UNIVERSIDADE FEDERAL DE SANTA CATARINA

MÁRIO HEIDRICH VICENTIM - 0423840-0

SISTEMA DE GERENCIAMENTO DE ATIVIDADES EXTRACURRICULARES

FLORIANÓPOLIS, 30 DE JUNHO DE 2008.

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

MÁRIO HEIDRICH VICENTIM - 0423840-0

SISTEMA DE GERENCIAMENTO DE ATIVIDADES EXTRACURRICULARES

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação, orientado pelo professor José Eduardo De Lucca, Msc.

FLORIANÓPOLIS, 30 DE JUNHO DE 2008.

MÁRIO HEIDRICH VICENTIM - 0423840-0

SISTEMA DE GERENCIAMENTO DE ATIVIDADES EXTRACURRICULARES

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

Orientador: José Eduardo De Lucca, Msc.

Banca Examinadora

Vitório Bruno Mazzola,

Delson de Valois Santos,

Djali Avelino Valois.

À minha família e a minha amada

namorada Carolina Ramos Berbel Angulski.

AGRADECIMENTOS

Agradeço primeiramente a minha família e a minha namorada pelas palavras de incentivo e apoio.

Ao meu orientador De Lucca pela dedicação, atenção, oportunidade de aprendizado e principalmente por ter acreditado em mim.

As professoras do Studio de Dança pela ajuda, disponibilidade e paciência em explicar as suas rotinas administrativas.

E finalmente, a Deus por ter me dado forças para vencer mais uma etapa de minha vida.

RESUMO

A ampla necessidade de organização, agilidade e transparência no gerenciamento de atividades extracurriculares já não são mais supridas pelos atuais meios administrativos convencionais não informatizados. Pequenas instituições, colégios e cursos sofrem com a falta de ferramentas que os auxiliem em suas tarefas diárias.

O sistema proposto por este trabalho, projetado seguindo partes de algumas metodologias e melhores práticas de desenvolvimento, vem para tentar suprir esta carência, oferecendo algumas funcionalidades que tornarão mais fácil o gerenciamento destas atividades.

Palavras-chave: Metologias de desenvolvimento, Desenvolvimento Web, Gerenciamento, Extracurricular e Ferramentas.

ABSTRACT

The ample need of organization, agility and transparency in the management of extracurricular activities are no more supplied by the present conventional not computer-aided means of administration. Small institutions, schools and courses suffer with the lack of tools for help then in their journey work.

The system proposed by this study, projected following parts of some methodologies and best practices of development, came for try to supply this neediness, offering some functionalities that will make the management of this activities easier.

Key-Words: Methodologies of development, Web development, Management, Extracurricular and Tools.

SUMÁRIO

ÍNDICE DE FIGURAS

LISTA DE ABRAVIATURAS E SIGLAS

UML - Unified Modeling Language.

IEE - Instituto Estadual de Educação.

OMG - Object Management Group.

RUP - Rational Unified Process.

1. INTRODUÇÃO

O aumento da velocidade e da complexidade das mudanças implica a necessidade de utilização de ferramentas gerenciais (BRYSON, 1988:1 apud OLIVEIRA & SANCHES, 2003:4). Nas atividades extracurriculares, isto também ocorre. O processo manual de execução das tarefas pode acabar gerando problemas devido à sua total dependência da competência humana, o que pode acarretar em desorganização, erros e atrasos, sem falar na possibilidade de desvios e atos ilegais de pessoas que agem de má-fé.

Segundo CHIAVENATO (2000), a tecnologia proporciona uma eficiência maior, uma precisão maior e a liberação da atividade humana para tarefas mais complicadas e que exijam planejamento e criatividade.

É com base nesta visão que este trabalho idealiza o planejamento de um sistema que venha para tentar ajudar a solucionar estas necessidades e problemas. Oferecendo uma ferramenta que dê suporte para a execução das tarefas envolvidas nestas atividades. Isto fará com que a responsabilidade total humana diminua, o que não quer dizer q ela será totalmente eliminada, pois como diz REZENDE (2002) *"Tecnologia por tecnologia, sem planejamento, sem gestão e ação efetivas não traz contribuição para a empresa".*

Para uma melhor compreensão do domínio do problema este contará com o auxílio de profissionais do Studio de Dança do Instituto Estadual de Educação, que participarão como clientes deste sistema.

1.1. DELIMITAÇÃO DO TEMA

Este trabalho abordará o desenvolvimento de um sistema que se predispõe a atender qualquer atividade complementar ao ensino que necessite de ferramentas para auxiliar o gerenciamento de turmas e controle financeiro da mesma.

1.2. OBJETIVO GERAL

Implementar uma ferramenta gratuita para auxiliar o gerenciamento de turmas e controle financeiro de atividades extracurriculares e através disso auxiliar escolas e cursos que não tenham condições de possuir um sistema pago a organizarem suas atividades extracurriculares.

1.3. OBJETIVOS ESPECÍFICOS

Estudar conceitos de metodologias e boas práticas em análise e desenvolvimento de software vistas durante o curso, visando aprender mais sobre estas.

Aprender mais sobre desenvolvimento de projetos, e ter uma noção prática de como este processo funciona.

1.4. MOTIVAÇÕES

A identificação da necessidade que alguns cursos e escolas têm por sistemas os quais organizem suas atividades de uma forma que torne mais fácil a administração destes e que também auxilie o combate a fraudes e desvios que eventualmente ocorrem em nestes locais, devido à forma precária e manual como é feito este controle.

1.5. METODOLOGIA DE DESENVOLVIMENTO

Este trabalho é uma pesquisa aplicada onde o conhecimento adquirido será utilizado para um objetivo específico, o desenvolvimento de um sistema utilizando conceitos de metodologias e boas práticas em desenvolvimento de software.

A metodologia de pesquisa utilizada para reunir informações é a pesquisa bibliográfica assim como também a realização de entrevistas com os clientes do sistema, para ter uma base de conhecimento sobre o tema em questão.

1.6. ORGANIZAÇÃO DO TRABALHO

Este trabalho é composto por 6 capítulos e está organizado da seguinte maneira: No primeiro capítulo está esta introdução, que descreve o escopo deste trabalho. O segundo capítulo apresenta conceitos de metodologias, notações e boas práticas utilizadas. O capítulo 3 trará uma descrição sobre os requisitos do sistema,

que foram obtidos através do cliente do sistema, o Studio de dança do IEE. O Capítulo 4 Apresentará a descrição das etapas de análise e desenvolvimento do sistema. A implementação é descrita no capítulo 5 e por último no capitulo 6 virão à conclusão e os trabalhos futuros.

2 CONCEITOS ENVOLVIDOS

O sistema será planejado usando o paradigma de orientação a objetos, e se baseará nos moldes da engenharia de software, tendo uma fase de pesquisa, onde serão estudadas as ferramentas utilizadas, os conceitos necessários para o desenvolvimento do projeto e alguma outra informação necessária ao sistema. Terá também a fase de levantamento de requisitos e de análise, utilizando para isto algumas noções de UML, RUP e do Modelo Incremental. Sendo estes passos não necessariamente seqüenciais na sua execução, podendo alguns deles ser cíclicos.

2.1. PROCESSO DE DESENVOLVIMENTO DE SISTEMAS

Segundo PRESSMAN (2002, apud LUSTOSA, 2007:2), “Processo de desenvolvimento de sistemas, ou de software, é um roteiro que o ajuda a criar em tempo hábil um resultado de alta qualidade como produto”.

Um processo de software é formado por um conjunto de passos de processo parcialmente ordenados, relacionados com artefatos, pessoas, recursos, estruturas organizacionais e restrições, tendo como objetivo produzir e manter os produtos de software finais requeridos (REIS, 2002).

2.2. UML

A UML é uma linguagem para modelagem visual, que oferece suporte a diversas linguagens de programação, podendo ser aplicada a um processo de desenvolvimento de software. (SILVA, 2001:17)

A UML foi desenvolvida por Grady Booch, James Rumbaugh, e Ivar Jacobson. De acordo com ESMIN(????) “A UML é o sucessor de um conjunto de métodos de análise e projeto orientados a objeto que está, atualmente, em processo de padronização pela OMG”.

BEZERRA (2007:14) define UML como uma linguagem constituída de elementos gráficos utilizados na modelagem que permitem representar os conceitos do paradigma da orientação a objetos.

A UML é formada por vários elementos de modelo que representam as diferentes partes de um sistema de software. Os elementos UML são usados para criar diagramas, que tem a função de representar uma parte ou um fluxo do sistema.

De acordo com JACOBSON (2000, Apud AMORIM, 2007:2), a UML nos oferece diversos diagramas flexíveis, que podem ser adaptados a qualquer tipo de sistema, incluindo projetos de tempo real e para Web.

2.2.1. DIAGRAMAS UML

Segundo BOOCH (2000, apud MODESTO, 2006:31) um diagrama é a representação gráfica de um conjunto de elementos do sistema.

A UML disponibiliza nove desses diagramas que permite diferentes partes do modelo de um sistema (BOOCH, 2000, apud MODESTO, 2006:32):

- § Diagrama de Seqüência;
- § Diagrama de Caso de Uso;
- § Diagrama de Classes;
- § Diagrama de Objetos;
- § Diagrama de componentes;
- § Diagrama de Implantação;
- § Diagrama de Colaboração
- § Diagrama de Estados
- § Diagrama de Atividades

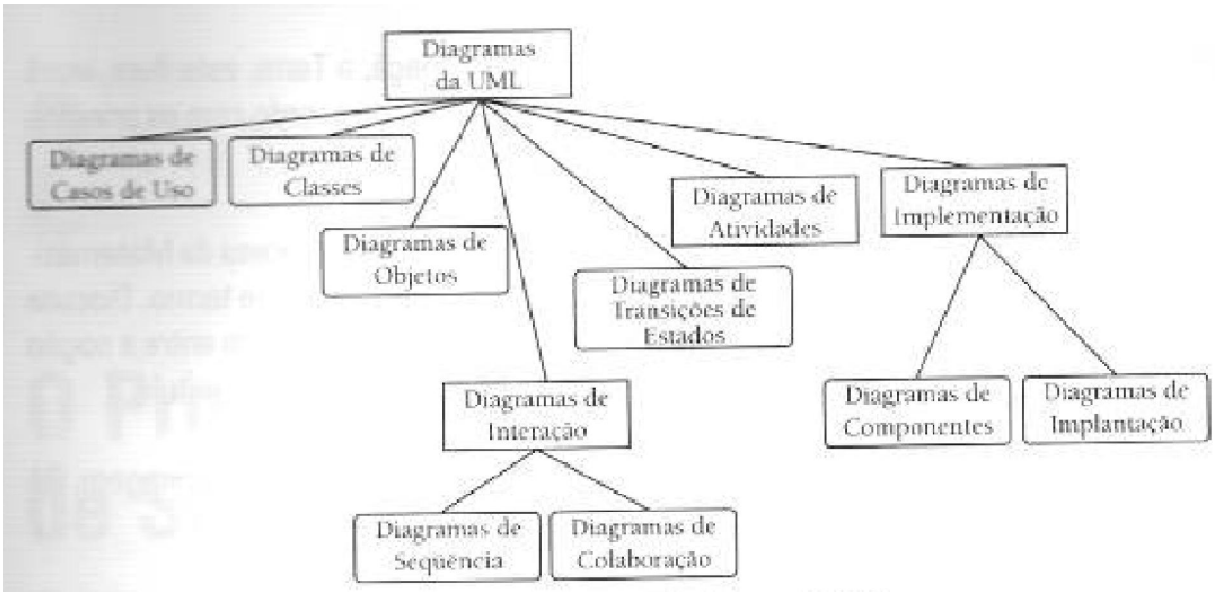


Figura 2.1 - Diagramas da UML

Neste trabalho serão utilizados os diagramas de caso de uso, de classe e de seqüência, abaixo seguem suas definições:

Diagrama De Casos De Uso:

Um diagrama de caso de uso representa um conjunto de cenários identificados, útil aos usuários do sistema. Segundo SILVA (2001:37) sua estrutura é formada principalmente por três elementos:

Caso de Uso - Descrição de um cenário do sistema. Usa elementos gráficos e textuais. Um caso de uso consiste em uma interação entre um usuário e o sistema (BOOCH, 2000 apud MODESTO, 2006:33). É um modo onde são definidos processos genéricos que serão manipulados pelo sistema, estabelecendo assim, um conjunto de funcionalidades relacionadas ao sistema (JACOBSON, 1992 apud MODESTO, 2006:33). FOWLER (2005, apud Farias, 2006:14) define casos de uso como uma técnica para capturar requisitos funcionais de um sistema. Segundo ele, os casos de uso descrevem as interações típicas entre os usuários do sistema e o sistema.

Ator - Solicitante dos serviços disponíveis no Caso de uso.

Relacionamentos - Servem para organizar os casos de usos, os mais utilizados são a generalização, a inclusão e a Extensão.

Através deste diagrama é possível captar o comportamento desejado de um sistema, por meio da análise dos requisitos necessários para a construção do sistema. Além disto, também permite documentar todos os requisitos de um sistema, organizá-los, eliminando as redundâncias e também identificar os riscos possíveis para a fase de construção do sistema. (SILVA, 2001:37).

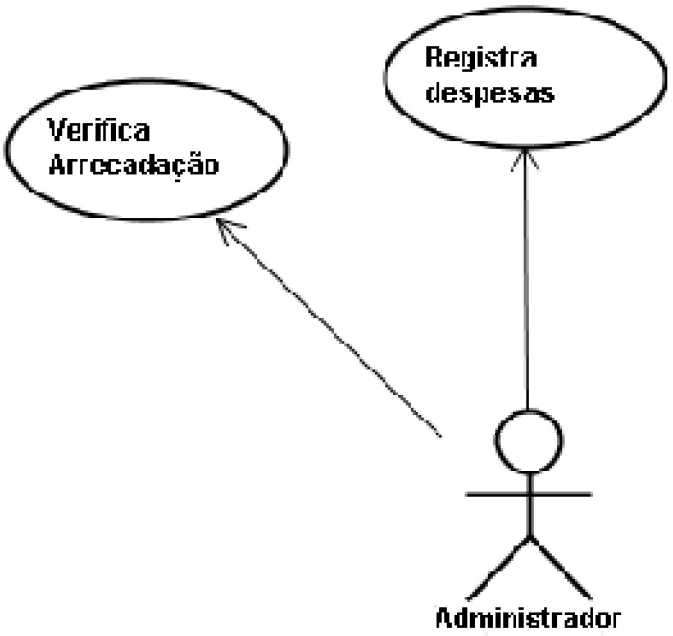


Figura 2.2 - Exemplo de um Diagrama de Caso de Uso.

Diagrama de Classes:

Segundo SILVA (2001:39) o diagrama de classes representa a estrutura de um sistema orientado a objeto, aonde são exibidos suas classes e seus respectivos relacionamentos e colaborações.

Uma classe descreve as propriedades (atributos) e comportamentos (métodos) de um objeto. São usadas para representar os papéis dos elementos do modelo que oferecem o comportamento necessário para atender aos requisitos funcionais especificados pelos casos de uso e aos requisitos não funcionais descritos nas especificações. (MODESTO, 2006:37).

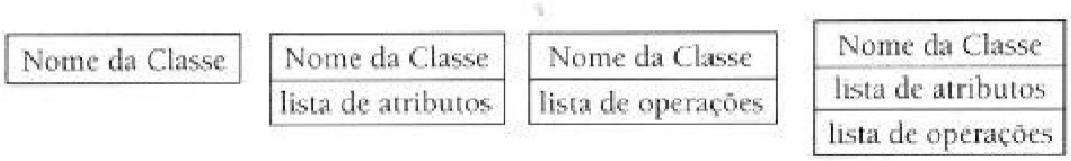


Figura 2.3 - Possíveis Notações de uma classe Fonte: BEZERRA (2007:98).

Segundo BEZERRA (2007:98) uma classe é representada através de uma caixa, com o nome da classe no primeiro compartimento, os atributos, que são as informações que um objeto armazena, logo abaixo no segundo compartimento e por fim as operações, ações que os objetos de uma classe sabem realizar, no terceiro compartimento.

Figura 2.4 - Possíveis representações de uma mesma classe. Fonte: BEZERRA (2007:98)

Diagrama de Seqüência:

Um Diagrama de Seqüência foca a ordem de tempo das mensagens, mostrando um conjunto de objetos e mensagens enviadas e recebidas por esses objetos. (MODESTO, 2006).

Um Diagrama de Seqüência é composto pelos seguintes elementos:

- **Atores**: São entidades externas que interagem com o sistema e que solicitam serviços, gerando dessa forma eventos que iniciam processos.
- **Objetos**: Representam as instâncias das classes representadas no processo. Os objetos são ilustrados como retângulos. Eles compõem a dimensão horizontal (->).
- **Gate**: Indica um ponto em que a mensagem pode ser transmitida para dentro ou para fora do fragmento de interação.
- **Fragmento**: Fragmentos de interação como: Alt (Alternativa), Opt (Opcional), Break (Parar), Loop (Repetição) e outras.
- **Linha de vida**: Compõem a dimensão vertical (tempo). A dimensão vertical é a seqüência onde a vida do objeto durante a interação representada.

Os Diagramas de Seqüência devem ser construídos com base na informação disponível nos casos de uso, dessa forma diz-se que diagramas de seqüência desenham os casos de uso (BOOCH, 2000, apud MODESTO, 2006).

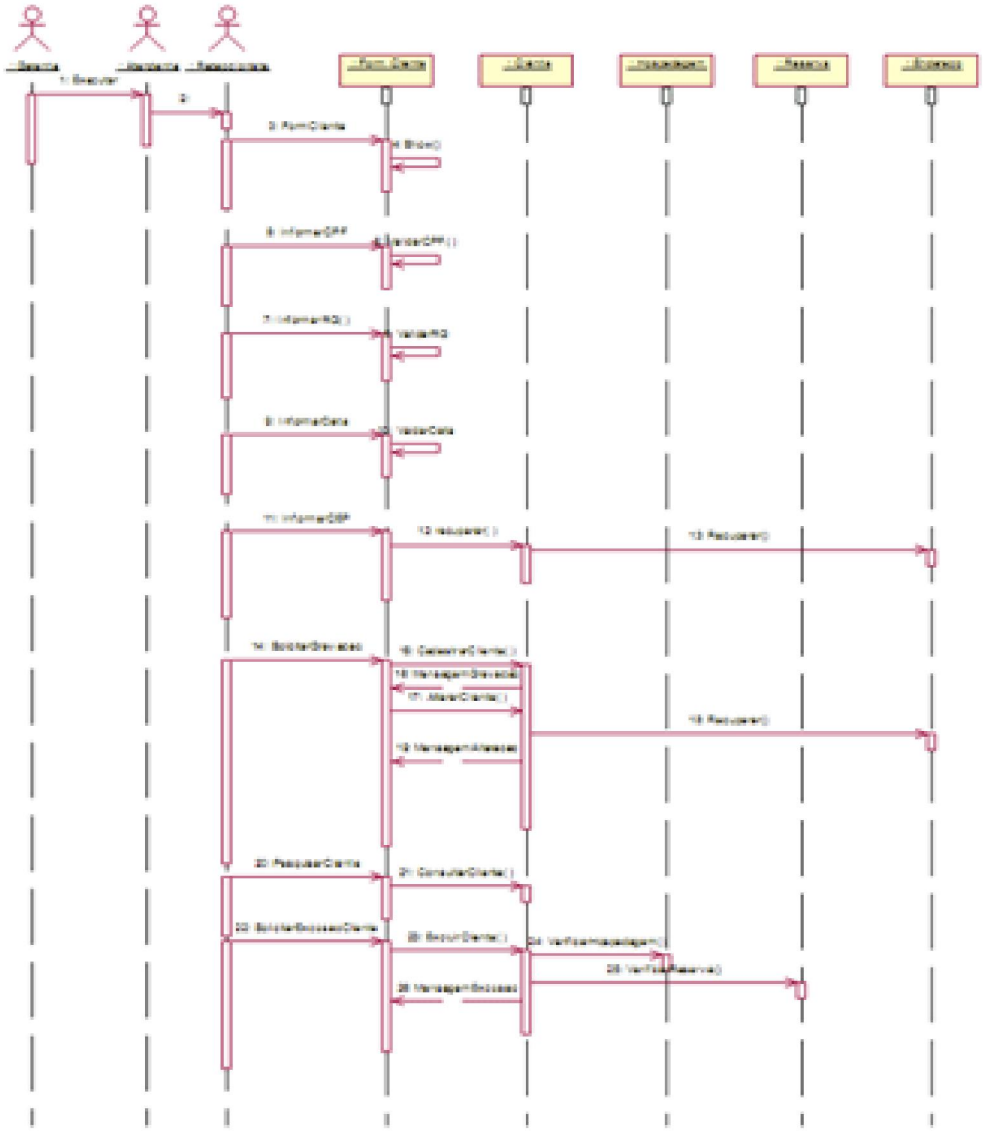


Figura 2.5 - Diagrama de Seqüencia Fonte:MODESTO (2006:59)

2.2. MODELO INCREMENTAL

Segundo ROCHA (2002), O Modelo Incremental foi desenvolvido através da combinação entre os modelos linear e prototipação. O desenvolvimento é dividido em etapas, denominadas “incrementos”, que produzirão incrementalmente o

sistema, até a sua versão final. Em cada incremento é realizado todo o ciclo do desenvolvimento de software, do planejamento aos testes do sistema já em funcionamento. Cada etapa produz um sistema totalmente funcional, apesar de ainda não cobrir todos os requisitos. O Modelo Incremental apresenta diversas vantagens para o desenvolvimento de um software, especialmente se os requisitos não estão claros inicialmente.

De acordo com BEZERRA (2008) “O modelo incremental e iterativo foi proposto como uma resposta aos problemas encontrados no modelo em cascata. Um processo de desenvolvimento segundo esse modelo divide o desenvolvimento de software em *iterações*. Em cada iteração, são realizadas as atividades de análise, projeto, implementação e testes para *uma parte* do sistema. Esta característica contrasta com o modelo em cascata, no qual as fases de análise, projeto, implementação e testes são realizados uma única vez para o sistema *como um todo*. Ele complementa dizendo que o modelo incremental e iterativo pode ser visto como a aplicação do modelo em cascata várias vezes: *o software é desenvolvido em incrementos, e cada incremento é desenvolvido segundo uma "mini-cascata"*. (O modelo em cascata também pode ser visto como o modelo incremental e iterativo no qual há somente uma iteração.)”.

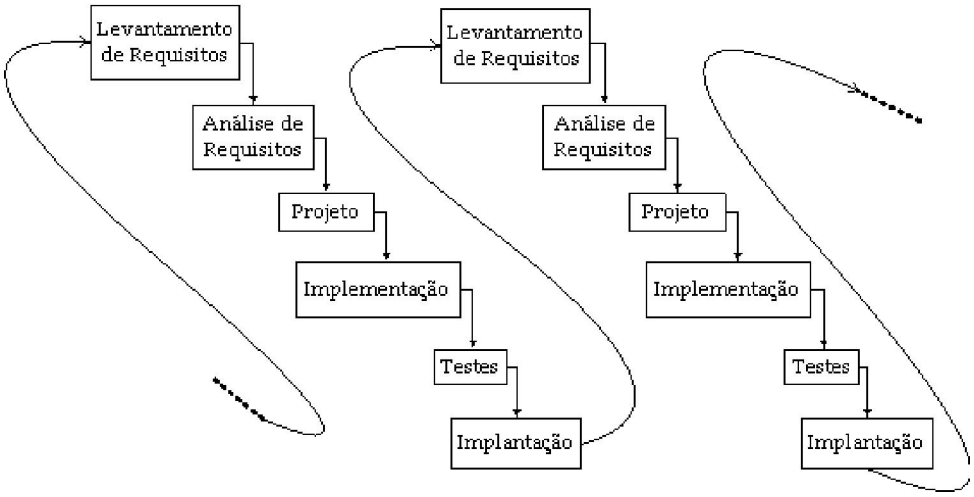


Figura 2.6 - Modelo Incremental. Fonte: Bezerra

DAVIS (1997, Apud ROCHA, 2002) cita algumas vantagens do modelo incremental:

- A construção de um sistema menor é sempre menos arriscada que a construção de um grande;
- Se um grande erro é cometido, apenas o último incremento é descartado;
- Reduzindo o tempo de desenvolvimento de um sistema, as chances de mudanças nos requisitos do usuário durante o desenvolvimento são menores.

2.3 RUP

Segundo FALBO (2000:6) o Processo Unificado proposto pela Rational foi criado para apoiar o desenvolvimento orientado a objetos, fornecendo uma forma sistemática para se obter reais vantagens no uso da Linguagem de Modelagem

Unificada. Segundo MORAES (2006:15) uma das características principais desse processo é a busca contínua pelo desenvolvimento de software com qualidade.

LUSTOSA (2007:8) diz que o RUP utiliza uma abordagem de orientação a objetos em sua concepção para a modelagem dos sistemas, e é projetado e documentado utilizando a notação UML para ilustrar os processos em ação.

O RUP está fundamentado em três princípios básicos: orientação a casos de uso, arquitetura e iteração. Ele é dito *dirigido a casos de uso*, pois são os casos de uso que orientam todo o processo de desenvolvimento. Com base no modelo de casos de uso, são criados uma série de modelos de análise, projeto e implementação, que realizam estes casos de uso. É *centrado em arquitetura*, pois defende a definição de um esqueleto para a aplicação (a arquitetura), a ganhar corpo gradualmente ao longo do desenvolvimento. Finalmente, o RUP é iterativo e incremental, oferecendo uma abordagem para particionar o trabalho em porções menores ou mini-projetos (FALBO, 2000:4).

Figura 2.7 - Modelo de desenvolvimento iterativo e incremental

O ciclo de vida de software do RUP é dividido em quatro fases seqüenciais, cada uma concluída por um marco principal, ou seja, cada fase é basicamente um intervalo de tempo entre dois marcos principais. As 4 fases são Iniciação, Elaboração, Construção e Transição. TANAKA (2008).

Segundo MORAES (2006:16) O RUP utiliza seis boas práticas de desenvolvimento de software, são elas:

- Desenvolver software iterativamente
- Gerenciar requisitos
- Utilizar arquiteturas baseadas em componentes
- Modelar o software visualmente
- Verificar a qualidade do software de forma contínua
- Controlar as mudanças do software

3. LEVANTAMENTO DE REQUISITOS

Segundo MOREIRA & PROTIL (2004:2) apesar do levantamento de requisitos ser uma atividade comum na análise de um sistema, não existe uma metodologia única para a definição dos mesmos. Sobre isto SANTOS (1999:10 apud MOREIRA & PROTIL, 2004:2) também fala “o analista deve identificar qual se adapta mais às características do ambiente em que está sendo desenvolvido o sistema de informação ou utilizar a combinação de diferentes abordagens”.

Com base nisto decidi por utilizar o padrão do UML e do RUP neste trabalho para realizar o levantamento dos requisitos do sistema.

Segundo KRUCHTEN (2003, apud Farias, 2006:14), o RUP define gerenciamento de requisitos como uma abordagem sistemática para localizar, documentar, organizar e controlar os requisitos variáveis do sistema, sendo os requisitos funções ou capacidades que o sistema deve atender

Para facilitar nesta definição, este trabalho terá como cliente o Studio de Dança do IEE, mas o produto final poderá ser utilizado para qualquer atividade extracurricular.

3.1. ENTREVISTA COM O CLIENTE

Para obter os requisitos foram feitas entrevistas, com as analistas de negócio, onde foram levantas algumas questões para coletar informações sobre as atividades desenvolvidas no Studio, para com isto identificar quais eram as necessidades do sistema. Foram escolhidas para analistas de negócio as professoras

Marlen Lingner Heidrich e Marina Lingner Heidrich de Carvalho. As questões abordadas foram:

- Quais são as atividades/processos de gerenciamento realizados neste Studio?
- Como são realizados?
- Quem faz?
- Qual a importância?
- Qual o resultado deste?
- Existe alguma outra informação relevante sobre esta atividade/processo?
- Quais informações sobre os funcionários e alunos são necessárias?
- Quais são as características de uma turma?
- Como funciona o controle financeiro?
- Quais são os relatórios feitos atualmente e quais são os relatórios que o sistema deve ter?
- Qual é o hardware disponível em que este sistema irá funcionar?
- Descrevam de forma simples, como vocês acham que este sistema deveria funcionar.

Com base nas respostas destas questões foram desenvolvidos os requisitos do sistema, que passaram por uma avaliação do cliente, sofrendo algumas alterações para enfim chegar a sua versão final.

3.2. REQUISITOS

Com base nas respostas das questões acima descritas foram definidos os requisitos do sistema, divididos em requisitos funcionais e requisitos não-funcionais.

3.2.1 REQUISITOS FUNCIONAIS

a) O sistema deverá incluir, editar e excluir os usuários do sistema. (Importância alta)

b) O sistema deverá incluir, editar e excluir as turmas existentes no Studio, bem como definir os alunos e o professor desta turma. (Importância alta)

c) O sistema deverá incluir, editar e excluir os alunos do Studio. (Importância alta)

d) O sistema deverá incluir, editar e excluir os professores do Studio. (Importância alta)

e) O sistema deverá incluir, editar e excluir as notas dos alunos de uma turma. (Importância variável)

f) O sistema deve permitir o registro do pagamento das mensalidades de um determinado aluno, fazendo o respectivo controle de acordo com a turma em que este está matriculado e registrando está informação para consultas e relatórios futuros. (Importância variável)

g) O sistema deve permitir o registro da freqüência de um aluno. (Importância alta)

h) O sistema deve permitir a consulta da situação do aluno, tanto em relação à freqüência quanto em relação aos pagamentos, e emitir relatório sobre isto. (Importância variável).

i) O sistema deve permitir a consulta da parte financeira, podendo verificar quanto foi arrecadado e quanto não foi pago. (Importância baixa).

j) O sistema deve permitir o controle (inserção, edição, consulta e exclusão de registros) das despesas fixas, incluindo nelas os salários dos professores e funcionários, e das despesas avulsas. (Importância baixa).

l) O sistema deverá gerar relatórios financeiros mensais contendo o quanto foi arrecadado, o quanto foi gasto e qual foi o lucro obtido. (Importância baixa).

m) As funcionalidades de controle nota e de controle de mensalidades não devem ser de uso obrigatório, assim como os relatórios referentes a estas. (importância variável).

n) O sistema deve ter controles de acesso, onde somente o usuário administrador deverá ter acesso a todo o sistema. Deverá ter também ter um usuário para os professores acessarem a parte de controle de notas e freqüência de alunos da turma pelo qual ele é responsável. Somente funcionários terão acesso ao sistema. (importância alta).

3.2.2 REQUISITOS NÃO-FUNCIONAIS

a) O sistema deve ser compatível com os browsers Internet Explorer 7 e Mozilla Firefox 3.0 (importância alta).

b) O sistema deverá ser implementado na linguagem de programação JAVA (importância alta).

c) O sistema deverá seguir o paradigma de orientação a objetos. (importância alta).

d) O sistema deverá ser de código-livre e deve ser gratuito. (importância alta).

e) O sistema deve usar o banco de dados MySQL.

f) O sistema deve possuir pelo menos uma tela para cada um dos requisitos funcionais. (importância baixa).

g) O sistema deverá oferecer a possibilidade de ser acessado via web. (importância alta).

h) O sistema deverá oferecer a possibilidade de ser acessado localmente. (importância alta).

i) O sistema deve utilizar o padrão de arquitetura em camadas. (importância alta).

4. ANÁLISE E PROJETO DO SISTEMA

Segundo Bezerra (2007:25) esta é a etapa onde é feito um estudo detalhado dos requisitos levantados na atividade anterior e a partir desse estudo, são construídos modelos para representar o sistema a ser construído.

FOWLER (2005, apud Farias, 2006:14) menciona que o RUP recomenda a utilização de casos de uso como método de organização de requisitos funcionais. Partindo deste princípio foram feitos o Diagrama de casos de uso, com auxílio da ferramenta CASE JUDE, e os casos de usos que abordam os requisitos previamente estabelecidos. Estes foram exibidos aos clientes, passaram por uma revisão e por fim foram aprovados pelo cliente.

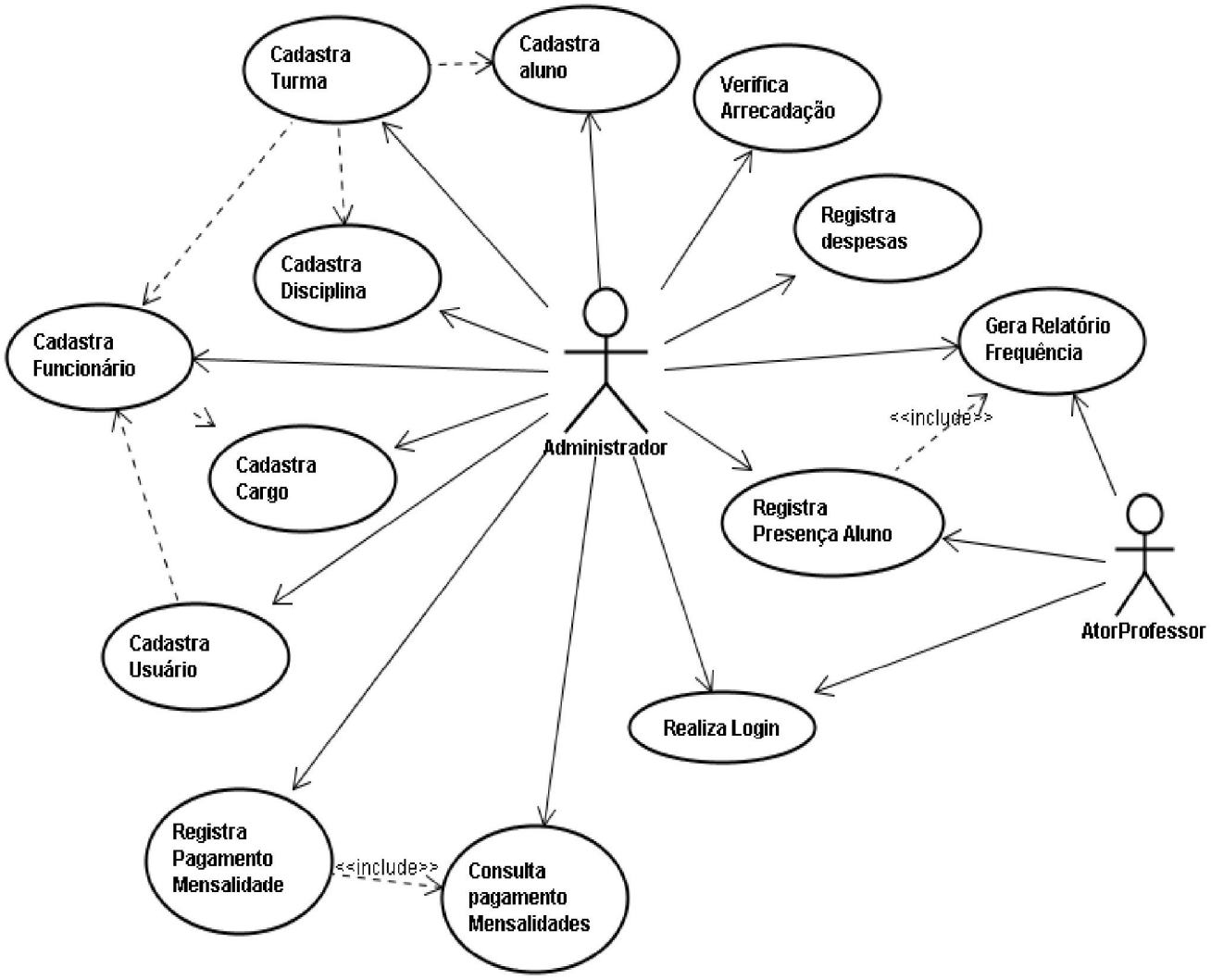


Figura 4.1 - Diagrama de Casos de Uso

Foram criados 11 casos de uso e dois Atores para o sistema (Fig.x). O ator Professor é o um indivíduo que leciona para uma ou mais turmas. O ator Administrador é um funcionário responsável pela administração do sistema, ele que fará os cadastros necessários e controlará a parte financeira, também podendo fazer registros de controle de uma turma caso haja necessidade.

No caso de uso “Cadastra Aluno”, o Administrador poderá cadastrar, consultar, alterar e excluir um aluno, assim como no “Cadastra Cargo” poderá fazer o

mesmo para um cargo e no “Cadastra Usuário” para um usuário, este último é responsável pelo cadastro dos usuários do sistema, definindo o tipo de usuário, se vai ser Administrador ou Professor. Os casos de uso “Cadastra Funcionário”, “Cadastra Usuário” e “Cadastra Turma” possuem dependência com outros casos de uso. No caso de uso “Cadastra Funcionário” é necessário ter previamente os cargos cadastrados para poder cadastrar um funcionário. No “Cadastra Usuário” existe a necessidade de ter um funcionário para cada usuário cadastrado. Já no caso de uso “Cadastra Turma” é preciso ter os alunos que fazem aquela aula (ou pelo menos parte deles) cadastrados previamente, para serem adicionados a turma. E é necessário também ter pelo menos uma disciplina e um professor cadastrados no sistema.

No caso de uso “Registra Pagamento de Mensalidades” são feitos os pagamentos das mensalidades, possui também acesso ao “Consulta Pagamento de Mensalidades” onde o usuário pode consultar os pagamentos de um aluno ou de uma turma e emitir relatório. Funcionamento parecido tem o “Registra Presença do Aluno” que tem acesso ao “Gera Relatório de Freqüência”, a diferença nestes é que tanto o professor quanto o administrador podem ter acesso.

No caso de uso “Registra Despesas” são registradas despesas como pagamentos de salários aos funcionários, compra de material, entre outras. No “Verifica Arrecadação” o administrador pose verificar o valor total arrecadado, as despesas totais e o balanço geral. Todas estas opções podem ser filtradas por ano ou por mês. O caso de uso “Cadastra Disciplinas” é um cadastro simples, com opções de inserir, editar, consultar e excluir uma disciplina.

E por fim, no caso de uso “Realiza Login”, o administrador e o professor realizam o acesso ao sistema. É este caso de uso que determina a que parte do sistema cada um terá acesso.

Este diagrama final foi obtido após a segunda iteração, tendo sido acrescentado os casos de usos “Cadastra Cargos” e “Consulta Pagamento Mensalidades”, sendo o primeiro adicionado devido a necessidade de uma maior flexibilidade na definição dos cargos antes pré-definidos. Já o segundo fazia parte do “Registra Pagamentos Mensalidades” e foi separado devido a sua complexidade e ao fato dele poder ser também acessado à parte.

Para cada um dos casos de uso foi feito um documento de detalhamento de caso de uso de acordo com o formato expandido (BEZERRA, 2007:65); segue abaixo o detalhamento do caso de uso “Cadastro Turma”, para consultar os outros detalhamentos, vide anexo.

Castra Turma (CSU04)

Sumário: Administrador utiliza sistema para cadastrar uma turma, definindo seus alunos.

Ator Primário: Administrador.

Precondições: O administrador ter realizado acesso no sistema e existirem alunos, disciplinas e professores cadastrados no sistema.

Fluxo Principal:

1. O administrador solicita a inclusão de uma nova turma.

2. O sistema libera os campos Nome Turma, Professores, com uma lista de professores, Disciplina, com uma lista de disciplina e Alunos, com uma lista de alunos, Tem Mensalidade, Dia Aula.

3. O administrador preenche o campo Nome Turma.

4. O administrador seleciona um ou mais professores da lista e escolhe a opção incluir.

5. O sistema inclui o(s) professore(s) selecionado(s) na lista Professores Escolhidos.

6. O administrador seleciona uma disciplina da lista e escolhe a opção incluir.

7. O sistema inclui a disciplina selecionada no campo Disciplina Escolhida.

8. O administrador seleciona um ou mais alunos da lista e escolhe a opção incluir.

9. O sistema inclui o(s) aluno(s) selecionado(s) na lista Alunos Escolhidos.

10. O administrador seleciona a opção Tem Mensalidade.

11. O sistema libera os campos Valor Mensalidade, Mês Inicial e Mês Final.

12. O usuário preenche os campos Valor Mensalidade, Mês Inicial e Mês Final.

13. O usuário preenche os campos Dia Aula e Duração Aula e escolhe a opção inserir.

14. O sistema inclui o Dia na lista Dias Letivos.

15. O administrador escolhe a opção Gravar.

16. O sistema grava a turma.

Fluxo Alternativo (1):

1. No passo 5, o administrador pode selecionar um ou mais professores na lista e clicar em remover.

2. O sistema remove da lista os professores escolhidos.

3. O administrador pode voltar ao passo 4, do Fluxo Principal.

Fluxo Alternativo (2):

1. No passo 7, o administrador pode escolher a opção Remover para remover a disciplina.

2. O sistema remove a disciplina escolhida.

3. O administrador pode voltar ao passo 6, do Fluxo Principal.

Fluxo Alternativo (3):

1. No passo 9, o administrador pode selecionar um ou mais alunos na lista e clicar em remover.

2. O sistema remove da lista os alunos escolhidos.

3. O administrador pode voltar ao passo 8, do Fluxo Principal.

Fluxo Alternativo (4):

1. O passo 10, o administrador pode não selecionar a opção Tem Mensalidade. Neste caso, os passos 11 e 12, devem ser ignorados.

Fluxo Alternativo (5):

1. No passo 12, o administrador pode desmarcar o campo Tem Mensalidade.

2. O sistema esconde os campos Valor Mensalidade, Mês Inicial e Mês Final, limpando seus valores.

Fluxo Alternativo (6):

1. No passo 14, o administrador poderá voltar no passo 13 e incluir um novo dia letivo.

Fluxo Alternativo (7):

1. No passo 14, o administrador poderá escolher para remover um dia letivo da lista.

Fluxo de Exceção (RN01):

O campo valor Mensalidade deve ser um campo monetário.

Fluxo de Exceção (RN02):

O campo Nome Turma é de preenchimento obrigatório. Devem ser selecionados pelo menos um professor, um aluno e uma disciplina.

Caso a opção tem mensalidade esteja marcada os campos Valor Mensalidade, Mês Inicial e Mês Final são obrigatórios.

Caso qualquer uma dessas regras não seja cumprida, o sistema deve emitir uma mensagem de aviso.

Diagrama de Classes

Com base no detalhamento dos casos de uso foram obtidas algumas classes candidatas. Estas classes foram avaliadas e destas foram retiradas as classes utilizadas no diagrama de classes da Fig. 3, que representa o caso de uso "Cadastra Turma", e nos demais diagramas de classe.

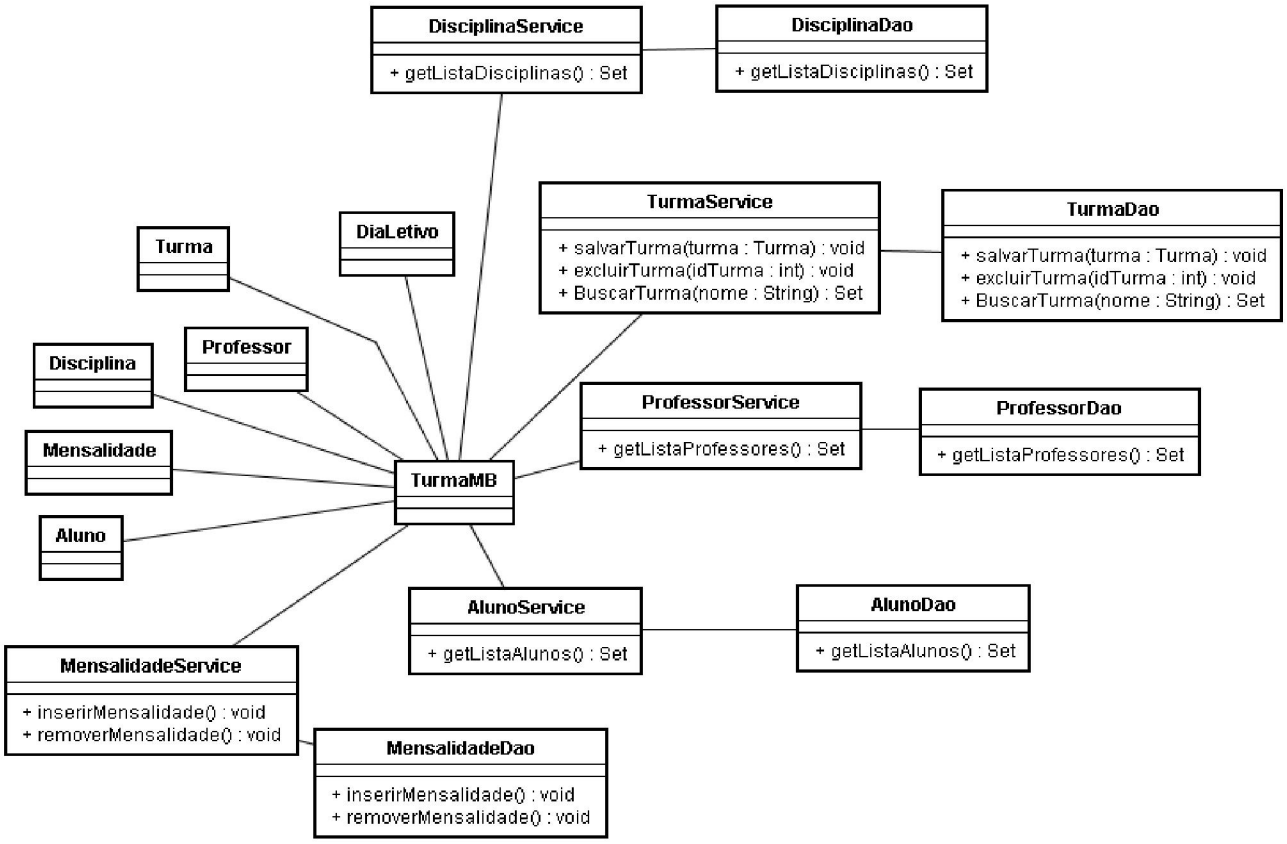


Figura 4.2 - Diagrama de Classes

5. IMPLEMENTAÇÃO

A partir dos diagramas gerados nas fases anteriores iniciou-se a fase de implementação. Primeiramente foi gerado o banco de dados da aplicação com o auxílio da ferramenta MySQL Workbench, que dado um modelo ER gera o script para criação do banco. O modelo utilizado pode ser visto na fig. 5.1. O Script Gerado está no Anexo A.

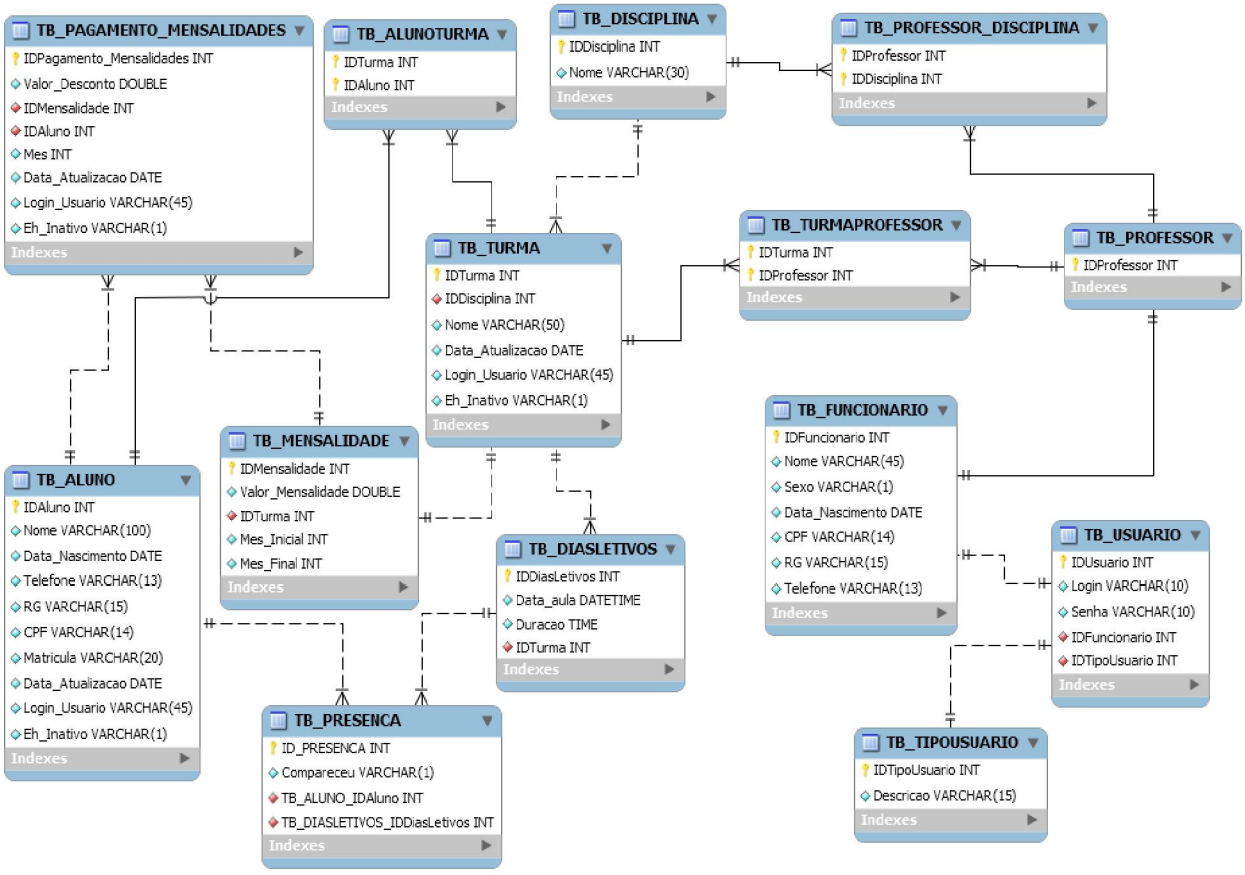


Figura 5.1 - Modelo ER

O próximo passo foi a configuração do ambiente de programação. A linguagem de programação utilizada no projeto foi Java, versão Enterprise (JEE) e a IDE de programação na qual a aplicação foi desenvolvida é o Eclipse Ganymede.

Foi utilizada uma nomenclatura para nome de classes, métodos, variáveis, para gerar um código limpo, de fácil entendimento, com qualidade e que seguisse um padrão. Na comunicação e persistência de objetos foram adotados os padrões de projeto VO[1] e DAO[2] respectivamente. A arquitetura do sistema foi construída de acordo com o Projeto Arquitetural, explicitado no Apêndice C. As páginas JSF foram geradas com auxílio da ferramenta Triangle[3] e o uso dos frameworks RichFaces[4] e MyFaces[5].

5.1 FERRAMENTAS UTILIZADAS

Para o desenvolvimento deste sistema foram utilizadas as seguintes ferramentas:

- IDE de Programação: Eclipse Ganymede.
- Banco de Dados: MySQL Server 5.0.
- Servidor de Aplicação: Tomcat 6.0.
- Ferramenta CASE: JUDE Community 1.5.2.
- Editor para modelagem de banco de dados: MySQL Workbench.
- Ferramenta de gerência de banco de dados: MySQL Administrator 1.2.
- Ferramenta gráfica de acesso ao banco de dados: MySQL Query Browser 1.2.
- Frameworks JSF: MyFaces e RichFaces.
- Ferramenta de geração de código: Triangle.

[1] http://www.mundooo.com.br/php/modules.php?name=News&file=print&sid=483
[2] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html
[3] ????
[4] http://livedemo.exadel.com/richfaces-demo/index.jsp
[5] http://myfaces.apache.org/tobago/demo.html

§ Ferramentas de design e criação de relatórios: IReport e JasperReports.

6 CONCLUSÕES

Este capítulo apresenta as conclusões obtidas no desenvolvimento deste trabalho e também os possíveis trabalhos futuros que podem ser feitos tendo este como base.

6.1 TRABALHO REALIZADO

Na primeira fase do projeto de pesquisa foi adquirido o embasamento teórico necessário para o desenvolvimento do sistema proposto. Foram estudados o processo de desenvolvimento de sistemas, a UML, o RUP e o modelo incremental, sucintamente descritos no capítulo 2, e também ferramentas para a implementação do sistema descritas no capitulo 5.

A segunda etapa do projeto foi de interação com os clientes do sistema, onde foram definidos os requisitos do sistema e a as prioridades de desenvolvimento. Nesta etapa a comunicação com o cliente foi fundamental, pois foi possível entender as reais necessidades que o sistema proposto deveria atender. Foram feitas diversas alterações até chegar ao resultado obtido, descrito no capítulo 3.

Na terceira etapa, com base no que foi obtido no capítulo 3 e estudado no capítulo 2, foi realizada a fase de análise e projeto (capítulo 4). Nesta fase foram elaborados os diagramas e casos de usos necessários para uma melhor

implementação do sistema. Aconteceram algumas mudanças nesta fase, mas, como as prioridades do sistema já estavam bem definidas, foram poucas em comparação com a fase anterior.

A quarta etapa foi a de implementação do sistema, onde foi necessário definir algumas tecnologias que iriam ser utilizadas no auxílio deste antes de passar para o trabalho prático de desenvolvimento do sistema em si. Esta etapa acabou sendo a mais longa, devido à equipe de desenvolvimento ser formada por um único membro e a quantidade de trabalho manual necessária.

6.2 ANÁLISE CRÍTICA

Neste trabalho foi possível perceber a importância de ter um forte embasamento teórico sobre os conceitos envolvidos. Somente a partir disto é possível utilizar de forma adequada e eficaz as técnicas e boas práticas pregadas pelas teorias estudas, e saber diferenciar o que pode ser aproveitado em um determinado projeto e o que não se aplica a ele.

Foi possível perceber a importância da realização de análise e projetos bem feitos e validados, pois com os problemas detectados na codificação era necessário às vezes retornar a fase de análise e projeto. Isto também mostra a importância da fase de testes, aonde os erros são detectados.

O Modelo Incremental foi muito útil para produzir um sistema mais adequado às necessidades do usuário. Pois o cliente tinha um retorno mais rápido, devido aos protótipos exibidos ao longo do processo, e o modelo também ajudava a obter uma definição de requisitos mais detalhada, já que o número de requisitos a ser estudado em cada iteração é menor. Nesta parte de requisitos o uso de alguns

conceitos do RUP também foi muito válido, como a utilização de casos de uso como método de organização dos requisitos funcionais.

O uso dos diagramas da UML foi fundamental para definir a modelagem do sistema e o processo de desenvolvimento. Através destes diagramas foi possível identificar quais eram as maiores dificuldades que seriam encontradas na fase de implementação do sistema, e simplificar este processo de desenvolvimento, diminuindo essencialmente as dúvidas encontradas nesta fase e reduzindo a necessidade de retornos a fase de análise.

Ao ser analisado o sistema demonstrou-se adequado as necessidades do cliente, e mostrou que o uso de práticas consagradas facilita amplamente a produção de um sistema, uma vez que é muito mais fácil andar por caminhos já trilhados e que rumam à direção certa.

6.3 TRABALHOS FUTUROS

Uma possibilidade de trabalho futuro relativo a este é a ampliação do sistema, oferecendo a possibilidade de acessos aos alunos, com espaços para atividades como conferir notas, tirar dúvidas, confraternizar com outros alunos, armazenar arquivos relativos às disciplinas praticadas, transformando o sistema em um portal, onde o aluno possa interagir obtendo um maior aproveitamento da disciplina oferecida. Isto também seria válido para os professores, oferecendo a eles uma maior possibilidade de divulgação dos conceitos envolvidos na sua disciplina, além de mais interação com seus alunos.

Também poderiam ser feitos estudos de outras metodologias de desenvolvimento, onde o sistema seria refeito utilizando outra abordagem (Utilizando XP ou Scrum, por exemplo).

Um estudo comparativo com outros sistemas também seria uma idéia válida, tendo em vista que ferramentas como o Moodle[6] e o Sistema de Gerenciamento Escolar (SGE[7]) possuem funcionalidade semelhante que poderiam ser adaptadas as necessidades deste sistema.

[6] Moodle - http://moodle.org/
[7] SGE - http://www.meusdownloads.com.br/p.jsp?ppID=d155

BIBLIOGRAFIA

CHIAVENATO, Idalberto. **Introdução à Teoria Geral da Administração**. 6ª ed. Rio de Janeiro: Campus, 2000. 700 p.;

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**. 2ª ed. Rio de Janeiro: Brasport, 2002.358 p.;

FARIAS, Tiago Moraes de Miranda. **Aplicação de Padrões ao Processo de Desenvolvimento de Software RUP.** Recife, 2006;

FALBO, Ricardo de Almeida. **A Experiência na Definição de um Processo Padrão Baseado no Processo Unificado**. Vitória, 2000;

OLIVEIRA, Alexandre Contento de & SANCHES, Erineide Ross. **Planejamento Estratégico em Organizações Sem Fins Lucrativos**. 2003;

ROCHA, Camila Ribeiro. **Estudo do Modelo Incremental para Desenvolvimento de Software Utilizando o Paradigma de Orientação a Objetos**. Projeto Final de Curso (Monografia) - Universidade Federal de Goiás, Instituto de Informática, Goiânia. 2002;

MOREIRA, Vilmar Rodrigues & PROTIL, Roberto Max. **Levantamento de requisitos de um sistema de informações voltado ao apoio à gestão logística hospitalar: O caso da Santa Casa de Misericórdia de Curitiba/Aliança Saúde.**

LUSTOSA, Rodrigo Bastos & NÓBREGA, Diogo Lins. **PROCESSO DE DESENVOLVIMENTO DE SISTEMAS: a Fase de Análise de Requisitos como diferencial de Qualidade na produção de Sistemas de Informação.** João Pessoa. 2007;

AMORIM, Rainer Xavier de, PASSOS,' Odette Mestrinho & FILHO, Luiz Guilherme. **Análise E Modelagem De Um Sistema Web Para Reservas De Laboratórios Da Escola Superior De Tecnologia – est.** 2007

SILVA, Douglas Marcos da. **UML - Guia De Consulta Rápida**. Ed. Novatec. São Paulo. 2001

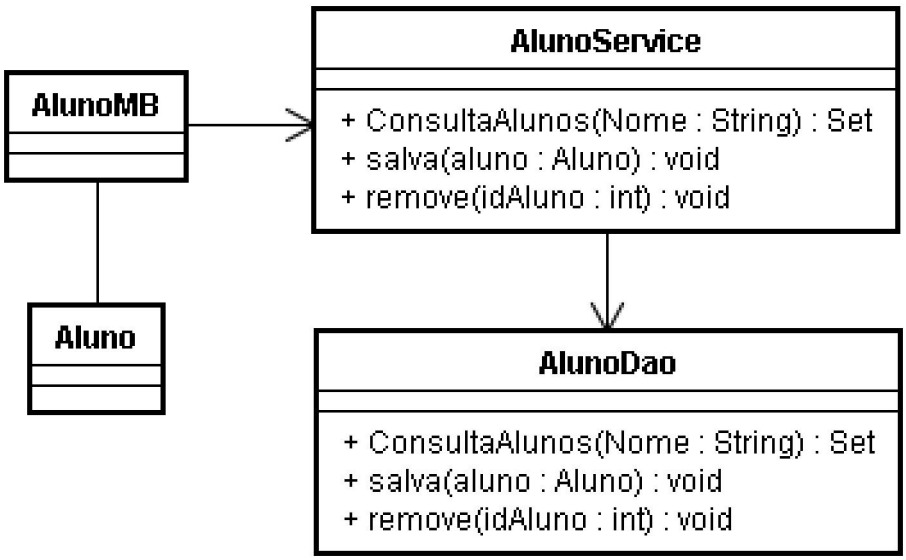
BEZERRA, Eduardo. **Princípios De Análise E Projeto De Sistemas Com UML**. Ed. Campus. 2006

MODESTO, Lisandro Rogério. **Teste Funcional Baseado em Diagrama da UML**. Marília. 2006.

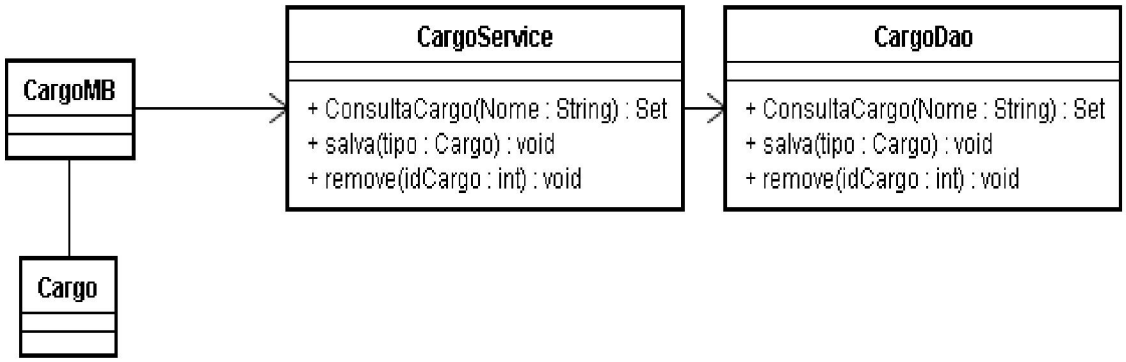
APÊNDICE A

CASOS DE USO

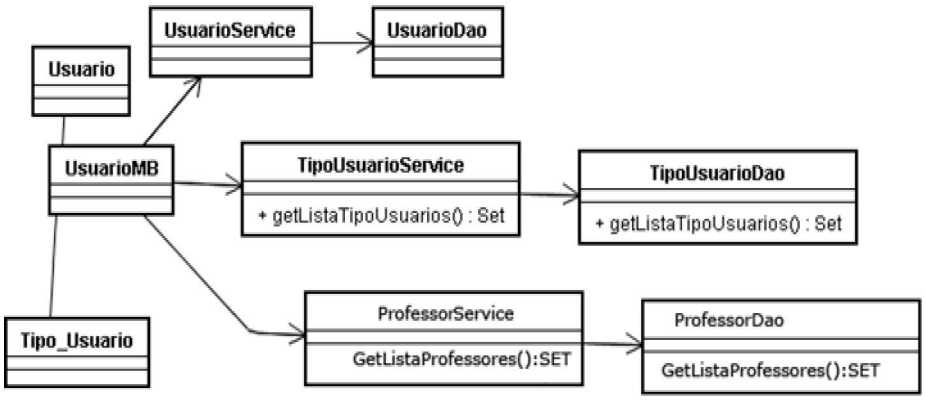
Cadastra Aluno:



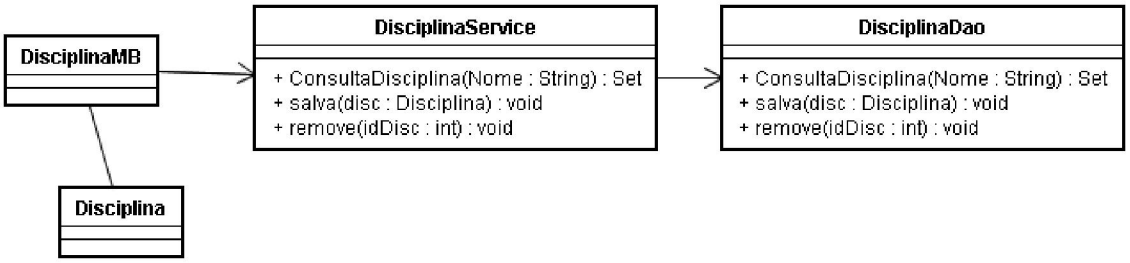
Cadastra Cargo



Cadastra Usuário:



Cadastra Disciplina:



APÊNDICE B

DIAGRAMAS DE CLASSE

Casos de Uso

Castra Usuário (CSU05)

Sumário: Administrador utiliza sistema para cadastrar um usuário, definindo o tipo de usuário.

Ator Primário: Administrador.

Precondições: O administrador ter realizado acesso no sistema, deve existir um funcionário que não esteja associado a um usuário, registrado previamente no sistema, e as constantes tipo usuário (Administrador e Professor) devem estar registrados na tabela tipo_Usuário.

Fluxo Principal:

1. O administrador solicita a inclusão de um novo usuário.

2. O sistema libera os campos Login, o campo Funcionários com uma lista de Funcionários, o campo Senha, o campo Confirma Senha e o campo tipo Usuário com as opções Administrador e Funcionário.

3. O administrador preenche o campo Login.

4. O administrador seleciona um funcionário da lista.

5. O sistema inclui o nome do funcionário selecionado no campo Nome Usuário.

6. O administrador preenche os campos Senha e Confirma Senha.

7. O administrador seleciona um dos tipos de usuário da lista.

8. O administrador seleciona a opção Salvar.

9. O sistema salva o usuário.

Fluxo Alternativo (1):

1. No passo 1, o administrador pode escolher a opção consultar.

2. O sistema exibe a tela de consulta com os campos Login e Nome Usuário.

3. O administrador preenche um ou mais campos da tela de consulta e escolhe a opção pesquisar.

4. O sistema retorna os dados dos usuários encontrados.

5. O administrador seleciona a opção Editar.

6. O sistema retorna a tela de cadastro de usuário com os campos preenchidos com os valores correspondentes ao usuário selecionado, em modo de edição.

7. Retorna ao passo 2 do Fluxo Principal.

Fluxo Alternativo (2):

1. No passo 5, do Fluxo Alternativo 1, o administrador pode escolher a opção Remover para remover o usuário.

2. O sistema remove o usuário escolhido.

3. O administrador pode voltar ao passo 1, do Fluxo Principal.

Fluxo de Exceção (RN01):

O valor do campo Senha e o valor do Campo Login devem ser o mesmo. Caso o administrador digite valores diferentes, o sistema deve exibir a mensagem: “os campos Senha e Confirmar Senha devem ser iguais!”.

Fluxo de Exceção (RN02):

O sistema só deverá exibir funcionários que não estejam associados a um outro usuário.

Fluxo de Exceção (RN03):

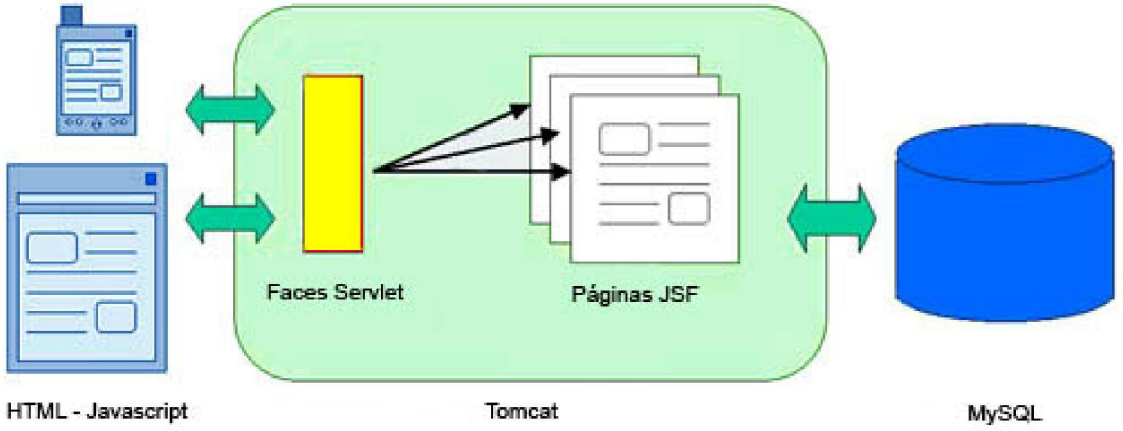
Não é permitida a alteração do campo Nome Funcionário e do campo Login.

Fluxo de Exceção (RN04):

Os campos Login, Nome Usuário, Senha, Confirma Senha e Tipo Usuário são de preenchimento obrigatório. Caso algum deles não seja preenchido o sistema deve exibir a mensagem: “o campo X é de preenchimento obrigatório!”.

APÊNDICE C

PROJETO ARQUITETURAL



Descrição

A arquitetura utilizada segue o padrão n-camadas, que separa a apresentação, as camadas de negócio e a camada de persistência de dados. De acordo com o padrão definido, as camadas serão implementadas em tecnologias Web, abertas e open source.

Na interface serão utilizadas páginas JSF com uso de Javascript e Ajax para operações de validação de entrada e para direcionar algumas ações implementadas. A arquitetura utiliza tecnologia Web, que funcionará tanto em ambientes intranets como em ambiente internet.

A seguir estão descritas características do serviço de hospedagem de páginas:

- Servidor Web e servidor Java Server Faces: Tomcat.
- Páginas JSF: as requisições do browser se destinarão a páginas JSF, que obedecerão a algumas restrições de conteúdo.

- Nenhuma regra de negócio ou acesso a dados será realizada em código dentro das páginas JSF. Toda lógica da aplicação, bem como acesso a dados, deve ser implementada em componentes JSF Managed Beans a serem referenciados nas páginas JSF;
- A persistência dos objetos deve ser implementada sob o padrão de projeto DAO (http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html);
- A comunicação entre as camadas deve ser implementada sob o padrão de projeto VO (http://www.mundooo.com.br/php/modules.php?name=News&file=print&sid=483);
- Será utilizado o SGBD relacional MySQL (http://dev.mysql.com/downloads/mysql/5.0.html), por ser um software open source e capaz de suportar os requisitos existentes no desenvolvimento da aplicação.
- Para a criação dos relatórios será utilizada a ferramenta IReport 3.1.1 (http://sourceforge.net/project/showfiles.php?group_id=64348) em conjunto com o Jasper Reports 3.0.1
- (http://sourceforge.net/project/showfiles.php?group_id=36382&package_id=28579&release_id=618591&abmode=1).

ANEXO A - SCRIPT SQL PARA GERAÇÃO DO BANCO DE DADOS

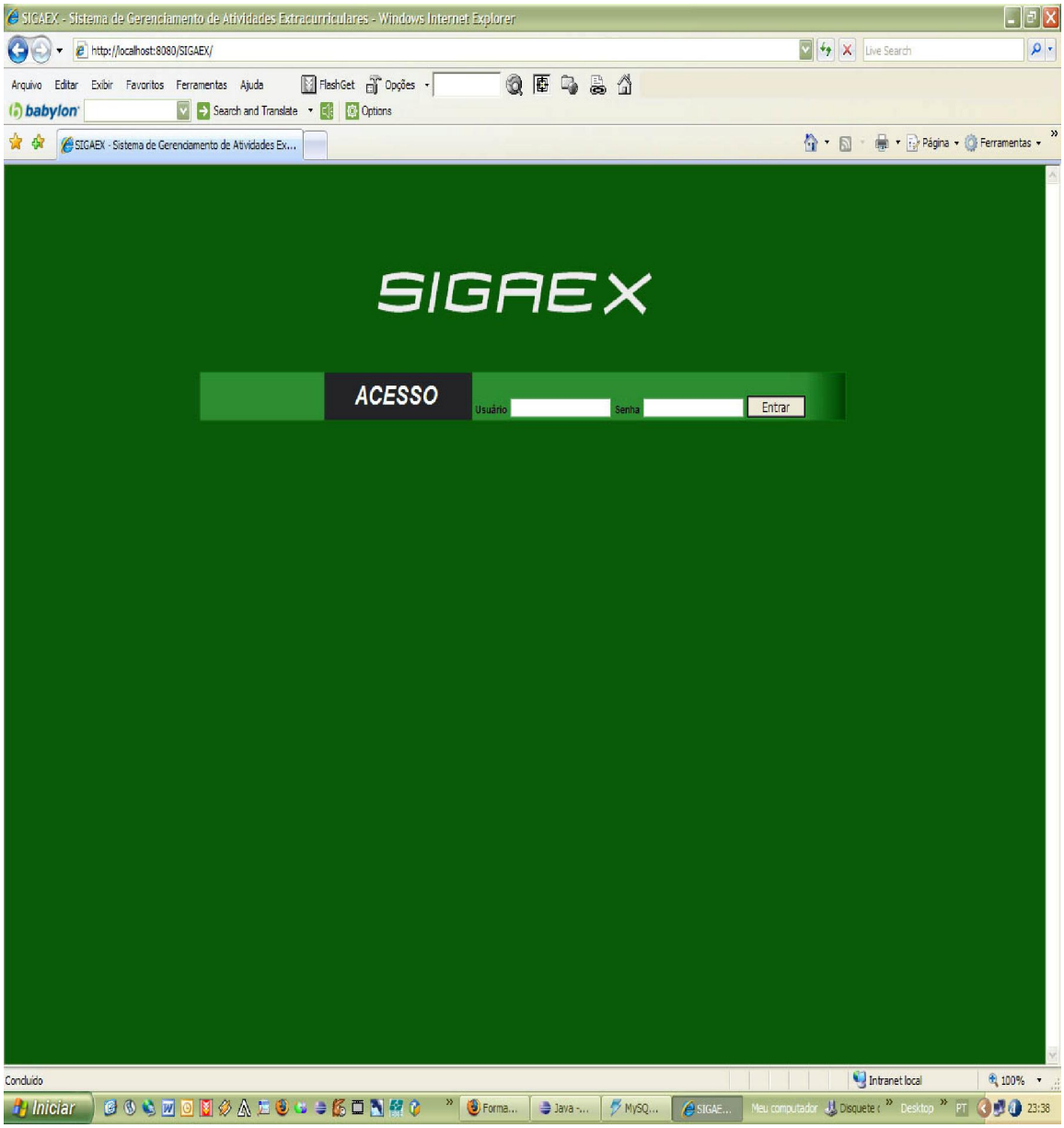
```sql
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL';

CREATE SCHEMA IF NOT EXISTS `SIGAEX` DEFAULT CHARACTER SET latin1 COLLATE latin1_swedish_ci ;
USE `SIGAEX`;

-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_ALUNO`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_ALUNO` (
  `IDAluno` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Nome` VARCHAR(100) NOT NULL ,
  `Data_Nascimento` DATE NOT NULL ,
  `Telefone` VARCHAR(13) NULL ,
  `RG` VARCHAR(15) NOT NULL ,
  `CPF` VARCHAR(14) NULL ,
  `Matricula` VARCHAR(20) NOT NULL ,
  `Data_Atualizacao` DATE NULL ,
  `Login_Usuario` VARCHAR(45) NULL ,
  `Eh_Inativo` VARCHAR(1) NOT NULL DEFAULT 'N' ,
  PRIMARY KEY (`IDAluno`) ,
  INDEX `TB_Aluno_Nome` (`Nome` ASC) )
AUTO_INCREMENT = 1
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_DISCIPLINA`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_DISCIPLINA` (
  `IDDisciplina` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Nome` VARCHAR(30) NOT NULL ,
  PRIMARY KEY (`IDDisciplina`) )
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_TURMA`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_TURMA` (
  `IDTurma` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `IDDisciplina` INT UNSIGNED NOT NULL,
  `Nome` VARCHAR(50) NOT NULL ,
  `Data_Atualizacao` DATE NULL ,
  `Login_Usuario` VARCHAR(45) NULL ,
  `Eh_Inativo` VARCHAR(1) NULL DEFAULT 'N' ,
  PRIMARY KEY (`IDTurma`) ,
  INDEX `TB_Turma_Nome` (`Nome` ASC) ,
  CONSTRAINT `fk_{4B5F7B75-42A7-4A59-81FC-0E73FC061680}`
    FOREIGN KEY (`IDDisciplina` )
    REFERENCES `SIGAEX`.`TB_DISCIPLINA` (`IDDisciplina` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;
```

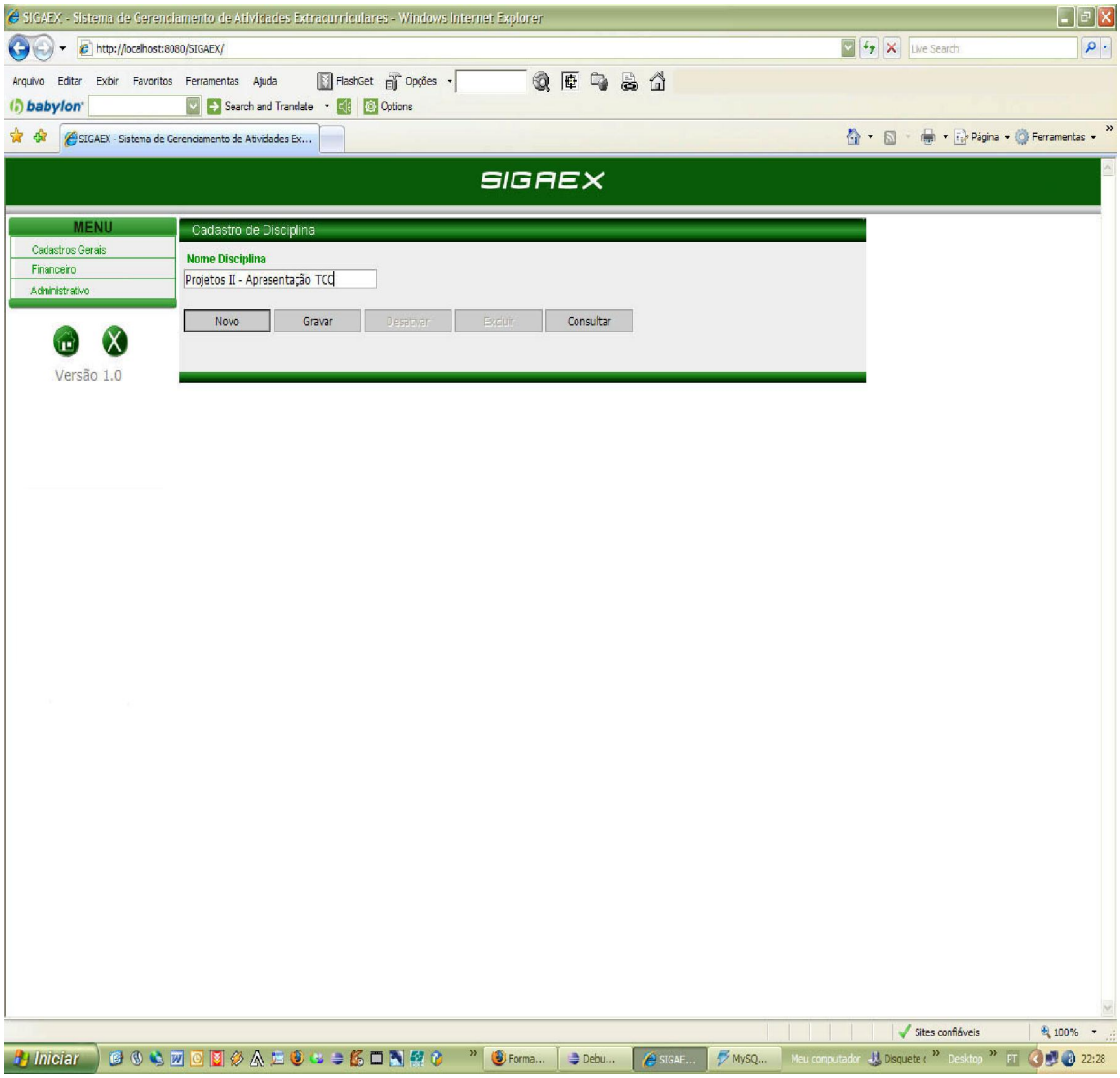
```
-- Table `SIGAEX`.`TB_ALUNOTURMA`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_ALUNOTURMA` (
  `IDTurma` INT UNSIGNED NOT NULL ,
  `IDAluno` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`IDTurma`, `IDAluno`) ,
  INDEX `TB_Aluno_has_TB_Turma_FKIndex1` (`IDAluno` ASC) ,
  INDEX `TB_Aluno_has_TB_Turma_FKIndex2` (`IDTurma` ASC) ,
  CONSTRAINT `FK_ALUNO`
    FOREIGN KEY (`IDAluno` )
    REFERENCES `SIGAEX`.`TB_ALUNO` (`IDAluno` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `FK_TURMA`
    FOREIGN KEY (`IDTurma` )
    REFERENCES `SIGAEX`.`TB_TURMA` (`IDTurma` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_DIASLETIVOS`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_DIASLETIVOS` (
  `IDDiasLetivos` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Data_aula` DATETIME NOT NULL ,
  `Duracao` TIME NOT NULL ,
  `IDTurma` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`IDDiasLetivos`) ,
  INDEX `fk_TB_DIASLETIVOS_TB_TURMA` (`IDTurma` ASC) ,
  CONSTRAINT `fk_TB_DIASLETIVOS_TB_TURMA`
    FOREIGN KEY (`IDTurma` )
    REFERENCES `SIGAEX`.`TB_TURMA` (`IDTurma` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_FUNCIONARIO`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_FUNCIONARIO` (
  `IDFuncionario` INT NOT NULL AUTO_INCREMENT ,
  `Nome` VARCHAR(45) NOT NULL ,
  `Sexo` VARCHAR(1) NOT NULL ,
  `Data_Nascimento` DATE NOT NULL ,
  `CPF` VARCHAR(14) NOT NULL ,
  `RG` VARCHAR(15) NULL ,
  `Telefone` VARCHAR(13) NULL ,
  PRIMARY KEY (`IDFuncionario`) ,
  INDEX `IndNome` (`Nome` ASC) )
ENGINE = InnoDB;

```

```
-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_PROFESSOR`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_PROFESSOR` (
  `IDProfessor` INT NOT NULL ,
  PRIMARY KEY (`IDProfessor`) ,
  INDEX `IDProfessor` (`IDProfessor` ASC) ,
  CONSTRAINT `IDProfessor`
    FOREIGN KEY (`IDProfessor` )
    REFERENCES `SIGAEX`.`TB_FUNCIONARIO` (`IDFuncionario` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;

-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_TIPOUSUARIO`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_TIPOUSUARIO` (
  `IDTipoUsuario` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Descricao` VARCHAR(15) NULL ,
  PRIMARY KEY (`IDTipoUsuario`) ,
  INDEX `index1` (`Descricao` ASC) )
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;

-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_USUARIO`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_USUARIO` (
  `IDUsuario` INT UNSIGNED NOT NULL AUTO_INCREMENT ,
  `Login` VARCHAR(10) NOT NULL ,
  `Senha` VARCHAR(10) NOT NULL ,
  `IDFuncionario` INT NOT NULL ,
  `IDTipoUsuario` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`IDUsuario`) ,
  INDEX `fk_TB_FUNCIONARIO` (`IDFuncionario` ASC) ,
  INDEX `fk_TB_USUARIO_TB_TIPOUSUARIO` (`IDTipoUsuario` ASC) ,
  CONSTRAINT `fk_TB_FUNCIONARIO`
    FOREIGN KEY (`IDFuncionario` )
    REFERENCES `SIGAEX`.`TB_FUNCIONARIO` (`IDFuncionario` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TB_USUARIO_TB_TIPOUSUARIO`
    FOREIGN KEY (`IDTipoUsuario` )
    REFERENCES `SIGAEX`.`TB_TIPOUSUARIO` (`IDTipoUsuario` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
PACK_KEYS = 0
ROW_FORMAT = DEFAULT;

-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_MENSALIDADE`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_MENSALIDADE` (
  `IDMensalidade` INT NOT NULL AUTO_INCREMENT ,
  `Valor_Mensalidade` DOUBLE NOT NULL ,
  `IDTurma` INT UNSIGNED NOT NULL ,
  `Mes_Inicial` INT NOT NULL ,
  `Mes_Final` INT NOT NULL ,
  PRIMARY KEY (`IDMensalidade`) ,
  INDEX `fk_TB_Turma` (`IDTurma` ASC) ,
  CONSTRAINT `fk_TB_Turma`
    FOREIGN KEY (`IDTurma` )
    REFERENCES `SIGAEX`.`TB_TURMA` (`IDTurma` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```sql
-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_PAGAMENTO_MENSALIDADES`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_PAGAMENTO_MENSALIDADES` (
  `IDPagamento_Mensalidades` INT NOT NULL ,
  `Valor_Desconto` DOUBLE NULL ,
  `IDMensalidade` INT NOT NULL ,
  `IDAluno` INT UNSIGNED NOT NULL ,
  `Mes` INT NOT NULL ,
  `Data_Atualizacao` DATE NULL ,
  `Login_Usuario` VARCHAR(45) NULL ,
  `Eh_Inativo` VARCHAR(1) NULL DEFAULT 'N' ,
  PRIMARY KEY (`IDPagamento_Mensalidades`) ,
  INDEX `fk_Mensalidade` (`IDMensalidade` ASC) ,
  INDEX `fk_TB_Aluno` (`IDAluno` ASC) ,
  CONSTRAINT `fk_Mensalidade`
    FOREIGN KEY (`IDMensalidade` )
    REFERENCES `SIGAEX`.`TB_MENSALIDADE` (`IDMensalidade` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE,
  CONSTRAINT `fk_TB_Aluno`
    FOREIGN KEY (`IDAluno` )
    REFERENCES `SIGAEX`.`TB_ALUNO` (`IDAluno` )
    ON DELETE RESTRICT
    ON UPDATE CASCADE)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_TURMAPROFESSOR`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_TURMAPROFESSOR` (
  `IDTurma` INT UNSIGNED NOT NULL ,
  `IDProfessor` INT NOT NULL ,
  PRIMARY KEY (`IDTurma`, `IDProfessor`) ,
  INDEX `fk_TB_TURMA_has_TB_PROFESSOR_TB_TURMA` (`IDTurma` ASC) ,
  INDEX `fk_TB_TURMA_has_TB_PROFESSOR_TB_PROFESSOR` (`IDProfessor` ASC) ,
  CONSTRAINT `fk_TB_TURMA_has_TB_PROFESSOR_TB_TURMA`
    FOREIGN KEY (`IDTurma` )
    REFERENCES `SIGAEX`.`TB_TURMA` (`IDTurma` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TB_TURMA_has_TB_PROFESSOR_TB_PROFESSOR`
    FOREIGN KEY (`IDProfessor` )
    REFERENCES `SIGAEX`.`TB_PROFESSOR` (`IDProfessor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);


-- -----------------------------------------------------
-- Table `SIGAEX`.`TB_PROFESSOR_DISCIPLINA`
-- -----------------------------------------------------
CREATE  TABLE IF NOT EXISTS `SIGAEX`.`TB_PROFESSOR_DISCIPLINA` (
  `IDProfessor` INT NOT NULL ,
  `IDDisciplina` INT UNSIGNED NOT NULL ,
  PRIMARY KEY (`IDProfessor`, `IDDisciplina`) ,
  INDEX `fk_TB_PROFESSOR_has_TB_DISCIPLINA_TB_PROFESSOR` (`IDProfessor` ASC) ,
  INDEX `fk_TB_PROFESSOR_has_TB_DISCIPLINA_TB_DISCIPLINA` (`IDDisciplina` ASC) ,
  CONSTRAINT `fk_TB_PROFESSOR_has_TB_DISCIPLINA_TB_PROFESSOR`
    FOREIGN KEY (`IDProfessor` )
    REFERENCES `SIGAEX`.`TB_PROFESSOR` (`IDProfessor` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_TB_PROFESSOR_has_TB_DISCIPLINA_TB_DISCIPLINA`
    FOREIGN KEY (`IDDisciplina` )
    REFERENCES `SIGAEX`.`TB_DISCIPLINA` (`IDDisciplina` )
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

ANEXO B - IMAGENS DO SISTEMA




SIGAEX - Sistema de Gerenciamento de Atividades Extracurriculares - Windows Internet Explorer
http://localhost:8080/SIGAEX/
SIGAEX
MENU
Cadastros Gerais
Financeiro
Administrativo
Versão 1.0
Cadastro de Disciplina
Nome Disciplina
Projetos II - Apresentação TCC
Novo
Gravar
Desativar
Excluir
Consultar
Iniciar

Autor

Mário Heidrich Vicentim

Resumo

O seguinte artigo busca introduzir alguns conceitos a respeito do desenvolvimento do sistema SIGAEX

Abstract

The following article has the purpose to introduce concepts about the development of SIGAEX system.

1. Objetivo do sistema:

O SIGAEX tem por objetivo ser uma ferramenta gratuita para auxiliar o gerenciamento de turmas e controle financeiro de atividades extracurriculares.

Foi criado baseado na identificação da necessidade que alguns cursos e escolas têm por sistemas os quais organizem suas atividades de uma forma que torne mais fácil a administração destes e que também auxilie o combate a fraudes.

2. Planejamento do sistema

O sistema foi planejado seguindo o paradigma de orientação a objetos, e baseado nos moldes da engenharia de software. Foram utilizadas também algumas noções de UML, RUP e do Modelo Incremental.

Para auxiliar no planejamento do sistema foram realizadas entrevistas com professoras do Studio de Dança do Instituto Estadual de Educação, da onde foram retirados os requisitos do sistema.

3 Análise do sistema

Para o desenvolvimento deste sistema foram criados 11 casos de uso e dois Atores. Para cada um dos casos de uso foi feito um documento de detalhamento de caso de uso de acordo com o formato expandido. Com base no detalhamento dos casos de uso foram obtidas as classes utilizadas no sistema.

4 Implementação

A partir dos diagramas gerados nas fases anteriores foi gerado o banco de dados da aplicação com o auxílio da ferramenta MySQL Workbench.

O próximo passo foi a configuração do ambiente de programação. A linguagem de programação utilizada no projeto foi Java, versão Enterprise (JEE) e a IDE de programação na qual a aplicação foi desenvolvida é o Eclipse Ganymede.

Foi utilizada uma nomenclatura para nome de classes, métodos, variáveis, para gerar um código limpo, de fácil entendimento, com qualidade e que seguisse um padrão. Na comunicação e persistência de objetos foram adotados os padrões de projeto VO[1] e DAO[2] respectivamente. A arquitetura do sistema foi construída de acordo com o Projeto Arquitetural, explicitado no Apêndice C. As páginas JSF foram geradas com auxílio da ferramenta Triangle[3] e o uso dos frameworks RichFaces[4] e MyFaces[5].

5 Ferramentas Utilizadas

Para o desenvolvimento deste sistema foram utilizadas as seguintes ferramentas:

- IDE de Programação: Eclipse Ganymede.
- Banco de Dados: MySQL Server 5.0.
- Servidor de Aplicação: Tomcat 6.0.
- Ferramenta CASE: JUDE Community 1.5.2.

---

[1] http://www.mundooo.com.br/php/modules.php?name=News&file=print&sid=483
[2] http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html
[3] ????
[4] http://livedemo.exadel.com/richfaces-demo/index.jsp
[5] http://myfaces.apache.org/tobago/demo.html

§ Editor para modelagem de banco de dados: MySQL Workbench.

§ Ferramenta de gerência de banco de dados: MySQL Administrator 1.2.

§ Ferramenta gráfica de acesso ao banco de dados: MySQL Query Browser 1.2.

§ Frameworks JSF: MyFaces e RichFaces.

§ Ferramenta de geração de código: Triangle.

§ Ferramentas de design e criação de relatórios: IReport e JasperReports.