5CONTECSI June 04 - 06, 2008 São Paulo, Brazil

5th International Conference on Information Systems and Technology Management
5º Congresso Internacional de Gestão da Tecnologia e Sistema de Informação
De 04 a 06 de Junho de 2008 - São Paulo - Brasil

RF-1112

WEBMAKER - AN ENVIRONMENT FOR AUTOMATIC GENERATION OF WEB APPLICATIONS

Leonardo Luiz Sarmento Schneider (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - leoschneider@gmail.com
Ronaldo Ribeiro Goldschmidt (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - rribeiro@univercidade.edu.br
Eduardo Bezerra da Silva (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - edubezerra@gmail.com
Jorge de Abreu Soares (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - jorge@univercidade.edu.br
Colaboradores
Luiz Fernando Gomes da Silva Júnior (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - luizf_silva@click21.com.br
Rafael Castaneda Ribeiro (Centro Universitário da Cidade do Rio de Janeiro, RJ, Brasil) - rafaelcastaneda@gmail.com

With the growth of Internet, the existence of tools that accelerate the process of developing web applications has become increasingly important. Frequently, Information Technology (IT) areas in companies do not supply the demand for new computational systems. This article aims at describing an environment, called WebMaker, that allows non-developer users create transactional semantic web applications without the need of prior knowledge in programming languages. The user provides information about data he wants to view or monitor and the own environment creates the web application. With WebMaker, applications are created using the new concept of Semantic Web, where data is organized and stored in a way that not only people but also machines can understand the meaning of the available content. This capability opens many other possibilities for integrating systems, searches and other web applications. In summary, WebMaker offers modern companies a way of reducing IT´s backlog for new computational systems.

Keywords: webmaker, web applications, semantic web applications, integrating systems

WEBMAKER - UM AMBIENTE PARA GERAÇÃO AUTOMÁTICA DE APLICAÇÕES WEB

Com o crescimento da Internet, vem se tornando cada vez mais necessária a existência de ferramentas que acelerem o processo de desenvolvimento de aplicações web. É crescente também o volume de demandas por novos sistemas de informação nas empresas modernas. Em geral as áreas de Tecnologia da Informação das organizações têm dificuldade em atender com agilidade todas as solicitações. Diante deste cenário, o presente artigo tem como objetivo descrever um ambiente, denominado WebMaker, que permita ao usuário não programador criar aplicações web transacionais semânticas sem a necessidade de possuir um conhecimento prévio em qualquer linguagem de programação. Tal ambiente se apresenta, portanto, como uma ferramenta que pode ser utilizada para acelerar a redução dos *backlogs* das áreas de Tecnologia da Informação das empresas no atendimento de seus usuários ávidos por novos sistemas computacionais. Através de formulários, o usuário fornece informações sobre os dados que deseja exibir e/ou controlar e o WebMaker cria a aplicação desde o seu modelo de dados até as páginas estáticas e/ou transacionais. As aplicações criadas utilizam o novo conceito de Web Semântica, onde os dados são organizados e catalogados de forma que não somente pessoas, mas também as máquinas possam entender o significado do conteúdo exibido, abrindo muitas outras possibilidades para integração e buscas entre sistemas e outras aplicações web.

Palavras-chave: webmaker, aplicações web, web semânticas, sistemas integrados.

WEBMAKER - UM AMBIENTE PARA GERAÇÃO AUTOMÁTICA DE APLICAÇÕES WEB

Leonardo Luiz Sarmento Schneider, Luiz Fernando Gomes da Silva Júnior, Ronaldo Ribeiro Goldschmidt, Jorge de Abreu Soares, Eduardo Bezerra da Silva, Rafael Castaneda Ribeiro

Núcleo de Projetos e Pesquisas em Aplicações Computacionais – NUPAC
Centro Universitário da Cidade do Rio de Janeiro – UniverCidade
Rio de Janeiro – RJ – Brasil

leoschneider@gmail.com, luizf_silva@click21.com.br, rribeiro@univercidade.edu.br, jorge@univercidade.edu.br, edubezerra@gmail.com, rafaelcastaneda@gmail.com

1. Introdução

O crescimento da internet tem tornado cada vez mais necessária a criação de aplicações na web. É crescente a demanda, sobretudo nas empresas modernas, pelo desenvolvimento de sistemas de informação em ambiente web. No entanto, a produtividade na criação de tais sistemas é invariavelmente afetada pelo fato de que para tal, é necessário que se tenha um conhecimento prévio de programação e de padrões de desenvolvimento de software. Isso faz com que seja necessária a contratação de terceiros e/ou treinamento de funcionários, o que gera maior custo e uma demora maior para atender à demanda pelo desenvolvimento de novos sistemas.

Diante do cenário descrito acima, o presente artigo tem como objetivo descrever um ambiente computacional, denominado WebMaker, que permite ao usuário leigo (não programador) criar aplicações web transacionais semânticas sem a necessidade de possuir um conhecimento prévio em qualquer linguagem de programação. Através de formulários, o usuário fornece informações sobre os dados que deseja exibir e/ou dados que deseja controlar e o ambiente cria a aplicação desde o seu modelo de dados até as páginas estáticas e/ou transacionais. As aplicações criadas utilizam o novo conceito de Web Semântica, onde os dados são organizados e catalogados de forma que não somente pessoas, mas também as máquinas possam entender o significado do conteúdo exibido, abrindo muitas outras possibilidades para integração e buscas entre sistemas e outras aplicações web. Para tanto, conceitos e recursos de ontologias [1] foram amplamente utilizados na concepção do WebMaker e dos sites por ele gerados.

Este artigo encontra-se dividido em mais quatro subseções. A seção dois descreve a estrutura do ambiente desenvolvido, assim como as tecnologias utilizadas nesse desenvolvimento. A seção três mostra exemplos de protótipos desenvolvidos com auxílio do WebMaker, e comenta os testes realizados bem como os resultados obtidos; Trabalhos relacionados e uma visão comparativa com o WebMaker são apresentados na seção quatro. A seção cinco resume as principais conclusões obtidas por meio dos resultados alcançados e fornece sugestões de melhorias futuras.

2. Abordagem Proposta

O WebMaker torna possível a geração de websites semânticos por usuários que não possuam experiência neste tipo de processo. Para atingir este objetivo, o sistema disponibiliza uma interface intuitiva e amigável onde o usuário pode manter várias aplicações web, tratadas como projetos pelo WebMaker.

Em cada um destes projetos, o usuário pode manter que dados os internautas que devem ter acesso ao futuro site devem possuir para acessar as áreas restritas do mesmo. Para ilustrar, em um projeto, o usuário do WebMaker pode definir que os internautas devem informar nome, sobrenome, endereço de e-mail, senha e sexo. Em outro projeto, o usuário do WebMaker pode definir que o internauta deva fornecer e-mail, nome de usuário e senha somente.

Também é possível ao usuário de nossa ferramenta manter grupos de informação, que são os elementos agrupadores de dados e a partir de onde são geradas as páginas componentes da aplicação web gerada. É também através do grupo de informação que o usuário cadastra os metadados a serem disponibilizados nas páginas geradas, de forma a torná-las visíveis e interpretáveis em buscas envolvendo semântica.

Como exemplo, um grupo de informação pode ser composto pelos dados de um CD. Este grupo de informação pode possuir itens, que são suas propriedades e atributos, tais como título, banda, ano de lançamento, assim como outros grupos de informação como músicas, que possuem nome, letra, compositor, autor. Com estes grupos de informação, o usuário do sistema pode criar novos itens complexos (ou seja, formados pelos grupos de informação) a serem exibidos no site de forma transparente.

Para gerar o website, o usuário do sistema precisa somente selecionar o projeto e solicitar a sua geração. O processo de criação de ontologias, metadados, páginas ASP.NET e JSP, classes e sua compilação são de responsabilidade do WebMaker.

A seguir, apresentamos o detalhamento do sistema WebMaker.

2.1. Ontologias nos websites gerados

A web semântica utiliza ontologias [4] para a estruturação das informações bem definidas que possam ser processadas automaticamente. Uma ontologia possibilita a criação de classes, suas propriedades e relacionamentos entre essas classes, com o objetivo de montar um vocabulário para viabilizar a leitura automática dessas informações.

Dentro do escopo deste trabalho, criamos algumas ontologias para as páginas que o sistema gerará automaticamente. Assim, de forma a torná-las semânticas, as páginas geradas pelo WebMaker podem possuir informações sobre elas mesmas tais como seu assunto, criador, tipo de objeto, formato e uma descrição do conteúdo da mesma. Para que isso seja possível, para cada página a ser gerada pelo sistema, é criada uma ontologia em OWL [1] para viabilizar o controle descrito anteriormente. Estes arquivos OWL foram criados a partir do modelo de metadados Dublin Core [2] de representação de dados.

2.2. Dublin Core (DC) Resource Description

O conjunto de metadados proposto pelo padrão Dublin Core é composto de 15 elementos [3], todos utilizados pelo WebMaker na geração das aplicações web:

- **Assunto (subject):** O tópico abordado pelo trabalho
- **Título (title):** Nome do objeto
- **Criador (creator):** Pessoa(s) responsável(eis) pelo conteúdo intelectual do objeto

- **Descrição (description):** Descrição do conteúdo do objeto
- **Editor (publisher):** Agente ou agência responsável por disponibilizar o serviço
- **Outro agente (contributor):** Pessoa(s) que fez (fizeram) contribuições significativas para o objeto
- **Data (date):** Data da publicação
- **Tipo de Objeto (type):** Gênero do objeto, se ficção, novela, poema ou dicionário
- **Formato (format):** Manifestação física do objeto. Exemplos são arquivos executáveis, do tipo texto ou PDF
- **Identificador (identifier):** Cadeia ou número utilizado para identificar unicamente aquele objeto
- **Relacionamento (relation):** Relacionamento com outros objetos
- **Fonte (source):** Outros objetos, eletrônicos ou físicos, dos quais este foi derivado (caso seja aplicável)
- **Linguagem (language):** Linguagem do conteúdo intelectual
- **Cobertura (coverage):** Localizações espaciais e durações temporais características do objeto
- **Direitos (rights):** Informação sobre os direitos acerca do objeto

2.3. Modelo de Ontologia em OWL

A seguir encontra-se descrito um modelo de ontologias criado em OWL que fica como documento anexo às páginas pertencentes a um determinado grupo de informação nos sistemas gerados pelo WebMaker.

Nesta ontologia, utilizamos o padrão Dublin Core (Seção 2.2) para facilitar a descrição das páginas geradas e de seu conteúdo. Utilizamos também dois atributos extras adicionados ao padrão Dublin Core que descrevem os dados de versionamento da página gerada. Estes atributos descrevem a versão atual do documento e o nome do arquivo OWL da versão anterior, para que assim possamos fazer o rastreamento das versões geradas.

Um exemplo da ontologia poderá ser visto na Figura 1 abaixo:

```
    <owl:Class rdf:ID="Identifier">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Language">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Relation">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Rights">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Subject">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Title">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>
    <owl:Class rdf:ID="Type">
        <rdfs:subClassOf rdf:resource="#Page"/>
    </owl:Class>


    <owl:Class rdf:ID="User"/>
    <owl:Class rdf:ID="Contributor">
        <rdfs:subClassOf rdf:resource="#User"/>
    </owl:Class>
    <owl:Class rdf:ID="Creator">
        <rdfs:subClassOf rdf:resource="#User"/>
    </owl:Class>
    <owl:Class rdf:ID="Publisher">
        <rdfs:subClassOf rdf:resource="#User"/>
    </owl:Class>


    <owl:Class rdf:ID="Version"/>
    <owl:Class rdf:ID="Coverage">
        <rdfs:subClassOf rdf:resource="#Version"/>
    </owl:Class>
    <owl:Class rdf:ID="Date">
        <rdfs:subClassOf rdf:resource="#Version"/>
    </owl:Class>
    <owl:Class rdf:ID="Number">
        <rdfs:subClassOf rdf:resource="#Version"/>
    </owl:Class>

</rdf:RDF>
```

Figura 1: Exemplo de Ontologia em OWL Gerada pelo WebMaker – Padrão Dublin Core

No exemplo acima, pode-se perceber como a ontologia é criada e de que forma os dados serão guardados para que a página tenha uma semântica associada.

Para cada página gerada, existirá uma classe *Page* que é responsável por conter todos os dados referentes àquela página. Como subclasses de *Page*, temos as classes *Description*, *Format*, *Identifier*, *Language*, *Relation*, *Rights*, *Subject*, *Title* e *Type* que

identificam e contêm informações relevantes à página. A seguir, descrevemos os significados que atribuímos a cada uma dessas subclasses.

Na classe *Description*, armazenamos a descrição textual da página gerada, para que, quando for esta for recuperada, seja possível identificar a função da página de forma clara e em linguagem natural.

A classe *Format* corresponde ao formato da página, seja ele XML, HTML, ASP.NET, JSP ou outro formato válido qualquer. Assim os agentes que recuperarem os dados da ontologia desta página podem identificar como devem processar essa página de forma adequada.

A classe *Identifier* contém um identificador único para a página (e.g., {550e8400-e29b-41d4-a716-446655440000}).

A classe *Language* contém a descrição da linguagem na qual a página foi gerada (e.g., pt-br ou en-us).

A classe *Relation* contém a descrição do nome de outras páginas que dizem respeito ao mesmo Grupo de Informação do qual a página gerada faz parte.

A classe *Rights* contem os direitos de distribuição do conteúdo da página gerada. O conteúdo será puramente descritivo para que possa ser consultado pelos agentes que farão inferência na ontologia da página.

As classes *Subject* e *Title* contêm informações sobre o assunto ao qual a página se refere e o título da página, respectivamente.

A classe *Type* diz respeito ao tipo da página, ou seja, se ela é de busca ou de detalhe de conteúdo, por exemplo. Esta classe representa um dado descritivo que possa ser consultado.

Também definimos a classe *User*, que é responsável pelos dados de usuário do sistema gerador de páginas. Abaixo desta classe temos as subclasses *Creator*, *Publisher* e *Contributor*, que contêm os dados dos usuários que atuaram na página gerada. As classes *Creator*, *Publisher* e *Contributor* contêm os mesmos dados sobre os usuários, como nome do usuário e e-mail.

Definimos também a classe *Version,* para controlar a versão atual da página gerada. Como subclasses de *Version*, temos ainda as subclasses *Coverage*, *Date* e *Number*, que contêm os dados referentes à versão da página. Na classe *Coverage*, definimos a URL para localização da página no website. A classe *Date* contém a data de criação da versão da página. A classe *Number* contém o número de versão atual da página.

2.4. Diagrama de Casos de Uso

Para facilitar a compreensão da estrutura funcional do WebMaker é apresentada na figura 2 uma visão geral das funcionalidades dessa ferramenta através de seu diagrama de casos de uso.

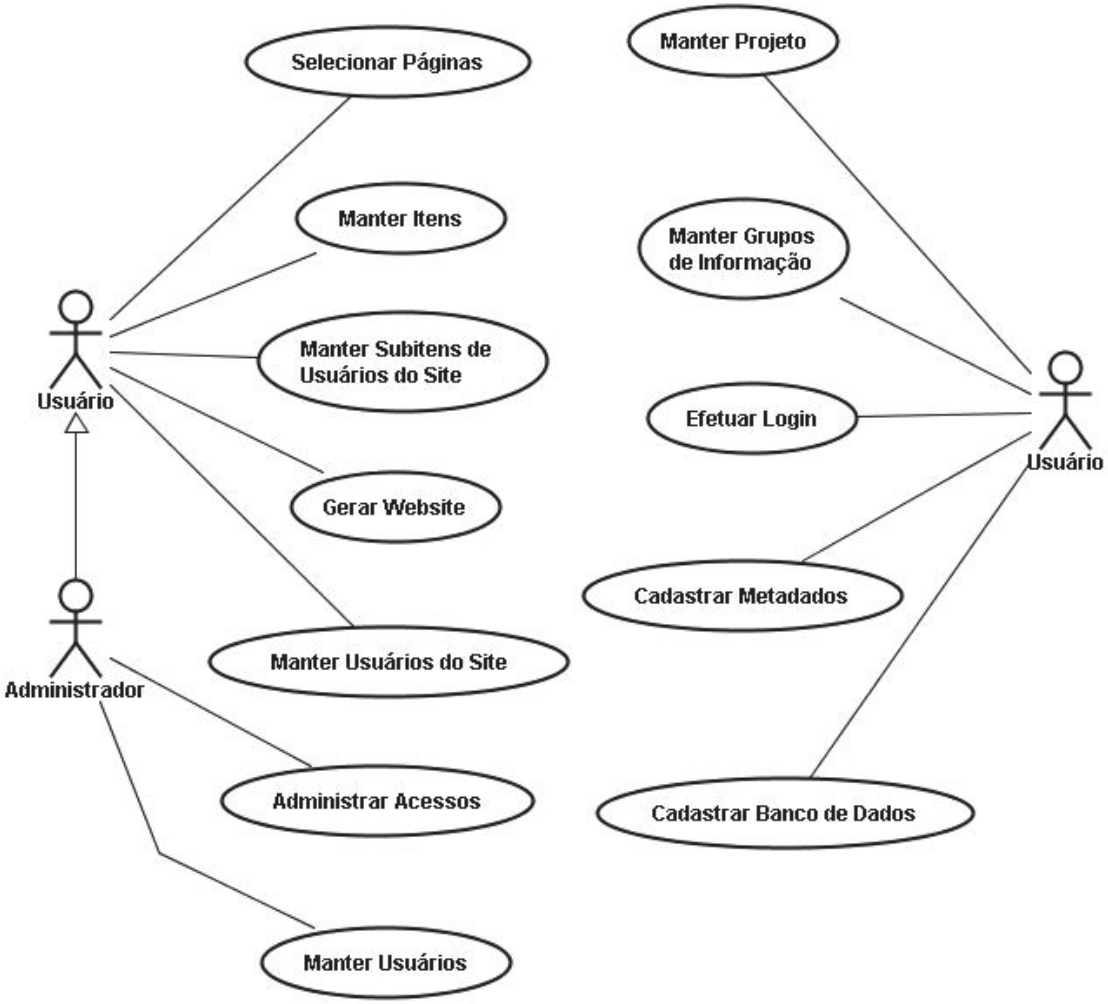


Figura 2: Diagrama de Casos de Uso do WebMaker

Dentre os casos de uso ilustrados no diagrama acima, podemos destacar como principais os casos de uso Manter Grupos de Informação e Gerar Website, que detalhamos adiante.

No caso de uso Manter Grupos de Informação o usuário tem a possibilidade de consultar, incluir, alterar e excluir cada um dos grupos de informação a serem manipulados pela aplicação a ser gerada. Dentro do cadastro de um grupo de informação, o usuário tem a possibilidade de realizar toda a manutenção dos seus itens, relacionamento entre os grupos de informação cadastrados, seleção de páginas e o cadastramento de metadados.

No cadastro de metadados, o usuário tem a possibilidade de alterar cada um dos itens de metadados e, com estes dados, realizar a geração dos arquivos de ontologias posteriormente.

Já o caso de uso Gerar Website é aquele no qual o usuário do WebMaker realiza a geração de todo o website. Neste caso de uso, o usuário é questionado somente sobre quais linguagens de programação deseja gerar o website e, a partir deste ponto, o sistema realiza a geração de todos os itens necessários para viabilizar o funcionamento do website. Maiores detalhes sobre esta geração são descritos na seção 2.6 deste artigo.

2.5. Diagrama de Classes

Para implementar o WebMaker, foram criadas várias classes conceituais conforme ilustrado no Diagrama de Classes da Figura 3.

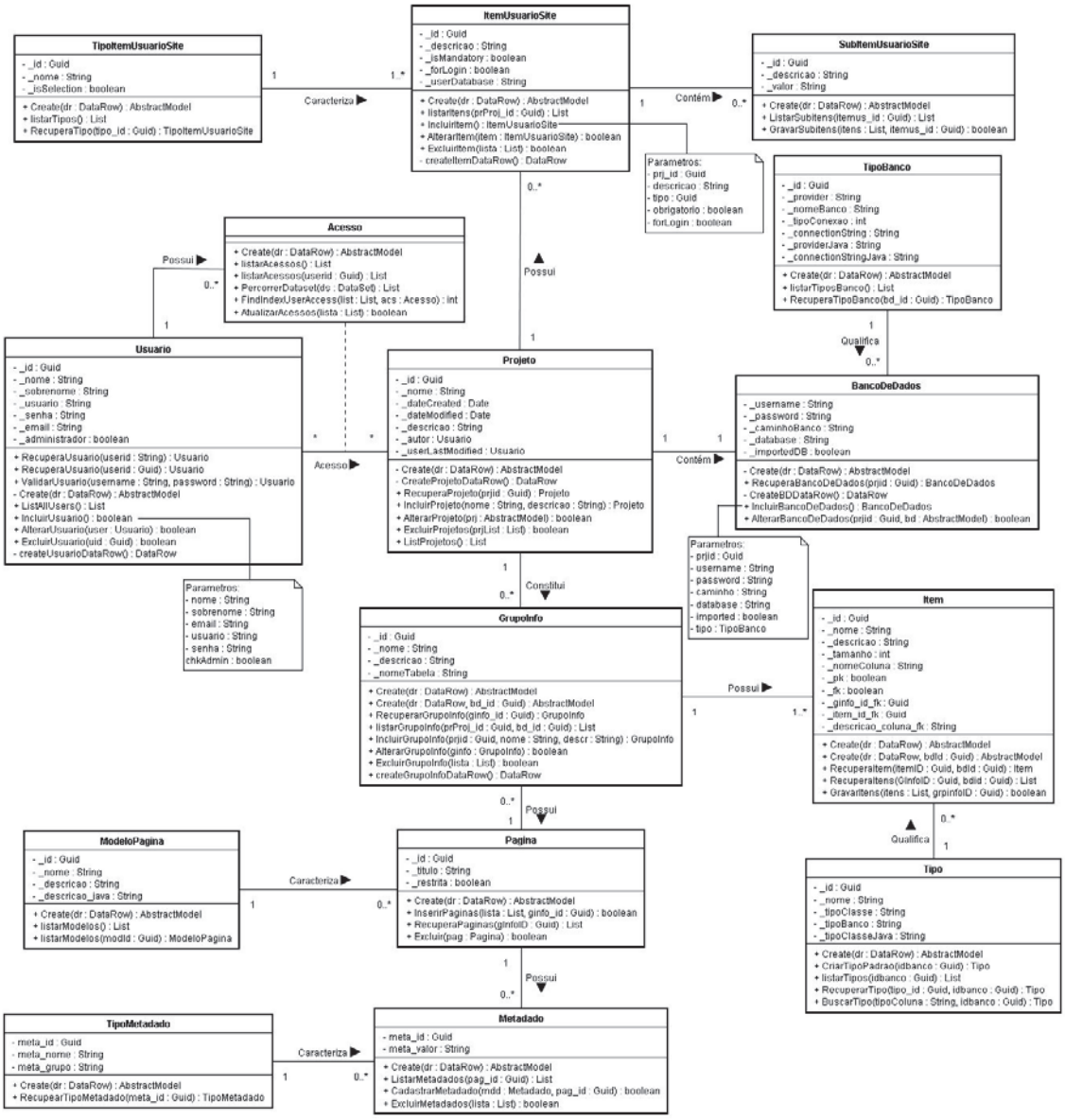


Figura 3: Diagrama de Classes do WebMaker

Neste diagrama de classes podemos visualizar a estrutura central do WebMaker. Conforme mencionamos anteriormente, cada aplicação web é tratada como um projeto pelo WebMaker. Na classe *Projeto*, podemos encontrar associações com as classes *GrupoInfo*, *BancoDeDados*, *Usuario* e *ItemUsuarioSite*, completando assim todo o conjunto de informações básicas necessário para a geração de um website. A classe *GrupoInfo* contém associações com *Item* e *Pagina*, que por sua vez possuem ligações com *Tipo* para o *Item* e *ModeloPagina* e *Metadado* para a página.

A classe *Metadado* contém informações sobre cada *Tipo* de *Metadado* (através da classe *TipoMetadado*) e *Pagina*.

Os usuários do WebMaker têm seus acessos a projetos controlados através da classe *Acesso*, que comporta informações para cada par *Usuario-Projeto*.

Cada projeto possui um banco de dados associado, através da classe BancoDeDados. O WebMaker atualmente dá suporte aos servidores de bancos de dados SQL Server Express, MySQL e PostgreSQL e estes são representados pela classe *TipoBanco*.

Cada projeto também pode conter informações sobre quais itens estarão disponíveis para os internautas. Cada um desses itens é armazenado na classe *ItemUsuarioSite*. Estes itens são classificados através da classe *TipoItemUsuarioSite* e podem conter subitens, que são armazenados pela classe *SubItemUsuarioSite*, como por exemplo: uma pessoa possui um atributo chamado Sexo, que será um item da classe *TipoItemUsuarioSite* e para este item existem dois subitens, masculino e feminino, que por sua vez são armazenados na classe *SubItemUsuarioSite*.

Através do conjunto de classes descrito acima, uma aplicação web pode ser gerada pelo WebMaker, possuindo todos os requisitos necessários de grupos de informação com seus itens, páginas a serem geradas com os seus arquivos de ontologias associados, itens que estarão disponíveis para os internautas se cadastrarem, banco de dados associado com tipo, local, usuário e senha e possuindo toda a interconexão entre os grupos de informação cadastrados, caso haja.

2.6. Detalhamento do caso de uso Gerar Website

O diagrama de atividades ilustrado na Figura 4 apresenta o fluxo de realização do caso de uso Gerar Website, um dos principais do WebMaker, pois é através dessa funcionalidade que ocorre toda a geração de código e ontologias dos projetos. O diagrama tem como objetivo representar os passos da criação de ontologias dos grupos de informação cadastrados, geração de classes baseadas nos grupos de informação e criação das páginas baseadas nos modelos cadastrados e selecionados.

Como pode se observar na ilustração da Figura 4, todo o fluxo de geração de uma aplicação web acontece no caso de uso Gerar Website. Após recuperar uma lista com todos os metadados disponíveis, o WebMaker, para cada uma das linguagens de programação selecionadas, cria toda a estrutura do website necessária para posterior compilação. Logo após, copia o código-fonte padrão, que é igual para todos os websites gerados.

A partir deste ponto, o sistema verifica se o usuário definiu o banco de dados como importado. Caso não seja, o WebMaker se encarrega de gerar todo o esquema de banco de dados (tabelas, colunas, restrições de integridade) para cada um dos grupos de informação associados ao projeto de website em questão. É nestas tabelas que todos os dados são cadastrados para cada um dos grupos de informação e ficam disponíveis para consultas futuras.

Após gerar o esquema do banco de dados, o sistema percorre novamente cada um dos grupos de informação para gerar as suas classes de domínio, controladoras e de DAO [5], garantindo assim que a aplicação web obedeça ao padrão MVC [6] de desenvolvimento. O mesmo processo é adotado para a criação das classes específicas para manter os dados dos internautas.

Ao final da geração da estrutura de classes da aplicação web, o sistema gera todas as páginas de cada um dos grupos de informação. Estas páginas podem ser de visão geral, de detalhes ou de administração, também com listas e detalhes. Ao final da geração das páginas o WebMaker cria os arquivos OWL com os dados cadastrados para cada uma das páginas disponíveis para a aplicação web.

Ao final do processo de geração, o WebMaker compila todo o código-fonte existente e notifica o usuário que a geração da aplicação web foi concluída com sucesso.

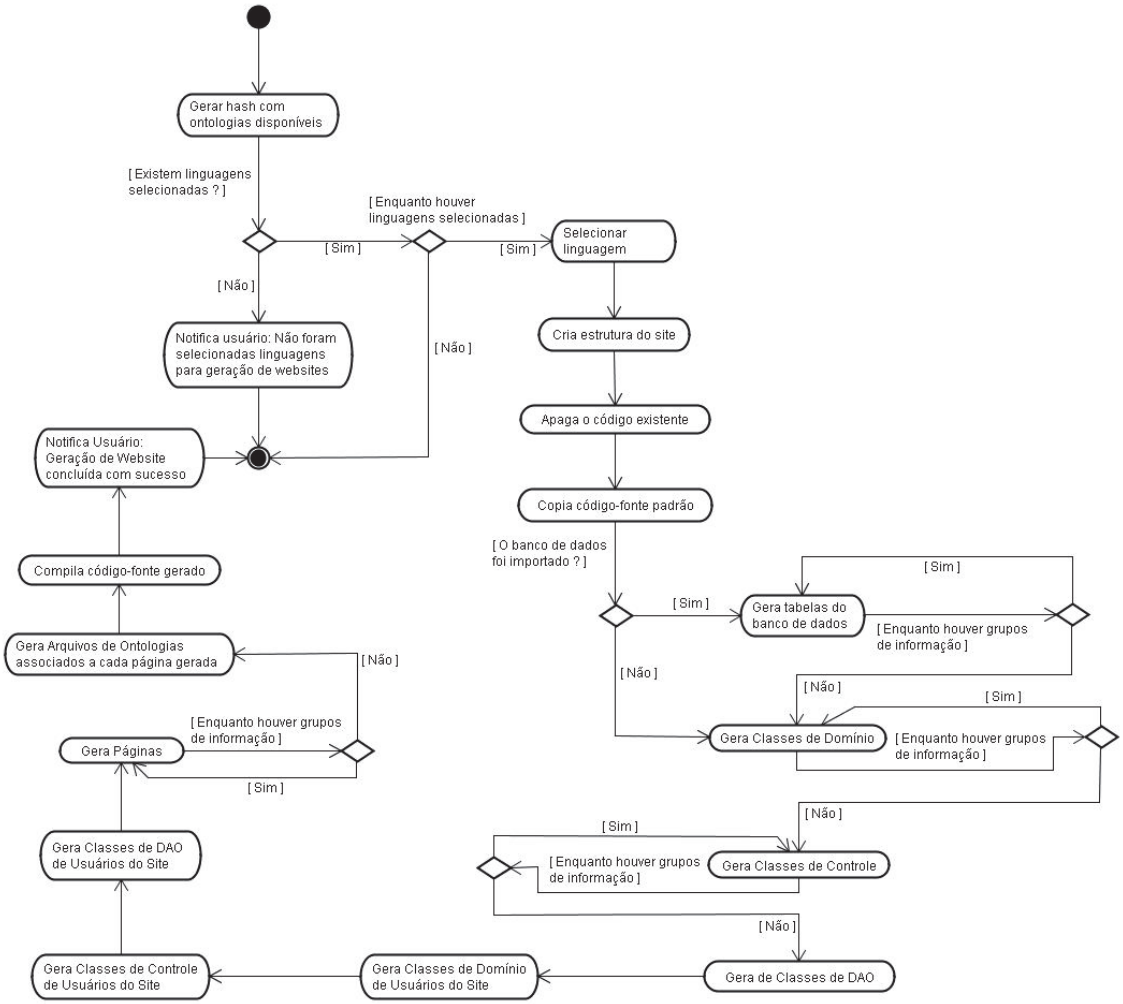


Figura 4: Diagrama de Atividades Gerar Website

3. Protótipo, Experimentos e Resultados

Esta seção tem por objetivos fornecer detalhes sobre a implementação do WebMaker, descrever a metodologia de validação adotada, além de apresentar os resultados obtidos e uma análise dos experimentos realizados.

3.1. Protótipo

O objetivo da ferramenta WebMaker é fornecer para qualquer usuário, sendo leigo (não programador) ou não, suporte na criação de aplicações web. Com esta ferramenta, o

usuário é capaz de criar aplicações web fornecendo apenas algumas informações da aplicação desejada.

A ferramenta foi desenvolvida inicialmente para dar suporte à geração de toda a estrutura necessária para a existência de uma aplicação web totalmente orientado a objetos utilizando a tecnologia .NET, incluindo o banco de dados SQL Server Express.

Porém, após uma versão inicial, a idéia foi modificada para que, além do módulo onde é gerada toda a estrutura a partir do zero, existisse um módulo adicional onde fosse possível utilizar um banco de dados já existente e as aplicações web pudessem ser geradas também utilizando a tecnologia JSP, atendendo assim a ambientes Windows e Linux. Também foi criado suporte para outros bancos de dados, tais como MySQL e PostgreSQL, tanto para a criação de um banco de dados novo como para a importação dos dados a partir de um banco de dados existente.

Para descrever a aplicação a ser gerada, o usuário necessita cadastrar algumas informações sobre a aplicação que deseja gerar.

Em primeiro lugar, o usuário deve criar um projeto, o qual será a base para a geração da aplicação. Conseqüentemente, cada projeto gerará uma aplicação.

O próximo passo que o usuário deverá seguir será o cadastro dos grupos de informação. Os grupos de informação, para a ferramenta, são como as tabelas para o banco de dados.

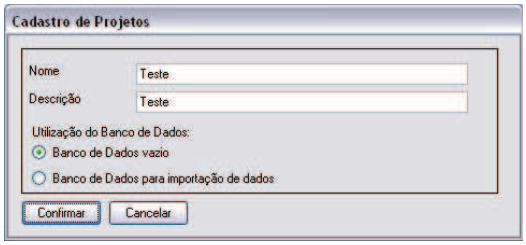
Para complementar o cadastro dos grupos de informação, existe também o cadastro dos itens de cada grupo. Os itens de um grupo de informação estão para o grupo de informação de informação como as colunas estão para uma tabela num banco de dados, ou seja, são nada mais do que atributos da entidade criada.

Além destes três cadastros o usuário deverá também cadastrar os modelos de página que deseja criar para cada grupo. Após a escolha dos modelos de páginas o usuário poderá realizar o cadastro de metadados para cada página selecionada.

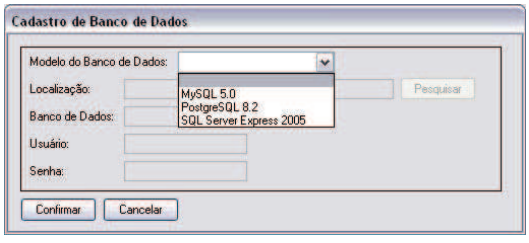
Para completar, o usuário pode definir itens para um cadastro de usuários da aplicação gerada. Podem ser definidos campos de texto, combos de seleção, listas de opções, etc.

É possível também escolher dentre três bancos de dados: SQL Server Express, PostgreSQL e MySQL.

A seguir serão apresentadas as telas do sistema a fim de dar melhor entendimento do mesmo e exemplificar o seu uso de maneira clara e intuitiva.

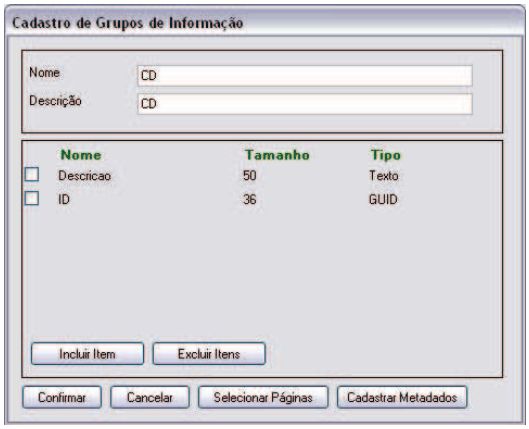


(a)



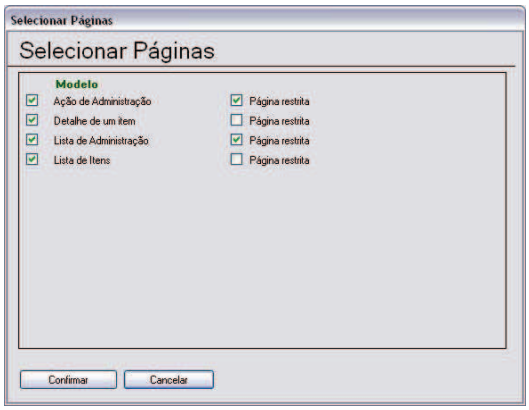
(b)

(c)

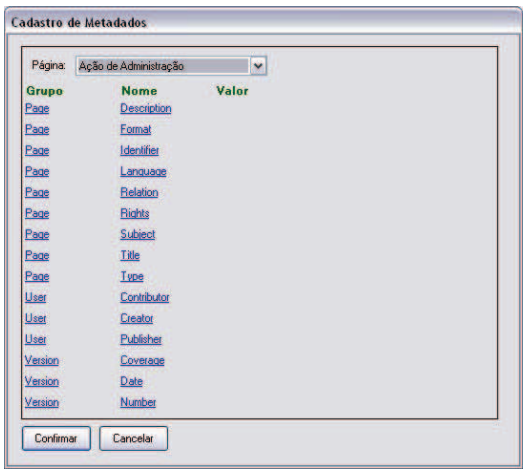


(d)

(e)



(f)



(g)

Figura 5: Telas do sistema gerador de websites. (a) Cadastro de Projetos; (b) Escolha de bancos de dados; (c) Opções principais do projeto selecionado; (d) Cadastro de Grupos de Informação; (e) Cadastro de Itens; (f) Selecionar Páginas; (g) Cadastro de Metadados.

3.2. Experimentos

Os experimentos com o WebMaker se concentraram na geração com sucesso do website para cada par linguagem de programação/servidor de banco de dados. Foram realizados testes para bancos de dados novos e importados e para as linguagens ASP.NET e JSP. Os testes envolveram a observação de recuperação de tipos de dados, tradução destes dados em grupos de informação, geração, compilação e uso do sistema.

Como testes mais relevantes a ser descrito, temos os realizados pelos alunos de graduação do curso de Informática do Instituto Superior de Tecnologia da FAETEC – Fundação de Apoio à Escola Técnica do Rio de Janeiro.

Os alunos da FAETEC utilizaram um banco de dados em PostgreSQL para importação pelo sistema. O banco de dados é composto de 85 tabelas, das quais foram selecionadas 17 tabelas de um subsistema de controle do acervo de uma biblioteca que os alunos precisavam desenvolver. Para realizar este desenvolvimento, os alunos da FAETEC utilizaram o WebMaker.

Inicialmente, o sistema WebMaker foi apresentado aos alunos de graduação em Informática da FAETEC através de um Manual do Usuário e de apresentação presencial, exibindo cada uma das telas do sistema e possibilidades de uso.

A partir deste ponto, os alunos iniciaram o uso do WebMaker e reportaram os erros encontrados através de e-mail e contato telefônico. Após as correções dos erros reportados, se tornou necessária outra visita aos alunos para identificação de refinamentos na ferramenta.

Após a correção dos erros encontrados e ajustes do WebMaker às variantes dos tipos de dados e associação entre tabelas do PostgreSQL, se tornou possível realizar a geração do website desejado.

Outros testes foram realizados, em condições diversas de importação e criação de bancos de dados novos para ambas as linguagens e, em todos, a aplicação web foi gerada com sucesso.

A seguir, apresentamos os resultados obtidos através dos testes realizados pelos alunos de graduação do Instituto Superior de Tecnologia da FAETEC.

3.3. Resultados

Durante os testes, ocorreram erros na execução principalmente com relação ao uso do banco de dados em PostgreSQL e à linguagem de programação JSP. Como a linguagem JSP é rígida em termos de tipos de dados, ocorreram erros de incompatibilidade entre os diversos tipos de dados entre as camadas de Controller e DAO.

Também se tornou necessário tratamento especial para o tipo de dado para data, sendo necessário também modificar o tipo de dados que manipula as datas no website gerado.

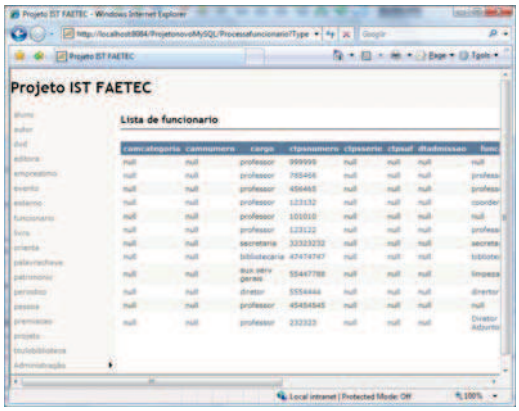
Após exaustivos testes no banco de dados produzido pelos alunos, foram identificados os erros de construção de SQLs e recuperação de tipos de dados, sendo devidamente tratados pelo sistema e corrigidos.

Identificou-se também erros de tradução dos tipos *java.util.UUID*, utilizado para o tratamento de GUIDs no sites em JSP, que foram tratados e corrigidos corretamente.

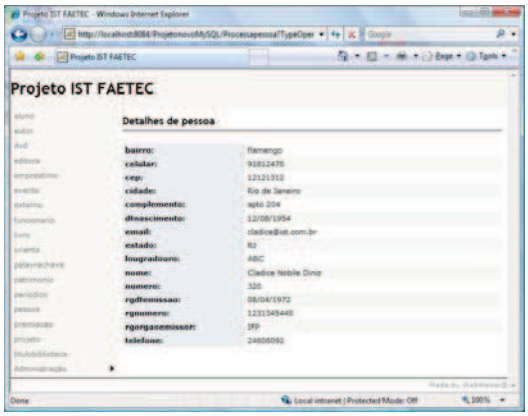
Houve também problemas na recuperação das chaves estrangeiras entre as tabelas do banco de dados em PostgreSQL, e tornaram-se necessárias novas consultas SQL no sistema para que o mesmo recuperasse corretamente as chaves estrangeiras somente das tabelas selecionadas pelo usuário e assim criar o relacionamento entre os grupos de informação.

A correção para os erros encontrados foram difundidos para os outros bancos de dados, no caso MySQL e SQL Server Express 2005, prevenindo assim a ocorrência de novos erros.

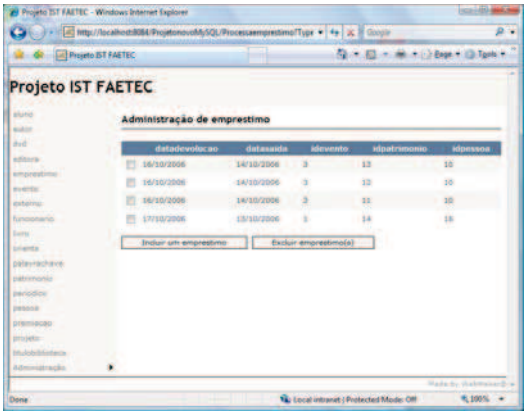
Por fim, ao concluir as correções no sistema, o website foi gerado conforme esperado. A seguir são apresentadas telas do referido website gerado pelos alunos.



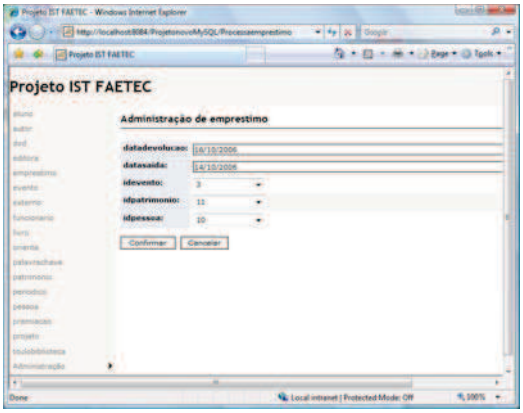
(a)



(b)



(c)



(d)

Figura 6: Telas de aplicação web gerada. (a) Lista de itens de um grupo de informação; (b) Detalhes de um item de um grupo de informação; (c) Lista de administração de itens de um grupo de informação; (d) Cadastro de um item de um grupo de informação.

4. Trabalhos Relacionados

Todos os projetos que são descritos nesta seção são gestores de conteúdos que fornecem a possibilidade de criação de várias formas de layout de websites e provêem diversos temas visuais.

Contrastando-os com o sistema proposto neste artigo, vale a pena lembrar que o WebMaker não tem como foco questões relacionadas à layout, fornecendo apenas um tipo de layout padrão. Em contrapartida, todos os gestores de conteúdo pesquisados são capazes de criar e gerenciar páginas apenas sobre grupos de informação separados uns dos outros, sem permitir a criação de relacionamentos entre os mesmos, fator que faz do sistema proposto por este artigo um diferencial no mundo dos geradores de aplicações web uma vez que o mesmo possui esta característica. A seguir, serão mostradas algumas características específicas de cada um dos projetos pesquisados.

- **Apache – Jakarta Jetspeed**

  Requer apenas a plataforma standard do J2EE para operar. (fator que motiva a sua utilização pelos defensores do open-source).

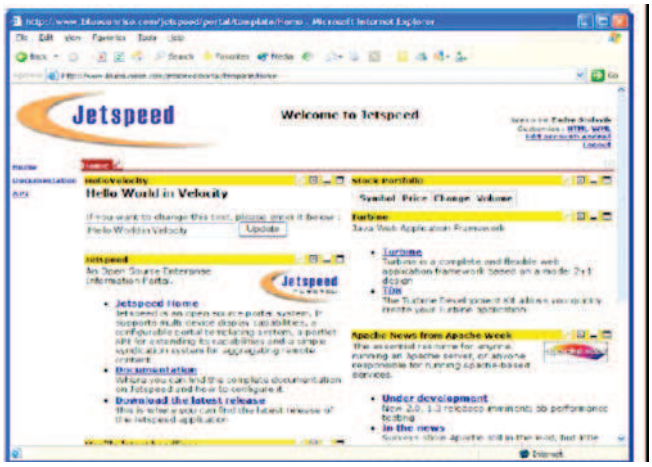


Figura 7: Interface do Jetspeed

- **BEA – Weblogic Portal**

  Necessita de um servidor Weblogic para funcionar, não satisfazendo o padrão standard J2EE.

Figura 8: Interface do BEA Weblogic

- **Vignette – Vignette Application Portal**

  Não requer um servidor proprietário para executar e satisfaz, portanto, o padrão standard J2EE. Possui alguns diferenciais como: possibilidade de arquitetura em duas camadas, permitindo a construção de sistemas distribuídos; suporte a integração com Web Services; suporte ao mecanismo WYSIWYG (*"What You See Is What You Get"*), o que permite a construção do site de maneira mais visual.

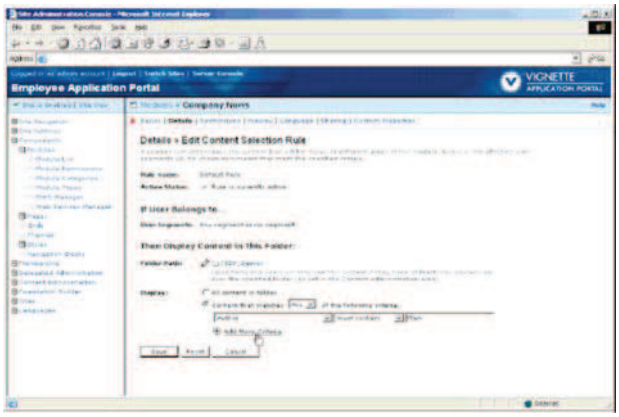


Figura 9: Interface do Vignette

- **IBM – WebSphere Portal**

  Possui um módulo dedicado a comércio eletrônico. Necessita de um servidor proprietário para rodar, o WebSphere Server, não seguindo, portanto, o padrão standard J2EE.

Figura 10: Interface do WebSphere

- **Microsoft – Content Management Server 2002**

  Mais facilmente utilizado, uma vez que sua plataforma de utilização é integrada ao Word, um dos editores de texto mais utilizados. Possui como problemas o fato de que a gestão de papéis só pode ser realizada através da própria ferramenta, fazendo com que os gestores necessitem da ferramenta instalada e de certos direitos sobre a máquina para manter o conteúdo criado, o que praticamente acaba com as possibilidades de gestão à distância. Outra desvantagem é o fato de que o Content Manager necessita do Sistema Operacional Windows 2000 Server ou Windows 2003 Server, e do Microsoft SQL Server 2000 como Banco de Dados para operar, o que faz com que este seja um dos ambientes estudados que mais torna o cliente dependente do fabricante.

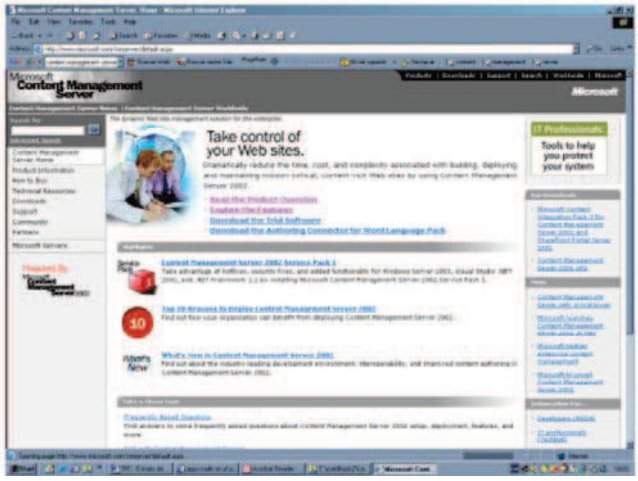


Figura 11: Interface do Microsoft Content Management Server 2002

- **Oracle – Oracle Portal**

  Possui praticamente todas as características do Vignette Application Portal, como possibilidade de criação de arquitetura em duas camadas, mecanismo WYSIWYG. Porém ao contrário do Vignette, este necessita do servidor Application Server e do sistema de banco de dados proprietário da Oracle para operar. Sendo assim, não satisfaz o padrão standard J2EE.

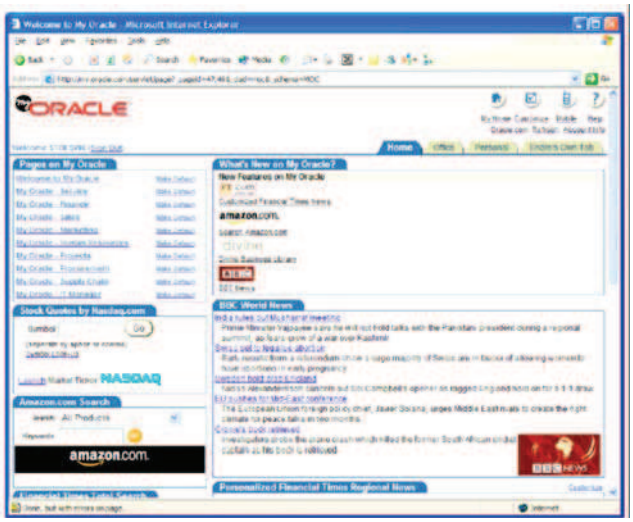


Figura 12: Interface do Oracle Portal

5. Considerações Finais

Com o crescimento da Internet, vem se tornando cada vez mais necessária a existência de ferramentas que acelerem o processo de desenvolvimento de aplicações web. É crescente também o volume de demandas por novos sistemas de informação nas empresas modernas. Em geral as áreas de Tecnologia da Informação das organizações têm dificuldade em atender com agilidade todas as solicitações. Diante deste cenário, o presente artigo teve como objetivo descrever o desenvolvimento e os testes de um ambiente computacional, o WebMaker, para ser utilizado na criação de websites transacionais que utilizassem páginas semânticas. O WebMaker permite gerar websites em

sua estrutura completa, desde o banco de dados até as páginas, bem como estruturas de ligação com o banco. Tal ambiente, que também pode ser utilizado para fins didáticos de ensino a programadores iniciantes, pode contribuir para a redução do *backlog* das áreas de TI das empresas para a produção ágil de novos sistemas de informação.

Com o WebMaker é possível criar não somente websites em sua estrutura, incluindo banco de dados, como também é possível gerar o website levando em consideração um banco de dados já existente. Esta segunda opção se tornou um desafio em particular, pois foi necessário realizar uma pesquisa a fundo dos bancos de dados suportados e também como o WebMaker deveria se comportar para identificar e traduzir os dados de um ambiente desconhecido em grupos de informação, conhecidos e facilmente manipuláveis pelo sistema.

Com a funcionalidade de Cadastro de Metadados, é possível cadastrar informações para todos os 15 elementos expostos pelo padrão Dublin Core da Web Semântica. Essas informações cadastradas para cada uma das páginas a serem criadas pelo WebMaker, são escritas levando-se em consideração a linguagem OWL, linguagem esta que tem sido uma das mais utilizadas na criação de ontologias. Estes arquivos OWL, por sua vez, são associados às páginas criadas.

Os resultados obtidos através dos testes realizados com o WebMaker indicam que os websites são criados com sucesso em qualquer opção oferecida, fornecendo condições padronizadas para a utilização das aplicações web geradas com sucesso sem a necessidade de experiência em linguagens de programação ou em manipulação de bancos de dados. Analisando tais resultados, é correto afirmar que o objetivo do sistema em gerar aplicações web foi atingido com sucesso.

É possível utilizar o WebMaker como ferramenta de aprendizado nas linguagens de programação ASP.NET e JSP, visto que o WebMaker produz websites funcionais nas duas linguagens.

O fato do WebMaker também gerar os websites obedecendo a arquitetura MVC também contribui de forma significativa no aprendizado para o desenvolvimento de websites utilizando orientação a objeto e padrões de projeto.

É possível relacionar algumas sugestões de melhorias, as quais foram encaminhadas como trabalhos futuros. Dentre estas, podemos citar:

- Possibilidades de escolha de quais itens de um grupo de informação serão exibidos na listagem no website gerado;
- Suporte à interligação entre os grupos de informação através de chaves compostas;
- Edição do seu layout em tempo de desenvolvimento e assim possibilitar uma maior personalização da aparência dos websites gerados;
- Suporte à inclusão de tipos de itens de Imagem para os grupos de informação, para que seja possível a inclusão de imagens em cada um dos registros;
- Suporte para outros bancos de dados, tais como Oracle e DB2 e de novas linguagens de programação, tais como PHP5, Ruby ou Python.

6. Referências

[1] W3C RECOMENDATION; OWL Web Ontology Language Semantics and Abstract Syntax; **W3C**; Fev. 2004; Disponível em <http://www.w3.org/TR/owl-absyn/>; Acesso em: 15 de maio de 2007;

[2] SOUZA, M.I.F.; VENDRUSCULO, L.G.; MELO, G.C.; **Metadados para descrição de recursos de informação eletrônica: utilização do padrão Dublin Core**; 2000; Disponível em <http://www.scielo.br/pdf/ci/v29n1/v29n1a10.pdf>; Acesso em: 15 de maio de 2007;

[3] BREITMAN, K.; **Web Semântica – A Internet do Futuro; 1 ed.**; Rio de Janeiro: LTC; 2006;

[4] BERNERS, T.B.; HENDLER, J.; LASSILA, O; The Semantic Web – A new form of Web content that is meaningful to computers Will unleash a revolution of new possibilities. **Scientific American**, New York, mai. 2001;

[5] CORE J2EE PATTERN CATALOG; Core J2EE Patterns – Data Access Object; **Sun Developer Network**; Dez. 2007; Disponível em <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>; Acesso em: 01 de janeiro de 2008;

[6] JAVA BLUEPRINTS; Model-View-Controller Architeture; **J2EE Patterns**; Dez. 2007; Disponível em <http://java.sun.com/blueprints/patterns/MVC-detailed.html>; Acesso em: 01 de janeiro de 2008;