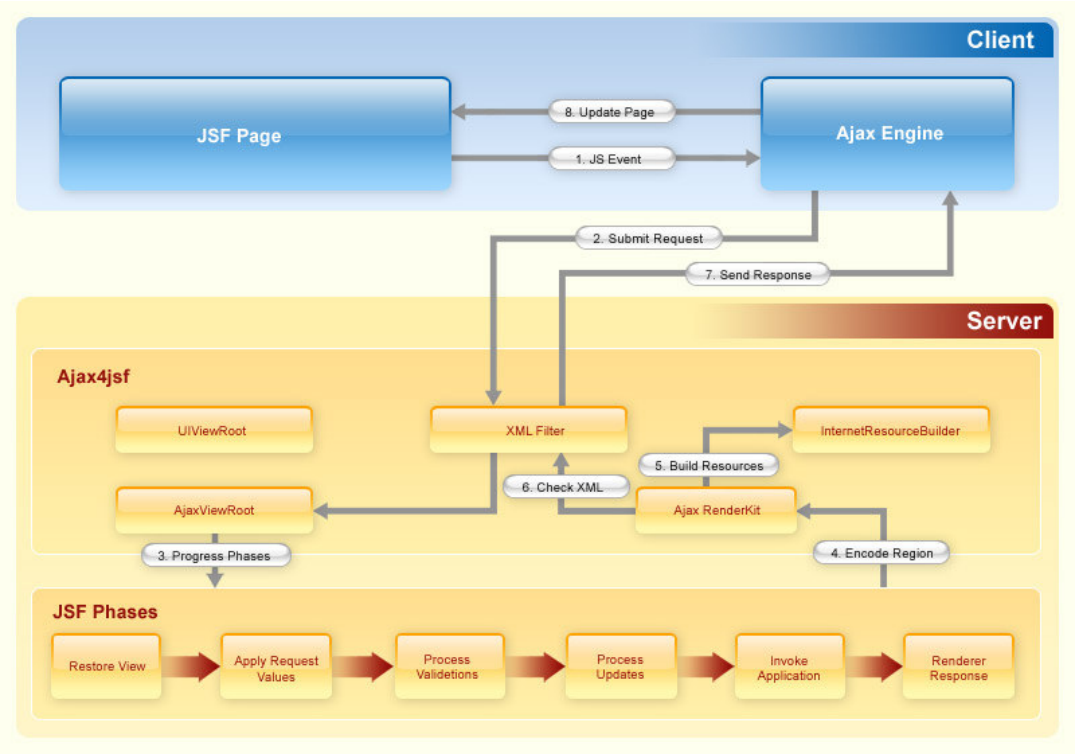


Figura 4.8 - Ajax4JSF dentro do ciclo de vida do JSF

4.4.8 RICHFACES

RichFaces é uma biblioteca de componentes JSF construídas utilizando o *framework* Ajax4JSF. Seus componentes proporcionam realizar funções como arrastar-e-soltar, criar tabelas de dados com paginação e ordenação, entre outros.

O RichFaces é *open-source*, está atualmente na versão 3.0.0 e é mantido e desenvolvido pela parceria realizada entre a Exadel e JBoss.

Esses componentes ajudam a melhorar a usabilidade do sistema, proporcionando o uso de técnicas que antes só estavam disponíveis em ambientes *desktop*.

A Figura 4.9 mostra os componentes disponibilizados pelo *RichFaces*.

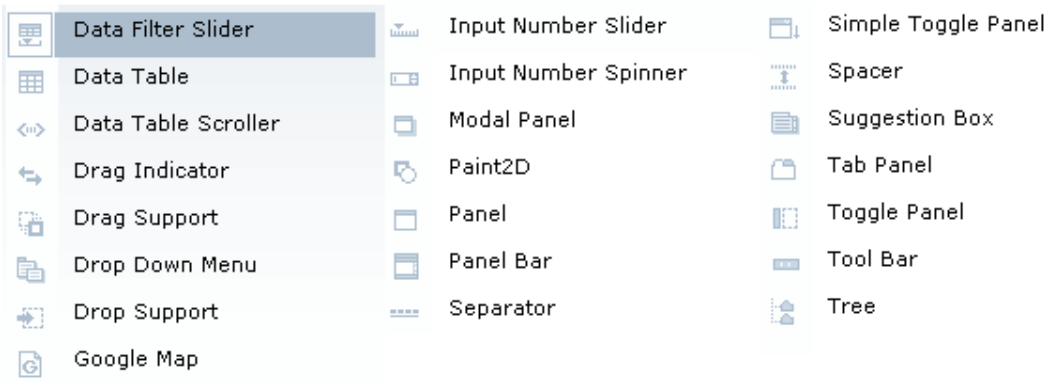


Figura 4.9 - Componentes disponibilizados pelo RichFaces

4.4.9 MYFACES TOMAHAWK

MyFaces Tomahawk, assim como o *RichFaces*, é uma biblioteca de componentes JSF. Ele faz parte do projeto *MyFaces* encubado pela Fundação de Software Apache. É *open-source* e atualmente está na versão 1.1.5. Seu logo é apresento na Figura 4.10.

Figura 4.10 - Logo do projeto MyFaces

O *Tomahawk* possui componentes que estendem componentes definidos na especificação JSF adicionando/modificando propriedades e comportamentos, como, por exemplo, o componente *<t:inputText>* que estende o componente *<h:inputText>*, que apresenta uma caixa de entrada de texto ao usuário. O *Tomahawk* também possui componentes utilitários que auxiliam o programador em tarefas durante a codificação, como, por exemplo, *<t:popup>* que apresenta para o usuário uma caixa de mensagem sobre a tela atual.

4.4.10 JSF SANDBOX

JSF Sandbox, assim como o *RichFaces*, é uma biblioteca de componentes JSF. Ele faz parte do projeto *JavaServer Faces RI* encubado pelo Projeto java.net. É *open-source* e atualmente está na versão 0.1.

O *JSF Sandbox* possui componentes que estendem componentes definidos na especificação JSF adicionando/modificando propriedades e comportamentos, como, por exemplo, o componente *<risb:selectItems>* que estende o componente *<f:selectItems>*, que apresenta as opções de uma caixa de seleção ao usuário. O *JSF Sandbox* também possui componentes utilitários que auxiliam o programador em tarefas durante a codificação, como, por exemplo, *<risb:contextMenu>* que apresenta para o usuário o menu de contexto, que aparece ao clicar o botão direito do mouse sobre a tela atual.

4.4.11 SPRING WEB FLOW

O *framework* Spring Web Flow (Web Flow) é o "C" do padrão de projeto MVC, ou seja, o controle da camada de apresentação. Ele será responsável por capturar os eventos gerados pelo usuário, invocar o modelo e devolver a resposta ao usuário, apresentando a tela. Seu logo é apresentado na Figura 4.11.

Figura 4.11 - Logo do Web Flow

O Web Flow é um sub-projeto do Spring Framework e é mantido e desenvolvido pela Interface21. É *open-source* e atualmente está na versão 1.0.3.

As principais características do Web Flow são:

- Implementa todas as funcionalidades da lógica de controle de uma aplicação web;
- Monta arquivos de fluxo que podem ser reutilizados na aplicação web;
- Possui integração com vários *frameworks* MVC, inclusive o JSF;
- Utiliza a mesma definição de fluxo caso mude o *framework* MVC;
- Controla automaticamente as ações de voltar, avançar e atualizar que podem ser geradas pelo usuário do navegador;
- Integração nativa com o Spring Framework;
- Método intuitivo de definição de fluxo através de diagramas de estados, conforme mostra a Figura 4.12.

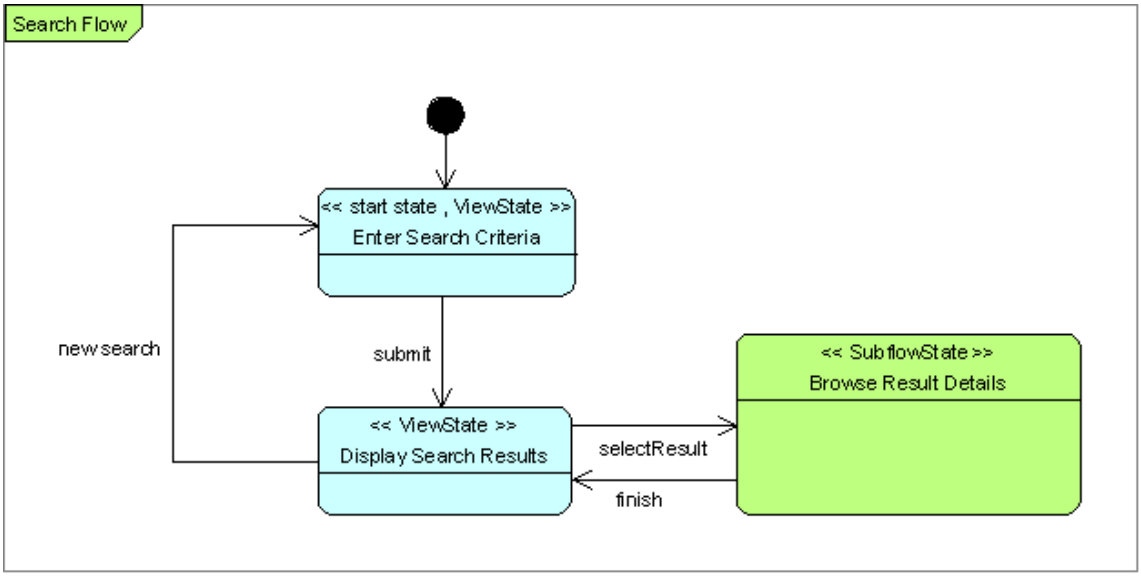


Figura 4.12 - Exemplo de um fluxo com Web Flow

A seguir serão descritos os *frameworks* e tecnologias que serão utilizados nesta camada, que são executados no cliente.

4.4.12 XHTML

A *Extensible HyperText Markup Language* (XHTML) é uma família de tipos de documentos e módulos que reproduzem e estendem a especificação HTML versão 4. A família XHTML é composta por tipos de documentos baseados no

formato XML, e são desenhados para trabalhar em conjunto com agentes baseados em XML. Um agente para XHTML pode ser um navegador web, como o Firefox, por exemplo.

A especificação HTML define como um documento HTML deve ser montado e interpretado pelo navegador web, tendo como resultado a apresentação da tela para o usuário.

A especificação XHTML 1.0 é o primeiro tipo de documento da família XHTML. É uma reformulação da árvore do tipo de documento HTML 4 para aplicações da especificação XML 1.0. A intenção é ser utilizado como uma linguagem de conteúdo em conformidade com a especificação XML e operar em agentes que interpretam a especificação HTML 4.

A especificação XHTML trás os seguintes benefícios:

- Documentos XHTML estão em conformidade com a especificação XML e podem ser lidos, editados e validados por ferramentas XML.
- Documentos XHTML podem ser interpretados da mesma forma, ou melhor, do que documentos escritos em HTML 4 em agentes em conformidade com o XHTML.
- Documentos XHTML possuem uma estrutura mais bem formada do que documentos HTML 4, pois os documentos devem seguir o formato XML.

A especificação XHTML é realizada pelo *World Wide Web Consortium* (W3C) e atualmente está sendo definida a versão 2.0. No projeto Aloha será utilizada a versão 1.0.

4.4.13 CSS

Cascading Style Sheets (CSS) é um mecanismo simples para adicionar estilos (como fontes, cores, espaços), em documentos web.

A especificação do CSS é realizada pelo *World Wide Web Consortium* (W3C) e atualmente está sendo definida a versão CSS3. No projeto Aloha será utilizada a versão CSS2.1.

O uso das formatações definidas na especificação CSS depende da implementação de cada navegador web. Os navegadores não são obrigados a suportar toda a especificação CSS. Assim deve-se tomar cuidado ao utilizar o CSS verificando se os navegadores que foram previstos como clientes do sistema suportam a formatação que se deseja utilizar.

4.4.14 JAVASCRIPT

JavaScript é uma linguagem de *script* baseada em objetos utilizada em páginas web. A Netscape implementa uma versão do JavaScript que compreende um conjuntos da funcionalidades previstas na especificação ECMA-262 3ª Edição (ECMAScript), que é a especificação padrão de linguagens de *script*.

As principais características do *JavaScript* são:

- Possibilitar a validação de formulários do lado do cliente;
- Interação com a página;
- Possui tipagem dinâmica;
- É interpretada e não compilada;
- Possui suporte a expressão regulares.

4.4.15 PROTOTYPE

O *Prototype* é um *framework JavaScript* que torna fácil o desenvolvimento de aplicações web dinâmicas. É desenvolvido e mantido por um conjunto de desenvolvimento chamado *Prototype Core Team*. É *open-source* e atualmente está na versão 1.5.1.

As principais características do *Prototype* são:

- Estende os objetos *JavaScript* criando funções que facilitam o desenvolvimento de páginas dinâmicas;
- Torna a aplicação web *cross-browser*, o que significa que o mesmo código pode ser executado em diferentes navegadores web;

- Implementa uma API simples para requisições AJAX.

Desenvolver páginas com *JavaScript* puro é trabalhoso. O *Prototype* tem como principal objeto facilitar a codificação da página. Sua logo é apresenta na Figura 4.13.

Figura 4.13 - Logo do Prototype

4.4.16 SCRIPT.ACULO.US

O *Script.aculo.us* é uma biblioteca de funções *JavaScript*. É implementado utilizando o *Prototype*.

As principais características do *Script.aculo.us* são:

- É fácil de utilizar e é *cross-browser*;
- Possui funções de animação, permitindo a movimentação, alteração de tamanho, forma, entre outras coisas, de objetos que fazem parte da página web;
- Possui função para arrastar e soltar;
- Possui funções utilitárias para manipulação do documento HTML (*DOM Utilities*).

Sua logo é apresentada na figura Figura 4.14.

Figura 4.14 - Logo do Script.aculo.us

4.4.17 RESUMO DA CAMADA DE APRESENTAÇÃO

O motivo de utilizar tantos *framework* e bibliotecas é para maximilizar o reuso de código. A Figura 4.15 mostra um resumo das tecnologias, *frameworks* e bibliotecas utilizadas na camada de apresentação.

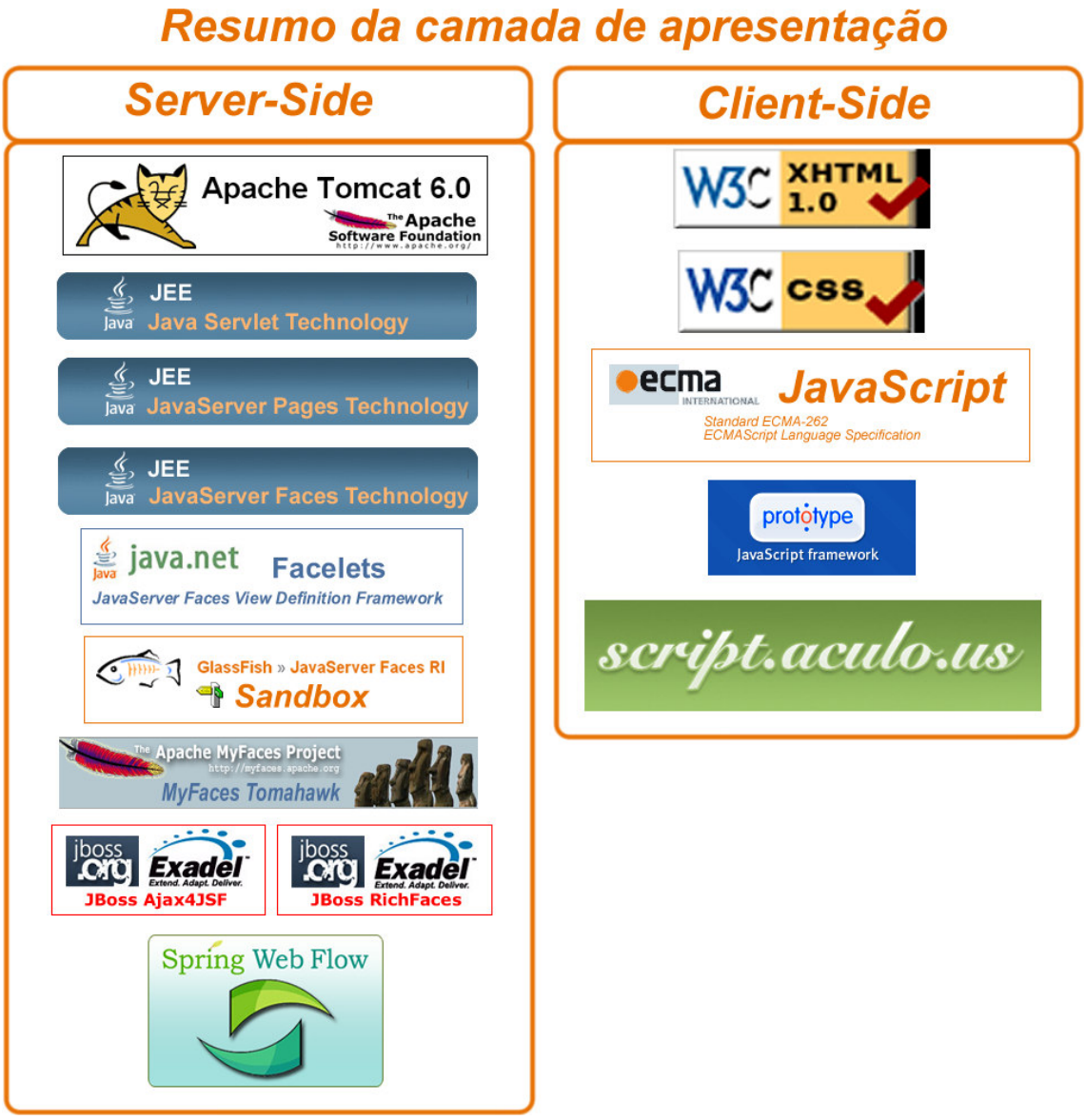


Figura 4.15 - Resumo da camada de aplicação

4.5 CAMADA DE SERVIÇO

A camada de serviço é o elo entre a camada de apresentação e a camada de lógica de negócio. Nesta camada são realizados alguns requisitos não funcionais do sistema, como abertura de transação, registro de erros, entre outros.

4.5.1 GERENCIADOR DE TRANSAÇÃO

As principais responsabilidades de um gerenciador de transação em um sistema são:

- Obter transação, se não existir, criar uma e retorná-la;
- Finalizar a transação com sucesso (*commit*);
- Finalizar a transação com erro (*roolback*).

Na infra-estrutura do Aloha a camada de serviço será responsável pela comunicação com o gerenciador de transação.

Uma transação será criada no momento em que a camada de apresentação acessar a camada de serviço solicitando a execução de uma lógica de negócio. Se a lógica de negócio for executada com sucesso a camada de serviço obtém o gerenciador de transação e invoca a operação de *commit*, fazendo com que as alterações realizadas durante a transação sejam persistidas. Mas se a camada de lógica de negócio lançar alguma exceção de erro, a camada de serviço obtém o gerenciador de transação e invoca a operação de *rollback*, desfazendo todas as alterações, retornando ao estado dos registros de antes do inicio da transação.

Será utilizado um gerenciador de transação local, ou seja, o sistema não suportará transações distribuídas.

4.5.2 REGISTRO DE ERROS

É uma boa prática realizar um registro de todos os erros que ocorrem no sistema, para servir de suporte para posterior averiguação do erro por parte do programador. Os erros em um sistema implementado com a linguagem Java são

gerador a partir do lançamento de uma exceção. O código abaixo demonstra o lançamento de uma exceção em Java:

throw new Exception("Descrição da Exceção");

A classe *Exception* do Java pode ser estendida, criando assim exceções customizadas. É uma boa prática dividir os erros nas seguintes situações:

- Erro gerado pela lógica de negócio do sistema;
- Erro gerado por algum dos *frameworks* ou componentes do sistema;
- Outros erros gerados.

Cada situação de erro pode ser gravada em um arquivo individual. Os erros gerados pelos *frameworks* e componentes utilizados pelo sistema devem ser analisados periodicamente, como uma maneira de se antecipar a uma provável reclamação do usuário.

4.5.3 SPRING FRAMEWORK

O Spring Framework (Spring) é um *framework* para aplicações Java/JEE que implementa soluções que todo sistema possui, por exemplo, um gerenciador de transação. O principal objetivo do Spring é encapsular toda a dificuldade de desenvolver uma aplicação Java/JEE e deixar que o programador se preocupe apenas em codificar as regras de negócio do sistema. Sua logo é apresentada na Figura 4.16.

Figura 4.16 - Logo do Spring

A Figura 4.17 mostra todos os componentes que o Spring implementa.

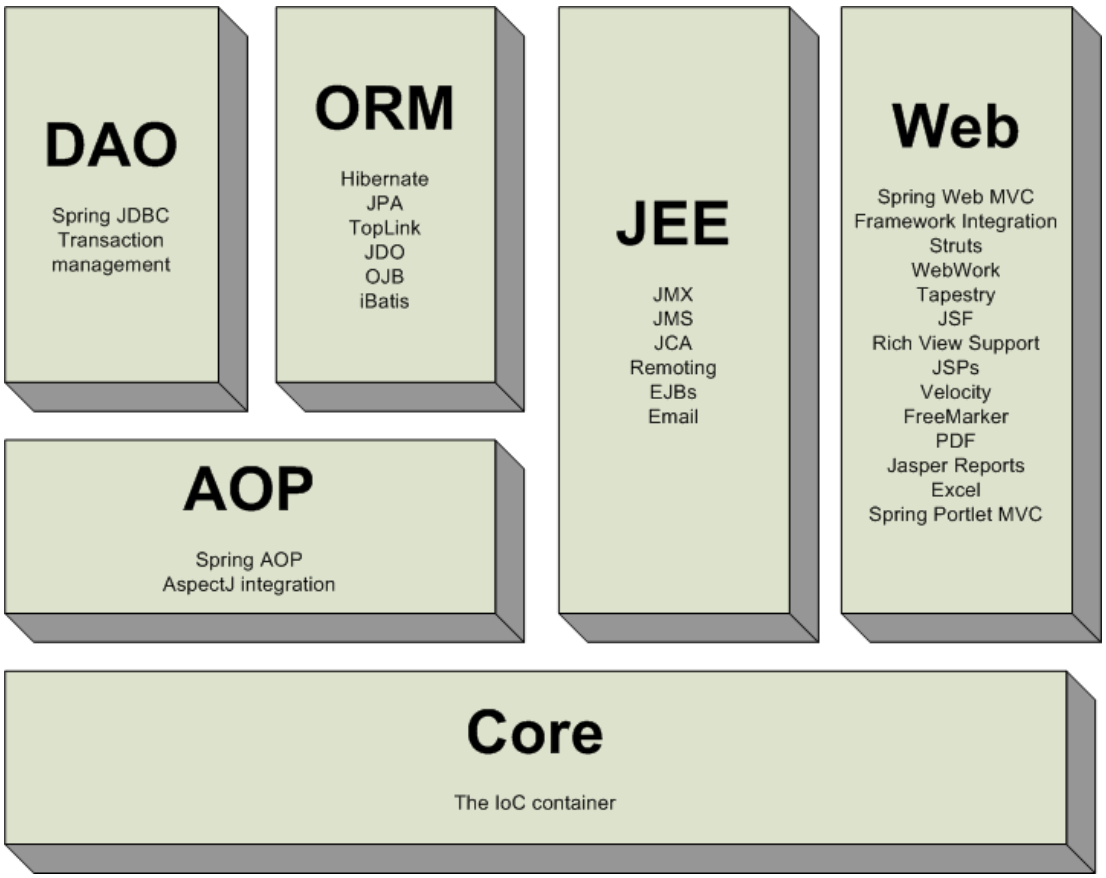


Figura 4.17 - Módulos e funcionalidades do Spring

O Aloha utilizará as seguintes implementações do Spring:

- Core – *IoC container*: todos os componentes Java utilizados em todas as camadas serão gerenciados pelo Spring. Durante a inicialização da aplicação o Spring instancia os componentes e disponibiliza ao programador através do padrão de projeto *Injection*;
- AOP – *Spring AOP e AspectJ Integration*: será utilizado para a gerência da execução dos métodos públicos da camada de apresentação, e também na implementação de um interceptador de chamadas de métodos entre a camada de apresentação e de negócio. Neste interceptador que será realizado o registro de erros quando ocorrer exceções, conforme descrito no item 4.5.2;
- WEB – JSF: os arquivos XHTML acessam diretamente componentes que fazem parte *container* de *beans* do Spring;

- JEE – Email: envio de e-mails;
- ORM – JPA: será utilizada a *Java Persistence API*;
- DAO – *Transaction Management*: será utilizado uma implementação do Spring como o gerenciador de transações do sistema. Este é o gerenciador que será acessado pela camada de serviço como descrito no item 4.5.1;

No capítulo 5 serão apresentados mais detalhes sobre a implementação do Aloha utilizando o Spring.

4.6 CAMADA DE NEGÓCIO

A camada de negócio é responsável por conter a codificação dos requisitos funcionais do sistema. Assim essa camada conterá apenas classes Java com a implementação dos componentes com a lógica de *negócio*.

Graças ao gerenciamento de transação e registro de erros que acontecem na camada de serviço, a camada de negócio não precisa conter nenhuma implementação para estas tarefas.

A Figura 4.18 apresenta um exemplo de como será uma classe da camada de negócio no Aloha. Percebe-se que a classe pode ter acesso a outras classes da camada de negócio, ou a camada de integração através das classes DAO.

As classes da camada de negócio serão gerenciadas pelo *container de beans* do Spring. Será de responsabilidade do Spring atribuir os valores das associações com outros componentes através do padrão *Injection*.

No exemplo da Figura 4.18 o componente *WorkSampleLogicImpl* possui associação com um componente de lógica de negócio (*TeacherLogic*), e duas associações com a camada de integração (*ActiveRecordDAO, WorkDAO*).

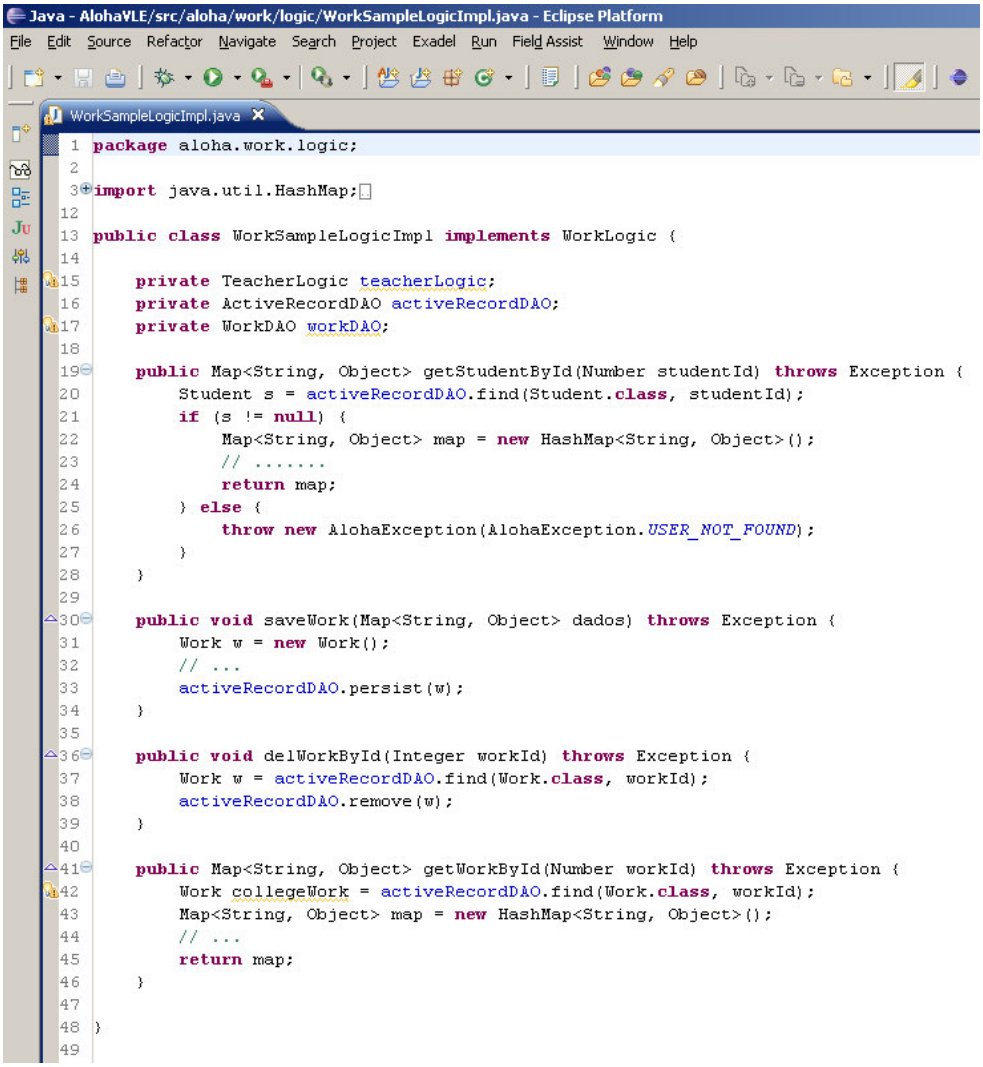


Figura 4.18 - Exemplo de uma classe de componente de lógica de negócio

4.7 CAMADA DE INTEGRAÇÃO

A camada de integração tem a responsabilidade de fornecer acesso à camada de negócio a outros sistemas utilizados no projeto. Estes sistemas podem ser Banco de Dados, *Web Services* externos, outros sistemas de informação, entre outros.

4.7.1 BANCO DE DADOS

O Aloha utiliza um banco de dados relacional para armazenar as informações. Um banco de dados relacional segue o Modelo Relacional. O modelo relacional é um modelo de dados, adequado a ser o modelo de um Sistema Gerenciador de Banco de Dados (SGDB), no qual todos os dados estão guardados em tabelas. Sua definição é teórica e baseada na lógica de predicados e na teoria dos conjuntos. O conceito Modelo Relacional foi criado por Edgar Frank Codd em 1970, sendo descrito no artigo *Relational Model of Data for Large Shared Data Banks* (WIKIPEDIA: MODELO RELACIONAL, 2007).

4.7.2 MAPEAMENTO OBJETO-RELACIONAL

O Mapeamento Objeto-Relacional (ORM) é uma técnica que mapeia as tabelas e seus atributos de um modelo relacional para classes e seus atributos no modelo de objetos. A Figura 4.19 exemplifica um mapeamento.

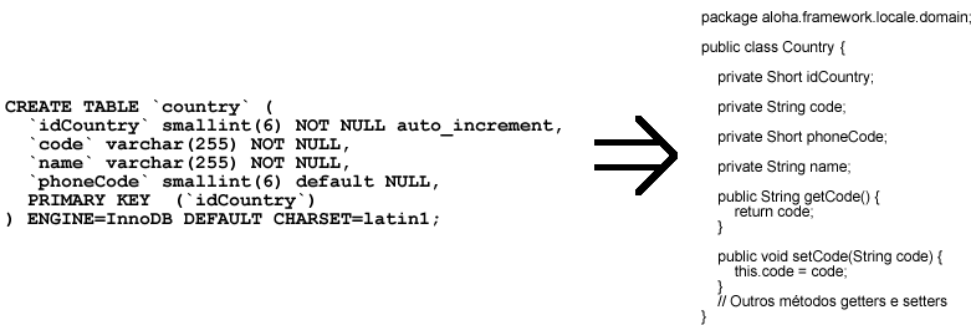


Figura 4.19- Exemplo de mapeamento Objeto-Relacional de uma tabela

Com a utilização de um *framework* que implementa esta técnica não é mais necessário a codificação de comandos na linguagem SQL para realizar operações no banco de dados, por exemplo:

- Gravar um registro (Comando *INSERT*);
- Atualizar um registro (Comando *UPDATE*);
- Excluir um registro (Comando *DELETE*);
- Selecionar um registro ou vários registros completos (COMANDO *SELECT * FROM tabela [WHERE id = 1]*).

Através do ORM um *framework* consegue realizar estas tarefas automaticamente. Isto representa um grande ganho de produtividade no processo de desenvolvimento de sistemas.

4.7.3 JAVA PERSISTENCE API

Java Persistence API (JPA) faz parte da especificação da tecnologia *Enterprise JavaBeansTM 3.0* do grupo *Enterprise Application Technologies* da linguagem Java. É especificada e mantida pela JSR 220 e atualmente está na versão 3.0. A JPA provê um modelo de persistência objeto-relacional utilizando *Plain Old Java Object* (POJO).

As principais características do JPA são:

- Prover uma especificação definindo um padrão para ORM em Java;
- Utilização de simples classes para representação de uma tabela do banco de dados;
- Utilização de anotações ou descrição em XML para definir o mapeamento do modelo relacional para o modelo de objetos, e vice-versa;
- Geração automática da estrutura relacional no banco de dados das anotações e descrições do arquivo XML.

A Figura 4.20 apresenta um exemplo de uma classe Java com as anotações especificadas pela JPA.

```java
package aloha.framework.locale.domain;

import javax.persistence.Basic;


@Entity
public class Country extends AbstractDomain {
    @Id
    @GeneratedValue(strategy = javax.persistence.GenerationType.AUTO)
    private Short idCountry;
    @Basic(optional=false)
    private String code;
    @Basic
    private Short phoneCode;
    @Basic(optional=false)
    private String name;
    public String getCode() {
        return code;
    }
    public void setCode(String code) {
        this.code = code;
    }
    // Outros métodos getters e setters
}
```

⇒

```
CREATE TABLE `country` (
  `idCountry` smallint(6) NOT NULL auto_increment,
  `code` varchar(255) NOT NULL,
  `name` varchar(255) NOT NULL,
  `phoneCode` smallint(6) default NULL,
  PRIMARY KEY  (`idCountry`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Figura 4.20 - Exemplo JPA

4.7.4 HIBERNATE

Hibernate é um *framework* poderoso e de alta performance que implementa os serviços de persistência no modelo objeto relacional e o serviço de consultas a banco de dados. Com o Hibernate pode-se desenvolves classes de persistência seguindo o modelo da orientação a objetos, incluindo associações, herança, polimorfismo, composição e coleções. Permite escrever consultas em uma linguagem estendida do SQL chamada *Hibernate Query Language* (SQL), e também permite realizar consultas com SQL nativo do banco de dados utilizado. A logo do Hibernate é apresentada na Figura 4.21.

Figura 4.21 - Logo do Hibernate

O Hibernate é desenvolvido e mantido pela Hibernate.org em parceria com a JBoss.org. É *open-source* e atualmente está na versão 3.2.4.SP1.

Através da extensão *Hibernate EntityManager* pode-se integrar o Hibernate com aplicações que utilizam a especificação EJB3 – JPA. O *Hibernate EntityManager* também é *open-source* e atualmente está na versão 3.3.0.

A Figura 4.22 apresenta os meios de como o *framework* de persistência Hibernate pode ser utilizado em aplicação Java ou até em aplicações *DOT.NET* da Microsoft.

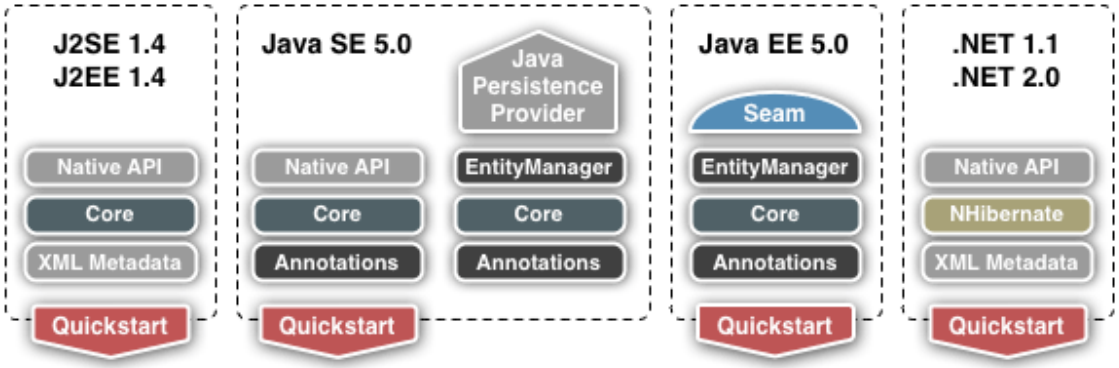


Figura 4.22 - Meios de utilização do Hibernate

4.7.5 OUTROS SISTEMAS

O Sistema Acadêmico de Graduação (CAGR) é o sistema utilizado na UFSC para as tarefas administrativas que envolvam os alunos da graduação. A partir do CAGR o Aloha buscará as informações necessárias nos domínios envolvidos no projeto, como Aluno, Professor, Cursos, Centros, entre outros.

Essa integração será através da tecnologia **Java Database Connect** (JDBC) acessando o banco de dados *SyBase* que é utilizado no CAGR.

4.8 UTILITÁRIOS

Existem funções que podem ser executadas em qualquer camada do sistema. Geralmente são funções que auxiliam a codificação. Essas classes com estas funções são chamadas de classes utilitárias.

Não necessariamente uma classe utilitária é implementada pelo projeto. Podem-se utilizar classes utilitárias desenvolvidas principalmente por projetos *open-source*, como as disponibilizadas pela *Jakarta Commons* da fundação Apache.

4.8.1 JAKARTA COMMONS

Commons é um subprojeto do projeto *Jakarta* incubado pela *Apache Software Fundation*. Seu principal objetivo é criar e manter componentes reutilizáveis em Java.

A Tabela 4.1 mostra os principais componentes mantidos pela *Apache Jakarta Commons*.

Tabela 4.1 - Principais Componentes Commons

Componentes	Descrição
Attributes	API de tempo de execução para adicionar meta dados à atributos assim como as tags *doclet*.
BeanUtils	APIs para introspeção e reflexão para beans Java de fácil utilização.
Betwixt	Serviços para mapear JavaBeans para documentos XML documents e vice versa.
Chain	Padrão de implementação *"Chain of Responsibility"*.
CLI	Parser de argumentos de aplicações via linha de comando.
Codec	Algorítmos genétricos de codificação/decodificação.
Collections	Extensão da API *Java Collections Framework*.
Configuration	Leitura de arquivos de configuração/preferências em vários formatos.
Daemon	Mecanismo alternative de invocação de código java usando unix-daemon.
DBCP	Serviço de pool de conexões para banco de dados.
DbUtils	Biblioteca utilizatária para a API JDBC.
Digester	Utilitário para mapeamento de XML para objetos Java.
Discovery	Ferramenta para localizar recursos pelo mapeamento de nomes de services/referencias para nomes de recursos.
EL	Interpretador para *Expression Language* definido pela especificação JSP 2.0.
Email	Biblioteca para envio de e-mail via Java.
FileUpload	Utilitário para realizar *upload* de arquivos via *servlets* e aplicações web.
HttpClient	Framework para trabalhar com o lado do cliente do protocol HTTP.
IO	Utilitários para I/O.
JCI	*Java Compiler Interface*
Jelly	Script baseado em XML e uma máquina de processamento.
Jexl	Linguagem de expressão que estende a Expression Language da JSTL.
JXPath	Utilitário para manipular *Java Beans* usando a sintaxe do *XPath*.
Lang	Prover mais funcionalidades para as classes do pacote *java.lang*.
Launcher	Inicializador de aplicações Java multiplataforma.

Logging	Implementações de APIs de *log*.
Math	Componentes para matemática e estatística.
Modeler	Mecanismo para criar *Model MBeans* compativeis com a especificação *JMX*.
Net	Coleção de utilitários para implementações de redes e protocolos.
Pool	Componente de pool generic.
Primitives	Pequeno, rápido e fácil de trabalhar com tipos primitivos em Java.
SCXML	Uma implementação da especificação do Gráfico de Estados XML.
Transaction	Implementação de vários níveis de locks, coleções transacionadas e transações em acesso a arquivos.
Validator	Framework que define validadores e regras de validação em um arquivo XML.
VFS	Componente *Virtual File System* para tratar arquivos, FTP, SMB, ZIP.

4.8.2 OBJECT-GRAPH NAVIGATION LANGUAGE

A Object-Graph Navigation Language (OGNL) é uma linguagem de expressão que permite obter e definir propriedades em objetos.

A OGNL é desenvolvida e mantida pela *OGNL Tecnology, INC*. É *open-source* e atualmente está na versão 2.6.9. Seu logo é apresentado na Figura 4.23.

Figura 4.23 - Logo do OGNL

A OGNL também é utilizada por outros *frameworks* como o Spring.

5 CONSTRUÇÃO DO ALOHA

5.1 INTRODUÇÃO

Este capítulo tem como objetivo apresentar os resultados obtidos com este estudo. O capitulo 5.2 apresenta os tipos de relacionamento existentes entre os membros de um Ambiente Virtual de Aprendizado, o capitulo 5.3 apresenta os resultados obtidos com a metodologia de desenvolvimento utilizada no projeto Aloha e o capitulo 5.4 apresenta as principais funcionalidades desenvolvidas no sistema em questão.

5.2 RELACIONAMENTOS

Os relacionamentos identificados que serão à base do desenvolvimento do projeto Aloha foram divididos em três categorias, explicadas nos itens a seguir.

5.2.1 ALUNO X ALUNO

O relacionamento entre alunos foi identificado no contexto de uma turma de uma disciplina e de uma equipe de desenvolvimento de um trabalho. Em uma turma os alunos precisam interagir entre si com o intuito de trocar conhecimento, informações ou experiências. Em uma equipe, os alunos interagem com um objetivo em comum, desenvolver um trabalho acadêmico solicitado por um professor.

A interação de alunos de diferentes turmas apesar de considerada importante, não foi contemplada na versão inicial do projeto Aloha, em virtude do tempo de desenvolvimento e da maior freqüência da interação entre alunos de uma mesma turma ou equipe.

5.2.2 ALUNO X PROFESSOR

A interação entre alunos e professores se dá por meio de uma turma de uma disciplina. Esta interação tem o objetivo de possibilitar a troca de conhecimento

entre ambas as partes e é possibilitada pela comunicação síncrona ou assíncrona e pelo compartilhamento de materiais didáticos. Como forma de auxiliar, direta ou indiretamente, esta troca de conhecimento e de contribuir para o aprendizado dos alunos, o professor faz uso de algumas atividades, como provas, trabalhos, chamadas, questionamentos que visam obter *feedback* dos alunos e outros. Portanto, otimizar e gerenciar estas atividades contribui para a troca de conhecimento entre os alunos e o professor e otimiza o aprendizado.

5.2.3 PROFESSOR X PROFESSOR

A interação entre professores ocorre principalmente entre aqueles que dominam assuntos em comum. Professores de uma mesma área de conhecimento interagem entre si a fim de trocar conhecimento e alinhar as metodologias de ensino e conteúdo de suas disciplinas.

As disciplinas de um curso de uma universidade estão agrupadas em áreas de conhecimento, como por exemplo, a área conhecimento de programação na Universidade Federal de Santa Catarina, é formada pelas disciplinas de Introdução à Programação Orientada a Objetos, Desenvolvimento de Sistemas Orientados a Objetos I, Estrutura de Dados, Desenvolvimento de Sistemas Orientados a Objetos II e Paradigmas de Programação.

Portanto, o agrupamento de professores se dá tendo em vista as disciplinas que lecionam.

5.3 METODOLOGIA

A metodologia de desenvolvimento de software utilizada no projeto Aloha foi a *EasyProcess* como descrito no capitulo 2 deste trabalho. Esta metodologia prevê que se construam alguns artefatos, como já explicados anteriormente nos itens daquele capitulo. Os artefatos construídos para o projeto Aloha são apresentados a seguir.

5.3.1 DEFINIÇÃO DE PAPÉIS

A Definição de Papéis do projeto Aloha é apresentada na Tabela 5.1.

Tabela 5.1 - Definição de Papéis

Equipe	**Papéis**
Adriano Manoel Demetrio	Gerente, Desenvolvedor, Usuário, Cliente, Testador
Arthur Martins Pereira	Gerente, Desenvolvedor, Usuário, Cliente, Testador

5.3.2 DOCUMENTO DE VISÃO

O Projeto Aloha visa criar um ambiente de relacionamento entre alunos e professores, entre alunos e alunos e entre professores e professores da Universidade Federal de Santa Catarina, onde será possível formalizar e agilizar processos inerentes aos relacionamentos existentes entre os mesmos. O Projeto Aloha visa estudar os diversos tipos de relacionamento existentes entre os usuários do sistema a fim de implementar uma solução que contribua para a eficiência dos mesmos. Este estudo se dará sobre observações do cotidiano no meio acadêmico do curso de Sistema de Informação.

O relacionamento Aluno-Professor é caracterizado pela existência de uma entidade de relacionamento Disciplina. Durante o andamento de uma disciplina, professores precisam tratar de assuntos referentes à própria disciplina com seus alunos, como por exemplo, a data de prova, disponibilização de material didático, divulgação das notas, etc. Além disso, professores podem informar seus alunos de algum evento relacionamento ao conteúdo da disciplina que esteja para acontecer, ou ainda de alguma notícia também relacionada com a sua disciplina.

O relacionamento Aluno-Aluno ocorre pela formação das equipes, as quais têm como objetivo reunir os integrantes para debater algum assunto referente a alguma atividade que os mesmos devem realizar em decorrência de um trabalho de uma disciplina. O relacionamento Aluno-Aluno acontece também durante uma disciplina, não necessariamente somente através de uma equipe, mas também entre os alunos que compõe a disciplina. Considerava-se que alunos podem contribuir com a disciplina, e esta contribuição atinge os demais membros da turma, como por

exemplo, alunos podem explicitar alguma experiência que tiveram relacionada com o assunto da disciplina para que os demais alunos aprendem com a mesma.

O relacionamento Professor-Professor é caracterizado pelos Grupos de Disciplina de Professores Afins. Estes grupos são formados por professores de uma mesma área de conhecimento. Exemplo: professores das disciplinas de Introdução a Banco de Dados, Banco de Dados I e Projetos de Banco de Dados podem se comunicar e interagir para trocar conhecimento e experiências.

Uma vez que o intuito do Projeto Aloha é atender principalmente aos alunos e professor da Universidade Federal de Santa Catarina, será estudada alguma forma de fazer a integração automática dos dados dos alunos, professores e suas respectivas disciplinas a fim de que, quando os usuários do sistema entrem no site do Projeto Aloha, todos os seus dados e seus relacionamentos já estejam preenchidos. Porém esta não será a intenção principal do projeto. Será estudada e avaliada a possibilidade de desenvolvimento da solução em questão durante o seu desenvolvimento. Será solicitado ao Núcleo de Processamento de Dados (NPD) da UFSC acesso ao banco de dados que contem as informações necessárias. Havendo a liberação do acesso e tempo hábil para programação da interface entre o Aloha e o NPD, então os dados serão apresentados carregados.

5.3.3 REQUISITOS FUNCIONAIS

Os requisitos funcionais do Projeto Aloha são:

- Cadastro das entidades de relacionamento entre os usuários do sistema (Turma, Equipe e GPDA).
- Formação dos relacionamentos.
- Gerenciamento das atividades existentes nas entidades de relacionamento entre os usuários do sistema.
    - Entidade Disciplina:
        - Agendamento de provas e trabalhos
        - Enquete
        - Questionários on-line

        - Fórum
        - Calendário de atividades
        - Noticias
        - Eventos
        - Disponibilização de material didático
    - Entidade Equipe:
        - Fórum
        - Compartilhamento de arquivos
    - Entidade GPDA
        - Fórum
        - Compartilhamento de arquivos

5.3.4 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais do Projeto Aloha são:

- Interface Web
- Utilização de Tecnologia Open Source
- Arquitetura MVC

5.3.5 PERFIL DO USUÁRIO

Alunos e professores da Universidade Federal de Santa Catarina, os quais possuem, em sua grande maioria, algum conhecimento em informática e contato com a internet.

5.3.6 OBJETIVOS DE USABILIDADE

Os objetivos de usabilidade são descritos na Tabela 5.2.

Tabela 5.2 - Objetivos de Usabilidade

Objetivo	**Mensuração**
Manter clareza na estrutura	Número de erros repetidos
Possuir telas simples	Observar dificuldades de navegação

5.3.7 MODELO DE TAREFA

Como descrito no item 2.4.1, a função do modelo de tarefa é esclarecer os detalhes de cada tarefa do sistema e oferecer um entendimento da interação do usuário com a interface do sistema. Porém esta função pode ser obtida através da comunicação direta com o cliente e pelo auxilio do protótipo de interface. Este artefato não foi construído para o projeto Aloha, uma vez que sua construção exige um investimento de tempo.

5.3.8 USER STORIES

A Tabela 5.3 apresenta as *UserStories* e suas respectivas estimativas de tempo de desenvolvimento identificadas para o Projeto Aloha.

Tabela 5.3 - *UserStories*

	User Story	**Tempo Estimado Inicial**
US1	Framework – Mensagem	4h
US2	Framework – Taglib – Validação e Formatação	4h
US3	Gerador Classes Domínio	4h
US4	Sincronizador UFSC – CAGR	4h
US5	Tela Padrão Sistema – Facelets	4h
US6	Visualizar Tela Inicial do Professor	4h
US7	Visualizar Turma do Professor	2h
US8	Visualizar Tela Inicial da Turma	4h
US9	Visualizar Trabalhos	2h
US10	Visualizar Trabalho	4h
US11	Incluir Trabalho	4h
US12	Excluir Trabalho	2h
US13	Visualizar e Atribuir Notas de Trabalho	4h
US14	Visualizar Avaliações	2h
US15	Visualizar Avaliação (Professor)	4h
US16	Incluir Avaliação On-line	12h
US17	Incluir Avaliação Presencial	4h
US18	Excluir Avaliação	2h
US19	Visualizar Notas de Avaliação On-line	4h
US20	Corrigir Questões Subjetivas de Avaliação On-line	4h
US21	Ver Resolução da Avaliação On-line de Aluno	4h

US22	Visualizar Notas da Avaliação Presencial dos Alunos	4h
US23	Visualizar Arquivos da Turma	2h
US24	Visualizar Arquivo da Turma	4h
US25	Incluir Arquivo da Turma	4h
US26	Excluir Arquivo da Turma	2h
US27	Visualizar Enquetes da Turma	2h
US28	Visualizar Enquete da Turma	4h
US29	Incluir Enquete da Turma	4h
US30	Excluir Enquete da Turma	2h
US31	Visualizar Resultado da Enquete	4h
US32	Visualizar e Enviar Mensagens da Turma	4h
US33	Visualizar GPDAs	4h
US34	Visualizar Tela Inicial do GPDA	4h
US35	Visualizar e Enviar Mensagens de GPDA	4h
US36	Visualizar Professores de GPDA	2h
US37	Visualizar Tela Inicial do Aluno	4h
US38	Visualizar as Turmas do Aluno	2h
US39	Visualizar a Tela Inicial da Turma	4h
US40	Visualizar Trabalhos da Turma	2h
US41	Visualizar Trabalho da Turma	4h
US42	Enviar Arquivo para Trabalho Individual	4h
US43	Visualizar Tela Inicial da Equipe	4h
US44	Adicionar Participantes na Equipe	4h
US45	Excluir Participantes da Equipe	2h
US46	Visualizar Mensagens da Equipe	4h
US47	Visualizar Materiais Auxiliares da Equipe	2h
US48	Visualizar Material Auxiliar da Equipe	4h
US49	Incluir Material Auxiliar para a Equipe	4h
US50	Excluir Material Auxiliar da Equipe	2h
US51	Visualizar Trabalhos PD da Equipe	2h
US52	Visualizar Trabalho PD da Equipe	4h
US53	Incluir Trabalho PD da Equipe	4h
US54	Excluir Trabalho PD da Equipe	2h
US55	Visualizar Avaliações	2h
US56	Visualizar Avaliação On-line	4h
US57	Responder Avaliação On-line	4h
US58	Visualizar Avaliação Presencial	4h
US59	Visualizar Materiais Didáticos da Turma	4h
US60	Responder Enquete	4h
US61	Visualizar/Atribuir Notas de Trabalho/Download	1h

US62	Visualizar Notas de Avaliação Online- Prova On-line	2h
US63	Infraestrutura – Parte1	104h
US64	Infraestrutura – Parte 2	56h
US65	Integração – CAGR – ALOHA	10h
US66	Integração – Autenticação Usuários	1h
US67	Integrar o Sistema junto com o Framework	80h

5.3.9 PROTÓTIPO DE INTERFACE

O protótipo de interface do projeto Aloha foi construído com baixo investimento de tempo e recurso, conforme pressupõe a metodologia YP. Alguns exemplos destes protótipos podem ser visualizados no Apêndice A.

5.3.10 TESTES DE ACEITAÇÃO

Os testes de aceitação, que têm como objetivo verificar se aquilo que o cliente pediu foi implementado com sucesso, são apresentados no Apêndice B deste trabalho.

5.3.11 PROJETO ARQUITETURAL

A Figura 5.1 representa o projeto arquitetural do projeto Aloha.

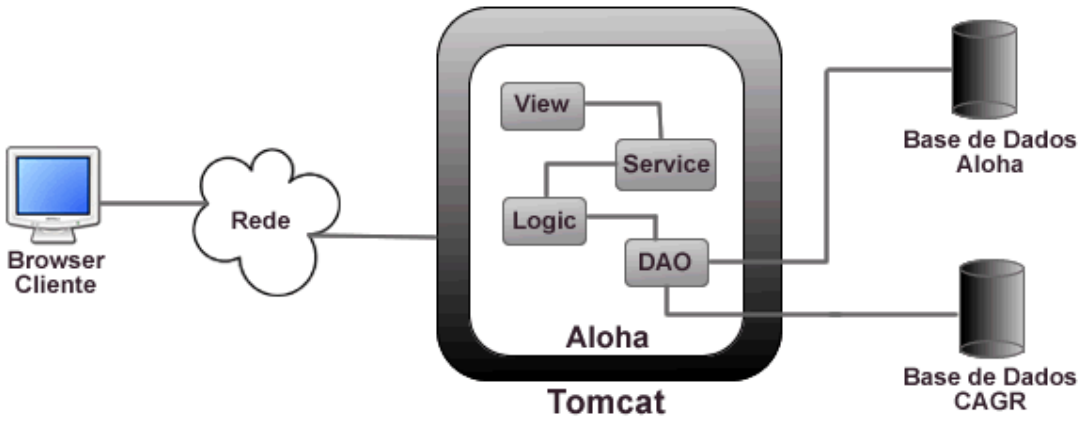


Figura 5.1 - Projeto Arquitetural

5.3.12 MODELAGEM LÓGICA DOS DADOS

Pelo fato do projeto Aloha utilizar o mapeamento objeto relacional, proporcionado pelo Hibernate, ao invés de criar o modelo lógico das tabelas do banco de dados, foi construído o Diagrama de Classes do projeto.

Na Figura 5.2 é apresentado o Diagrama de Classes relativo aos dados de Avaliações, Exercícios e Enquetes.

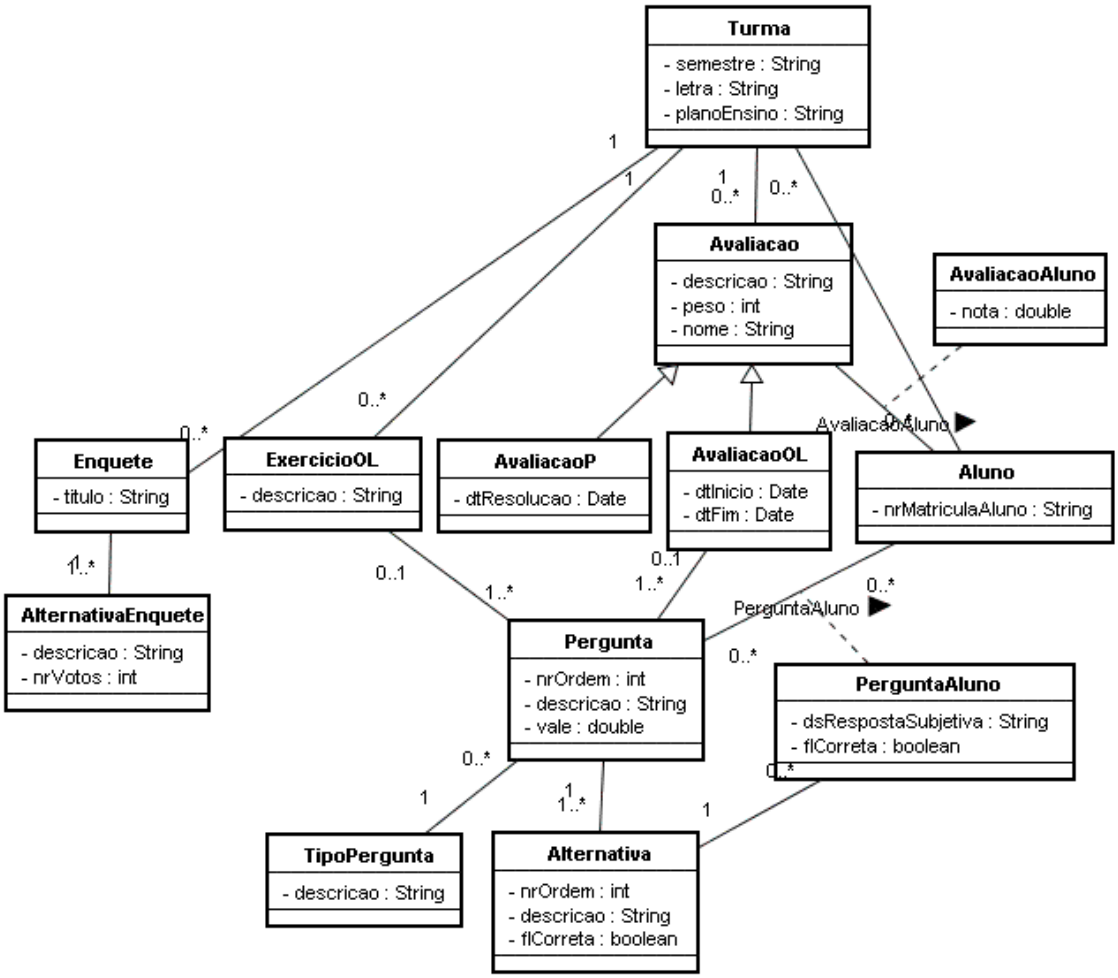


Figura 5.2 - Diagrama de Classes relativo a Avaliações, Exercícios e Enquetes

A Figura 5.3 representa o Diagrama de Classes relativo aos dados dos Trabalhos e Equipe.

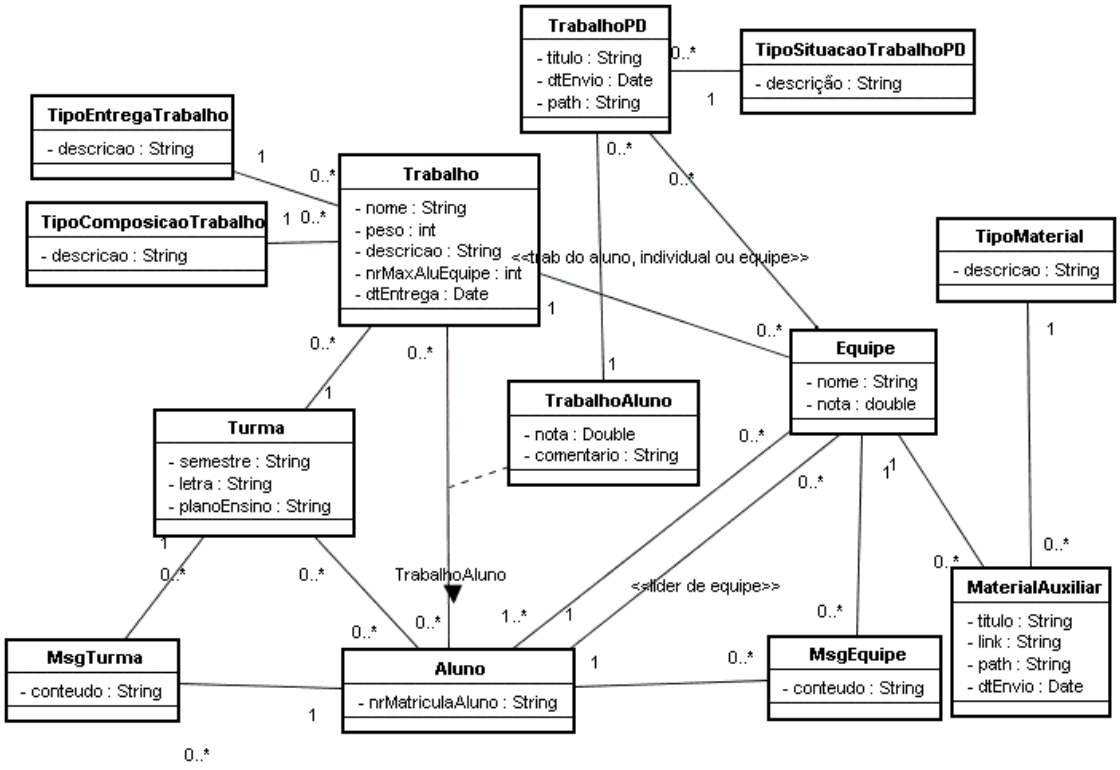


Figura 5.3 - Diagrama de Classes relativo aos Trabalhos e Equipe

Os dados da Turma, dos Eventos, das Notícias e dos GPDAs são apresentados na Figura 5.4.

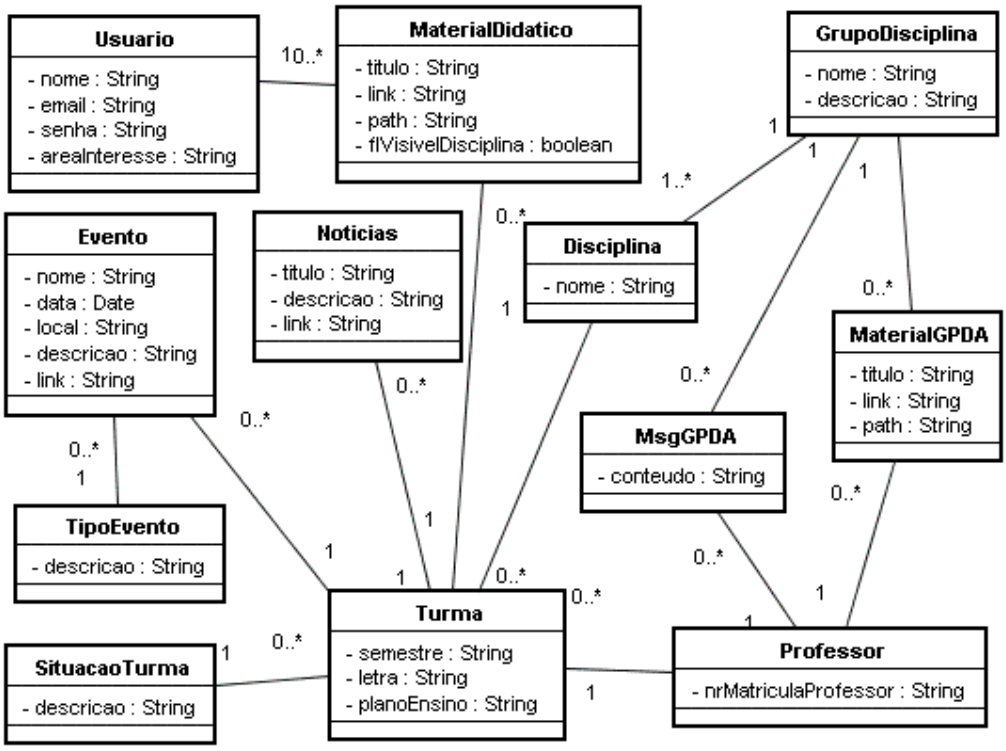


Figura 5.4 - Diagrama de Classes relativo a Turma, Eventos, Notícias e GPDA

Encerrando a Figura 5.5 apresenta como os dados dos usuários foram modelados no Diagrama de Classes.

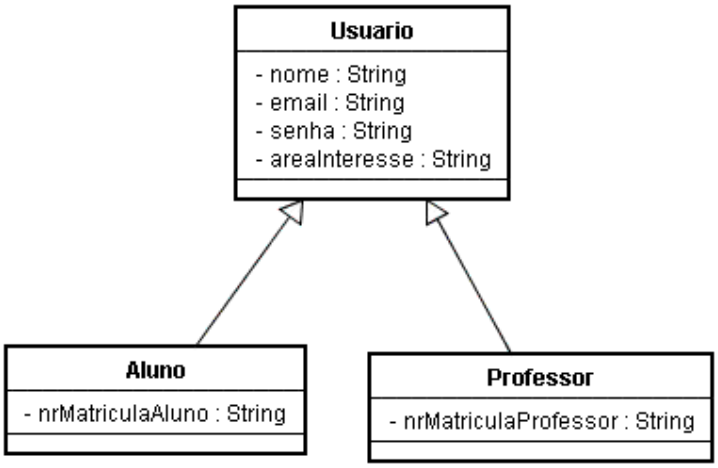


Figura 5.5 - Diagrama de Classes relativo a Usuário

5.3.13 MATRIZ DE COMPETÊNCIA

Foram feitas duas matrizes de competência. Na primeira matriz representada pela Tabela 5.4, verifica-se que os dois membros da equipe estavam participando do desenvolvimento do código do projeto Aloha. Na segunda matriz de competência, representada pela Tabela 5.5, percebe-se que enquanto um desenvolvedor se ausentou do desenvolvimento após adquirir algumas competências, o outro desenvolvedor se dedicou exclusivamente na implementação do mesmo.

Tabela 5.4 - Matriz de Competência de 23/04/2007 a 06/05/2007

Equipe	**Competências**	**Dedicação Semanal**
Adriano	Java, Facelets, Webflow, JPA, Spring, JSF, AJAX4J	12h
Arthur	Java, Spring, JSF	20h

Tabela 5.5 - Matriz de Competência de 07/05/2007 a 04/06/2007

Equipe	**Competências**	**Dedicação Semanal**
Adriano	Java, Facelets, Webflow, JPA, Spring, JSF, AJAX4J	40h
Arthur	Java, Facelets, Webflow, JPA, Spring, JSF	0h

5.3.14 PLANEJAMENTO DE RELEASE

Em virtude do tempo de desenvolvimento o projeto Aloha foi dividido em 2 *releases*. Cada *release* por sua vez foi dividido em duas iterações. O planejamento do primeiro *release* do projeto Aloha é apresentado na Tabela 5.6 e o do segundo na Tabela 5.7.

Tabela 5.6 - Planejamento do Release 1

Iteração	**User Story**	**Período**
Iteração 1	US1	26/4 – 29/4
Iteração 1	US2	25/4 – 26/4
Iteração 1	US3	29/4 – 29/4
Iteração 1	US4	25/4 – 26/4
Iteração 1	US5	23/4 – 25/4
Iteração 1	US6	25/4 – 26/4
Iteração 1	US7	26/4 – 26/4
Iteração 1	US8	26/4 – 29/4
Iteração 1	US9	29/4 – 29/4

Iteração 1	US10	29/4 – 30/4
Iteração 1	US11	30/4 – 03/5
Iteração 1	US12	03/5 – 06/5
Iteração 1	US13	06/5 – 06/5
Iteração 2	US63	07/5 – 20/5

Tabela 5.7 - Planejamento do Release 2

Iteração	**User Story**	**Período**
Iteração 1	US64	21/5 – 29/5
Iteração 2	US67	29/5 – 14/6
Iteração 2	US32	05/6 – 05/6
Iteração 2	US33	06/6 – 06/6
Iteração 2	US34	07/6 – 07/6
Iteração 2	US35	08/6 – 08/6
Iteração 2	US36	11/6 – 11/6
Iteração 2	US37	12/6 – 12/6
Iteração 2	US39	13/6 – 13/6
Iteração 2	US65	14/6 – 14/6
Iteração 2	US66	15/6 – 15/6

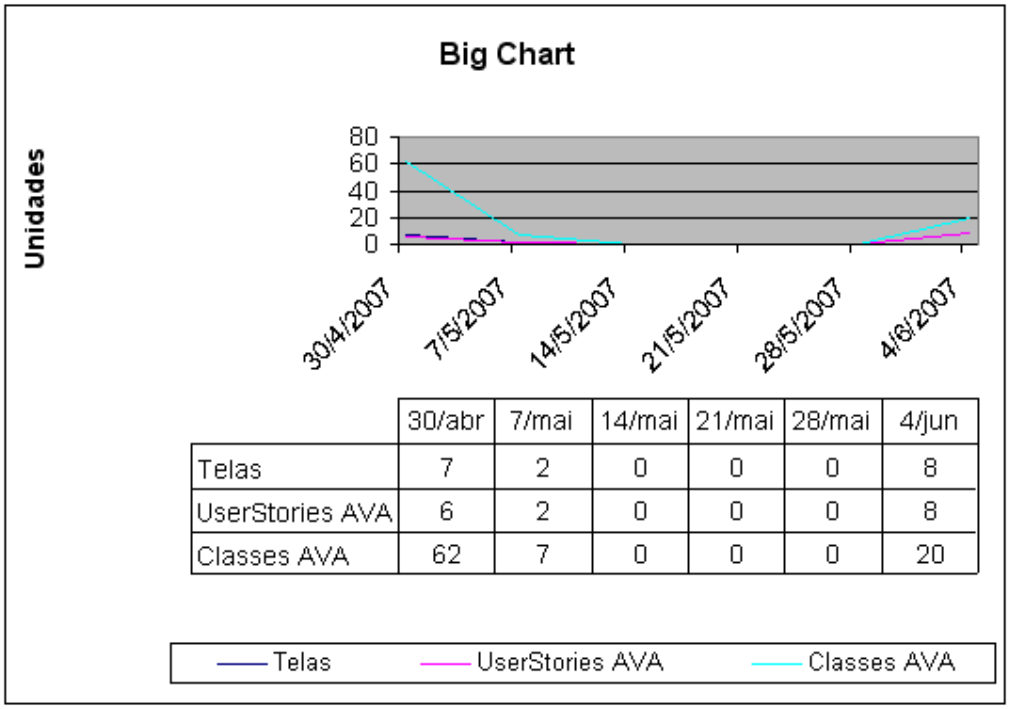
5.3.15 PLANEJAMENTO DE ITERAÇÃO

O planejamento das iterações é apresentado no Apêndice C.

5.3.16 BIG CHART

O gráfico *Big Chart*, explicado no item 2.7.1 deste trabalho, é apresentado na

Figura 5.6.



	30/abr	7/mai	14/mai	21/mai	28/mai	4/jun
Telas	7	2	0	0	0	8
UserStories AVA	6	2	0	0	0	8
Classes AVA	62	7	0	0	0	20

Figura 5.6 – Gráfico Big Chart

5.3.17 ANÁLISE DE RISCO

A tabela de análise de risco, descrita no item 2.7.1, é apresentada na Tabela 5.8.

Tabela 5.8 - Análise de Riscos

Data	Risco	Prio.	Resp.	Status	Providência
23/04	Obter acesso aos dados do CAGR	Alta	Adriano	Superado	Entrar em contato com o pessoal do npd para o desenvolvimento de Views
01/05	Tempo reduzido de desenvolvimento	Alta	Arthur	Vigente	Reduzir o escopo

5.4 IMPLEMENTAÇÃO

A interação entre os usuários do projeto Aloha, descritas no item 5.2, embasaram o desenvolvimento das funcionalidades do projeto, as quais, por sua vez, foram divididas em três ambientes de interação: o ambiente Equipe, o ambiente GPDA e o ambiente Turma.

5.4.1 AMBIENTE EQUIPE

Este ambiente de interação tem como objetivo dar base ao relacionamento existente entre os membros de uma equipe de trabalho. Este relacionamento ocorre entre aluno e aluno e tem como principal finalidade a viabilização de ferramentas de apoio para o sucesso do desenvolvimento do trabalho. As principais funcionalidades presentes neste módulo são:

- Compartilhamento de arquivos auxiliares à confecção do trabalho: esta funcionalidade permite que alunos compartilhem entre si documentos que podem contribuir para o desenvolvimento do trabalho proposto para a equipe. Por exemplo, uma vez que o objetivo do trabalho é fazer o desenvolvimento de um sistema de controle de locadoras em Java, com esta funcionalidade é possível que tutoriais de apoio sejam compartilhados entre os membros da equipe para auxiliar no desenvolvimento do sistema em questão.
- Compartilhamento de arquivos dos trabalhos desenvolvidos pelos alunos: esta funcionalidade permite que o trabalho proposto para equipe seja compartilhado pela equipe. Nesta funcionalidade, seguindo o exemplo proposto na funcionalidade anterior, o arquivo fonte do sistema de controle de locadoras seria compartilhado entre os membros da equipe e quando ele estivesse em condições de ser entregue ao professor, esta funcionalidade o enviaria automaticamente para o professor.
- Troca de mensagens entre os participantes da equipe: esta funcionalidade dispõe de uma área em que os alunos da equipe podem interagir entre si através de mensagens assíncronas.

5.4.2 AMBIENTE GPDA

GPDA é o acrônimo de Grupo de Professores de Disciplinas Afins. Este ambiente de relacionamento busca propiciar um relacionamento entre os diferentes professores de disciplinas que tem alguma relação. Professores de disciplinas afins podem através desde ambiente virtual alinhar as suas metodologias de ensino e

seus respectivos conteúdos. Este tipo de ambiente possibilita também que haja uma troca de conhecimento entre estes professores.

As principais funcionalidades presentes no ambiente GPDA são:

- Troca de mensagens entre os participantes do GPDA: esta funcionalidade dispõe de uma área em que os professores de disciplinas afins podem interagir entre si através de mensagens assíncronas.
- Compartilhamento de arquivos: esta funcionalidade permite que os professores pertencentes a um GPDA compartilhem arquivos. É possível que tais professores compartilhem por exemplo, o plano de ensino de suas disciplinas.

5.4.3 AMBIENTE TURMA

Este ambiente de interação tem como objetivo propiciar a interação entre aluno e professor e entre aluno e aluno de uma turma.

A interação entre aluno e professor ocorre através da execução das atividades inerentes a própria turma e as principais funcionalidades que dão suporte à esta interação são:

- Criação de avaliações on-line: esta funcionalidade permite que professores criem avaliações on-line por meio do Aloha, criando cada pergunta e suas respectivas alternativas, caracterizando-os como certas ou erradas e informando o peso de cada uma.
- Execução de avaliações on-line: o objetivo desta funcionalidade é permitir que o aluno, dentro do prazo, responda a uma avaliação on-line anteriormente criada.
- Cálculo automático do resultado das avaliações on-line: uma vez respondida a avaliação on-line, esta funcionalidade calcula e informa ao aluno o resultado da avaliação. Este *feedback* imediato permite que os alunos tenham rapidamente o conhecimento do seu desempenho e

possam assim tomar as medidas cabíveis para melhorar os seus pontos fracos.

- Controle e disponibilização de informações a respeito das avaliações realizadas em sala de aula: através desta funcionalidade os professores podem informar aos seus alunos a respeito de avaliações que serão realizadas em sala de aula. É permitido ao professor informar a data da prova e o conteúdo da mesma, assim como atribuir as notas de cada aluno pelo sistema, a fim de que esta nota contribua para o cálculo da média final do aluno.
- Controle e disponibilização de informações a respeito dos trabalhos da disciplina: Esta funcionalidade permite que os professores informem à turma sobre a necessidade de realização de um trabalho. O professor informa com esta funcionalidade as características do trabalho que ele deseja, como por exemplo, formato do trabalho, data de entrega, tipo de entrega entre outros. Se o trabalho em questão pode ser realizado em equipes, o ambiente Equipe, mencionado anteriormente, é habilitado para os membros da mesma.
- Escolha de alunos gerentes de equipes: esta funcionalidade é utilizada nos casos em que os trabalhos criados pelo professor, permitem que os trabalhos sejam feitos em equipes, por meio da funcionalidade mencionada anteriormente. Utilizando-se desta funcionalidade o professor escolhe alguns alunos gerentes de equipe, os quais por sua vez serão responsáveis pela formação suas respectivas equipes.
- Visualização de notas: esta funcionalidade permite ao aluno ter conhecimento de todas as suas notas, seja de trabalho ou avaliações. Possibilita também a visualização por parte dos alunos de suas médias finais.
- Disponibilização de material didático: para o embasamento teórico dos alunos, os professores precisam disponibilizar materiais didáticos. Esta funcionalidade propicia tal finalidade. A disponibilização de um material

didático pode ser por meio da indicação de um link na internet ou pelo *upload* de algum arquivo.

- Disponibilização do plano de ensino da disciplina: tal funcionalidade permite que os professores façam o *upload* do arquivo com o plano de ensino da disciplina. Desta forma os gastos com fotocópias e o desconhecimento do plano de ensino da disciplina pelos alunos são eliminados.
- Envio de contribuições pessoais por parte dos alunos para com a disciplina: esta funcionalidade possibilita que alunos contribuam para a troca de conhecimento em uma turma. Os alunos podem fazer *uploads* de arquivos, de sua autoria ou não, que venham a contribuir para a turma, disponibilizando assim mais material didático para a turma. Esta funcionalidade permite que os alunos explicitem o conhecimento ou experiência que tem em determinado assunto e compartilhem o mesmo com a turma.
- Divulgação de eventos relativos ao conteúdo da disciplina: tal funcionalidade permite que professores informem aos alunos sobre a realização de algum evento relativo a disciplina que venham a contribuir para o aprendizado dos alunos.
- Divulgação de noticias relativas ao conteúdo da disciplina: esta funcionalidade serve para que o professor informe os alunos sobre notícias pertinentes a disciplina. Com esta funcionalidade é permitido, por exemplo, que o professor da disciplina de Programação WEB informe organizadamente os seus alunos sobre o surgimento de algum *framework* que merece destaque e que seria de interesse dos alunos.
- Gerenciamento do calendário das atividades da turma: muitas das atividades pertencentes a uma turma, como trabalho, eventos, avaliações e outros, têm uma data associada. A funcionalidade em questão é uma forma de apresentar organizadamente, em forma de calendário, todas as datas importantes que os alunos devem ter conhecimento, a fim de se organizarem melhor.

- Criação de enquetes para turma; em algumas circunstâncias, uma votação ou alguma pergunta mais específica precisa ser feita aos membros da turma. Esta funcionalidade foi criada para prover esta interação entre professor e alunos.
- Troca de mensagens assíncronas entre os alunos e o professor: esta funcionalidade dispõe de uma área em que o professor da turma e os alunos da mesma podem interagir entre si através de mensagens assíncronas.

As funcionalidades que dão suporte para a interação entre aluno e aluno no ambiente Turma são:

- Troca de mensagens assíncronas entre o aluno e aluno: esta funcionalidade dispõe de uma área em que os alunos da turma podem interagir entre si através de mensagens assíncronas.
- Envio de contribuição pessoal por parte dos alunos para com a disciplina: como já mencionado anteriormente, esta funcionalidade tem como objetivo a troca de conhecimento entre os alunos da turma.

Cabe lembrar que o ambiente Equipe é habilitado para aqueles trabalhos em que o professor permite que se formem equipes. Portanto, pode-se dizer que o ambiente Equipe é um sub-ambiente de interação entre aluno e aluno no ambiente Turma.

5.5 CONFIGURAÇÃO DAS TECNOLOGIAS

A infra-estrutura de desenvolvimento faz a configuração para que todas as tecnologias selecionadas possam ser utilizadas de forma harmônica para o desenvolvimento do sistema.

Neste item serão mostradas as configurações necessárias para o funcionamento de cada tecnologia, *framework* e bibliotecas.

5.5.1 CONFIGURAÇÃO DO APACHE TOMCAT

O servidor web, ou *container servlets*, que será utilizado é o Apache Tomcat. Sua estrutura básica de diretórios e arquivo são apresentados na Figura 5.7.

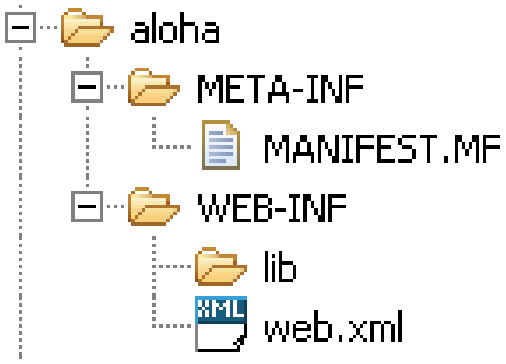


Figura 5.7 - Estrutura de diretórios e arquivos do Tomcat

O arquivo WEB-INF/web.xml contém as configurações do servidor, inclusive as configurações para inicialização de alguns *frameworks* e componentes. A seguir serão apresentadas as configurações realizadas para cada *framework*/tecnologia utilizada. Serão comentadas apenas as configurações que precisaram de uma definição de projeto, pois as outras configurações são padrão.

5.5.2 CONFIGURAÇÃO DO JSF

Foi utilizada a implementação de referência versão 1.2_04P02 do JSF. A configuração do JSF é realizada nos arquivos WEB-INF/web.xml e WEB-INF/faces-config.xml. Além da configuração é necessário copiar os arquivos com a implementação à pasta WEB-INF/lib.

A Figura 5.8 apresenta a configuração adicionada no arquivo web.xml.

```
<context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>server</param-value>
</context-param>
<context-param>
    <param-name>com.sun.faces.validateXml</param-name>
    <param-value>true</param-value>
</context-param>
<context-param>
    <param-name>com.sun.faces.verifyObjects</param-name>
    <param-value>false</param-value>
</context-param>
<context-param>
    <param-name>javax.faces.DEFAULT_SUFFIX</param-name>
    <param-value>.xhtml</param-value>
</context-param>

<listener>
    <listener-class>com.sun.faces.config.ConfigureListener</listener-class>
</listener>

<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.faces</url-pattern>
</servlet-mapping>
```

Figura 5.8 - Configuração do JSF

O JSF manterá o estado da árvore gerada no lado do servidor. O sufixo utilizado para os arquivos que possuem a codificação das interfaces gráficas é **.xhtml*. O mapeamento para acessar o *Faces Servlet* foi definido como "**.faces", ou seja,* para que o sistema acesse o ciclo de vida do JSF é necessário que o endereço colocado no navegador web termine com "*.faces*".

O arquivo WEB-INF/faces-config.xml é o principal arquivo de configuração do JSF. Como apresentado na Figura 5.9 foram definidas as seguintes configurações:

- O Web Flow é responsável pela navegação entre as interfaces gráficas;
- Foram definidos vários *Resolvers* de propriedades e variáveis;
- O local padrão do sistema foi definido como pt_BR;
- Foi adicionado um *listener* de fases como parte da configuração do Web Flow que será abordada mais tarde;

- Através da tag *render-kit* e *renderer* foram definidas novas classes de implementação do *Renderer* para as *tags* *<h:inputText>* e *<h:commandLink>*;
- O projeto do sistema Aloha definiu novos tipos de dados que serão utilizados nas interfaces gráficas. Para cada novo tipo de dados foi criado um conversor através da *tag* *<converter>*.

```
<application>
    <navigation-handler>aloha.framework.view.webflow.FlowNavigationHandler</navigation-handler>
    <variable-resolver>org.springframework.webflow.executor.jsf.DelegatingFlowVariableResolver</variable-r
    <property-resolver>org.springframework.webflow.executor.jsf.FlowPropertyResolver</property-resolver>
    <variable-resolver>org.springframework.webflow.executor.jsf.FlowVariableResolver</variable-resolver>
    <variable-resolver>org.springframework.web.jsf.DelegatingVariableResolver</variable-resolver>
    <variable-resolver>org.springframework.web.jsf.WebApplicationContextVariableResolver</variable-resolve
    <locale-config>
        <default-locale>pt_BR</default-locale>
        <supported-locale>pt_BR</supported-locale>
    </locale-config>
</application>
<lifecycle>
    <phase-listener>org.springframework.webflow.executor.jsf.FlowPhaseListener</phase-listener>
</lifecycle>

<render-kit>
    <renderer>
        <component-family>javax.faces.Input</component-family>
        <renderer-type>javax.faces.Text</renderer-type>
        <renderer-class>aloha.framework.view.jsf.renderer.HtmlInputTextRenderer</renderer-class>
    </renderer>
    <renderer>
        <component-family>javax.faces.Command</component-family>
        <renderer-type>javax.faces.Link</renderer-type>
        <renderer-class>aloha.framework.view.jsf.renderer.HtmlCommandLinkRenderer</renderer-class>
    </renderer>
</render-kit>
<converter>
    <converter-id>aloha.jsf.converter.textConverter</converter-id>
    <converter-class>aloha.framework.view.jsf.converter.TextConverter</converter-class>
</converter>
<converter>
    <converter-id>aloha.jsf.converter.passwordConverter</converter-id>
    <converter-class>aloha.framework.view.jsf.converter.PasswordConverter</converter-class>
</converter>
```

Figura 5.9 - Arquivo WEB-INF/faces-config.xml

5.5.3 CONFIGURAÇÃO DO FACELETS

A configuração do Facelets é realizada no arquivo WEB-INF/web.xml. Além da configuração é necessário copiar os arquivos com a implementação à pasta WEB-INF/lib.

A Figura 5.10 apresenta a configuração realizada para o Facelets.

```
<!-- Facelets Parameters -->
<context-param>
    <param-name>facelets.DEVELOPMENT</param-name>
    <param-value>true</param-value>
</context-param>
<context-param>
    <param-name>facelets.BUFFER_SIZE</param-name>
    <param-value>8192</param-value>
</context-param>
<context-param>
    <param-name>facelets.REFRESH_PERIOD</param-name>
    <param-value>1</param-value>
</context-param>
<context-param>
    <param-name>facelets.SKIP_COMMENTS</param-name>
    <param-value>true</param-value>
</context-param>
<!-- Ajax4Jsf parameters -->
<context-param>
    <param-name>org.ajax4jsf.VIEW_HANDLERS</param-name>
    <param-value>com.sun.facelets.FaceletViewHandler</param-value>
</context-param>
```

Figura 5.10 - Configuração do Facelets

Os principais parâmetros de configuração utilizados são:

- facelets.DEVELOPMENT: se verdadeiro ao ocorrer um erro é apresentado ao programador uma tela com informações relevantes à descoberta do erro. Utilizado como verdadeiro apenas em ambiente de desenvolvimento;
- facelets.SKIP_COMMENTS: se verdadeiro ignora os comentário HTML (Ex: *<!—Comentário -->*) não enviando-os para o cliente;
- org.ajax4jsf.VIEW_HANDLERS: este parâmetro integra a utilização do Facelets com o *framework* Ajax4JSF.

5.5.4 CONFIGURAÇÃO DO AJAX4JSF E RICHFACES

A configuração do Ajax4JSF é realizada no arquivo WEB-INF/web.xml. Além da configuração é necessário copiar os arquivos com a implementação à pasta WEB-INF/lib.

A Figura 5.11 apresenta a configuração do Ajax4JSF.

```
<!-- Filters -->
<filter>
    <display-name>Filter</display-name>
    <filter-name>A4J</filter-name>
    <filter-class>org.ajax4jsf.Filter</filter-class>
    <init-param>
        <param-name>forceparser</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>rewriteid</param-name>
        <param-value>false</param-value>
    </init-param>
    <init-param>
        <param-name>mime-type</param-name>
        <param-value>text/xml</param-value>
    </init-param>
    <init-param>
        <param-name>log4j-init-file</param-name>
        <param-value>WEB-INF/log4j.xml</param-value>
    </init-param>
    <init-param>
        <param-name>enable-cache</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>A4J</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
</filter-mapping>
```

Figura 5.11 - Configuração do Ajax4JSF

A configuração do Ajax4JSF introduz um filtro no qual passarão todas as requisições feitas ao servidor. Assim o *framework* Ajax4JSF adiciona código antes e depois do processamento efetivo da requisição.

Como a biblioteca de componentes RichFaces depende do Ajax4JSF esta configuração também é valida ao RichFaces.

5.5.5 CONFIGURAÇÃO DO TOMAHAWK

A configuração do Tomahawk é realizada no arquivo WEB-INF/web.xml. Além da configuração é necessário copiar o arquivo com a implementação à pasta WEB-INF/lib.

A Figura 5.12 apresenta a configuração.

```
<!-- Tomahawk config -->
<filter>
    <filter-name>MyFacesExtensionsFilter</filter-name>
    <filter-class>org.apache.myfaces.webapp.filter.ExtensionsFilter</filter-class>
    <init-param>
        <param-name>maxFileSize</param-name>
        <param-value>20m</param-value>
    </init-param>
</filter>

<filter-mapping>
    <filter-name>MyFacesExtensionsFilter</filter-name>
    <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>

<filter-mapping>
    <filter-name>MyFacesExtensionsFilter</filter-name>
    <url-pattern>/faces/myFacesExtensionResource/*</url-pattern>
</filter-mapping>
```

Figura 5.12 - Configuração do Tomahawk

Para seu funcionamento o *Tomahawk* necessita apenas da configuração *ExtensionFilter*. O *ExtensionFilter* é responsável por disponibilizar os recursos necessários, como imagens, *css, javascrips*, que as *tags* do *Tomahawk* utilizarão.

5.5.6 CONFIGURAÇÃO DO WEB FLOW

A configuração do Web Flow é realizada nos arquivos WEB-INF/web.xml, WEB-INF/faces-config.xml e arquivo /WEB-INF/webflow-config.xml. Além da configuração é necessário copiar o arquivo com a implementação à pasta WEB-INF/lib.

No arquivo *WEB-INF/web.xml* o *Web Flow* é configurado junto com o *Spring*. As configurações necessárias são apenas o parâmetro *contextConfigLocation* que contém os caminhos dos arquivo de configuração do *Spring* e do *Web Flow*, e a configuração do *ContextLoaderListener* que é executado ao iniciar o servidor, realizando as tarefas de inicialização de contexto do *Spring*. A Figura 5.13 apresenta estas configurações.

```
<!-- Spring and Web Flow parameters -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        classpath:/aloha/framework/applicationContext/applicationContext.xml,
        classpath:/aloha/framework/applicationContext/applicationContext-Engine.xml,
        classpath:/aloha/framework/applicationContext/applicationContext-JSF.xml,
        /WEB-INF/applicationContext.xml,
        /WEB-INF/webflow-config.xml
    </param-value>
</context-param>
<!-- Listners -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

Figura 5.13 - Configuração do Web Flow no arquivo WEB-INF/web.xml

No arquivo WEB-INF/faces-config.xml foram adicionas as configurações necessárias à integração entre o JSF e o Web Flow. As configurações realizadas são apresentadas na Figura 5.14.

```
<faces-config xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
    version="1.2">
    <application>
        <navigation-handler>aloha.framework.view.webflow.FlowNavigationHandler</navigation-handler>
        <variable-resolver>org.springframework.webflow.executor.jsf.DelegatingFlowVariableResolver</variable-resolver>
        <property-resolver>org.springframework.webflow.executor.jsf.FlowPropertyResolver</property-resolver>
        <variable-resolver>org.springframework.webflow.executor.jsf.FlowVariableResolver</variable-resolver>
        <variable-resolver>org.springframework.web.jsf.DelegatingVariableResolver</variable-resolver>
        <variable-resolver>org.springframework.web.jsf.WebApplicationContextVariableResolver</variable-resolver>
        <locale-config>
            <default-locale>pt_BR</default-locale>
            <supported-locale>pt_BR</supported-locale>
        </locale-config>
    </application>
    <lifecycle>
        <phase-listener>org.springframework.webflow.executor.jsf.FlowPhaseListener</phase-listener>
    </lifecycle>
```

Figura 5.14 - Configurações do Web Flow no arquivo WEB-INF/faces-config

As configurações realizadas foram:

- *application/navigation-handler*: foi utilizada uma implementação customizada ao Aloha. Em sua configuração padrão é utilizada a classe *org.springframework.webflow.executor.jsf.FlowNavigationHandler*, mas foi necessária a customização desta classe sendo assim criada a classe *aloha.framework.view.webflow.FlowNavigationHandler*. Na classe customizada foi alterada a forma com que se captura o parâmetro *_eventId*, que tem origem na interface com o usuário e é

utilizado internamente pelo Web Flow para definir ações e fluxo a serem realizados, conforme mostra a Figura 5.16.

```
package aloha.framework.view.webflow;

import javax.faces.application.NavigationHandler;
import javax.faces.context.FacesContext;

import aloha.framework.util.DataTypeUtils;

public class FlowNavigationHandler extends org.springframework.webflow.executor.jsf.FlowNavigationHandler {

    public void handleNavigation(FacesContext facesContext, String fromAction, String outcome,
            NavigationHandler originalNavigationHandler) {
        String eventId = facesContext.getExternalContext().getRequestParameterMap().get("_eventId");
        if (!DataTypeUtils.isEmpty(eventId) && outcome.equals("_eventId")) {
            fromAction = eventId;
            outcome = eventId;
        }
        super.handleNavigation(facesContext, fromAction, outcome, originalNavigationHandler);
    }
}
```

Figura 5.15 - Classe customizada para a configuração *application/navigation-handler*

- *application/variable-resolver e application/property-resolver*: permite que os programadores dos arquivo *.xhtml tenham acesso *container* de *beans* do Spring e aos escopos de variáveis criados pelo Web Flow;
- *lifecycle/ phase-listener*: adiciona um *listener* no ciclo de fases do JSF. Esta configuração permite que o Web Flow execute as tarefas necessárias a medida que o JSF executa seu ciclo de vida.

No arquivo WEB-INF/webflow-config.xml são configurado os componentes essências do Web Flow, como apresentado na Figura 5.16, que são:

- *flowExecutor*: instância a classe que gerenciará os fluxos criados na aplicação;
- flowRegistry: identifica o local onde estão os arquivos de fluxo. Os arquivos de fluxo fazem parte da implementação e não da configuração e serão mostrados mais adiante.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:flow="http://www.springframework.org/schema/webflow-config"
    xsi:schemaLocation="
           http://www.springframework.org/schema/beans
           http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
           http://www.springframework.org/schema/webflow-config
           http://www.springframework.org/schema/webflow-config/spring-webflow-config-1.0.xsd">

    <!-- Launches new flow executions and resumes existing executions -->
    <flow:executor id="flowExecutor" registry-ref="flowRegistry" repository-type="continuation" />

    <!-- Creates the registry of flow definitions for this application -->
    <flow:registry id="flowRegistry">
        <flow:location path="/WEB-INF/flows/**/*.xml" />
    </flow:registry>

</beans>
```

Figura 5.16 - Configuração do Web Flow no arquivo WEB-INF/webflow-config.xml

5.5.7 CONFIGURAÇÃO DO SPRING

A configuração do Spring no servidor web já foi apresentada no item 5.5.6. A configuração do Framework Spring está no arquivo WEB-INF/classes/aloha/*framework*/applicationContext/applicationContext.xml.

O Spring contêm as configurações de muitos componentes de infra-estrutura utilizados em todo o projeto como será mostrado nos itens a seguir.

5.5.7.1 Propriedades de Configuração

Componente que carrega configurações que podem ser acessadas por todo o contexto do Spring. As propriedades podem ser acessadas através da sintaxe *${nomeDaPropriedade}*. Sua configuração é apresentada na Figura 5.17.

```
<bean id="propertyConfigurer" class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="locations">
        <list>
            <value>META-INF/config/mail.properties</value>
            <value>META-INF/config/hibernate.commons.properties</value>
            <!-- MYSQL -->
            <value>META-INF/config/mysql/hibernate.properties</value>
            <value>META-INF/config/mysql/jdbc.properties</value>
            <!-- SQL SERVER -->
            <!--
                <value>META-INF/config/sqlserver/hibernate.properties</value>
                <value>META-INF/config/sqlserver/jdbc.properties</value>
            -->
        </list>
    </property>
</bean>
```

Figura 5.17 – Componente propertyConfigurer do Spring

5.5.7.2 Fonte de dados

Componente que representa a ligação com o banco de dados. Através deste componente é que são enviados os comandos ao *driver* de conexão do banco de dados. Sua configuração é apresentada na Figura 5.18.

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${jdbc.driverClassName}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
</bean>
```

Figura 5.18 - Componente dataSource do Spring

5.5.7.3 JPA

A *Java Persistence API* é configurada através do componente entityManagerFactory. A Figura 5.19 apresenta as principais configurações que são:

- dataSource: identifica a ligação com o banco de dados;
- jpaVendorAdapter: ídentifica o *framework* que implementa a especificação JPA que será utilizada. No Aloha será utilizado o Framework Hibernate;
- a classe PersistenceAnnotationBeanPostProcessor é responsável por identificar as anotações constantes nas classes de domínio implementas com JPA.

```
<bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="persistenceUnitName" value="AlohaFramework" />
    <property name="loadTimeWeaver">
        <bean class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver" />
    </property>
    <property name="jpaVendorAdapter">
        <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">
            <property name="databasePlatform" value="${hibernate.dialect}" />
            <property name="generateDdl" value="${hibernate.generateDdl}" />
            <property name="showSql" value="${hibernate.showSql}" />
        </bean>
    </property>
</bean>
<bean class="org.springframework.orm.jpa.support.PersistenceAnnotationBeanPostProcessor" />
```

Figura 5.19 - Componente JPA do Spring

5.5.7.4 Gerenciador de transação

Componente responsável por gerenciar as transações realizadas com banco de dados. Será utilizada a implementação para JPA do Spring, conforme apresenta a Figura 5.20. Para a instanciação do componente se faz necessária a informação do componente *entityManagerFactory*. A abertura e finalização da transação será realizada pelo Spring através da classe TransactionProxyFactoryBean. A instância desta classe define quando uma transação deve ser iniciada e quais definições devem utilizar, como exemplifica a Figura 5.21. Por padrão será adotado o seguinte:

- Todo método cujo nome inicia com *save, create, update* ou *remove* serão transacionados. Caso ainda não exista uma transação então uma será criada, caso contrário se utilizará a mesma configuração. Toda exceção lançada fará com que o gerenciador de transação desfaça todas as alterações da transação.
- Os demais métodos também serão transacionados, mas somente permitirão leituras.

```
<bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
    <property name="entityManagerFactory" ref="entityManagerFactory" />
</bean>
```

Figura 5.20 - Gerenciador de Transação do Spring

```
<bean abstract="true" id="parentService"
    class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
    <property name="preInterceptors">
        <list>
            <ref bean="serviceInterceptor" />
        </list>
    </property>
    <property name="transactionManager">
        <ref bean="transactionManager" />
    </property>
    <property name="transactionAttributes">
        <props>
            <prop key="save*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="create*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="update*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="remove*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="*">PROPAGATION_REQUIRED, readOnly, -Exception</prop>
        </props>
    </property>
</bean>

<aop:aspectj-autoproxy />

<tx:annotation-driven transaction-manager="transactionManager" />
```

Figura 5.21 - Gerenciador de inicio de transação do Spring

5.5.7.5 Interceptador

Todas as chamadas entre a camada de apresentação e a camada de negócio serão interceptadas pelo componente interceptador configurado no *Spring*. Este componente será responsável pelo tratamento das exceções e registro dos erros em arquivo.

A Figura 5.22 mostra a configuração do interceptador.

```
<bean id="serviceInterceptor"
      class="aloha.framework.layers.service.interceptor.ServiceInterceptor" />
```

Figura 5.22 - Configuração do interceptador

A Figura 5.23 apresenta a classe que implementa o interceptador.

```
package aloha.framework.layers.service.interceptor;

import java.io.FileWriter;

public class ServiceInterceptor implements MethodInterceptor, Serializable {

    public Object invoke(MethodInvocation invocation) throws Throwable {
        try {
            Object back = invocation.proceed();
            return back;
        } catch (LogicException e) {
            printStackTrace(e);
            saveLog(e, invocation);
            throw e;
        } catch (AlohaException e) {
            printStackTrace(e);
            saveLog(e, invocation);
            throw e;
        } catch (Exception e) {
            printStackTrace(e);
            saveLog(e, invocation);
            throw e;
        }
    }

    private void saveLog(Exception e, MethodInvocation invocation) {
        try {
            FileWriter fileWriter = new FileWriter("c:/aloha.log", true);
            PrintWriter log = new PrintWriter(fileWriter);
            log.append(DataTypeUtils.DATE_DMY_HMS.toString(new Date()));
            log.append("-");
            log.append("Method: " + invocation.getMethod());
            log.append("\n\r");
            log.append("Arguments: ");
            log.append(ArrayUtils.toString(invocation.getArguments()));
            log.append("\n\r");
            log.append("StackTrace: ");
            e.printStackTrace(log);
            log.close();
            fileWriter.close();
        } catch (Exception e1) {
            e1.printStackTrace();
        }
    }
```

Figura 5.23 - Implementação do interceptador

5.5.7.6 Internacionalização

Componente que cria a classe *ResourceBundle* a partir de uma lista de arquivos de propriedades com as mensagens internacionalizadas. Para complementar a implementação do Spring foi criada a classe *aloha.framework.message.ResourceBundle*, que permite o acesso a

internacionalização a partir do código dos arquivos *.xhtml. A configuração é apresentada na Figura 5.24.

```
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basenames">
        <list>
            <value>aloha/framework/core/metadata/component/bundle/component</value>
            <value>aloha/framework/core/metadata/domain/bundle/domain</value>
            <value>aloha/framework/core/metadata/project/bundle/project</value>
            <value>META-INF/resources/bundle/alohaFramework</value>
        </list>
    </property>
</bean>

<bean id="msg" class="aloha.framework.message.ResourceBundle" init-method="utils" />
```

Figura 5.24 - Internacionalização do *Spring*

5.6 INFRA-ESTRUTURA DE DESENVOLVIMENTO

A infra-estrutura do Aloha consiste na implementação de componentes que serão utilizados no desenvolvimento do caso de uso desde a interface gráfica até a camada de integração.

Os itens a seguir explicarão os principais componentes da infra-estrutura do Aloha.

5.6.1 TIPOS DE DADOS

Foram pré-definidos 16 tipos customizados de tipo de dado que serão utilizados no projeto. São eles:

- Data no formato dia/mês/ano;
- Data no formato mês/ano;
- Data no formato dia/mês/ano hora:minuto:segundo;
- Data no formato dia/mês/ano hora:minuto;
- Hora no formato hora:minuto:segundo;
- Hora no formato hora:minuto;
- Decimal com duas casas;
- Decimal com quatro casas;

- Número inteiro de 8 bits com sinal (BYTE);
- Número inteiro de 16 bits com sinal (SHORT);
- Número inteiro de 32 bits com sinal (INTEGER);
- Número inteiro de 64 bits com sinal (LONG);
- Texto do tipo senha;
- Texto;
- Flag. Permite os seguintes valores: verdadeiro ou falso;
- Arquivo.

Para cada um dos tipos customizados foi criado um conversor. Este conversor pode transformar um texto no tipo customizado, ou a partir do tipo transformar em texto.

Estas transformações serão muito utilizadas na interface gráfica.

5.6.2 TAGLIBS FACELETS

Foram codificadas taglibs com componentes visuais e taglibs com funções que podem ser acessadas através da codificação dos arquivos *.xhtml.

Para cada tipo de dado customizado foi criada uma taglib de entrada e uma de saída. Esta taglib implementa as restrições do tipo de dado para que force a entrada e saída no formato e valor correto da informação. Por exemplo, o tipo de dados dia/mês/ano permite apenas a entrada de caracteres numéricos e barras. A taglib realiza a validação da informação, no caso de entrada. Assim o valor de entrada 31/09/2007 será invalidado pela taglib.

A Figura 5.25 apresenta as taglibs de entrada e saída utilizadas no Aloha.

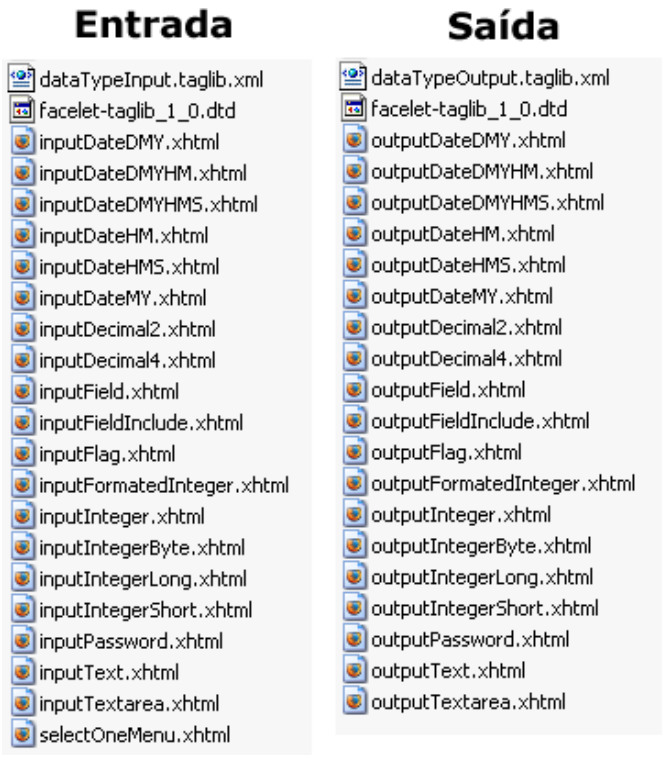


Figura 5.25 - *Taglibs* de entrada e saída

5.6.3 DOMÍNIO SIMPLES

Foram considerados como domínio simples todas as classes de domínio que possuem apenas as informações de nome e código, como por exemplo, as situações de uma turma.

Todas as classes de domínio simples estenderão a classe *aloha.framework.core.jpa.DescriptionDomain.DescriptionDomain*. A classe *DescriptionDomain* é apresentada na Figura 5.26.

```
package aloha.framework.core.jpa;

import aloha.framework.core.jpa.impl.DescriptionDomainImpl;

@javax.persistence.Entity
@javax.persistence.Inheritance(strategy = javax.persistence.InheritanceType.SINGLE_TABLE)
public abstract class DescriptionDomain extends DescriptionDomainImpl {

    public static final String CLASS_LIVE_CODE = null;

    @javax.persistence.Id
    @javax.persistence.GeneratedValue(strategy = javax.persistence.GenerationType.AUTO)
    private java.lang.Integer idDescriptionDomain;

    @javax.persistence.Basic(optional = false)
    @javax.persistence.Column(nullable = false, insertable = true, updatable = true, length = 50)
    private java.lang.String friendlyName;

    @javax.persistence.Basic(optional = false)
    @javax.persistence.Column(nullable = false, insertable = true, updatable = false)
    private java.lang.Byte code;
```

Figura 5.26 - Classe *DescriptionDomain*

As classes que estendem a classe *DescriptionDomain* podem apenas adicionar as constantes com os códigos de cada registro, se necessário, como exemplificado na Figura 5.27.

```
package aloha.domain;

import aloha.framework.core.jpa.DescriptionDomain;

@javax.persistence.Entity
public  class LearningClassState extends DescriptionDomain {

    public static final java.lang.Byte IN_PROGRESS = new java.lang.Byte("1");
    public static final java.lang.Byte FINISHED = new java.lang.Byte("2");

    public Boolean isInProgress() {
        return IN_PROGRESS.equals(getCode());
    }
    public Boolean isFinished() {
        return FINISHED.equals(getCode());
    }

}
```

Figura 5.27 - Exemplo de utilização da classe *DescriptionDomain*

Os dados de todas as classes que estendem a classe *DescriptionDomain* serão armazenadas em apenas uma tabela no banco de dados. A Figura 5.28 mostra como ficam armazenados os dados.

DTYPE	idDescriptionDomain	version	code	friendlyName
QuestionType	46	0	1	Subjective
CompositionTypeWork	45	0	2	Single
CompositionTypeWork	44	0	1	Team
LearningClassState	43	0	2	Finalizada
LearningClassState	42	0	1	Em andamento

Figura 5.28 - Forma de armazenamento dos dados na tabela *DescriptionDomain*

5.6.4 DOMÍNIO ABSTRATO

Todas as classes de domínio estenderão a classe abstrata *aloha.framework.core.jpa.AbstractDomain*. Isto permitira o uso de polimorfismo nas classes de domínio alem da possibilidade da implementação de métodos auxiliares que poderão ser utilizados por todas as classes de domínio.

5.6.5 TRATAMENTO DE EXCEÇÃO

Todas as exceções lançadas pelo Aloha terão que estender ou ser uma instancia da classe *aloha.framework.exception.AlohaException*.

A classe *AlohaException* permite a inserção de várias mensagens de exceção.

5.6.6 ACESSO A CAMADA DE INTERGRAÇÃO

Para acesso a camada de integração foi utilizado o padrão de projeto *ActiveRecordDAO*. Todo acesso às fontes de dados são feitas por uma interface chamada *ActiveRecordDAO*. Com este padrão caso seja alterada a tecnologia de ORM ou banco de dados, será necessário fazer somente uma nova implementação da interface *ActiveRecordDAO* para a nova tecnologia.

Para o projeto Aloha foi implementada a classe *aloha.framework.integration.impl.ActiveRecordJPA* devido ao uso da *Java Persistence API*.

A classe *ActiveRecordJPA* estende a classe *org.springframework.orm.jpa.support.JpaDaoSupport* do *Spring*, que possui métodos de acesso já implementados.

5.6.7 SUPER CLASSES DAS CAMADAS

As interfaces da camada de serviço dos casos de uso do Aloha estenderão a interface *aloha.framework.layers.service.Service* permitindo assim identificar através de polimorfismo todas as interfaces da camada de serviço.

As interfaces da camada lógica de negócio estenderão a interface *aloha.framework.layers.logic.Logic*. As classes que implementam a lógica de negócio estenderão a classe *aloha.framework.layers.logic.impl.LogicImpl*. A classe *LogicImpl* possui como atributo o objeto *activeRecordDAO* que fornecerá acesso a camada de integração à classe de lógica de negócio conforme a Figura 5.29.

```
package aloha.framework.layers.logic.impl;

import aloha.framework.integration.ActiveRecordDAO;

public abstract class LogicImpl implements Logic {

    protected ActiveRecordDAO activeRecordDAO;

    public void setActiveRecordDAO(ActiveRecordDAO activeRecordDAO) {
        this.activeRecordDAO = activeRecordDAO;
    }

}
```

Figura 5.29 - Classe *LogicImpl*

5.6.8 OBJECT MAP

As classes que implementam a interface *java.util.Map* são muito utilizadas em todos os sistemas. No Aloha foi criada a classe *aloha.framework.map.AlohaMap*. A AlohaMap estende a classe *java.util.LinkedHashMap*. A customização realizada foi a inclusão de métodos que obtém um objeto da Map realizando a tipagem do

mesmo. Por exemplo, a classe *AlohaMap* possui o seguinte método: *public Date getDate(Object key)*.

5.6.9 MENSAGENS

Existem três níveis de mensagens no sistema: sucesso, alerta e erro. No lançamento de uma exceção é definida a mensagem desta exceção e o seu nível.

A partir do nível da exceção a apresentação da mensagem ao usuário pode ser personalizada, por exemplo, colocando as mensagens de erro em vermelho, as mensagens de alerta em amarelo e as mensagens de sucesso em verde.

Outra funcionalidade para o nível de exceção é no registro de erros no interceptador. Pode-se escolher por registrar o erro apenas de mensagens do nível erro, por exemplo. A Figura 5.30 mostra o lançamento de uma exceção com uma mensagem de erro.

```
public void metodo() throws Exception {
    AlohaException alohaException = new AlohaException();
    alohaException.addErrorMessage("message", "parameters");
    throw alohaException;
}
```

Figura 5.30 - Exceção com mensagem

5.6.10 ACESSO A ARQUIVOS DE RECURSO PELO NAVEGADOR

O acesso aos arquivos como imagens, estilos, scripts, serão realizados através de um *servlet* pela interface gráfica.

A Figura 5.31 mostra a configuração do *servlet resources* no arquivo WEB-INF/web.xml.

```
<servlet>
    <servlet-name>resources</servlet-name>
    <servlet-class>aloha.framework.view.resources.Resources</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>resources</servlet-name>
    <url-pattern>/resources/*</url-pattern>
</servlet-mapping>
```

Figura 5.31 - Configuração do *servlet resources*

A Figura 5.32 apresenta o exemplo de uso do *servlet resources* pelo arquivo *.xhtml.

```
<link rel="stylesheet" type="text/css"
    href="${contextPath}/resources/aloha/framework/view/resources/css/alohaFramework.css" />
<link rel="stylesheet" type="text/css"
    href="${contextPath}/resources/aloha/framework/view/resources/css/external.css" />
<link rel="stylesheet" type="text/css"
    href="${contextPath}/resources/aloha/framework/view/resources/css/menu.css" />
<script type="text/javascript"
    src="${contextPath}/resources/aloha/framework/view/resources/js/prototype-1.5.0.js">
</script>
<script type="text/javascript"
    src="${contextPath}/resources/aloha/framework/view/resources/js/scriptaculous/scriptaculous.js">
</script>
<script type="text/javascript"
    src="${contextPath}/resources/aloha/framework/view/resources/js/alohaFramework.js">
</script>
```

Figura 5.32 - Uso do *servlet resources*

5.6.11 DEFINIÇÃO DAS CONFIGURAÇÕES INICIAIS

Assim como os *frameworks* utilizados o Aloha tem uma classe responsável por inicializar suas configurações. Esta responsabilidade é da classe *aloha.framework.view.servlet.ApplicationInit*. No momento esta classe inicializa apenas a localização padrão do sistema, que é Brasil, e armazena o contexto da aplicação para armazenar informações.

Esta classe é executada no momento que o Apache Tomcat é inicializado. Sua configuração é realizada no arquivo WEB-INF/web.xml. A Figura 5.33 apresenta a configuração.

```
<listener>
    <listener-class>aloha.framework.view.servlet.ApplicationInit</listener-class>
</listener>
```

Figura 5.33 - Configuração do *ApplicationInit*

6 CONCLUSÕES

Muitas são as vantagens da utilização de uma Ambiente Virtual de Aprendizado em uma instituição de ensino, como citado no capítulo 3 deste trabalho. Porém, um estudo sobre os diversos tipos de AVAs se faz necessário com o intuito de identificar aquele que melhor se adéqüe às atividades da instituição. Em muitas das vezes é mais vantajoso desenvolver um Ambiente Virtual de Aprendizado especialmente para a instituição, como ocorreu com 38% das instituições de ensino do Reino Unido em 2005, conforme Tabela 4.

Um Ambiente Virtual de Aprendizado desenvolvido com foco nas atividades e cotidiano de uma instituição de ensino, em específico, possibilita que o mesmo seja mais coerente e fiel às necessidades da instituição em questão. Esta vantagem é obtida pelo fato de que o processo de desenvolvimento do AVA é realizado inteiramente com base nos dados e necessidades daquela instituição. Além disso, a integração com o MIS e LibMS da instituição é facilitada, pois nenhuma transformação de dados é necessária, visto que o modelo dos dados é semelhante.

O projeto Aloha, desenvolvido especialmente para a Universidade Federal de Santa Catarina, apresenta alguns diferenciais em relação aos demais Ambientes Virtuais de Aprendizado. O projeto Aloha dispõe de um ambiente exclusivo para equipes de alunos que precisam desenvolver um trabalho acadêmico, provendo funcionalidades de compartilhamento de arquivo e comunicação. Além de dar base para as interações entre aluno e aluno, o projeto Aloha dispõe de um ambiente totalmente projetado para as interações entre professor e professor, com a finalidade de prover aos mesmos uma ferramenta que auxilia no melhoramento de técnicas de ensino e troca de conhecimentos e experiência.

Com o intuito de prover a integração entre o Aloha e o MIS da UFSC, representado pelo sistema CAGR, foi planejado um processo de sincronização que busca os dados do CAGR e os atualiza na base de dados do Aloha. Portanto a entrada dos dados pessoais dos alunos e professores é feita automaticamente,

assim como a configuração adequada do Ambiente Virtual de Aprendizado para cada aluno e professor, com base nas suas disciplinas cursadas atualmente.

Outro aspecto interessante no projeto Aloha foi a escolha da tecnologia. Para o desenvolvimento do projeto Aloha somente foram utilizadas tecnologias *open-source* e a linguagem de programação Java. Portanto, a contribuição externa para o projeto é facilitada e bem-vinda. Uma vez que a linguagem de programação utilizada foi o Java, é totalmente viável a utilização do Aloha para o ensino de disciplinas relativas ao desenvolvimento de software utilizando a linguagem em questão, possibilitando assim que o projeto Aloha cresça e evolua com base nas necessidades da UFSC.

Em relação ao *EasyProcess*, a metodologia de desenvolvimento de software utilizada para o desenvolvimento do Aloha, observou-se que a mesma é uma metodologia bastante nova e que por isso dispõe de poucas referências bibliográficas, fato este que restringe o aprendizado e dificulta de resolução de dúvidas referentes a metodologia. Porém a metodologia é bastante simples e eficiente para o desenvolvimento de projetos acadêmicos ou de pequeno e médio porte, pois projetos desta dimensão apresentam poucas regras de negócio e a equipe de desenvolvimento geralmente é pequena.

Uma fator positivo presente no *EasyProcess* é o rápido processo de especificação do sistema, com pouca documentação, adiantando o tempo para codificação. A utilização de *UserStories* juntamente com o auxilio do protótipo de interface e o contato com o cliente foram suficientes para entendimento da tarefa e embasaram o desenvolvimento.

Portanto, percebeu-se que além da utilização dos Ambientes Virtuais de Aprendizado estar em ascensão, o desenvolvimento focado para uma instituição apresentada-se como uma alternativa de maior ajustamento às necessidades. Notou-se também que a utilização da metodologia de desenvolvimento *EasyProcess* ofereceu o suporte necessário para o desenvolvimento do sistema.

6.1 RECOMENDAÇÕES PARA TRABALHOS FUTUROS

Sugere-se como trabalho futuro a continuação deste trabalho por meio da identificação de novas funcionalidades que possam contribuir para o sucesso do processo de ensino e aprendizagem na Universidade Federal de Santa Catarina. Como exemplo de novas funcionalidades cita-se a criação de salas de *chat* para dar suporte à comunicação síncrona, o suporte a vídeo conferências e o envio de mensagens das novas atualizações de conteúdo no Aloha para os usuários por meio de mensagens no formato *Feed RSS*. A visualização das notas, eventos e notícias, a participação em enquetes e a resolução de avaliações on-line são funcionalidades que poderiam ser realizadas via celular. Outra sugestão seria a criação de um ambiente de relacionamento dentro do projeto Aloha que possibilitasse a interação entre todos os alunos da universidade, não se restringindo a equipes ou turmas.

Além do desenvolvimento de novas funcionalidades, seria interessante uma maior integração entre o AVA Aloha, o MIS e o LibMS da UFSC. Portanto um trabalho futuro seria tornar o Ambiente Virtual de Aprendizado o mais próximo possível de ser um MLE, como descrito no item 3.4.4.

7 REFERÊNCIAS

ALLEN, Paul; BAMBARA, Joseph. J2EE: **Guia Oficial de Certificação**. Trad. Edson Furmankiewicz - Rio de Janeiro: Campus, 2003. ISBN 85-352-1300-7.

(APACHE MYFACES: Tomahawk, 2007)
APACHE MYFACES. **MyFaces Tomahawk**. Versão 1.5.0. Disponível em http://myfaces.apache.org/tomahawk/index.html. Acessado em 21/05/2007.

(APACHE: Tomcat, 2007)
APACHE. **Tomcat**. Disponível em http://tomcat.apache.org/. Acessado em 25/05/2007.

CATLEY, Paul. **Enhancing Learning through Online Assessment**. Disponível em: <http://www.jisc.ac.uk/uploaded_documents/oxfordv2.doc>. Acessado em 4 jan 2007.

CHASEN, Michael. **Letter to Clients**. 2005. Disponível em: <http://library.blackboard.com/docs/company/Letter_to_Clients.pdf>. Acessado em 21 mai 2007.

DEITEL, H. M.; DEITEL, P. J. **Java: como programar**. 4. ed. Porto Alegre: Bookman, 2003. ISBN 85-363-0123-6.

DIAS, Renato Mendes. **Especificações de características de ambientes de "e-learning"**. 2002. Tese (Mestrado em Engenharia de Produção). Curso de Pós-graduação em Engenharia de Produção. Universidade Federal de Santa Catarina, Florianópolis, 2002.

DONALD, Keith. **Spring Web Flow: Reference Documentation**. Versão 1.0.3. 2007. Disponível em http://static.springframework.org/spring-webflow/docs/current/reference/index.html. Acessado em 25/05/2007.

FERREIRA, Aurélio Buarque de Holanda. **Dicionário da Língua Portuguesa**. 2002. Disponível em: <http://www2.uol.com.br/aurelio/>.

FRANCO, Marcelo Araújo, CORDEIRO, Luciana Meneghel;CASTILLO, Renata A. Fonseca de. **O ambiente virtual de aprendizagem e sua incorporação na Unicamp**. 2003. Disponível em: <http://www.scielo.br/pdf/ep/v29n2/a11v29n2.pdf>. Acessado em 2 jan 2007.

GARCIA, Francilene Procópio et al. 2007. **easyProcess – Processo de desenvolvimento de software para o DSC**. Disponível em http://www.dsc.ufcg.edu.br/~yp/. Acessado em 3 jan 2007.

GARCIA, Francilene Procópio et al. **easyProcess - Um Processo de Desenvolvimento de Software.** 2004. Disponível em: http://repositorio.viadigital.org.br/docman/view.php/7/7/Metodologia_de_**Desenvolvimento**.pdf. Acessado em 4 jan 2007.

GARCIA, Francilene Procópio et al. **easYProcess: Um Processo de Desenvolvimento para Uso no Ambiente Acadêmico**. 2004. Disponível em: <http://www.gustavowagner.com/paginaPessoal/documentos/artigos/YPsbc2004-GustavoWagnerMendes.pdf>. Acessado em 3 jan 2007.

GIL, Antônio Carlos. **Como Elaborar Projetos de Pesquisa**. 3. ed. São Paulo: Atlas, 1991.
HOOKOM, Jacob. **Facelets - JavaServer Faces View Definition Framework, Developer Documentation**. 2006. Disponível em https://facelets.dev.java.net/nonav/docs/dev/docbook.html. Acessado em 21/01/2007.

HUNT, Malcolm; PARSONS, Dave; FLEMING, Andrew. **A REVIEW OF THE RESEARCH LITERATURE ON THE USE OF MANAGED LEARNING ENVIRONMENTS AND VIRTUAL LEARNING ENVIRONMENTS IN EDUCATION, AND A CONSIDERATION OF THE IMPLICATIONS FOR SCHOOLS IN THE UNITED KINGDOM**. Disponível em: < www.becta.org.uk/page_documents/research/VLE_report.pdf >. Acessado em 20 mai 2007.

JARVIS, Janis et al. **A Cross-Cluster Qualitative Study from the External Evaluation**. 2005. Disponível em: <http://www.evaluation.icttestbed.org.uk/files/mis_systems_report.pdf>. Acessado em 4 jan 2007.

(JCP2, 2007)
JAVA COMMUNITY PROCESS . **JCP 2: Process Document**. Disponível em http://www.jcp.org/en/procedures/jcp2. Acessado em 21/01/2007.

(JCP: SERVLET, 2007)
JAVA COMMUNITY PROCESS. **JSR 154: JavaTM Servlet 2.4 Specification**. Disponível em http://jcp.org/en/jsr/detail?id=154. Acessado em 21/01/2007.

(JCP: JSP, 2007)
JAVA COMMUNITY PROCESS. **JSR 245: JavaServerTM Pages 2.1**. Disponível em http://jcp.org/en/jsr/detail?id=245. Acessado em 21/01/2007.

(JCP: JSF, 2007)
JAVA COMMUNITY PROCESS. **JSR 252: JavaServer Faces 1.2**. Disponível em http://jcp.org/en/jsr/detail?id=252. Acessado em 21/01/2007.

(JAVA.NET: JSF Sandbox, 2007)
JAVA.NET. **GlassFish, JavaServer Faces RI, Sandbox**. Versão 0.1. Disponível em https://javaserverfaces.dev.java.net/sandbox. Acessado em 21/05/2007.

(JBOSS: Ajax4jsf, 2007)
JBOSS. **Ajax4jsf**. Versão 1.1.0. Disponível em http://labs.jboss.com/jbossajax4jsf/. Acessado em 21/05/2007.

(JBOSS: RichFaces, 2007)
JBOSS. **RichFaces**. Versão 3.0.0. Disponível em http://labs.jboss.com/jbossrichfaces/. Acessado em 21/05/2007.

JENKINS Martin;,BROWNE Tom; WALKER Richard. **VLE Surveys. A longitudinal perspective between March 2001, March 2003 and March 2005 for higher education in the United Kingdom**. 2005. Disponível em : <http://www.ucisa.ac.uk/groups/tlig/vle/vle_survey_2005.pdf>. Acessado em 4 jan 2007.

JISC. **Managed Learning Environments: A Workshop run by JISC Assist, 29 February and 7 March 2000. Final Report. JISC**. 2000. Disponível em: <http://www.jisc.ac.uk/index.cfm?name=event_report_mle>. Acessado em 4 nov 2006.

JOHNSON, Rod. **Spring Frameowrk: The Spring Framework - Reference Documentation**. Versão 2.0.5. 2007. http://static.springframework.org/spring/docs/2.0.x/reference/index.html. Acessado em 25/05/2007.

LARMAN, Craig. **Utilizando UML e Padrões**: uma Introdução à análise e ao Projeto Orientado a Objeto. Trad. Luiz A Meirelles Salgado - Porto Alegre: Bookman, 2000. ISBN 85-730-651-8.

MEDEIROS, F. P. **Projeto e implementação do módulo TAOS-Graph da ferramenta iTAOS para análise e modelagem da tarefa**. 2003. Dissertação (Mestrado) – Universidade Federal de Campina Grande, Campina Grande, Paraíba, Fevereiro / 2003.

MORGAN, Glenda. **Faculty Use of Course Management System**. 2003. Disponível em: <http://www.educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf> Acessado em 4 jan 2007.

PEDROSO, Deucélia Eva. **Interfaces Gráficas em Ambientes de E-learning: Caso VIASK**. 2002. Tese (Mestrado em Engenharia de Produção). Curso de Pós-graduação em Engenharia de Produção. Universidade Federal de Santa Catarina, Florianópolis, 2002

ROSENBERG, Marc J. **E-Learning**. 1. ed. São Paulo: Makron Books, 2002.

SILVA, Edna Lúcia da; MENEZES, Estera Muszkat. **Metodologia da Pesquisa e Elaboração de Dissertação**. 2001. Disponível em: <http://projetos.inf.ufsc.br/arquivos/Metodologia da Pesquisa 3a edicao.pdf>. Acessado em: 16 out 2006.

SILVA, Valdete Teixeira. **Avaliação de um Ambiente Virtual de Aprendizagem: Análise de Ferramentas**. 2003. Tese (Doutorado em Engenharia de Produção). Curso de Pós-graduação em Engenharia de Produção. Universidade Federal de Santa Catarina, Florianópolis, 2003

SULEMAN, Shakeel**. Use of Communications Tools within VLEs**. 2003. Disponível: <http://ferl.becta.org.uk/display.cfm?resID=5497>. Acessado em 27 dec 2006.

VASCONCELOS, Cesar Rocha. **XPU – Um Modelo Para o Desenvolvimento de Sistemas Centrado no Usuário**. 2004. Dissertação (Mestrado) – Universidade Federal de Campina Grande, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande, Paraíba, Fevereiro / 2004.

(W3C: CSS, 2007)
W3C. **Cascading Style Sheets home page**. Disponível em http://www.w3.org/Style/CSS/. Acessado em 28/05/2007.

(W3C: XHTML, 2007)
W3C. **XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)**. Disponível em http://www.w3.org/TR/2002/REC-xhtml1-20020801/. Acessado em 28/05/2007.

(WIKIPEDIA: AJAX, 2007)
WIKIPEDIA. **AJAX (programação)**. 2007. Última atualização: 29/05/2007 19:56 por Luís Felipe Braga . Disponível em

http://pt.wikipedia.org/wiki/AJAX_%28programa%C3%A7%C3%A3o%29. Acessado em 01/06/2007.

(WIKIPEDIA: DHTML, 2007)
WIKIPEDIA. **DHTML**. 2007. Disponível em http://pt.wikipedia.org/wiki/DHTML. Acessado em 01/06/2007.

(WIKIPEDIA: MODELO RELACIONAL, 2007)
WIKIPEDIA. **MODELO RELACIONAL**. 2007. Disponível em http://pt.wikipedia.org/wiki/Modelo_Relacional. Acessado em 01/06/2007.

(WIKIPEDIA:OSI, 2007)
WIKIPEDIA. **Modelo OSI**. Última atualização: 08/02/2007 09:39 por Epinheiro. Disponível em http://pt.wikipedia.org/wiki/Modelo_OSI. Acessado em 21/02/2007.

WILSON, Scott. **Interoperability: Why and how**. 2002. Disponível em: < www.jisc.ac.uk/uploaded_documents/Interoperability%20(Scott%20Wilson).ppt>. Acessado em 21 mai 2007.

8 APÊNDICE A – PROTÓTIPO DE INTERFACE

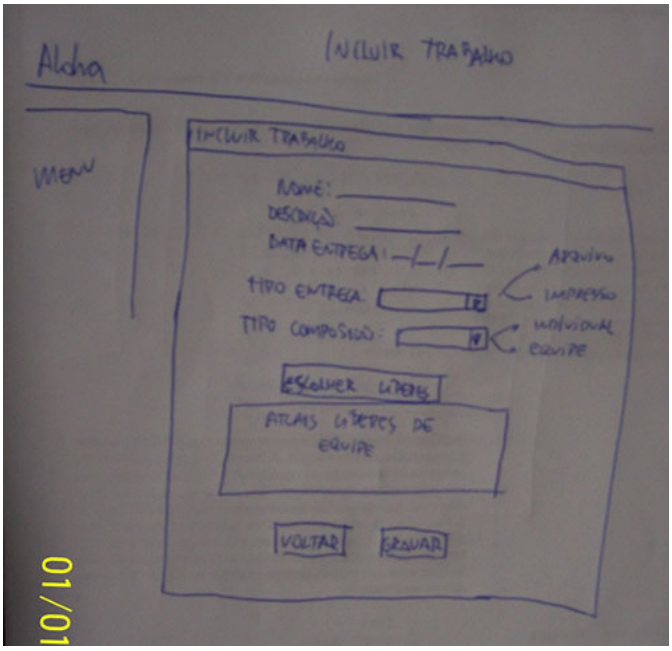


Figura 8.1 - Protótipo de Tela de Incluir Trabalho

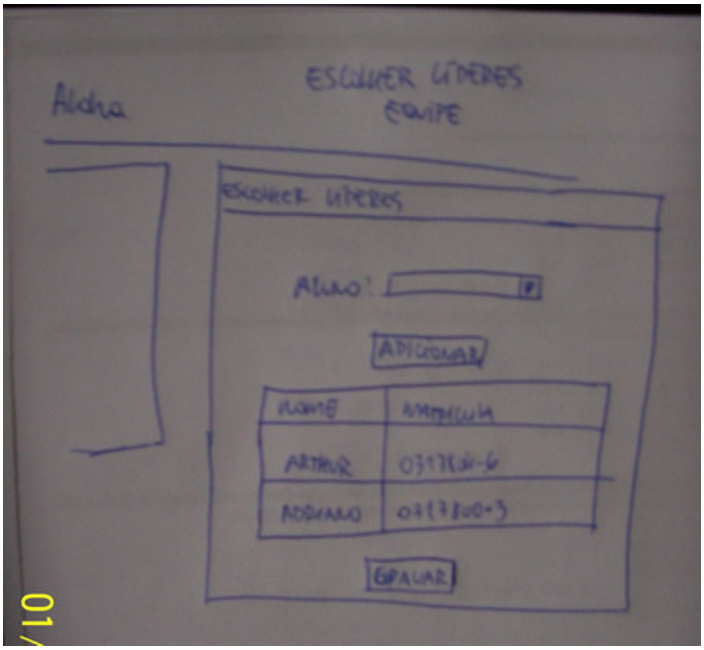


Figura 8.2 - Prototipo de Tela de Escolher Lider de Equipe

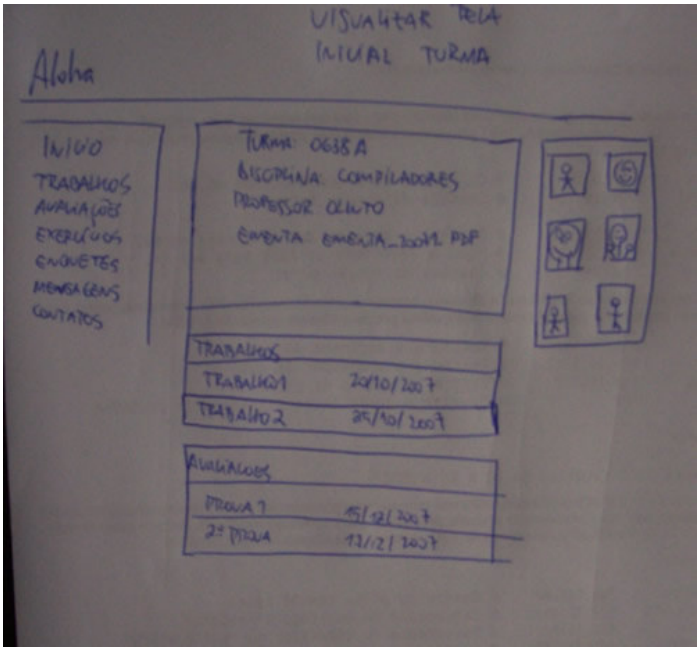


Figura 8.3- Protótipo de Tela de Visualizar Tela Inicial da Turma

9 APÊNDICE B – TESTES DE ACEITAÇÃO

	User Story
US4	Sincronizador UFSC – CAGR
	Efetuar o login no sistema utilizando a matricula e senha de aluno. Os dados pessoais e as turmas dos alunos devem aparecer.
US6	Visualizar Tela Inicial do Professor
	Efetuar o login no sistema utilizando a matricula e senha de professor. Os dados pessoais, as turmas e os gpdas do professor devem aparecer.
US7	Visualizar Turmas do Professor
	Na tela inicial do professor clicar em uma das turmas. Deve aparecer uma listagem com as turmas do professor.
US8	Visualizar Tela Inicial da Turma
	Ao entrar na tela da turma, deve-se mostrar uma listagem de trabalhos, avaliações e mensagens da turma.
US9	Visualizar Trabalhos
	Deve-se mostrar uma listagem de trabalhos da turma.
US10	Visualizar Trabalho
	Deve-se visualizar todos os dados de um trabalho previamente cadastrado
US11	Incluir Trabalho
	Ao incluir um trabalho, Não informar todos os dados obrigatórios. O cadastro não deve ser efetuado.
	Ao incluir um trabalho, Informar todos os dados obrigatórios. O cadastro deve ser efetuado.
US12	Excluir Trabalho
	Selecionar um trabalho de uma listagem e ao exclui-lo o trabalho deve sair desta listagem.
US13	Visualizar e Atribuir Notas de Trabalho
	Ao selecionar um trabalho deve-se apresentar as notas de todos os alunos, agrupados por equipe ou não, com espaço para o professor alterar ou incluir a nota. Ao atribuir a nota, a listagem deve ser atualizada.
US14	Visualizar Avaliações
	Deve-se mostrar uma listagem de avaliações da turma.
US15	Visualizar Avaliação (Professor)
	Deve-se mostrar todos os dados a avaliação.
US16	Incluir Avaliação On-line
	Incluir uma avaliação on-line com perguntas e alternativas e com todos os dados obrigatórios. O cadastro deve ser efetuado.
	Incluir uma avaliação on-line sem perguntas e alternativas e com todos os dados obrigatórios. O cadastro não deve ser efetuado.
	Incluir uma avaliação on-line com perguntas e alternativas e sem os dados obrigatórios. O cadastro não deve ser efetuado.
US17	Incluir Avaliação Presencial

	Incluir uma avaliação presencial com os dados obrigatórios. O cadastro deve ser efetuado.
	Incluir uma avaliação presencial sem os dados obrigatórios. O cadastro deve ser efetuado.
US18	Excluir Avaliação
	Selecionar uma avaliação listagem e ao exclui-la. A avaliação deve sair desta listagem.
US19	Visualizar Notas de Avaliação On-line
	Ao selecionar uma avaliação on-line deve-se apresentar as notas de todos os alunos. Se a avaliação tiver questão subjetiva deve-se mostrar uma mensagem de que a nota ainda não foi calcula pois a questão subjetiva ainda falta ser corrigida.
US20	Corrigir Questões Subjetivas de Avaliação On-line
	Ao corrigi uma questão subjetiva, a listagem de notas deve ser atualizada corretamente.
US21	Ver Resolução da Avaliação On-line de Aluno
	Ao seleciona um aluno, mostrar cada pergunta da prova e suas respectivas alternativas selecionadas e respostas subjetivas.
US22	Visualizar Notas da Avaliação Presencial dos Alunos
	Ao selecionar uma avaliação, mostrar uma listagem com a nota de todos os alunos da Tuma.
US23	Visualizar Arquivos da Turma
	Deve-se mostrar uma listagem com os arquivos que a Turma possui.
US24	Visualizar Arquivo da Turma
	Deve-se poder visualizar todos os dados dos arquivos da Turma e se houver, deve-se ser possível fazer o download do arquivo.
US25	Incluir Arquivo da Turma
	Incluir um arquivo do tipo link informando todos os campos obrigatórios. O cadastro deve ser efetuado.
	Incluir um arquivo do tipo link sem informar os campos obrigatórios. O cadastro não deve ser efetuado.
	Incluir um arquivo do tipo arquivo informando todos os campos obrigatórios. O cadastro deve ser efetuado.
	Incluir um arquivo do tipo arquivo sem fazer o upload do arquivo.O cadastro não deve ser efetuado.
US26	Excluir Arquivo da Turma
	Selecionar uma arquivo listagem e ao exclui-lo. O arquivo deve sair desta listagem.
US27	Visualizar Enquetes da Turma
	Deve-se mostrar uma listagem com as enquetes da Turma
US28	Visualizar Enquete da Turma
	Deve-se mostrar todos os campos da enquete.
US29	Incluir Enquete da Turma
	Incluir um enquete informando todos os campos obrigatórios. O cadastro deve ser efetuado.

	Incluir um enquete sem informar todos os campos obrigatórios. O cadastro não deve ser efetuado.
US30	Excluir Enquete da Turma
	Selecionar uma enquete da listagem e ao exclui-la. A enquete deve sair desta listagem.
US31	Visualizar Resultado da Enquete
	Ao selecionar uma enquete, uma percentagem de votos em cada alternativa da enquete deve ser apresentada.
US32	Visualizar e Enviar Mensagens da Turma
	Deve ser possível visualizar uma listagem com todas as mensagens da turma e também deve ser possível poder adicionar uma mensagem nesta listagem.
US33	Visualizar GPDAs
	Deve-se apresentar uma listagem com todos os GPDAs de um professor.
US34	Visualizar Tela Inicial do GPDA
	Ao selecionar um GPDA, todos os dados do GPDA devem ser apresentados.
US35	Visualizar e Enviar Mensagens de GPDA
	Deve ser possível visualizar uma listagem com todas as mensagens do GPDA e também deve ser possível poder adicionar uma mensagem nesta listagem.
US36	Visualizar Professores de GPDA
	Ao selecionar um GPDA, uma listagem com apresentando os professores de um GPDA deve ser mostrada.
US37	Visualizar Tela Inicial do Aluno
	Efetuar o login no sistema utilizando a matricula e senha de aluno. Os dados pessoais e as turmas do aluno devem aparecer.
US38	Visualizar as Turmas do Aluno
	Deve aparecer uma listagem das turmas do aluno.
US39	Visualizar a Tela Inicial da Turma
	Ao selecionar uma turma, todos os dados da turma devem aparecer.
US40	Visualizar Trabalhos da Turma
	Ao selecionar uma turma, deve-se mostrar uma listagem com todos os trabalhos da turma.
US41	Visualizar Trabalho da Turma
	Quando o aluno selecionar um trabalho de uma turma, deve-se mostrar todos os dados do trabalho.
US42	Enviar Arquivo para Trabalho Individual
	Se o aluno estiver fazendo o trabalho individualmente. Ao fazer o upload do arquivo, uma mensagem de confirmação de recebimento de arquivo deve aparecer.
US43	Visualizar Tela Inicial da Equipe

	Se o trabalho permitir que os alunos se organizem em equipes, um botão fazendo a linkagem com a equipe deve aparecer. Todos os dados da equipe devem aparecer.
US44	Adicionar Participantes na Equipe
	Selecionar um aluno dentre os alunos de uma listagem e adicionar na lista de participantes da equipe. A adição deve ser efetuada com sucesso.
	Selecionar um aluno que já esteja na listagem de participantes da equipe, uma mensagem de erro deve aparecer.
US45	Excluir Participantes da Equipe
	Ao selecionar um participante da listagem, o mesmo deve sair da listagem.
US46	Visualizar Mensagens da Equipe
	Deve ser possível visualizar uma listagem com todas as mensagens da equipe e também deve ser possível poder adicionar uma mensagem nesta listagem.
US47	Visualizar Materiais Auxiliares da Equipe
	Na tela da equipe deve-se apresentar uma listagem com os materiais didáticos da equipe.
US48	Visualizar Material Auxiliar da Equipe
	Ao selecionar um material auxiliar da equipe, deve-se exibir todos os dados do material assim como deve ser possível fazer o download do mesmo.
US49	Incluir Material Auxiliar para a Equipe
	Incluir um arquivo do tipo arquivo informando todos os campos obrigatórios. O cadastro deve ser efetuado.
	Incluir um arquivo do tipo arquivo sem fazer o upload do arquivo.O cadastro não deve ser efetuado.
US50	Excluir Material Auxiliar da Equipe
	Ao selecionar um material de uma listagem para exclusão, o material deve sair da listagem.
US51	Visualizar Trabalhos PD da Equipe
	Deve-se apresentar uma lista dos trabalhos propriamente ditos da equipe.
US52	Visualizar Trabalho PD da Equipe
	Ao selecionar um trabalho todos os dados do trabalho propriamente dito devem aparecer.
US53	Incluir Trabalho PD da Equipe
	Incluir um arquivo informando todos os campos obrigatórios. O cadastro deve ser efetuado.
	Incluir um arquivo do tipo arquivo sem fazer o upload do arquivo.O cadastro não deve ser efetuado.
US54	Excluir Trabalho PD da Equipe
	Ao selecionar um trabalho propriamente dito de uma listagem para exclusão, o trabalho deve sair da listagem.
US55	Visualizar Avaliações
	Ao visualizar as avaliações, uma listagem deve ser exibida.
US56	Visualizar Avaliação On-line

	Ao selecionar uma avaliação on-line todos os dados da mesma devem ser exibidos.
US57	Responder Avaliação On-line
	Selecionar uma avaliação on-line e responder as questões. A nota final deve ser calculada automaticamente.
	Selecionar uma avaliação on-line e não responder as questões. A nota final não deve ser calculada automaticamente.
US58	Visualizar Avaliação Presencial
	Ao selecionar uma avaliação presencial todos os dados da mesma devem ser exibidos.
US59	Visualizar Materiais Didáticos da Turma
	Deve ser apresentada uma listagem dos materiais didáticos da turma.
US60	Responder Enquete
	Selecionar um enquete e selecionar uma das alternativas da enquete. O resultado parcial deve ser apresentado.

10 APÊNDICE C – PLANEJAMENTO DE ITERAÇÃO

Tabela 10.1 - Plano da primeira iteração do primeiro release

UserStory	Responsável	Estimativa de Tempo (horas)	Tempo Real (horas)	Status
US1	Adriano	4	4	
US2	Adriano	4	4	
US3	Adriano	4	4	
US4	Adriano	4	0	
US5	Adriano	4	4	
US6	Arthur	4	5	
US7	Arthur	2	2	
US8	Arthur	4	5	
US9	Arthur	2	3	
US10	Arthur	4	4	
US11	Arthur	8	10	
US12	Arthur	2	1	
US13	Arthur	4	6	

Tabela 10.2 - Plano da segunda iteração do primeiro release

UserStory	**Descrição**	**Responsável**	**Estimativa de Tempo (horas)**	**Tempo Real (horas)**	**Status**
US63.1	Definição do Layout do Sistema	Adriano	5	5	
US63.2	Gerenciador de Mensagens	Adriano	3	3	
US63.3	Controle de exceções	Adriano	3	3	
US63.4	Lógica de validação de unicidade	Adriano	10	10	
US63.5	Lógica de validação de campos obrigatórios	Adriano	10	10	
US63.6	Componente de criação de EJB-QL	Adriano	7	7	
US63.7	Utilitários Gerais sobre Dados	Adriano	35	35	
US63.8	Componente genérico de acesso a banco de dados (DAO)	Adriano	6	6	
US63.9	Interceptador de chamadas da camada de serviço	Adriano	4	4	
US63.10	Gravação de logs de erros da camada de lógica de *negócio*	Adriano	6	6	
US63.11	Componente HashMap com cast para tipos	Adriano	8	8	
US63.12	Respositório de mensagens (Resouce Bundle)	Adriano	2	2	
US63.13	Implementação das classes abstratas de teste	Adriano	2	2	
US63.14	Taglib facelets de funções para as taglibs de entrada e saída de dados	Adriano	3	3	

Tabela 10.3 - Plano da primeira iteração do segundo release

UserStory	Descrição	Responsável	Estimativa de Tempo (horas)	Tempo Real (horas)	Status
US64.1	Tratamento de exceção na camada de apresentação com Aspecto	Adriano	22	22	
US64.2	Taglib facelets para entrada de dados	Adriano	10	10	
US64.3	Taglib facelets para saida de dados	Adriano	10	10	
US64.4	Taglib facelets genérica para tabela de resultado de pesquisa	Adriano	2	2	
US64.5	Taglib facelets genérica para formulário de inclusão e edição	Adriano	2	2	
US64.6	Taglib facelets genérica para detalhes de um registro	Adriano	2	2	
US64.7	Taglib facelets de funções para as taglibs de tabela, formulário e detalhes	Adriano	2	2	
US64.8	Configuração do applicationContext.xml do Spring	Adriano	5	5	
US64.9	Configuração do web.xml	Adriano	5	5	
US64.10	Configuração do faces-config e web-flow	Adriano	2	2	
US64.11	Configuração do Ambiente de Desenvolvimento Eclipse	Adriano	4	4	

Tabela 10.4 - Plano da segunda iteração do segundo release

UserStory	Responsável	Estimativa de Tempo (horas)	Tempo Real (horas)	Status
US67	Adriano	80	80	
US32	Arthur	6	6	
US33	Arthur	6	6	
US34	Arthur	6	6	
US35	Arthur	6	6	
US36	Arthur	6	7	
US37	Arthur	6	6	
US39	Arthur	6	7	
US65	Arthur	10	10	
US66	Adriano	1	0	

11 ANEXOS

11.1 ARTIGO

Aloha
Um Ambiente de Relacionamento para Ensino

Adriano Manoel Demetrio[1], Arthur Martins Pereira[2]

[1]Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

{adrianod,arthurmp}@inf.ufrgs.br

Abstract. *This composition presents the development of Aloha Project, a Virtual Learning Environment with focus on Federal University of Santa Catarina. It also presents the technologies and the software development methodology used on the project.*

Resumo. *Este artigo apresenta o desenvolvimento do projeto Aloha, um Ambiente Virtual de Aprendizado com foco na Universidade Federal de Santa Catarina. Apresenta também as tecnologias e a metodologia de desenvolvimento de software utilizada.*

1. INTRODUÇÃO

É comum observar no curso de Sistemas de Informações da Universidade Federal de Santa Catarina que professores utilizam-se de suas páginas pessoais e do e-mail como principal forma de comunicar eletronicamente seus alunos a respeito das atividades de suas disciplinas, sejam elas, trabalhos, avaliações, resultado de provas, avisos importantes e outros.

Desta forma, muitas vezes, as informações se encontram descentralizadas ou ausentes, dependendo da vontade do professor em manter sua página pessoal atualizada e interessante para os alunos. Além disso, a comunicação entre professores e alunos fica desorganizada na caixa de mensagens de cada um.

Estas desvantagens e outras mais são eliminadas com a utilização de um Ambiente Virtual de Aprendizado (AVA).

Um AVA é um sistema que tem como finalidade contribuir para o processo de ensino e aprendizado, podendo ser utilizado para ensino a distância ou como suporte para aulas presenciais. Existem muitos AVAs no mercado atualmente, com destaque para o Moodle e para o Blackboard, porém o desenvolvimento de Ambiente Virtual de Aprendizado para a própria universidade possibilita um maior ajustamento entre as necessidades e o sistema em questão.

Portanto, o projeto Aloha surge como mais uma alternativa de utilização de Ambientes Virtuais de Aprendizado na Universidade Federal de Santa Catarina, uma vez que a mesma está iniciando o processo de utilização do Moodle.

O projeto Aloha buscou elencar os relacionamentos existentes entre professores e alunos como forma de embasar a identificação das características do software. Para o desenvolvimento do sistema em questão foi utilizada a metodologia de desenvolvimento de software EasyProcess e a tecnologias open-source.

Com resultado do desenvolvimento do projeto Aloha, obteve-se um Ambiente Virtual de Aprendizado que dá base para os relacionamentos existentes entre aluno e aluno, entre aluno e professor assim como entre professor e professor. Tal característica se dá por meio da existência dos ambientes de interação, chamados de ambiente Equipe, ambiente Turma e ambiente GPDA.

2. AMBIENTES VIRTUAIS DE APRENDIZADO

Em um mundo altamente competitivo, o aprendizado torna-se o grande diferencial. Ferramentas que tornam este processo mais ágil assumem um papel cada vez mais importante. Sendo assim, os Ambientes Virtuais de Aprendizado (AVA) vêm para suprir esta necessidade e segundo JISC (2000) a impressão principal é que eles terão um impacto significativo no processo de aprendizagem e ensino no futuro.

Segundo Morgan (2003), os Ambientes Virtuais de Aprendizado (AVA) estão tendo um grande papel na infra-estrutura de tecnologia da educação superior na atualidade. O AVA, segundo definição da mesma autora, pode ser definido como

um software especificamente projetado para instituições de ensino e estudantes utilizarem no ensino e no aprendizado.

As principais características de um Ambiente Virtual de Aprendizado dizem respeito ao gerenciamento das atividades de uma turma de aprendizado, ao controle e disponibilização dos materiais didáticos da mesma, a avaliação da performance dos alunos e a comunicação entre os membros da turma.

Um ambiente de aprendizado virtual propicia controle das atividades acadêmicas visto que algumas funções inerentes ao cotidiano das salas de aulas são gerenciadas pelo software. É possível, portanto, efetuar a chamada on-line dos alunos presentes, agendar trabalhos e avaliações, divulgar o plano de ensino da disciplina, divulgar dados de eventos e noticias relevantes à disciplina e outras funcionalidades. Além disso, pode-se armazenar e divulgar o material da disciplina em questão, servindo como repositório único de todas as disciplinas da universidade, tornando a pesquisa mais acessível aos seus alunos.

Muitos Ambientes Virtuais de Aprendizado dispõem de funcionalidades relativas a criação e resolução de provas e exercícios on-line. Estas funcionalidades, conforme descrito por Catley (2004), tem a capacidade de propiciar um maior engajamento dos alunos para com as suas disciplinas. Por serem automáticas, tais funcionalidades permitem o retorno imediato, ou seja, é possível saber a correção dos exercícios e avaliações on-line em questão de poucos segundos. Com isso o aluno obtém um feedback do seu desempenho mais rapidamente, induzindo o mesmo a focar nos seus erros e procurar o suporte dos professores, seja on-line ou presencialmente na sala de aula, a fim de suprir suas carências o mais rápido possível. Nota-se então que tal sistema é diferente do tradicional sistema de aprendizado em que, após a realização da prova espera-se, em grande parte das vezes, um considerável tempo até obter o resultado das mesmas. Por conseqüência, o atraso da busca da correção dos erros por parte dos estudantes, resulta em notas baixas e reprovação. Além disso, estas funcionalidades são mais econômicas e rápidas, visto que reduzem o custo com fotocópias e agilizam a elaboração e correção das mesmas.

Outro fator interessante relativo às funcionalidades de avaliação e trabalhos on-line, é que estas funcionalidades, em alguns AVAs, auxiliam o professor a monitorar o desempenho dos alunos. Exercícios e avaliações on-line podem a critério do professor, serem respondidas uma ou várias vezes pelos alunos. A cada tentativa de resolução do exercício ou avaliação o Ambiente Virtual de Aprendizado registra o desempenho do aluno. Com isso é possível saber quantas vezes o discente efetuou o teste e como foi o seu desempenho neles. Podendo assim identificar se os alunos após perceberem seus erros, empenhou-se em corrigi-los, e conseqüentemente aprendeu mais.

Outro fator importante presente nos AVAs é a possibilidade de comunicação entre os seus participantes. Suleman (2003) define esta comunicação como síncrona e assíncrona. Comunicação assíncrona é todo tipo de mensagem postada para ser lida posteriormente, como por exemplo, fóruns, e-mails e noticeboards, os quais são informações em que somente certos usuários têm acesso a criação. Por exemplo, um professor poderia utilizar um noticeboard para comunicar informações administrativas de suas disciplinas, como a data da próxima prova, e os demais alunos poderiam somente visualizar tais mensagens, diferentemente de um fórum. Por sua vez, comunicação síncrona é a comunicação em tempo real, obtida, por exemplo, através de chats ou vídeo-conferências, e que permitem conectar pessoas que por questões de distância não poderiam estar na sala de aula.

A comunicação entre os membros do AVA dá suporte para o trabalho colaborativo, ou seja, propicia que os integrantes de uma turma acordem questões relativas aos seus trabalhos de aula, utilizando-se das funcionalidades de comunicação do AVA. Estas funcionalidades quebram as barreiras impostas pela distância, pelo tempo e pela timidez. As dificuldades inerentes a distâncias são eliminadas uma vez que única limitação para o acesso ao processo de aprendizagem é uma conexão a internet. As mensagens assíncronas permitem que pessoas troquem mensagens nas suas horas mais convenientes e impedem que incompatibilidades nos horários dos integrantes atrapalhem o rendimento do trabalho, eliminando assim as barreiras do tempo. E conforme descrito por Nancy Chick da Universidade de Wisconsin em Morgan (2003), a utilização do AVA encoraja a participação dos alunos tímidos, portanto a timidez torna-se menos

prejudicial ao aprendizado. A comunicação presente no AVA também dá base para o suporte dos professores e monitores para com os alunos.

Em uma instituição de ensino, a mesma pode utilizar somente o Ambiente Virtual de Aprendizado, ou então utilizar o mesmo integrado com MIS ou LibMS da instituição. O MIS, *Management Information System*, é o sistema gerenciador de informações e segundo Jarvis (JARVIS et al 2005), é um sistema que permite que faculdades ou escolas gerenciem os elementos essenciais do seu negócio, incluindo os números dos alunos, construções, quadro de horários e orçamento, e que permite tomar decisões, planejar e responder aos pedidos de informações dos gerentes e demais solicitantes. Portanto, o MIS é o sistema que armazena os dados pessoais dos alunos, assim como as suas disciplinas matriculadas, sua grade de horário, suas notas e outros cadastros afins. Logo, torna-se plausível uma integração entre o AVA e o MIS no tocante à troca de informações, com o intuito de evitar entrada manual de dados excessiva e duplicação de dados.

Além da interação com o MIS, o AVA pode comunicar-se com o sistema gerenciador da biblioteca da universidade, mais conhecido como LibMS (Library Management System), a fim de fazer consultas de recursos e reservas.

Segundo Wilson (2002), esta integração entre Ambiente Virtual de Aprendizado, MIS e LibMS compõe o Managed Learning Environment (MLE), como ilustrado na Figura 2 a seguir.

Figura 4 - Partes do MLE

3. EASYPROCESS

O EasyProcess (YP) é uma metodologia de desenvolvimento de software criada com o intuito de atender o desenvolvimento de projetos acadêmicos. Esta metodologia foi idealizada pela Professora Drª Francilene Procópio Garcia da Universidade Federal de Campina Grande (UFCG) e foi concebida no ambiente do grupo PET Computação desta mesma universidade, no ano de 2003.

O YP foi moldado com base nas características de um projeto de desenvolvimento de software acadêmico, como por exemplo, o escopo pequeno, em virtude da duração de um semestre, e a utilização de tecnologias consideradas estado da arte. Além disto, esta metodologia prima pela boa produtividade do projeto, amparado pelo uso de boas ferramentas de apoio e pela geração mínima de artefatos de software. O EasyProcess é uma metodologia simples e de fácil entendimento pelos integrantes da equipe, porém é completo e robusto a ponto de gerar produtos de qualidade.

Esta metodologia está apoiada em práticas do XP, RUP e Agile Modeling, utilizando o que há de mais adequado em cada uma delas em relação ao ambiente acadêmico.

4. TECNOLOGIAS

O desenvolvimento foi realizado utilizando a análise, projeto e programação orientada a objetos. A linguagem de programação escolhida foi a Java. O sistema foi dividido em quatro camadas: apresentação, serviço, lógica de negocio e integração. Cada camada tem suas responsabilidades.

4.1 CAMADA DE APRESENTAÇÃO

As responsabilidades da camada de apresentação são:

- Apresentar as informações através de textos e imagens;
- Obter as entradas de dados;
- Capturar os eventos realizados pelo usuário;
- Processar os eventos;

- Direcionar o sistema conforme a resposta do evento;
- Apresentar mensagens geradas pelo sistema; e
- Dar suporte a internacionalização.

O servidor utilizado foi o Apache Tomcat versão 6.0.13 que dá suporte aversão servlet 2.5 e ao Java Server Pages 2.1. Foi utilizado o Java Server Faces 1.2 como framework MVC. A parte de controle do MVC foi realizada com o uso do Spring Web Flow. Para criar os templates de telas foi utilizado o Facelets que também foi utilizado para criar taglibs. Para habilitar o Ajax no sistema foi utilizado o framework Ajax4JSF e a biblioteca de componentes Ajax RichFaces. Outros bibliotecas de componentes foram utilizadas como a JSF Sandbox e a Apache MyFaces Tomahawk. Na parte do navegador web foi utilizado o formato XHTML de dados e o uso de estilos CSS. Para deixar as páginas mais dinâmicas foi utilizado Javascript e algumas bibliotecas com funções prontas como a Prototype e a Scriptaculus.

4.2 CAMADA DE SERVIÇO

As responsabilidades da camada de serviço são:

- Iniciar a transação que será utilizada na camada de lógica de negócio;
- Finalizar a transação utilizada com sucesso (COMMIT) ou erro (ROLLBACK);
- Registro de erros.

O gerenciador de transação utilizado foi o JpaTransactionManager do Framework Spring. Este gerenciador executas todas as tarefas necessárias sobre o gerenciamento de transações. Para o registro de erro foi utilizado a parte de orientação a aspecto do Spring através da classe MethodInterceptor. Através desta classe foi criado um interceptador entra a camada de apresentação e a camada de lógica de negócio. Ao ocorrer um erro este interceptador capturar esse erro e registro em um arquivo.

4.3 CAMADA DE LÓGICA DE NEGÓCIO

As responsabilidades da camada de lógica de negócio são:

- Implementar as regras de negócio dos casos de uso;
- Lançar erros caso ocorram;
- Acessar outros componentes de negócio.

Esta camada possui apenas classes Java apenas com a implementação das regras definidas nos casos de uso. São as chamadas classes POJO (Plain Old Java Object). Todas as classes da camada de lógica de negocio são instanciadas e gerenciadas pelo container de beans do Spring.

4.4 CAMADA DE INTEGRAÇÃO

As responsabilidades da camada de integração são:

- Prover acesso a camada de lógica de negócio;
- Obter e persistir informação em banco de dados;
- Buscar informações em outros sistemas (CAGR);

O acesso ao banco de dados é realizado através da especificação Java Persistence API (JPA) que faz parte da especificação do Enterprise Java Beans 3.0. O framework utilizado junto com o JPA foi o Hibernate. Alem de acesso ao banco de dados a camada de integração fornece acesso a outros sistemas, no caso do Aloha será acessada a base de dados do Sistema Acadêmico de Graduação (CAGR) da UFSC através de JDBC.

5. IDENTIFICAÇÃO DOS RELACIONAMENTOS

Os relacionamentos identificados que basearão o desenvolvimento do projeto Aloha foram divididos em três categorias

5.1 ALUNO X ALUNO

O relacionamento entre alunos foi identificado no contexto de uma turma de uma disciplina e de uma equipe de desenvolvimento de um trabalho. Em uma turma os alunos precisam interagir entre si com o intuito de trocar conhecimento, informações ou experiências. Em uma equipe, os aluno interagem com um objetivo em comum, desenvolver um trabalho acadêmico solicitado por um professor.

5.2 ALUNO X PROFESSOR

A interação entre alunos e professores se dá por meio de uma turma de uma disciplina. Esta interação tem o objetivo de possibilitar a troca de conhecimento entre ambas as partes, e é possibilitada pela comunicação síncrona ou assíncrona e pelo compartilhamento de materiais didáticos. Como forma de auxiliar, direta ou indiretamente, esta troca de conhecimento e de contribuir para o aprendizado dos alunos, o professor faz uso de algumas atividades, como provas, trabalhos, chamadas, questionamentos que visam obter feedback dos alunos e outros. Portanto, otimizar e gerenciar estas atividades contribui para a troca de conhecimento entre os alunos e professor e otimiza o aprendizado.

5.3 PROFESSOR X PROFESSOR

A interação entre professores ocorre principalmente entre aqueles que dominam assuntos em comum. Professores de uma mesma área de conhecimento interagem entre si a fim de trocar conhecimento e alinhar as metodologias de ensino e conteúdo de suas disciplinas.

As disciplinas de um curso de uma universidade estão agrupadas em áreas de conhecimento, como por exemplo, a área conhecimento de programação na Universidade Federal de Santa Catarina, é formada pelas disciplinas de Introdução à Programação Orientada a Objetos, Desenvolvimento de Sistemas Orientados a Objetos I, Estrutura de Dados, Desenvolvimento de Sistemas Orientados a Objetos II e Paradigmas de Programação.

Portanto o agrupamento de professores se dá tendo em vista as disciplinas que lecionam.

6. FUNCIONALIDADES

As funcionalidades desenvolvidas para o projeto Aloha foram embasadas nos relacionamento existentes entre professores e alunos. Estas funcionalidades foram divididas em três ambientes de interação.

6.1 AMBIENTE EQUIPE

Este ambiente de interação tem como objetivo dar base ao relacionamento existente entre os membros de uma equipe de trabalho. Este relacionamento ocorre entre aluno e aluno e tem como principal finalidade a viabilização de ferramentas de apoio para o sucesso do desenvolvimento do trabalho. As principais funcionalidades são:

- Compartilhamento de arquivos auxiliares para o trabalho.
- Compartilhamento de arquivos dos trabalhos propriamente ditos
- Troca de mensagens entre os participantes da equipe.

6.2 AMBIENTE TURMA

Este ambiente de interação tem como objetivo propiciar a interação entre aluno e professor e entre aluno e aluno de uma turma.

A interação entre aluno e professor ocorre através da execução das atividades inerentes da própria turma e as principais funcionalidades que dão suporte à esta interação são:

- Criação de avaliações on-line.
- Execução de avaliações on-line.
- Calculo automático do resultado das avaliações on-line.

- Controle e disponibilização de informações a respeito das avaliações realizadas em sala de aula.
- Controle e disponibilização de informações a respeito dos trabalhos da disciplina.
- Escolha de alunos gerentes de equipes.
- Visualização de notas.
- Disponibilização de material didático.
- Disponibilização da ementa da disciplina.
- Envio de contribuições pessoais por parte dos alunos para com a disciplina.
- Divulgação de eventos relativos ao conteúdo da disciplina.
- Divulgação de noticias relativas ao conteúdo da disciplina.
- Gerenciamento do calendário das atividades da turma.
- Criação de enquetes para turma.
- Troca de mensagens assíncronas entre os alunos e o professor.

As funcionalidades que dão suporte para a interação entre aluno e aluno no ambiente Turma são:

- Troca de mensagens assíncronas entre o aluno e aluno.
- Envio de contribuição pessoal por parte dos alunos para com a disciplina.

O ambiente de interação Equipe é habilitado para aqueles trabalhos em que o professor permite que se formem equipes. Portanto pode-se dizer que o ambiente Equipe é um sub-ambiente de interação entre aluno e aluno no ambiente Turma.

6.3 AMBIENTE GPDA

GPDA é o acrônimo de Grupo de Professores de Disciplinas Afins. Este ambiente de relacionamento busca propiciar um relacionamento entre os diferentes

professores de disciplinas que tem alguma relação em comum. Professores de disciplinas como, por exemplo, Banco de Dados I, Banco de Dados II e Projetos de Banco de Dados podem através desde ambiente virtual alinhar as suas metodologias de ensino e seus respectivos conteúdos. Este tipo de ambiente possibilita também que haja uma troca de conhecimento entre estes professores.

As principais funcionalidades presentes no ambiente GPDA são:

- Troca de mensagens entre os participantes do GPDA.
- Compartilhamento de arquivos.

7. REFERENCES

Morgan, Glenda. (2003), "Faculty Use of Course Management System", http://www.educause.edu/ir/library/pdf/ers0302/rs/ers0302w.pdf.

JISC. (2000), "Managed Learning Environments: A Workshop run by JISC Assist, 29 February and 7 March 2000. Final Report. JISC", http://www.jisc.ac.uk/index.cfm?name=event_report_mle.

Catley, Paul. (2004), "Enhancing Learning through Online Assessment", http://www.jisc.ac.uk/uploaded_documents/oxfordv2.doc.

Suleman, Shakeel. (2003), "Use of Communications Tools within VLEs", http://ferl.becta.org.uk/display.cfm?resID=5497.

Jarvis, Janis et al. (2005), "A Cross-Cluster Qualitative Study from the External Evaluation", http://www.evaluation.icttestbed.org.uk/files/mis_systems_report.pdf.

Wilson, Scott. (2002), ".Interoperability: Why and how", www.jisc.ac.uk/uploaded_documents/Interoperability%20(Scott%20Wilson).ppt.

12 GLOSSÁRIO

Map – Interface de programação Java cuja implementação permite o armazenamento de informações na qual cada informação é associada a uma chave. Assim a partir da chave é possível obter a informação armazenada em uma Map. No Java existem várias implementações de Map como: HashMap, LinkedHashMap.