DESENVOLVIMENTO DE UMA FERRAMENTA ASSISTENTE PARA CRIAÇÃO DE APLICAÇÕES CRUD EM JAVA NA WEB

Carlos Renato de Souza Perri, Francisco Assis da Silva, Leandro Luiz de Almeida

Faculdade de Informática – Universidade do Oeste Paulista (UNOESTE) – Presidente Prudente – SP – Brasil.
carlosperri@gmail.com, chico, llalmeida}@unoeste.br

RESUMO

Devido à necessidade de informatização dos processos de negócio, o armazenamento de informações relevantes em Banco de Dados e a disponibilização desses dados na rede mundial de computadores, este projeto propõe o desenvolvimento de uma ferramenta geradora de aplicações para Web escritas em Java que construa cadastros e movimentações para realização de operações CRUD (Create, Retrieve, Update, Delete), ou seja, armazenamento, busca, atualização e deleção. Esta ferramenta é um software, a partir do qual, o programador insere o script de criação de um Banco de Dados e, ao definir parâmetros na ferramenta, são gerados os códigos-fonte das aplicações em Java para Web. Existe a necessidade de se produzir aplicações Web em Java com baixo tempo de produção, pois construir essas aplicações utilizando métodos usuais de desenvolvimento de software demanda-se certo período de tempo. A implementação desta ferramenta vem, também, para mudar esse conceito e a forma de desenvolvimento Web em Java, pois ela servirá como um assistente, facilitando a produção das aplicações Java Web. As aplicações geradas utilizam as tecnologias Servlets e JSPs, o framework Hibernate e também a biblioteca Javascript jQuery.
Palavras-chaves: Aplicações Web; Servlets; JSP; Hibernate; jQuery; CRUD.

DEVELOPING AN ASSISTANT TOOL TO CREATE CRUD APPLICATIONS IN JAVA WEB

ABSTRACT

Due to the need for computerization of business processes, storage of relevant information in databases and making these information available on Internet, this project proposes to develop a tool for generating Web applications written in Java, that build functionalities to perform CRUD (Create, Retrieve, Update, Delete), ie, storage, read, update and deletion. The software is a tool, from which, the programmer inserts the script to create a database and, after setting the parameters into the tool, source code of Java Web applications are generated. There is a need to create Web applications in Java with low production time, because building these applications using common methods for development takes much time. The implementation of this tool is also to change that concept and the way of developing Java Web applications, because this tool will be used as an assistant, becoming easier to create Java Web applications. The generated applications use the technologies Servlets and JSPs, the Hibernate framework and the jQuery JavaScript library.
Keywords: WEB applications; Servlets;JSP; Hibernate; jQuery; CRUD.

1. INTRODUÇÃO

Nos dias de hoje, torna-se cada vez maior o uso da Internet, sendo esta utilizada para diversos fins, que variam desde o lazer até a manipulação de dados empresariais.

Devido a essa crescente necessidade da informatização de processos de negócios, bem como a disponibilização dessas informações na rede mundial de computadores, nota-se a ausência de uma ferramenta geradora de aplicações Web feitas em Java, para que o processo de desenvolvimento dessas aplicações acompanhe a demanda existente.

Desenvolver aplicações Java para Web com as tecnologias existentes é uma tarefa bastante custosa e, para muitos, é também complexa. O projeto de desenvolvimento dessa ferramenta geradora de aplicações Java Web permitirá desenvolver aplicações mais rapidamente, trazendo uma nova maneira de programação, sem que o programador necessite escrever linhas de código. Segundo a CodeGeneration [CodeGeneration 2009], com o uso de ferramentas geradoras de código-fonte, o esforço gasto no desenvolvimento de aplicações Web em Java é minimizado, além de livrar o programador da árdua tarefa de escrever código redundante diversas vezes. As aplicações que a ferramenta gera são padronizadas, trazendo como consequência, a diminuição da chance de erros comparando-se à escrita de programas linha a linha de código.

A ferramenta utiliza, nas aplicações geradas, as tecnologias Servlets e JSPs, Hibernate e jQuery. Essas aplicações seguem o padrão *Model View Controller* (MVC).

2. A FERRAMENTA

A ferramenta deste trabalho é um software projetado capaz de gerar o código-fonte de aplicações Java Web, para realização de operações CRUD (*Create*, *Retrieve*, *Update*, *Delete*), ou seja, armazenamento, busca, atualização e deleção, assim como a exibição de informações na Web.

Este tipo de software presta apoio às atividades da Engenharia de Software, sendo classificada como uma ferramenta CASE (do inglês *Computer-Aided Software Engineering* - Engenharia de Software Auxiliada por Computador). Esta classificação abrange todas as ferramentas baseadas em computadores que auxiliam atividades de engenharia de software, desde análise de requisitos e modelagem até programação e testes. Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software

Esta ferramenta encaixa-se na categoria dos geradores de código-fonte baseados em *templates*, presentes em ambiente de produção de software como opção para a geração automática e massiva de aplicações, seja de maneira completa ou parcial, permitindo com que os programadores possam criar aplicações mais facilmente e rapidamente, com muito menos esforço comparando-se à forma habitual de programar, já que podem viabilizar a geração de código de uma aplicação a partir de *templates* e parâmetros de entrada. Com o uso deste software tem-se um código padronizado e com menos possibilidades de erros, visto que o código gerado se baseia em um *template* utilizado e também na filosofia de trabalho da ferramenta. Tais aspectos estão fortemente relacionados à qualidade do software: o produto final é uma aplicação em cujo código-fonte se percebe um estilo único de programação.

Segundo a CodeGenaration [CodeGenaration 2009] (que é um recurso para engenheiros de software e arquitetos de interesse na geração de código – comunidade constituída por artigos, entrevistas e um banco de dados de

ferramentas de geração) ao se utilizar de geradores de código têm-se as seguintes vantagens: a) padrão de qualidade: o código feito a mão tem sua qualidade variando durante o ciclo de vida de um projeto: pode começar no alto e decair ou vice e versa, não é constante. Contrariamente, o código gerado por uma ferramenta assistente aumenta sua qualidade com o tempo, pois os erros encontrados podem ser uniformemente reparados na base do código, já que pode-se lançar novas versões da ferramenta; b) a familiaridade com o gerador por parte do programador também ajuda bastante, visto que dependendo dos seus conhecimentos, ele mesmo pode fazer reparos ou personalizações na ferramenta (quando esta é código aberto) e não somente no código gerado; c) consistência: as aplicações geradas são consistentes na estrutura de classes e nomeação de variáveis; d) produtividade: os geradores constroem o código numa fração muita curta de tempo, economizando horas de trabalho (muitas vezes repetitivo). Isso libera o desenvolvedor para outras tarefas que exijam soluções mais criativas. e) abstração: os geradores frequentemente fornecem uma camada de abstração entre o projeto e a base do código, elevando-se o nível de abstração e facilitando o manuseio das regras de negócio.

A idéia da ferramenta apresentada neste artigo consiste em inserir o *script* de definição de Banco de Dados e por meio da parametrização da ferramenta (escolha dos *templates* – cadastros e/ou movimentações) gerar os códigos-fontes dessas aplicações. Esses códigos-fontes são JSPs e Servlets, que fazem uso de arquivos de mapeamento e classes de Banco de Dados.

Considere o modelo Entidade-Relacionamento, ilustrado na Figura 1.

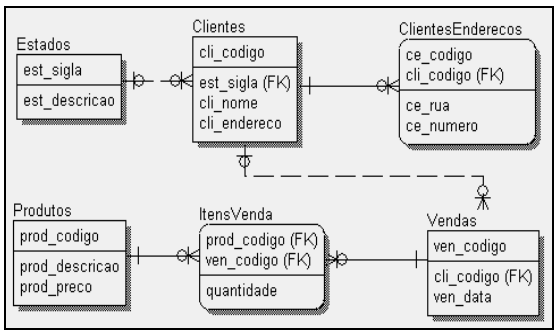

Figura 1. Modelo Entidade-Relacionamento

Com o *script* de definição do modelo ilustrado na Figura 1, é solicitada ao usuário a inserção deste na tela inicial da Ferramenta, ilustrada pela Figura 2.

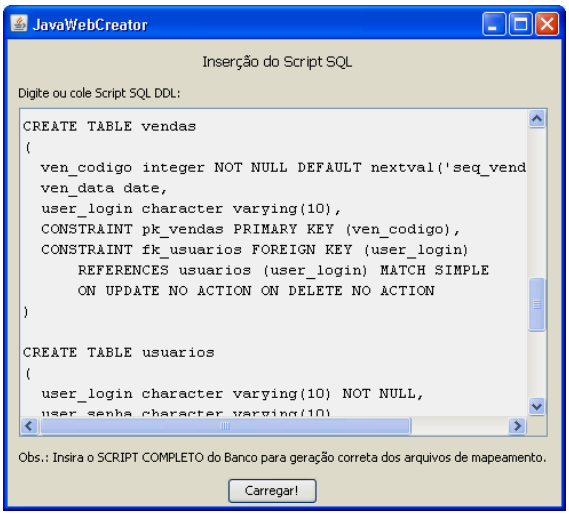

Figura 2. Tela de inserção do *script* do Banco de Dados

Após o carregamento do *script* na ferramenta, são exibidos os nomes de tabelas em uma tela de parametrização, conforme ilustrado na Figura 3. Nessa tela serão escolhidos quais cadastros e movimentações deseja-se gerar.

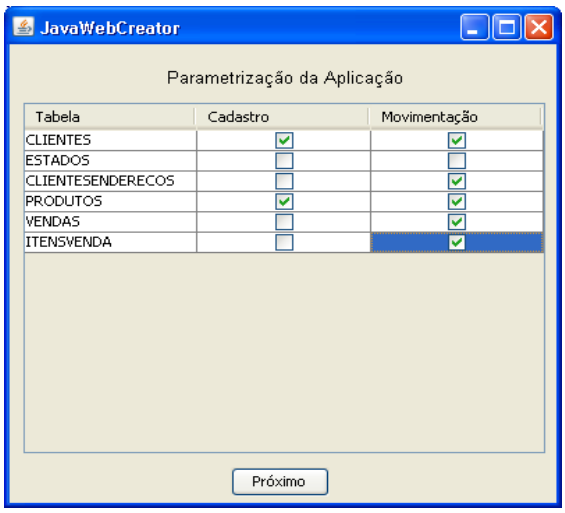

Figura 3. Parametrização da ferramenta

Feita a escolha dos *templates*, é então feito o detalhamento de cada *template*. Em cada um deles o usuário pode visualizar as chaves (primárias e estrangeiras) e atributos pertencentes a determinada tabela.

Nos *templates* de cadastros, as chaves primárias podem ser configuradas escolhendo-se o nome para exibição no formulário ("*Label*") e se a chave é auto-incremento ou não. Para as chaves estrangeiras pode ser definido qual é o campo para exibição ("*display field*"), qual o nome para exibição do *display field* no formulário ("*Label*"); se o campo será obrigatório ou não e se o campo será ou não um filtro de consulta na tela de consulta do Cadastro. Já para os atributos podem ser definidos: o *label*, se o campo é obrigatório ou não e se é filtro de consulta, conforme ilustrado na Figura 4.

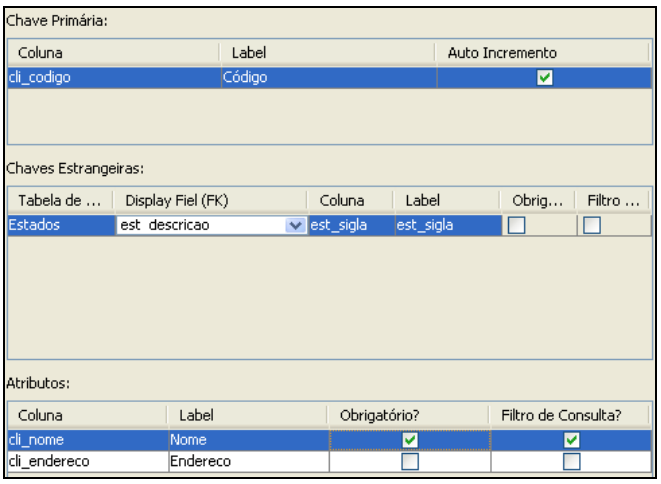

Figura 4. Configuração do *template* de cadastros

Ao ser finalizada a geração dos cadastros, a próxima etapa é a geração das movimentações, sendo que primeiramente é apresentada a tela para o apontamento específico das tabelas envolvidas na movimentação, já que a ferramenta reconhece dois tipos de movimentações: um que envolve três tabelas (tipo 1), conforme ilustrado na Figura 5 e um que envolve duas tabelas (tipo 2), conforme ilustrado na figura 6.

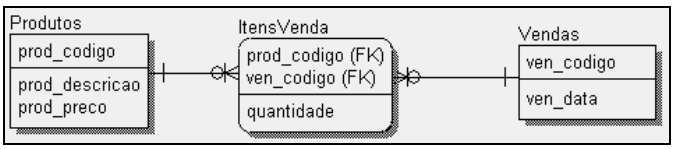

Figura 5. Tabelas que compõem a movimentação de tipo 1

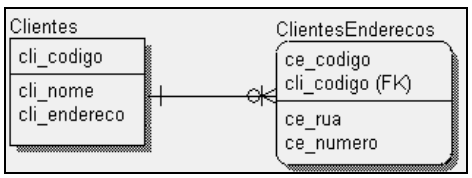

Figura 6. Tabelas que compõem a movimentação de tipo 2

Este apontamento de tabelas é realizado na tela ilustrada pela Figura 7.

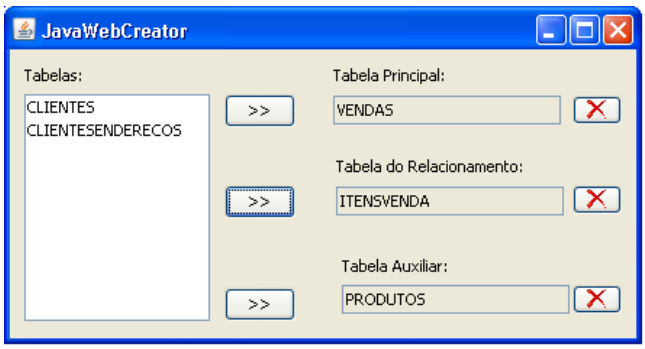

Figura 7. Apontamento de tabelas da movimentação

Caso seja escolhida a movimentação do tipo 1 (três tabelas) devem ser apontadas: a tabela principal, a tabela do relacionamento e a tabela auxiliar, conforme ilustrado na Figura 7. Caso contrário (tipo 2 – duas tabelas) devem ser apontadas: a tabela principal e a tabela do relacionamento, somente.

Feito o apontamento das tabelas, caso seja movimentação do tipo 1, será exibida a tela de configuração da tabela principal (semelhante à ilustração da Figura 4), já que a ferramenta considera que neste tipo de movimentação a chave desta tabela ainda será gerada ao se executar a aplicação. Em seguida será exibida a tela de configuração da tabela do relacionamento (Figura 8), para esta movimentação.

Caso seja escolhida a movimentação do tipo 2 (duas tabelas), apenas será exibida a tela de configuração da tabela do relacionamento, visto que neste tipo de movimentação a ferramenta considera a tabela principal apenas como uma chave estrangeira já existente na Base de Dados. A tela de configuração da tabela de relacionamento é ilustrada na Figura 8.

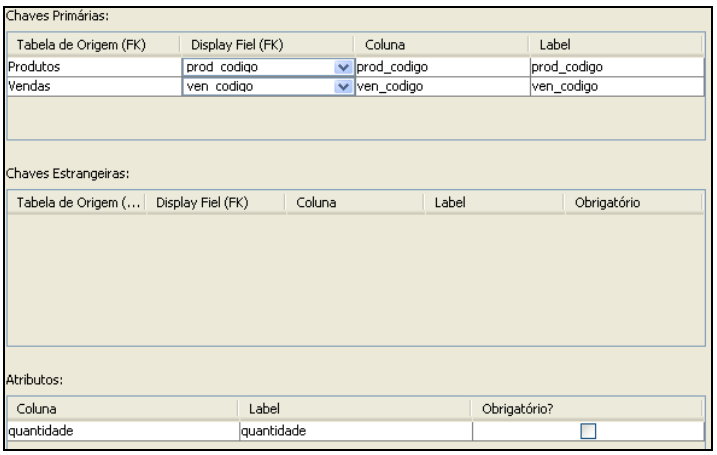

Figura 8. Configuração da tabela de relacionamento

Ao passar para a próxima etapa, é questionado ao programador (usuário da ferramenta) se este deseja mais alguma movimentação. Em caso positivo, a aplicação volta à tela de apontamento de tabelas ilustrada na Figura 7. Nesta tela encontram-se as tabelas escolhidas, anteriormente, no processo de parametrização da aplicação, conforme ilustrado na Figura 3. Finalizada a configuração de cada *template* escolhido, os arquivos referentes aos JSPs e Servlets são criados.

Por fim é gerado, pela ferramenta, o arquivo (“web.xml”) de distribuição dos Servlets, ilustrado na Figura 9. Para geração deste arquivo foi utilizada uma estrutura de dados do tipo Lista Encadeada, que armazena em cada um de seus nós o nome de um JSP e seu respectivo Servlet.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" x
    <servlet>
        <servlet-name>clienteServlet</servlet-name>
        <servlet-class>Classes.clienteServlet</servlet-class>
    </servlet>
    <servlet>
        <servlet-name>emprestimosServlet</servlet-name>
        <servlet-class>Classes.emprestimosServlet</servlet-class
    </servlet>
    <servlet>
        <servlet-name>ITENSVENDAServlet</servlet-name>
        <servlet-class>Classes.ITENSVENDAServlet</servlet-class>
    </servlet>
    <servlet>
```

Figura 9. Arquivo “web.xml”

2.1. Como proceder com a aplicação gerada

Baseando-se no modelo de dados ilustrado pela Figura 1, supõe-se que tenham sido gerados os cadastros de Clientes, Estados, Produtos, Usuários, suas respectivas telas de consulta e as seguintes movimentações: Controle de Endereços à Clientes (Tabela de Clientes e ClientesEnderecos – movimentação do tipo 2) e Controle de Vendas (Tabela de Vendas, Produtos e ItensVenda – movimentação do tipo 1). Essas aplicações são representadas pelos arquivos ilustrados na Figura 10.

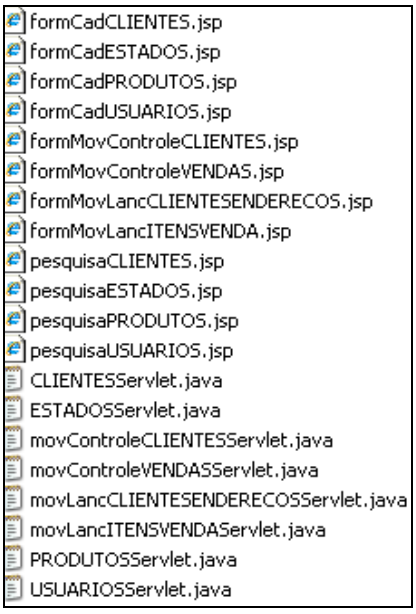

Figura 10. Aplicações geradas

Esses arquivos encontram-se no diretório “geracao”, também criado pela ferramenta. Junto a esses arquivos encontram-se outros diretórios e arquivos, ilustrados pela Figura 11. Na raiz deste diretório encontra-se também o arquivo de sessão do Hibernate – HibernateUtil.java.

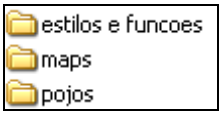

Figura 11. Conteúdo do diretório “geracao”

No diretório “estilos e funcoes” encontram-se arquivos de folhas de estilo (css) utilizados nas páginas, bem como algumas funções e bibliotecas JavaScript (inclusive o jQuery) necessárias para o correto funcionamento das páginas. No dirétorio “maps” encontram-se os arquivos XML gerados pela ferramenta que representam o mapeamento objeto relacional de todas as tabelas do modelo de dados, conforme ilustrado pela Figura 12. Esses mapeamentos são necessários para o funcionamento e operações no Banco de Dados que o *framework* Hibernate irá executar.

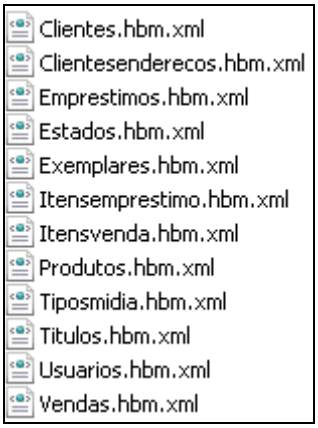

Figura 12. Arquivos de mapeamento

No diretório “pojos” encontram-se classes Java geradas pela ferramenta que, também, representam o mapeamento objeto relacional de todas as tabelas do modelo de dados, conforme ilustrado pela Figura 13. Essas classes também são necessárias para o correto funcionamento do *framework* Hibernate.

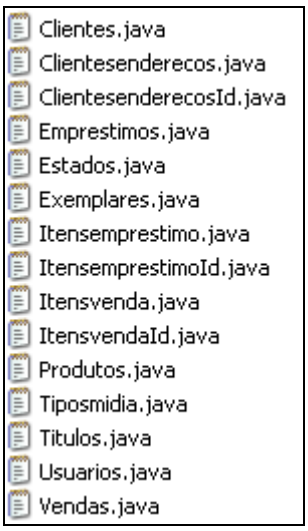

Figura 13. Classes de Banco de Dados

Neste momento tem-se a aplicação gerada e todos os arquivos necessários para o funcionamento dessas aplicações. Esses arquivos poderiam ser utilizados em qualquer ambiente de desenvolvimento Java, como por exemplo, a IDE NetBeans e a IDE Eclipse.

No caso de utilização da IDE NetBeans, uma aplicação Web gerada inicialmente tem a estrutura de arquivos ilustrada pela Figura 14.

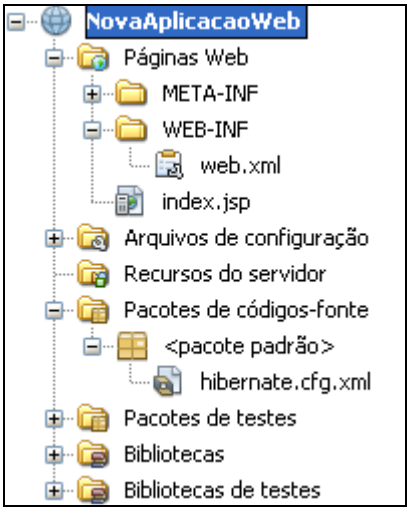

Figura 14. Estrutura inicial de arquivos

Para executar a aplicação gerada, na pasta “Pacotes de códigos-fonte” deve-se criar dois novos pacotes: “model” (onde estarão localizadas as classes Java de Bancos de Dados e o arquivo de sessão do Hibernate - HibernateUtil.java) e “control”, onde estarão localizados os Servlets.

De posse dessa estrutura de arquivos, deve-se copiar os arquivos *.jsp gerados para a pasta “Páginas Web”; os Servlets desses arquivos devem ser copiados para o pacote “control”; o arquivo de sessão do Hibernate (HibernateUtil.java) e as classes Java de Banco de Dados devem ser copiadas para a o pacote “model”; os arquivos de mapeamento (*.hbm.xml) devem ser copiados para o pacote-padrão, porém deveriam ser copiados para o “control”, por se tratar de uma aplicação MVC, mas se os copiar para o pacote “control”, o *framework* Hibernate não os encontra. O conteúdo da pasta “estilos e funcoes” (que também se encontra na pasta dos arquivos gerados pela ferramenta – diretório “geracao”) deve ser copiado para a pasta “Páginas Web”. Por fim deve ser copiado o arquivo de distribuição dos Servlets (“web.xml”) gerado pela ferramenta para a pasta WEB-INF (sub-pasta de “Páginas Web”).

No pacote-padrão, o arquivo “hibernate.cfg.xml” criado pela IDE deve ser excluído. O arquivo “hibernate.cfg.xml” gerado pela ferramenta deve ser copiado para este mesmo pacote.

Neste momento, tem-se a estrutura de arquivos da aplicação como ilustrado na Figura 15.

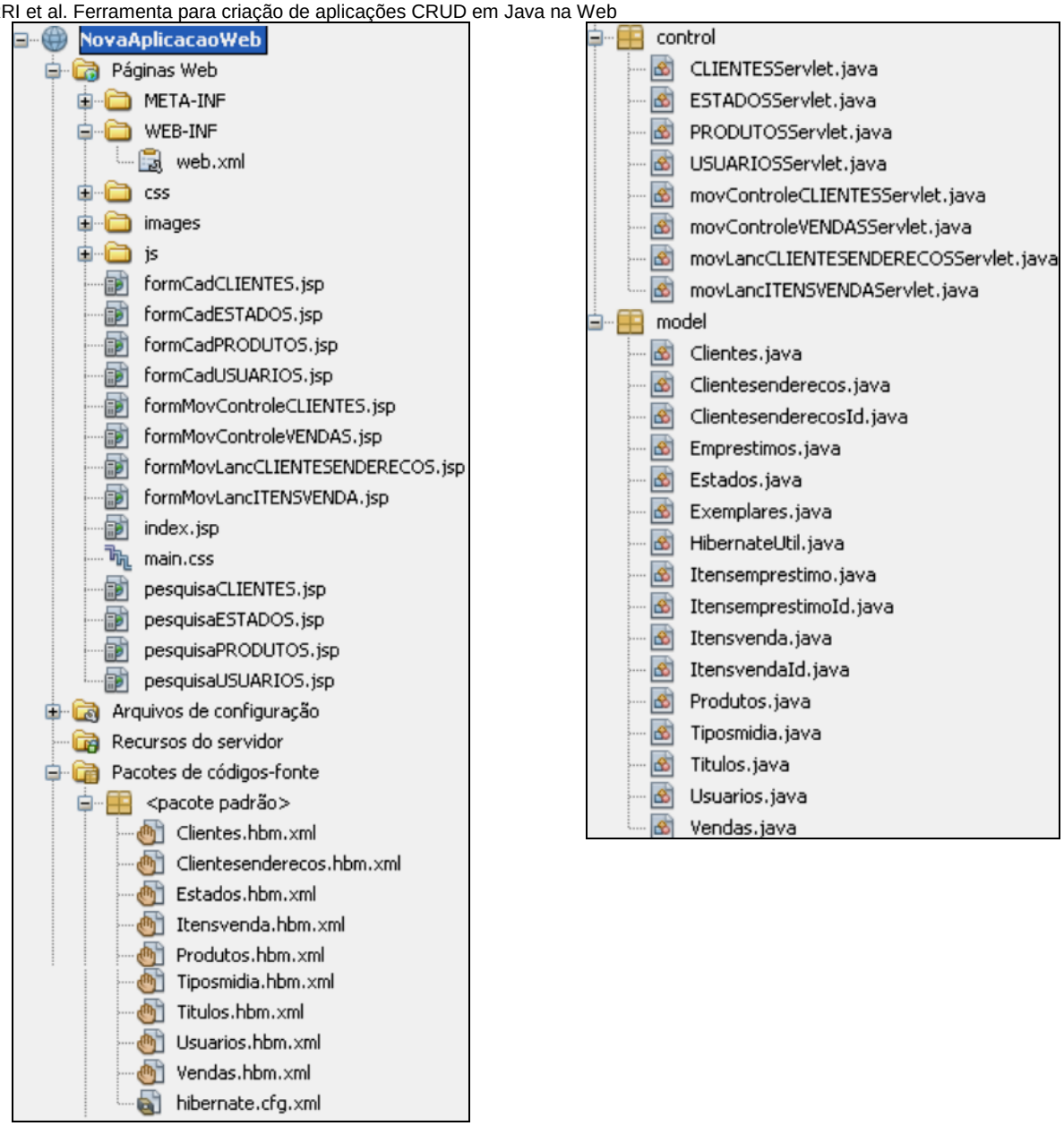

Figura 15. Estrutura de arquivos da aplicação

Caso sejam apontados erros (nos JSPs e/ou Servlets), estes serão erros de tipo de dados, já que a ferramenta geradora não faz verificação de tipos nos campos das tabelas. Esses erros têm de ser corrigidos manualmente pelo programador, selecionando o tipo correto para o campo no qual está sendo apontado erro (nas instruções *get* e *set*).

3. SOBRE AS TECNOLOGIAS UTILIZADAS

3.1. Servlets & JSPs

Algumas informações descritas nesta subseção foram obtidas em Bashan et. al. [Basham et. al. 2008].

Para se construir uma aplicação Web é preciso se utilizar do Java, dos Servlets e dos JSPs, pois as antigas e estáticas páginas HTML estão em desuso. Nos dias de hoje, os usuários esperam por sites dinâmicos, interativos e customizáveis. Com o uso dessas tecnologias, um site na Web se torna uma aplicação Web.

Quando é preciso se ter páginas instantâneas (páginas criadas dinamicamente e que não existiam antes do processamento da

solicitação) e capacidade para escrever ou salvar dados no servidor (escrever em um arquivo ou Banco de Dados) é necessário o papel desempenhado por uma classe “helper”, pois o servidor não possui recursos para implementar o salvamento de dados no servidor, por exemplo.

O JSP é a solução para se colocar o Java dentro de uma página HTML. Uma página JSP é bem parecida com uma página HTML, com a diferença de que pode-se inserir Java (e itens relacionados ao Java) dentro da página. Portanto, é realmente semelhante a inserir uma variável na página HTML.

3.2. O padrão de design MVC

Com o MVC (*Model-View-Controller*) a lógica de negócio não fica apenas separada da apresentação, ela sequer sabe que existe uma apresentação. A essência do MVC é que se pode separar a lógica de negócio da apresentação, mas com algo entre elas (classe “helper”) para que a lógica de negócio possa agir independentemente como uma classe Java reutilizável, sem precisar saber nada sobre a “View” (camada de apresentação).

3.3. Hibernate

Algumas informações descritas nestas subseções foram obtidas em FJ-21 [FJ-21 2009] e FJ-28 [FJ-28 2009].

O *framework* Hibernate tem como função minimizar a complexidade associada aos aplicativos Java, baseado no modelo orientado a objeto, que necessitam trabalhar com um Banco de Dados tradicional do modelo relacional (presente na maior parte dos SGBDs).

Sua característica mais importante é a transformação das classes do Java para tabelas de dados (e dos tipos de dados Java para os da SQL). Este processo é denominado “Mapeamento Objeto Relacional”, conforme ilustrado nas Figuras 16 e 17.

```
<hibernate-mapping>
    <class name="Produtos" table="produtos" schema="public">
        <id name="prodCodigo" type="int">
            <column name="prod_codigo" />
            <generator class="assigned" />
        </id>
        <property name="prodDescricao" type="string">
            <column name="prod_descricao" />
        </property>
        <property name="prodPreco" type="java.lang.Float">
            <column name="prod_preco" precision="8" scale="8" />
        </property>
        <set name="itensvendas" inverse="true">
            <key>
                <column name="prod_codigo" not-null="true" />
            </key>
            <one-to-many class="Itensvenda" />
        </set>
    </class>
</hibernate-mapping>
```

Figura 16. Mapeamento Objeto Relacional (arquivo XML)

```
import java.util.HashSet;
import java.util.Set;

/**
 * Produtos generated by hbm2java
 */
public class Produtos  implements java.io.Serializable {


     private int prodCodigo;
     private String prodDescricao;
     private Float prodPreco;
     private Set<Itensvenda> itensvendas = new HashSet<Itensvenda>(0);

    public Produtos() {
    }
```

Figura 17. Mapeamento Objeto Relacional (classe Java)

3.3.1 ORM

ORM ou “Mapeamento Objeto Relacional” é uma nova técnica de desenvolvimento empregada para reduzir o custo associado à programação orientada a objetos utilizando Banco de Dados Relacionais. As tabelas de dados do Banco são representadas através de classes e as linhas de cada tabela são representadas como instâncias das classes correspondentes. Com esta nova técnica, o programador pode abstrair os detalhes de como construir as consultas SQL. Ele fará uso de uma interface de programação simples que faz todo o restante do trabalho de persistência. Não é

preciso uma relação direta entre as tabelas de dados e as classes da aplicação. A relação entre as tabelas de onde originam os dados e os objetos que os disponibilizarão é configurada pelo desenvolvedor, isolando o código do programa das modificações à organização dos dados nas tabelas do Banco de Dados.

A maneira de configuração deste mapeamento é dependente da linguagem e ambiente de desenvolvimento utilizado. Como exemplo, o desenvolvedor que faz uso do Hibernate na linguagem Java pode usar arquivo XML ou o sistema de anotações (*Java Persistence* API *Annotations*) que a linguagem tem incluída.

3.3.2 HQL (*Hibernate Query Language*)

Assim como o Hibernate implementa uma forma de persistência de dados orientada a objetos, é natural que também apresente como necessidade uma linguagem de manipulação de dados própria para este paradigma. Neste cenário o SQL demonstra-se insuficiente, sendo necessária uma linguagem própria para o Hibernate. E esta linguagem é a *Hibernate Query Language*, a HQL.

A HQL é uma linguagem que pretende ser mais fácil e poderosa que o SQL. Ela permite trabalhar com os mapeamentos da orientação a objetos, e ainda, por uma questão de compatibilidade retrógrada com as Bases de Dados já existentes, suporta o SQL nativo, já que nem em todos os Bancos de Dados previamente existentes a HQL é aplicável. Pode haver um banco de dados já construído, impraticável de ser reconstruído pelo seu gigantismo e que não suporta a HQL gerado pelo Hibernate. Para garantir compatibilidade retrógrada com este tipo de banco, o Hibernate juntamente com a interface Query, implementa outra interface, a interface SQLQuery (na verdade, a interface SQLQuery extende a interface Query). Os objetos criados a partir desta interface apresentam o método createSQLQuery(), que gera o SQL puro suportado por estes Bancos de Dados que só trabalham com SQL nativo. Supostamente o SQL gerado pela HQL já é otimizado, e caso ocorra uma perda em consultas críticas deve-se otimizar a HQL, e não o SQL gerado. A HQL foi inspirada no SQL e por sua vez inspirou o EJB-QL, a linguagem de manipulação de dados do EJB 3.

A HQL suporta ainda os chamados “métodos agregados”, equivalentes às funções agregadas do SQL. A única diferença é que na HQL os métodos agregados são adaptados para a realidade dos objetos persistidos do Hibernate. Alguns exemplos de métodos agregados: avg, count, max, min e sum.

Algumas vantagens do Hibernate: suporta classes desenvolvidas com agregações, herança, polimorfismo, composições e coleções; permite a escrita de consultas tanto através de uma linguagem própria (HQL) como também através de SQL; é um *framework* não intrusivo, ou seja, não restringe a arquitetura da aplicação e implementa a especificação *Java Persistente* API (JPA) [Rocha et. al. 2009].

O Hibernate constrói chamadas SQL e livra o desenvolvedor do trabalho manual da transformação dos dados resultantes, mantendo o programa praticamente portável para quaisquer Bancos de Dados SQL, graças ao suporte de uma linguagem própria e poderosa chamada *Hibernate Query Language* - HQL, que suporta e amplia o SQL nativo. Estas facilidades, todavia, cobram seu preço ao causar um pequeno custo no aumento do tempo de execução.

Em JPA [JPA 2008] encontram-se testes que foram realizados (durante 30 minutos) com o *framework*, obtendo-se os resultados mostrados na Tabela 1.

Tabela 1: Dados coletados em teste de desempenho do Hibernate

Nº de consultas + inserções realizadas	12687
Número de consultas realizadas	3080
Número de inserções executadas	9607
Max. Mem. ocupada durante o teste (Mb)	130
Mem. ocupada após o teste (Mb)	79

Com a Tabela 1, pode-se notar que o Hibernate consumiu bastante memória, gerando um *overhead* (processamento em excesso).

3.4 Ajax

Algumas informações descritas nesta subseção foram obtidas em Mclaughlin [Mclaughlin 2008].

Com o Ajax é possível trazer aos aplicativos Web uma aparência de aplicativo *desktop* sofisticado. O Ajax permite a construção de aplicativos sofisticados de Internet que sejam mais interativos, fáceis de usar e tenham maior capacidade de resposta. Uma abordagem totalmente diferente para a programação na Web.

O Ajax é a atual geração de aplicativos Web, onde o JavaScript, alguma HTML dinâmica e um pouco de XML podem tornar os aplicativos mais rápidos e fazer com que se comportem como aplicativos *desktop* dinâmicos.

Os aplicativos Ajax podem se comunicar com um servidor Web sem o navegador ter que sempre recarregar e redesenhar a página inteira, porém o Ajax não lida apenas com interfaces de usuário aperfeiçoadas.

Além de eliminar as incômodas recargas de página, o JavaScript de um aplicativo Ajax se comunica com o servidor Web assincronamente, ou seja, o código JavaScript fará uma solicitação ao servidor, mas enquanto o servidor estiver trabalhando em segundo plano, pode-se inserir dados em formulário Web e até mesmo efetuar clique em botões. Logo em seguida, quando o servidor tiver terminado, o código poderá apenas atualizar a parte da página que tiver mudado, porém não é necessário aguardar, ressaltando-se o poder das solicitações assíncronas. Ao se combinar solicitações assíncronas com atualização de páginas sem recarga ("*refresh*" ou "*round-trip*") tem-se aplicativos Ajax.

3.4.1 jQuery

A jQuery [jQuery 2009] é uma famosa biblioteca JavaScript que implementa diversos recursos adicionais ao JavaScript comum dos *browsers*. Possui vários efeitos visuais já implementados e prontos para uso. Além disso, já possui muitas APIs para trabalhar com Ajax de forma simples e produtiva.

A biblioteca jQuery é baseada em um conceito de encadeamento (*chaining*). As chamadas de seus métodos são encadeadas uma após a outra, o que cria código muito simples de ser lido.

Os *plugins* da jQuery utilizados nos JSPs gerados pela ferramenta foram: **Nicejforms** – para aparência de formulários; **ThickBox** – tela de consulta "modal" e **Validate** – validação de campos obrigatórios.

3.5 Trabalhos Correlatos

Muitos trabalhos foram encontrados objetivando a geração de código-fonte, principalmente de aplicações *desktop*, como é o caso da ferramenta para gerar código-fonte de aplicações comerciais na linguagem de programação Java, a partir das informações contidas no catálogo do banco de dados [Horaguti 2005]. Ferramentas semelhantes à exposta neste artigo estão disponíveis no mercado, por exemplo, tem-se o BcodeGen [Cipriano 2007] e o CodeCharge [CodeCharge 2009].

O BcodeGen é uma ferramenta feita em C#.NET para geração de aplicações CRUD para Web em PHP5. Neste software a geração de código é feita a partir de uma descrição da base

de dados, é necessário obter essa descrição e ter estruturas de dados que permitam representar e manipular essa informação dentro do programa. A Base de Dados é descrita em XML, que é interpretado de forma a obter as várias tabelas e respectivos campos. Para fazer representar a Base de Dados internamente foram criadas algumas classes cujo objetivo é representar uma Base de Dados no Modelo Relacional. Com estas classes, além de representar a Base de Dados, é também necessário manipular os dados obtidos, para fazer a geração de código propriamente dita. Esse software pode ser encontrado na Universidade Lusófona de Humanidades e Tecnologias - Departamento de Ciências da Comunicação, Artes e Tecnologias de Informação (Portugal) – já que se trata de um Projeto de Licenciatura em Informática. Porém, nos dias atuais, com as novas tecnologias existentes, como o Ajax, essa ferramenta pode ser aprimorada.

O CodeCharge é um gerador de códigos para aplicações Web no qual é possível criar Web sites nas linguagens (ASP, ASP.Net/C#, ColdFusion, JSP, PHP e Perl) com acesso a qualquer Banco de Dados Relacional (incluindo Oracle, Sybase, mySQL, Microsoft SQL Server e Microsoft Access que sejam passíveis de conexão via JET, ODBC, JDBC, ADO, DBI e PHPLib). Os códigos gerados rodam em qualquer versão do Windows, Unix e Linux. Este software pode ser encontrado em [CodeCharge 2009], porém é uma ferramenta limitada e de uso restrito, pois se trata de um software comercial e para utilizá-lo é necessário aquisição de licenças de uso. Também foram relatados problemas na geração de código JSP que estão sendo resolvidos pela empresa desenvolvedora.

Algumas das principais diferenças dessas ferramentas em relação à ferramenta exposta neste artigo são descritas a seguir: No que se diz respeito ao BcodeGen, é o uso da tecnologia Ajax (melhorando assim a interface com o usuário e também permitindo programação assíncrona), e a geração do código das aplicações Web é na linguagem Java. Já em relação à CodeCharge a principal diferença é a gratuidade da ferramenta geradora de código-fonte de aplicações Web.

4. CONCLUSÕES

É importante que os programadores utilizem a ferramenta, já que traz melhoria na produtividade, poupa trabalho repetitivo e pouco criativo, deixando tempo livre para outras atividades, melhorando consequentemente a qualidade de vida dos programadores. Vale ressaltar que a qualidade do código gerado pela ferramenta vai depender diretamente da habilidade do programador em manuseá-la, aplicando ajustes necessários, etc.

Projetos criados com geradores de código podem apresentar falhas. No entanto, a aplicação de baterias exaustivas de testes, conhecimento profundo da ferramenta e bom planejamento ajudam a corrigir falhas, por isso é importante manter ferramenta em atualização/manutenção constante, inclusive para aprimoramento/ampliação de *templates* e recursos que a ferramenta tem potencial para oferecer aos programadores.

REFERÊNCIAS

Basham, B., Sierra, K., Bates, B. (2008), Use a Cabeça! Servlets & JSP. AltaBooks. 2ª edição.

Cipriano, B. Ferramenta de Geração de Código para Aplicações Web. 2007. Relatório da disciplina de Projeto da Licenciatura em Informática – Departamento de Ciências da Comunicação, Artes e Tecnologias de Informação, Universidade Lusófona de Humanidades e Tecnologias, Portugal.

CodeGeneration. Disponível em: http://www.codegeneration.net. Acesso em: 30 de novembro de 2009.

FJ-21 Java para desenvolvimento Web. Caelum Ensino e Soluções em Java. Disponível em: www.caelum.com.br. Acesso em: 26 de fev. de 2009.

FJ-28. Caelum Ensino e Soluções em Java. Disponível em: www.caelum.com.br.

Horaguti, S. H. S. Ferramenta Geradora de Código-Fonte de Aplicações Comerciais em Java, 2005. Monografia (Graduação em Ciência da Computação) – Faculdade de Informática de Presidente Prudente, UNOESTE, Presidente Prudente.

JPA implementations comparison: Hibernate, Toplink Essentials, Openjpa, Eclipselink, 2008. Disponível em: http://terrazadearavaca.blogspot.com/2008/12/jpa-implementations-comparison.html. Acesso em: 30 de novembro de 2009.

jQuery. Disponível em: http://jquery.com/. Acesso em: 3 de março de 2009.

Mclaughlin, B. Use a Cabeca Ajax. AltaBooks, 2008.

Rocha, G.; Filho, H.; Jurity, R. Camada de Persistência de Dados para Aplicações Java: O Hibernate. Artigo, Instituto de Matemática – Universidade Federal da Bahia (UFBA). Disponível em: http://www.mauroborges.com.br/downloads/Setor%20de%20TI/POO/Hibernate.pdf. Acesso em: 20 de março de 2009.

Yes Software. CodeCharge, 2009. Disponível em: http://www.codecharge.com.br/. Acesso em: 30 de novembro de 2009.