

Framework web baseado em taglibs para gerenciamento de conteúdo

Daniel Naschenweng

Alexander Roberto Valdameri - Orientador

Roteiro da apresentação

- Contextualização
- Objetivos
- Motivação
- Revisão bibliográfica
- Especificação
- Implementação
- Operacionalidade
- Conclusões
- Relevância Pessoal

Contextualização

- O desenvolvimento web exige hoje do *web designer* conhecimentos de programação para a utilização dos frameworks existentes.
- Implementação de um *framework* web que permite ao *web designer* o desenvolvimento de *sites* dinâmicos de conteúdos, através de uma biblioteca de *tags*.

Objetivos

- Implementar um *framework* web visando facilitar a futura construção de *web sites* de conteúdo com estrutura de dados e diagramação visual flexíveis;
- Desenvolver uma biblioteca de *tags*;
- Produzir uma ferramenta para gerenciamento de conteúdos.

Motivação

- O desenvolvimento deste trabalho se deu pela falta de um *framework* web que seja voltado para o *web designer*. Os *frameworks* disponíveis no mercado exigem conhecimento de programação para sua utilização, o que dificulta o processo de desenvolvimento atual.

DESENVOLVIMENTO WEB BASEADO EM BIBLIOTECAS DE *TAGS*

- O uso de bibliotecas de *tags* personalizadas faz com que se possa escrever as páginas da interface sem utilizar código de programação.
- Com o uso delas é possível alterar o *layout* da página sem modificar a lógica e alterar esta última sem influir no *layout*

ELEMENTOS DE UMA TAG PERSONALIZADA

- *Tags* personalizadas podem possuir até três partes:
 - a) A *tag* inicial, como `<demo:hello>`;
 - b) A *tag* final, como `</demo:hello>`;
 - c) O corpo da *tag* que é todo o conteúdo entre a *tag* inicial e a final, podendo conter elementos de *script*, textos HTML e até mesmo outras *tags*.

COMPARTILHAMENTO DE OBJETOS

- Outro recurso é a cooperação entre *tags*, isto é realizado através de objetos compartilhados. Ex. *tags* aninhadas:

```
<tt:outerTag>  
  <tt:innerTag />  
</tt:outerTag>
```

DESCRITORES DE BIBLIOTECAS DE TAGS

- O descritor mapeia o nome da *tag* usada na página JSP e a classe Java que implementa a *tag*. Este descritor é denominado *Tag Library Descriptor* (TLD).

Exemplo de TLD:

```
<?XML version="1.0" encoding="ISSO-8859-1" ?>
<!DOCTYPE taglib PUBLIC
    "-//SUN Microsystems, Inc. //DTD JSP Tag Library 1.1//EN"
    "http://java.sun.com/dtd/web-jsptaglibrary_1_1.dtd">
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>1.1</jsp-version>
  <short-name>mut</short-name>
  <info>Utility Tags for JSP.</info>
  <uri>http://www.taglib.com/wdjsp/tlds/mut_1_0.tld</uri>
  <tag>
    <name>forProperty</name>
    <tag-class>com.taglib.wdjsp.mut.ForPropertyTag</tag-class>
    <tei-class>com.taglib.wdjsp.mut.ForPropertyTEI</tei-class>
    <body-content>JSP</body-content>
    <attribute>
      <name>attr1</name>
      <required>>true</required>
      <rtexprvalue>>true</rtexprvalue>
      <type>java.lang.String</type>
    </attribute>
  </tag>
</taglib>
```

TRATADORES DE *TAGS*

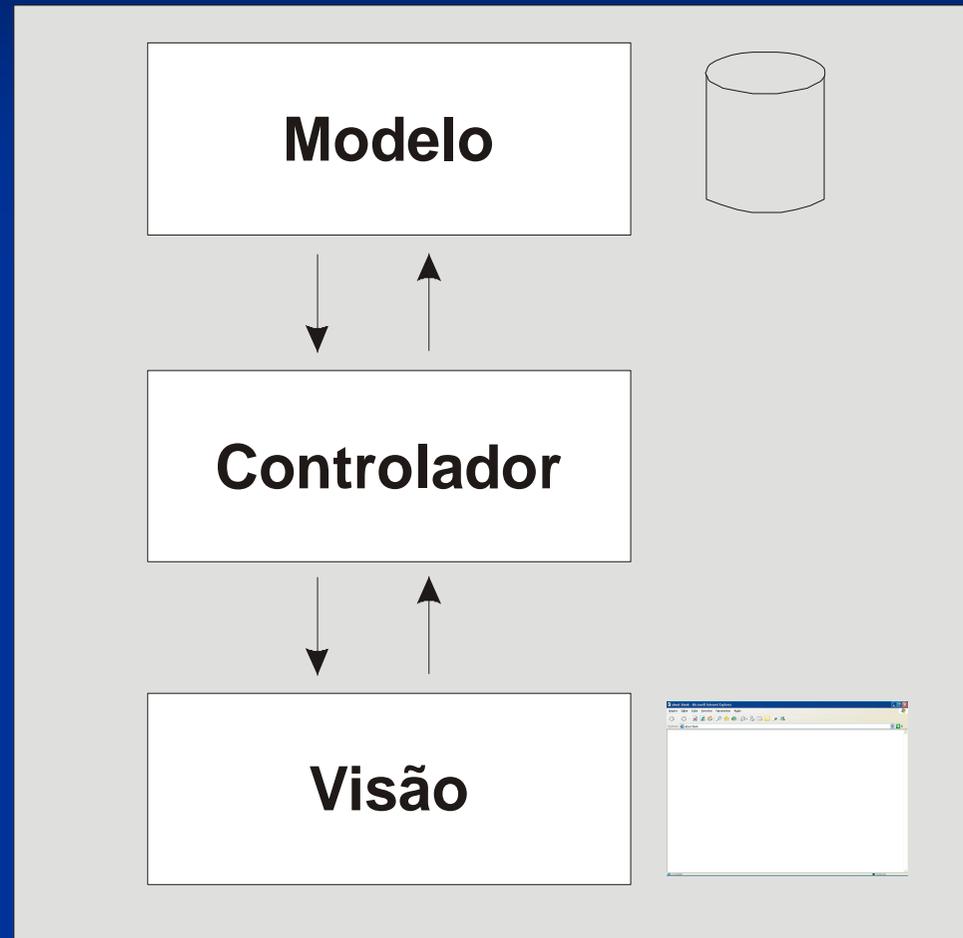
Quando o *container* web executa uma página JSP que possui uma *tag* personalizada JSP, são executadas operações por um objeto denominado de tratador de *tags*.

Este tratador de *tags* deve implementar a interface Tag ou BodyTag do J2EE.

MODELO-VISÃO- CONTROLADOR (MVC)

- O *framework* para interfaces gráficas MVC divide uma aplicação em três partes ou papéis: modelo (dados e classes de negócios), visão (apresentação) e controle.
- A grande vantagem do padrão MVC é simplificar a manutenção e evolução de aplicações web, bem como o reuso de seus componentes.

Cenário de interação do *framework* MVC na web



GERENCIADOR DE CONTEÚDOS

- No conceito de CMS, o conteúdo representa informações puras ainda não formatadas para a exibição em *sites*. Este conteúdo pode ser, por exemplo, arquivos-texto, imagens, planilhas, entre outros.

TRABALHOS CORRELATOS

- O *framework* Struts (STRUTS, 2005) visa facilitar aos desenvolvedores a criação de aplicações web baseadas em tecnologias Java *servlets* e JSP;
- Com a utilização do Struts o desenvolvedor preocupa-se mais com as regras de negócios do que com a infraestrutura do *site*.

TRABALHOS CORRELATOS

- Jakarta TagLibs (JAKARTA TAGLIBS, 2005) que é um projeto *open source* onde se encontra a implementação de referência (RI) da JSTL;
- Entre as várias *tags* disponibilizadas estão *tags* de formatação, acesso à banco de dados, manipulação de XML e funções para uso diverso.

TRABALHOS CORRELATOS

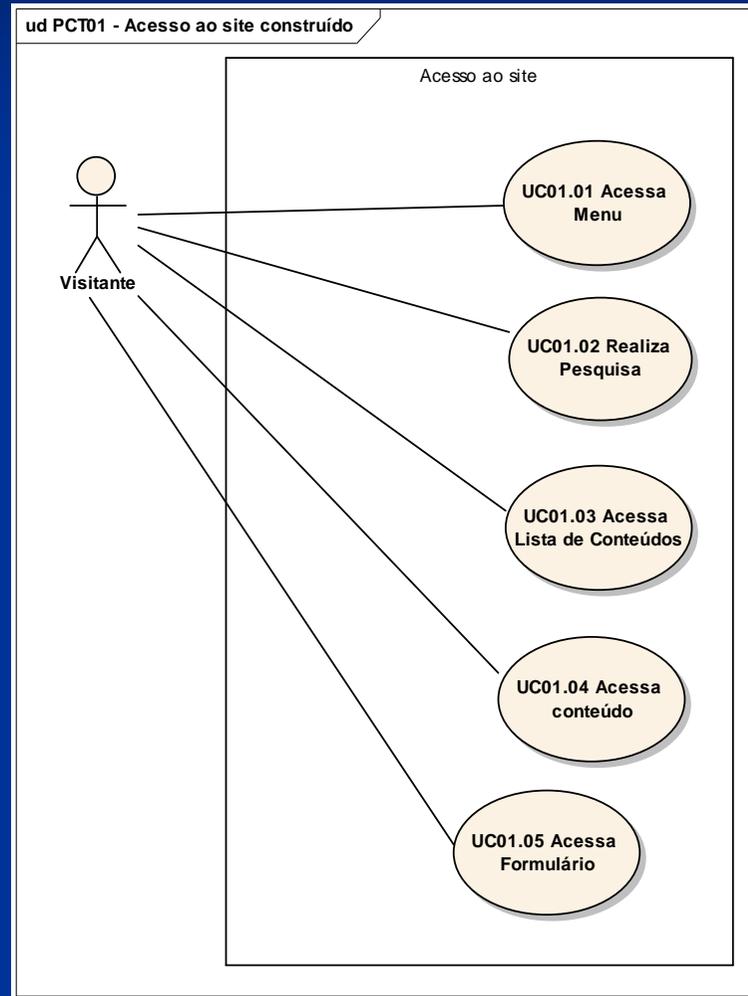
- O *framework* Makumba (MAKUMBA, 2005) implementado em Java oferece uma *Application Program Interface* (API) e uma biblioteca de *tags* para o desenvolvimento rápido de aplicações web;
- A estrutura de dados é descrita em um arquivo texto de acordo com uma linguagem específica.

TRABALHOS CORRELATOS

- Em Moratelli (2002) é realizado um estudo sobre sistemas de gerenciamento de conteúdos para ambiente web, visando facilitar o gerenciamento de conteúdo de um *site*.
- Também apresenta a especificação e implementação deste sistema utilizando a linguagem PHP: *Hipertext Preprocessor* (PHP).

ESPECIFICAÇÃO

Diagrama de Casos de Uso



ESPECIFICAÇÃO

Diagrama de Casos de Uso

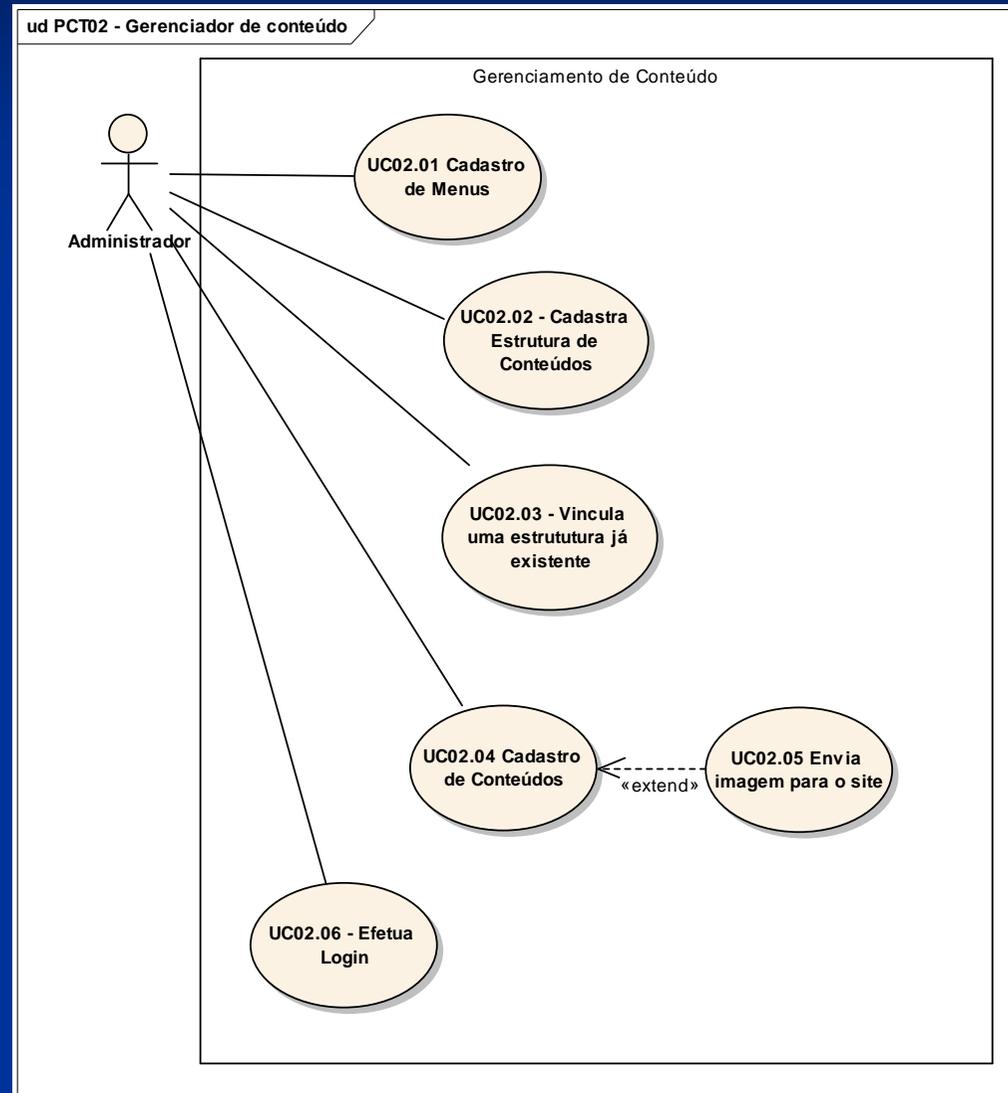


Diagrama de Classes

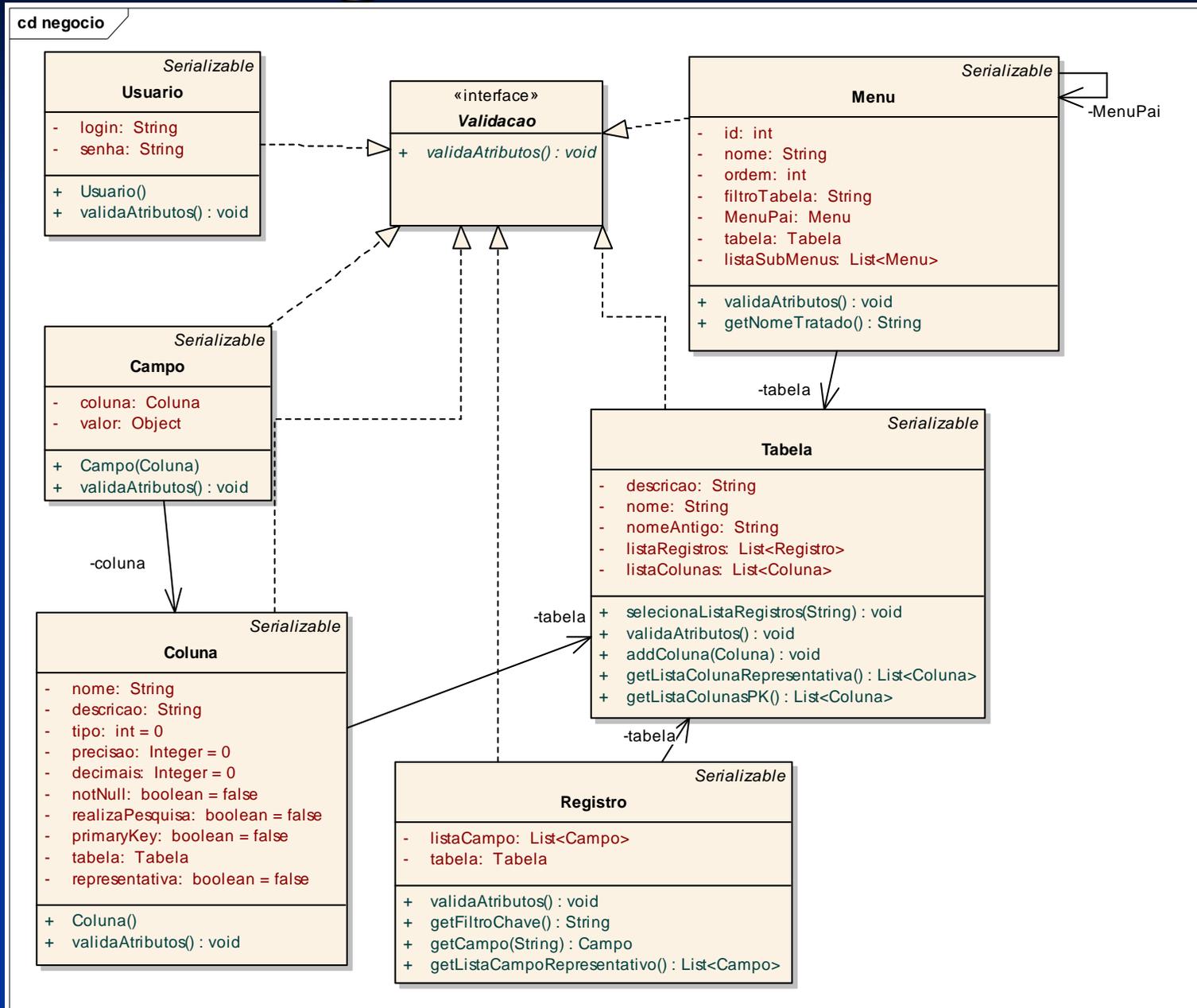


Diagrama de Classes

cd taglib

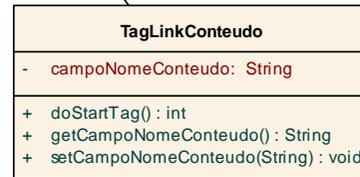
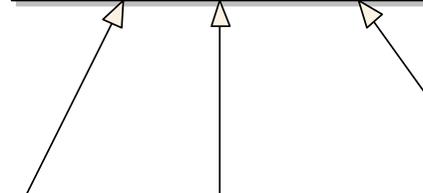
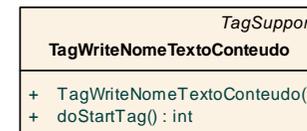
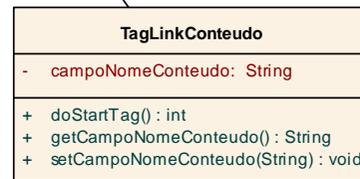
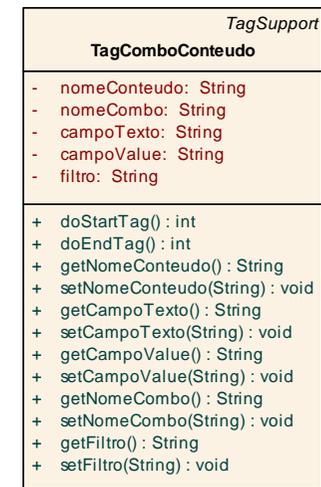
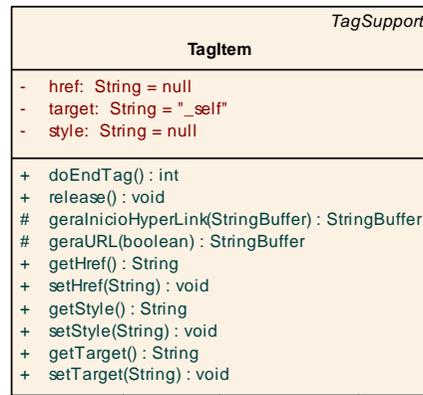


Diagrama de Classes

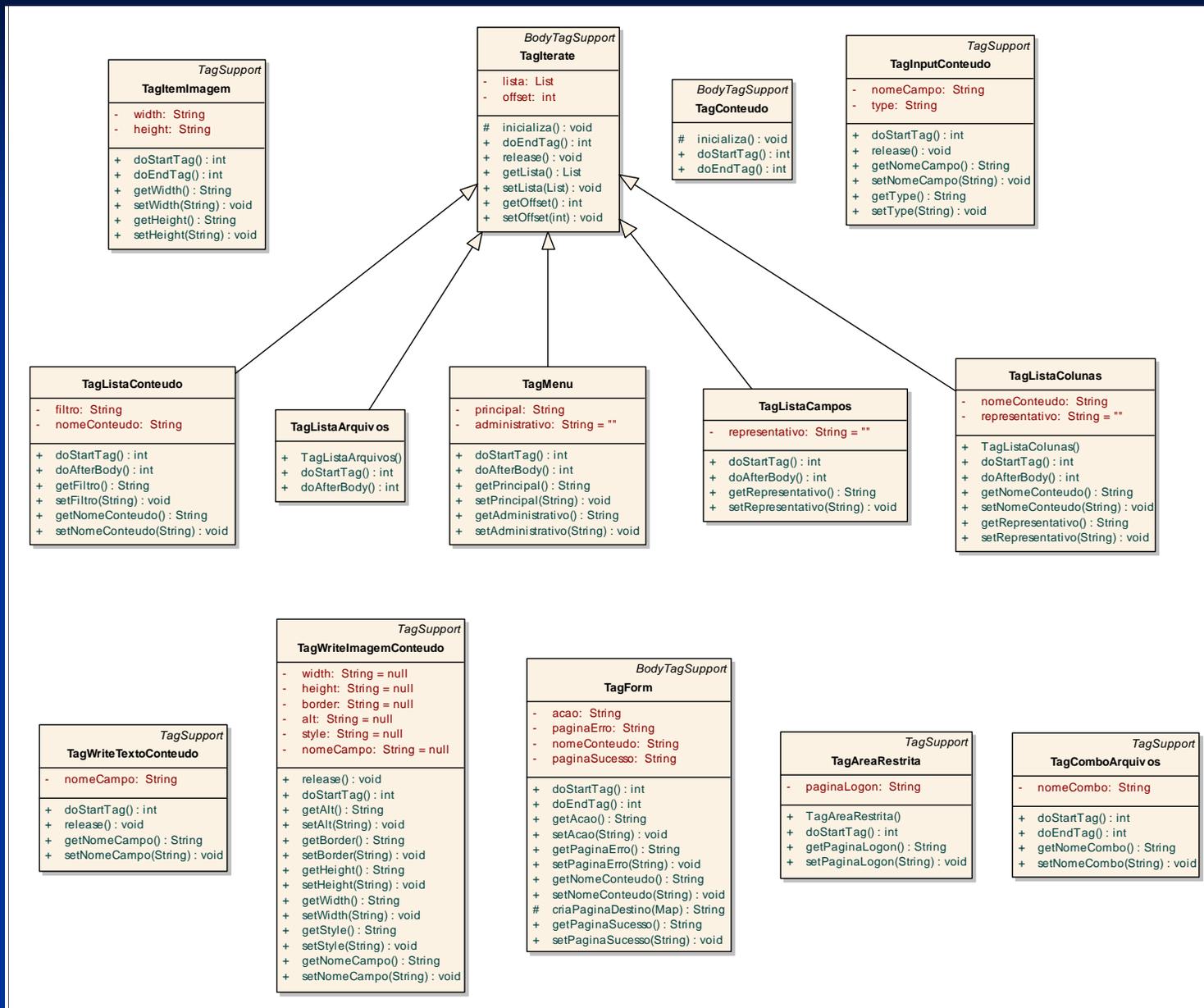
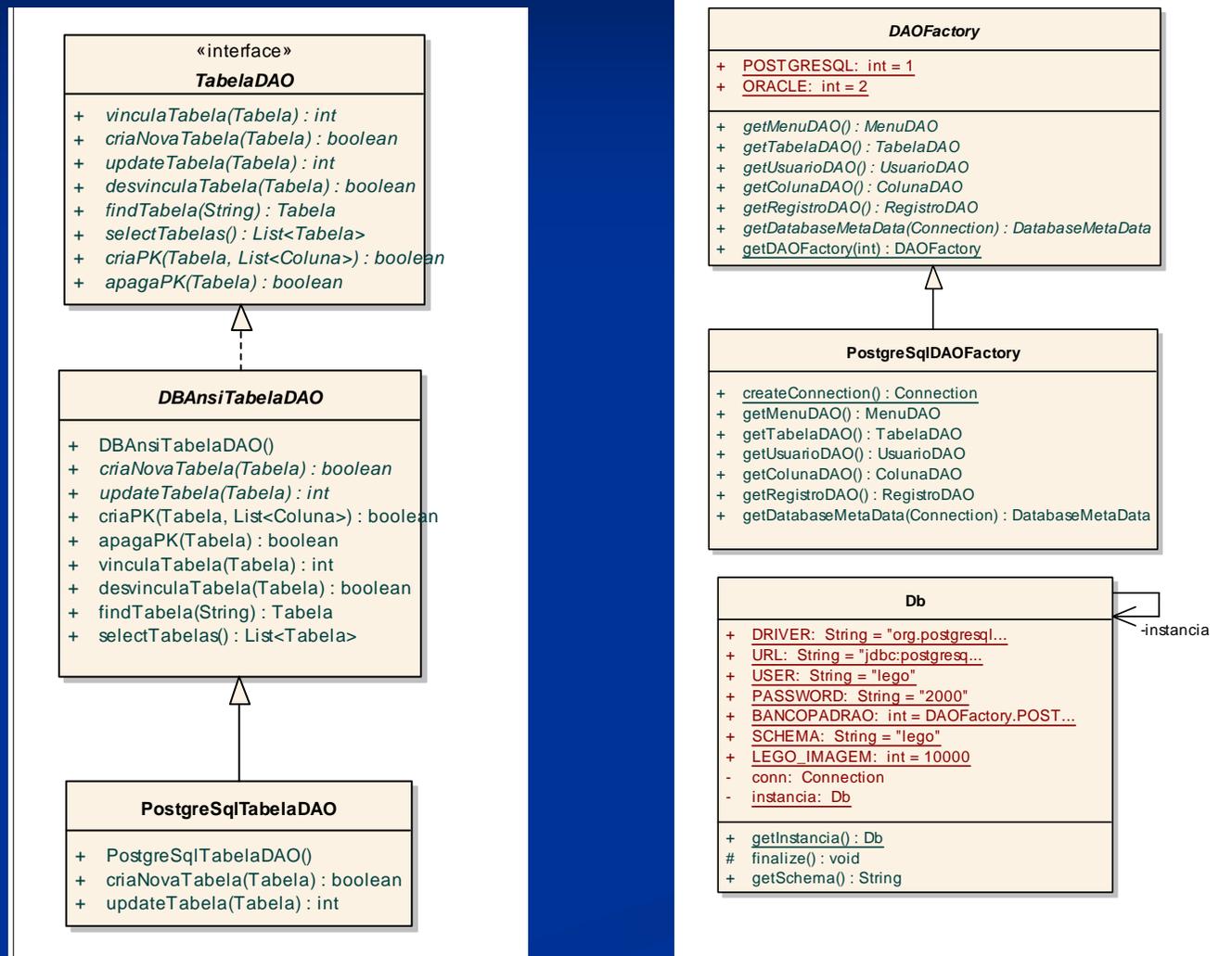


Diagrama de Classes



DESCRIÇÃO DAS TAGS

<tag>

```
<name>menu</name>
```

```
<tagclass>br.com.lego.taglib.TagMenu</tagclass>
```

```
<bodycontent>JSP</bodycontent>
```

```
<info>
```

O atributo principal é obrigatório e define se um menu é o menu principal ou se é um sub-menu.

O atributo administrativo somente é utilizado pela área de gerenciamento do próprio *framework*. Para o desenvolvimento de um *web site* este atributo pode ser ignorado.

```
</info>
```

```
<attribute>
```

```
  <name>principal</name>
```

```
  <required>>true</required>
```

```
  <rtexprvalue>>true</rtexprvalue>
```

```
</attribute>
```

```
<attribute>
```

```
  <name>administrativo</name>
```

```
  <required>>false</required>
```

```
  <rtexprvalue>>true</rtexprvalue>
```

```
</attribute>
```

```
</tag>
```

Classe tratadora de *tag*

```
public class TagIterate extends BodyTagSupport {
    private List lista;
    private int offset;
    protected void inicializa() {
        this.setBodyContent(null); // Limpa o corpo da tag
        this.setLista(null);
        this.setOffset(0);
        Db.getInstancia().setConnection((HttpServletRequest)
pageContext.getRequest());
    }

    /**
     * Método para tratamento do fim de tag de iteração.
     * @return Retorna EVAL_PAGE para que continue a ser tratado o restante
da página.
     * @throws javax.servlet.jsp.JspException Exceção JSP
     */
    public int doEndTag() throws JspException {
        JspWriter writer = pageContext.getOut();
        try {
            writer.print(this.getBodyContent() != null ?
this.getBodyContent().getString() : "");
        } catch (IOException IOe) {
            throw new JspException(IOe.getMessage());
        }
        return EVAL_PAGE;
    }
}
(...)
```

JSP utilizando a *tag* menu

```
<table width="800" border="0" align="center" cellpadding="0"
      cellspacing="0" bgcolor="#999999">
  <tr>
    <td>
      <table width="798" border="0" align="center" cellpadding="0"
            cellspacing="0" bgcolor="#FFFFFF">
        <tr>
          <td colspan="2" align="center">
            <lego:menu principal="S" administrativo="sim">
              <td width="15px">&nbsp;</td>
              <td width="" align="center">
                <lego:itemMenu href="/adm/*.jsp" />
              </td>
            </lego:menu>
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<table width="798" border="0" align="center" cellpadding="0"
      cellspacing="0" bgcolor="#999999">
  <tr><td height="2"></td>
</tr>
</table>
```

JSP utilizando outras *tags*

```
<lego:listaConteudo>
  <table width="550" border="1" align="center" cellpadding="2"
    cellspacing="0" bordercolor="#666666" bgcolor="#CCCCCC">
    <tr>
      <td>
        <table width="100%" border="0" align="center" cellpadding="1"
          cellspacing="2">
          <tr bgcolor="#CCCCCC">
            <td bgcolor="" width="50%">
              <lego:listaCampos representativo="sim">
                <lego:escreveNomeCampoConteudo />:
              <lego:escreveCampoConteudo /><br>
                </lego:listaCampos>
            </td>
            <lego:form acao="excluir">
              <td bgcolor="" width="50%" align="right">
                <lego:itemConteudo
href="/adm/editar_conteudo.jsp" target="" style="">
                  
                </lego:itemConteudo>
                <input type="image" name="imageField"
src="../img/excluir.GIF">
              </td>
            </lego:form>
          </tr>
        </table>
      </td>
    </tr>
  </table>
</lego:listaConteudo>
```

Operacionalidade

- Para demonstrar a operacionalidade do *framework*, escolheu-se a área de gerência.
- Ela foi desenvolvida utilizando as *tags* do próprio *framework*.

Autenticação de Usuário

Projeto de TCC Lego Designer - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço http://localhost:8084/LegoDesigner/adm/ Ir Links >>

Projeto de TCC Lego Designer

Estrutura	Cadastro	Menus	Usuários	Arquivos
---------------------------	--------------------------	-----------------------	--------------------------	--------------------------

Login:

Senha:

Concluído Intranet local

Tela de alteração de Estruturas

Projeto de TCC Lego Designer - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço <http://localhost:8084/LegoDesigner/adm/estrutura.jsp?idMenu=1> Ir Links >>

Projeto de TCC Lego Designer

- Estrutura
- Cadastro
- Menus
- Usuários
- Arquivos

Alteração de Estrutura

clientes	 
produtos	 

Concluído Intranet local

Alteração de colunas

The screenshot shows a web browser window titled "Projeto de TCC Lego Designer - Microsoft Internet Explorer". The address bar displays the URL: `http://localhost:8084/LegoDesigner/adm/estr_listaCampos.jsp?nomeConteudo=lego_tabela&filtroConteudo=tabela_nome%3D%27produtos%27&idMenu=1`. The page content includes a navigation menu with the following items: **Estrutura**, **Cadastro**, **Menus**, **Usuários**, and **Arquivos**. Below the menu, a section titled "Colunas da estrutura selecionada:" contains a table with two rows:

Colunas da estrutura selecionada:	
id	 
nome	 

The status bar at the bottom of the browser window shows "Concluído" on the left and "Intranet local" on the right.

Inclusão de nova coluna

Projeto de TCC Lego Designer - Microsoft Internet Explorer

Arquivo Editar Exibir Favoritos Ferramentas Ajuda

Endereço http://localhost:8084/LegoDesigner/adm/incluir_coluna.jsp?nomeConteudo=lego_coluna Ir Links >>

Projeto de TCC Lego Designer

Estrutura **Cadastro** **Menus** **Usuários** **Arquivos**

Inserir nova Coluna

Inserindo:

Nome:

Descricao:

Chave Primária:

Não aceitar Nulo?

Casas Decimais:

Tamanho:

Tipo:

Utilizada para pesquisa?

Nome Conteúdo:

Representativa?

Concluído Intranet local

Edição de registros

The screenshot shows a Microsoft Internet Explorer browser window displaying a web application. The browser's address bar shows the URL: `http://localhost:8084/LegoDesigner/adm/cad_listaConteudo.jsp?nomeConteudo=clientes`. The page title is "Projeto de TCC Lego Designer".

The application interface features a navigation menu with the following items: **Estrutura**, **Cadastro**, **Menus**, **Usuários**, and **Arquivos**. The "Cadastro" menu item is currently selected.

Below the navigation menu, there is a section titled "Escolha o registro a editar:" followed by a table of records. Each record includes an ID and a name, with edit and delete icons to the right.

Escolha o registro a editar:	
id: 2 nome: João	 
id: 5 nome: Mariane	 
id: 1 nome: Daniel	 
id: 6 nome: Beltrano	 
id: 3 nome: Mané	 
id: 4 nome: Gabriela	 

At the bottom of the browser window, the status bar shows "Concluído" on the left and "Intranet local" on the right.

Conclusões

- Implementado utilizando o conceito de biblioteca de *tags*;
- Visual ficou independente da programação
- Diagramação visual dos *sites* desenvolvidos tornou-se flexível;
- Ferramenta de gerência de conteúdos permite que os dados sejam estruturados diretamente pelo administrador do web *site*;

Conclusões

- O *framework* obedece ao padrão MVC, onde o modelo é gerenciado pela camada DAO, a visualização pelas *tags* disponibilizadas e o controle pelas classes controladoras das *tags*;
- Área de gerência de conteúdos foi construída com o próprio *framework*;
- Este trabalho alcançou todos os seus objetivos.

Relevância Pessoal

- Aplicação de conhecimentos em projetos de software;
- Conhecimentos adquiridos na área de desenvolvimento web em Java;
- Satisfação na conclusão de um projeto que considero interessante e útil.