UNIVERSIDADE FEDERAL DE SANTA CATARINA

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

Alysson Marques

Marco Aurélio Silva

UM FRAMEWORK PARA PERSISTÊNCIA DE DADOS EM PHP

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciências da Computação.

Orientador: José Eduardo De Lucca

Florianópolis-SC, julho de 2004.

UM FRAMEWORK PARA PERSISTÊNCIA DE DADOS EM PHP

Alysson Marques

Marco Aurélio Silva

_____
José Mazzucco Jr.
Coordenador do Curso

_____
José Eduardo De Lucca
Orientador

_____
João Bosco Mangueira Sobral

Banca Examinadora:

_____
Marcelo Thiry Comicholli da Costa

_____
Mario Antonio Ribeiro Dantas

Florianópolis, Julho de 2004.

“O conhecimento não é imediato.”

Dílson Lucala da Costa

AGRADECIMENTOS

Agradecemos aos nossos pais, professores e em especial a Deus, pela honra de ter nos concebido membros da 992,

a Pior Turma de Todos os Tempos.

RESUMO

O desenvolvimento de aplicações para WEB possui características que se repetem para diferentes contextos, sendo talvez a necessidade de se armazenar e manipular informações em bases de dados a principal delas.

Este trabalho relata algumas dificuldades encontradas no desenvolvimento de aplicações em PHP com acesso a base de dados e apresenta um proposta para soluções destes problemas.

Esta proposta consiste na implementação de um framework de suporte a persistência de dados fazendo uso da linguagem de programação PHP e os principais bancos de dados utilizados no mercado atual.

Este framework consiste basicamente de três geradores, que irão gerar automaticamente classes que visam facilitar e acelerar o processo de desenvolvimento, seja com as funções de inserir, remover, atualizar e procurar objetos no banco de dados, seja com os formulários de acesso que serão gerados.

ABSTRACT

The development of applications for WEB have characteristics that repeat for different contexts, being the necessity of storing and manipulating information in databases the main one of them.

This work tells to some difficulties found in the development of applications in PHP with access to the database and presents a proposal for solutions of these problems.

This proposal consists of the implementation of one framework of support persistence of data making use of the programming language PHP and the main data bases used in the current market.

This framework basically consists of three generators, that will automatically generate class that they aim at to facilitate and to speed up the development process, either with the functions to insert, to remove, to update and to look objects in the data base, either form of access that will be generated.

SUMÁRIO

ÍNDICE DE FIGURAS

1 Introdução

O desenvolvimento de aplicações para WEB possui características que se repetem para diferentes contextos, sendo talvez a necessidade de se armazenar e manipular informações em bases de dados a principal delas.

A linguagem PHP é bastante utilizada nessas aplicações, pois, além de ser de fácil aprendizado, oferece funções de acesso as principais bases de dados. Porém essas funções não são padronizadas, o que reduz a portabilidade destas aplicações.

O conceito de orientação a objetos (OO) trouxe diversas facilidades para o desenvolvimento de software em geral, porém quando este conceito é associado ao uso de informações armazenadas em bancos de dados faz-se necessária a utilização de uma metodologia de manipulação dessas informações pelas classes envolvidas na lógica dos sistemas.

A essência deste trabalho é a criação de um artefato que proporcione portabilidade e utilize o conceito de OO associado a padrões de projeto com o objetivo de reduzir o esforço de desenvolvimento de aplicações em PHP com acesso a base de dados.

2 Tecnologias Estudadas

2.1 PHP

O PHP foi criado por Rasmus Lerdorf inicialmente como alguns scripts em Perl para contabilizar estatísticas de acesso para seu currículo on-line. Estes scripts receberam o nome de “Personal Home Page Tools”, e como outras funcionalidades foram sendo inseridas, como acesso a bancos de dados, logo o autor escreveu uma implementação muito maior em C. Rasmus resolveu disponibilizar o código fonte do PHP/FI (Personal Home Page/Forms Interpreter) para que todos pudessem ter acesso, e também usá-lo, bem como consertar bugs e melhorar o código.

O PHP é hoje uma das linguagens de programação mais utilizadas no desenvolvimento de aplicações para WEB. Existem vários motivos para este sucesso, entre eles o fato de ser uma linguagem de código aberto, de fácil aprendizagem, ser portável a vários Sistemas Operacionais e não necessitar de muitos recursos computacionais.

2.2 eXtensible Markup Language (XML)

XML é uma linguagem de marcação em formato texto simples e altamente flexível destinada a descrever e estruturar informações. Tornou-se um padrão aceito mundialmente por possibilitar a comunicação entre diferentes aplicações, independentemente de plataforma ou tecnologia proprietária. Além disso, mensagens em XML podem ser transferidas pela rede através de protocolos padrões da Internet, tais como o HTTP.

Um documento XML é composto de tags que contêm o significado dos dados que elas próprias contêm. Ao contrário da HTML (HyperText Markup Language), na XML o conjunto de tags não é pré-determinado, ou seja, é possível criar um conjunto de tags personalizado para cada documento de acordo com o desejo ou necessidade do usuário ou aplicação. Resumindo, XML oferece uma estrutura padrão pela qual é possível criar outras estruturas.

Com isso, torna-se necessária a existência de um método para avaliar se um documento XML está de acordo com a estrutura esperada para um determinado tipo de aplicação. Isto pode ser feito comparando o documento em questão com um DTD (Document Type Definition) ou com um XSD (XML Schema Definition). DTDs e XSDs descrevem a estrutura de um documento XML, informando qual tipo de dados cada tag do XML pode conter, a ordem e hierarquia em que estas podem se encontrar no documento.

Infelizmente, DTDs não são um meio flexível para definição de formatos de documentos XML, pois não oferecem facilidade para expressar tipos de dados ou relações estruturais complexas.

Quando consideramos XSDs, torna-se importante entender o conceito de XML namespace. XML namespace é um método para evitar conflitos de nome. Para explicarmos o que são conflitos de nome colocaremos um exemplo que segue abaixo:

Vamos supor dois documentos XML. O primeiro contém a seguinte estrutura:

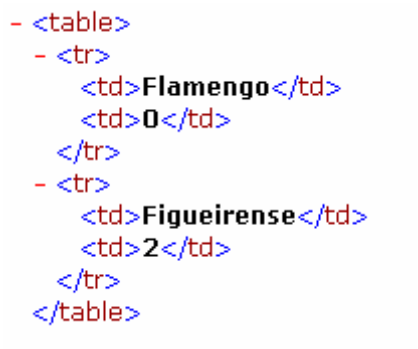

Figura 1 - Exemplo de Estrutura XML

E o segundo contém a seguinte estrutura:

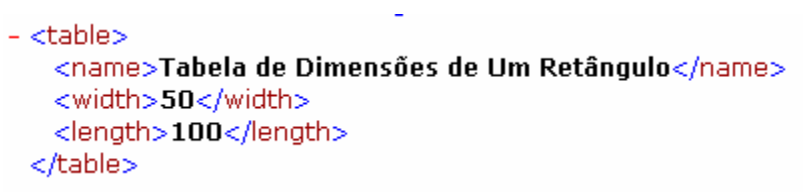

Figura 2 - Exemplo de Estrutura XML

Se estes dois documentos forem interpretados juntos teremos um conflito de nomes, pois o elemento <table> foi definido com estruturas diferentes.

Vamos resolver este conflito adicionando o atributo xmlns no elemento <table>. Logo teremos no primeiro arquivo a seguinte estrutura:

Figura 3 - Exemplo de Estrutura XML com atributo xmlns

Já no segundo, teremos a estrutura a seguir:

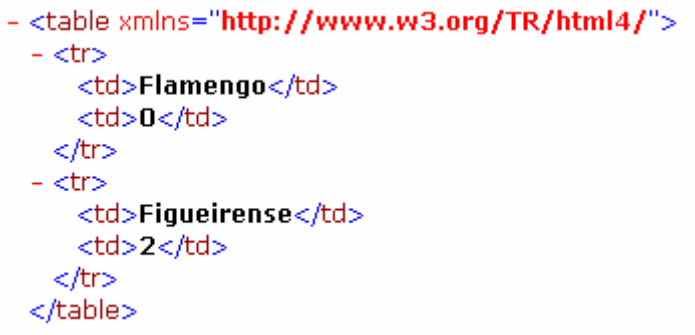

Figura 4 - Exemplo de Estrutura XML com atributo xmlns

2.3 AdoDB

2.3.1 O que é ADOdb

ADOdb (Active Data Objects DataBase) é uma biblioteca de abstração de banco de dados que foi criada por John Lim. Ela fornece uma API comum para vários bancos de dados diferentes, fazendo com quem haja portabilidade entre esses bancos de dados suportados, além de aumentar a reusabilidade de código. Atualmente ADOdb tem drivers para vários bancos de dados como MySQL, PostgreSQL, Oracle, Interbase, Microsoft

SQL Server, Access, FoxPro, Sybase entre outros.

Existem, hoje, várias bibliotecas de abstração de banco de dados para PHP, como PEAR DBI, Metabase e PHPLib. A biblioteca ADOdb foi utilizada por ser uma das mais completas e eficientes, tanto que alguns projetos de código aberto como PostNuke, por exemplo, a utilizam.

Algumas das características que fazem com que ADOdb seja melhor que as outras bibliotecas de abstração são:

- Fornecer suporte para trabalhar com inserts e updates que podem ser facilmente adapatados para vários banco de dados. Possui métodos para manuseio de de datas, concatenação de strings, etc.
- Se um sistema de metatipos para mapear tipos de dados equivalents entre banco de dados diferentes.

2.3.2 Como Funciona

ADOdb cria uma camada de abstração entre a aplicação PHP e o banco de dados, fazendo com que seja transparente para a aplicação qual banco de dados estamos utilizando. O que ela faz é pegar os métodos particulares de cada driver, como o MySQL por exemplo, e mapeá-los para novos métodos, comuns a todos os drivers.

A Figura 5 mostra como ocorre a interação entre aplicações PHP sem usar uma camada de abstração de banco de dados e depois mostra como

ocorre quando a usamos.

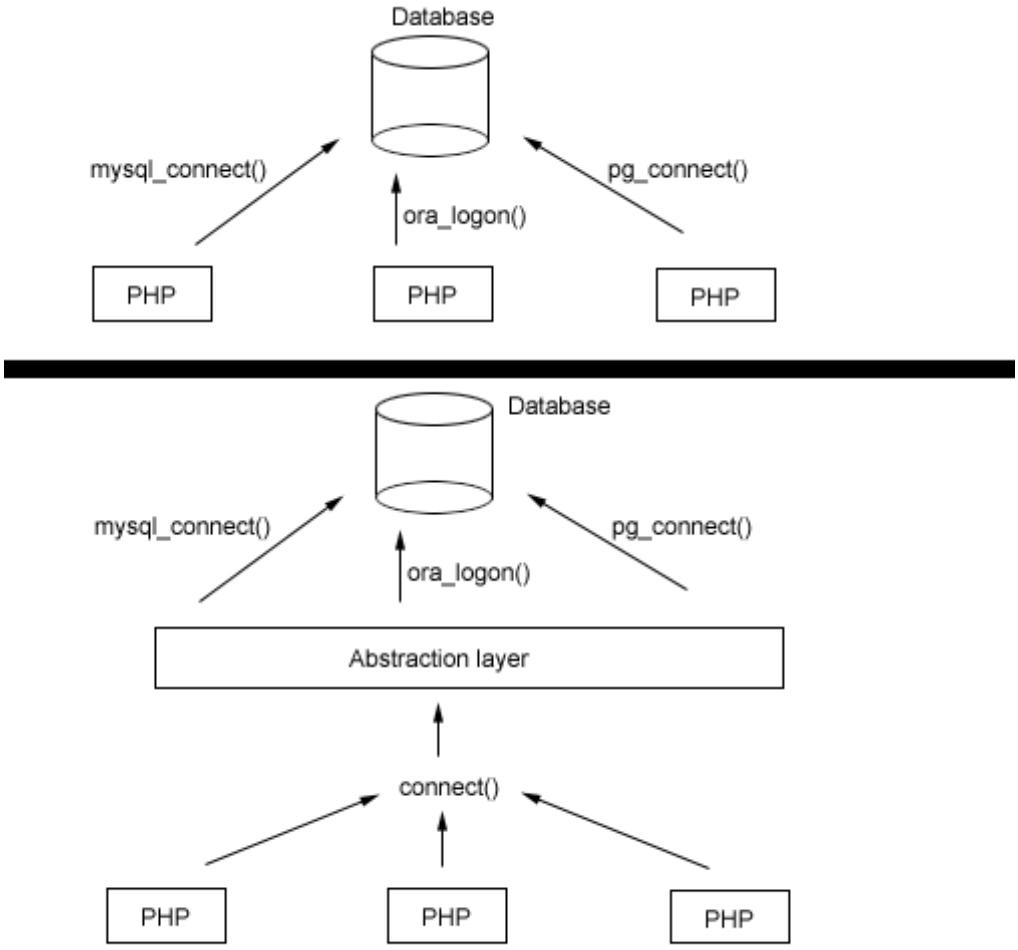

Figura 5 - Conectando no banco dados com uma camada de abstração

2.4 Framework

2.4.1 Definição

Um “Framework” é o projeto de um conjunto de objetos que colaboram entre si para execução de um conjunto de responsabilidades. Um framework reusa análise, projeto e código. Ele reusa análise porque

descreve os tipos de objetos importantes e como um problema maior pode ser dividido em problemas menores. Ele reusa projeto porque contém algoritmos abstratos e descreve a interface que o programador deve implementar e as restrições a serem satisfeitas pela implementação. Ele reusa código porque torna mais fácil desenvolver uma biblioteca de componentes compatíveis e porque a implementação de novos componentes pode herdar grande parte de seu código das super-classes abstratas. Apesar de todos os tipos de reuso serem importantes, o reuso de análise e de projeto são os que mais compensam a longo prazo [Johnson 91].

2.4.2 Classficação de Frameworks

Os frameworks são classificados em três grupos [Fayad 97]: Frameworks de infraestrutura do sistema, frameworks de integração de “middleware” e frameworks de aplicação empresarial.

Os frameworks de infra-estrutura do sistema simplificam o desenvolvimento da infra-estrutura de sistemas portáveis e eficientes, como por exemplo os sistemas operacionais, sistemas de comunicação, interfaces com o usuário e ferramentas de processamento de linguagem. Em geral são usados internamente em uma organização de software e não são vendidos a clientes diretamente.

Os frameworks de integração de “middleware” são usados, em geral, para integrar aplicações e componentes distribuídos. Eles são projetados para melhorar a habilidade de desenvolvedores em modularizar, reutilizar e

estender sua infra-estrutura de software para funcionar “seamlessly”[1] em um ambiente distribuído. Exemplos dessa classe de framework são o “Object Request Broker” (ORB), “middleware” orientado a mensagens e bases de dados transacionais.

Os frameworks de aplicação empresarial estão [2]voltados a domínios de aplicação mais amplos e são a pedra fundamental para atividades de negócios das empresas, como por exemplo sistemas de telecomunicações, aviação, manufatura e engenharia financeira. Frameworks dessa classe são mais caros para desenvolver ou comprar, mas podem dar um retorno substancial do investimento, já que permitem o desenvolvimento de aplicações e produtos diretamente.

2.5 Padrões de Projeto

Um padrão de projeto é uma solução encontrada para um determinado problema dentro de um contexto de desenvolvimento de um software que pode ser reutilizada na resolução de problemas similares em outros contextos.

Nas próximas duas seções serão abordados os padrões de projeto utilizados neste trabalho, Value Object e Facade.

[1] Seamlessly – Sem emendas. O tipo de framework mencionado faz a integração de componentes distribídos de forma transparente ao usário.

2.5.1 Value Object

O *Value Object* é um dos mais importantes e utilizados padrões de projeto. As aplicações J2EE (Java 2 Entrerprise Edition) implementam componentes de negócios do lado servidor como beans de sessão e beans de entidade. Alguns métodos expostos pelos componentes de negócios retornam dados para o cliente. Freqüentemente, o cliente chama os métodos *get* de um objeto de negócios várias vezes até obter todos os valores dos atributos. Para reduzir o número de chamadas remotas e evitar a sobrecarga de rede, os dados podem ser recuperados em uma única chamada que retorna um *Value Object*. Um *Value Object* encapsula os dados de negócio. Quando o cliente solicita ao *enterprise bean* os dados de negócios, o *enterprise bean* pode criar o objeto de dados, preenchê-lo com seus valores de atributos e passá-lo por valor para o cliente (ALUR e CRUPI e MALKS, 2002) (FERLIN,2004).

Neste trabalho utilizamos o conceito de *Value Object* utilizado em J2EE para proporcionar uma melhor representação de registros de uma tabela de banco de dados em um objeto.

2.5.2 Fachada – Facade

O acesso a bancos de dados requer que as aplicações executem várias chamadas de comandos SQL, como INSERT, UPDATE, DELETE e SELECT. A utilização de comandos SQL diretamente na implementação de regras de negócios torna o código de difícil leitura e manutenção. Para resolver

este problema, é criada uma classe facade, que torna as chamadas de comandos SQL transparente para a regra de negócio.

Utilizado em conjunto com o padrão *Value Object,* o padrão fachada implementa esta camada de abstração, que faz a integração banco de dados relacional-regra de negócio. Quando quiser remover um registro de uma tabela, o programador simplesmente chamará um método REMOVE de um objeto da classe facade, passando como um parâmetro um objeto VO que terá como valor do atributo ID_PK a chave primária da tabela correspondente. Isso deixa a regra de negócio muito mais compreensível, pois o programador não terá que utilizar um comando SQL como “DELETE FROM TABLE1 WHERE ID_PK=1”.

3 Um Framework para Persistência de Dados em PHP

No decorrer deste capítulo mostraremos com mais detalhes as características, funcionalidades, objetivos, desenvolvimento, dificuldades e as soluções encontradas no desenvolvimento deste projeto.

3.1 Descrição dos Problemas

3.1.1 Portabilidade

Apesar de o PHP ter suporte a vários tipos diferentes de bancos de dados, as funções de acesso a essas bases de dados não são padronizadas, o que dificulta a portabilidade das aplicações desenvolvidas. Quando se altera a base de dados é necessário que se altere todo o código envolvido com o acesso a base de dados, trocando as funções da base antiga pelas funções de acesso da nova base.

3.1.2 Orientação a Objetos e bases de dados relacionais

Um problema comum quando se usa orientação objetos, é decidir qual abordagem utilizar para passar os dados que estão encapsulados num objeto para o banco de dados. Geralmente, o programador tem toda a lógica de negócio orientada a objeto, e na hora de salvar, extrai os dados do objeto e monta uma SQL para inserção ou alteração dos dados no banco.

3.1.3 Geração de formulários

A maioria das aplicações desenvolvidas para WEB possui uma parte pública e uma parte com acesso restrito, onde os administradores do sistema podem manipular os dados das tabelas da aplicação. A parte de acesso restrito consiste, de uma maneira geral, em formulários que executam as seguintes ações: alteração, consulta, inclusão e exclusão de dados.

A implementação desses formulários é uma tarefa que consome um tempo de desenvolvimento considerável, que poderia ser dispensado para outras tarefas. Este processo pode ser automatizado, já que as etapas de construção dos formulários são muito similares para todas as tabelas da aplicação.

3.2 Proposta de Solução

O trabalho proposto visa a criação de um framework que proporcione mais facilidade, rapidez e portabilidade no desenvolvimento de aplicações em PHP com acesso a base de dados SQL.

O problema da portabilidade entre bancos de dados foi resolvido utilizando a biblioteca ADOdb. Através dela, o framework se adapta facilmente aos principais bancos de dados utilizados no mercado, bastando ao usuário apenas selecionar para qual banco de dados ele deseja gerar suas classes.

Com algumas adaptações e adequações ao ambiente do PHP, os padrões de projeto *Value Object* e *Facade* foram utilizados para resolver o problema de persistência de dados na programação orientada à objetos.

3.2.1 Diagrama de Classes

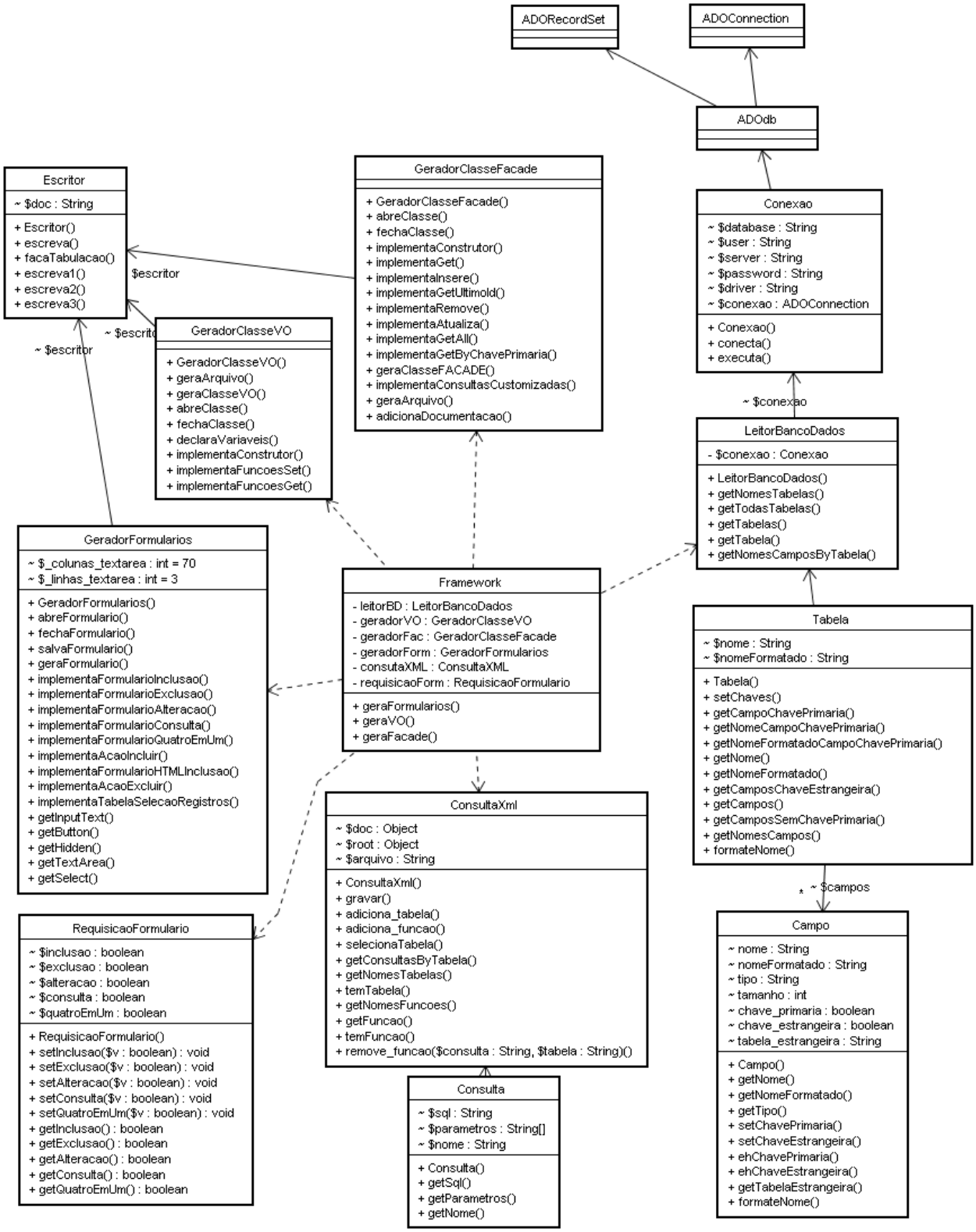

Figura 6 - Diagrama de Classes

4 Implementação

O Framework faz a geração automática das classes *Value Object* (VO) e *Session Facade* (Facade) e também dos formulários para manipulação de dados.

A classe VO (seção 2.5.1) encapsula um registro de uma tabela do banco de dados num objeto. Além disso ela fornece métodos para manipulação de seus atributos através das funções getAtributo, que retorna o valor do atributo, e setAtributo, que modifica o valor do atributo no objeto.

A classe Facade (seção 2.5.2) faz a ligação entre o banco de dados e o VO. É ela quem implementa todas as consultas SQL necessárias para que um registro de uma tabela seja “transformado” num objeto VO e também o caminho inverso, ou seja, pega um objeto VO e o transforma num registro de uma tabela.

Para gerar os formulários de uma tabela, todas as suas colunas são lidas e para cada uma é criado um campo de formulário em HTML, que dependendo do tipo de formulário irá servir para inserção, exclusão, visualização ou edição.

Para realizar essas funções, o sistema foi dividido nas seguintes classes: Leitor de Banco de Dados, Gerador de classes VO, Gerador de Classes

Facade e Gerador de Formulários. Para dar suporte a estas classes foram criadas as classes Tabela, Campo, Conexão, ConsultaXML, Consulta e Escritor.

4.1 Convenções Adotadas

Antes de abordar as classes que integram o sistema, faz-se necessário a especificação das convenções adotadas para o uso do framework proposto.

4.1.1 Chaves Primárias

No projeto do banco de dados, a nomenclatura para as chaves primárias deverá ser da seguinte forma: “id_” seguido pelo nome da tabela. Por exemplo, para a tabela Pessoa o nome da chave primária deverá ser “id_Pessoas”.

4.1.2 Chaves Estrangeiras

De maneira similar a nomenclatura das chaves primárias, as chaves estrangeiras também deverão ter o prefixo “id_” seguido pelo nome da tabela relacionada. Por exemplo, para a tabela Tipo_Pessoa o nome da chave primária deverá ser “id_Tipo_Pessoa”.

4.1.3 Formatação dos Nomes

Todas as tabelas e os campos no banco de dados seguem o padrão nome_tabela, nome_campo , id_nome_tabela. Quando convertidos para objetos, esses nomes serão transformados para nomeTabela, nomeCampo e idNomeTabela

4.2 Classes Auxiliares

4.2.1 Tabela

A classe Tabela é uma das classes mais importantes do Framework, pois é ela quem possui todas as informações que as classes geradoras precisam. Tem como atributo o nome da tabela, o nome formatado e um array de Campos. A Figura 7 mostra o diagrama da classe Tabela.

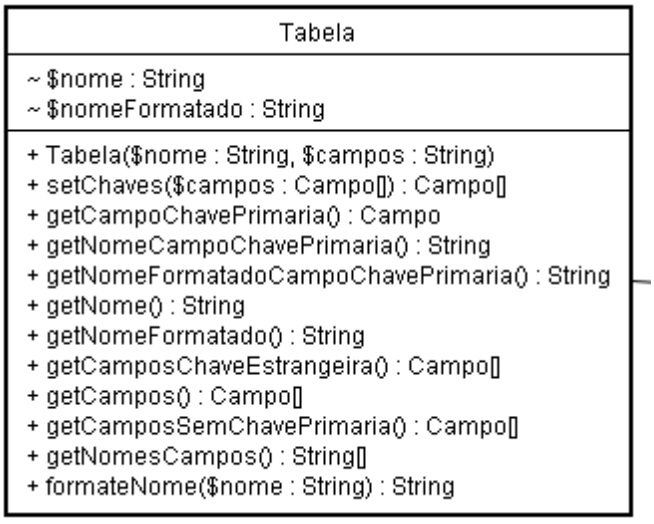

Figura 7 – Diagrama da Classe Tabela

As principais funções são:

- GetCampoChavePrimaria(): Retorna um objeto Campo que representa o campo chave primária da tabela.

- GetCampos(): Retorna um array com todos os objetos Campo da tabela.

- GetCamposSemChavePrimaria(): Retorna um array com todos os objetos Campo, exceto o campo chave primária. Esta função foi necessária pois o GeradorClasseFacade implementa funções de getByCampo para todos os campos da tabela, mas a função getByChavePrimária é especial pois ela não retorna um array de objetos como os outros getByCampo, e sim um único objeto.

4.2.2 Campo

A classe Campo se relaciona diretamente com a classe Tabela. Possui como atributo o nome do campo, nomeFormatado, tipo de campo e tamanho. Possui ainda o atributos booleanos chave_primaria, que indica se o campo é chave primária, chave_estrangeira que indica se o campo é chave estrangeira e o atributo tabela_estrangeira, que é uma string que informa qual a tabela_estrangeira o campo faz referência. A Figura 8 mostra o diagrama da classe Campo.

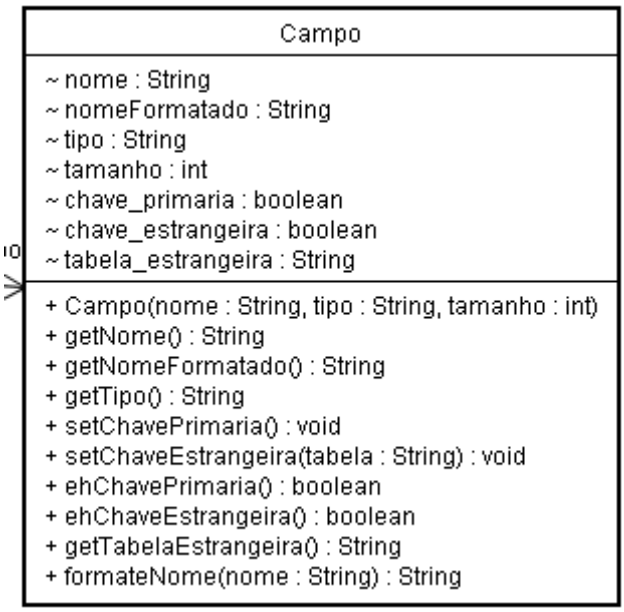

Figura 8 - Diagrama da Classe Campo

4.2.3 Conexao

A classe conexão, como o próprio nome diz, é quem faz a conexão com o banco de dados. É ela quem recebe toda as informações sobre o banco de dados, como endereço do servidor, nome do banco de dados, nome de usuário, senha e o tipo de banco de dados. Possui o atributo conexão, que é um objeto ADOConnection, que representa a conexão com o banco de dados. A Figura 9 mostra o diagrama da classe Conexao.

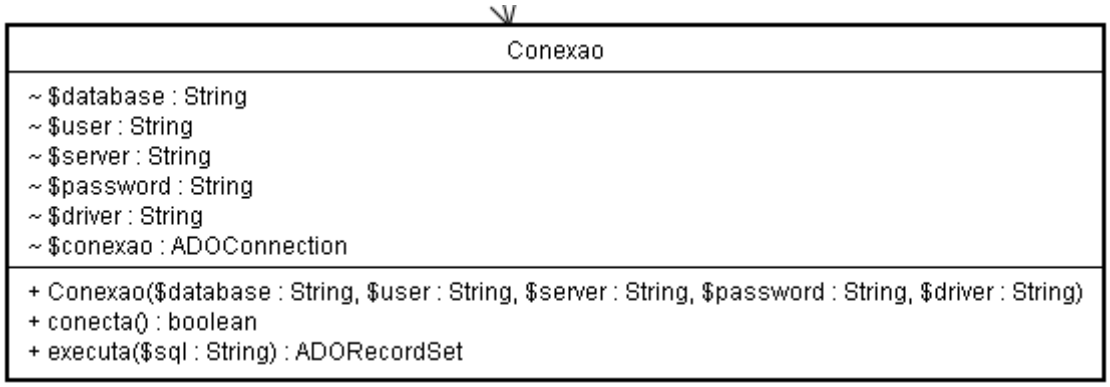

Figura 9 - Diagrama da Classe Conexão

4.2.4 Consulta

Representa uma consulta SQL. Possui uma string sql que é o código SQL, parâmetros que é uma array de string com os nomes do parâmetros e atributo o nome. A Figura 10 mostra o diagrama da classe Consulta.

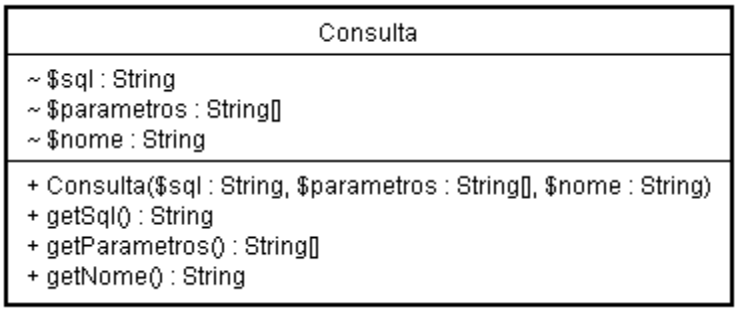

Figura 10 - Diagrama da Classe Consulta

4.2.5 Escritor

Uma classe bem simples, cuja função é auxiliar os geradores na formatação dos arquivos que serão gerados, facilitando a leitura das

classes e formulários gerados. A Figura 11 mostra o diagrama da classe Escritor.

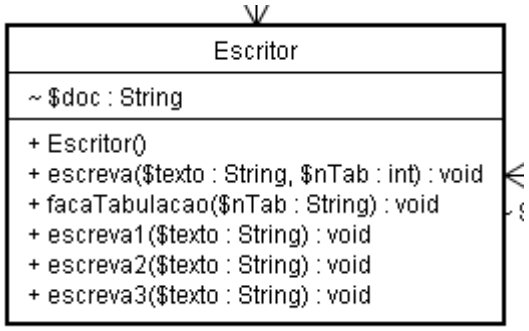

Figura 11 - Diagrama da Classe Escritor

4.2.6 RequisicaoFormulario

A função desta classe é informar ao gerador de formulário qual ou quais formulários deverão ser gerados. Possui como atributos cinco booleans ($inclusao, $exclusao, $alteracao, $consulta e $quatroEmUm) além de funções get e set para cada um destes atributos.

4.3 Leitor de Banco de Dados

Esta classe, como o próprio nome diz, é responsável pela leitura do bancde dados. Sua função é fornecer as informações necessárias para que as classes geradoras façam o seu trabalho. Para fazer isto, o Leitor possui um objeto Conexão como atributo e as seguintes funções:

- getNomesTabelas(): Retorna um array com o nome de todas as tabelas do banco de dados;
- getTodasTabelas(): retorna um array de objetos Tabela com todas a tabelas do banco
- getTabelas(NomesTabelas): retorna um array com objetos do tipo Tabela de acordo com o array de nomes passado como parâmeto.
- getTabela(NomeTabela): retorna um objeto Tabela.
- getNomesCamposByTabela(NomeTabela): Retorna um array com os nomes dos campos da tabela.

Figura 12 - Diagrama da Classe LeitorBancoDados

4.4 Definição das Classes VO

Uma classe VO é uma abstração de uma tabela do banco de dados. Ela possui apenas um atributo campos, que é um array que possui uma entrada para cada campo existente na tabela. Além disso, a classe VO possui, para cada campo, método GetByCampo que serve para recuperar

o valor do atributo, e método setCampo, que serve para modificar o valor do atributo.

4.5 Gerador de classes VO

O gerador de classes VO recebe um objeto Tabela, que foi criado pelo Leitor de Banco de Dados. O gerador então utiliza as funções deste objeto para poder escrever em um arquivo a classe VO propriamente dita. As principais funções do gerador são

- implementaConstrutor(Campos, NomeFormatado): Este método escreve no arquivo o construtor da classe VO. Inicialmente, a classe VO teria para campo da tabela uma declaração de atributo. Mais tarde, tornou-se necessário obter um array com todos os atributos do VO, e para facilitar a obtenção deste array modificamos a classe VO, fazendo com que ela possua um array de atributos. Com isto, o construtor passou a inicializar o array, declarando uma entrada no array para cada Campo passado como parâmetro.

- declaraVariaveis(): Devido a modificação na estrutura da classe VO que foi mencionada acima, este método deixou de declarar um atributo para cada campo e passou a declara apenas um único atributo do tipo array.

- implementaFuncoesSet(Campos): Este método implementa as funções set para todos os Campos recebidos como parâmetro.

- implementaFuncoesGet(Campos): Este método implementa as funções get para todos os Campos recebidos como parâmetro.

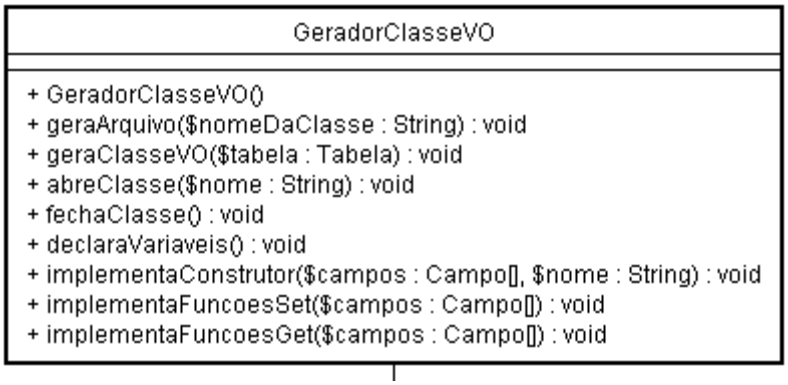

Figura 13 - Diagrama da Classe GeradorClasseVO

4.6 Definição das classes Facade

Uma classe Facade possui como atributo um objeto Conexao, e várias funções, sendo que algumas serão geradas por padrão e outras através de parâmetros pré-definidos pelo usuário.

4.6.1 Funções Geradas por Padrão

4.6.1.1 insere(objetoVO)

Insere um registro na tabela que guardará os valores contidos no objeto VO passado como parâmetro. Esta função tem como retorno o valor da chave primária que foi criada para o registro inserido. Caso tenha ocorrido algum erro e o registro não tenha sido inserido, o retorno será o *boolean false*.

A Figura 14 mostra um exemplo de código que utiliza da função insere(objetoVO).

```
$pessoaVO = new PessoaVO();
$pessoaVO->setNome("José Eduardo De Lucca");
$pessoaFacade = new PessoaFacade();
$id_pessoa=$pessoaFacade->insere($pessoaVO);
echo($id_pessoa);
```

Figura 14 - Exemplo de código para função insere(objetoVO)

O resultado deste código será:

55

4.6.1.2 getByCampo(param1)

Retorna um array de objetos VO contendo os registros que contem o valor passado como parâmetro no campo especificado pelo nome da própria função.

A Figura 15 mostra um exemplo de código que utiliza da função getByCampo(param1).

```
$pessoaFacade = new PessoaFacade();
$pessoaVO = $pessoaFacade->getByNome("%José%");
print_r($pessoaVO);
```

Figura 15 - Exemplo de Código para função getByCampo(param1)

O resultado deste código é mostrado na Figura 16.

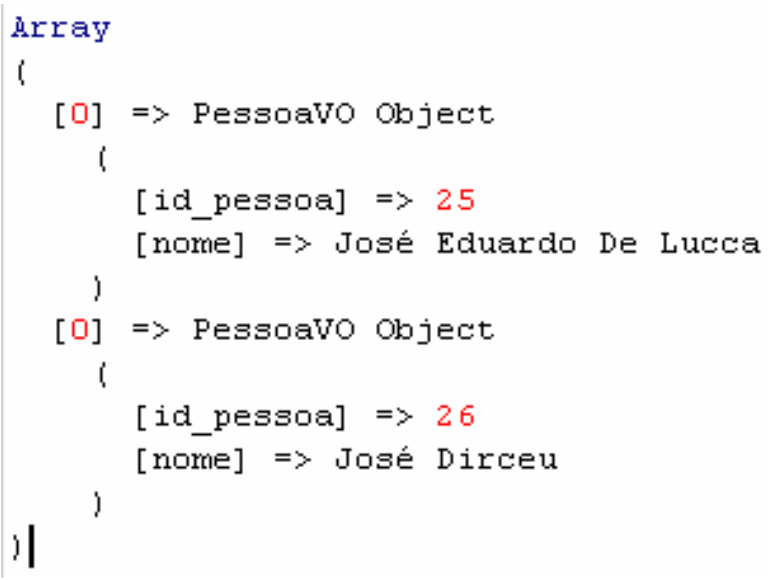

Figura 16 - Resultado do Código da Figura 15

4.6.1.3 getByPK(param1)

Retorna um objeto VO contendo o registro que contem o valor passado como parâmetro na sua chave primária.

A Figura 17 mostra um exemplo de código que utiliza da função getByPK(param1).

```
$pessoaFacade = new PessoaFacade();
$pessoaVO = $pessoaFacade->getByPK(25);
print_r($pessoaVO);
```

Figura 17 - Exemplo de Código da função getByPK(param1)

O resultado deste código será mostrado na Figura 18.

```
PessoaVO Object
(
    [id_pessoa] => 25
    [nome] => José Eduardo De Lucca
)
```

Figura 18 - Resultado do Código da Figura 16

4.6.1.4 atualiza(objetoVO)

Esta função atualiza o registro que corresponde ao objeto VO passado como parâmetro. Somente serão atualizados os campos que tiverem valor diferente de nulo. A função retornará um booliano, que corresponderá ao sucesso ou não da operação.

A Figura 19 mostra um exemplo de código que utiliza da função atualiza(ObjetoVO).

```
$pessoaVO = new PessoaVO();
$pessoaVO->setNome("Juliano Romani");
$pessoaFacade = new PessoaFacade();
$atualizou =$pessoaFacade->atualiza($pessoaVO);
if($atualizou)
    echo("atualizou!");
else
    echo("não atualizou");
```

Figura 19 - Exemplo de Código da Função atualiza(objetoVO)

Se a operação for realizada com sucesso, o resultado deste código será:

atualizou!

4.6.1.5 atualizaCompleto(objetoVO)

Esta função atualiza o registro que corresponde ao objeto VO passado como parâmetro. Todos os campos serão atualizados independente se tiverem valor nulo ou não. A função retornará um booliano, que corresponderá ao sucesso ou não da operação.

A Figura 20 mostra um exemplo de código que utiliza da função atualizaCompleto(ObejtoVO).

```
$pessoaVO = new PessoaVO();
$pessoaVO->setNome("Juliano Romani");
$pessoaFacade = new PessoaFacade();
$atualizou =$pessoaFacade->atualizaCompleto($pessoaVO);
if($atualizou)
    echo("atualizou!");
else
    echo("não atualizou");
```

Figura 20 - Exemplo de Código da Função atualizaCompleto(ObjetoVO)

Se a operação for realizada com sucesso, o resultado deste código será:

atualizou!

4.6.1.6 remove(objetoVO)

Esta função remove o registro que corresponde ao objeto VO passado como parâmetro. A função retornará um booliano, que corresponderá ao sucesso ou não da operação.

A Figura 21 mostra um exemplo de código que utiliza da função

remove(ObjetoVO).

```
$pessoaVO = new PessoaVO();
$pessoaVO->setIdPessoa(26);
$pessoaFacade = new PessoaFacade();
$removeu =$pessoaFacade->remove($pessoaVO);
if($removeu)
    echo("removeu!");
else
    echo("não removeu");
```

Figura 21 - Exemplo de Código da Função remove(ObjetoVO)

Se a operação for realizada com sucesso, o resultado deste código será:

removeu!

4.6.2 Funções Customizadas pelo Usuário

4.6.2.1 Consulta XML

Esta classe tem a finalidade de dar suporte ao Gerador de Facades, consultando e gravando num arquivo XML. O sistema tem uma interface para que o usuário insira, altere ou apague as funções de uma tabela, e ela se encarregará de fazer as alterações no arquivo XML. É ela quem informa ao Gerador de Facade quais as funções que deverão ser implementadas, passando um array de objetos do tipo Funcao. A Figura 22 mostra o diagrama da classe ConsultaXML.

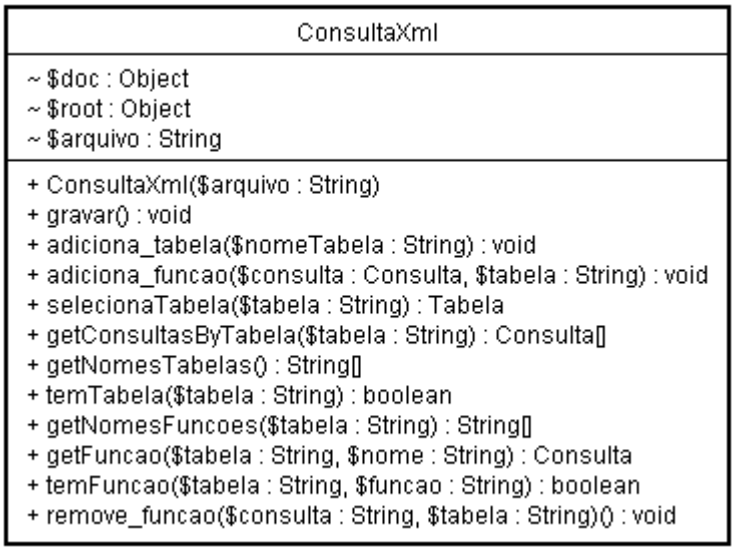

Figura 22 - Diagrama da Classe ConsultaXML

```
<?xml version="1.0" ?>
- <tabelas>
  - <pessoa>
    - <funcoes>
      - <funcao nome="getByPkeNome">
        - <parametros>
            <parametro valor="id" />
            <parametro valor="nome" />
          </parametros>
          <sql>select * from pessoa where id_pessoa=$id and nome=$nome</sql>
        </funcao>
      - <funcao nome="getByPkOuNome">
        - <parametros>
            <parametro valor="id" />
            <parametro valor="nome" />
          </parametros>
          <sql>select * from pessoa where id_pessoa=$id or nome=$nome</sql>
        </funcao>
      </funcoes>
    </pessoa>
  </tabelas>
```

Figura 23 - Exemplo de Consulta Customizada em Arquivo XML

4.7 Gerador de classes Facade

Assim como o gerador de classes VO, o gerador de classes Facade também recebe um objeto Tabela do Leitor de Banco de Dados. O gerador então utiliza as funções deste objeto para poder escrever em um arquivo a classe Facade. Seus principais Métodos são:

- Métodos que implementam os Get: Este métodos implementam um laço que pega o resultado da consulta ao banco de dados, coloca os valores de todos os campos retornados num objeto VO e em seguida adiciona o objeto VO num array que será o retorno da função.
    - implementaGet(Tabela) : Este método recupera todos os campos excluindo o campo que é chave primária. Para cada campo será implementado uma função getByCampo, que retornará um array de objetos VO.
    - implementaGetByChavePrimaria(Tabela): Este método recupera o campo chave primária e implementa a função getByChavePrimaria. Esta função é implementada separadamente dos outros get's pois ao invés de retornar um array de objetos VO retorna apenas um objeto VO, ou seja, retornará apenas o objeto que está no primeiro índice do array.

- implementaGetAll(Tabela): Este método implementa a função que retorna um array com todos os objetos que existem no banco de dados.

- Método implementaAtualiza(Tabela, Tipo): Este método implementa as funções atualiza e atualizaCompleto. Para gerar o cláusula SQL, que irá fazer o *update* dos dados no banco de dados, foi utilizado a função getUpdateSQL(ADORecordSet, Campos) do ADOdb. Para isso, primeiro é feita uma consulta ao banco de dados passando como parâmetro a chave primária do VO e que retornará um ADORecordSet. Em seguida a função getUpdateSQL verifica quais campos estão diferente no ADORecordSet e nos Campos do VO, e irá colocar na cláusula SQL todos os campos que estiverem diferentes. A diferença entre as funções atualiza e atualizaCompleto é que quando um atributo está com o valor nulo no VO e tem algum valor diferente de nulo no banco de dados, a função atualiza não atualizará o valor para nulo, enquanto que a função atualiza completo atualizará o valor no banco. Isto é feito com a definição da constante ADODB_FORCE_NULLS, fazendo que ela seja 1 para atualizar completo, e 0 para o atualiza.

- Método implementaInsere(Tabela): Este método implementa a função insere. Similarmente ao método implementaAtualiza, faz uso de função do ADOdb getInsertSQL(NomeTabela,Campos). A função getInsertSQL é um pouco mais simples que a função getUpdateSQL,

pois só precisa do nome da tabela e de um array com os campos que serão inseridos. Ela irar gerar uma cláusula SQL que irá inserir todos os campos que não forem nulos. Para gerar a chave primária do registro que será inserido, é chamada a função getUltimoId, que retorna o maior valor de chave primária da tabela, e incrementado o resultado em 1.

- Método implementaRemove(Tabela): Este método implementa a função que irá remover o VO, ele simplesmente gera a cláusula SQL que irá apagar o registro do banco pela chave primária do VO.

- Método implementaConsultasCustomizadas(Tabela, Consultas): Este método implementa as funções que foram feitas pelo usuário. Para cada objeto consulta será criada uma função cujo nome, parâmetro e cláusula SQL será retirado deste objeto.

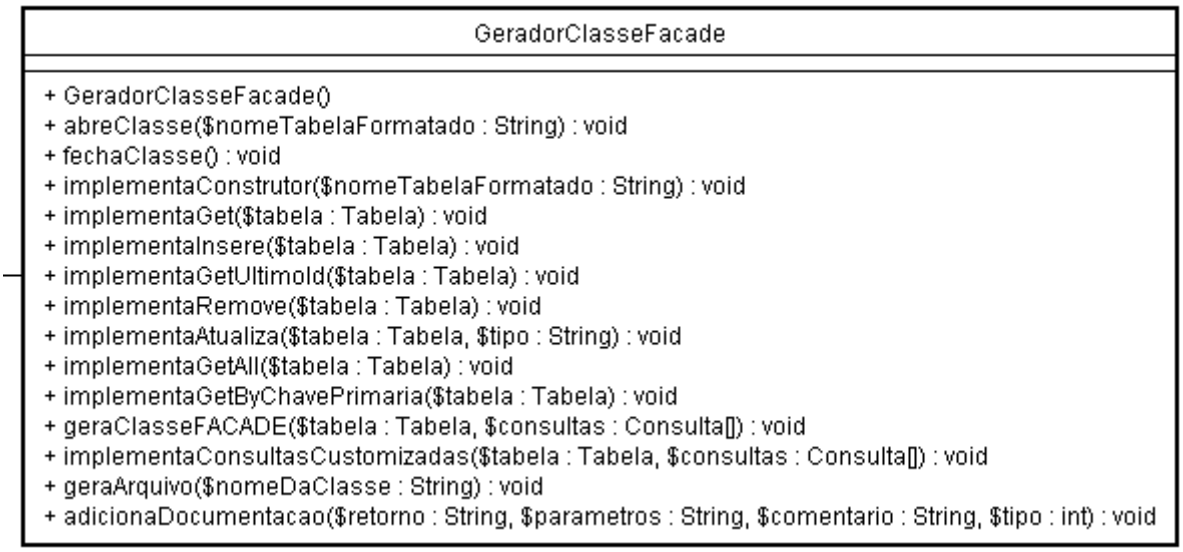

Figura 24 - Diagrama da Classe GeradorClasseFacade

4.8 Definição dos Formulários de Manipulação de Dados

Em nossa implementação construímos um gerador para os seguintes tipos de formulários de manipulação de dados:

4.8.1 Formulário de Inclusão

Este formulário mostra inicialmente um formulário HTML para entrada de dados e um script PHP que implementa a inclusão dos dados informados quando o formulário HTML é submetido.

4.8.2 Formulário de Exclusão

Este formulário mostra inicialmente uma tabela HTML contendo os registros de uma tabela no banco de dados para que o usuário escolha qual registro deseja excluir, chamaremos esta tabela de tabela de seleção de registros. Quando o usuário seleciona o registro a página é recarregada e um script em PHP faz a exclusão do registro, em seguida é mostrada novamente a tabela de seleção de registros atualizada.

4.8.3 Formulário de Alteração

Este formulário mostra inicialmente uma tabela de seleção de registros, quando o usuário seleciona um registro o formulário é recarregado e desta vez é mostrado um formulário HTML contendo os campos de entrada de dados já preenchidos com as informações contidas no registro selecionado. Quando este formulário é submetido, a página é novamente

recarregada, um script PHP faz a alteração do registro e em seguida é mostrada novamente a tabela de seleção de registros.

4.8.4 Formulário de Consulta

Este formulário mostra uma tabela de seleção de registros para que o usuário selecione o registro que será consultado, quando isto é feito o formulário é recarregado e é mostrada um tabela HTML contendo as informações do registro selecionado.

4.8.5 Formulário "Quatro em Um"

Este formulário mostra uma tabela de seleção de registros que contém no lado de cada registro as opções de exclusão, alteração e consulta para o registro relacionado. A opção de inclusão é visualizada no final da tabela. As funções de inclusão, exclusão, alteração e consulta são feitas da mesma forma dos outros formulários.

4.9 Gerador de Formulários

O gerador de formulários recebe um objeto Tabela, dois arrays de objetos Campo ($campos_visiveis e $campos_estrangeiros) e um objeto RequisicaoFormulario.

Do objeto Tabela o gerador extrai as informações necessárias para a criação das tabelas de seleção de registros, dos formulários HTML e das ações que serão executadas quando estes formulários forem submetidos. O array $campos_visiveis contém os Campos que deverão ser visualizados

na tabela de seleção de registros, o $campos_estrangeiros contém os campos das tabelas estrangeiras que deverão ser exibidos sempre que uma chave estrangeira for encontrada. O objeto RequisicaoFormulario informa ao gerador quais os tipos de formulários gerar.

Os principais métodos do gerador de formulários podem ser divididos da seguinte forma.

Métodos de Implementação de Formulários

Estes métodos montam a implementação dos formulários invocando os métodos de implementação de ações, da tabela de seleção de registros e de formulários HTML, de acordo com a estrutura do formulário solicitado. São os seguintes:

implementaFormularioInclusao($tabela, $campos_estrangeiros)

Recebe um objeto Tabela ($tabela) e um array de Campo ($campos_estrangeiros) que contém os campos das tabelas estrangeiras que deverão ser mostrados para cada chave estrangeira encontrada em $tabela.
Implementa o formulário de inclusão invocando os métodos implementaAcaoIncluir($tabela) e implementaFormularioHTMLInclusao($tabela, $campos_estrangeiros). A implementação é gravada em arquivo.

implementaFormularioExclusao(Tabela, Campo[], Campo[])

Recebe um objeto Tabela ($tabela) e dois arrays de Campo ($campos_visiveis e $campos_estrangeiros) que contém os campos que deverão ser visualizados na tabela de seleção de registros e o campos das tabelas estrangeiras que deverão ser mostrados para cada chave estrangeira encontrada em $tabela.

Implementa o formulário de exclusão invocando os métodos implementaAcaoExcluir($tabela) e implementaTabelaSelecaoRegistros($tabela, $campos_visiveis, $campos_estrangeiros, “excluir”). A implementação é gravada em arquivo.

implementaFormularioAlteracao(Tabela, Campo[], Campo[])

Recebe um objeto Tabela ($tabela) e dois arrays de Campo ($campos_visiveis e $campos_estrangeiros) que contém os campos que deverão ser visualizados na tabela de seleção de registros e os campos das tabelas estrangeiras que deverão ser mostrados para cada chave estrangeira encontrada em $tabela.

Implementa o formulário de exclusão invocando os métodos implementaAcaoAlterar($tabela, $campos_estrangeiros) e implementaTabelaSelecaoRegistros($tabela, $campos_visiveis, $campos_estrangeiros, “alterar”). A implementação é gravada em arquivo.

implementaFormularioConsulta(Tabela, Campo[], Campo[])

Recebe um objeto Tabela ($tabela) e dois arrays de Campo ($campos_visiveis e $campos_estrangeiros) que contém os campos que deverão ser visualizados na tabela de seleção de registros e os campos das tabelas estrangeiras que deverão ser mostrados para cada chave estrangeira encontrada em $tabela.

Implementa o formulário de consulta invocando os métodos implementaAcaoConsultar($tabela, $campos_estrangeiros) e implementaTabelaSelecaoRegistros($tabela, $campos_visiveis, $campos_estrangeiros, “consultar”). A implementação é gravada em arquivo.

implementaFormularioQuatroEmUm(Tabela, Campo[], Campo[])

Recebe um objeto Tabela ($tabela) e dois arrays de Campo ($campos_visiveis e $campos_estrangeiros) que contém os campos que deverão ser visualizados na tabela de seleção de registros e os campos das tabelas estrangeiras que deverão ser mostrados para cada chave estrangeira encontrada em $tabela.

Implementa o formulário “Quatro em Um” invocando os métodos: implementaAcaoIncluir($tabela), implementaAcaoExcluir($tabela), implementaAcaoAlterar($tabela), implementaAcaoConsultar($tabela), implementaFormularioHTMLInclusao($tabela,$campos_estrangeiros), implementaTabelaSelecaoRegistros($tabela,$campos_visiveis,

$campos_estrangeiros,“excluir”), implementaTabelaSelecaoRegistros($tabela,$campos_visiveis, $campos_estrangeiros,“alterar”) e implementaTabelaSelecaoRegistros($tabela,$campos_visiveis, $campos_estrangeiros,“consultar”). A implementação é gravada em arquivo.

Métodos de Implementação de Ações

A implementação dos métodos de implementação de ações leva em conta a existência de um objeto Facade já declarado nos formulários pela função abreFormulario(Tabela) e implementa as ações que serão executadas quando um formulário HTML de inclusão de dados for submetido ou quando o usuário seleciona um registro de uma tabela de seleção de registros.

implementaAcaoIncluir(Tabela)

Recebe um objeto Tabela ($tabela) e implementa as ações que serão executadas quando o formulário HTML de inclusão de dados for submetido. Essas ações são as seguintes:

- Criação de um objeto VO;
- Preenchimento deste objeto com os valores passados pelo formulário HTML;

- Invocação da função insere do objeto Facade já declarado no formulário passando como parâmetro o objeto VO preenchido;
- Impressão de mensagem de erro ao sucesso.

implementaAcaoExcluir(Tabela)

Recebe um objeto Tabela ($tabela) e implementa as ações que serão executadas quando o usuário selecionar um registro da tabela de seleção de registros. Essas ações são as seguintes:

- Criação de um objeto VO já preenchido com os valores do registro selecionado através da função getByCampo(param) do objeto Facade passando como parâmetro a chave primaria do registro;
- Invocação da função remove(obj) do objeto Facade já declarado no formulário passando como parâmetro o objeto VO preenchido;
- Impressão de mensagem de erro ao sucesso.

implementaAcaoAlterar(Tabela, Campo[])

Recebe um objeto Tabela ($tabela) e um array de Campo ($campos_estrangeiros). Implementa um formulário HTML de alteração de

dados e as ações que serão tomadas quando este formulário for submetido.

O formulário é gerado pegando cada Campo de $tabela e implementando o campo HTML específico de acordo com o tipo de dados do Campo. Se o Campo de $tabela for chave estrangeira então o Campo implementado será o Campo correspondente contido em $campos_estrangeiros.

As ações implementadas são as seguintes:

- Criação de um objeto VO já preenchido com os valores do registro selecionado através da função getByCampo(param) do objeto Facade passando como parâmetro a chave primaria do registro;
- Invocação da função atualizaCompleto(obj) do objeto Facade já declarado no formulário passando como parâmetro o objeto VO preenchido;
- Impressão de mensagem de erro ou sucesso.

implementaAcaoConsultar(Tabela, Campo[])

Recebe um objeto Tabela ($tabela) e implementa as ações que serão executadas quando o usuário selecionar um registro da tabela de seleção de registros. Essas ações são as seguintes:

- Criação de um objeto VO já preenchido com os valores do registro selecionado através da função getByCampo(param) do objeto Facade passando como parâmetro a chave primaria do registro;

- Impressão de um tabela HTML contendo os valores do registro selecionado

- Impressão de mensagem de erro (se for o caso).

implementaFormularioHTMLInclusao(Tabela, Campo[])

Recebe um objeto Tabela ($tabela) e um array de Campo ($campos_estrangeiros). O formulário é gerado pegando cada Campo de $tabela e implementando o campo HTML específico de acordo com o tipo de dados do Campo. Se o Campo de $tabela for chave estrangeira então o Campo implementado será o Campo correspondente contido em $campos_estrangeiros.

implementaTabelaSelecaoRegistros(Tabela, Campo[], Campo[], String)

Recebe um objeto Tabela ($tabela), dois arrays de Campo ($campos_visiveis e $campos_estrangeiros) e uma String ($tipo). Implementa uma tabela HTML que mostra para cada registro de $tabela os campos contidos em $campos_visiveis. Quando um destes campos for

chave estrangeira, o campo que será mostrado é o campo correspondente de $campos_estrangeiros.

Se o valor passado em $tipo for igual a “4em1”, será implementado ao lado de cada registro 3 links para as opções de exclusão, alteração e consulta e no final da tabela 1 link para inclusão de novo registro.

Caso contrário cada registro será um link para opção contida em $tipo, nesse caso os valores válidos para $tipo são “excluir”, “consultar” ou “alterar”.

abreFormulario(Tabela)

Além de escrever a tag de abertura do PHP “<?” esta função:

- escreve o requerimento da classe Conexao e da classe Facade relativa ao objeto Tabela passado como parâmetro;
- escreve a criação de um objeto da classe Facade relativa ao objeto Tabela passado como parâmetro.

5 Conclusão e Trabalhos Futuros

O desenvolvimento de aplicações para WEB possui características que se repetem para diferentes contextos, sendo talvez a necessidade de se armazenar e manipular informações em bases de dados a principal delas.

Neste trabalho foram apresentadas as características destas aplicações e abordados alguns problemas encontrados em seu desenvolvimento. Para estes problemas, elaboramos uma proposta de solução que consiste na implementação de um framework para persistência de dados fazendo uso da linguagem de programação PHP.

Os geradores que integram este framework proporcionam ao desenvolvedor economia de esforço e de tempo de desenvolvimento ao gerar automaticamente classes que utilizam os conceitos dos padrões de projeto Value Object e Facade e que desta forma, permitem o uso de OO em conjunto com bancos de dados relacionais de forma transparente. Isto proporciona maior clareza no entendimento do código facilitando sua manutenção. Adicionalmente, o desenvolvedor é poupado de uma tarefa repetitiva que é a geração de formulários para manipulação de dados. Todas essas características mostram o quanto a utilização de padrões de projeto pode ser útil na implementação de aplicações.

Para trabalhos futuros, propomos o desenvolvimento deste framework para a linguagem PHP versão 5, que trará novos recursos, principalmente no que diz respeito a OO e tratamento de exceções.

Referências

1. **A HISTÓRIA DO PHP E PROJETOS RELACIONADOS**. Disponível em: http://www.php.net/manual/pt_BR/history.php. Acesso em: 10/07/2004.

2. **ADODB DATABASE ABSTRACTION LIBRARY FOR PHP**. Disnponível em: http://adodb.sourceforge.net. Acesso em: 10/07/2004.

3. **ADODB LIBRARY FOR PHP MANUAL**. Disponível em: http://phplens.com/lens/adodb/docs-adodb.htm. Acesso em: 10/07/2004.

4. **PHP APPLICATION DEVELOPMENT WITH ADODB (PART 1)**. Disponível em: http://www.melonfire.com/community/columns/trog/article.php?id=142. Acesso em: 10/07/2004.

5. **PHP APPLICATION DEVELOPMENT WITH ADODB (PART 2)**. Disponível em: http://www.melonfire.com/community/columns/trog/article.php?id=144. Acesso em: 10/07/2004.

6. **ADODB DATABASE LIBRARY FOR PHP: CREATE PORTABLE DB APPS**. Disponível em: http://php.weblogs.com/adodb. Acesso em: 10/07/2004.

7. [Johnson 91] Johnson, Ralph E.; Russo, Vincent. ***Reusing Object-Oriented Designs***.

8. [Fayad 97] Fayad, M.E.; Schmidt, D.C. (eds) ***Object-Oriented Applications Frameworks***. Communications of the ACM, V. 40, nº 10, p. 32-38, 1997.

9 [FERLIN2004] Ferlin, Adriano **Modelo para Controle de concorrência para Objetos de Valor**

UM FRAMEWORK PARA PERSISTÊNCIA DE DADOS EM PHP

Alysson Marques

Marco Aurélio Silva

{marco}{alysson}@inf.ufsc.br

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

UNIVERSIDADE FEDERAL DE SANTA CATARINA

RESUMO

O desenvolvimento de aplicações para WEB possui características que se repetem para diferentes contextos, sendo talvez a necessidade de se armazenar e manipular informações em bases de dados a principal delas.

Este trabalho relata algumas dificuldades encontradas no desenvolvimento de aplicações em PHP com acesso a base de dados e apresenta um proposta para soluções destes problemas.

Esta proposta consiste na implementação de um framework de suporte persistência de dados fazendo uso da linguagem de programação PHP e os principais bancos de dados utilizados no mercado atual.

Este framework, consiste basicamente de três geradores, que irão gerar automaticamente classes que visam facilitar e acelerar o processo de desenvolvimento, seja com as funções de inserir, remover, atualizar e procurar objetos no banco de dados, seja com os formulários de acesso que serão gerados.

ABSTRACT

The development of applications for WEB have characteristics that repeat for different contexts, being the necessity of storing and manipulating information in databases the main one of them.

This work tells to some difficulties found in the development of applications in PHP with access to the database and presents a proposal for solutions of these problems.

This proposal consists of the implementation of one framework of support persistence of data making use of the programming language PHP and the main data bases used in the current market.

This framework, basically consists of three generators, that will automatically generate class that they aim at to facilitate and to speed up the development process, either with the functions to insert, to remove, to update and to look objects in the data base, either form of access that will be generated.

1 Definição das Classes Value Object (VO)

Uma classe VO é uma abstração de uma tabela do banco de dados. Ela possui apenas um atributo campos, que é um array que possui uma entrada para cada campo existente na tabela. Além disso, a classe VO possui, para cada campo, método GetByCampo que serve para recuperar o valor do atributo, e método setCampo, que serve para modificar o valor do atributo.

2 Gerador de classes VO

O gerador de classes VO recebe um objeto Tabela, que foi criado pelo Leitor de Banco de Dados. O gerador então utiliza as funções deste objeto para poder escrever em um arquivo a classe VO propriamente dita.

3 Definição das classes Facade

Uma classe Facade possui como atributo um objeto Conexao, e várias funções, sendo que algumas serão geradas por padrão e outras através de parâmetros pré-definidos pelo usuário.

3.1 Funções Geradas por Padrão

insere(objetoVO)

getByCampo(param1)

getByPK(chave)

atualiza(objetoVO)

atualizaCompleto(objetoVO)

remove(objetoVO)

3.2 Funções Customizadas pelo Usuário

Consulta XML - Esta classe tem a finalidade de dar suporte ao Gerador de Facades, consultando e gravando num arquivo XML. O sistema tem uma interface para que o usuário insira, altere ou apague as funções de uma tabela, e ela se encarregará de fazer as alterações no arquivo XML. É ela quem informa ao Gerador de Facade quais as funções que deverão ser implementadas, passando um array de objetos do tipo Funcao.

4 Gerador de classes Facade

Assim como o gerador de classes VO, o gerador de classes Facade também recebe um objeto Tabela do Leitor de Banco de Dados. O gerador então utiliza as funções deste objeto para poder escrever em um arquivo a classe Facade. Seus principais Métodos são:

Métodos que implementam os Get: Este métodos implementam um laço que pega o resultado da consulta ao banco de dados, coloca os valores de todos os campos retornados num objeto VO e em seguida adiciona o objeto VO num array que será o retorno da função.

implementaGet(Tabela) : Este método recupera todos os campos excluindo o campo que é chave primária. Para cada campo será implementado uma função getByCampo, que retornará um array de objetos VO.

implementaGetByChavePrimaria(Tabela): Este método recupera o campo chave primária e implementa a função getByChavePrimaria. Esta função é implementada separadamente dos outros get's pois ao invés de retornar um array de objetos VO retorna apenas um objeto VO, ou seja, retornará apenas o objeto que está no primeiro índice do array.

implementaGetAll(Tabela): Este método implementa a função que retorna um array com todos os objetos que existem no banco de dados.

Método implementaAtualiza(Tabela, Tipo): Este método implementa as funções atualiza e atualizaCompleto. Para gerar o cláusula SQL, que irá fazer o *update* dos dados no banco de dados, foi utilizado a função getUpdateSQL(ADORecordSet, Campos) do ADOdb. Para isso, primeiro é feita uma consulta ao banco de dados passando como parâmetro a chave primária do VO e que retornará um ADORecordSet. Em seguida a função getUpdateSQL verifica quais campos estão diferente no ADORecordSet e nos Campos do VO, e irá colocar na cláusula SQL todos os campos que estiverem diferentes. A diferença entre as funções atualiza e atualizaCompleto é que quando um atributo está com o valor nulo no VO e tem algum valor diferente de nulo no banco de dados, a função atualiza não atualizará o valor para nulo, enquanto que a função atualiza completo atualizará o valor no banco. Isto é feito com a definição da constante ADODB_FORCE_NULLS, fazendo que ela seja 1 para atualizar completo, e 0 para o atualiza.

Método implementaInsere(Tabela): Este método implementa a função insere. Similarmente ao método implementaAtualiza, faz uso de função do ADOdb getInsertSQL(NomeTabela,Campos). A função getInsertSQL é um pouco mais simples que a função getUpdateSQL, pois só precisa do nome da tabela e de um array com os campos que serão inseridos. Ela irar gerar uma cláusula SQL que irá inserir todos os campos que não forem nulos. Para gerar a chave primária do registro que será inserido, é chamada a função getUltimoId, que retorna o maior valor de chave primária da tabela, e incrementado o resultado em 1.

Método implementaRemove(Tabela): Este método implementa a função que irá remover o VO, ele simplesmente gera a cláusula SQL que irá apagar o registro do banco pela chave primária do VO.

Método implementaConsultasCustomizadas(Tabela, Consultas): Este método implementa as funções que foram feitas pelo usuário. Para cada objeto consulta será criada uma função cujo nome, parâmetro e cláusula SQL será retirado deste objeto.

5 Definição dos Formulários de Manipulação de Dados

Em nossa implementação construímos um gerador para os seguintes tipos de formulários de manipulação de dados:

5.1 Formulário de Inclusão

Este formulário mostra inicialmente um formulário HTML para entrada de dados e um script PHP que implementa a inclusão dos dados informados quando o formulário HTML é submetido.

5.2 Formulário de Exclusão

Este formulário mostra inicialmente uma tabela HTML contendo os registros de uma tabela no banco de dados para que o usuário escolha qual registro deseja excluir, chamaremos esta tabela de tabela de seleção de registros. Quando o usuário seleciona o registro a página é recarregada e um script em PHP faz a exclusão do registro, em seguida é mostrada novamente a tabela de seleção de registros atualizada.

5.3 Formulário de Alteração

Este formulário mostra inicialmente uma tabela de seleção de registros, quando o usuário seleciona um registro o formulário é recarregado e desta vez é mostrado um formulário HTML contendo os campos de entrada de dados já preenchidos com as informações contidas no registro selecionado. Quando este formulário é submetido, a página é novamente recarregada, um script PHP faz a alteração do registro e em seguida é mostrada novamente a tabela de seleção de registros.

5.4 Formulário de Consulta

Este formulário mostra uma tabela de seleção de registros para que o usuário selecione o registro que será consultado, quando isto é feito o formulário é recarregado e é mostrada um tabela HTML contendo as informações do registro selecionado.

5.5 Formulário "Quatro em Um"

Este formulário mostra uma tabela de seleção de registros que contém no lado de cada registro as opções de exclusão, alteração e consulta para o registro relacionado. A opção de inclusão é visualizada no final da tabela. As funções de inclusão, exclusão, alteração e consulta são feitas da mesma forma dos outros formulários.

6 Gerador de Formulários

O gerador de formulários recebe um objeto Tabela, dois arrays de objetos Campo ($campos_visiveis e $campos_estrangeiros) e um objeto RequisicaoFormulario.

Do objeto Tabela o gerador extrai as informações necessárias para a criação das tabelas de seleção de registros, dos formulários HTML e das ações que serão executadas quando estes formulários forem submetidos. O array $campos_visiveis contém os Campos que deverão ser visualizados na tabela de seleção de registros, o $campos_estrangeiros contém os campos das tabelas estrangeiras que deverão ser exibidos sempre que uma chave estrangeira for encontrada. O objeto RequisicaoFormulario informa ao gerador quais os tipos de formulários gerar.

7 Conclusão

O desenvolvimento de aplicações para WEB possui características que se repetem para diferentes contextos, sendo talvez a necessidade de se armazenar e manipular informações em bases de dados a principal delas.

Neste trabalho foram apresentadas as características destas aplicações e abordados alguns problemas encontrados em seu desenvolvimento. Para estes problemas, elaboramos uma proposta de solução que consiste na implementação de um framework para persistência de dados fazendo uso da linguagem de programação PHP.

Os geradores que integram este framework proporcionam ao desenvolvedor economia de esforço e de tempo de desenvolvimento ao gerar automaticamente classes que utilizam os conceitos dos padrões de projeto Value Object e Facade e que desta forma, permitem o uso de OO em conjunto com bancos de dados relacionais de forma transparente. Isto proporciona maior clareza no entendimento do código facilitando sua manutenção. Adicionalmente, o desenvolvedor é poupado de uma tarefa repetitiva que é a geração de formulários para manipulação de dados. Todas essas características mostram o quanto a utilização de padrões de projeto pode ser útil na implementação de aplicações.

Referências

1. **A HISTÓRIA DO PHP E PROJETOS RELACIONADOS**. Disponível em: http://www.php.net/manual/pt_BR/history.php. Acesso em: 10/07/2004.

2. **ADODB DATABASE ABSTRACTION LIBRARY FOR PHP**. Disnponível em: http://adodb.sourceforge.net. Acesso em: 10/07/2004.

3. **ADODB LIBRARY FOR PHP MANUAL**. Disponível em: http://phplens.com/lens/adodb/docs-adodb.htm. Acesso em: 10/07/2004.

4. **PHP APPLICATION DEVELOPMENT WITH ADODB (PART 1)**. Disponível em: http://www.melonfire.com/community/columns/trog/article.php?id=142. Acesso em: 10/07/2004.

5. **PHP APPLICATION DEVELOPMENT WITH ADODB (PART 2)**. Disponível em: http://www.melonfire.com/community/columns/trog/article.php?id=144. Acesso em: 10/07/2004.

6. **ADODB DATABASE LIBRARY FOR PHP: CREATE PORTABLE DB APPS**. Disponível em: http://php.weblogs.com/adodb. Acesso em: 10/07/2004.

7. [Johnson 91] Johnson, Ralph E.; Russo, Vincent. ***Reusing Object-Oriented Designs***.

8. [Fayad 97] Fayad, M.E.; Schmidt, D.C. (eds) ***Object-Oriented Applications Frameworks***. Communications of the ACM, V. 40, nº 10, p. 32-38, 1997.

Código Fonte

ARQUIVO: LeitorBancoDados.php

```
<?php

require_once "Tabela.php";
require_once "Campo.php";

class LeitorBancoDados{
	var $conexao;

	function LeitorBancoDados($conexao){
		$this->conexao= $conexao->conexao;
	}

	function executa($sql){
		return $this->conexao->Execute($sql);
	}

	function getNomesTabelas(){
		return $this->conexao->MetaTables('TABLES');
	}

	function getTodasTabelas(){
		return $this->getTabelas($this->getNomesTabelas());
	}

	function getTabelas($array){
		$tabelas = array();
		 foreach ($array AS $nomeTabela){
			$campos = array();
			$colunas = $this->conexao-
>MetaColumns($nomeTabela);
			if($colunas!=NULL){
				foreach ($colunas AS $objetoColuna){
					$campos[] = new Campo($objetoColuna-
>name,$objetoColuna->type,$objetoColuna->max_length);
				}
				$tabelas[] = new Tabela($nomeTabela,$campos);
			}
			}
			return $tabelas;
	}

	function getTabela($nome){
		$table =$this->getTabelas($tabela = array($nome));
		if (!sizeof($table)) return false;
```

```
            return $table[0];
        }

        function getNomesCamposByTabela($nome){
            $tabela = $this->getTabela($nome);
            return $tabela->getNomesCampos();
        }
}
```

ARQUIVO: ConsultaXml.php

```
<?php
require_once "Consulta.php";

class ConsultaXML {
        var $doc;
        var $root;
        var $arquivo;
        //construtor
        function ConsultaXml($arquivo){
            $this->doc = domxml_open_file(realpath($arquivo));
            $this->root=$this->doc->document_element();
            $this->arquivo=$arquivo;
    }

    function gravar() {
            $fp = fopen ($this->arquivo, "w");
            fwrite($fp, $this->doc->dump_mem());
            fclose($fp);
        }
        /**
        * adiciona uma tabela, caso ela nao exista
        */
        function adiciona_tabela($nomeTabela){
            $array = $this->doc-
>get_elements_by_tagname($nomeTabela);
            if(sizeof($array)==0){ //teste para ver se a tabela ja existe

                $Novatabela = $this->doc-
>create_element($nomeTabela);

                $funcoes=$this->doc->create_element("funcoes");
                $Novatabela->append_child($funcoes);
                $this->root->append_child($Novatabela);
            }
        }
```

```
        function remove_funcao($consulta,$tabela){
                if ($this->temFuncao($tabela,$consulta)){
                        //selecionando a tabela
                        $tables = $this->doc->get_elements_by_tagname($tabela);
                        $tabela = $tables[0];
                        //selecionando as funcoes
                        $funcoes= $tabela->get_elements_by_tagname("funcoes");
                        $funcoes=$funcoes[0];
                        $filhos = $funcoes->child_nodes();
                        //selecionar o maldito filho
                        foreach ($filhos as $filho){
                                if($filho->get_attribute("nome")==$consulta)
                                        $funcoes->remove_child($filho);
                        }
                        $this->gravar();
                }
        }

        function adiciona_funcao($consulta,$tabela){
                if (!$this->temFuncao($tabela,$consulta->getNome())){
                        if (!$this->temTabela($tabela))
                                $this->adiciona_tabela($tabela);
                        $nomeFuncao = $consulta->getNome();
                        $sql= $consulta->getSql();
                        $parametros= $consulta->getParametros();
                        //selecionando a tabela
                        $tables = $this->doc->get_elements_by_tagname($tabela);

                        $tabela = $tables[0];
                        //selecionando as funcoes
                        $funcoes= $tabela->get_elements_by_tagname("funcoes");
                        $funcao=$this->doc->create_element("funcao");
                        $funcao->set_attribute("nome",$nomeFuncao);
                        //criando o nodo dos parametros e adicionando no nodo da funcao
                        $params=$this->doc->create_element("parametros");
                        $funcao->append_child($params);
                        //criando e adicionando os parametros
                         foreach ($parametros AS $param){
                         $parametro = $this->doc->create_element("parametro");
                         $parametro->set_attribute("valor",$param);
                         $params->append_child($parametro);
```

```
                }
                //criando o nodo elemento sql;
                $cdata=$this->doc->create_cdata_section($sql);
                $sqltag = $this->doc->create_element("sql");
                $sqltag->append_child($cdata);
                $funcao->append_child($sqltag);

                //adicionando a funcao completa
                $funcoes[0]->append_child($funcao);
                $this->gravar();
            }
        }
        function mostraArvore(){
            echo htmlentities($this->doc->dump_mem()) ;
        }

        function print_a($array){
            echo "<pre>";
            print_r($array);
            echo "</pre>";
        }

        //seleciona uma tabela no documento
        function selecionaTabela($tabela){
            $tables = $this->doc->get_elements_by_tagname($tabela);
            if (sizeof($tables)>0)
            return $tables[0];
            else return false;
        }

        //retorna um array de consultas de uma tabela
        function getConsultasByTabela($tabela){
            $array = array();
            //selecionando a tabela
            $tabela= $this->selecionaTabela($tabela);
            if($tabela!=NULL){
            //selecionando as funcoes
            $funcoes= $tabela->get_elements_by_tagname("funcoes");
            $funcao = $funcoes[0]->get_elements_by_tagname("funcao");
             foreach ($funcao as $func){
                $nome = $func->get_attribute("nome");
                $sql = $func->get_elements_by_tagname("sql");
                $sql= $sql[0]->get_content();
                $parametros = $func->get_elements_by_tagname("parametros");
                $parametros = $parametros[0]->child_nodes ();
                $param = array();
```

```
                foreach($parametros as $par){
                    $param[] = $par->get_attribute("valor");
                }
                $consulta = new consulta($sql,$param,$nome);
                $array[] = $consulta;
             }
            return $array;
            }
            else return false;
      }
      //retorna um array com o nome de todas as tabelas que existem no
documento
      function getNomesTabelas(){
            $tabelas =$this->doc-
>get_elements_by_tagname("tabelas");

            $tabelas = $tabelas[0]->child_nodes();
            $array = array();
            foreach($tabelas as $tab){
                $array[]=$tab->node_name();
            }
            return $array;
      }

      //boolean retorna true se tem a tabela no documento
      function temTabela($tabela){
            return in_array($tabela,$this->getNomesTabelas());
      }
      //retorna um array com os nomes de todas as funcoes da tabela
      function getNomesFuncoes($tabela){
            if ($tabela = $this->selecionaTabela($tabela)){
            $array=array();
            $funcoes= $tabela->get_elements_by_tagname("funcoes");
            $funcao = $funcoes[0]-
>get_elements_by_tagname("funcao");
             foreach ($funcao as $func){
                $array[] = $func->get_attribute("nome");
            }
            return $array;
            }
            else return false;
      }

      function getFuncao($tabela,$nome){
            $consultas = $this->getConsultasByTabela($tabela);

             foreach ($consultas as $con){
```

```
                if ($nome==$con->getNome())
                return $con;
            }
            return NULL;
        }

        //boolean retorna se existe a funcao na tabela
        function temFuncao($tabela,$funcao) {
            $tem = false;
            if ($this->temTabela($tabela)){
                $tem = in_array($funcao,$this->getNomesFuncoes($tabela));
            }
            return $tem;
        }

}

?>
```

ARQUIVO: Consulta.php

```
<?php

class Consulta {

        var $sql;
        var $parametros;
        var $nome;

        function Consulta($sql,$parametros,$nome){
            $this->sql=$sql;
            $this->parametros=$parametros;
            $this->nome=$nome;
        }

        function getSql(){
            return $this->sql;
        }
        function getParametros(){
            return $this->parametros;
        }
        function getNome(){
            return $this->nome;
        }
```

```
}
```

ARQUIVO: Conexao.php

```
<?php

//require_once realpath("")."\adodb\adodb.inc.php";
//require_once "adodb/adodb.inc.php";
class Conexao{
	var $database;
	var $user;
	var $server;
	var $password;
	var $driver;
	var $conexao;
	var $conectado;

	function Conexao($database,$user,$server,$password,$driver){
		$this->conexao= ADONewConnection($driver);
		$this->user=$user;
		$this->server=$server;
		$this->password=$password;
		$this->database=$database;
		$this->driver=$driver;

		$this->conecta();
	}

	function conecta(){
		if ($this->conexao->Connect($this->server, $this->user,
$this->password, $this->database)){
		return true;
		}
		else
		return false;
	}

	function estaConectado(){
		return $this->conexao;
	}

	function executa($sql){
		return $this->conexao->Execute($sql);
	}

}
```

ARQUIVO: Campo.php

```
<?php
class Campo{
	var $nome;
	var $nomeFormatado;
	var $tipo;
	var $tamanho;
	var $chave_primaria;
	var $chave_estrangeira;
	var $tabela_estrangeira;

	function Campo($nome, $tipo,$tamanho){
		$this->nome=$nome;
		$this->tipo=$tipo;
		$this->chave_primaria=FALSE;
		$this->chave_estrangeira=FALSE;
		$this->tabela_estrangeira=NULL;
		$this->tamanho=$tamanho;
		$this->nomeFormatado=$this->formateNome($nome);
	}

	function getNome(){
		return $this->nome;
	}

	function getNomeFormatado(){
		return $this->nomeFormatado;
	}

	function getTipo(){
		return $this->tipo;
	}
	function setChavePrimaria(){
		$this->chave_primaria=TRUE;
		$this->chave_estrangeira=FALSE;
		$this->tabela_estrangeira=NULL;
	}

	function setChaveEstrangeira($tabela) {
		$this->chave_primaria=FALSE;
		$this->chave_estrangeira=TRUE;
		$this->tabela_estrangeira=$tabela;
	}

	function ehChavePrimaria(){
```

```
            return $this->chave_primaria;
        }

        function ehChaveEstrangeira(){
            return $this->chave_estrangeira;
        }

        function getTabelaEstrangeira(){
            return $this->tabela_estrangeira;
        }

        function formateNome($nome) {
            $pedaco= explode("_", $nome);
            $novo_nome= "";
            for ($i= 0; $i < sizeof($pedaco); $i++)
                $novo_nome.= substr_replace($pedaco[$i],
strtoupper(substr($pedaco[$i], 0, 1)), 0, 1);
            return $novo_nome;
        }


}

?>
```

ARQUIVO: Escritor.php

```
<?php

class Escritor {

        var $doc;

        function Escritor () {
            $this->doc= "";
        }

        function escreva($texto, $nTab) {
            $this->facaTabulacao($nTab);
            $this->doc.= $texto;
        }

        function facaTabulacao($nTab) {
            for ($i= 0; $i < $nTab; $i++) {
                $this->doc.= "\t";
            }
```

```
	}

	function escreva1($texto){
		$this->facaTabulacao(1);
		$this->doc.= $texto . "\n\r";
	}

	function escreva2($texto){
		$this->facaTabulacao(2);
		$this->doc.= $texto . "\n\r";
	}

	function escreva3($texto){
		$this->facaTabulacao(3);
		$this->doc.= $texto . "\n\r";
	}

	function esc_nl($texto, $nTab) {
		$this->doc.= "\n";
		$this->facaTabulacao($nTab);
		$this->doc.= $texto;
	}

	function reseta() {
		$this->doc= "";
	}

}

?>
```

ARQUIVO: Tabela.php

```
<?php
require_once "Campo.php";
class Tabela{
	var $nome;
	var $nomeFormatado;
	var $campos;

	function Tabela($nome, $campos){
		$this->nome=$nome;
		$this->campos=$this->setChaves($campos);
		$this->nomeFormatado=$this->formateNome($nome);
	}
```

```
function setChaves($campos){
	$camposN =array();
	foreach ($campos as $campo){
	$sufix=substr($campo->nome,0,3);
		if("id_"== $sufix){
		$table=substr($campo->nome,3);
			if ($this->nome==$table){
				$campo->setChavePrimaria();

			}
			else
				$campo->setChaveEstrangeira($table);
		}
		$camposN[]=$campo;
	}
	return $camposN;
}

function getCampoChavePrimaria(){
	foreach($this->campos as $campo)
		if ($campo->ehChavePrimaria())
			return $campo;
}

function getNomeCampoChavePrimaria(){
$campo = $this->getCampoChavePrimaria();
	return $campo->getNome();
}

function getNomeFormatadoCampoChavePrimaria(){
	$campo =$this->getCampoChavePrimaria();
	return $campo->getNomeFormatado();
}

function getNome(){
	return $this->nome;
}

function getNomeFormatado(){
	return $this->nomeFormatado;
}

function getCamposChaveEstrangeira(){
	$tables = array();
	foreach($this->campos as $campo)
		if ($campo->ehChaveEstrangeira())
			$tables[] = $campo;
	return $tables;
```

```
	}

	function getCampos(){
		return $this->campos;
	}

	function getCampo($nomeCampo){
		foreach ($this->campos as $campo)
			if ($campo->getNome()==$nomeCampo)
				return $campo;

		return false;
	}

	function getCamposSemChavePrimaria(){
		$campos = array();
		foreach ($this->campos as $campo)
			if (!$campo->ehChavePrimaria())
				$campos[]=$campo;

		return $campos;
	}

	function getNomesCampos(){
		$nomes= array();
		foreach ($this->campos as $campo){
			$nomes[]=$campo->getNome();
		}
		return $nomes;
	}

	function formateNome($nome) {
		$pedaco= explode("_", $nome);
		$novo_nome= "";
		for ($i= 0; $i < sizeof($pedaco); $i++)
			$novo_nome.= substr_replace($pedaco[$i],
strtoupper(substr($pedaco[$i], 0, 1)), 0, 1);
		return $novo_nome;
	}

}

?>
```

ARQUIVO: RequisicaoFormulario.php

```php
<?php

class RequisicaoFormulario {

	var $inclusao;
	var $exclusao;
	var $alteracao;
	var $consulta;
	var $quatroEmUm;

	function TipoFormulario() {
		$this->inclusao= false;
		$this->exclusao= false;
		$this->alteracao= false;
		$this->consulta= false;
		$this->quatroEmUm= false;
		return $this;
	}

	function setInclusao($v) {
		$this->inclusao= $v;
	}

	function setExclusao($v) {
		$this->exclusao= $v;
	}

	function setAlteracao($v) {
		$this->alteracao= $v;
	}

	function setConsulta($v) {
		$this->consulta= $v;
	}

	function setQuatroEmUm($v) {
		$this->quatroEmUm= $v;
	}

	function getInclusao() {
		return $this->inclusao;
	}

	function getExclusao() {
		return $this->exclusao;
	}

	function getAlteracao() {
```

```
            return $this->alteracao;
        }

        function getConsulta() {
            return $this->consulta;
        }

        function getQuatroEmUm() {
            return $this->quatroEmUm;
        }
}

?>
```

ARQUIVO: GeradorClasseFacade.php

```
<?php
require_once "Escritor.php";
require_once "Consulta.php";

class GeradorCLasseFacade{

        var $escritor;

        function GeradorClasseFacade(){
            $this->escritor= new Escritor();
        }

        /**
        * @return void
        * @param String $nomeTabelaFormatado
        * @desc Escreve o começo da classe
        */
        function abreClasse($nomeTabelaFormatado){
            $this->escritor->escreva("<?php \n\n", 0);
            $this->escritor->escreva("require_once
\"$nomeTabelaFormatado"."VO.php\";\n\n", 0);
            $this->escritor->escreva("class
$nomeTabelaFormatado"."Facade {\n\n", 0);
            $this->escritor->escreva("\n\r", 0);
            $this->escritor->escreva1("var \$msgErro;");
            $this->escritor->escreva1("var \$con;\n");
        }

        /**
        * @return void
```

```
	* @desc Escreve o fim da classe
	*/
	function fechaClasse(){
		$this->escritor->escreva("}\n\n", 0);
		$this->escritor->escreva("?>", 0);
	}

	/**
	* @return void
	* @param String $nomeTabelaFormatado
	* @desc Escreve o construtor da função
	*/
	function implementaConstrutor($nomeTabelaFormatado){
		$this->adicionaDocumentacao("$nomeTabelaFormatado"."Facade",array('Conexao','$con'),"Construtor");
		$this->escritor->escreva1("function $nomeTabelaFormatado"."Facade(\$con) {");
		$this->escritor->escreva2("\$this->msgErro=\"\";");
		$this->escritor->escreva2("\$this->con= \$con;");
		$this->escritor->escreva1("}");
		$this->escritor->escreva("\n\r", 0);
	}

	/**
	* @return void
	* @desc Escreve a função getErro
	*/
	function implementaGetErro(){
		$this->adicionaDocumentacao("String MsgErro",array("",""),"Retorna a mensagem de erro");
		$this->escritor->escreva1("function getErro() {");
		$this->escritor->escreva2("return \$this->msgErro;");
		$this->escritor->escreva1("}");
	}

	/**
	* @return void
	* @param Tabela $tabela
	* @desc Escreve as funções getByCampo para todos os campos da tabela, exceto a getByChavePrimaria
	*/
	function implementaGet($tabela){
		$campos = $tabela->getCampos();
		$camposSemPK = $tabela->getCamposSemChavePrimaria();
		$nomeTabela = $tabela->getNome();
		$nomeTabelaFormatado = $tabela->getNomeFormatado();
		foreach ($camposSemPK as $campo){
```

```
                $nomeFormatado=$campo->getNomeFormatado();
                $nome = $campo->getNome();
                $tipo = $campo->getTipo();
                $this->adicionaDocumentacao("array
$nomeTabelaFormatado"."VO",array($tipo,'$busca'),"Método que busca
por $nomeFormatado");
                $this->escritor->escreva1("function
getBy$nomeFormatado(\$busca) {\n\r", 1);
                $this->escritor->escreva2("\$resource= \$this->con-
>Executa(\"select * from $nomeTabela where $nome = '\$busca'\");");
                $this->escritor->escreva2("if (!\$resource) {");
                $this->escritor->escreva3("\$this->msgErro=\$this-
>con->conexao->errorMsg();");
                $this->escritor->escreva3("return false;");
                $this->escritor->escreva2("}");
                $this->escritor->escreva2("\$valores= \$resource-
>GetArray();");
                $this->escritor->escreva2("\$num_rows=
sizeof(\$valores);");
                $this->escritor->escreva2("\$obj= '';");
                $this->escritor->escreva2("for (\$i= 0; \$i <
\$num_rows; \$i++) {");
                $this->escritor->escreva3("\$valor= \$valores[\$i];");
                $this->escritor->escreva3("\$obj[\$i]= new
$nomeTabelaFormatado"."VO();");
                foreach ($campos as $campo){
                    $nomeFormatado=$campo-
>getNomeFormatado();
                    $nome = $campo->getNome();
                    $this->escritor->escreva3("\$obj[\$i]-
>set$nomeFormatado(\$valor[\"$nome\"]);");
                }
                $this->escritor->escreva3("}");
                $this->escritor->escreva2("return \$obj;");
                $this->escritor->escreva1("}\n");
            }
        }

        /**
        * @return void
        * @param Tabela $tabela
        * @desc Escreve a função que insere o objeto
        */
        function implementaInsere($tabela){
            $nomeTabela = $tabela->getNome();
            $nomeTabelaF = $tabela->getNomeFormatado() . "VO";
            $campoPK = $tabela->getCampoChavePrimaria();
            $nomePKF = $campoPK->getNomeFormatado();
```

```
            $this->adicionaDocumentacao("int
$nomePKF",array($nomeTabelaF,'$obj'),"Método que insere um
$nomeTabelaF");
            $this->escritor->escreva1("function insere(\$obj) {");
            $this->escritor->escreva2("\$id=\$this->getUltimoId()+1;");
            $this->escritor->escreva2("\$obj->set$nomePKF"."(\$id);");
            $this->escritor-
>escreva2("\$nomeTabela=\"$nomeTabela\";");
            $this->escritor->escreva2("\$sql= \$this->con->conexao-
>GetInsertSQL(&\$nomeTabela,\$obj->campos);");
            $this->escritor->escreva2("\$res= \$this->con-
>Executa(\$sql);");
            $this->escritor->escreva2("if(\$res) return \$id;");
            $this->escritor->escreva2("else {");
            $this->escritor->escreva3("\$this->msgErro=\$this->con-
>conexao->errorMsg();");
            $this->escritor->escreva3("return FALSE;");
            $this->escritor->escreva2("}");
            $this->escritor->escreva1("}\n");

      }

      /**
      * @return void
      * @param Tabela $tabela
      * @desc Escreve o método que retorna o ultimo ID inserido na
tabela
      */
      function implementaGetUltimoId($tabela){
            $nomeTabela=$tabela->getNome();
            $chavePK=$tabela->getCampoChavePrimaria();
            $nomePK=$chavePK->getNome();
            $this->adicionaDocumentacao("int
UltimoId",array("",""),"Método que retorna o ultimo id inserido");
            $this->escritor->escreva1("function getUltimoId(){");
            $this->escritor->escreva2("\$id= \$this->con->conexao-
>GetCol(\"select max($nomePK".") from $nomeTabela\");");
            $this->escritor->escreva3("if (sizeof(\$id)<1) return 0;");
            $this->escritor->escreva2("return \$id[0];");
            $this->escritor->escreva1("}\n");
      }

      /**
      * @return void
      * @param Tabela $tabela
      * @desc Escreve a função que remove o objeto
      */
      function implementaRemove($tabela){
```

```
            //implementacao da funcao del
            $nomeTabela = $tabela->getNome();
            $nomeTabelaF = $tabela->getNomeFormatado() . "VO";
            $campoPK = $tabela->getCampoChavePrimaria();
            $pkF= $campoPK->getNomeFormatado();
            $pk = $campoPK->getNome();
            $this->adicionaDocumentacao("boolean",array($nomeTabelaF,'$obj'),"Método que remove um $nomeTabelaF");
            $this->escritor->escreva1("function remove(\$obj) {");
            $this->escritor->escreva2("\$sql= \"delete from $nomeTabela where $pk = \".\$obj->get$pkF();");
            $this->escritor->escreva2("if(\$this->con->Executa(\$sql))");
            $this->escritor->escreva2("return true; else {");
            $this->escritor->escreva2("\$this->msgErro=\$this->con->conexao->errorMsg();");
            $this->escritor->escreva2("return false;}");
            $this->escritor->escreva1("}\n");


      }

      /**
      * @return void
      * @param Tabela $tabela
      * @desc Escreve as funções Atualiza e AtualizaCompleto
      */
      function implementaAtualiza($tabela,$tipo=""){
            $nomeTabela = $tabela->getNome();
            $nomeTabelaF = $tabela->getNomeFormatado() . "VO";
            $nomePk = $tabela->getNomeCampoChavePrimaria();
            $campoPK=$tabela->getCampo($nomePk);
            $this->adicionaDocumentacao("boolean",array($nomeTabelaF,'$obj'),"Método que atualiza um $nomeTabelaF");
            $this->escritor->escreva1("function atualiza$tipo"."(\$obj) {");
            $this->escritor->escreva2("\$id= \$obj->get".$campoPK->getNomeFormatado()."();");
            $this->escritor->escreva2("\$rs = \$this->con->Executa(\"select * from $nomeTabela where $nomePk = \$id\");");
            if ($tipo=="Completo")
            $this->escritor->escreva2("define('ADODB_FORCE_NULLS',1);");
            $this->escritor->escreva2("\$sql = \$this->con->conexao->GetUpdateSQL(&\$rs,\$obj->campos);");
            $this->escritor->escreva2("\$resource= \$this->con->Executa(\$sql);");
```

```
            $this->escritor->escreva2("if(\$resource) return true; else
{");
            $this->escritor->escreva2("\$this->msgErro=\$this->con-
>conexao->errorMsg();");
            $this->escritor->escreva2("return false;}");
            if ($tipo=="Completo")
            $this->escritor-
>escreva2("define('ADODB_FORCE_NULLS',0);");
            $this->escritor->escreva1("}\n");


      }

      /**
      * @return void
      * @param Tabela $tabela
      * @desc Escreve a função que insere o objeto
      */
      function implementaGetAll($tabela){

            $campos = $tabela->getCampos();
            $nomeTabela = $tabela->getNome();
            $nomeTabelaFormatado = $tabela->getNomeFormatado() .
"VO";
            $this->adicionaDocumentacao("array
$nomeTabelaFormatado",array($nomeTabelaFormatado,'$obj'),"Método
que busca todos os $nomeTabelaFormatado");
            $this->escritor->escreva1("function getAll() {");
            $this->escritor->escreva2("\$obj= array();");
            $this->escritor->escreva2("\$resource= \$this->con-
>Executa(\"select * from $nomeTabela \");");
            $this->escritor->escreva2("if (!\$resource) {");
            $this->escritor->escreva3("\$this->msgErro=\$this->con-
>conexao->errorMsg();");
            $this->escritor->escreva3("return false;");
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("\$valores= \$resource-
>GetArray();");
            $this->escritor->escreva2("\$num_rows=
sizeof(\$valores);");
            $this->escritor->escreva2("for (\$i= 0; \$i < \$num_rows;
\$i++) {");
            $this->escritor->escreva2("\$valor= \$valores[\$i];");
            $this->escritor->escreva3("\$obj[\$i]= new
$nomeTabelaFormatado"."();");
            foreach ($campos as $campo){
                  $nomeFormatado=$campo->getNomeFormatado();
                  $nome = $campo->getNome();
```

```
                $this->escritor->escreva3("\$obj[\$i]-
>set$nomeFormatado(\$valor[\"$nome\"]);");
            }
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("return \$obj;");
            $this->escritor->escreva1("}\n");

        }

        /**
        * @return void
        * @param Tabela $tabela
        * @desc Escreve a função getByChavePrimaria
        */
        function implementaGetByChavePrimaria($tabela){

            $campos = $tabela->getCampos();
            $nomeTabela = $tabela->getNome();
            $campoPK=$tabela->getCampoChavePrimaria();
            $chavePK=$campoPK->getNome();
            $chavePKF=$campoPK->getNomeFormatado();
            $nomeTabelaFormatado = $tabela->getNomeFormatado()
."VO";


            $this->adicionaDocumentacao("Object
$nomeTabelaFormatado",array($nomeTabelaFormatado,'$obj'),"Método
que busca um objeto $nomeTabelaFormatado pela chave primaria");
            $this->escritor->escreva1("function
getBy$chavePKF"."(\$busca) {");
            $this->escritor->escreva2("\$resource= \$this->con-
>Executa(\"select * from $nomeTabela where
$chavePK=\$busca\");");
            $this->escritor->escreva2("if (!\$resource) {");
            $this->escritor->escreva3("\$this->msgErro=\$this->con-
>conexao->errorMsg();");
            $this->escritor->escreva3("return false;");
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("\$valores= \$resource-
>GetArray();");
            $this->escritor->escreva2("\$num_rows=
sizeof(\$valores);");
            $this->escritor->escreva2("if(\$num_rows) {");
//          $this->escritor->escreva2("\$obj= '';");
//          $this->escritor->escreva2("for (\$i= 0; \$i < \$num_rows;
\$i++) {");
            $this->escritor->escreva3("\$valor= \$valores[0];");
```

```
            $this->escritor->escreva3("\$obj= new
$nomeTabelaFormatado"."();");
            foreach ($campos as $campo){
                  $nomeFormatado=$campo->getNomeFormatado();
                  $nome = $campo->getNome();
                  $this->escritor->escreva3("\$obj-
>set$nomeFormatado(\$valor[\"$nome\"]);");
            }
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("if(\$num_rows) return \$obj; else
return false;");
            $this->escritor->escreva1("}\n");
      }
      /**
      * @return void
      * @param Tabela $tabela
      * @param Consulta[] $consultas
      * @desc Método que chama todas as funções necessárias para gerar
a classeFacade
      */
      function geraClasseFACADE($tabela,$consultas) {
            $this->escritor->reseta();
            $this->abreClasse($tabela->getNomeFormatado());
            $this->implementaConstrutor($tabela-
>getNomeFormatado());
            $this->implementaInsere($tabela);
            $this->implementaRemove($tabela);
            $this->implementaAtualiza($tabela);
            $this->implementaAtualiza($tabela,"Completo");
            $this->implementaGetByChavePrimaria($tabela);
            $this->implementaGetAll($tabela);
            $this->implementaGet($tabela);
            $this-
>implementaConsultasCustomizadas($tabela,$consultas);
            $this->implementaGetUltimoId($tabela);
            $this->implementaGetErro();
            $this->fechaClasse();
            $this->geraArquivo($tabela->getNomeFormatado());

      }


      /**
      * @return void
      * @param Tabela $tabela
      * @param Consulta[] $consultas
      * @desc Escreve todas as funções customizadas pelo usuário
      */
```

```
    function implementaConsultasCustomizadas($tabela,$consultas){
        $nomeTabelaFormatado=$tabela->getNomeFormatado();
        $campos=$tabela->getCampos();
        foreach ($consultas as $consulta){
            $nomeConsulta= $consulta->getNome();
            $parametros = $consulta->getParametros();
            $sql=$consulta->getSql();
            $temp="function $nomeConsulta(";
            $i=0;
            foreach($parametros as $parametro){
                $temp.="$$parametro";
                if ($i < (sizeof($parametros)-1))
                $temp.= ",";
                $i++;
            }
            $temp.="){";
            $this->escritor->escreva1($temp);
            $this->escritor->escreva2("\$resource= \$this->con-
>Executa(\" $sql \");");
            $this->escritor->escreva2("if (!\$resource) {");
            $this->escritor->escreva3("\$this->msgErro=\$this-
>con->conexao->errorMsg();");
            $this->escritor->escreva3("return false;");
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("\$valores= \$resource-
>GetArray();");
            $this->escritor->escreva2("\$num_rows=
sizeof(\$valores);");
            $this->escritor->escreva2("for (\$i= 0; \$i <
\$num_rows; \$i++) {");
            $this->escritor->escreva3("\$valor= \$valores[\$i];");
            $this->escritor->escreva3("\$obj[\$i]= new
$nomeTabelaFormatado"."VO();");
            foreach ($campos as $campo){
                $nomeFormatado=$campo-
>getNomeFormatado();
                $nome = $campo->getNome();
                $this->escritor->escreva3("\$obj[\$i]-
>set$nomeFormatado(\$valor[\"$nome\"]);");
            }
            $this->escritor->escreva2("}");
            $this->escritor->escreva2("return \$obj;");
            $this->escritor->escreva1("}\n");
        }
    }

    /**
    * @return void
```

```
	* @param String $nomeDaClasse
	* @desc Grava o arquivo
	*/
	function geraArquivo($nomeDaClasse) {
		$fp = fopen
("arquivosGerados/$nomeDaClasse"."Facade.php", "w");
		fwrite($fp,$this->escritor->doc);
		fclose($fp);
		echo "Arquivo $nomeDaClasse"."Facade.php
gerado.\n<br>";
	}

	/**
	* @return void
	* @param String $retorno
	* @param Array $parametros
	* @param String $comentario
	* @param int $tipo
	* @desc Adiciona a documentação nas funções criadas.
	*/
	function
adicionaDocumentacao($retorno,$parametros,$comentario,$tipo=0){
		$this->escritor->escreva("/**\n",1);
		$this->escritor->escreva("* @return $retorno\n",1);
		if ($tipo)
		foreach ($parametros as $param)
		$this->escritor->escreva("* @param $param[0]
$param[1]\n",1);
		else
		$this->escritor->escreva("* @param $parametros[0]
$parametros[1]\n",1);
		$this->escritor->escreva("* @desc $comentario\n",1);
		$this->escritor->escreva("*/\n",1);
	}


}

?>
```

ARQUIVO: GeradorClasseVO.php

```
<?php
require_once "Escritor.php";
require_once "Tabela.php";
```

```
class GeradorClasseVO{

	var $escritor;

	function GeradorClasseVO() {
		$this->escritor= new Escritor();


	}

	function geraArquivo($nomeDaClasse) {
		$fp = fopen ("arquivosGerados/$nomeDaClasse"."VO.php",
"w");
		fwrite($fp,$this->escritor->doc);
		fclose($fp);
		echo "Arquivo $nomeDaClasse"."VO.php gerado.\n<br>";
	}

//retorna um string com a implementacao da classeVO
	function geraClasseVO($tabela) {

		$this->abreClasse($tabela->getNomeFormatado());
		$campos = $tabela->getCampos();
		$this->declaraVariaveis();
		$this->implementaConstrutor($tabela-
>getNomesCampos(),$tabela->getNomeFormatado());
		$this->escritor->escreva("\n\r", 0);
		$this->implementaFuncoesGet($campos);
		$this->escritor->escreva("\n\r", 0);
		$this->implementaFuncoesSet($campos);

		$this->escritor->escreva("\n\r", 0);
		$this->fechaClasse();
		$this->geraArquivo($tabela->getNomeFormatado());
		return $this->escritor->doc;

	}

	function abreClasse($nome){
		$this->escritor->doc= "";
		$this->escritor->escreva("<?php \n\n",0);
		$this->escritor->escreva("class $nome"."VO {\n\n",0);

	}

	function fechaClasse(){
		$this->escritor->escreva("}\n\n", 1);
		$this->escritor->escreva("?>", 0);
```

```
	}
	function declaraVariaveis(){
		$this->escritor->escreva("var \$campos;\r", 1);
	}


	function implementaConstrutor($campos,$nome){
		$this->escritor->escreva("function $nome"."VO(){\r", 1);

		$this->escritor->escreva("\$this->campos= array();\r", 2);

		foreach ($campos as $campo){
			$this->escritor->escreva("\$this-
>campos['$campo']=NULL;\r", 2);
		}
		$this->escritor->escreva("}\r", 1);
	}
	function implementaFuncoesSet($campos){
		foreach ($campos as $campo){
			$campo1 = $campo->getNomeFormatado();
			$campo2 = $campo->getNome();
			$this->escritor->escreva("function
set$campo1($$campo1) {\r", 1);
			$this->escritor->escreva(" \$this-
>campos['$campo2']=$$campo1;\r", 2);
			$this->escritor->escreva("}\n\n\r", 1);
		}
	}

	function implementaFuncoesGet($campos){
		foreach ($campos as $campo){
			$campo1 = $campo->getNomeFormatado();
			$campo2 = $campo->getNome();
			$this->escritor->escreva("function get$campo1() {\r",
1);
			$this->escritor->escreva("return \$this-
>campos['$campo2'];\r", 2);
			$this->escritor->escreva("}\n\n\r", 1);
		}
	}
}

?>
```

ARQUIVO: GeradorFormularios.php

```
<?php

require_once "funcoes.php";
require_once "config.php";
require_once "classes/Escritor.php";

class GeradorFormularios {

	var $escritor;
	var $_colunas_textarea= 70;
	var $_linhas_textarea= 3;
	var $leitor;

	function GeradorFormularios($leitor) {
		$this->escritor= new Escritor();
		$this->leitor= $leitor;
	}

	function abreFormulario($tabela) {
		$this->escritor->doc= "";
		$this->escritor->escreva("<?php\n", 0);
		$this->escritor->esc_nl("require_once \"conexao.inc.php\";",
0);
		$this->escritor->esc_nl("require_once \"".$tabela-
>getNomeFormatado()."Facade.php\";", 0);
		$this->escritor->esc_nl("\n\$".$tabela->getNome()."Facade=
new ".$tabela->getNomeFormatado()."Facade(\$con);", 0);
	}

	function fechaFormulario() {
		$this->escritor->esc_nl("?>", 0);
	}

	function salvaFormulario($arquivo) {
		$fp = fopen ($arquivo, "w");
		fwrite($fp,$this->escritor->doc);
		fclose($fp);
		echo "Arquivo $arquivo gerado.\n<br>";
	}

	function geraFormulario($tabela, $campos_visiveis,
$campos_estrangeiros, $requisicao_formulario) {
		$this->abreFormulario($tabela);
		if($requisicao_formulario->getInclusao()) $this-
>implementaFormularioInclusao($tabela, $campos_estrangeiros);
		if($requisicao_formulario->getExclusao()) $this-
>implementaFormularioExclusao($tabela, $campos_visiveis,
$campos_estrangeiros);
```

```
            if($requisicao_formulario->getAlteracao()) $this->implementaFormularioAlteracao($tabela, $campos_visiveis, $campos_estrangeiros);
            if($requisicao_formulario->getConsulta()) $this->implementaFormularioConsulta($tabela, $campos_visiveis, $campos_estrangeiros);
            if($requisicao_formulario->getQuatroEmUm()) $this->implementaFormularioQuatroEmUm($tabela, $campos_visiveis, $campos_estrangeiros);
        }

        function implementaFormularioInclusao($tabela, $campos_estrangeiros) {
            $this->abreFormulario($tabela);
            $this->implementaGetPOST();
            $this->implementaGetREQUEST();
            $this->escritor->esc_nl("\$_mostraFormInc= true;", 0);
            $this->implementaAcaoIncluir($tabela);
            $this->implementaFormularioHTMLInclusao($tabela, $campos_estrangeiros);
            $this->fechaFormulario();
            $this->salvaFormulario("arquivosGerados/formInclusao".$tabela->getNomeFormatado().".php");
        }

        function implementaFormularioExclusao($tabela, $campos_visiveis, $campos_estrangeiros) {
            $this->abreFormulario($tabela);
            $this->implementaGetPOST();
            $this->implementaGetREQUEST();
            $this->implementaAcaoExcluir($tabela);
            $this->implementaTabelaSelecaoRegistros($tabela, $campos_visiveis, $campos_estrangeiros, "excluir");
            $this->fechaFormulario();
            $this->salvaFormulario("arquivosGerados/formExclusao".$tabela->getNomeFormatado().".php");
        }

        function implementaFormularioAlteracao($tabela, $campos_visiveis, $campos_estrangeiros) {
            $this->abreFormulario($tabela);
            $this->implementaGetPOST();
            $this->implementaGetREQUEST();
            $this->implementaAcaoAlterar($tabela, $campos_estrangeiros);
```

```
            $this->implementaTabelaSelecaoRegistros($tabela,
$campos_visiveis, $campos_estrangeiros, "alterar");
            $this->fechaFormulario();
            $this-
>salvaFormulario("arquivosGerados/formAlteracao".$tabela-
>getNomeFormatado().".php");
      }

      function implementaFormularioConsulta($tabela, $campos_visiveis,
$campos_estrangeiros) {
            $this->abreFormulario($tabela);
            $this->implementaGetPOST();
            $this->implementaGetREQUEST();
            $this->implementaAcaoConsultar($tabela,
$campos_estrangeiros);
            $this->implementaTabelaSelecaoRegistros($tabela,
$campos_visiveis, $campos_estrangeiros, "consultar");
            $this->fechaFormulario();
            $this-
>salvaFormulario("arquivosGerados/formConsulta".$tabela-
>getNomeFormatado().".php");
      }

      function implementaFormularioQuatroEmUm($tabela,
$campos_visiveis, $campos_estrangeiros) {
            $this->abreFormulario($tabela);
            $this->implementaGetPOST();
            $this->implementaGetREQUEST();
            $this->escritor->esc_nl("if(!isset(\$_mostraFormInc))
\$_mostraFormInc= false;", 0);
            $this->implementaAcaoIncluir($tabela);
            $this->implementaFormularioHTMLInclusao($tabela,
$campos_estrangeiros);
            $this->implementaAcaoExcluir($tabela);
            $this->implementaAcaoConsultar($tabela,
$campos_estrangeiros);
            $this->implementaAcaoAlterar($tabela,
$campos_estrangeiros);
            $this->implementaTabelaSelecaoRegistros($tabela,
$campos_visiveis, $campos_estrangeiros, "4em1");
            $this->fechaFormulario();
            $this->salvaFormulario("arquivosGerados/form4em1".$tabela-
>getNomeFormatado().".php");
      }

      function implementaAcaoIncluir($tabela) {
            $this->escritor->esc_nl("\n//=====>>> INICIO DA ACAO
INCLUIR", 0);
```

```
            $this->escritor->esc_nl("if(isset(\$acao) && (\$acao
=='incluir')) {", 0);
            $this->escritor->esc_nl("if(\$_POST) {", 1);
            $this->escritor->esc_nl("\$registroVO= new ".$tabela-
>getNomeFormatado()."VO();", 2);
            $campos= $tabela->getCampos();
            foreach($campos as $campo)
                  $this->escritor->esc_nl("\$registroVO->set".$campo-
>getNomeFormatado()."(\$".$campo->getNome().");", 2);
            $this->escritor->esc_nl("if(\$".$tabela->getNome()."Facade-
>insere(\$registroVO))", 2);
            $this->escritor->esc_nl("echo \"Registro inserido com
sucesso!\";", 3);
            $this->escritor->esc_nl("else {", 2);
            $this->escritor->esc_nl("echo \"Erro ao inserir
registro!<br>\";", 3);
            $this->escritor->esc_nl("echo \"(\".\$".$tabela-
>getNome()."Facade->getErro().\")\";", 3);
            $this->escritor->esc_nl("}", 2);
            $this->escritor->esc_nl("echo \"<p><a
href=\\\"\".\$_SERVER['PHP_SELF'].\"\\\">Voltar</a></p>\";", 2);
            $this->escritor->esc_nl("\$_mostraFormInc= false;", 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//===>>> FIM DA ACAO
INCLUIR\n", 0);
      }

      function implementaFormularioHTMLInclusao($tabela,
$campos_estrangeiros) {
            $this->escritor->esc_nl("//===>>> INICIO DO FORMULARIO
INCLUSAO", 0);
            $this->escritor->esc_nl("if (\$_mostraFormInc) {", 0);
            $this->escritor->esc_nl("?>", 0);
            $this->escritor->esc_nl("<form method=\"post\"
action=\"<?php echo \"\$PHP_SELF\"; ?>\">\n", 0);
            $ce= 0;
            $campos= $tabela->getCampos();
            foreach($campos AS $campo) {
                  if($campo->ehChaveEstrangeira()) {
                        $campoE= $campos_estrangeiros[$ce];
                        $tabelaE= $this->leitor->getTabela($campo-
>tabela_estrangeira);
                        $pK_TE= $tabelaE->getCampoChavePrimaria();
                        $ce++;
                        $this->escritor->esc_nl("<p>", 0);
                        $this->escritor->esc_nl(formateNome($campo-
>tabela_estrangeira).":<br>", 1);
```

```
                        $this->escritor->esc_nl("<select
name=\"".$campo->getNome()."\">", 1);
                        $this->escritor->esc_nl("<?", 1);
                        $this->escritor->esc_nl("require_once
\"".$tabelaE->getNomeFormatado()."Facade.php\";", 1);
                        $this->escritor->esc_nl("\$".$tabelaE-
>getNome()."Facade= new ".$tabelaE-
>getNomeFormatado()."Facade(\$con);", 1);
                        $this->escritor->esc_nl("\$opts= \$".$tabelaE-
>getNome()."Facade->getAll();", 1);
                        $this->escritor->esc_nl("foreach(\$opts as \$opt)
{", 1);
                        $this->escritor->esc_nl("echo \"<option
value=\\\"\".\$opt->get".$pK_TE-
>getNomeFormatado()."().\"\\\">\".\$opt->get".$campoE-
>getNomeFormatado()."().\"</option>\";", 2);
                        $this->escritor->esc_nl("}", 1);
                        $this->escritor->esc_nl("?>", 1);
                        $this->escritor->esc_nl("</select>", 1);
                        $this->escritor->esc_nl("</p>", 0);
                }
                else {
                        $this->escritor->escreva("<p>\n", 0);
                        $this->escritor->escreva(formateNome($campo-
>getNome()).":<br>\n", 1);
                        if(substr($campo->tipo, 0, 3) == "set") $tipo=
"set";
                        else $tipo= $campo->tipo;
                        switch($tipo) {
                                case "set":
                                        $this->escritor->escreva($this-
>getSelect($campo, 1), 0);
                                        break;
                                default:
                                        if($campo->tamanho > $this-
>_colunas_textarea)
                                                $this->escritor->escreva($this-
>getTextArea($campo), 1);
                                        else
                                                $this->escritor->escreva($this-
>getInputText($campo)."\n", 1);
                                        break;
                        }
                        $this->escritor->esc_nl("</p>", 0);
                }
        }
        $this->escritor->esc_nl("<p>".$this->getButton("submit",
"Inserir Registro")."</p>", 0);
```

```
            $this->escritor->esc_nl("<p>".$this->getHidden("acao",
"incluir")."</p>", 0);
            $this->escritor->esc_nl("</form>\n", 0);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//===>>> FIM DO FORMULARIO
INCLUSAO\n", 0);
      }

      function implementaAcaoExcluir($tabela) {
            $chave_primaria= $tabela->getCampoChavePrimaria();
//chave primaria
            $cpf= $chave_primaria->getNomeFormatado();
            $this->escritor->esc_nl("\n//===>>> INICIO DA ACAO
EXCLUIR", 0);
            $this->escritor->esc_nl("if(isset(\$acao) && (\$acao
=='excluir') && isset(\$id)) {", 0);
            $this->escritor->esc_nl("\$registro= \$".$tabela-
>getNome()."Facade->getBy$cpf(\$id);", 1);
            $this->escritor->esc_nl("if(\$".$tabela->getNome()."Facade-
>remove(\$registro))", 1);
            $this->escritor->esc_nl("echo \"Registro excluído com
sucesso!\";", 2);
            $this->escritor->esc_nl("else {", 1);
            $this->escritor->esc_nl("echo \"Erro ao excluir registro!\";",
2);
            $this->escritor->esc_nl("echo \"(\".\$".$tabela-
>getNome()."Facade->getErro().\")\";", 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("echo \"<p><a
href=\\\"\".\$_SERVER['PHP_SELF'].\"\\\">Voltar</a></p>\";", 1);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//===>>> FIM DA ACAO EXCLUIR",
0);
      }

      function implementaAcaoConsultar($tabela, $campos_estrangeiros)
{
            $chave_primaria= $tabela->getCampoChavePrimaria();
//chave primaria
            $cpf= $chave_primaria->getNomeFormatado();
            $campos= $tabela->getCampos();
            $this->escritor->esc_nl("\n//===>>> INICIO DA ACAO
CONSULTAR", 0);
            $this->escritor->esc_nl("if(isset(\$acao) && (\$acao
=='consultar') && isset(\$id)) {", 0);
            $this->escritor->esc_nl("\$registro= \$".$tabela-
>getNome()."Facade->getBy$cpf(\$id);", 1);
```

```
            $this->escritor->esc_nl("if(\$registro) {", 1);
            $ce= 0;
            foreach($campos as $campo) {
                  if($campo->chave_estrangeira) {
                        $campoE= $campos_estrangeiros[$ce];
                        $ce++;
                        $this->escritor->esc_nl("require_once
\"".formateNome($campo->tabela_estrangeira)."Facade.php\";", 2);
                        $this->escritor-
>esc_nl("\$".formateNome($campo->tabela_estrangeira)."Facade= new
".formateNome($campo->tabela_estrangeira)."Facade(\$con);", 2);
                        $this->escritor->esc_nl("\$regVO=
\$".formateNome($campo->tabela_estrangeira)."Facade-
>getBy".$campo->getNomeFormatado()."(\$registro->get".$campo-
>getNomeFormatado()."());", 2);
                        $this->escritor->esc_nl("echo
\"<p>".formateNome($campo->tabela_estrangeira).":<br>\";", 2);
                        $this->escritor->esc_nl("if(\$regVO) echo
\$regVO->get".$campoE->getNomeFormatado()."(); else echo \$registro-
>get".$campo->getNomeFormatado()."().\"</p>\";", 2);
                        }
                  else {
                        $this->escritor->esc_nl("echo \"<p>".$campo-
>getNomeFormatado().":<br>\";", 2);
                        $this->escritor->esc_nl("echo \$registro-
>get".$campo->getNomeFormatado()."().\"</p>\";", 2);
                  }
            }
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("else {", 1);
            $this->escritor->esc_nl("echo \"Erro ao excluir registro!\";",
2);
            $this->escritor->esc_nl("echo \"(\".\$".$tabela-
>getNome()."Facade->getErro().\")\";", 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("echo \"<p><a
href=\\\"\".\$_SERVER['PHP_SELF'].\"\\\">Voltar</a></p>\";", 1);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//===>>> FIM DA ACAO
CONSULTAR", 0);
      }

      function implementaAcaoAlterar($tabela, $campos_estrangeiros) {
            $chave_primaria= $tabela->getCampoChavePrimaria();
//chave primaria
            $cpf= $chave_primaria->getNomeFormatado();
            $campos= $tabela->getCampos();
```

```
            $this->escritor->esc_nl("\n//=====>>> INICIO DA ACAO
ALTERAR", 0);
            $this->escritor->esc_nl("if(isset(\$acao) && (\$acao
=='alterar') && isset(\$id)) {", 0);
            $this->escritor->esc_nl("\$registro= \$".$tabela-
>getNome()."Facade->getBy$cpf(\$id);", 1);
            $this->escritor->esc_nl("if(\$_POST) {", 1);
            foreach($campos as $campo) {
                  $this->escritor->esc_nl("\$registro->set".$campo-
>getNomeFormatado()."(\$".$campo->getNome().");", 2);

            }
            $this->escritor->esc_nl("if(\$".$tabela->getNome()."Facade-
>atualizaCompleto(\$registro))", 2);
            $this->escritor->esc_nl("echo \"Registro alterado com
sucesso!<br>\";", 3);
            $this->escritor->esc_nl("else {", 2);
            $this->escritor->esc_nl("echo \"Erro ao excluir registro!\";",
3);
            $this->escritor->esc_nl("echo \"(\".\$".$tabela-
>getNome()."Facade->getErro().\")\";", 3);
            $this->escritor->esc_nl("}", 2);
            $this->escritor->esc_nl("echo \"<p><a
href=\\\"\".\$_SERVER['PHP_SELF'].\"\\\">Voltar</a></p>\";", 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("else {", 1);
            $this->implementaFormularioHTMLAlteracao($tabela,
$campos_estrangeiros, 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//=====>>> FIM DA ACAO ALTERAR",
0);
      }

      function implementaFormularioHTMLAlteracao($tabela,
$campos_estrangeiros, $tab) {
            $chave_primaria= $tabela->getCampoChavePrimaria();
//chave primaria
            $cpf= $chave_primaria->getNomeFormatado();
            $campos= $tabela->getCampos();
            $this->escritor->esc_nl("//=====>>> INICIO DO FORMULARIO
ALTERACAO", $tab);
            $this->escritor->esc_nl("?>", $tab);
            $this->escritor->esc_nl("<form method=\"post\"
action=\"\">\n", $tab);
            $ce= 0;
            foreach($campos AS $campo) {
                  if($campo->ehChaveEstrangeira()) {
```

```
                        $campoE= $campos_estrangeiros[$ce];
                        $tabelaE= $this->leitor->getTabela($campo-
>tabela_estrangeira);
                        $pK_TE= $tabelaE->getCampoChavePrimaria();
                        $ce++;
                        $this->escritor->esc_nl("<p>", 0);
                        $this->escritor->esc_nl(formateNome($campo-
>tabela_estrangeira).":<br>", 1);
                        $this->escritor->esc_nl("<select
name=\"".$campo->getNome()."\">", 1);
                        $this->escritor->esc_nl("<?", 1);
                        $this->escritor->esc_nl("require_once
\"".$tabelaE->getNomeFormatado()."Facade.php\";", 1);
                        $this->escritor->esc_nl("\$".$tabelaE-
>getNome()."Facade= new ".$tabelaE-
>getNomeFormatado()."Facade(\$con);", 1);
                        $this->escritor->esc_nl("\$opts= \$".$tabelaE-
>getNome()."Facade->getAll();", 1);
                        $this->escritor->esc_nl("foreach(\$opts as \$opt)
{", 1);
                        $this->escritor->esc_nl("echo \"<option
value=\\\"\".\$opt->get".$pK_TE->getNomeFormatado()."().\"\\\"\";
if(\$opt->get".$pK_TE->getNomeFormatado()."() == \$registro-
>get".$campo->getNomeFormatado()."()) echo \" selected\"; echo
\">\".\$opt->get".$campoE->getNomeFormatado()."().\"</option>\";",
2);
                        $this->escritor->esc_nl("}", 1);
                        $this->escritor->esc_nl("?>", 1);
                        $this->escritor->esc_nl("</select>", 1);
                        $this->escritor->esc_nl("</p>", 0);
                }
                else {
                        $this->escritor->escreva("<p>\n", ($tab+1));
                        $this->escritor->escreva(formateNome($campo-
>getNome()).":<br>", ($tab+2));
                        if(substr($campo->tipo, 0, 3) == "set") $tipo=
"set";
                        else $tipo= $campo->tipo;
                        switch($tipo) {
                                case "set":
                                        $opcoes= getOpcoesSet($campo-
>tipo);
                                        $this->escritor->esc_nl("<select
name=\"".$campo->getNome()."\">\n", ($tab+2));
                                        foreach($opcoes AS $op) {
                                                $this->escritor->esc_nl("<option
value=\"$op\"<? if(\$registro->get".$campo->getNomeFormatado()."()
== \"$op\") echo \" selected\";?>>$op</option>\n", ($tab+3));
```

```
                                        }
                                    $this->escritor->esc_nl("</select>\n",
($tab+2));
                                    break;
                                default:
                                    if($campo->tamanho > $this-
>_colunas_textarea)
                                        $this->escritor-
>esc_nl("<textarea name=\"".$campo->getNome()."\" cols=\"".$this-
>_colunas_textarea."\" rows=\"".$this->_linhas_textarea."\" value=\"<?
echo \$registro->get".$campo->getNomeFormatado()."();
?>\"></textarea>", ($tab+2));
                                    else
                                        $this->escritor->esc_nl("<input
type=\"text\" name=\"".$campo->getNome()."\" maxlength=\"".$campo-
>tamanho."\" value=\"<? echo \$registro->get".$campo-
>getNomeFormatado()."(); ?>\">", ($tab+2));
                                    break;
                            }
                        }
                        $this->escritor->esc_nl("</p>", ($tab+1));
                }
                $this->escritor->esc_nl("<p>".$this->getButton("submit",
"Alterar Registro")."</p>", ($tab+1));
                $this->escritor->esc_nl("<p>".$this->getHidden("acao",
"alterar")."</p>", ($tab+1));
                $this->escritor->esc_nl("<p>".$this->getHidden("id", "<?
echo \$id; ?>")."</p>", ($tab+1));
                $this->escritor->esc_nl("</form>\n", ($tab+1));
                $this->escritor->esc_nl("<?php", $tab);
                $this->escritor->esc_nl("//=====>>> FIM DO FORMULARIO
ALTERACAO\n", $tab);
        }


        function implementaTabelaSelecaoRegistros($tabela,
$campos_visiveis, $campos_estrangeiros, $tipo) {
                $colspan= sizeof($campos_visiveis);
                $chave_primaria= $tabela->getCampoChavePrimaria();
//chave primaria
                $cpf= $chave_primaria->getNomeFormatado();
                $this->escritor->esc_nl("\n//=====>>> INICIO DA TABELA
SELECAO REGISTROS", 0);
                $this->escritor->esc_nl("if(!isset(\$acao)) {", 0);
                $this->escritor->esc_nl("\$registros= \$".$tabela-
>getNome()."Facade->getALL();", 0);
                $this->escritor->esc_nl("?>", 0);
```

```
            $this->escritor->esc_nl("<table border=1 cellspacing=2>",
0);
            $aux= "<tr><td colspan=$colspan>Campos</td>";
            if($tipo == "4em1")    $aux.= "<td colspan=3>Ações</td>";
            $aux.= "<tr>";
            $this->escritor->esc_nl($aux, 1);
            $this->escritor->esc_nl("<tr>", 1);
            foreach($campos_visiveis as $campo_visivel)
                  if($campo_visivel->ehChaveEstrangeira())
                        $this->escritor-
>esc_nl("<td>".formateNome($campo_visivel-
>tabela_estrangeira)."</td>", 2);
                  else
                        $this->escritor->esc_nl("<td>".$campo_visivel-
>getNomeFormatado()."</td>", 2);
            if($tipo == "4em1")
                  $this->escritor-
>esc_nl("<td>Editar</td>\n\t\t<td>Excluir</td>\n\t\t<td>Consultar</t
d>", 2);
            $this->escritor->esc_nl("</tr>", 1);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("if(sizeof(\$registros)) {", 0);
            $this->escritor->esc_nl("?>", 0);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("foreach(\$registros as \$registro) {",
1);
            $this->escritor->esc_nl("\$id= \$registro->get$cpf();", 2);
            $this->escritor->esc_nl("?>", 0);
            $this->escritor->esc_nl("<tr>", 2);
            $ce= 0;
            foreach($campos_visiveis as $campo_visivel) {
                  if($campo_visivel->chave_estrangeira) {
                        $campoE= $campos_estrangeiros[$ce];
                        $ce++;
                        $this->escritor->esc_nl("<?php", 0);
                        $this->escritor->esc_nl("require_once
\"".formateNome($campo_visivel->tabela_estrangeira)."Facade.php\";",
2);
                        $this->escritor-
>esc_nl("\$".formateNome($campo_visivel-
>tabela_estrangeira)."Facade= new ".formateNome($campo_visivel-
>tabela_estrangeira)."Facade(\$con);", 2);
                        $this->escritor->esc_nl("\$regVO=
\$".formateNome($campo_visivel->tabela_estrangeira)."Facade-
>getBy".$campo_visivel->getNomeFormatado()."(\$registro-
>get".$campo_visivel->getNomeFormatado()."());", 2);
                        $this->escritor->esc_nl("?>", 0);
                        if($tipo == "4em1")
```

```
                        $this->escritor->esc_nl("<td><?php
if(\$regVO) echo \$regVO->get".$campoE->getNomeFormatado()."();
else echo \$registro->get".$campo_visivel-
>getNomeFormatado()."();?></td>", 2);
                    else
                        $this->escritor->esc_nl("<td><a
href=\"?acao=$tipo&id=<? echo \$id; ?>\"><?php if(\$regVO) echo
\$regVO->get".$campoE->getNomeFormatado()."(); else echo \$registro-
>get".$campo_visivel->getNomeFormatado()."();?></a></td>", 2);
                    }
                else
                    if($tipo == "4em1")
                        $this->escritor->esc_nl("<td><?php echo
\$registro->get".$campo_visivel->getNomeFormatado()."(); ?></td>",
3);
                    else
                        $this->escritor->esc_nl("<td><a
href=\"?acao=$tipo&id=<? echo \$id; ?>\"><?php echo \$registro-
>get".$campo_visivel->getNomeFormatado()."(); ?></a></td>", 3);
                    }
            if($tipo == "4em1") {
                $this->escritor->esc_nl("<td align=\"center\"><a
href=\"<?php echo \$PHP_SELF.\"?id=\".\$id;
?>&acao=alterar\">ALT</a></td>", 3);
                $this->escritor->esc_nl("<td align=\"center\"><a
href=\"<?php echo \$PHP_SELF.\"?id=\".\$id;
?>&acao=excluir\">EXC</a></td>", 3);
                $this->escritor->esc_nl("<td align=\"center\"><a
href=\"<?php echo \$PHP_SELF.\"?id=\".\$id;
?>&acao=consultar\">CON</a></td>", 3);
            }
            $this->escritor->esc_nl("</tr>", 2);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("}", 2);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("else {", 0);
            $this->escritor->esc_nl("?>", 0);
            if($tipo == "4em1")    $aux= $colspan+3; else $aux=
$colspan;
            $this->escritor->esc_nl("<tr><td colspan=$colspan>Esta
tabela está vazia.</td></tr>", 1);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("}", 1);
            $this->escritor->esc_nl("?>", 0);
            if($tipo == "4em1")
                $this->escritor->esc_nl("<tr><td
colspan=".($colspan+3)."><a href=\"<?php echo \$PHP_SELF;
```

```
?>?acao=incluir&_mostraFormInc=true\">Incluir
Registro</a></td></tr>", 1);
            $this->escritor->esc_nl("</table>", 0);
            $this->escritor->esc_nl("<?php", 0);
            $this->escritor->esc_nl("}", 0);
            $this->escritor->esc_nl("//===>>> FIM DA TABELA
SELECAO REGISTROS\n", 0);
            }

      function implementaGetPOST() {
            $this->escritor->esc_nl("if(\$_POST) {", 0);
            $this->escritor->esc_nl("\$keysPOST=
array_keys(\$_POST);", 1);
            $this->escritor->esc_nl("foreach(\$keysPOST as
\$keyPOST)", 1);
            $this->escritor->esc_nl("\${\$keyPOST}=
\$_POST[\$keyPOST];", 2);
            $this->escritor->esc_nl("}", 1);
      }

      function implementaGetREQUEST() {
            $this->escritor->esc_nl("if(\$_REQUEST) {", 0);
            $this->escritor->esc_nl("\$keysREQUEST=
array_keys(\$_REQUEST);", 1);
            $this->escritor->esc_nl("foreach(\$keysREQUEST as
\$keyREQUEST)", 1);
            $this->escritor->esc_nl("\${\$keyREQUEST}=
\$_REQUEST[\$keyREQUEST];", 2);
            $this->escritor->esc_nl("}", 1);
      }

      function getInputText($campo) {
            return "<input type=\"text\" name=\"".$campo-
>getNome()."\" maxlength=\"".$campo->tamanho."\">";
            }

      function getButton($tipo, $rotulo) {
            return "<input type=\"$tipo\" name=\"submit\"
value=\"$rotulo\">";
            }

      function getHidden($nome, $valor) {
            return "<input type=\"hidden\" name=\"$nome\"
value=\"$valor\">";
            }

      function getTextArea($campo) {
```

```
            return "<textarea name=\"".$campo->getNome()."\"
cols=\"".$this->_colunas_textarea."\" rows=\"".$this-
>_linhas_textarea."\"></textarea>";
            }
      function getSelect($campo, $tab) {
            $e= new Escritor();
            $opcoes= getOpcoesSet($campo->tipo);
            $e->escreva("<select name=\"".$campo-
>getNome()."\">\n", $tab);
            foreach($opcoes AS $op)
                  $e->escreva("<option
value=\"$op\">$op</option>\n", $tab+1);
            $e->escreva("</select>\n", $tab);
            return $e->doc;
            }

      }


?>
```