PAULO HENRIQUE CONSOLI

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO
SISTEMA DE GERENCIAMENTO ACADÊMICO
DECISION

ASSIS
2009

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO SISTEMA DE GERENCIAMENTO ACADÊMICO DECISION

PAULO HENRIQUE CONSOLI

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

Orientador:_____

Analisador (1):_____

Analisador (2):_____

Assis
2009

PAULO HENRIQUE CONSOLI

DESENVOLVIMENTO DE SOFTWARE ACADÊMICO SISTEMA DE GERENCIAMENTO ACADÊMICO DECISION

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

Orientador:_____

Área de Concentração:_____

_____

Assis
2009

DEDICATÓRIA

À minha filha pelo esforço, dedicação e compreensão em todos os momentos desta e caminhadas.

Aos colegas do curso de Tecnologia em Processamento de Dados que demonstraram companheirismo durante a sofrida caminhada ao longo desses anos.

Em especial, a minha esposa e melhor amiga Eliane que acreditou e continua acreditando em mim.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois sem Ele, nada seria possível e não estaria aqui, desfrutando destes momentos que nos são tão importantes.

Aos professores do curso de Tecnologia em Processamento de Dados da FEMA pela paciência e dedicação ao longo desses anos.

Ao Prof. Talo, pela paciência, pelo apoio e pelo acompanhamento desse trabalho.

Ao Prof. Almir, pelas longas horas de ensinamento da ferramenta utilizada para o desenvolvimento desse trabalho.

RESUMO

Este trabalho tem como finalidade colocar em prática todo conhecimento adquirido ao longo do curso, visando à criação de um sistema de gerenciamento acadêmico, com o intuito de disponibilizar um cadastro atualizado de alunos, professores e disciplinas, assim como, facilitar o processo de matriculas, possibilitando um acompanhamento preciso de toda vida acadêmica dos alunos.

A análise é feita utilizando modelagem UML. Toda a parte de programação foi desenvolvida com a utilização das ferramentas Visual C – Sharp, Banco de Dados SQL – Server.

Palavras- chave: C – Sharp. Gerenciamento. Acadêmico.

ABSTRACT

This work hás as purpose to put in practice every acquired knowledge along the course, seeking to the creation of a system of academic administration, with the intention of marking available na updated register of students, teachers and disciplines, as well as, to facilitate the processo f you register, marking possible a necessary attendance of a lifetime academic of the students.

The analysis is made using modelling UML. The whole programming part was developed with tools Visual C'S use Sharp, Banco SQL Server.

Keywords: C. Sharp. Administration. Academic.

LISTA DE LUSTRAÇOES

LISTA DE ABREVIATURAS

CLR	-	Common Language Runtime
MSIL	-	Microsoft Intermediate Language
JIT	-	Just In Time
CLS	-	Common Language Specification
C#	-	C Sharp
SQL	-	Structured Query Language

SUMÁRIO

1. INTRODUÇÃO

O Sistema Acadêmico DECISION, desenvolvido para Web, tem como principal objetivo controlar eficientemente toda a área acadêmica e pedagógica dos cursos, bem como a emissão de relatórios gerenciais e de controles, com a finalidade de facilitar a administração da escola, tornando – a mais eficaz e trazendo mais benefícios para si e para os alunos.

O sistema será implementado buscando atender todas as necessidades da escola, inclusive possibilitando eventuais atualizações, ou seja, fornecer compatibilidade para inclusão de novas funções, emissão de novos relatórios e até mesmo alguma modificação referente à padronização da escola.

O sistema será desenvolvido com a ferramenta “Visual C - Sharp (C#)”, permitindo elaborar uma interface de fácil manuseio e agradável para o usuário podendo aplicar-se na elaboração do *layout* do sistema. Para armazenamento das informações, será utilizado o aplicativo de Banco de Dados “SQL Server”, o qual caracteriza-se com uma interface simples, objetiva, atendendo com total eficiência e exatidão a implementação do sistema. Para gerar os relatórios será utilizada a ferramenta “Crystal Reports 8.0”, que oferece uma boa visualização dos dados.

2. OBJETIVOS

O objetivo principal deste sistema visa disponibilizar um cadastro atualizado dos alunos, professores e disciplinas , assim como, agilizar o processo de matricula, possibilitando um acompanhamento preciso de toda vida acadêmica dos alunos.

Espera-se também que este sistema contribua de forma significativa para as futuras decisões tomadas pelos dirigentes da escola. O sistema fornecerá controles detalhados e preciso, além de relatórios eficientes e de fácil visualização, para que tarefas rotineiras tornem-se menos cansativas e com menor probabilidade de erros.

3. USUÁRIOS DO SISTEMA

O sistema deverá possuir um modulo de segurança que devera ser responsável pela autenticação de usuários, identificando a visão diferenciada que cada um deverá ter quanto a funcionalidades do sistema de Controle Acadêmico.

Usuário Diretoria: único autorizado a efetuar alterações cadastrais, seja de inclusão, exclusão ou atualização de informações sobre as entidades do sistema e acesso a todas as informações contidas no sistema (aluno, professor ou disciplinas).

Usuário Secretaria: responsável para efetuar todos os cadastros e lançamentos de notas, faltas,etc.

Usuário Aluno: receberá identificação única, poderá consultar informações a respeito de sua vida acadêmica ou informações a respeito das disciplinas.

4. LISTA DE FUNÇÕES

O sistema deverá possibilitar:

Cadastro de Usuários
Cadastro de aluno;
Cadastro de professor;
Cadastro de matérias;
Cadastro de curso;
Cadastro de apostila;
Cadastro de nota;
Cadastro de faltas;

Relatório de notas
Relatório de apostilas;
Relatório de faltas;
Relatório de matéria;
Relatório de professor;
Relatório de curso;

5. LISTA DE EVENTOS

N°	Descrição	Evento	Use case	Resposta
1	Diretoria cadastra novo usuário	Cadastro de usuário	Cadastrar usuário	Mgs 1
2	Secretaria cadastra aluno	Cadastro de aluno	Cadastrar aluno	Mgs 2
3	Secretaria cadastra professor	Cadastro de professor	Cadastrar professor	Mgs 3
4	Secretaria cadastra matéria	Cadastro de matéria	Cadastrar matéria	Mgs 4
5	Secretaria cadastra curso	Cadastro de curso	Cadastrar curso	Mgs 5
6	Secretaria cadastra apostila	Cadastro de apostila	Cadastrar apostila	Mgs 6
7	Secretaria cadastra nota	Cadastro de nota	Cadastrar nota	Mgs 7
8	Secretaria cadastra falta	Cadastro de falta	Cadastrar falta	Mgs 8
9	Aluno solicita relatório de notas	Geração relatório de notas	Gerar relatório de notas	Mgs 9
10	Aluno solicita relatório de apostilas	Geração relatório de apostilas	Gerar relatório de apostilas	Mgs 10
11	Aluno solicita relatório de faltas	Geração relatório de faltas	Gerar relatório de faltas	Mgs 11
12	Diretoria solicita relatório de matérias	Geração relatório de matérias	Gerar relatório de matérias	Mgs 12
13	Diretoria solicita relatório de professores	Geração relatório de professor	Gerar relatório de professor	Mgs 13
14	Diretoria solicita relatório de cursos	Geração relatório de cursos	Gerar relatório de curso	Mgs 14

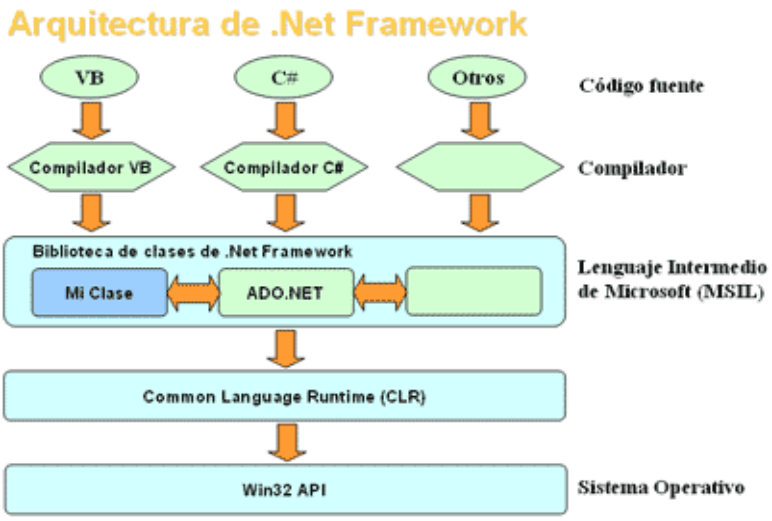
6. PLATAFORMA E LINGUAGEM DE IMPLEMENTAÇÃO

6.1 Plataforma Microsoft.NET

A nova tecnologia de Microsoft oferece soluções aos problemas de programação atuais, como são a administração de código ou a programação para Internet. Para aproveitar ao máximo as características de .Net é necessário entender a arquitetura básica na que está implementada esta tecnologia e assim se beneficiar de todas as características que oferece esta nova plataforma.

O Framework de .Net é uma infra-estrutura sobre a que se reúne todo um conjunto de linguagens e serviços que simplificam enormemente o desenvolvimento de aplicações. Mediante esta ferramenta se oferece um ambiente de execução altamente distribuído, que permite criar aplicações robustas e escaláveis. Os principais componentes deste ambiente são:

- Linguagens de compilação
- Biblioteca de classes de .Net
- CLR (Common Language Runtime)



Arquitetura .Net

Atualmente, o Framework de .Net é uma plataforma não incluída nos diferentes sistemas operacionais distribuídos por Microsoft, pelo qual é necessária sua instalação prévia à execução de programas criados mediante .Net.

.Net Framework suporta múltiplas linguagens de programação e embora cada linguagem tenha suas características próprias, é possível desenvolver qualquer tipo de aplicação com qualquer destas linguagens. Existem mais de 30 linguagens adaptadas a .Net, desde as mais conhecidas como C# (C Sharp), Visual Basic ou C++ até outras linguagens menos conhecidas como Perl ou Cobol.

6.2 Common Language Runtime (CLR)

O CLR é o verdadeiro núcleo do Framework de .Net, já que é o ambiente de execução no qual se encarregam as aplicações desenvolvidas nas distintas linguagens, ampliando o conjunto de serviços que oferece o sistema operacional padrão Win32.

A ferramenta de desenvolvimento compila o código fonte de qualquer uma das linguagens suportadas por .Net em um mesmo código, denominado código intermediário (MSIL, Microsoft Intermediate Linguaje). Para gerar tal código o compilador se baseia no Common Linguaje Specification (CLS) que determina as regras necessárias para criar código MSIL compatível com o CLR.

Desta forma, indistintamente da ferramenta de desenvolvimento utilizada e da linguagem escolhida, o código gerado é sempre o mesmo, já que o MSIL é a única linguagem que entende diretamente o CLR. Este código é transparente ao desenvolvimento da aplicação já que o gera automaticamente o compilador.

Entretanto, o código gerado em MSIL não é código máquina e, portanto, não pode se executar diretamente. Necessita-se um segundo passo no qual uma ferramenta denominada compilador JIT (Just-In-Time) gera o código máquina real que se executa na plataforma que tenha o computador.

Desta forma se consegue com .Net certa independência da plataforma, já que cada plataforma pode ter seu compilador JIT e criar seu próprio código máquina a partir do código MSIL.

A compilação JIT realiza o CLR à medida que se invocam os métodos no programa e, o código executável obtido, se armazena na memória cache do computador, sendo recompilado só quando se produz alguma mudança no código fonte.

6.3 Introdução à Linguagem C#

C# (CSharp) é uma linguagem de programação orientada a objetos criada pela Microsoft, faz parte da sua plataforma .Net. A companhia baseou C# na linguagem C++ e Java.

A linguagem C# foi criada junto com a arquitetura .NET. Embora existam várias outras linguagens que suportam essa tecnologia (como VB.NET, C++, J#), C# é considerada a linguagem símbolo do .NET pelas seguintes razões:

* Foi criada praticamente do zero para funcionar na nova plataforma, sem preocupações de compatibilidade com código de legado.

* O compilador C# foi o primeiro a ser desenvolvido.

* A maior parte das classes do .NET Framework foram desenvolvidas em C#.

A criação da linguagem, embora tenha sido feita por vários desenvolvedores, é atribuída principalmente a Anders Hejlsberg, hoje um Distinguished Engineer na Microsoft. Anders Hejlsberg era desenvolvedor de compiladores na Borland, e entre suas criações mais conhecidas estão o Turbo Pascal e o Delphi.

Muitos pensam que o nome C# viria de uma sobreposição de 4 símbolos "+" dando a impressão de "++++". Na verdade o "#" de C# refere-se ao sinal musical (sustenido), que aumenta em 1/2 tom uma nota musical. O símbolo real seria o ♯ e não o #, porém, devido à limitação de telas, fontes e alguns browsers, no momento da normalização junto a ECMA, fora especificado apenas que o nome da linguagem seria uma letra C maiúscula (U+0043) e o sinal "#" (U+0023), facilitando assim, publicações e artigos com um caractere encontrado facilmente dos layouts de teclado padrões. Desta forma, caso o nome fosse usado em português, seria "C-Sustenido" (ou "Dó-Sustenido"), e não "C-cerquilha".

C# (pronuncia-se "cê chárp" em português ou "sí Sharp" para o inglês) é, de certa forma, a linguagem de programação que mais diretamente reflete a plataforma. NET sobre a qual todos os programas .NET executam. C# está de tal forma ligada a esta plataforma que não existe o conceito de código não-gerenciado (unmanaged code) em C#. Suas estruturas de dados primitivas são objetos que correspondem a tipos em .NET. A desalocação automática de memória por garbage colletor além de várias de suas abstrações tais como

classes, interfaces, delegados e exceções são nada mais que a exposição explicita recursos do ambiente. NET.

Quando comparada com C e C++, a linguagem é restrita e melhorada de várias formas incluindo:

* Ponteiros e aritmética sem checagem só podem ser utilizados em uma modalidade especial chamada modo inseguro (unsafe mode). Normalmente os acessos a objetos são realizados através de referências seguras, as quais não podem ser invalidadas e normalmente as operações aritméticas são checadas contra sobrecarga (overflow).

* Objetos não são liberados explicitamente, mas através de um processo de coleta de lixo (garbage collector) quando não há referências aos mesmos, previnindo assim referências inválidas.

* Destrutores não existem. O equivalente mais próximo é a interface Disposable, que juntamente com a construção using block permitem que recursos alocados por um objeto sejam liberados prontamente. Também existem finalizadores, mas como em Java sua execução não é imediata.

* Como no Java, não é permitida herança múltipla, mas uma classe pode implementar várias interfaces abstratas. O objetivo principal é simplificar a implementação do ambiente de execução.

* C# é mais seguro com tipos que C++. As únicas conversões implícitas por default são conversões seguras, tais como ampliação de inteiros e conversões de um tipo derivado para um tipo base. Não existem conversões implícitas entre inteiros e variáveis lógicas ou enumerações. Não existem ponteiros nulos (void pointers) (apesar de referências para Object serem parecidas). E qualquer conversão implícita definida pelo usuário deve ser marcada explicitamente, diferentemente dos construtores de cópia de C++.

* A sintaxe para a declaração de vetores é diferente ("int [] a = new int[5]" ao invés de "int a[5]").

* Membros de enumeração são colocados em seu próprio espaço de nomes (namespace)

* C# não possui modelos (templates), mas C# 2.0 possui genéricos (generics).

* Propriedades estão disponíveis, as quais permitem que métodos sejam chamados com a mesma sintaxe de acesso a membros de dados.

* Recursos de reflexão completos estão disponíveis

Apesar de C# ser freqüentemente tido como similar a Java, existem uma série de diferenças importantes, mas a maioria é implementada de forma diferenciada em ambas as linguagens.

Por exemplo, o Java não implementa propriedades, mas permite a utilização de métodos Get e Set que realizam o mesmo processo.

Outros detalhes são:

* O Java não implementa o goto como estrutura de controle, mas o C# sim, apesar de ser pouco usual.
*O Java utiliza comentários Javadoc e o C# utiliza comentários baseados em XML.

* O C# possui indexadores. O Java tem Listeners.

* O Java utiliza a JVM, o C# o .Net Framework, Mono e DotGnu.

* Um dos principais editores do Java é o Eclipse, o do C# é o Visual Studio. Mas ambos têm compiladores de linha de comando.

* O Java pode ser compilado em qualquer plataforma, o C# possui compiladores para Windows, Windows Mobile, Linux, Mac OS X e Solaris. A plataforma .Net da Microsoft provê compiladores para Windows e Windows Mobile (.Net Compact Framework), já a plataforma Mono provê compiladores para Windows, Linux, Mac OS X e Solaris.

No final, ambos possuem recursos similares, sendo possível o programador escolher a linguagem com a qual mais simpatiza.

Ao contrário das outras linguagens de programação, nenhuma implementação de C# atualmente inclui qualquer conjunto de bibliotecas de classes ou funções. Ao invés disso, C# está muito vinculada ao framework .Net/[Mono (projeto)|Mono]], do qual C# obtém suas classes ou funções de execução. O código é organizado em um conjunto de namespaces que agrupam as classes com funções similares. Por exemplo: System.Drawing para gráficos, System.Collections para estrutura de dados e System.Windows.Forms para o sistema Windows Form.

Um nível de organização superior é fornecido pelo conceito de montador (assembly). Um montador pode ser um simples arquivo ou multiplos arquivos ligados juntos (como em al.exe) que podem conter muitos namespaces ou objetos. Programas que precisam de classes para realizar uma função em particular podem se referenciar a montadores como System. Drawing.dll e System.Windows.Forms.dll assim como a biblioteca core (conhecida como mscorlib.dll na implementação da Microsoft).

7. Ambiente de Desenvolvimento

7.1 Visual Studio 2008

O Visual Studio é um conjunto de ferramentas de desenvolvimentos que permite aos desenvolvedores de software a resolver problemas complexos e criar novas soluções.

A função do Visual Studio é aprimorar o processo de desenvolvimento para tornar mais fácil, simples e satisfatório o trabalho de criação. Com o Visual Studio, os desenvolvedores de software se beneficiam com uma experiência de produto integrado que inclui ferramentas, servidores e serviços.

A Microsoft Visual Studio 2008 baseia se em três fundamentos:

* Desenvolvimento rápido de aplicativos;

* Colaboração eficaz em equipe;

* Experiências de usuários inovadores.

8. SQL Server 2005

Em 27 de março de 1987, o presidente da Microsoft na época, Jon Shirley, e o co-fundador e presidente da Sybase, Mark Hoffman, assinaram um acordo onde a Microsoft obteria direitos exclusivos ao produto DataServer da Sybase para o OS/2. Já a Sybase além de obter os royalties da Microsoft, ganharia credibilidade com o endosso de sua tecnologia pela Microsoft.

Para ganhar aceitação no mundo dos bancos de dados para PC, onde o dBASE da Ashton-Tate tinha boa parte do mercado, o “novo” sistema de gerenciamento de banco de dados da Microsoft (licenciado pela Sybase) precisaria interessar à grande comunidade do dBASE. E a maneira mais direta de fazer isso era fazer a Ashton-Tate endossar o produto, e foi exatamente o que a Microsoft fez, um acordo com a Ashton-Tate

Em novembro de 2005 é lançado ao publico o SQL SERVER 2005 (codinome Yukon). O SQL Server 2005 é uma plataforma abrangente de banco de dados que fornece recursos de gerenciamento de dados com ferramentas de BI (Business Intelligence) integradas. Esta versão possui aprimoramentos significantes no modelo de segurança da plataforma de banco de dados, fornecendo um controle mais preciso e permitindo uma maior segurança dos dados.

9. Diagramas

9.1 Diagrama de use case – Movimentação Aluno

O diagrama de caso de uso é um ponto importante na organização e modelagem das principais funcionalidades de um sistema.

Use Case é a especialização de seqüências de ações para atender a uma funcionalidade do sistema, interagindo com seus agentes.

O Caso de Uso abaixo mostra o aluno interagindo com partes do sistema.

Cada usuário terá uma senha para acessar partes do sistema.

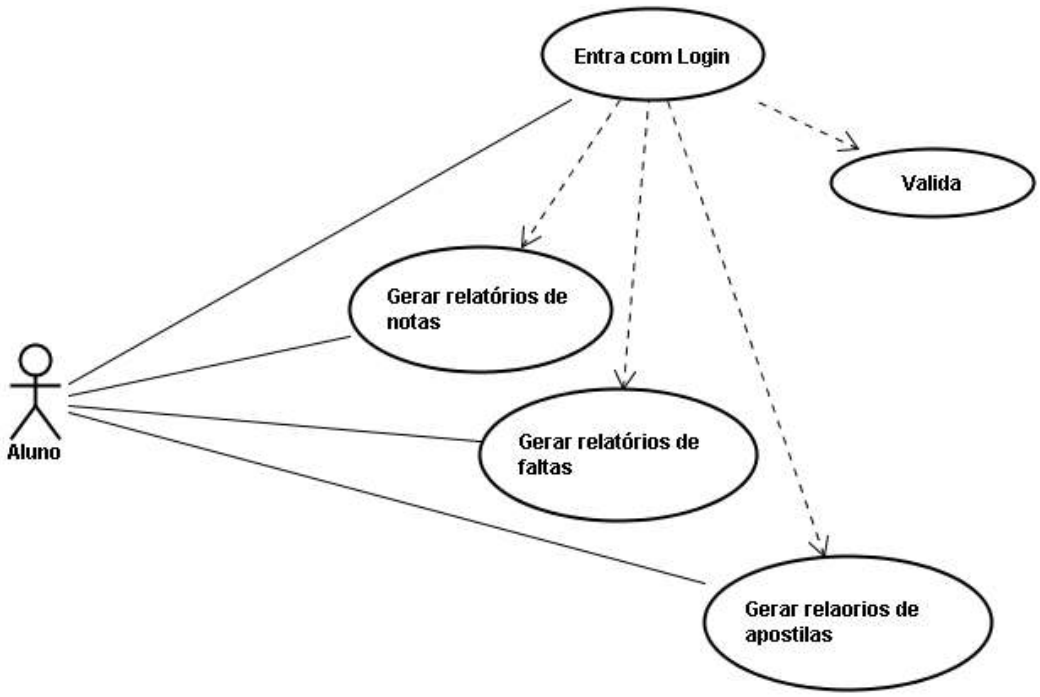


Diagrama de Use Case - Movimentação Aluno

9.2 Diagrama de use case – Movimentação Diretoria

O Diagrama a seguir mostra o cadastro de novos usuários e o acompanhamento realizado pela Diretoria.

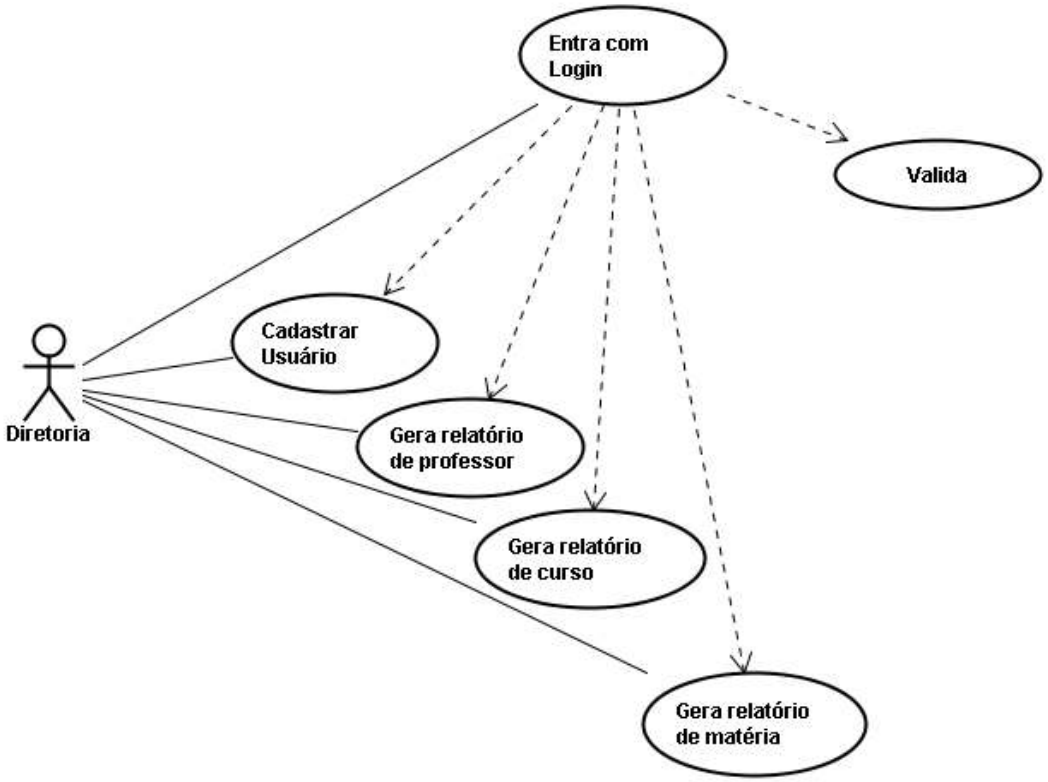


Diagrama de Use Case - Movimentação Diretoria

9.3 Diagrama de use case – Movimentação Secretaria

O Diagrama abaixo mostra as atividades realizadas pela secretaria.

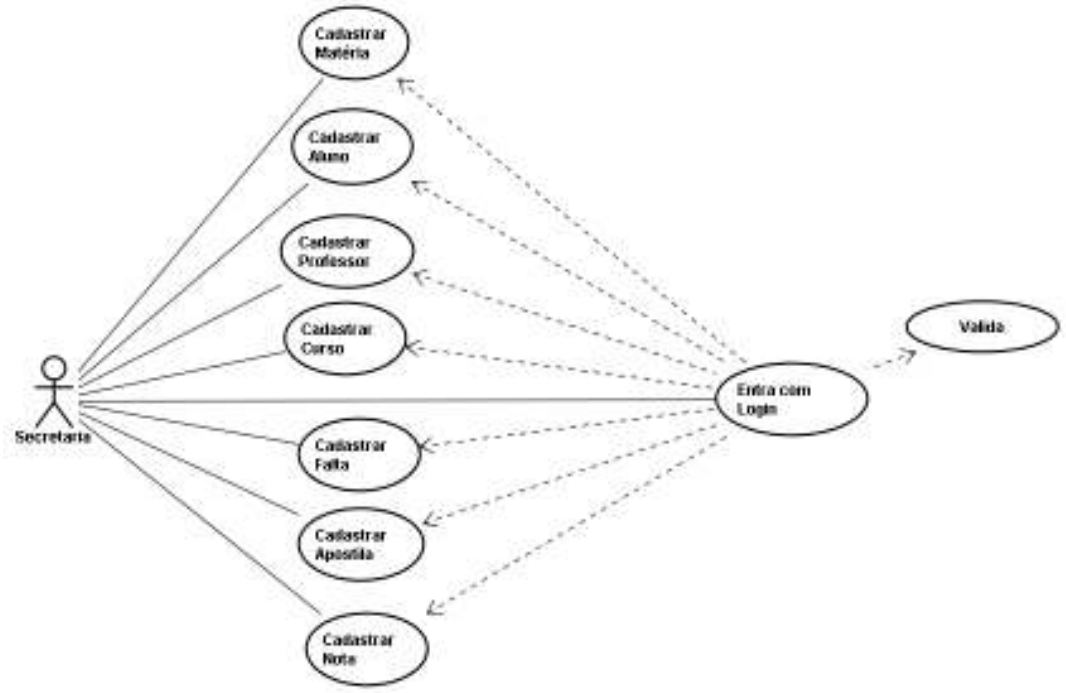


Diagrama de Use Case - Movimentação Secretaria

9.4 Diagrama de use case – Movimentação Geral

O Diagrama a seguir mostra todos os autores e suas atividades dentro do sistema.

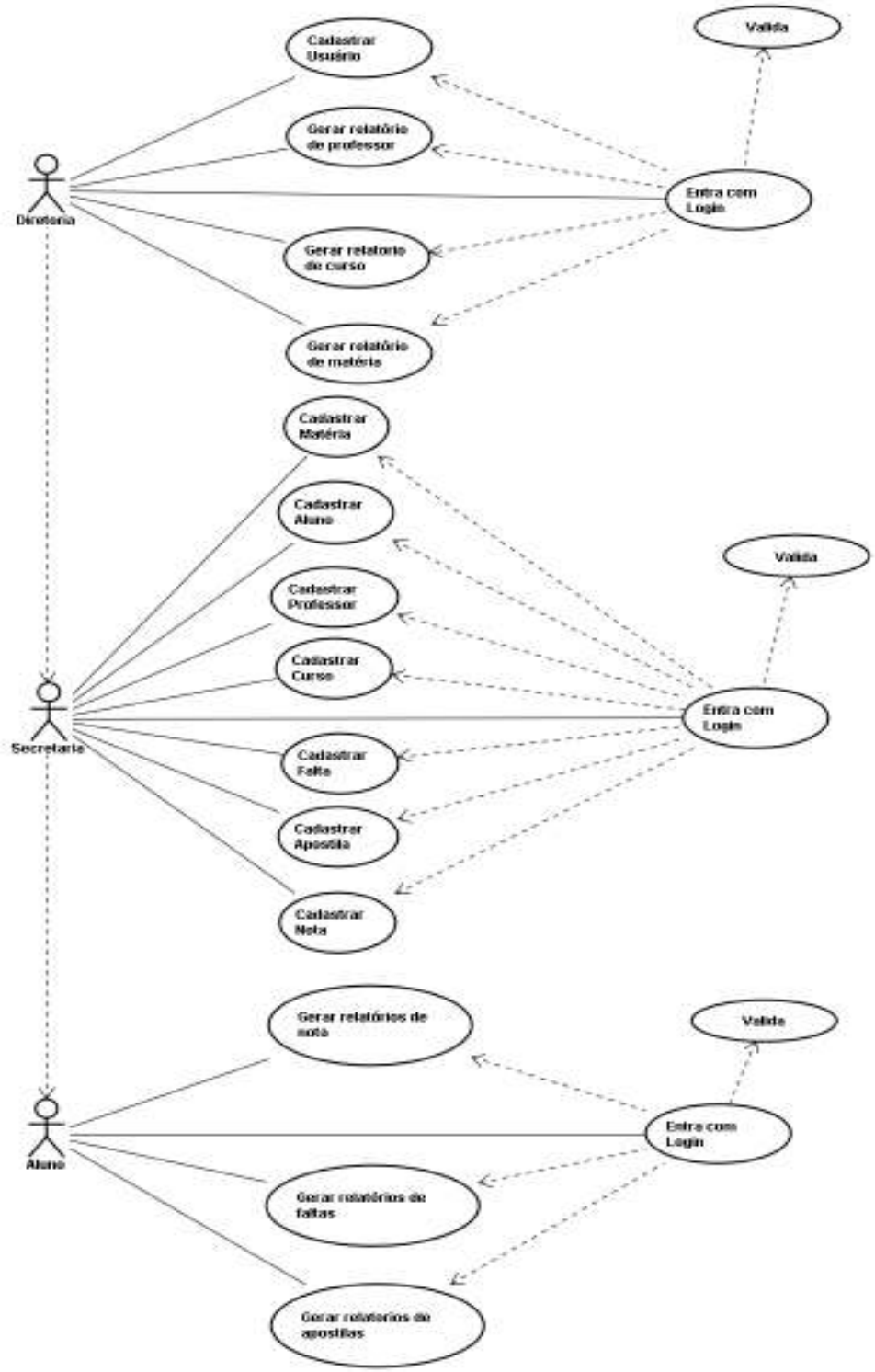


Diagrama de Use Case – Movimentação Geral

10. DESCRIÇÃO DA USE CASE

10.1 Descrição de use case – Autenticação de Usuário

A descrição de caso a seguir descreve a ação do autor.

A secretaria fará a movimentação no cadastro do aluno como mostra o diagrama abaixo.

Descrição de Use Case – Autenticação dos Usuários

O autor faz uma autenticação na tela de login do sistema.

1	**Autenticar Usuários**
Descrição	Fazer a autenticação dos usuários.
Pré – Condição	Usuários são cadastrados no Sistema Decision.
Atores	Diretoria, Secretaria, Professor, Aluno
Prioridade	Essencial
Cenário Principal	1 – A tela de login do sistema Decision é iniciada. 2 – O usuário digita nome e senha para ser feita a autenticação. 3 – O sistema confirma a autenticação e em seguida é aberta a tela do sistema Decision.
Cenário Alternativo	O sistema não confirma a autenticação.

10.2 Descrição de use case – Cadastro de Aluno

A secretaria fará a movimentação no cadastro do aluno como mostra o diagrama abaixo.

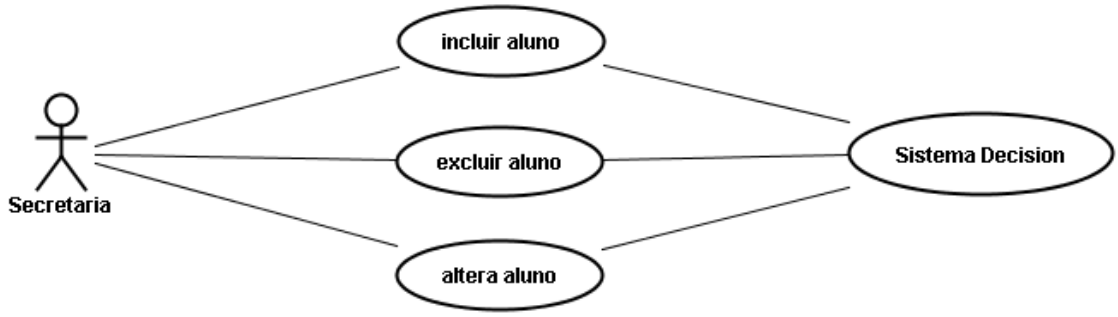


Diagrama de Use Case – Cadastro de Aluno

O autor efetua um cadastro de aluno na tela Cadastro de Aluno.

2	**Incluir Aluno**
Descrição	Inclusão é feita após o aluno ter fornecido todos os dados necessários para seu cadastro.
Pré – Condição	O aluno mostra interesse em fazer parte da Instituição.
Atores	Secretaria, Aluno
Prioridade	Essencial
Cenário Principal	1 – Sistema abre tela para cadastro do aluno. 2 – Secretaria preenche dados cadastrais do aluno. 3 – Secretaria salva as informações.
Cenário Alternativo	O cadastro do aluno só será concluído se todas as informações estiverem completas.

2-1	**Excluir Aluno**
Descrição	Para a exclusão de um aluno o sistema verifica se o mesmo é cadastrado,
Pré – Condições	O aluno precisa desistir da Instituição
Atores	Secretaria, Aluno
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de alunos para efetuar a exclusão. 2 – A secretaria seleciona a opção onde é feita a exclusão. 3 – A secretaria seleciona salvar. 4 – O sistema efetua a exclusão do aluno.
Cenário Alternativo	A secretaria não confirma a exclusão
2-2	**Alterar Aluno**
Descrição	Para efetuar a alteração do aluno o cadastro deverá estar incorreto ou incompleto. A secretaria faz a alteração e salva.
Pré – Condições	Dados incorretos ou incompletos
Atores	Secretaria, Aluno
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de alunos para efetuar a alteração. 2 – A secretaria preenche os dados do aluno par a alteração. 3 – A secretaria seleciona salvar. 4 – O sistema efetua a alteração do aluno.
Cenário Alternativo	A alteração do cadastro do aluno só será concluída se todas as informações estiverem completas

10.3 Descrição de use case – Cadastro de Usuário

No próximo diagrama a diretoria movimenta o cadastro de usuários.

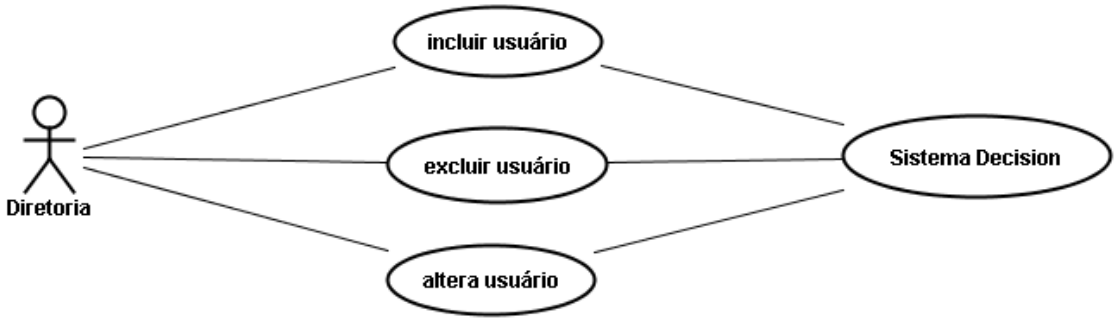


Diagrama de Use Case – Cadastro de Usuários

O autor cadastra usuários na tela Cadastro de Usuários.

3	**Incluir Usuário**
Descrição	Inclusão é feita após o usuário ter fornecido todos os dados necessários para seu cadastro.
Pré – Condição	O usuário tem que fazer parte do quadro de funcionários da Instituição.
Atores	Diretoria, Usuário
Prioridade	Essencial
Cenário Principal	1 – Sistema abre tela para cadastro do usuário. 2 – Diretoria preenche o perfil do usuário. 3 – Diretoria salva as informações. 4 – Sistema faz inclusão do perfil do usuário.
Cenário Alternativo	O cadastro do usuário só será concluído se todas as informações estiverem completas.

3-1	Excluir Usuário
Descrição	Para a exclusão de um usuário o mesmo não poderá fazer parte do quadro funcional da Instituição.
Pré – Condições	O usuário foi demitido ou pediu demissão.
Atores	Diretoria, Usuário
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de usuário para efetuar a exclusão. 2 – Diretoria seleciona o perfil do usuário e efetua a exclusão. 3 – Diretoria seleciona salvar. 4 – O sistema efetua a exclusão do usuário.
Cenário Alternativo	Diretoria não confirma a exclusão

3-2	Alterar Usuário
Descrição	A alteração e feita caso o usuário tenha mudado de perfil.
Pré – Condições	Usuário foi remanejado de setor
Atores	Diretoria, Usuário
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de usuário para efetuar a alteração. 2 – Diretoria seleciona o novo perfil do usuário e efetua a alteração. 3 – O sistema efetua a alteração do usuário.
Cenário Alternativo	A alteração do cadastro do usuário só será concluída se todas as informações estiverem completas

10.4 Descrição de use case – Cadastro de Professor

No próximo diagrama a secretaria movimenta o cadastro de professores.

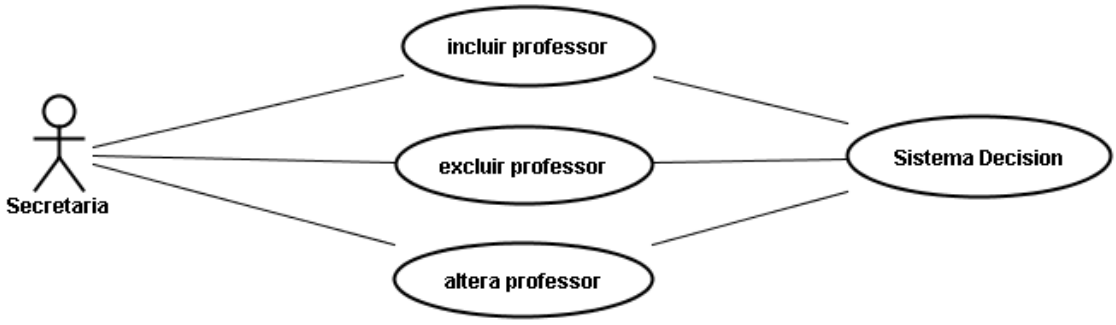


Diagrama de Use Case – Cadastro de Professor

O autor efetua um cadastro de professor na tela Cadastro de Professor.

4	**Incluir Professor**
Descrição	Inclusão é feita após o professor ter fornecido todos os dados necessários para seu cadastro.
Pré – Condição	O professor é contratado para ministrar aulas na Instituição.
Atores	Secretaria, Professor
Prioridade	Essencial
Cenário Principal	1 – Sistema abre tela para cadastro do professor. 2 – Secretaria preenche dados cadastrais do professor. 3 – Secretaria salva as informações.
Cenário Alternativo	O cadastro do professor só será concluído se todas as informações estiverem completas.

4-1	Excluir Professor
Descrição	Para a exclusão de um professor o sistema verifica se o mesmo é cadastrado.
Pré – Condições	O professor por qualquer motivo saiu da Instituição.
Atores	Secretaria, Professor
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de professor para efetuar a exclusão. 2 – A secretaria seleciona a opção onde é feita a exclusão. 3 – A secretaria seleciona salvar. 4 – O sistema efetua a exclusão do professor.
Cenário Alternativo	A secretaria não confirma a exclusão

4-2	Alterar Professor
Descrição	Para efetuar a alteração do professor o cadastro deverá estar incorreto ou incompleto. A secretaria faz a alteração e salva.
Pré – Condições	Professor ministrará outra disciplina.
Atores	Secretaria, Professor
Prioridade	Importante
Cenário Principal	1 – Sistema abre a tela de cadastro de professor para efetuar a alteração. 2 – A secretaria preenche os dados do professor par a alteração. 3 – A secretaria seleciona salvar. 4 – O sistema efetua a alteração do professor.
Cenário Alternativo	A alteração do cadastro do professor só será concluída se todas as informações estiverem completas

11. Diagrama de classe

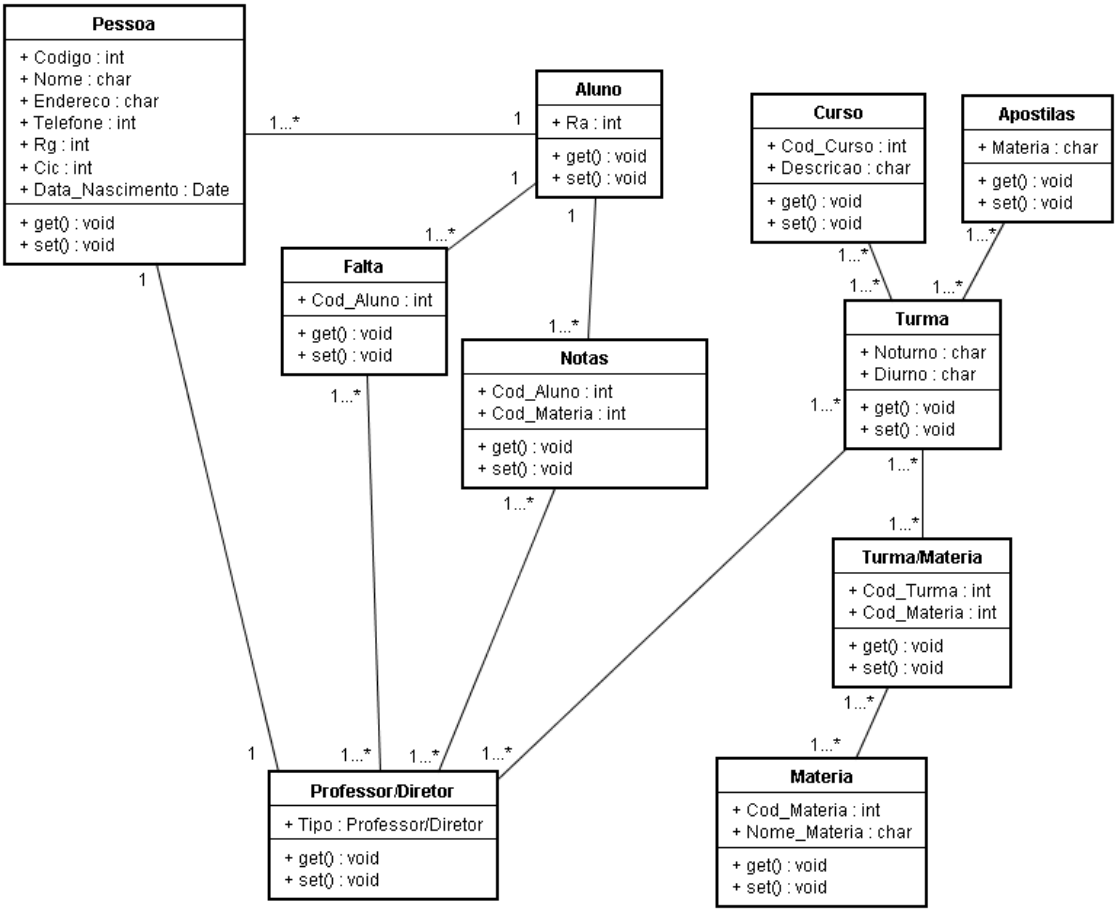


Diagrama de Use Case – Diagrama de Classe

12. Diagrama Entidade Relacionamento

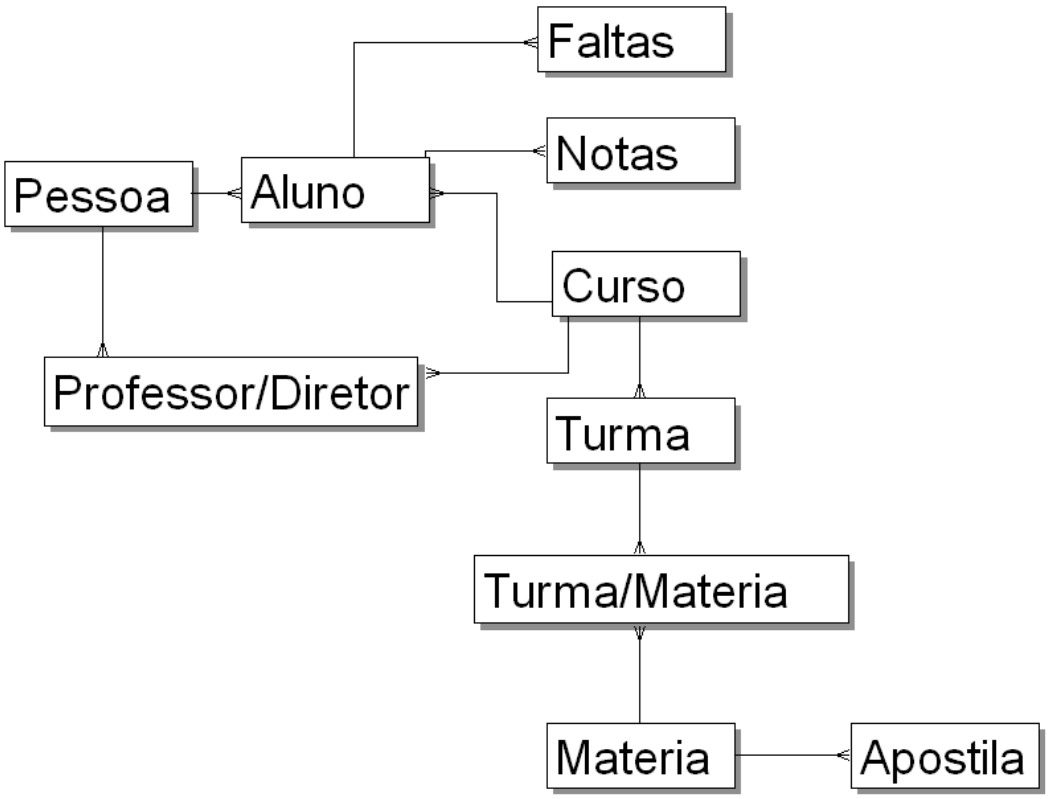


Diagrama Entidade Relacionamento

13. Conclusão

Neste período de estudos pode-se concluir que esta tecnologia C# que é considerada freqüentemente como similar ao Java, existe uma série de diferenças importantes, mas a maioria é implementada de forma diferenciada em ambas as linguagens.

O sistema desenvolvido, apesar de simples tem suas funções necessárias para ser implantado na escola, podendo ao decorrer do tempo sendo modelado e readequado conforme surgir novas necessidades.

Ele possui comandos de fácil acesso, armazenamento de dados com confiabilidade, pois usa o Banco de Dados SQL Server assim facilitando a tarefa do usuário e também podendo diminuir os custos operacionais.

14. Referencias Bibliográficas

Prado, Antonio Francisco. Curso de pós-graduação “Lato sensu” Especialização em Computação. Assis. Marco de 2006.

BLAHA, Michael; RUMBAUGH, James. Modelagem e Projetos Baseados em Objetos com UML2. Rio de Janeiro: Editora Campus, 2006.

SOUKUP, Ron; Desvendando o Microsoft SQL Server 6.5.
Rio de Janeiro: Editora Campus – 1998.

WILLE, Christoph;Apresentando C#.
São Paulo: Berkeley Brasil, 2001.

FREEZE, S. Wayne; SQL Guia de Referência do Programador.
Rio de Janeiro: Editpra Ciência Moderna Ltda, 1998.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John.
Padrões de Projetos: Soluções Reutilizáveis de Software Orientado a Objetos.
Porto Alegre: Editora Bookman, 2000.

“UML Linguagem de Modelagem Unificada”
<http://inf.unisul.br/~osmarjr/download/apostilauml.html>acesso em 27 de abril de 2009.

“Arquitetura básica da plataforma .Net. Descrição do Framework e seus principais componentes: Linguagens, biblioteca de classes e CLR.”
<http://criarweb.com/artigos/net.framework.html> acesso em 15 de Agosto de 2009.