FERRAMENTA PARA CONVERSÃO DE INTERFACES JAVA SWING EM HTML E JAVASCRIPT

Acadêmica: Marciele Fernanda Severo

Orientadora: Joyce Martins

Roteiro

- Introdução
  - Objetivos do trabalho
- Fundamentação teórica
  - Geradores de código, Componentes de interface e tratadores de evento em Java, HTML, Javascript, Motor de *templates* Velocity
- Desenvolvimento do sistema
  - Requisitos principais, Especificação, Resultados e discussão e Trabalhos correlatos
- Conclusão
  - Extensões

Introdução

- Demanda por sistemas web
  - Vantagens
- Reengenharia de software
  - Tradução do programa
  - Economia

Objetivos do trabalho

- converter componentes de interface Swing
- converter tratamentos de eventos de interface
- utilizar *templates* para configurar o formato do código de saída
- analisar o grau de compatibilidade entre o código gerado e o código de entrada da ferramenta

Fundamentação Teórica

- Geradores de código
- Componentes de interface e tratadores de evento em Java
- HTML e Javascript
- Motor de *templates* Velocity

Desenvolvimento do Sistema

Requisitos Principais funcionais

- Converter componentes de interface Java Swing para HTML
- Converter tratadores de eventos da linguagem Java para JavaScript

Requisitos principais não funcionais

- Manter o layout original dos componentes de interface e a funcionalidade dos tratadores de evento
- Utilizar templates para a geração do código de saída
- Utilizar linguagem Java 5.0 para implementação da ferramenta
- Executar em ambiente Windows XP e Vista
- Executar no navegador Internet Explorer 7

Primeira e segunda etapa para a geração de código

jLabel5 = new JLabel(); jLabel5.setText(""); jLabel5.setPreferredSize(new Dimension(17, 20));	

Template para conversão do JLabel


```
<font name=$!{Component.getName()}
   style= "position: absolute;font-family:
$!{Component.getFontFamily()};
   font-size: $!{Component.getFontSize()};
   font-weight: bold;
   width: $!{Component.getSizeWidth()};
   height: $!{Component.getSizeHeight()};
   left: $!{Component.getPositionLeft()};
   top: $!{Component.getPositionTop()};">
$!{Component.getText()}
  </font>
```

```
 <font name=jLabel5
   style= "position:
absolute;font-family:
dialog;
   font-size: 12;
   font-weight: bold;
   width: 17;
   height: 20;
   left: 0;
   top: 0;">
 </font>
```

Escrevendo a função Javascript

<HTML> <HEAD> <TITLE> $!{JFrameGetTitle} </TITLE> </HEAD> <BODY> $!{funcJavaScript} </BODY> </HTML>	String f = verifyFunctions(); **if** (f.length() > 26) { context.put("funcJavascript", f); }

Diagrama de casos de uso

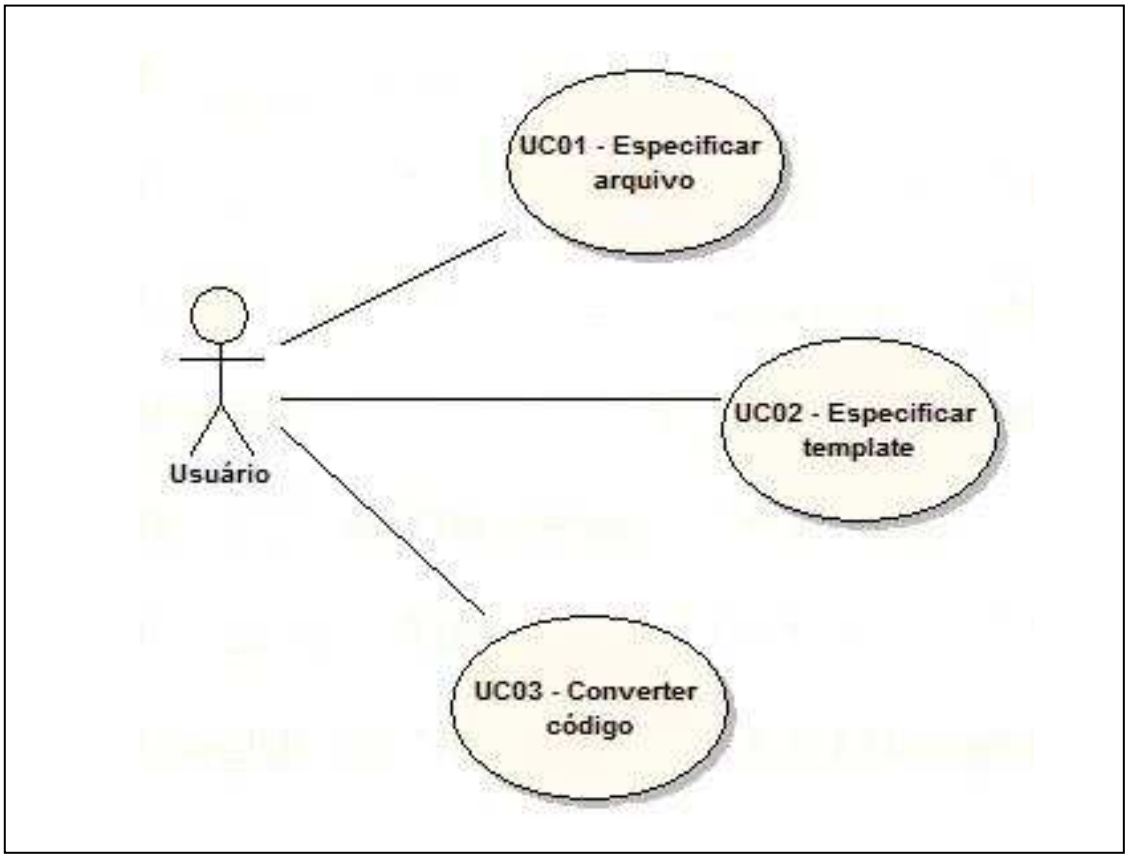


Diagrama de classes

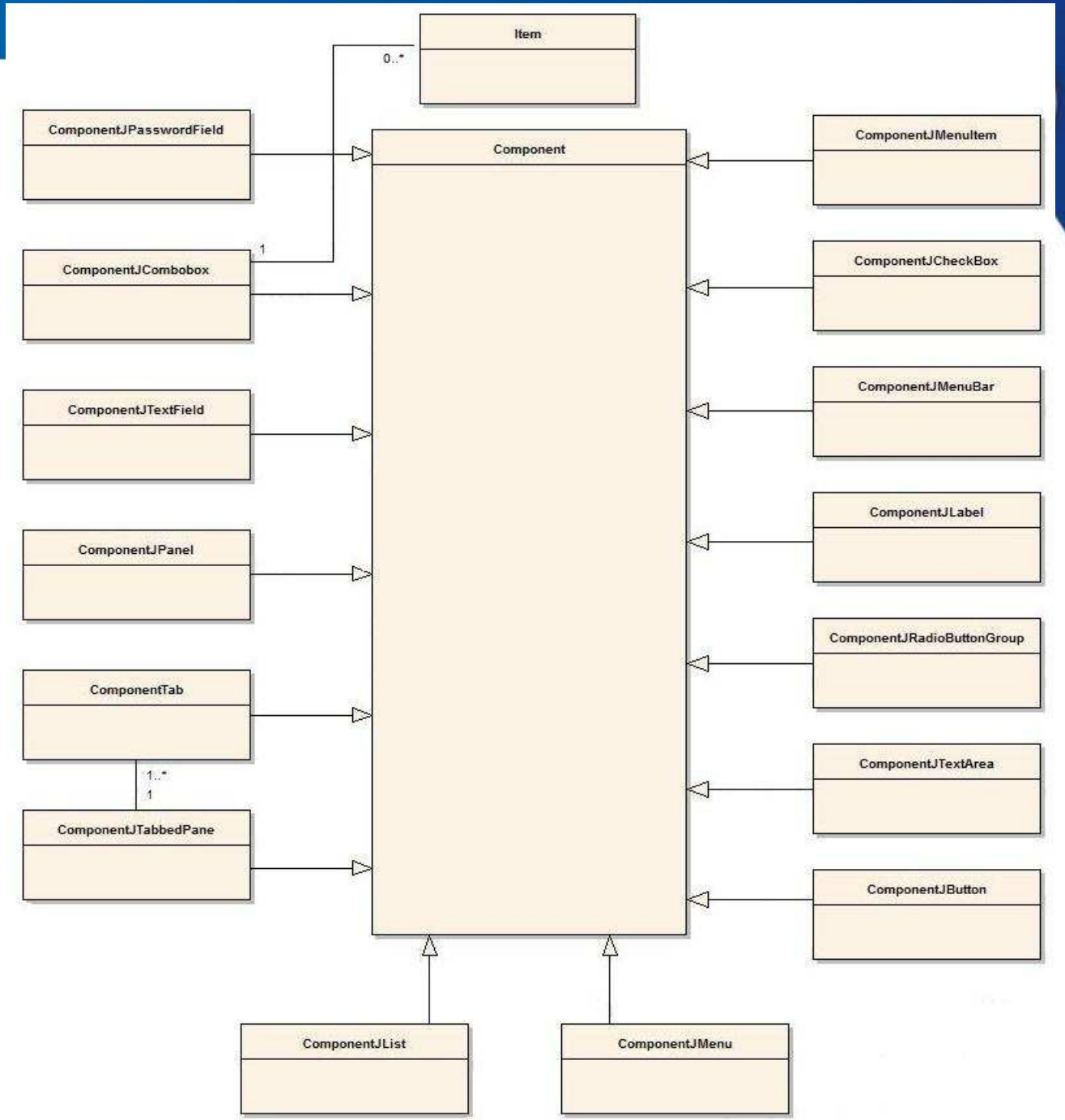


Diagrama de classes

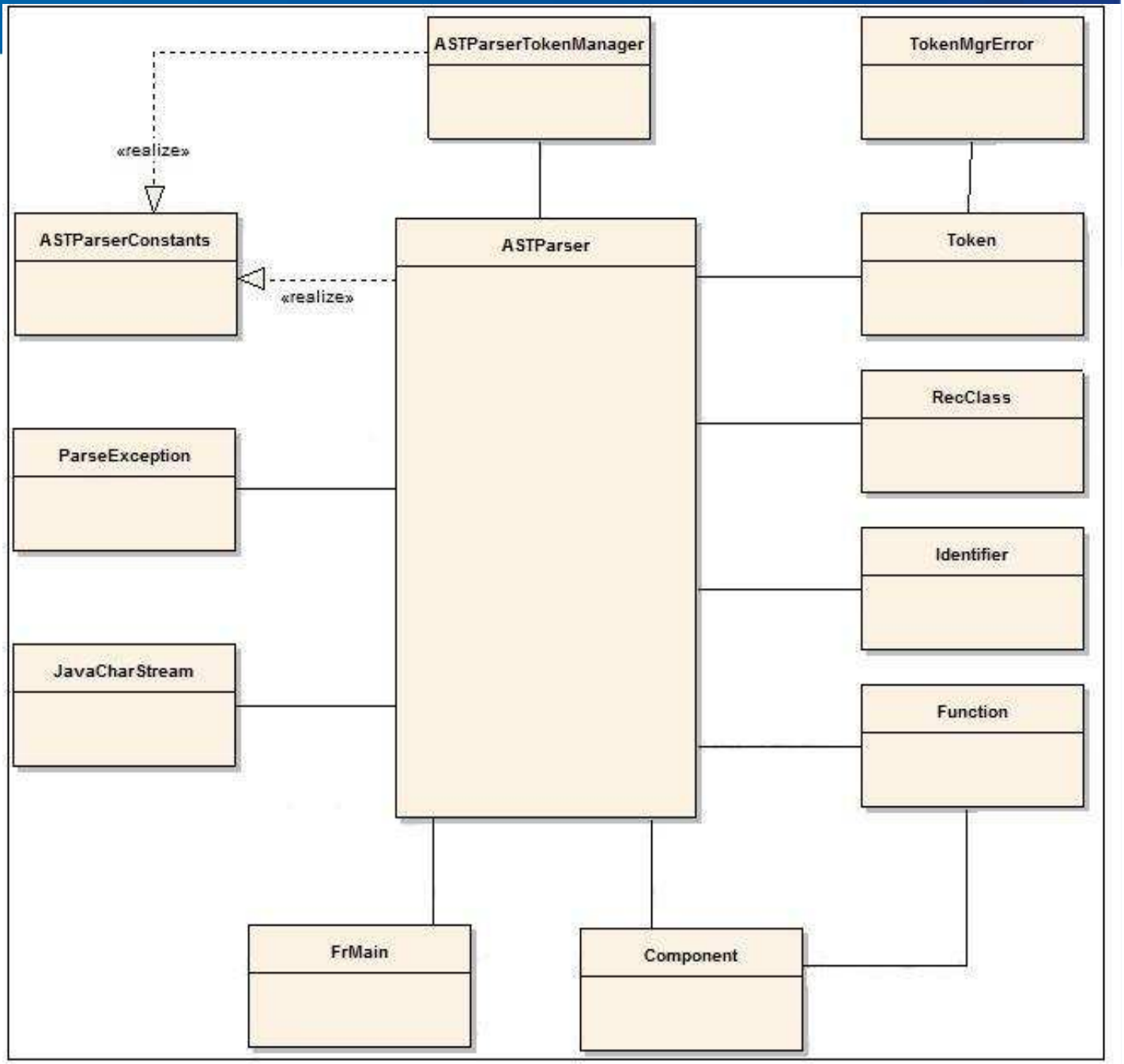


Diagrama de sequência

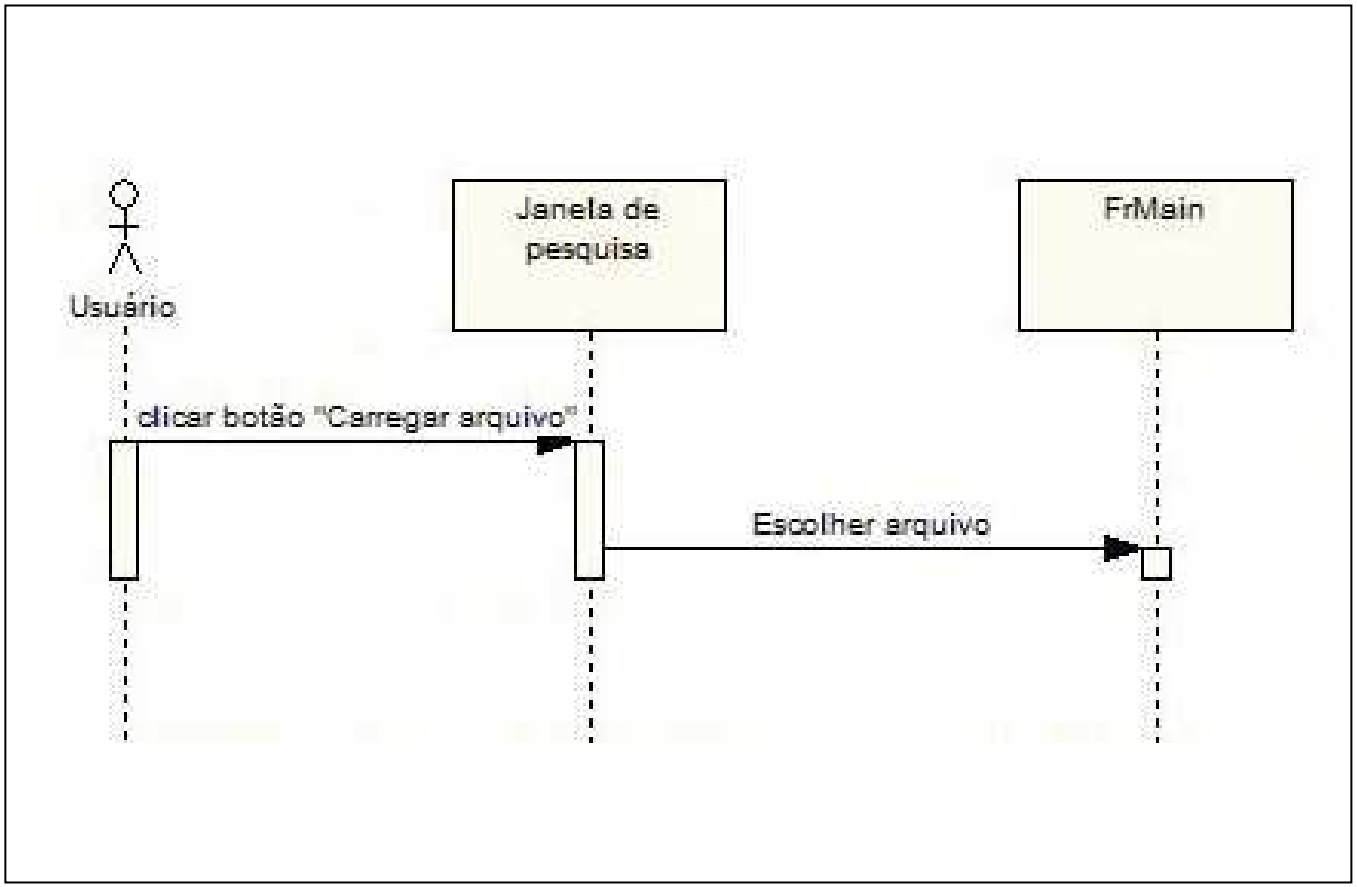


Diagrama de sequência

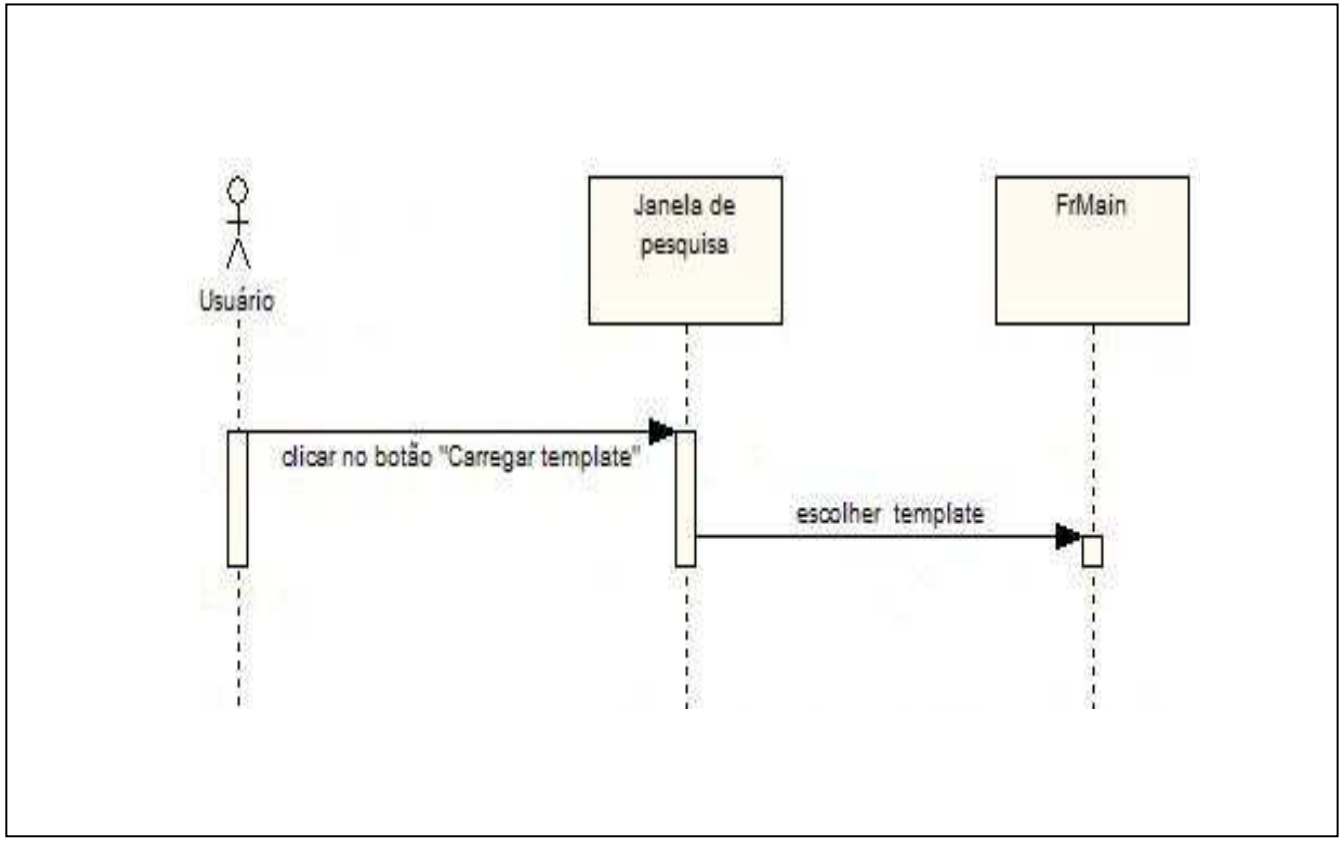
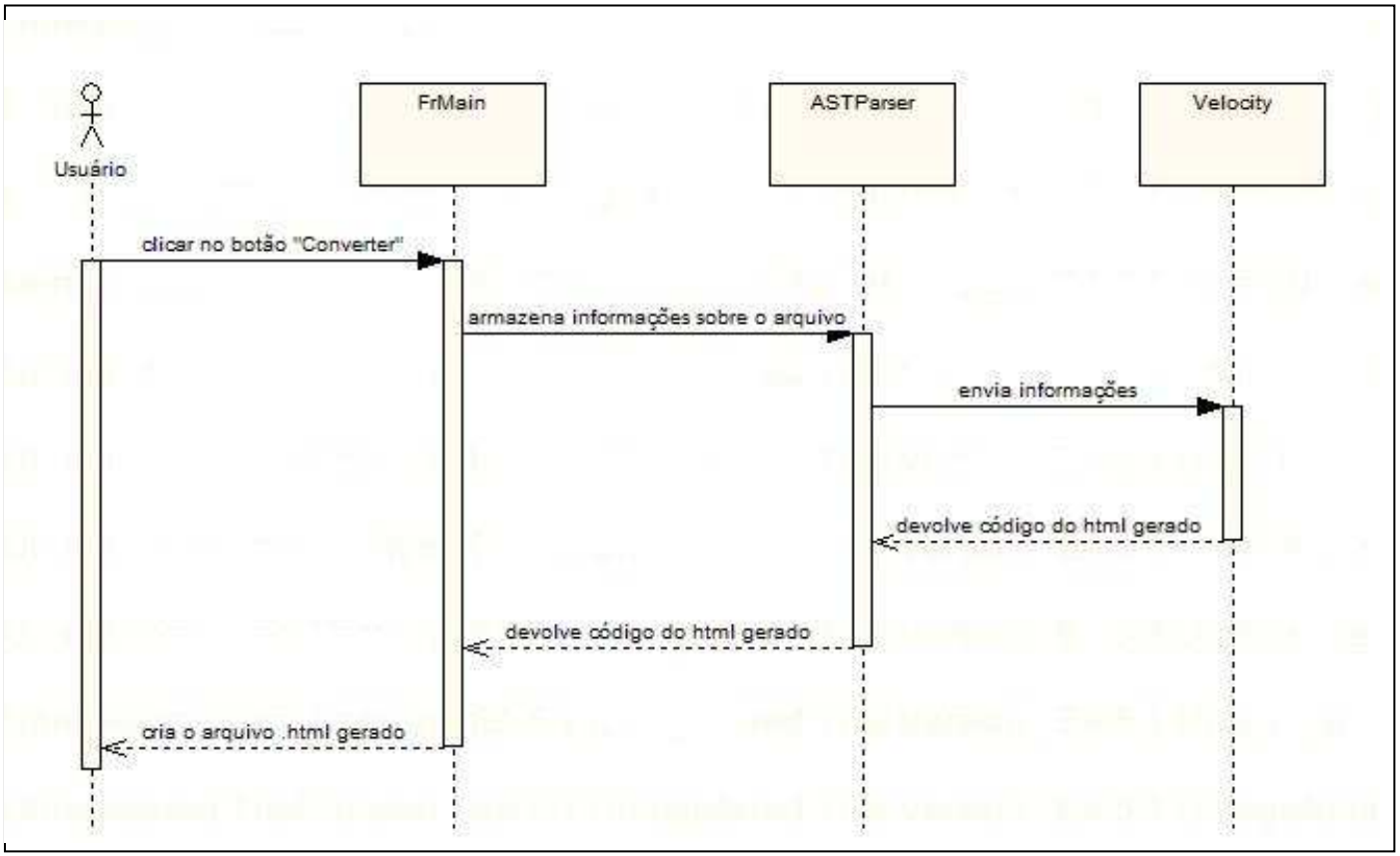
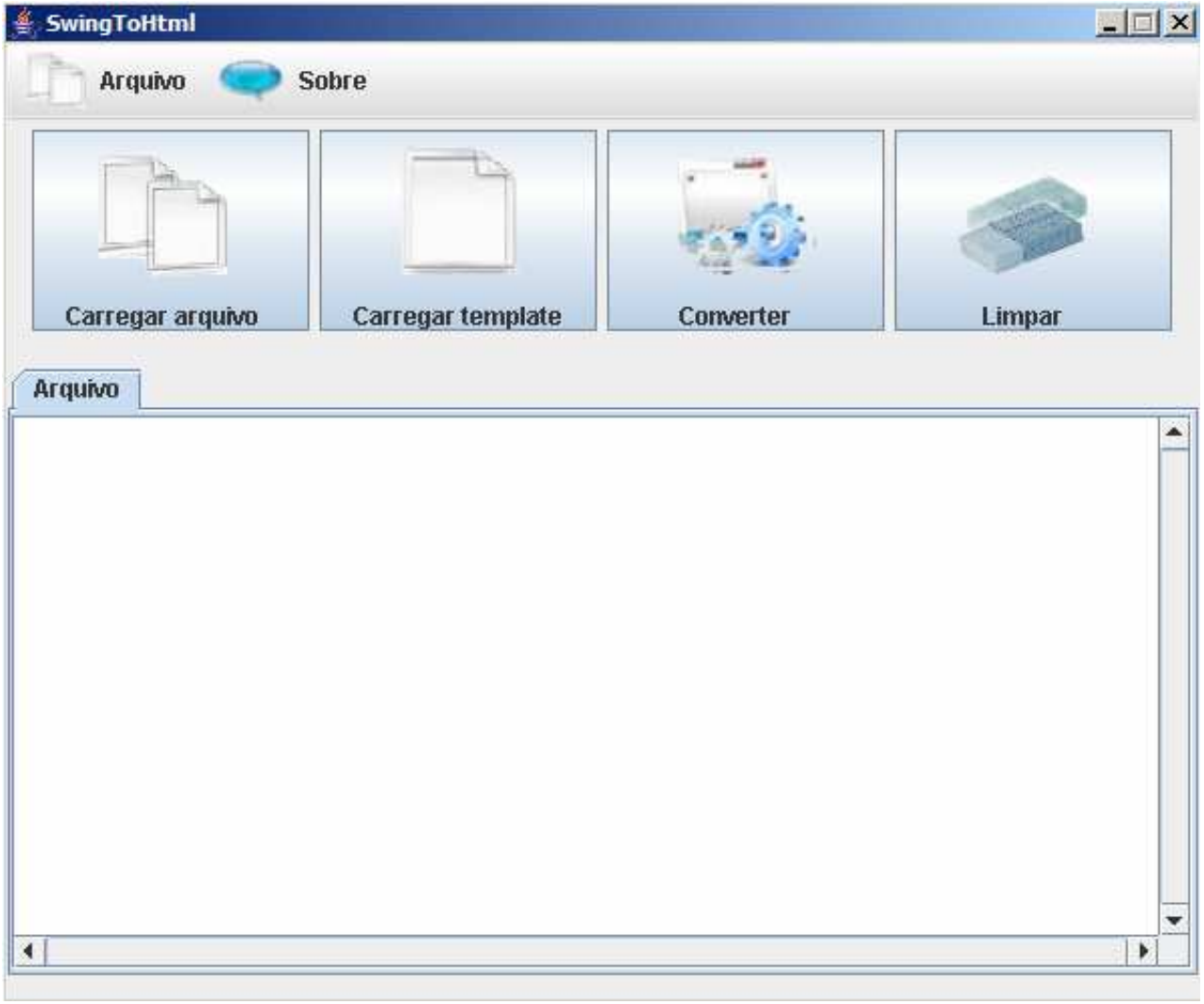


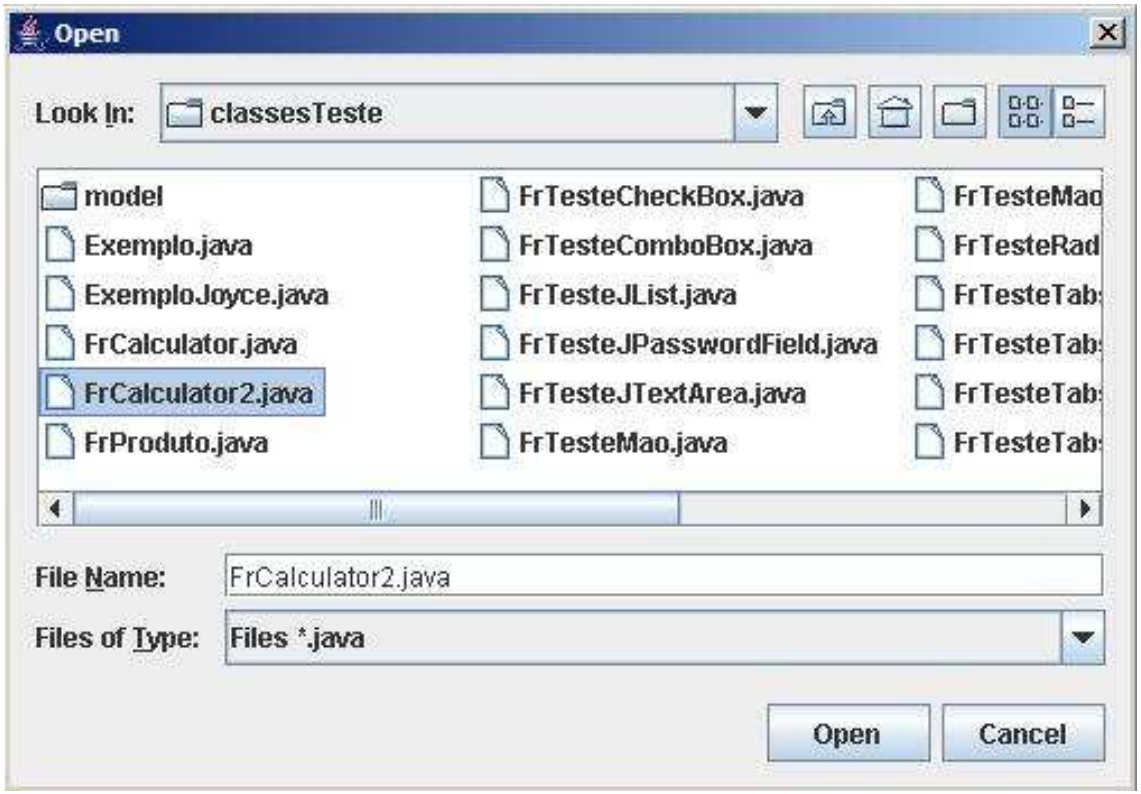
Diagrama de sequência



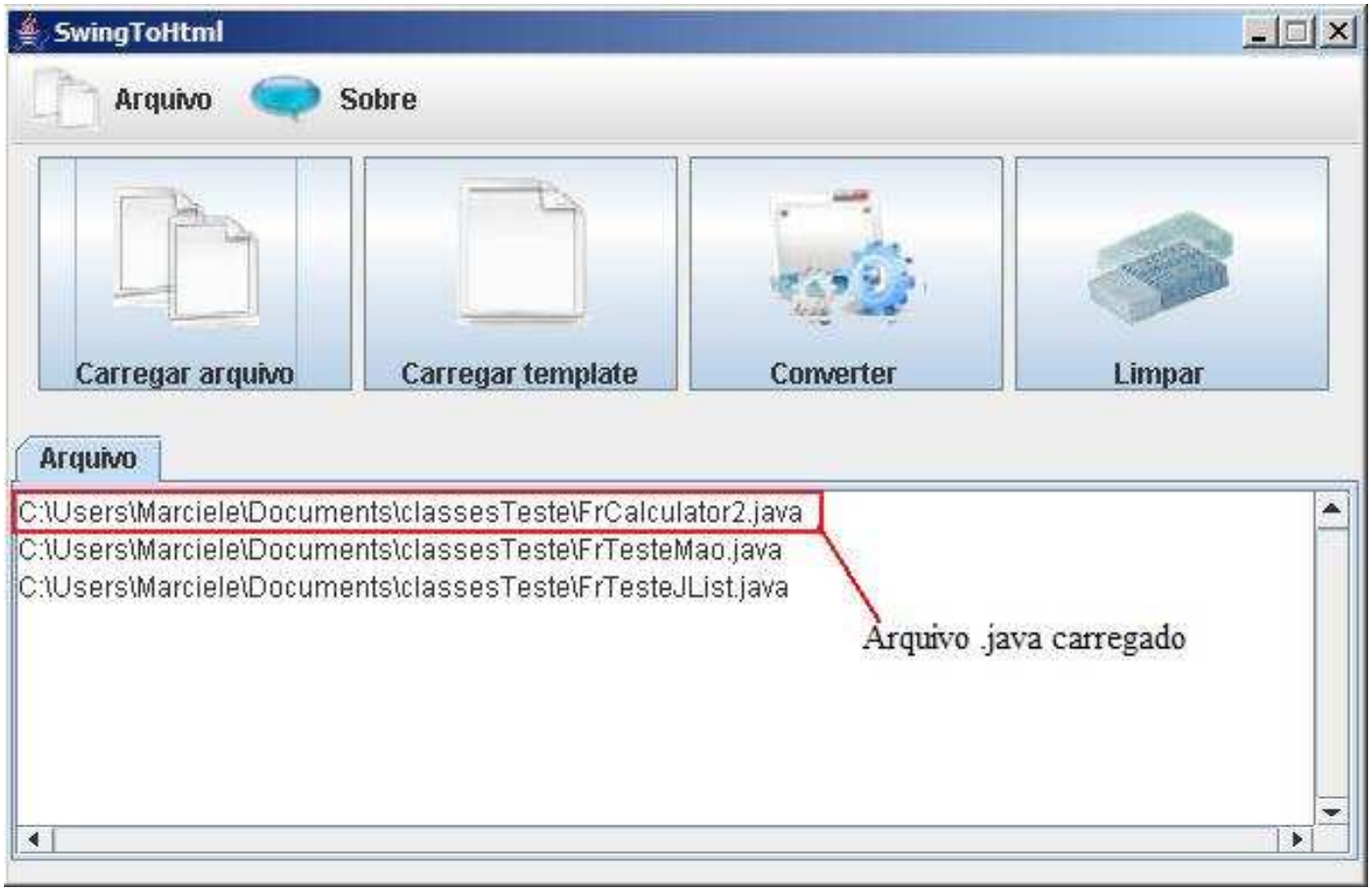
Utilizando a ferramenta



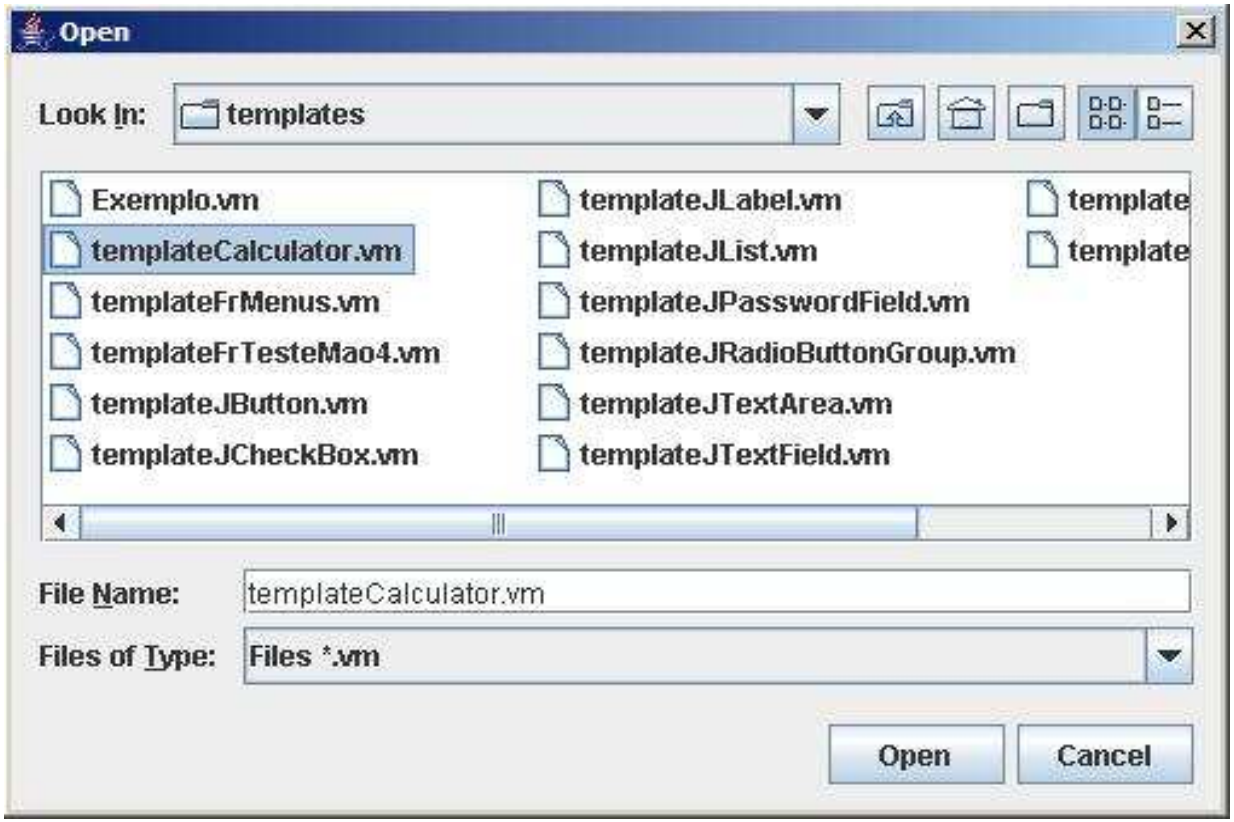
Utilizando a ferramenta



Utilizando a ferramenta

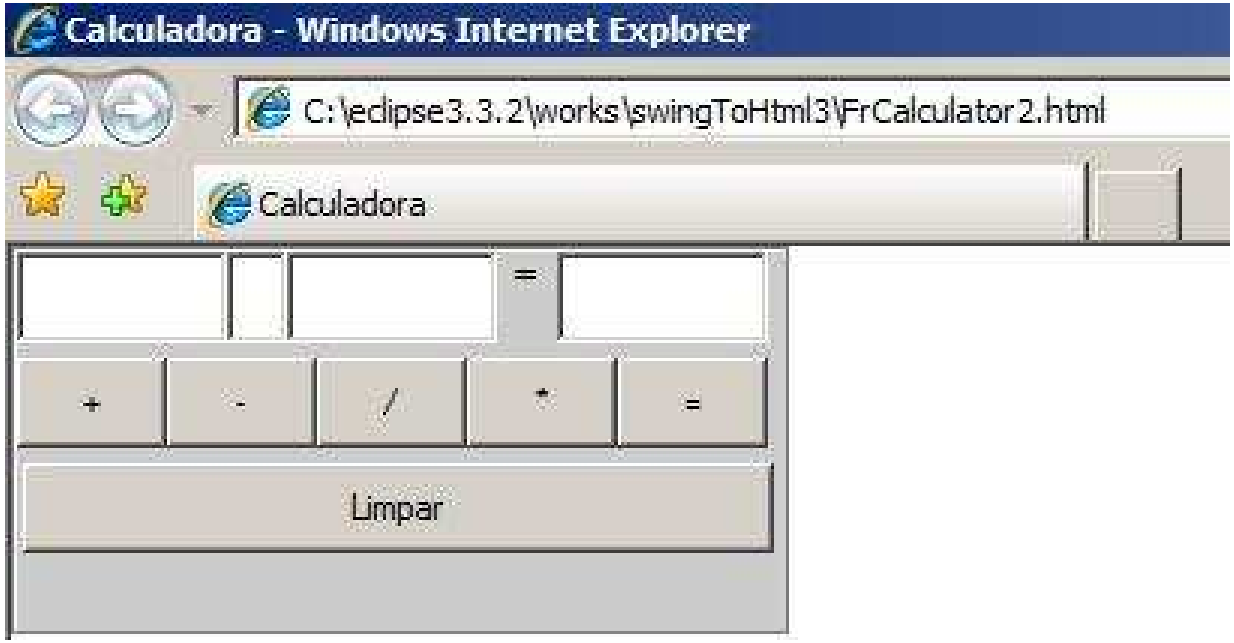


Utilizando a ferramenta



Utilizando a ferramenta

Utilizando a ferramenta



Resultados e discussão

CARACTERÍSTICAS	AjaxSwing	DelphiToWeb	Converte Forms	SwingToHtml
conversão de componentes de interface	sim	22	15	14
conversão de tratamento de eventos	sim	não	sim	sim
linguagem de entrada	arquivos `.java`	arquivos .DFM	arquivos Oracle Forms	arquivos `.java`
linguagem de saída	HTML, AJAX	HTML, .LZX, XML	`.java`	HMTL, JavaScript
uso de analisadores (léxico, sintático e semântico) para leitura dos arquivos de entrada	-	sim	sim	sim
uso de *templates* para geração de código	-	não	sim	sim

Conclusão

- Agilidade no desenvolvimento de aplicações
- Produtividade e qualidade no uso de *templates*
- A ferramenta atingiu seu maior objetivo – Conversão de componentes de interfaces
- Dificuldades com posicionamento dos componentes

Extensões

- implementar a conversão de mais componentes já que a ferramenta está limitada à conversão de 14 componentes;
- aumentar a implementação dos tratadores de eventos, a fim de que se possa ampliar a conversão do código dentro dos tratadores;
- criar templates para gerar código para tratamento de eventos (métodos);
- melhorar o posicionamento dos componentes de interface, a fim de chegar mais próximo do layout original.

Obrigado!

O sucesso geralmente vem para aqueles que estão muito ocupados para estarem procurando por ele.

Henry David Thoreau