

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

Geração automática de classes C# a partir de um Dataset

Vitor Martins Alves (40657)
Hermano Andrade (40634)
Ernesto Costa (40618)
Tiago Marques (41034)

Supervisão: Pedro Rangel Henriques
Daniela da Cruz

3 de Julho de 2008

Data de Recepção	
Responsável	
Avaliação	
Observações	

Resumo

O trabalho foi realizado no contexto da disciplina de Laboratórios IV, 3ºano de LEI, e consistiu no desenvolvimento de uma aplicação com a funcionalidade de gerar automaticamente classes C# a partir de um Dataset em formato XML. Assim sendo, para cada tabela será gerada uma classe C# em que os atributos serão as colunas da tabela e os métodos serão os gets/sets desses atributos. A linguagem escolhida para o seu desenvolvimento foi Visual C#, uma linguagem orientada a objectos que faz parte da plataforma .Net da Microsoft. O programa permite ao utilizador a escolha das tabelas e dos campos que deverão gerar as classes, assim como o tipo dos atributos. Através desta aplicação, espera-se criar um importante auxílio aos programadores, poupando tempo e trabalho.

Área de Aplicação: Engenharia de software

Palavras-Chave: Dataset, XML, Classes, C#.

Agradecimentos

Ao longo deste semestre tivemos a oportunidade de contar com o apoio de diversas pessoas que, directa ou indirectamente, deram o seu contributo no desenvolvimento deste projecto.

Em primeiro lugar, agradecemos a todos aqueles que, com o seu saber, a sua colaboração e o seu apoio crítico, dispuseram do seu tempo para debater connosco orientações e práticas à medida que tentámos atingir os objectivos propostos e a que nos propusemos.

Estamos especialmente gratos aos professores Dr. Pedro Rangel Henriques e Dra. Daniela da Cruz pelas suas valiosas sugestões, pensamentos e tempo dispendido, que tornaram o nosso trabalho mais objectivo e simples, bem como pela sua contínua disponibilidade.

Um especial agradecimento ao aluno finalista da LESI, Pedro Cabral, presentemente inserido num estágio em que uma das principais ferramentas é a plataforma .NET, que nos deu alguma ajuda relativamente á sintaxe desta.

Naturalmente, qualquer eventual omissão ou erro é da nossa inteira responsabilidade.

Conteúdo

Agradecimentos	i
Conteúdo	ii
Lista de Figuras	iv
Lista de Tabelas	v
1 Introdução	1
2 Enunciado do Problema	2
3 Concepção da Solução / Arquitectura do Sistema	3
3.1 Análise de Requisitos	3
3.2 Arquitectura do Sistema	4
3.3 Estrutura de Dados	4
3.3.1 Campo	4
3.3.2 Campos	4
3.3.3 Tabela	5
3.3.4 Tabelas	5
3.3.5 Exemplo dos ficheiros de input e output	5
3.3.6 Diagrama de Classes	7
4 Implementação	8
4.1 Linguagens de Programação	8
4.2 Tecnologias	9
4.2.1 Microsoft .NET	9
4.2.2 XML	9
4.3 Implementação do Sistema	9
4.3.1 Parsing do DataSet	9
4.3.2 Selecção das Tabelas e Campos	9
4.3.3 Geração das Classes	10
4.3.4 Sistema da Aplicação	10

5	Testes e Resultados	12
6	Conclusão	16
	Bibliografia	17
	Referências WWW	18
	Glossário	19

Lista de Figuras

3.1	Arquitectura do Sistema	4
3.2	Diagrama de Classes dos Dados da Aplicação	7
4.1	Sistema da Aplicação	11
5.1	Erro XML	12
5.2	Programa	13
5.3	Inserir nome do projecto	14

Lista de Tabelas

Capítulo 1

Introdução

O projecto enquadrou-se na disciplina de Laboratórios IV, do 3º ano da Licenciatura em Engenharia Informática, e consistiu no desenvolvimento de uma aplicação que nos permita a geração automática de classes C# a partir de um Dataset. Para cada tabela irá ser gerada uma classe C# em que os atributos serão as colunas da tabela e os métodos serão os gets/sets desses atributos. A aplicação foi desenvolvida para o Departamento Informática, da Universidade do Minho, fazendo parte de um projecto SIGON.2, um sistema de informação real, suportado na web, para submissão, avaliação e aprovação, acompanhamento e auditoria de candidaturas a programas de financiamento para o desenvolvimento regional do Norte de Portugal.

Depois de uma fase de avaliação do problema, pareceu-nos mais lógico optar pelo uso da linguagem Visual C#, uma vez que temos por objectivo obter resultados relacionados com esta linguagem. Poderia-mos ter optado entre outras, por exemplo, pelo uso da linguagem XSL, para o tratamento de documentos XML, mas como o Visual C# também nos permite extrair informação de ficheiros XML decidimos torná-lo a nossa opção. Relativamente ao interface foi idealizado de forma simples mas com algum cuidado em criar um ambiente agradável e intuitivo, sofrendo por isso algumas alterações ao longo da criação do projecto. O objectivo deste projecto, como já foi referido, será então criar uma aplicação que ao receber um Dataset, o compile, gerando um conjunto de classes em C#.

Dos diversos projectos propostos, este foi um dos que nos motivou mais interesse, pois iria permitir um consolidar de conhecimentos recebidos ao longo do curso esperando também adquirir conhecimentos e prática na manipulação das linguagens de programação C#.net e XML, os quais são para já algo limitados.

Após termos apresentado o projecto, no capítulo três será exposta a concepção da aplicação, onde irão ser abordados diversos assuntos que nos permitiram passar para a implementação. No capítulo quatro será exposta a implementação da aplicação, onde fazemos uma breve referência às tecnologias usadas, daremos uma explicação pormenorizada do desenvolvimento da aplicação, entre outros assuntos. Em seguida, serão mostrados testes e resultados, e por fim chegaremos à conclusão que irá dar uma elucidação sobre todo o trabalho desenvolvido, juntamente com uma análise crítica e possíveis trabalhos futuros.

Capítulo 2

Enunciado do Problema

O SIGON.2 é um sistema de informação real, suportado na web, para submissão, avaliação e aprovação, acompanhamento e auditoria de candidaturas a programas de financiamento para o desenvolvimento regional do Norte de Portugal. Preenchidos os formulários e carregados os anexos, as candidaturas são gravadas num base de dados central, que vai servir de suporte a todo o sistema de Back-Office. através do qual os membros da Entidade Gestora do Programa Operacional (EGPO) vão fazer a avaliação e aprovação das candidaturas, bem como acompanhar todas as tarefas ao longo da execução dos projectos e o controlo final, após o encerramento. O sistema está a ser desenvolvido em .NET, usando ASP, C# e SQL-Server2005. Pretende-se, neste projecto, criar um Gerador de C# que tome como ponto de partida o Dataset que se está a usar numa Aplicação (o SIGON.2 será usado como caso de estudo para testar o Gerador a desenvolver) e gere para cada tabela uma classe C# em que os atributos serão as colunas da tabela e os métodos serão os gets/sets desses atributos. Ou seja, pretende-se criar uma aplicação que tem como input um ficheiro no formato XML que contem um Dataset e partir daí gere como output um conjunto de classes no formato C#.

Capítulo 3

Concepção da Solução / Arquitectura do Sistema

3.1 Análise de Requisitos

Para o desenvolvimento e implementação deste projecto, tivemos a liberdade de pesquisar na Internet por aplicações do mesmo género, para assim adoptarmos alguma ideias. Contudo, depois de algum tempo de pesquisa e com alguma surpresa não encontramos nenhuma aplicação deste tipo. Ficamos então com a possibilidade de desenvolver uma aplicação sem influências de outras já existentes tornando ainda mais interessante este projecto. Tomámos em conta vários aspectos importantes que mereceram a nossa atenção desde o início. Desde logo, a nível de software de implementação, o requisito fundamental era encontrar uma ferramenta que permitisse ter uma interacção adequada com ficheiros em formato XML, já que esses seriam a fonte de toda a informação com a qual pretendíamos trabalhar. A escolha acabou por recair na linguagem Visual C#, do pacote de programas Visual Studio 2005, da plataforma .NET da Microsoft (mais tarde iremos fazer uma análise do porquê desta escolha, bem como falar das alternativas existentes). Depois de extrair toda a informação do ficheiro, esta seria guardada num estrutura de dados, que achámos ser a mais adequada (num próximo capítulo iremos revelar uma análise mais detalhada sobre essa estrutura), pois a nossa intenção foi dar a possibilidade de escolha ao utilizador, relativamente às tabelas e aos campos que deveriam gerar as classes. Um factor revelante, foi o facto de o formato do ficheiro XML que contém o dataset, apenas nos permitir extrair o nome das tabelas, o nome dos campos de cada tabela e o valor desses mesmos campos, e como pretendíamos gerar as classes com os tipos dos atributos correctos, sentimos a necessidade do uso de expressões regulares para sabermos o correcto tipo de cada atributo. No entanto alguns campos das tabelas poderão estar vazios. Nesse caso, o tipo predefinido será string. Sendo assim, fizemos questão de possibilitar ao utilizador a mudança do tipo dos atributos caso achasse necessário. Relativamente à interface da aplicação, existia a importância da simplicidade da mesma. Tomámos então a decisão de criar uma aplicação com as funcionalidades rapidamente acessíveis, de modo intuitivo, fácil compreensão e com a informação necessária ao utilizador para a geração das classes pretendidas.

3.2 Arquitectura do Sistema

Através da imagem 3.1 é demonstrado de uma forma simples como foi idealizada a arquitectura do sistema.



Figura 3.1: Arquitectura do Sistema

3.3 Estrutura de Dados

Nesta secção vamos apresentar os dados mais importantes para a aplicação e um exemplo do ficheiro de input e output. Tentado abstrair-nos da linguagem de programação, podemos dizer que os dados que teriam de ser sempre guardados pela aplicação são os campos e as tabelas. Todos estes dados teriam de ser guardados numa estrutura eficiente. Nós optamos por gravá-los em colecções, nomeadamente através da Classe `CollectionBase`. Decidimos usar colecções devido à sua enorme capacidade, sempre que os elementos são inseridos a capacidade da colecção é aumentada através de realocação e também devido à sua eficiência e facilidade de inserção, remoção e procura.

3.3.1 Campo

A classe `Campo` vai servir para guardar a informação de cada campo do dataset, logo é constituída por uma string nome que permite guardar o nome dos campos, uma string tipo que ficará com o tipo desse campo e também um atributo booleano que nos vai permitir saber se o utilizador pretende utilizar esse campo para gerar as classes pretendidas, neste caso o atributo chama-se `boxCheck`.

3.3.2 Campos

A classe `Campos` é unicamente constituída por uma colecção da classe `Campo`. A colecção utilizada, foi a `CollectionBase`. A escolha desta colecção, como já foi referido, deve-se à sua eficiência e facilidade de inserção, remoção e procura.

3.3.3 Tabela

A classe Tabela guarda a informação referente a cada tabela do dataset, e é constituída por uma string nome que permite guardar o nome das tabelas, um atributo do tipo Campos, permitindo ter assim a colecção de campos referentes a essa tabela e também um atributo booleano, tal como na classe Campo, que nos vai permitir saber se o utilizador pretende utilizar essa tabela para gerar as classes pretendidas, neste caso o atributo também se chama boxCheck.

3.3.4 Tabelas

A classe Tabelas é semelhante á classe Campos, é apenas constituída por uma colecção da classe Tabela. A colecção utilizada, também foi a CollectionBase, devido aos motivos já enunciados.

3.3.5 Exemplo dos ficheiros de input e output

Exemplo de um ficheiro XML contendo um dataset, com apenas 6 tabelas e alguns campos, que gerariam caso o utilizador pretendesse no máximo 6 classes distintas.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<NewDataSet>
  <dados>
    <eixo>3</eixo>
    <objectivo>15</objectivo>
    <regulamento>6</regulamento>
    <avisos>6</avisos>
  </dados>
  <parceiro>
    <nipc>505359863</nipc>
    <designacao>Instituto de História e Arte ãCristis-I.H.A.C.</designacao>
    <endereco>Rua de Santa Margarida, ºn 181, 4710 - 306 Braga</endereco>
    <concelho>101020303</concelho>
    <freguesia>010102030351</freguesia>
  </parceiro>
  <projecto-dados>
    <idProjecto>0</idProjecto>
    <designacao>çãInventariio da Arquidiocese de Braga</designacao>
    <dataInicio>2008-06-01</dataInicio>
    <dataFim>2010-05-31</dataFim>
  </projecto-dados>
  <dimensao>
    <idProjecto>0</idProjecto>
    <dimensao-financiamento>1</dimensao-financiamento>
    <dimensao-territorial>1</dimensao-territorial>
    <tema-id>-1</tema-id>
  </dimensao>
  <temas>
    <idProjecto>0</idProjecto>
    <tema-id>-1</tema-id>
    <tema>58</tema>
    <percentagem>100</percentagem>
  </temas>
  <articulacoes>
    <idProjecto>0</idProjecto>
    <resposta>>false</resposta>
    <tipo-continuacao-programa />
    <tipo-continuacao-codigo />
    <tipo-complementar-programa />
    <tipo-complementar-codigo />
  </articulacoes>
</NewDataSet>
```

Exemplo de um ficheiro de saída, ou seja, uma classe C#. Neste caso será uma classe relativamente ao dataset mostrado anteriormente, apenas da tabela projecto-dados com os campos idProjecto, designacao e dataInicio. De salientar que todos os hífens tanto nos nomes das tabelas como nos campos, serão substituídos pelo underscore devido á sintaxe da linguagem C#.

```
using System;
using System.Collections.Generic;
using System.Text;

namespace ProjetoLabs4
{
    class projecto_dados
    {
        private int idProjecto;
        private string designacao;
        private DateTime dataInicio;

        public projecto_dados()
        {
            this.idProjecto = 0;
            this.designacao = "";
            this.dataInicio = new DateTime();
        }

        public projecto_dados(int idProjecto, string designacao, DateTime
dataInicio)
        {
            this.idProjecto = idProjecto;
            this.designacao = designacao;
            this.dataInicio = dataInicio;
        }

        public int getIdProjecto ()
        {
            return idProjecto;
        }

        public string getDesignacao()
        {
            return designacao;
        }

        public DateTime GetDataInicio()
        {
            return dataInicio;
        }

        public void setIdProjecto (int idProjecto)
        {
            this.idProjecto=idProjecto;
        }

        public void setDesignacao (string designacao)
        {
            this.designacao= designacao;
        }

        public void setDataInicio (DateTime dataInicio)
        {
            this.dataInicio = dataInicio;
        }
    }
}
```

3.3.6 Diagrama de Classes

Na figura 3.2 podemos ver os dados descritos neste capítulo, já normalizados para uma aplicação C#. Nela também podemos visualizar o nome dos atributos e de todos os métodos a serem usados.

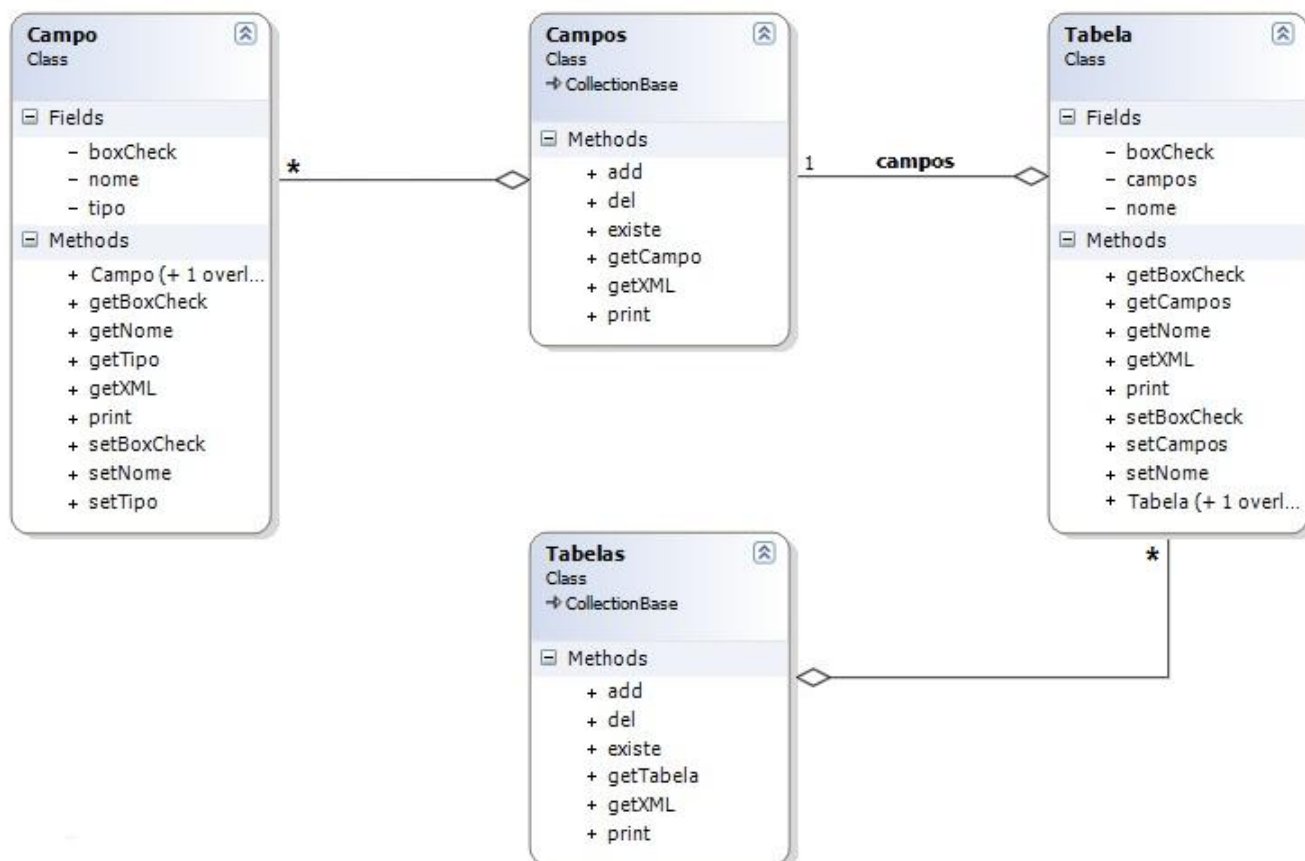


Figura 3.2: Diagrama de Classes dos Dados da Aplicação

Capítulo 4

Implementação

Como já foi referido o software utilizado para a implementação desta aplicação, foi o Visual C#, do pacote de programas Visual Studio 2005, da plataforma .NET da Microsoft. Neste capítulo, iremos falar sobre a escolha desta linguagem e as alternativas consideradas, vamos também referenciar as tecnologias utilizadas na implementação do projecto, e também explicar como foi implementado o sistema.

4.1 Linguagens de Programação

Depois de estudado o problema, chegou a altura de decidirmos qual seria a linguagem a usar para criar a implementação do software e foram nos propostas varias alternativas, todas com suporte para a linguagem XML, sendo este o critério mais importante para a escolha da linguagem em que o programa iria ser desenvolvido. As linguagens propostas foram, a linguagem Java usando o ambiente Netbeans, a linguagem XSL, e por fim a linguagem C# da plataforma .NET da Microsoft. Em relação ao Java, e como já foi referido seria utilizado o ambiente Netbeans, e devido há já existência de alguma experiência relativamente a este ambiente, através de trabalhos feitos em outras cadeiras, seria então de esperar que a escolha recaísse sobre ele. Mas, como um dos principais objectivos que tínhamos com a realização deste projecto seria também adquirir conhecimentos em outras linguagens, foi nossa decisão colocarmos de parte esta linguagem. Relativamente á linguagem XSL, é uma linguagem muito poderosa relativamente á manipulação de ficheiros XML, pois consegue transformar um documento XML em outros formatos de documentos (HTML, TeX, PostScript, RTF), além de poder utilizar alguns elementos de estilo disponíveis em CSS. A linguagem XSL pode ser utilizada para acrescentar aspectos de apresentação aos elementos de um documento XML. Desta forma, é possível criar múltiplas representações da mesma informação a partir de vários documentos XSL diferentes aplicados a um único documento XML. Logo esta linguagem estaria bem colocada para ser utilizada no tratamento de documentos XML, no entanto, a nossa opção acabou por recair pela linguagem Visual C#, do pacote Visual Studio 2005, da plataforma .NET da Microsoft. Depois de alguma pesquisa verificamos a existência de uma ferramenta bastante poderosa para a manipulação de ficheiros XML, através da classe XmlDocument, sendo este um requisito fundamental para a escolha da linguagem. Outro dos nossos objectivos era ganhar conhecimento e prática de novas linguagens, o que acontecia com esta linguagem pois, os nossos conhecimentos ainda eram bastantes limitados relativamente á plataforma .NET. O último motivo que nos levou a optar por esta linguagem, foi o facto de todo o sistema SIGON.2 estar a ser desenvolvido em .NET.

Assim sendo nada melhor que desenvolvermos também a nossa aplicação nesta linguagem, o que acabou por ser uma escolha bastante acertada e que alcançou todas as nossas expectativas em relação á linguagem.

4.2 Tecnologias

4.2.1 Microsof .NET

Microsoft .NET é uma iniciativa da Microsoft em que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET, pode ser executado em qualquer dispositivo ou plataforma que possua um framework: a "Plataforma .NET"(.NET Framework). Com ideia semelhante à plataforma Java, o programador deixa de escrever código para um sistema ou dispositivo específico, e passa a escrever para a plataforma .NET.

4.2.2 XML

XML é uma linguagem de anotação para documentos que contêm informação estruturada. O seu principal objectivo é facilitar a partilha de dados entre diferentes sistemas, em particular via Internet. Separa o conteúdo da formatação, concentrando-se na estrutura da informação e não na sua aparência. É uma linguagem simples e legível tanto para pessoas como para software. É independente de plataforma e relativamente imune a mudanças na tecnologia.

4.3 Implementação do Sistema

Nesta secção, vamos revelar como foi implementado o sistema, que se baseou em três fases para atingir o estado final da aplicação. Para complementar a explicação do sistema e tornar a compreensão deste mais simples é exibida a figura 4.1.

4.3.1 Parsing do DataSet

Uma das principais fases da elaboração do projecto era fazer um parsing que nos permitisse extrair a informação de um ficheiro XML. E como já foi referido anteriormente, foi utilizada a classe XmlDocument. Esta é uma classe bastante poderosa, suporta métodos que nos permitem efectuar operações sobre o documento como um todo (por exemplo, coloca-lo em memória ou guardar o XML para um ficheiro). Esta classe, permite-nos também uma forma de ver e manipular todos os nós de um documento XML, o que nos permitiu, ler toda a informação do ficheiro XML, e preencher a estrutura de dados. Uma nota importante, foi o necessário uso de expressões regulares, para saber os tipos dos atributos, pois no ficheiro XML não existia informação sobre o tipo de atributo. Para isso, a plataforma .NET permite-nos utilizar a classe Regex, esta contém métodos que nos permite o uso de expressões regulares, facilitando o trabalho a qualquer programador.

4.3.2 Selecção das Tabelas e Campos

Depois de efectuado o parsing e preenchida a estrutura, foi nossa intenção proporcionar ao utilizador da aplicação, uma forma que lhe permitisse controlar quais as tabelas e os campos que

iriam gerar as classes. O objectivo era obter uma forma de selecção intuitiva e simples, objectivo esse que nos parece ter sido atingido. A forma como foi implementado foi algo trabalhosa, utilizaram-se duas listas, uma contem as tabelas e a outra os campos. Sempre que existia alguma mudança em qualquer das listas, através da utilização de eventos, era accionada uma acção que verifica qual a modificação e efectua as alterações necessárias na estrutura de dados, efectuando sempre que preciso também mudanças na outra lista. É importante referir, que devido á inexistência de informação sobre qual o tipo dos atributos no ficheiro XML, e mesmo utilizando expressões regulares para sabermos o tipo, existem campos que estão vazios, neste caso o tipo predefinido era string, sendo assim demos também a possibilidade ao utilizador de alterar o tipo de qualquer atributo sempre que tal achasse.

4.3.3 Geração das Classes

Por fim, depois de o utilizador decidir as quais classes e os atributos escolhidos para formar as classes, estas poderão então ser criadas. Como existem algumas classes e campos que continham hífen estes foram substituidos pelo caracter underscore, devido á sintaxe da linguagem C#. Antes de criar as classes, será pedido ao utilizador para inserir o nome do projecto a que essas classes se destinam, pois este é referenciado nas classes C#.

4.3.4 Sistema da Aplicação

Na figura 4.1, está descrito o modo de funcionamento da aplicação através de uma forma bastante simples, com o intuito de ajudar o leitor a ter uma boa percepção sobre a implementação do sistema.

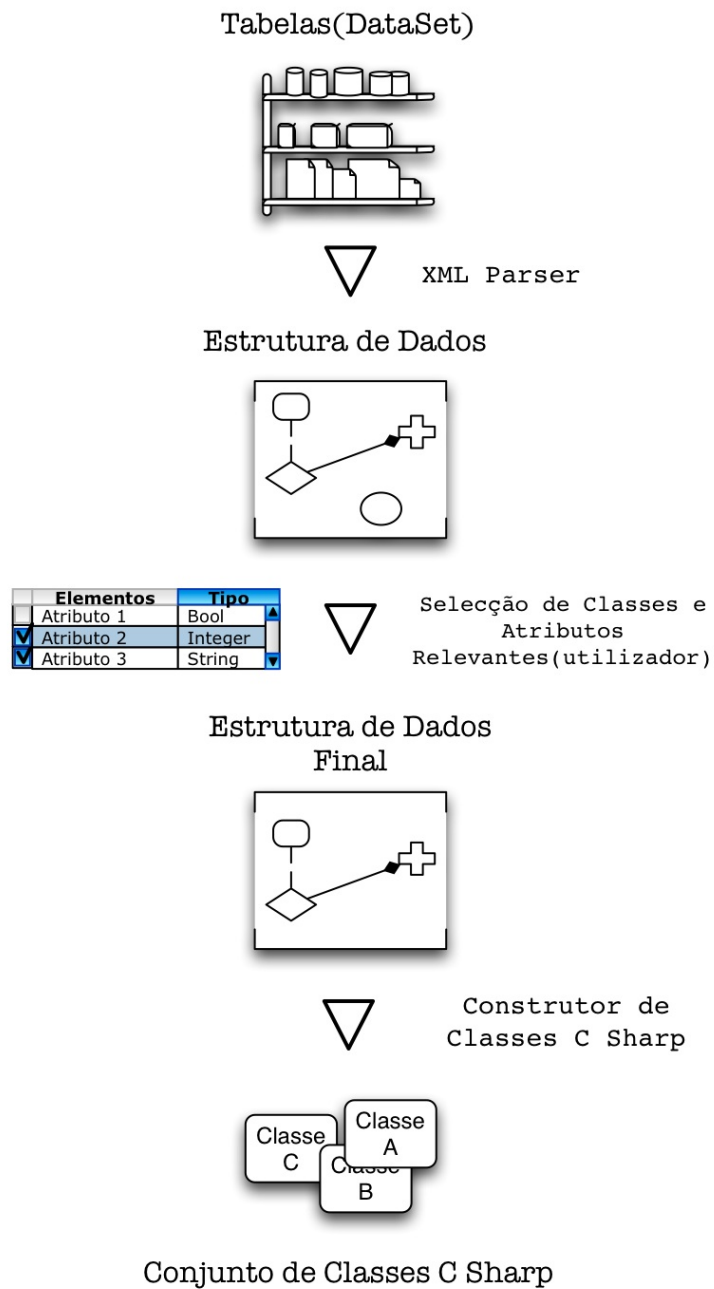


Figura 4.1: Sistema da Aplicação

Capítulo 5

Testes e Resultados

Neste capítulo iremos apresentar algumas imagens da aplicação desenvolvida e resultados obtidos através da sua utilização.

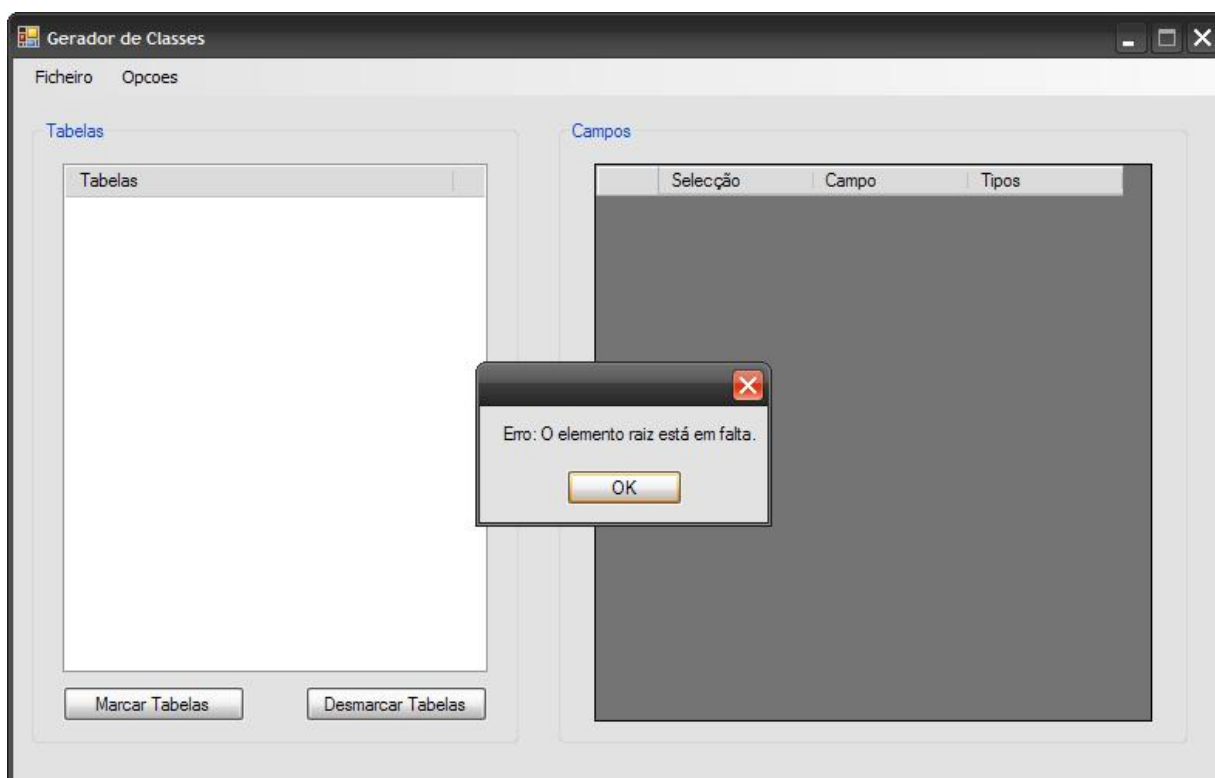


Figura 5.1: Erro XML

Depois de o programa ser executado, e abrirmos um ficheiro XML, caso exista algo de errado nesse ficheiro, é mostrado ao utilizador uma mensagem com o erro que ocorreu ao tentar extrair a informação dele. Como podemos ver através da imagem 5.1, que neste caso sendo um ficheiro XML em branco deu o erro "O elemento da raiz está em falta."

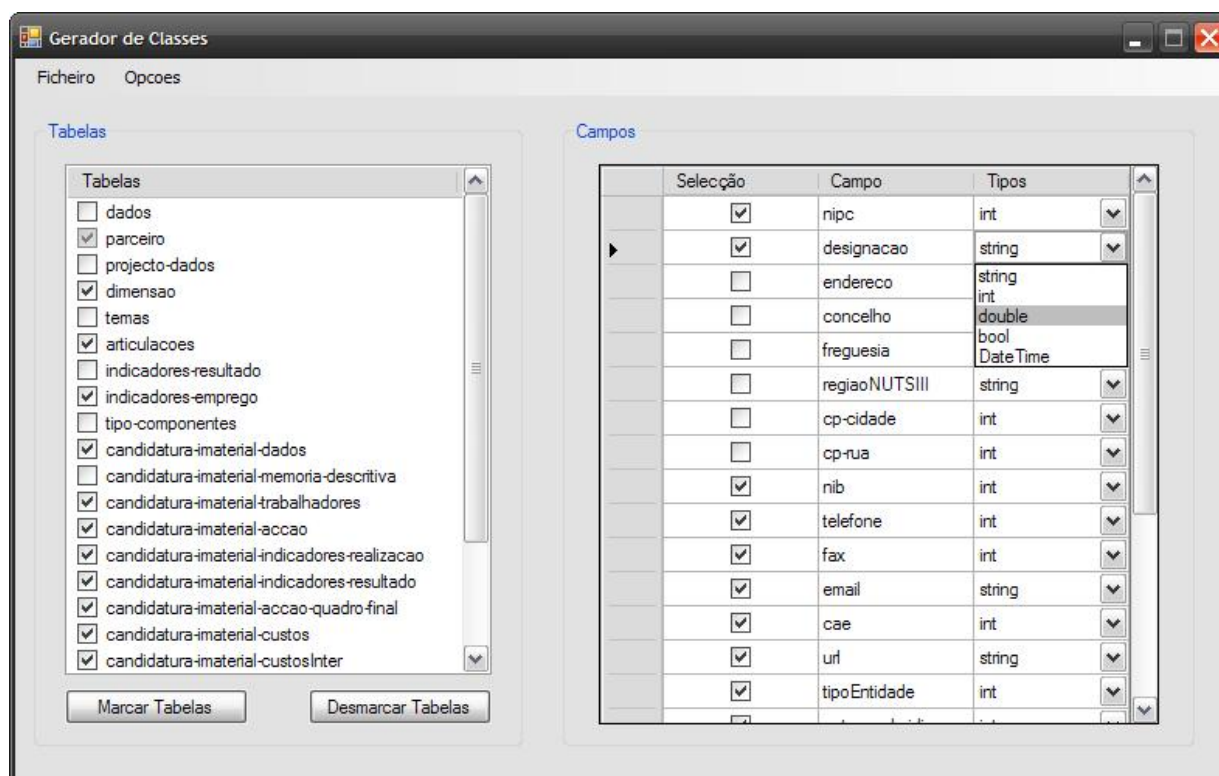


Figura 5.2: Programa

Depois de ler o ficheiro, e extrair a informação, as tabelas e campos serão disponibilizadas pelo programa, estando marcados por definição. Na janela principal existem dois itens no menu, Ficheiro e Opções. Seleccionando o item Ficheiro, temos as opções de Abrir ficheiro e Sair. Já no item Opções, vamos poder gerar as classes, clicando em Gerar classes. Existem também os botões Marcar e Desmarcar tabelas, que nos facilitam o trabalho caso apenas seja necessário gerar poucas classes. Temos a possibilidade de alterar o tipo dos campos, marcar e desmarcar tabelas e campos. Podemos ter uma melhor noção do programa através da imagem 5.2 e visualizar como é disposta a informação.

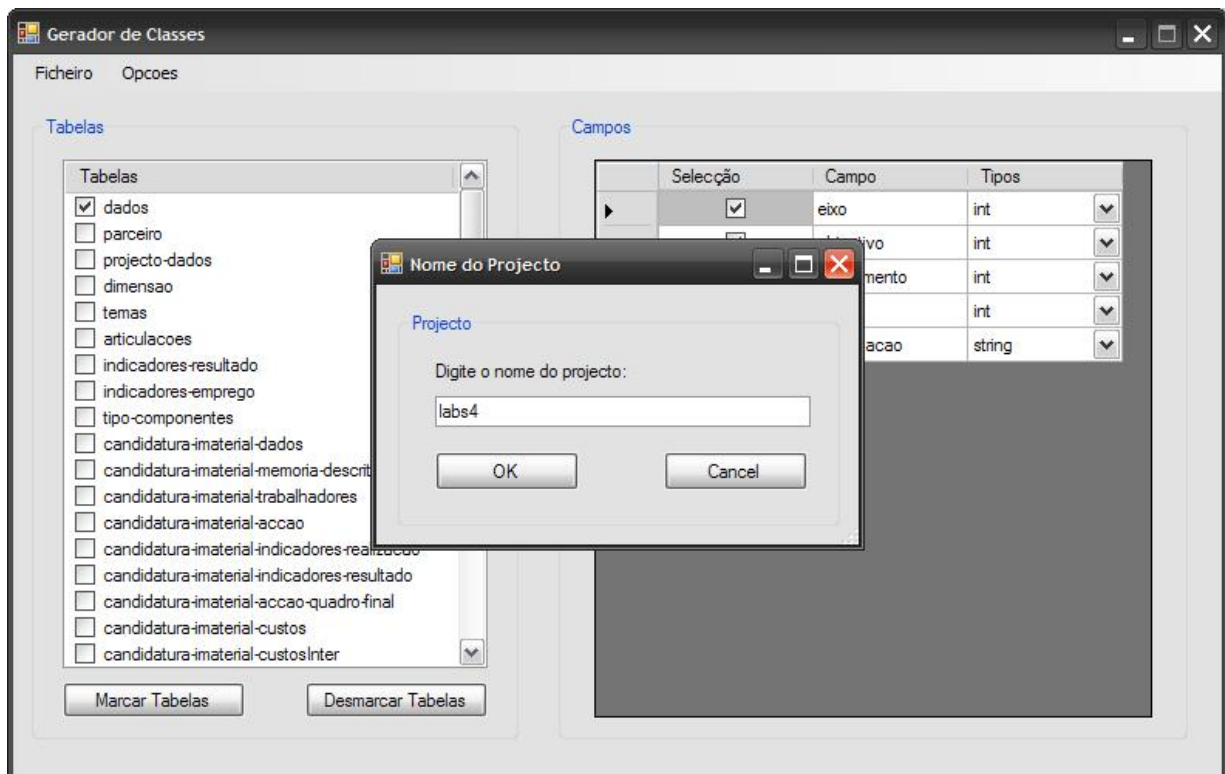


Figura 5.3: Inserir nome do projecto

Depois de termos escolhido as tabelas e campos, podemos então, gerar as classes. No entanto, antes de serem geradas, será pedido ao utilizador para escrever o nome do projecto em que as classes serão inseridas, ja que é informação que as classes C# têm no seu código. A imagem 5.3 mostra como é tudo é processado.No caso da figura 5.3 seria gerada uma classe com o nome dados.cs com o seguinte conteúdo:

```
using System;
using System.Collections.Generic;
using System.Text;

namespace labs4
{
    class dados
    {
        private int eixo;
        private int objetivo;
        private string designacao;

        public dados()
        {
            eixo = 0;
            objetivo = 0;
            designacao = "";
        }

        public dados(int eixo, int objetivo, string designacao)
        {
            this.eixo = eixo;
            this.objetivo = objetivo;
            this.designacao = designacao;
        }

        public dados(dados outro)
        {
            this.eixo = outro.getEixo();
            this.objetivo = outro.getObjetivo();
            this.designacao = outro.getDesignacao();
        }

        public int getEixo()
        {
            return this.eixo;
        }

        public int getObjetivo()
        {
            return this.objetivo;
        }

        public string getDesignacao()
        {
            return this.designacao;
        }

        public void setEixo(int eixo)
        {
            this.eixo=eixo;
        }

        public void setObjetivo(int objetivo)
        {
            this.objetivo=objetivo;
        }

        public void setDesignacao(string designacao)
        {
            this.designacao=designacao;
        }
    }
}
```

Capítulo 6

Conclusão

Foi nossa intenção efectuar um relatório com todo cuidado possível, tendo como objectivo a inclusão de toda a informação que achámos relevante. Desde a concepção do sistema até á sua implementação, existiu a preocupação de expor todo o trabalho desenvolvido de forma clara e simples. Utilizámos algumas imagens, esquemas e uma breve explicação de conceitos, esperando deste modo facilitar a leitura do relatório e compreensão do trabalho por parte do leitor.

Quanto ao trabalho desenvolvido, e após uma análise cuidadosa dos resultados, afirmamo-nos bastante satisfeitos com o desempenho da nossa aplicação. Todas as funcionalidades propostas foram implementadas e o seu funcionamento preenche perfeitamente as expectativas, desde o parser XML até ao interface simples e agradável, que foi uma das preocupações presentes desde o início. Não podemos deixar de referir que se tratou de um projecto bastante trabalhoso. Factores como o desconhecimento inicial, da nossa parte, das linguagens C# e XML, amplamente utilizadas na implementação da nossa aplicação, são de realçar, tal como o facto da inexistência de um software do mesmo tipo, pelo qual pesquisámos sem qualquer sucesso, afim de extrair ideias que nos pudessem ser úteis.

Tendo em conta estes factos, deu-nos uma alegria imensa ter como resultado final uma aplicação totalmente operacional e bastante *user-friendly*, bem como os conhecimentos adquiridos sobre vários componentes da plataforma .Net, objectivo que esteve sempre presente na forma como encarámos o desafio proposto neste projecto.

Como trabalho futuro foram pensadas algumas funcionalidades que tornariam este projecto ainda mais interessante e útil. Entre estas, a possibilidade de gerar classes em várias linguagens, permitindo assim ao utilizador escolher a que melhor se aplicaria ao trabalho que está a desenvolver. Relativamente ao tipo de input da aplicação, poderiam ser implementadas formas alternativas de aceder à informação desejada, como por exemplo, em vez da utilização de um ficheiro XML como fonte, poderíamos, por exemplo, ligar directamente a uma base de dados.

Em suma, pensamos que os objectivos a que nos propusemos inicialmente, foram cumpridos na totalidade, bem como alguns que surgiram durante o estudo e implementação do projecto. Não entrámos em considerações que pudessem sacrificar o cumprimento dos prazos estabelecidos ou o cumprimento satisfatório do que nos propusemos a realizar. No entanto, existem bastantes funcionalidades que poderiam ser implementadas para enriquecer o projecto, das quais já demos alguns exemplos.

Bibliografia

Hoffman, Kevin, *Microsoft Visual C# 2005 Unleashed*, Sams, 2006

Marshall, Donis, *Programming Microsoft Visual C# 2005: The Language*, Microsoft Press, 2006

Gittleman, Art, *C#.NET Illuminated*, Jones and Bartlett Publishers, 2005

Referências WWW

01 **support.microsoft.com**

Página de suporte e ajuda da microsoft, que fornece bastante ajuda a muitos níveis. No nosso caso, foi um importante apoio em algumas dúvidas sobre C#.net.

02 **msdn.microsoft.com**

Página disponibilizada pela microsoft, que contém muita informação sobre todas as linguagens da plataforma .NET. Desde tutoriais a exemplos, foi a esta página que mais recorreremos e em primeiro lugar.

03 **www.c-sharpcorner.com/**

Uma página, que contém muita informação sobre C#.net e não só, que nos permitiu tirar algumas informações.

04 **www.codeproject.com**

O Code Project tem como principal função, providenciar aos programadores uma forma de trocar ideias, e conseguir assim disponibilizar muitos recursos de ajuda.

05 **www.linhadecodigo.com.br**

Uma página em português do Brasil, que contém também muita informação sobre as mais variadas linguagens de programação, inclusive C#.net.

06 **www.ebookee.com**

Uma página que contém bastantes tutoriais sobre os mais variados temas, inclusive linguagens de programação.

07 **en.wikipedia.org**

Página principal da wikipedia, enciclopédia online livre, assente num modelo colaborativo, onde se pode encontrar informações sobre quase tudo.

Glossário

XML *Extensible Markup Language*

XSL *Extensible Stylesheet Language*

HTML *Hypertext Markup Language*

RTF *Rich Text Format*

CSS *Cascading Style Sheets*