UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

QEA - INTEGRAÇÃO ENTRE A FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS WEB QUELLON E O ENTERPRISE ARCHITECT

BRUNA EMERICH DALL OLIVO DE SOUZA

BLUMENAU
2011

2011/2-07

BRUNA EMERICH DALL OLIVO DE SOUZA

QEA - INTEGRAÇÃO ENTRE A FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS WEB QUELLON E O ENTERPRISE ARCHITECT

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

Prof. Jacques Robert Heckmann, Mestre - Orientador

BLUMENAU
2011

2011/2-07

QEA - INTEGRAÇÃO ENTRE A FERRAMENTA PARA DESENVOLVIMENTO DE SISTEMAS WEB QUELLON E O ENTERPRISE ARCHITECT

Por

BRUNA EMERICH DALL OLIVO DE SOUZA

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

Presidente:	Prof. Jacques Robert Heckmann, Mestre – Orientador, FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB

Blumenau, 06 de dezembro de 2011.

Dedico este trabalho à meus pais Karla e Júlio, pessoas que me ensinaram que pra vencer na vida é preciso dedicação, garra, caráter, coragem e educação.

AGRADECIMENTOS

A Deus, por me dar forças diante das dificuldades.

A minha irmã Mariah e meu cunhado Mayco, pela amizade, carinho e amor.

Obrigada aos meus pais, por me ajudarem a concluir essa etapa tão preciosa pra mim. Obrigada por me ensinarem que sem educação não somos nada, que a batalha será difícil, mas a recompensa será grande. Obrigada por me apoiarem sempre.

Agradeço à minha melhor amiga Camila Labes, por estar sempre ao lado, por compartilhar comigo todo seu conhecimento, por me ajudar tanto a concluir essa etapa, por me mostrar o caminho a seguir para chegar à solução quando eu só conseguia ver o problema. Obrigada de coração pela tua amizade, por confiar em mim. Obrigada por ser quem você é!

Ao meu namorado Ricardo Linhares, obrigada por entrar na minha vida e me deixar entrar na tua. Obrigada pela confiança, pelo amor, carinho e amizade. Obrigada por estar sempre ao meu lado me apoiando e ajudando.

Agradeço a toda equipe Quellon, por me apoiarem, com um agradecimento especial à Taynara Bittelbrunn, que sempre esteve preocupada e disposta a me ajudar a concluir este trabalho.

Agradeço ao meu orientador Jacques Robert Heckmann, por acreditar no meu potencial, obrigada por me ajudar a concluir esse trabalho.

Faça o que pode, com o que tem, onde estiver.

Roosevelt

RESUMO

Este trabalho apresenta a ferramenta QEA para integrar a ferramenta de desenvolvimento para sistemas *web* Quellon e o *Enterprise Architect*. Ela automatiza a passagem da estrutura lógica de um sistema exposto em um diagrama de classes no *Enterprise Architect*, para o banco de dados do sistema que está sendo desenvolvido na ferramenta Quellon. Como resultado, a ferramenta permite reduzir os erros que o trabalho manual pode inserir, além de agilizar o processo de criação da base de dados. A ferramenta tratará ligações do tipo associação e multiplicidades um para muitos.

Palavras-chave: Ferramentas CASE. Diagrama de Classes. Banco de Dados.

ABSTRACT

This paper presents the QEA tool for integrating development tool for web systems Quellon and *Enterprise Architect*. It automates the passage of the logical structure of a system exposed in a class diagram in *Enterprise Architect* for the database system being developed in Quellon tool. As a result, the tool allows to reduce the errors that manual labor can enter, and streamline the process of creating the database. The tool treat type connector and associated manifolds one to many.

Keywords: CASE tools. Class Diagram. Database.

LISTA DE ILUSTRAÇÕES

Figura 1: Formulário de um sistema Quellon 17
Figura 2: Tela de *login* 18
Figura 3: Enterprise Architect 20
Figura 4: Criando uma nova tabela em um sistema que utiliza Quellon 24
Figura 5: Selecionando o tipo do membro que será criado 25
Figura 6: Criando um membro do tipo *string* 25
Figura 7: Diagrama de Atividades do sistema atual 26
Figura 8: Tela do TCC de Raphael Marcos Batista 27
Figura 9: Tela do TCC de André Luis Becker 27
Quadro 1: Requisitos funcionais 29
Quadro 2: Requisitos não funcionais 29
Figura 10: Diagrama de caso de uso 30
Figura 11: Exportação do XML do EA 31
Figura 12: Parte do XML do diagrama 31
Figura 13: Leitura das tabelas do XML do EA 32
Figura 14: Chamada da DLL Quellon.Update para criação de tabelas 32
Figura 15: Leitura dos campos 33
Figura 16: Chamada da DLL Quellon.Update para criação de campos 33
Figura 17: Criando os tipos de campos da Quellon 34
Figura 18: Linguagem Quellon 34
Figura 19: Tipos de dados Quellon 35
Figura 20: Leitura das FKs dentro do XML do EA 35
Figura 21: Visual Studio 38
Figura 22: Diagrama de Atividades do funcionamento da ferramenta QEA 38
Figura 23: Pasta com os arquivos da ferramenta de integração 39
Figura 24: Tela inicial da ferramenta 40
Figura 25: Tela de cadastro do sistema 40
Figura 26: Erro ao cadastrar sistema 41
Figura 27: Botão para importação do diagrama de classes 41
Figura 28: Importar Arquivo 42
Figura 29: Tabelas que estão no XML que será importado 42

Figura 30: Status da Importação 43
Figura 31: Atualização finalizada 43
Figura 32: Código *log* de erro 43
Figura 33: exemplo do arquivo logErro.txt 43
Figura 34: Código tabelas criadas 44
Figura 35: Exemplo do arquivo tabelasCriadas.txt 44
Figura 36: *Select* antes da importação do diagrama de classe 44
Figura 37: Diagrama de classes com duas classes 45
Figura 38: *Select* após importação do diagrama de classes 45
Figura 39: Diagrama de classes de uma locadora 46
Figura 40: *Select* em aaa_objects após importação do diagrama de classes 46
Figura 41: *Select* em aaa_members após importação do diagrama de classes 47
Figura 42: Objetos criados no sistema 47
Figura 43: Campos criados na tabela DVD 48
Quadro 3: Caso de uso “Cadastrar Sistema” 54
Quadro 4: Caso de uso “Gerar estrutura” 55

LISTA DE SIGLAS

CLS - *Commom Language Specification*

C# - C-Sharp

DLL – *Dynamic Library Link*

EA - *Enterprise Architect*

FK - *Foreign Key*

IDE - Ambiente de Desenvolvimento Integrado

OMG - *Object Management Group*

RF - Requisitos Funcionais

RNF - Requisitos Não Funcionais

UML - *Unified Modeling Language*

XAML – *eXtensible Markup Language Metadata Interchange*

XMI - *XML Metadata Interchange*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 12
1.1 OBJETIVOS DO TRABALHO 13
1.2 ESTRUTURA DO TRABALHO 14
2 FUNDAMENTAÇÃO TEÓRICA 15
2.1 DIAGRAMA DE CLASSE 15
2.1.1 Classes 15
2.1.2 Relacionamento entre objetos (associação) 16
2.1.3 Atributos 16
2.2 QUELLON 17
2.3 ENTERPRISE ARCHITECT 19
2.4 BANCO DE DADOS 20
2.5 XMI 21
2.6 MAPEAMENTO DE OBJETOS PARA O MODELO RELACIONAL 22
2.6.1 Mapeamento de classes em tabelas 22
2.6.2 Mapeamento de atributos em colunas 23
2.6.3 Mapeamento de associações 23
2.7 SISTEMA ATUAL 23
2.8 TRABALHOS CORRELATOS 26
3 DESENVOLVIMENTO DA FERRAMENTA 28
3.1 LEVANTAMENTOS DE INFORMAÇÕES 28
3.2 ESPECIFICAÇÃO 28
3.2.1 Requisitos Funcionais 29
3.2.2 Requisitos Não Funcionais 29
3.2.3 Casos de Uso 29
3.3 IMPLEMENTAÇÃO 30
3.3.1 Técnicas utilizadas 30
3.3.2 Ferramentas utilizadas 36
3.3.2.1 C# na plataforma .NET 36
3.3.2.2 Plataforma .NET 37
3.3.2.3 Visual Studio 37
3.3.3 Operacionalidade da ferramenta 38

3.3.3.1 Instalação do executável 39
3.3.3.2 Adição do sistema 39
3.3.3.3 Importação dos dados 41
3.4 RESULTADOS E DISCUSSÃO 48
4 CONCLUSÕES 50
4.1 EXTENSÕES 50
REFERÊNCIAS BIBLIOGRÁFICAS 52
APÊNDICE A – DETALHAMENTO DOS CASOS DE USO 54

1 INTRODUÇÃO

Ferramentas CASE facilitam o desenvolvimento de qualquer software.

> Ferramentas de engenharia de software apoiada por computador (CASE) cobrem a atividade do processo de software e aquelas atividades guarda-chuva aplicadas ao longo do processo. Pode ser utilizada tanto em atividades gerenciais quanto técnicas. (PRESSMAN, 2002, p. 823).

Percebe-se muitas vezes que existe a necessidade de ferramentas CASE comunicarem-se, interagirem entre si. Isso normalmente ocorre porque uma mesma organização utiliza ferramentas criadas por fabricantes distintos, como por exemplo, uma ferramenta para análise do software e outra ferramenta para o seu desenvolvimento.

> O ambiente I-CASE combina mecanismos de integração para dados, ferramentas e interação homem/computador. A integração das ferramentas pode ser projetada sob medida por fornecedores que trabalham juntos, ou alcançada por intermédio de software de gestão fornecido como parte do repositório. (PRESSMAN, 2002, p.823).

Diagramas são os meios utilizados para a visualização dos modelos do sistema a ser desenvolvido. Bons diagramas facilitam a compreensão deste sistema (BOOCH; RUMBAUGH; JACOBSON, 2000, p.89).

Furlan (1998, p.91) afirma que "o diagrama de classes é a essência da UML. Trata-se de um diagrama mostrando uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações".

Os diagramas de classe são os diagramas encontrados com maior freqüência na modelagem de sistemas orientados a objetos. Eles são importantes para a visualização, a especificação e a documentação de modelos estruturais, assim como também para a construção de sistemas executáveis por intermédio de engenharia de produção reversa (BOOCH; RUMBAUGH; JACOBSON, 2000, p.104).

Existem várias ferramentas que geram diagramas de classes no mercado. A seleção de uma delas depende da necessidade de cada organização. Uma ferramenta eficaz e utilizada por muitas organizações na região de Blumenau, é o *Enterprise Architect* (EA). Com ela pode-se projetar todo um software a ser construído, pois abrange todas as fases do ciclo de vida do desenvolvimento de software, desde o levantamento das necessidades, desenvolvimento e manutenção, além de manter a documentação.

Segundo Sparx Systems (2011), o EA é uma ferramenta que pertence à companhia Sparx Systems Ply Ltda, um membro contribuinte do *Object Management Group* (OMG). A

empresa é especializada em ferramentas de modelagem visual para o planejamento, a concepção e a construção de sistemas de software complexos.

Com clientes que vão desde a indústria aeroespacial e engenharia automobilística para finanças, defesa, governo, entretenimento e comunicações, a Sparx Systems Ply Ltda é um fornecedor líder de soluções inovadoras baseadas em *Unified Modeling Language* (UML) e suas especificações relacionadas.

O produto líder de vendas da companhia, o EA, recebeu várias homenagens desde o lançamento comercial em agosto de 2000. Agora na versão 9.2, o EA é a ferramenta de *design* de escolha para mais de 250.000 usuários registrados em todo mundo (SPARX SYSTEMS, 2011).

A empresa Quellon do Brasil Sistemas S.A está no mercado há 9 anos com a missão de criar, manter, comercializar e implantar ferramentas de desenvolvimento e sistemas de informação para internet. A empresa tem como principais acionistas os sócios da Fácil Informática, HB.Sis Informática e Ellevo Soluções em Tecnologia da Informação e conta com 9 colaboradores no seu quadro de funcionários. A ferramenta Quellon foi desenvolvida utilizando a plataforma .NET na linguagem C#.

Com o surgimento da necessidade de integrar a ferramenta Quellon com uma ferramenta de que possibilitasse criar diagramas de classes, a empresa Quellon do Brasil Sistemas S.A optou por integrar sua ferramenta ao EA, devido ao fato de importantes clientes utilizarem esta ferramenta para projetarem seus sistemas.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é fazer uma integração entre as ferramentas EA e Quellon, integrando o Diagrama de Classes do EA à estrutura física de banco de dados da ferramenta Quellon. Os objetivos específicos do trabalho são:

a) automatizar a passagem da estrutura de banco de dados exposta em diagrama de classes, para que tabelas, campos e *foreign keys* sejam criados apenas no EA e através de métodos de integração seja criada a estrutura física nas bases de dados dos sistemas que utilizam a ferramenta Quellon;
b) identificar dentro do padrão XMI como são armazenados os modelos de classes;
c) definir o mapeamento entre os diagramas de classes e o Quellon;

d) determinar conversões de tipos de dados existentes no EA para os oferecidos no Quellon.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo é apresentada uma introdução ao assunto abordado e os objetivos a serem alcançados pelo trabalho.

No segundo capítulo é feita uma revisão bibliográfica para o entendimento do trabalho, incluindo temas como diagrama de classes, Quellon, *Enterprise Architect*, banco de dados, XMI, mapeamento de objetos para o modelo relacional, uma descrição do sistema atual e por fim trabalhos correlatos.

O terceiro capítulo apresenta um levantamento de informações. Apresenta ainda os requisitos do problema tratado neste trabalho, assim como a especificação da ferramenta QEA, a implementação, a operacionalidade demonstrando o seu uso prático e os resultados e discussões tendo em vista o resultado da pesquisa realizada.

O quarto capítulo apresenta as conclusões e sugestões de extensão e melhorias deste trabalho para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão tratados os temas sobre diagrama de classes, Quellon, *Enterprise Architect*, banco de dados, XMI e mapeamento de objetos para o modelo relacional. Posteriormente relata-se sobre o sistema atual, bem como os trabalhos correlatos com objetivos semelhantes ao deste trabalho.

2.1 DIAGRAMA DE CLASSE

O diagrama de classes é utilizado na construção do modelo de classes desde o nível de análise até o nível de especificação. De todos os diagramas da UML, esse é o mais rico em termos de notação (BEZERRA, 2002, p.97). Os diagramas de classe utilizam classes e interfaces para documentar detalhes sobre as entidades que formam seu sistema e as relações estáticas entre elas (PILONE; PITMAN, 2006, p.5). Os diagramas de classe são uns dos mais fundamentais tipos de diagramas da UML, pois proporcionam uma maneira de capturar a estrutura "física" do sistema (PILONE; PITMAN, 2006, p.11). Um diagrama de classes mostra um conjunto de classes, interfaces, colaborações e seus relacionamentos (BOOCH, 2000, p.4).

Um diagrama de classes é composto por vários elementos, a seguir pode-se ter uma visão mais completa dos itens: classes, relacionamento entre objetos (associação) e atributos, que são os itens tratados na ferramenta QEA.

2.1.1 Classes

As classes são blocos de construção mais importante de qualquer software orientado a objetos. Uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamento e semântica (BOOCH, 2000, p.47). Uma classe representa um grupo de coisas que têm estado e comportamento comuns (PILONE; PITMAN, 2006, p.11).

Bezerra (2002, p.97) explica que uma classe é representada através de uma caixa com,

no máximo, três compartimentos exibidos. No primeiro compartimento (de cima para baixo) é exibido o nome da classe. No segundo compartimento, são declarados os atributos. Finalmente, no terceiro compartimento, são declaradas as operações.

2.1.2 Relacionamento entre objetos (associação)

Um relacionamento é uma conexão entre itens. Um relacionamento é representado graficamente como um caminho, com tipos diferentes de linhas, para diferenciar os tipos de relacionamentos (BOOCH, 2000, p.62). Para representar o fato de que objetos podem se relacionar uns com os outros, existe um elemento do diagrama de classes chamado associação. Este elemento representa relacionamentos que são formados entre objetos durante a execução do sistema. Uma associação é representada no diagrama de classes através de um segmento de reta ligando as classes às quais os objetos relacionados pertencem (BEZERRA, 2002, p.99).

Uma associação é um relacionamento estrutural que especifica objetos de um item conectado a objetos de outro item. A partir de uma associação conectando duas classes, se é capaz de navegar do objeto de uma classe até o objeto de outra classe e vice-versa (BOOCH, 2000, p.63).

2.1.3 Atributos

O atributo é a menor unidade em que si possui significância própria e inter-relacionada com o conceito lógico da qual a classe pertence. Apresenta um princípio de atomicidade, ou seja, do armazenamento de um valor simples em uma célula (FURLAN, 1998, p.102). Os detalhes de uma classe são representados como atributos. Os atributos podem ser tipos primitivos simples ou relações com outros objetos complexos (PILONE; PITMAN, 2006, p.12). Os atributos correspondem à descrição dos dados armazenados pelos objetos de uma classe. A cada atributo de uma classe está associado um conjunto de valores que esse atributo pode assumir (BEZERRA, 2002, p.98).

2.2 QUELLON

Segundo a Quellon do Brasil Sistemas S.A, a ferramenta Quellon é utilizada para desenvolvimento de aplicações *web* na plataforma .NET, porém, a ferramenta não é somente uma ferramenta de desenvolvimento. Deve-se entendê-la como um conjunto de ferramentas que envolvem todo o processo de desenvolvimento de um produto, tais como a criação de telas, a segurança, os componentes, as atualizações, as customizações, entre outros.

A criação de telas dos sistemas que utilizam Quellon é toda feita pela própria ferramenta. Após criar um objeto e seus campos, o usuário escolhe quais campos deseja adicionar na interface, com isso, a ferramenta Quellon monta o formulário com os campos escolhidos. A Figura 1 ilustra um formulário de um sistema que utiliza Quellon.

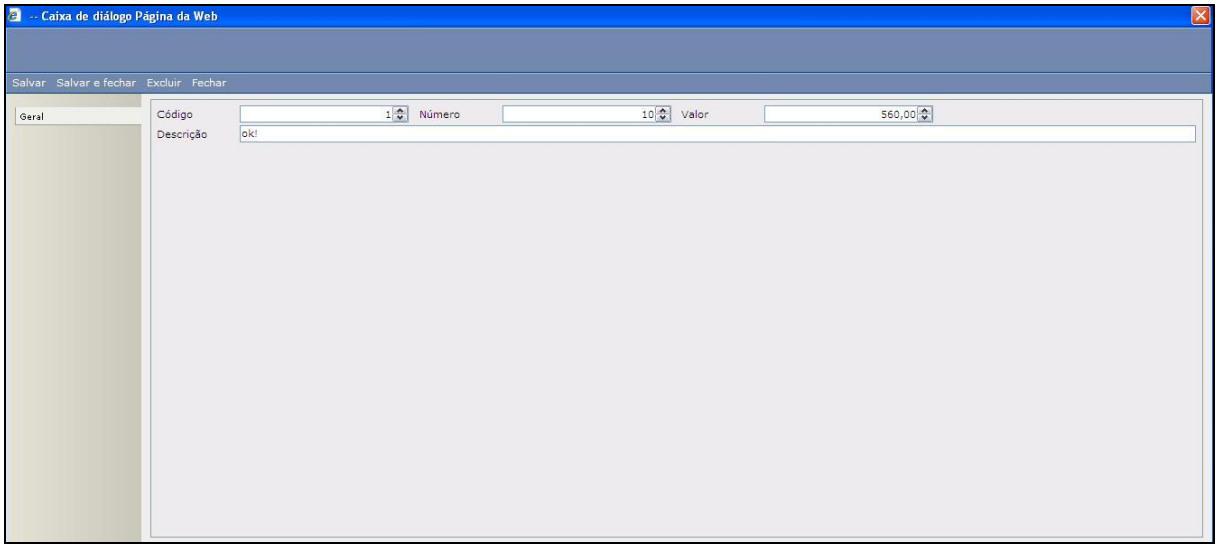


Figura 1: Formulário de um sistema Quellon

Os sistemas que utilizam Quellon possuem tipos de campos específicos, que são eles:

a) *string*: armazena uma cadeia de caracteres;
b) *int*: armazena números inteiros;
c) *numeric*: armazena números com ou sem vírgula;
d) *bool*: usado em campos que requerem apenas ‘sim’ ou ‘não’;
e) *datetime*: armazena a data, a hora, ou ambos;
f) *lookup*: aponta para um registro em outra tabela;
g) *memo*: utilizado em campos de texto grande;
h) *money*: armazena valores monetários;
i) *password*: armazena uma senha;
j) *document*: armazena um documento;

k) *image*: armazena uma imagem;

l) URL: armazena um *link* para um *site*;

m) *e-mail*: armazena um *e-mail*;

n) cor: armazena uma cor.

A ferramenta permite também a criação de páginas ASPX, feitas pelo próprio programador, e para isso, disponibiliza um pacote de componentes na dll Quellon.Lib, como componentes *TextBox*, *Numeric*, *DateTime* entre outros.

A segurança da ferramenta Quellon é tratada por grupos de usuários. Um grupo pode conter vários usuários e um usuário pode estar em vários grupos. A ferramenta permite configurar cada módulo, página, itens e até registros, garantido que cada usuário irá ter acesso apenas aquilo que lhe é permitido.

É possível fazer customizações utilizando a ferramenta Quellon. Alteração em cores, tela de *login* e dicionário são permitidas e mantidas, não sendo perdidas em cada atualização. A Figura 2 mostra uma tela de *login* de um sistema Quellon.

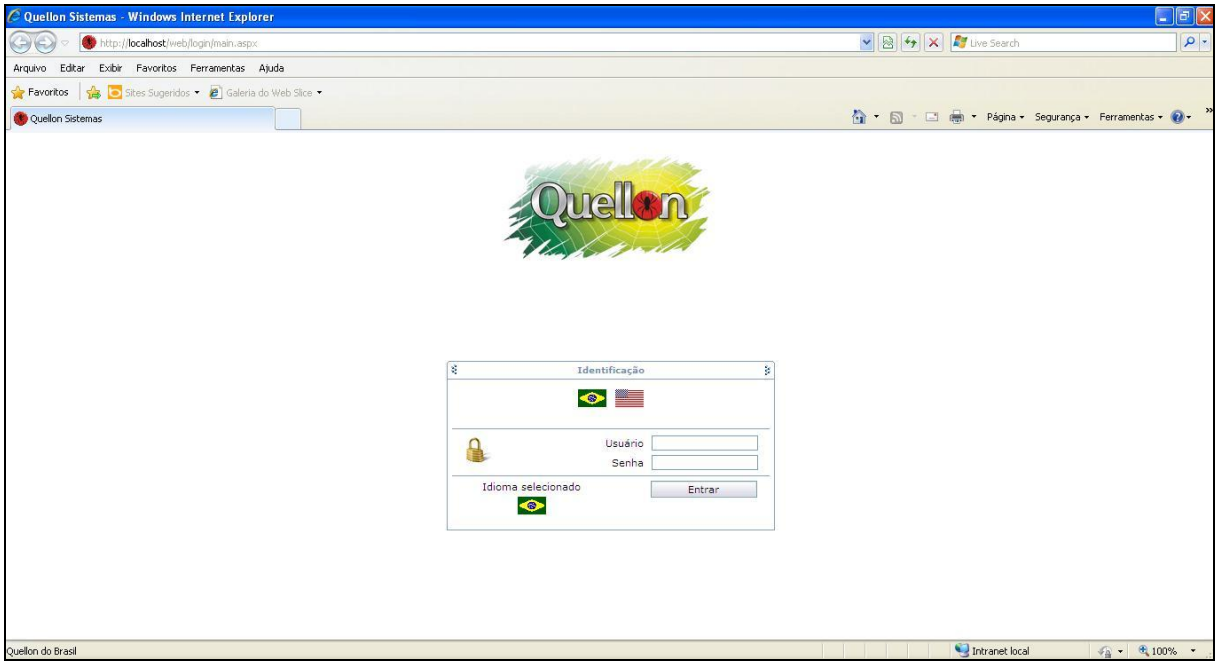


Figura 2: Tela de *login*

A ferramenta Quellon foi concebida utilizando a plataforma .NET na linguagem C#. Outra característica importante da ferramenta é que seu fundamento acontece quase por completo na *web*, ou seja, é uma ferramenta para desenvolver aplicativos e sistemas *web* (QUELLON DO BRASIL SISTEMAS S.A, 2004). Algumas características da ferramenta Quellon são:

a) trabalha com dois bancos de dados: Microsoft SQL Server e Oracle;
b) desenvolvimento de telas, também sendo possível criar uma tela toda pelo programador (ASPX);
c) componentes de tela próprios;
d) gerador de relatórios;
e) filtros;
f) manipulação de dados;
g) importação e exportação de dados;
h) atualizador do sistema.

2.3 ENTERPRISE ARCHITECT

Segundo Sparx Systems (2006), o EA é uma ferramenta CASE baseada na UML. O EA é utilizado no desenho e construção de projetos de sistemas de software e abrange todas as fases do ciclo de desenvolvimento do projeto do sistema de software, desde o levantamento das necessidades, o desenvolvimento até a manutenção.

De acordo com Lima (2005, p.41), o EA é uma ferramenta que cobre todos os aspectos do ciclo de desenvolvimento, fornecendo suporte para teste, manutenção e controle de mudanças e requisitos, além de diagramas e modelos UML 2. Algumas características do EA são:

a) modelagem de sistemas de software de acordo com a notação UML 2.1;
b) integração com o Visual Studio .Net e Eclipse;
c) importação e exportação de modelos no formatos XML;
d) controle de versão e comparação de modelos e criação de baseline;
e) automação e customização do EA nas linguagens VB, C++ e Delphi.

A Figura 3 ilustra o EA.

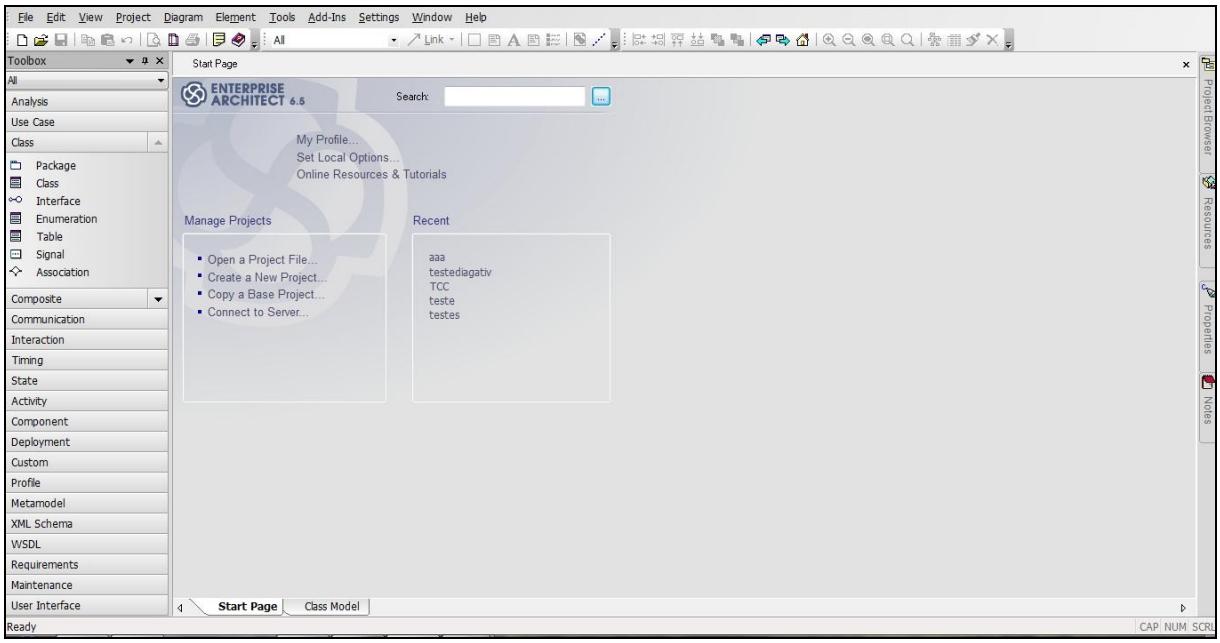


Figura 3: Enterprise Architect

2.4 BANCO DE DADOS

Os bancos de dados surgiram aproximadamente em meados dos anos 60, decorrentes da possibilidade dos computadores armazenarem e gerenciarem grandes quantidades de dados em meios de armazenamento permanente de acesso direto e eficiente a cada dado e da necessidade de estruturar esses dados e prover rotinas padronizadas de acesso a eles (SETZER; NASSU, 1999, p.1). Os bancos de dados e os sistemas de banco de dados se tornaram componentes essenciais no cotidiano da sociedade moderna (ELMASRI; NAVATHE, 2005, p.3)

De acordo com Date (1991, p.5), o sistema de banco de dados é basicamente um sistema de manutenção de registros por computador, ou seja, um sistema cujo objetivo global é manter as informações e torná-las disponíveis quando solicitadas. O sistema do banco de dados proporciona à empresa o controle centralizado dos seus dados operacionais (DATE, 1991, p.13).

Monteiro (2004, p.167) afirma que banco de dados é uma coleção abrangente, organizada e inter-relacionada de dados armazenados em um meio físico, com o objetivo de evitar ou minimizar duplicidade de informação, otimizar a eficácia de seu tratamento, permitindo o acesso, de diversas formas, a uma grande variedade de informações.

Um banco de dados é uma coleção de dados relacionados. Os dados são fatos que podem ser gravados e que possuem um significado implícito (ELMASRI; NAVATHE, 2005, p.4). Existem muitas vantagens se utilizar um sistema de banco de dados:

a) é compacto: não há necessidade de arquivos de papéis volumosos;
b) é rápido: a máquina pode recuperar e modificar os dados muito mais rapidamente do que o ser humano;
c) importa em menos trabalho braçal: elimina a maior parte do tedioso trabalho manual de arquivamento;
d) tem fluxo corrente: disponibilidade de informações certas e atualizadas a qualquer momento, basta pedir;
e) mantém o controle centralizado: reduz a redundância, a inconsistência pode ser evitada (até certo ponto), pode compartilhar os dados, pode manter a integridade.

2.5 XMI

Muitos dos esforços na engenharia de software dirigem-se à construção de ferramentas de apoio às tarefas comuns nessa área, como modelagem, desenvolvimento, manutenção e documentação. Para aproveitar melhor as funcionalidades oferecidas por essas ferramentas é interessante ter a possibilidade de trocar informações entre elas. Dessa forma, várias ferramentas independentes podem ser usadas em conjunto para formarem um poderoso ambiente de desenvolvimento e manutenção.

Para conseguir essa troca de informações, foram desenvolvidas várias linguagens comuns que toda ferramenta poderia interpretar. Essas linguagens permitem salvar os dados em uma ferramenta e ler esses mesmos dados dentro de outra ferramenta. Uma dessas linguagens, que está ganhando muito apoio, é a XMI definida pela OMG.

O XMI é um padrão da OMG que permite representar objetos usando o XML. Ele especifica mecanismos de conversão de modelos em documentos XML e vice-versa. Com respeito à modelação, o XMI especifica uma estrutura de representação de modelos conforme o metamodelo UML. O principal objetivo do XMI é permitir a interoperação e utilização dos modelos UML de forma independente das plataformas, linguagens, repositórios e ferramentas CASE (SILVA; VIDEIRA, 2005, p.290).

2.6 MAPEAMENTO DE OBJETOS PARA O MODELO RELACIONAL

O modelo relacional se fundamenta no conceito de relação. A maioria dos sistemas de banco de dados desenvolvidos são relacionais. Um sistema relacional é aquele em que os dados são percebidos pelo usuário como tabelas e os operadores geram novas tabelas a partir das antigas (DATE, 1991, p.21).

De acordo com Bezerra (2007, p.339) um conceito importante de modelo relacional é o de chave primária. Uma chave primária é uma coluna ou um conjunto de colunas cujos valores podem ser utilizados para identificar unicamente cada linha de uma relação. Outro conceito importante é o de chave estrangeira. Linhas de uma relação podem estar associadas a linhas de outras relações.

Quando um Sistema de Gerenciamento de Banco de Dados Relacional deve ser utilizado como mecanismo de armazenamento persistente de informações para um sistema de software orientado a objetos, há a necessidade de se realizar o mapeamento dos valores de atributos de objetos persistentes do sistema para tabelas. Abaixo serão tratados os tópicos mapeamento de classes em tabelas, mapeamento de atributos em colunas e mapeamento de associações, explicando a associação do tipo um para muitos, que são os mapeamentos tratados pela ferramenta QEA.

2.6.1 Mapeamento de classes em tabelas

O mapeamento de classes pode ser feito mediante a paridade entre classe e tabela, ou seja, uma classe é mapeada para uma tabela. Este mapeamento direto de classes para tabelas representa a forma mais simples de mapeamento, tornando mais fácil o entendimento e a manutenção de uma aplicação. Com um modelo bastante simples isto pode ser feito. Porém, nem sempre é simples assim. No caso de uma estrutura hierárquica, várias classes podem ser mapeadas para uma tabela, como também uma classe pode ser mapeada para várias tabelas.

2.6.2 Mapeamento de atributos em colunas

Ao tratar do mapeamento de atributos de uma classe para colunas em tabelas de um banco de dados relacional, deve-se levar em conta que os atributos podem ser de tipos de dados simples ou primários como: inteiros, ponto flutuante, caracteres, boleanos e binários, mas também podem ser de tipos de dados complexos como tipos baseados em outras classes. Os atributos podem ainda ser multivalorados (lista de objetos), o que viola as regras de normalização do modelo relacional. Além disso, podem existir atributos de controle ou utilizados em cálculos, que geralmente não necessitam ser mapeados.

Desta forma, os atributos simples podem ser mapeados diretamente para colunas em uma tabela, já os atributos complexos e multivalorados podem necessitar de tabelas adicionais para seu armazenamento. Estes atributos complexos geralmente possuem características recursivas, ou seja, são classes que possuem outros atributos e assim sucessivamente.

2.6.3 Mapeamento de associações

As associações entre classes no modelo orientado a objetos é conceitualmente bastante similar ao relacionamento entre tabelas no modelo relacional. Este fato permite que tais associações sejam mapeadas para relacionamentos, podendo utilizar chaves estrangeiras ou tabelas auxiliares.

A associação do tipo um para muitos, a associação tratada pela ferramenta QEA, é mapeada colocando o atributo identificador da classe referenciada na classe que o referência, criando assim o conceito de uma chave estrangeira no modelo relacional.

2.7 SISTEMA ATUAL

Atualmente a ferramenta Quellon não faz integração com ferramenta alguma de modelagem de software. O analista que está desenvolvendo seu produto com a ferramenta

Quellon disponibiliza o Diagrama de Classe e o desenvolvedor precisa analisar o diagrama e criar a tabela manualmente no sistema, utilizando a interface da Quellon, o que exige muito tempo, além de possibilitar erros. Para se criar uma nova tabela em sistema que utiliza Quellon, o usuário precisa acessar o sistema, ir ao módulo Segurança, clicar no botão Novo, digitar no campo Nome o nome que deseja que sua tabela tenha e clicar em Salvar, conforme mostrado na Figura 4. Esse processo exige muito tempo, além de possibilitar que o programador digite errado o nome da tabela.

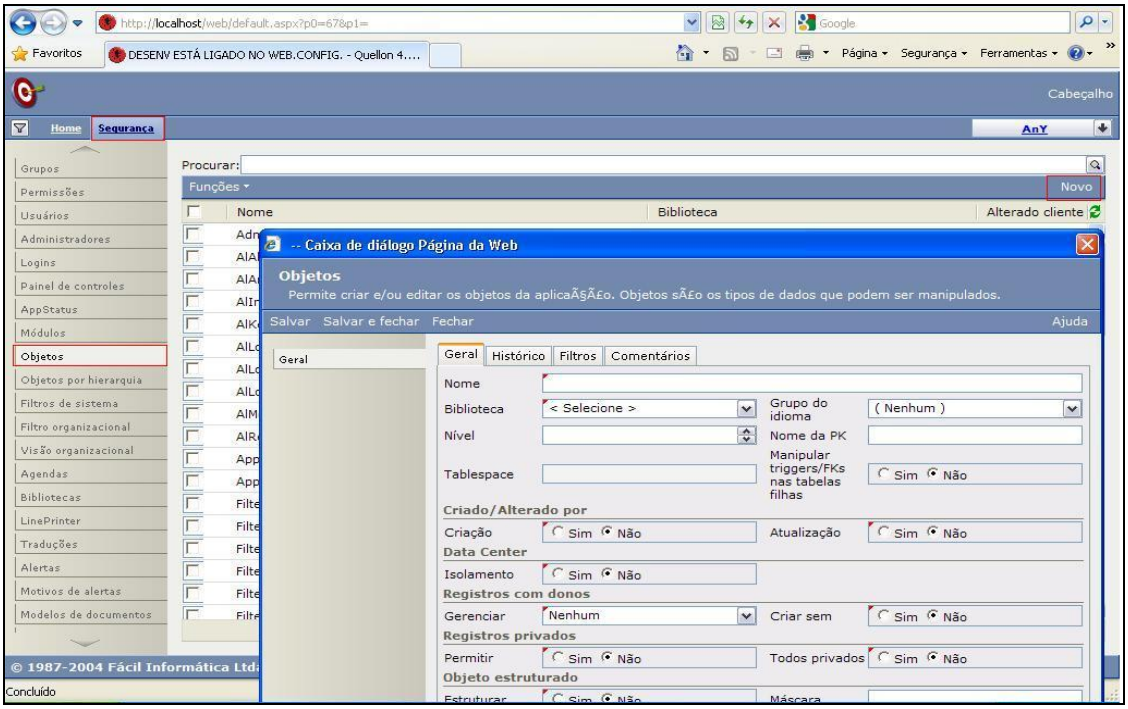


Figura 4: Criando uma nova tabela em um sistema que utiliza Quellon

Após a criação da tabela, abaixo da *tab* Geral é criada a *tab* Membros, onde é possível criar os campos que pertencem àquela tabela. Para criar um campo é necessário clicar na *tab* membros, clicar em Novo, selecionar o tipo de campo que deseja criar, conforme a Figura 5. Após isso, será apresentada uma nova tela para criar o membro conforme o tipo selecionado. Existem campos comuns em todos os tipos de membro, como por exemplo: Nome, Legenda, Visibilidade. Porém, existem campos específicos para cada tipo de membro, por exemplo, para campos do tipo *string*, deve-se dizer o tamanho, uma máscara (se quiser), entre outros, conforme Figura 6.

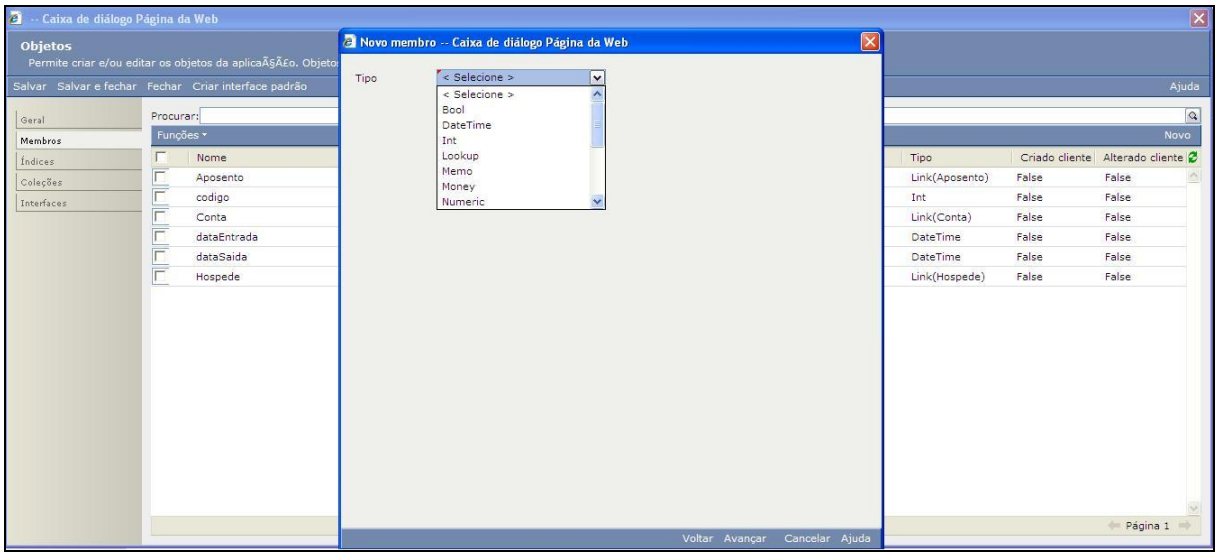


Figura 5: Selecionando o tipo do membro que será criado

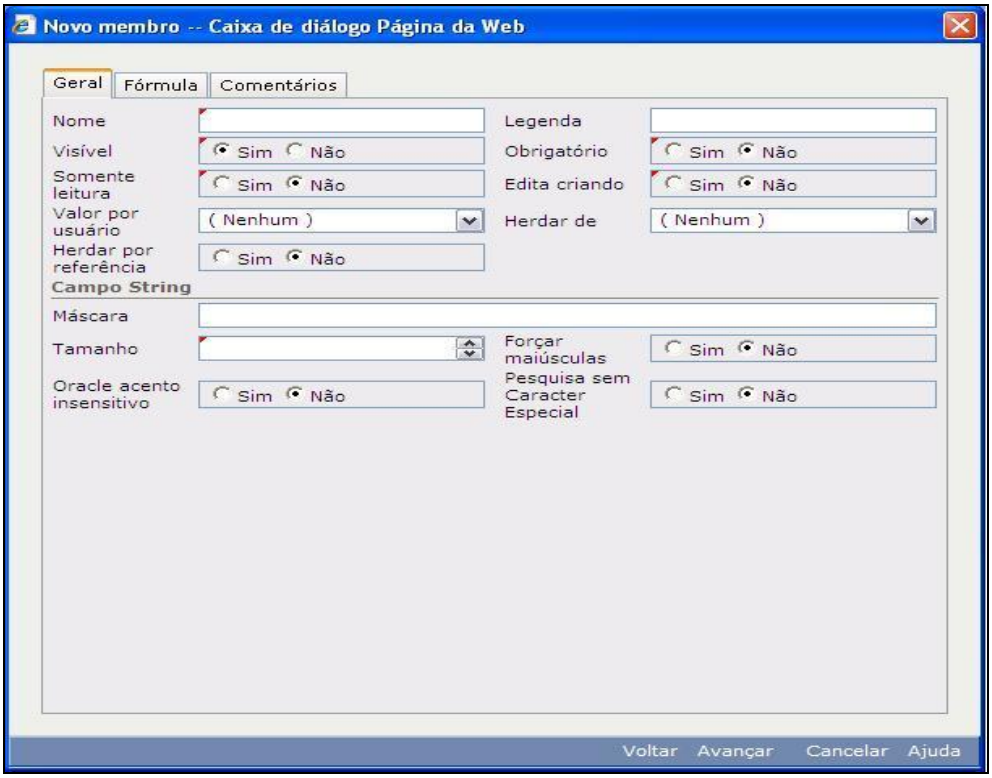


Figura 6: Criando um membro do tipo *string*

Na Figura 7 o sistema atual é representado em um diagrama de atividades. Após o analista do sistema disponibilizar o diagrama de classes para o programador, o programador precisa analisar o diagrama de classes e mapear cada tabela dentro do sistema. Com a tabela criada é preciso criar os membros dentro de cada tabela e assim ter sua base de dados pronta.

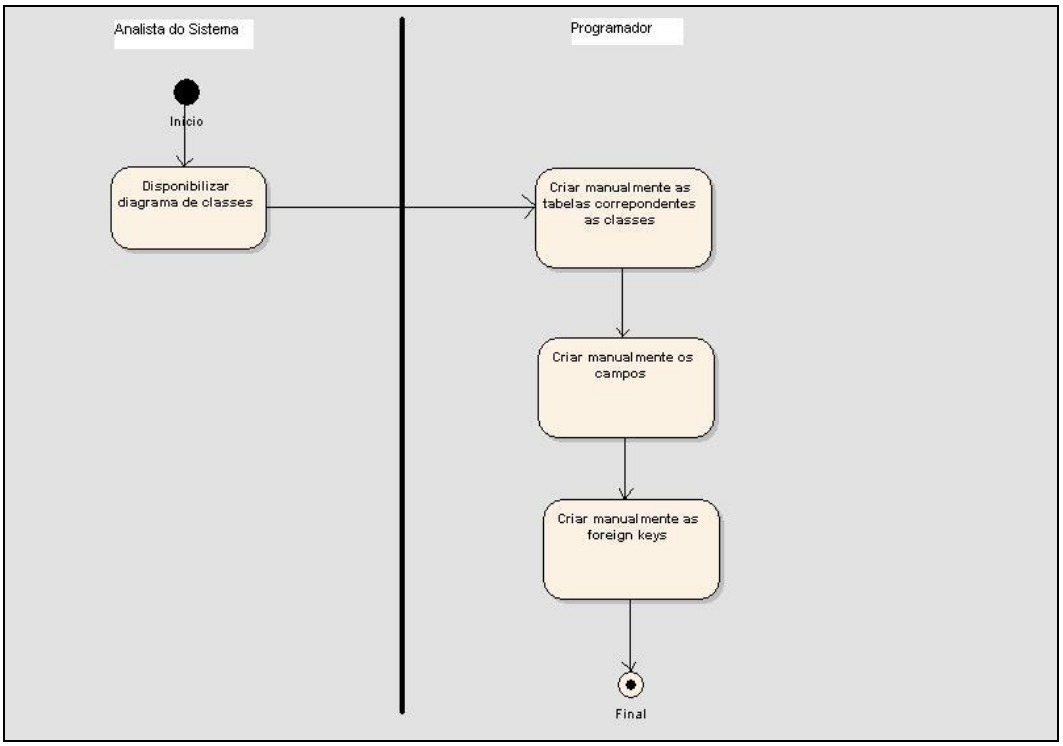


Figura 7: Diagrama de Atividades do sistema atual

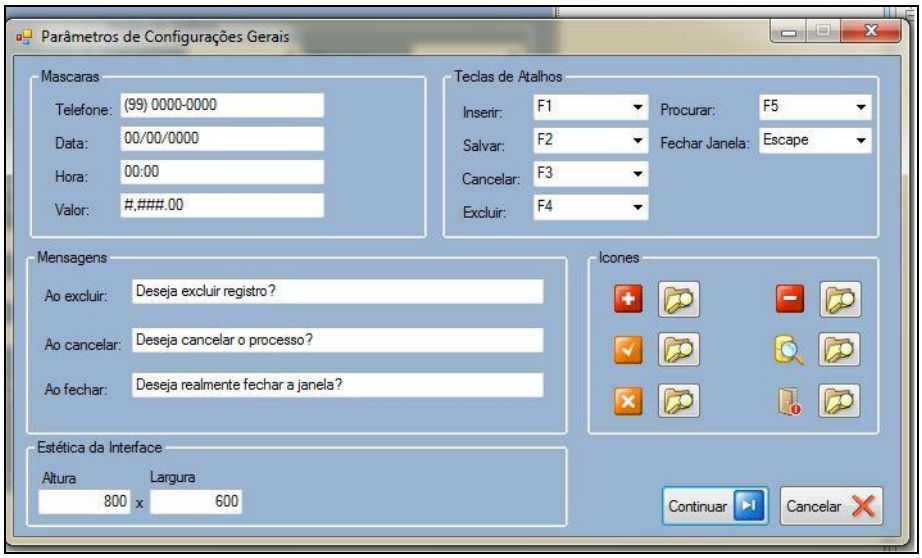
2.8 TRABALHOS CORRELATOS

Batista (2007) desenvolveu como Trabalho de Conclusão de Curso (TCC), uma ferramenta de gerência de requisitos de software integrada ao EA. Essa ferramenta tinha como objetivo disponibilizar uma extensão da ferramenta CASE EA para especificação e gerenciamento de requisitos. Tinha como objetivo também, disponibilizar através da ferramenta uma documentação de requisitos baseado no modelo IEEE-830-1998. Na Figura 8 tem-se a tela onde o sistema de Batista (2007) verifica os alertas.

Fonte: Batista (2007).

Figura 8: Tela do TCC de Raphael Marcos Batista

Já Becker (2009) desenvolveu como TCC uma ferramenta para construção de interfaces de software a partir de diagrama de classes. Essa ferramenta tinha como objetivo construir interfaces conforme critérios ergonômicos de usabilidade a partir de diagrama de classes. Tinha como objetivo também, obter informações do diagrama de classes gerado pela ferramenta CASE EA, traduzir as informações obtidas para a linguagem XML, filtrar o código gerado e adaptá-lo para gerar interfaces gráficas para aplicações *standalone*, conforme critérios ergonômicos de usabilidade e permitir configurar valores para atender os critérios ergonômicos de usabilidade. A Figura 9 mostra uma tela da ferramenta criada por Becker (2009).



Fonte: Becker (2009).

Figura 9: Tela do TCC de André Luis Becker

3 DESENVOLVIMENTO DA FERRAMENTA

Este capítulo apresenta as seções sobre o levantamento de informações do sistema, detalhes sobre a especificação e a modelagem desenvolvida. Na seção de implementação do sistema tem-se as técnicas e ferramentas utilizadas e demonstra-se a operacionalidade do sistema através de um estudo de caso, apresentando-se ao final a seção com os resultados e discussões.

3.1 LEVANTAMENTOS DE INFORMAÇÕES

Após o EA ser escolhido como a ferramenta que seria integrada ao Quellon, foi definido que seria feita a integração do diagrama de classes, onde as classes serão as tabelas, os atributos serão os campos e as ligações serão as *foreign Keys*, e que o processo de importação dos dados deveria ser feito através da funcionalidade de importação de XML.

A ferramenta QEA deverá ser responsável por criar a estrutura física de banco de dados necessária, que estará exposta no diagrama de classes, criando e excluindo campos e tabelas, de acordo com o diagrama que estará sendo importado. O processo de integração entre as duas ferramentas acontecerá em forma de comparação das duas estruturas. O sistema irá manter na base de dados apenas as classes e atributos que estiverem no diagrama: ao criar uma classe no diagrama e importar o XML para a base de dados, essa classe e seus atributos serão criados fisicamente como tabelas e campos; ao excluir uma classes do diagrama e realizar a importação do XML, a mesma será excluída da base de dados, perdendo assim todos os dados nela contidos. Esse mesmo procedimento acontecerá com os campos. A base de dados do cliente sempre estará atualizada de acordo com o diagrama.

3.2 ESPECIFICAÇÃO

Os Requisitos Funcionais (RF) apresentam as funcionalidades e descrevem o comportamento do sistema. Os Requisitos Não Funcionais (RNF) descrevem as restrições que se

aplicam a usabilidade, portabilidade e segurança do sistema.

3.2.1 Requisitos Funcionais

O Quadro 1 apresenta os requisitos funcionais previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o caso de uso associado.

Requisitos Funcionais	**Caso de Uso**
RF01: A ferramenta deverá permitir o cadastro de sistemas	UC.001
RF02: O sistema deverá permitir a importação do XML exportado pelo EA	UC.002
RF03: O sistema deverá gerar a estrutura física necessária dentro do banco de dados.	UC.002
RF04: O sistema deverá emitir um documento informando quais tabelas foram criadas no banco de dados	UC.002
RF05: Em caso de erro na importação deste XML, o sistema deverá gerar um documento, contendo um *log* de erro	UC.002

Quadro 1: Requisitos funcionais

3.2.2 Requisitos Não Funcionais

O Quadro 2 lista os requisitos não funcionais previstos para o sistema.

Requisitos Não Funcionais
RNF01: O XML a ser lido pelo Quellon deverá estar utilizando o padrão XMI
RNF02: O sistema deverá ser construído utilizando a linguagem C#

Quadro 2: Requisitos não funcionais

3.2.3 Casos de Uso

Esta subseção apresenta o diagrama de caso de uso da ferramenta. O sistema possui 2 casos de uso:

a) UC.001 – cadastrar sistema;
b) UC.002 - gerar estrutura.

Para melhor entendimento do projeto, o detalhamento dos casos de uso se encontra no Apêndice A. A Figura 10 ilustra o diagrama de caso de uso.

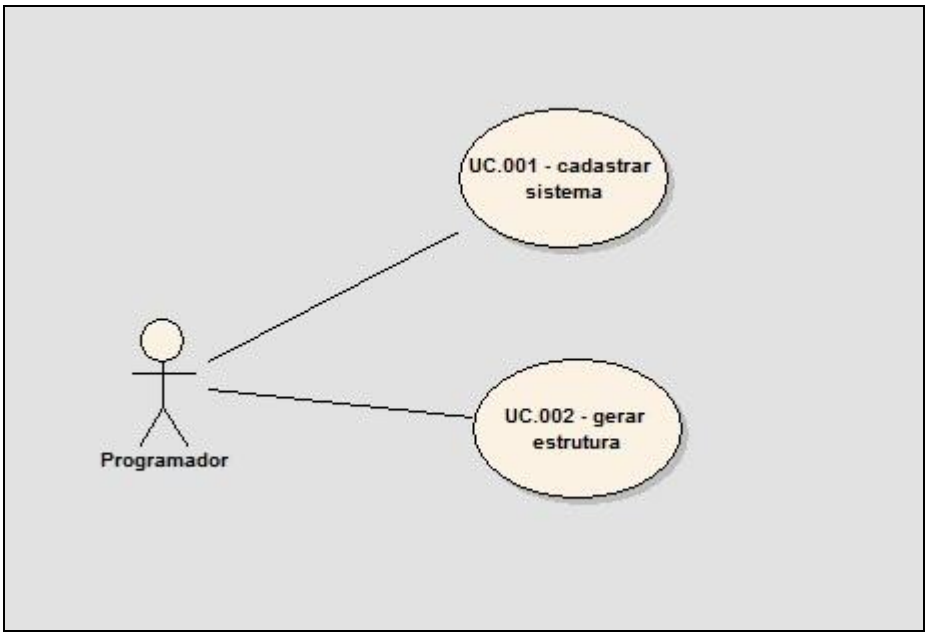


Figura 10: Diagrama de caso de uso

3.3 IMPLEMENTAÇÃO

Nesta seção serão apresentadas informações sobre a implementação da ferramenta, as técnicas e ferramentas utilizadas e a sua operacionalidade.

3.3.1 Técnicas utilizadas

A ferramenta QEA foi desenvolvida em linguagem C#/ASP.NET, sobre a plataforma Microsoft .Net, com versão 4.0 de *framework*. Para codificação foi utilizado o Ambiente de Desenvolvimento Integrado (IDE) Microsoft Visual Studio 2010.

O XML a ser exportado do EA deverá estar no formato UML 2.1 (XMI 2.1), conforme indicado na Figura 11.

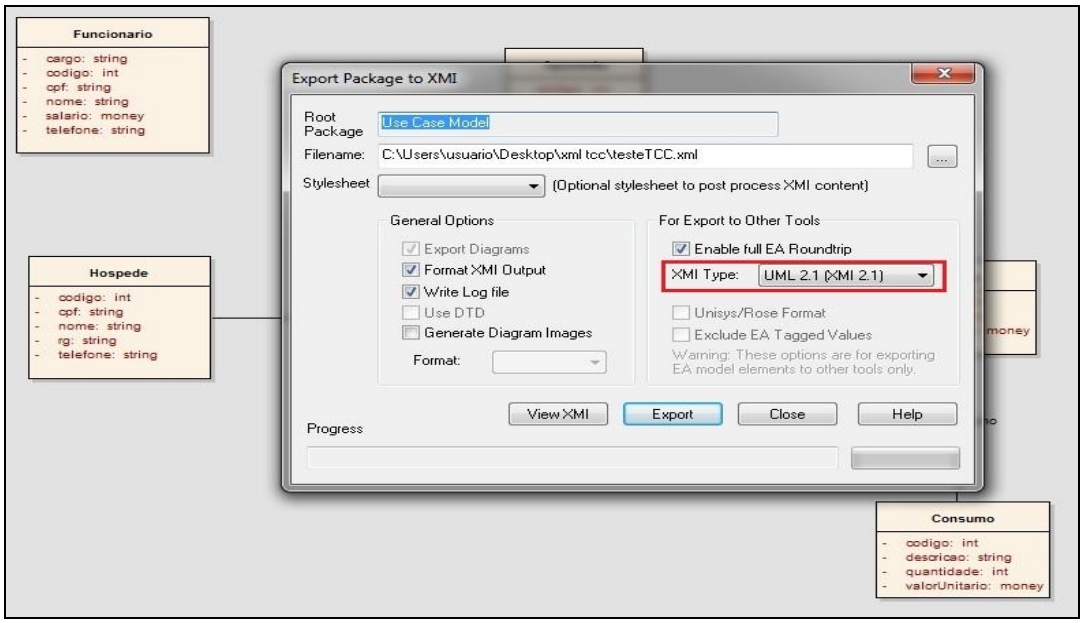


Figura 11: Exportação do XML do EA

A Figura 12 ilustra um XML de um diagrama exportado do EA.

```
<packagedElement xmi:type="uml:Class" xmi:id="EAID_1E732646_39F0_47a7_BD1E_DD06753B4099" name="Consumo" visibility="public">
- <ownedAttribute xmi:type="uml:Property" xmi:id="EAID_7B4B0082_AE0A_4ce4_A557_FAA51121F016" name="codigo" visibility="private" isDerived="false">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000009_AE0A_4ce4_A557_FAA51121F016" value="1" />
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000010_AE0A_4ce4_A557_FAA51121F016" value="1" />
  </ownedAttribute>
- <ownedAttribute xmi:type="uml:Property" xmi:id="EAID_BEE47EAD_DE9B_4fac_BDA6_F55788AE6F78" name="descricao" visibility="private" isDerived="false">
    <type xmi:idref="EAID_8BE1915A_B562_4f90_AFBA_129BE1DC09F5" />
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000011_DE9B_4fac_BDA6_F55788AE6F78" value="1" />
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000012_DE9B_4fac_BDA6_F55788AE6F78" value="1" />
  </ownedAttribute>
- <ownedAttribute xmi:type="uml:Property" xmi:id="EAID_64AB1269_BFB2_4f19_89B2_CFDB86F3112A" name="quantidade" visibility="private" isDerived="false">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000013_BFB2_4f19_89B2_CFDB86F3112A" value="1" />
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000014_BFB2_4f19_89B2_CFDB86F3112A" value="1" />
  </ownedAttribute>
- <ownedAttribute xmi:type="uml:Property" xmi:id="EAID_F3DCAE2B_BC51_4f9f_9CEC_C969256926DE" name="valorUnitario" visibility="private" isDerived="false">
    <lowerValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000015_BC51_4f9f_9CEC_C969256926DE" value="1" />
    <upperValue xmi:type="uml:LiteralInteger" xmi:id="EAID_LI000016_BC51_4f9f_9CEC_C969256926DE" value="1" />
  </ownedAttribute>
</packagedElement>
<packagedElement xmi:type="uml:Association" xmi:id="EAID_8415B9B9_1069_408c_AAC5_446273BE285F" name="id_Consumo" visibility="public">
  <memberEnd xmi:idref="EAID_dst15B9B9_1069_408c_AAC5_446273BE285F" />
  <memberEnd xmi:idref="EAID_src15B9B9_1069_408c_AAC5_446273BE285F" />
- <ownedEnd xmi:type="uml:Property" xmi:id="EAID_src15B9B9_1069_408c_AAC5_446273BE285F" visibility="public"
    association="EAID_8415B9B9_1069_408c_AAC5_446273BE285F" isOrdered="false" isDerived="false" isDerivedUnion="false" aggregation="none">
    <type xmi:idref="EAID_A7AA8262_D512_49bd_89AE_2C24F14C9371" />
  </ownedEnd>
```

Figura 12: Parte do XML do diagrama

Para leitura do XML exportado pelo EA, utiliza-se a biblioteca *System.XML* da própria ferramenta de desenvolvimento. Foi desenvolvido um protótipo (conforme código ilustrado na Figura 13), o qual implementa uma varredura do XML do EA, onde é feita a primeira leitura das tabelas, e colocado todos os IDs em um dicionário.

```
Dictionary<string, string> tabelasComIds = new Dictionary<string, string>();

XmlDocument xml = new XmlDocument();
//abrir um arquivo xml
xml.Load("C:\\Documents and Settings\\Administrador\\Desktop\\xmlExportado.xml");

//buscar todos os nodos que sejam do tipo element (no xml a classe fica dentro desse nodo tipo element)
XmlNodeList listaNodos = xml.GetElementsByTagName("element");

//percorrer todos os nodos dessa lista de nodos elements, procurando pelo nodo que seja do tipo classe
foreach (XmlNode nodo in listaNodos)
{
    //elementos que são tipo classe tem o atributo type
    if (nodo.Attributes["xmi:type"] != null)
    {
        //verificar se o nodo da vez é uma classe (que será uma tabela)
        if (nodo.Attributes["xmi:type"].Value == "uml:Class")
        {
            //escreve o nome da tabela na tela, na ferramenta será o comando de create table
            System.Console.WriteLine(string.Format("Nome da classe: {0}", nodo.Attributes["name"].Value));

            //coloca a tabela no dicionário:
            tabelasComIds.Add(nodo.Attributes["xmi:idref"].Value, nodo.Attributes["name"].Value);
        }
    }
}
```

Figura 13: Leitura das tabelas do XML do EA

Com o nome das novas tabelas identificadas, em primeiro lugar, todas elas devem ser criadas. Esse processo foi utilizado para que não ocorram erros ao criar tabelas que possuem *foreign keys* apontam para tabelas que ainda não existem na base de dados. Para criação das tabelas no banco de dados foi utilizada a DLL da Quellon, chamada de Quellon.Update, que possui o método Update, no qual é feita a comparação do XML enviado por parâmetro com a base de dados atual, criando as tabelas que ainda não existem na base de dados e excluindo tabelas que não existem no XML exportado. A própria DLL verifica se o banco de dados utilizado por aquele sistema é Oracle ou Microsoft SQL Server, montando assim, o comando SQL de acordo com o banco de dados do sistema que está sendo atualizado. Na Figura 14 pode-se verificar a chamada desta DLL.

```
string fileName = frm.FileName;
StreamReader sr = new StreamReader(fileName);
string arquivoXmlExportado = sr.ReadToEnd();
arquivoXmlExportado = AlteraXmlEAParaXmlQuellonCriandoTabelas(arquivoXmlExportado);
sr.Close();
m_Update = new Quellon.Update.Update(arquivoXmlExportado, SystemObjects, sysInfo.Caller, "");
```

Figura 14: Chamada da DLL Quellon.Update para criação de tabelas

Com todas as tabelas criadas no banco de dados, deve-se ler o valor dos campos destas tabelas. A Figura 15 mostra o código utilizado para leitura dos campos.

```
//verificar se a classe tem atributos, que serão os campos da tabela
if (nodo.HasChildNodes)
{
    //os atributos propriamente ditos ficam dentro do node "attributes" do xml, ele vai ter varios "attribute"
    //e cada um deles é um atributo, ou seja um campo
    XmlNode atributos = nodo.SelectSingleNode("attributes");

    foreach (XmlNode atributo in atributos.ChildNodes)
    {

        string membro = atributo.Attributes["name"].Value;
        string tipo = atributo.SelectSingleNode("properties").Attributes["type"].Value;
```

Figura 15: Leitura dos campos

A criação dos campos no banco de dados é feita da mesma forma que a criação das tabelas, chamando a DLL Quellon.Update, conforme Figura 16.

```
string fileName = frm.FileName;
StreamReader sr = new StreamReader(fileName);
string arquivoXmlExportado = sr.ReadToEnd();
arquivoXmlExportado = AlteraXmlEAParaXmlQuellonCriandoCampos(arquivoXmlExportado);
sr.Close();
m_Update = new Quellon.Update.Update(arquivoXmlExportado, SystemObjects, sysInfo.Caller, "");
```

Figura 16: Chamada da DLL Quellon.Update para criação de campos

Ao criar um diagrama de classes para ser importado na ferramenta QEA, os campos cadastrados nesse diagrama devem respeitar a nomenclatura dos tipos de campos da Quellon. Estes tipos de campos dentro do EA são vistos como tipos de dados (*datatypes*) de um produto ou uma linguagem de programação. A Figura 17 mostra como cadastrar os tipos de dados dentro do EA, no qual deve-se primeiramente adicionar um novo produto (clicando em *Add Product*), em seguida clicar-se no botão *New*, quando deve ser informado o tipo do dado. Para finalizar é preciso clicar no botão *Save*. Também pode-se criar novos tipos de dados em um produto já existente.

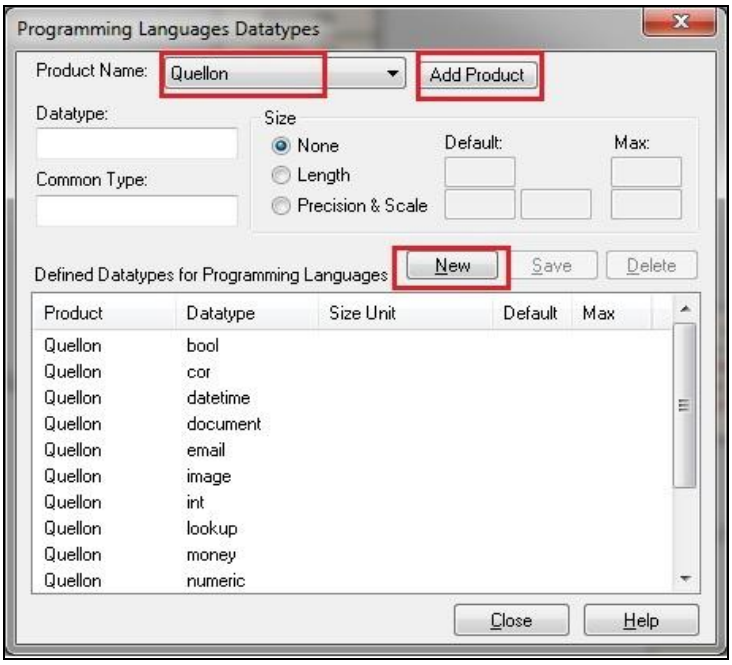


Figura 17: Criando os tipos de campos da Quellon

Ao criar-se uma classe, é preciso selecionar o tipo de linguagem a ser utilizada. Serão apresentados os tipos de dados já cadastrados anteriormente, conforme mostrados nas Figuras 18 e 19.

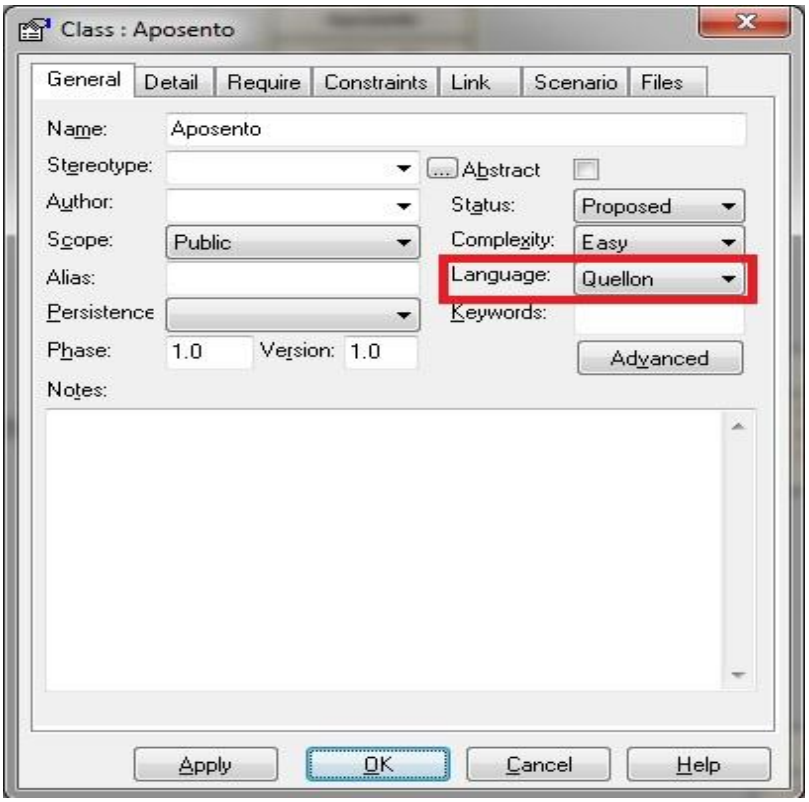


Figura 18: Linguagem Quellon

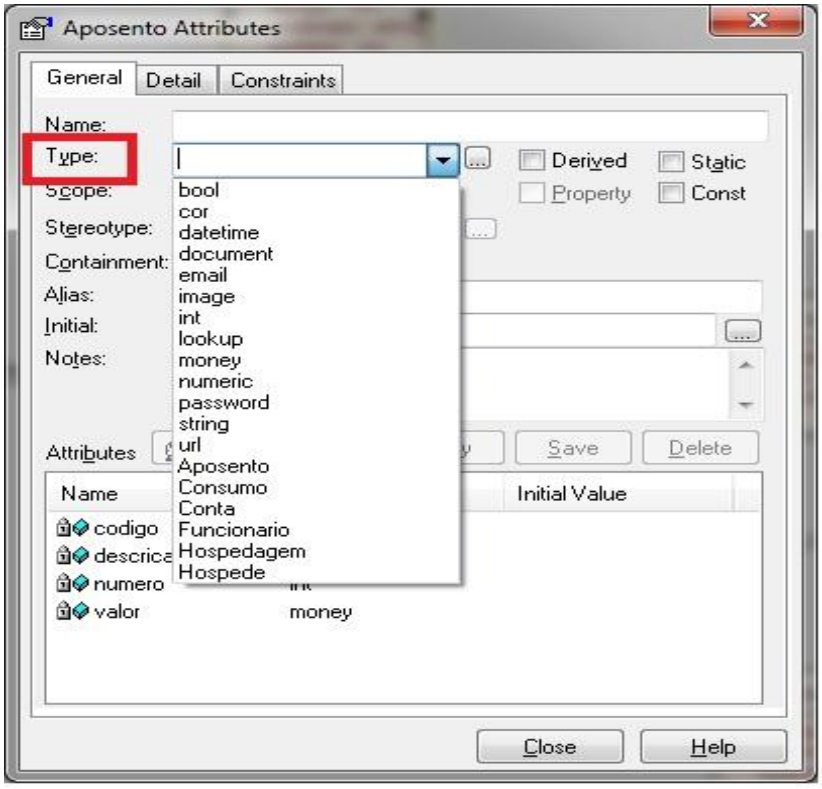


Figura 19: Tipos de dados Quellon

Uma chave estrangeira (FK) é uma coluna ou combinação de colunas que é usada para estabelecer ou impor uma relação entre os registros de duas tabelas (MSDN, 2011c). Destaca-se na Figura 20 o código utilizado para leitura das FKs dentro deste XML.

```
foreach (XmlNode nodoConnector in listaConectores)
{
    XmlNode conetores = nodoConnector.SelectSingleNode("connector");
    if (conetores != null)
    {
        foreach (XmlNode conector in conetores)
        {
            if (conector.Name == "target")
            {
                if (conector.SelectSingleNode("type") != null)
                {
                    XmlNode tipos = conector.SelectSingleNode("type");
                    string multiplicidade = tipos.Attributes["multiplicity"].Value;
                    if(multiplicidade.Equals("1"))
                    {
                        idTabelaDestino = conector.Attributes["xmi:idref"].Value;
                        nomeTabelaDestino = tabemasComIds[idTabelaDestino];
                    }
                    if (multiplicidade.Equals("*"))
                    {
                        idTabelaOrigem = conector.Attributes["xmi:idref"].Value;
                        nomeTabelaOrigem = tabemasComIds[idTabelaOrigem];
                    }
                }
            }
```

Figura 20: Leitura das FKs dentro do XML do EA

Para que a ferramenta QEA possa localizar a ligação entre duas classes no diagrama, esta deve ser feita através de uma associação (*Associate*). Por questões de escopo, limitou-se a localização e criação de apenas ligações com multiplicidade um para muitos, onde o campo de ligação entre as tabelas será criado na tabela que possui multiplicidade *. Para melhor visualização dos campos, pode-se também utilizar campos *lookup* dentro das classes, para que o usuário visualize melhor quais são todos os campos que pertencem àquela classe, mas isso será utilizado apenas para visualização, pois a ferramenta QEA não criará campos *lookup* definidos pelo usuário. Os campos *lookups* sempre serão criados através das ligações entre duas tabelas. O nome do campo que será criado será sempre o nome da tabela para qual ele aponta.

3.3.2 Ferramentas utilizadas

A seguir são listadas e detalhadas as ferramentas utilizadas para o desenvolvimento da ferramenta QEA.

3.3.2.1 C# na plataforma .NET

Para o desenvolvimento da ferramenta QEA, foi escolhida a linguagem C# na plataforma .NET. De acordo com Lippman (2003, p.7), C# é uma linguagem criada pela Microsoft e introduzida com o Visual Studio.NET. Para Haddad (2001, p.3), ela é uma linguagem que abrange o poder e a versatilidade do Visual Basic, a força e a criatividade do C++ e a facilidade do Jscript, trazendo como vantagens os recursos do ambiente Windows, a validação dos dados, os tratamentos de erros, não precisar registrar componentes, a manipulação de banco de dados não conectados, entre outros.

O MSDN (2011d) afirma que, C# é uma linguagem orientada o objeto, elegante e fortemente tipada, o que permite aos desenvolvedores criarem uma variedade de aplicativos seguros e robustos que são executados no .NET Framework. Pode ser utilizada para criar aplicações tradicionais no Windows, *Web Services* baseados em XML, componentes distribuídos, aplicativos servidor-cliente, aplicativo de banco de dados, entre outros. A sintaxe do C# é altamente expressiva, mas também é simples e fácil de aprender. Como uma linguagem orientada a objetos, ela suporta os conceitos de encapsulamento, herança e

polimorfismo.

3.3.2.2 Plataforma .NET

O .NET é uma plataforma de desenvolvimento para *web* que inclui diversas ferramentas, como por exemplo, o SQL Server para banco de dados e o Visual Studio.NET para desenvolvimento (HADDAD, 2001, p. 2). Segundo o MSDN (2011b), o .NET Framework é um ambiente para desenvolvimento e execução que permite o funcionamento conjunto de linguagens de programação e bibliotecas diferentes, tendo em vista a criação de aplicativos para Windows, *web*, dispositivos móveis e Office. É um componente essencial do Windows que oferece suporte à criação e execução da próxima geração de aplicativos e serviços XML da *web*.

O .Net Framework foi projetado para atingir os seguintes objetivos:

a) fornecer um ambiente de programação orientado a objetos consistente, quer o código seja armazenado e executado localmente, seja executado localmente, mas distribuído pela internet ou seja executado remotamente;
b) fornecer um ambiente de execução que minimize conflitos de versionamento de publicação;
c) fornecer um ambiente de execução que promova a execução segura de código criado por desconhecidos ou código de terceiros com baixo nível de segurança;
d) fornecer um ambiente de execução que elimine os problemas de desempenho dos ambientes interpretados ou com scripts;
e) para tornar a experiência do desenvolvedor consistente, através de diversos tipos de aplicativos baseados no Windows e aplicativos baseados na *web*.

3.3.2.3 Visual Studio

O Visual Studio.NET é a IDE no qual os desenvolvedores trabalham para criar programas para a plataforma .NET (MSDN, 2011a). Segundo Camara (2005, p. 10), o Visual Studio .NET é usado como ferramenta de desenvolvimento para desenhar sistemas que podem ser orientados a objetos e utilizam-se de formulários, componentes e páginas para internet. A idéia proposta é utilizar a ferramenta de desenvolvimento como um facilitador para a

utilização de objetos e formulários. A ferramenta trabalha com qualquer linguagem que esteja dentro dos requisitos da *Commom Language Specification* (CLS). Na Figura 21 é ilustrada uma das visões da IDE Visual Studio 2010.

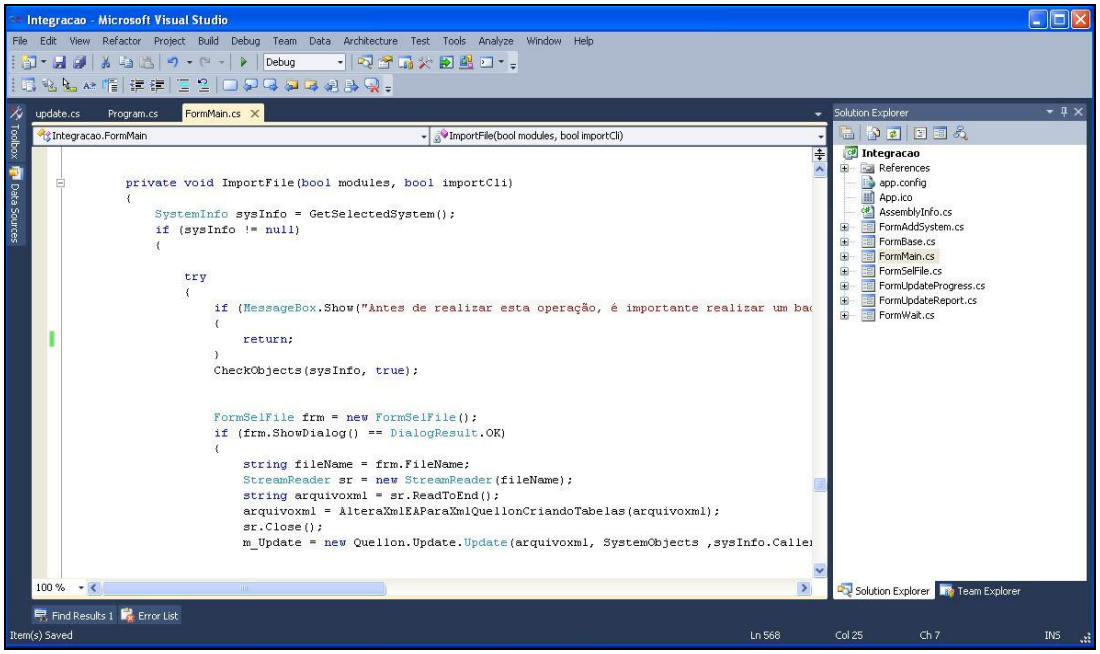


Figura 21: Visual Studio

3.3.3 Operacionalidade da ferramenta

Esta subseção apresenta as telas da ferramenta QEA e suas funcionalidades. A Figura 22 ilustra um diagrama de atividades descrevendo os passos para funcionamento da ferramenta QEA.

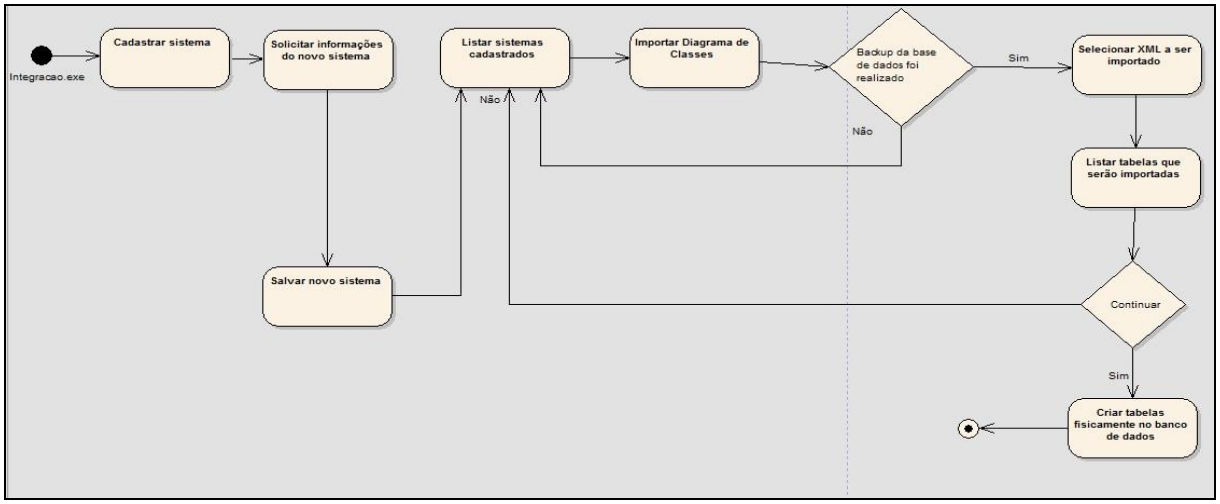


Figura 22: Diagrama de Atividades do funcionamento da ferramenta QEA

3.3.3.1 Instalação do executável

A ferramenta QEA é disponibilizada para os clientes da Quellon em um pacote com todos os arquivos necessários para o seu funcionamento. O primeiro passo para o uso da ferramenta será executar o arquivo chamado `Integracao.exe` que se encontra dentro deste pacote, conforme exibido na Figura 23.

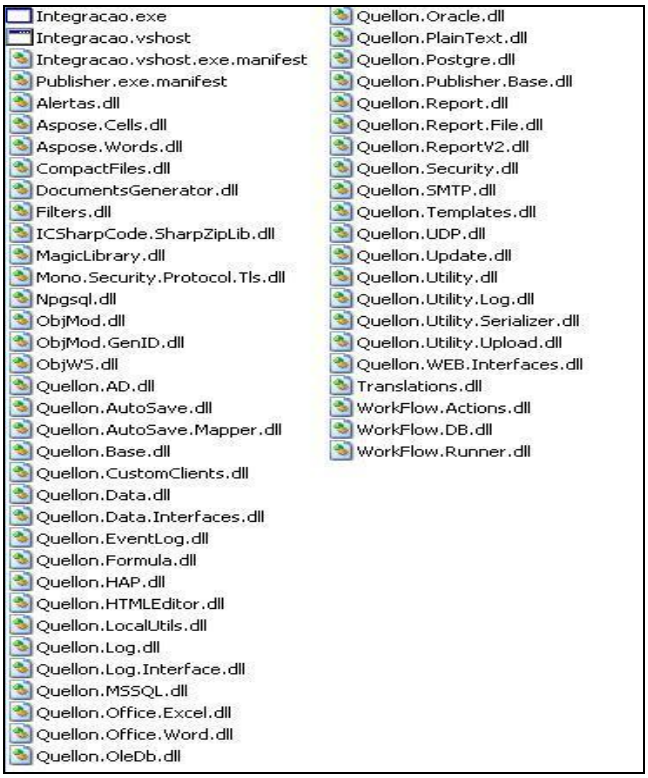


Figura 23: Pasta com os arquivos da ferramenta de integração

3.3.3.2 Adição do sistema

Após iniciar-se a ferramenta QEA, surge a sua tela inicial (Figura 24), na qual é preciso selecionar apenas o botão “Adicionar Sistema”, para cadastrar o sistema no qual deseja que os dados sejam importados.

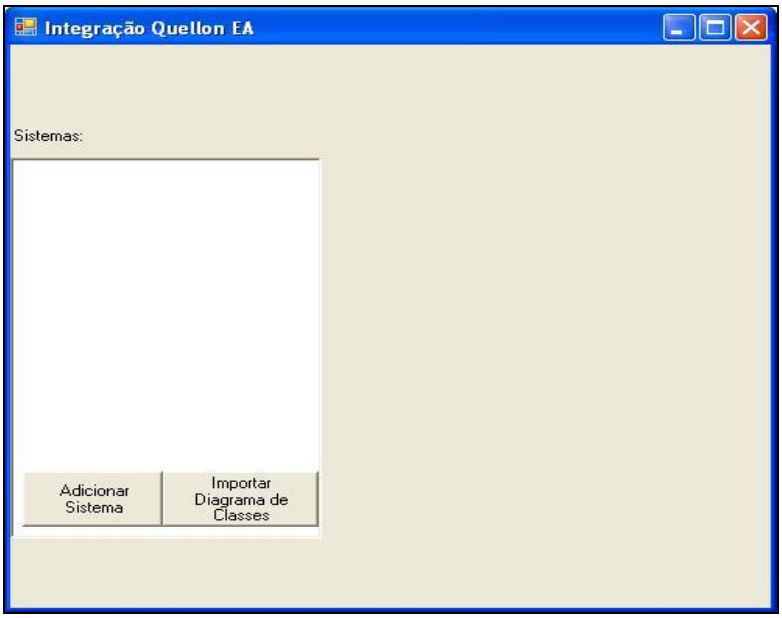


Figura 24: Tela inicial da ferramenta

Será então apresentada uma tela na qual deverá ser cadastrado o nome do sistema, o endereço de acesso, o usuário e senha para conectar-se ao sistema cadastrado, conforme demonstra a Figura 25.

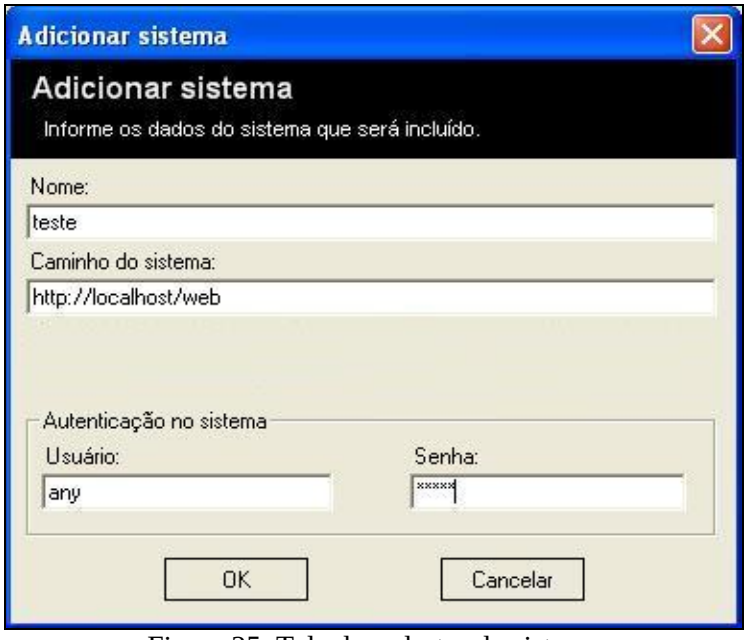


Figura 25: Tela de cadastro do sistema

Ao clicar no botão “OK”, a ferramenta validará se o sistema informado existe e se o usuário e senha são válidos. Caso uma dessas validações não ocorra, é apresentada uma tela informando o erro, conforme exemplificado na Figura 26.

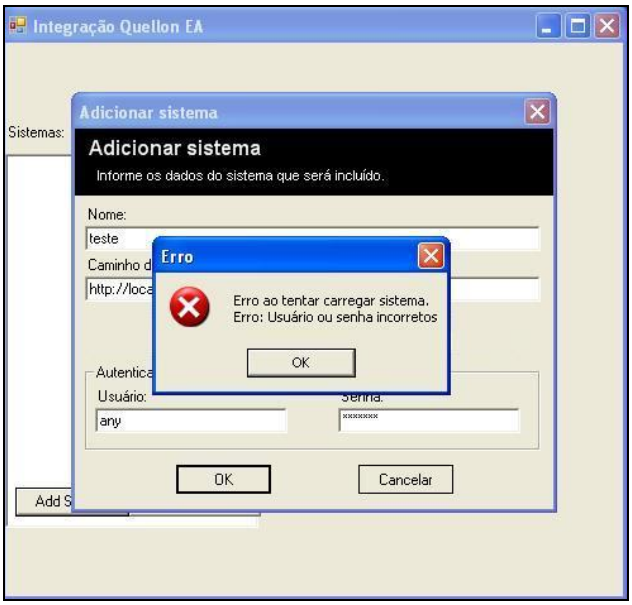


Figura 26: Erro ao cadastrar sistema

3.3.3.3 Importação dos dados

Após o cadastramento do sistema ser completado, um registro será adicionado na lista de sistemas, conforme ilustrado na Figura 27. Para realizar a importação de dados, é preciso clicar o nome do sistema e selecionar a opção “Importar Diagrama de Classes”.

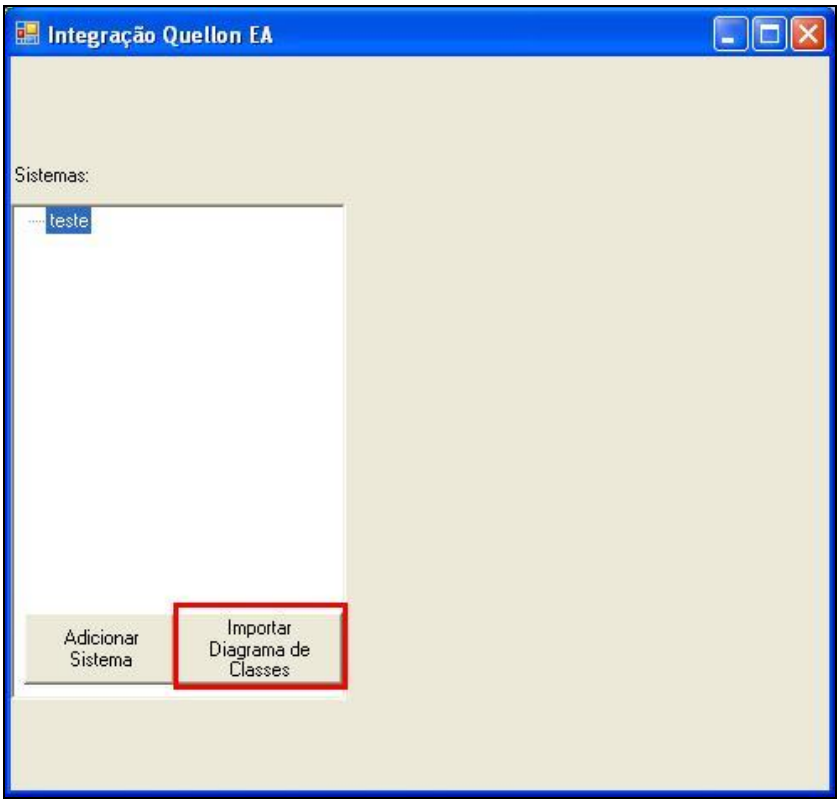


Figura 27: Botão para importação do diagrama de classes

Como próximo passo, é preciso selecionar o caminho do arquivo que deverá ser importado pela ferramenta, conforme Figura 28.

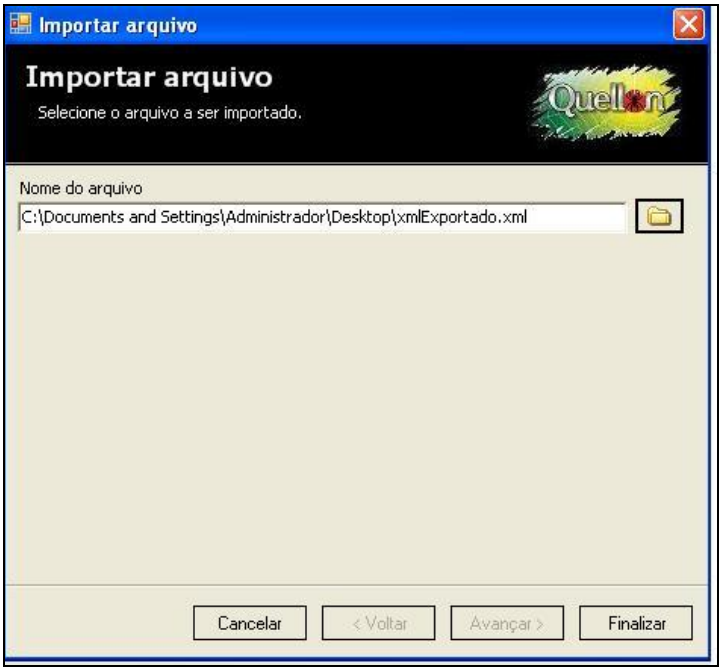


Figura 28: Importar Arquivo

Ao clicar no botão “Finalizar”, a ferramenta apresentará uma tela onde será possível visualizar o nome de todas as tabelas que estão no XML a ser importado, onde é possível cancelar a importação, clicando em Cancelar, ou inicializar a importação, clicando em OK, conforme ilustrado na Figura 29.

Figura 29: Tabelas que estão no XML que será importado

Clicando em OK, a ferramenta inicializará a leitura do XML e a geração da estrutura física dentro do banco de dados. Para o usuário, será mostrada uma tela com o status do andamento da importação, conforme Figura 30.

Figura 30: Status da Importação

No término da importação, a ferramenta exibirá uma mensagem ao usuário, informando que a atualização foi finalizada, conforme Figura 31.

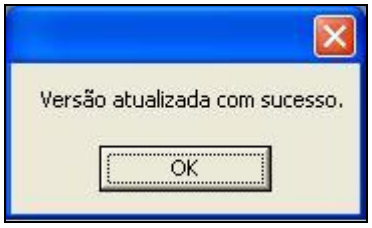


Figura 31: Atualização finalizada

Caso haja erro durante a atualização será também criado um arquivo .txt dentro do pacote que contém a ferramenta de integração. A Figura 32 ilustra o código utilizado para criação deste arquivo.

```
StreamWriter logErro = new StreamWriter(m_diretorio + @"\logErro.txt", true, Encoding.ASCII);
logErro.Write(string.Format("Erro: {0}.", err.Message));
logErro.WriteLine();
logErro.Close();
```

Figura 32: Código *log* de erro

A Figura 33 demonstra um exemplo do arquivo `logErro.txt`.

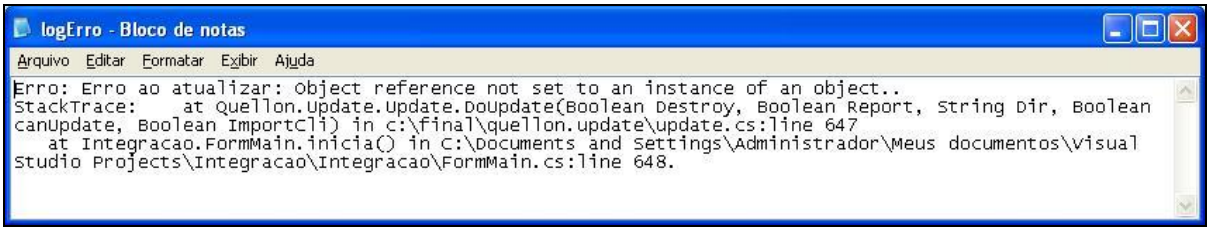


Figura 33: exemplo do arquivo logErro.txt

Após o término da atualização, é possível verificar quais tabelas foram criadas no banco de dados através da análise do arquivo `tabelasCriadas.txt`. A Figura 34 mostra o código utilizado para criação deste arquivo.

```
if (_Obj == null)
{
    string nomeTabelaCriada = _NodeObject.Attributes[ExportConst.Name].Value;

    StreamWriter tabelaCriada = new StreamWriter(m_Diretorio + @"\tabelasCriadas.txt", true, Encoding.ASCII);
    tabelaCriada.Write(string.Format("Tabela criada: {0}. \n",nomeTabelaCriada));
    tabelaCriada.Close();


}
```

Figura 34: Código tabelas criadas

A Figura 35 ilustra um exemplo do arquivo `tabelasCriadas.txt`

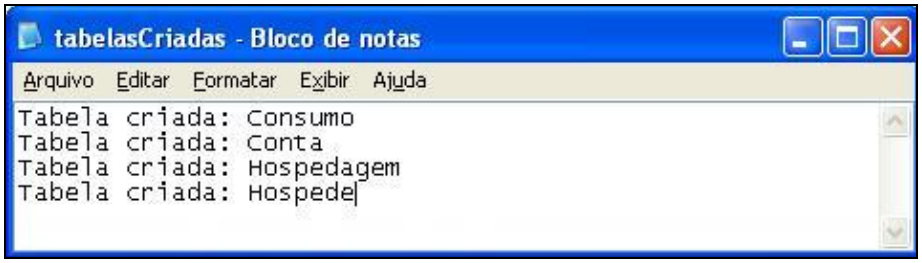
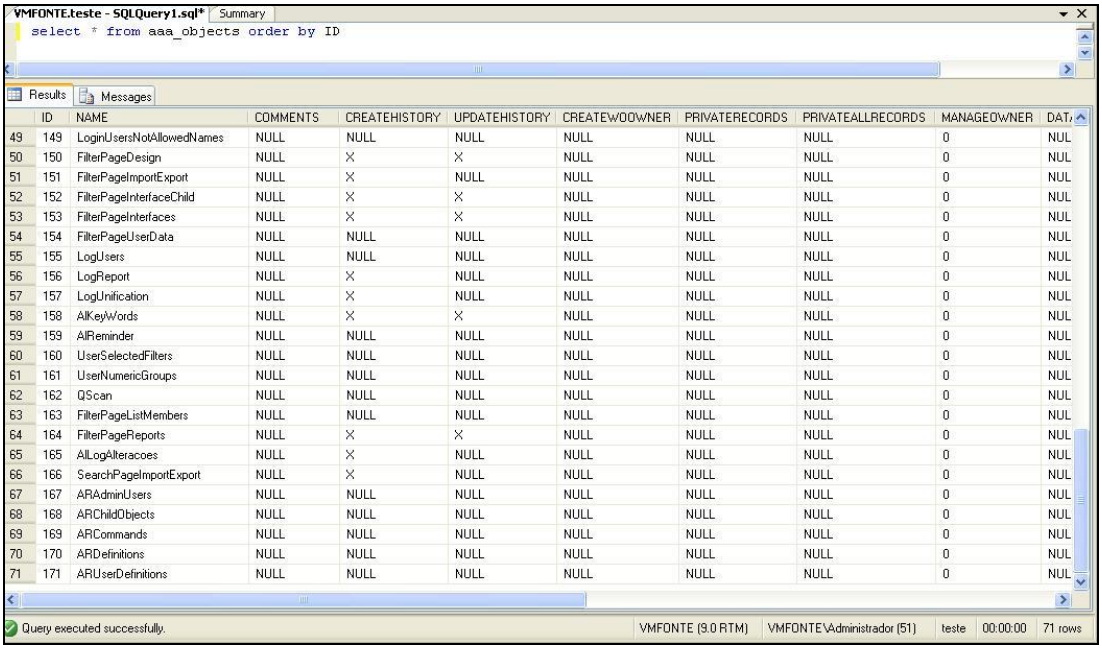


Figura 35: Exemplo do arquivo tabelasCriadas.txt

Para demonstrar o resultado gerado pela ferramenta serão criados dois exemplos. Na Figura 36, verifica-se que existem somente tabelas da ferramenta Quellon na base de dados utilizada.



	ID	NAME	COMMENTS	CREATEHISTORY	UPDATEHISTORY	CREATEWOOWNER	PRIVATERECORDS	PRIVATEALLRECORDS	MANAGEOWNER	DAT
49	149	LoginUsersNotAllowedNames	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
50	150	FilterPageDesign	NULL	X	X	NULL	NULL	NULL	0	NUL
51	151	FilterPageImportExport	NULL	X	NULL	NULL	NULL	NULL	0	NUL
52	152	FilterPageInterfaceChild	NULL	X	X	NULL	NULL	NULL	0	NUL
53	153	FilterPageInterfaces	NULL	X	X	NULL	NULL	NULL	0	NUL
54	154	FilterPageUserData	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
55	155	LogUsers	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
56	156	LogReport	NULL	X	NULL	NULL	NULL	NULL	0	NUL
57	157	LogUnification	NULL	X	NULL	NULL	NULL	NULL	0	NUL
58	158	AllKeyWords	NULL	X	X	NULL	NULL	NULL	0	NUL
59	159	AllReminder	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
60	160	UserSelectedFilters	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
61	161	UserNumericGroups	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
62	162	QScan	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
63	163	FilterPageListMembers	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
64	164	FilterPageReports	NULL	X	X	NULL	NULL	NULL	0	NUL
65	165	AllLogAlteracoes	NULL	X	NULL	NULL	NULL	NULL	0	NUL
66	166	SearchPageImportExport	NULL	X	NULL	NULL	NULL	NULL	0	NUL
67	167	ARAdminUsers	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
68	168	ARChildObjects	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
69	169	ARCommands	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
70	170	ARDefinitions	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL
71	171	ARUserDefinitions	NULL	NULL	NULL	NULL	NULL	NULL	0	NUL

Figura 36: *Select* antes da importação do diagrama de classe

No primeiro exemplo será importado pela ferramenta um diagrama de classes com duas classes. A Figura 37 ilustra o diagrama que será importado pela ferramenta QEA.

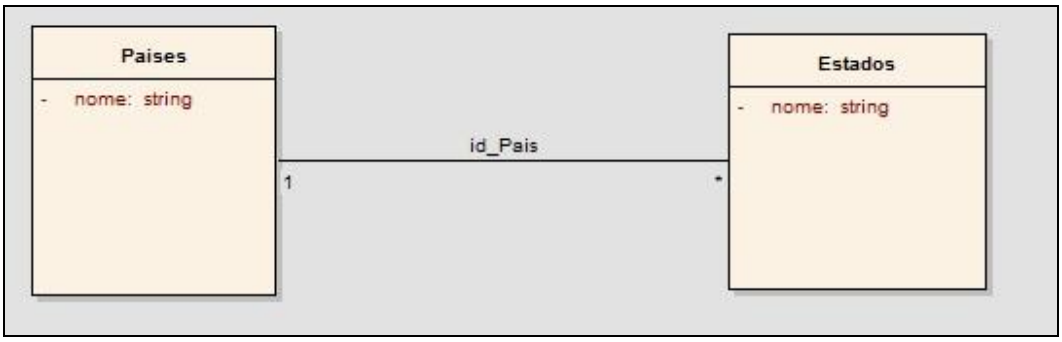
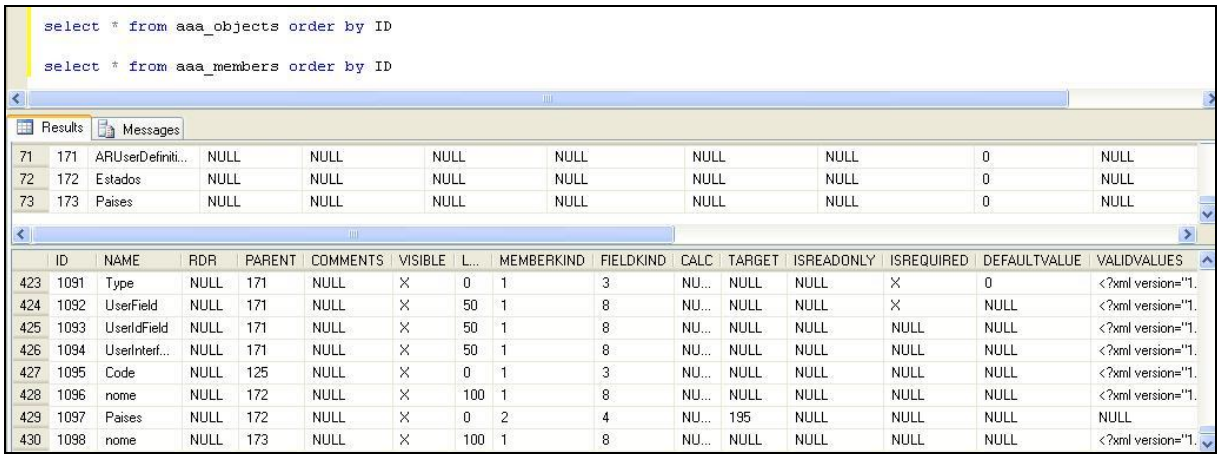


Figura 37: Diagrama de classes com duas classes

Após a importação do XML pela ferramenta QEA, é possível verificar na Figura 38 que as tabelas Paises e Estados foram criadas, juntamente com o campo Paises na tabela de Estados. A ferramenta Quellon cria automaticamente a *primary key* dentro de qualquer tabela criada. Esse campo é do tipo *Int.* e se chama ID, ele é preenchido sempre com o valor mais alto de um ID já existente somado de 1.



```
select * from aaa_objects order by ID

select * from aaa_members order by ID
```

Results | Messages

71	171	ARUserDefiniti...	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
72	172	Estados	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
73	173	Paises	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL

	ID	NAME	RDR	PARENT	COMMENTS	VISIBLE	L...	MEMBERKIND	FIELDKIND	CALC	TARGET	ISREADONLY	ISREQUIRED	DEFAULTVALUE	VALIDVALUES
423	1091	Type	NULL	171	NULL	X	0	1	3	NU...	NULL	NULL	X	0	<?xml version="1.
424	1092	UserField	NULL	171	NULL	X	50	1	8	NU...	NULL	NULL	X	NULL	<?xml version="1.
425	1093	UserIdField	NULL	171	NULL	X	50	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.
426	1094	UserInterf...	NULL	171	NULL	X	50	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.
427	1095	Code	NULL	125	NULL	X	0	1	3	NU...	NULL	NULL	NULL	NULL	<?xml version="1.
428	1096	nome	NULL	172	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.
429	1097	Paises	NULL	172	NULL	X	0	2	4	NU...	195	NULL	NULL	NULL	NULL
430	1098	nome	NULL	173	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.

Figura 38: *Select* após importação do diagrama de classes

No segundo exemplo será importado pela ferramenta QEA um diagrama de classes com 7 classes e diversos atributos e ligações entre as classes. Na Figura 39 verifica-se o diagrama que será importado.

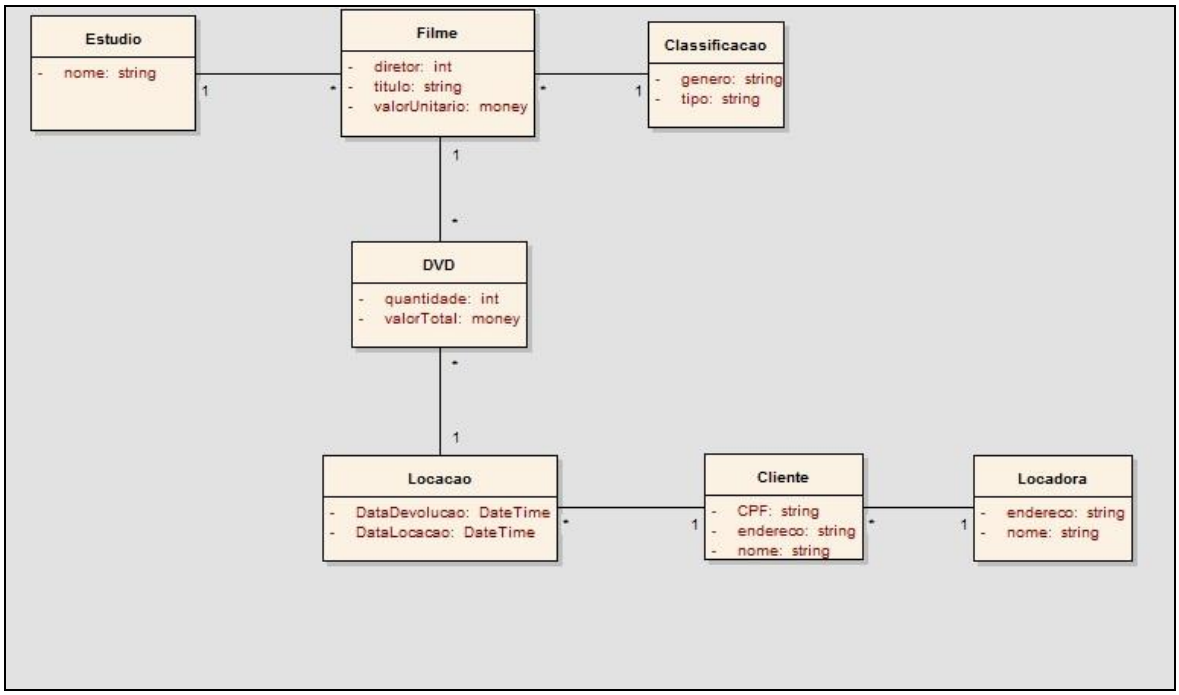


Figura 39: Diagrama de classes de uma locadora

Com a importação finalizada com sucesso, verifica-se na Figura 40 que todas as tabelas foram criadas na base de dados, assim como seus atributos (conforme Figura 41). A tabela `aaa_objects` é a tabela onde a ferramenta Quellon adiciona todos os objetos do sistema, a tabela `aaa_members` é a tabela onde estão armazenados todos os campos do sistema.

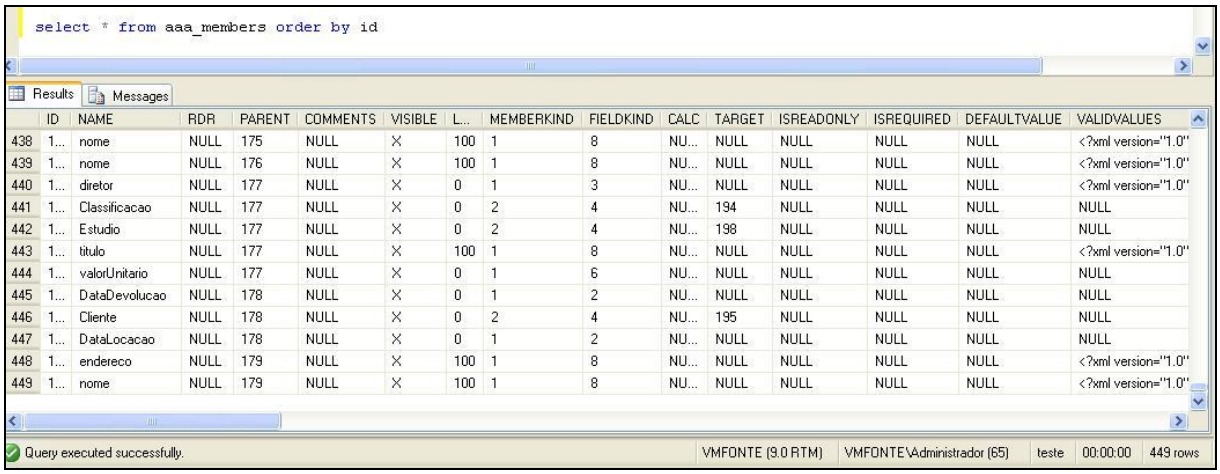
```
select * from aaa_objects order by id
```

Results | Messages

	ID	NAME	COMMENTS	CREATEHISTORY	UPDATEHISTORY	CREATEWOOWNER	PRIVATERECORDS	PRIVATEALLRECORDS	MANAGEOWNER	DATACENTERVERS
68	168	ARChildObjects	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
69	169	ARCommands	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
70	170	ARDefinitions	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
71	171	ARUserDefiniti...	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
72	172	Classificacao	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
73	173	Cliente	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
74	174	DVD	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
75	175	Estado	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
76	176	Estudio	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
77	177	Filme	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
78	178	Locacao	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL
79	179	Locadora	NULL	NULL	NULL	NULL	NULL	NULL	0	NULL

Query executed successfully. | VMFONTE (9.0 RTM) | VMFONTE\Administrador (65) | teste | 00:00:00 | 79 rows

Figura 40: *Select* em aaa_objects após importação do diagrama de classes



```
select * from aaa_members order by id
```

Results | Messages

	ID	NAME	RDR	PARENT	COMMENTS	VISIBLE	L...	MEMBERKIND	FIELDKIND	CALC	TARGET	ISREADONLY	ISREQUIRED	DEFAULTVALUE	VALIDVALUES
438	1...	nome	NULL	175	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"
439	1...	nome	NULL	176	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"
440	1...	diretor	NULL	177	NULL	X	0	1	3	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"
441	1...	Classificacao	NULL	177	NULL	X	0	2	4	NU...	194	NULL	NULL	NULL	NULL
442	1...	Estudio	NULL	177	NULL	X	0	2	4	NU...	198	NULL	NULL	NULL	NULL
443	1...	titulo	NULL	177	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"
444	1...	valorUnitario	NULL	177	NULL	X	0	1	6	NU...	NULL	NULL	NULL	NULL	NULL
445	1...	DataDevolucao	NULL	178	NULL	X	0	1	2	NU...	NULL	NULL	NULL	NULL	NULL
446	1...	Cliente	NULL	178	NULL	X	0	2	4	NU...	195	NULL	NULL	NULL	NULL
447	1...	DataLocacao	NULL	178	NULL	X	0	1	2	NU...	NULL	NULL	NULL	NULL	NULL
448	1...	endereco	NULL	179	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"
449	1...	nome	NULL	179	NULL	X	100	1	8	NU...	NULL	NULL	NULL	NULL	<?xml version="1.0"

Query executed successfully. | VMFONTE (9.0 RTM) | VMFONTE\Administrador (65) | teste | 00:00:00 | 449 rows

Figura 41: *Select* em aaa_members após importação do diagrama de classes

Na Figura 42 verificamos as tabelas criadas dentro do sistema.

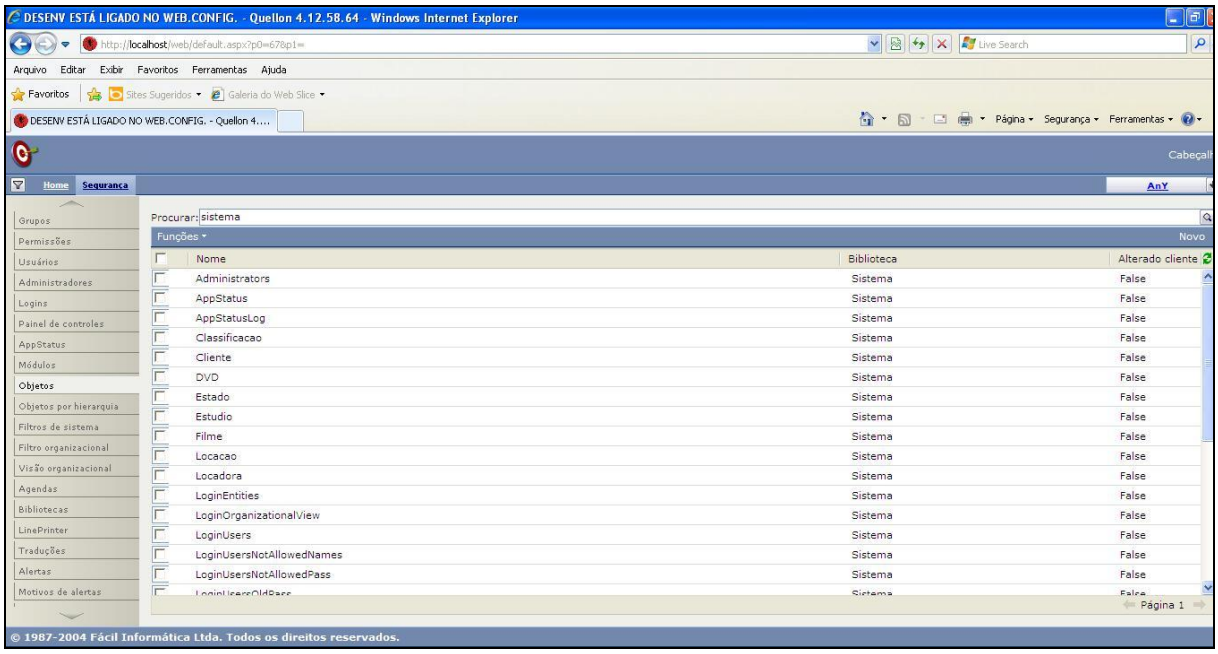


Figura 42: Objetos criados no sistema

Na Figura 43 verificamos os campos criados na tabela DVD.

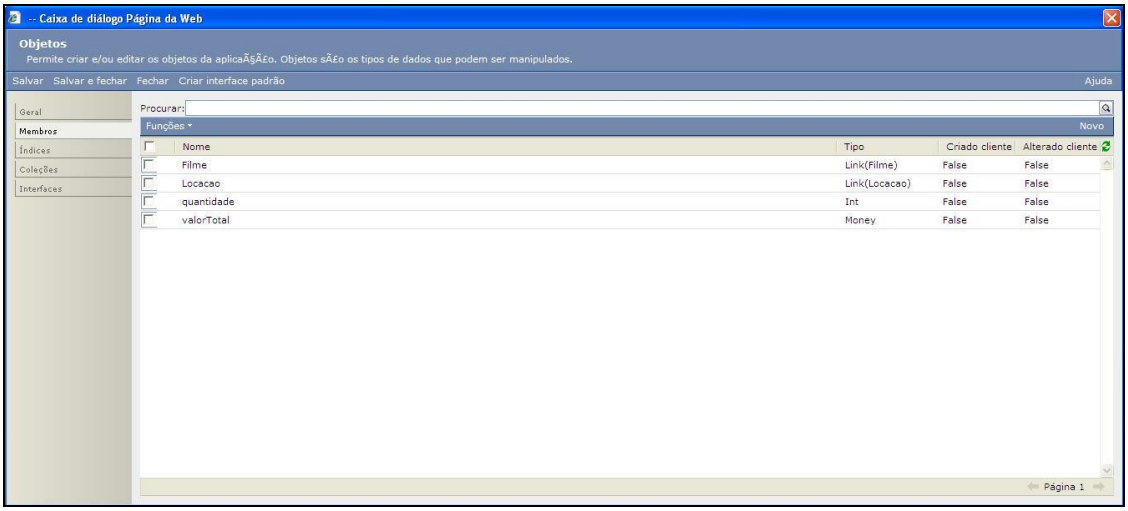


Figura 43: Campos criados na tabela DVD

3.4 RESULTADOS E DISCUSSÃO

A ferramenta QEA foi desenvolvida visando atender a necessidade de integrar a ferramenta Quellon com uma ferramenta de que possibilitasse criar diagramas de classes. Com importantes clientes utilizando o EA como ferramenta para projetarem seus sistemas, a Quellon Sistemas do Brasil S.A optou por integrar sua ferramenta de desenvolvimento para sistemas *web* ao EA.

Com a integração concluída, a ferramenta foi testada pela equipe de qualidade da Quellon Sistemas do Brasil S.A. Foram feitos testes com diagramas simples, de apenas duas classes e uma ligação entre elas e com diagramas mais completos, com 16 classes e várias ligações entre as classes, sempre utilizando ligações um para muitos. Foi possível perceber que a utilização da ferramenta diminui o tempo para criar das bases de dados e reduz as chances de erros na criação dos tipos ou nome dos dados a ser inseridos na base de dados.

Comparando com os trabalhos correlatos apresentados, verifica-se que Batista (2007) utilizou Borland Delphi 6 como software para desenvolver sua ferramenta. O acesso ao repositório do EA foi realizado através de uma biblioteca de interface de automação criada a partir do Borland Delphi 6. Becker (2009) utilizou a linguagem de programação C# e a IDE

Visual Studio 2008 para construção de sua ferramenta. Utilizou a opção de exportação em formato XMI do EA e a biblioteca *System.XML* para leitura dos dados exportados.

4 CONCLUSÕES

Neste trabalho é apresentada a ferramenta QEA que possibilita a integração entre o *Enterprise Architect* e a ferramenta para desenvolvimento de sistemas *web* Quellon. A ferramenta integra o modelo de classes do EA à base de dados da Quellon, criando na base de dados automaticamente todas as tabelas, campos e *foreign keys* que estão expostas em um diagrama de classes no EA. A ferramenta sugere a extinção da criação manual do banco de dados de um sistema que utiliza Quellon, eliminando a interação humana.

A ferramenta foi desenvolvida na linguagem de programação C# / ASP.NET, assim como a ferramenta Quellon.

A ferramenta atingiu os objetivos propostos, e como resultado tem-se a eliminação do re-trabalho e brechas para possíveis erros, como criação de tabelas ou campos errôneos, que faziam com que a base de dados perdesse referências para o diagrama de classes. Além disso, a ferramenta contribui para que o projeto seja entregue com mais rapidez, diminuindo seus custos.

A ferramenta não trata relacionamentos entre duas classes quando esses forem generalização, composição, agregação ou dependência. A ligação entre duas classes deverá ser sempre uma associação e a multiplicidade deverá ser sempre um para muitos.

Assim, pelos resultados obtidos, o estudo de caso avalia positivamente a proposta da ferramenta e demonstra que sua aplicação trará benefícios para clientes da Quellon.

4.1 EXTENSÕES

Para trabalhos futuros, tem-se como sugestão possibilitar ao usuário escolher quais tabelas devem ser criadas no banco de dados ao importar o XML exportado pelo EA, através de uma tela com uma lista de todas as tabelas que estão no XML exportado e que possibilite ao usuário escolher quais ele deseja criar. Como a ferramenta QEA foi implementado neste trabalho, é feita uma comparação entre todas as tabelas que estão no XML com todas as tabelas que estão no banco de dados, para criar apenas as que não estão na base de dados e excluir da base de dados as tabelas que não estão no XML. Com isso, se no diagrama de classes de um grande sistema (com muitas classes) for criada apenas uma nova tabela, o

processo de importação pela ferramenta tornar-se-ia lento.

Sugere-se também que a ferramenta trate quaisquer multiplicidades possíveis (não apenas 1..*) e todos os tipos de relacionamentos possíveis entre classes (não apenas associação).

REFERÊNCIAS BIBLIOGRÁFICAS

BATISTA, Raphael Marcos. **Ferramenta de gerencia de requisitos de software integra com Enterprise Architect**. 2007. 67 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau

BECKER, André Luis. **Ferramenta para construção de interfaces de software a partir de diagrama de classes**. 2009. 31 f. Trabalho de Conclusão de Curso (Bacharel em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML.** Rio de Janeiro: Elsevier, 2002. 286 p.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML.** 2ed. Rio de Janeiro: Elsevier, 2007. 369 p.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivair. **UML guia do usuário**. Rio de Janeiro: Campus, 2000. 472p. Tradução de: The unified modeling language user guide.

CAMARA, Fábio. **Dominando o Visual Studio .NET com C#.** 2ed. Florianópolis: Visual Books, 2005. 394 p.

DATE, C.J. **Introdução a sistemas de banco de dados.** Rio de Janeiro: Campus, 1991. 674 p.

ELMASRI, Raméz; NAVATHE, Shamkant B. **Sistemas de banco de dados**. São Paulo: Pearson Education do Brasil, 2005. 724 p.

FURLAN, José Davi. **Modelagem de objetos através de UML**. São Paulo: Makron Books, 1998. 329p. Tradução de: The Unified Modeling Language.

HADDAD, Renato Ibrahim. **C#: aplicações e soluções**. 2.ed. São Paulo: Érica, 2001. 398 p.

HAY, David C. **Princípios de modelagem de dados**. São Paulo: Makron Books, 1999. 271 p. Tradução de: Data Model Patterns.

LIMA, Adilson da Silva. **UML 2.0 do requisito à solução**. 1.ed. São Paulo: Érica, 2005. 326 p.

LIPPMAN, Stanley B. **C#: um guia prático**. Porto Alegre: Bookman, 2003. 316 p. Tradução de: C# primer: a practical approach.

MONTEIRO, Emiliano Soares. **Projetos de sistemas e banco de dados.** Rio de Janeiro: Brasport, 2004. 321 p.

MSDN. **Introdution to the C# Language and the .NET Framework.** [S.1]. 2011d. Disponível em: <http://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>. Acesso em: 03 nov. 2011.

MSDN. **Restrições FOREIGN KEY.** [S.1]. 2011c. Disponível em: <http://msdn.microsoft.com/pt-br/library/ms175464.aspx>. Acesso em: 16 out. 2011.

MSDN. **Visão geral conceitual do .NET Framework.** [S.1]. 2011b. Disponível em: <http://msdn.microsoft.com/pt-br//library/zw4w595w.aspx>. Acesso em 03 nov. 2011.

MSDN. **Visual C# Developer Center.** [S.1]. 2011a. Disponível em: <http://msdn.microsoft.com/pt-br/vcsharp/dd919145.aspx >. Acesso em 03 nov. 2011.

PILONE, Dan; PITMAN, Neil. **UML 2:** rápido e prático: guia de referência**.** Rio de Janeiro: Alta Books, 2006. 191 p.

PRESSMAN, Roger S. **Engenharia de software**. 5 ed. Rio de Janeiro: McGraww-Hill, 2002. 843 p. Tradução de : Software engeneering: a practitionaer's approach.

QUELLON DO BRASIL SISTEMAS S.A. **Quellon.** Blumenau, 2004. Disponível em: <http://www.quellon.com/website/pt_br/solucoes_quellon.asp >. Acesso em: 08 set. 2011.

SETZER, Waldemar W.;NASSU, Eugênio A. **Bancos de dados orientados a objetos** Ed. Blucher, 1999. 122 p.

SILVA, Alberto; VIDEIRA, Carlos. **UML metodologias e ferramentas case**. 2. Ed. Lisboa, 2005. 357 p.

SPARX SYSTEMS. **Sparx Systems**. Creswick, 2011. Disponível em: < http://www.sparxsystems.com/about.html >. Acesso em: 15 out. 2011.

SPARX SYSTEMS. **Enterprise Architect**: user guide. Version 6.5 [S.l], 2006. Documento eletrônico disponibilizado com o ambiente Enterprise Architect 6.5.

APÊNDICE A – DETALHAMENTO DOS CASOS DE USO

Este Apêndice contém o detalhamento dos casos de uso previstos no diagrama apresentado na seção 3.3.1. O Quadro 3 ilustra o caso de uso “Cadastrar Sistema”.

Caso de uso – UC.001 – Cadastrar sistema
Ator: Programador
Objetivo: A ferramenta deverá permitir o cadastro de sistemas

Pós-condições: Sistema cadastrado na ferramenta

Cenário Principal

1. Programador digita URL de acesso ao sistema.
2. Programador digita usuário de acesso ao sistema.
3. Programador digita senha de acesso ao sistema.
4. Ferramenta cadastra sistema.

Cenário de Exceção

No passo 1 do cenário principal, se programador digitar URL inválida, ferramenta emite mensagem de erro.

No passo 2 do cenário principal, se programador digitar usuário inválido, ferramenta emite mensagem de erro.

No passo 3 do cenário principal, se programador digitar senha inválida, ferramenta emite mensagem de erro.

Quadro 3: Caso de uso “Cadastrar Sistema”

O Quadro 4 ilustra o caso de uso “Gerar estrutura”.

Caso de uso – UC.002 – Gerar estrutura
Ator: Programador
Objetivo: A ferramenta deverá gerar toda estrutura física em um banco de dados
Pré-condições: Sistema deverá ter sido cadastrado

Cenário Principal

1. A ferramenta lê o arquivo XML importado.
2. A ferramenta gera a estrutura física de banco de dados do sistema.
3. A ferramenta emite documento informando quais tabelas foram criadas na base de dados.

Cenário Exceção

No passo 1 do cenário principal, se o arquivo XML não estiver no padrão XMI, a ferramenta emite mensagem de erro.

No passo 2 do cenário principal, caso ocorra erro ao gerar a estrutura física no banco de dados, a ferramenta gera um documento, contendo o log do erro.

Quadro 4: Caso de uso “Gerar estrutura”