UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

SISTEMA DE GESTÃO ESCOLAR OBJETO-RELACIONAL UTILIZANDO BANCO DE DADOS CACHÉ

JULIANO WALTER BRUNE

BLUMENAU
2007

2007/1-21

JULIANO WALTER BRUNE

SISTEMA DE GESTÃO ESCOLAR OBJETO-RELACIONAL UTILIZANDO BANCO DE DADOS CACHÉ

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Wilson Carli - Orientador

BLUMENAU
2007

2007/1-21

SISTEMA DE GESTÃO ESCOLAR OBJETO-RELACIONAL UTILIZANDO BANCO DE DADOS CACHÉ

Por

JULIANO WALTER BRUNE

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Wilson Pedro Carli, Mestre – Orientador, FURB
Membro:	Prof. Mauro Marcelo Mattos, Doutor – FURB
Membro:	Prof. Alexander Roberto Waldameri, Mestre – FURB

Blumenau, 2 de agosto de 2007

Dedico este trabalho ao meu filho Pedro, que ainda nem nasceu, mas já faz parte da minha vida.

AGRADECIMENTOS

À Deus, por mostrar sempre o melhor caminho e me guiar com seu amor.

À minha Mãe, pelos conselhos e pela constante presença em minha vida.

Ao meu Pai, pelo caráter e honestidade.

À minha Esposa pelo companheirismo, amor e cumplicidade.

À minha família pelo conforto.

A vida nos foi dada por Deus para a empregarmos em benefício da Humanidade.

Bulow

RESUMO

Este trabalho apresenta um estudo sobre as principais características do BDC, resultando no desenvolvimento de um sistema de gestão escolar aplicado a uma escola de ensino fundamental. No decorrer do trabalho é apresentado o BDC, suas ferramentas de desenvolvimento e manutenção, seu suporte a objetos e a integração com outras tecnologias.
Palavras-chave: Sistema de gestão. Orientação a objetos. Banco de dados. Caché.

ABSTRACT

This work describe a study about the main characteristics of Caché Database. Showing development and maintenance tools, objects support and integration with new technologies. It was developed a small school management system to validate the work.
Key-words: System of management. Orientation the objects. Data base. Caché.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

BDC – Banco de dados Caché

BDOO – Banco de dados orientado a objetos

BDR – Banco de dados relacional

CAD - *Computer-Aided Design*

COS – *Caché Object Script*

CSP – *Caché Server Pages*

EJB – *Enterprise Java Beans*

FK – *Foreign key*

GUI – *Grafic user interface*

ID – *Object Identify*

IDE - *Integrated Development Environment*

OCI – *Oracle Calling* Interface

ODBC – *Open database connectivity*

OO – Orientação a Objetos

SGBD – Sistema gerenciador de banco de dados

SGBDOO – Sistema gerenciador de banco de dados orientado a objetos

SGBDOR – Sistema gerenciador de banco de dados objeto-relacional

SGE – Sistema de Gestão Escolar

SQL – *Structed Query Language*

UML – *Unified Modeling Language*

VB – *Visual Basic*

XML - *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Segundo Tiobe Software (2006), em 2006 as linguagens de programação mais utilizadas no mundo foram: Java, C, C++ e Visual Basic. Entre as quatro linguagens de programação mais utilizadas, três são orientadas a objetos, fazendo com que esse paradigma de programação assumisse o primeiro plano no cenário de desenvolvimento de aplicações. Estas linguagens de programação orientadas a objetos são mais utilizadas devido aos seus ricos modelos de dados e ao suporte a conceitos que melhoram a produtividade, tais como encapsulamento, herança e polimorfismo.

Entretanto, uma grande parcela dos dados ainda reside em bancos de dados relacionais, provocando assim um aumento no tempo de desenvolvimento, pois é preciso mapear os dados para que eles possam ser compreendidos e manipulados pela tecnologia orientada a objetos. “As vantagens da tecnologia de objetos são diminuídas quando os objetos de um banco de dados resultantes têm que ser forçados ao modelo relacional bidimensional” (INTERSYSTEMS, 2005b). O modelo relacional suporta uma pequena quantidade de tipos de dados que se mostram adequados para aplicações convencionais. Porém, quando utiliza-se o modelo orientado a objetos para desenvolvimento de aplicações, tem-se a necessidade de um esforço adicional de programação para adaptá-los ao modelo relacional. Os objetos podem ter estruturas complexas fazendo com que se tenha um trabalho a mais de mapeamento objeto-relacional.

Segundo Intersystems (2005b), “O uso e a complexidade de aplicações de Tecnologia da Informação (TI) estão explodindo e os sistemas de hoje tem exigências e processos crescentes que superam as capacidades da tecnologia relacional”.

Armazenar objetos em um banco de dados relacional é uma tarefa complexa. O modelo relacional não possui os mecanismos necessários para representar características simples do modelo orientado a objeto como a herança e o polimorfismo. Nesse cenário, um banco de dados relacional não traz vantagem. “A escolha de um banco de dados orientado a objeto é a evolução natural” (INTERSYSTEMS, 2002b).

Tendo em vista a crescente utilização de linguagens de programação orientada a objetos, propõe-se neste trabalho o desenvolvimento de um SGE, utilizando para seu desenvolvimento a linguagem de programação Visual Basic e a ferramenta Caché Studio, que é uma IDE para desenvolver e depurar aplicações com Caché. Esta ferramenta é disponibilizada pelo próprio servidor de aplicações do Caché. O BDC será utilizado para a

persistência de todos os objetos do sistema bem como para a criação das classes.

Conforme entrevista com a administração do Colégio Madre Francisca Lampel, observou-se a necessidade de um controle acadêmico que pudesse solucionar os problemas de ordem administrativa e educacional encontradas na instituição. O Colégio possui um sistema de gestão escolar limitado, pois não utiliza banco de dados para persistência dos dados e sim um sistema de arquivos indexados. O sistema atual é desenvolvido na linguagem Cobol. Após uma consultoria realizada na instituição por uma empresa externa, observou-se a necessidade da atualização do SGE por outro que possuísse os requisitos que serão apresentados no capítulo 3.1

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho é mostrar através da construção de um SGE, as características de orientação a objeto suportadas pelo BDC (Herança, polimorfismo e encapsulamento).

Os objetivos específicos são:

a) desenvolver um sistema de gestão escolar (*desktop*), utilizando os recursos de orientação a objetos existentes no BDC;
b) atualizar a administração de ensino, materiais e biblioteca;

1.2 ESTRUTURA DO TRABALHO

Este trabalho encontra-se estruturado em forma de capítulos.

No Capítulo 1 são apresentados a introdução, as justificativas, os objetivos e a organização do trabalho.

No Capítulo 2 é apresentada a fundamentação teórica, onde serão abordados diversos temas relacionados ao BDC e ao sistema de gestão escolar.

No Capítulo 3 é descrito todo o processo para o desenvolvimento do sistema, sua especificação, os requisitos que o sistema deverá ter e a sua implementação.

No Capítulo 4 são apresentadas as conclusões e sugestões para trabalhos futuros.

No Capítulo 5 são apresentadas as referências bibliográficas.

2 FUNDAMENTAÇÃO TEÓRICA

Para que os objetivos propostos neste trabalho possam ser alcançados, faz-se necessário abordar os aspectos relevantes do BDC, suas características OO, sua estrutura interna de armazenamento, suas ferramentas de desenvolvimento e as interações com outras tecnologias.

2.1 BANCO DE DADOS RELACIONAL

Segundo Boscarioli (2006, p. 1), na década de 60 as informações eram armazenadas em arquivos seqüenciais, com o crescimento e aprimoramento dos sistemas, viu-se a necessidade da criação de um sistema de armazenamento de dados independente de aplicação. Surge em meados dos anos 60 os primeiros SGBD. No modelo de dados relacional as informações são armazenadas em tabelas bidimensionais, limitadas a linhas e colunas. Através de chaves primarias é possível fazer referência aos dados contidos em uma ou mais tabelas. Um conceito importante do modelo relacional é da integridade referencial, cujo papel é de garantir que um dado referenciado não seja excluído, perdendo assim a sua referência.

2.2 BANCO DE DADOS ORIENTADO A OBJETO

Em partir dos anos 80 as aplicações e o processamento das máquinas evoluíram, permitindo modelagens de dados mais complexas. Com está crescente evolução, surgiram sistemas robustos que necessitavam de uma imensa camada de dados. Para esses sistemas como, por exemplo, o CAD, o modelo de dados relacional não era mais adequado (GUALBERTO, 2005).

Surge na década de 80 o conceito de BDOO. Primeiramente como trabalhos acadêmicos de pesquisa em universidades e centros de pesquisa, posteriormente em uma ferramenta viável e comercial (GUALBERTO, 2005).

O BDOO permite a persistência de dados complexos sem a necessidade de grandes mapeamentos e codificações. O modelo oferece suporte a características da programação OO, como herança, polimorfismo e encapsulamento. Os modelos de um BDOO estão diretamente relacionados com a construção das classes do sistema. Dentro de um BDOO, o objeto pode possuir diversos atributos, porém, existe um atributo que é criado automaticamente na sua construção e é responsável pela identidade do objeto, o ID. Esse ID pode parecer muito com as chaves primárias usadas em modelos relacionais. Porem, as chaves primárias apenas identificam um registro em um relacionamento, já o ID identifica o objeto em toda a base de dados. Pode-se ter dois objetos com os mesmos estados, porém suas identidades serão diferentes (GUALBERTO, 2005).

2.3 BANCO DE DADOS ORIENTADO A OBJETO VS BANCO DE DADOS RELACIONAL

Mesmo tendo diferentes formas de armazenamento, os dois modelos de bancos de dados servem para o mesmo propósito, o de armazenar dados necessários à manutenção das aplicações.

Uma das diferenças mais importantes entre o modelo relacional e o orientado a objeto, é o fato de que o modelo relacional não tem a capacidade de armazenar dados complexos de forma simples. Sua estrutura bidimensional não possibilita o armazenamento de um objeto de forma única, tendo a necessidade de partir o objeto e força-lo em diferentes tabelas relacionais. O custo para remontar esse objeto é alto e necessita de grandes mapeamentos objeto-relacional (GUALBERTO, 2005).

Segundo BORBA (2006, p. 62), a vantagem mais significativa do modelo OO é que o resultado obtido é o mais próximo do modo como pensamos. Uma classe definida terá uma correspondência com algo real, facilitando a compreensão.

Uma solução encontrada para armazenamento de herança em um BDR é colocar em uma tabela todos os atributos da superclasse, ou ainda, definir tabelas para a superclasse e para a classe derivada separadamente, sendo necessária a utilização de junções para a remontagem do objeto. Em um BDOO a passagem do modelo de classes para o Banco é direta.

BDOO tem vantagens sobre BDR pois:

a) executam mais rapidamente aplicações transacionais;

b) trabalham com objetos complexos mais efetivamente;

c) oferecem mais produtividade ao desenvolvedor;

d) são mais facilmente gerenciáveis.

No quadro 1 é mostrada uma comparação das técnicas do modelo relacional e de objetos.

Conceitos do modelo de objetos	Modelo relacional	Modelo de objetos	Benefícios do modelo de objetos
Abstração de dados	Entidades de intersecção e indexação para representar referências entre tuplas	ID para representar diretamente as referências entre objetos	Esquemas mais simples para representar dados complexos
Herança	Tipo codificado e Programado	Hierarquias de classe	Representação direta das referências entre tipos e subtipos e suporte para a especialização de cada subtipo
Encapsulamento	Tipo codificado e programado, usualmente com bibliotecas	Fornece encapsulamento embutido para assegurar a execução do código correto nos dados corretos	Reduz o código de aplicação e a chance de erro de execução do código errado em dados corretos

Fonte: Borba (2006, p. 64).

Quadro 1 - Comparação das técnicas do modelo relacional e de objetos

2.4 BANCO DE DADOS PÓS-RELACIONAL CACHÉ

Desenvolvido pela Intersystems e lançado no mercado em 1997, o Caché é um BD Pós-Relacional de alta performance. É composto por um servidor de aplicações com capacidade de programação em objetos e integração com uma ampla variedade de tecnologias (SAMARY, 2004a).

O servidor de aplicações do Caché oferece:

a) máquina virtual que roda duas linguagens de *scripting* embutidas (COS e Basic);

b) acesso aos servidores de dados multidimensionais Caché no mesmo e em outros

computadores com roteamento transparente;

c) software de conectividade com *caching* no lado cliente, para permitir acesso a objetos Caché de todas as tecnologias comumente usadas, incluindo Java, C++, C#, COM, .NET, VB e DELPHI;
d) compatibilidade com SOAP e XML;
e) acesso a SQL usando ODBC e JDBC;
f) acesso a BDR;
g) CSP para criação de páginas dinâmicas;
h) Caché Studio.

Por não ser um banco de dados apenas multidimensional, apenas orientado a objetos ou apenas relacional(SQL), foi dado o nome de BD Pós-Relacional, uma evolução do modelo relacional. O armazenamento interno dos dados é feito através de arrays multidimensionais chamado de globais, porém os dados podem ser acessados na forma de objetos, SQL ou com o acesso direto as globais (MULLER, 2006)

Segundo Atkinson (2006, p. 61 apud BORBA, 1989), algumas regras devem ser obedecidas para que um BD seja caracterizado com sendo OO. Uma delas é o suporte a objetos complexos, identidade de objetos (ID) e encapsulamento. O BDC suporta essas três regras e portanto pode ser caracterizado com sendo OO.

2.4.1 Armazenamento de dados complexos

Segundo Gualberto (2005), objetos complexos são construções formadas por coleções, arrays, tuplas ou listas. Essas formações são aplicadas a objetos simples (inteiros, *strings*). No modelo relacional essas construções são criadas através de relacionamentos entre diversas tabelas. No modelo OO, os construtores são em geral multidimensionais, ou seja, qualquer construtor pode ser aplicado a qualquer objeto. No modelo relacional o construtor apenas pode ser aplicado a tuplas, e o construtor de registro apenas a valores atômicos.

Segundo Borba (2006, p. 64), “A idéia básica de um SGBDOO é armazenar diretamente qualquer modelo complexo de dados em um banco de dados, sem limitações para os conceitos mantidos pela orientação a objetos”.

Na quadro 2 é mostrado um exemplo de dados complexos implementado no BDC.

```
[1] Class User.Turma Extends %Persistent [ ClassType = persistent, ProcedureBlock ]
[2] {
[3]
[4] Property Disciplina As Disciplinas [ Collection = array ];
[4]                                    .
[5]                                    .
[6]                                    .
[7] ClassMethod adicionarDisciplinasTurma(idDisciplina As %Integer, objTurma As Turma) As Turma
[8] {
[9]                                    .
[10]                                   .
[11]                                   .
[12] do objTurma.Disciplina.SetAt(##class(User.Disciplinas).%OpenId(idDisciplina),count)
[13] do objTurma.%Save()
[14] }
[15]
[16] quit objTurma
[17]
[18] }
```

Quadro 2 - Dados complexos no Caché

Observa-se na linha 4 (quatro) do quadro 2, que a classe `Turma` possui um *array* do tipo objeto de `Disciplinas`. Essa declaração permite adicionar diversos objetos de `Disciplinas` a um objeto de `Turma`. Na linha 12 (doze) do quadro 2 é mostrado o comando de inclusão de um objeto de `Disciplinas` no *array*. Na figura 1 é mostrada a estrutura da global `Turma`. Observa-se que a `Turma` com o ID igual a 1 (um) possui 3 (três) disciplinas em seu *array* de Disciplinas, referenciados pelos seus próprios ID.

Figura 1 - Estrutura da global Turma

2.4.2 Globais

No BDC os dados não são armazenados em tabelas bidimensionais e sim a globais. Globais são arrays multidimensionais persistentes. É a forma como o Caché armazena as informações (em baixo nível). A estrutura interna de uma global é multidimensional. Pode-se observar a estrutura interna de uma global na figura 2. Observa-se ainda que a global `Pessoa` armazena objetos de `Professor`, `Aluno e Fornecedor`. A figura 2 demonstra também como é a forma de armazenamento quando se tem uma relação de herança.

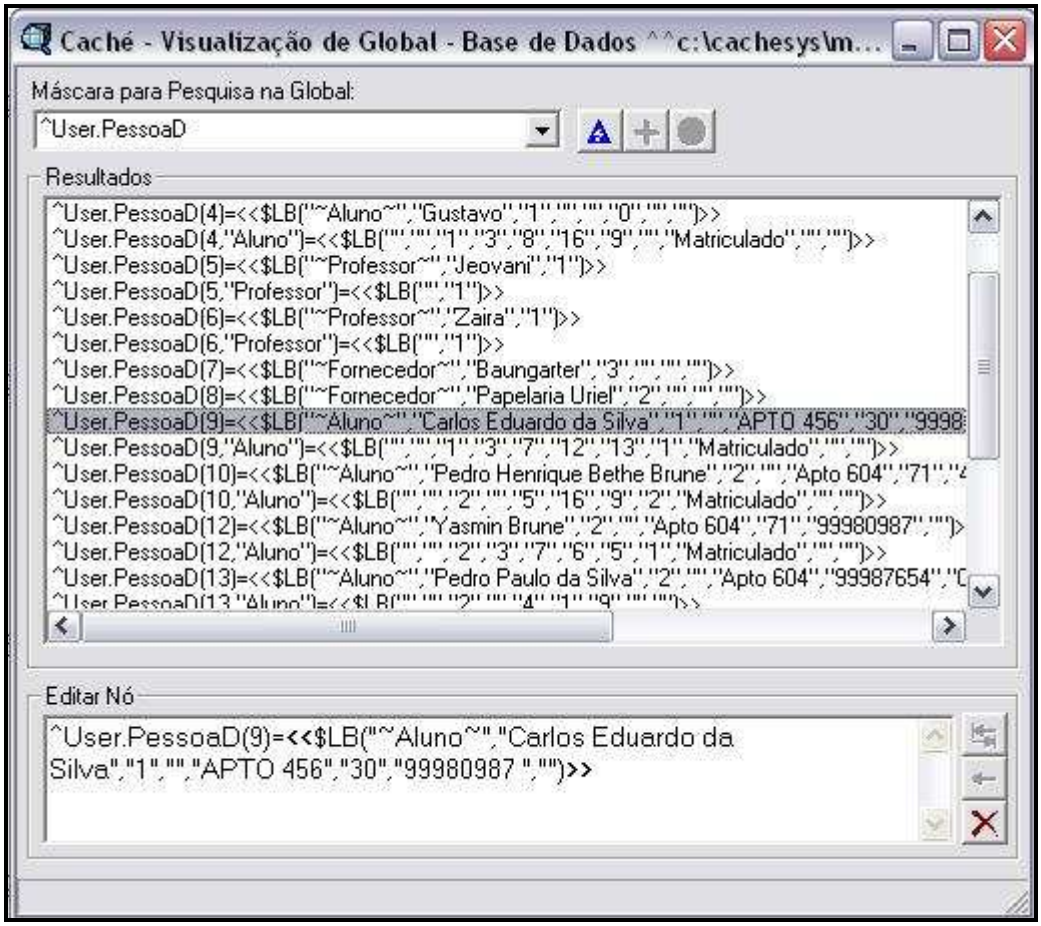


Figura 2 - Estrutura da global `Pessoa`

2.4.3 Base de dados no Caché

Segundo Muller (2006), base de dados é o local físico onde ficam contidas as globais do sistema. O armazenamento das globais é dado em um arquivo chamado cache.dat. A partir

da versão 5 do Caché, as bases de dados são armazenadas em blocos de 8k, permitindo um tamanho máximo de 32 *terabytes* de dados**.**

2.4.4 Namespace

É uma entidade lógica que agrupa uma ou mais bases de dados em uma única unidade. Essa referência existe somente dentro do Caché. O *namespace* separa o código da aplicação e referencia sua localização física na base de dados. Todo e qualquer acesso aos dados deve ser feito através do *namespace*. Por padrão, na instalação do caché são criadas os seguintes *namespaces*: *USER, SAMPLES, DOCBOOK, %CACHELIB e %SYS* (MULLER, 2006).

2.4.5 Mecanismo multidimensional de dados

Diferente do modelo relacional que forçam os objetos em duas ou mais tabelas relacionais, o Caché armazena os dados em tabelas multidimensionais. O fato de armazenar dados em tabelas multidimensionais faz, com que se tenha um modelo de dados altamente coerente com a descrição das classes de objetos, diminuindo o custo de mapeamentos existentes na tecnologia relacional (INTERSYSTEMS, 2004).

O termo “multidimensional” significa que pode-se indexar os dados por quantos parâmetros forem necessários, pois eles não estão limitados a linhas e colunas. A estrutura multidimensional do Caché juntamente com o suporte a herança, polimorfismo e encapsulamento, caracteriza o banco como sendo também orientado a objetos (SOUZA, 2005, p. 19).

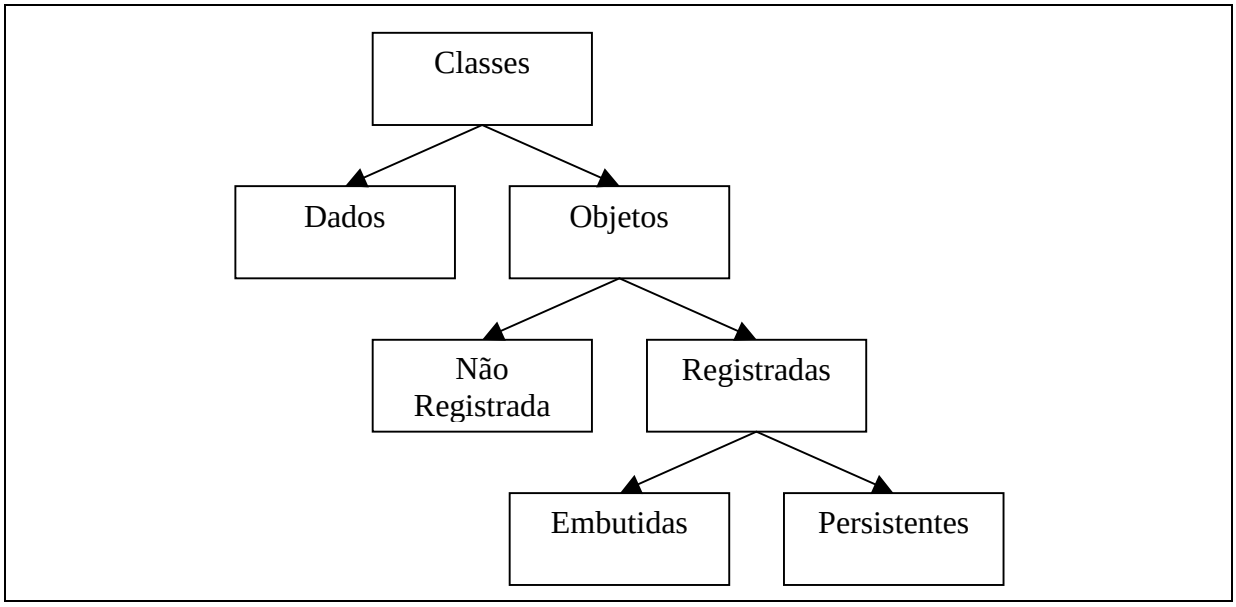
Quando modela-se dados complexos do mundo real e utiliza-se para persistência tecnologias relacionais, necessita-se fragmentar as informações, para que estas caibam em tabelas bidimensionais, exigindo um custo de processamento para realizar a fragmentação e montagem desses objetos. Dados multidimensionais não necessitam serem montados a partir de partes. O servidor de dados do Caché elimina está sobrecarga de processamento existente na tecnologia relacional (INTERSYSTEMS, 2002d).

2.4.6 Classes

Segundo Booch (2006, p. 71 apud BORBA, 2000, p. 49), “classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamento e semântica”.

Segundo Boscarioli (2003), classes são agrupamentos de objetos de um mesmo tipo, que possuem comportamentos (operações) e propriedades (atributos e relação) em comum.

No sistema proposto, a criação das classes é feita utilizando a IDE Caché Studio disponibilizada junto ao próprio BDC. Pode-se desenvolver toda a regra de negócio utilizando está ferramenta. As classes criadas no Caché Studio podem ser projetadas para Java, Visual Basic e Rational Rose. Existem diferentes tipos de classe no Caché dentre a qual a mais utilizada é a classe persistente. O quadro 3 mostra os tipos de classes suportadas no Caché.



Fonte: Souza (2005, p. 21).

Quadro 3 - Tipos de classes do Caché

2.4.6.1 Relacionamento entre classes

As classes no BDC podem se relacionar de diversas maneiras. Em um BDR o relacionamento é feito utilizando as FK. O problema encontrado no modelo relacional é que muitas vezes o índice utilizado para se construir a referência é um próprio dado da tabela. Se

houvesse um erro na digitação do dado, teria que ser alterado em todas as tabelas que fizessem referência. O problema foi resolvido com a criação do ID, cuja criação é de responsabilidade do BD.

O Quadro 4 exemplifica os tipos de relacionamentos encontrados do BDC.

Tipo de relacionamento	Descrição
Referência simples	Em uma referência simples, um objeto simplesmente aponta para um outro objeto através de seu ID. Se o outro objeto for apagado, o primeiro não saberá disto e, ao tentar abri-lo, ocorrerá um erro de <INVALID OREF>. Isto é, não existe integridade referencial.
Relacionamento	Um relacionamento é uma referência entre objetos de duas classes, mantida através do ID dos objetos envolvidos. As diferenças entre um Relacionamento e uma Referência simples são: • O Caché garante a integridade referencial.• O relacionamento aparece para o mundo relacional como uma foreing key.• A entidade referenciada é mantida em uma lista dinâmica nos objetos que a referenciam (inverse property). Um relacionamento pode ser de dois tipos: One-to-Many e Parent-Children, isto é, “um para muitos” e “pai filho” (onde os filhos seriam as “entidades fracas”).
Composição/Agregação	Composição/Agregação Uma referência simples para um objeto de uma classe do tipo serial. Quando o objeto que mantém a referência é salvo, o objeto serial referenciado é serializado e salvo junto com ele.
Foreing Key relacional	Um relacionamento entre objetos mantido através de uma ou mais colunas que juntas representam um outro objeto (linha de uma outra tabela). O Caché garante a integridade referencial.

Fonte: Samary (2004b, p. 20).

Quadro 4 - Tipos de relacionamentos encontrados no BDC

2.4.7 Atributos e Métodos

Segundo Borba (2006), atributo é um valor de dados guardado pelos objetos de uma classe. Pode-se criar diferentes objetos com os mesmos atributos, porém eles terão identificadores diferentes (ID).

Os atributos e métodos do sistema foram criados no Caché Studio. Esses métodos e atributos podem ser acessados por diversas tecnologias. Uma delas é o Visual Basic, onde foi construída a interface GUI do sistema. Um exemplo de atributos e métodos podem ser vistos

no quadro 5.

```
Class User.Professor Extends User.Pessoa [ ClassType = persistent, ProcedureBlock ]
{
Property disciplinasLeciona As Disciplinas [ Collection = array ];
Property turmasLeciona As Turma [ Collection = array ];
ClassMethod adicionarDisciplinas(idDisciplinas As %Integer, objProfessor As Professor) As
Professor
{
 Set count = objProfessor.disciplinasLeciona.Count()
 Set count = count + 1
 do objProfessor.disciplinasLeciona.SetAt(##class(User.Disciplinas).%OpenId(idDisciplinas),count)
 do objProfessor.%Save()
 quit objProfessor
}
```

Quadro 5 - Atributos e métodos no Caché

2.4.8 Queries

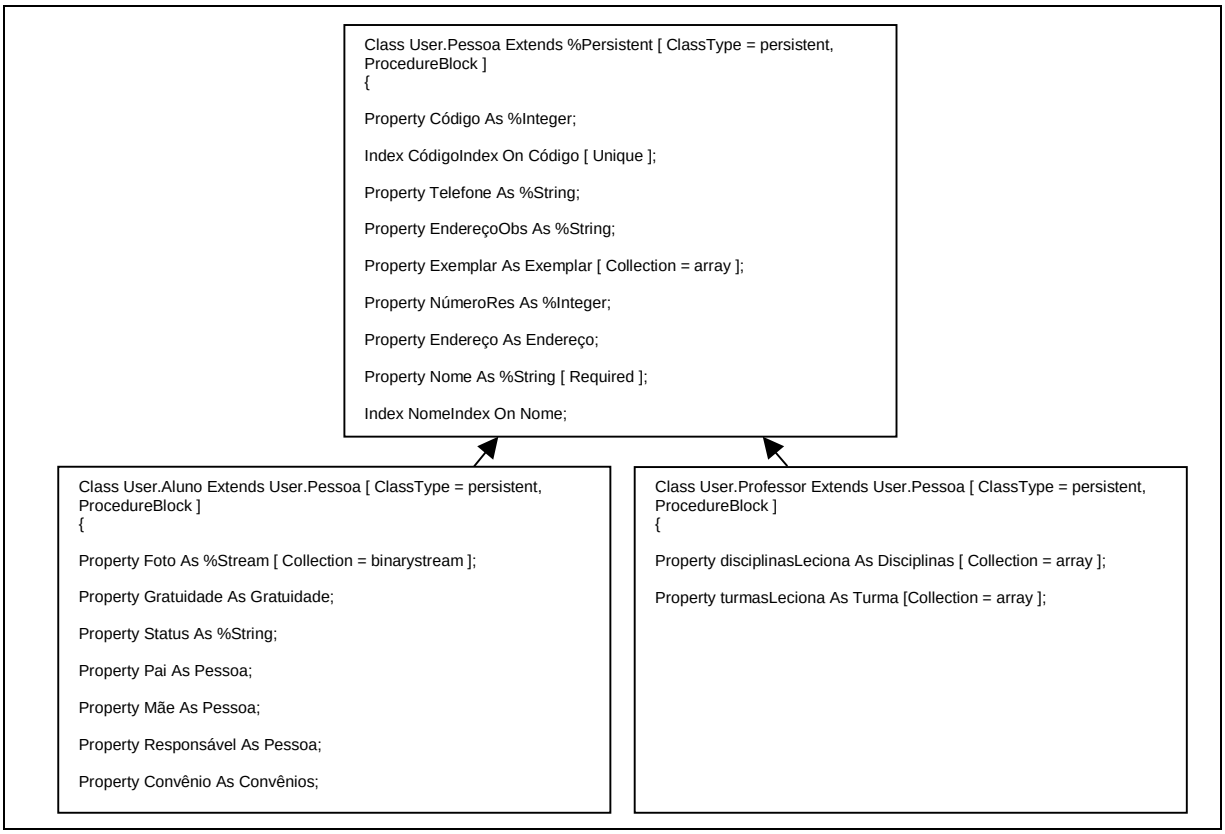
As *queries* são definições, muito parecidas com os métodos. Porém são acessadas de forma relacional. É uma das formas de acesso aos dados no Caché. Através dessas *queries* é possível gerar consultas SQL dinâmicas em um BDOO. No quadro 6 pode-se observar a estrutura de uma *querie* implementada no BDC.

```
Query Estados(nome As %String = "") As %SQLQuery(CONTAINID = 1, ROWSPEC =
"ID:%Integer,Nome:%String(MAXLEN=30), nomePais:Pais", SELECTMODE = "RUNTIME")
{
 SELECT ID, Nome, nomePais->nome
 FROM Estado
 WHERE (Nome %STARTSWITH :nome)
 ORDER BY Nome
}
```

Quadro 6 - Estrutura de uma *querie*

2.4.9 Herança

É a capacidade de derivar objetos de uma classe (superclasse). As classes derivadas implementam os atributos e métodos da superclasse, além de implementar atributos e métodos específicos de sua própria classe. A relação da subclasse com a superclasse é de subordinação (INTERSYSTEMS, 2005b). Por exemplo, a classe `Aluno` estende da classe `Pessoa`, pois `Aluno` "é uma" `Pessoa`. `Professor` estende da classe `Pessoa`, pois também "é uma" `Pessoa`. A herança neste caso faz com que as classes `Aluno` e `Professor` implementem os mesmos atributos e métodos da classe `Pessoa`, não havendo a necessidade de repetir os métodos comuns entre as duas classes. Apenas atributos e métodos específicos de cada classe devem ser implementados. No quadro 7 é mostrado um exemplo de herança no Caché. A classe `Aluno` e `Professor` herdam da classe `Pessoa`. Os atributos e métodos comuns às classes `Aluno` e `Professor` são implementados na classe `Pessoa`, os demais atributos e métodos peculiares a cada uma das classes são implementados em suas próprias classes.



Quadro 7 - Herança no Caché

2.4.10 Encapsulamento

Um objeto pode conter métodos públicos e privados. Os métodos públicos podem ser acessados por outros objetos e os privados apenas por métodos do próprio objeto. Assim, a aplicação não tem necessidade de saber como é o funcionamento interno do objeto, ela apenas trata dos métodos que são públicos, o que um método privado executa internamente é irrelevante para a aplicação. Com o encapsulamento, as classes podem ser melhoradas sem que haja necessidade de nenhuma alteração na aplicação (INTERSYSTEMS, 2005b).

Em um posto de gasolina não necessitamos conhecer o funcionamento interno da bomba de combustível, apenas espera-se dela que encha o tanque e que calcule o valor a pagar. No decorrer do tempo, a bomba pode sofrer mudanças internas para garantir melhor eficiência. Essa mudança não interfere na função da bomba que é de abastecer o veículo e calcular o valor. Traçando uma relação com métodos públicos de uma classe, tem-se na classe bomba, o método encher o tanque e calcular o valor com sendo públicos e demais métodos como pegar o combustível do reservatório sendo métodos privativos da classe.

2.4.11 Polimorfismo

É o principio pelo qual duas ou mais classes derivadas de uma superclasse possam invocar métodos com a mesma assinatura, porém com comportamentos diferentes. Por exemplo, tem-se a superclasse `Figura` e as subclasses `Quadrado` e `Círculo`. Na superclasse `Figura` tem-se o método `desenhar`, esse método é implementado de maneiras diferentes em cada uma das subclasses. Quando é referenciado um método da superclasse é verificado em tempo de execução qual classe derivada que executará o método. Esse processo é chamado de ligação tardia. É necessário que os métodos tenham a mesma identificação (assinatura) (RICARTE, 2001).

Segundo Nassu (2006, p. 70 apud BORBA, 1999), “polimorfismo pode ser definido como um mecanismo que permite que o nome de um mesmo método possa ser definido em várias classes, com implementações diferentes em cada uma delas”.

2.4.12 Java

O Caché da suporte a Java e o acesso se da de três diferentes formas:

a) através do acessso SQL via JDBC;
b) através de projeções Java, onde as classes em Caché são projetadas como classes Java. Porem, neste caso as classes devem ser criadas no Caché para depois serem projetadas;
c) e a última forma de acesso é com EJB.

As projeções fazem com que os desenvolvedores Java ganhem tempo quando da persistência de seus objetos (INTERSYSTEMS, 2002a).

2.4.13 Visual Basic

Qualquer aplicação capaz de instanciar componentes *ActiveX* é capaz de instanciar objetos diretamente do BDC sem nenhum tipo de mapeamento. Dentro do VB é usado o componente *CacheObject* para manipular os objetos do BDC. No quadro 8 observa-se alguns exemplos do uso do CacheObject dentro do VB.

Código	Descrição
Public m_factory As CacheObject.Factory	Declaração do fabricante de objetos
Public m_object As CacheObject.ObjInstance	Declaração de uma instância de objetos
Set m_object = m_factory.OpenId(“User.Aluno”, 1)	Objeto Aluno com ID = 1 é atribuido a variável m_object
Set m_object = m_factory.New(“User.Aluno”)	Criação de um novo objeto de Aluno
Set m_object.Nome = “Juliano Walter Brune”	Atributo Nome do objeto aluno recebe uma *String*

Quadro 8 - Uso do *CacheObject* no VB

2.4.14 ODBC

ODBC é a mais popular forma de acesso a bancos de dados, pois abstrai detalhes da conexão como: endereço do servidor, porta e drivers. Essas informações são encapsuladas através da fonte de dados(*datasource*). Sua função é a de transformar requisições SQL genéricas em chamadas específicas ao banco de dados associado. Logo na instalação do Caché, as configurações necessárias para a conexão ODBC são realizadas. Por padrão

aparecem os *datasources* *CacheSample* e *CachéUser*. Essas duas fontes de dados permitem o acesso aos *namespaces* "*Samples*" e "*User*" (AFONSO, 2003).

2.4.15 Rational Rose

O Caché *RoseLink* é um *add-in* fornecido pela Intersystems à popular ferramenta de modelagem de objetos Rational Rose. Ele fornece toda a interface de engenharia para ida e volta entre o Caché e o Rational Rose. Classes construídas no Rational Rose podem ser importadas para o Caché e vice versa (SOUZA, 2005, p. 29).

2.4.16 Dreamweaver

O Caché vem com um *add-in* para o ambiente de projeto de páginas web do Dreamweaver, dando aos usuários uma maneira de criar páginas em CSP utilizando recursos deste editor de páginas web.

2.4.17 Linguagens suportadas

O Caché suporta duas linguagens de programação, o COS e o Caché Basic, muito similar à linguagem basic. Ambas dão suporte a objetos, SQL e acesso direto aos dados (multidimensional).

2.4.17.1 COS

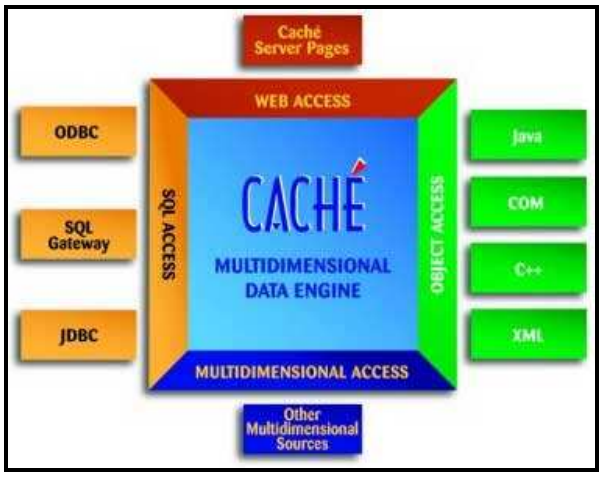
COS é a linguagem de programação original do Caché. A linguagem permite a mistura de métodos de acesso aos dados, podendo acessá-los como objetos, como tabelas relacionais (SQL) ou como arrays multidimensionais.

2.4.17.2 Basic

Incorporado ao Caché, a linguagem Caché basic veio para fornecer aos usuários do visual basic uma maneira fácil de começar a utilizar o Caché. O Caché basic inclui extensões que permitem que ele rode na maquina virtual do Caché, possibilitando assim como no COS, o acesso aos dados através de objetos, SQL ou multidimensionalmente. Rotinas escritas em Caché basic podem chamar rotinas escritas em COS e vice versa (INTERSYSTEMS, 2002c).

2.4.18 As formas de acesso aos dados no Caché

O Banco de dados Caché suporta o acesso aos seus dados de três diferentes formas. Possui o acesso através de objeto, acesso SQL (Relacional) e acesso direto (Multidimensional) (figura 3).



Fonte: Intersystems (2004, p. 1).

Figura 3 - Formas de acesso aos dados no Caché

2.4.18.1 Acesso através de objetos

Uma das formas para acessar e manipular objetos é através dos métodos. De forma geral, esse acesso é o mais conhecido por desenvolvedores de aplicações OO. No quadro 9 é apresentado um exemplo de acesso através de objeto. O método recebe o ID do objeto, chama o objeto que possui esse ID, armazena no atributo `Status` do objeto o valor “Matriculado” e

posteriormente salva o objeto.

```
Method atulizarAluno(idAluno As %Integer)
                                    {
Set objAluno = ##class(User.Aluno).%OpenId(idAluno)
Set objAluno.Status = "Matriculado"
do objAluno.%Save()
}
```

Quadro 9 - Acesso através de objetos

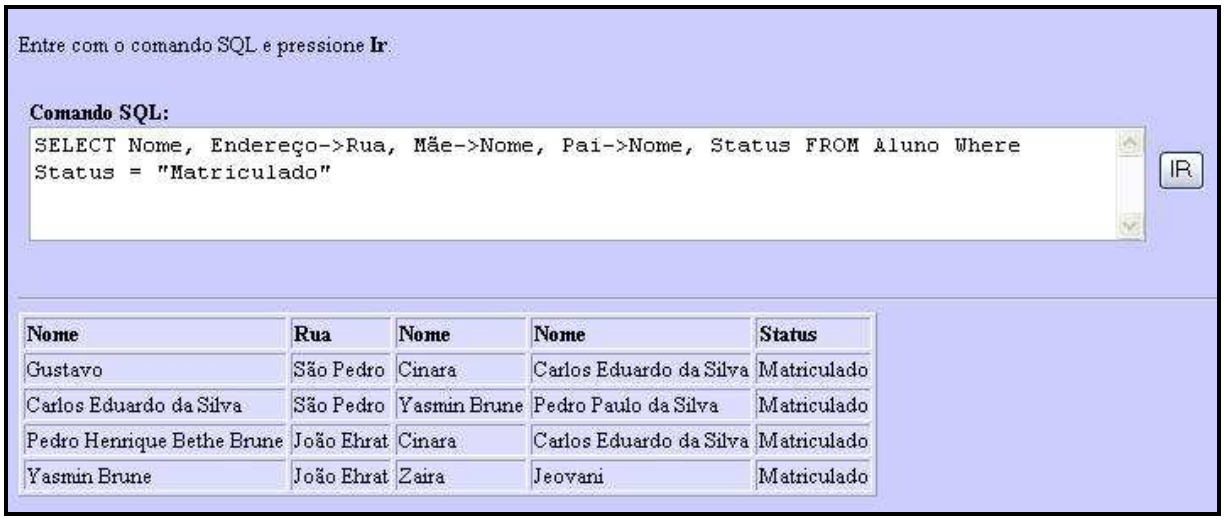
2.4.18.2 Acesso através de SQL

O SQL se tornou um padrão para consultas em banco de dados. Porém, mesmo ele tendo origem nos BDR, não fica restrito apenas ao modelo relacional. O Caché suporta SQL padrão como linguagem de consulta e de atualização. O SQL no Caché foi estendido para incluir capacidades de objetos. Na figura 4 é mostrado o acesso ao objeto `Aluno` através do SQL. Nota-se que os atributos `Endereço`, `Mãe` e `Pai` do objeto `Aluno` listaram apenas números. Esses números são ID de outros objetos referenciados pelo objeto `Aluno`.

Nome	Endereço	Mãe	Pai	Status
Gustavo	1	3	1	Matriculado
Carlos Eduardo da Silva	1	5	1	Matriculado
Pedro Henrique Bethe Brune	2	3	1	Matriculado
Yasmin Brune	2	3	1	Matriculado

Figura 4 - Acesso ao objeto `Aluno` através do SQL

Na figura 5 é mostrado o acesso sobre o mesmo objeto `Aluno` suprimindo o ID. Uma característica do Caché é que, sempre que é definida uma classe no BD, é fornecido automaticamente acesso total SQL aos dados.



Nome	Rua	Nome	Nome	Status
Gustavo	São Pedro	Cinara	Carlos Eduardo da Silva	Matriculado
Carlos Eduardo da Silva	São Pedro	Yasmin Brune	Pedro Paulo da Silva	Matriculado
Pedro Henrique Bethe Brune	João Ehrat	Cinara	Carlos Eduardo da Silva	Matriculado
Yasmin Brune	João Ehrat	Zaira	Jeovani	Matriculado

Figura 5 - Acesso ao objeto `Aluno` através do SQL suprimindo o ID

Segundo Arbegaus (2003) o Caché apresenta três diferentes interfaces para o acesso SQL:

a) ODBC;

b) JDBC;

c) OCI.

2.4.18.3 Acesso direto aos dados

Uma referência direta ao BD é na verdade uma referência a um array multidimensional precedida pelo caractere circunflexo “^”. O quadro 10 demonstra como é feito o acesso direto aos dados no caché. Este exemplo foi feito utilizando-se do terminal do Caché.

```
USER>Set ^Pessoa(“Juliano”) = “Estudante da Furb”
USER>Set x=^Pessoa(“Juliano”)
USER>W X
Estudante da Furb
USER>
```

Quadro 10 - Acesso direto aos dados no Caché

2.4.19 Ferramentas do caché

O BDC disponibiliza algumas ferramentas para criação e manutenção de sistemas. Nos subitens seguintes serão mostradas essas ferramentas.

2.4.19.1 Caché Studio

O Caché Studio é a IDE do Caché, disponibilizada junto ao próprio BDC. No Caché Studio são definidas as classes e criada todas as regras do negócio. No Caché Studio além da definição das classes, pode-se criar as páginas CSP para acesso web. Possui checagem de *sintaxe* e *debug*. Na figura 6 tem-se uma visão de uma aplicação desenvolvida no ambiente Caché Studio.

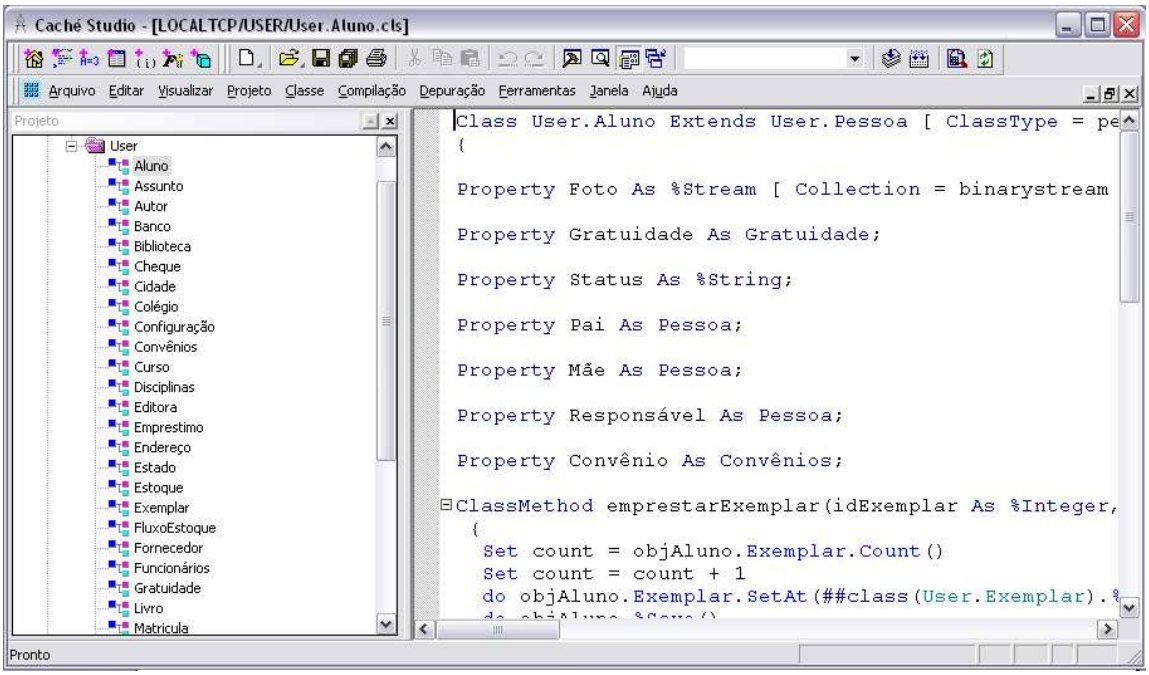


Figura 6 - IDE Caché Studio

2.4.19.2 Caché explorer

O Caché Explorer gerencia/administra o BD com suas classes associadas, globais e rotinas. Nele pode-se visualizar os *namespaces* e as globais. A figura 7 mostra o ambiente Caché explorer.

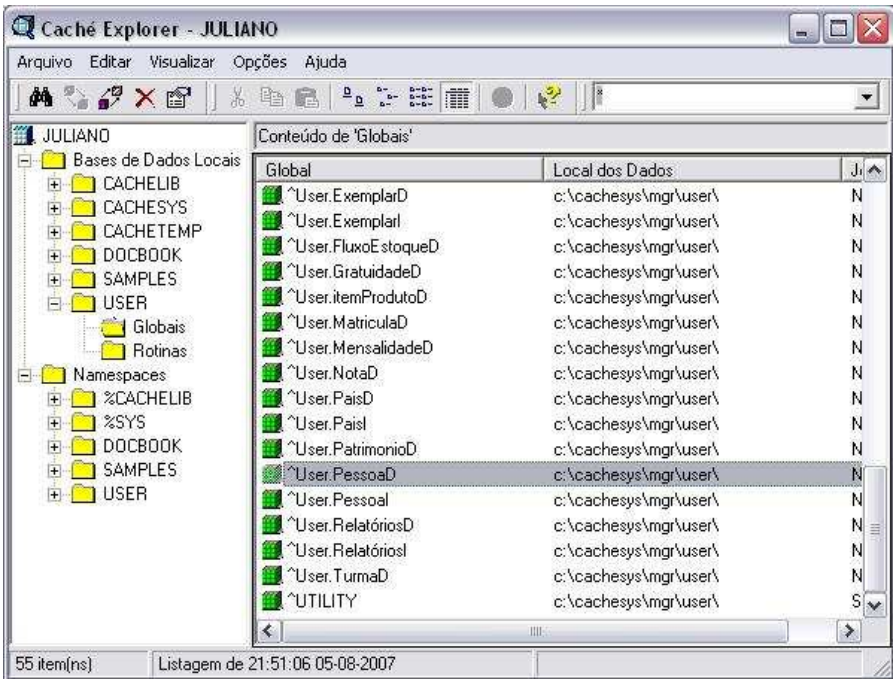


Figura 7 - Ambiente Caché Explorer

2.4.19.3 Gerenciador Sql

Segundo Intersystems (2005a) “com o gerenciador SQL você pode gerenciar todos os aspectos de acesso relacional ao Caché, por exemplo, trabalhar em tabelas, views, store procedures. A figura 8 mostra o gerenciador SQL.

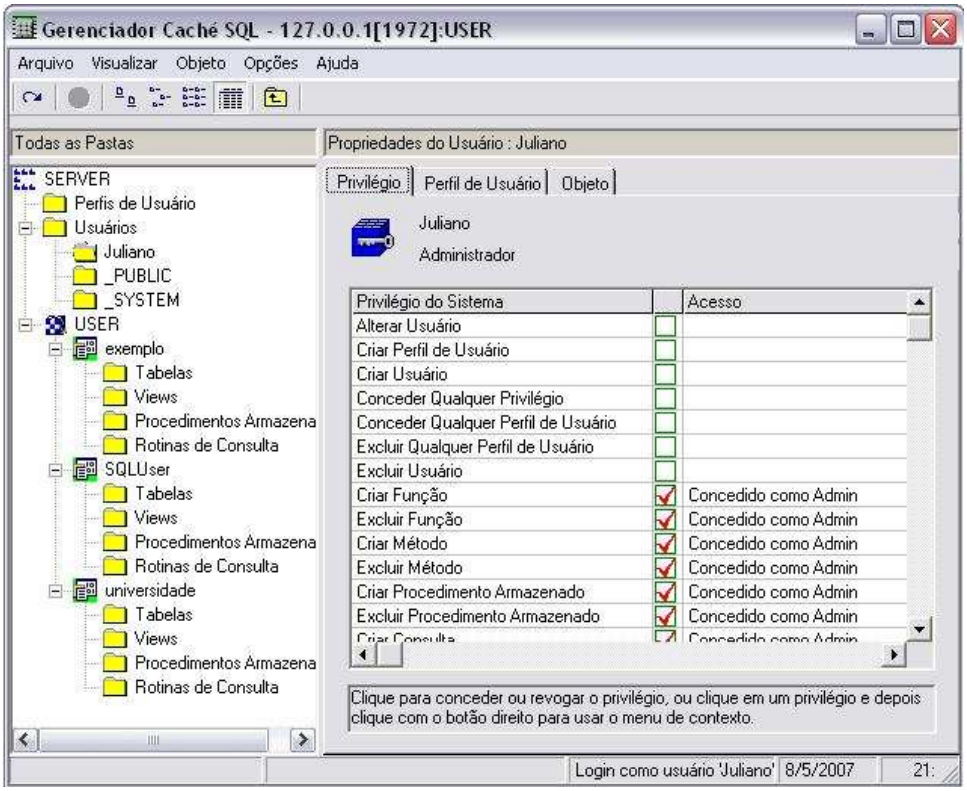


Figura 8 - Gerenciador SQL

2.4.19.4 Caché Terminal

É o terminal de manipulação e consulta do Caché. Nele pode-se executar comando em caractere no formato COS. Através do terminal pode-se acessar e manipular uma global de forma direta (multidimensionalmente).

2.4.19.5 Painel de controle

O painel de controle é utilizado para administrar o Caché. Pode-se criar contas de usuários no Caché, gerenciar backups das bases de dados, calcular o tamanho das bases e gerenciar espelhamentos. A figura 9 apresenta o painel de controle do Caché.

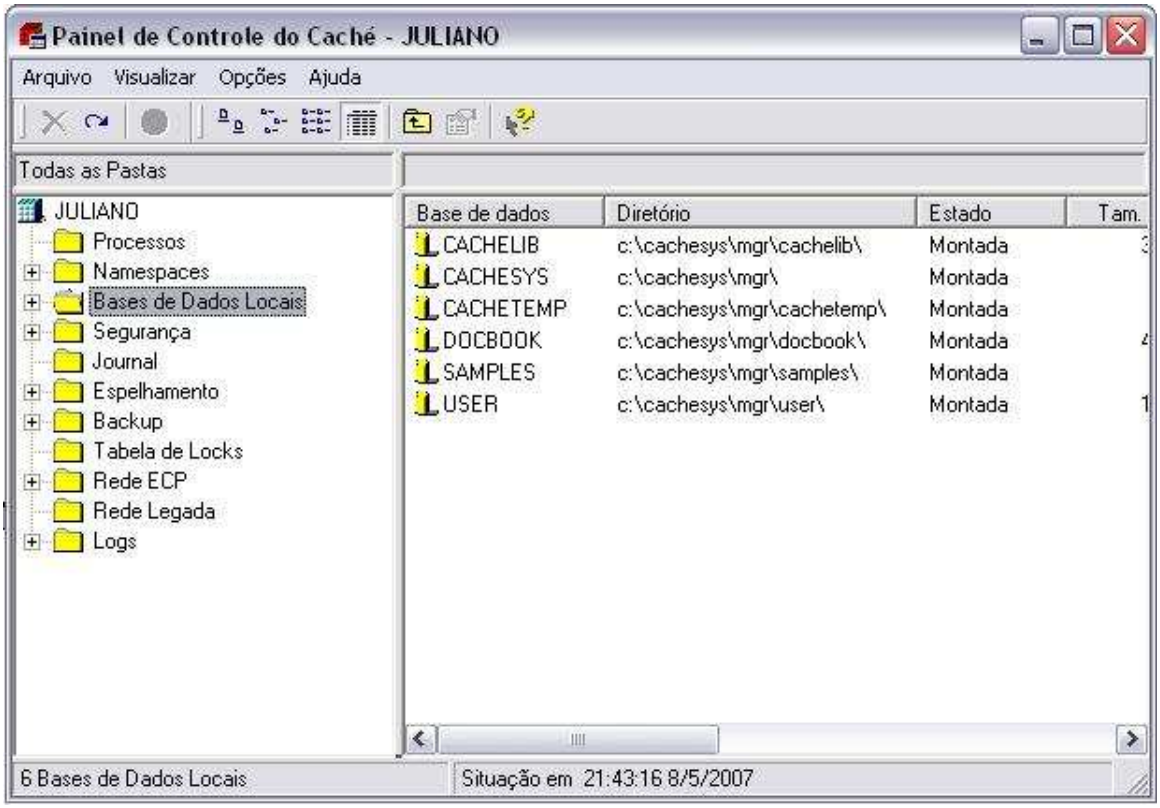


Figura 9 - Painel de controle do Caché

2.4.19.6 Documentação

O Caché possui uma completa documentação. A documentação é disponibilizada juntamente com a instalação do BDC. Está documentação não está traduzida para o português, porém é vasta e conta com tutoriais, materiais de referência e exemplos. A figura 10 apresenta a tela inicial da documentação do Caché.

Figura 10 - Documentação do Caché

2.5 TRABALHOS CORRELATOS

Em Obenaus (2000), é proposto um protótipo de uma aplicação comercial utilizando BDC com interface web. O trabalho explora características do banco de dados para desenvolvimento de aplicações voltadas à internet, como por exemplo, o Caché Studio.

Em Souza (2005), é apresentado um estudo sobre o BDC, seus conceitos e técnicas mais utilizadas. A implementação deste trabalho consiste na criação de um sistema de vendas para a web. A linguagem usada para o desenvolvimento da aplicação foi o CSP.

Em Arbegaus (2003), é apresentado um estudo do BDC. Foi desenvolvida uma aplicação de reservas de vaga em eventos acadêmicos via web utilizando o BDC com Java. Neste trabalho as regras de negócio foram escritas na IDE Caché Studio e a aplicação web foi

escrita em JSP, mostrando assim a interação do BDC com Java. A forma de acesso escolhida para o desenvolvimento do aplicativo foi através do SQL via JDBC.

3 DESENVOLVIMENTO DO TRABALHO

Neste capítulo tem-se a apresentação dos requisitos principais do trabalho, a especificação do sistema e os aspectos relacionados com sua implementação.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Em entrevista com a administração do Colégio Madre Francisca Lampel fez-se o levantamento de informações relevantes para o desenvolvimento do sistema. O sistema de gestão escolar proposto deve atender alguns requisitos funcionais principais dentre os quais observa-se:

a) os cadastros gerais de alunos;
b) cadastrar professores;
c) cadastrar funcionários;
d) cadastrar fornecedores;
e) cadastrar cursos;
f) cadastrar turmas;
g) cadastrar disciplinas;
h) cadastrar materiais;
i) cadastrar bancos;
j) cadastrar exemplares;
k) cadastrar notas;
l) cadastrar patrimônio;
m) controlar o estoque (entrada e saída de material escolar);
n) controlar as turmas;
o) cadastrar exemplares;
p) emprestar exemplar;
q) devolver exemplar;
r) executar a matrícula do aluno e gerar as parcelas de cobrança;
s) gerar os diários de classe;
t) emitir os boletins;

u) emitir relatórios de parcelas de cobrança em aberto;

v) disponibilizar mediante senha, um terminal *web* para acesso remoto ao sistema.

3.2 ESPECIFICAÇÃO

A especificação do sistema foi feita utilizando a UML. Os diagramas apresentados são os de casos de uso, o de classe e o de atividades. Os principais diagramas de casos de uso e de atividades foram separados por módulos para uma melhor visualização.

Na figura 11 está apresentado o caso de uso da matrícula.

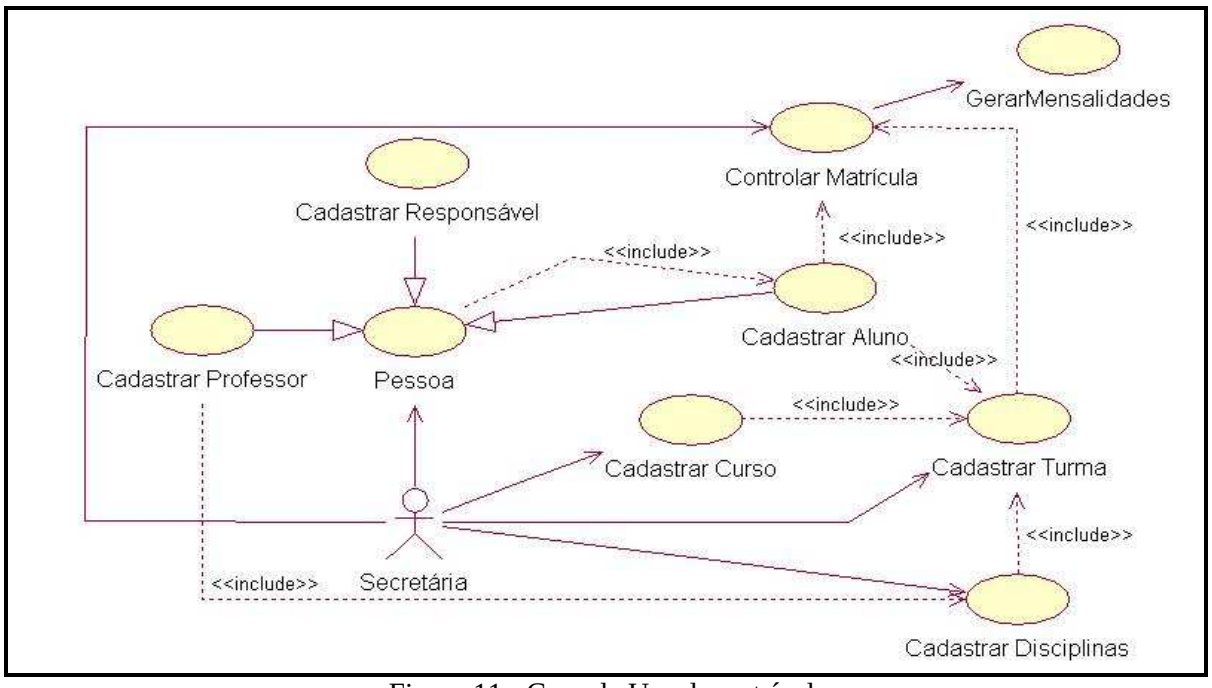


Figura 11 - Caso de Uso da matrícula

Na figura 12 está apresentado o caso de uso da biblioteca.

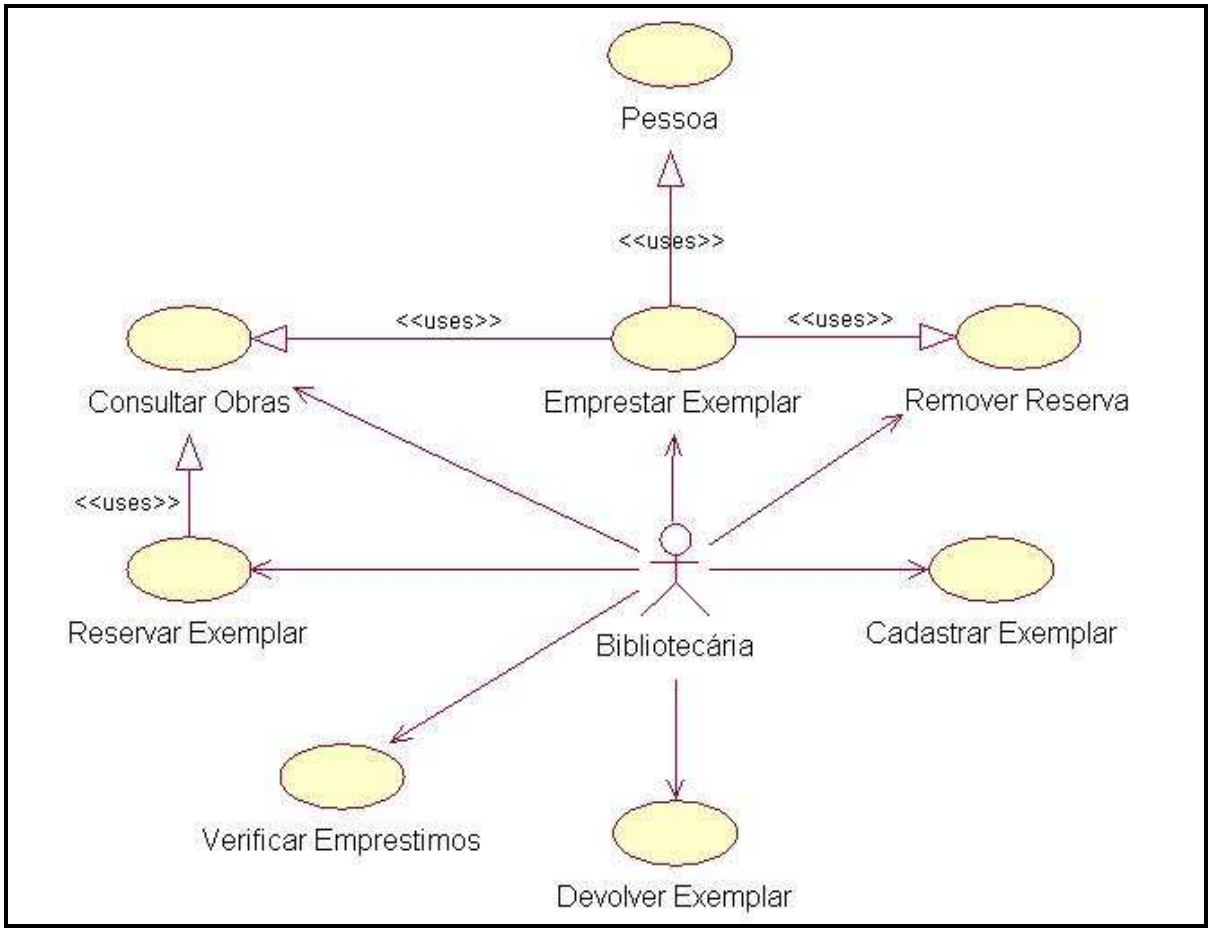


Figura 12 - Caso de uso da biblioteca.

Na figura 13 está apresentado o caso de uso do pagamento da mensalidade.

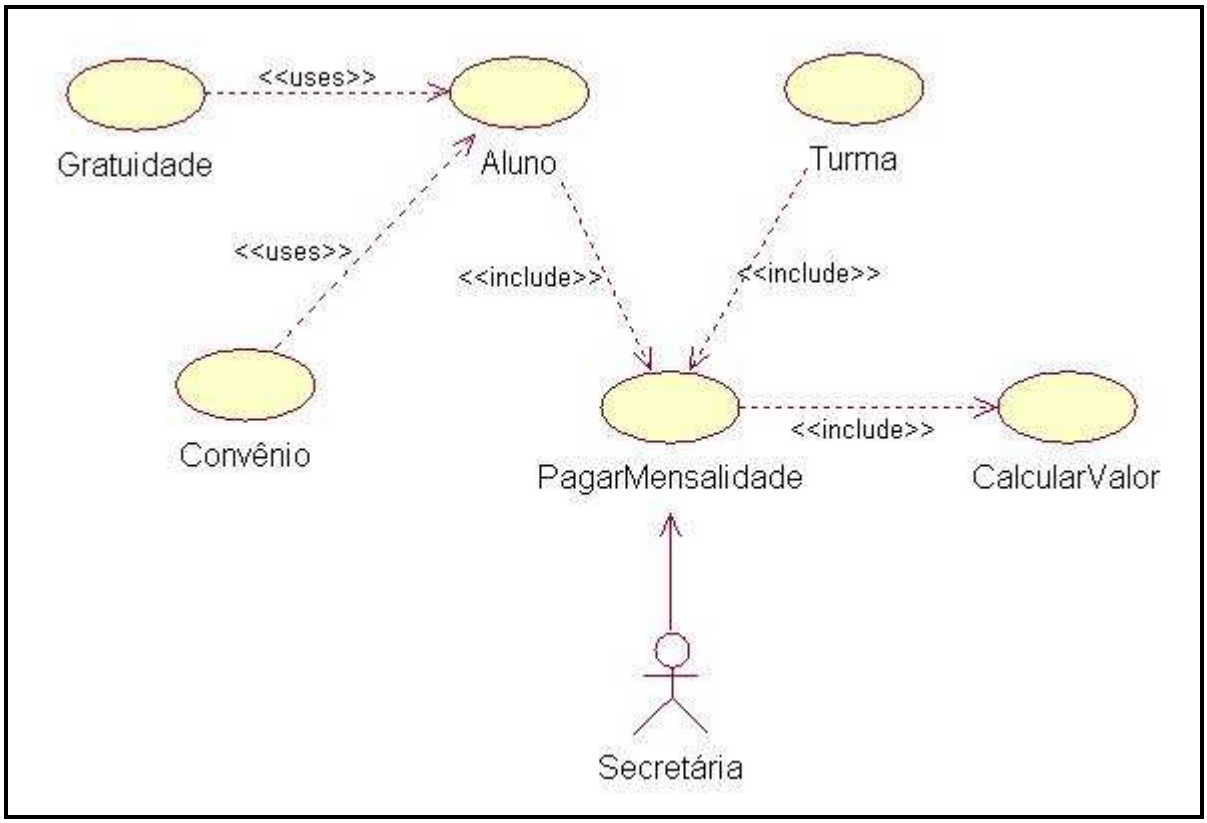


Figura 13 - Caso de uso pagamento da mensalidade

Na figura 14 está representado o diagrama de classes. O diagrama de casos de uso e o de classes foram feitos utilizando a ferramenta Rational Rose.

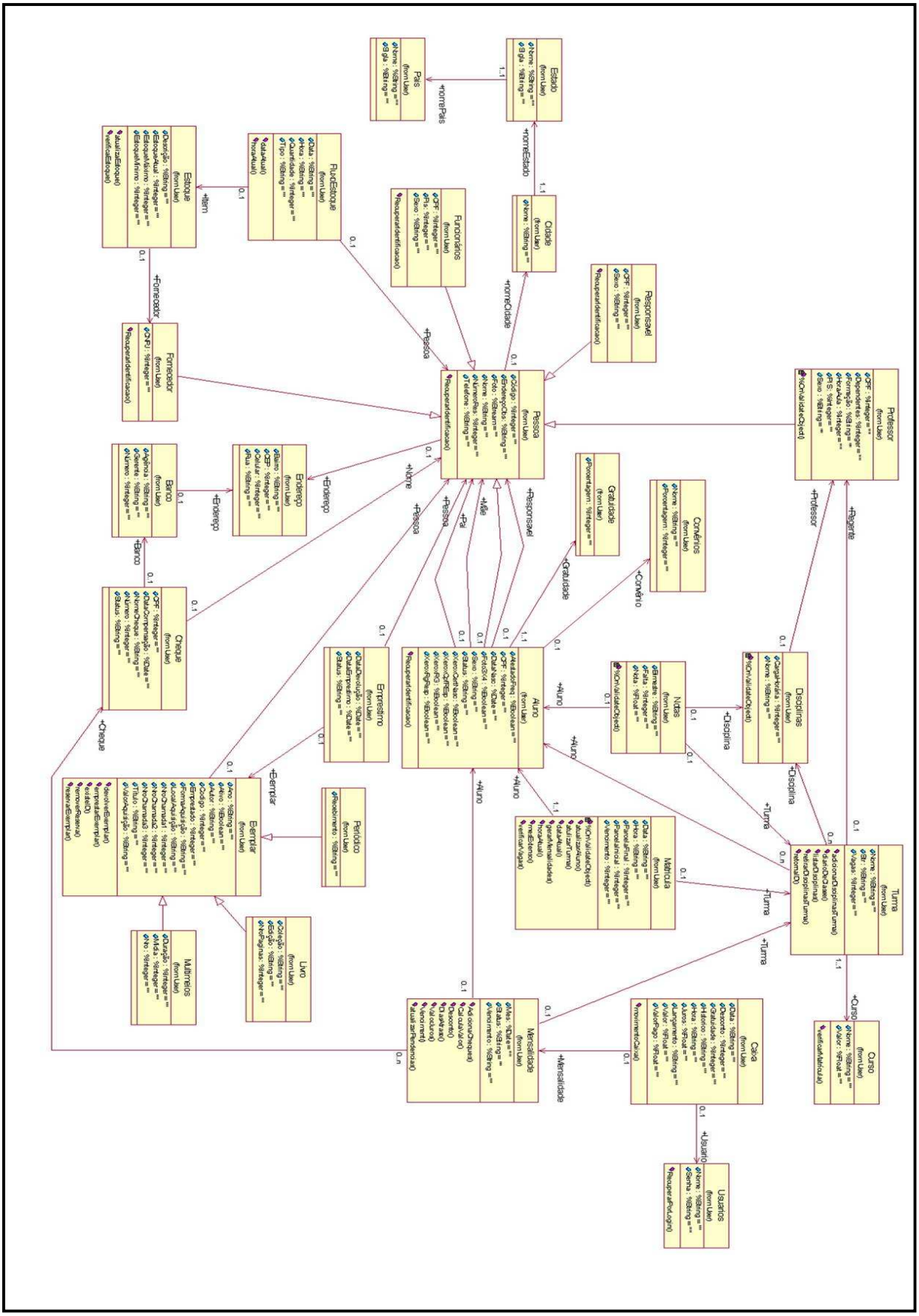


Figura 14 - Diagrama de classes do sistema

Na figura 15 está apresentado o diagrama de atividade da matrícula.

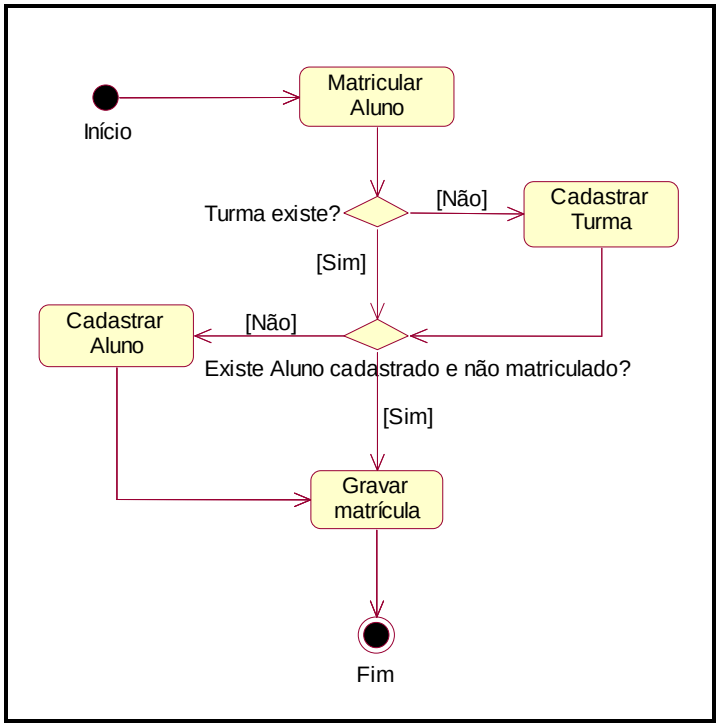


Figura 15 – Diagrama de atividades da matrícula

Na figura 16 está apresentado o diagrama de atividade do cadastro de aluno.

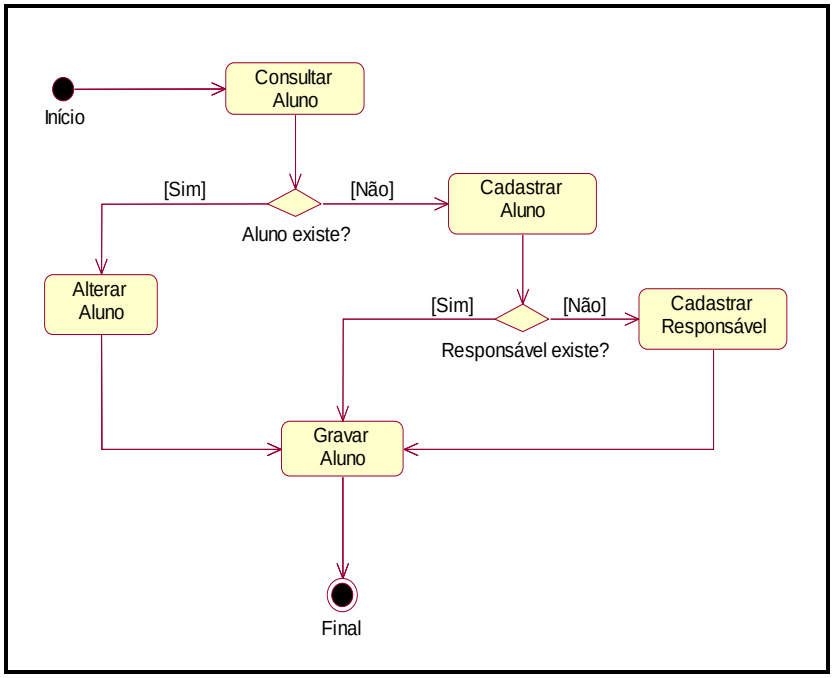


Figura 16 – Diagrama de atividades do cadastro de aluno

Na figura 17 está apresentado o diagrama de atividades do empréstimo de exemplares.

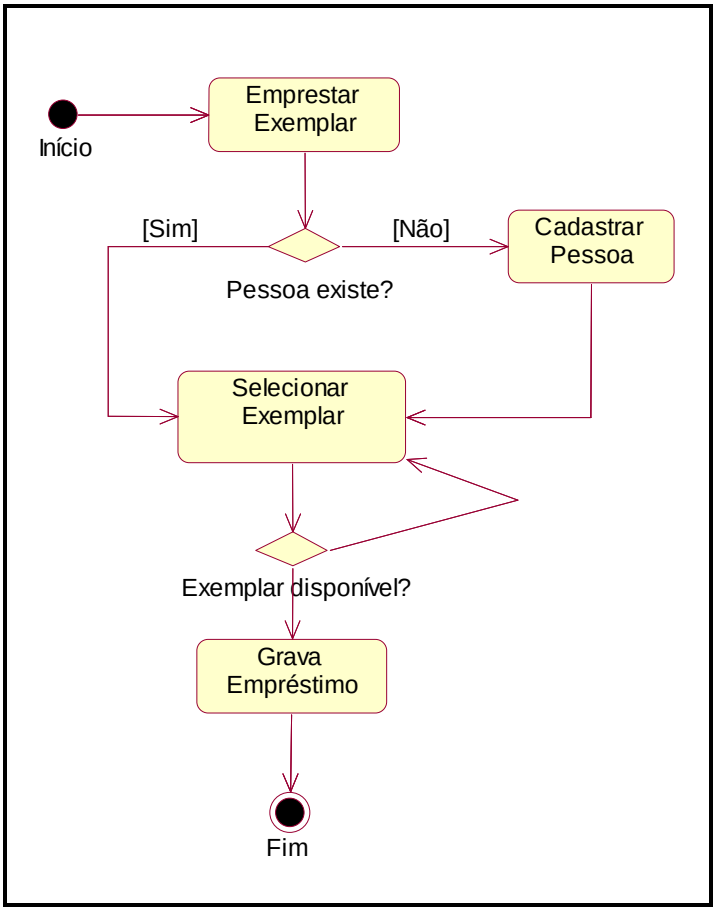


Figura 17 – Diagrama de atividades do empréstimo de exemplares

No quadro 11 é apresentada uma descrição detalhada das principais classes do sistema.

Classes	Detalhe
Aluno	Armazena dados relativos aos alunos
Banco	Armazena dados relativos aos Bancos
Cheque	Armazena dados relativos aos cheques enviados ou recebidos
Caixa	Armazena toda a movimentação financeira do sistema
Convênio	Armazena dados relativos aos convênios existentes ou aceitos
Cursos	Armazena dados relativos aos cursos disponíveis
Editora	Armazena dados sobre a editora do exemplar
Disciplina	Armazena dados sobre as disciplinas existentes nos cursos
Empréstimo	Realiza o empréstimo dos exemplares da biblioteca
Endereço	Armazena o endereço da classe a qual for agregada (é uma classe serial[1])
Estado	Armazena uma lista de Estados
Estoque	Armazena os produtos disponíveis e sua quantidade em estoque
Exemplar	Uma Superclasse estendida para as classes Livros, Multimeios e Periódicos
FluxodeEstoque	Armazena dados relativos as movimentações de entrada e saída de materiais
Fornecedor	Armazena dados relativos aos fornecedores
Funcionário	Armazena dados relativos aos funcionários
Gratuidade	Armazena dados relativos aos percentuais de gratuidade que serão dados
Livro	Armazena dados relativos aos livros
Matrícula	Armazena dados relativos a matrícula
Mensalidade	Armazena dados relativos aos boletos de cobrança
Multimeio	Armazena dados relativos aos multimeios
Notas	Armazena dados relativos as notas dos alunos
Pais	Armazena dados relativos aos pais ou responsáveis
Patrimônio	Armazena dados relativos ao patrimônio
Periódico	Armazena dados relativos aos periódicos
Pessoa	Uma Superclasse estendida para as classes Aluno, Professor, Pais, Fornecedores
Professor	Armazena dados relativos aos professores
Relatório	Responsável por organizar oois relatórios mais elaborados do sistema
Turma	Armazena dados relativos a turmas

Quadro 11 - Exemplificação das classes do sistema

[1] Classes seriais não possuem objetos próprios, são armazenadas juntamente com uma classe persistente (relação de agregação).

3.3 IMPLEMENTAÇÃO

Nesta etapa do trabalho será mostrado as ferramentas utilizadas para o desenvolvimento da aplicação, as técnicas utilizadas e os resultados obtidos.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento da aplicação foi utilizado o BDC como forma de armazenamento dos objetos, criação das classes e construção de toda a regra do negócio. O Caché disponibiliza uma IDE chamada Caché Studio onde são criadas as classes da aplicação. A partir do Rational Rose foi possível fazer a exportação das classes para dentro do banco de dados Caché.

O VB foi utilizado para criação da interface *desktop* ao usuário. A comunicação do VB com o Caché foi feita através da instanciação do *ActiveX CacheObject*, permitindo assim, que fossem instanciados objetos sem nenhum mapeamento prévio.

A figura 18 apresenta o caminho no Rational Rose para se importar classes criadas do Caché Studio. As figuras 19 e 20 completam os passos para a criação do diagrama de classes.

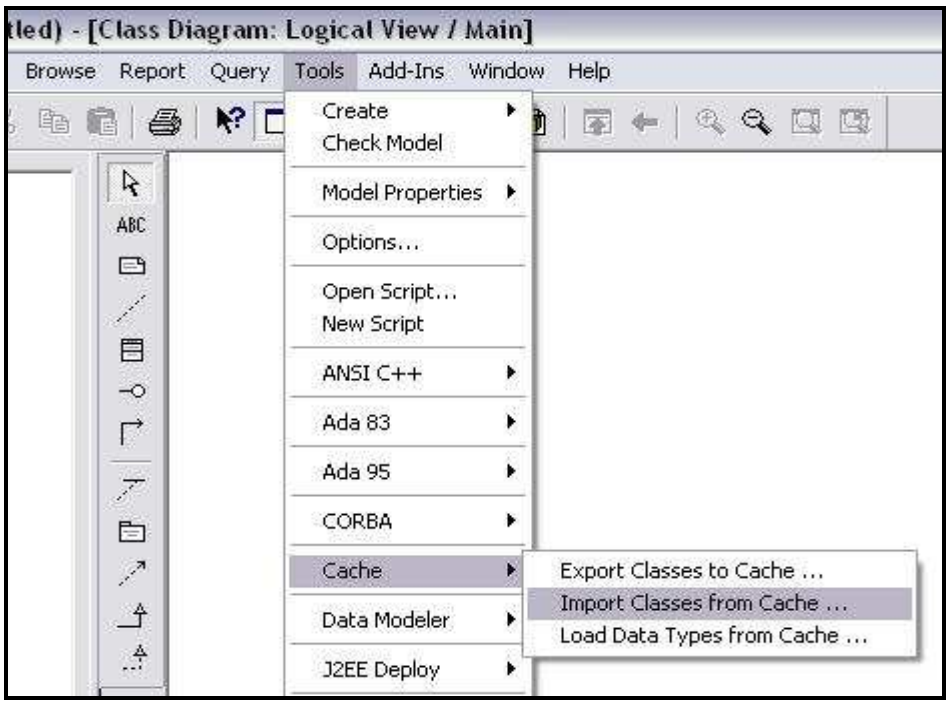


Figura 18 – Importação de classes do Caché no Rational Rose

Na figura 19 é apresentado o Rose Caché Link e a tela de conexão com o BDC.

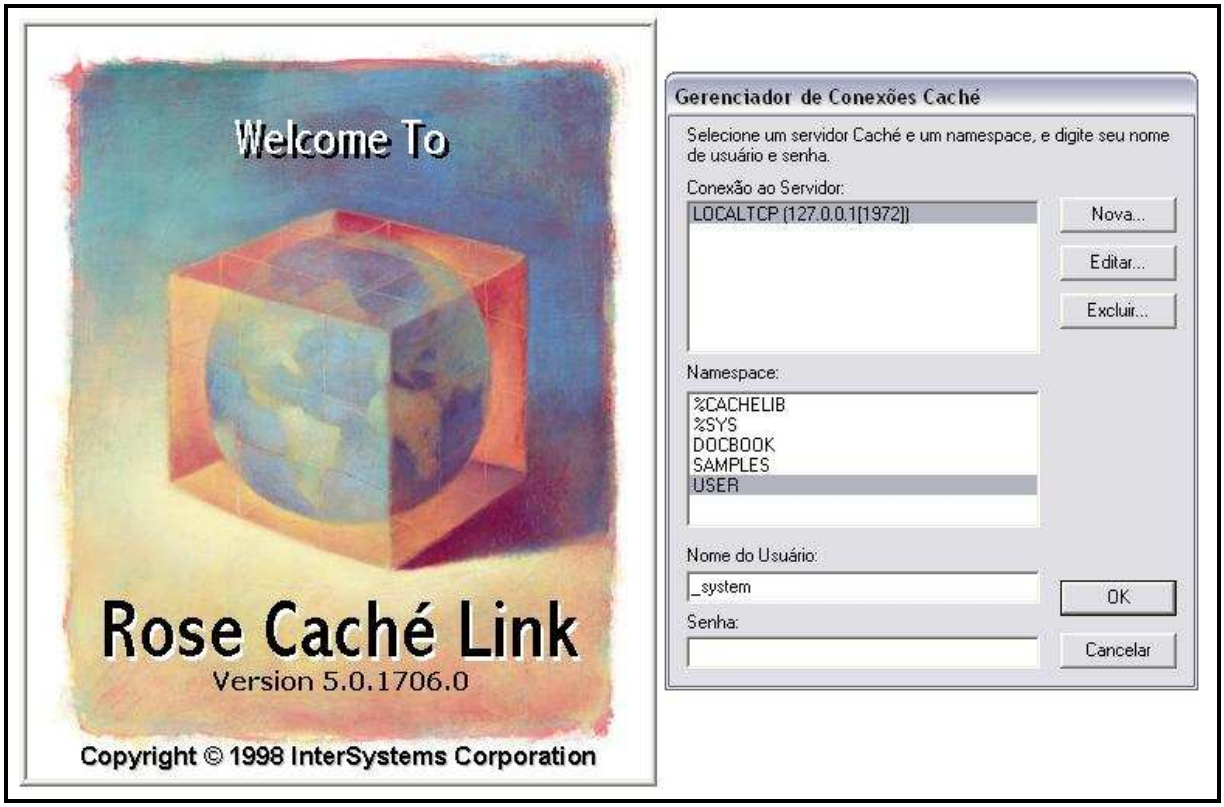


Figura 19 – Caché Rose Link

Na figura 20 é apresentada as classes do BDC, possibilitando ao usuário escolher sobre quais classes deseja criar o diagrama de classes.

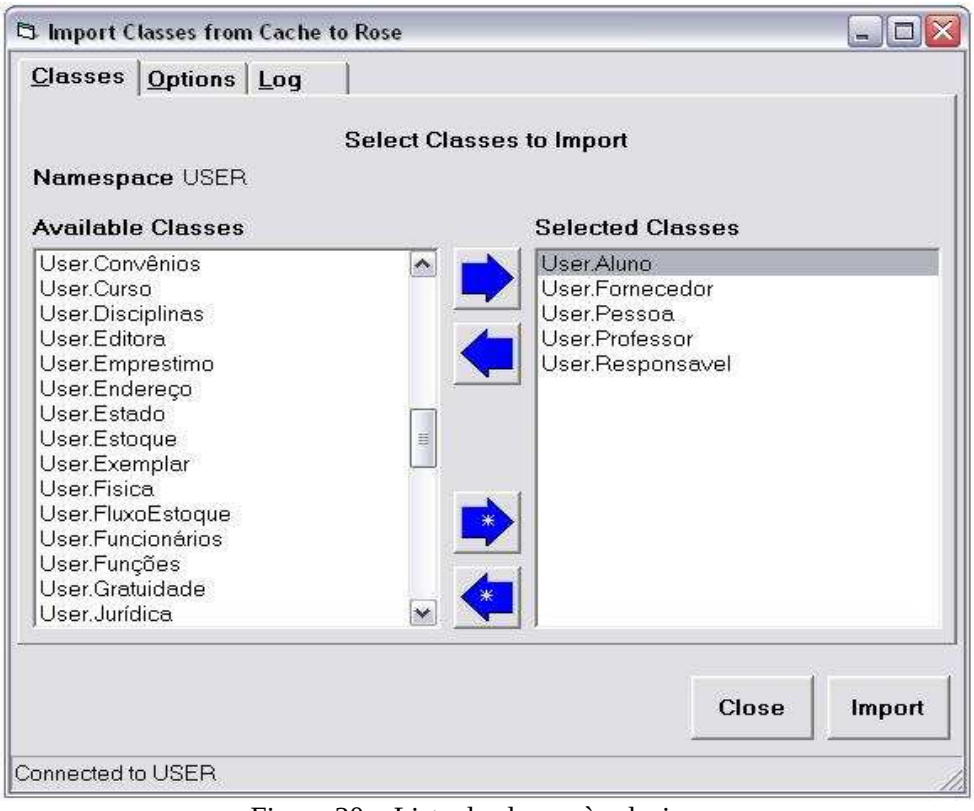


Figura 20 – Lista de classes à selecionar

Na figura 21 é apresentado o resultado da importação.

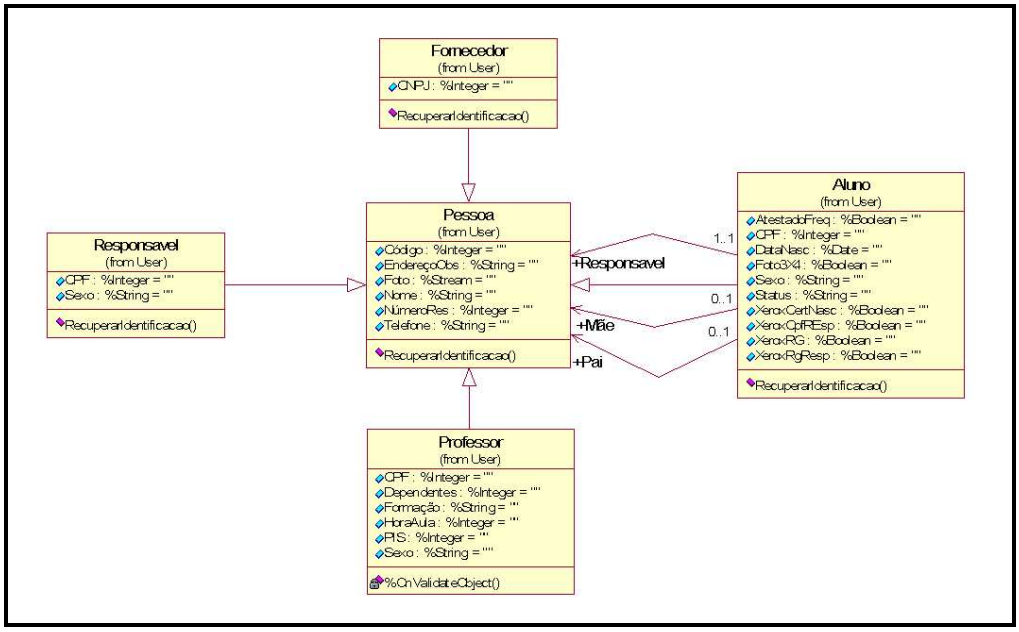


Figura 21 – Diagrama de classes

3.3.2 Operacionalidade da implementação

Na figura 22 é apresentada a tela de *login* do sistema.

Figura 22 - Tela de *login* do sistema

Na figura 23 é apresentada, a tela principal do sistema.

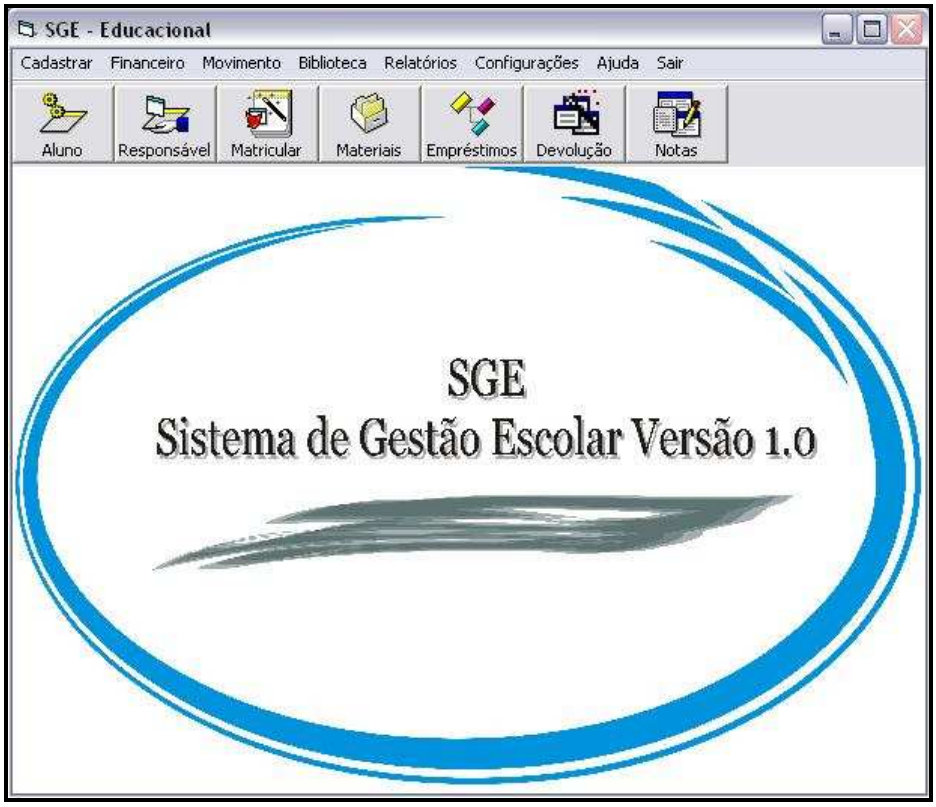


Figura 23 - Tela principal do sistema

Na figura 24 é apresentado a tela de cadastro de alunos (Objeto `Aluno`). Nota-se que todos os campos *combobox* mostram o valor de um atributo de outro objeto. As demais telas de cadastro são semelhantes aos da figura 24.

Figura 24 - Tela de cadastro de alunos

Na figura 25 é apresentada, a tela de localização de objetos. Em todo o sistema é usada essa mesma tela para localizar os objetos em diferentes momentos. No campo `Query` são listadas as *queries* criadas no BDC. No subcapítulo 2.2.8 é exemplificada a criação das *queries.*

Figura 25 - Tela de localização de objetos

Na figura 26 é apresentada, a tela de empréstimos de exemplares. O empréstimo é realizado associando-se um objeto de pessoa a um ou mais objetos de exemplares.

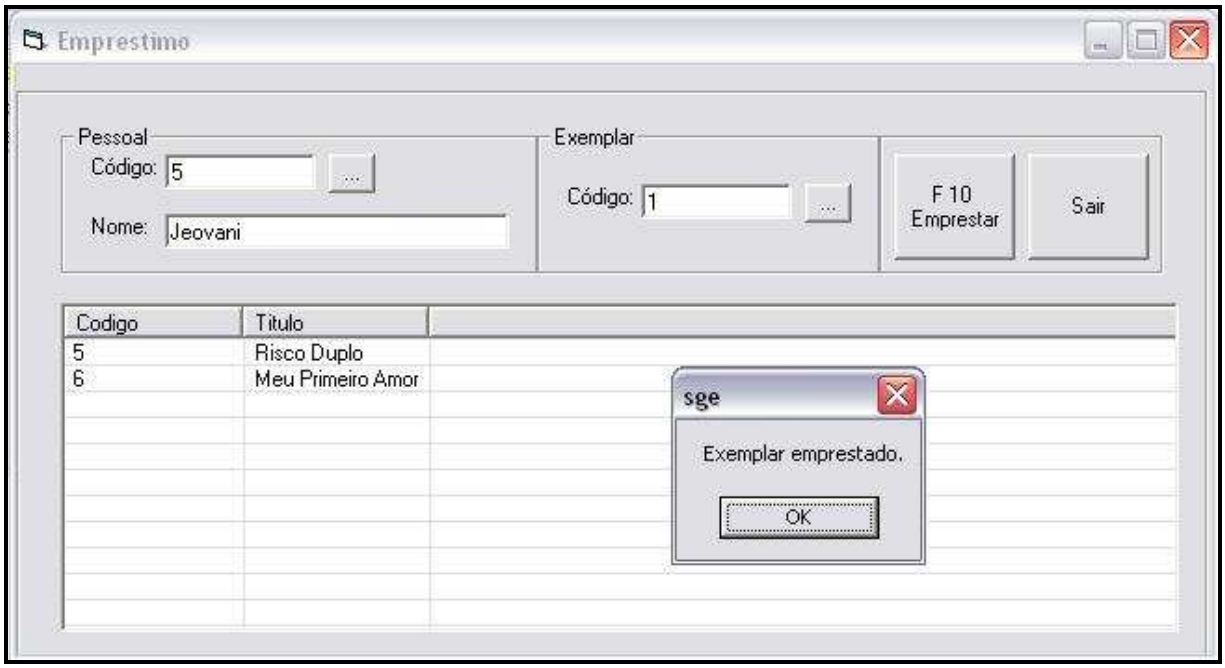


Figura 26 - Tela de empréstimos de exemplares

Na figura 27 é apresentada, a tela de movimento de materiais.

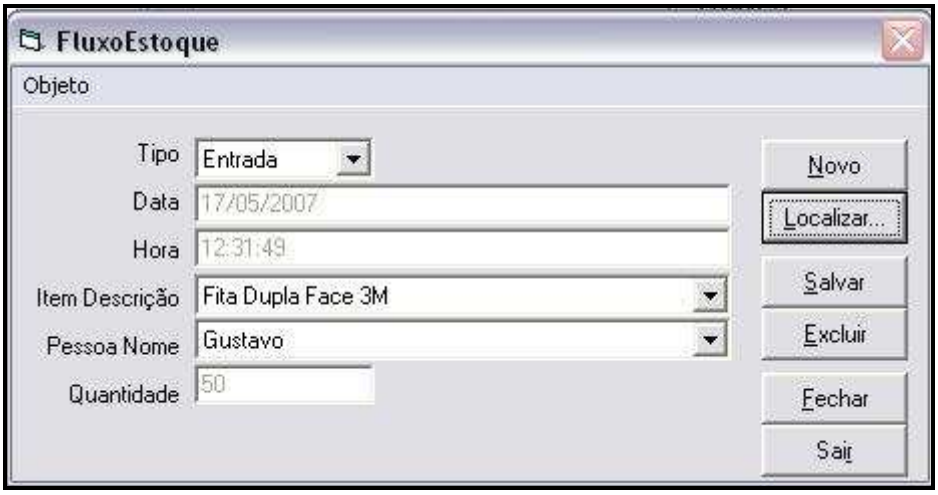


Figura 27 - Tela de fluxo de estoque

Na figura 28 é apresentada, a tela de cadastro de turmas. Observa-se nesse cadastro um exemplo de armazenamento de dados complexos relatado no sub-capítulo 2.2.1. A Classe `Turma` possui um atributo do tipo array de objetos de `Disciplina`. Na figura 28 pode-se observar a listagem de três objetos de `Disciplina`.

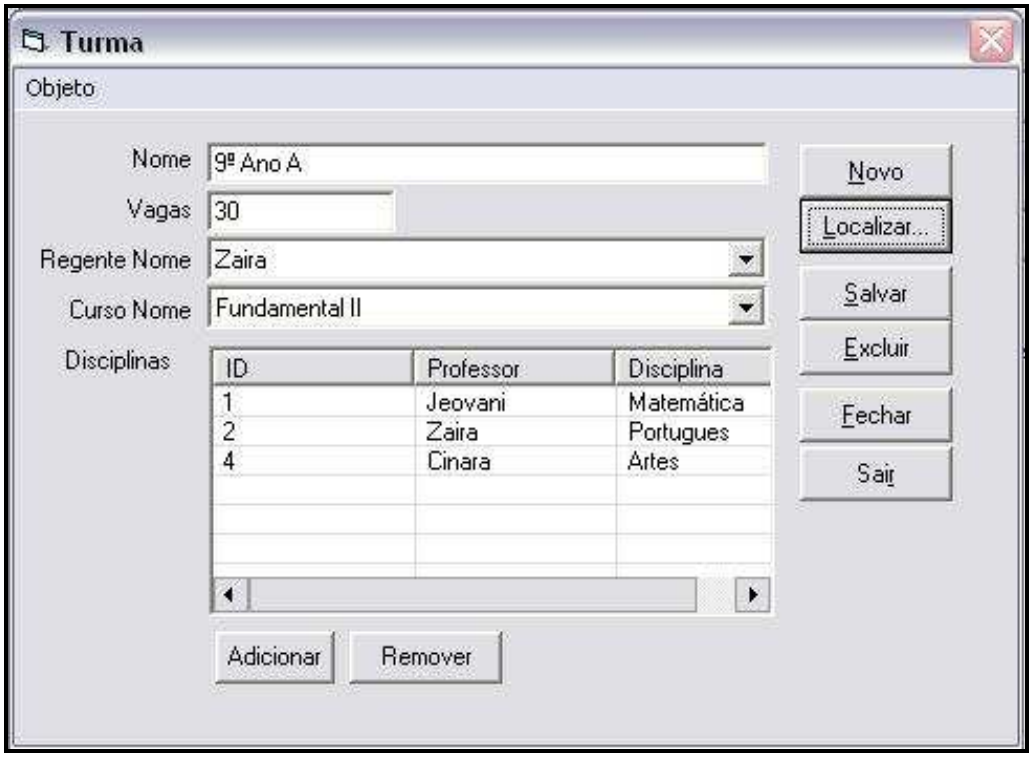


Figura 28 - Tela de cadastro de turmas

Na figura 29 é apresentada, a tela de pagamento das mensalidades, com suas deduções e acréscimos.

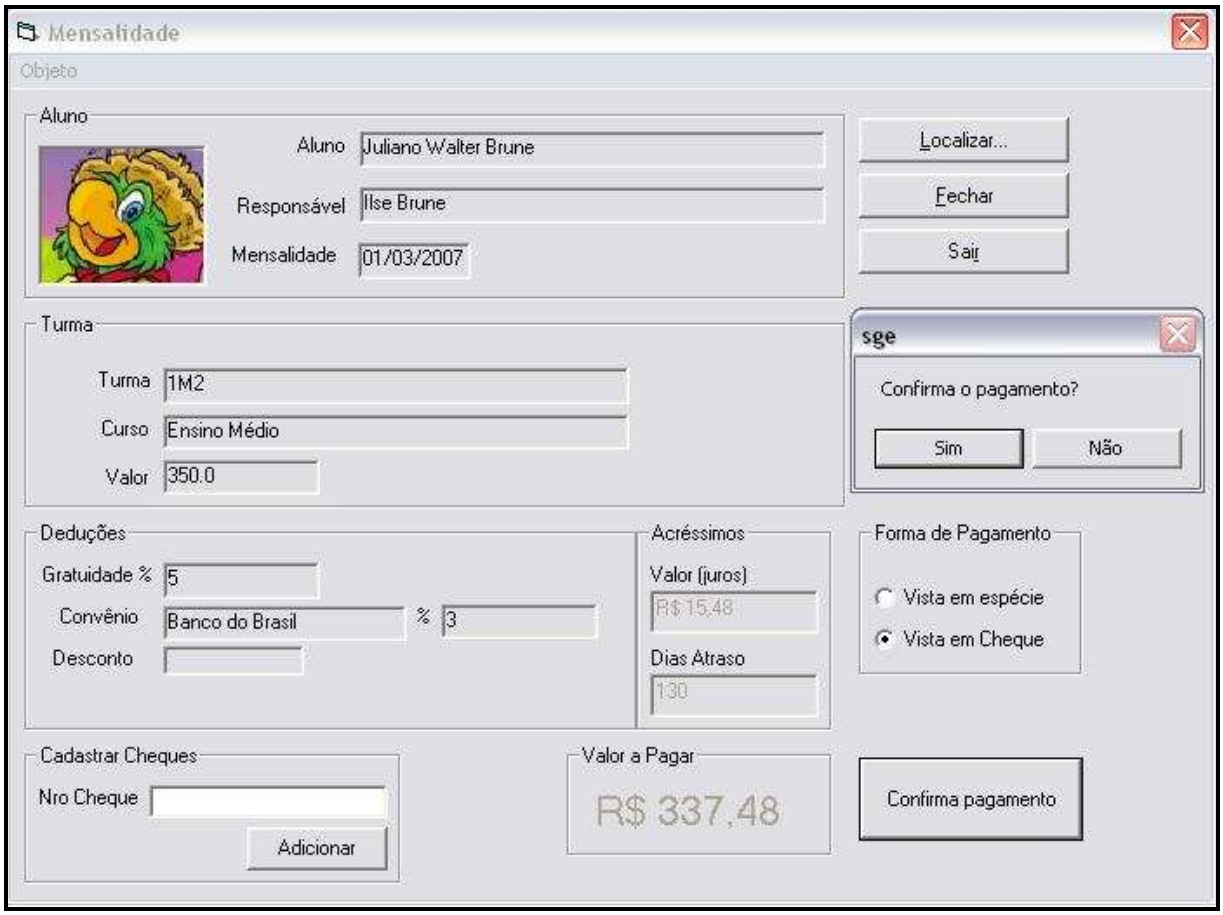
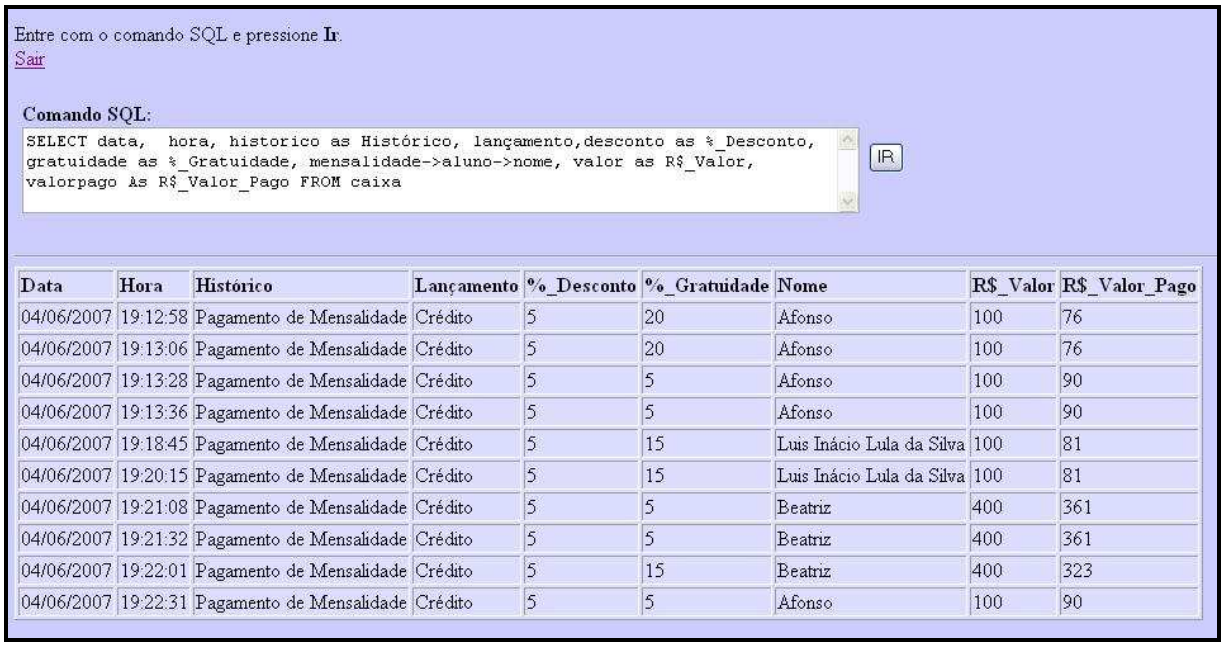


Figura 29 - Tela de pagamento da mensalidade

Na figura 30 é apresentada a tela de login para acesso web. Através do acesso pela Internet o administrador do sistema poderá contar com um console para inserção de comandos SQL.

Figura 30 - Tela de login para acesso web

Na figura 31 é apresentado o terminal web para acesso remoto ao sistema, utilizando-se de comandos SQL. Através desse terminal pode-se manipular todas as globais do sistema, executar consultas e atualizações dos registros.


Entre com o comando SQL e pressione **Ir**.
Sair

Comando SQL:

```
SELECT data,  hora, historico as Histórico, lançamento,desconto as %_Desconto,
gratuidade as %_Gratuidade, mensalidade->aluno->nome, valor as R$_Valor,
valorpago As R$_Valor_Pago FROM caixa
```

IR

Data	Hora	Histórico	Lançamento	%_Desconto	%_Gratuidade	Nome	R$_Valor	R$_Valor_Pago
04/06/2007	19:12:58	Pagamento de Mensalidade	Crédito	5	20	Afonso	100	76
04/06/2007	19:13:06	Pagamento de Mensalidade	Crédito	5	20	Afonso	100	76
04/06/2007	19:13:28	Pagamento de Mensalidade	Crédito	5	5	Afonso	100	90
04/06/2007	19:13:36	Pagamento de Mensalidade	Crédito	5	5	Afonso	100	90
04/06/2007	19:18:45	Pagamento de Mensalidade	Crédito	5	15	Luis Inácio Lula da Silva	100	81
04/06/2007	19:20:15	Pagamento de Mensalidade	Crédito	5	15	Luis Inácio Lula da Silva	100	81
04/06/2007	19:21:08	Pagamento de Mensalidade	Crédito	5	5	Beatriz	400	361
04/06/2007	19:21:32	Pagamento de Mensalidade	Crédito	5	5	Beatriz	400	361
04/06/2007	19:22:01	Pagamento de Mensalidade	Crédito	5	15	Beatriz	400	323
04/06/2007	19:22:31	Pagamento de Mensalidade	Crédito	5	5	Afonso	100	90

Figura 31 - Terminal web para acesso remoto ao sistema

No quadro 12 são descritos os relatórios mais importantes do sistema.

Relatórios Acadêmicos
Relação de alunos
Relação de alunos matriculados
Relação de alunos cadastrados
Boletins
Diário de classe
Relatórios Financeiros
Parcelas em aberto
Relação de alunos inadimplentes
Fluxo de Caixa
Cheques a compensar
Biblioteca
Relação do acervo
Relação de livros emprestados
Relação de alunos em débito
Administração
Relação de materiais em estoque
Relação de patrimônio
Relação do fluxo do estoque
Relação de colaboradores
Relação de alunos com gratuidade

Quadro 12 - Relatórios mais importantes do SGE

Na figura 32 é apresentada, a tela de relatório de aluno. Todos os relatórios obtém os dados a partir de classes do tipo *view*, implementadas no BDC. Através de uma conexão ODBC pode-se acessar essas classes para a obtenção dos dados. Dentro das classes do tipo *view* são programados os comandos SQL.

A formatação descrita na figura 28 é igual para todos os relatórios do sistema. Contendo o cabeçalho da escola, o nome do relatório e as colunas referentes.

Figura 32 - Tela de relatórios do sistema

Na figura 33 é apresentado os métodos responsáveis pelo empréstimo e devolução dos exemplares.

```
 //Cria objeto Emprestimo e passa os ID de Exemplar e de Pessoa

ClassMethod emprestarExemplar(idExemplar, idPessoa As %Integer) As Exemplar
  {
    Set objEmp = ##class(User.Emprestimo).%New()
    Set objEmp.Exemplar = ##class(User.Exemplar).%OpenId(idExemplar)
    Set objEmp.Pessoa = ##class(User.Pessoa).%OpenId(idPessoa)
    Set objEmp.Exemplar.Emprestado = 1
    Set objEmp.Exemplar.Pessoa = ##class(User.Pessoa).%OpenId(idPessoa)
    Set objEmp.Status = "Emprestimo"
    Set objEmp.DataEmprestimo =  $PIECE($HOROLOG,",",1)
    Set objEmp.DataDevolução =  $PIECE($HOROLOG,",",1)+15
    do objEmp.%Save()
    quit objEmp
  }

 //Devolve o Exemplar e muda o valor de "Emprestado" retirando o ID da Pessoa

ClassMethod devolverExemplar(IdExemplar As %Integer)
  {
     Set objExemp = ##class(User.Exemplar).%OpenId(IdExemplar)
     Set objExemp.Emprestado = 0
     Set objExemp.Pessoa = ""
     do objExemp.%Save()
  }
```

Figura 33 – Código fonte empréstimo e devolução

Na figura 34 é apresentado a *query* responsável por listar objetos da classe Pessoa, exemplificando assim, o acesso através de SQL.

```
//Query responsável por listar pessoas habilitadas a realizar empréstimos
//de exemplares. Alunos, professores, pais, funcionários podem realizar
//empréstimos, Fornecedores não. x__classname guarda informações sobre
//o tipo de PESSOA.

Query ExemplarPessoa(Nome As %String, I As %Integer) As %SQLQuery(CONTAINID = 1,
ROWSPEC = "Id: %Integer,Nome:%String(MAXLEN=30)", SELECTMODE = "RUNTIME")
 {
 SELECT Id,Nome FROM Pessoa
 WHERE (Nome %STARTSWITH :Nome) and (x__classname <> "~Fornecedor~")
 ORDER BY Nome
 }
```

Figura 34 – Código fonte *query* `ExemplarPessoa`

Na figura 35 é listada a função (VB) que acessa as queries criadas no Caché.

```
Public Function procurarExemplar(X)
    Principal.actionClose
    Set Principal.CacheQuery1.Factory = Principal.m_factory
    Principal.CacheQuery1.ClassName = m_classname
    Principal.CacheQuery1.MaxToDisplay = 30000
    Principal.CacheQuery1.StringOnlyMode True, DisplayMode
    id = Principal.CacheQuery1.FindId
    If id = "" Then Exit Function
    Set Principal.m_object = Principal.m_factory.OpenId(m_classname, id)
    If Principal.m_object Is Nothing Then
        MsgBox "Não foi possível abrir o objeto."
        Principal.actionClose
    End If
    If Principal.m_object.Emprestado And X = 1 Then
        MsgBox "Exemplar emprestado."
        Principal.actionClose
    Else
       If X = 1 Then
           frmEmprestar.mostrarExemplar (id)
        Else
          If Principal.m_object.Emprestado Then
             frmDevolver.mostrarExemplar (id)
          Else
             MsgBox "Exemplar já devolvido."
             Principal.actionClose
          End If
        End If
    End If
```

Figura 35 – Código fonte (VB) acesso a queries

Na figura 36 é apresentado o método que atualiza as vagas de cada turma e adiciona o aluno matriculado a sua turma.

```
//Quando efetuada a matricula, atualiza-se as vagas da turma
//e insere um objeto de Aluno dentro da turma cadastrada

Method atulizarTurma(objTurma As Turma, idAluno As %Integer) As Turma
 {
    Set objTurma.Vagas = objTurma.Vagas - 1
    Set count = objTurma.Aluno.Count()
    Set count = count + 1
    do objTurma.Aluno.SetAt(##class(User.Aluno).%OpenId(idAluno),count)
    do objTurma.%Save()
    quit objTurma

 }
```

Figura 36 – Código fonte da atualização de turma

Na figura 37 é apresentado o método responsável por validar o usuário e senha do sistema.

```
//Verifica o Login e senha digitados

ClassMethod RecuperarPorLogin(Nome As %String, Senha As %String) As %String
 {
        Set x = 0
        Set ok = $system.Status.OK()

        Do  // Try
        {
            Set rs = ##class(%Library.ResultSet).%New()
            Set ok = rs.Prepare("select ID from Usuarios where Nome = ? and Senha = ?")
            If 'ok Quit // Exception!
            Set ok = rs.Execute(Nome,Senha)
            If 'ok Quit // Exception!

            //Se entrar no laço é porque Existe o Nome e Senha informados
            If rs.Next()
            {
               Set x = 1
            }
        }
        While 0
        Quit x


 }
```

Figura 37 – Código fonte da verificação do usuário e senha do sistema

Na figura 38 é apresentado o código (VB) que chama o método `RecuperarPorLogin` passando como parâmetros o usuário e senha digitado.

```
Private Sub cmdOK_Click()
    Set Principal.m_object = Principal.m_factory.New("User.Usuarios")
    'Verifica o Usuário e Senha
    If Principal.m_object.RecuperarPorLogin(txtUserName.text, txtPassword.text) Then
        Principal.Show
        LoginSucceeded = True
        Me.Hide
    Else
        MsgBox "Senha inválida, tente novamente!", , "Login"
        txtPassword.SetFocus
        SendKeys "{Home}+{End}"
    End If
Principal.actionClose
End Sub
```

Figura 38 – Código fonte (VB) verificação de usuário e senha

3.4 RESULTADOS E DISCUSSÃO

Foram apresentados alguns trabalhos correlatos. A relação entre eles e o trabalho proposto são:

a) em Obenaus (2000), o trabalho proposto apenas dá enfoque na construção de um sistema simples de compra pela internet. Seu modelo de dados é bastante pequeno e sua aplicação bem limitada. Já no sistema desenvolvido, criou-se um ambiente utilizando o acesso através de objeto e SQL. Todos os requisitos que foram apresentados fazem com que o sistema seja completo e não apenas um módulo.

b) em Souza (2005), o trabalho é bastante fundamentado com relação ao BDC. Apresenta uma pesquisa abrangente, porem sua aplicação é bastante pequena (5 classes), desenvolvida apenas com o intuito de exemplificar as ferramentas do BDC. O sistema foi desenvolvido exclusivamente para o ambiente web. Já no trabalho apresentado foi dado enfoque principal às formas de acesso (objetos e SQL) e as interações com o VB para a criação do sistema *desktop* contendo ainda um modulo para web *(*acesso as notas). O sistema apresenta uma aplicação mais completa, pois seu modelo de dados mostra a interação de 47 classes, possibilitando uma visão mais detalhada do uso do BDC em uma aplicação OO;

c) em Arbegaus (2003), o trabalho apresentado, além de possuir uma pesquisa sobre o BDC, exemplifica através de uma sistema de reserva de vagas em eventos, o uso

do BDC com Java. O modelo de dados é pequeno e o acesso aos dados se da através de JDBC. O sistema foi desenvolvido exclusivamente para o ambiente web utilizando a linguagem JSP. Já no sistema apresentado, utiliza-se do acesso através de ODBC, possibilitando o uso das classes *view*, aliando assim o desempenho do BDC ao *left join* nativo da notação de setas. Outro tipo de acesso utlizado no sistema é o acesso através de objetos. Métodos escritos dentro do BDC são invocados de dentro da aplicação no VB ou mesmo através de outros métodos.

De modo geral as diferenças encontradas entre os três trabalhos correlatos apresentados e o trabalho desenvolvido é que o SGE proposto é mais completo, desenvolvido para ambiente *desktop* utilizando-se de diferentes formas de acesso aos dados no BDC (objetos e SQL). A união dessas formas de acesso sobre uma mesma estrutura de classes exemplifica de maneira transparente como o BDC trabalha com o que tem de melhor dos dois mundos (OO e Relacional).

4 CONCLUSÕES

Nos subitens seguintes, procurou-se apresentar de forma sucinta os principais resultados alcançados, as limitações estabelecidas e algumas sugestões para trabalhos futuros.

4.1 CONSIDERAÇÕES SOBRE O SISTEMA DE GESTÃO ESCOLAR COM CACHÉ

O Caché se apresentou como uma ferramenta que possibilitou que todas as regras do negócio fossem desenvolvidas no mesmo. O VB foi utilizado para a criação da parte GUI do sistema apresentando facilidades no acesso aos dados no BDC.

Um ponto observado no desenvolvimento do trabalho diz respeito a manutenção do sistema. Como a regra do negócio foi toda desenvolvida no próprio BDC, foi facilitado em diversos momentos a alteração e manutenção das classes.

De forma geral, os objetivos propostos foram alcançados através do embasamento teórico sobre o BDC e da implementação do SGE conforme os requisitos levantados.

4.2 VANTAGENS E LIMITAÇÕES

O trabalho desenvolvido possibilita um entendimento melhor sobre as principais características do BDC, fundamentado em inúmeros exemplos práticos dispostos no decorrer do trabalho e na própria construção do sistema.

O SGE por sua vez, demonstra o controle de inúmeras requisições de um ambiente escolar, mostrando na prática, diversas formas de acesso aos dados no BDC, exemplos de herança, associação, agregação, polimorfismo e encapsulamento.

As principais limitações do sistema são:

a) os usuários do sistema poderão efetuar qualquer tipo de alteração;
b) os relatórios apresentados são fixos, não podendo ser alterados;
c) foi utilizado apenas referência simples;
d) não foram implementadas as possíveis configurações do sistema.

4.3 EXTENSÕES

A partir desse trabalho de conclusão de curso pode-se aprofundar os estudos com relação ao BDC e a utilização com outras linguagens de programação como Java e Delphi.

Pode-se construir um sistema robusto inteiramente web para exemplificar o uso do CSP de forma mais abrangente.

Pode-se traçar um comparativo entre os bancos de dados relacionais existentes no mercado com o BDC, executando testes de performance.

Por último pode-se demonstrar a conversão dos dados de um BDR para o BDC. As técnicas e ferramentas utilizadas para a conversão e as perdas se houverem.

REFERÊNCIAS BIBLIOGRÁFICAS

AFONSO, Caricio. **Caché e ODBC:** Intersystems ShortCuts. [S.l.], 2005. Disponível em: <http://www.intersystems.com.br/isc/downloads/wp/WP_ODBC.pdf>. Acesso em: 13 maio. 2007.

ARBEGAUS, Aloisio. **Estudo do SGBD "Caché" com uma aplicação na reserva de vagas em eventos acadêmicos via web.** 2003. 75 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BOSCARIOLI, Clodis et al. **Uma reflexão sobre bancos de dados orientados a objetos.** [S.l.], 2006. Disponível em: <http://conged.deinfo.uepg.br/artigo4.pdf>. Acesso em: 23 maio. 2007.

BORBA, Sueli de Fátima P. **Metodologia para implantação de modelos multidimensionais em banco de dados orientado a objetos.** 2006. 189 f. Tese (Doutorado em Engenharia de Produção) – Programa de Pós-Graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis.

GUALBERTO, João. **Trabalho sobre banco de dados orientado a objeto**. [S.l.], 1998. Disponível em: <http://www.sqlmagazine.com.br/apostilas.asp>. Acesso em: 29 maio. 2007.

INTERSYSTEMS. **Componentes Caché:** Caché & Java**.** [S.l.], 2002a. Disponível em: <http://www.intersystems.com.br/isc/downloads/wp/Cache5_e_Java.pdf>. Acesso em 25 mar. 2007.

_____. **Uma nova era na tecnologia dos bancos de dados.** [S.l.], 2002b. Disponível em: <http://www.intersystems.com.br/isc/downloads/vantagensbeneficios/WhitepaperUmanovaeradeBD.pdf>. Acesso em: 25 fev. 2007.

_____. **Componentes Caché:** linguagens de programação do Caché**.** [S.l.], 2002c. Disponível em: <http://www.intersystems.com.br/isc/downloads/vantagensbeneficios/CACHE%20Basic.PDF>. Acesso em: 12 maio. 2007.

_____. **Objetos e SQL:** a fusão de duas tecnologias atuais em um banco de dados multidimensional. [S.l.], 2002d. Disponível em: <http://www.intersystems.com.br/downloads/Fusion.pdf>. Acesso em: 07 abr. 2007.

_____. **Introdução executiva do Caché.** [S.l.], 2004. Disponível em: <http://www.intersystems.com.br/isc/CacheTecnoOQIntroExec.csp>. Acesso em: 9 mar. 2007.

_____. **Tecnologia Caché.** [S.l.], 2005a. Disponível em: <http://www.intersystems.com.br/exemplos.htm>. Acesso em: 10 mar. 2007.

_____. **Guia da tecnologia Caché.** [S.l.], 2005b. Disponível em: <http://www.intersystems.com.br/isc/downloads/guiatecnologia/Cap1.pdf>. Acesso em: 23 abr. 2007.

MULLER, Jaime. **Base de dados, namespaces e globais**. [S.l.], 2006. Disponível em: <http://www.imasters.com.br/artigo/4062/bancodedados/bases_de_dados_namespaces_e_globais/>. Acesso em: 15 mar. 2007.

OBENAUS, Maurício Rogério. **Protótipo de uma aplicação comercial utilizando banco de dados Caché com interface web.** 2000. 85 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

RICARTE, Ivan L. Marques. **Polimorfismo.** [S.l.], 2000. Disponível em: <http://www.dca.fee.unicamp.br/cursos/PooJava/polimorfismo.html>. Acesso em: 15 abr. 2007.

SAMARY, Amir. **Conhecendo o Caché.** [S.l.], 2004a. Disponível em: <http://br.groups.yahoo.com/group/cache-br/files/Tutoriais/>. Acesso em: 12 nov. 2006.

_____, Amir. **Introdução a orientação a objetos no Caché.** 2004b. Disponível em: <http://www.dc.ufscar.br/~marilde/Graduacao/Segundo%20semestre/2006/Cache/Orientacao_OO_Cache.pdf>. Acesso em: 13 abr. 2007.

SOUZA, Flávio R. de Carvalho. **Desenvolvimento de aplicações utilizando bancos de dados pós-relacionais**. 2005. 51 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências da Natureza, Universidade Federal do Piauí, Teresina.

TIOBE SOFTWARE. **November headline:** Ruby is rocketing skywards. [S.l.], 2006. Disponível em: <www.tiobe.com/tpci.htm>. Acesso em: 14 fev. 2007.