

Universidade do Minho  
Conselho de Cursos de Engenharia  
Licenciatura em Engenharia Informática



Universidade do Minho

## **Disciplina de Laboratórios de Informática IV**

Ano Lectivo de 2007/2008

# **Gestão de Back-Office de Tabelas Auxiliares de uma Base de dados de Candidaturas**

Rodrigo Carvalho (38055)  
Arlindo Pires (38586)

**Supervisão:** Pedro Rangel Henriques  
Daniela Cruz

4 de Julho de 2008

Data de Recepção	
Responsável	
Avaliação	
Observações	

## **Resumo**

Este projecto surge com a necessidade do *Sistema de Informação para Gestão do ON.2* (SIGON) ter de elaborar a administração de parte das tabelas contidas na *Base de Dados* (BD), para isso precisamos de sistematizar todo este processo com vista a um menor esforço da manutenção e actualização da aplicação *Web*.

**Área de Aplicação:** Desenvolvimento Web ...

**Palavras-Chave:** Back-Office, Front-Office, Reflection, Reverse Engineering, Bases de Dados, Sistemas de Informação ...

# Conteúdo

Conteúdo	i
Lista de Figuras	ii
<b>1 Introdução</b>	<b>1</b>
<b>2 Enunciado do Problema</b>	<b>3</b>
<b>3 Concepção da Solução / Arquitectura do Sistema</b>	<b>4</b>
<b>4 Implementação</b>	<b>7</b>
4.1 Tecnologias . . . . .	7
4.2 Linguagens . . . . .	8
4.3 Conceitos gerais . . . . .	8
4.3.1 <i>Reverse Engineering</i> . . . . .	8
4.3.2 <i>Back-Office vs Front-Office</i> . . . . .	8
4.3.3 <i>Reflection</i> . . . . .	8
4.3.4 <i>ASP.NET</i> . . . . .	9
4.4 Descrição da Implementação . . . . .	9
<b>5 Exemplos ilustrativos do funcionamento do SI</b>	<b>13</b>
<b>6 Conclusão</b>	<b>20</b>
<b>Bibliografia</b>	<b>21</b>
<b>A Diagramas</b>	<b>22</b>
A.0.1 Diagrama de classes respeitantes ao DAO do Front-Office SIGON . . . . .	22
<b>Glossário</b>	<b>27</b>

# Lista de Figuras

3.1	Modelo Conceptual . . . . .	5
3.2	Diagrama de classes do Back-Office . . . . .	6
5.1	Página inicial . . . . .	13
5.2	Visualizar tabela . . . . .	14
5.3	Seleccionar Registo . . . . .	15
5.4	Inserir registo . . . . .	16
5.5	Editar registo . . . . .	17
5.6	Notificação de erro . . . . .	18
5.7	Descrição de erro . . . . .	19
A.1	Diagrama de Classes 1 . . . . .	23
A.2	Diagrama de Classes 2 . . . . .	24
A.3	Diagrama de Classes 3 . . . . .	25
A.4	Diagrama de Classes 4 . . . . .	26

# Capítulo 1

## Introdução

O nosso projecto consiste na implementação do **Back-Office** do SIGON. O *Programa Operacional do Norte* (ON.2), tem como objectivo apoiar o desenvolvimento da região norte de Portugal. O sistema foi idealizado e implementado para facilitar todo o processo de candidaturas a *Fundos Europeus de Desenvolvimento Regional* (FEDER). Este processo tem como estados, o registo da entidade no sistema, a submissão da candidatura, aprovação ou reprovação do projecto proposto, consulta do estado de avaliação sobre um pedido ou projecto a que tenha permissões.

A motivação presente no nosso trabalho, prende-se com o facto de que em aplicações de grande escala é necessária uma ferramenta que automatize a gestão e/ou administração dos dados relacionados com o *Sistema de Informação* (SI). Deste modo facilita-se o procedimento de alteração da BD do sistema.

Tendo em vista a simplificação de utilização e a eficácia da aplicação no contexto referido anteriormente, concebemos a interacção com o utilizador do seguinte modo( podemos visualizar mais detalhadamente no Capítulo 7):

1. A página inicial apresenta como opção seleccionar a tabela da BD desejada, com o intuito de visualizar e/ou alterar a mesma.
2. Depois da opção tomada anteriormente, podemos visualizar os seus registos de forma tabular( explicitando o nome dos atributos).
3. De seguida é possível seleccionar, remover ou inserir uma linha na tabela.
4. No caso de selecção é mostrado mais detalhadamente o tuplo em questão, permitindo a sua adulteração, contudo se a opção for de remoção, processa-se a actualização da informação e regressa ao ponto 2.

Toda esta implementação foi realizada sobre a plataforma *ASP.Net*, pois permite o desenvolvimento de aplicações *Web* dinâmicas a partir de um conjunto de classes e/ou controlos (está presente o conceito de *object oriented*), dando continuidade ao trabalho já realizado.

Após esta breve introdução procedemos á descrição do problema, onde iremos expôr todos os objectivos a atingir a nós propostos (Capítulo 2). No capítulo seguinte relatamos a forma como analisamos o problema, conseguindo assim uma abstracção de como estruturar a arquitectura do sistema. Também explicamos quais as decisões tomadas acerca das estruturas de dados e algoritmos para a implementação(desenvolvido mais detalhadamente no Capítulo 4).

Na conclusão fazemos referência aos aspectos mais importantes caracterizados neste relatório, e

ao estado actual da solução. Examinamos também de forma crítica todos os métodos aplicados na concepção do trabalho realizado. Por fim, salientamos quais os melhoramentos a efectuar de forma a potenciar a nossa aplicação *Web*.

## Capítulo 2

# Enunciado do Problema

Neste projecto pretende-se implementar uma aplicação *Web* que tenha como função gerir o **Back-Office** de tabelas auxiliares de uma BD de candidaturas, sendo estas partes integrantes do SIGON. Para tal os objectivos a cumprir são os seguintes:

- Compreender a arquitectura do SIGON, focando principalmente a sua camada de dados.
- Identificar as classes responsáveis pelas operações de gerenciamento de dados, tipos de classes, estruturação (métodos e variáveis de instância), e também a forma como se ligam entre elas (composição, agregação, especialização, generalização...).
- Especificar o diagrama de classes referente ao ponto anterior.
- Esquematizar o modelo conceptual da BD, e perceber as dependências entre tabelas.
- Construir o diagrama de classes do projecto em questão.
- Apresentar uma solução que permita a visualização, inserção, remoção e a actualização das tabelas contempladas.

## Capítulo 3

# Concepção da Solução / Arquitectura do Sistema

Primeiramente analisamos o diagrama de classes do SIGON com vista a compreender a estrutura do sistema informático.

De seguida capturamos todos os requisitos que o nosso sistema deverá cumprir, com vista a descobrir quais os objectos que asseguram essas funcionalidades, de que forma se relacionam, e quais os seus atributos e métodos associados.

Para a clarificação do problema resolvemos, arquitectar o modelo conceptual da BD do SIGON (somente as tabelas interligadas ao nosso **Back-Office**), com isto foi-nos possível mapear (com toda a informação já reunida) o modelo relacional para o diagrama de classes do sistema em desenvolvimento.

Decidimos então definir uma classe abstracta "*Tabela*", com o intuito de que todas as operações sobre as relações da BD são as mesmas (manipulação dos registos), distinguindo-se apenas o algoritmo de cada uma delas, pois os seus atributos são diferentes, e há casos de dependências na BD, deste modo cada tabela da BD tem uma classe com o mesmo nome (subclasse de "*Tabela*") e que trata todos os métodos sobre os registos.

Por ultimo idealizamos a interface com o utilizador separando o *Web Design* do tratamento de eventos. No *Web Design* preocupamo-nos fundamentalmente com o utilizador final, garantindo que atinja os seus objectivos de maneira agradável e intuitiva. O tratamento de eventos é feito totalmente apartir de uma classe cujos métodos se responsabilizam por esta tarefa.

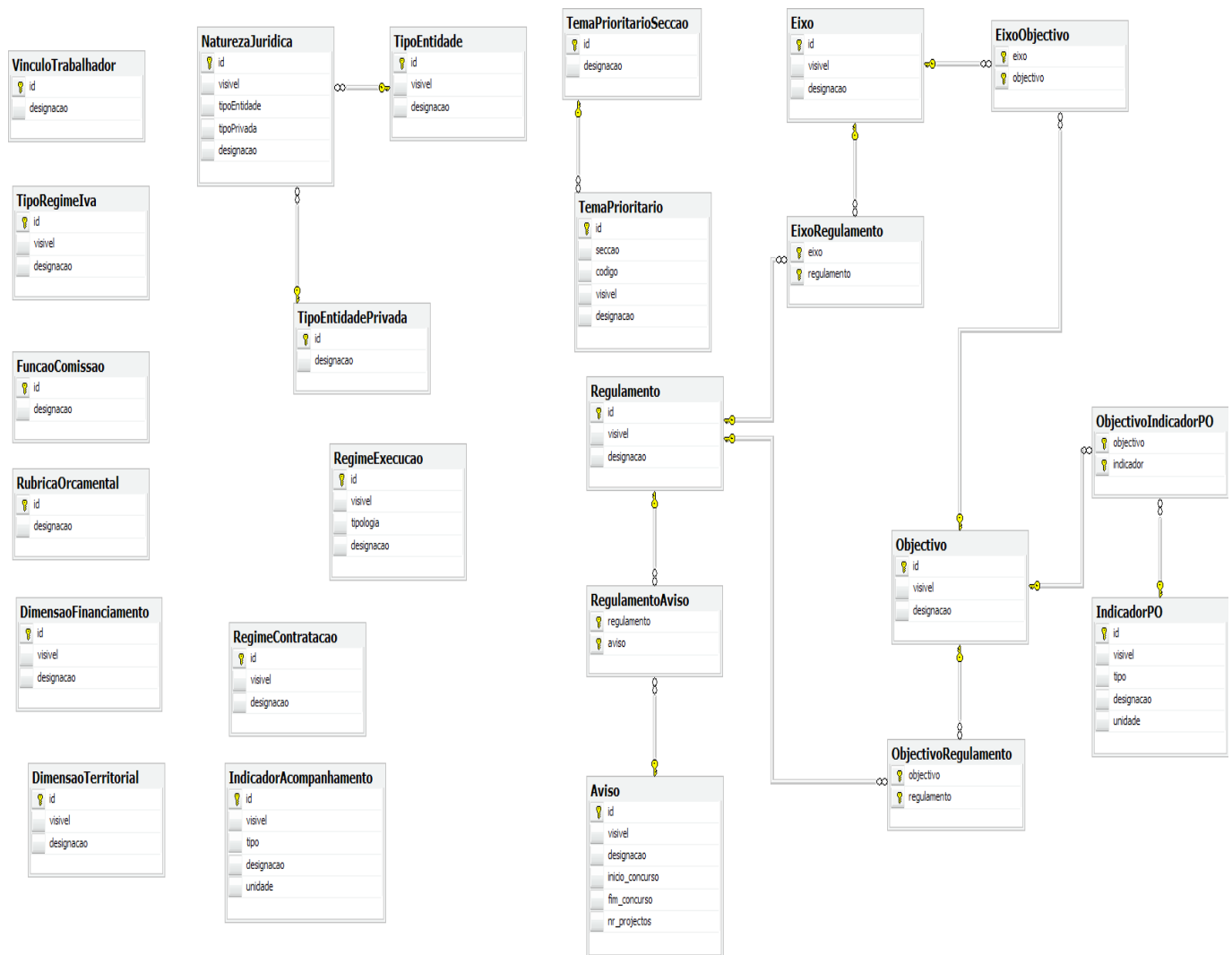


Figura 3.1: Modelo Conceptual

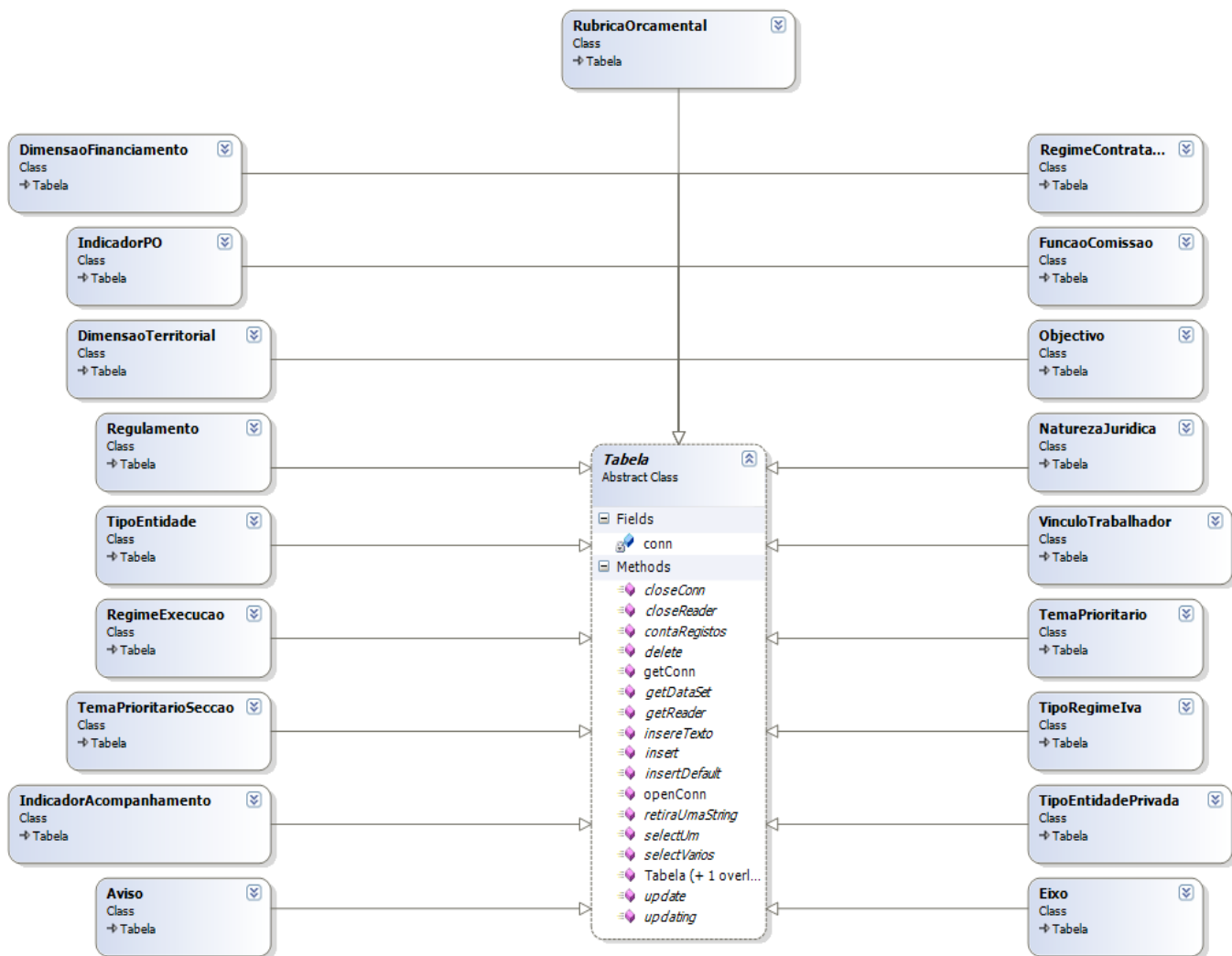


Figura 3.2: Diagrama de classes do Back-Office

# Capítulo 4

## Implementação

Neste Capítulo explicitamos quais as tecnologias por nós utilizadas, enunciamos as linguagens de programação a que recorreremos, procedemos à contextualização dos conceitos mais relevantes, para que o leitor compreenda a forma como construímos a solução da nossa aplicação *Web*, por último descrevemos a sua implementação.

### 4.1 Tecnologias

- *Integrated Development Environment (IDE) Microsoft Visual Studio:*
  - O Microsoft Visual Studio é um pacote de programas da Microsoft, para desenvolvimento de Software, especialmente dedicado, ao framework .NET e às linguagens Visual Basic (VB), C , C++ (C Plus Plus), C# (C Sharp) e J# (Jey Sharp). Também é um grande produto de desenvolvimento na área web, usando a plataforma do ASP.NET. As linguagens com maior frequência nessa plataforma são: VB.NET (Visual Basic .Net) e o C#.
- *IDE Microsoft SQL Server Management Studio:*
  - SQL Server Management Studio é uma ferramenta incluída com o SQL Server( SGBD-Sistema Gestor de Base de Dados) para configurar, gerir e administrar todos os componentes no Microsoft SQL Server. A ferramenta inclui editores e ferramentas gráficas que trabalham com objectos e funcionalidades do servidor.
- *IDE Visual Paradigm*
  - *Visual Paradigm* é uma ferramenta integrada para UML avançada que suporta o ciclo de desenvolvimento de software - análise, projeto, implementação, teste e depuração.
- *Plugin Visual Paradigm*
  - Integrado com o *Visual Studio* permite o *Reverse Engineering* de código-fonte para UML.

## 4.2 Linguagens

- UML - A *Unified Modeling Language* (UML) é uma linguagem de modelação unificada. Esta linguagem não é uma metodologia de desenvolvimento, o que significa que ela não nos diz o que fazer ou como projectar o sistema, mas sim, serve de auxílio a visualizar o desenho e a comunicação entre objectos.
- C# (*c sharp*) - Linguagem de programação orientada aos objectos.
- SQL - *Structured Query Language*. Implementa os conceitos definidos no modelo relacional.

## 4.3 Conceitos gerais

### 4.3.1 *Reverse Engineering*

Vem de encontro ao que o próprio nome indica, é o processo inverso ao da produção de software, ou seja, através do código-fonte ou do "executável" de uma aplicação gera-se informação( recorrendo a modelos ou outras representações) sobre a arquitectura do sistema. Permitindo, deste modo, uma visão com um nível de abstracção superior sobre o programa em questão.

O *Reverse Engineering* revela-se óptimo em situações como:

- Perda do código-fonte( tendo em mão o "executável").
- Perda da documentação com a modelagem do software.
- Manutenção e actualização das aplicações, aquando da falta de documentação.

### 4.3.2 *Back-Office vs Front-Office*

#### ***Back-office:***

Sistema de informação que tem como função gerir e/ou administrar os dados presentes num dado contexto.

As aplicações de *Back-Office* são normalmente para a utilização de gerente(s) e/ou administrador(es) de sistemas.

#### ***Front-office:***

Sistema de informação que tem como função interagir com os dados presentes num dado contexto.

As aplicações de *Front-office* são construídas para os utilizadores dos sistemas.

### 4.3.3 *Reflection*

Consiste em descobrir em tempo de execução quais os tipos presentes no sistema e suas associações, assim como permitir a manipulação desses tipos. Os mecanismos de *Reflection* são bastante poderosos e permitem muita flexibilidade, por exemplo, é possível chamar métodos "por nome" em objectos dos quais apenas é conhecida uma referência, ou descobrir todas as relações entre classes e objectos.

#### 4.3.4 ASP.NET

O *ASP.NET* foi lançado pela primeira vez em 2002 pela *Microsoft*, tendo como antecessor o *ASP*.

A plataforma *ASP.NET* permite o desenvolvimento de aplicações *Web* dinâmicas a partir de um conjunto de classes e/ou controlos. Portanto está presente o conceito de OO( *Object Oriented*) na estruturação deste tipo de aplicações, e a simplificação do manuseamento dos controlos HTML.

#### ADO.NET

A plataforma .NET oferece um conjunto de classes que permitem o acesso á base de dados a que chamou de ADO.NET.

O *ADO.NET* traz consigo o conceito de *data provider*, cada provider é responsável por estabelecer a ligação a uma base de dados através de um determinado protocolo, sendo de destacar que permite efectuar ligação à base de dados SQL Server.

- *SqlConnection*- Permite fazer a conexão à base de dados, recebendo como parâmetro, uma *string* contendo a *path* para a localização da BD.
- *SqlCommand*- Funciona como um interpretador de comandos SQL, recebendo como parâmetros, uma *string* incluindo o comando SQL a executar e uma instância do *SqlConnection*.
- *SqlDataReader*- Armazena em memória os dados provenientes da execução de um comando SQL sobre a base de dados.

#### Controlos *Data Source*

Conjunto de controlos capazes de obter dados provenientes de uma base de dados.

- *DataSet*- Armazena em memória os dados provenientes da base de dados depois de efectuado um comando SQL.

#### Controlos *Data Bound*

Tem controlos capazes de apresentar os dados fornecidos pelos controlos *Data Source*, entre os quais destacam-se:

- *GridView*- Controlo capaz de apresentar os dados provenientes de uma base de dados, em formato tabular. Além disso, pode-se activar botões para seleccionar, remover e alterar nas linhas presentes da tabela, permitindo também a ordenação e paginação dos dados.
- *DetailsView*- Controlo idêntico ao *GridView*, com a restrição de, apenas mostrar em forma de caixa, uma linha de uma tabela da base de dados.

## 4.4 Descrição da Implementação

No âmbito do projecto escolhido e das ferramentas indicadas pelos orientadores, começamos por explorar as potencialidades do **Visual Studio**, visto que foi o primeiro contacto com este IDE

ao longo do nosso percurso académico, para isso elaboramos uma série de exemplos puramente académicos com vista a ambientarmo-nos á ferramenta.

Seguidamente, pesquisamos mais aprofundadamente o conceito, a abstracção do **Reverse Engineering**, porém encontramos algumas ferramentas que automatizam o processo, destacando-se entre eles o plugin disponibilizado pelo **Visual Paradigm** para integrar no **Visual Studio**, e também a aplicação **Free-Source** denominada **AutoDiagrammer**, cujo autor *Sacha Barber* disponibilizou na *Web*. Neste contexto, em vez de construirmos um diagrama de classes no **Visual Studio**, optamos por utilizar a primeira ferramenta referida de modo a obter o diagrama de classes automaticamente, esta solução custou-nos tempo de pesquisa mas foi altamente rentável pois o objectivo foi concretizado maquinalmente. Depois de escolhidas as tabelas da BD em que iríamos operar, concebemos o modelo conceptual respectivo com a ajuda do **SQL Server Management Studio**.

Numa fase posterior, desenhamos o diagrama de classes em UML recorrendo ao IDE **Visual Paradigm**, estruturando assim a nossa camada de dados.

Relativamente à página *Web*, os *Web Controls* utilizados foram *labels*, para indicar ao utilizador as opções ou notificar erros( previamente anunciados por um **alert de JavaScript chamado através do C#**), *Buttons*, para confirmar a escolha de tabela e para inserir novos registos á BD, *DropDownList*, usada para seleccionar de entre uma lista de itens o nome da tabela pretendida. Esta última, apresenta a lista de itens de uma forma dinâmica, já que os seus valores são encontrados após uma query á BD. Ainda na construção da página, utilizamos controlos de *DataBound* sendo eles, o *GridView* e *DetailsView*.

Referente á classe associada ao aspx, que trata todos os eventos gerados (efectuando o tratamento de excepções) pelos *Web Controls* e *DataBound Controls* tendo métodos que foram realizados programaticamente com *c#*( *pagging*, *rowdeleting*, *item updating*, *selectedindex*,...).

Código para seleccionar o indice:

```
public void SelectedIndexChanged(GridView Grid, DetailsView Details)
{
    int id;
    string nomeTabela="";
    nomeTabela = DropDownList1.Text.Trim();
    Tabela tabela = this.constructor(nomeTabela);
    if (Grid.SelectedDataKey != null)
        id = (int)Grid.SelectedDataKey.Value;
    else id = tabela.contaRegistos();
    tabela.selectUm(id);
    string [] keys;
    keys = new string [] { "id" };
    Details.DataKeyNames = keys;
    Details.DataSource = tabela.getReader();
    Details.DataBind();
    tabela.closeConn();
}

protected void GridView1_SelectedIndexChanged(object sender, EventArgs e)
{
    try
    {
        this.SelectedIndexChanged(GridView1, AvisoDetails);
        AvisoDetails.Visible = true;
    }
}
```

```

catch (Exception exc)
{
    Label4.Text = exc.Message;
    Label4.Visible = true;
    this.MsgBox("Erro!! Ver descricao em baixo");
    GridView1.Visible = false;
}

```

Uma inovação que podemos referir foi o facto de termos criado um método que recebe uma *string* com o nome de uma classe, e com base nisto cria uma instância( objecto) em *runtime* do seu objecto correspondente, retornando-a como resultado. Isto foi possível devido á propriedade do *c# Reflection*.

Este método teve aplicação prática aquando do tratamento do evento da selecção da tabela pretendida, pois é obtido o nome da mesma pelo meio da *dropdownlist* em forma de texto, para que o objecto da camada de dados seja carregado em memória, sendo este responsável pelas operações de manuseamento de dados da respectiva tabela da BD.

Código reflection:

```

private Assembly assembly = typeof(Tabela).Assembly;
private string assemblyName = assembly.FullName;;

public Tabela constructor(string nomeTabela)
{
    Tabela tabela = null;
    Object[] paramters = new Object[0];
    Assembly ass = Assembly.ReflectionOnlyLoad(assemblyName);
    AssemblyName a = ass.GetName();
    Assembly asse = Assembly.Load(a);
    Type tipo = asse.GetType(nomeTabela);
    ConstructorInfo[] fi = tipo.GetConstructors();
    tabela = (Tabela)fi[0].Invoke(paramters);
    return tabela;
}

```

Acerca da camada de dados temos a salientar que definimos uma classe abstracta "Tabela" que possui como atributo uma instância de *SqlConnection*, pois todas as suas subclasses necessitam de uma ligação á BD, nesta classe são definidos todos os métodos abstractos que têm de ser cumpridos nas subclasses( como exemplos os métodos de inserção e remoção de registos). Assumindo estes exemplos, queremos demonstrar que em cada uma das subclasses os procedimentos são diferentes, pois as suas tabelas correspondentes têm atributos distintos, e por vezes existem dependências entre tabelas e é necessário efectuar a alteração em tantas, quanto o número de relações.

Código para eliminar um tuplo de uma tabela com relações a outras:

```

public override void delete(object id)
{
    int entidade = this.getEntidade(id);
    int privada = this.getPrivada(id);
    string sql = "delete from NaturezaJuridica where id=@id";
    SqlCommand cmd = new SqlCommand(sql, getConn());
    cmd.Parameters.AddWithValue("@id", (int)id);
    getConn().Open();
    cmd.ExecuteNonQuery();
    closeConn();
    TipoEntidade tipoE = new TipoEntidade();
    tipoE.delete(entidade);
    tipoE.closeConn();
    TipoEntidadePrivada tipo = new TipoEntidadePrivada();
}

```

```
tipo.delete(privada);  
tipo.closeConn();  
}
```

## Capítulo 5

# Exemplos ilustrativos do funcionamento do SI



Figura 5.1: Página inicial

back office

# SIGON.2

SISTEMA DE INFORMAÇÃO DE GESTÃO DO ON.2

Escolha uma das tabelas:

		id	visivel	designacao	inicio_concurso	fim_concurso	nr_projectos
Selecionar	Eliminar	1	<input checked="" type="checkbox"/>	GAEPC/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	8
Selecionar	Eliminar	2	<input checked="" type="checkbox"/>	AVQA/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	7
Selecionar	Eliminar	3	<input checked="" type="checkbox"/>	AVL/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	6
Selecionar	Eliminar	4	<input checked="" type="checkbox"/>	S/1/2007	21-12-2007 0:00:00	11-04-2008 0:00:00	0
Selecionar	Eliminar	5	<input checked="" type="checkbox"/>	PRU/1/2007	21-12-2007 0:00:00	11-04-2008 0:00:00	3

1 2 3 4 5 6 7

Figura 5.2: Visualizar tabela

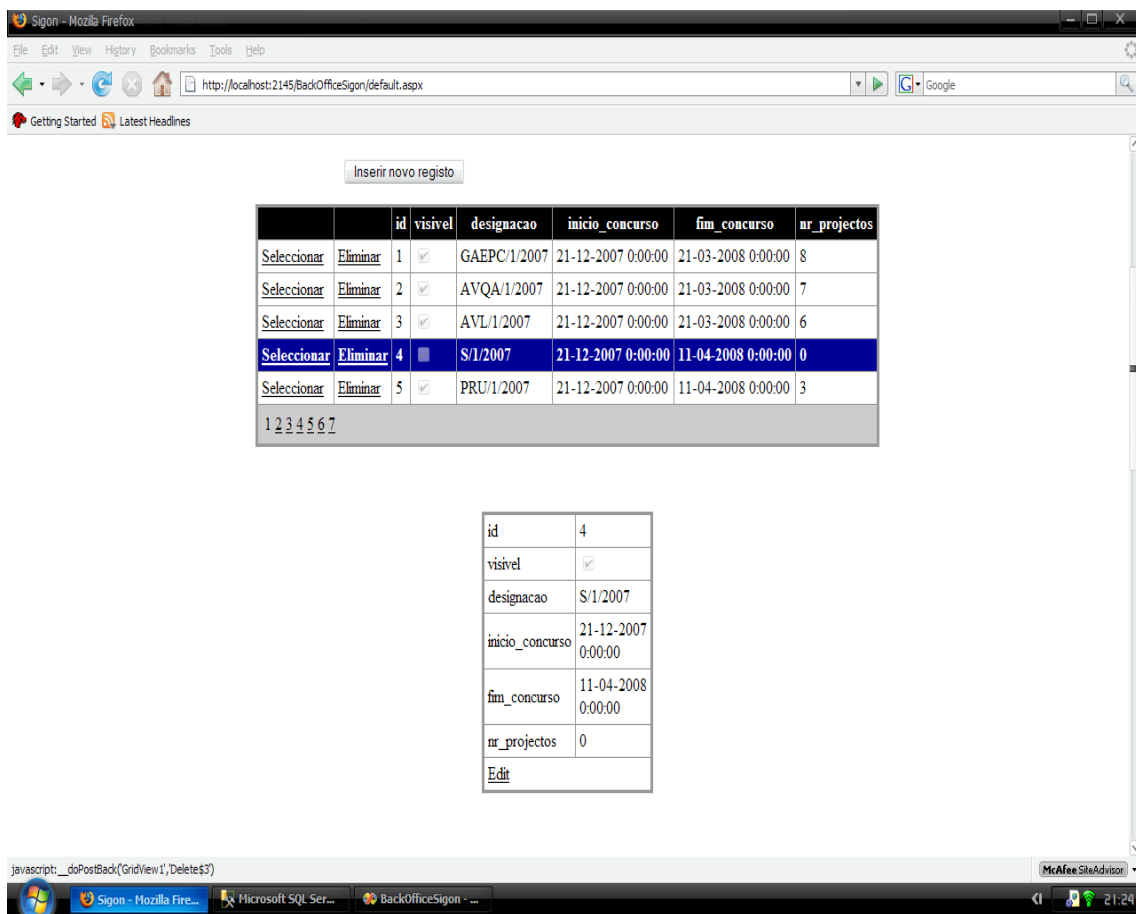


Figura 5.3: Seleccionar Registo

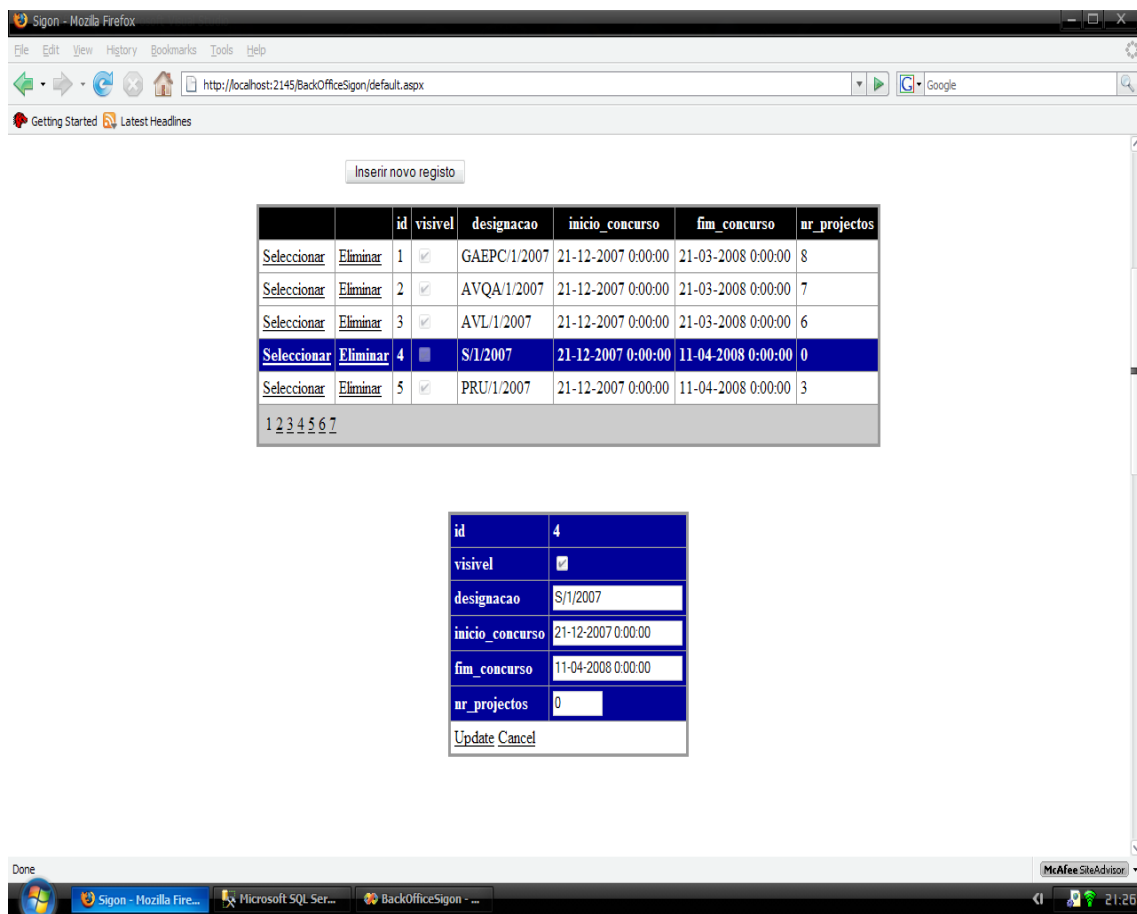
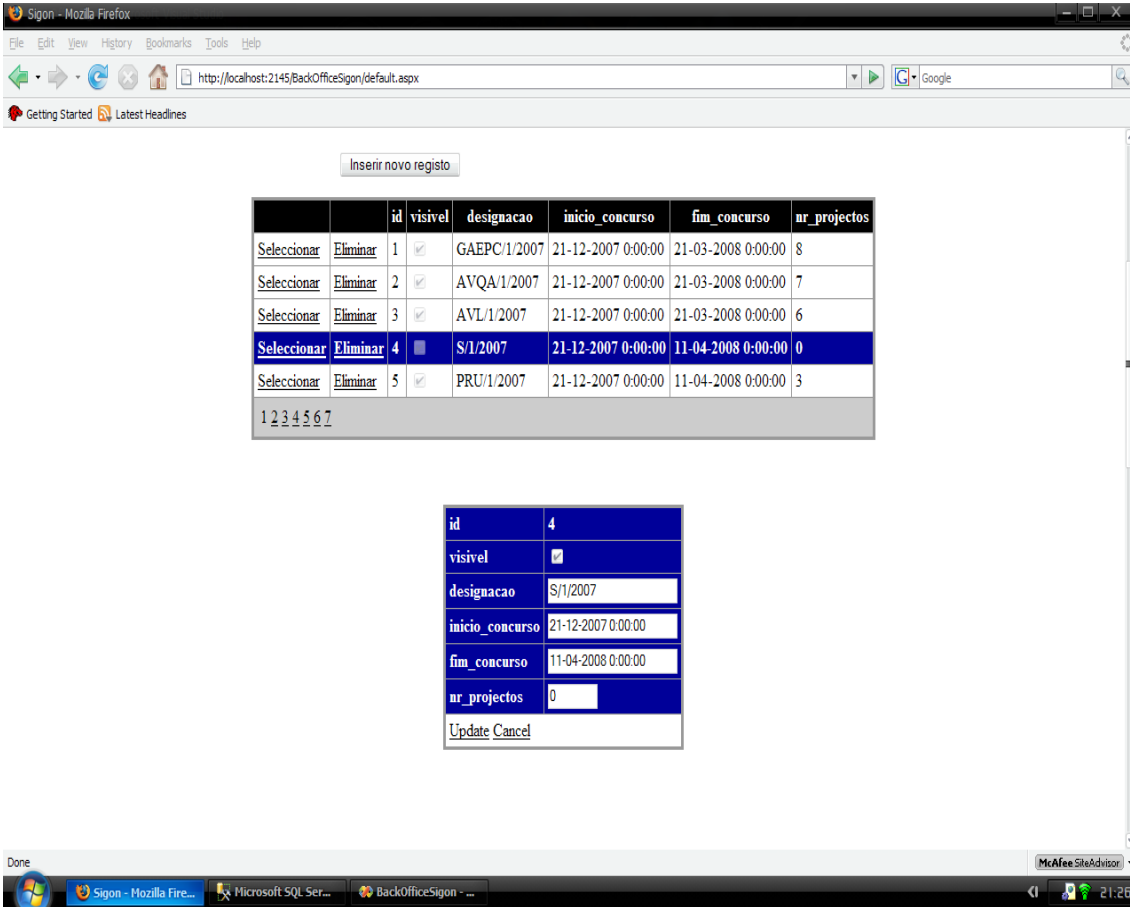


Figura 5.4: Inserir registo



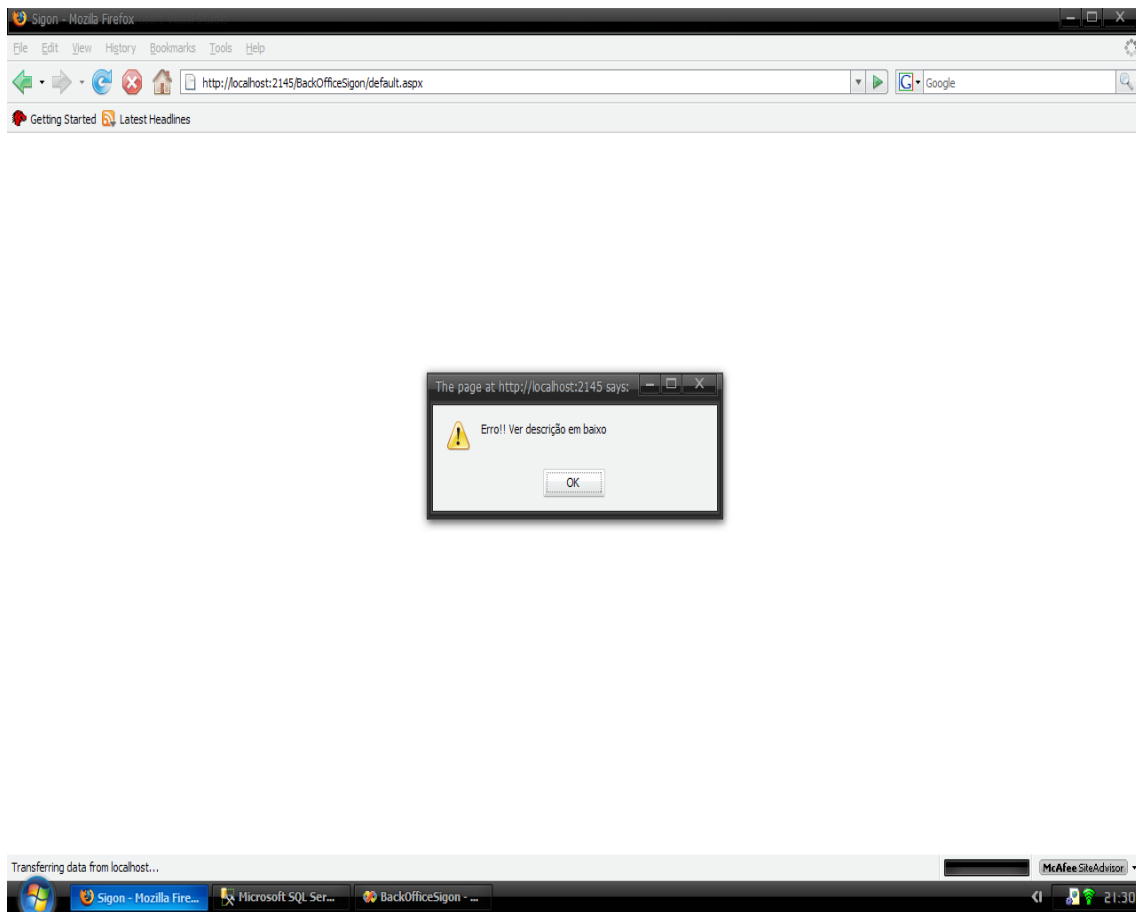
Inserir novo registo

		id	visivel	designacao	inicio_concurso	fim_concurso	nr_projectos
<a href="#">Selecionar</a>	<a href="#">Eliminar</a>	1	<input checked="" type="checkbox"/>	GAEPC/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	8
<a href="#">Selecionar</a>	<a href="#">Eliminar</a>	2	<input checked="" type="checkbox"/>	AVQA/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	7
<a href="#">Selecionar</a>	<a href="#">Eliminar</a>	3	<input checked="" type="checkbox"/>	AVL/1/2007	21-12-2007 0:00:00	21-03-2008 0:00:00	6
<a href="#">Selecionar</a>	<a href="#">Eliminar</a>	4	<input checked="" type="checkbox"/>	S/1/2007	21-12-2007 0:00:00	11-04-2008 0:00:00	0
<a href="#">Selecionar</a>	<a href="#">Eliminar</a>	5	<input checked="" type="checkbox"/>	PRU/1/2007	21-12-2007 0:00:00	11-04-2008 0:00:00	3

1 2 3 4 5 6 7

id	4
visivel	<input checked="" type="checkbox"/>
designacao	S/1/2007
inicio_concurso	21-12-2007 0:00:00
fim_concurso	11-04-2008 0:00:00
nr_projectos	0
<a href="#">Update</a> <a href="#">Cancel</a>	

Figura 5.5: Editar registo



**Figura 5.6:** Notificação de erro



Figura 5.7: Descrição de erro

## Capítulo 6

# Conclusão

Num sistema de informação de média ou grande escala, é fundamental possuir uma ferramenta de *Back-Office* para que se automatize todo o exercício de manutenção de um sistema. Perante a óptica do programador, a curto prazo, parece ser uma solução muito penosa, todavia, a médio e longo prazo é compensado todo o esforço levado a cabo, pois deixa de haver necessidade do programador interagir directamente e manualmente com o SGBD.

Analizando todo o trabalho realizado, podemos afirmar que conseguimos atingir todos os objectivos propostos. Embora tenha existido uma fase de progressão menos produtiva, devido ao facto, de ser o nosso primeiro contacto com o desenvolvimento de aplicações Web utilizando a plataforma *ASP.NET*, perdemos assim algum tempo em pesquisa, contudo ganhamos produtividade em fases posteriores.

Outra observação pertinente sobre o projecto, é que, em caso de haver uma "simbiose" deste *Back-Office* com o *Front-Office* já existente na realidade, o nosso sistema passaria a ter uma validação de *login's* para o(s) gestor(es) e/ou administrador(es) efectuarem a manutenção dos dados do SIGON. Neste contexto seria mais perspicaz e eficiente realizar todas as alterações aos registos apenas no fim da sessão, diminuindo assim os acessos à BD.

Como trabalho futuro poderíamos melhorar o aspecto gráfico (*WebDesign*), e estender a aplicação às restantes tabelas.

# Bibliografia

- [1] Luís Abreu, *ASP.NET 2.0 - Curso Completo*, 3ª edição, 2006.
- [2] Cristian Darie & Zak Ruvalcaba, *Build Your Own ASP.NET 2.0 Web Site Using C# & VB*, 2nd edition, 2006.
- [3] Matthew MacDonald and Mario Szpuszta, *Pro ASP.NET 2.0 in C# 2005*, 2005.
- [4] Damien Foggon, *Beginning ASP.NET 2.0 Databases: From Novice to Professional*, 2006.
- [5] Paulo Marques e Hernâni Pedroso, *C# 2.0*, 4ª edição, 2005.
- [6] Luís Damas, *SQL - Structured Query Language*, 8ª edição, 2005.
- [7] Outras Fontes:

<http://pt.wikipedia.org/wiki>

<http://www.codeproject.com/KB/cs/AutoDiagrammer.aspx>

[http://www.codeguru.com/csharp/csharp/cs\\_misc/reflection/article.php/c4257/](http://www.codeguru.com/csharp/csharp/cs_misc/reflection/article.php/c4257/)

[http://www.codeproject.com/KB/cs/C\\_\\_Reflection\\_Tutorial.aspx](http://www.codeproject.com/KB/cs/C__Reflection_Tutorial.aspx)

<http://www.linhadecodigo.com.br/Artigo.aspx?id=1518>

<http://www.csharp-station.com/>

## Apêndice A

# Diagramas

### A.0.1 Diagrama de classes respeitantes ao DAO do Front-Office SIGON

Dividimos o diagrama em 4 sub-diagramas, devido ao número extenso de classes.

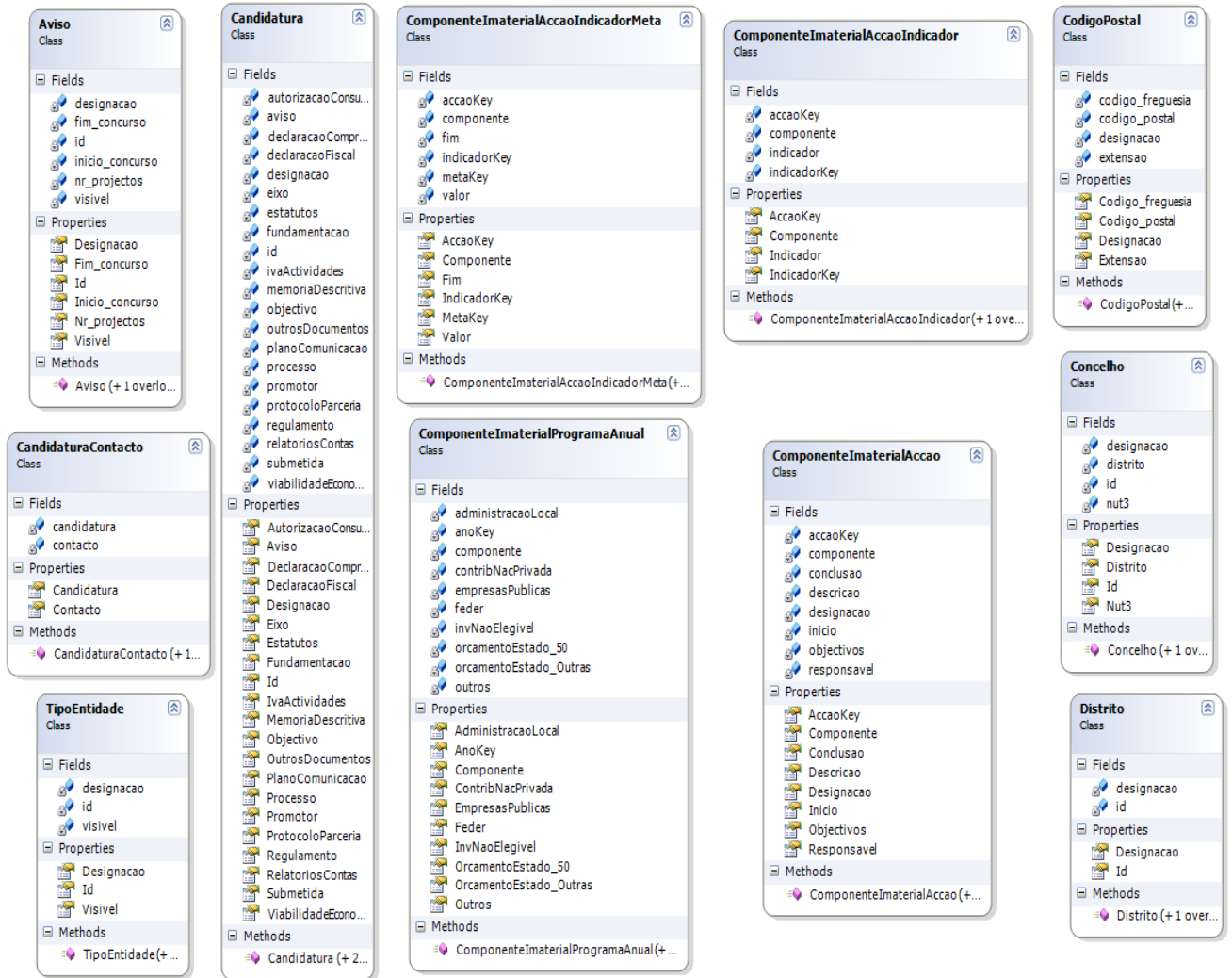


Figura A.1: Diagrama de Classes 1

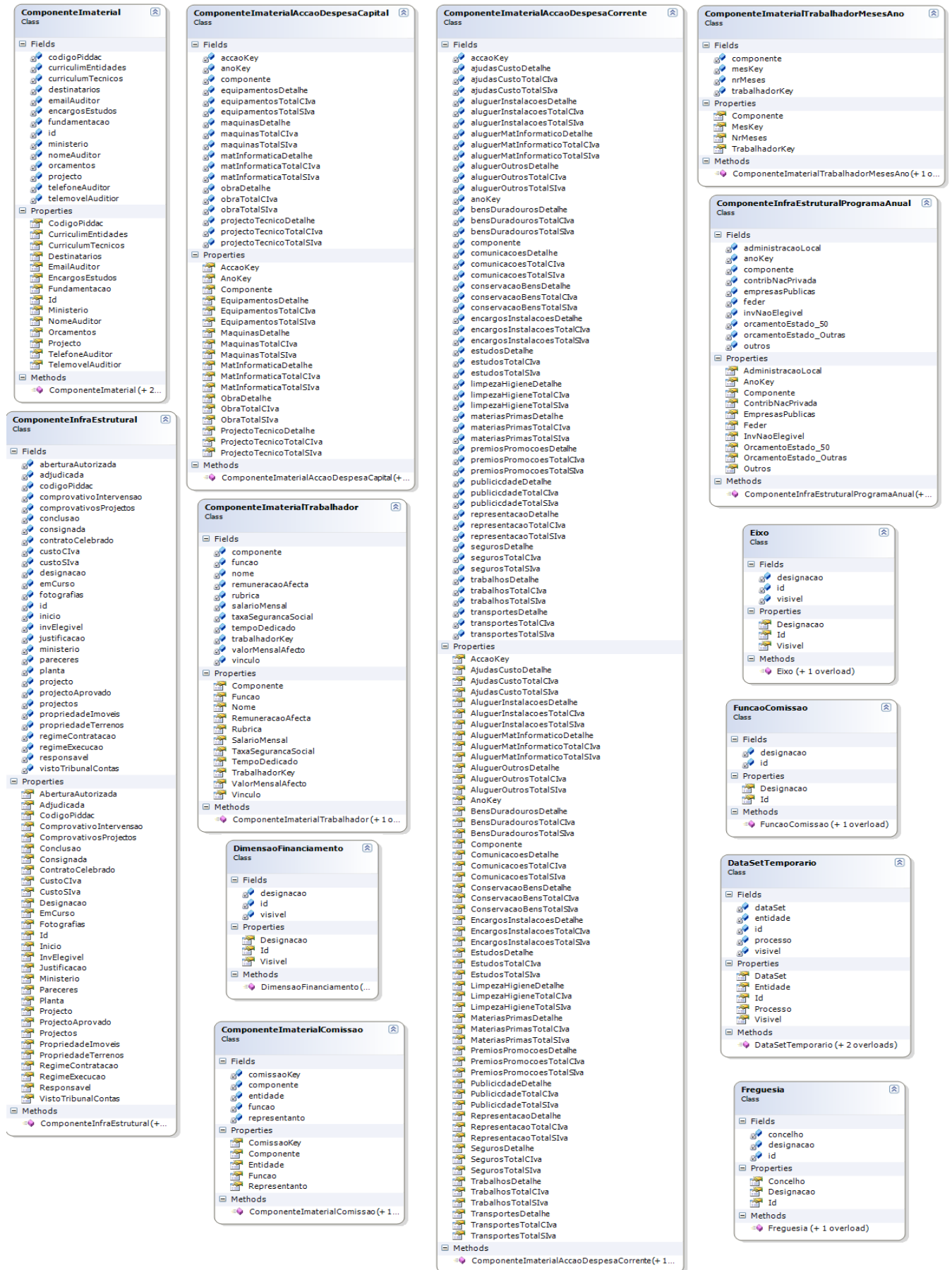


Figura A.2: Diagrama de Classes 2

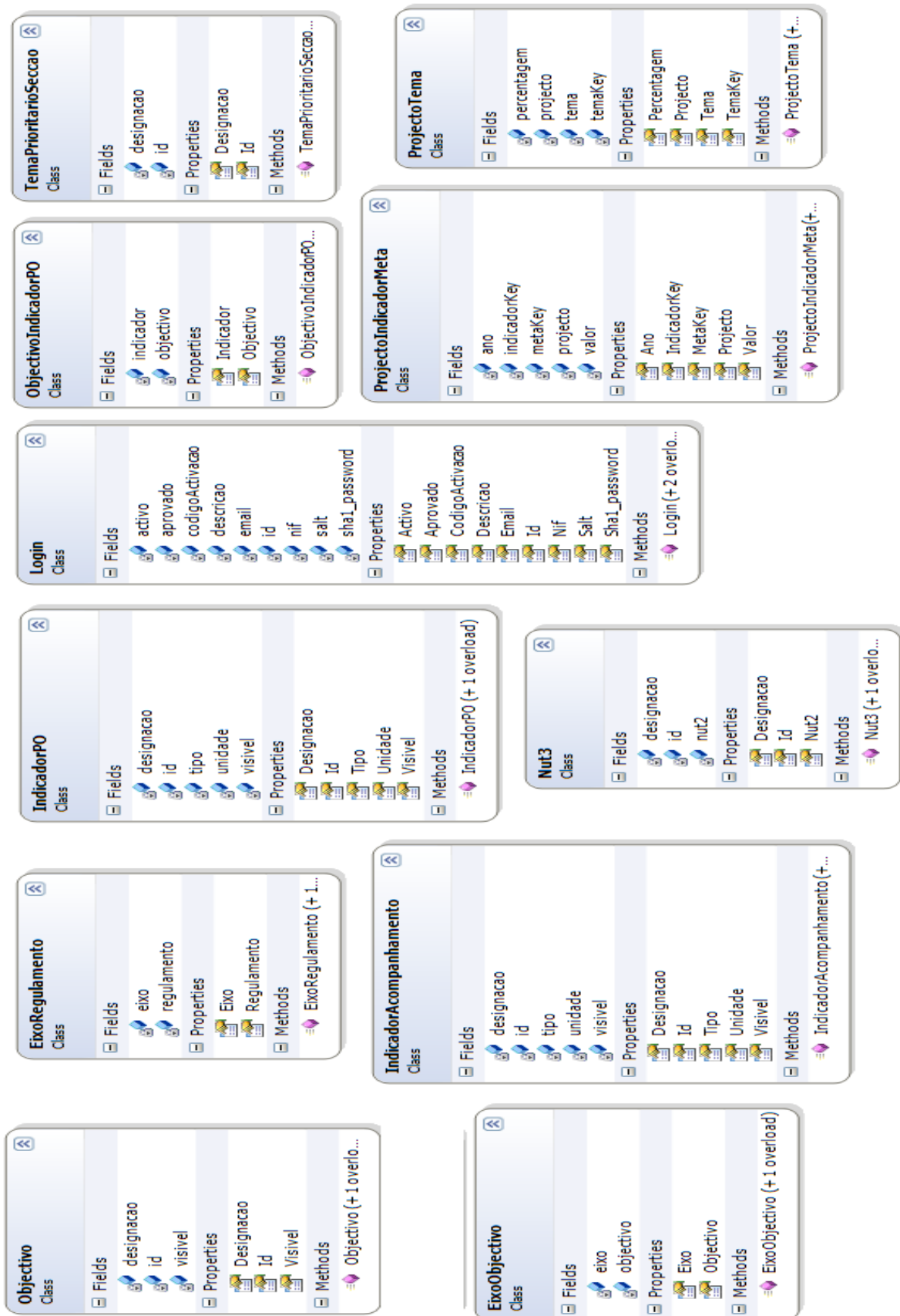


Figura A.3: Diagrama de Classes 3

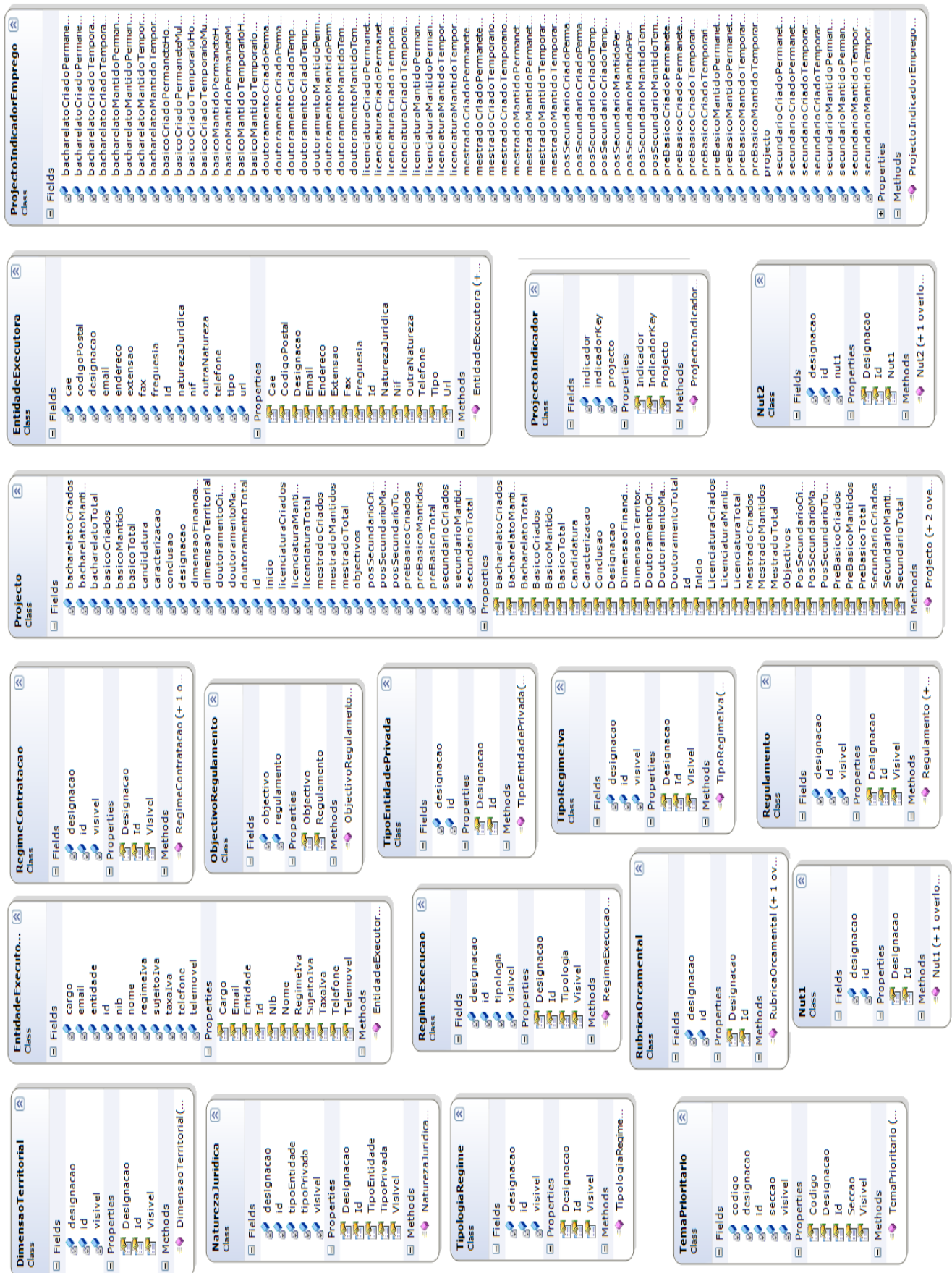


Figura A.4: Diagrama de Classes 4

# Glossário

**DI** Departamento de Informática

**HTML** *Hypertext Markup Language* - um conjunto de etiquetas e regras de anotação de texto (de acordo com SGML) usado na criação de documentos hipertexto para serem divulgados na Internet e implementados pelos browsers WWW. É um standard mantido pela W3C

**IDE** *Integrated Development Environment*

**SIGON** *Sistema de Informação para Gestão do ON.2*

**ON.2** *Programa Operacional do Norte*

**FEDER** *Fundos Europeus de Desenvolvimento Regional*

**SI** *Sistema de Informação*

**BD** *Base de Dados*

**DAO** *Data Access Object*