WebCode
Componente web para edição de código fonte

Leonardo Zorzo Carbone

Orientador: Adilson Vahldick

Roteiro da Apresentação

- Introdução
- Fundamentação Teórica
- Desenvolvimento
- Conclusão
- Extensões

Introdução

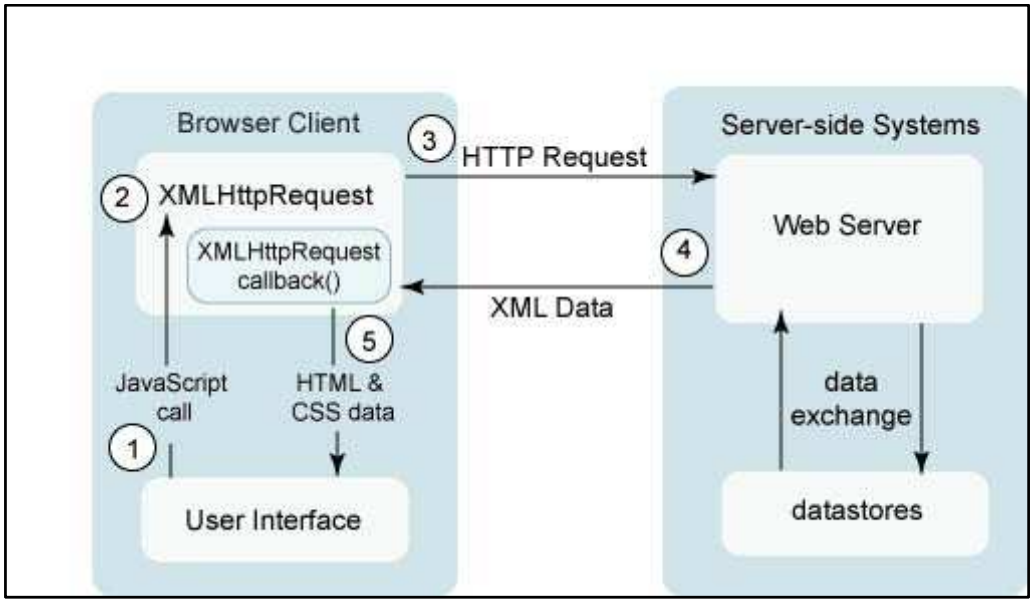
- Objetivos do Trabalhos
  - Disponibilizar um mecanismo para definir e anexar plugins
  - Disponibilizar meios para abrir e salvar o código digitado
  - Criar meios para inserir o componente em uma página HTML
  - Disponibilizar recursos de *syntax highlighting*, auto complemento de código e numeração de linhas

Fundamentação Teórica

- AJAX
- DOM
- JSON
- JSF
- Trabalhos Correlatos

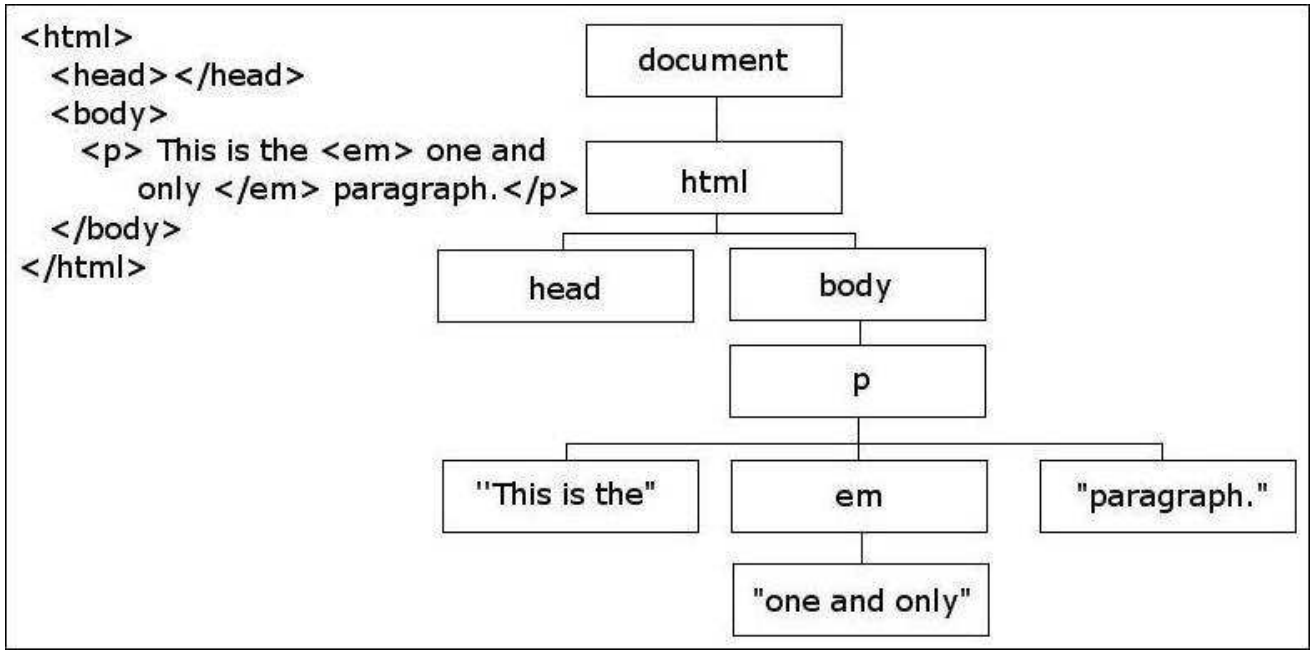
Fundamentação Teórica

- AJAX
  - CSS
  - DOM
  - XML
  - JavaScript
  - XMLHttpRequest



Fundamentação Teórica

- DOM
  - Estrutura de árvore
  - Variável document (JavaScript)
  - 5 divisões definidas pela W3C



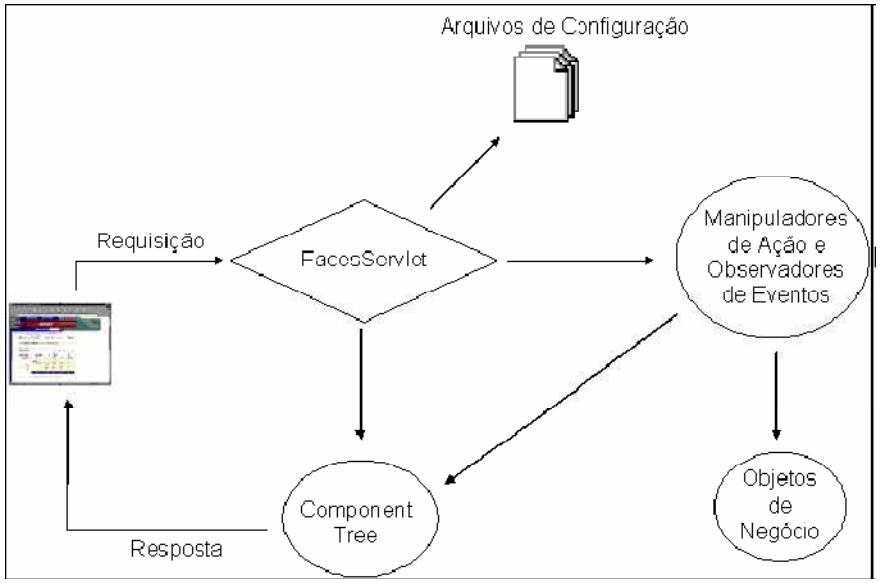
Fundamentação Teórica

- JSON
  - Formato para intercâmbio de dados
  - Representação de objetos complexos
  - Requisições AJAX
  - FlexJSON

```
{
 var stringJSON = ‘({“Pessoa”:{‘nome’:’Leonardo’,’idade’:’26’}})’;
 var novoObjeto = eval(stringJSON);
 alert(novoObjeto.nome);
}
```

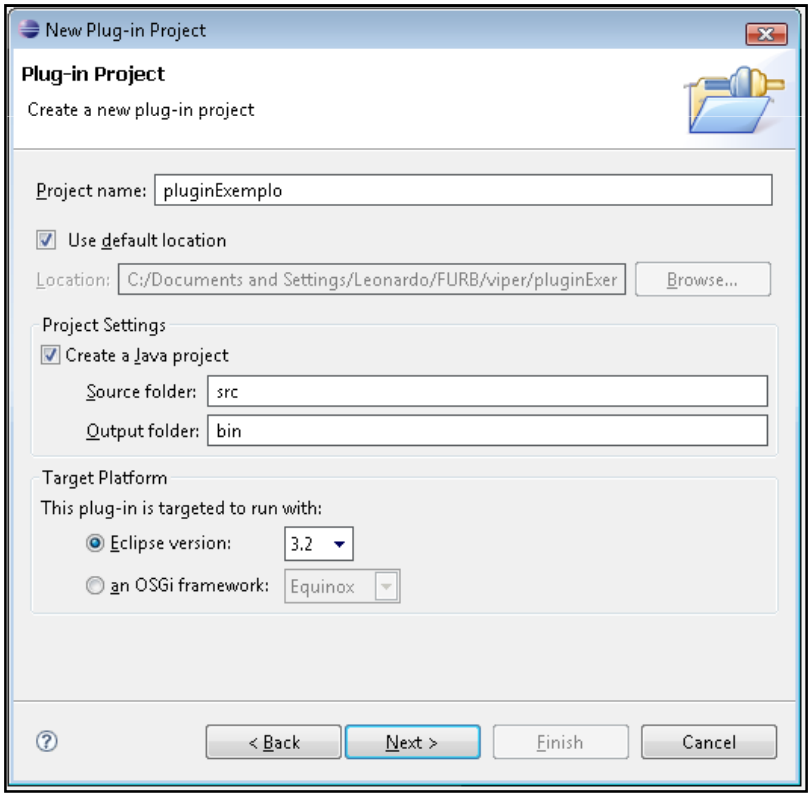
Fundamentação Teórica

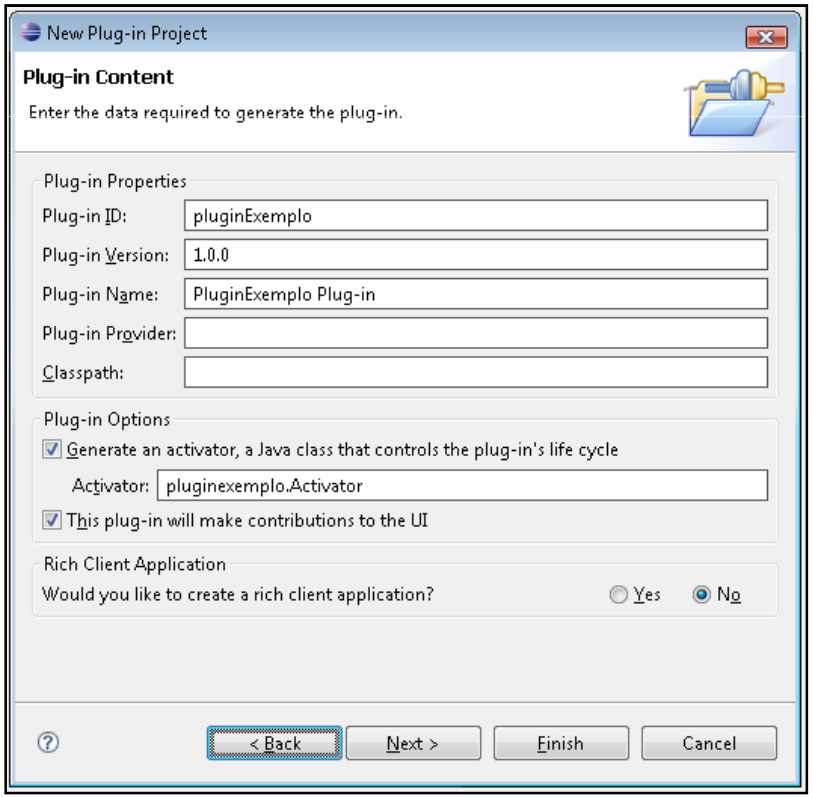
- JSF
  - Conjuntos de componentes reutilizáveis
  - Arquitetura MVC
  - Modelo de navegação declarativo
  - Ciclo de vida de requisições bem definido
  - Componentes customizados



Fundamentação Teórica
Trabalhos Correlatos

- Eclipse
  - Ambiente de desenvolvimento personalizado
  - Pontos de extensão para criação de *plugins*





Fundamentação Teórica
Trabalhos Correlatos

- CodePress
  - Bibliotecas JavaScript e arquivos CSS

```
<head>
   <link type="text/css" href="languages/codepress-php.css"
rel="stylesheet" id="cp-lang-style" />
<script type="text/javascript" src="codepress.js"></script>
<script type="text/javascript">
   CodePress.language = 'PHP';
</script>
</head>
```

```
<?php
// Very simple implementation of server side script

if(isset($_GET['file'])) {
        $file = basename($_GET['file']);
        $full_file = $path['server'].'/'.$path['webdocs'].'/'.$path['files']."/".$
        if(file_exists($full_file)) {
                $code = file_get_contents($full_file);
                $code = preg_replace("/>/","&gt;",$code);
                $code = preg_replace("/</","&lt;",$code);
                $language = getLanguage($file);
        }
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xht
<html>
<head>
        <title>CodePress - Real Time Syntax Highlighting Editor written in JavaScr
        <link type="text/css" href="languages/codepress-<?=$language?>.css" rel="s
```

Desenvolvimento

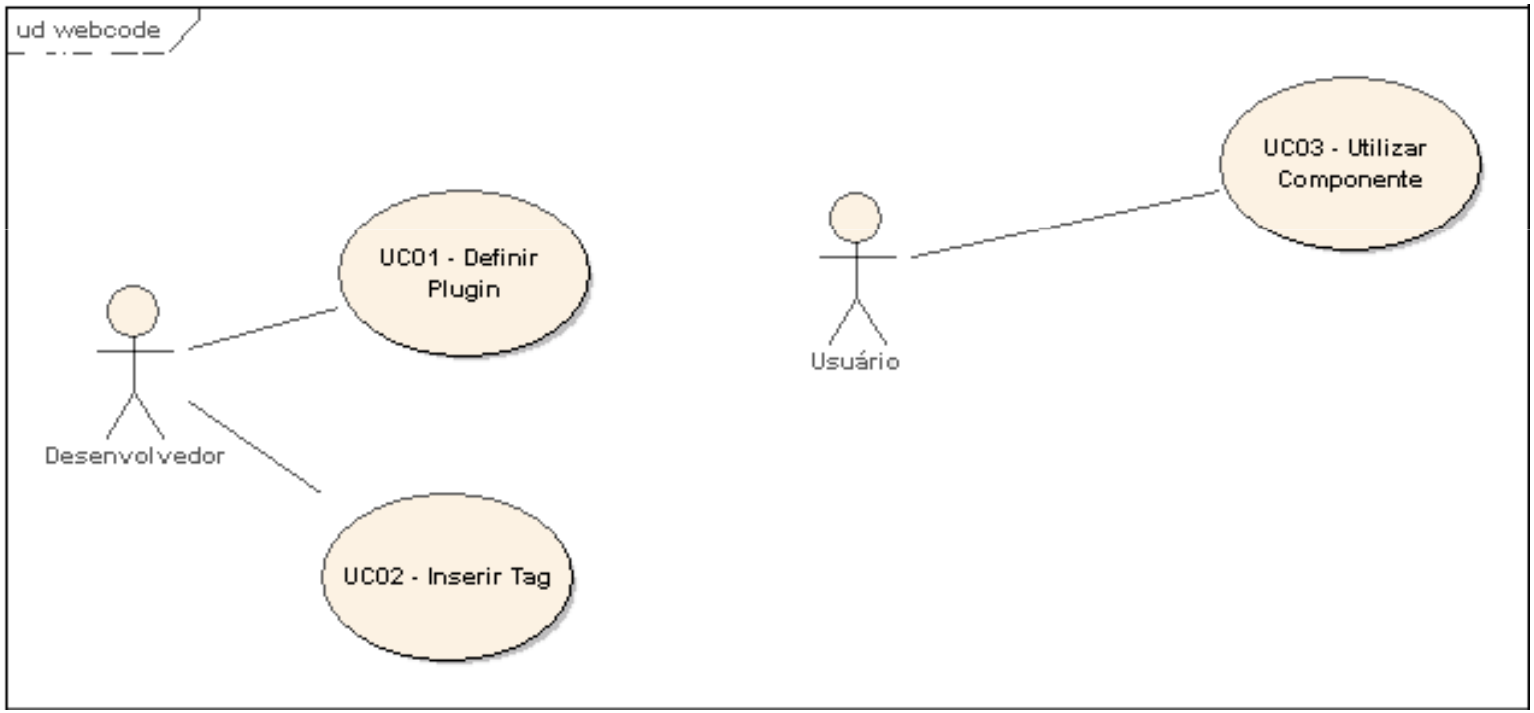
- Requisitos
- Casos de Uso
- Diagramas de Classes e Sequência
- Implementação
- Operacionalidade
- Resultados

Desenvolvimento

- Requisitos
  - Disponibilizar meios para abrir e salvar o código digitado (RF);
  - Disponibilizar mecanismo para definir e anexar plugins (RF);
  - Disponibilizar recursos de numeração de linhas, auto complemento e *syntax highlighting* (RF);
  - Reconhecer nativamente a linguagem Java (RF);
  - Ser implementado utilizando as linguagens Java e JavaScript e o *framework* JSF (RNF);
  - Ser compatível com os navegadores Internet Explorer e Firefox (RNF).

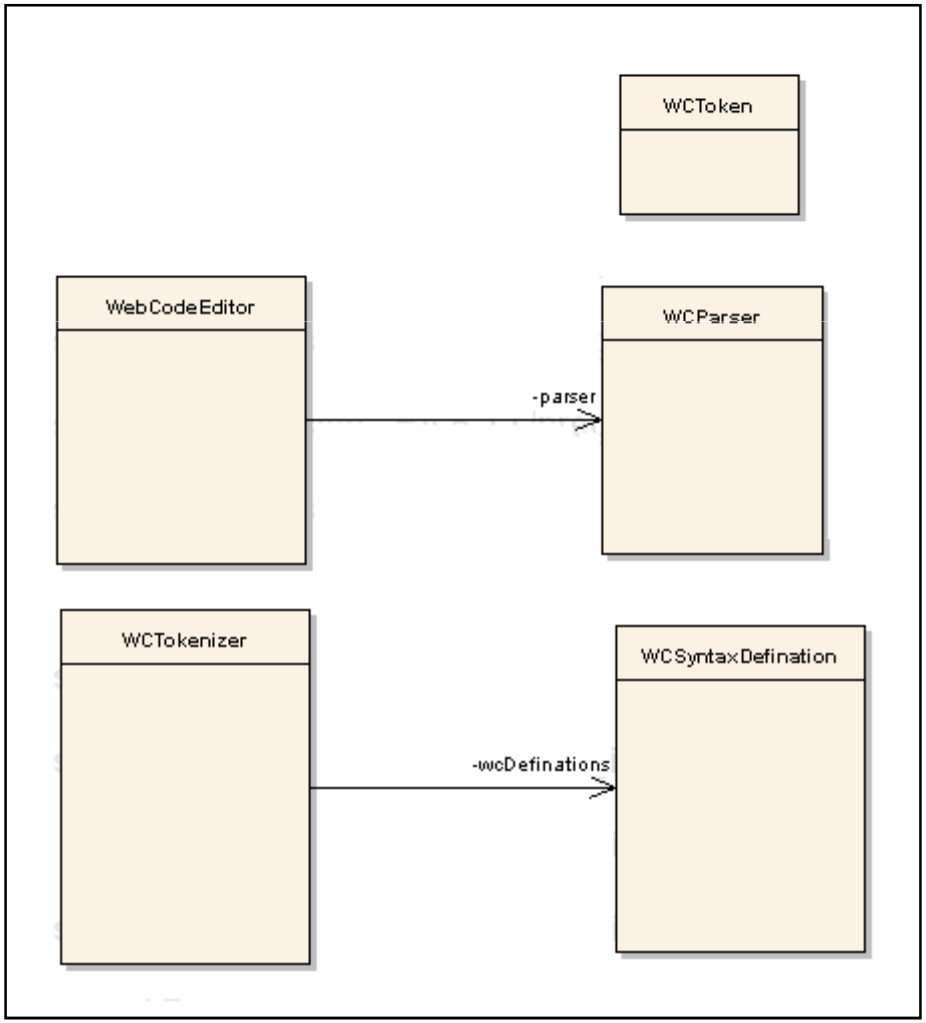
Desenvolvimento

- Casos de Uso



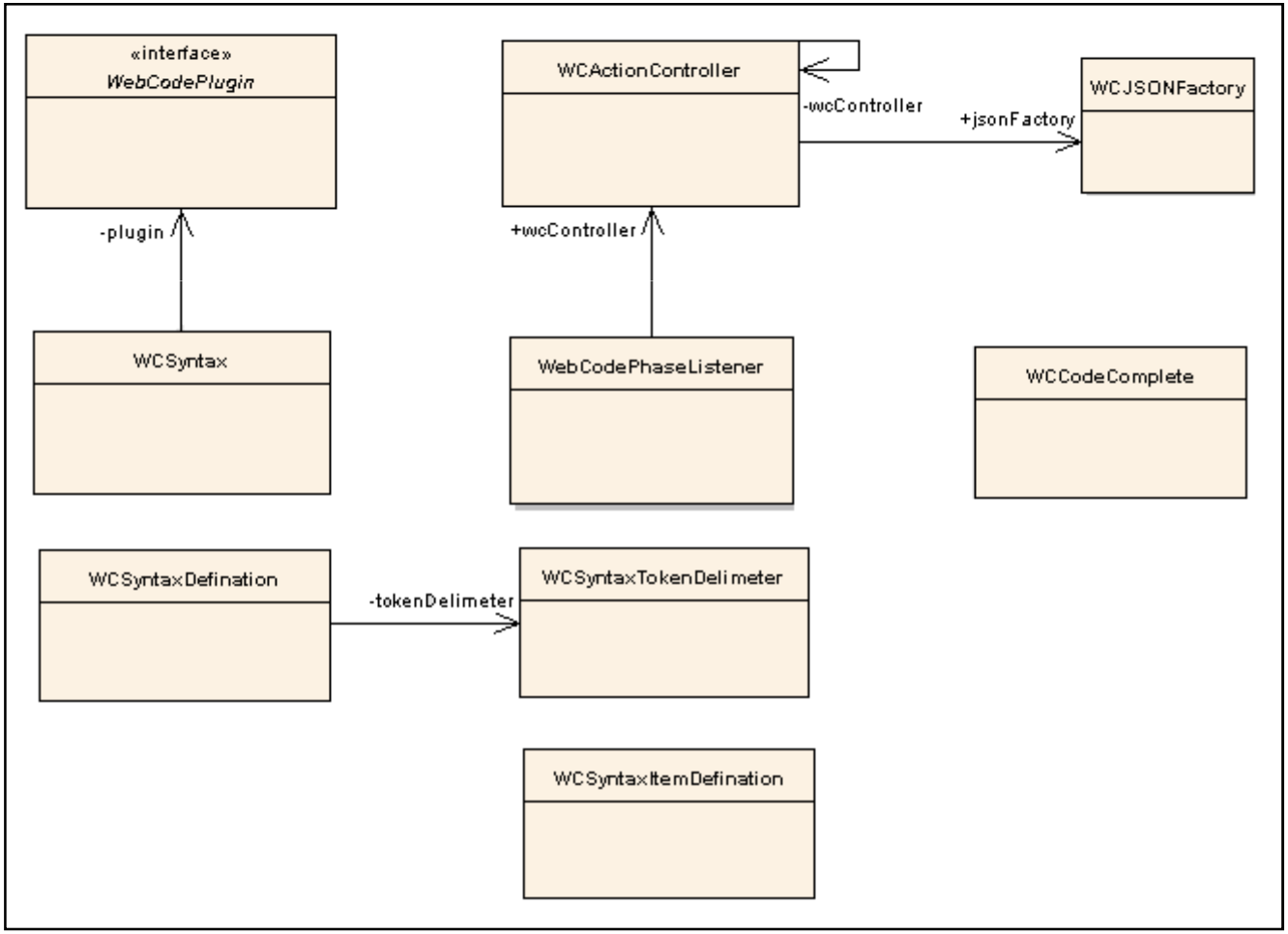
Desenvolvimento

- Diagramas de Classes: Módulo JavaScript

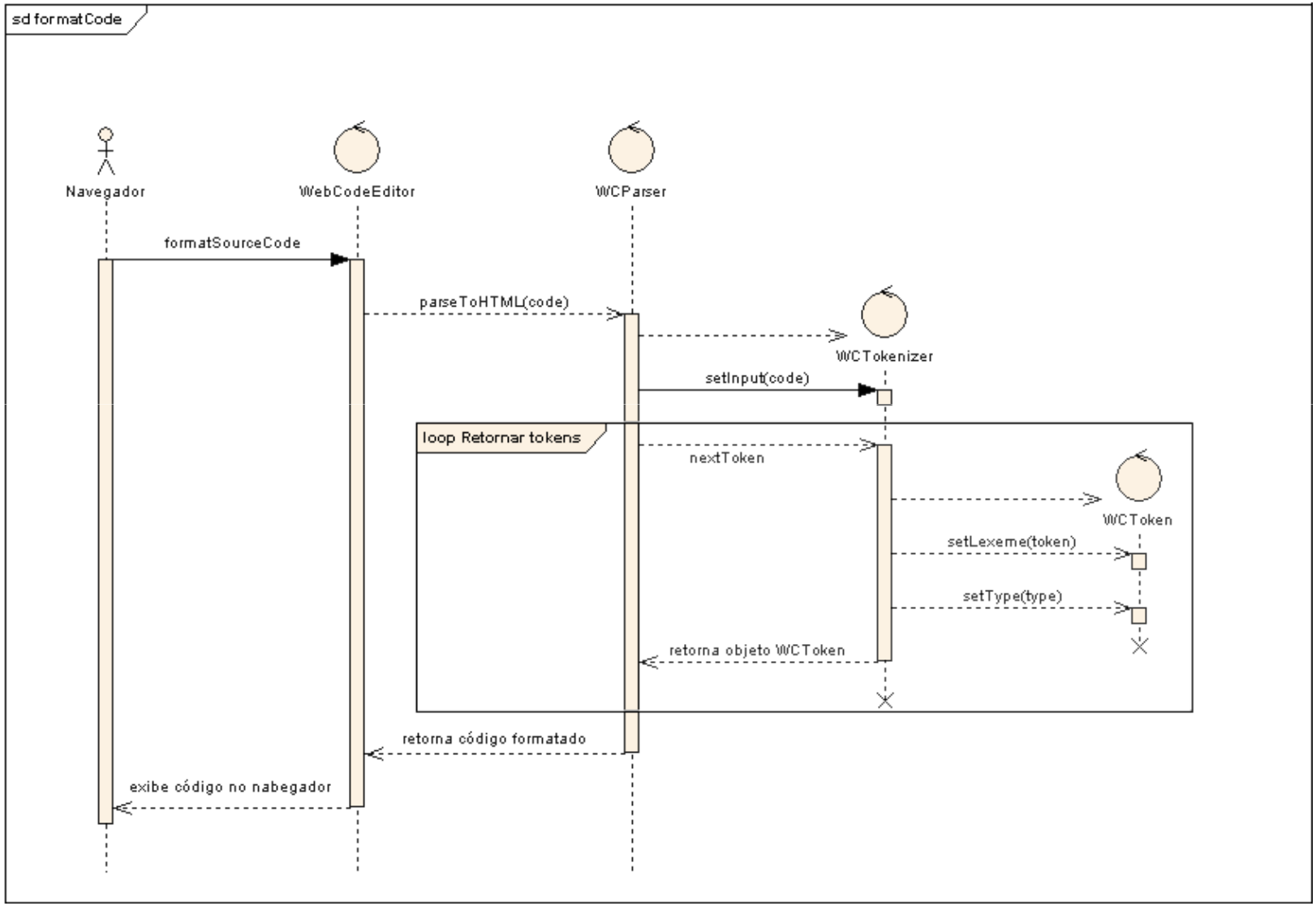


Desenvolvimento

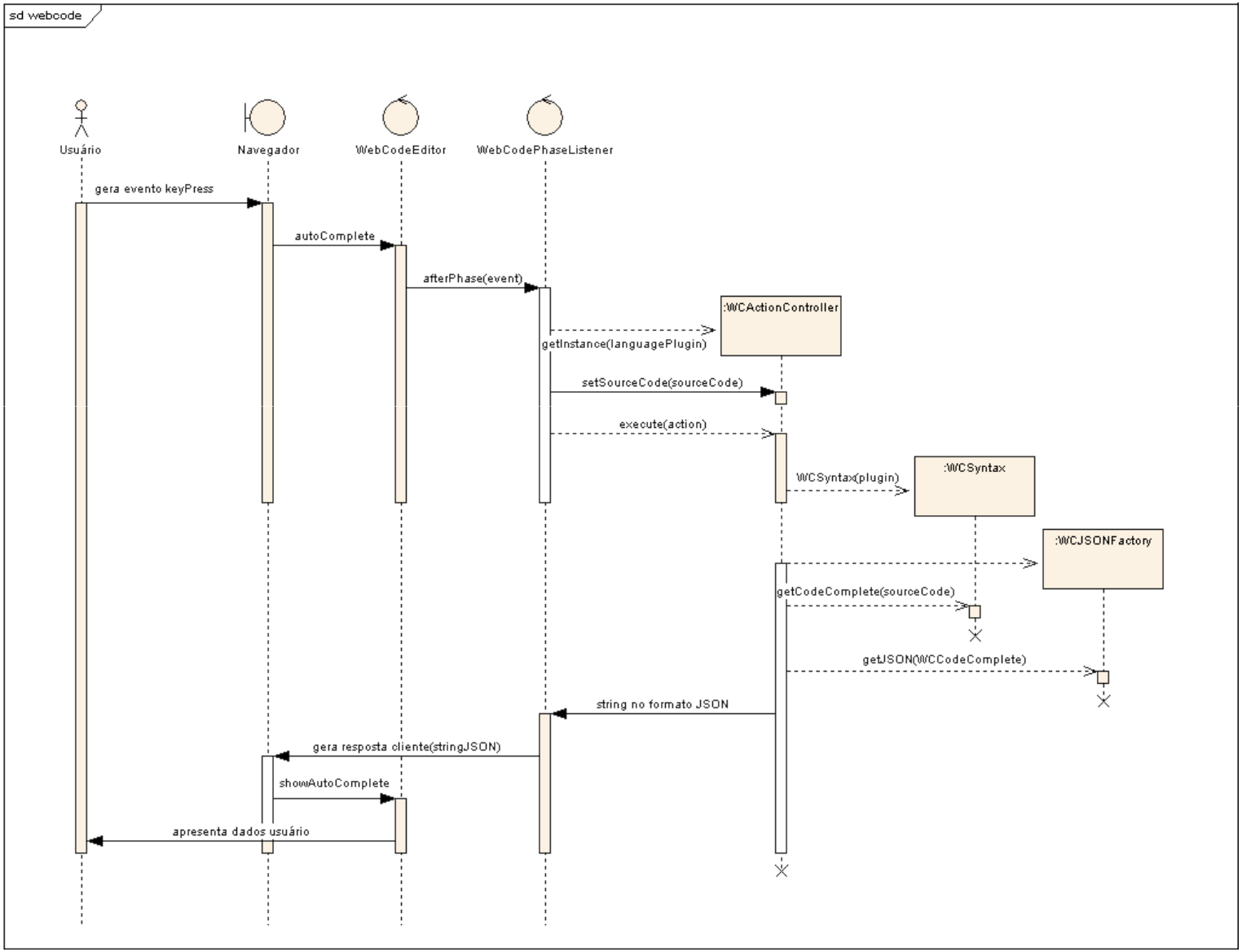
- Diagramas de Classes: Módulo Execução



Desenvolvimento



Desenvolvimento



Desenvolvimento

- Implementação
  - Módulo JSF
  - Módulo JavaScript
  - Módulo de Execução

Desenvolvimento

- Implementação: Módulo JSF (WebCodeRenderer)

Método	Parâmetros	Descrição
`beginRenderMainForm()`	`FacesContext`	Responsável por inserir o código JavaScript que instância a classe `WebCodeEditor`.
`endRenderMainForm()`	`FacesContext`	Finaliza a estrutura do componente.
`renderStatusBar()`	`FacesContext`	Método que cria a *status bar* do componente.
`renderWebCodeEditor()`	`FacesContext`, `UIComponent`	Método responsável por definir o `iframe` que corresponde a área de edição e por definir as funções JavaScript que serão executadas quando o componente for carregado no navegador.
`renderWebCodeToolBar()`	`FacesContext`, `WebCode`	Método responsável por criar a barra de ferramentas do componente e atribuir as funções JavaScript do objeto `WebCodeEditor` para cada botão da barra.

Desenvolvimento

- Implementação: Módulo JSF (WebCodeRenderer)

```
protected void beginRenderMainForm(FacesContext facesContext) throws IOException
{
    this.attributes.clear();
    this.attributes.addAttribute(HtmlConstants.TYPE,"text/javascript");
    this.tags.openScriptTag(this.attributes);
    this.insertText(facesContext,"var wc = new WebCodeEditor();");
    this.tags.closeScriptTag();

    this.attributes.clear();
    this.attributes.addAttribute(HtmlConstants.ID, WebCodeIds.MainForm + this.componentId);
    this.attributes.addAttribute(HtmlConstants.ALIGN, this.align);
    this.tags.openDivTag(this.attributes);
}
```

```
protected void renderWebCodeEditor(FacesContext facesContext,
                                   UIComponent component) throws IOException
{
    String onLoad = "";
    onLoad = "wc.setContentWindow(this.contentWindow);" +
             "wc.setDesignModeOn();" +
             "wc.getContentWindow()." + this.getJS_AttachEvent(this.getJS_KeyDownEvent(),
             "wc.keyDownEvent") + ";" +
             "wc.init('" + WebCodeConstants.AJAX_URI + "');";

    this.attributes.clear();
    this.attributes.addAttribute(HtmlConstants.CLASS,"codeArea");
    this.attributes.addAttribute(HtmlConstants.WIDTH, this.width);
    this.attributes.addAttribute(HtmlConstants.HEIGHT, this.height);
    this.attributes.addAttribute(HtmlConstants.ID,WebCodeIds.CodeArea);
    this.attributes.addAttribute(HtmlConstants.SRC,this.getResourceURL(facesContext,
                                 WebCodeConstants.RESOURCE_PATH + "/codeArea.html"));
    this.attributes.addAttribute(HtmlConstants.ON_LOAD_IE,onLoad);



    this.tags.openIFrameTag(this.attributes);
    this.tags.closeIFrameTag();
}
```

Desenvolvimento

- Implementação: Módulo JSF (WebCodePhaseListener)

```
public class WebCodePhaseListener implements PhaseListener
{
   private PhaseId phaseId = PhaseId.RENDER_RESPONSE;
   private WCActionController wcController = null;

   private void controlAjaxRequest(PhaseEvent event)
   {
       this.phaseId = PhaseId.RESTORE_VIEW;
       String requestType = event.getFacesContext().getViewRoot().getViewId();
       // Se não for uma requisição ajax, disparada pelo objeto javascript WebCodeEditor, sai do método.
       if (!requestType.trim().equals(WebCodeConstants.AJAX_URI.trim()))
          return;
       // Recupera Request.
       HttpServletRequest request = (HttpServletRequest)event.getFacesContext().getExternalContext().getRequest();
       // Recupera Response.
       HttpServletResponse response = (HttpServletResponse)event.getFacesContext().getExternalContext().getResponse();
       // Se não vier o parâmetro action, sai do método e não faz nenhum processo.
       String actionParameter = request.getParameter("action");
       if (actionParameter == null)
           return;
       WebCode component = (WebCode)request.getSession().getAttribute("webcode");
       if (this.wcController == null)
         this.wcController = WCActionController.getInstance(component.getLanguagePlugin());

       String sourceCode = "";
       if (request.getParameter("code") != null)
           sourceCode = request.getParameter("code");

       this.wcController.setSourceCode(sourceCode);
       try
       {
          wcActions[] wcAction = wcActions.values();
          String resultAction =  this.wcController.execute(wcAction[Integer.parseInt(actionParameter)]);

          response.setContentType("text/xml");
          response.setHeader("Cache-Control", "no-cache");
          response.setHeader("Pragma", "no-cache");
          response.setDateHeader("Expires",-1);
          response.setCharacterEncoding("iso-8859-1");
          response.getWriter().write(resultAction);
          event.getFacesContext().responseComplete();
       }
       catch (WCException ex)
```

Desenvolvimento

- Implementação: Módulo JavaScript (WebCodeEditor.js)

```
this.init = function(ajaxURI)
{

 ajaxUri = "faces"  + ajaxURI;
 if (!initialization)
 {

    try
    {
       framework.ajaxRequest("GET",false,getPluginInfoCallback,null,null,
                             onErrorCallback,ajaxUri+"?action=" +
                             wcActions.wcaGetPluginInfo,"");
       framework.ajaxRequest("GET",false,loadSyntaxDefinationCallback,null,null,
                             onErrorCallback,ajaxUri+"?action=" +
                             wcActions.wcaGetSyntaxDefination,"");
       initialization = true;
       contentWindow.focus();
       if (framework.isIE())
        {
          contentWindow.setInterval(formatSourceCode, 3000);
        }
       else
          {
             if (framework.isGecko())
                window.setInterval(formatSourceCode,3000);
          }
    }
    catch (exception)
    {
       onErrorCallback(exception.message);
    }
 }
};
```

Desenvolvimento

- Implementação: Módulo JavaScript (WebCodeEditor.js)

```
clearCodeTags = function(sourceCode)
{
   sourceCode = sourceCode.replace(/<br>/gi,'\n');
   sourceCode = sourceCode.replace(/<\/p>/g,'\n');
   sourceCode = sourceCode.replace(/<p>/gi,'\n');
   sourceCode = sourceCode.replace(/<.*?>/g,'');
   sourceCode = sourceCode.replace(/&lt;/g,'<');
   sourceCode = sourceCode.replace(/&gt;/g,'>');
   sourceCode = sourceCode.replace(/ /g,' ')
   sourceCode = sourceCode.replace(/ /g,' ');

   return sourceCode;
}

formatSourceCode = function()
{
   try
   {
   var code = contentWindow.document.body.innerHTML;
   if (code.trim().equals(""))
      return;

   contentWindow.document.selection.createRange().text = "\u2008";
   code = contentWindow.document.body.innerHTML;
   code = parser.parseToHTML(clearCodeTags(code))

   contentWindow.document.body.innerHTML = code;
   var range = contentWindow.document.body.createTextRange();
   if (range.findText("\u2008"))
      {
          range.select();
          range.text = '';
      }
   }
   catch (exception)
   {
       onErrorCallback(exception.message);
   }
};
```

Desenvolvimento

- Implementação: Módulo Execução (WCActionController)

```
public String execute(wcActions action)throws WCException
{
   if (action.equals(wcActions.wcaCodeComplete) ||action.equals(wcActions.wcaFormatCode))
      if (this.sourceCode.trim().equals(""))
         return this.jsonFactory.getJSON(new WCError("Error on execute action '" +
                                         action.name() + "': Source Code undefined."));

   try
   {
      String result = "";

      WebCodePlugin plugin = this.loadPlugin();
      WCSyntax wcSyntax = new WCSyntax(plugin);

      this.jsonFactory = new WCJSONFactory();
      switch (action)
      {
          case wcaCodeComplete        : result = this.jsonFactory.getJSON(
                                        wcSyntax.getCodeComplete(this.sourceCode));break;
          case wcaFormatCode          : result = this.jsonFactory.getJSON(
                                        wcSyntax.getFormatCode(this.sourceCode));break;
          case wcaGetSyntaxDefination : result = this.jsonFactory.getJSON(
                                        wcSyntax.getSyntaxDefination());break;
          case wcaGetPluginInfo       : result = this.jsonFactory.getJSON(plugin);break;
      }
      return result;
   }
   catch (WCException ex)
   {
      return this.jsonFactory.getJSON(new WCError("Error on execute action '" +
                                      action.name() + "'\n" + ex.toString()));
   }
}
```

Desenvolvimento

- Implementação: Módulo Execução (WCActionController)

```
private WebCodePlugin loadPlugin() throws WCException
{
   try
   {
      Class<?> loadClass = Class.forName(this.languagePlugin);
      Constructor loadConstructor = loadClass.getConstructor();
      return (WebCodePlugin)loadConstructor.newInstance();
   }
   catch (Exception ex)
   {
      throw new WCException("WCActionController loadPlugin error: Plugin: '" +
      this.languagePlugin + "'\n" + ex.toString());
   }
}
```

Desenvolvimento

- Operacionalidade
  - Implementação do *plugin*
  - Declaração de bibliotecas
    - flexjson.jar
    - webcode.jar
  - Declaração de *taglibs*
  - Declaração *tag* webcode na página

Desenvolvimento

- Operacionalidade
  - Implementação do *plugin*

```
package br.furb.inf.tcc.imp.webcode.plugin;

import br.furb.inf.tcc.imp.webcode.WCException;
import br.furb.inf.tcc.imp.webcode.syntax.WCSyntaxDefination;
import br.furb.inf.tcc.imp.webcode.syntax.code.WCCodeComplete;

public interface WebCodePlugin
{
   public abstract WCSyntaxDefination getWebCodeSyntaxDefinations() throws WCException;
   public abstract WCCodeComplete getWebCodeComplete(String sourceCode) throws WCException;
   public abstract String getPluginVersion() throws WCException;
   public abstract String getPluginName() throws WCException;
}
```

Desenvolvimento

- Operacionalidade
  - Implementação do *plugin*

```
private void setReservedWords()
{
    this.wcSyntaxItem = new WCSyntaxItemDefination();
    this.wcSyntaxItem.setColor("Blue");
    this.wcSyntaxItem.setBold(true);
    this.wcSyntaxItem.setItalic(true);

    this.wcSyntaxItem.addItem("abstract");
    this.wcSyntaxItem.addItem("boolean");
    this.wcSyntaxItem.addItem("break");
    this.wcSyntaxItem.addItem("byte");
    this.wcSyntaxItem.addItem("case");
    this.wcSyntaxItem.addItem("catch");
    this.wcSyntaxItem.addItem("char");
    this.wcSyntaxItem.addItem("class");
    this.wcSyntaxItem.addItem("const");
    this.wcSyntaxItem.addItem("continue");
    this.wcSyntaxItem.addItem("default");
    this.wcSyntaxItem.addItem("do");
    this.wcSyntaxItem.addItem("double");
    this.wcSyntaxItem.addItem("else");
    this.wcSyntaxItem.addItem("extends");
    this.wcSyntaxItem.addItem("final");
    this.wcSyntaxItem.addItem("finally");
    this.wcSyntaxItem.addItem("float");
    this.wcSyntaxItem.addItem("for");
    this.wcSyntaxItem.addItem("if");
    this.wcSyntaxItem.addItem("implements");
    this.wcSyntaxItem.addItem("import");
```

Desenvolvimento

- Operacionalidade
  - Implementação do *plugin*

```
private void setPredefinedClasses()
{
    this.wcSyntaxItem = new WCSyntaxItemDefination();
    this.wcSyntaxItem.setColor("Red");
    this.wcSyntaxItem.setItalic(true);

    this.wcSyntaxItem.addItem("String");
    this.wcSyntaxItem.addItem("Integer");
    this.wcSyntaxItem.addItem("Double");
    this.wcSyntaxItem.addItem("File");
    this.wcSyntaxItem.addItem("PrintStream");
    this.wcSyntaxItem.addItem("PrintWriter");

    this.wcDefination.addSyntaxItem(this.wcSyntaxItem);
}
```

Desenvolvimento

- Operacionalidade
  - Implementação do *plugin*

```
public String getPluginVersion()
{
   return "1.0.0";
}

public String getPluginName()
{
    return "JavaPlugin";
}

 public WCSyntaxDefination getWebCodeSyntaxDefinations()
 {
     this.wcDefination.setLiteralColor("Orange");
     this.wcDefination.setCommentColor("Green");
     this.wcDefination.setNumberColor("Gray");

     return this.wcDefination;
 }
```

Desenvolvimento

- Operacionalidade
  - Declaração de *taglibs* e *tag* webcode

```
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<%@taglib prefix="t" uri="http://tcc.imp.webcode"%>
```

```
<f:view>
   <t:webCode componentId="webCode" align="center"
              languagePlugin="br.furb.inf.tcc.imp.webcode.plugin.JavaPlugin"
              onOpen="abrirDados" onSave="salvarDados">

   </t:webCode>
</f:view>
```

Desenvolvimento

- Resultados
  - Comparação com trabalhos correlatos

	Eclipse	**CodePress**	**WebCode**
Ambiente	Desktop	Web	Web
Recurso de plugins	Possui	Não Possui	Possui
Criação de plugins	Complexa	Não Possui	Simples
Abrir e salvar código	Possui	Não Possui	Possui
Auto complemento de código	Possui	Não possui	Possui
Syntax highlighting	Possui	Possui	Possui
Reconhecimento das linguagens	Por meio de plugins	CSS, Classes Java Script	Por meio de plugins

Conclusão

- Objetivos alcançados
- Utilização do componente por meio de *tag* customizada
- Reutilização do componente
- Problemas com estudo de caso (WebIde)

Extensões

- Recursos de análise sintática e semântica
- Configuração do editor
- Novos atributos para a *tag* webcode
- Recursos para ocultar blocos de código
- Aperfeiçoar recursos de auto complemento