Universidade Federal de Santa Catarina
Centro Tecnológico
Departamento de Informática e Estatística
Curso de Ciências da Computação

Uma Ferramenta de Apoio ao ORM Gendal para Mapeamento e Consultas

Autor: Diego Magno da Silva

Monografia de Conclusão do Curso de Ciências da Computação

Orientador: Ronaldo Santos Mello

Florianópolis, julho de 2012.

Diego Magno da Silva

Uma Ferramenta de Apoio para Mapeamento e Consultas ao ORM Gendal

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de Bacharel, do curso de Ciências da Computação na Universidade Federal de Santa Catarina.

Profº orientador: Ronaldo Santos Mello

Florianópolis, 2012.

Resumo

Em uma aplicação orientada a objetos que utiliza um banco de dados relacional para persistência de seus dados, uma ferramenta de ORM (mapeamento objeto-relacional) é importante para integrar os objetos da aplicação ao banco de dados. Uma solução neste contexto é o ORM Gendal (*Generic Data Access Library*), um *framework* .Net, que é utilizado pela IDE Visual Studio. Os principais pontos fortes do Gendal são a simplicidade de definição dos mapeamentos e definição de consultas complexas. O DBClassMapper é uma ferramenta desenvolvida para o Visual Studio 2010 com a finalidade de auxiliar a utilização do Gendal. Ela provê uma melhoria de usabilidade ao Gendal ao permitir a construção interativa e simples de mapeamentos e consultas complexas ao banco de dados. Comparado com trabalhos relacionados, verifica-se que a Microsoft vem melhorando o seu ORM Entity Framework visando facilitar a sua utilização. O problema é que mesmo no Visual Studio 2010 o Entity Framework não oferece ferramentas de suporte para mapeamento. Já o o Hibernate (outro ORM para .Net) não tem nenhuma ferramenta para essa IDE e as ferramentas existentes para outras plataformas não permitem a construção interativa de consultas a partir das classes de objetos. Estas deficiências motivaram o projeto do DBClassMapper sobre o ORM Gendal.

Abstract

In an object-oriented application that uses a relational database for persistence of their data, an ORM tool is important to integrate the application objects to the database. A solution in this context is ORM Gendal (Generic Data Access Library), a .Net framework, which is used by the Visual Studio IDE. Its main strengths are the definition of the mapping and definition of complex queries. DBClassMapper is a tool designed for Visual Studio 2010 in order to assist the use of Gendal. The main facilities of DBClassMapper are the interactive construction and simple mappings and complex queries to the database. Compared to related work, we note that Microsoft is improving its ORM Entity Framework to facilitate its use. The problem is that even in the Visual Studio 2010, the Entity Framework does not support tools for mapping. Hibernate (another ORM for .Net) has no tool for this IDE, and existing tools for other platforms do not allow interactive construction of queries from the classes of objects. These limitations have motivated the design of DBClassMapper for Gendal.

Sumário

Lista de Figuras

Lista de Reduções

- ORM: Object relational-mapping
- EF: Entity Framework
- SGBD: Sistema gerenciador de banco de dados
- SQL: Structured Query Language
- IDE: Integrated Development Environment
- CRUD: Create, Read, Update and Delete
- VB: Visual Basic
- TCC: Trabalho de Conclusão de Curso
- MVVM: Model View View-Model

1. Introdução

1.1. Visão Geral

Em um sistema orientado a objetos em que a utilização de um banco de dados para persistência é indispensável, o suporte de uma tecnologia *Object-Relational Mapping* (ORM) torna-se parte importante no auxílio para integrar a persistência relacional dos dados com a orientação a objetos do sistema. Com esta tecnologia, o programador não precisa se preocupar com os comandos em linguagem SQL [9]. O Gendal é um ORM para o framework .Net que foi criado por Diego Magno da Silva. Ele é utilizado pela IDE Visual Studio e pode ser baixado no site www.deltacon.com.br. A utilização de um ORM aumenta a produção de uma equipe, pois reduz o problema da impedância ou *impedance mismatch* [1], uma vez que o programador da aplicação não precisa se preocupar com a construção de comandos na linguagem SQL para realizar a definição e manipulação de dados [9].

O Gendal resolveu o problema da falta de um ORM que funcionasse com o banco de dados Caché da Intersystems. O Caché é um banco de dados orientado a objetos, mas que tem uma camada relacional para compatibilidade de SQL, conexões ODBC, dentre outros. Por ser um banco de dados muito específico, os ORMs mais utilizados no .Net não dão suporte a ele como o NHibernate [11] e o Entity Framework da Microsoft [4]. O Gendal nasceu basicamente para tentar resolver este problema, mas como a sua criação foi moldada nas premissas do que um ORM deveria ser, acabou por se tornar um ORM muito fácil e simples de se utilizar.

Em uma breve comparação, é possível observar as principais diferenças entre o Gendal e seus dois principais concorrentes, o NHibernate e o Entity Framework, como já citados. O NHibernate teve a sua última versão para o .Net 2.0 [11], sendo que o mesmo está na versão 4.0, além de não ter nenhuma ferramenta auxiliadora para o Visual Studio. Mesmo assim, o NHibernate é um bom ORM que tem muita flexibilidade, e é esse excesso de flexibilidade que o deixa um pouco mais complexo, em termos de uso, que o Gendal. O Entity Framework requer conhecimentos de SQL ou de Lambda Expressions para se gerar consultas mais complexas [8], ou seja, consultas que envolvem várias tabelas. Neste contexto, o Gendal é uma biblioteca extremamente mais simples e funcional.

Apesar de ser um ORM simples, o Gendal sozinho ainda requer muito trabalho manual para os mapeamentos e mesmo sendo totalmente orientado a objetos, algumas consultas podem se tornar complexas de se formular. Estes problemas existem também nos outros ORMs e a maneira mais simples para resolvê-los foi criando uma ferramenta que os auxiliasse.

1.2. Objetivos

1.2.1. Geral

Desenvolver uma ferramenta para auxiliar o mapeamento e a geração de consultas no ORM Gendal para a IDE Visual Studio 2010, reduzindo consideravelmente o trabalho manual de se mapear as classes para as tabelas e

auxiliando na geração de consultas, desde as mais simples até as mais complexas.

1.2.2. Específicos

- Oferecer uma interface amigável e ao mesmo tempo poderosa para auxiliar o Gendal;
- Realizar comparações de mapeamento com e sem a ferramenta;
- Realizar testes de usabilidade para a geração de consultas complexas;
- Apresentar as diferenças de mapeamentos entre classes e tabelas, auxiliando o usuário a atualizar seus mapeamentos.

1.3. Metodologia

Para alcançar os objetivos descritos, este trabalho irá seguir as etapas abaixo elencadas:

- Estudo sobre ORMs em contexto geral;
- Estudo sobre o Gendal;
- Estudo sobre Plug-ins para o Visual Studio 2010
- Implementação do Plug-in
- Estudo de Caso

Na primeira etapa é mostrado um estudo geral de ORMs e como esse tipo de biblioteca deve funcionar.

Na segunda etapa é mostrado o funcionamento básico do ORM Gendal, passando pela conexão e mapeamento e indo até a geração de consultas com alguns extras no final.

Na terceira etapa é mostrada uma breve explicação do funcionamento de plug-ins no Visual Studio 2010.

A quarta etapa refere-se à implementação da ferramenta, que consiste na geração de código para mapear classes e gerar consultas. Além disso, a ferramenta deverá poder se conectar a um banco de dados para comparar classes já mapeadas para serem atualizadas.

A quinta e última etapa irá mostrar um Estudo de Caso, incluindo um trabalho acadêmico feito em cima do Gendal e utilizando a ferramenta.

1.4. Estrutura do Trabalho

Este trabalho está dividido da seguinte forma:

- Capítulo 1: Introdução - Apresentação geral, mostrando melhorias possíveis para um ORM, tecnologias envolvidas, comparações e a solução para a melhoria;
- Capítulo 2: Fundamentação teórica – Explicação sobre ORMs, algumas comparações e a visão da criação do Gendal.
- Capítulo 3: O ORM Gendal – Estudo sobre as funcionalidades básicas do Gendal e a motivação para se criar uma ferramenta para ele.

- Capítulo 4: DBClassMapper, a ferramenta de auxilio – Implementação da ferramenta de auxilio do Gendal. Ele contém uma breve explicação de plug-ins para o Visual Studio 2010.
- Capítulo 5: Construtor de Consultas – Mostra como utilizar a ferramenta para o auxílio na criação de consultas.
- Capítulo 6: Estudo de Caso – Este capítulo apresenta um trabalho acadêmico que utilizou o Gendal e o DBClassMapper para a sua implementação.
- Capítulo 7: Conclusão – Apresenta a conclusão sobre este trabalho e futuros possíveis trabalhos que podem ser desenvolvidos.

2. Fundamentação teórica

2.1. Definição de um ORM

Este capítulo apresenta uma breve explicação sobre o que é ORM, como ele deveria realmente ser e funcionar, bem como uma visão geral das principais bibliotecas para .Net.

2.1.1. ORM

A sigla ORM significa *Object Relational-Mapping*, ou Mapeamento Objeto-relacional em português. Um ORM basicamente mapeia objetos de um programa para tabelas relacionais de um banco de dados e vice-versa.

Na prática, um ORM deve ser uma biblioteca que auxilia os desenvolvedores na persistência de seus objetos definidos por classes. Este auxílio deve ser suficiente para evitar o conhecimento profundo de dialetos SQL e o tempo gasto com o ajuste de códigos de *C.R.U.D (Create, Read, Update e Delete)*. Como visto a seguir, nem todas as ferramentas de ORM seguem esta prática.

Uma definição mais clara e objetiva de ORM é a encontrada na Wikipédia

Mapeamento objeto-relacional (ou ORM, do inglês: Object-relational mapping) é uma técnica de desenvolvimento utilizada para reduzir a impedância da programação orientada a objetos utilizando bancos de dados relacionais. As tabelas do banco de dados são representadas através de classes e os registros de cada tabela são representados como instâncias das classes correspondentes.

Com esta técnica, o programador não precisa se preocupar com os comandos em linguagem SQL; ele irá usar uma interface de programação simples que faz todo o trabalho de persistência.

Não é necessária uma correspondência direta entre as tabelas de dados e as classes do programa. A relação entre as tabelas onde originam os dados e o objeto que os disponibiliza é configurada pelo programador, isolando o código do programa das alterações à organização dos dados nas tabelas do banco de dados.

A forma como este mapeamento é configurado depende da ferramenta que estamos a usar. Como exemplo, o programador que use Hibernate na linguagem Java pode usar arquivos XML ou o sistema de anotações que a linguagem providencia.

2.1.2. Funcionamento

Uma solução ORM pode ser encarada como um sistema com classes de objetos (como pessoa, produto, setor, e etc.). Nesse sistema as funcionalidades e regras estão bem definidas, basta agora persistir os dados que os usuários irão inserir no sistema. Para isso será necessário para cada classe criar funções e procedimentos para salvar, ler, atualizar e apagar estes dados de um banco de dados. Esta tarefa será quase tão grande quanto foi a de programar as funcionalidades e regras do sistema, mas com uma diferença, tudo que é feito para persistir os dados é basicamente trabalho maçante e repetitivo, pois as funções *C.R.U.D.* (salvar, ler, atualizar e apagar) são muito similares para qualquer classe. Um ORM simplesmente tem que resolver este problema sem

que os desenvolvedores conheçam SQL, e para ser mais específico, toda a utilização do ORM deverá ser o mais orientado a objetos possível, pois assim nenhum desenvolvedor iria perder tempo aprendendo novas maneiras de programar.

2.2. ORM existentes para .Net e Comparações

O Gendal foi desenvolvido na plataforma .Net da Microsoft e por este mesmo motivo iremos compará-lo a outros ORMs voltados para esta mesma plataforma. Algumas comparações feitas aqui somente serão bem entendidas após a leitura dos capítulos que se sucedem, pois algumas comparações tratam de funcionalidades ainda não bem explicadas do Gendal.

2.2.1. Microsoft Entity Framework

Principal e único ORM da Microsoft, esta complicada biblioteca já teve mais mudanças drásticas do que o primeiro SGBD suportado por ela, o SQL Server. O EF [3], que é como será chamado daqui em diante, começou muito complexo e ao mesmo tempo simples. Bastava gerar um diagrama de classes no Visual Studio (IDE utilizada para o .Net, SQL Server e EF) que a partir deste diagrama você já conseguiria gerar o esquema de dados para o SQL Server e ter seus objetos. O problema é que apenas um arquivo era gerado para todas as suas classes e esse mesmo arquivo já continha todo o mapeamento entre as suas classes e o SQL Server. Como se isto já não deixasse o entendimento complicado o suficiente, na hora de consultar dados era utilizado uma linguagem chamada de *Linq to SQL* [6, 7], derivada do *Linq to Object*. Esta linguagem visa

basicamente consultar dados de uma determinada lista no caso de *Linq to Object*, e em uma determinada tabela no caso de *Linq to SQL*, mas que para a sua programação era totalmente baseada em SQL. A definição básica de ORM nos remete para abstrair o SQL.

A Microsoft decidiu então lançar uma nova versão do EF totalmente diferente da anterior, mas que mantivesse total compatibilidade com as versões anteriores. A nova versão do EF veio resolver os problemas que todos tinham com a versão anterior. Neste novo formato o EF agora deixa as suas classes serem independentes e em qualquer arquivo, ou seja, mais fácil para manutenção e agora é legível. Para retirar o *Linq to SQL*, a Microsoft decidiu utilizar expressões lambda, as quais já eram utilizadas em objetos do .Net. O problema é que estas expressões lambda são mais complicadas de se entender do que o próprio *Linq to SQL*. Não é mais necessário saber SQL para consultar dados, mas é necessário saber criar uma expressão lambda para cada consulta diferenciada. A arquitetura do EF tem muitas camadas e requer conhecimento de todas elas para o seu funcionamento (Figura 1). Este é o ponto forte do Gendal em relação ao EF, não é necessário saber nada especial para se gerar uma consulta, basta apenas saber orientação a objetos como será visto no capitulo que explica sobre consultas no Gendal.

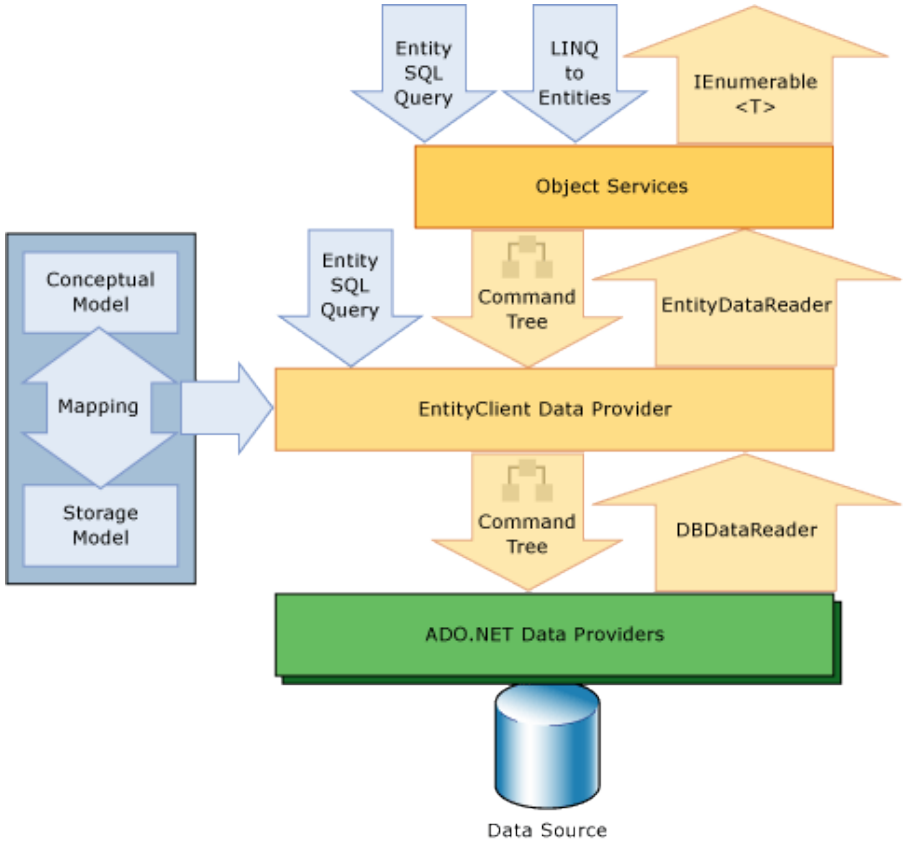


Figura 1. Arquitetura do Entity Framework

2.2.2. O NHibernate – Versão para .Net 2.0 do Hibernate

O Hibernate é um ORM amplamente utilizado em Java e foi criado a partir dele o NHibernate [11], uma versão para o .Net. Um dos problemas que já faz o Gendal ser necessário ao .Net é que o NHibernate não evoluiu a partir do .Net 2.0, o que deixa certa insegurança em utilizar algo que não se sabe se terá suporte futuro. Este ORM tem uma flexibilidade grande quanto a sua utilização e por este mesmo motivo requer grande atenção dos desenvolvedores para não escreverem códigos complicados.

O Gendal, por ser mais simples, objetivo e intuitivo não requer grandes cuidados, pois se não contarmos com as varias possibilidades de consultas que geram o mesmo resultado, ou seja, obter os mesmo resultados através de diferentes consultas, existe quase nenhuma ambiguidade para a geração de códigos do Gendal, tornando-o extremamente fácil de dar manutenção e entender o que está escrito.

A figura 2 abaixo mostra a estrutura do Hibernate. Esta imagem foi retirada do site oficial mais atualizado do Hibernate e mostra ao lado direito claramente a descontinuação com o .Net e ao centro o fortalecimento com o Java. Nota-se que só existem ferramentas auxiliares para o Hibernate do Java, tornando as ferramentas do Gendal para .Net extremamente poderosas no conceito de ORM para o .Net.

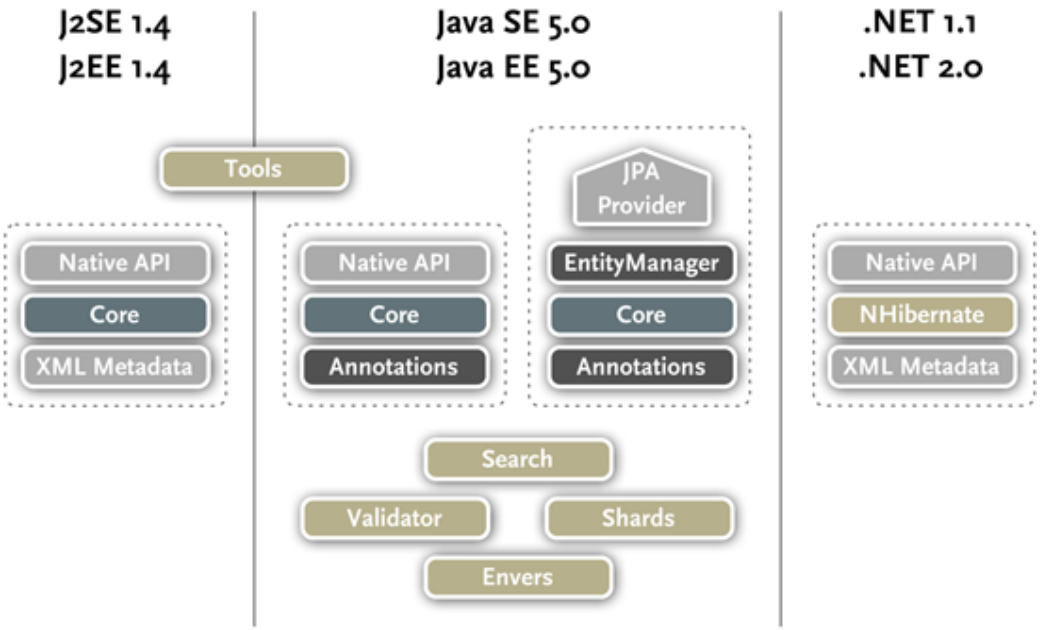


Figura 2. Estrutura do Hibernate

3. O ORM Gendal

3.1. Proposta Geral

Neste capitulo é mostrado o surgimento do ORM Gendal, suas facilidades e ao final é explicado como funciona o seu mapeamento. Apenas para ilustrar, a figura 3 mostra a estrutura do Gendal em um sistema que utiliza interfaces gráficas e regras de negócio para a sua implementação. Os círculos em azul representam classes e as suas interações quando são estão ligadas. O hexágono alaranjado representa uma interface gráfica que é gerada a partir de uma classe. O quadrado azul representa uma base de dados.

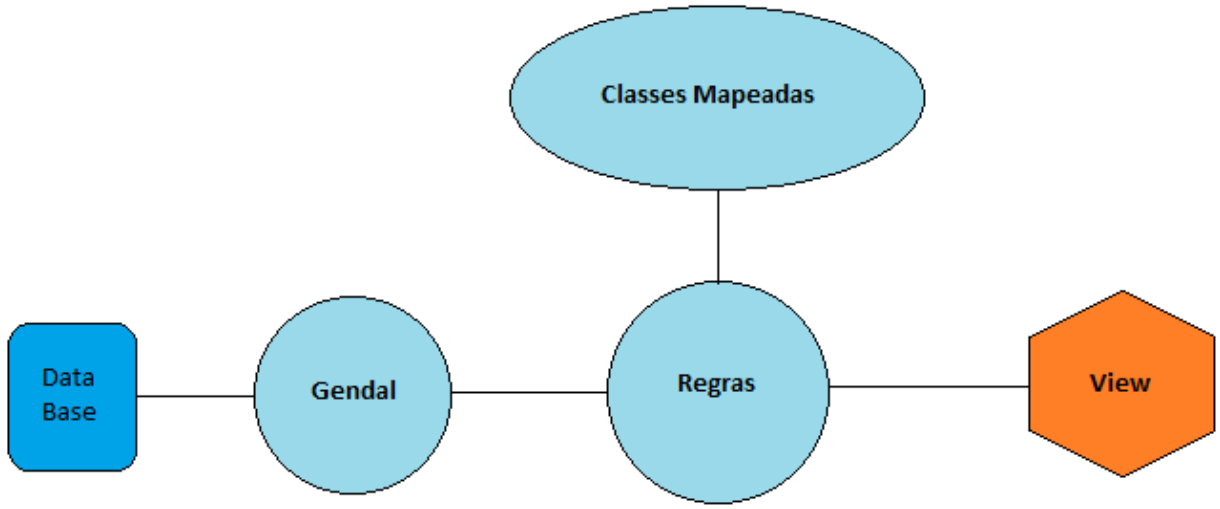


Figura 3. Estrutura do Gendal

3.1.1. Compatibilidade com o SGBD Caché

O Gendal surgiu da necessidade de um ORM para o banco de dados Caché da Intersystems. Mas mesmo a necessidade sendo especificamente para o Caché, não se queria ficar preso ao mesmo, ou seja, o ORM deveria ser multibanco. Foi a partir desta ideia que o Gendal começou a ser concebido. Sem

ter muito conhecimento de como outros ORM funcionavam, o Gendal foi criado sem vícios e a partir da definição básica do que é um ORM

O banco de dados Caché da Intersystems é um banco internamente relacional, mas que tem uma biblioteca orientada a objetos por fora para a sua utilização. Isto por si só já é como se existisse um ORM específico do Caché, mas que quando utilizado deixa o desenvolvedor totalmente preso ao banco, pois esta funcionalidade é exclusiva deste banco de dados. Este é o principal motivo de não ter sido utilizado a própria orientação a objetos do Caché.

3.1.2. Código orientado a objetos

Toda a utilização do Gendal é orientada a objetos. Não é necessário aprender SQL, expressões lambda, ou qualquer outra tecnologia, linguagem ou paradigma.

Um ORM por definição deveria abstrair o máximo possível de um banco de dados relacional e tentar facilitar a vida dos desenvolvedores. Quando são criadas novas linguagens, mudados paradigmas e até reaproveitado conceitos como o SQL, ao invés de abstrair o banco relacional, estão sendo criadas novas dificuldades ou até reutilizando dificuldades já encontradas antes. Toda a utilização do Gendal foi concebida para o paradigma da Orientação a Objeto e nada dele foge disto, ao contrário do EF, por exemplo.

3.1.3. Mapeamento por Anotações

Existem duas principais maneiras para se mapear objetos de um sistema para tabelas relacionais. Uma delas é através de arquivos (XML, por exemplo)

onde nestes arquivos estão contidos, além dos dados de mapeamento, alguns dados duplicados das classes para se saber de qual classe e quais propriedades cada mapeamento pertence. A outra maneira, que é utilizada pelo Gendal, é através de anotações nas próprias classes de objetos, ou seja, cada propriedade, além da própria classe, recebe anotações (no caso do Gendal atributos). Estas anotações servem para mapear os objetos para as tabelas relacionais e utiliza-se o mínimo possível de anotações, mas sempre o suficiente, para se obter o melhor mapeamento possível. Isto simplifica o entendimento dos mapeamentos, pois eles estão presentes nos seus objetos.

A utilização de mapeamento por arquivos não se demonstrou mais expressiva do que a de anotações. Ela simplesmente é uma maneira diferenciada de mapeamento que quando combinada com a de anotações pode atrapalhar a legibilidade do código fonte do programa.

3.2. Mapeamento no Gendal

Como já explicado anteriormente, o Gendal utiliza a forma de anotações para mapear as classes. Nesta seção é explicado como são os mapeamentos do Gendal em duas seções. O primeiro irá mostrar os atributos de mapeamentos utilizados do *namespace Data Annotations* do próprio .Net e a segunda seção mostra os atributos criados especificamente para o Gendal.

3.2.1. Atributos com Data Annotations

Data Annotations é um *namespace* do .Net que contém classes de atributos para anotações genéricas. Estas anotações têm uma variedade de

utilizações no .Net, mas aqui será focada a sua utilização somente como atributos para mapear as classes objeto para tabelas relacionais.

Para uma melhor explicação, os atributos criados para os mapeamentos são simplesmente classes que herdam da classe *Attribute* do .Net e tem propriedades e nomes significativos à sua utilidade. A utilização de atributos pré-definidos pelo *Data Annotations* apenas simplificou o trabalho da criação de alguns atributos, mas não teve nenhum ganho no ORM além deste. Este *namespace* deveria tentar cobrir várias anotações de várias coisas não citadas aqui, mas pela falta de alguns atributos é que foram criados outros atributos que serão descritos na próxima seção.

Tentando simplificar a explicação geral do mapeamento imagine uma classe com o nome de cliente que contém três propriedades, id, nome, idade. Desta classe surgiram objetos que serão do tipo pessoa. Para persistir esse objeto será então criado um banco de dados com a tabela SQLUser.Cliente com os campos id, nome e idade. O mapeamento já parece um pouco óbvio. Um objeto da classe pessoa será uma linha da tabela SQLUser.Cliente e cada propriedade da classe será vinculada com cada campo correspondente da tabela. A figura 4 exemplifica como ficaria este mapeamento no Visual Studio 2010 utilizando a linguagem Visual Basic.

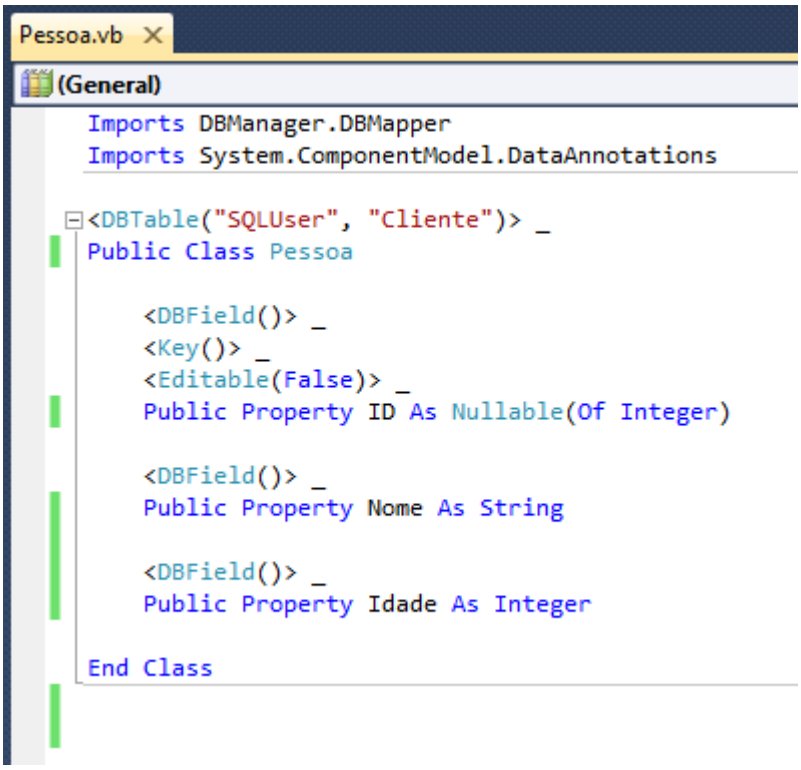


Figura 4. Exemplo de mapeamento no Gendal

Com este exemplo tem-se uma boa ideia de como os mapeamentos funcionam e com isso abaixo segue uma lista com todos os atributos do Data Annotations e para que eles são utilizados no Gendal.

Key: Indica se a propriedade é uma chave ou não, podendo várias propriedades ser chaves (indicando chave múltipla). Para chaves alternativas é utilizado o DBAlternateKey, que será explicado nos atributos específicos do Gendal.

Editable: Indica se a propriedade é editável ou não. Em um banco de dados normalmente é visto como o inverso do *AllowNull*.

Required: Indica se a propriedade é requerida ou não, ou seja, se é obrigatório ter algum dado diferente de vazio preenchido. Caso uma propriedade

Required esteja vazia o Gendal irá retornar um erro indicando que a propriedade em questão deveria ser preenchida e veio vazia.

Association: Indica uma associação entre classes, mais conhecido nos bancos de dados como um relacionamento entre tabelas. Existem três parâmetros obrigatórios: Nome da Relação, Nome das Chaves nesta Classe, Nome das Chaves da Classe relacionada. O Gendal utiliza este atributo para gerar os joins entre classes podendo ser utilizado filtros e ordenações mais avançadas.

TimeStamp: Utilizado para indicar se uma propriedade deve ser entendida como Data e Hora precisas. Muito utilizado para comparações e ordenação de Datas e Horas.

ConcurrencyCheck: Esta propriedade é utilizada para verificar concorrência entre atualizações de dados. Quando for requisitado ao Gendal salvar um objeto no banco, se este objeto tiver propriedades ConcurrencyCheck ele irá analisar para ver se os valores destas propriedades continuam os mesmos desde o momento em que o usuário requisitou o objeto. Caso forem diferentes isto indica que alguém ou algo mudou este objeto no meio do tempo de sua utilização, com isso o Gendal retorna um objeto contendo as propriedades que estão no banco. Isto é muito útil pois assim o desenvolvedor pode escolher o que fazer com o que ele tem em mãos. Funciona como um controle de versão de cada objeto e é extremamente utilizado em sistemas que precisam de controle de concorrência sem bloquear os objetos, pois apenas quando eles estão diferentes é que é levantado o questionamento de o que fazer

com as diferenças. Este é um atributo bem interessante, mas aprofundar mais nele fugiria do foco.

3.2.2. Atributos do Gendal

Como já explicado no capitulo anterior, estes atributos foram criados especificamente para o Gendal, já que o *Data Annotations* não conseguiu cobrir toda a necessidade do Gendal.

A utilização destes atributos é combinada com o uso do *Data Annotations* e por isso não há necessidade para maiores explicações. Abaixo segue a lista com os atributos específicos do Gendal e para que são utilizados.

DBTable: Utilizado como atributo de classe, este indica para qual tabela a classe está sendo mapeada.

DBField: Utilizado como atributo de propriedade, este indica para qual campo da tabela esta propriedade está sendo mapeada.

DBAlternateKey: Indica se a propriedade é uma chave alternativa, podendo existir várias chaves alternativas.

DBStream: Indica se a propriedade é um *Stream* (mais conhecido como *Blob* ou *Byte Array*). Este atributo surgiu com a necessidade de buscar objetos sem carregar seus *streams*, por razões de desempenho. *Streams* de objetos não são carregados por padrão, são opcionais

DBDateTime: O .Net não tem um diferenciação entre variáveis Date e Time muito boa, sempre misturando as duas e fazendo uma só chamada de DateTime. Por este motivo este atributo foi criado, com ele é possível separar campos específicos de Datas e Horas.

3.3. Conexão com SGBDs no Gendal

O Gendal suporta uma lista de bancos aceitos por conexão nativa, onde a conexão é feita por uma biblioteca disponibilizada pelo próprio fabricante do SGBD. Além desta lista é possível se conectar com qualquer outro banco através de conexões ODBC que são suportadas pelo Gendal.

Segue uma lista com todos os bancos de dados aceitos nativamente pelo Gendal, ou seja, que utilizam bibliotecas próprias dos fabricantes:

- Caché 4.1.6
- Caché 2010
- Caché 2012
- SQL Server 2005
- SQL Server 2008

Para conectar em um banco de dados que ainda não esteja na lista, basta utilizar a conexão por ODBC. Se houver a necessidade de adicionar um novo banco nativo ao Gendal, basta implementar nas classes de *Providers* importando a biblioteca do fabricante lá dentro. Esta parte não será aprofundada por ser muito extensa e não ser o foco principal deste trabalho.

3.3.1. O DBFactory

O DBFactory é a classe utilizada pelo Gendal para se obter uma conexão com o SGBD desejado. A sua instanciação requer apenas o IP ou nome do servidor onde se encontra a base de dados e a string de conexão necessária que pode variar dependendo do banco utilizado. Caso for utilizar conexão ODBC basta colocar ‘DNS=[Nome da conexão ODBC]’. Para saber o que é e como

criar uma conexão ODBC basta ler o apêndice que vem em anexo deste trabalho. Conexões ODBCs existem unicamente para cobrir os bancos que não tem suas bibliotecas de acesso implementadas no Gendal.

A figura 5 mostra um exemplo de criação de um objeto DBFactory:

Figura 5. Utilização do DBFactory

3.3.2. A Classe Gendal

O DBFactory é utilizado como principal parâmetro de criação do Gendal. Tendo uma instância de um DBFactory basta instanciar um Gendal passando para ele o respectivo DBFactory. É a partir disto que o Gendal saberá como se conectar a um SGBD.

Todos os comandos para salvar, deletar, buscar e fazer o que bem entender com os seus objetos no banco de dados estão presentes na classe Gendal. Esta classe é quem deu o nome deste ORM pois ela é o coração do ORM. Seu Significado é simples, mas quer dizer tudo: *Gen: Generic; d: data; a: access; l: library*, ou seja, Biblioteca Genérica de Acesso a Dados.

A figura 6 exemplifica a criação de um objeto Gendal:

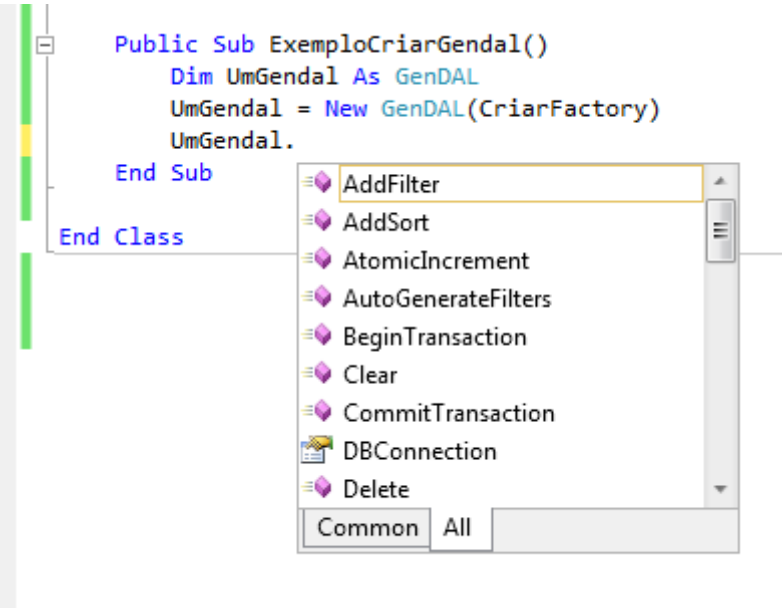


Figura 6. Utilização do Gendal

3.4. Salvando e Buscando Objetos Individuais

Será tratado aqui da funcionalidade básica de salvar e buscar objetos individuais, ou seja, não será tratado como trabalhar com listas de objetos. As listas serão tratadas na próxima seção onde serão abordados filtros, ordenação e outros.

3.4.1. Criando um Objeto e Salvando

Todo programa orientado a objetos requer as classes que definem os objetos. Como o Gendal é orientado a objetos também existirá a necessidade destas classes. Nesta parte será suposto que existe um código que já tenha suas classes definidas e mapeadas de acordo com o capítulo de mapeamentos visto anteriormente.

Com os pré-requisitos prontos será criado então no programa um objeto e populado suas propriedades com os dados necessários para que ele possa ser

salvo (propriedades onde o campo original do banco não pode ser *null* devem ser preenchidas). Com este objeto em mãos e com um Gendal já instanciado com um DBFactory, basta agora pedir que o Gendal salve este objeto no banco de dados. A figura 7 mostra exatamente isso:

```
Public Function CriarESalvarUmaPessoa() As Boolean
    Dim DBLink As New GenDAL(CriarFactory)
    Dim UmaPessoa As New Pessoa

    UmaPessoa.Nome = "Diego Magno"
    UmaPessoa.Idade = 25

    Return DBLink.Save(UmaPessoa)
End Function
```

Figura 7. Salvar um objeto no Gendal

O *Save* do Gendal serve tanto para atualizar como criar objetos. Um *boolean* é retornado pela função para avisar se o objeto foi salvo com sucesso ou não. Caso uma propriedade obrigatória não fosse preenchida, um erro seria levantado e não chegaria a retornar *False*. Esta função somente irá retornar *False* quando por um motivo desconhecido o banco não aceitar o *Insert* ou *Update* e nenhum erro for levantado. Nestes casos será necessária a análise do desenvolvedor para saber o que está acontecendo no banco de dados.

3.4.2. Buscando um Objeto

Para buscar um objeto do banco é muito simples, basta instanciar um objeto da classe desejada e preencher as chaves destes objetos. Com as chaves preenchidas basta em uma instância do Gendal chamar a função Open([Objeto]). Imediatamente o Gendal busca o objeto do banco e preenche com todas as informações que estão mapeadas.

A figura 8 ilustra um objeto do tipo Pessoa sendo instanciado, tendo a sua chave preenchida e sendo buscado pelo Gendal. Existem maneiras mais apropriadas e genéricas para se fazer o mesmo, todos os exemplos utilizados aqui estão sendo o mais didático e simples possível.

```
Public Function BuscarUmaPessoa() As Pessoa
    Dim DBLink As New GenDAL(CriarFactory)
    Dim UmaPessoa As New Pessoa

    UmaPessoa.ID = 1

    If Not DBLink.Open(UmaPessoa) Then
        Throw New Exception("Erro ao tentar buscar a pessoa de ID = 1")
    End If

    Return UmaPessoa
End Function
```

Figura 8. Buscar um objeto no Gendal

3.4.3. Apagando um Objeto

Remover um objeto através do Gendal é muito fácil, por isso deve-se utilizar esta chamada com cuidado. Basta chamar a função Delete([Objeto]).

A função aceita tanto objetos que foram trazidos pelo próprio Gendal quanto objetos que estejam apenas com os campos que são chaves preenchidos. Após executado é retornado um valor do tipo *Integer* contendo o número de objetos afetados (ou pode-se entender também como o número de linhas afetadas no banco de dados).

A figura 9 abaixo mostra como pode ser feito para apagar um objeto utilizando a maneira de preencher as chaves. Para utilizar a técnica de apagar um objeto trazido pelo Gendal basta buscar o objeto como visto no capítulo anterior e aplicar a mesma função neste objeto.

```
Public Function ApagarUmaPessoa() As Integer
    Dim DBLink As New GenDAL(CriarFactory)
    Dim UmaPessoa As New Pessoa

    UmaPessoa.ID = 1

    Return DBLink.Delete(UmaPessoa)
End Function
```

Figura 9. Apagar um objeto

3.5. Trabalhando com Listas de Objetos

O Gendal utiliza a classe de lista do .Net como base para gerenciar as suas listas. Essa classe de lista é amplamente utilizada para tratar usualmente as listas de objetos em qualquer programa orientado a objetos no .Net.

Com essa classe de listas é possível ordenar, filtrar, adicionar, remover dentre outras coisas os objetos contidos na lista. Para isso é necessário utilizar *Linq to Object* ou expressões lambda. Esta lista não será aprofundada neste trabalho, principalmente por ser utilizada a parte básica desta lista para exemplificação e por ser uma classe muito extensa e que foge ao foco do trabalho.

3.5.1. Buscando Listas de Objetos

Trazer listas de objetos de um banco de dados utilizando o Gendal é algo muito prático e natural. Não se precisa prática nem habilidade. Basta saber que tipo de objetos fazem-se necessários trazer. Será apresentado neste capítulo apenas a maneira mais simples de se trazer listas de objetos, o que não é algo indicado de se fazer no dia-a-dia, pois isto significa trazer todos os objetos, ou seja, todas as linhas de uma tabela. A maneira mais correta de se trazer listas

será mostrada no capítulo seguinte. Veja este capítulo apenas como uma básica introdução a listas.

Da mesma forma que das outras execuções anteriores, basta instanciar o Gendal com um DBFactory e pedir para ele lhe retornar uma lista com a função List(of [Tipo]), onde [Tipo] é o tipo de objeto que se deseja listar. Essa execução irá buscar todos os objetos do banco de dados do tipo solicitado, se for testar desta maneira é aconselhado usar em uma tabela com poucas linhas.

O Gendal gera SQLs internamente para executar tais consultas. Para controle de o que é de quem nas filtragens e ordenações é utilizado um *Alias* em cada tabela vinculada a uma classe. Por padrão é utilizado o Schema junto da tabela, como por exemplo SQLUser.Pessoa. É possível personalizar estes *Alias* passando uma string no primeiro parâmetro da chamada do List(of [Tipo])([Alias]). A figura 10 ilustra um método que faz isto.

```
Public Function ListarTodasAsPessoas() As List(Of Pessoa)
    Dim DBLink As New GenDAL(CriarFactory)
    Return DBLink.List(Of Pessoa)("P")
End Function
```

Figura 10. Buscar listas no Gendal

3.5.2. Filtros para Buscar Listas

A maneira mais eficiente para se trazer listas é adicionar sempre pelo menos um filtro. O Gendal tem uma lista de filtros que é aberta para total edição, podendo-se adicionar, remover e alterar filtros à vontade. A cada execução de uma consulta esta lista é esvaziada supondo que os filtros adicionados eram apenas para aquela consulta. Para a utilização de filtros é aconselhado a utilização de *Alias* para melhor identificar qual filtro será de qual objeto. Isto será

muito importante quando chegarmos à utilização de joins e consultas mais complexas.

Os filtros podem ser adicionados diretamente na lista de filtros ou pela chamada AddFilter onde existem diversas configurações de parâmetros para se adequar de acordo com cada necessidade de filtro, inclusive *Between* e etc.

A seguir temos uma imagem (Figura 11) exemplificando uma listagem com filtros e já utilizando *Alias* como aconselhado.Neste exemplo são mostrados filtros sendo adicionados diretamente na lista de filtros e também pela chamada de AddFilter. A cada execução os filtros são limpos, não sendo reaproveitados.

```
Public Function ListaFiltrada() As List(Of Pessoa)
    Dim DBLink As New GenDAL(CriarFactory)
    Dim UmFiltro As New Filter

    UmFiltro.Field = "P.Idade"
    UmFiltro.OperatorType = OperatorType.Equal
    UmFiltro.Value = 25
    DBLink.Filters.Add(UmFiltro)

    DBLink.AddFilter("P.Nome", OperatorType.LikeStartsWith, "Diego")

    Return DBLink.List(Of Pessoa)("P")
End Function
```

Figura 11. Listas filtradas no Gendal

Como visto no exemplo para filtrar basta inserir qual campo será filtrado, o tipo de comparação (ou operador) e o valor que será comparado. Abaixo segue uma lista com os operadores aceitos e o que cada um deles faz:

Equal: Operador de igualdade. O valor deve ser igual.

Different: Operador de diferente. O valor deve ser diferente.

Between: Devem ser passados dois valores e os valores do banco devem estar dentre esses dois valores indicados.

GreaterEqualsThan: Operador maior ou igual. Os campos devem ter o valor maior ou igual ao valor passado.

GreaterThan: Operador maior do que. O valor dos campos devem ser maiores do que o valor passado.

LessEqualsThan: Operador menor ou igual. Os campos devem ter o valor menor ou igual ao valor passado.

LessThan: Operador menor do que. Os campos devem ter o valor menor do que o valor passado.

O operador Like é utilizado para comparações de strings (Varchar) e tem várias utilidades na busca de strings ou pedaços de strings. Este assunto não será aprofundado neste trabalho, basta entender que Like é um operador para Strings.

LikeEndsWith: Procura Strings que terminem com o valor passado indiferente do que há no inicio dela.

LikeHaving: Procura Strings que contenham em qualquer parte dela todas as strings passadas separadas por espaço. Ex.: Procuramos com LikeHaving na String ‘Eu adoro monografias sobre ORM’ usando de filtro a String ‘adoro sobre’. Esta busca irá retornar True, esta string contém o valor filtrado e a linha da tabela que a contém será retornada.

LikePure: Equivalente a utilizar o operador Equal com Strings.

LikeStartsEndsWith: Procura Strings que iniciem e terminem com o valor passado.

LikeStartsWith: Procura String que iniciem com o valor passado.

3.5.3. Ordenações

As ordenações são muito parecidas com os filtros, com a diferença que ao invés de ser uma lista de filtros é uma lista de ordenações, e ao invés de filtrar irá ordenar.

A ordenação segue todos os princípios da filtragem: a lista é limpa após a execução de uma consulta e o *Alias* é utilizado para melhor representar a propriedade de qual objeto que está sendo ordenada. A principal diferença entre os dois é que a lista de ordenação não é aberta para edição. Para adicionar uma ordem existe apenas a função AddSort.

A figura 12 ilustra um exemplo de como ordenar uma lista simples. Na seção de joins serão mostrados os filtros e ordenações com vários objetos. Caso ficar alguma dúvida sobre estes conceitos, a seção de joins poderá se tornar de difícil compreensão.

```
Public Function ListaOrdenanda() As List(Of Pessoa)
    Dim DBLink As New GenDAL(CriarFactory)

    DBLink.AddSort("P.Nome", OrderMode.Desc)

    Return DBLink.List(Of Pessoa)("P")
End Function
```

Figura 12. Ordenar no Gendal

O exemplo mostra apenas uma ordenação sendo adicionada, mas é possível adicionar quantas ordenações forem necessárias e em qualquer ordem.

3.5.4. Remover múltiplos Objetos

Boas práticas de programação sempre dizem que se pudermos evitar apagar um dado de uma tabela e apenas indicá-lo, através de uma *Flag,* que aquele dado já não é mais válido, devemos fazê-lo, evitando diversos problemas futuros. Seguindo esta prática é que este capítulo foi criado.

Sim, é possível apagar múltiplos dados no Gendal, mas é aconselhado, desde já, a utilizar isto com muito cuidado. Basta apenas utilizar a chamada DeleteFiltering(Of [Tipo]) passando filtros no Gendal igual como se faz na busca que os dados serão apagados utilizando os filtros. Esta seção poderia não existir e ter ficado junto da seção de filtros, mas a sua existência vem a frisar um alerta. Utilizar esta chamada somente quando realmente se faz necessário e quando tiver certeza do que está fazendo.

Esta função não tem *Alias* pelo fato de remoções não poderem ter joins nem nada que faça existir ambiguidade de campos no SQL.

A figura 13 ilustra um exemplo simples e perigoso.

```
Public Function DeletarFiltrando() As Boolean
    Dim DBLink As New GenDAL(CriarFactory)

    DBLink.AddFilter("ID", OperatorType.GreaterEqualsThan, 0)

    Return DBLink.DeleteFiltering(Of Pessoa)()
End Function
```

Figura 13. Remoção com filtro no Gendal

Este exemplo simplesmente apaga todos os dados da tabela SQLUser.Pessoa.

As próximas seções irão tratar de assuntos mais avançados. Caso até aqui algo não tenha ficado bem claro ou ainda há muita duvida no que foi

passado até agora, é aconselhado reler algumas seções e passar para as próximas quando estiver com as outras seções melhor entendidas, principalmente as que tratam de listas e filtros.

3.5.5. Joins Orientados a Objeto

Esta é a parte do Gendal que, quando bem compreendida, facilita em muito as consultas de dados. É neste ponto que o EF requer conhecimentos de SQL para utilizar Linq to SQL ou de expressões lambda para formular consultas baseadas em métodos. O Gendal simplesmente requer um bom entendimento da orientação a objeto que as consultas já saem naturalmente.

Revisando a seção das listas (seção 3.5), veremos que utilizamos um parâmetro para o *Alias* do objeto em destaque. Além desse *Alias* podemos também adicionar uma lista de Joins que irão se relacionar com o objeto principal da busca de listas. Além disso, cada um dos Joins pode conter internamente uma lista de Joins que irão se relacionar com o Join específico. Por exemplo, se tivermos um objeto NotaFiscal que se relaciona com Pessoa e um objeto Produto que se relaciona com NotaFiscal e quisermos buscar uma lista de pessoas que compraram um determinado produto? Partindo do básico, vamos pedir uma lista de pessoas:

```
DBLink.List(Of Pessoa)("P")
```

Esta chamada irá trazer uma lista com todas as pessoas, mas queremos apenas as pessoas que compraram um determinado produto. Para isso precisaremos indicar no Gendal que queremos Pessoas que compraram:

```
Dim JoinNF As New DBJoin(GetType(NotaFiscal), "N")
```

Esta declaração define um Join com NotaFiscal e diz que NotaFiscal irá utilizar o *Alias* 'N'. Mas esta declaração sozinha não nos diz nada ainda. Precisamos associar esta Join a outra classe de objetos. Como NotaFiscal tem uma relação com Pessoa, então iremos inserir o Join de NotaFiscal na Pessoa, como Pessoa é o objeto principal de busca, iremos então adicionar o Join diretamente na busca:

```
DBLink.List(Of Pessoa)("P", JoinNF)
```

Agora esta chamada irá trazer apenas pessoas que compraram, mas queremos pessoas que compraram um determinado produto, então temos que pegar Notas Fiscais que tem produtos vinculados a elas:

```
Dim JoinP As New DBJoin(GetType(Produto), "PD")
```

Esta declaração define um Join com Produto e diz que Produto irá utilizar o *Alias* 'PD'. Já sabemos que precisamos adicionar este join em outro objeto. Como Produto tem uma relação com NotaFiscal iremos então inserir o Join de Produto no Join de NotaFiscal:

```
JoinNF.AddJoins(JoinP)
```

Esta chamada adiciona no Join de NotaFiscal o Join de Produto, ou seja, Produto se relaciona com NotaFiscal (é possível fazer o contrário, como também é possível indicar se será um *Left Outer, Right Outer* ou outro tipo de join, mas para simplificarmos e não aprofundarmos mais do que já estamos, será tratado somente este exemplo para melhor entendimento).

Agora para pegarmos uma lista de Pessoas que compraram um determinado produto precisamos apenas filtrar o produto:

```
DBLink.AddFilter("PD.Codigo", OperatorType.Equal, 1)
```

Tudo que foi feito até agora foi orientado a objetos e de maneira natural. Nesta forma os joins ficam declarados e são indicados de traz para frente como os filtros e ordenações também foram. Tudo é declarado antes da chamada da lista, por isto há a inversão de declaração comparando com SQL. Para adicionarmos um Join dentro de outro Join, o primeiro já precisa estar declarado, e para adicionarmos todos os Joins na chamada da lista eles já precisam estar declarados, por isso ocorre esta inversão. O código final completo para esta consulta ficará assim de acordo com a imagem a Figura 14.

```
Public Function ListarPessoasQueCompraramProdutoX() As List(Of Pessoa)
    Dim DBLink As New GenDAL(CriarFactory)
    Dim JoinP As New DBJoin(GetType(Produto), "PD")
    Dim JoinNF As New DBJoin(GetType(NotaFiscal), "N")

    JoinNF.AddJoins(JoinP)
    DBLink.AddFilter("PD.Codigo", OperatorType.Equal, 1)

    Return DBLink.List(Of Pessoa)("P", JoinNF)
End Function
```

Figura 14. Joins no Gendal

O GetType([Tipo]) utilizado serve apenas para pegar a classe do objeto e é especifico da linguagem Visual Basic, o C#, que também utiliza .Net, não requer esta chamada, basta apenas passar o nome da classe.

Com estas poucas linhas fizemos totalmente orientado a objetos e sem necessitar conhecimento algum de SQL uma consulta que traz Pessoas que compraram Produtos com o código 1. A mesma consulta em SQL seria:

```
SELECT P.ID, P.Nome, P.Idade
FROM SQLUser.Pessoa P
        INNER JOIN SQLUser.NotaFiscal N ON N.Pessoa = P.ID
        INNER JOIN SQLUser.Produto PD ON PD.NotaFiscal = N.ID
WHERE PD.Codigo = 1
```

3.6. Funções especiais

Nesta seção serão descritas algumas funções especiais do Gendal que servem para facilitar ainda mais a sua utilização. As transações foram inseridas aqui somente para não confundir a explicação das seções anteriores, mas elas são tão simples de se utilizar como se não houvesse o Gendal.

Tanto a seção do Incremento Atômico quanto a seção de Paginação requerem a leitura da seção de Transações, então é aconselhável evitar pular tal seção antes de ler os outros.

3.6.1. Transações

Transação é algo extremamente simples e fácil de utilizar no Gendal, mas existem alguns cuidados que devem ser tomados para se evitar problemas que nem sempre aparecem em testes e vão ocorrer onde menos deveriam.

Para se iniciar uma transação basta executar a chamada de BeginTransaction. Dependendo do banco de dados o BeginTransaction irá iniciar imediatamente a transação ou irá iniciar apenas quando a primeira chamada que executa algum SQL pelo Gendal for executado. É bom prestar muita atenção quanto a isto, pois códigos executados dentro de uma transação que ainda não começou, dependendo do código, pode gerar falhas e erros. O Caché somente inicia a transação após executado algum SQL depois do BeginTransaction, já o SQL Server inicia imediatamente a transação independentemente de algo já ter sido executado ou não. Como nem todos os bancos de dados foram testados, é sempre aconselhável, antes de iniciar o desenvolvimento de um sistema com o Gendal, analisar este comportamento.

Para se efetivar uma transação basta executar *CommitTransaction*. Esta chamada imediatamente efetiva um *Commit* de tudo que foi feito na transação para o banco de dados persistir e finalizar a transação.

Para desfazer uma transação, basta chamar *RollBackTransaction*. É com este que devemos tomar muito cuidado. Se for iniciada uma transação com um objeto do Caché e por algum motivo o código não chegar a executar *Commit*, mas também não executar *RollBack* e este objeto do Caché for reaproveitado em outro lugar e neste lugar ocorrer um *RollBack*, tudo que foi feito na suposta primeira transação irá ser perdido também, pois na realidade houve apenas uma transação para as duas. Então muito cuidado na execução de *Commits* e *RollBacks*, sempre colocar códigos que peguem qualquer erro que ocorrer e sempre finalizar as transações corretamente. Esses problemas não são específicos do Gendal, mas continuam existindo nele também.

Para utilização de *Multi-Thread* no Gendal é importante utilizar um objeto de Gendal e um DBFactory para cada Thread. A utilização de mesma conexão para Multi-Threads pode causar sequências de envios totalmente erradas, mesmo que no seu código estejam sincronizadas. Transações em *Multi-Thread* também requerem isto, principalmente, pois dependendo do banco de dados as transações são executadas de maneiras diferentes e podem uma sobrepor a outra mesmo antes de seu código finalizar por completo. A figura 15 ilustra um exemplo de transação.

```
Public Function ListaFiltrada() As List(Of Pessoa)
    Dim DBLink As New GenDAL(CriarFactory)
    Dim UmFiltro As New Filter
    Dim Pessoas As List(Of Pessoa)

    Try
        'Inicia a transacao - dependendo do banco a transacao ainda nao existira
        DBLink.BeginTransaction()

        UmFiltro.Field = "P.Idade"
        UmFiltro.OperatorType = OperatorType.Equal
        UmFiltro.Value = 25
        DBLink.Filters.Add(UmFiltro)

        DBLink.AddFilter("P.Nome", OperatorType.LikeStartsWith, "Diego")

        'Aqui e o primeiro SQL executado pelo Gendal, dependendo do banco a transacao sera criada aqui
        Pessoas = DBLink.List(Of Pessoa)("P")

        'Commit na transacao
        DBLink.CommitTransaction()
    Catch ex As Exception
        'Caso algum erro ocorrer tem que garantir o RollBack para finalizar a transacao
        DBLink.RollbackTransaction()
        Throw New Exception("Erro inesperado, Inner Exception com maiores detalhes", ex)
    End Try

    Return Pessoas
End Function
```

Figura 15. Transação no Gendal

3.6.2. Incremento Atômico

O incremento atômico tem um nome bem sugestivo. Essa função executa um incremento em quantos campos forem requisitados com qualquer valor de maneira atômica, ou seja, essa chamada garante o incremento sem que ninguém mexa naqueles valores até seu código liberar. Para isso é necessária e obrigatória à utilização da Transação aqui.

Para um melhor entendimento vamos supor que queremos aumentar a quantidade de um determinado produto, ou seja, queremos incrementar o estoque do produto em mais 1, por exemplo. Se buscarmos a quantidade deste produto, alterarmos, e depois gravarmos existe a chance de outro usuário no meio do processo ter alterado este valor. Para resolver este problema é que surgiu o Incremento Atômico. Esta função é capaz de incrementar o valor

requisitado e só liberar para outro usuário fazê-lo também assim que seu código tiver liberado a transação.

O funcionamento é muito simples, basta iniciar uma transação e dentro desta transação executar a função AtomicIncrement do Gendal, passando qual o objeto que será atualizado, os campos e as quantidades a serem incrementadas (quantidades negativas decrementam). Exemplificado na figura 16.

```
Public Sub AumentarQuantidadeProduto()
    Dim DBLink As New GenDAL(CriarFactory)
    Dim MeuProduto As New Produto With {.ID = 1}

    Try
        DBLink.BeginTransaction()

        'Incremento da quantidade em +1
        DBLink.AtomicIncrement(Of Produto)(MeuProduto, New DBIncrementField("Quantidade", +1))

        'Busca do objeto produto, dentro desta transacao a quantidade nunca ira mais mudar ate
        'a sua transacao finalizar garantindo atomicidade
        DBLink.Open(MeuProduto)

        DBLink.CommitTransaction()
    Catch ex As Exception
        DBLink.RollbackTransaction()
        Throw ex
    End Try
End Sub
```

Figura 16. Incremento Atômico no Gendal

3.6.3. Paginação

A paginação é uma funcionalidade do Gendal muito dependente do banco de dados que está sendo utilizado, mas mesmo assim vale a pena dar uma breve explicação sobre ela, pois ela resolve muitos problemas de sistemas web.

Um dos maiores problemas de sistemas web é a utilização da banda de dados. Os usuários querem sempre listas e relatórios que muitas vezes são muito grandes para terem um bom desempenho no tráfego até o destino final. A paginação veio para resolver este problema. Quando for pedir uma lista de objetos no Gendal, basta indicar nele que será utilizada paginação e a quantidade de objetos por página. Com esta configuração feita, basta solicitar

uma página específica no Gendal, ou seja, mesmo que a sua lista seja muito grande, serão enviados ao usuário final somente as páginas requisitadas. Antes era tudo enviado para o usuário final e a paginação era toda feita na ponta final, agora com a paginação feita no próprio servidor, o tráfego de dados na rede fica extremamente reduzido. Os desempenhos em testes rápidos foram bem surpreendentes.

Infelizmente a paginação é totalmente dependente do SGBD. Para a paginação funcionar, o banco de dados tem que dar suporte a cache de consultas e poder retornar pedaços da consulta ao invés dela sempre inteira. A princípio, os bancos de dados Caché 2010 e o SQL Server 2008 dão suporte a este tipo de implementação e neles foi possível fazer alguns testes e aprovar a paginação.

4. Ferramenta DBClassMapper

O DBClassMapper é a ferramenta desenvolvida neste trabalho de conclusão e seu objetivo é auxiliar o ORM Gendal. Mesmo o Gendal auxiliando o desenvolvimento de sistemas que necessitem persistência de dados, ele também necessita de um auxílio para melhorar a sua usabilidade. O DBClassMapper fornece um suporte ao Gendal totalmente integrado a IDE Visual Studio 2010. A princípio, a ferramenta gera códigos apenas para o Visual Basic, porém, com alguns ajustes é possível gerar códigos para outras linguagens, como o C#.

A tela inicial do DBClassMapper, mostrada na figura 17, é bem simples e tem botões para cada finalidade específica da ferramenta. Cada seção que se sucede explica o seu funcionamento.

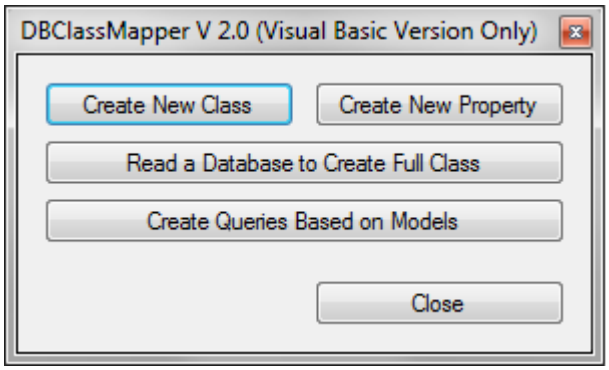


Figura 17. Janela principal do DBClassMapper

4.1. Mapeador Manual

O DBClassMapper tem telas para auxiliar a criação manual de classes e propriedades e a criação automatizada das mesmas. Nesta seção são mostradas as telas que auxiliam manualmente a criação de mapeamentos de classes e propriedades.

4.1.1. Criando um mapeamento de classe

Para criar uma nova classe no projeto já mapeada utiliza-se a tela *New Class*, acessada pelo botão *Create New Class*. Nesta tela (figura 18) existem três campos a serem preenchidos, o *Class Name*, *Schema* e *Table*. O *Class Name* é o nome da classe que será criada, o *Schema* é o nome do esquema do banco de dados para o qual esta classe será mapeada e o *Table* é o nome da tabela do banco de dados ao qual também será mapeada. Apenas o preenchimento do campo *Class Name* é obrigatório. Os outros, se ficarem em branco, são interpretados como a tabela tendo o mesmo nome da classe e o esquema sendo o esquema padrão do banco de dados.

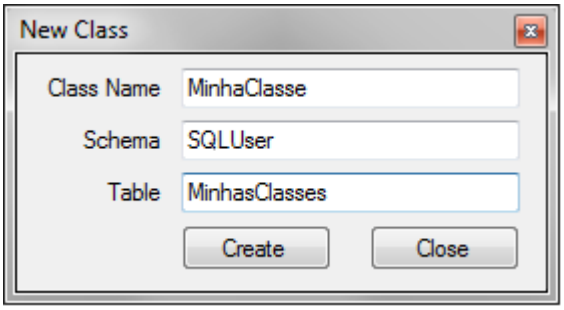


Figura 18. Tela New Class do DBClassMapper

4.1.2. Criando e Mapeando Propriedades

Uma propriedade é criada e mapeada em uma classe através da tela *New Property* (figura 19) e chamado pelo botão *Create New Property*. Nesta tela existem diversas especificações para criar uma nova propriedade e já mapeá-la. Apenas o tipo de mapeamento (*Mapper Type*), o nome da propriedade e o tipo de dado são obrigatórios.

O mapper type é composto por sete tipos de mapeamento:

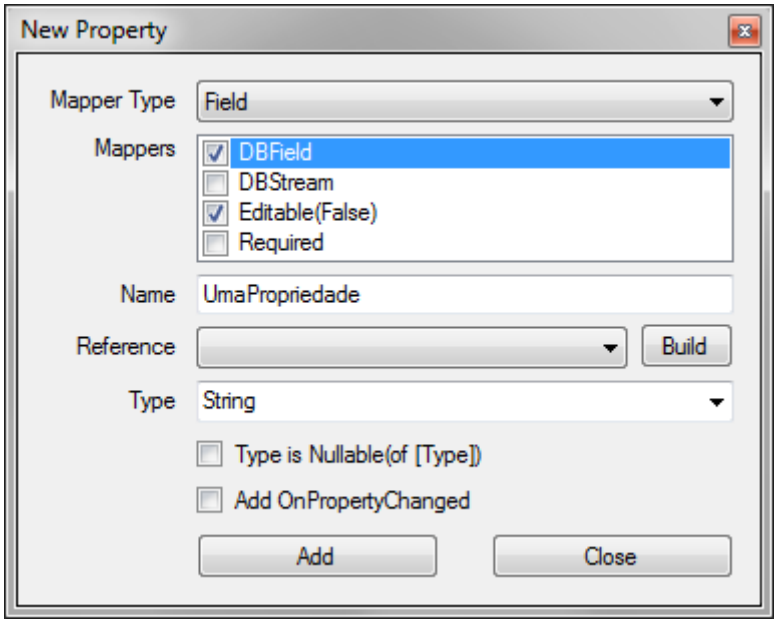


Figura 19. Tela New Property do DBClassMapper

None: Este tipo de mapeamento indica que a propriedade não será mapeada. Neste caso, a ferramenta irá criar a propriedade sem nenhum mapeamento. Ele serve apenas como um auxílio na criação de propriedades que não serão persistidas.

Field: Indica que a propriedade é mapeada para um campo do banco de dados. No Field é possível indicar o nome do campo da tabela diferente do nome da propriedade e a ferramenta sempre irá criar a propriedade supondo que o nome dela será o mesmo do campo da tabela. Isto facilita e agiliza o processo de criação. Caso haja a necessidade de alterar, basta no código indicar o nome no atributo Field.

Key: Este indica que o campo será uma chave primária. Vários campos com este atributo indicam chave múltipla para o Gendal.

Alternate Key: Propriedades deste tipo de mapeamento indicam que a propriedade é uma chave alternativa. Funciona como o Key e pode ser utilizado em Joins quando especificado.

Foreign Key: Indica um mapeamento de chave estrangeira. Usualmente uma chave estrangeira de um banco de dados significa um apontamento de um objeto de uma classe do sistema.

Date Time: Indica que a propriedade é um capo de data, hora ou data e hora. Existe basicamente para tratar problemas de data e hora que variam de SGBD para SGBD.

Time Stamp: Indica que a propriedade será utilizada para validar acesso a uma linha da tabela. Quando algum sistema executando efetiva um Open em um objeto, ou seja, lê uma linha do banco e preenche o objeto, esta propriedade é comparada no momento de dar um Save (Update no banco). Caso o valor esteja diferente, isso significa que alguém ou alguma coisa alterou o objeto enquanto ele estava sendo manipulado. Caso esteja igual, este valor é apenas alterado, dependendo do seu tipo, e atualizado na base de dados. No caso de diferente, fica a critério do desenvolvedor o que fazer com os dados. Ele pode sobrescrever, cancelar a atualização, fazer merge ou o que o desenvolvedor achar melhor para o sistema.

Após indicar qual o tipo de mapeamento, irão surgir os mapeadores propriamente ditos para serem selecionados na caixa dos Mappers. Alguns mapeadores são obrigados a existir dependendo do tipo de mapeamento. Por isso, os mapeadores que não puderem ser deselecionados são obrigados a

existir. Os mapeadores que aparecem na caixa dependem necessariamente do tipo de mapeador. Cada mapeador é um atributo que será colocado na propriedade criada. No capitulo do Gendal existe a lista destes mapeadores já com a explicação do que cada um faz.

Após indicar o nome no campo Name, existe uma caixa de seleção com o nome de *Reference* e ao seu lado um botão com o nome de Build. Esta caixa traz todas as referências que o projeto em questão tem de outros projetos ou bibliotecas. Isto serve para preencher a caixa de tipos de dados (Type) com objetos de classes, caso o tipo de dado for um relacionamento, por exemplo. O botão Build existe apenas para uma melhor usabilidade, este botão compila o projeto todo novamente e isto serve para que novas classes criadas apareçam na caixa *Reference*. O projeto pode ser normalmente compilado pelo Visual Studio e irá aparecer também na caixa. O local onde o projeto é compilado é indiferente para a ferramenta.

A caixa de seleção *Type* tem os tipos de dados mais comuns (*integer*, *Double, string*, etc). Caso alguma referência tenha sido selecionada no campo de *Reference*, as classes serão listadas no *Type* também, por exemplo, Pessoa, Produto, etc.

A caixa de marcação *Type is Nullable (of [Type])* indica se a propriedade pode ter valores nulos, considerando que campos nos bancos de dados também podem ser nulos. A caixa de marcação *Add On PropertyChanged* não será tratada neste trabalho pois é especifica do padrão de desenvolvimento *MVVM (Model View View-Model).*

Com todos os campos obrigatórios preenchidos, basta clicar no botão *Add* que a ferramenta irá adicionar a propriedade na classe que estiver em foco no projeto.

```
Imports DBManager.DBMapper
Imports System.ComponentModel.DataAnnotations

<DBTable("SQLUser", "Fornecedor")> _
Public Class Fornecedor

#Region "Fornecedor_Fields"
    Private _ID as Integer

#End Region

#Region "Fornecedor_Properties"

    <DBField()> _
    <Key()> _
    <Editable(False)> _
    <Required()> _
    Public Property ID as Integer
        Get
            Return _ID
        End Get
        Set(ByVal value As Integer)
            _ID = value
        End Set
    End Property

#End Region

End Class
```

Figura 20. Classe criada pelo DBClassMapper

4.1.3. Relacionamentos

Os relacionamentos requerem uma atenção extra no DBClassMapper pois, por mais que seja simples criar um relacionamento, a automatização pode deixar o usuário sem entender o que está acontecendo.

Um relacionamento é criado quando o tipo de mapeador Foreign Key é selecionado. Na caixa de dados deve ser selecionado qual objeto é o relacionamento. Com estes dados, a ferramenta busca a chave do objeto

selecionado e cria a propriedade com a chave estrangeira sendo a chave do objeto (se for múltipla, ela será criada com todas as chaves). O cuidado a ser tomado é em entender qual chave no mapeamento está indicada como a chave estrangeira e qual é o nome da propriedade local. No atributo criado existem os parâmetros *ThisKey* e *OtherKeys*. O *ThisKey* indica o nome da chave local da propriedade e o *OtherKeys* indica as chaves estrangeiras do objeto relacionado. O DBClassMapper preenche tudo isso automaticamente, mas é bom entender de onde vem cada valor para evitar problemas de manutenção mais tarde. A figura 21 mostra um relacionamento criado pelo DBClassMapper.

```
<DBField()> _
<Association("PessoaFK", "Pessoa", "ID")> _
Public Property Pessoa as Pessoa
    Get
        Return _Pessoa
    End Get
    Set(ByVal value As Pessoa)
        _Pessoa = value
    End Set
End Property
```

Figura 21. Relacionamento com DBClassMapper

4.2. Mapeamento a partir de uma Base de Dados

O DBClassMapper tem funcionalidades específicas para ler um banco de dados e recuperar todo os seus esquemas, as suas tabelas e campos. Esta é a parte chamada de Mapeamento Automatizado que tem o nome de *DataBase Mapper Generator* e é chamada pelo botão *Read a Database to Create Full Class*. A partir de uma base de dados é possível criar todas as classes e propriedades já mapeadas, ou seja, todo o trabalho de mapear pode ser feito em

questão de poucos minutos se a base de dados já existir. Este é o caminho reverso das tabelas para as classes.

4.2.1. Leitura do Banco de Dados via ODBC

O mapeador automatizado, por questões de compatibilidade, acessa os SGBDs apenas por ODBC, pois este padrão de acesso acessa os SGBDs sempre da mesma maneira e com os mesmos comandos, tornando padronizada a automatização do mapeamento e não necessitando de implementações a mais para cada banco de dados. O Gendal além do ODBC também tem conexões nativas, pois os fabricantes de cada SGBD dizem que o desempenho dos SGBDs é superior com acesso nativo.

Na tela do mapeador (figura 22) existe um campo denominado DNS no qual são listadas as conexões ODBCs já existentes no Windows. Caso a conexão ainda não exista, basta clicar em *New DSN* para criar uma nova conexão. Cada configuração de conexão ODBC é totalmente dependente do SGBD e por isto não será tratado nenhum caso específico de conexão neste trabalho. Esta será a conexão utilizada para se ler o banco de dados e pegar as informações necessárias.

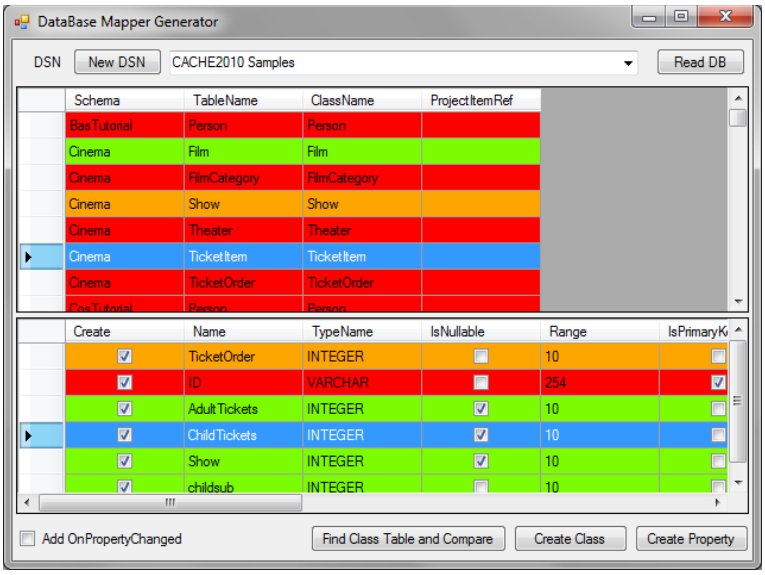


Figura 22. DBClassMapper Comparador

4.2.2. Criando Classes

Com a configuração para a conexão pronta, basta clicar em Read DB que o DBClassMapper irá iniciar a leitura das tabelas do SGBD. Será questionado ao usuário se ele deseja comparar as tabelas com as classes existentes no projeto. Se indicado que sim, a ferramenta irá fazer toda a comparação e colorir as informações sobre tabelas dependendo de como está a classe em comparação com o banco de dados. Abaixo segue a lista com as possíveis cores para tabelas (figura 22):

Verde: Indica que a tabela está totalmente compatível com a classe.

Laranja: Indica que a tabela está parcialmente compatível. Não quer dizer que seja necessário fazer alguma alteração, apenas indica alguma diferença. As diferenças podem indicar campos que não tem mapeamento na classe ou campos que tem mapeamentos mas o tipo de dado está diferente. Não ter um campo mapeado ou ter um campo com tipo diferente pode ser por questões de

projetos, por exemplo, na base de dados o campo é um *Integer*, mas na classe é uma String e por questões de projeto foi definido assim.

Vermelho: Indica que a tabela não está mapeada, ou seja, mesmo existindo no SGBD o sistema não conhece esta tabela.

Para criar a classe referente à tabela basta clicar em *Create Class*. A classe será criada já com todos os seus campos mapeados em propriedades da nova classe. A classe será criada dentro do projeto da pasta que estiver selecionada em questão. A figura 23 mostra um diagrama de sequencia de como o DBClassMapper faz para criar uma classe mapeada.

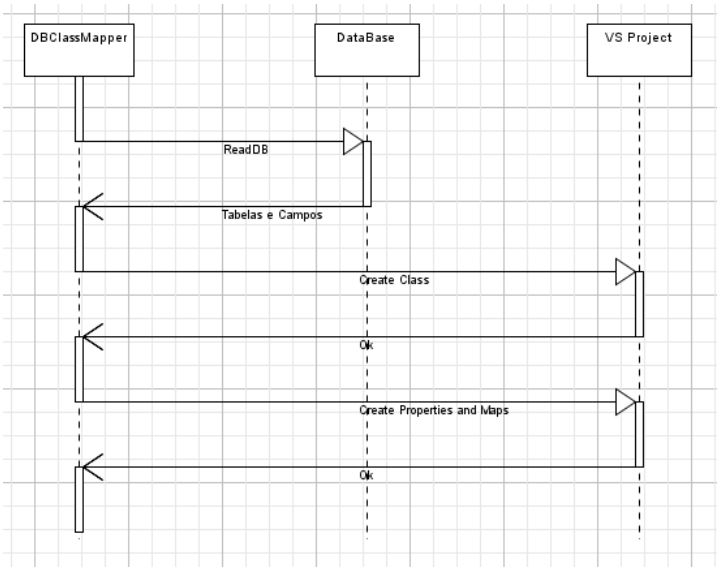


Figura 23. Criando uma Classe Mapeada

4.2.3. Criando Propriedades

As propriedades seguem um padrão parecido com o das tabelas para comparação e coloração. Com as tabelas comparadas basta dar um duplo clique na tabela desejada que os campos irão listar na tabela abaixo sendo comparados com as classe existente, caso existe a classe. Abaixo segue a lista com as cores dos campos na tabela e o que representam (figura 22):

Verde: O campo está totalmente compatível com a propriedade da sua classe.

Laranja: O tipo de dado do campo está diferente do tipo de dado da propriedade. Não quer dizer que está errado, apenas que está diferente e pode estar diferente por questões de projeto.

Vermelho: O campo não existe na classe em questão.

Esta parte serve principalmente para atualizar as classes com novas alterações no banco de dados, novos campos em tabelas já existentes. Basta selecionar a classe que se deseja criar e clicar em *Create Property*, a propriedade será criada dentro da classe já existente.

4.3. Construtor de Consultas

Outra importante função do DBClassMapper é a geração de consultas baseadas no modelo objeto já mapeado. Após ter todo o modelo de classes mapeado para o banco de dados correspondente, o Construtor de Consultas, chamado de *Query Builder,* tem a capacidade de ler todos os mapeamentos das classes e a partir deles montar árvores lógicas sobre os relacionamentos entre os mapeamentos. Com essa árvore lógica fica muito fácil para o desenvolvedor definir qualquer consulta visualmente, sem precisar lembrar como o modelo é necessariamente.

A Figura 24 mostra a interface gráfica do *Query Builder,* que é chamada pelo botão *Create Queries Based on Models,* e um exemplo de consulta. Na parte central ficam os nós que correspondem às classes envolvidas na consulta,

no lado esquerdo são listadas todas as classes que tem algum relacionamento com o nó selecionado e na parte inferior temos o código da consulta gerado. As guias da parte inferior são uma lista de Filtros e Ordenações que podem ser colocados na ordem que for desejada.

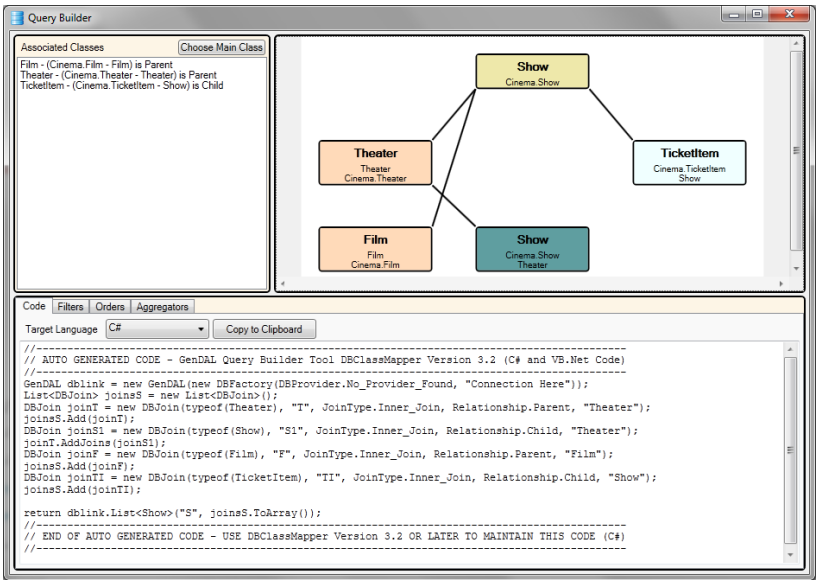


Figura 24. Query Builder - Visão Geral

4.3.1. Nó Principal da Consulta

Quando o *Query Builder* é aberto aparece uma nova janela listando todos os mapeamentos encontrados no projeto conforme mostra a Figura 24. Caso os mapeamentos estejam em um projeto diferente, basta selecionar a referência deste projeto na lista de referências (indicado no rodapé da nova janela) e mandar buscar novamente os mapeamentos (somente projetos ou arquivos de bibliotecas referenciados pelo projeto ao qual se está gerando a consulta serão listados). Nessa lista deve-se selecionar o nó principal da consulta, ou seja, a classe a partir da qual se obterá uma lista. Depois de selecionada a classe principal, um nó irá aparecer na janela central e já será gerado um código correspondente na parte inferior da interface. Analisando o código é possível ver

que se trata de uma consulta para a qual são listados todos os objetos existentes no banco de dados referentes àquela classe.

Na barra lateral esquerda é possível visualizar todas as classes que têm algum relacionamento com o nó principal. Nesta mesma barra existe um botão chamado *Choose Main Class*. Este botão serve para selecionar um novo nó principal (Figura 25). Esta ação substitui o nó anterior e remove todos os vínculos que existiam.

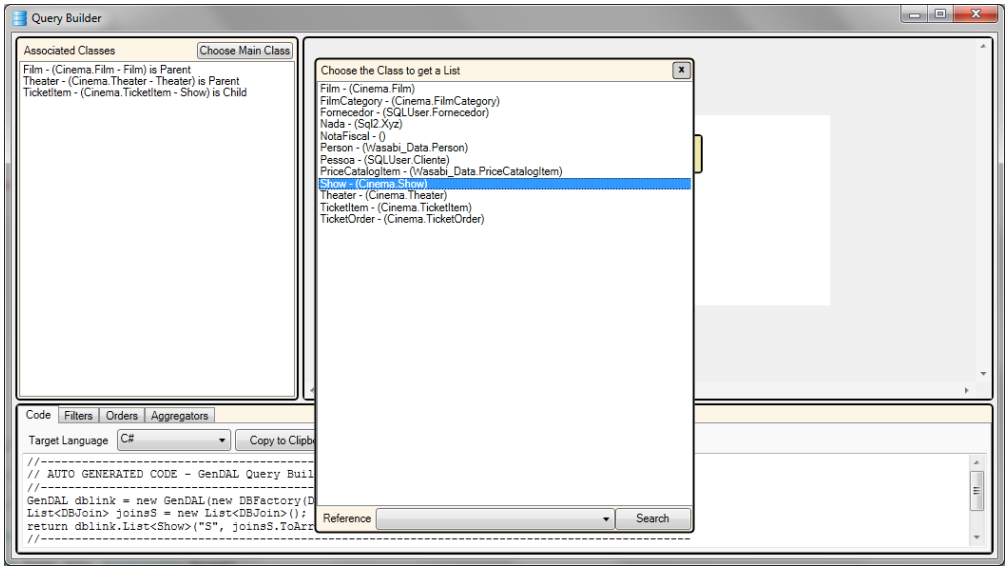


Figura 25. Query Builder - Escolher Nó Principal

4.3.2. Adicionando Nós

Selecionado o nó principal, observa-se que na lista lateral esquerda surgem seus relacionamentos. Com um duplo clique sobre qualquer um dos relacionamentos, o mesmo é adicionado a um nó pertencente ao nó selecionado em questão, no caso o nó principal.

Selecionando outros nós é possível ver e selecionar seus relacionamentos na barra lateral esquerda. Uma árvore correspondente à consulta será construída a partir dessas ações. Ela é baseada na ordem de

relacionamento selecionado pelo desenvolvedor, indicando quem é *parent* e *child* de quem. Muito importante é não confundir o relacionamento dos nós com o relacionamento do mapeamento. Relacionamentos mapeados independentemente se são *child* ou *parent* sempre estarão abaixo de seu nó relacionado e, na visão de uma árvore, todos eles são *childs* daquele nó. Na Figura 26 é possível visualizar uma estrutura onde o nó principal tem *childs* e *parents*, e na consulta, se olharmos na visão do nó principal, todos eles são *childs* para o nó. Essa estratégia de visualização, com o nó principal no topo, visa destacar o nó do qual desejamos recuperar dados.

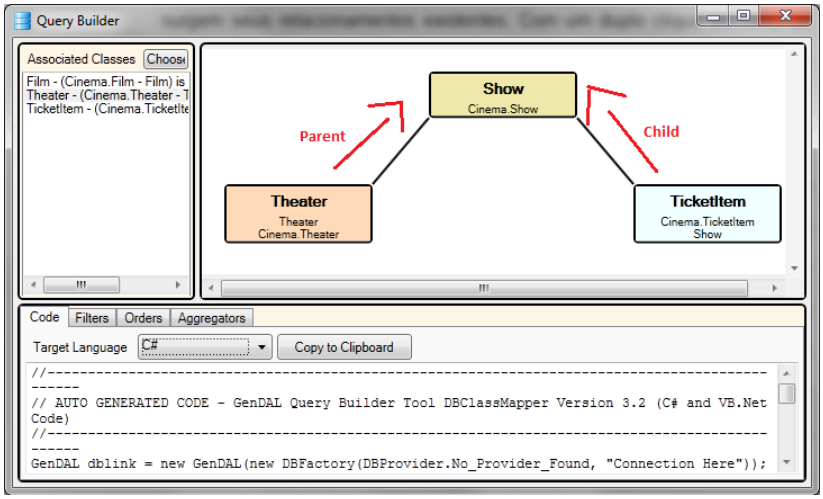


Figura 26. Query Builder – Nodos Parent e Child

4.3.3. Propriedades dos Nós

Clicando com o botão direito do mouse sobre um nó é possível removê-lo ou editá-lo. Quando um nó é removido, toda a sua estrutura abaixo é também removida. Editando as suas propriedades, é possível alterar o *alias* utilizado na consulta, alterar o tipo do *join* (*inner, left outer ou right outer*), além de adicionar filtros e ordenações. A Figura 27 mostra as propriedades de um nó da consulta (TicketItem).

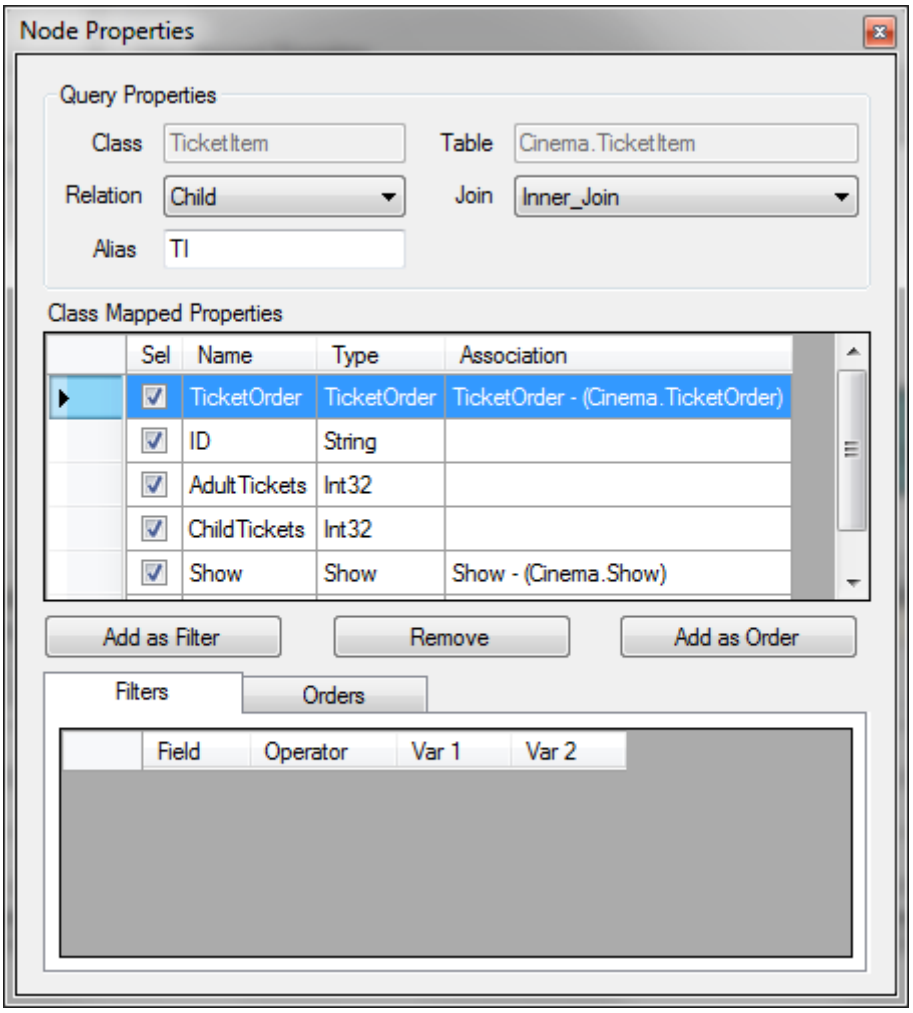


Figura 27.Query Builder – Propriedades de um Nó

A lista de propriedades de um nó já possui todas as suas propriedades, inclusive indicando se aquela propriedade está relacionada com alguma outra classe na coluna *association*. Para adicionar um filtro ou ordenação, basta selecionar a propriedade desejada e clicar no botão *Add as Filter* ou *Add as Order*.

É possível alterar a ordem em que os filtros e ordenações são executados nas consultas a partir das listas na parte inferior. Isto é importante para definir a ordenação e alguma otimização desejada na consulta (Figura 28).

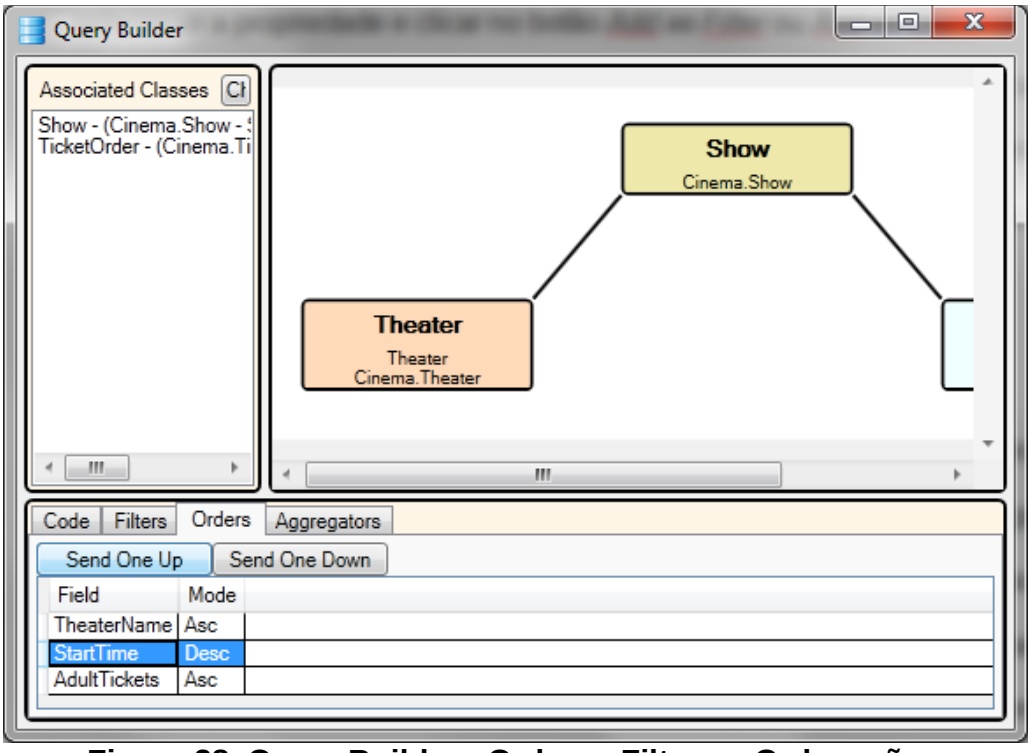


Figura 28. Query Builder - Ordenar Filtros e Ordenações

É possível fazer agregações nas listas conforme mostra a Figura 29. Como são listas de objetos fortemente tipados, as propriedades que são agregadas substituem as propriedades originais na lista. O exemplo da Figura 28 mostra um agregador agrupando por filmes do show e agregando o preço do tíquete adulto para pegar a média. O Gendal irá montar uma lista de shows agrupadas por filme tendo a propriedade *AdultPrice* preenchida com a média do preço adulto destes filmes em cada show.

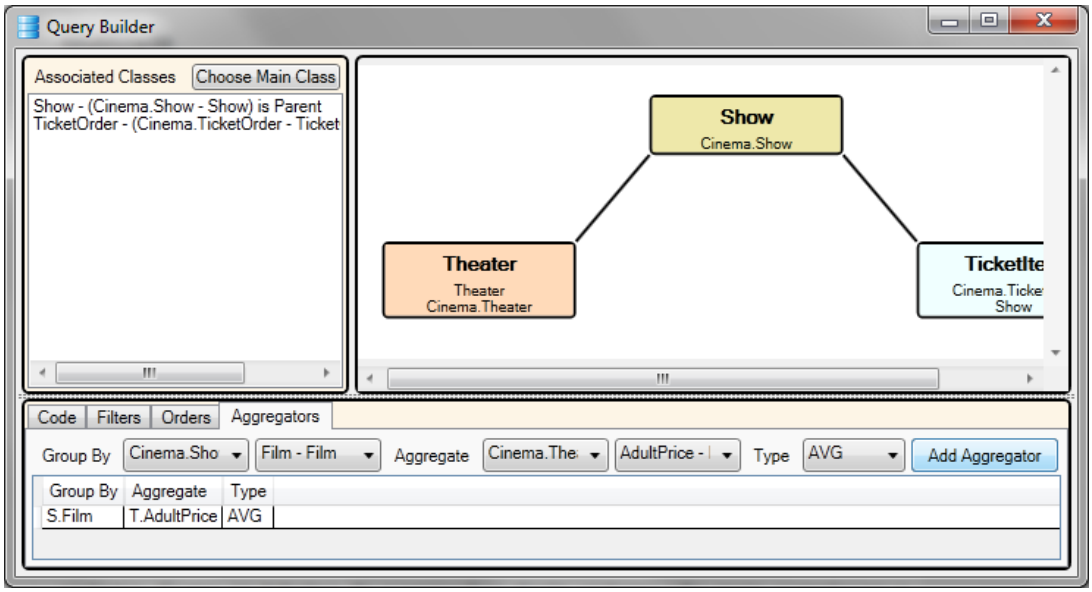


Figura 29. Query Builder - Agregadores

4.3.4. Geração da Consulta

Uma consulta é gerada automaticamente pelo DBClassMapper na barra inferior da tela. Nesta mesma barra é possível selecionar a linguagem destino da consulta. Atualmente, as linguagens suportadas são as mesmas do Gendal: C# e VB.Net. A Figura 30 mostra um diagram de sequencia exemplificando a criação de uma consulta.

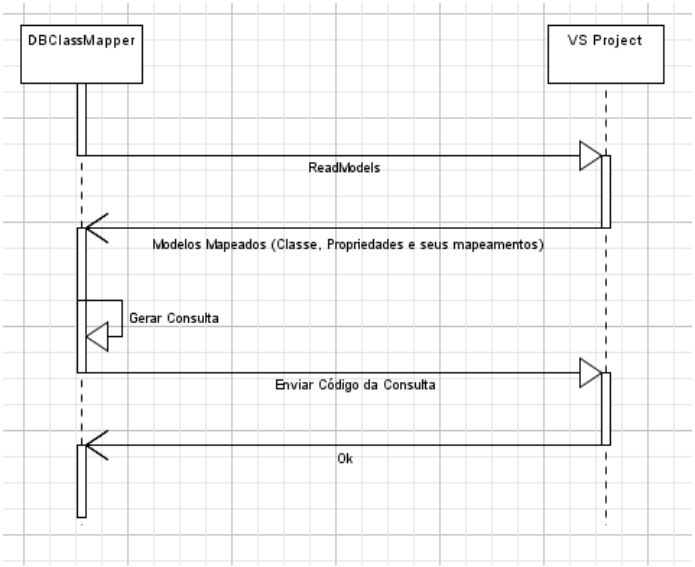


Figura 30. Gerando uma Consulta

As Figuras 31 e 32 mostram consultas reais que foram criadas no DBClassMapper para verificar se o usuário tem uma permissão específica em um determinado sistema e gerar uma lista de permissões que um usuário tem, respectivamente. Vale observar que a Figura 30 mostra como fica a consulta já inserida no código do programa após a utilização do *Query Builder*. Como as tabelas de permissões deste sistema são mais complexas e não tão intuitivas, a consulta ficou extensa, mas quando vista no DBClassMapper é mais facilmente entendida.

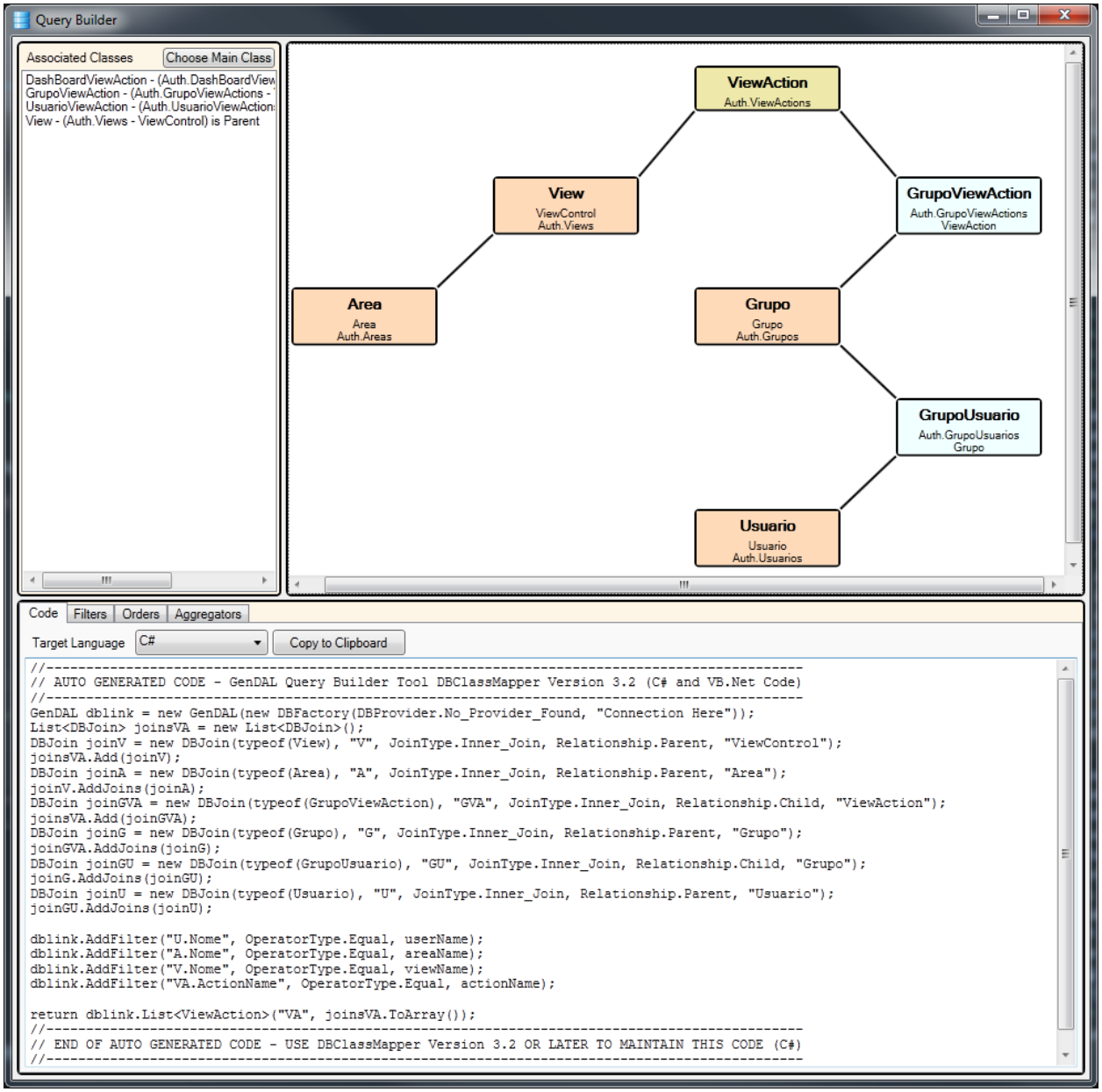


Figura 31. Query Builder - Verificar Permissão

```
public List<ViewAction> ListUserViewActions(Usuario u)
{
    //------------------------------------------------------------------------------------  ---------
    // AUTO GENERATED CODE - GenDAL Query Builder Tool DBClassMapper Version 3.0 (C# and VB.Net Code)
    //------------------------------------------------------------------------------------  ---------
    GenDAL dblink = new GenDAL(Factory);
    List<DBJoin> joinsVA = new List<DBJoin>();
    DBJoin joinUVA = new DBJoin(typeof(UsuarioViewAction), "UVA", JoinType.Inner_Join, Relationship.Child);
    joinsVA.Add(joinUVA);

    dblink.AddFilter("UVA.Usuario", OperatorType.Equal, u.ID);

    return dblink.List<ViewAction>("VA", joinsVA.ToArray());
    //------------------------------------------------------------------------------------  ---------
    // END OF AUTO GENERATED CODE - USE DBClassMapper Version 3.0 OR LATER TO MAINTAIN THIS CODE (C#)
    //------------------------------------------------------------------------------------  ---------
}
```

Figura 32. Query Builder - Listar Permissões

5. Estudo de Caso

Um outro trabalho de conclusão de curso (TCC) do INE/UFSC utilizou o Gendal e o DBClassMapper como as principais ferramentas para a persistência de seus dados. O titulo do TCC é “Sistema de Alta disponibilidade para Gerenciamento de Pedidos e Estoque”, TCC do curso de Ciências da Computação desenvolvido pela aluna Mariell Schappo. Trata-se de um sistema distribuído de bases de dados para a geração de pedidos, ou seja, a persistência de dados neste trabalho é muito importante, as consultas devem ser extremamente rápidas e não pode haver falhas na persistência de dados. Este trabalho se mostrou adequado como um cenário para mostrar o poder e a confiabilidade do Gendal e para mostrar a agilidade que o DBClassMapper dá à geração dos mapeamentos e consultas.

Este sistema trabalha com estoques totalmente distribuídos, ou seja, partes do estoque encontram-se em diferentes servidores, mas todos eles sabem a quantidade total. Quando um servidor recebe um pedido que requer uma quantidade igual ou menor ao seu estoque local, este mesmo servidor confirma o pedido, atualiza seu estoque local e envia aos outros servidores a atualização do estoque total e seu pedido gerado. Já quando um servidor não tem estoque local suficiente, mas sabe que globalmente existe quantidade suficiente, este servidor envia uma requisição aos outros para confirmar se há realmente estoque disponível e já requisita que algum dos servidores ceda a quantidade necessária para completar o pedido. Caso tudo ocorra bem, o servidor recebe a confirmação para completar o pedido e já o envia aos outros.

Na situação do estoque não ser suficiente ou os servidores retornarem que não há estoque global para tal requisição, o servidor recusa o pedido avisando o motivo.

Segue abaixo algumas figuras (Figura 33, Figura 34 e Figura 35) ilustrando o uso do Gendal e do DBClassMapper no TCC citado.

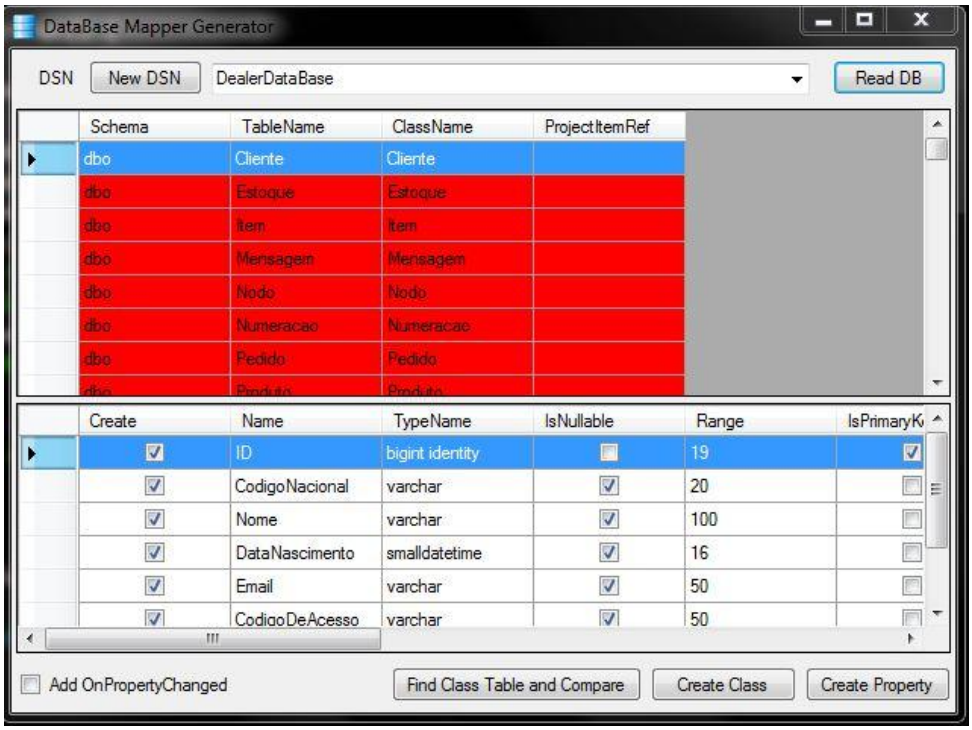
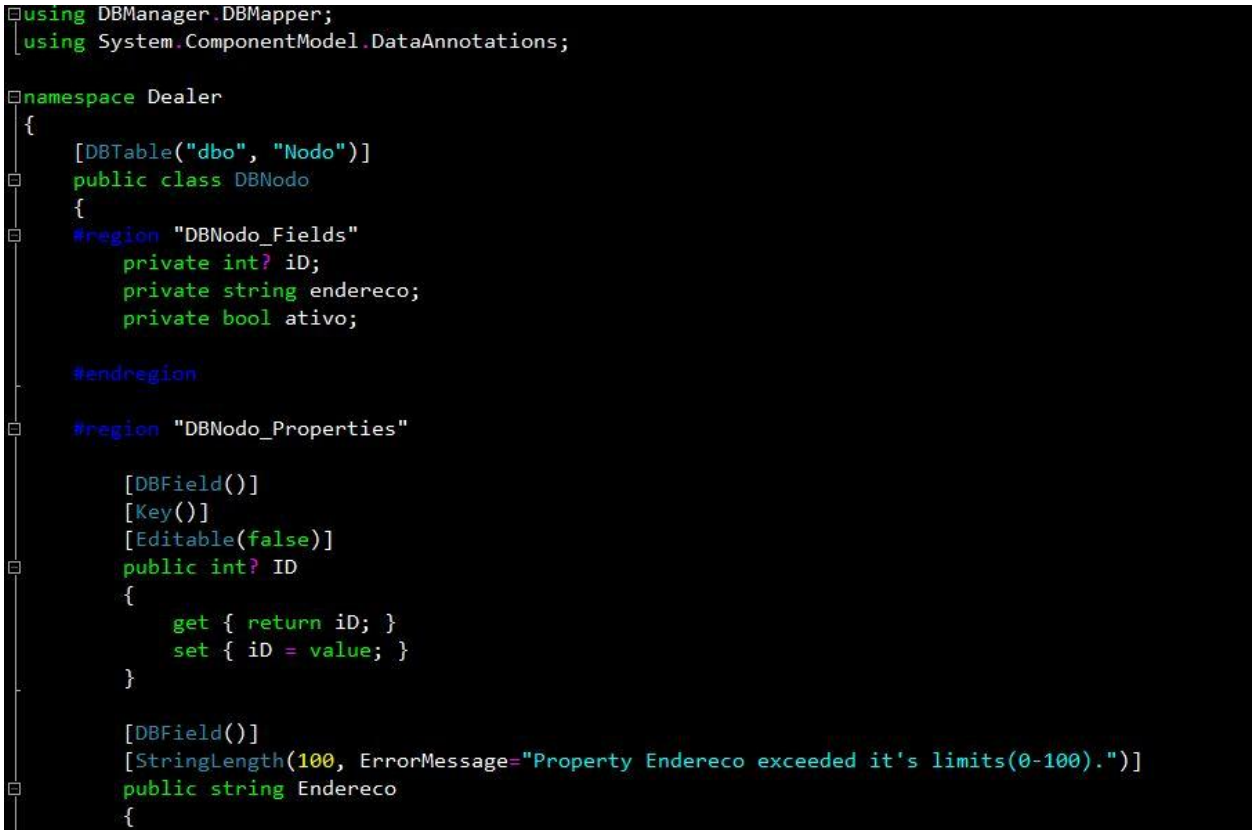


Figura 33. Estudo de Caso – Mapeando

A Figura 33 mostra a utilização do mapeador no Estudo de Caso. A parte superior da interface com o usuário mostra as tabelas do banco de dados do sistema de pedidos e a parte inferior mostra os campos da tabela selecionada. As linhas em vermelho indicam tabelas que ainda não tem mapeamento no programa.



```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;

namespace Dealer
{
    [DBTable("dbo", "Nodo")]
    public class DBNodo
    {
    #region "DBNodo_Fields"
        private int? iD;
        private string endereco;
        private bool ativo;

    #endregion

    #region "DBNodo_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public int? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        [StringLength(100, ErrorMessage="Property Endereco exceeded it's limits(0-100).")]
        public string Endereco
        {
```

Figura 34. Estudo de Caso – Classe Mapeada

A Figura 34 mostra uma classe objeto mapeada para uma tabela com o suporte do DBClassMapper. No caso, a classe DBNodo está sendo mapeada para a tabela dbo.Nodo. Esta classe também tem 3 propriedades sendo mapeadas para campos da tabela: iD, endereco e ativo. Na Figura 34 é possível ver somente o mapeamento de iD e endereco.

```
public class ImpCliente:Cliente
{
    public int Cadastrar() {
        int i = 0;
        DBCliente dbC = new DBCliente();

        dbC.Bloqueado = this.Bloqueado;
        dbC.CodigoDeAcesso = this.CodigoDeAcesso;
        dbC.CodigoNacional = this.CodigoNacional;
        dbC.DataNascimento = this.DataNascimento;
        dbC.Email = this.EMail;
        dbC.ID = null;
        dbC.MsgBloqueado = this.MensagemBloqueio;
        dbC.Nome = this.Nome;

        DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
        GenDAL gd = new GenDAL(f);

        try
        {
            i = gd.Save(ref dbC);
        }
        catch (Exception err)
        {
            i = -1;
        }

        return i;
    }
```

Figura 35. Estudo de Caso – Persistindo um Objeto

Na Figura 35 observa-se um objeto do tipo DBCliente tendo as suas propriedades populadas e logo em seguida o Gendal é instanciado e utilizado para salvar, ou seja, persistir o objeto populado no banco de dados. Isto foi possível pois o objeto DBCliente foi mapeado através do DBClassMapper.

6. Conclusão

O Gendal é um ORM alternativo ao Entity Framework da Microsoft para o Visual Studio que tem as suas curvas de aprendizagem. Com o DBClassMapper essas curvas foram reduzidas através do mapeador, que lê o banco de dados e já gera as classes mapeadas, e do gerador de consultas, que com apenas alguns cliques permite a geração de consultas extremamente complexas.

Ao utilizar o mapeador do DBClassMapper, o usuário instintivamente começa a entender como os mapeamentos funcionam, podendo fazer alterações manuais para melhorar os mapeamentos em algum aspecto, como por exemplo já indicar descrições mais legíveis para as chaves estrangeiras no mapeamento de *Association*.

O Gendal está sendo utilizado amplamente em um novo sistema ERP para a Web. A geração da consulta com o *Query Builder* é a parte mais complexa de se entender e utilizar no Gendal. De acordo com a equipe deste sistema, com o DBClassMapper, ficou muito mais simples gerar consultas do que mapear tabelas, mostrando assim, a contribuição do DBClassMapper aliado ao Gendal. Com poucas consultas geradas, o usuário já começa a entender como as consultas no Gendal funcionam e ocorre a mesma situação com o mapeador, ou seja, o usuário começa a ter a capacidade de manualmente fazer alterações para melhorar as consultas, como por exemplo, alterando os *Alias* das tabelas para ficarem mais legíveis para outros programadores.

Com isso, conclui-se que o DBClassMapper é uma importante ferramenta na utilização do Gendal. Sem o DBClassMapper, o trabalho de mapear e gerar

consultas seria mais complicado e a curva de aprendizagem da própria biblioteca seria muito mais elevada. Entende-se que os objetivos pretendidos com o desenvolvimento do DBClassMapper foram alcançados.

O DBClassMapper presta um importante apoio ao Gendal ao ler o banco de dados e mapear as tabelas mas, o inverso não foi desenvolvido, ou seja, ainda não é possível gerar as tabelas do banco de dados a partir das classes. Um importante trabalho futuro seria desenvolver tal funcionalidade, deixando o DBClassMapper mais flexível, ou seja, não obrigando o desenvolvedor a primeiramente criar o banco de dados para depois mapeá-lo. Esta funcionalidade também auxiliaria o Gendal no sentido de torná-lo um ORM multibanco, ou seja, se as classes já existem, poder-se-ia criar as tabelas em qualquer banco de dados.

7. Referências Bibliográficas

1. AMBLER, SCOTT W. ***Agile Database Techniques***. 1.ed. Nova Yorque: Wiley & Sons, 2003.
2. ROBINSON, Simon; NAGEL, Christian; WATSON, Karli; GLYNN, Jay; SKINNER, Morgan, EVJEN, Bill. **Professional C#.** Terceira edição, editora Wrox.
3. MSDN, Queries in LINQ to Entities. Disponível em: <http://msdn.microsoft.com/en-us/library/bb399367.aspx>. Acesso em: 26 jun. 2012
4. MSDN, Generating Models and Mappings. Disponível em: <http://msdn.microsoft.com/en-us/library/bb399596.aspx>. Acesso em: 26 jun. 2012
5. MSDN, Mapping a Conceptual Model to Storage Model. Disponível em: <http://msdn.microsoft.com/en-us/library/bb399232.aspx>. Acesso em: 26 jun. 2012
6. MSDN, Query Syntax Examples: Filtering. Disponível em: <http://msdn.microsoft.com/en-us/library/bb738636.aspx>. Acesso em: 26 jun. 2012
7. MSDN, Query Syntax Examples: Ordering. Disponível em: <http://msdn.microsoft.com/en-us/library/bb738627.aspx>. Acesso em: 26 jun. 2012
8. MSDN, Query Expression Syntax Examples: Join Operators. Disponível em: <http://msdn.microsoft.com/en-us/library/bb896266.aspx>. Acesso em: 26 jun. 2012

9. WIKIPÉDIA, Mapeamento objeto-relacional. Disponível em: <http://pt.wikipedia.org/wiki/Mapeamento_objeto-relacional>. Acesso em: 26 jun. 2012
10. JBoss Community, Hibernate Tools for Eclipse and Ant. Disponível em: <http://www.hibernate.org/subprojects/tools.html>. Acesso em: 26 jun. 2012
11. JBoss Community, Hibernate Getting Started Guide. Disponível em: <http://docs.jboss.org/hibernate/orm/4.1/quickstart/en-US/html/>. Acesso em: 26 jun. 2012

APÊNDICE A – Artigo 1

DBClassMapper: Uma Ferramenta de Apoio ao Mapeamento e Consultas ao *ORM* Gendal

Diego Magno da Silva, Ronaldo dos Santos Mello

Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil.

{diegomagno,ronaldo}@inf.ufsc.br

Resumo. *Este artigo apresenta o DBClassMapper, uma ferramenta de apoio ao mapeamento e consultas ao ORM Gendal cujo principal diferencial é gerar consultas a partir das classes de objetos e poder comparar estas classes com as tabelas relacionais do banco de dados, facilitando a atualização dos mapeamentos. Esta ferramenta foi criada para a IDE Visual Studio 2010 da Microsoft, a qual utiliza o .Net como principal framework de desenvolvimento. A novidade do DBClassMapper é que o desenvolvedor pode gerar consultas complexas a partir de poucos cliques e com muito pouco conhecimento de SQL.*

Abstract. *This paper presents the DBClassMapper, a tool supporting the mapping and queries to the ORM Gendal whose main difference is generating queries from the classes of objects and to compare these classes to relational tables in the database, making it easier to update the mappings. This tool is designed for Visual Studio 2010 IDE from Microsoft, which uses the .Net as the primary development framework. The novelty of DBClassMapper is that the developer can generate complex queries from a few clicks and with very little knowledge of SQL.*

1. Introdução

Em uma aplicação orientada a objetos que utiliza um banco de dados relacional para persistência de seus dados, uma ferramenta de *ORM* é importante para integrar os objetos da aplicação ao banco de dados. A utilização de um *ORM* aumenta a produção de uma equipe, pois reduz o problema da impedância ou *impedance mismatch* [Ambler, 2003], uma vez que o programador da aplicação não precisa se preocupar com a construção de comandos na linguagem SQL para realizar a definição e manipulação de dados [ORM, 2012]. Uma solução neste contexto é o *ORM* Gendal (*Generic Data Access Library*). *Gendal* é um *ORM* para o *framework .Net*, que é bem utilizado pela *IDE Visual Studio*, ambas da Microsoft. Os principais pontos fortes do *Gendal* são a simplicidade de definição dos mapeamentos e definição de consultas complexas.

O *DBClassMapper* é uma ferramenta desenvolvida para o Visual Studio 2010 com a finalidade de auxiliar a utilização do *Gendal*. Especificamente, *DBClassMapper* provê uma melhoria de usabilidade ao *Gendal* pois facilita a manipulação da biblioteca ao permitir a construção interativa e simples de mapeamentos e consultas complexas ao banco de dados. .

Comparando com trabalhos relacionados, verifica-se que a Microsoft vem melhorando o seu *ORM Entity Framework* visando facilitar a sua utilização. O problema é que mesmo no Visual Studio 2010 o *Entity Framework* não oferece ferramentas de suporte para mapeamento. Já o o *Hibernate* (outro *ORM* para *.Net*) não tem nenhuma ferramenta para essa IDE, e as ferramentas existentes para outras plataformas não permitem a construção interativa de consultas a partir das classes de objetos. Estas deficiências motivaram o projeto do *DBClassMapper* sobre o *ORM Gendal*.

O *DBClassMapper* é uma ferramenta criada para reduzir o trabalho massivo de mapear classes para um esquema relacional. Com este objetivo, foram criadas estruturas que auxiliam o desenvolvedor a definir classes e suas propriedades já mapeadas, ou seja, a ferramenta cria a propriedade e seu mapeamento junto. Além

disso, a ferramenta também auxilia a criação de classes e propriedades que não são mapeadas.. Quando já existe um banco de dados e algumas classes já mapeadas, a tarefa torna-se saber o que e aonde deverá ser ajustado. Este problema foi resolvido com uma nova funcionalidade que lê o banco de dados e compara todas as tabelas com todas as classes, já trazendo um relatório completo do que é necessário criar ou alterar, e apontando o que esta diferente. Para simplificar, a partir desta comparação é possível com poucos cliques já criar as classes e as propriedades faltantes, além de também poder alterar as que têm diferenças.

2. Trabalhos Relacionados

Algumas soluções comerciais para *ORM* e suas ferramentas encontram-se disponíveis, como o *Hibernate* com suas ferramentas de apoio para o Eclipse, e o *Entity Framework* da Microsoft com algumas facilidades para o Visual Studio. A vantagem do *DBClassMapper* em relação ao *Hibernate* para o Eclipse é que não existe uma ferramenta totalmente integrada que ofereça todas as funcionalidades providas por ela. As ferramentas de apoio ao *Hibernate* são normalmente separadas, ou seja, diferentes funcionalidades estão em diferentes ferramentas, sendo necessário utilizar várias delas para executar certa tarefa. Já o *Entity Framework* tem apenas algumas funções de auxílio para a sua utilização, as quais sem elas não seria possível executá-lo, ou seja, não existe nenhuma ferramenta que auxilie, por exemplo, a gerar uma consulta ao banco de dados.

Um importante diferencial do *DBClassMapper* é a sua grande portabilidade para qualquer SGBD que tenha acesso via ODBC. O *Gendal*, *ORM* ao qual o *DBClassMapper* oferece suporte, pode se conectar com qualquer SGBD, desde que este tenha sido implementado ou tenha acesso através de ODBC. Desta forma, a grande maioria dos SGBDs pode ser utilizado pelo *Gendal* e pelo *DBClassMapper*.

3. Ferramenta DBClassMapper: Utilização e Funcionalidades

A ferramenta *DBClassMapper* foi desenvolvida para o Visual Studio 2010 da Microsoft em *Visual Basic .Net 4.0* como um *AddIn* (denominação dada aos plug-ins no *Visual Studio*). A sua instalação é muito simples e rápida: basta executar o seu arquivo de instalação no Windows (XP ou 7). Quando o *Visual Studio* for aberto posteriormente, a ferramenta já irá aparecer no menu *Tools*, como mostra Figura 1.

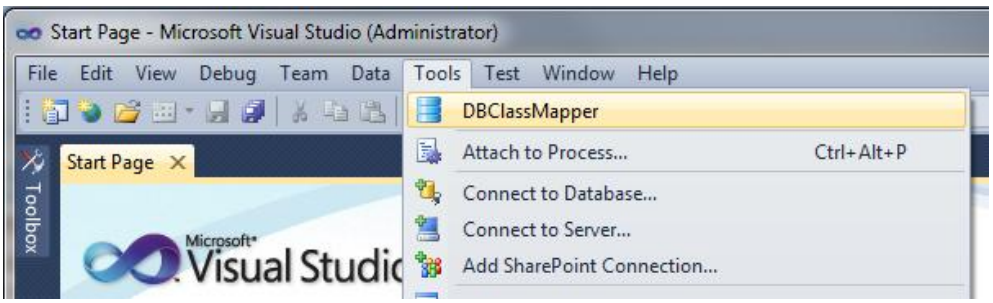


Figura 1. *DBClassMapper* no menu *Tools* do Visual Studio 2010

O *DBClassMapper* possui várias facilidades integradas em uma única ferramenta. A sua tela de entrada, mostrada na Figura 2, apresenta essas funcionalidades. .

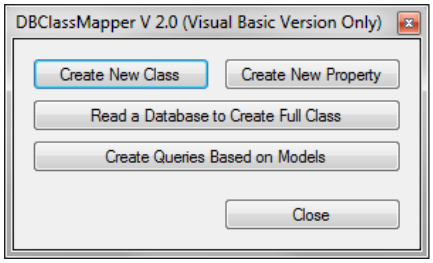


Figura 2. Tela principal d*o* DBClassMapper

Nesta versão, a ferramenta gera códigos apenas para *Visual Basic*. Como trabalho futuro, pretende-se adicionar à geração de códigos para outras linguagens como o C# ou C++). A primeira funcionalidade (*Create New Class*) O primeiro botão (superior à esquerda) abre uma tela para auxiliar a criação de classes mapeadas individualmente.. A segunda funcionalidade (*Create New Property*) irá abrir uma tela para auxiliar a criação das propriedades mapeadas individualmente. A terceira funcionalidade (botão no centro irá abrir uma tela que permitirá comparar um esquema de banco de dados com as classes mapeadas do código fonte do projeto que está ativo no momento e a última funcionalidade (parte inferior) irá abrir uma tela na qual será possível gerar consultas utilizando as classes já mapeadas. Cada uma destas funcionalidades é explicada a seguir.

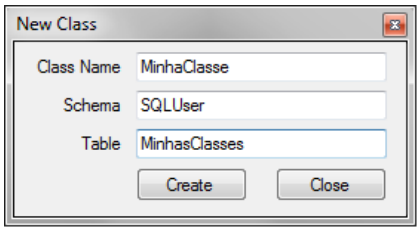


Figura 3. Tela *Create New Class*

A Figura 3 mostra a tela para criar um novo mapeamento de classe, acessível pelo botão *Create New Class*. Nesta tela basta indicar o nome da classe, o nome da tabela e o esquema do banco de dados já existente, que esta tabela estará. Apenas o nome da classe é obrigatório. Caso os outros estejam vazios, o *Gendal* irá supor que o nome da tabela é o mesmo da classe e que é utilizado o esquema padrão do SGBD.

A Figura 4 mostra a tela referente à funcionalidade *Create New Property*. Nesta tela se indica o tipo de mapeamento (o qual pode ser *None*, ou seja, não será mapeada, ou tipos padrões como key, alternate key, field, etc). Na sequência, é possível indicar propriedades gerais que um atributo em uma tabela pode ter (a lista varia de acordo com o tipo de mapeamento selecionado), bem como o nome da propriedade. O campo *Reference* indica uma referência a um tipo de objeto, sendo possível selecionar este tipo de objeto no campo *Type* para estabelecer um relacionamento. Por fim, informa-se o tipo de dado e os dois últimos *check boxes* definem, respectivamente,se a propriedade pode ser nula e o ultimo serve exclusivamente para classes de programas que utilizam o padrão de programação *MVVM (Model View View-Model)*.

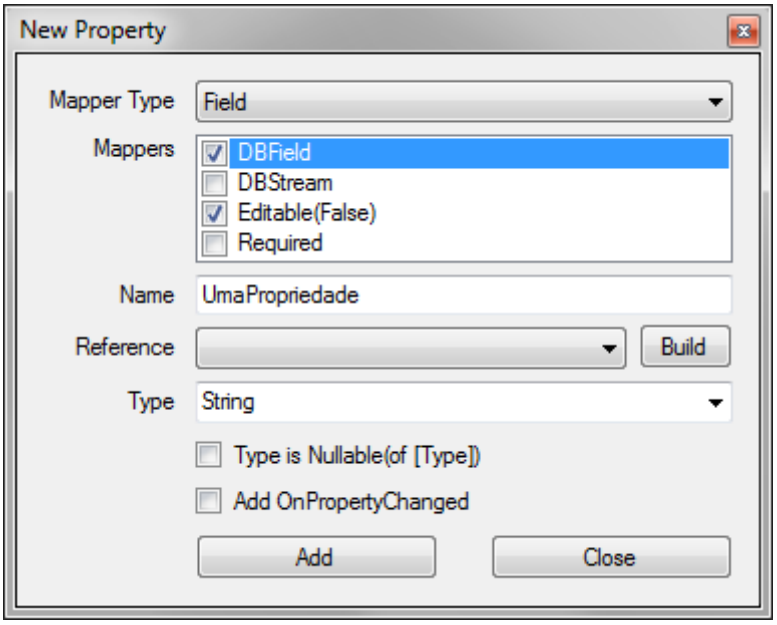


Figura 4. Tela *Create New Property*

A Figura 5 mostra a tela para se fazer comparações entre o esquema de um banco de dados com as classes mapeadas do sistema, para fins de criação e atualização das propriedades (vide Figura 6). No DSN é indicado um ODBC já criado previamente no Windows para se conectar com o SGBD. No botão *New* DSN é possível criar este ODBC. Após indicada a conexão, basta clicar no botão Read DB. O usuário será questionado se deseja fazer a comparação entre as tabelas do banco e as suas classes.

Na visualização de tabelas é mostrado em vermelho as tabelas que não tem vinculo com alguma classe; em laranja as que têm vinculo, mas estão diferentes em termos de propriedades mapeadas; e em verde as que estão equivalentes. Ao dar um duplo clique em uma tabela, os seus atributos são comparados com as propriedades mapeadas da classe. Atributos em vermelho são os que não têm qualquer vinculo com alguma propriedade; Atributos em laranja tem algum vinculo, mas possuem alguma diferença (tipo de dado diferente, por exemplo), e atributos em verde estão equivalentes. Vale ressaltar que, uma tabela ou atributo destacado em laranja não significa que seu mapeamento esteja errado. A finalidade aqui é apenas evidenciar que ele não esta totalmente equivalente, mas, mesmo assim, um desenvolvedor pode desejar um mapeamento desta forma por motivos de projeto.

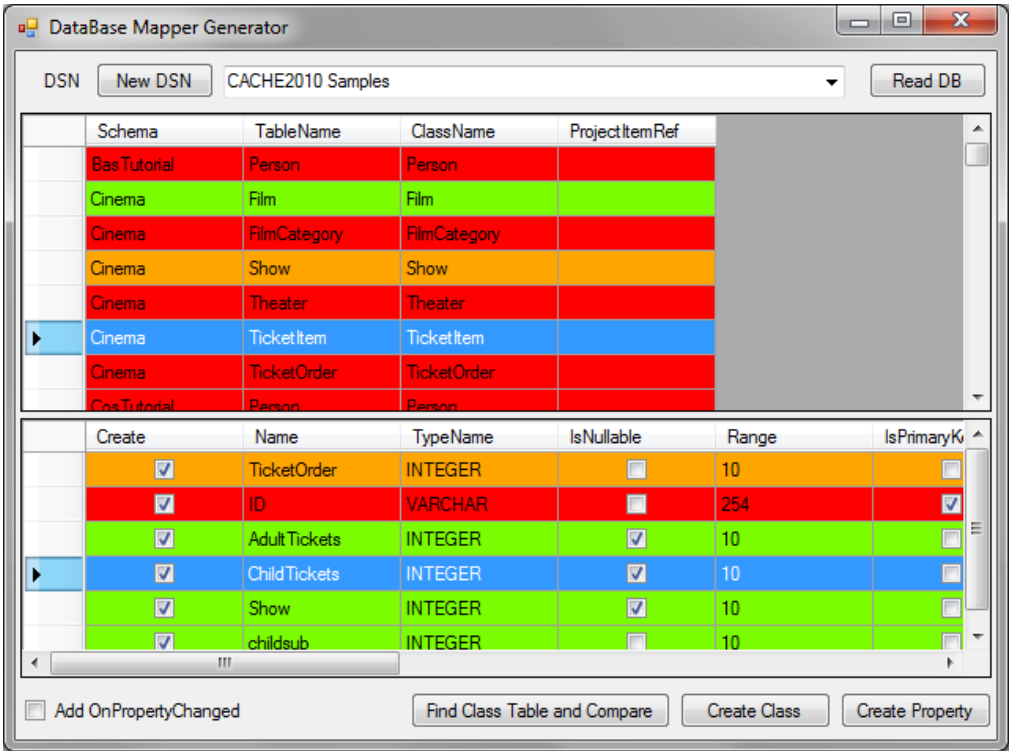


Figura 5. Tela correspondente à *Read a Database to Create Full Class*

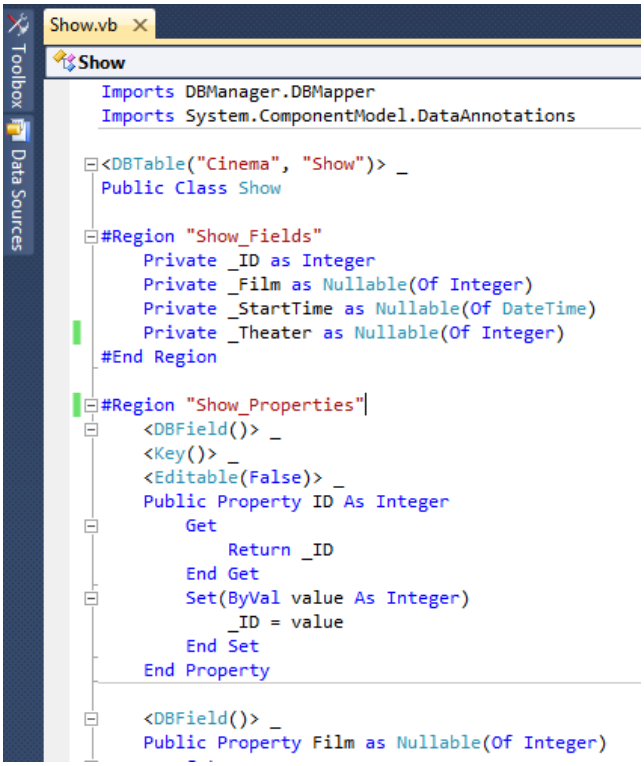


Figura 6. Exemplo de código gerado a partir do comparador

A Figura 7 mostra a tela correspondente à funcionalidade *Create Queries Based on Models*, que é um importante diferencial do *DBClassMapper*. Esta tela permite a definição de consultas baseadas nas classes de objetos No lado esquerdo da tela existem duas listas: a primeira mostra todas as classes de objetos que estão mapeadas e a

segunda mostra as classes que estão diretamente relacionadas à classe selecionada na primeira lista.

A formulação de uma consulta é realizada arrastando classes para a área de montagem de consultas à direita. Ao serem posicionados nesta área, os relacionamentos existentes entre as classes vão sendo exibidos. Em cada uma das classes pode-se adicionar filtros interativamente, como mostrado na Figura 6. Ainda, é possível indicar um *Alias* para a tabela e indicar uma ordenação para o resultado.

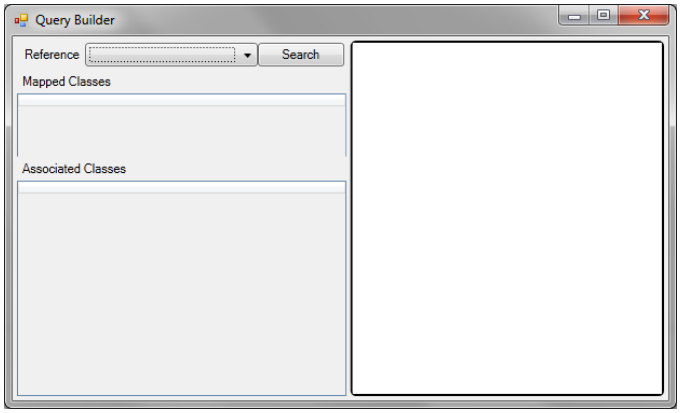


Figura 7. Tela *Create Queries Based on Models*

4. Conclusão

Este artigo apresenta *DBClassMapper*, uma ferramenta gráfica para um usuário desenvolvedor de aplicações orientadas a objetos que integra diversas funcionalidades relacionadas ao processo de mapeamento objeto-relacional. Ela foi especificamente desenvolvida para dar suporte ao *ORM Gendal* com qualquer SGBD.

A principal contribuição d*o DBClassMapper*, se comparada com ferramentas similares, é justamente o fato de disponibilizar diversas funcionalidades de outras já existentes integradas em uma única ferramenta, além de ser a primeira ferramenta para o *ORM Gendal* e uma das poucas (talvez a única) existente para um *ORM* para o *.Net* no *Visual Studio 2010*. Outra inovação importante da ferramenta é o gerador interativo de consultas, funcionalidade não disponível em outros *ORMs*..

Trabalhos futuros incluem a possibilidade de a partir das classes gerar as tabelas e campos do banco de dados, uma vez que, atualmente o esquema do banco deve existir a *priori*, experimentos de usabilidade no gerador de consultas e a disponibilidade da ferramenta em outras plataformas.

Referências

http://msdn.microsoft.com/en-us/data/ee712907, Acesso em: 24/06/2012

http://www.hibernate.org/subprojects/tools.html, Acesso em: 24/06/2012

Ambler, Scott W. ***Agile Database Techniques***. 1.ed. Nova Yorque: Wiley & Sons, 2003.

http://pt.wikipedia.org/wiki/ORM, Acessado em: 24/06/2012

APÊNDICE B – Artigo 2

DBClassMapper: Uma Ferramenta de Apoio ao Mapeamento e Consultas ao *ORM* Gendal

Diego Magno da Silva, Ronaldo dos Santos Mello

Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil.

{diegomagno,ronaldo}@inf.ufsc.br

Resumo. *Este artigo apresenta o DBClassMapper, uma ferramenta de apoio ao mapeamento e consultas ao ORM Gendal cujo principal diferencial é gerar consultas a partir das classes de objetos e poder comparar estas classes com as tabelas relacionais do banco de dados, facilitando a atualização dos mapeamentos. Esta ferramenta foi criada para a IDE Visual Studio 2010 da Microsoft, a qual utiliza o .Net como principal framework de desenvolvimento. A novidade do DBClassMapper é que o desenvolvedor pode gerar consultas complexas a partir de poucos cliques e com muito pouco conhecimento de SQL.*

Abstract. *This paper presents the DBClassMapper, a tool supporting the mapping and queries to the ORM Gendal whose main difference is generating queries from the classes of objects and to compare these classes to relational tables in the database, making it easier to update the mappings. This tool is designed for Visual Studio 2010 IDE from Microsoft, which uses the .Net as the primary development framework. The novelty of DBClassMapper is that the developer can generate complex queries from a few clicks and with very little knowledge of SQL.*

1. Introdução

Em uma aplicação orientada a objetos que utiliza um banco de dados relacional para persistência de seus dados, uma ferramenta de *ORM* é importante para integrar os objetos da aplicação ao banco de dados. A utilização de um *ORM* aumenta a produção de uma equipe, pois reduz o problema da impedância ou *impedance mismatch* [Ambler, 2003], uma vez que o programador da aplicação não precisa se preocupar com a construção de comandos na linguagem SQL para realizar a definição e manipulação de dados [ORM, 2012]. Uma solução neste contexto é o *ORM* Gendal (*Generic Data Access Library*). *Gendal* é um *ORM* para o *framework .Net*, que é bem utilizado pela *IDE Visual Studio*, ambas da Microsoft. Os principais pontos fortes do *Gendal* são a simplicidade de definição dos mapeamentos e definição de consultas complexas.

O *DBClassMapper* é uma ferramenta desenvolvida para o Visual Studio 2010 com a finalidade de auxiliar a utilização do *Gendal*. Especificamente, *DBClassMapper* provê uma melhoria de usabilidade ao *Gendal* pois facilita a manipulação da biblioteca ao permitir a construção interativa e simples de mapeamentos e consultas complexas ao banco de dados.

2. Ferramenta DBClassMapper: Utilização e Funcionalidades

A ferramenta *DBClassMapper* foi desenvolvida para o Visual Studio 2010 da Microsoft em *Visual Basic .Net 4.0* como um *AddIn* (denominação dada aos plug-ins no *Visual Studio*). A sua instalação é muito simples e rápida: basta executar o seu arquivo de instalação no Windows (XP ou 7). Quando o *Visual Studio* for aberto posteriormente, a ferramenta já irá aparecer no menu *Tools*, como mostra Figura 1.

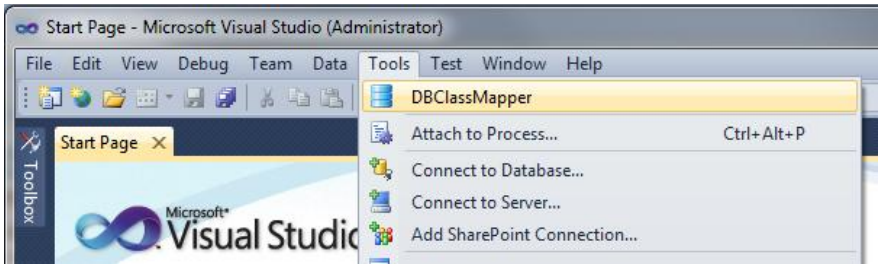


Figura 1. *DBClassMapper* no menu *Tools* do Visual Studio 2010

O *DBClassMapper* possui várias facilidades integradas em uma única ferramenta. A sua tela de entrada, mostrada na Figura 2, apresenta essas funcionalidades. .

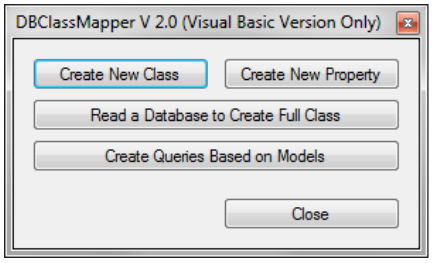


Figura 2. Tela principal d*o* DBClassMapper

A primeira funcionalidade (*Create New Class*) O primeiro botão (superior à esquerda) abre uma tela para auxiliar a criação de classes mapeadas individualmente. A segunda funcionalidade (*Create New Property*) irá abrir uma tela para auxiliar a criação das propriedades mapeadas individualmente. A terceira funcionalidade (botão no centro irá abrir uma tela que permitirá comparar um esquema de banco de dados com as classes mapeadas do código fonte do projeto que está ativo no momento e a última funcionalidade (parte inferior) irá abrir uma tela na qual será possível gerar consultas utilizando as classes já mapeadas. Cada uma destas funcionalidades é explicada a seguir.

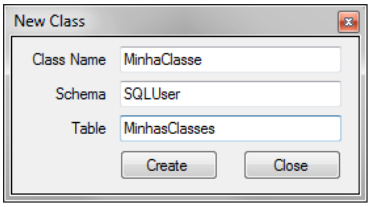


Figura 3. Tela *Create New Class*

A Figura 3 mostra a tela para criar um novo mapeamento de classe. Nesta tela basta indicar o nome da classe, o nome da tabela e o esquema do banco de dados já existente, que esta tabela estará.

A Figura 4 mostra a tela referente à funcionalidade *Create New Property*. Nesta tela se indica o tipo de mapeamento (o qual pode ser *None*, ou seja, não será mapeada, ou tipos padrões como key, alternate key, field, etc). Na sequência, é possível indicar propriedades gerais que um atributo em uma tabela pode ter (a lista varia de acordo com o tipo de mapeamento selecionado), bem como o nome da propriedade. O campo *Reference* indica uma referência a um tipo de objeto, sendo possível selecionar este tipo de objeto no campo *Type* para estabelecer um relacionamento. Por fim, informa-se o tipo de dado e os dois últimos *check boxes* definem,

respectivamente,se a propriedade pode ser nula e o ultimo serve exclusivamente para classes de programas que utilizam o padrão de programação *MVVM (Model View View-Model)*.

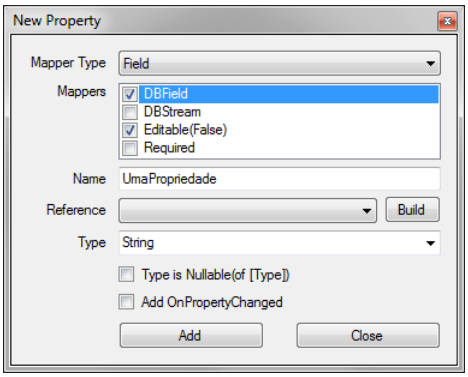


Figura 4. Tela *Create New Property*

A Figura 5 mostra a tela para se fazer comparações entre o esquema de um banco de dados com as classes mapeadas do sistema, para fins de criação e atualização das propriedades. No DSN é indicado um ODBC já criado previamente no Windows para se conectar com o SGBD. No botão *New* DSN é possível criar este ODBC. Após indicada a conexão, basta clicar no botão *Read DB*. O usuário será questionado se deseja fazer a comparação entre as tabelas do banco e as suas classes.

Na visualização de tabelas é mostrado em vermelho as tabelas que não tem vinculo com alguma classe; em laranja as que têm vinculo, mas estão diferentes em termos de propriedades mapeadas; e em verde as que estão equivalentes.

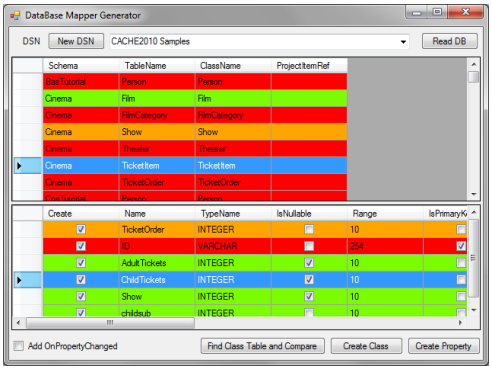


Figura 5. Tela correspondente à *Read a Database to Create Full Class*

A Figura 6 mostra a tela correspondente à funcionalidade *Create Queries Based on Models*, que é um importante diferencial do *DBClassMapper*. Esta tela permite a definição de consultas baseadas nas classes de objetos já mapeados. No lado esquerdo da tela existe a lista de classes que se relacionam com o nodo selecionado, sendo que o primeiro nodo é o primeiro a ser definido a partir da lista de todas as classes mapeadas. Ao centro estão todos os nodos e embaixo ficam as listas de filtros, ordenações e agregações, além do próprio código gerado para a consulta.

Clicando com o botão direito em cima de um nodo é possível definir os filtros, ordenações, *alias* e tipo de join que será realizado (*inner, left, right*)

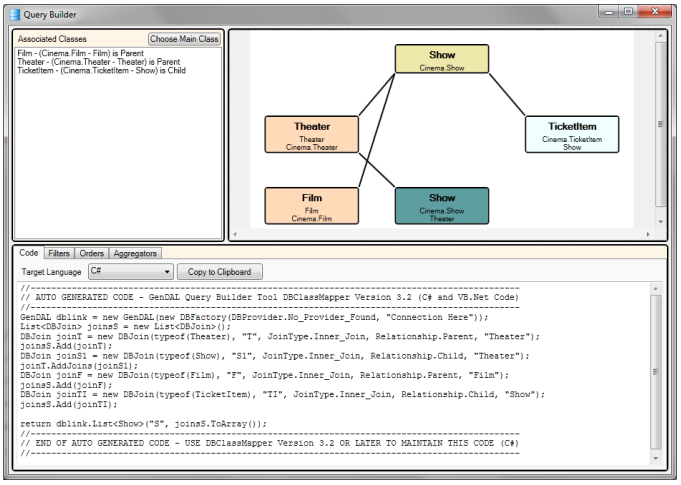


Figura 7. Tela *Create Queries Based on Models*

3. Conclusão

Este artigo apresenta *DBClassMapper*, uma ferramenta gráfica para um usuário desenvolvedor de aplicações orientadas a objetos que integra diversas funcionalidades relacionadas ao processo de mapeamento objeto-relacional. Ela foi especificamente desenvolvida para dar suporte ao *ORM Gendal* com qualquer SGBD.

A principal contribuição d*o DBClassMapper*, se comparada com ferramentas similares, é justamente o fato de disponibilizar diversas funcionalidades de outras já existentes integradas em uma única ferramenta, além de ser a primeira ferramenta para o *ORM Gendal* e uma das poucas (talvez a única) existente para um *ORM* para o *.Net* no *Visual Studio 2010*. Outra inovação importante da ferramenta é o gerador interativo de consultas, funcionalidade não disponível em outros *ORMs.*.

Trabalhos futuros incluem a possibilidade de à partir das classes gerar as tabelas e campos do banco de dados, uma vez que, atualmente o esquema do banco deve existir a *priori*, experimentos de usabilidade no gerador de consultas e a disponibilidade da ferramenta em outras plataformas.

Referências

http://msdn.microsoft.com/en-us/data/ee712907, Acesso em: 24/06/2012

http://www.hibernate.org/subprojects/tools.html, Acesso em: 24/06/2012

Ambler, Scott W. ***Agile Database Techniques***. 1.ed. Nova Yorque: Wiley & Sons, 2003.

http://pt.wikipedia.org/wiki/ORM, Acessado em: 24/06/2012

APÊNDICE C – Código fonte da aplicação

AssociationInfo.vb

```
Imports System.Collections.Generic
Imports System.Xml.Serialization

<Serializable()> _
Public Class AssociationInfo

    Public Property PropertyName As String
    Public Property ClassName As String
    Public Property Schema As String
    Public Property Table As String
    Public Property AssociatingKeys As New List(Of AssociationKey)
    Public Property IsChild As Boolean = False

    Public ReadOnly Property FullName As String
        Get
            If PropertyName = "" Then
                Return String.Format("{0}.{1}", Schema, Table)
            Else
                If IsChild Then
                    Return String.Format("{0}.{1}{2}{3}", Schema, Table, vbCrLf,
PropertyName)
                Else
                    Return String.Format("{0}{1}{2}.{3}", PropertyName, vbCrLf, Schema,
Table)
                End If
            End If
        End Get
    End Property

    Public ReadOnly Property GetKey As String
        Get
            Return String.Format("{0}{1}{2}", Schema, Table, ClassName)
        End Get
    End Property

    Public ReadOnly Property GetNodeKey As String
        Get
            Return String.Format("{0}{1}{2}{3}{4}", Schema, Table, ClassName,
PropertyName, IsChild.ToString)
        End Get
    End Property

    Public ReadOnly Property Description As String
        Get
            Dim S As String = Schema
            If S <> "" Then S = String.Concat(S, ".")
            Return String.Format("{0} - ({1}{2} - {3}) is {4}", ClassName, S, Table,
PropertyName, IIf(IsChild, "Child", "Parent"))
        End Get
    End Property

    Public Overrides Function ToString() As String
```

```
        Dim S As String = Schema
        If S <> "" Then S = String.Concat(S, ".")
        Return String.Format("{0} - ({1}{2})", ClassName, S, Table)
    End Function

End Class
```

AssociationKey.vb

```
<Serializable()> _
Public Class AssociationKey

    Public Property ThisKey As String
    Public Property OtherKey As String

End Class
```

ClassInfo.vb

```
Imports System.Collections.Generic

<Serializable()> _
Public Class ClassInfo
    Public Property Schema As String
    Public Property Table As String
    Public Property ClassName As String
    Public Property Props As New List(Of PropInfo)

    Public ReadOnly Property FullName As String
        Get
            Return String.Format("{0}.{1}", Schema, Table)
        End Get
    End Property

    Public ReadOnly Property GetKey As String
        Get
            Return String.Format("{0}{1}{2}", Schema, Table, ClassName)
        End Get
    End Property

    Public Overrides Function ToString() As String
        Dim S As String = Schema
        If S <> "" Then S = String.Concat(S, ".")
        Return String.Format("{0} - ({1}{2})", ClassName, S, Table)
    End Function

End Class
```

ClassNode.vb

```
Imports System.Windows
Imports System.Collections.Generic
Imports DBManager
Imports System.Text
Imports DBManager.DBImprover
```

```vb
Imports System.Collections.ObjectModel
Imports DBManager.DBWarder
Imports System.Windows.Media

Public Class ClassNode
    Public Property UIElement As UIElement
    Public Property ClassInfo As ClassInfo
    Public Property AssociationInfo As AssociationInfo
    Public Property Parents As New List(Of ClassNode)
    Public Property Childs As New List(Of ClassNode)

    Public Property Relationship As Relationship = Relationship.Parent
    Public Property JoinType As JoinType = JoinType.Left_Outer_Join
    Public Property SQLAlias As String
    Public Property Filters As New ObservableCollection(Of ToolDBFilter)
    Public Property Orders As New ObservableCollection(Of Order)
    Public Property SpecifiedFields As Boolean = False
    Public Property NodeColor As Brush

    Public Sub New(ByRef UIElement As UIElement, ByVal ClassInfo As ClassInfo, ByVal
Relationship As Relationship)
        Me.UIElement = UIElement
        Me.ClassInfo = ClassInfo
        Me.Relationship = Relationship
        Me.JoinType = JoinType.Inner_Join

        GenerateAlias()
    End Sub

    Public Sub New(ByRef UIElement As UIElement, ByVal ClassInfo As ClassInfo, ByVal
AssociationInfo As AssociationInfo, Relationship As Relationship)
        Me.UIElement = UIElement
        Me.ClassInfo = ClassInfo
        Me.AssociationInfo = AssociationInfo
        Me.Relationship = Relationship
        Me.JoinType = JoinType.Inner_Join

        GenerateAlias()
    End Sub

    Private Sub GenerateAlias()
        Dim T As New StringBuilder

        For Each C As Char In ClassInfo.ClassName
            If Char.IsUpper(C) Then
                T.Append(C)
            End If
        Next

        If T.Length > 0 Then
            SQLAlias = T.ToString
        End If
    End Sub

    Public Overrides Function ToString() As String
        Return String.Format("{0} - {1}", ClassInfo.FullName, SQLAlias)
    End Function
End Class
```

DBSchema.vb

```
Imports System.Data.Odbc
Imports System.Collections.Generic

Public Class DBSchema

    Private _Conn As OdbcConnection
    Private _Tables As New List(Of TableInfo)

    Public ReadOnly Property OdbcConnection As OdbcConnection
        Get
            Return _Conn
        End Get
    End Property

    Public ReadOnly Property Tables As List(Of TableInfo)
        Get
            Return _Tables
        End Get
    End Property


    Public Sub New(ByVal ConnectionString As String)
        Try
            _Conn = New OdbcConnection(ConnectionString)
            _Conn.Open()
            _Conn.Close()
        Catch ex As Exception
            Throw New Exception(String.Format("Unable to Connect to Database using {0} as
the ConnectionString. Check InnerException.", ConnectionString), ex)
        End Try

        ReloadDBSchema()
    End Sub

    Public Sub ReloadDBSchema()
        Try
            RefreshTables()
        Catch ex As Exception
            Throw New Exception("Unable to get the Schemas from DataBase. Check
InnerException.", ex)
        End Try
    End Sub

    Private Sub RefreshTables()
        Dim DT As DataTable
        Dim LastSchema As String = ""
        Dim LastTable As String = ""
        Dim TI As New TableInfo
        Dim FI As New FieldInfo
        Dim Item1 As String
        Dim Item2 As String

        _Conn.Open()
        DT = _Conn.GetSchema("Columns")
        _Tables.Clear()
```

```
        For Each R As DataRow In DT.Rows
            Item1 = CStr(R.Item(1))
            Item2 = CStr(R.Item(2))

            If LastSchema <> Item1 Or LastTable <> Item2 Then
                LastSchema = Item1
                LastTable = Item2
                TI = New TableInfo
                TI.Schema = Item1
                TI.TableName = Item2
                TI.ClassName = Item2
                _Tables.Add(TI)
            End If

            FI = New FieldInfo
            FI.Name = CStr(R.Item(3))
            FI.TypeName = CStr(R.Item(5))
            FI.Range = CDbl(R.Item(6))
            FI.IsNullable = CBool(R.Item(10))
            TI.Fields.Add(FI)
        Next

        _Conn.Close()
    End Sub

End Class
```

DLLInfo.vb

```
Imports System.Collections.Generic

<Serializable()> _
Public Class DLLInfo

    Public Property DLLPath As String
    Public Property Mappeds As List(Of FillComboList)

End Class
```

DocWriter.vb

```
Imports EnvDTE
Imports EnvDTE80
Imports System.Text

Public Class DocWriter

    Public Property AppObject As DTE2
    Public Property InstallFolder As String

    Public ReadOnly Property FileExtension As String
        Get
            Return
AppObject.ActiveDocument.FullName.Substring(AppObject.ActiveDocument.FullName.LastIndexOf
(".") + 1)
```

```
        End Get
    End Property

    Public Sub New(ByVal AppObject As DTE2, ByVal InstallFolder As String)
        Me.AppObject = AppObject
        Me.InstallFolder = InstallFolder
    End Sub

    Public Function CreateClass(ByVal Table As TableInfo) As ProjectItem
        If FileExtension = "vb" Then
            Return CreateVBClass(Table)
        ElseIf FileExtension = "cs" Then
            Return CreateCSClass(Table)
        Else
            Throw New ApplicationException("Only VB and C# files are supported.")
        End If
    End Function

    Public Function CreateCSClass(ByVal Table As TableInfo) As ProjectItem
        Dim Proj As Project = AppObject.DTE.ActiveDocument.ProjectItem.ContainingProject
        Dim Sol As Solution = Proj.DTE.Solution
        Dim NewClass As ProjectItem
        Dim TDoc As TextDocument
        Dim Edit As EditPoint

        Try
            'Criar Classe
            NewClass = CreateNewCSClass(Table.ClassName)
            NewClass.Open()
            NewClass.Document.Activate()

            'Criar Ponto de Edição
            TDoc = CType(NewClass.Document.Object("TextDocument"), TextDocument)
            Edit = TDoc.StartPoint.CreateEditPoint

            'Criar Mapeamento
            Edit.FindPattern("XNAMESPACE")
            Edit.Delete(10)
            Edit.Insert(AppObject.ActiveDocument.ProjectItem.ContainingProject.Name)

            'Criar Mapeamento
            Edit.FindPattern("//MAP")
            Edit.Delete(5)
            Edit.Insert(String.Format("[DBTable({0})]", MountTableMap(Table.Schema,
Table.TableName)))

            'Renomear Classe
            Edit.FindPattern("Class1")
            Edit.Delete(6)
            Edit.Insert(Table.ClassName)

            MakeFieldsRegion()
            MakePropsRegions()

            NewClass.Document.Save()
            NewClass.Save()

            Return NewClass
```

```
        Catch ex As Exception
            Throw New Exception(String.Format("Error on creating new class.{0}{0}{1}",
vbCrLf, ex.Message), ex)
        End Try
    End Function

    Public Function CreateVBClass(ByVal Table As TableInfo) As ProjectItem
        Dim Proj As Project = AppObject.DTE.ActiveDocument.ProjectItem.ContainingProject
        Dim Sol As Solution = Proj.DTE.Solution
        Dim NewClass As ProjectItem
        Dim TDoc As TextDocument
        Dim Edit As EditPoint

        Try
            'Criar Classe
            NewClass = CreateNewVBClass(Table.ClassName)
            NewClass.Open()
            NewClass.Document.Activate()

            'Criar Ponto de Edição
            TDoc = CType(NewClass.Document.Object("TextDocument"), TextDocument)
            Edit = TDoc.StartPoint.CreateEditPoint

            'Criar Mapeamento
            Edit.FindPattern("'MAP")
            Edit.Delete(4)
            Edit.Insert(String.Format("<DBTable({0})> _", MountTableMap(Table.Schema,
Table.TableName)))

            'Renomear Classe
            Edit.FindPattern("Class1")
            Edit.Delete(6)
            Edit.Insert(Table.ClassName)

            MakeFieldsRegion()
            MakePropsRegions()

            NewClass.Document.Save()
            NewClass.Save()

            Return NewClass
        Catch ex As Exception
            Throw New Exception(String.Format("Error on creating new class.{0}{0}{1}",
vbCrLf, ex.Message), ex)
        End Try
    End Function

    Private Function CreateNewCSClass(ByVal ClassName As String) As ProjectItem
        Dim SelItem As ProjectItem = AppObject.SelectedItems.Item(1).ProjectItem
        Dim Result As ProjectItem

        If SelItem.ProjectItems.Count = 0 Then
            'Não é pasta, então pega parent (que deve ser uma pasta ou projeto) e
adiciona
            Result =
SelItem.Collection.Parent.ProjectItems.AddFromTemplate(String.Format("{0}\csclass.cs",
_InstallFolder), String.Format("{0}.cs", ClassName))
        Else
```

```
            'É uma pasta, então adiciona
            Result = SelItem.ProjectItems.AddFromTemplate(String.Format("{0}\csclass.cs",
_InstallFolder), String.Format("{0}.cs", ClassName))
        End If

        Return Result
    End Function

    Private Function CreateNewVBClass(ByVal ClassName As String) As ProjectItem
        Dim SelItem As ProjectItem = AppObject.SelectedItems.Item(1).ProjectItem
        Dim Result As ProjectItem

        If SelItem.ProjectItems.Count = 0 Then
            'Não é pasta, então pega parent (que deve ser uma pasta ou projeto) e
adiciona
            Result =
SelItem.Collection.Parent.ProjectItems.AddFromTemplate(String.Format("{0}\vbclass.vb",
_InstallFolder), String.Format("{0}.vb", ClassName))
        Else
            'É uma pasta, então adiciona
            Result = SelItem.ProjectItems.AddFromTemplate(String.Format("{0}\vbclass.vb",
_InstallFolder), String.Format("{0}.vb", ClassName))
        End If

        Return Result
    End Function

    Private Function MountTableMap(Optional ByVal SchemaName As String = "", Optional
ByVal TableName As String = "") As String
        Dim Result As New StringBuilder

        If SchemaName <> "" Then
            Result.Append(String.Format("""{0}""", SchemaName))
        End If

        If TableName <> "" Then
            Result.Append(String.Format(", ""{0}""", TableName))
        End If

        Return Result.ToString
    End Function

    Public Sub CreateProperty(ByVal Field As FieldInfo, Optional ByVal
CreateMapAttributes As Boolean = True, Optional ByVal AddOnPropertyChanged As Boolean =
False)
        If FileExtension = "vb" Then
            CreateVBProperty(Field, CreateMapAttributes, AddOnPropertyChanged)
        ElseIf FileExtension = "cs" Then
            CreateCSProperty(Field, CreateMapAttributes, AddOnPropertyChanged)
        Else
            Throw New ApplicationException("Only VB and C# files are supported.")
        End If
    End Sub

    Public Sub CreateVBProperty(ByVal Field As FieldInfo, Optional ByVal
CreateMapAttributes As Boolean = True, Optional ByVal AddOnPropertyChanged As Boolean =
False)
        Dim TextDoc As TextDocument
```

```
        Dim Edit As EditPoint
        Dim PropType As String = FixDBTypeTOVBType(Field.TypeName)
        Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name

        ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

        If Field.IsNullable And IsPrimitiveType(PropType) Then
            PropType = String.Format("Nullable(Of {0})", PropType)
        End If

        TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
        Edit = TextDoc.StartPoint.CreateEditPoint

        'Private Fields Making
        '-------------------------------------------------------------------------------
---
        MakeFieldsRegion()
        Edit.FindPattern(String.Format("#Region ""{0}_Fields""", ClassName))
        Edit.FindPattern("#End Region")
        Edit.LineUp()
        Edit.Indent()
        Edit.Insert(String.Format("Private _{0} as {1}", Field.Name, PropType))
        Edit.Insert(vbCrLf)
        '-------------------------------------------------------------------------------
---

        'Public Properties Making
        '-------------------------------------------------------------------------------
---
        MakePropsRegions()
        Edit.FindPattern(String.Format("#Region ""{0}_Properties""", ClassName))
        Edit.FindPattern("#End Region")
        Edit.LineUp()

        If CreateMapAttributes Then
            GenerateAttributes(Field, PropType, Edit)
            Edit.Insert(vbCrLf)
            Edit.Indent()
        End If

        Edit.Insert(String.Format("Public Property {0} as {1}", Field.Name, PropType))
        Edit.Insert(vbCrLf)
        Edit.Indent(, 2)
        Edit.Insert("Get")
        Edit.Insert(vbCrLf)
        Edit.Indent(, 3)
        Edit.Insert(String.Format("Return _{0}", Field.Name))
        Edit.Insert(vbCrLf)
        Edit.Indent(, 2)
        Edit.Insert("End Get")
        Edit.Insert(vbCrLf)
        Edit.Indent(, 2)
        Edit.Insert(String.Format("Set(ByVal value As {0})", PropType))

        Edit.Insert(vbCrLf)
        Edit.Indent(, 3)
        Edit.Insert(String.Format("_{0} = value", Field.Name))
```

```
        If AddOnPropertyChanged Then
            Edit.Insert(vbCrLf)
            Edit.Indent(, 3)
            Edit.Insert(String.Format("OnPropertyChanged(Me, Function(T) Me.{0})",
Field.Name))
        End If

        Edit.Insert(vbCrLf)
        Edit.Indent(, 2)
        Edit.Insert("End Set")
        Edit.Insert(vbCrLf)
        Edit.Indent()
        Edit.Insert("End Property")
        Edit.Insert(vbCrLf)
        '-------------------------------------------------------------------------------
---
    End Sub

    Public Sub CreateCSProperty(ByVal Field As FieldInfo, Optional ByVal
CreateMapAttributes As Boolean = True, Optional ByVal AddOnPropertyChanged As Boolean =
False)
        Dim TextDoc As TextDocument
        Dim Edit As EditPoint
        Dim PropType As String = FixDBTypeTOCSType(Field.TypeName)
        Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name
        Dim PrivateFieldName As String = String.Concat(Field.Name(0).ToString.ToLower,
Field.Name.Substring(1))

        ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

        If Field.IsNullable And IsPrimitiveType(PropType) Then
            PropType = String.Format("Nullable<{0}>", PropType)
        End If

        TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
        Edit = TextDoc.StartPoint.CreateEditPoint

        'Private Fields Making
        '-------------------------------------------------------------------------------
---
        MakeFieldsRegion()
        Edit.FindPattern(String.Format("#region ""{0}_Fields""", ClassName))
        Edit.FindPattern("#endregion")
        Edit.LineUp()
        Edit.Indent(, 2)
        Edit.Insert(String.Format("private {0} {1};", PropType, PrivateFieldName))
        Edit.Insert(vbCrLf)
        '-------------------------------------------------------------------------------
---

        'Public Properties Making
        '-------------------------------------------------------------------------------
---
        MakePropsRegions()
        Edit.FindPattern(String.Format("#region ""{0}_Properties""", ClassName))
        Edit.FindPattern("#endregion")
        Edit.LineUp()
```

```
            If CreateMapAttributes Then
                GenerateAttributes(Field, PropType, Edit)
                Edit.Insert(vbCrLf)
                Edit.Indent()
            End If

            Edit.Indent()
            Edit.Insert(String.Format("public {0} {1}", PropType, Field.Name))
            Edit.Insert(vbCrLf)
            Edit.Indent(, 2)
            Edit.Insert("{")
            Edit.Insert(vbCrLf)
            Edit.Indent(, 3)
            Edit.Insert(String.Format("get {{ return {0}; }}", PrivateFieldName))
            Edit.Insert(vbCrLf)
            Edit.Indent(, 3)
            Edit.Insert(String.Format("set {{ {0} = value;", PrivateFieldName))

            If AddOnPropertyChanged Then
                Edit.Insert(vbCrLf)
                Edit.Indent(, 4)
                Edit.Insert(String.Format("OnPropertyChanged(Me, T => Me.{0});", Field.Name))
                Edit.Insert(vbCrLf)
                Edit.Indent(, 3)
                Edit.Insert("}")
            Else
                Edit.Insert(" }")
            End If

            Edit.Insert(vbCrLf)
            Edit.Indent(, 2)
            Edit.Insert("}")
            Edit.Insert(vbCrLf)
            '--------------------------------------------------------------------------------
---
        End Sub

        Public Sub MakePropsRegions()
            If FileExtension = "vb" Then
                MakeVBPropsRegions()
            ElseIf FileExtension = "cs" Then
                MakeCSPropsRegions()
            Else
                Throw New ApplicationException("Only VB and C# files are supported.")
            End If
        End Sub

        Public Sub MakeVBPropsRegions()
            Dim FindRes As Boolean
            Dim Edit As EditPoint
            Dim TextDoc As TextDocument
            Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name

            ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

            TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("Class " & ClassName)
```

```
        FindRes = Edit.FindPattern(String.Format("#Region ""{0}_Properties""",
ClassName))

        If Not FindRes Then
            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("Class " & ClassName)

            Edit.FindPattern(String.Format("#Region ""{0}_Fields""", ClassName))
            Edit.FindPattern("#End Region")
            Edit.LineDown()
            Edit.EndOfLine()
            Edit.Insert(vbCrLf)
            Edit.Insert(String.Format("#Region ""{0}_Properties""", ClassName))
            Edit.Insert(vbCrLf)
            Edit.Insert(vbCrLf)
            Edit.Insert("#End Region")

            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("Class " & ClassName)
            Edit.FindPattern(String.Format("#Region ""{0}_Properties""", ClassName))
        End If
    End Sub

    Public Sub MakeCSPropsRegions()
        Dim FindRes As Boolean
        Dim Edit As EditPoint
        Dim TextDoc As TextDocument
        Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name

        ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

        TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
        Edit = TextDoc.StartPoint.CreateEditPoint
        Edit.FindPattern("class " & ClassName)
        FindRes = Edit.FindPattern(String.Format("#region ""{0}_Properties""",
ClassName))

        If Not FindRes Then
            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("class " & ClassName)

            Edit.FindPattern(String.Format("#region ""{0}_Fields""", ClassName))
            Edit.FindPattern("#endregion")
            Edit.LineDown()
            Edit.EndOfLine()
            Edit.Insert(vbCrLf)
            Edit.Indent()
            Edit.Insert(String.Format("#region ""{0}_Properties""", ClassName))
            Edit.Insert(vbCrLf)
            Edit.Insert(vbCrLf)
            Edit.Indent()
            Edit.Insert("#endregion")

            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("class " & ClassName)
            Edit.FindPattern(String.Format("#region ""{0}_Properties""", ClassName))
        End If
    End Sub
```

```
Public Sub MakeFieldsRegion()
    If FileExtension = "vb" Then
        MakeVBFieldsRegion()
    ElseIf FileExtension = "cs" Then
        MakeCSFieldsRegion()
    Else
        Throw New ApplicationException("Only VB and C# files are supported.")
    End If
End Sub

Public Sub MakeVBFieldsRegion()
    Dim FindRes As Boolean
    Dim Edit As EditPoint
    Dim TextDoc As TextDocument
    Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name

    ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

    TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
    Edit = TextDoc.StartPoint.CreateEditPoint
    Edit.FindPattern(String.Format("Class {0}", ClassName))
    FindRes = Edit.FindPattern(String.Format("#Region ""{0}_Fields""", ClassName))

    If Not FindRes Then
        Edit = TextDoc.StartPoint.CreateEditPoint
        Edit.FindPattern(String.Format("Class {0}", ClassName))

        Edit.LineDown()
        Edit.EndOfLine()
        Edit.Insert(vbCrLf)
        Edit.Insert(String.Format("#Region ""{0}_Fields""", ClassName))
        Edit.Insert(vbCrLf)
        Edit.Insert(vbCrLf)
        Edit.Insert("#End Region")
        Edit.Insert(vbCrLf)
        Edit.Insert(vbCrLf)

        Edit = TextDoc.StartPoint.CreateEditPoint
        Edit.FindPattern("Class " & ClassName)
        Edit.FindPattern(String.Format("#Region ""{0}_Fields""", ClassName))
    End If
End Sub

Public Sub MakeCSFieldsRegion()
    Dim FindRes As Boolean
    Dim Edit As EditPoint
    Dim TextDoc As TextDocument
    Dim ClassName As String = AppObject.ActiveDocument.ProjectItem.Name

    ClassName = ClassName.Substring(0, InStr(ClassName, ".") - 1)

    TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
    Edit = TextDoc.StartPoint.CreateEditPoint
    Edit.FindPattern(String.Format("class {0}", ClassName))
    FindRes = Edit.FindPattern(String.Format("#region ""{0}_Fields""", ClassName))

    If Not FindRes Then
```

```
            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern(String.Format("class {0}", ClassName))

            Edit.LineDown()
            Edit.EndOfLine()
            Edit.Insert(vbCrLf)
            Edit.Indent()
            Edit.Insert(String.Format("#region ""{0}_Fields""", ClassName))
            Edit.Insert(vbCrLf)
            Edit.Insert(vbCrLf)
            Edit.Indent()
            Edit.Insert("#endregion")
            Edit.Insert(vbCrLf)
            Edit.Insert(vbCrLf)

            Edit = TextDoc.StartPoint.CreateEditPoint
            Edit.FindPattern("class " & ClassName)
            Edit.FindPattern(String.Format("#region ""{0}_Fields""", ClassName))
        End If
    End Sub

    Private Sub GenerateAttributes(ByVal Field As FieldInfo, ByVal FixedType As String,
ByRef Edit As EditPoint)
        Dim FCLType As FillComboList = Field.FCLType
        Dim Att As String
        Dim AGField As String = ""
        Dim AGFilter As String = ""

        For Each F As FillComboList In Field.Mappers
            Att = ""

            Select Case F.Key
                Case "AssociationAttribute"
                    If FCLType Is Nothing Then
                        'Att = String.Format("<Association(""{0}FK"", ""{1}"", ""{2}"")>
_", Field.Name, FixedType, "[OtherKeys]")
                        Att = String.Format("<Association(""{0}FK"", ""{0}"", ""{1}"")>
_", Field.Name, "[OtherKeys]")
                    Else
                        'Att = String.Format("<Association(""{0}FK"", ""{1}"", ""{2}"")>
_", Field.Name, FCLType.Data, FCLType.ObjectKeys)
                        Att = String.Format("<Association(""{0}FK"", ""{0}"", ""{1}"")>
_", Field.Name, FCLType.ObjectKeys)
                    End If

                Case "DBDateTime"
                    If FixedType <> "DateTime" AndAlso FixedType <> "DBType.DBDateTime"
Then
                        Att = String.Format("<DBDateTime({0})> _", FixedType)
                    Else
                        Att = CStr(F.Data)
                    End If

                Case "StringLength"
                    If FixedType.ToLower = "string" Then
                        Att = String.Format("<StringLength({0}, ErrorMessage:=""Property
{1} exceeded it's limits(0-{0})."")> _", Field.Range, Field.Name)
                    End If
```

```
                Case "AGField"
                    AGField = ", AutoGenerateField := true"

                Case "AGFilter"
                    AGFilter = ", AutoGenerateFilter := true"

                Case Else
                    Att = CStr(F.Data)

            End Select

            If Not String.IsNullOrEmpty(Att) Then
                Edit.Insert(vbCrLf)
                Edit.Indent()

                If FileExtension = "cs" Then
                    Att = Att.Replace("<", "[").Replace(">", "]").Replace(" _",
"").Replace(":=", "=").Replace("False", "false")
                    Edit.Indent()
                End If

                Edit.Insert(Att)
            End If
        Next

        Edit.Insert(vbCrLf)
        Edit.Indent()

        Att = String.Format("<Display(Name := ""{0}"", ShortName := ""{0}""{1}{2})> _",
Field.Name, AGField, AGFilter)

        If FileExtension = "cs" Then
            Att = Att.Replace("<", "[").Replace(">", "]").Replace(" _", "").Replace(":=",
"=").Replace("False", "false")
            Edit.Indent()
        End If

        Edit.Insert(Att)
    End Sub

    Public Function FixDBTypeTOVBType(ByVal DBType As String) As String
        Dim Result As String
        Dim TypeName As String

        TypeName = DBType.ToUpper

        Select Case TypeName

            'BOOLEANS
            Case "BIT"
                Result = "Boolean"

                'INTEGERS
            Case "TINYINT"
                Result = "Byte"
            Case "SMALLINT"
                Result = "Short"
```

```
        Case "INTEGER"
            Result = "Integer"
        Case "LONG"
            Result = "Long"
        Case "INT"
            Result = "Integer"
        Case "BIGINT"
            Result = "Long"

            'FLOATERS
        Case "FLOAT"
            Result = "Single"
        Case "DOUBLE"
            Result = "Double"
        Case "NUMERIC"
            Result = "Decimal"
        Case "REAL"
            Result = "Single"
        Case "DECIMAL"
            Result = "Decimal"

            'CHARS
        Case "CHAR"
            Result = "Char"
        Case "VARCHAR"
            Result = "String"

            'DATE AND TIMES
        Case "DATE"
            Result = "DateTime"
        Case "TIME"
            Result = "DateTime"
        Case "TIMESTAMP"
            Result = "DateTime"

        Case Else
            Result = DBType

    End Select

    Return Result
End Function

Public Function FixDBTypeTOCSType(ByVal DBType As String) As String
    Dim Result As String
    Dim TypeName As String

    TypeName = DBType.ToUpper

    Select Case TypeName

        'BOOLEANS
        Case "BIT"
            Result = "bool"

            'INTEGERS
        Case "TINYINT"
            Result = "byte"
```

```
            Case "SMALLINT"
                Result = "short"
            Case "INTEGER"
                Result = "int"
            Case "LONG"
                Result = "long"
            Case "INT"
                Result = "int"
            Case "BIGINT"
                Result = "long"

                'FLOATERS
            Case "FLOAT"
                Result = "float"
            Case "DOUBLE"
                Result = "double"
            Case "NUMERIC"
                Result = "decimal"
            Case "REAL"
                Result = "float"
            Case "DECIMAL"
                Result = "decimal"

                'CHARS
            Case "CHAR"
                Result = "char"
            Case "VARCHAR"
                Result = "string"

                'DATE AND TIMES
            Case "DATE"
                Result = "DateTime"
            Case "TIME"
                Result = "DateTime"
            Case "TIMESTAMP"
                Result = "DateTime"

            Case Else
                Result = DBType

        End Select

        Return Result
    End Function

    Public Shared Function IsPrimitiveType(ByVal TypeName As String) As Boolean
        Select Case TypeName.ToUpper
            Case "BOOLEAN"
            Case "BYTE"
                'Date e DateTime não são primitivos realmente
                'Case "DATE"
                'Case "DATETIME"
            Case "DECIMAL"
            Case "DOUBLE"
            Case "INTEGER"
            Case "INT32"
            Case "LONG"
            Case "INT64"
```

```
            Case "SBYTE"
            Case "SHORT"
            Case "INT16"
            Case "SINGLE"
                'String não é um primitivo realmente
                'Case "STRING"
            Case "UINTEGER"
            Case "UINT32"
            Case "ULONG"
            Case "UINT64"
            Case "USHORT"
            Case "UINT16"
            Case Else
                Return False
        End Select

        Return True
    End Function

    Public Function GetTableInfoWithoutFields(Optional ByVal TDoc As TextDocument =
Nothing) As TableInfo
        Dim TI As New TableInfo
        Dim TextDoc As TextDocument
        Dim Edit As EditPoint
        Dim C As String = ""
        Dim Table As New StringBuilder
        Dim Schema As New StringBuilder
        Dim ClassName As New StringBuilder

        If TDoc Is Nothing Then
            TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"),
TextDocument)
        Else
            TextDoc = TDoc
        End If

        Edit = TextDoc.StartPoint.CreateEditPoint

        '-------------------------------------------------------------------------------
---------------------------------------
        ' MONTAGEM DO MAPEAMENTO DA CLASSE PARA UMA TABELA
        '-------------------------------------------------------------------------------
---------------------------------------
        'Procura o mapeamento
        If Not Edit.FindPattern("DBTable(") Then
            Throw New Exception("Class isn't mapped to a Table")
        End If

        'Busca Schema
        Edit.CharRight(8)
        While C <> """" And C <> "," And C <> ">" And C <> "]"
            Edit.CharRight()
            Schema.Append(C)
            C = Edit.GetText(1)
        End While

        'Se C for uma virgula é porque o esquema não esta definido explicitamente
```

```
        'Caso não for virgula tem que procurar a virgula, caso for ')' não tem nada
definido
        If C = "," Or C = ">" Or C = "]" Then
            Schema.Clear()
        Else
            While C <> "," And C <> ")" And C <> ">" And C <> "]"
                Edit.CharRight()
                C = Edit.GetText(1)
            End While
        End If

        'Se C for ')' então não tem table definido, caso contrario buscar o table
        If C <> ")" And C <> ">" And C <> "]" Then
            'Aqui dentro suponho que estamos no EditPoint onde tem a virgula que separa
Table de Schema no mapeamento (Codigo acima deve garantir isso)
            'Busca o inicio do Table
            While C <> """"
                Edit.CharRight()
                C = Edit.GetText(1)
            End While

            'Busca Schema
            C = ""
            While C <> """"
                Edit.CharRight()
                Table.Append(C)
                C = Edit.GetText(1)
            End While
        End If

        'Procura a definição da classe
        If Not (Edit.FindPattern("Public Class") Or Edit.FindPattern("public class"))
Then
            Throw New Exception("Couldn't find the Class definition in document. Make
sure you are in a VB or C# Class.")
        End If

        'Busca ClassName
        Edit.CharRight(12)
        C = ""
        While C <> vbCrLf OrElse C <> " "
            Edit.CharRight()
            ClassName.Append(C)
            C = Edit.GetText(1)
        End While

        TI.Schema = Schema.ToString
        TI.TableName = Table.ToString
        TI.ClassName = ClassName.ToString
        '-------------------------------------------------------------------------------
---------------------------------------
        ' FIM DA MONTAGEM DO MAPEAMENTO DA CLASSE PARA UMA TABELA
        '-------------------------------------------------------------------------------
---------------------------------------

        Return TI
    End Function
```

```
    Public Function GetFieldInfoWithoutMappings(ByVal FieldToFind As FieldInfo, Optional
ByVal TDoc As TextDocument = Nothing) As FieldInfo
        If FileExtension = "vb" Then
            Return GetVBFieldInfoWithoutMappings(FieldToFind, TDoc)
        ElseIf FileExtension = "cs" Then
            Return GetCSFieldInfoWithoutMappings(FieldToFind, TDoc)
        Else
            Throw New ApplicationException("Only VB and C# files are supported.")
        End If
    End Function

    Public Function GetVBFieldInfoWithoutMappings(ByVal FieldToFind As FieldInfo,
Optional ByVal TDoc As TextDocument = Nothing) As FieldInfo
        Dim FI As FieldInfo
        Dim TextDoc As TextDocument
        Dim Edit As EditPoint
        Dim FullTypeName As String = FieldToFind.TypeName

        If TDoc Is Nothing Then
            TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"),
TextDocument)
        Else
            TextDoc = TDoc
        End If

        Edit = TextDoc.StartPoint.CreateEditPoint
        FullTypeName = FixDBTypeTOVBType(FieldToFind.TypeName)

        If IsPrimitiveType(FullTypeName) And FieldToFind.IsNullable Then
            FullTypeName = String.Format("Nullable(of {0})", FullTypeName)
        End If

        If Edit.FindPattern(String.Format("Public Property {0} as {1}", FieldToFind.Name,
FullTypeName)) Then
            FI = New FieldInfo
            FI.Name = FieldToFind.Name
            FI.TypeName = FieldToFind.TypeName
            FI.IsNullable = FieldToFind.IsNullable
        ElseIf Edit.FindPattern(String.Format("Public Property {0} as ",
FieldToFind.Name)) Then
            FI = New FieldInfo
            FI.Name = FieldToFind.Name
            Edit.CharRight(26)
            FI.TypeName = Edit.GetText(Edit.LineLength - 27)
        Else
            FI = New FieldInfo
        End If

        Return FI
    End Function

    Public Function GetCSFieldInfoWithoutMappings(ByVal FieldToFind As FieldInfo,
Optional ByVal TDoc As TextDocument = Nothing) As FieldInfo
        Dim FI As FieldInfo
        Dim TextDoc As TextDocument
        Dim Edit As EditPoint
        Dim FullTypeName As String = FieldToFind.TypeName
```

```
        If TDoc Is Nothing Then
            TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"),
TextDocument)
        Else
            TextDoc = TDoc
        End If

        Edit = TextDoc.StartPoint.CreateEditPoint
        FullTypeName = FixDBTypeTOVBType(FieldToFind.TypeName)

        If IsPrimitiveType(FullTypeName) And FieldToFind.IsNullable Then
            FullTypeName = String.Format("Nullable<{0}>", FullTypeName)
        End If

        If Edit.FindPattern(String.Format("public {0} {1}", FullTypeName,
FieldToFind.Name)) Then
            FI = New FieldInfo
            FI.Name = FieldToFind.Name
            FI.TypeName = FieldToFind.TypeName
            FI.IsNullable = FieldToFind.IsNullable
        ElseIf Edit.FindPattern(String.Format("{0}", FieldToFind.Name)) Then
            FI = New FieldInfo
            FI.Name = FieldToFind.Name
            Edit.CharRight(7)
            FI.TypeName = Edit.GetText(Edit.LineLength - 8)
        Else
            FI = New FieldInfo
        End If

        Return FI
    End Function

    Public Sub FindPropertyAndSetFocus(ByVal PropertyName As String)
        Dim TextDoc As TextDocument
        Dim Edit As EditPoint

        TextDoc = CType(AppObject.ActiveDocument.Object("TextDocument"), TextDocument)
        Edit = TextDoc.StartPoint.CreateEditPoint
        PropertyName = String.Format("Public Property {0} as ", PropertyName)

        Edit.TryToShow()
        If Edit.FindPattern(PropertyName) Then
            Edit.TryToShow()
        Else
            Throw New Exception(String.Format("Couldn't find the {0} in document.",
PropertyName.Substring(0, PropertyName.Length - 4)))
        End If
    End Sub

    Public Function FindMappedClassInTables(ByRef TablesGrid As DataGridView, ByRef
CodeEl As CodeElement) As DataGridViewRow
        Dim TDoc As TextDocument
        Dim TIClass As TableInfo
        Dim TITable As TableInfo
        Dim Result As DataGridViewRow = Nothing

        If CodeEl.IsCodeType Then
            'Caso projectitem esteja fechado ele é aberto
```

```
            If Not CodeEl.ProjectItem.IsOpen Then CodeEl.ProjectItem.Open()
            'Cria uma referencia de facil acesso ao documento do projectitem se não for
nothing
            If Not CodeEl.ProjectItem Is Nothing AndAlso Not CodeEl.ProjectItem.Document
Is Nothing Then
                TDoc = CType(CodeEl.ProjectItem.Document.Object("TextDocument"),
TextDocument)
                'Busca o mapeamento do DBTable como TableInfo
                Try
                    TIClass = GetTableInfoWithoutFields(TDoc)

                    For Each R As DataGridViewRow In TablesGrid.Rows
                        TITable = DirectCast(R.DataBoundItem, TableInfo)
                        If TIClass.TableName = TITable.TableName AndAlso TIClass.Schema =
TITable.Schema AndAlso TITable.TableName <> "" Then
                            TITable.ProjectItemRef = CodeEl.ProjectItem
                            R.DefaultCellStyle.BackColor = Drawing.Color.LawnGreen
                            Result = R
                            Exit For
                        End If
                    Next
                Catch Ex As Exception
                End Try
            End If
        End If

        Return Result
    End Function
End Class
```

DrawProps.vb

```
Imports System.Windows.Media

Public Class DrawProps
    Public Property Color As Brush = Brushes.BlanchedAlmond
    Public Property LeftReduceMyHalf As Boolean = False
    Public Property LeftReduceMyTotal As Boolean = False
    Public Property LeftMod As Double = 0
    Public Property TopMod As Double = 0

    Public Sub New(ByVal LeftReduceMyHalf As Boolean, ByVal Color As Brush)
        Me.LeftReduceMyHalf = LeftReduceMyHalf
        Me.Color = Color
    End Sub

    Public Sub New(ByVal Color As Brush, ByVal LeftReduceMyTotal As Boolean)
        Me.LeftReduceMyTotal = LeftReduceMyTotal
        Me.Color = Color
    End Sub
End Class
```

FieldInfo.vb

```
Imports System.Collections.Generic
```

```vb
<Serializable()> _
Public Class FieldInfo

    Private _Name As String

    Public Property Create As Boolean = True

    Public Property Name As String
        Get
            Return _Name
        End Get
        Set(ByVal value As String)
            _Name = value
            If value.ToUpper = "ID" Then
                IsPrimaryKey = True
                IsEditable = False
            End If
        End Set
    End Property

    Public Property TypeName As String
    Public Property IsNullable As Boolean
    Public Property Range As Double
    Public Property IsPrimaryKey As Boolean
    Public Property IsAlternateKey As Boolean
    Public Property IsForeignKey As Boolean
    Public Property IsRequired As Boolean
    Public Property IsEditable As Boolean = True
    Public Property IsTimeStamp As Boolean
    Public Property IsConcurrencyCheck As Boolean
    Public Property IsStream As Boolean

    Public Property FCLType As FillComboList
    Public Property Mappers As List(Of FillComboList)

    Public Overrides Function ToString() As String
        Return String.Format("{0} {{{1}}}", Name, TypeName)
    End Function

End Class
```

FillComboList.vb

```vb
<Serializable()> _
Public Class FillComboList

    Public Property Key As String
    Public Property Name As String
    Public Property Data As Object
    Public Property AlwaysChecked As Boolean
    Public Property ObjectKeys As String
    Public Property ClassInfo As ClassInfo

    Public Sub New()

    End Sub
```

```
    Public Sub New(ByVal Key As String, ByVal Name As String, Optional ByVal Data As
Object = Nothing, Optional ByVal AlwaysChecked As Boolean = False, Optional ByVal
ObjectKeys As String = "", Optional ByVal ClassInfo As ClassInfo = Nothing)
        Me.Key = Key
        Me.Name = Name
        Me.Data = Data
        Me.AlwaysChecked = AlwaysChecked
        Me.ObjectKeys = ObjectKeys
        Me.ClassInfo = ClassInfo
    End Sub

    Public Overrides Function ToString() As String
        Return Name
    End Function

End Class
```

Lines.vb

```
Imports System.Windows
Imports System.Collections.Generic

Public Class Lines
    Public Property OwnerNode As ClassNode
    Public Property Owner As UIElement
    Public Property MyLines As New List(Of SlaveLine)
    Public Property TheirLines As New List(Of SlaveLine)
    Public Property OrigPoint As Point

    Public Sub New(Owner As UIElement)
        Me.Owner = Owner
    End Sub

    Public Sub MoveLines()
        Dim X As Double = CDbl(Owner.GetValue(Window.LeftProperty))
        Dim Y As Double = CDbl(Owner.GetValue(Window.TopProperty))
        MoveMyLines(X, Y)
        MoveTheirLines(X, Y)
    End Sub

    Public Sub MoveMyLines(X As Double, Y As Double)
        Dim MyXOffSet As Double
        Dim MyYOffSet As Double
        Dim TheirXOffSet As Double
        Dim TheirYOffSet As Double

        For Each S As SlaveLine In MyLines
            MyXOffSet = 0
            TheirXOffSet = 0
            If CDbl(Owner.GetValue(Window.LeftProperty)) <
CDbl(S.OtherOwner.GetValue(Window.LeftProperty)) Then
                MyXOffSet = CDbl(Owner.GetValue(Window.ActualWidthProperty))
            Else
                TheirXOffSet = CDbl(S.OtherOwner.GetValue(Window.ActualWidthProperty))
            End If

            MyYOffSet = 0
```

```
            TheirYOffSet = 0
            If CDbl(Owner.GetValue(Window.TopProperty)) <
CDbl(S.OtherOwner.GetValue(Window.TopProperty)) Then
                MyYOffSet = CDbl(Owner.GetValue(Window.ActualHeightProperty))
            Else
                TheirYOffSet = CDbl(S.OtherOwner.GetValue(Window.ActualHeightProperty))
            End If

            S.Line.X1 = X + MyXOffSet
            S.Line.Y1 = Y + MyYOffSet
            S.Line.X2 = CDbl(S.OtherOwner.GetValue(Window.LeftProperty)) + TheirXOffSet
            S.Line.Y2 = CDbl(S.OtherOwner.GetValue(Window.TopProperty)) + TheirYOffSet
        Next
    End Sub

    Public Sub MoveTheirLines(X As Double, Y As Double)
        Dim XOffSet As Double
        Dim YOffSet As Double
        Dim TheirXOffSet As Double
        Dim TheirYOffSet As Double

        For Each S As SlaveLine In TheirLines
            XOffSet = 0
            TheirXOffSet = 0
            If CDbl(Owner.GetValue(Window.LeftProperty)) <
CDbl(S.OtherOwner.GetValue(Window.LeftProperty)) Then
                XOffSet = CDbl(Owner.GetValue(Window.ActualWidthProperty))
            Else
                TheirXOffSet = CDbl(S.OtherOwner.GetValue(Window.ActualWidthProperty))
            End If

            YOffSet = 0
            TheirYOffSet = 0
            If CDbl(Owner.GetValue(Window.TopProperty)) <
CDbl(S.OtherOwner.GetValue(Window.TopProperty)) Then
                YOffSet = CDbl(Owner.GetValue(Window.ActualHeightProperty))
            Else
                TheirYOffSet = CDbl(S.OtherOwner.GetValue(Window.ActualHeightProperty))
            End If

            S.Line.X1 = CDbl(S.OtherOwner.GetValue(Window.LeftProperty)) + TheirXOffSet
            S.Line.Y1 = CDbl(S.OtherOwner.GetValue(Window.TopProperty)) + TheirYOffSet
            S.Line.X2 = X + XOffSet
            S.Line.Y2 = Y + YOffSet
        Next
    End Sub
End Class
```

Mapper.vb

```
Imports System.Collections.Generic
Imports VSLangProj
Imports EnvDTE
Imports System.IO
Imports System.Xml.Serialization
Imports EnvDTE80
```

```vb
Public Class Mapper

    Public Shared Function GetMappeds(ByVal Proj As Project, ByVal InstallFolder As
String, Optional ByVal Ref As Reference = Nothing) As List(Of FillComboList)
        'Dim Proj As Project = _AppObject.ActiveDocument.ProjectItem.ContainingProject
        Dim Result As New List(Of FillComboList)
        'Dim Reg As RegistryKey
        Dim AppPath As String
        Dim PPath As String
        Dim SerializePath As String

        'FullPath tem o caminho completo da solution (não do arquivo .sln ou .vbp e sim
da localicação ex.: "C:\VSProj\MeuProjeto\")
        Dim FullPath As String = Proj.Properties.Item("FullPath").Value.ToString()
        'OutputPath tem o caminho parcial no qual o desenvolvedor compilou esse projeto
pela ultima vez (ex.: "\bin\Debug\")
        Dim OutputPath As String =
Proj.ConfigurationManager.ActiveConfiguration.Properties.Item("OutputPath").Value.ToStrin
g()
        'OutputFile tem o nome do ultimo arquivo compilado (ex.: "MeuPrograma.dll")
        Dim OutputFile As String =
Proj.Properties.Item("OutputFileName").Value.ToString()
        'OutputFullName combina tudo que foi pego acima em um caminho válido para o
ultimo arquivo compilado
        Dim OutputFullName As String = Path.Combine(FullPath, OutputPath)
        OutputFullName = Path.Combine(OutputFullName, OutputFile)

        'Caminho onde o DBClassMapper foi instalado
        AppPath = InstallFolder
        'Caminho do aplicativo que será chamado para serializar
        PPath = String.Format("{0}\AssemblyInspector.exe", AppPath)
        'Montagem do caminho onde está o arquivo serializado
        SerializePath = GetSerializePath()

        'Busca mapeados do projeto
        Try
            Result.AddRange(SerializeAndDeserializeMappeds(PPath, OutputFullName,
SerializePath))
        Catch ex As Exception
            MsgBox(String.Format("Erro ao buscar mapeamentos do projeto.{0}{0}{1}",
vbCrLf, ex.Message))
        End Try

        'Busca mapeados da referencia selecionada
        Try
            If Not Ref Is Nothing Then
                Result.AddRange(SerializeAndDeserializeMappeds(PPath, Ref.Path,
SerializePath))
            End If
        Catch ex As Exception
            MsgBox(String.Format("Erro ao buscar mapeamentos da referencia
selecionada.{0}{0}{1}", vbCrLf, ex.Message))
        End Try

        Return Result
    End Function

    ''' <summary>
```

```
    ''' Pega todas as referencias locais do projeto
    ''' </summary>
    ''' <returns>Lista de referencias prontas para preencher um ComboBox</returns>
    ''' <remarks>Não carrega referencias que não são copias locais. Tem esse
comportamento para evitar listar muita DLLs desnecessárias.</remarks>
    Public Shared Function GetReferences(ByVal ApplicationObject As DTE2) As List(Of
FillComboList)
        Dim Proj As VSProject =
DirectCast(ApplicationObject.ActiveDocument.ProjectItem.ContainingProject.Object,
VSProject)
        Dim Result As New List(Of FillComboList)

        For Each Ref As Reference In Proj.References
            'Adiciona as referencias na combo para depois poder buscar as classes
mapeadas
            'A propriedade CopyLocal = True é utilizada para filtrar somente as DLLs
locais
            'retirando assim as DLLs do System e etc as quais não serão utilizadas
            If Ref.CopyLocal Then Result.Add(New FillComboList(Ref.Identity, Ref.Name,
Ref))
        Next

        Return Result
    End Function

    Private Shared Function GetSerializePath() As String
        Dim Path As String = String.Format("{0}\.deltacon\.dbclassmapper",
My.Computer.FileSystem.SpecialDirectories.AllUsersApplicationData)

        If Not Directory.Exists(Path) Then
            Directory.CreateDirectory(Path)
        End If

        Return String.Format("{0}\Mappeds.xml", Path)
    End Function

    Private Shared Function SerializeAndDeserializeMappeds(ByVal PPath As String, ByVal
DLLPath As String, ByVal SerializePath As String) As List(Of FillComboList)
        Dim XS As New XmlSerializer(GetType(DLLInfo))
        Dim P As New System.Diagnostics.Process
        Dim FS As FileStream = Nothing
        Dim DI As DLLInfo
        Dim Result As New List(Of FillComboList)
        Dim Args As String = String.Format("""{0}"" ""{1}""", DLLPath, SerializePath)

        Try
            'Chamada do processo que irá serializar os mapeamentos da DLL passada
            P.StartInfo = New ProcessStartInfo(PPath, Args)
            P.StartInfo.WindowStyle = ProcessWindowStyle.Hidden

            'Inicia o Processo
            If P.Start() Then
                'Aguarda o processo terminar (Feito com While para o ProgressBar
funcionar sem utilizar multi-thread explicitamente)
                'If P.WaitForExit(60000) Then
                While Not P.HasExited
                    System.Windows.Forms.Application.DoEvents()
                    System.Threading.Thread.Sleep(10)
```

```
                End While

                'Se o processo retornar um erro uma exceção é levantada aqui
                If Not P.ExitCode = 0 Then
                    Throw New Exception(String.Format("AssemblyInspector retornou erro ao
tentar serializar as classes mapeadas.{0}{0}Returned Code: {1}", vbCrLf, P.ExitCode))
                End If

                'Deserialização do arquivo serializado e captura dos arquivos mapeados
                FS = New FileStream(SerializePath, FileMode.Open, FileAccess.Read)
                DI = DirectCast(XS.Deserialize(FS), DLLInfo)
                FS.Flush()

                Result = DI.Mappeds
                'Else
                '    Throw New Exception("Excedeu o tempo de espera para carregar os
assemblys.")
                'End If
            Else
                Throw New Exception("Não foi possível iniciar o AssemblyInspector.")
            End If

            Return Result
        Catch ex As Exception
            'Qualquer erro é jogado para quem chamou esta função
            Throw New Exception(ex.Message, ex)
        Finally
            'Verifica se o processo é algo, se for ele verifica se ainda esta aberto, se
estiver ele fecha e por ultimo esvazia
            If Not IsNothing(P) Then
                If Not P.HasExited Then P.Close()
                P = Nothing
            End If
            'Verifica se o FileStream existe para poder fecha-lo
            If Not IsNothing(FS) Then
                Try
                    FS.Close()
                    FS = Nothing
                Catch ex2 As Exception
                End Try
            End If
            'Manda o GC coletar
            GC.Collect()
        End Try
    End Function

End Class
```

PropInfo.vb

```
Imports System.Collections.Generic

<Serializable()> _
Public Class PropInfo

    Public Property Selected As Boolean = True
    Public Property FieldName As String
```

```
    Public Property PropName As String
    Public Property TypeName As String
    Public Property IsPrimaryKey As Boolean
    Public Property IsAlternateKey As Boolean
    Public Property Association As AssociationInfo

    Public Overrides Function ToString() As String
        Dim F As String = FieldName
        If F <> "" Then F = String.Format("({0})", F)
        Return String.Format("{0}{1} - {2}", PropName, F, TypeName)
    End Function

End Class
```

SlaveLine.vb

```
Imports System.Windows
Imports System.Windows.Shapes

Public Class SlaveLine
    Public Property OtherOwner As UIElement
    Public Property Line As Line

    Public Sub New(ByRef OtherOwner As UIElement, ByRef Line As Line)
        Me.OtherOwner = OtherOwner
        Me.Line = Line
    End Sub
End Class
```

TableInfo.vb

```
Imports System.Collections.Generic
Imports EnvDTE

<Serializable()> _
Public Class TableInfo

    Public Property Schema As String
    Public Property TableName As String
    Public Property ClassName As String
    Public Property Fields As New List(Of FieldInfo)
    Public Property ProjectItemRef As ProjectItem

    Public Overrides Function ToString() As String
        Dim SchemaTemp As String

        If Schema = "" Then
            SchemaTemp = ""
        Else
            SchemaTemp = String.Format("{0}.", Schema)
        End If

        Return String.Format("{0}{1}", SchemaTemp, TableName)
    End Function

    Public Function IsEmpty() As Boolean
```

```
        If Schema = "" AndAlso TableName = "" AndAlso Fields.Count = 0 Then
            Return True
        Else
            Return False
        End If
    End Function

End Class
```

ToolDBFilter.vb

```
Imports DBManager.DBImprover
Imports DBManager.DBWarder

Public Class ToolDBFilter
    Inherits DBFilter

    Public Property Var1 As Object
    Public Property Var2 As Object
    Public Property Node As ClassNode

End Class
```

ToolDBOrder.vb

```
Imports DBManager.DBImprover
Imports DBManager.DBWarder

Public Class ToolDBOrder
    Inherits Order

    Public Property Node As ClassNode

    Public Sub New(Field As String, Mode As OrderMode)
        MyBase.New(Field, Mode)
    End Sub

End Class
```

frmAddFilter.vb

```
Imports System.Drawing
Imports DBManager.DBImprover
Imports DBManager.DBWarder

Public Class frmAddFilter

    Private Node As ClassNode
    Private Prop As PropInfo
    Private NewFilter As ToolDBFilter

    Public Sub New(ByRef Node As ClassNode, ByRef SelectedProp As PropInfo, ByRef
NewFilter As ToolDBFilter)

        ' This call is required by the designer.
```

```
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        Me.Node = Node
        Me.Prop = SelectedProp
        Me.NewFilter = NewFilter
        Me.NewFilter.Node = Me.Node
    End Sub

    Private Sub frmAddFilter_FormClosing(sender As Object, e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        If NewFilter.Field = "" Then NewFilter = Nothing
    End Sub

    Private Sub frmAddFilter_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        LoadBase()
    End Sub

    Private Sub LoadBase()
        Me.Size = New Size(Me.Size.Width, 62)

        For Each O As OperatorType In [Enum].GetValues(GetType(OperatorType))
            cmbOperator.Items.Add(O)
        Next

        cmbOperator.SelectedItem = OperatorType.Equal

        txtField.Text = Prop.PropName

        Dim TTP As ToolTip

        TTP = New ToolTip
        TTP.SetToolTip(txtVar1, "Variable or Value to generate the Gendal Code to
Query.")

        TTP = New ToolTip
        TTP.SetToolTip(txtTestVal1, "Value to Test this filter in the Query.")

        TTP = New ToolTip
        TTP.SetToolTip(txtVar2, "Variable or Value to generate the Gendal Code to Query.
This is used in Between filters.")

        TTP = New ToolTip
        TTP.SetToolTip(txtTestVal2, "Value to Test this filter in the Query. This is used
in Between filters.")
    End Sub

    Private Sub btnAddFilter_Click(sender As System.Object, e As System.EventArgs)
Handles btnAddFilter.Click
        If txtVar1.Text = "" Then
            MsgBox("Var 1 is required.")
            txtVar1.Focus()
            Exit Sub
        End If

        If cmbOperator.SelectedItem.Equals(OperatorType.Between) Then
            If txtVar2.Text = "" Then
```

```
                MsgBox("Var 2 is required in between.")
                txtVar2.Focus()
                Exit Sub
            End If
        End If

        NewFilter.Field = txtField.Text
        NewFilter.OperatorType = DirectCast(cmbOperator.SelectedItem, OperatorType)
        NewFilter.Value = txtTestVal1.Text
        NewFilter.Var1 = txtVar1.Text

        If cmbOperator.SelectedItem.Equals(OperatorType.Between) Then
            NewFilter.ValueTo = txtTestVal2.Text
            NewFilter.Var2 = txtVar2.Text
        End If

        Me.Close()
    End Sub

    Private Sub cmbOperator_SelectedIndexChanged(sender As System.Object, e As
System.EventArgs) Handles cmbOperator.SelectedIndexChanged
        If cmbOperator.SelectedItem.Equals(OperatorType.Between) Then
            Me.Size = New Size(Me.Size.Width, 92)
        Else
            Me.Size = New Size(Me.Size.Width, 62)
        End If
    End Sub
End Class
```

frmAddOrder.vb

```
Imports DBManager.DBImprover
Imports DBManager.DBWarder

Public Class frmAddOrder

    Private Node As ClassNode
    Private Prop As PropInfo
    Private NewOrder As ToolDBOrder

    Public Sub New(ByRef Node As ClassNode, ByRef SelectedProp As PropInfo, ByRef
NewOrder As ToolDBOrder)

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        Me.Node = Node
        Me.Prop = SelectedProp
        Me.NewOrder = NewOrder
        Me.NewOrder.Node = Me.Node
    End Sub

    Private Sub frmAddOrder_FormClosing(sender As Object, e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        If NewOrder.Field = "" Then NewOrder = Nothing
```

```
    End Sub

    Private Sub frmAddOrder_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load
        txtField.Text = Prop.PropName

        For Each O As OrderMode In [Enum].GetValues(GetType(OrderMode))
            cmbOrder.Items.Add(O)
        Next

        cmbOrder.SelectedItem = OrderMode.Asc
    End Sub

    Private Sub btnAddOrder_Click(sender As System.Object, e As System.EventArgs) Handles
btnAddOrder.Click
        NewOrder.Field = txtField.Text
        NewOrder.Mode = DirectCast(cmbOrder.SelectedItem, OrderMode)
        Me.Close()
    End Sub
End Class
```

frmDBReader.vb

```
Imports Microsoft.Win32
Imports System.Collections.Generic
Imports EnvDTE80
Imports System.IO
Imports EnvDTE

Public Class frmDBReader

    Private _AppObject As DTE2
    Private _InstallFolder As String
    Private _Schema As DBSchema
    Private _DW As DocWriter
    Private _SchemaIsReady As Boolean

    Public Sub New(ByVal ApplicationObject As DTE2, ByVal InstallFolder As String)

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        _AppObject = ApplicationObject
        _InstallFolder = InstallFolder
        _DW = New DocWriter(_AppObject, _InstallFolder)
    End Sub

    Private Sub frmDBReader_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        Dim MainWin As New frmMainWindow(_AppObject)
        MainWin.Show()
        Me.Dispose()
    End Sub
```

```
    Private Sub frmDBReader_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        grdTables.AutoGenerateColumns = True
        grdFields.AutoGenerateColumns = True
        cmbDSN.Text = My.Settings.LastDSN
        lblCreating.Dock = DockStyle.Fill
        pgbCreating.Dock = DockStyle.Bottom
        chkOnPropertyChanged.Checked = My.Settings.CheckAddPropChange
        GetDSNs()
    End Sub

    Private Sub GetDSNs()
        Dim Reg As RegistryKey
        Dim DSNs As New List(Of String)

        Reg = Registry.LocalMachine.OpenSubKey("Software\ODBC\ODBC.INI\ODBC Data
Sources", True)

        For Each DSN As String In Reg.GetValueNames
            DSNs.Add(DSN)
        Next

        cmbDSN.Items.Clear()
        cmbDSN.Items.AddRange(DSNs.ToArray)
    End Sub

    Private Sub btnNewDSN_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewDSN.Click
        Try
            Me.TopMost = False

Shell(Path.Combine(String.Concat(Environment.GetEnvironmentVariable("windir"), "\"),
"system32\odbcad32.exe"), AppWinStyle.NormalFocus, True)
            GetDSNs()
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.Message)
            Me.TopMost = True
        Finally
            Me.TopMost = True
        End Try
    End Sub

    Private Sub btnReadDB_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReadDB.Click
        ReadDataBase(MsgBox("Deseja gerar a comparação entre as tabelas do banco com as
classes existentes?", vbYesNo) = vbYes)
    End Sub

    Private Sub ReadDataBase(Optional ByVal DoCrossClassesWithTables As Boolean = False)
        Try
            pgbCreating.Style = ProgressBarStyle.Marquee
            pgbCreating.MarqueeAnimationSpeed = 10
            SetCreating(True)
            _SchemaIsReady = False

            System.Windows.Forms.Application.DoEvents()
```

```
            Dim T As New Threading.Thread(New
Threading.ParameterizedThreadStart(AddressOf GetSchema))
            T.IsBackground = True
            T.SetApartmentState(Threading.ApartmentState.MTA)
            T.Start()

            While Not _SchemaIsReady
                System.Windows.Forms.Application.DoEvents()
                Threading.Thread.Sleep(10)
            End While

            grdTables.DataSource = _Schema.Tables

            pgbCreating.Style = ProgressBarStyle.Blocks

            If DoCrossClassesWithTables Then
                grdTables.Refresh()
                System.Windows.Forms.Application.DoEvents()
                CrossClassesWithTables()
            End If
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.Message)
            Me.TopMost = True
        Finally
            SetCreating(False)
        End Try
    End Sub

    Private Sub GetSchema()
        Try
            _Schema = New DBSchema(String.Format("DSN={0}", cmbDSN.Text))
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.Message)
            Me.TopMost = True
        Finally
            _SchemaIsReady = True
        End Try
    End Sub

    Private Sub CrossClassesWithTables()
        Dim Solution As EnvDTE.Solution = _AppObject.Solution
        Dim Count As Long = 0
        Dim TotalCE As Long

        Try
            'Colore tudo de vermelho para depois pintar de verde as que forem encontradas
            For Each R As DataGridViewRow In grdTables.Rows
                R.DefaultCellStyle.BackColor = Drawing.Color.Red
            Next

            For Each P As Project In Solution.Projects
                If P.CodeModel Is Nothing Then
                    'É uma pasta de projeto
                    CheckRecursiveProjectItens(P.ProjectItems)
                Else
                    'É um codemodel de projeto
```

```
                    Count = 0
                    TotalCE = P.CodeModel.CodeElements.Count
                    For Each D As CodeElement In P.CodeModel.CodeElements
                        CheckMapDiffs(grdTables, D)
                        Count += 1
                        pgbCreating.Value = CInt(100 * Count / TotalCE)
                        System.Windows.Forms.Application.DoEvents()
                    Next
                End If
            Next
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.Message)
            Me.TopMost = True
        End Try
    End Sub

    Private Sub CheckMapDiffs(ByVal TableGrid As DataGridView, ByVal D As CodeElement)
        Dim RowTI As DataGridViewRow = _DW.FindMappedClassInTables(grdTables, D)

        If Not RowTI Is Nothing Then
            If Not ClassHasNoSignificantChanges(DirectCast(RowTI.DataBoundItem,
TableInfo).Fields, DirectCast(D.ProjectItem.Document.Object("TextDocument"),
TextDocument)) Then
                RowTI.DefaultCellStyle.BackColor = Drawing.Color.Orange
            End If
        End If
    End Sub

    Private Sub CheckRecursiveProjectItens(ByVal PItens As ProjectItems)
        Dim Count As Long = 0
        Dim TotalCE As Long

        For Each PJ As ProjectItem In PItens
            If Not PJ.SubProject Is Nothing Then
                Try
                    If PJ.SubProject.CodeModel Is Nothing Then
                        CheckRecursiveProjectItens(PJ.SubProject.ProjectItems)
                    Else
                        Count = 0
                        TotalCE = PJ.SubProject.CodeModel.CodeElements.Count
                        For Each D As CodeElement In PJ.SubProject.CodeModel.CodeElements
                            CheckMapDiffs(grdTables, D)
                            Count += 1
                            pgbCreating.Value = CInt(100 * Count / TotalCE)
                            System.Windows.Forms.Application.DoEvents()
                        Next
                    End If
                Catch ex1 As NotImplementedException
                    CheckRecursiveProjectItens(PJ.SubProject.ProjectItems)
                Catch ex2 As Exception
                    Throw ex2
                End Try
            End If
        Next
    End Sub
```

```
    Private Sub grdTables_CellDoubleClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles grdTables.CellDoubleClick
        Dim TI As TableInfo

        If grdTables.SelectedRows.Count > 0 Then
            TI = DirectCast(grdTables.SelectedRows(0).DataBoundItem, TableInfo)
            grdFields.DataSource = TI.Fields
            If Not TI.ProjectItemRef Is Nothing Then
                TI.ProjectItemRef.Open()
                TI.ProjectItemRef.Document.Activate()
                ComparePropertiesWithFields()
            Else
                Me.TopMost = False
                MsgBox("Couldn't find a Class that Matches with this table.")
                Me.TopMost = True
            End If
        End If
    End Sub

    Private Sub grdTables_SelectionChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles grdTables.SelectionChanged
        Dim TI As TableInfo

        If grdTables.SelectedRows.Count > 0 Then
            TI = DirectCast(grdTables.SelectedRows(0).DataBoundItem, TableInfo)
            grdFields.DataSource = TI.Fields
        End If
    End Sub

    Private Sub cmbDSN_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmbDSN.SelectedIndexChanged
        My.Settings.LastDSN = cmbDSN.Text
    End Sub

    Private Sub btnCreateClass_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCreateClass.Click
        Dim TI As TableInfo

        If grdTables.SelectedRows.Count = 0 Then
            MsgBox("Select one table to create a class from it.")
            Exit Sub
        End If

        TI = DirectCast(grdTables.SelectedRows(0).DataBoundItem, TableInfo)

        If TI.TableName = "" Then
            MsgBox("Table without a name isn't valid to map.")
            Exit Sub
        End If

        Try
            SetCreating(True)
            CreateClass(TI, grdTables.SelectedRows(0))
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.ToString)
            Me.TopMost = True
        Finally
```

```
            SetCreating(False)
        End Try
    End Sub

    Private Sub btnCreateProperty_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCreateProperty.Click
        If grdFields.SelectedRows.Count = 0 Then
            MsgBox("Select one field to create a property from it.")
            Exit Sub
        End If

        CreateSelectedProperties()
    End Sub

    Private Sub CreateSelectedProperties()
        Try
            Dim Count As Long = 0
            SetCreating(True)
            For Each R As DataGridViewRow In grdFields.SelectedRows
                Count += 1
                CreateProperty(DirectCast(R.DataBoundItem, FieldInfo))
                R.DefaultCellStyle.BackColor = Drawing.Color.LawnGreen
                pgbCreating.Value = CInt((Count * 100) / grdFields.SelectedRows.Count)
                System.Windows.Forms.Application.DoEvents()
            Next

            'Se tudo estiver Verde então colorir a row da grid da tabela de verde também
            Dim AllGreen As Boolean = True
            For Each R As DataGridViewRow In grdFields.Rows
                If R.DefaultCellStyle.BackColor <> Drawing.Color.LawnGreen Then
                    AllGreen = False
                    Exit For
                End If
            Next

            If AllGreen Then
                grdTables.SelectedRows(0).DefaultCellStyle.BackColor =
Drawing.Color.LawnGreen
            End If
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.ToString)
            Me.TopMost = True
        Finally
            SetCreating(False)
        End Try
    End Sub

    Private Sub SetCreating(ByVal Creating As Boolean)
        pgbCreating.Value = 0
        lblCreating.Visible = Creating
        pgbCreating.Visible = Creating
    End Sub

    Private Sub CreateClass(ByVal TI As TableInfo, ByRef RowTI As DataGridViewRow)
        Dim Total As Long = TI.Fields.Count
        Dim Count As Long = 0
        Dim PI As ProjectItem
```

```
        PI = DirectCast(RowTI.DataBoundItem, TableInfo).ProjectItemRef
        PI = _DW.CreateClass(TI)

        For Each F As FieldInfo In TI.Fields
            If F.Create Then
                CreateProperty(F)
            End If
            Count += 1
            pgbCreating.Value = CInt(100.0 * Count / Total)
            System.Windows.Forms.Application.DoEvents()
        Next

        PI.Save()
        RowTI.DefaultCellStyle.BackColor = Drawing.Color.LawnGreen
        TI.ProjectItemRef = PI
    End Sub

    Private Sub CreateProperty(ByVal FI As FieldInfo)
        Dim FixedType As String = _DW.FixDBTypeTOVBType(FI.TypeName)
        Dim DBType As String

        FI.Mappers = New List(Of FillComboList)
        FI.Mappers.Add(New FillComboList("DBField", "DBField", "<DBField()> _"))

        If FixedType = "String" Then
            FI.Mappers.Add(New FillComboList("StringLength", "StringLength",
String.Format("<StringLength({0}, ErrorMessage:=""Property {1} exceeded it's limits(0-
{0})."")> _", FI.Range, FI.Name)))
        End If

        If FixedType = "DateTime" Then
            Select Case FI.TypeName.ToUpper
                Case "DATE"
                    DBType = "DBType.DBDate"

                Case "TIME"
                    DBType = "DBType.DBTime"

                Case Else
                    DBType = ""

            End Select

            FI.Mappers.Add(New FillComboList("DBDateTime", "DBDateTime",
String.Format("<DBDateTime({0})> _", DBType)))
        End If

        If FI.IsPrimaryKey Then
            FI.Mappers.Add(New FillComboList("KeyAttribute", "Key", "<Key()> _"))
        End If

        If FI.IsAlternateKey Then
            FI.Mappers.Add(New FillComboList("DBAlternateKey", "DBAlternateKey",
"<DBAlternateKey()> _"))
        End If

        If FI.IsForeignKey Then
```

```
            FI.Mappers.Add(New FillComboList("AssociationAttribute", "Association",
"<Association()> _"))
        End If

        If FI.IsRequired Then
            FI.Mappers.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _"))
        End If

        If Not FI.IsEditable Then
            FI.Mappers.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _"))
        End If

        If FI.IsTimeStamp Then
            FI.Mappers.Add(New FillComboList("TimeStampAttribute", "Time Stamp",
"<TimeStamp()> _"))
        End If

        If FI.IsConcurrencyCheck Then
            FI.Mappers.Add(New FillComboList("ConcurrencyCheckAttribute", "Concurrency
Check", "<ConcurrencyCheck()> _"))
        End If

        If FI.IsStream Then
            FI.Mappers.Add(New FillComboList("DBStream", "DBStream", "<DBStream()> _"))
        End If

        _DW.CreateProperty(FI, True, chkOnPropertyChanged.Checked)
    End Sub

    Private Sub chkOnPropertyChanged_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs)
        My.Settings.CheckAddPropChange = chkOnPropertyChanged.Checked
    End Sub

    Private Sub frmDBReader_ResizeBegin(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeBegin
        Me.Opacity = 0.15
    End Sub

    Private Sub frmDBReader_ResizeEnd(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeEnd
        Me.Opacity = 1
    End Sub

    Private Sub btnbtnFindClassTableAndCompare_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnFindClassTableAndCompare.Click
        FindTableAndCompare()
    End Sub

    Private Sub FindTableAndCompare()
        Try
            Dim TI As TableInfo = _DW.GetTableInfoWithoutFields()
            Dim RowTI As TableInfo
            Dim Found As Boolean = False

            For Each D As DataGridViewRow In grdTables.Rows
```

```
                RowTI = DirectCast(D.DataBoundItem, TableInfo)
                If RowTI.TableName = TI.TableName And RowTI.Schema = TI.Schema Then
                    grdTables.ClearSelection()
                    D.Selected = True
                    grdTables.CurrentCell = D.Cells(0)
                    Found = True
                    Exit For
                End If
            Next

            If Found Then
                ComparePropertiesWithFields()
                MsgBox("Similar Table Found and Selected!")
            Else
                MsgBox("Couldn't find a table that looks like the mapped class you are
in.")
            End If
        Catch ex As Exception
            MsgBox(ex.Message)
        End Try
    End Sub

#Region "Comparison same Methods in Two Methods - Need good idea to unify"
    '-------------------------------------------------------------------------------
-------------------------------------------------------------
    ' DIDNT FIND A GOOD IDEA TO DON'T SEPARATE THIS TWO METHODS
    '-------------------------------------------------------------------------------
-------------------------------------------------------------

    Private Sub ComparePropertiesWithFields()
        Dim RowFI As FieldInfo
        Dim FI As FieldInfo

        For Each D As DataGridViewRow In grdFields.Rows
            RowFI = DirectCast(D.DataBoundItem, FieldInfo)
            FI = _DW.GetFieldInfoWithoutMappings(RowFI)

            If RowFI.Name = FI.Name Then
                'Nomes iguais então field já existe
                If RowFI.TypeName = FI.TypeName And RowFI.IsNullable = FI.IsNullable Then
                    'Tipos iguais então field provavelmente é o mesmo mapeado (pode haver
diferenças de mapeamento, mas isto não define que mudou no banco)
                    'SETAR LINHA EM VERDE
                    D.DefaultCellStyle.BackColor = Drawing.Color.LawnGreen
                Else
                    'Tipos diferentes então algo foi mudado no banco de dados
                    'SETAR LINHA EM LARANJADO
                    D.DefaultCellStyle.BackColor = Drawing.Color.Orange
                End If
            Else
                'Não existe este Field definido na classe como propriedade, provavelmente
é um campo novo na base de dados
                'SETAR LINHA EM VERMELHO
                D.DefaultCellStyle.BackColor = Drawing.Color.Red
            End If
        Next
    End Sub
```

```
    Private Function ClassHasNoSignificantChanges(ByVal Fields As List(Of FieldInfo),
ByVal TDoc As TextDocument) As Boolean
        Dim FI As FieldInfo

        For Each F As FieldInfo In Fields
            FI = _DW.GetFieldInfoWithoutMappings(F, TDoc)

            If F.Name = FI.Name Then
                'Nomes iguais então field existe
                If Not (F.TypeName = FI.TypeName And F.IsNullable = FI.IsNullable) Then
                    'Tipos diferentes então algo foi mudado no banco de dados
                    Return False
                End If
            Else
                'Não existe este Field definido na classe como propriedade, provavelmente
é um campo novo na base de dados
                Return False
            End If
        Next

        Return True
    End Function

    '------------------------------------------------------------------------------------
----------------------------------------------------------------
#End Region

    Private Sub grdFields_CellDoubleClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles grdFields.CellDoubleClick
        Try

_DW.FindPropertyAndSetFocus(DirectCast(grdFields.Rows(e.RowIndex).DataBoundItem,
FieldInfo).Name)
        Catch ex As Exception
            Me.TopMost = False
            MsgBox(ex.Message)
            Me.TopMost = True
        End Try
    End Sub
End Class
```

frmMainWindow.vb

```
Imports EnvDTE80
Imports Microsoft.Win32

Public Class frmMainWindow

    Private _AppObject As DTE2
    Private _InstallFolder As String

    Public Sub New(ByVal ApplicationObject As DTE2)

        ' This call is required by the designer.
        InitializeComponent()
```

```
        ' Add any initialization after the InitializeComponent() call.
        _AppObject = ApplicationObject
        _InstallFolder = GetInstallFolder()
    End Sub

    Private Function GetInstallFolder() As String
        Dim Reg As RegistryKey

        'Busca no registry o caminho onde o DBCLassMapper foi instalado
        Reg = Registry.LocalMachine.OpenSubKey("Software\Deltacon\DBClassMapper", True)

        'Try
        '    Reg =
Registry.LocalMachine.OpenSubKey("Software\Microsoft\VisualStudio\10.0\AutomationOptions\
LookInFolders", True)
        'Catch ex As Exception
        '    Try
        '        Reg =
Registry.LocalMachine.OpenSubKey("Software\Microsoft\VisualStudio\9.0\AutomationOptions\L
ookInFolders", True)
        '    Catch ex2 As Exception
        '        MsgBox("Could not find the folder where DBClassMapper were installed.
Execute setup of DBClassMapper to fix this problem.")
        '        Return ""
        '    End Try
        'End Try

        Return CStr(Reg.GetValue("DBClassMapperPath"))
    End Function

    Private Sub btnNewProperty_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewProperty.Click
        Try
            Dim NewProp As New frmNewProperty(_AppObject, _InstallFolder)
            NewProp.Show()
            Me.Close()
        Catch ExNull As NullReferenceException
            MsgBox("No document selected for creating a new property!",
MsgBoxStyle.Exclamation)
        Catch Ex As Exception
            MsgBox(Ex.ToString)
        End Try
    End Sub

    Private Sub btnNewClass_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewClass.Click
        Try
            Dim NewClass As New frmNewClass(_AppObject, _InstallFolder)
            NewClass.Show()
            Me.Close()
        Catch Ex As Exception
            MsgBox(Ex.ToString)
        End Try
    End Sub

    Private Sub btnReadDataBase_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnReadDataBase.Click
        Try
```

```
            Dim DataBaseReader As New frmDBReader(_AppObject, _InstallFolder)
            DataBaseReader.Show()
            Me.Close()
        Catch Ex As Exception
            MsgBox(Ex.ToString)
        End Try
    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClose.Click
        Me.Close()
    End Sub

    Private Sub frmMainWindow_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        If _InstallFolder = "" Then
            btnNewClass.Enabled = False
            btnNewProperty.Enabled = False
        End If
    End Sub

    Private Sub frmMainWindow_ResizeBegin(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeBegin
        Me.Opacity = 0.15
    End Sub

    Private Sub frmMainWindow_ResizeEnd(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeEnd
        Me.Opacity = 1
    End Sub

    Private Sub btnQueryBuilder_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnQueryBuilder.Click
        Try
            If _AppObject.ActiveDocument Is Nothing Then
                MsgBox("No document selected for creating queries", vbExclamation)
                Exit Sub
            End If
            Dim QueryBuilder As New frmQueryBuilder(_AppObject, _InstallFolder)
            QueryBuilder.Show()
            Me.Close()
        Catch Ex As Exception
            MsgBox(Ex.ToString)
        End Try
    End Sub
End Class
```

frmNewClass.vb

```
Imports EnvDTE80
Imports EnvDTE
Imports System.Text

Public Class frmNewClass

    Private _AppObject As DTE2
```

```
    Private _InstallFolder As String

    Public Sub New(ByVal ApplicationObject As DTE2, ByVal InstallFolder As String)

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        _AppObject = ApplicationObject
        _InstallFolder = InstallFolder
    End Sub

    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClose.Click
        Me.Close()
    End Sub

    Private Sub btnCreate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCreate.Click
        If IsValid() Then
            Try
                CreateClass()
                Me.Close()
            Catch ex As Exception
                MsgBox(ex.Message)
            End Try
        End If
    End Sub

    Private Sub CreateClass()
        Dim DW As DocWriter
        Dim TI As TableInfo

        DW = New DocWriter(_AppObject, _InstallFolder)
        TI = New TableInfo
        TI.TableName = txtTable.Text
        TI.Schema = txtSchema.Text
        TI.ClassName = txtClassName.Text
        DW.CreateClass(TI)
    End Sub

    Private Function IsValid() As Boolean
        If txtClassName.Text = "" Then
            MsgBox("Missing the Class Name!")
            Return False
        End If

        Return True
    End Function

    Private Sub frmNewClass_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        Dim MainWin As New frmMainWindow(_AppObject)
        MainWin.Show()
        Me.Dispose()
    End Sub
```

```
    Private Sub frmNewClass_ResizeBegin(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeBegin
        Me.Opacity = 0.15
    End Sub

    Private Sub frmNewClass_ResizeEnd(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeEnd
        Me.Opacity = 1
    End Sub

    Private Sub frmNewClass_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

    End Sub
End Class
```

frmNewProperty.vb

```
Imports EnvDTE80
Imports EnvDTE
Imports VSLangProj
Imports System.CodeDom
Imports System.Text
Imports System.Text.RegularExpressions
Imports System.Collections.Generic
Imports System.Reflection
Imports System.IO
Imports System.Security.Policy
Imports Microsoft.Win32
Imports System.Xml.Serialization
Imports System.Drawing

Public Class frmNewProperty

    Private _AppObject As DTE2
    Private _InstallFolder As String
    Private _LastProj As Project
    Private _VSTypes As New List(Of FillComboList)
    Private _DBTypes As New List(Of FillComboList)
    Private _BlockRefs As Boolean = False

    Public Sub New(ByVal ApplicationObject As DTE2, ByVal InstallFolder As String)

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        _AppObject = ApplicationObject
        _InstallFolder = InstallFolder
    End Sub

    'Public Sub Start(ByVal ApplicationObject As DTE2)
    '    _AppObject = ApplicationObject
    'End Sub
```

```
    Private Sub frmNewProperty_Activated(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Activated
        Dim ActualProj As Project =
_AppObject.ActiveDocument.ProjectItem.ContainingProject

        'Bloqueia para as referencias não ficarem carregando enquanto o combo é ajustado
        _BlockRefs = True

        'Caso o ultimo projeto não tiver sido definido ainda ou for diferente do atual
então recarrega os assemblys da combo
        If _LastProj Is Nothing Then
            cmbReference.Items.Clear()
            cmbReference.Items.AddRange(Mapper.GetReferences(_AppObject).ToArray)
            LoadTypesCombo()
            _LastProj = ActualProj
        ElseIf Not _LastProj.Equals(ActualProj) Then
            cmbReference.Items.Clear()
            cmbReference.Items.AddRange(Mapper.GetReferences(_AppObject).ToArray)
            LoadTypesCombo()
            _LastProj = ActualProj
        End If

        _BlockRefs = False
    End Sub

    Private Sub frmNewProperty_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        Dim MainWin As New frmMainWindow(_AppObject)
        MainWin.Show()
        Me.Dispose()
    End Sub

    Private Sub NewProperty_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown
        Select Case e.KeyCode
            Case Keys.Escape
                btnClose_Click(Nothing, New EventArgs())
        End Select
    End Sub

    Private Sub NewProperty_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Me.Icon = My.Resources.Database

        'Ajuste do Label a da ProgressBar do Assembly Loading
        lblLoading.Size = New Size(348, 255)
        lblLoading.Location = New Point(8, 5)
        pgbLoading.Size = New Size(347, 22)
        pgbLoading.Location = New Point(9, 154)

        LoadLastCheckedBoxes()
        LoadCombosAndLists()

        Me.Refresh()
    End Sub

    Private Sub LoadCombosAndLists()
        'Montagem das listas com os tipos definidos
```

```
        Mount_PrimitiveTypes()
        Mount_DBTypes()

        'Montagem do combo que contem todos os tipos de objetos
        cmbType.DisplayMember = "Name"
        cmbType.ValueMember = "Key"

        'Montagem do combo que contem os tipos mapeaveis
        cmbMapperType.DisplayMember = "Name"
        cmbMapperType.ValueMember = "Key"
        cmbMapperType.Items.AddRange(GetMapperTypes.ToArray)
        SelectLastType()

        'Montagem do combo das referencias
        cmbReference.DisplayMember = "Name"
        cmbReference.ValueMember = "Data"
        cmbReference.Items.Clear()
        cmbReference.Items.AddRange(Mapper.GetReferences(_AppObject).ToArray)
        SelectLastReference()

        'Montagem da lista de Mappers
        clbMappers.CheckOnClick = True
        clbMappers.FormatString = "Key|Name|Data"
    End Sub

    Private Sub LoadLastCheckedBoxes()
        chkNullableOf.Checked = My.Settings.CheckNullableOf
        chkOnPropertyChanged.Checked = My.Settings.CheckAddPropChange
    End Sub

    Private Sub SelectLastReference()
        Dim SavedKey As String = My.Settings.LastReference

        If SavedKey <> "" Then
            For Each F As FillComboList In cmbReference.Items
                If F.Key = SavedKey Then
                    Try
                        cmbReference.SelectedItem = F
                    Catch Ex As Exception
                        'Do Nothing
                    End Try
                End If
            Next
        End If
    End Sub

    Private Sub SelectLastType()
        Dim SavedKey As String = My.Settings.LastMapperType
        cmbMapperType.SelectedIndex = 0

        If SavedKey <> "" Then
            For Each F As FillComboList In cmbMapperType.Items
                If F.Key = SavedKey Then
                    Try
                        cmbMapperType.SelectedItem = F
                    Catch Ex As Exception
                        'Do Nothing
                    End Try
```

```
                End If
            Next
        End If
    End Sub

    ' ''' <summary>
    ' ''' Pega todas as referencias locais do projeto
    ' ''' </summary>
    ' ''' <returns>Lista de referencias prontas para preencher um ComboBox</returns>
    ' ''' <remarks>Não carrega referencias que não são copias locais. Tem esse
comportamento para evitar listar muita DLLs desnecessárias.</remarks>
    'Private Function GetReferences(ByVal ApplicationObject As DTE2) As List(Of
FillComboList)
    '    Dim Proj As VSProject =
DirectCast(ApplicationObject.ActiveDocument.ProjectItem.ContainingProject.Object,
VSProject)
    '    Dim Result As New List(Of FillComboList)

    '    For Each Ref As Reference In Proj.References
    '        'Adiciona as referencias na combo para depois poder buscar as classes
mapeadas
    '        'A propriedade CopyLocal = True é utilizada para filtrar somente as DLLs
locais
    '        'retirando assim as DLLs do System e etc as quais não serão utilizadas
    '        If Ref.CopyLocal Then Result.Add(New FillComboList(Ref.Identity, Ref.Name,
Ref))
    '    Next

    '    Return Result
    'End Function

    ''' <summary>
    ''' Montagem da lista de mapeaveis
    ''' </summary>
    ''' <remarks></remarks>
    Private Function GetMapperTypes() As List(Of FillComboList)
        Dim Result As New List(Of FillComboList)

        Result.Add(New FillComboList("MP0", "None"))
        Result.Add(New FillComboList("MP1", "Field"))
        Result.Add(New FillComboList("MP2", "Key"))
        Result.Add(New FillComboList("MP3", "Alternate Key"))
        Result.Add(New FillComboList("MP4", "Foreign Key"))
        Result.Add(New FillComboList("MP5", "Date Time"))
        Result.Add(New FillComboList("MP6", "Time Stamp"))

        Return Result
    End Function

    ''' <summary>
    ''' Montagem da lista de tipos do VB
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub Mount_PrimitiveTypes()
        If FileExtension = "vb" Then
            Mount_VBPrimitiveTypes()
        ElseIf FileExtension = "cs" Then
            Mount_CSPrimitiveTypes()
```

```
        Else
            Throw New Exception("Only VB and C# files are supported.")
        End If
    End Sub

    ''' <summary>
    ''' Montagem da lista de tipos do VB
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub Mount_VBPrimitiveTypes()
        _VSTypes.Add(New FillComboList("VB0", "Byte", "Byte"))
        _VSTypes.Add(New FillComboList("VB1", "Short", "Short"))
        _VSTypes.Add(New FillComboList("VB2", "Integer", "Integer"))
        _VSTypes.Add(New FillComboList("VB3", "Long", "Long"))
        _VSTypes.Add(New FillComboList("VB4", "Single", "Single"))
        _VSTypes.Add(New FillComboList("VB5", "Double", "Double"))
        _VSTypes.Add(New FillComboList("VB6", "Boolean", "Boolean"))
        _VSTypes.Add(New FillComboList("VB7", "Char", "Char"))
        _VSTypes.Add(New FillComboList("VB8", "Decimal", "Decimal"))
        _VSTypes.Add(New FillComboList("VB9", "String", "String"))
    End Sub

    ''' <summary>
    ''' Montagem da lista de tipos do CS
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub Mount_CSPrimitiveTypes()
        _VSTypes.Add(New FillComboList("VB0", "byte", "byte"))
        _VSTypes.Add(New FillComboList("VB1", "short", "short"))
        _VSTypes.Add(New FillComboList("VB2", "int", "int"))
        _VSTypes.Add(New FillComboList("VB3", "long", "long"))
        _VSTypes.Add(New FillComboList("VB4", "single", "single"))
        _VSTypes.Add(New FillComboList("VB5", "double", "double"))
        _VSTypes.Add(New FillComboList("VB6", "bool", "bool"))
        _VSTypes.Add(New FillComboList("VB7", "char", "char"))
        _VSTypes.Add(New FillComboList("VB8", "decimal", "decimal"))
        _VSTypes.Add(New FillComboList("VB9", "string", "string"))
    End Sub

    ''' <summary>
    ''' Montagem da lista de tipos do DBManager
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub Mount_DBTypes()
        _DBTypes.Add(New FillComboList("DB0", "DateTime", "DateTime"))
        _DBTypes.Add(New FillComboList("DB1", "DBType.DBDate", "DateTime"))
        _DBTypes.Add(New FillComboList("DB2", "DBType.DBTime", "DateTime"))
        _DBTypes.Add(New FillComboList("DB3", "DBType.DBDateTime", "DateTime"))
        _DBTypes.Add(New FillComboList("DB4", "DBType.DBTimeStamp", "DateTime"))
    End Sub

    Private Sub cmbMapperType_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles cmbMapperType.SelectedIndexChanged
        Dim SelItem As FillComboList = DirectCast(cmbMapperType.SelectedItem,
FillComboList)

        My.Settings.LastMapperType = DirectCast(cmbMapperType.SelectedItem,
FillComboList).Key
```

```
        clbMappers.Items.Clear()

        If SelItem.Key = "MP0" Then
            clbMappers.Enabled = False
        Else
            clbMappers.Enabled = True
            clbMappers.Items.Add(New FillComboList("DBField", "DBField", "<DBField()> _",
True), True)
        End If

        Select Case SelItem.Key
            Case "MP1"
                LoadFieldAttributes()
            Case "MP2"
                LoadKeyAttributes()
            Case "MP3"
                LoadAKAttributes()
            Case "MP4"
                LoadFKAttributes()
            Case "MP5"
                LoadDateTimeAttributes()
            Case "MP6"
                LoadTimeStampSttributes()
        End Select

        LoadTypesCombo()
    End Sub

    Private Sub LoadFieldAttributes()
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _"))
        clbMappers.Items.Add(New FillComboList("DBStream", "DBStream", "<DBStream()> _"))
        clbMappers.Items.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _"))
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub

    Private Sub LoadKeyAttributes()
        clbMappers.Items.Add(New FillComboList("DBDateTime", "DBDateTime",
"<DBDateTime()> _"))
        clbMappers.Items.Add(New FillComboList("KeyAttribute", "Key", "<Key()> _", True),
True)
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _", True), True)
        clbMappers.Items.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _", True), True)
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _", True), True)
        clbMappers.Items.Add(New FillComboList("StringLength", "StringLength",
"<StringLength(50, ErrorMessage:=""Property exceeded it's limits(0-50)."")> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub

    Private Sub LoadAKAttributes()
```

```
        clbMappers.Items.Add(New FillComboList("DBDateTime", "DBDateTime",
"<DBDateTime()> _"))
        clbMappers.Items.Add(New FillComboList("DBAlternateKey", "DBAlternateKey",
"<DBAlternateKey()> _", True), True)
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _"), True)
        clbMappers.Items.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _", True), True)
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _", True), True)
        clbMappers.Items.Add(New FillComboList("StringLength", "StringLength",
"<StringLength(50, ErrorMessage:=""Property exceeded it's limits(0-50).""")> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub

    Private Sub LoadFKAttributes()
        clbMappers.Items.Add(New FillComboList("AssociationAttribute", "Association",
"<Association()> _", True), True)
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _"), True)
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _"))
        clbMappers.Items.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _"))
        clbMappers.Items.Add(New FillComboList("StringLength", "StringLength",
"<StringLength(50, ErrorMessage:=""Property exceeded it's limits(0-50).""")> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub

    Private Sub LoadDateTimeAttributes()
        clbMappers.Items.Add(New FillComboList("DBDateTime", "DBDateTime",
"<DBDateTime()> _", True), True)
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _"), True)
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _"))
        clbMappers.Items.Add(New FillComboList("EditableAttribute", "Editable(False)",
"<Editable(False)> _"))
        clbMappers.Items.Add(New FillComboList("ConcurrencyCheckAttribute", "Concurrency
Check", "<ConcurrencyCheck()> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub

    Private Sub LoadTimeStampSttributes()
        clbMappers.Items.Add(New FillComboList("TimeStampAttribute", "Time Stamp",
"<TimeStamp()> _", True), True)
        clbMappers.Items.Add(New FillComboList("DBIndexAttribute", "DBIndex",
"<DBIndex()> _"), True)
        clbMappers.Items.Add(New FillComboList("RequiredAttribute", "Required",
"<Required()> _"))
        clbMappers.Items.Add(New FillComboList("ConcurrencyCheckAttribute", "Concurrency
Check", "<ConcurrencyCheck()> _"))
        clbMappers.Items.Add(New FillComboList("AGField", "Auto Generate Field", ""))
        clbMappers.Items.Add(New FillComboList("AGFilter", "Auto Generate Filter", ""))
    End Sub
```

```
    Private Sub btnClose_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClose.Click
        Me.Close()
    End Sub

    Private Sub btnAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnAdd.Click
        Dim DW As DocWriter
        Dim FI As FieldInfo

        Try
            If Validar() Then
                DW = New DocWriter(_AppObject, _InstallFolder)
                FI = New FieldInfo
                FI.Name = txtName.Text
                FI.FCLType = DirectCast(cmbType.SelectedItem, FillComboList)
                FI.IsNullable = chkNullableOf.Checked

                If cmbType.SelectedItem Is Nothing Then
                    FI.TypeName = cmbType.Text
                Else
                    FI.TypeName = CStr(DirectCast(cmbType.SelectedItem,
FillComboList).Data)
                End If

                FI.Mappers = New List(Of FillComboList)
                For Each F As FillComboList In clbMappers.CheckedItems
                    FI.Mappers.Add(F)
                Next

                DW.CreateProperty(FI, DirectCast(cmbMapperType.SelectedItem,
FillComboList).Key <> "MP0", chkOnPropertyChanged.Checked)
            End If
        Catch ExNull As NullReferenceException
            MsgBox("Não existe documento selecionado para inserir a nova propriedade!",
MsgBoxStyle.Exclamation)
        Catch Ex As Exception
            MsgBox(Ex.ToString, MsgBoxStyle.Critical)
        End Try
    End Sub

    Private Sub SetAssemblyLoading(ByVal Loading As Boolean, Optional ByVal LoaderText As
String = "Loading...")
        lblLoading.Text = LoaderText
        lblLoading.Visible = Loading
        'Feito para ressetar o valor do marquee
        pgbLoading.Style = ProgressBarStyle.Blocks
        pgbLoading.Style = ProgressBarStyle.Marquee
        pgbLoading.Visible = Loading
        pgbLoading.BringToFront()
        Me.Refresh()
    End Sub

    Private Function Validar() As Boolean
        Dim NotFound As Boolean = True

        If cmbType.Text = String.Empty Then
```

```
            MsgBox("Indique um Type")
            cmbType.Focus()
            Return False
        End If

        If txtName.Text = String.Empty Then
            MsgBox("Indique o Name")
            txtName.Focus()
            Return False
        End If

        Return True
    End Function

    Private Sub cmbReference_SelectedIndexChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles cmbReference.SelectedIndexChanged
        My.Settings.LastReference = DirectCast(cmbReference.SelectedItem,
FillComboList).Key
        If Not _BlockRefs Then LoadTypesCombo()
    End Sub

    ''' <summary>
    ''' Carrega o combo de tipos pegando além dos tipos nativos do vb e dos tipos
especiais do DBManager os tipos mapeados nas classes
    ''' </summary>
    ''' <remarks></remarks>
    Private Sub LoadTypesCombo()
        Dim SelRefObj As FillComboList = DirectCast(cmbReference.SelectedItem,
FillComboList)
        Dim SelMPType As FillComboList = DirectCast(cmbMapperType.SelectedItem,
FillComboList)
        Dim LoadDateTime As Boolean = False
        If SelMPType Is Nothing Then Exit Sub

        cmbType.Items.Clear()

        'Caso o mapeador puder ser um DateTime então carregar os tipos DateTime
        For Each F As FillComboList In clbMappers.Items
            If CStr(F.Data) = "DateTime" Then
                LoadDateTime = True
                Exit For
            End If
        Next

        'Mapeamentos Nativos se não for especificadamente DateTime
        If SelMPType.Key <> "MP5" AndAlso SelMPType.Key <> "MP6" Then
cmbType.Items.AddRange(_VSTypes.ToArray)

        'Mapeamentos de DateTime
        If SelMPType.Key = "MP5" OrElse LoadDateTime Then
cmbType.Items.AddRange(_DBTypes.ToArray)

        'Mapeamento do Time Stamp
        If SelMPType.Key = "MP6" Then cmbType.Items.Add(New FillComboList("DateTime",
"DateTime", "DateTime"))

        'Mapeamentos de Objetos (Classes)
        If SelMPType.Key = "MP4" Then
```

```
            'Bloqueia o AddIn
            SetAssemblyLoading(True, "Loading Assembly...")

            If SelRefObj Is Nothing Then

cmbType.Items.AddRange(Mapper.GetMappeds(_AppObject.ActiveDocument.ProjectItem.Containing
Project, _InstallFolder).ToArray)
            Else

cmbType.Items.AddRange(Mapper.GetMappeds(_AppObject.ActiveDocument.ProjectItem.Containing
Project, _InstallFolder, DirectCast(SelRefObj.Data, Reference)).ToArray)
            End If

            'Libera o AddIn
            SetAssemblyLoading(False)
        End If
    End Sub

    'Private Function GetMappeds(Optional ByVal Ref As Reference = Nothing) As List(Of
FillComboList)
    '    Dim Proj As Project = _AppObject.ActiveDocument.ProjectItem.ContainingProject
    '    Dim Result As New List(Of FillComboList)
    '    'Dim Reg As RegistryKey
    '    Dim AppPath As String
    '    Dim PPath As String
    '    Dim SerializePath As String

    '    'FullPath tem o caminho completo da solution (não do arquivo .sln ou .vbp e sim
da localicação ex.: "C:\VSProj\MeuProjeto\")
    '    Dim FullPath As String = Proj.Properties.Item("FullPath").Value.ToString()
    '    'OutputPath tem o caminho parcial no qual o desenvolvedor compilou esse projeto
pela ultima vez (ex.: "\bin\Debug\")
    '    Dim OutputPath As String =
Proj.ConfigurationManager.ActiveConfiguration.Properties.Item("OutputPath").Value.ToStrin
g()
    '    'OutputFile tem o nome do ultimo arquivo compilado (ex.: "MeuPrograma.dll")
    '    Dim OutputFile As String =
Proj.Properties.Item("OutputFileName").Value.ToString()
    '    'OutputFullName combina tudo que foi pego acima em um caminho válido para o
ultimo arquivo compilado
    '    Dim OutputFullName As String = Path.Combine(FullPath, OutputPath)
    '    OutputFullName = Path.Combine(OutputFullName, OutputFile)

    '    'Bloqueia o AddIn
    '    SetAssemblyLoading(True, "Loading Assembly...")

    '    'Caminho onde o DBClassMapper foi instalado
    '    AppPath = _InstallFolder
    '    'Caminho do aplicativo que será chamado para serializar
    '    PPath = String.Format("{0}\AssemblyInspector.exe", AppPath)
    '    'Montagem do caminho onde está o arquivo serializado
    '    SerializePath = GetSerializePath()

    '    'Busca mapeados do projeto
    '    Try
    '        Result.AddRange(SerializeAndDeserializeMappeds(PPath, OutputFullName,
SerializePath))
    '    Catch ex As Exception
```

```
    '        MsgBox(String.Format("Erro ao buscar mapeamentos do projeto.{0}{0}{1}",
vbCrLf, ex.Message))
    '    End Try

    '    'Busca mapeados da referencia selecionada
    '    Try
    '        If Not Ref Is Nothing Then
    '            Result.AddRange(SerializeAndDeserializeMappeds(PPath, Ref.Path,
SerializePath))
    '        End If
    '    Catch ex As Exception
    '        MsgBox(String.Format("Erro ao buscar mapeamentos da referencia
selecionada.{0}{0}{1}", vbCrLf, ex.Message))
    '    End Try

    '    'Libera o AddIn
    '    SetAssemblyLoading(False)

    '    Return Result
    'End Function

    'Private Function GetSerializePath() As String
    '    Dim Path As String = String.Format("{0}\.deltacon\.dbclassmapper",
My.Computer.FileSystem.SpecialDirectories.AllUsersApplicationData)

    '    If Not Directory.Exists(Path) Then
    '        Directory.CreateDirectory(Path)
    '    End If

    '    Return String.Format("{0}\Mappeds.xml", Path)
    'End Function

    'Private Function SerializeAndDeserializeMappeds(ByVal PPath As String, ByVal DLLPath
As String, ByVal SerializePath As String) As List(Of FillComboList)
    '    Dim XS As New XmlSerializer(GetType(DLLInfo))
    '    Dim P As New System.Diagnostics.Process
    '    Dim FS As FileStream = Nothing
    '    Dim DI As DLLInfo
    '    Dim Result As New List(Of FillComboList)
    '    Dim Args As String = String.Format("""{0}"" ""{1}""", DLLPath, SerializePath)

    '    Try
    '        'Chamada do processo que irá serializar os mapeamentos da DLL passada
    '        P.StartInfo = New ProcessStartInfo(PPath, Args)
    '        P.StartInfo.WindowStyle = ProcessWindowStyle.Hidden

    '        'Inicia o Processo
    '        If P.Start() Then
    '            'Aguarda o processo terminar (Feito com While para o ProgressBar
funcionar sem utilizar multi-thread explicitamente)
    '            'If P.WaitForExit(60000) Then
    '            While Not P.HasExited
    '                System.Windows.Forms.Application.DoEvents()
    '                System.Threading.Thread.Sleep(10)
    '            End While

    '            'Se o processo retornar um erro uma exceção é levantada aqui
    '            If Not P.ExitCode = 0 Then
```

```
    '                Throw New Exception(String.Format("AssemblyInspector retornou erro
ao tentar serializar as classes mapeadas.{0}{0}Returned Code: {1}", vbCrLf, P.ExitCode))
    '            End If

    '            'Deserialização do arquivo serializado e captura dos arquivos mapeados
    '            FS = New FileStream(SerializePath, FileMode.Open, FileAccess.Read)
    '            DI = DirectCast(XS.Deserialize(FS), DLLInfo)
    '            FS.Flush()

    '            Result = DI.Mappeds
    '            'Else
    '            '    Throw New Exception("Excedeu o tempo de espera para carregar os
assemblys.")
    '            'End If
    '        Else
    '            Throw New Exception("Não foi possível iniciar o AssemblyInspector.")
    '        End If

    '        Return Result
    '    Catch ex As Exception
    '        'Qualquer erro é jogado para quem chamou esta função
    '        Throw New Exception(ex.Message, ex)
    '    Finally
    '        'Verifica se o processo é algo, se for ele verifica se ainda esta aberto, se
estiver ele fecha e por ultimo esvazia
    '        If Not IsNothing(P) Then
    '            If Not P.HasExited Then P.Close()
    '            P = Nothing
    '        End If
    '        'Verifica se o FileStream existe para poder fecha-lo
    '        If Not IsNothing(FS) Then
    '            Try
    '                FS.Close()
    '                FS = Nothing
    '            Catch ex2 As Exception
    '            End Try
    '        End If
    '        'Manda o GC coletar
    '        GC.Collect()
    '    End Try
    'End Function

    Private Sub clbMappers_ItemCheck(ByVal sender As Object, ByVal e As
System.Windows.Forms.ItemCheckEventArgs) Handles clbMappers.ItemCheck
        Dim FCL As FillComboList = DirectCast(DirectCast(sender,
CheckedListBox).Items(e.Index), FillComboList)
        If FCL.AlwaysChecked Then e.NewValue = CheckState.Checked
    End Sub

    Private Sub chkNullableOf_CheckedChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles chkNullableOf.CheckedChanged
        My.Settings.CheckNullableOf = chkNullableOf.Checked
    End Sub

    Private Sub chkOnPropertyChanged_CheckedChanged(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles chkOnPropertyChanged.CheckedChanged
        My.Settings.CheckAddPropChange = chkOnPropertyChanged.Checked
    End Sub
```

```
    Private Sub frmNewProperty_ResizeBegin(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeBegin
        Me.Opacity = 0.15
    End Sub

    Private Sub frmNewProperty_ResizeEnd(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.ResizeEnd
        Me.Opacity = 1
    End Sub

    Private Sub btnBuild_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnBuild.Click
        Dim Sol As Solution = _AppObject.DTE.Solution

        SetAssemblyLoading(True, "Building Solution...")

        Sol.SolutionBuild.Build(False)

        If Sol.SolutionBuild.BuildState = vsBuildState.vsBuildStateNotStarted Then
            MsgBox("Error on Building Solution. CHeck Output for more details.")
        Else
            While Sol.SolutionBuild.BuildState = vsBuildState.vsBuildStateInProgress
                System.Windows.Forms.Application.DoEvents()
                System.Threading.Thread.Sleep(10)
            End While
        End If

        If Sol.SolutionBuild.BuildState = vsBuildState.vsBuildStateDone Then
            MsgBox("Build finished (It doesn't means it built successfull.). Check Output
for more details.")
        Else
            MsgBox("For some reason Build has Not Finished and has stopped building.
Check Output for more details.")
        End If

        SetAssemblyLoading(False)
    End Sub

    Private ReadOnly Property FileExtension As String
        Get
            Return
_AppObject.ActiveDocument.FullName.Substring(_AppObject.ActiveDocument.FullName.LastIndex
Of(".") + 1)
        End Get
    End Property
End Class
```

frmNodeProperties.vb

```
Imports DBManager.DBImprover
Imports DBManager
Imports System.Collections.Generic
Imports DBManager.DBWarder

Public Class frmNodeProperties
```

```
    Private Node As ClassNode
    Private OrdenedFilters As List(Of ToolDBFilter)
    Private OrdenedOrders As List(Of ToolDBOrder)

    Public Sub New(ByRef Node As ClassNode, ByRef OrdenedFilters As List(Of
ToolDBFilter), ByRef OrdenedOrders As List(Of ToolDBOrder), Optional BlockJoin As Boolean
= False)
        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        Me.Node = Node
        Me.OrdenedFilters = OrdenedFilters
        Me.OrdenedOrders = OrdenedOrders

        If BlockJoin Then
            cmbJoin.Enabled = False
            cmbRelation.Enabled = False
        End If
    End Sub

    Private Sub frmNodeProperties_FormClosing(sender As Object, e As
System.Windows.Forms.FormClosingEventArgs) Handles Me.FormClosing
        Node.JoinType = DirectCast(cmbJoin.SelectedItem, JoinType)
        Node.Relationship = DirectCast(cmbRelation.SelectedItem, Relationship)
        Node.SQLAlias = txtAlias.Text

        Node.SpecifiedFields = False
        For Each P As PropInfo In Node.ClassInfo.Props
            If Not P.Selected Then
                Node.SpecifiedFields = True
                Exit For
            End If
        Next
    End Sub

    Private Sub frmNodeProperties_Load(sender As System.Object, e As System.EventArgs)
Handles MyBase.Load
        LoadBase()
        LoadNode()
    End Sub

    Private Sub LoadBase()
        grdProps.AutoGenerateColumns = False
        grdFilters.AutoGenerateColumns = False
        grdOrders.AutoGenerateColumns = False

        For Each JT As JoinType In [Enum].GetValues(GetType(JoinType))
            cmbJoin.Items.Add(JT)
        Next

        For Each R As Relationship In [Enum].GetValues(GetType(Relationship))
            cmbRelation.Items.Add(R)
        Next
    End Sub

    Private Sub LoadNode()
```

```
        txtClass.Text = Node.ClassInfo.ClassName
        txtTable.Text = String.Format("{0}.{1}", Node.ClassInfo.Schema,
Node.ClassInfo.Table)
        cmbJoin.SelectedItem = Node.JoinType
        cmbRelation.SelectedItem = Node.Relationship
        txtAlias.Text = Node.SQLAlias

        grdProps.DataSource = Node.ClassInfo.Props

        'Só pode preencher se tiver algo na lista, caso contrário existe um bug do .Net
que da erro de index -1
        'A solução foi inserir esta gambiarra
        If Node.Orders.Count > 0 Then
            grdOrders.DataSource = Node.Orders
        End If

        'Só pode preencher se tiver algo na lista, caso contrário existe um bug do .Net
que da erro de index -1
        'A solução foi inserir esta gambiarra
        If Node.Filters.Count > 0 Then
            grdFilters.DataSource = Node.Filters
        End If
    End Sub

    Private Sub btnAddOrder_Click(sender As System.Object, e As System.EventArgs) Handles
btnAddOrder.Click
        If grdProps.SelectedRows.Count = 0 Then Exit Sub

        Dim NewOrder As New ToolDBOrder("", OrderMode.Asc)
        Dim FormOrder As New frmAddOrder(Node,
DirectCast(grdProps.SelectedRows(0).DataBoundItem, PropInfo), NewOrder)

        FormOrder.ShowDialog(Me)

        If NewOrder IsNot Nothing AndAlso NewOrder.Field <> "" Then
            tabBottom.SelectTab(1)
            grdOrders.DataSource = Nothing
            Node.Orders.Add(NewOrder)
            grdOrders.DataSource = Node.Orders
            OrdenedOrders.Add(NewOrder)
        End If
    End Sub

    Private Sub btnRemove_Click(sender As System.Object, e As System.EventArgs) Handles
btnRemove.Click
        If tabBottom.SelectedTab.Equals(pagFilters) Then
            RemoveFilter()
        Else
            RemoveOrder()
        End If
    End Sub

    Private Sub RemoveOrder()
        If grdOrders.SelectedRows.Count = 0 Then Exit Sub
        Dim O As ToolDBOrder = DirectCast(grdOrders.SelectedRows(0).DataBoundItem,
ToolDBOrder)

        grdOrders.DataSource = Nothing
```

```
        Node.Orders.Remove(O)
        grdOrders.DataSource = Node.Orders
        OrdenedOrders.Remove(O)
    End Sub

    Private Sub RemoveFilter()
        If grdFilters.SelectedRows.Count = 0 Then Exit Sub
        Dim F As ToolDBFilter = DirectCast(grdFilters.SelectedRows(0).DataBoundItem,
ToolDBFilter)

        grdFilters.DataSource = Nothing
        Node.Filters.Remove(F)
        grdFilters.DataSource = Node.Filters
        OrdenedFilters.Remove(F)
    End Sub

    Private Sub btnAddFilter_Click(sender As System.Object, e As System.EventArgs)
Handles btnAddFilter.Click
        If grdProps.SelectedRows.Count = 0 Then Exit Sub

        Dim NewFilter As New ToolDBFilter()
        Dim FormFilter As New frmAddFilter(Node,
DirectCast(grdProps.SelectedRows(0).DataBoundItem, PropInfo), NewFilter)

        FormFilter.ShowDialog(Me)

        If NewFilter IsNot Nothing AndAlso NewFilter.Field <> "" Then
            tabBottom.SelectTab(0)
            grdFilters.DataSource = Nothing
            Node.Filters.Add(NewFilter)
            grdFilters.DataSource = Node.Filters
            OrdenedFilters.Add(NewFilter)
        End If
    End Sub
End Class
```

frmQueryBuilder.vb

```
Imports EnvDTE80

Public Class frmQueryBuilder
    Private _AppObject As DTE2
    Private _InstallFolder As String

    Public Sub New(ByVal ApplicationObject As DTE2, ByVal InstallFolder As String)

        ' This call is required by the designer.
        InitializeComponent()

        ' Add any initialization after the InitializeComponent() call.
        _AppObject = ApplicationObject
        _InstallFolder = InstallFolder

        Dim QB As wpfQueryBuilder = DirectCast(eleQBuilder.Child, wpfQueryBuilder)
        QB.InstallFolder = _InstallFolder
        QB.AppObject = _AppObject
```

```
    End Sub
End Class
```

wpfQueryBuilder.xaml.vb

```
Imports System.Collections.Generic
Imports EnvDTE80
Imports VSLangProj
Imports System.Text
Imports System.Linq
Imports System.Windows
Imports System.Windows.Input
Imports System.Windows.Controls
Imports System.Windows.Shapes
Imports System.Windows.Media
Imports DBManager
Imports Microsoft.Win32
Imports Microsoft.VisualBasic.Interaction
Imports DBManager.DBWarder
Imports DBManager.DBConnector
Imports DBManager.DBMapper
Imports DBManager.DBImprover

Public Class wpfQueryBuilder

#Region "Properties"
    Public Property OrdenedFilters As New List(Of ToolDBFilter)
    Public Property OrdenedOrders As New List(Of ToolDBOrder)
    Public Property Aggregators As New List(Of Aggregator)
    Public Property Languages As New List(Of String)

#End Region

#Region "ClassAndJoins_Fields"
    Private Property MainNode As ClassNode
    Private Property SelectedNode As ClassNode
#End Region

#Region "Window_Fields"
    Public Property AppObject As DTE2
    Public Property InstallFolder As String
    Public Property Classes As Dictionary(Of String, ClassInfo)
    Public Property Refs As List(Of FillComboList)
#End Region

#Region "Canvas_Fields"
    'Nodes
    Private IsMouseDraggingObj As Boolean = False
    Private DraggingElement As UIElement
    Private LastDragColor As Brush
    Private OrigX As Double
    Private OrigY As Double
    'Rectangle
    Private StartPoint As Point
    Private Rectangle As Rectangle
    Private IsDraggingRectangle As Boolean
```

```
    Private SelectedElements As New List(Of Grid)
#End Region

#Region "Methods"
    Private Sub Start()
        Me.GetReferences()
        Me.GetDictionary()
        Me.GetAggregTypes()
        'Me.GetDSNs()

        cmbReference.DisplayMemberPath = "Name"
        cmbReference.SelectedValuePath = "Data"
        cmbReference.ItemsSource = Refs

        lsbClasses.DisplayMemberPath = "Value"
        lsbClasses.SelectedValuePath = "Key"
        lsbClasses.ItemsSource = Classes.OrderBy(Function(x) DirectCast(x.Value,
ClassInfo).ToString)

        lsbAssociated.DisplayMemberPath = "Description"

        Languages.Add("Visual Basic")
        Languages.Add("C#")
        cmbTargetLanguage.ItemsSource = Languages
        cmbTargetLanguage.SelectedItem = "Visual Basic"
    End Sub

    Private Sub GetReferences()
        Refs = Mapper.GetReferences(AppObject)
    End Sub

    Private Sub GetDictionary()
        Dim FCL As List(Of FillComboList)
        Dim Erros As New StringBuilder

        Mouse.SetCursor(Cursors.Wait)

        If cmbReference.SelectedItem Is Nothing Then
            FCL =
Mapper.GetMappeds(AppObject.ActiveDocument.ProjectItem.ContainingProject, _InstallFolder)
        Else
            FCL =
Mapper.GetMappeds(AppObject.ActiveDocument.ProjectItem.ContainingProject, _InstallFolder,
DirectCast(DirectCast(cmbReference.SelectedItem, FillComboList).Data, Reference))
        End If

        Classes = New Dictionary(Of String, ClassInfo)

        For Each F As FillComboList In FCL
            Try
                Classes.Add(F.ClassInfo.FullName, F.ClassInfo)
            Catch exA As ArgumentException
                If exA.Message = "An item with the same key has already been added." Then
                    Erros.AppendFormat("{0}{1}", vbCrLf, F.ToString)
                Else
                    Throw exA
                End If
            End Try
```

```
        Next

        Mouse.SetCursor(Cursors.Arrow)

        If Erros.Length > 0 Then
            MsgBox(String.Format("Found some duplicated mappers:{0}{1}", vbCrLf, Erros))
        End If
    End Sub

    Private Sub GetAggregTypes()
        cmbType.ItemsSource = [Enum].GetValues(GetType(AggregateFunction)).Cast(Of
AggregateFunction).ToList
    End Sub

    Private Sub EmptyHolders()
        IsMouseDraggingObj = False
        IsDraggingRectangle = False

        If Rectangle IsNot Nothing Then
            SelectNodes(Rectangle)
            canMain.Children.Remove(Rectangle)
            Rectangle = Nothing
        End If
    End Sub

    Private Sub SelectNodes(R As Rectangle)
        Dim OSX As Double = CDbl(Rectangle.GetValue(Canvas.LeftProperty))
        Dim OSY As Double = CDbl(Rectangle.GetValue(Canvas.TopProperty))
        Dim OEX As Double = CDbl(Rectangle.GetValue(Canvas.LeftProperty)) +
CDbl(Rectangle.GetValue(Canvas.ActualWidthProperty))
        Dim OEY As Double = CDbl(Rectangle.GetValue(Canvas.TopProperty)) +
CDbl(Rectangle.GetValue(Canvas.ActualHeightProperty))
        Dim GSX As Double
        Dim GSY As Double
        Dim GEX As Double
        Dim GEY As Double

        For Each U As UIElement In canMain.Children
            If U.GetType.Equals(GetType(Grid)) Then
                Dim G As Grid = DirectCast(U, Grid)
                If G.Tag IsNot Nothing Then
                    GSX = CDbl(G.GetValue(Canvas.LeftProperty))
                    GSY = CDbl(G.GetValue(Canvas.TopProperty))
                    GEX = CDbl(G.GetValue(Canvas.LeftProperty)) +
CDbl(G.GetValue(Canvas.ActualWidthProperty))
                    GEY = CDbl(G.GetValue(Canvas.TopProperty)) +
CDbl(G.GetValue(Canvas.ActualHeightProperty))

                    If GSX > OSX AndAlso GSY > OSY AndAlso GEX < OEX AndAlso GEY < OEY
Then
                        DirectCast(DirectCast(G.Children(0), Border).Child,
Grid).Background = Brushes.CadetBlue
                        SelectedElements.Add(G)
                    End If
                End If
            End If
        Next
    End Sub
```

```
    Private Function CreateObj(Info As Object, left As Double, top As Double, Optional
DrawProps As DrawProps = Nothing) As Grid
        Dim InfoKey As String
        Dim InfoClassName As String
        Dim InfoFullName As String
        Dim AI As AssociationInfo
        Dim CI As ClassInfo

        If Info.GetType.Equals(GetType(AssociationInfo)) Then
            AI = DirectCast(Info, AssociationInfo)
            InfoKey = AI.GetKey
            InfoClassName = AI.ClassName
            InfoFullName = AI.FullName
        ElseIf Info.GetType.Equals(GetType(ClassInfo)) Then
            CI = DirectCast(Info, ClassInfo)
            InfoKey = CI.GetKey
            InfoClassName = CI.ClassName
            InfoFullName = CI.FullName
        Else
            Throw New Exception(String.Format("CreateObj method needs a ClassInfo or
AssociationInfo class. It received {0}", Info.GetType().Name))
        End If

        Dim MainGrid As New Grid With {.Name = InfoKey}
        Dim OutBorder As New Border With {.BorderBrush = Brushes.Black, .CornerRadius =
New CornerRadius(3, 3, 3, 3), .BorderThickness = New Thickness(2)}
        Dim InnerGrid As New Grid With {.Background = Brushes.OldLace}
        Dim MainLabel As New Label With {.Content = InfoClassName, .FontSize = 14,
.FontWeight = FontWeights.Bold, .HorizontalContentAlignment =
Windows.HorizontalAlignment.Center, .MinWidth = 128}
        Dim DescLabel As New TextBlock With {.Text = InfoFullName, .FontSize = 10,
.TextAlignment = TextAlignment.Center}
        Dim DropDown As New ContextMenu
        Dim NewMenuItem As MenuItem

        MainGrid.Children.Add(OutBorder)
        OutBorder.Child = InnerGrid

        MainGrid.SetValue(Window.LeftProperty, left)
        MainGrid.SetValue(Window.TopProperty, top)

        InnerGrid.RowDefinitions.Add(New RowDefinition With {.Height = GridLength.Auto})
        InnerGrid.RowDefinitions.Add(New RowDefinition With {.Height = GridLength.Auto})
        InnerGrid.ColumnDefinitions.Add(New ColumnDefinition With {.Width =
GridLength.Auto})
        InnerGrid.ColumnDefinitions.Add(New ColumnDefinition With {.Width =
GridLength.Auto})

        Grid.SetRow(MainLabel, 0)
        Grid.SetColumn(MainLabel, 0)
        InnerGrid.Children.Add(MainLabel)

        Grid.SetRow(DescLabel, 1)
        Grid.SetColumn(DescLabel, 0)
        InnerGrid.Children.Add(DescLabel)

        'Menu to Remove the Node
```

```
        NewMenuItem = New MenuItem With {.Header = "Remove"}
        AddHandler NewMenuItem.Click, AddressOf Remove_Click
        DropDown.Items.Add(NewMenuItem)
        'Menu to see the properties of the node
        NewMenuItem = New MenuItem With {.Header = "Properties"}
        AddHandler NewMenuItem.Click, AddressOf Properties_Click
        DropDown.Items.Add(NewMenuItem)
        'Includes the ContextMenu
        InnerGrid.ContextMenu = DropDown

        canMain.Children.Add(MainGrid)
        Me.UpdateLayout()

        'Ajustes no Obj após a sua criação (depende do seu tamanho após criado)
        If DrawProps IsNot Nothing Then
            InnerGrid.Background = DrawProps.Color

            If DrawProps.LeftReduceMyHalf Then
                left -= CDbl(MainGrid.GetValue(Window.ActualWidthProperty)) / 2
            End If

            If DrawProps.LeftReduceMyTotal Then
                left -= CDbl(MainGrid.GetValue(Window.ActualWidthProperty))
            End If

            left += DrawProps.LeftMod
            top += DrawProps.TopMod

            'Ajusta a posicao do grid caso já tenha um grid na mesma posicao
            FixGridPosition(top, left)

            'Ajuste do tamanho do canvas caso o objeto ultrapasse o limite
            FixCanvasSize(MainGrid, top, left)

            MainGrid.SetValue(Window.LeftProperty, left)
            MainGrid.SetValue(Window.TopProperty, top)
        End If

        AddHandler MainGrid.MouseLeftButtonDown, AddressOf Obj_MouseLeftButtonDown
        AddHandler MainGrid.MouseLeftButtonUp, AddressOf Obj_MouseLeftButtonUp
        AddHandler MainGrid.MouseRightButtonDown, AddressOf Obj_MouseRightButtonDown
        AddHandler MainGrid.MouseRightButtonUp, AddressOf Obj_MouseRightButtonUp
        MainGrid.Tag = New Lines(MainGrid)

        Return MainGrid
    End Function

    Private Sub FixCanvasSize(MainGrid As Grid, top As Double, ByRef left As Double)
        Dim ExtraDist As Double = 20
        Dim bottom As Double = top + MainGrid.ActualHeight
        Dim right As Double = left + MainGrid.ActualWidth

        If bottom > canMain.ActualHeight Then canMain.SetValue(HeightProperty, bottom +
+ExtraDist)
        If right > canMain.ActualWidth Then canMain.SetValue(WidthProperty, right +
ExtraDist)

        'Caso especial e mais complicado de ajustar
```

```
        If left < 0 Then
            Dim Enlarge As Double = (left * -1) + ExtraDist
            'Aumenta o tamanho do canvas
            canMain.SetValue(WidthProperty, canMain.ActualWidth + Enlarge) '+20 para dar
uma distancia da lateral
            'Joga o Node que será desenhado ao limite esquerdo com a distancia lateral
            left = ExtraDist
            'Movimenta todos os outros nodes para ajustar conforme novo tamanho
            For Each U As UIElement In canMain.Children
                If U.GetType.Equals(GetType(Grid)) Then
                    Dim G As Grid = DirectCast(U, Grid)
                    If G.Tag IsNot Nothing Then
                        G.SetValue(Window.LeftProperty,
CDbl(G.GetValue(Window.LeftProperty)) + Enlarge)
                        DirectCast(G.Tag, Lines).MoveLines()
                    End If
                End If
            Next
        End If
    End Sub

    Private Sub FixGridPosition(ByRef top As Double, ByRef left As Double)
        Dim t As Long = CLng(Math.Round(top))
        Dim l As Long = CLng(Math.Round(left))
        Dim gt As Long
        Dim gl As Long
        Dim difTA As Long
        Dim difTB As Long
        Dim difLA As Long
        Dim difLB As Long
        'Dim DT As Long = 26
        'Dim DL As Long = 64
        Dim DT As Long = 42
        Dim DL As Long = 96

        For Each U As UIElement In canMain.Children
            If U.GetType.Equals(GetType(Grid)) Then
                Dim G As Grid = DirectCast(U, Grid)
                If G.Tag IsNot Nothing Then
                    gt = CLng(Math.Round(CDbl(G.GetValue(Window.TopProperty))))
                    gl = CLng(Math.Round(CDbl(G.GetValue(Window.LeftProperty))))

                    difTA = gt - DT
                    difTB = gt + DT
                    difLA = gl - DL
                    difLB = gl + DL

                    If (t > difTA And t < difTB) And (l > difLA And l < difLB) Then
                        top = top + 100
                        FixGridPosition(top, left)
                        Exit For
                    End If
                End If
            End If
        Next
    End Sub

    Private Sub CreateLine(ByRef Grid1 As Grid, ByRef Grid2 As Grid)
```

```
        Dim Line As New Line() With {.Stroke = Brushes.Black, .StrokeThickness = 2}
        Dim MyXOffSet As Double
        Dim MyYOffSet As Double
        Dim TheirXOffSet As Double
        Dim TheirYOffSet As Double

        canMain.Children.Add(Line)
        Me.UpdateLayout()

        MyXOffSet = 0
        TheirXOffSet = 0
        If CDbl(Grid1.GetValue(Window.LeftProperty)) <
CDbl(Grid2.GetValue(Window.LeftProperty)) Then
            MyXOffSet = CDbl(Grid1.GetValue(ActualWidthProperty))
        Else
            TheirXOffSet = CDbl(Grid2.GetValue(ActualWidthProperty))
        End If

        MyYOffSet = 0
        TheirYOffSet = 0
        If CDbl(Grid1.GetValue(Window.TopProperty)) <
CDbl(Grid2.GetValue(Window.TopProperty)) Then
            MyYOffSet = CDbl(Grid1.GetValue(ActualHeightProperty))
        Else
            TheirYOffSet = CDbl(Grid2.GetValue(ActualHeightProperty))
        End If

        Line.X1 = CDbl(Grid1.GetValue(Window.LeftProperty)) + MyXOffSet
        Line.Y1 = CDbl(Grid1.GetValue(Window.TopProperty)) + MyYOffSet
        Line.X2 = CDbl(Grid2.GetValue(Window.LeftProperty)) + TheirXOffSet
        Line.Y2 = CDbl(Grid2.GetValue(Window.TopProperty)) + TheirYOffSet

        DirectCast(Grid1.Tag, Lines).MyLines.Add(New SlaveLine(DirectCast(Grid2,
UIElement), Line))
        DirectCast(Grid2.Tag, Lines).TheirLines.Add(New SlaveLine(DirectCast(Grid1,
UIElement), Line))
    End Sub

    Private Sub DestroyNodes(ByRef Node As ClassNode)
        Dim A As Lines
        Dim Ls As New List(Of SlaveLine)
        Dim CNs As List(Of ClassNode) = GetNodesFullList(Node)

        For Each CN As ClassNode In CNs
            A = DirectCast(DirectCast(CN.UIElement, Grid).Tag, Lines)

            Ls.AddRange(A.MyLines)
            Ls.AddRange(A.TheirLines)

            'Remove all lines
            For Each S As SlaveLine In Ls
                canMain.Children.Remove(S.Line)
            Next

            'Remove the Obj
            canMain.Children.Remove(CN.UIElement)
        Next
```

```
        RemoveNoveFromList(Node, MainNode, Nothing)
    End Sub

    Private Sub DestroyNodeFiltersAndOrders(Node As ClassNode)
        For Each F As ToolDBFilter In Node.Filters
            OrdenedFilters.Remove(F)
        Next

        For Each O As ToolDBOrder In Node.Orders
            OrdenedOrders.Remove(O)
        Next
    End Sub

    Private Function RemoveNoveFromList(ByRef NodeToRemove As ClassNode, ByRef
NodeToSearchIn As ClassNode, ByRef ListToRemoveFrom As List(Of ClassNode)) As Boolean
        If NodeToSearchIn.Equals(NodeToRemove) Then
            If ListToRemoveFrom Is Nothing Then
                NodeToRemove = Nothing
            Else
                ListToRemoveFrom.Remove(NodeToRemove)
            End If
            Return True
        End If

        For Each N As ClassNode In NodeToSearchIn.Parents
            If RemoveNoveFromList(NodeToRemove, N, NodeToSearchIn.Parents) Then
                Return True
            End If
        Next

        For Each N As ClassNode In NodeToSearchIn.Childs
            If RemoveNoveFromList(NodeToRemove, N, NodeToSearchIn.Childs) Then
                Return True
            End If
        Next

        Return False
    End Function

    Private Function GetNodesFullList(ByRef Node As ClassNode) As List(Of ClassNode)
        Dim R As New List(Of ClassNode)

        If Node IsNot Nothing Then R.Add(Node)

        For Each N As ClassNode In Node.Parents
            R.AddRange(GetNodesFullList(N))
        Next

        For Each N As ClassNode In Node.Childs
            R.AddRange(GetNodesFullList(N))
        Next

        Return R
    End Function

    Private Sub FixObjs()
        If MainNode Is Nothing Then Exit Sub
```

```
        Dim CNs As List(Of ClassNode) = GetNodesFullList(MainNode)
        Dim MaxWidth As Double = CDbl(canMain.GetValue(ActualWidthProperty))
        Dim MaxHeight As Double = CDbl(canMain.GetValue(ActualHeightProperty))

        If Not Me.IsLoaded Then Exit Sub

        For Each CN As ClassNode In CNs
            FixObj(CN)
        Next
    End Sub

    Private Sub FixObj(Node As ClassNode)
        Dim Width As Double
        Dim Height As Double
        Dim Left As Double
        Dim Top As Double
        Dim Right As Double
        Dim Bottom As Double

        Width = CDbl(Node.UIElement.GetValue(Window.ActualWidthProperty))
        Height = CDbl(Node.UIElement.GetValue(Window.ActualHeightProperty))
        Left = CDbl(Node.UIElement.GetValue(Window.LeftProperty))
        Top = CDbl(Node.UIElement.GetValue(Window.TopProperty))
        Right = Left + Width
        Bottom = Top + Height

        If Left < 0 Then
            Node.UIElement.SetValue(Window.LeftProperty, 0.1)
        ElseIf Right > MaxWidth Then
            Node.UIElement.SetValue(Window.LeftProperty, MaxWidth - Width)
        End If

        If Top < 0 Then
            Node.UIElement.SetValue(Window.TopProperty, 0.1)
        ElseIf Bottom > MaxHeight Then
            Node.UIElement.SetValue(Window.TopProperty, MaxHeight - Height)
        End If

        DirectCast(DirectCast(Node.UIElement, Grid).Tag, Lines).MoveLines()
    End Sub

    Private Sub GetParenstAndChilds(CI As ClassInfo)
        Dim Props As List(Of PropInfo) = CI.Props
        Dim Assocs As New List(Of AssociationInfo)
        Dim Assoc As AssociationInfo

        For Each KVC As KeyValuePair(Of String, ClassInfo) In lsbClasses.Items
            Dim C As ClassInfo = KVC.Value

            Dim LQ As IEnumerable(Of AssociationInfo) = From P As PropInfo In C.Props
                                                        Where (P.Association IsNot Nothing
AndAlso (P.Association.Schema = CI.Schema And P.Association.Table = CI.Table))
                                                        Select P.Association

            'Adicionando Childs
            For Each A As AssociationInfo In LQ
                Assoc = New AssociationInfo
                Assoc.ClassName = C.ClassName
```

```
                Assoc.Table = C.Table
                Assoc.Schema = C.Schema
                Assoc.IsChild = True
                Assoc.AssociatingKeys = A.AssociatingKeys
                Assoc.PropertyName = A.PropertyName
                Assocs.Add(Assoc)
            Next
        Next

        'Adicionando Parents
        Assocs.AddRange(From P As PropInfo In Props Where P.Association IsNot Nothing
Select P.Association)

        lsbAssociated.ItemsSource = Assocs.OrderBy(Function(x) x.Description)
    End Sub
#End Region

#Region "Event_Handlers"
    Private Sub btnCopyToClipboard_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnCopyToClipboard.Click
        Clipboard.SetText(txtCode.Text)
    End Sub

    Private Sub cmbTargetLanguage_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles
cmbTargetLanguage.SelectionChanged
        If MainNode IsNot Nothing Then
            GenerateCode(MainNode)
        End If
    End Sub

    Private Sub btnChooseClass_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnChooseClass.Click
        grdClasses.Visibility = Windows.Visibility.Visible
    End Sub

    Private Sub wpfQueryBuilder_Loaded(sender As Object, e As
System.Windows.RoutedEventArgs) Handles Me.Loaded
        Start()
    End Sub

    Private Sub btnSearch_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnSearch.Click
        Me.GetDictionary()

        lsbClasses.DisplayMemberPath = "Value"
        lsbClasses.SelectedValuePath = "Key"
        lsbClasses.ItemsSource = Classes.OrderBy(Function(x) DirectCast(x.Value,
ClassInfo).ToString)
    End Sub

    Private Sub wpfQueryBuilder_SizeChanged(sender As Object, e As
System.Windows.SizeChangedEventArgs) Handles Me.SizeChanged
        'FixObjs()
    End Sub

    Private Sub lsbClasses_MouseDoubleClick(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles lsbClasses.MouseDoubleClick
```

```
        Dim CI As ClassInfo = DirectCast(lsbClasses.SelectedItem, KeyValuePair(Of String,
ClassInfo)).Value
        Dim Create As Boolean = False
        Dim DP As DrawProps
        Dim Obj As UIElement

        If MainNode Is Nothing Then
            Create = True
        Else
            If MsgBox("A Main Node already exists. Overwrite it?", vbYesNo) = vbYes Then
                If MainNode.Parents.Count > 0 Or MainNode.Childs.Count > 0 Then
                    If MsgBox("All Parents and Child Nodes will be destroyed! Continue
with it?", vbYesNo) = vbYes Then
                        DestroyNodes(MainNode)
                        Create = True
                    End If
                Else
                    DestroyNodes(MainNode)
                    Create = True
                End If
            End If
        End If

        If Create Then
            canMain.SetValue(HeightProperty, 200.0)
            canMain.SetValue(WidthProperty, 400.0)
            Me.UpdateLayout()
            DP = New DrawProps(True, Brushes.PaleGoldenrod)
            Obj = CreateObj(CI, canMain.ActualWidth / 2, 20, DP)
            MainNode = New ClassNode(Obj, CI, Relationship.None)
            MainNode.NodeColor = DP.Color
            DirectCast(DirectCast(Obj, Grid).Tag, Lines).OwnerNode = MainNode
            MainNode.JoinType = JoinType.None
            SelectedNode = MainNode
            GenerateCode(MainNode)
        End If

        LoadAggregNodes()
        grdClasses.Visibility = Windows.Visibility.Hidden
    End Sub

    Private Sub lsbAssociated_MouseDoubleClick(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles lsbAssociated.MouseDoubleClick
        If lsbAssociated.SelectedItem Is Nothing Then Exit Sub

        Dim AI As AssociationInfo = DirectCast(lsbAssociated.SelectedItem,
AssociationInfo)
        Dim Key As String = AI.GetKey
        Dim Name As String = AI.ToString
        Dim Left As Double
        Dim Top As Double
        Dim Obj As UIElement
        Dim CI As ClassInfo
        Dim NewNode As ClassNode
        Dim DP As DrawProps

        If SelectedNode Is Nothing Then Exit Sub
```

```
        If NodeIsRepeated(SelectedNode, AI.GetNodeKey) Then
            MsgBox("Node already exists.")
            Exit Sub
        End If

        Top = CDbl(SelectedNode.UIElement.GetValue(Window.TopProperty))

        If AI.IsChild Then
            'New Node as a Child - IF AND ELSE Blocks are totally different
            If SelectedNode.Childs.Count > 0 Then Top =
CDbl(SelectedNode.Childs.Last.UIElement.GetValue(Window.TopProperty))
            Left = CDbl(SelectedNode.UIElement.GetValue(Window.LeftProperty)) +
CDbl(SelectedNode.UIElement.GetValue(Window.ActualWidthProperty)) + 50
            CI = FindClassInfo(AI)
            DP = New DrawProps(Brushes.Azure, False)
            Obj = CreateObj(AI, Left, Top + 100, DP)
            NewNode = New ClassNode(Obj, CI, AI, Relationship.Child)
            NewNode.NodeColor = DP.Color
            FixAlias(NewNode)
            SelectedNode.Childs.Add(NewNode)
        Else 'New Node as a Parent - IF AND ELSE Blocks are totally different
            If SelectedNode.Parents.Count > 0 Then Top =
CDbl(SelectedNode.Parents.Last.UIElement.GetValue(Window.TopProperty))
            Left = CDbl(SelectedNode.UIElement.GetValue(Window.LeftProperty)) - 50
            CI = FindClassInfo(AI)
            DP = New DrawProps(Brushes.PeachPuff, True)
            Obj = CreateObj(AI, Left, Top + 100, DP)
            NewNode = New ClassNode(Obj, CI, AI, Relationship.Parent)
            NewNode.NodeColor = DP.Color
            FixAlias(NewNode)
            SelectedNode.Parents.Add(NewNode)
        End If
        DirectCast(DirectCast(Obj, Grid).Tag, Lines).OwnerNode = NewNode
        CreateLine(DirectCast(SelectedNode.UIElement, Grid), DirectCast(Obj, Grid))
        'FixObj(NewNode)
        LoadAggregNodes()
        GenerateCode(MainNode)
    End Sub

    Private Sub LoadAggregNodes()
        Dim Ns As List(Of ClassNode) = GetNodesFullList(MainNode)
        cmbGroupTable.ItemsSource = Ns
        'cmbGroupField.ItemsSource = Nothing
        cmbAggregTable.ItemsSource = Ns
        'cmbAggregField.ItemsSource = Nothing
    End Sub

    Private Sub FixAlias(ByRef Node As ClassNode)
        For Each N As ClassNode In GetNodesFullList(MainNode)
            If N.SQLAlias = Node.SQLAlias Then
                Node.SQLAlias = GetNextAlias(Node.SQLAlias)
                FixAlias(Node)
                Exit For
            End If
        Next
    End Sub

    Private Function GetNextAlias(ByVal SQLAlias As String) As String
```

```
        Dim Number As New StringBuilder
        Dim NamePart As New StringBuilder

        For Each C As Char In SQLAlias
            If Char.IsDigit(C) Then
                Number.Append(C)
            Else
                NamePart.Append(Number)
                NamePart.Append(C)
                Number.Clear()
            End If
        Next

        If Number.Length = 0 Then Number.Append("0")

        Return String.Format("{0}{1}", NamePart.ToString, CStr(CDbl(Number.ToString) +
1))
    End Function

    Private Function NodeIsRepeated(Node As ClassNode, Key As String) As Boolean
        For Each CN As ClassNode In GetNodesFullList(Node)
            If CN.AssociationInfo IsNot Nothing AndAlso CN.AssociationInfo.GetNodeKey =
Key Then
                Return True
            End If
        Next

        Return False
    End Function

    Private Function FindClassInfo(AssociationInfo As AssociationInfo) As ClassInfo
        For Each KVC As KeyValuePair(Of String, ClassInfo) In lsbClasses.Items
            Dim CI As ClassInfo = KVC.Value
            If CI.GetKey = AssociationInfo.GetKey Then
                Return CI
            End If
        Next
        Return Nothing
    End Function

    Private Sub lsbClasses_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles lsbClasses.SelectionChanged
        If lsbClasses.SelectedItems.Count = 0 Then Exit Sub
        GetParenstAndChilds(DirectCast(lsbClasses.SelectedItems.Item(0), KeyValuePair(Of
String, ClassInfo)).Value)
    End Sub

    Private Sub cmbGroupTable_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles cmbGroupTable.SelectionChanged
        cmbGroupField.ItemsSource = DirectCast(cmbGroupTable.SelectedItem,
ClassNode).ClassInfo.Props
    End Sub

    Private Sub cmbAggregTable_SelectionChanged(sender As System.Object, e As
System.Windows.Controls.SelectionChangedEventArgs) Handles
cmbAggregTable.SelectionChanged
        cmbAggregField.ItemsSource = DirectCast(cmbAggregTable.SelectedItem,
ClassNode).ClassInfo.Props
```

```
    End Sub

    Private Sub cmbAddAggreg_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles cmbAddAggreg.Click
        If cmbGroupField.SelectedIndex = -1 Then
            MsgBox("Select a Field to Group By.")
            Exit Sub
        End If

        If cmbAggregField.SelectedIndex = -1 Then
            MsgBox("Select a Field to Aggregate.")
            Exit Sub
        End If

        If cmbType.SelectedIndex = -1 Then
            MsgBox("Select the Aggregator Type.")
            Exit Sub
        End If

        Dim Ag As New Aggregator
        Dim GCN As ClassNode = DirectCast(cmbGroupTable.SelectedItem, ClassNode)
        Dim GPI As PropInfo = DirectCast(cmbGroupField.SelectedItem, PropInfo)
        Dim ACN As ClassNode = DirectCast(cmbAggregTable.SelectedItem, ClassNode)
        Dim API As PropInfo = DirectCast(cmbAggregField.SelectedItem, PropInfo)

        Ag.GroupByField = String.Format("{0}.{1}", GCN.SQLAlias, GPI.PropName)
        Ag.AggregateField = String.Format("{0}.{1}", ACN.SQLAlias, API.PropName)
        Ag.AggregatorFunction = DirectCast(cmbType.SelectedItem, AggregateFunction)

        dtgAggreg.ItemsSource = Nothing
        Aggregators.Add(Ag)
        dtgAggreg.ItemsSource = Aggregators
        GenerateCode(MainNode)
    End Sub

    Private Sub dtgAggreg_MouseDoubleClick(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles dtgAggreg.MouseDoubleClick
        If dtgAggreg.SelectedIndex = -1 Then Exit Sub
        If MsgBox("Confirm removing this aggregator?", vbYesNo) = vbNo Then Exit Sub
        Dim Item As Aggregator = DirectCast(dtgAggreg.SelectedItem, Aggregator)

        dtgAggreg.ItemsSource = Nothing
        Aggregators.Remove(Item)
        dtgAggreg.ItemsSource = Aggregators
    End Sub

    Private Sub scrMain_MouseLeftButtonUp(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles scrMain.MouseLeftButtonUp
        EmptyHolders()
    End Sub

    'Private Sub btnNewDSN_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnNewDSN.Click
    '    Try
    '
Shell(System.IO.Path.Combine(String.Concat(Environment.GetEnvironmentVariable("windir"),
"\"), "system32\odbcad32.exe"), AppWinStyle.NormalFocus, True)
    '        GetDSNs()
```

```
    '    Catch ex As Exception
    '        MsgBox(ex.Message)
    '    End Try
    'End Sub

    'Private Sub GetDSNs()
    '    Dim Reg As RegistryKey

    '    Reg = Registry.LocalMachine.OpenSubKey("Software\ODBC\ODBC.INI\ODBC Data
Sources", True)
    '    cmbDSN.Items.Clear()

    '    For Each DSN As String In Reg.GetValueNames
    '        cmbDSN.Items.Add(DSN)
    '    Next
    'End Sub

    Private Sub scrMain_MouseMove(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles scrMain.MouseMove
        FrameMouseMove(e)
    End Sub

#End Region

#Region "Objs_Events"
    Private Sub Obj_MouseLeftButtonDown(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs)
        Dim InnerGrid As Grid
        Dim MainGrid As Grid

        If canMain.Children.Contains(DirectCast(sender, UIElement)) Then
            'Fix the color os the last selected
            If DraggingElement IsNot Nothing Then
                InnerGrid = DirectCast(DirectCast(DirectCast(DraggingElement,
Grid).Children(0), Border).Child, Grid)
                InnerGrid.Background = LastDragColor
            End If

            'Activate dragging and saves the mouse start position for reference
            IsMouseDraggingObj = True
            DraggingElement = DirectCast(sender, UIElement)
            SelectedNode = GetNodeFromUIElement(DraggingElement)

            'Sets the color of the selected Grid
            MainGrid = DirectCast(DraggingElement, Grid)
            InnerGrid = DirectCast(DirectCast(MainGrid.Children(0), Border).Child, Grid)
            If InnerGrid.Background.Equals(DirectCast(MainGrid.Tag,
Lines).OwnerNode.NodeColor) Then
                ClearSelections()
            End If
            LastDragColor = InnerGrid.Background
            InnerGrid.Background = Brushes.CadetBlue
            SelectedElements.Add(DirectCast(DraggingElement, Grid))

            'Calculate Coordenates
            For Each G As Grid In SelectedElements
```

```
                DirectCast(G.Tag, Lines).OrigPoint = New
Point(CDbl(G.GetValue(Window.LeftProperty)) - e.GetPosition(canMain).X,
CDbl(G.GetValue(Window.TopProperty)) - e.GetPosition(canMain).Y)
            Next

            'List the Parents and Child of the selected Node
            GetParenstAndChilds(GetNodeFromUIElement(DraggingElement).ClassInfo)
        End If
    End Sub

    Private Function GetNodeFromUIElement(UIElement As UIElement) As ClassNode
        Dim Node = From CN As ClassNode In GetNodesFullList(MainNode) Where
CN.UIElement.Equals(DirectCast(DirectCast(DraggingElement, Grid).Tag, Lines).Owner)
Select CN
        Return Node.First
    End Function

    Private Sub Obj_MouseRightButtonDown(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs)
        Dim InnerGrid As Grid

        If canMain.Children.Contains(DirectCast(sender, UIElement)) Then
            'Fix the color os the last selected
            If DraggingElement IsNot Nothing Then
                InnerGrid = DirectCast(DirectCast(DirectCast(DraggingElement,
Grid).Children(0), Border).Child, Grid)
                InnerGrid.Background = LastDragColor
            End If

            DraggingElement = DirectCast(sender, UIElement)
            SelectedNode = GetNodeFromUIElement(DraggingElement)

            'Sets the color of the selected Grid
            InnerGrid = DirectCast(DirectCast(DirectCast(DraggingElement,
Grid).Children(0), Border).Child, Grid)
            LastDragColor = InnerGrid.Background
            InnerGrid.Background = Brushes.CadetBlue
        End If

        Obj_MouseLeftButtonDown(sender, e)
    End Sub

    Private Sub Obj_MouseLeftButtonUp(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs)
        EmptyHolders()
    End Sub

    Private Sub Obj_MouseRightButtonUp(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs)
        EmptyHolders()
    End Sub

    Private Sub Remove_Click(sender As Object, e As System.Windows.RoutedEventArgs)
        If MsgBox("Removing this Node will remove his Parents and Childs Nodes as well!
Continue with it?", vbYesNo) = vbNo Then Exit Sub
        DestroyNodeFiltersAndOrders(SelectedNode)
        DestroyNodes(SelectedNode)
        SelectedNode = Nothing
```

```
        lsbAssociated.ItemsSource = New List(Of AssociationInfo)
        GenerateCode(MainNode)
    End Sub

    Private Sub Properties_Click(sender As Object, e As System.Windows.RoutedEventArgs)
        Dim Win As frmNodeProperties

        If SelectedNode.Equals(MainNode) Then
            Win = New frmNodeProperties(SelectedNode, OrdenedFilters, OrdenedOrders,
True)
        Else
            Win = New frmNodeProperties(SelectedNode, OrdenedFilters, OrdenedOrders)
        End If

        Win.ShowDialog()
        GenerateCode(MainNode)
        FixGrids()
    End Sub

    Private Sub FixGrids()
        dtgOrders.ItemsSource = New List(Of Order)
        dtgOrders.ItemsSource = OrdenedOrders
        dtgFilters.ItemsSource = New List(Of DBFilter)
        dtgFilters.ItemsSource = OrdenedFilters
    End Sub
#End Region

#Region "Canvas_Events"
    ''' <summary>
    ''' Clicking in any place a canvas that doesn't have an obj will unselect any
selected objs
    ''' </summary>
    Private Sub canMain_MouseLeftButtonDown(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles canMain.MouseLeftButtonDown
        Dim MainGrid As Grid
        Dim InnerGrid As Grid
        Dim Node As ClassNode

        If DraggingElement IsNot Nothing And Not IsMouseDraggingObj Then
            MainGrid = DirectCast(DraggingElement, Grid)
            Node = DirectCast(MainGrid.Tag, Lines).OwnerNode
            InnerGrid = DirectCast(DirectCast(MainGrid.Children(0), Border).Child, Grid)
            InnerGrid.Background = Node.NodeColor
            DraggingElement = Nothing
            ClearSelections()
        End If

        If Not IsMouseDraggingObj Then
            IsDraggingRectangle = True
            StartPoint = e.GetPosition(canMain)

            Rectangle = New Rectangle With {.Stroke = Brushes.Gray, .StrokeThickness = 2}
            Canvas.SetLeft(Rectangle, StartPoint.X)
            Canvas.SetTop(Rectangle, StartPoint.Y)
            canMain.Children.Add(Rectangle)
        End If
    End Sub
```

```
    Private Sub ClearSelections()
        'Limpa as seleções atuais
        For Each G As Grid In SelectedElements
            DirectCast(DirectCast(G.Children(0), Border).Child, Grid).Background =
DirectCast(G.Tag, Lines).OwnerNode.NodeColor
        Next
        SelectedElements.Clear()
    End Sub

    ''' <summary>
    ''' If mouse is dragging an obj than move the dragging element and his lines
    ''' </summary>
    Private Sub canMain_MouseMove(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles canMain.MouseMove
        FrameMouseMove(e)
    End Sub

    Private Sub FrameMouseMove(e As System.Windows.Input.MouseEventArgs)
        If IsMouseDraggingObj Then
            Dim MovX As Double
            Dim MovY As Double
            Dim OP As Point

            For Each G As Grid In SelectedElements
                OP = DirectCast(G.Tag, Lines).OrigPoint
                MovX = e.GetPosition(canMain).X + OP.X
                MovY = e.GetPosition(canMain).Y + OP.Y

                If MovX < 0 Then
                    MovX = 0
                ElseIf MovX + G.ActualWidth > canMain.ActualWidth Then
                    MovX = canMain.ActualWidth - G.ActualWidth
                End If

                If MovY < 0 Then
                    MovY = 0
                ElseIf MovY + G.ActualHeight > canMain.ActualHeight Then
                    MovY = canMain.ActualHeight - G.ActualHeight
                End If

                G.SetValue(Window.TopProperty, MovY)
                G.SetValue(Window.LeftProperty, MovX)
                DirectCast(G.Tag, Lines).MoveLines()
            Next

        ElseIf IsDraggingRectangle Then
            Dim Pos As Point = e.GetPosition(canMain)

            Dim X As Double = Math.Min(Pos.X, StartPoint.X)
            Dim Y As Double = Math.Min(Pos.Y, StartPoint.Y)

            Dim W As Double = Math.Max(Pos.X, StartPoint.X) - X
            Dim H As Double = Math.Max(Pos.Y, StartPoint.Y) - Y

            Rectangle.Width = W
            Rectangle.Height = H

            Canvas.SetLeft(Rectangle, X)
```

```
            Canvas.SetTop(Rectangle, Y)
        End If
    End Sub

    Private Sub canMain_MouseUp(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles canMain.MouseUp
        EmptyHolders()
    End Sub

    Private Sub canMain_MouseLeave(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles canMain.MouseLeave
        If e.LeftButton = MouseButtonState.Released Then
            EmptyHolders()
        End If
    End Sub

    Private Sub canMain_MouseEnter(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles canMain.MouseEnter
        If e.LeftButton = MouseButtonState.Released Then
            EmptyHolders()
        End If
    End Sub
#End Region

#Region "Close_Events"
    Private Sub brdClose_MouseEnter(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles brdClose.MouseEnter
        brdClose.Background = Brushes.White
    End Sub

    Private Sub brdClose_MouseLeave(sender As Object, e As
System.Windows.Input.MouseEventArgs) Handles brdClose.MouseLeave
        brdClose.Background = Brushes.LightGray
    End Sub

    Private Sub brdClose_MouseLeftButtonDown(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles brdClose.MouseLeftButtonDown
        brdClose.Background = Brushes.Gray
    End Sub

    Private Sub brdClose_MouseLeftButtonUp(sender As Object, e As
System.Windows.Input.MouseButtonEventArgs) Handles brdClose.MouseLeftButtonUp
        brdClose.Background = Brushes.LightGray
        grdClasses.Visibility = Windows.Visibility.Hidden
    End Sub
#End Region

#Region "Code Generator"
    Private MainJoinsList As String

    Private Sub GenerateCode(ByRef Node As ClassNode)
        Dim SpecifiedFields As Boolean = False

        For Each N As ClassNode In GetNodesFullList(Node)
            If N.SpecifiedFields Then
                SpecifiedFields = True
                Exit For
            End If
```

```
        Next

        If CStr(cmbTargetLanguage.SelectedItem) = "Visual Basic" Then
            GenerateVBCode(Node, SpecifiedFields)
        Else
            GenerateCSCode(Node, SpecifiedFields)
        End If
    End Sub

    'Private Sub btnTest_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnTest.Click
    '    Dim DBLink As New GenDAL(New DBFactory(DBMapper.DBProvider.ODBC,
String.Format("DSN={0}", cmbDSN.Text)))
    'End Sub
#End Region

#Region "Code Generator VB"
    Private Sub GenerateVBCode(ByRef Node As ClassNode, SpecifiedFields As Boolean)
        Dim Code As New StringBuilder
        Dim Joins As String = GetVBJoins(Node, String.Format("{0}Joins", Node.SQLAlias),
SpecifiedFields)
        Dim FiltersOrders As String = GetVBFiltersAndOrders(Node)
        Dim Aggregs = GetVBAggregators()

        MainJoinsList = String.Format("{0}Joins", Node.SQLAlias)

        Code.AppendFormat("'------------------------------------------------------------
----------------------------------{0}", vbCrLf)
        Code.AppendFormat("' AUTO GENERATED CODE - GenDAL Query Builder Tool
DBClassMapper Version 3.2 (C# and VB.Net Code){0}", vbCrLf)
        Code.AppendFormat("'------------------------------------------------------------
----------------------------------{0}", vbCrLf)
        Code.AppendFormat("Dim DBLink as New GenDAL(New
DBFactory(DBProvider.No_Provider_Found, ""Connection Here""))){0}", vbCrLf)
        Code.AppendFormat("Dim {0} as New List(of DBJoin){1}", MainJoinsList, vbCrLf)

        If Joins.Length > 0 Then Code.AppendFormat("{0}{1}", Joins, vbCrLf)
        If FiltersOrders.Length > 0 Then Code.AppendFormat("{0}{1}", FiltersOrders,
vbCrLf)
        If Aggregs.Length > 0 Then Code.AppendFormat("{0}{1}", Aggregs, vbCrLf)

        Code.AppendFormat("Return DBLink.List(of {0})(""{1}"", {1}Joins.ToArray)",
Node.ClassInfo.ClassName, Node.SQLAlias)
        Code.AppendFormat("{0}'------------------------------------------------------------
------------------------------------{0}", vbCrLf)
        Code.AppendFormat("' END OF AUTO GENERATED CODE - USE DBClassMapper Version 3.2
OR LATER TO MAINTAIN THIS CODE (VB){0}", vbCrLf)
        Code.AppendFormat("'------------------------------------------------------------
----------------------------------{0}", vbCrLf)

        txtCode.Text = Code.ToString
    End Sub

    Private Function GetVBJoins(ByRef TopNode As ClassNode, ByVal JoinVarToAdd As String,
SpecifiedFields As Boolean) As String
        Dim Join As New StringBuilder
        Dim SF As New StringBuilder
```

```
        If SpecifiedFields Then
            For Each P As PropInfo In TopNode.ClassInfo.Props
                If P.Selected Then SF.AppendFormat(", ""{0}""", P.PropName)
            Next

            If SF.Length > 0 Then Join.AppendFormat("DBLink.AddFields(""{0}""{1}){2}",
TopNode.SQLAlias, SF, vbCrLf)
        End If

        For Each N As ClassNode In TopNode.Parents
            Join.AppendFormat("Dim Join{0} as New DBJoin(GetType({1}), ""{0}"",
JoinType.{2}, Relationship.{3}, ""{4}""){5}", N.SQLAlias, N.ClassInfo.ClassName,
N.JoinType, N.Relationship, N.AssociationInfo.PropertyName, vbCrLf)

            If JoinVarToAdd = MainJoinsList Then
                Join.AppendFormat("{0}.Add(Join{1}){2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            Else
                Join.AppendFormat("{0}.AddJoins(Join{1}){2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            End If

            If N.AssociationInfo IsNot Nothing Then
                Join.Append(GetVBJoins(N, String.Format("Join{0}", N.SQLAlias),
SpecifiedFields))
            End If
        Next

        For Each N As ClassNode In TopNode.Childs
            Join.AppendFormat("Dim Join{0} as New DBJoin(GetType({1}), ""{0}"",
JoinType.{2}, Relationship.{3}, ""{4}""){5}", N.SQLAlias, N.ClassInfo.ClassName,
N.JoinType, N.Relationship, N.AssociationInfo.PropertyName, vbCrLf)

            If JoinVarToAdd = MainJoinsList Then
                Join.AppendFormat("{0}.Add(Join{1}){2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            Else
                Join.AppendFormat("{0}.AddJoins(Join{1}){2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            End If

            If N.AssociationInfo IsNot Nothing Then
                Join.Append(GetVBJoins(N, String.Format("Join{0}", N.SQLAlias),
SpecifiedFields))
            End If
        Next

        Return Join.ToString
    End Function

    Private Function GetVBFiltersAndOrders(Node As ClassNode) As String
        Dim Filters As New StringBuilder
        Dim Orders As New StringBuilder
        Dim BrakeLine As String = ""

        For Each F As ToolDBFilter In OrdenedFilters
            If F.Var2 Is Nothing OrElse F.Var2.Equals(String.Empty) Then
```

```
                Filters.AppendFormat("DBLink.AddFilter(""{0}.{1}"", OperatorType.{2},
{3}){4}", F.Node.SQLAlias, F.Field, F.OperatorType, F.Var1, vbCrLf)
            Else
                Filters.AppendFormat("DBLink.AddFilter(""{0}.{1}"", {2}, {3}){4}",
F.Node.SQLAlias, F.Field, F.Var1, F.Var2, vbCrLf)
            End If
        Next

        For Each O As ToolDBOrder In OrdenedOrders
            Orders.AppendFormat("DBLink.AddSort(""{0}.{1}"", OrderMode.{2}){3}",
O.Node.SQLAlias, O.Field, O.Mode, vbCrLf)
        Next

        If Filters.Length > 0 And Orders.Length > 0 Then BrakeLine = vbCrLf

        Return String.Format("{0}{1}{2}", Filters, BrakeLine, Orders)
    End Function

    Private Function GetVBAggregators() As String
        Dim Ags As New StringBuilder

        For Each A As Aggregator In Aggregators
            Ags.AppendFormat("DBLink.AddAggregator(New Aggregator(""{0}"", ""{1}"",
AggregateFunction.{2})){3}", A.GroupByField, A.AggregateField,
A.AggregatorFunction.ToString, vbCrLf)
        Next

        Return Ags.ToString
    End Function
#End Region

#Region "Code Generator CS"
    Private Sub GenerateCSCode(ByRef Node As ClassNode, SpecifiedFields As Boolean)
        Dim Code As New StringBuilder
        MainJoinsList = String.Format("joins{0}", Node.SQLAlias)
        Dim Joins As String = GetCSJoins(Node, MainJoinsList, SpecifiedFields)
        Dim FiltersOrders As String = GetCSFiltersAndOrders(Node)
        Dim Aggregs As String = GetCSAggregators()

        Code.AppendFormat("//------------------------------------------------------------
----------------------------------{0}", vbCrLf)
        Code.AppendFormat("// AUTO GENERATED CODE - GenDAL Query Builder Tool
DBClassMapper Version 3.2 (C# and VB.Net Code){0}", vbCrLf)
        Code.AppendFormat("//------------------------------------------------------------
----------------------------------{0}", vbCrLf)
        Code.AppendFormat("GenDAL dblink = new GenDAL(new
DBFactory(DBProvider.No_Provider_Found, ""Connection Here""));{0}", vbCrLf)
        Code.AppendFormat("List<DBJoin> {0} = new List<DBJoin>();{1}", MainJoinsList,
vbCrLf)

        If Joins.Length > 0 Then Code.AppendFormat("{0}{1}", Joins, vbCrLf)
        If FiltersOrders.Length > 0 Then Code.AppendFormat("{0}{1}", FiltersOrders,
vbCrLf)
        If Aggregs.Length > 0 Then Code.AppendFormat("{0}{1}", Aggregs, vbCrLf)

        Code.AppendFormat("return dblink.List<{0}>(""{1}"", {2}.ToArray());",
Node.ClassInfo.ClassName, Node.SQLAlias, MainJoinsList)
```

```
        Code.AppendFormat("{0}//------------------------------------------------------------
-----------------------------------{0}", vbCrLf)
        Code.AppendFormat("// END OF AUTO GENERATED CODE - USE DBClassMapper Version 3.2
OR LATER TO MAINTAIN THIS CODE (C#){0}", vbCrLf)
        Code.AppendFormat("//------------------------------------------------------------
---------------------------------{0}", vbCrLf)

        txtCode.Text = Code.ToString
    End Sub

    Private Function GetCSJoins(ByRef TopNode As ClassNode, ByVal JoinVarToAdd As String,
SpecifiedFields As Boolean) As String
        Dim Join As New StringBuilder
        Dim SF As New StringBuilder

        If SpecifiedFields Then
            For Each P As PropInfo In TopNode.ClassInfo.Props
                If P.Selected Then SF.AppendFormat(", ""{0}""", P.PropName)
            Next

            If SF.Length > 0 Then Join.AppendFormat("dblink.AddFields(""{0}""{1});{2}",
TopNode.SQLAlias, SF, vbCrLf)
        End If

        For Each N As ClassNode In TopNode.Parents
            Join.AppendFormat("DBJoin join{0} = new DBJoin(typeof({1}), ""{0}"",
JoinType.{2}, Relationship.{3}, ""{4}"");{5}", N.SQLAlias, N.ClassInfo.ClassName,
N.JoinType, N.Relationship, N.AssociationInfo.PropertyName, vbCrLf)

            If JoinVarToAdd = MainJoinsList Then
                Join.AppendFormat("{0}.Add(join{1});{2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            Else
                Join.AppendFormat("{0}.AddJoins(join{1});{2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            End If

            If N.AssociationInfo IsNot Nothing Then
                Join.Append(GetCSJoins(N, String.Format("join{0}", N.SQLAlias),
SpecifiedFields))
            End If
        Next

        For Each N As ClassNode In TopNode.Childs
            Join.AppendFormat("DBJoin join{0} = new DBJoin(typeof({1}), ""{0}"",
JoinType.{2}, Relationship.{3}, ""{4}"");{5}", N.SQLAlias, N.ClassInfo.ClassName,
N.JoinType, N.Relationship, N.AssociationInfo.PropertyName, vbCrLf)

            If JoinVarToAdd = MainJoinsList Then
                Join.AppendFormat("{0}.Add(join{1});{2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            Else
                Join.AppendFormat("{0}.AddJoins(join{1});{2}", JoinVarToAdd, N.SQLAlias,
vbCrLf)
            End If

            If N.AssociationInfo IsNot Nothing Then
```

```
                Join.Append(GetCSJoins(N, String.Format("join{0}", N.SQLAlias),
SpecifiedFields))
            End If
        Next

        Return Join.ToString
    End Function

    Private Function GetCSFiltersAndOrders(Node As ClassNode) As String
        Dim Filters As New StringBuilder
        Dim Orders As New StringBuilder
        Dim BrakeLine As String = ""

        For Each F As ToolDBFilter In OrdenedFilters
            If F.Var2 Is Nothing OrElse F.Var2.Equals(String.Empty) Then
                Filters.AppendFormat("dblink.AddFilter(""{0}.{1}"", OperatorType.{2},
{3});{4}", F.Node.SQLAlias, F.Field, F.OperatorType, F.Var1, vbCrLf)
            Else
                Filters.AppendFormat("dblink.AddFilter(""{0}.{1}"", {2}, {3});{4}",
F.Node.SQLAlias, F.Field, F.Var1, F.Var2, vbCrLf)
            End If
        Next

        For Each O As ToolDBOrder In OrdenedOrders
            Orders.AppendFormat("dblink.AddSort(""{0}.{1}"", OrderMode.{2});{3}",
O.Node.SQLAlias, O.Field, O.Mode, vbCrLf)
        Next

        If Filters.Length > 0 And Orders.Length > 0 Then BrakeLine = vbCrLf

        Return String.Format("{0}{1}{2}", Filters, BrakeLine, Orders)
    End Function

    Private Function GetCSAggregators() As String
        Dim Ags As New StringBuilder

        For Each A As Aggregator In Aggregators
            Ags.AppendFormat("dblink.AddAggregator(new Aggregator(""{0}"", ""{1}"",
AggregateFunction.{2}));{3}", A.GroupByField, A.AggregateField,
A.AggregatorFunction.ToString, vbCrLf)
        Next

        Return Ags.ToString
    End Function
#End Region

#Region "Orders"
    Private Sub btnUpOrder_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnUpOrder.Click
        If dtgOrders.SelectedItem Is Nothing Then Exit Sub
        If dtgOrders.Items.Count = 1 Then Exit Sub

        Dim O As ToolDBOrder = DirectCast(dtgOrders.SelectedItem, ToolDBOrder)
        Dim i As Integer = OrdenedOrders.IndexOf(O)
        OrdenedOrders.Remove(O)

        If i = 0 Then
            OrdenedOrders.Insert(0, O)
```

```
        Else
            OrdenedOrders.Insert(i - 1, O)
        End If

        FixGrids()
        dtgOrders.SelectedItem = O
        GenerateCode(MainNode)
    End Sub

    Private Sub btnDownOrder_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnDownOrder.Click
        If dtgOrders.SelectedItem Is Nothing Then Exit Sub
        If dtgOrders.Items.Count = 1 Then Exit Sub

        Dim O As ToolDBOrder = DirectCast(dtgOrders.SelectedItem, ToolDBOrder)
        Dim i As Integer = OrdenedOrders.IndexOf(O)

        If O.Equals(OrdenedOrders.Last) Then Exit Sub

        OrdenedOrders.Remove(O)

        If i > OrdenedOrders.IndexOf(OrdenedOrders.Last) Then
            OrdenedOrders.Insert(OrdenedOrders.IndexOf(OrdenedOrders.Last), O)
        Else
            OrdenedOrders.Insert(i + 1, O)
        End If

        FixGrids()
        dtgOrders.SelectedItem = O
        GenerateCode(MainNode)
    End Sub

    Private Sub btnUpFilter_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnUpFilter.Click
        If dtgFilters.SelectedItem Is Nothing Then Exit Sub
        If dtgFilters.Items.Count = 1 Then Exit Sub

        Dim F As ToolDBFilter = DirectCast(dtgFilters.SelectedItem, ToolDBFilter)
        Dim i As Integer = OrdenedFilters.IndexOf(F)
        OrdenedFilters.Remove(F)

        If i = 0 Then
            OrdenedFilters.Insert(0, F)
        Else
            OrdenedFilters.Insert(i - 1, F)
        End If

        FixGrids()
        dtgFilters.SelectedItem = F
        GenerateCode(MainNode)
    End Sub

    Private Sub btnDownFilter_Click(sender As System.Object, e As
System.Windows.RoutedEventArgs) Handles btnDownFilter.Click
        If dtgFilters.SelectedItem Is Nothing Then Exit Sub
        If dtgFilters.Items.Count = 1 Then Exit Sub

        Dim F As ToolDBFilter = DirectCast(dtgFilters.SelectedItem, ToolDBFilter)
```

```
        Dim i As Integer = OrdenedFilters.IndexOf(F)

        If F.Equals(OrdenedFilters.Last) Then Exit Sub

        OrdenedFilters.Remove(F)

        If i > OrdenedFilters.IndexOf(OrdenedFilters.Last) Then
            OrdenedFilters.Insert(OrdenedFilters.IndexOf(OrdenedFilters.Last), F)
        Else
            OrdenedFilters.Insert(i + 1, F)
        End If

        FixGrids()
        dtgFilters.SelectedItem = F
        GenerateCode(MainNode)
    End Sub
#End Region

End Class
```

csclass.cs

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;

namespace XNAMESPACE
{
    //MAP
    public class Class1
    {

    }
}
```

vbclass.vb

```
Imports DBManager.DBMapper
Imports System.ComponentModel.DataAnnotations

'MAP
Public Class Class1

End Class
```

Connect.vb

```
Imports System
Imports Microsoft.VisualStudio.CommandBars
Imports Extensibility
Imports EnvDTE
Imports EnvDTE80
Imports Microsoft.Win32
Imports System.Runtime.Remoting
Imports System.Runtime.Serialization
```

```
Public Class Connect

    Implements IDTExtensibility2
    Implements IDTCommandTarget

    Private _applicationObject As DTE2
    Private _addInInstance As AddIn
    Private _GetCommandBarsHandler As CommandBarEvents
    Private _DBClassMapperHandler As CommandBarEvents
    'Private _SafeDomain As AppDomain

    '''<summary>Implements the constructor for the Add-in object. Place your
initialization code within this method.</summary>
    Public Sub New()

    End Sub

    '''<summary>Implements the OnConnection method of the IDTExtensibility2 interface.
Receives notification that the Add-in is being loaded.</summary>
    '''<param name='application'>Root object of the host application.</param>
    '''<param name='connectMode'>Describes how the Add-in is being loaded.</param>
    '''<param name='addInInst'>Object representing this Add-in.</param>
    '''<remarks></remarks>
    Public Sub OnConnection(ByVal application As Object, ByVal connectMode As
ext_ConnectMode, ByVal addInInst As Object, ByRef custom As Array) Implements
IDTExtensibility2.OnConnection
        _applicationObject = CType(application, DTE2)
        _addInInstance = CType(addInInst, AddIn)

        If connectMode = ext_ConnectMode.ext_cm_UISetup Then

            Dim commands As Commands2 = CType(_applicationObject.Commands, Commands2)
            Dim toolsMenuName As String = "Tools"

            'Place the command on the tools menu.
            'Find the MenuBar command bar, which is the top-level command bar holding all
the main menu items:
            Dim commandBars As CommandBars = CType(_applicationObject.CommandBars,
CommandBars)
            Dim menuBarCommandBar As CommandBar = commandBars.Item("MenuBar")

            'Find the Tools command bar on the MenuBar command bar:
            Dim toolsControl As CommandBarControl =
menuBarCommandBar.Controls.Item(toolsMenuName)
            Dim toolsPopup As CommandBarPopup = CType(toolsControl, CommandBarPopup)

            Try
                'Add a command to the Commands collection:
                'Dim command As Command = commands.AddNamedCommand2(_addInInstance,
"DBClassMapper", "DBClassMapper", "Executes the command for DBClassMapper", False,
My.Resources.Database, Nothing, CType(vsCommandStatus.vsCommandStatusSupported, Integer)
+ CType(vsCommandStatus.vsCommandStatusEnabled, Integer),
vsCommandStyle.vsCommandStylePictAndText,
vsCommandControlType.vsCommandControlTypeButton)
                Dim command As Command

                If _applicationObject.Version = "10.0" Then
```

```
                    command = commands.AddNamedCommand2(_addInInstance, "DBClassMapper",
"DBClassMapper", "Executes the command for DBClassMapper", False, My.Resources.Database,
Nothing, CType(vsCommandStatus.vsCommandStatusSupported, Integer) +
CType(vsCommandStatus.vsCommandStatusEnabled, Integer),
vsCommandStyle.vsCommandStylePictAndText,
vsCommandControlType.vsCommandControlTypeButton)
                Else
                    command = commands.AddNamedCommand(_addInInstance, "DBClassMapper",
"DBClassMapper", "Executes the command for DBClassMapper", True, 13, Nothing,
CType(vsCommandStatus.vsCommandStatusSupported, Integer) +
CType(vsCommandStatus.vsCommandStatusEnabled, Integer) +
vsCommandStyle.vsCommandStylePictAndText +
vsCommandControlType.vsCommandControlTypeButton)
                End If

                'Find the appropriate command bar on the MenuBar command bar:
                command.AddControl(toolsPopup.CommandBar, 1)

            Catch argumentException As System.ArgumentException
                'If we are here, then the exception is probably because a command with
that name
                '  already exists. If so there is no need to recreate the command and we
can
                '  safely ignore the exception.
            End Try

        End If
    End Sub

    '''<summary>Implements the OnDisconnection method of the IDTExtensibility2 interface.
Receives notification that the Add-in is being unloaded.</summary>
    '''<param name='disconnectMode'>Describes how the Add-in is being unloaded.</param>
    '''<param name='custom'>Array of parameters that are host application
specific.</param>
    '''<remarks></remarks>
    Public Sub OnDisconnection(ByVal disconnectMode As ext_DisconnectMode, ByRef custom
As Array) Implements IDTExtensibility2.OnDisconnection
        'AppDomain.Unload(_SafeDomain)
    End Sub

    '''<summary>Implements the OnAddInsUpdate method of the IDTExtensibility2 interface.
Receives notification that the collection of Add-ins has changed.</summary>
    '''<param name='custom'>Array of parameters that are host application
specific.</param>
    '''<remarks></remarks>
    Public Sub OnAddInsUpdate(ByRef custom As Array) Implements
IDTExtensibility2.OnAddInsUpdate
    End Sub

    '''<summary>Implements the OnStartupComplete method of the IDTExtensibility2
interface. Receives notification that the host application has completed
loading.</summary>
    '''<param name='custom'>Array of parameters that are host application
specific.</param>
    '''<remarks></remarks>
    Public Sub OnStartupComplete(ByRef custom As Array) Implements
IDTExtensibility2.OnStartupComplete
    End Sub
```

```
    '''<summary>Implements the OnBeginShutdown method of the IDTExtensibility2 interface.
Receives notification that the host application is being unloaded.</summary>
    '''<param name='custom'>Array of parameters that are host application
specific.</param>
    '''<remarks></remarks>
    Public Sub OnBeginShutdown(ByRef custom As Array) Implements
IDTExtensibility2.OnBeginShutdown
    End Sub

    '''<summary>Implements the QueryStatus method of the IDTCommandTarget interface. This
is called when the command's availability is updated</summary>
    '''<param name='commandName'>The name of the command to determine state for.</param>
    '''<param name='neededText'>Text that is needed for the command.</param>
    '''<param name='status'>The state of the command in the user interface.</param>
    '''<param name='commandText'>Text requested by the neededText parameter.</param>
    '''<remarks></remarks>
    Public Sub QueryStatus(ByVal commandName As String, ByVal neededText As
vsCommandStatusTextWanted, ByRef status As vsCommandStatus, ByRef commandText As Object)
Implements IDTCommandTarget.QueryStatus
        If neededText = vsCommandStatusTextWanted.vsCommandStatusTextWantedNone Then
            If commandName = "DBClassMapper.Connect.DBClassMapper" Then
                status = CType(vsCommandStatus.vsCommandStatusEnabled +
vsCommandStatus.vsCommandStatusSupported, vsCommandStatus)
            Else
                status = vsCommandStatus.vsCommandStatusUnsupported
            End If
        End If
    End Sub

    '''<summary>Implements the Exec method of the IDTCommandTarget interface. This is
called when the command is invoked.</summary>
    '''<param name='commandName'>The name of the command to execute.</param>
    '''<param name='executeOption'>Describes how the command should be run.</param>
    '''<param name='varIn'>Parameters passed from the caller to the command
handler.</param>
    '''<param name='varOut'>Parameters passed from the command handler to the
caller.</param>
    '''<param name='handled'>Informs the caller if the command was handled or
not.</param>
    '''<remarks></remarks>
    Public Sub Exec(ByVal commandName As String, ByVal executeOption As
vsCommandExecOption, ByRef varIn As Object, ByRef varOut As Object, ByRef handled As
Boolean) Implements IDTCommandTarget.Exec
        handled = False
        If executeOption = vsCommandExecOption.vsCommandExecOptionDoDefault Then
            If commandName = "DBClassMapper.Connect.DBClassMapper" Then
                Try
                    Dim MainWin As New frmMainWindow(_applicationObject)
                    MainWin.Show()
                Catch ex As Exception
                    MsgBox(ex.Message)
                End Try

                handled = True
                Exit Sub
            End If
        End If
```

```
    End Sub

End Class
```

InstallAddIn.vb

```
Imports System.ComponentModel
Imports System.Configuration.Install
Imports Microsoft.Win32
Imports System.IO
Imports System.Collections.Specialized

Public Class InstallAddIn

    Public Sub New()
        MyBase.New()

        'This call is required by the Component Designer.
        InitializeComponent()

        'Add initialization code after the call to InitializeComponent

    End Sub

    Private Sub InstallAddIn_AfterInstall(ByVal sender As Object, ByVal e As
System.Configuration.Install.InstallEventArgs) Handles Me.AfterInstall
        Dim RegBase As RegistryKey
        Dim RegSub As RegistryKey
        Dim RegDelta As RegistryKey
        Dim Caminho As String = ""
        Dim ExisteVS10 As Boolean = False
        Dim ExisteVS12 As Boolean = False

        'Pega o caminho da instalação
        Try
            Caminho = Me.Context.Parameters.Item("assemblyPath")
            Caminho = Caminho.Substring(0, Caminho.LastIndexOf("\"))
        Catch Ex As Exception
            MsgBox(Ex.ToString)
            MyBase.Rollback(e.SavedState)
            MyBase.Uninstall(e.SavedState)
            Exit Sub
        End Try

        'Abre o registro do VS (Requer VS 2010/2012)
        Try
            RegBase = Registry.LocalMachine.OpenSubKey("Software\Microsoft\VisualStudio")
            If RegBase Is Nothing Then
                Throw New Exception("Could not load RegKey
[Software\Microsoft\VisualStudio]")
            End If
        Catch Ex As Exception
            MsgBox("DBManager requer o Visual Studio 2010/2012 Instalado para funcionar.
Instale o Visual Studio e execute o Setup novamente.")
            MyBase.Rollback(e.SavedState)
            MyBase.Uninstall(e.SavedState)
            Exit Sub
```

```
        End Try

        'Tenta gerar o registry que indica o local onde foi instalado o DBClassMapper
        Try
            RegDelta =
Registry.LocalMachine.CreateSubKey("Software\Deltacon\DBClassMapper")
            RegDelta.SetValue("DBClassMapperPath", Caminho)
        Catch ex As Exception
            MsgBox(ex.ToString)
            MyBase.Rollback(e.SavedState)
            MyBase.Uninstall(e.SavedState)
            Exit Sub
        End Try

        'Tenta gerar os registrys necessários para o VS 2010
        Try
            RegSub = RegBase.OpenSubKey("10.0", True)
            If RegSub Is Nothing Then
                Throw New Exception("Visual Studio 2010 not found.")
            End If
            Try
                RegSub = RegSub.CreateSubKey("AutomationOptions\LookInFolders")
                RegSub.SetValue(Caminho, "")
                'RegSub.SetValue("DBClassMapperPath", Caminho
            Catch Ex As Exception
                MsgBox(Ex.ToString)
                MyBase.Rollback(e.SavedState)
                MyBase.Uninstall(e.SavedState)
                Exit Sub
            End Try
            ExisteVS10 = True
        Catch

        End Try

        'Tenta gerar os registrys necessários para o VS 2012
        Try
            RegSub = RegBase.OpenSubKey("11.0", True)
            If RegSub Is Nothing Then
                Throw New Exception("Visual Studio 2012 not found.")
            End If
            Try
                RegSub = RegSub.CreateSubKey("AutomationOptions\LookInFolders")
                RegSub.SetValue(Caminho, "")
                'RegSub.SetValue("DBClassMapperPath", Caminho)
            Catch Ex As Exception
                MsgBox(Ex.ToString)
                MyBase.Rollback(e.SavedState)
                MyBase.Uninstall(e.SavedState)
                Exit Sub
            End Try
            ExisteVS12 = True
        Catch

        End Try

        'Copia o DBClassMapper.AddIn e o DBClassMapper.dll para
C:\ProgramData\Microsoft\MSEnvShared\Addins
```

```
        Try
            Dim ProgramData As String =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.CommonApplicationData)
            ProgramData = String.Format("{0}\Microsoft\MSEnvShared\Addins", ProgramData)
            Directory.CreateDirectory(ProgramData)
            File.Copy(String.Format("{0}\DBClassMapper.AddIn", Caminho),
String.Format("{0}\DBClassMapper.AddIn", ProgramData), True)
            File.Copy(String.Format("{0}\DBClassMapper.dll", Caminho),
String.Format("{0}\DBClassMapper.dll", ProgramData), True)
            File.Copy(String.Format("{0}\DBManager2.dll", Caminho),
String.Format("{0}\DBManager2.dll", ProgramData), True)
        Catch ex As Exception
            MsgBox(ex.ToString)
            MyBase.Rollback(e.SavedState)
            MyBase.Uninstall(e.SavedState)
            Exit Sub
        End Try

        'Caso não identificar o VS 2010/2012 ele avisa e volta a instalação
        If Not (ExisteVS10 Or ExisteVS12) Then
            MsgBox("A Versão do Visual Studio deve ser a 2010/2012. Esta versão não foi
encontrada.")
            MyBase.Rollback(e.SavedState)
            MyBase.Uninstall(e.SavedState)
            Exit Sub
        End If

    End Sub

    Private Sub InstallAddIn_BeforeUninstall(ByVal sender As Object, ByVal e As
System.Configuration.Install.InstallEventArgs) Handles Me.BeforeUninstall
        Dim Caminho As String
        Dim Reg As RegistryKey

        'Monta o caminho da instalação
        Try
            Caminho = Me.Context.Parameters.Item("assemblyPath")
            Caminho = Caminho.Substring(0, Caminho.LastIndexOf("\"))
        Catch Ex As Exception
            MsgBox(Ex.ToString)
            MyBase.Rollback(e.SavedState)
            Exit Sub
        End Try

        'Apaga o diretório do Mappeds.XML com todos os arquivos e subpastas junto
        Try
            Directory.Delete(String.Format("{0}\.deltacon\.dbclassmapper",
My.Computer.FileSystem.SpecialDirectories.AllUsersApplicationData), True)
        Catch
        End Try

        'Apaga o AddIn do ProgramData
        Try
            Dim ProgramData As String =
System.Environment.GetFolderPath(System.Environment.SpecialFolder.CommonApplicationData)
            ProgramData = String.Format("{0}\Microsoft\MSEnvShared\Addins", ProgramData)
            File.Delete(String.Format("{0}\DBClassMapper.AddIn", ProgramData))
            File.Delete(String.Format("{0}\DBClassMapper.dll", ProgramData))
```

```
            File.Delete(String.Format("{0}\DBManager2.dll", ProgramData))
        Catch
        End Try

        'Apaga os registry do VS 2010
        Try
            Reg =
Registry.LocalMachine.OpenSubKey("Software\Microsoft\VisualStudio\10.0\AutomationOptions\
LookInFolders", True)
            Reg.DeleteValue(Caminho)
            'Reg.DeleteValue("DBClassMapperPath")
        Catch
        End Try

        'Apaga os registry do VS 2012
        Try
            Reg =
Registry.LocalMachine.OpenSubKey("Software\Microsoft\VisualStudio\11.0\AutomationOptions\
LookInFolders", True)
            Reg.DeleteValue(Caminho)
            'Reg.DeleteValue("DBClassMapperPath")
        Catch
        End Try

    End Sub
End Class
```

AssemblySearcher.vb

```
Imports System.Collections.Generic
Imports System.Reflection
Imports System.IO
Imports System.Text
Imports DBClassMapper
Imports System.Xml.Serialization
Imports Microsoft.Win32
Imports System.ComponentModel.DataAnnotations

Public Class AssemblySearcher

    Public Shared Sub SearchMappeds(ByVal DLLPath As String, ByVal WritePath As String)
        Dim Asm As Assembly
        Dim FC As FillComboList
        Dim Fills As New List(Of FillComboList)
        Dim ObjToSerialize As New DLLInfo
        Dim XS As New XmlSerializer(GetType(DLLInfo))
        Dim FS As FileStream

        Try
            Asm = Assembly.LoadFrom(DLLPath)
            For Each T As Type In Asm.GetTypes
                Try
                    FC = GetDBTableAttributeFromDLL(T)
                    If Not IsNothing(FC) Then Fills.Add(FC)
                Catch ex As Exception
                    'MsgBox(String.Format("{0} não foi carregada.{1}{1}{2}", T.Name,
vbCrLf, ex.Message))
```

```
                    Console.WriteLine(String.Format("{0} não foi carregada.{1}{1}{2}",
T.Name, vbCrLf, ex.Message))
                End Try
            Next
        Catch ex As Exception
            Throw New Exception(String.Format("Couldn't load DLL {0}.{1}{1}{2}", DLLPath,
vbCrLf, ex.Message), ex)
        End Try

        ObjToSerialize.DLLPath = DLLPath
        ObjToSerialize.Mappeds = Fills

        Try
            FS = New FileStream(WritePath, FileMode.Create, FileAccess.ReadWrite)
            XS.Serialize(FS, ObjToSerialize)
            FS.Flush()
            FS.Close()
        Catch ex As Exception
            Throw New Exception(String.Format("Couldn't serialize DLL {0}.{1}{1}{2}",
DLLPath, vbCrLf, ex.Message), ex)
        End Try
    End Sub

    Private Shared Function GetDBTableAttributeFromDLL(ByVal T As Type) As FillComboList
        Dim Result As FillComboList = Nothing
        Dim Mapeado As New StringBuilder
        Dim TypeName As String
        Dim Schema As String
        Dim Table As String

        For Each A In T.GetCustomAttributes(True)
            If A.GetType.Name = "DBTable" Then
                TypeName = CStr(T.Name)
                Schema = CStr(A.Schema)
                Table = CStr(A.Table)

                If Schema <> "" Then
                    Mapeado.AppendFormat("{0}.", CStr(Schema))
                End If

                If Table = "" Then
                    Mapeado.Append(TypeName)
                Else
                    Mapeado.Append(Table)
                End If

                Result = New FillComboList(String.Format("ObjTypeGUID:{0}",
CStr(T.GUID.ToString)), String.Format("{0} ({1})", TypeName, Mapeado.ToString), TypeName,
, GetKeys(T), GetClassInfo(Table, Schema, T))
            End If
        Next

        Return Result
    End Function

    Private Shared Function GetKeys(ByRef T As Type) As String
        Dim Result As New StringBuilder
```

```
        Dim lQuery = From P In T.GetProperties()
                     From Chave In P.GetCustomAttributes(GetType(KeyAttribute), False)
                     From Field In P.GetCustomAttributes(False).Where(Function(X)
X.GetType.Name = "DBField")
                     Select P, Field

        For Each R In lQuery
            If R.Field.FieldName = "" Then
                Result.AppendFormat("{0}, ", R.P.Name)
            Else
                Result.AppendFormat("{0}, ", R.Field.FieldName)
            End If
        Next

        If Result.Length > 2 Then
            Result.Remove(Result.Length - 2, 2)
        End If

        Return Result.ToString
    End Function

    Private Shared Function GetClassInfo(ByVal Table As String, ByVal Schema As String,
ByVal T As Type) As ClassInfo
        Dim Result As New ClassInfo
        Dim FI As PropInfo = Nothing
        Dim LastProp As String = ""
        Dim OtherKeys() As String
        Dim ThisKeys() As String
        Dim AssocKey As AssociationKey
        Dim Mapped As Boolean = False

        'Dim lQuery = From P In T.GetProperties()
        '             From Att In P.GetCustomAttributes(False)
        '             From Field In P.GetCustomAttributes(False).Where(Function(X)
X.GetType.Name = "DBField")
        '             Select P, Field, Att

        Dim lQuery = From P In T.GetProperties()
                     From Att In P.GetCustomAttributes(False)
                     Select P, Att

        Result.Table = Table
        Result.Schema = Schema
        Result.ClassName = T.Name

        For Each R In lQuery
            If R.P.Name <> LastProp Then
                If Mapped Then Result.Props.Add(FI)

                FI = New PropInfo
                FI.PropName = R.P.Name
                If R.P.PropertyType.Name = "Nullable`1" Then
                    FI.TypeName = R.P.PropertyType.GetGenericArguments(0).Name
                Else
                    FI.TypeName = R.P.PropertyType.Name
                End If
                Mapped = False
            End If
```

```
            Select Case R.Att.GetType.Name
                Case "DBField"
                    Mapped = True
                    FI.FieldName = R.Att.FieldName

                Case "KeyAttribute"
                    FI.IsPrimaryKey = True

                Case "DBAlternateKey"
                    FI.IsAlternateKey = True

                Case "AssociationAttribute"
                    Mapped = True
                    Dim AssocAtt As AssociationAttribute = R.Att
                    OtherKeys = CStr(AssocAtt.OtherKey).Split(",")
                    ThisKeys = CStr(AssocAtt.ThisKey).Split(",")
                    If OtherKeys.Count <> ThisKeys.Count Then
                        Throw New Exception(String.Format("Error at {0} maps in {1}
AssociationAttribute. OtherKeys and ThisKeys needs to have the same number of Keys",
T.Name, R.P.Name))
                    End If
                    FI.Association = New AssociationInfo
                    For Each A In R.P.PropertyType.GetCustomAttributes(True)
                        If A.GetType.Name = "DBTable" Then
                            FI.Association.ClassName = R.P.PropertyType.Name
                            FI.Association.Schema = A.Schema
                            FI.Association.Table = A.Table
                            FI.Association.PropertyName = R.P.Name
                        End If
                    Next
                    If FI.Association.Table = "" Then FI.Association.Table =
R.P.PropertyType.Name
                    For I = 0 To OtherKeys.Count - 1
                        AssocKey = New AssociationKey
                        AssocKey.ThisKey = Trim(ThisKeys(I))
                        AssocKey.OtherKey = Trim(OtherKeys(I))
                        FI.Association.AssociatingKeys.Add(AssocKey)
                    Next

                Case "EditableAttribute"
                    'Nothing to do

                Case "RequiredAttribute"
                    'Nothing to do

                Case "DBDateTime"
                    'Nothing to do

                Case "DBStream"
                    'Nothing to do

                Case "StringLength"
                    'Case "StringRange"
                    'Nothing to do

                Case "TimeStampAttribute"
                    'Nothing to do
```

```
                Case "ConcurrencyCheckAttribute"
                    'Nothing to do

            End Select

            LastProp = R.P.Name
        Next

        If Mapped Then Result.Props.Add(FI)

        Return Result
    End Function

    'Private Shared Function GetMapperInfo(ByVal Name As String, ByRef T As Type,
Optional ByVal DefinedAssociatedTypes As List(Of String) = Nothing) As MapperInfo
    '    Dim Result As New MapperInfo
    '    Dim AI As AssociationInfo = Nothing
    '    Dim MI As MapperInfo = Nothing
    '    Dim LastProp As Object = "Empty"
    '    Dim FieldName As String = ""
    '    Dim OtherKeys() As String
    '    Dim ThisKeys() As String
    '    Dim AssocKey As AssociationKey
    '    Dim AddAsKey As Boolean = False
    '    Dim FieldFound As Boolean = False
    '    Dim Ignore As Boolean = False
    '    Dim I As Long

    '    If DefinedAssociatedTypes Is Nothing Then
    '        DefinedAssociatedTypes = New List(Of String)
    '    End If

    '    Result.Name = Name
    '    Result.TypeName = T.Name

    '    Dim lQuery = From P In T.GetProperties()
    '           From Att In P.GetCustomAttributes(False)
    '           From Field In P.GetCustomAttributes(False).Where(Function(X)
X.GetType.Name = "DBField")
    '         Select P, Field, Att
    '         Order By P.Name, Att.GetType.Name Descending

    '    For Each R In lQuery
    '        If Not R.P.Equals(LastProp) Then
    '            If FieldFound Then
    '                If AddAsKey Then
    '                    Result.KeysName.Add(FieldName)
    '                    AddAsKey = False
    '                End If

    '                If AI Is Nothing AndAlso MI IsNot Nothing Then
    '                    Result.MappedProperties.Add(MI)
    '                ElseIf AI IsNot Nothing Then
    '                    Result.Associateds.Add(AI)
    '                End If
    '                FieldFound = False
    '                Ignore = False
```

```
    '            End If

    '            MI = New MapperInfo
    '            MI.TypeName = R.P.PropertyType.Name
    '            AI = Nothing
    '        End If

    '        If Not Ignore Then
    '            If R.Att.GetType.Name = "DBField" Then
    '                If R.Att.FieldName = "" Then
    '                    FieldName = R.P.Name
    '                Else
    '                    FieldName = R.Att.FieldName
    '                End If
    '                MI.Name = FieldName
    '                FieldFound = True

    '                For Each S As String In DefinedAssociatedTypes
    '                    If FieldName = S Then
    '                        Ignore = True
    '                        Exit For
    '                    End If
    '                Next
    '            End If

    '            If R.Att.GetType.Equals(GetType(KeyAttribute)) Then
    '                AddAsKey = True
    '            ElseIf R.Att.GetType.Equals(GetType(AssociationAttribute)) Then
    '                AI = New AssociationInfo
    '                AI.Associated = MI
    '                MI.MappedProperties.Add(GetMapperInfo(FieldName, R.P.PropertyType))

    '                OtherKeys = CStr(R.Att.OtherKey).Split(",")
    '                ThisKeys = CStr(R.Att.ThisKey).Split(",")

    '                If OtherKeys.Count <> ThisKeys.Count Then
    '                    Throw New Exception(String.Format("Error at {0} maps in {1}
AssociationAttribute. OtherKeys and ThisKeys needs to have the same number of Keys",
T.Name, R.P.Name))
    '                End If

    '                For I = 0 To OtherKeys.Count - 1
    '                    AssocKey = New AssociationKey
    '                    AssocKey.ThisKey = Trim(ThisKeys(I))
    '                    AssocKey.OtherKey = Trim(OtherKeys(I))
    '                    AI.AssociantingKeys.Add(AssocKey)
    '                Next

    '                DefinedAssociatedTypes.Add(FieldName)
    '            End If
    '        End If

    '        LastProp = R.P
    '    Next

    '    If FieldFound Then
    '        If AddAsKey Then
    '            Result.KeysName.Add(FieldName)
```

```
    '            'AddAsKey = False
    '        End If

    '        If AI Is Nothing AndAlso MI IsNot Nothing Then
    '            Result.MappedProperties.Add(MI)
    '        ElseIf AI IsNot Nothing Then
    '            Result.Associateds.Add(AI)
    '        End If
    '        'FieldFound = False
    '    End If

    '    Return Result
    'End Function

End Class
```

ModMain.vb

```
Imports System.Collections.ObjectModel

Module ModMain

    Sub Main()
        Try
            Dim Args As ReadOnlyCollection(Of String) = My.Application.CommandLineArgs

            If Args.Count = 2 Then
                AssemblySearcher.SearchMappeds(Args(0), Args(1))
            ElseIf Args.Count = 3 Then

            Else
                MsgBox("AssemblyInspector precisa de 2 (dois) Argumentos em sua chamada
de execução. O primeiro é o caminho completo da DLL a ser inspecionada. Ex.:
C:\MinhasDLLs\UmaDLL.dll' e o segundo é o caminho completo onde será criado o arquivo
serializado.")
            End If
        Catch ex As Exception
            MsgBox(String.Format("Erro executar SearchMappeds.{0}{0}{1}", vbCrLf,
ex.Message))
        End Try
    End Sub

End Module
```

QueryTester.vb

```
Imports System.Reflection
Imports DBManager.DBWarder
Imports System.IO
Imports System.Xml.Serialization

Public Class QueryTester

    Public Sub Execute(ByVal DLLPath As String, ByVal QueryPath As String, ByVal ListPath
As String)
        'Load The Assembly File
```

```
        Dim Asm As Assembly = LoadAssembly(DLLPath)
        If Asm Is Nothing Then Exit Sub

        'Deserialize The GenDAL Query
        Dim DBLink As GenDAL = DeserializeQuery(QueryPath)
        If DBLink Is Nothing Then Exit Sub


    End Sub

    Private Function LoadAssembly(ByVal DLLPath As String) As Assembly
        Try
            Return Assembly.LoadFile(DLLPath)
        Catch ex As Exception
            MsgBox(ex.ToString)
            Return Nothing
        End Try
    End Function

    Private Function DeserializeQuery(ByVal QueryPath As String) As GenDAL
        Try
            Dim FS As FileStream
            Dim XS As New XmlSerializer(GetType(GenDAL))
            Dim Result As GenDAL

            FS = New FileStream(QueryPath, FileMode.Open, FileAccess.Read)
            Result = XS.Deserialize(FS)
            FS.Close()
            Return Result
        Catch ex As Exception
            MsgBox(ex.ToString)
            Return Nothing
        End Try
    End Function

End Class
```

Cliente.vb

```
Imports System.ComponentModel.DataAnnotations
Imports DBManager.DBMapper

Public Class Cliente

#Region "Cliente_Fields"
    Private _ID As Nullable(Of Integer)
    Private _CP as String
    Private _Nome as String

#End Region

#Region "Cliente_Properties"

    <DBField()> _
    <Key()> _
    <Editable(False)> _
    Public Property ID As Nullable(Of Integer)
```

```
        Get
            Return _ID
        End Get
        Set(ByVal value As Nullable(Of Integer))
            _ID = value
        End Set
    End Property

    <DBField()> _
    <DBAlternateKey()> _
    <Editable(False)> _
    Public Property CP as String
        Get
            Return _CP
        End Get
        Set(ByVal value As String)
            _CP = value
        End Set
    End Property

    <DBField()> _
    Public Property Nome as String
        Get
            Return _Nome
        End Get
        Set(ByVal value As String)
            _Nome = value
        End Set
    End Property

#End Region

End Class
```

Nota.vb

```
Imports System.ComponentModel.DataAnnotations
Imports DBManager.DBMapper

Public Class Nota

#Region "Nota_Fields"
    Private _ID As Nullable(Of Integer)
    Private _NR as String
    Private _Data as Nullable(Of DateTime)
    Private _Cliente as Cliente

#End Region

#Region "Nota_Properties"

    <DBField()> _
    <Key()> _
    <Editable(False)> _
    Public Property ID As Nullable(Of Integer)
        Get
            Return _ID
```

```
        End Get
        Set(ByVal value As Nullable(Of Integer))
            _ID = value
        End Set
    End Property

    <DBField()> _
    <DBAlternateKey()> _
    <Editable(False)> _
    Public Property NR as String
        Get
            Return _NR
        End Get
        Set(ByVal value As String)
            _NR = value
        End Set
    End Property

    <DBField()> _
    <DBDateTime()> _
    Public Property Data As Nullable(Of DateTime)
        Get
            Return _Data
        End Get
        Set(ByVal value As Nullable(Of DateTime))
            _Data = value
        End Set
    End Property

    <DBField()> _
    <Association("ClienteFK", "Cliente", "ID")> _
    Public Property Cliente As Cliente
        Get
            Return _Cliente
        End Get
        Set(ByVal value As Cliente)
            _Cliente = value
        End Set
    End Property

#End Region

End Class
```

NotaProduto.vb

```
Imports System.ComponentModel.DataAnnotations
Imports DBManager.DBMapper

Public Class NotaProduto

#Region "NotaProduto_Fields"
    Private _ID As Nullable(Of Integer)
    Private _Nota as Nota
    Private _Produto as Produto
    Private _Valor as Nullable(Of Double)
```

```
#End Region

#Region "NotaProduto_Properties"

    <DBField()> _
    <Key()> _
    <Editable(False)> _
    Public Property ID As Nullable(Of Integer)
        Get
            Return _ID
        End Get
        Set(ByVal value As Nullable(Of Integer))
            _ID = value
        End Set
    End Property

    <DBField()> _
    <Association("NotaFK", "Nota", "ID")> _
    Public Property Nota As Nota
        Get
            Return _Nota
        End Get
        Set(ByVal value As Nota)
            _Nota = value
        End Set
    End Property

    <DBField()> _
    <Association("ProdutoFK", "Produto", "ID")> _
    Public Property Produto As Produto
        Get
            Return _Produto
        End Get
        Set(ByVal value As Produto)
            _Produto = value
        End Set
    End Property

    <DBField()> _
    Public Property Valor as Nullable(Of Double)
        Get
            Return _Valor
        End Get
        Set(ByVal value As Nullable(Of Double))
            _Valor = value
        End Set
    End Property

#End Region

End Class
```

Produto.vb

```
Imports System.ComponentModel.DataAnnotations
Imports DBManager.DBMapper
```

```
Public Class Produto

#Region "Produto_Fields"
    Private _ID as Nullable(Of Integer)
    Private _CD as String
    Private _Nome as String

#End Region

#Region "Produto_Properties"

    <DBField()> _
    <Key()> _
    <Editable(False)> _
    Public Property ID as Nullable(Of Integer)
        Get
            Return _ID
        End Get
        Set(ByVal value As Nullable(Of Integer))
            _ID = value
        End Set
    End Property

    <DBField()> _
    <DBAlternateKey()> _
    <Editable(False)> _
    Public Property CD as String
        Get
            Return _CD
        End Get
        Set(ByVal value As String)
            _CD = value
        End Set
    End Property

    <DBField()> _
    Public Property Nome as String
        Get
            Return _Nome
        End Get
        Set(ByVal value As String)
            _Nome = value
        End Set
    End Property

#End Region

End Class
```

frmAddFilter.Designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmAddFilter
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
```

```
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
    Finally
        MyBase.Dispose(disposing)
    End Try
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
<System.Diagnostics.DebuggerStepThrough()> _
Private Sub InitializeComponent()
    Me.cmbOperator = New System.Windows.Forms.ComboBox()
    Me.Label2 = New System.Windows.Forms.Label()
    Me.Label1 = New System.Windows.Forms.Label()
    Me.txtField = New System.Windows.Forms.TextBox()
    Me.btnAddFilter = New System.Windows.Forms.Button()
    Me.Label3 = New System.Windows.Forms.Label()
    Me.txtVar1 = New System.Windows.Forms.TextBox()
    Me.Label4 = New System.Windows.Forms.Label()
    Me.txtTestVal1 = New System.Windows.Forms.TextBox()
    Me.Label5 = New System.Windows.Forms.Label()
    Me.txtTestVal2 = New System.Windows.Forms.TextBox()
    Me.Label6 = New System.Windows.Forms.Label()
    Me.txtVar2 = New System.Windows.Forms.TextBox()
    Me.SuspendLayout()
    '
    'cmbOperator
    '
    Me.cmbOperator.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
    Me.cmbOperator.FormattingEnabled = True
    Me.cmbOperator.Location = New System.Drawing.Point(206, 11)
    Me.cmbOperator.Name = "cmbOperator"
    Me.cmbOperator.Size = New System.Drawing.Size(121, 21)
    Me.cmbOperator.TabIndex = 9
    '
    'Label2
    '
    Me.Label2.AutoSize = True
    Me.Label2.Location = New System.Drawing.Point(152, 15)
    Me.Label2.Name = "Label2"
    Me.Label2.Size = New System.Drawing.Size(48, 13)
    Me.Label2.TabIndex = 8
    Me.Label2.Text = "Operator"
    '
    'Label1
    '
    Me.Label1.AutoSize = True
    Me.Label1.Location = New System.Drawing.Point(11, 15)
    Me.Label1.Name = "Label1"
    Me.Label1.Size = New System.Drawing.Size(29, 13)
    Me.Label1.TabIndex = 7
```

```
Me.Label1.Text = "Field"
'
'txtField
'
Me.txtField.Enabled = False
Me.txtField.Location = New System.Drawing.Point(46, 12)
Me.txtField.Name = "txtField"
Me.txtField.Size = New System.Drawing.Size(100, 20)
Me.txtField.TabIndex = 6
'
'btnAddFilter
'
Me.btnAddFilter.Location = New System.Drawing.Point(499, 9)
Me.btnAddFilter.Name = "btnAddFilter"
Me.btnAddFilter.Size = New System.Drawing.Size(75, 23)
Me.btnAddFilter.TabIndex = 5
Me.btnAddFilter.Text = "Add Filter"
Me.btnAddFilter.UseVisualStyleBackColor = True
'
'Label3
'
Me.Label3.AutoSize = True
Me.Label3.Location = New System.Drawing.Point(344, 15)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(32, 13)
Me.Label3.TabIndex = 11
Me.Label3.Text = "Var 1"
'
'txtVar1
'
Me.txtVar1.Location = New System.Drawing.Point(382, 12)
Me.txtVar1.Name = "txtVar1"
Me.txtVar1.Size = New System.Drawing.Size(100, 20)
Me.txtVar1.TabIndex = 10
'
'Label4
'
Me.Label4.AutoSize = True
Me.Label4.Location = New System.Drawing.Point(625, 12)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(55, 13)
Me.Label4.TabIndex = 13
Me.Label4.Text = "Test Val 1"
'
'txtTestVal1
'
Me.txtTestVal1.Location = New System.Drawing.Point(686, 8)
Me.txtTestVal1.Name = "txtTestVal1"
Me.txtTestVal1.Size = New System.Drawing.Size(100, 20)
Me.txtTestVal1.TabIndex = 12
'
'Label5
'
Me.Label5.AutoSize = True
Me.Label5.Location = New System.Drawing.Point(625, 38)
Me.Label5.Name = "Label5"
Me.Label5.Size = New System.Drawing.Size(55, 13)
Me.Label5.TabIndex = 17
```

```
        Me.Label5.Text = "Test Val 2"
        '
        'txtTestVal2
        '
        Me.txtTestVal2.Location = New System.Drawing.Point(686, 34)
        Me.txtTestVal2.Name = "txtTestVal2"
        Me.txtTestVal2.Size = New System.Drawing.Size(100, 20)
        Me.txtTestVal2.TabIndex = 16
        '
        'Label6
        '
        Me.Label6.AutoSize = True
        Me.Label6.Location = New System.Drawing.Point(344, 41)
        Me.Label6.Name = "Label6"
        Me.Label6.Size = New System.Drawing.Size(32, 13)
        Me.Label6.TabIndex = 15
        Me.Label6.Text = "Var 2"
        '
        'txtVar2
        '
        Me.txtVar2.Location = New System.Drawing.Point(382, 38)
        Me.txtVar2.Name = "txtVar2"
        Me.txtVar2.Size = New System.Drawing.Size(100, 20)
        Me.txtVar2.TabIndex = 14
        '
        'frmAddFilter
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(580, 68)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.txtTestVal2)
        Me.Controls.Add(Me.Label6)
        Me.Controls.Add(Me.txtVar2)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.txtTestVal1)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.txtVar1)
        Me.Controls.Add(Me.cmbOperator)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.txtField)
        Me.Controls.Add(Me.btnAddFilter)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "frmAddFilter"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
        Me.Text = "Add Filter"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents cmbOperator As System.Windows.Forms.ComboBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents txtField As System.Windows.Forms.TextBox
    Friend WithEvents btnAddFilter As System.Windows.Forms.Button
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents txtVar1 As System.Windows.Forms.TextBox
```

```
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents txtTestVal1 As System.Windows.Forms.TextBox
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents txtTestVal2 As System.Windows.Forms.TextBox
    Friend WithEvents Label6 As System.Windows.Forms.Label
    Friend WithEvents txtVar2 As System.Windows.Forms.TextBox
End Class
```

frmAddOrder.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmAddOrder
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.btnAddOrder = New System.Windows.Forms.Button()
        Me.txtField = New System.Windows.Forms.TextBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.cmbOrder = New System.Windows.Forms.ComboBox()
        Me.SuspendLayout()
        '
        'btnAddOrder
        '
        Me.btnAddOrder.Location = New System.Drawing.Point(354, 3)
        Me.btnAddOrder.Name = "btnAddOrder"
        Me.btnAddOrder.Size = New System.Drawing.Size(75, 23)
        Me.btnAddOrder.TabIndex = 0
        Me.btnAddOrder.Text = "Add Order"
        Me.btnAddOrder.UseVisualStyleBackColor = True
        '
        'txtField
        '
        Me.txtField.Enabled = False
        Me.txtField.Location = New System.Drawing.Point(47, 6)
        Me.txtField.Name = "txtField"
        Me.txtField.Size = New System.Drawing.Size(100, 20)
        Me.txtField.TabIndex = 1
```

```
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(29, 13)
        Me.Label1.TabIndex = 2
        Me.Label1.Text = "Field"
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(168, 9)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(33, 13)
        Me.Label2.TabIndex = 3
        Me.Label2.Text = "Order"
        '
        'cmbOrder
        '
        Me.cmbOrder.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.cmbOrder.FormattingEnabled = True
        Me.cmbOrder.Location = New System.Drawing.Point(207, 5)
        Me.cmbOrder.Name = "cmbOrder"
        Me.cmbOrder.Size = New System.Drawing.Size(121, 21)
        Me.cmbOrder.TabIndex = 4
        '
        'frmAddOrder
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(441, 33)
        Me.Controls.Add(Me.cmbOrder)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.txtField)
        Me.Controls.Add(Me.btnAddOrder)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "frmAddOrder"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
        Me.Text = "Add Order By"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents btnAddOrder As System.Windows.Forms.Button
    Friend WithEvents txtField As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents cmbOrder As System.Windows.Forms.ComboBox
End Class
```

frmDBReader.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmDBReader
```

```
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Dim resources As System.ComponentModel.ComponentResourceManager = New
System.ComponentModel.ComponentResourceManager(GetType(frmDBReader))
        Me.sptMain = New System.Windows.Forms.SplitContainer()
        Me.sptTop = New System.Windows.Forms.SplitContainer()
        Me.btnReadDB = New System.Windows.Forms.Button()
        Me.btnNewDSN = New System.Windows.Forms.Button()
        Me.cmbDSN = New System.Windows.Forms.ComboBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.grdTables = New System.Windows.Forms.DataGridView()
        Me.sptBottom = New System.Windows.Forms.SplitContainer()
        Me.grdFields = New System.Windows.Forms.DataGridView()
        Me.chkOnPropertyChanged = New System.Windows.Forms.CheckBox()
        Me.btnFindClassTableAndCompare = New System.Windows.Forms.Button()
        Me.btnCreateClass = New System.Windows.Forms.Button()
        Me.btnCreateProperty = New System.Windows.Forms.Button()
        Me.lblCreating = New System.Windows.Forms.Label()
        Me.pgbCreating = New System.Windows.Forms.ProgressBar()
        CType(Me.sptMain, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.sptMain.Panel1.SuspendLayout()
        Me.sptMain.Panel2.SuspendLayout()
        Me.sptMain.SuspendLayout()
        CType(Me.sptTop, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.sptTop.Panel1.SuspendLayout()
        Me.sptTop.Panel2.SuspendLayout()
        Me.sptTop.SuspendLayout()
        CType(Me.grdTables, System.ComponentModel.ISupportInitialize).BeginInit()
        CType(Me.sptBottom, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.sptBottom.Panel1.SuspendLayout()
        Me.sptBottom.Panel2.SuspendLayout()
        Me.sptBottom.SuspendLayout()
        CType(Me.grdFields, System.ComponentModel.ISupportInitialize).BeginInit()
        Me.SuspendLayout()
        '
        'sptMain
        '
        Me.sptMain.Dock = System.Windows.Forms.DockStyle.Fill
```

```
        Me.sptMain.Location = New System.Drawing.Point(3, 3)
        Me.sptMain.Name = "sptMain"
        Me.sptMain.Orientation = System.Windows.Forms.Orientation.Horizontal
        '
        'sptMain.Panel1
        '
        Me.sptMain.Panel1.Controls.Add(Me.sptTop)
        '
        'sptMain.Panel2
        '
        Me.sptMain.Panel2.Controls.Add(Me.sptBottom)
        Me.sptMain.Size = New System.Drawing.Size(617, 431)
        Me.sptMain.SplitterDistance = 226
        Me.sptMain.TabIndex = 6
        '
        'sptTop
        '
        Me.sptTop.Dock = System.Windows.Forms.DockStyle.Fill
        Me.sptTop.FixedPanel = System.Windows.Forms.FixedPanel.Panel1
        Me.sptTop.IsSplitterFixed = True
        Me.sptTop.Location = New System.Drawing.Point(0, 0)
        Me.sptTop.Name = "sptTop"
        Me.sptTop.Orientation = System.Windows.Forms.Orientation.Horizontal
        '
        'sptTop.Panel1
        '
        Me.sptTop.Panel1.Controls.Add(Me.btnReadDB)
        Me.sptTop.Panel1.Controls.Add(Me.btnNewDSN)
        Me.sptTop.Panel1.Controls.Add(Me.cmbDSN)
        Me.sptTop.Panel1.Controls.Add(Me.Label1)
        '
        'sptTop.Panel2
        '
        Me.sptTop.Panel2.Controls.Add(Me.grdTables)
        Me.sptTop.Size = New System.Drawing.Size(617, 226)
        Me.sptTop.SplitterDistance = 32
        Me.sptTop.TabIndex = 8
        '
        'btnReadDB
        '
        Me.btnReadDB.Anchor = CType((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.btnReadDB.Location = New System.Drawing.Point(539, 4)
        Me.btnReadDB.Name = "btnReadDB"
        Me.btnReadDB.Size = New System.Drawing.Size(75, 23)
        Me.btnReadDB.TabIndex = 3
        Me.btnReadDB.Text = "Read DB"
        Me.btnReadDB.UseVisualStyleBackColor = True
        '
        'btnNewDSN
        '
        Me.btnNewDSN.Location = New System.Drawing.Point(48, 4)
        Me.btnNewDSN.Name = "btnNewDSN"
        Me.btnNewDSN.Size = New System.Drawing.Size(75, 23)
        Me.btnNewDSN.TabIndex = 2
        Me.btnNewDSN.Text = "New DSN"
        Me.btnNewDSN.UseVisualStyleBackColor = True
        '
```

```
        'cmbDSN
        '
        Me.cmbDSN.Anchor = CType((System.Windows.Forms.AnchorStyles.Left Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.cmbDSN.FormattingEnabled = True
        Me.cmbDSN.Location = New System.Drawing.Point(129, 6)
        Me.cmbDSN.Name = "cmbDSN"
        Me.cmbDSN.Size = New System.Drawing.Size(395, 21)
        Me.cmbDSN.TabIndex = 1
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(30, 13)
        Me.Label1.TabIndex = 4
        Me.Label1.Text = "DSN"
        '
        'grdTables
        '
        Me.grdTables.AllowUserToAddRows = False
        Me.grdTables.AllowUserToDeleteRows = False
        Me.grdTables.AllowUserToOrderColumns = True
        Me.grdTables.AllowUserToResizeRows = False
        Me.grdTables.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.grdTables.Dock = System.Windows.Forms.DockStyle.Fill
        Me.grdTables.Location = New System.Drawing.Point(0, 0)
        Me.grdTables.MultiSelect = False
        Me.grdTables.Name = "grdTables"
        Me.grdTables.ReadOnly = True
        Me.grdTables.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect
        Me.grdTables.Size = New System.Drawing.Size(617, 190)
        Me.grdTables.TabIndex = 4
        '
        'sptBottom
        '
        Me.sptBottom.Dock = System.Windows.Forms.DockStyle.Fill
        Me.sptBottom.FixedPanel = System.Windows.Forms.FixedPanel.Panel2
        Me.sptBottom.IsSplitterFixed = True
        Me.sptBottom.Location = New System.Drawing.Point(0, 0)
        Me.sptBottom.Name = "sptBottom"
        Me.sptBottom.Orientation = System.Windows.Forms.Orientation.Horizontal
        '
        'sptBottom.Panel1
        '
        Me.sptBottom.Panel1.Controls.Add(Me.grdFields)
        '
        'sptBottom.Panel2
        '
        Me.sptBottom.Panel2.Controls.Add(Me.chkOnPropertyChanged)
        Me.sptBottom.Panel2.Controls.Add(Me.btnFindClassTableAndCompare)
        Me.sptBottom.Panel2.Controls.Add(Me.btnCreateClass)
        Me.sptBottom.Panel2.Controls.Add(Me.btnCreateProperty)
        Me.sptBottom.Size = New System.Drawing.Size(617, 201)
        Me.sptBottom.SplitterDistance = 165
```

```
        Me.sptBottom.TabIndex = 0
        '
        'grdFields
        '
        Me.grdFields.AllowUserToAddRows = False
        Me.grdFields.AllowUserToDeleteRows = False
        Me.grdFields.AllowUserToOrderColumns = True
        Me.grdFields.AllowUserToResizeRows = False
        Me.grdFields.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.grdFields.Dock = System.Windows.Forms.DockStyle.Fill
        Me.grdFields.Location = New System.Drawing.Point(0, 0)
        Me.grdFields.Name = "grdFields"
        Me.grdFields.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect
        Me.grdFields.Size = New System.Drawing.Size(617, 165)
        Me.grdFields.TabIndex = 5
        '
        'chkOnPropertyChanged
        '
        Me.chkOnPropertyChanged.Anchor = CType((System.Windows.Forms.AnchorStyles.Bottom
Or System.Windows.Forms.AnchorStyles.Left), System.Windows.Forms.AnchorStyles)
        Me.chkOnPropertyChanged.AutoSize = True
        Me.chkOnPropertyChanged.Location = New System.Drawing.Point(3, 9)
        Me.chkOnPropertyChanged.Name = "chkOnPropertyChanged"
        Me.chkOnPropertyChanged.Size = New System.Drawing.Size(144, 17)
        Me.chkOnPropertyChanged.TabIndex = 21
        Me.chkOnPropertyChanged.Text = "Add OnPropertyChanged"
        Me.chkOnPropertyChanged.UseVisualStyleBackColor = True
        '
        'btnFindClassTableAndCompare
        '
        Me.btnFindClassTableAndCompare.Anchor =
CType((System.Windows.Forms.AnchorStyles.Bottom Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.btnFindClassTableAndCompare.Location = New System.Drawing.Point(248, 5)
        Me.btnFindClassTableAndCompare.Name = "btnFindClassTableAndCompare"
        Me.btnFindClassTableAndCompare.Size = New System.Drawing.Size(165, 23)
        Me.btnFindClassTableAndCompare.TabIndex = 20
        Me.btnFindClassTableAndCompare.Text = "Find Class Table and Compare"
        Me.btnFindClassTableAndCompare.UseVisualStyleBackColor = True
        '
        'btnCreateClass
        '
        Me.btnCreateClass.Anchor = CType((System.Windows.Forms.AnchorStyles.Bottom Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.btnCreateClass.Location = New System.Drawing.Point(419, 5)
        Me.btnCreateClass.Name = "btnCreateClass"
        Me.btnCreateClass.Size = New System.Drawing.Size(96, 23)
        Me.btnCreateClass.TabIndex = 6
        Me.btnCreateClass.Text = "Create Class"
        Me.btnCreateClass.UseVisualStyleBackColor = True
        '
        'btnCreateProperty
        '
        Me.btnCreateProperty.Anchor = CType((System.Windows.Forms.AnchorStyles.Bottom Or
System.Windows.Forms.AnchorStyles.Right), System.Windows.Forms.AnchorStyles)
        Me.btnCreateProperty.Location = New System.Drawing.Point(521, 5)
```

```
        Me.btnCreateProperty.Name = "btnCreateProperty"
        Me.btnCreateProperty.Size = New System.Drawing.Size(96, 23)
        Me.btnCreateProperty.TabIndex = 7
        Me.btnCreateProperty.Text = "Create Property"
        Me.btnCreateProperty.UseVisualStyleBackColor = True
        '
        'lblCreating
        '
        Me.lblCreating.Font = New System.Drawing.Font("Microsoft Sans Serif", 12.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lblCreating.Location = New System.Drawing.Point(253, 186)
        Me.lblCreating.Name = "lblCreating"
        Me.lblCreating.Size = New System.Drawing.Size(116, 65)
        Me.lblCreating.TabIndex = 25
        Me.lblCreating.Text = "In Progress..."
        Me.lblCreating.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
        Me.lblCreating.Visible = False
        '
        'pgbCreating
        '
        Me.pgbCreating.Location = New System.Drawing.Point(250, 207)
        Me.pgbCreating.Name = "pgbCreating"
        Me.pgbCreating.Size = New System.Drawing.Size(122, 23)
        Me.pgbCreating.TabIndex = 26
        Me.pgbCreating.Visible = False
        '
        'frmDBReader
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(623, 437)
        Me.Controls.Add(Me.pgbCreating)
        Me.Controls.Add(Me.lblCreating)
        Me.Controls.Add(Me.sptMain)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "frmDBReader"
        Me.Padding = New System.Windows.Forms.Padding(3)
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "DataBase Mapper Generator"
        Me.TopMost = True
        Me.sptMain.Panel1.ResumeLayout(False)
        Me.sptMain.Panel2.ResumeLayout(False)
        CType(Me.sptMain, System.ComponentModel.ISupportInitialize).EndInit()
        Me.sptMain.ResumeLayout(False)
        Me.sptTop.Panel1.ResumeLayout(False)
        Me.sptTop.Panel1.PerformLayout()
        Me.sptTop.Panel2.ResumeLayout(False)
        CType(Me.sptTop, System.ComponentModel.ISupportInitialize).EndInit()
        Me.sptTop.ResumeLayout(False)
        CType(Me.grdTables, System.ComponentModel.ISupportInitialize).EndInit()
        Me.sptBottom.Panel1.ResumeLayout(False)
        Me.sptBottom.Panel2.ResumeLayout(False)
        Me.sptBottom.Panel2.PerformLayout()
        CType(Me.sptBottom, System.ComponentModel.ISupportInitialize).EndInit()
        Me.sptBottom.ResumeLayout(False)
        CType(Me.grdFields, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
```

```
    End Sub
    Friend WithEvents sptMain As System.Windows.Forms.SplitContainer
    Friend WithEvents sptTop As System.Windows.Forms.SplitContainer
    Friend WithEvents btnReadDB As System.Windows.Forms.Button
    Friend WithEvents btnNewDSN As System.Windows.Forms.Button
    Friend WithEvents cmbDSN As System.Windows.Forms.ComboBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents grdTables As System.Windows.Forms.DataGridView
    Friend WithEvents sptBottom As System.Windows.Forms.SplitContainer
    Friend WithEvents grdFields As System.Windows.Forms.DataGridView
    Friend WithEvents btnCreateClass As System.Windows.Forms.Button
    Friend WithEvents btnCreateProperty As System.Windows.Forms.Button
    Friend WithEvents lblCreating As System.Windows.Forms.Label
    Friend WithEvents pgbCreating As System.Windows.Forms.ProgressBar
    Friend WithEvents btnFindClassTableAndCompare As System.Windows.Forms.Button
    Friend WithEvents chkOnPropertyChanged As System.Windows.Forms.CheckBox
End Class
```

frmMainWindow.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmMainWindow
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.btnNewClass = New System.Windows.Forms.Button()
        Me.btnNewProperty = New System.Windows.Forms.Button()
        Me.btnReadDataBase = New System.Windows.Forms.Button()
        Me.btnClose = New System.Windows.Forms.Button()
        Me.btnQueryBuilder = New System.Windows.Forms.Button()
        Me.SuspendLayout()
        '
        'btnNewClass
        '
        Me.btnNewClass.Location = New System.Drawing.Point(12, 12)
        Me.btnNewClass.Name = "btnNewClass"
        Me.btnNewClass.Size = New System.Drawing.Size(125, 23)
        Me.btnNewClass.TabIndex = 0
```

```
        Me.btnNewClass.Text = "Create New Class"
        Me.btnNewClass.UseVisualStyleBackColor = True
        '
        'btnNewProperty
        '
        Me.btnNewProperty.Location = New System.Drawing.Point(147, 12)
        Me.btnNewProperty.Name = "btnNewProperty"
        Me.btnNewProperty.Size = New System.Drawing.Size(125, 23)
        Me.btnNewProperty.TabIndex = 1
        Me.btnNewProperty.Text = "Create New Property"
        Me.btnNewProperty.UseVisualStyleBackColor = True
        '
        'btnReadDataBase
        '
        Me.btnReadDataBase.Location = New System.Drawing.Point(12, 41)
        Me.btnReadDataBase.Name = "btnReadDataBase"
        Me.btnReadDataBase.Size = New System.Drawing.Size(260, 23)
        Me.btnReadDataBase.TabIndex = 2
        Me.btnReadDataBase.Text = "Read a Database to Create Full Class"
        Me.btnReadDataBase.UseVisualStyleBackColor = True
        '
        'btnClose
        '
        Me.btnClose.Location = New System.Drawing.Point(147, 111)
        Me.btnClose.Name = "btnClose"
        Me.btnClose.Size = New System.Drawing.Size(125, 23)
        Me.btnClose.TabIndex = 3
        Me.btnClose.Text = "Close"
        Me.btnClose.UseVisualStyleBackColor = True
        '
        'btnQueryBuilder
        '
        Me.btnQueryBuilder.Location = New System.Drawing.Point(12, 70)
        Me.btnQueryBuilder.Name = "btnQueryBuilder"
        Me.btnQueryBuilder.Size = New System.Drawing.Size(260, 23)
        Me.btnQueryBuilder.TabIndex = 4
        Me.btnQueryBuilder.Text = "Create Queries Based on Models"
        Me.btnQueryBuilder.UseVisualStyleBackColor = True
        '
        'frmMainWindow
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(284, 146)
        Me.Controls.Add(Me.btnQueryBuilder)
        Me.Controls.Add(Me.btnClose)
        Me.Controls.Add(Me.btnReadDataBase)
        Me.Controls.Add(Me.btnNewProperty)
        Me.Controls.Add(Me.btnNewClass)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "frmMainWindow"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "DBClassMapper V 3.2"
        Me.TopMost = True
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents btnNewClass As System.Windows.Forms.Button
```

```
    Friend WithEvents btnNewProperty As System.Windows.Forms.Button
    Friend WithEvents btnReadDataBase As System.Windows.Forms.Button
    Friend WithEvents btnClose As System.Windows.Forms.Button
    Friend WithEvents btnQueryBuilder As System.Windows.Forms.Button
End Class
```

frmNewClass.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmNewClass
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.txtClassName = New System.Windows.Forms.TextBox()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.txtSchema = New System.Windows.Forms.TextBox()
        Me.txtTable = New System.Windows.Forms.TextBox()
        Me.Label3 = New System.Windows.Forms.Label()
        Me.btnClose = New System.Windows.Forms.Button()
        Me.btnCreate = New System.Windows.Forms.Button()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(12, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(63, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Class Name"
        '
        'txtClassName
        '
        Me.txtClassName.Location = New System.Drawing.Point(81, 6)
        Me.txtClassName.Name = "txtClassName"
        Me.txtClassName.Size = New System.Drawing.Size(169, 20)
        Me.txtClassName.TabIndex = 0
```

```
'
'Label2
'
Me.Label2.AutoSize = True
Me.Label2.Location = New System.Drawing.Point(29, 35)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(46, 13)
Me.Label2.TabIndex = 2
Me.Label2.Text = "Schema"
'
'txtSchema
'
Me.txtSchema.Location = New System.Drawing.Point(81, 32)
Me.txtSchema.Name = "txtSchema"
Me.txtSchema.Size = New System.Drawing.Size(169, 20)
Me.txtSchema.TabIndex = 1
'
'txtTable
'
Me.txtTable.Location = New System.Drawing.Point(81, 58)
Me.txtTable.Name = "txtTable"
Me.txtTable.Size = New System.Drawing.Size(169, 20)
Me.txtTable.TabIndex = 2
'
'Label3
'
Me.Label3.AutoSize = True
Me.Label3.Location = New System.Drawing.Point(41, 61)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(34, 13)
Me.Label3.TabIndex = 4
Me.Label3.Text = "Table"
'
'btnClose
'
Me.btnClose.DialogResult = System.Windows.Forms.DialogResult.Cancel
Me.btnClose.Location = New System.Drawing.Point(175, 84)
Me.btnClose.Name = "btnClose"
Me.btnClose.Size = New System.Drawing.Size(75, 23)
Me.btnClose.TabIndex = 4
Me.btnClose.Text = "Close"
Me.btnClose.UseVisualStyleBackColor = True
'
'btnCreate
'
Me.btnCreate.DialogResult = System.Windows.Forms.DialogResult.OK
Me.btnCreate.Location = New System.Drawing.Point(81, 84)
Me.btnCreate.Name = "btnCreate"
Me.btnCreate.Size = New System.Drawing.Size(75, 23)
Me.btnCreate.TabIndex = 3
Me.btnCreate.Text = "Create"
Me.btnCreate.UseVisualStyleBackColor = True
'
'frmNewClass
'
Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
Me.ClientSize = New System.Drawing.Size(262, 116)
```

```
        Me.Controls.Add(Me.btnCreate)
        Me.Controls.Add(Me.btnClose)
        Me.Controls.Add(Me.txtTable)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.txtSchema)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.txtClassName)
        Me.Controls.Add(Me.Label1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "frmNewClass"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "New Class"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents txtClassName As System.Windows.Forms.TextBox
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents txtSchema As System.Windows.Forms.TextBox
    Friend WithEvents txtTable As System.Windows.Forms.TextBox
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents btnClose As System.Windows.Forms.Button
    Friend WithEvents btnCreate As System.Windows.Forms.Button
End Class
```

frmNewProperty.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmNewProperty
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Me.cmbMapperType = New System.Windows.Forms.ComboBox()
        Me.Label1 = New System.Windows.Forms.Label()
        Me.Label2 = New System.Windows.Forms.Label()
        Me.txtName = New System.Windows.Forms.TextBox()
        Me.Label3 = New System.Windows.Forms.Label()
```

```
Me.cmbType = New System.Windows.Forms.ComboBox()
Me.btnClose = New System.Windows.Forms.Button()
Me.btnAdd = New System.Windows.Forms.Button()
Me.Label7 = New System.Windows.Forms.Label()
Me.cmbReference = New System.Windows.Forms.ComboBox()
Me.chkOnPropertyChanged = New System.Windows.Forms.CheckBox()
Me.clbMappers = New System.Windows.Forms.CheckedListBox()
Me.Label4 = New System.Windows.Forms.Label()
Me.chkNullableOf = New System.Windows.Forms.CheckBox()
Me.btnBuild = New System.Windows.Forms.Button()
Me.lblLoading = New System.Windows.Forms.Label()
Me.pgbLoading = New System.Windows.Forms.ProgressBar()
Me.SuspendLayout()
'
'cmbMapperType
'
Me.cmbMapperType.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
Me.cmbMapperType.FormattingEnabled = True
Me.cmbMapperType.Location = New System.Drawing.Point(88, 12)
Me.cmbMapperType.Name = "cmbMapperType"
Me.cmbMapperType.Size = New System.Drawing.Size(268, 21)
Me.cmbMapperType.TabIndex = 0
'
'Label1
'
Me.Label1.AutoSize = True
Me.Label1.Location = New System.Drawing.Point(12, 15)
Me.Label1.Name = "Label1"
Me.Label1.Size = New System.Drawing.Size(70, 13)
Me.Label1.TabIndex = 0
Me.Label1.Text = "Mapper Type"
'
'Label2
'
Me.Label2.AutoSize = True
Me.Label2.Location = New System.Drawing.Point(47, 112)
Me.Label2.Name = "Label2"
Me.Label2.Size = New System.Drawing.Size(35, 13)
Me.Label2.TabIndex = 2
Me.Label2.Text = "Name"
'
'txtName
'
Me.txtName.Location = New System.Drawing.Point(88, 109)
Me.txtName.Name = "txtName"
Me.txtName.Size = New System.Drawing.Size(268, 20)
Me.txtName.TabIndex = 2
'
'Label3
'
Me.Label3.AutoSize = True
Me.Label3.Location = New System.Drawing.Point(51, 165)
Me.Label3.Name = "Label3"
Me.Label3.Size = New System.Drawing.Size(31, 13)
Me.Label3.TabIndex = 6
Me.Label3.Text = "Type"
'
'cmbType
```

```
'
Me.cmbType.FormattingEnabled = True
Me.cmbType.Location = New System.Drawing.Point(88, 162)
Me.cmbType.Name = "cmbType"
Me.cmbType.Size = New System.Drawing.Size(268, 21)
Me.cmbType.TabIndex = 5
'
'btnClose
'
Me.btnClose.DialogResult = System.Windows.Forms.DialogResult.Cancel
Me.btnClose.Location = New System.Drawing.Point(235, 238)
Me.btnClose.Name = "btnClose"
Me.btnClose.Size = New System.Drawing.Size(121, 23)
Me.btnClose.TabIndex = 9
Me.btnClose.Text = "Close"
Me.btnClose.UseVisualStyleBackColor = True
'
'btnAdd
'
Me.btnAdd.DialogResult = System.Windows.Forms.DialogResult.OK
Me.btnAdd.Location = New System.Drawing.Point(88, 238)
Me.btnAdd.Name = "btnAdd"
Me.btnAdd.Size = New System.Drawing.Size(121, 23)
Me.btnAdd.TabIndex = 8
Me.btnAdd.Text = "Add"
Me.btnAdd.UseVisualStyleBackColor = True
'
'Label7
'
Me.Label7.AutoSize = True
Me.Label7.Location = New System.Drawing.Point(25, 138)
Me.Label7.Name = "Label7"
Me.Label7.Size = New System.Drawing.Size(57, 13)
Me.Label7.TabIndex = 16
Me.Label7.Text = "Reference"
'
'cmbReference
'
Me.cmbReference.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
Me.cmbReference.FormattingEnabled = True
Me.cmbReference.Location = New System.Drawing.Point(88, 135)
Me.cmbReference.Name = "cmbReference"
Me.cmbReference.Size = New System.Drawing.Size(215, 21)
Me.cmbReference.TabIndex = 3
'
'chkOnPropertyChanged
'
Me.chkOnPropertyChanged.AutoSize = True
Me.chkOnPropertyChanged.Location = New System.Drawing.Point(88, 214)
Me.chkOnPropertyChanged.Name = "chkOnPropertyChanged"
Me.chkOnPropertyChanged.Size = New System.Drawing.Size(144, 17)
Me.chkOnPropertyChanged.TabIndex = 7
Me.chkOnPropertyChanged.Text = "Add OnPropertyChanged"
Me.chkOnPropertyChanged.UseVisualStyleBackColor = True
'
'clbMappers
'
Me.clbMappers.FormattingEnabled = True
```

```
Me.clbMappers.Location = New System.Drawing.Point(88, 39)
Me.clbMappers.Name = "clbMappers"
Me.clbMappers.Size = New System.Drawing.Size(268, 64)
Me.clbMappers.TabIndex = 1
'
'Label4
'
Me.Label4.AutoSize = True
Me.Label4.Location = New System.Drawing.Point(34, 39)
Me.Label4.Name = "Label4"
Me.Label4.Size = New System.Drawing.Size(48, 13)
Me.Label4.TabIndex = 20
Me.Label4.Text = "Mappers"
'
'chkNullableOf
'
Me.chkNullableOf.AutoSize = True
Me.chkNullableOf.Location = New System.Drawing.Point(88, 191)
Me.chkNullableOf.Name = "chkNullableOf"
Me.chkNullableOf.Size = New System.Drawing.Size(149, 17)
Me.chkNullableOf.TabIndex = 6
Me.chkNullableOf.Text = "Type is Nullable(of [Type])"
Me.chkNullableOf.UseVisualStyleBackColor = True
'
'btnBuild
'
Me.btnBuild.Location = New System.Drawing.Point(309, 133)
Me.btnBuild.Name = "btnBuild"
Me.btnBuild.Size = New System.Drawing.Size(47, 23)
Me.btnBuild.TabIndex = 4
Me.btnBuild.Text = "Build"
Me.btnBuild.UseVisualStyleBackColor = True
'
'lblLoading
'
Me.lblLoading.Location = New System.Drawing.Point(243, 186)
Me.lblLoading.Name = "lblLoading"
Me.lblLoading.Size = New System.Drawing.Size(113, 22)
Me.lblLoading.TabIndex = 23
Me.lblLoading.Text = "Loading..."
Me.lblLoading.TextAlign = System.Drawing.ContentAlignment.MiddleCenter
Me.lblLoading.Visible = False
'
'pgbLoading
'
Me.pgbLoading.Location = New System.Drawing.Point(246, 209)
Me.pgbLoading.MarqueeAnimationSpeed = 10
Me.pgbLoading.Name = "pgbLoading"
Me.pgbLoading.Size = New System.Drawing.Size(110, 22)
Me.pgbLoading.Style = System.Windows.Forms.ProgressBarStyle.Marquee
Me.pgbLoading.TabIndex = 24
Me.pgbLoading.Visible = False
'
'frmNewProperty
'
Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
Me.CancelButton = Me.btnClose
```

```
        Me.ClientSize = New System.Drawing.Size(368, 273)
        Me.Controls.Add(Me.pgbLoading)
        Me.Controls.Add(Me.lblLoading)
        Me.Controls.Add(Me.btnBuild)
        Me.Controls.Add(Me.chkNullableOf)
        Me.Controls.Add(Me.Label4)
        Me.Controls.Add(Me.clbMappers)
        Me.Controls.Add(Me.chkOnPropertyChanged)
        Me.Controls.Add(Me.Label7)
        Me.Controls.Add(Me.cmbReference)
        Me.Controls.Add(Me.btnAdd)
        Me.Controls.Add(Me.btnClose)
        Me.Controls.Add(Me.Label3)
        Me.Controls.Add(Me.cmbType)
        Me.Controls.Add(Me.txtName)
        Me.Controls.Add(Me.Label2)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.cmbMapperType)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.HelpButton = True
        Me.KeyPreview = True
        Me.MaximizeBox = False
        Me.MinimizeBox = False
        Me.Name = "frmNewProperty"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen
        Me.Text = "New Property"
        Me.TopMost = True
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents cmbMapperType As System.Windows.Forms.ComboBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents txtName As System.Windows.Forms.TextBox
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents cmbType As System.Windows.Forms.ComboBox
    Friend WithEvents btnClose As System.Windows.Forms.Button
    Friend WithEvents btnAdd As System.Windows.Forms.Button
    Friend WithEvents Label7 As System.Windows.Forms.Label
    Friend WithEvents cmbReference As System.Windows.Forms.ComboBox
    Friend WithEvents chkOnPropertyChanged As System.Windows.Forms.CheckBox
    Friend WithEvents clbMappers As System.Windows.Forms.CheckedListBox
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents chkNullableOf As System.Windows.Forms.CheckBox
    Friend WithEvents btnBuild As System.Windows.Forms.Button
    Friend WithEvents lblLoading As System.Windows.Forms.Label
    Friend WithEvents pgbLoading As System.Windows.Forms.ProgressBar
End Class
```

frmNodeProperties.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmNodeProperties
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
```

```
<System.Diagnostics.DebuggerNonUserCode()> _
Protected Overrides Sub Dispose(ByVal disposing As Boolean)
    Try
        If disposing AndAlso components IsNot Nothing Then
            components.Dispose()
        End If
    Finally
        MyBase.Dispose(disposing)
    End Try
End Sub

'Required by the Windows Form Designer
Private components As System.ComponentModel.IContainer

'NOTE: The following procedure is required by the Windows Form Designer
'It can be modified using the Windows Form Designer.
'Do not modify it using the code editor.
<System.Diagnostics.DebuggerStepThrough()> _
Private Sub InitializeComponent()
    Me.GroupBox1 = New System.Windows.Forms.GroupBox()
    Me.txtAlias = New System.Windows.Forms.TextBox()
    Me.Label6 = New System.Windows.Forms.Label()
    Me.cmbJoin = New System.Windows.Forms.ComboBox()
    Me.cmbRelation = New System.Windows.Forms.ComboBox()
    Me.Label3 = New System.Windows.Forms.Label()
    Me.txtTable = New System.Windows.Forms.TextBox()
    Me.Label4 = New System.Windows.Forms.Label()
    Me.Label2 = New System.Windows.Forms.Label()
    Me.txtClass = New System.Windows.Forms.TextBox()
    Me.Label1 = New System.Windows.Forms.Label()
    Me.grdProps = New System.Windows.Forms.DataGridView()
    Me.Label5 = New System.Windows.Forms.Label()
    Me.btnAddFilter = New System.Windows.Forms.Button()
    Me.btnAddOrder = New System.Windows.Forms.Button()
    Me.tabBottom = New System.Windows.Forms.TabControl()
    Me.pagFilters = New System.Windows.Forms.TabPage()
    Me.grdFilters = New System.Windows.Forms.DataGridView()
    Me.pagOrders = New System.Windows.Forms.TabPage()
    Me.grdOrders = New System.Windows.Forms.DataGridView()
    Me.colFieldOrder = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colMode = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.btnRemove = New System.Windows.Forms.Button()
    Me.colField = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colOperator = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colVar1 = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colTestVal1 = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colVar2 = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colTestVal2 = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colSelect = New System.Windows.Forms.DataGridViewCheckBoxColumn()
    Me.colName = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colType = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.colAssociation = New System.Windows.Forms.DataGridViewTextBoxColumn()
    Me.GroupBox1.SuspendLayout()
    CType(Me.grdProps, System.ComponentModel.ISupportInitialize).BeginInit()
    Me.tabBottom.SuspendLayout()
    Me.pagFilters.SuspendLayout()
    CType(Me.grdFilters, System.ComponentModel.ISupportInitialize).BeginInit()
    Me.pagOrders.SuspendLayout()
```

```
CType(Me.grdOrders, System.ComponentModel.ISupportInitialize).BeginInit()
Me.SuspendLayout()
'
'GroupBox1
'
Me.GroupBox1.Controls.Add(Me.txtAlias)
Me.GroupBox1.Controls.Add(Me.Label6)
Me.GroupBox1.Controls.Add(Me.cmbJoin)
Me.GroupBox1.Controls.Add(Me.cmbRelation)
Me.GroupBox1.Controls.Add(Me.Label3)
Me.GroupBox1.Controls.Add(Me.txtTable)
Me.GroupBox1.Controls.Add(Me.Label4)
Me.GroupBox1.Controls.Add(Me.Label2)
Me.GroupBox1.Controls.Add(Me.txtClass)
Me.GroupBox1.Controls.Add(Me.Label1)
Me.GroupBox1.Location = New System.Drawing.Point(12, 12)
Me.GroupBox1.Name = "GroupBox1"
Me.GroupBox1.Size = New System.Drawing.Size(413, 105)
Me.GroupBox1.TabIndex = 1
Me.GroupBox1.TabStop = False
Me.GroupBox1.Text = "Query Properties"
'
'txtAlias
'
Me.txtAlias.Location = New System.Drawing.Point(60, 74)
Me.txtAlias.Name = "txtAlias"
Me.txtAlias.Size = New System.Drawing.Size(121, 20)
Me.txtAlias.TabIndex = 18
'
'Label6
'
Me.Label6.AutoSize = True
Me.Label6.Location = New System.Drawing.Point(22, 77)
Me.Label6.Name = "Label6"
Me.Label6.Size = New System.Drawing.Size(29, 13)
Me.Label6.TabIndex = 17
Me.Label6.Text = "Alias"
'
'cmbJoin
'
Me.cmbJoin.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
Me.cmbJoin.FormattingEnabled = True
Me.cmbJoin.Location = New System.Drawing.Point(235, 47)
Me.cmbJoin.Name = "cmbJoin"
Me.cmbJoin.Size = New System.Drawing.Size(172, 21)
Me.cmbJoin.TabIndex = 16
'
'cmbRelation
'
Me.cmbRelation.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList
Me.cmbRelation.FormattingEnabled = True
Me.cmbRelation.Location = New System.Drawing.Point(60, 47)
Me.cmbRelation.Name = "cmbRelation"
Me.cmbRelation.Size = New System.Drawing.Size(121, 21)
Me.cmbRelation.TabIndex = 15
'
'Label3
'
```

```
        Me.Label3.AutoSize = True
        Me.Label3.Location = New System.Drawing.Point(203, 50)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New System.Drawing.Size(26, 13)
        Me.Label3.TabIndex = 14
        Me.Label3.Text = "Join"
        '
        'txtTable
        '
        Me.txtTable.Enabled = False
        Me.txtTable.Location = New System.Drawing.Point(235, 21)
        Me.txtTable.Name = "txtTable"
        Me.txtTable.Size = New System.Drawing.Size(172, 20)
        Me.txtTable.TabIndex = 13
        '
        'Label4
        '
        Me.Label4.AutoSize = True
        Me.Label4.Location = New System.Drawing.Point(8, 50)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New System.Drawing.Size(46, 13)
        Me.Label4.TabIndex = 12
        Me.Label4.Text = "Relation"
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Location = New System.Drawing.Point(195, 24)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New System.Drawing.Size(34, 13)
        Me.Label2.TabIndex = 11
        Me.Label2.Text = "Table"
        '
        'txtClass
        '
        Me.txtClass.Enabled = False
        Me.txtClass.Location = New System.Drawing.Point(60, 21)
        Me.txtClass.Name = "txtClass"
        Me.txtClass.Size = New System.Drawing.Size(121, 20)
        Me.txtClass.TabIndex = 10
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Location = New System.Drawing.Point(22, 24)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New System.Drawing.Size(32, 13)
        Me.Label1.TabIndex = 9
        Me.Label1.Text = "Class"
        '
        'grdProps
        '
        Me.grdProps.AllowUserToAddRows = False
        Me.grdProps.AllowUserToDeleteRows = False
        Me.grdProps.AllowUserToOrderColumns = True
        Me.grdProps.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
```

```
        Me.grdProps.Columns.AddRange(New System.Windows.Forms.DataGridViewColumn()
{Me.colSelect, Me.colName, Me.colType, Me.colAssociation})
        Me.grdProps.Location = New System.Drawing.Point(12, 137)
        Me.grdProps.MultiSelect = False
        Me.grdProps.Name = "grdProps"
        Me.grdProps.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect
        Me.grdProps.Size = New System.Drawing.Size(413, 135)
        Me.grdProps.TabIndex = 2
        '
        'Label5
        '
        Me.Label5.AutoSize = True
        Me.Label5.Location = New System.Drawing.Point(12, 121)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New System.Drawing.Size(124, 13)
        Me.Label5.TabIndex = 3
        Me.Label5.Text = "Class Mapped Properties"
        '
        'btnAddFilter
        '
        Me.btnAddFilter.Location = New System.Drawing.Point(12, 278)
        Me.btnAddFilter.Name = "btnAddFilter"
        Me.btnAddFilter.Size = New System.Drawing.Size(120, 23)
        Me.btnAddFilter.TabIndex = 4
        Me.btnAddFilter.Text = "Add as Filter"
        Me.btnAddFilter.UseVisualStyleBackColor = True
        '
        'btnAddOrder
        '
        Me.btnAddOrder.Location = New System.Drawing.Point(299, 278)
        Me.btnAddOrder.Name = "btnAddOrder"
        Me.btnAddOrder.Size = New System.Drawing.Size(120, 23)
        Me.btnAddOrder.TabIndex = 5
        Me.btnAddOrder.Text = "Add as Order"
        Me.btnAddOrder.UseVisualStyleBackColor = True
        '
        'tabBottom
        '
        Me.tabBottom.Controls.Add(Me.pagFilters)
        Me.tabBottom.Controls.Add(Me.pagOrders)
        Me.tabBottom.Location = New System.Drawing.Point(12, 307)
        Me.tabBottom.Name = "tabBottom"
        Me.tabBottom.SelectedIndex = 0
        Me.tabBottom.Size = New System.Drawing.Size(413, 151)
        Me.tabBottom.SizeMode = System.Windows.Forms.TabSizeMode.Fixed
        Me.tabBottom.TabIndex = 6
        '
        'pagFilters
        '
        Me.pagFilters.Controls.Add(Me.grdFilters)
        Me.pagFilters.Location = New System.Drawing.Point(4, 22)
        Me.pagFilters.Name = "pagFilters"
        Me.pagFilters.Padding = New System.Windows.Forms.Padding(3)
        Me.pagFilters.Size = New System.Drawing.Size(405, 125)
        Me.pagFilters.TabIndex = 0
        Me.pagFilters.Text = "Filters"
        Me.pagFilters.UseVisualStyleBackColor = True
```

```
        '
        'grdFilters
        '
        Me.grdFilters.AllowUserToAddRows = False
        Me.grdFilters.AllowUserToDeleteRows = False
        Me.grdFilters.AllowUserToOrderColumns = True
        Me.grdFilters.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.grdFilters.Columns.AddRange(New System.Windows.Forms.DataGridViewColumn()
{Me.colField, Me.colOperator, Me.colVar1, Me.colTestVal1, Me.colVar2, Me.colTestVal2})
        Me.grdFilters.Location = New System.Drawing.Point(4, 6)
        Me.grdFilters.MultiSelect = False
        Me.grdFilters.Name = "grdFilters"
        Me.grdFilters.ReadOnly = True
        Me.grdFilters.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect
        Me.grdFilters.Size = New System.Drawing.Size(396, 115)
        Me.grdFilters.TabIndex = 3
        '
        'pagOrders
        '
        Me.pagOrders.Controls.Add(Me.grdOrders)
        Me.pagOrders.Location = New System.Drawing.Point(4, 22)
        Me.pagOrders.Name = "pagOrders"
        Me.pagOrders.Padding = New System.Windows.Forms.Padding(3)
        Me.pagOrders.Size = New System.Drawing.Size(405, 125)
        Me.pagOrders.TabIndex = 1
        Me.pagOrders.Text = "Orders"
        Me.pagOrders.UseVisualStyleBackColor = True
        '
        'grdOrders
        '
        Me.grdOrders.AllowUserToAddRows = False
        Me.grdOrders.AllowUserToDeleteRows = False
        Me.grdOrders.AllowUserToOrderColumns = True
        Me.grdOrders.ColumnHeadersHeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersHeightSizeMode.AutoSize
        Me.grdOrders.Columns.AddRange(New System.Windows.Forms.DataGridViewColumn()
{Me.colFieldOrder, Me.colMode})
        Me.grdOrders.Location = New System.Drawing.Point(4, 6)
        Me.grdOrders.MultiSelect = False
        Me.grdOrders.Name = "grdOrders"
        Me.grdOrders.ReadOnly = True
        Me.grdOrders.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.FullRowSelect
        Me.grdOrders.Size = New System.Drawing.Size(396, 115)
        Me.grdOrders.TabIndex = 4
        '
        'colFieldOrder
        '
        Me.colFieldOrder.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colFieldOrder.DataPropertyName = "Field"
        Me.colFieldOrder.HeaderText = "Field"
        Me.colFieldOrder.Name = "colFieldOrder"
        Me.colFieldOrder.ReadOnly = True
        Me.colFieldOrder.Width = 54
        '
```

```
        'colMode
        '
        Me.colMode.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colMode.DataPropertyName = "Mode"
        Me.colMode.HeaderText = "Mode"
        Me.colMode.Name = "colMode"
        Me.colMode.ReadOnly = True
        Me.colMode.Width = 59
        '
        'btnRemove
        '
        Me.btnRemove.Location = New System.Drawing.Point(156, 278)
        Me.btnRemove.Name = "btnRemove"
        Me.btnRemove.Size = New System.Drawing.Size(120, 23)
        Me.btnRemove.TabIndex = 7
        Me.btnRemove.Text = "Remove"
        Me.btnRemove.UseVisualStyleBackColor = True
        '
        'colField
        '
        Me.colField.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colField.DataPropertyName = "Field"
        Me.colField.HeaderText = "Field"
        Me.colField.Name = "colField"
        Me.colField.ReadOnly = True
        Me.colField.Width = 54
        '
        'colOperator
        '
        Me.colOperator.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colOperator.DataPropertyName = "OperatorType"
        Me.colOperator.HeaderText = "Operator"
        Me.colOperator.Name = "colOperator"
        Me.colOperator.ReadOnly = True
        Me.colOperator.Width = 73
        '
        'colVar1
        '
        Me.colVar1.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colVar1.DataPropertyName = "Var1"
        Me.colVar1.HeaderText = "Var 1"
        Me.colVar1.Name = "colVar1"
        Me.colVar1.ReadOnly = True
        Me.colVar1.Width = 57
        '
        'colTestVal1
        '
        Me.colTestVal1.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colTestVal1.DataPropertyName = "Value"
        Me.colTestVal1.HeaderText = "Test Val 1"
        Me.colTestVal1.Name = "colTestVal1"
        Me.colTestVal1.ReadOnly = True
        Me.colTestVal1.Visible = False
```

```
        Me.colTestVal1.Width = 80
        '
        'colVar2
        '
        Me.colVar2.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colVar2.DataPropertyName = "Var2"
        Me.colVar2.HeaderText = "Var 2"
        Me.colVar2.Name = "colVar2"
        Me.colVar2.ReadOnly = True
        Me.colVar2.Width = 57
        '
        'colTestVal2
        '
        Me.colTestVal2.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colTestVal2.DataPropertyName = "ValueTo"
        Me.colTestVal2.HeaderText = "Test Val 2"
        Me.colTestVal2.Name = "colTestVal2"
        Me.colTestVal2.ReadOnly = True
        Me.colTestVal2.Visible = False
        Me.colTestVal2.Width = 80
        '
        'colSelect
        '
        Me.colSelect.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.ColumnHeader
        Me.colSelect.DataPropertyName = "Selected"
        Me.colSelect.Frozen = True
        Me.colSelect.HeaderText = "Sel"
        Me.colSelect.Name = "colSelect"
        Me.colSelect.Resizable = System.Windows.Forms.DataGridViewTriState.[False]
        Me.colSelect.Width = 28
        '
        'colName
        '
        Me.colName.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colName.DataPropertyName = "PropName"
        Me.colName.Frozen = True
        Me.colName.HeaderText = "Name"
        Me.colName.Name = "colName"
        Me.colName.ReadOnly = True
        Me.colName.ToolTipText = "Name of the Property"
        Me.colName.Width = 60
        '
        'colType
        '
        Me.colType.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colType.DataPropertyName = "TypeName"
        Me.colType.HeaderText = "Type"
        Me.colType.Name = "colType"
        Me.colType.ReadOnly = True
        Me.colType.Width = 56
        '
        'colAssociation
        '
```

```
        Me.colAssociation.AutoSizeMode =
System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells
        Me.colAssociation.DataPropertyName = "Association"
        Me.colAssociation.HeaderText = "Association"
        Me.colAssociation.Name = "colAssociation"
        Me.colAssociation.ReadOnly = True
        Me.colAssociation.Width = 86
        '
        'frmNodeProperties
        '
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(436, 467)
        Me.Controls.Add(Me.btnRemove)
        Me.Controls.Add(Me.tabBottom)
        Me.Controls.Add(Me.btnAddOrder)
        Me.Controls.Add(Me.btnAddFilter)
        Me.Controls.Add(Me.Label5)
        Me.Controls.Add(Me.grdProps)
        Me.Controls.Add(Me.GroupBox1)
        Me.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedToolWindow
        Me.Name = "frmNodeProperties"
        Me.StartPosition = System.Windows.Forms.FormStartPosition.CenterParent
        Me.Text = "Node Properties"
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox1.PerformLayout()
        CType(Me.grdProps, System.ComponentModel.ISupportInitialize).EndInit()
        Me.tabBottom.ResumeLayout(False)
        Me.pagFilters.ResumeLayout(False)
        CType(Me.grdFilters, System.ComponentModel.ISupportInitialize).EndInit()
        Me.pagOrders.ResumeLayout(False)
        CType(Me.grdOrders, System.ComponentModel.ISupportInitialize).EndInit()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents GroupBox1 As System.Windows.Forms.GroupBox
    Friend WithEvents cmbJoin As System.Windows.Forms.ComboBox
    Friend WithEvents cmbRelation As System.Windows.Forms.ComboBox
    Friend WithEvents Label3 As System.Windows.Forms.Label
    Friend WithEvents txtTable As System.Windows.Forms.TextBox
    Friend WithEvents Label4 As System.Windows.Forms.Label
    Friend WithEvents Label2 As System.Windows.Forms.Label
    Friend WithEvents txtClass As System.Windows.Forms.TextBox
    Friend WithEvents Label1 As System.Windows.Forms.Label
    Friend WithEvents Label5 As System.Windows.Forms.Label
    Friend WithEvents btnAddFilter As System.Windows.Forms.Button
    Friend WithEvents btnAddOrder As System.Windows.Forms.Button
    Friend WithEvents tabBottom As System.Windows.Forms.TabControl
    Friend WithEvents pagFilters As System.Windows.Forms.TabPage
    Friend WithEvents pagOrders As System.Windows.Forms.TabPage
    Friend WithEvents btnRemove As System.Windows.Forms.Button
    Friend WithEvents grdProps As System.Windows.Forms.DataGridView
    Friend WithEvents txtAlias As System.Windows.Forms.TextBox
    Friend WithEvents Label6 As System.Windows.Forms.Label
    Friend WithEvents grdFilters As System.Windows.Forms.DataGridView
    Friend WithEvents grdOrders As System.Windows.Forms.DataGridView
    Friend WithEvents colFieldOrder As System.Windows.Forms.DataGridViewTextBoxColumn
```

```
    Friend WithEvents colMode As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colField As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colOperator As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colVar1 As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colTestVal1 As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colVar2 As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colTestVal2 As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colSelect As System.Windows.Forms.DataGridViewCheckBoxColumn
    Friend WithEvents colName As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colType As System.Windows.Forms.DataGridViewTextBoxColumn
    Friend WithEvents colAssociation As System.Windows.Forms.DataGridViewTextBoxColumn
End Class
```

frmQueryBuilder.Design.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerated()> _
Partial Class frmQueryBuilder
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the component list.
    <System.Diagnostics.DebuggerNonUserCode()> _
    Protected Overrides Sub Dispose(ByVal disposing As Boolean)
        Try
            If disposing AndAlso components IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As System.ComponentModel.IContainer

    'NOTE: The following procedure is required by the Windows Form Designer
    'It can be modified using the Windows Form Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()> _
    Private Sub InitializeComponent()
        Dim resources As System.ComponentModel.ComponentResourceManager = New
System.ComponentModel.ComponentResourceManager(GetType(frmQueryBuilder))
        Me.eleQBuilder = New System.Windows.Forms.Integration.ElementHost()
        Me.WpfQueryBuilder1 = New DBClassMapper.wpfQueryBuilder()
        Me.SuspendLayout()
        '
        'eleQBuilder
        '
        Me.eleQBuilder.Dock = System.Windows.Forms.DockStyle.Fill
        Me.eleQBuilder.Location = New System.Drawing.Point(0, 0)
        Me.eleQBuilder.Name = "eleQBuilder"
        Me.eleQBuilder.Size = New System.Drawing.Size(804, 532)
        Me.eleQBuilder.TabIndex = 0
        Me.eleQBuilder.Text = "ElementHost1"
        Me.eleQBuilder.Child = Me.WpfQueryBuilder1
        '
        'frmQueryBuilder
        '
```

```
        Me.AutoScaleDimensions = New System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New System.Drawing.Size(804, 532)
        Me.Controls.Add(Me.eleQBuilder)
        Me.Icon = CType(resources.GetObject("$this.Icon"), System.Drawing.Icon)
        Me.Name = "frmQueryBuilder"
        Me.Text = "Query Builder"
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents eleQBuilder As System.Windows.Forms.Integration.ElementHost
    Friend WpfQueryBuilder1 As DBClassMapper.wpfQueryBuilder
End Class
```

wpfQueryBuilder.xaml

```
<UserControl x:Class="wpfQueryBuilder"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="600" d:DesignWidth="800">
    <Grid>
        <Grid Margin="2,2,2,2">
            <Grid.RowDefinitions>
                <RowDefinition Height="*" />
                <RowDefinition Height="3" />
                <RowDefinition Height="160" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="300" />
                <ColumnDefinition Width="4" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>

            <!-- GRID DO CANVAS ONDE SÃO JOGADAS TODAS AS CLASSES DENTRO -->
            <Grid Grid.Row="0" Grid.Column="2" Margin="0,2,2,0">
                <Border BorderThickness="2" BorderBrush="Black" CornerRadius="4,4,4,4">
                    <ScrollViewer VerticalScrollBarVisibility="Visible"
HorizontalScrollBarVisibility="Visible" x:Name="scrMain">
                        <Canvas Background="White" x:Name="canMain" MinWidth="400"
MinHeight="200" />
                    </ScrollViewer>
                </Border>
            </Grid>

            <!-- GRID DO MENU LATERAL ESQUERDO -->
            <Grid Grid.Row="0" Grid.Column="0" Margin="2,2,0,0">
                <Border BorderThickness="2" BorderBrush="Black" CornerRadius="4,4,4,4"
Background="OldLace">
                    <Grid Margin="2,2,2,2">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="Auto" />
                            <RowDefinition Height="*" />
                        </Grid.RowDefinitions>
```

```
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="*" />
                        </Grid.ColumnDefinitions>

                        <Label Grid.Row="0" Grid.Column="0" x:Name="lblAssociated"
Content="Associated Classes" />
                        <Button Grid.Row="0" Grid.Column="1" x:Name="btnChooseClass"
Content="Choose Main Class" MaxWidth="140" Margin="2,2,2,2" HorizontalAlignment="Right"
/>

                        <ListBox Grid.Row="1" Grid.ColumnSpan="2" x:Name="lsbAssociated"
MinHeight="200" />
                    </Grid>
                </Border>
            </Grid>

            <!-- COLUMN SPLITTER -->
            <GridSplitter Grid.Row="0" Grid.Column="1" ResizeDirection="Columns"
                          HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
Width="Auto" Height="Auto"/>

            <!-- ROW SPLITTER -->
            <GridSplitter Grid.Row="1" Grid.ColumnSpan="3" ResizeDirection="Rows"
                          HorizontalAlignment="Stretch" VerticalAlignment="Stretch"
Width="Auto" Height="Auto"/>

            <!-- GRID DO RODAPÉ ONDE FICAM OS CODIGOS GERADOS E OS TESTES -->
            <Grid Grid.Row="2" Grid.ColumnSpan="3" Margin="2,0,2,2">
                <Border BorderThickness="2" BorderBrush="Black" CornerRadius="4,4,4,4"
Background="OldLace">
                    <TabControl>
                        <TabItem Header="Code">
                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto" />
                                    <RowDefinition Height="*" />
                                </Grid.RowDefinitions>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*" />
                                </Grid.ColumnDefinitions>

                                <StackPanel Orientation="Horizontal"
HorizontalAlignment="Stretch">
                                    <Label Content="Target Language" Margin="2,2,2,2" />
                                    <ComboBox x:Name="cmbTargetLanguage" MinWidth="120"
MaxWidth="160" HorizontalAlignment="Stretch" Margin="2,2,2,2" />
                                    <Button x:Name="btnCopyToClipboard" Content="Copy to
Clipboard" MinWidth="120" MaxWidth="160" HorizontalAlignment="Stretch" Margin="2,2,2,2"
/>
                                </StackPanel>

                                <TextBox Grid.Row="1" Grid.Column="0" IsReadOnly="True"
Height="Auto" TextWrapping="Wrap" AcceptsReturn="True"
                                         VerticalScrollBarVisibility="Auto"
Name="txtCode" FontFamily="Courier New" FontSize="12" />
                            </Grid>
                        </TabItem>
```

```
                        <TabItem Header="Filters">
                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto" />
                                    <RowDefinition Height="*" />
                                </Grid.RowDefinitions>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*" />
                                </Grid.ColumnDefinitions>

                                <StackPanel Grid.Row="0" Grid.Column="0"
Orientation="Horizontal">
                                    <Button Content="Send One Up" MinWidth="100"
MaxWidth="120" x:Name="btnUpFilter" />
                                    <Button Content="Send One Down" MinWidth="100"
MaxWidth="120" x:Name="btnDownFilter" />
                                </StackPanel>

                                <DataGrid Grid.Row="1" Grid.Column="0"
                                          IsReadOnly="True" x:Name="dtgFilters"
CanUserReorderColumns="False"
                                          AutoGenerateColumns="False"
CanUserAddRows="False" CanUserResizeRows="False" SelectionMode="Single">
                                    <DataGrid.Columns>
                                        <DataGridTextColumn Header="Field"
IsReadOnly="True" Binding="{Binding Field}" />
                                        <DataGridTextColumn Header="Operator"
IsReadOnly="True" Binding="{Binding OperatorType}"/>
                                        <DataGridTextColumn Header="Var 1"
IsReadOnly="True" Binding="{Binding Var1}"/>
                                        <!--<DataGridTextColumn Header="Test Value 1"
IsReadOnly="True" Binding="{Binding Value}"/>-->
                                        <DataGridTextColumn Header="Var 2"
IsReadOnly="True" Binding="{Binding Var2}"/>
                                        <!--<DataGridTextColumn Header="Test Value 2"
IsReadOnly="True" Binding="{Binding ValueTo}"/>-->
                                    </DataGrid.Columns>
                                </DataGrid>
                            </Grid>
                        </TabItem>

                        <TabItem Header="Orders">
                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto" />
                                    <RowDefinition Height="*" />
                                </Grid.RowDefinitions>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*" />
                                </Grid.ColumnDefinitions>

                                <StackPanel Grid.Row="0" Grid.Column="0"
Orientation="Horizontal">
                                    <Button Content="Send One Up" MinWidth="100"
MaxWidth="120" x:Name="btnUpOrder" />
                                    <Button Content="Send One Down" MinWidth="100"
MaxWidth="120" x:Name="btnDownOrder" />
```

```
                            </StackPanel>

                            <DataGrid Grid.Row="1" Grid.Column="0"
                                      IsReadOnly="True" x:Name="dtgOrders"
CanUserReorderColumns="False"
                                      AutoGenerateColumns="False"
CanUserAddRows="False" CanUserResizeRows="False" SelectionMode="Single">
                                <DataGrid.Columns>
                                    <DataGridTextColumn Header="Field"
IsReadOnly="True" Binding="{Binding Field}" />
                                    <DataGridTextColumn Header="Mode"
IsReadOnly="True" Binding="{Binding Mode}"/>
                                </DataGrid.Columns>
                            </DataGrid>
                        </Grid>
                    </TabItem>

                    <TabItem Header="Aggregators">
                        <Grid>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="AUTO" />
                                <RowDefinition Height="*" />
                            </Grid.RowDefinitions>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="*" />
                            </Grid.ColumnDefinitions>

                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="AUTO" />
                                </Grid.RowDefinitions>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="AUTO" />
                                    <ColumnDefinition Width="*" />
                                    <ColumnDefinition Width="*" />
                                    <ColumnDefinition Width="AUTO" />
                                    <ColumnDefinition Width="*" />
                                    <ColumnDefinition Width="*" />
                                    <ColumnDefinition Width="AUTO" />
                                    <ColumnDefinition Width="AUTO" />
                                    <ColumnDefinition Width="AUTO" />
                                </Grid.ColumnDefinitions>

                                <Label Grid.Row="0" Grid.Column="0" Margin="2,2,2,2"
Content="Group By" />
                                <ComboBox Grid.Row="0" Grid.Column="1"
Margin="2,2,2,2" x:Name="cmbGroupTable" />
                                <ComboBox Grid.Row="0" Grid.Column="2"
Margin="2,2,2,2" x:Name="cmbGroupField" />

                                <Label Grid.Row="0" Grid.Column="3" Margin="2,2,2,2"
Content="Aggregate" />
                                <ComboBox Grid.Row="0" Grid.Column="4"
Margin="2,2,2,2" x:Name="cmbAggregTable" />
                                <ComboBox Grid.Row="0" Grid.Column="5"
Margin="2,2,2,2" x:Name="cmbAggregField" />
```

```
                                    <Label Grid.Row="0" Grid.Column="6" Margin="2,2,2,2"
Content="Type" />
                                    <ComboBox Grid.Row="0" Grid.Column="7" Width="70"
Margin="2,2,2,2" x:Name="cmbType" />

                                    <Button Grid.Row="0" Grid.Column="8" Content="Add
Aggregator" Width="100" Margin="2,2,2,2" x:Name="cmbAddAggreg" />
                                </Grid>

                                <DataGrid Grid.Row="1" Grid.Column="0"
                                          IsReadOnly="True" x:Name="dtgAggreg"
CanUserReorderColumns="False"
                                          AutoGenerateColumns="False"
CanUserAddRows="False" CanUserResizeRows="False" SelectionMode="Single">
                                    <DataGrid.Columns>
                                        <DataGridTextColumn Header="Group By"
IsReadOnly="True" Binding="{Binding GroupByField}" />
                                        <DataGridTextColumn Header="Aggregate"
IsReadOnly="True" Binding="{Binding AggregateField}"/>
                                        <DataGridTextColumn Header="Type"
IsReadOnly="True" Binding="{Binding AggregatorFunction}"/>
                                    </DataGrid.Columns>
                                </DataGrid>
                            </Grid>
                        </TabItem>

                        <!--<TabItem Header="Test">
                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto" />
                                    <RowDefinition Height="*" />
                                </Grid.RowDefinitions>
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="*" />
                                </Grid.ColumnDefinitions>

                                <Grid Grid.Row="0" Grid.Column="0">
                                    <Grid.RowDefinitions>
                                        <RowDefinition Height="Auto" />
                                    </Grid.RowDefinitions>
                                    <Grid.ColumnDefinitions>
                                        <ColumnDefinition Width="Auto" />
                                        <ColumnDefinition Width="Auto" />
                                        <ColumnDefinition Width="*" />
                                        <ColumnDefinition Width="Auto" />
                                        <ColumnDefinition Width="Auto" />
                                    </Grid.ColumnDefinitions>

                                    <Label Grid.Row="0" Grid.Column="0"
Content="Connection" Margin="2,2,2,2" />
                                    <Button Grid.Row="0" Grid.Column="1" Content="New
DSN" MinWidth="100" Margin="2,2,2,2" Name="btnNewDSN" />
                                    <ComboBox Grid.Row="0" Grid.Column="2"
Margin="2,2,2,2" Name="cmbDSN" />
                                    <Button Grid.Row="0" Grid.Column="3" Content="Test"
MinWidth="100" Margin="2,2,2,2" Name="btnTest" />
                                    <Button Grid.Row="0" Grid.Column="4" Content="See
Results" MinWidth="100" Margin="2,2,2,2" Name="btnSeeResults" />
```

```
                                </Grid>

                                <TextBox Grid.Row="1" Grid.Column="0" IsReadOnly="True"
Height="Auto" TextWrapping="Wrap"
                                         AcceptsReturn="True"
VerticalScrollBarVisibility="Auto" FontFamily="Courier New" FontSize="12" />

                            </Grid>
                        </TabItem>-->
                    </TabControl>
                </Border>
            </Grid>
        </Grid>

        <!-- GRID QUE FICA ESCONDIDO E É UTILIZADO PARA SELECIONAR A CLASSE PRINCIPAL -->
        <Grid MaxWidth="400" Width="Auto" MaxHeight="500"  Height="Auto"
Background="OldLace" x:Name="grdClasses">
            <Border BorderThickness="2" BorderBrush="Black" CornerRadius="4,4,4,4">
                <Grid>
                    <Grid.RowDefinitions>
                        <RowDefinition Height="Auto" />
                        <RowDefinition Height="*" />
                        <RowDefinition Height="Auto" />
                    </Grid.RowDefinitions>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="*" />
                    </Grid.ColumnDefinitions>

                    <Grid>
                        <Label x:Name="lblClasses" Content="Choose the Class to get a
List" />
                        <Border Width="19" Height="18" Margin="2,2,2,2"
HorizontalAlignment="Right"
                                BorderThickness="1" BorderBrush="Black"
CornerRadius="2,2,2,2" Background="LightGray"
                                x:Name="brdClose">
                            <Label Content="x" FontWeight="Bold" Margin="0,-5, 0, 0" />
                        </Border>
                    </Grid>
                    <ListBox Grid.Row="1" Grid.Column="0" x:Name="lsbClasses"
                                  MinHeight="200" MaxHeight="500" MaxWidth="400" />

                    <Grid Grid.Row="2" Grid.Column="0" Margin="2,2,2,2">
                        <Grid.RowDefinitions>
                            <RowDefinition Height="Auto" />
                        </Grid.RowDefinitions>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="Auto" />
                            <ColumnDefinition Width="*" />
                            <ColumnDefinition Width="Auto" />
                        </Grid.ColumnDefinitions>

                        <Label Grid.Row="0" Grid.Column="0" x:Name="lblRef"
Content="Reference" />
                        <ComboBox Grid.Row="0" Grid.Column="1" x:Name="cmbReference"
MinWidth="160" MaxHeight="24" />
```

```
                        <Button Grid.Row="0" Grid.Column="2" x:Name="btnSearch"
MinWidth="80" MaxWidth="140" MaxHeight="24" Content="Search" />

                    </Grid>
                </Grid>
            </Border>
        </Grid>
    </Grid>
</UserControl>
```