Nesta seção você encontra artigos voltados para diferentes abordagens de apoio ao desenvolvimento de projetos de software

Computação em Nuvem

Uma aplicação prática usando o Google App Engine

Gabriella Castro Barbosa Costa
gabriellacbc@gmail.com
Bacharel em Sistemas de Informação pelo Centro de Ensino Superior de Juiz de Fora e Técnica em Informática Industrial pelo CEFET-MG. Atua como desenvolvedora Web desde 2008.

Marco Antônio Pereira Araújo
maraujo@devmedia.com.br
É Doutor e Mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ, Especialista em Métodos Estatísticos Computacionais e Bacharel em Informática pela UFJF, professor do curso de Bacharelado em Ciência da Computação da FAGOC, professor dos Cursos de Bacharelado em Sistemas de Informação do Centro de Ensino Superior de Juiz de Fora, da Faculdade Metodista Granbery e da Universidade Severino Sombra, professor do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Fundação Educacional D. André Arcoverde, Analista de Sistemas da Prefeitura de Juiz de Fora, Editor da Engenharia de Software Magazine.

Em departamentos de Tecnologia da Informação, há cerca de duas décadas atrás, já se agrupavam computadores de forma a simular um único computador com maior capacidade computacional. A clusterização, como é chamada, permitiu a comunicação entre computadores que, neste contexto, são também denominados nós, equilibrando a demanda computacional. Usando softwares de gerenciamento de clusters, os processos são executados na CPU (*Central Processing Unit* - Unidade Central de Processamento), com maior disponibilidade de processamento naquele instante.

Em 1990, surge uma nova tecnologia, conhecida por *Grid Computing*, ou Computação em Grade, que teve como premissa uma analogia com o fornecimento de energia elétrica. Se recursos como energia podem ser fornecidos por

De que se trata o artigo?
Este artigo apresenta o processo de desenvolvimento de uma aplicação Web que faz uso da plataforma de desenvolvimento App Engine do Google. O aplicativo a ser desenvolvido oferece os recursos de um dicionário da língua portuguesa e um tradutor Português/Inglês e Inglês/Português.

Para que serve?
Classificado no modelo de serviço PaaS (Platform as a Service – Plataforma como Serviço), o Google App Engine oferece serviços de desenvolvimento de aplicações em nuvem, permitindo aos desenvolvedores a criação dos próprios aplicativos web utilizando a infraestrutura das aplicações do Google.

Em que situação o tema é útil?
Plataformas ofertadas como serviço permitem que o usuário faça uso de toda uma plataforma de desenvolvimento de aplicações voltadas para a Computação em Nuvem – um novo paradigma da computação que vem ganhando força desde 2006, caracterizando-se principalmente por uma profunda modificação dos modelos tradicionais de utilização dos recursos de TI. As aplicações que fazem uso do Google App Engine são fáceis de criar, manter e escalar de acordo com a necessidade do usuário.

terceiros, isso também poderia ocorrer com os recursos de computação. A Computação em Grade expandiu os conceitos de clusters, pois possibilita um melhor aproveitamento dos recursos, utilizando-se de momentos ociosos dos recursos computacionais das máquinas interligadas, pertencentes a diferentes domínios e distribuídas geograficamente. Grades computacionais são muito utilizadas para fins científicos, como em grandes universidades.

A Computação em Nuvem, ou *Cloud Computing*, utiliza-se dos conceitos da Computação em Grade, oferecendo recursos de *data centers* a usuários que não querem preocupar-se em adquirir a infraestrutura necessária para montar uma grade computacional. Este novo paradigma vem ganhando força desde 2006, caracterizando-se principalmente por uma profunda modificação dos modelos tradicionais de utilização dos recursos de TI.

Muitas pessoas acreditam que a Computação em Nuvem é um novo paradigma que será responsável por grandes transformações no atual mundo de TI. Outras acreditam, simplesmente, que este paradigma é apenas uma evolução repaginada da já conhecida *Utility Computing* ou Computação Utilitária.

A Computação em Nuvem pode ser resumidamente definida como um modelo tecnológico que permite o acesso a um conjunto de recursos computacionais através de uma rede, por demanda, de forma rápida e sem a necessidade de grandes configurações.

Este paradigma tem como principal característica mudar o foco dos departamentos de TI para projetos estratégicos, ao invés de simplesmente manter recursos em funcionamento (como os *data centers*), enquanto reduz custos operacionais e de capital.

Segundo Taurion (2009), a ideia da Computação em Nuvem é bastante sedutora, pois propõe uma forma de utilizar recursos ociosos, diminuir custos para as organizações, possibilitar aos usuários domésticos acesso amplo aos seus dados e aplicativos através de qualquer dispositivo com acesso à Internet e, tudo isso, sem pagar por licenças de uso de software. Por outro lado, o uso deste modelo está altamente subordinado a uma Internet de boa velocidade, consistente e estável. A interoperabilidade entre diferentes nuvens pode ser vista como um problema, já que ainda não existem padrões claramente definidos. Deve-se levar em conta também a questão da segurança no tráfego de dados confidenciais através da nuvem.

O avanço das aplicações Web 2.0, a disseminação da Internet banda larga e a pressão por melhores infraestruturas computacionais nas organizações são fatores que influenciam e favorecem a utilização deste paradigma emergente.

Cada vez mais, as organizações estão voltando-se para a Computação em Nuvem como um meio de baixo custo e de rápido tempo de entrega de soluções ao mercado. Grandes empresas como Amazon, Google, Salesforce e IBM já fazem uso deste paradigma e vêm obtendo satisfatórios resultados.

O Google *App Engine* possibilita a execução de aplicativos Web na infraestrutura de nuvem própria do Google, aproveitando todo o poder de processamento da empresa. Seus recursos são escaláveis, de forma dinâmica, à medida que o tráfego e a necessidade de armazenamento aumentam, tornando simples e fácil a manutenção dos aplicativos que fazem uso desta plataforma.

Neste contexto, com intuito de apresentar como se dá o desenvolvimento deste tipo de aplicação, neste artigo será desenvolvida uma aplicação prática, usando o Google *App Engine*, plataforma de desenvolvimento em nuvem gratuita, para demonstrar uma das soluções de Computação em Nuvem.

O Google *App Engine*

O Google é considerado como uma das empresas pioneiras na utilização da Computação em Nuvem. O exemplo de maior destaque é o mecanismo de busca ofertado por esta empresa que é usado por milhares de pessoas. Como o mecanismo requer uma grande capacidade de processamento e necessita acessar milhares de megabytes de dados, a solução encontrada pela empresa foi agrupar centenas de milhares de servidores em *data centers* espalhados por todo o mundo.

Para aproveitar a imensa capacidade da nuvem computacional, o Google passou a disponibilizar outros serviços além da busca, como o *YouTube*, Google *Maps*, Google *Apps* e também o Google *App Engine*.

Classificado no modelo de serviço PaaS (*Platform as a Service* - Plataforma como Serviço), o Google *App Engine* oferece serviços de desenvolvimento de aplicações em nuvem, permitindo aos desenvolvedores a criação dos próprios aplicativos web utilizando a infraestrutura das aplicações Google.

Como os aplicativos alocados no Google *App Engine* encontram-se na nuvem do Google, não há necessidade de manter servidores. As aplicações para esta plataforma são fáceis de criar, manter e escalar de acordo com a necessidade do usuário. Além disso, o Google fornece um ambiente de desenvolvimento local que simula o Google *App Engine* em qualquer computador desktop.

Atualmente são suportadas as linguagens de programação Python e Java. O armazenamento de dados faz uso do *Big Table*, um sistema de armazenamento distribuído de gerenciamento de dados estruturados que se destina ao armazenamento em larga escala.

O ambiente de desenvolvimento Java, que será usado no decorrer deste trabalho, é compatível com o padrão *Servlet* Java e também possibilita o uso de JSPs (*Java Server Pages*).

Como principal vantagem desta plataforma de desenvolvimento em Nuvem, pode-se citar o fato de a utilização da infraestrutura do Google estar associada à credibilidade que a empresa ganhou no decorrer dos anos e também à garantia de disponibilidade ofertada.

O uso da plataforma é gratuito dentro das cotas estabelecidas pelo Google (disponíveis para acesso através do link 'Limites de uso dos recursos do Google *App Engine*') e o usuário consegue ter acesso aos recursos consumidos por sua aplicação através de um painel de monitoramento – *dashboard* – onde também é possível efetuar o controle de versões e visualizar relatórios dos usuários e tarefas ao utilizar o aplicativo.

Outra grande vantagem é a opção dada ao usuário de possuir um servidor local para testes das aplicações antes de enviá-la à nuvem, sendo que somente a API para envio de e-mails não funciona localmente (para desenvolvedores que utilizam como ferramenta de desenvolvimento o Eclipse, a utilização da infraestrutura do Google ainda pode ser simplificada através da instalação de um *plugin*).

Figura 1. CloudDictionary - Dicionário

Figura 2. CloudDictionary -Tradutor

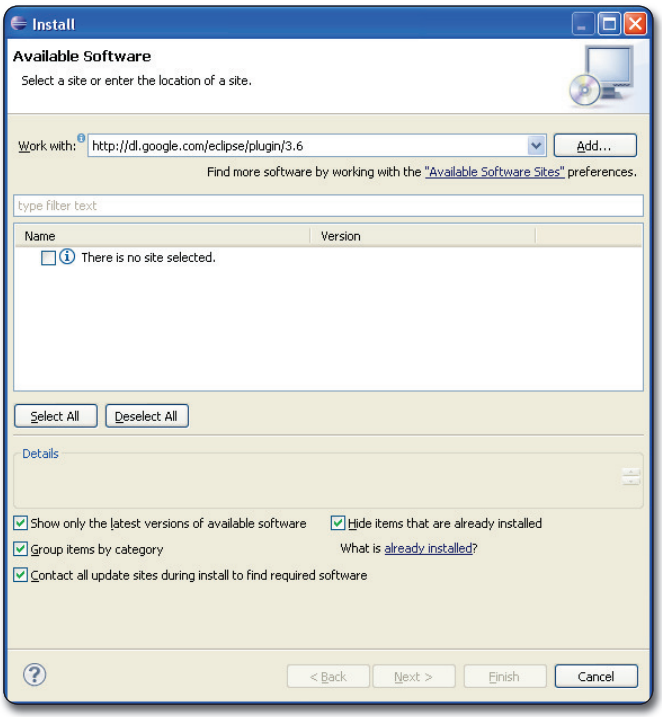

Figura 3. Instalação do plugin da plataforma *App Engine* para Eclipse: *Install New Software*

Como o uso gratuito do Google *App Engine* possui algumas limitações quanto ao número de chamadas a cada recurso oferecido por dia e por minuto, caso os limites disponíveis sejam excedidos, o usuário poderá efetuar um *upgrade* de conta, pagando pelos recursos extras a serem utilizados.

CloudDictionary: a aplicação a ser desenvolvida

Será desenvolvida uma aplicação Web em nuvem, em Java, que faz uso da plataforma de desenvolvimento *App Engine* do Google e oferece os recursos de um dicionário da língua portuguesa, conforme exibido na **Figura 1**, e também um tradutor de palavras desta língua para o inglês e vice versa, conforme a **Figura 2**. Tal aplicação teve como motivação verificar, de forma prática, a viabilidade, vantagens e desvantagens do Google *App Engine*.

A aplicação desenvolvida, chamada CloudDictionary, faz uso da integração de um serviço que encontra-se fora da nuvem – Dicionário Michaelis – por meio da chamada de uma URL externa.

Configuração do ambiente de desenvolvimento

Para o desenvolvimento desta aplicação foi utilizado o ambiente de desenvolvimento gratuito Eclipse SDK (*Software Development Kit* - Kit de Desenvolvimento de Software), versão 3.6.0, cujo download encontra-se disponível em http://www.eclipse.org/downloads/, pois existe um *plugin* para este ambiente que possibilita o desenvolvimento e o *upload* das aplicações para a nuvem do Google de forma mais fácil.

Após a instalação do Eclipse, deve-se proceder a instalação do *plugin* para o Eclipse da *App Engine*, de acordo com os seguintes passos:

Passo 1: No menu 'Help', clicar em 'Install New Software'.

Passo 2: Em 'Workwith', conforme **Figura 3**, colar a url: *http://dl.google.com/eclipse/plugin/3.6* e clicar em 'Add...'.

Passo 3: Selecione todos os checkboxes e em seguida no botão Next (**Figura 4**) para a instalação do plugin, do Google App Engine Java SDK e do Google Web Toolkit SDK.

Passo 4: Na próxima tela checar se os 3 itens a serem instalados, conforme definido anteriormente estão presentes; se sim, clique em '*Next*'.

Passo 5: Selecionar '*I accept the terms of the license agreements*' e, em seguida, clique em '*Finish*' (**Figura 5**).

Passo 6: Após a instalação, clicar em '*Restart Now*' para reiniciar o Eclipse e concluir instalação do *plugin* do Google *App Engine*.

Após seguir esses passos, o ambiente de desenvolvimento já está pronto para a criação de aplicações usando a Google *App Engine*. Ao reiniciar o Eclipse, serão exibidos três novos ícones, conforme a **Figura 6**.

Nota

É possível desenvolver uma aplicação sem a utilização desse ambiente específico, sendo necessário apenas o uso do SDK do Google App Engine para Java – o download encontra-se em *http://code.google.com/intl/pt-BR/appengine/downloads.html*.

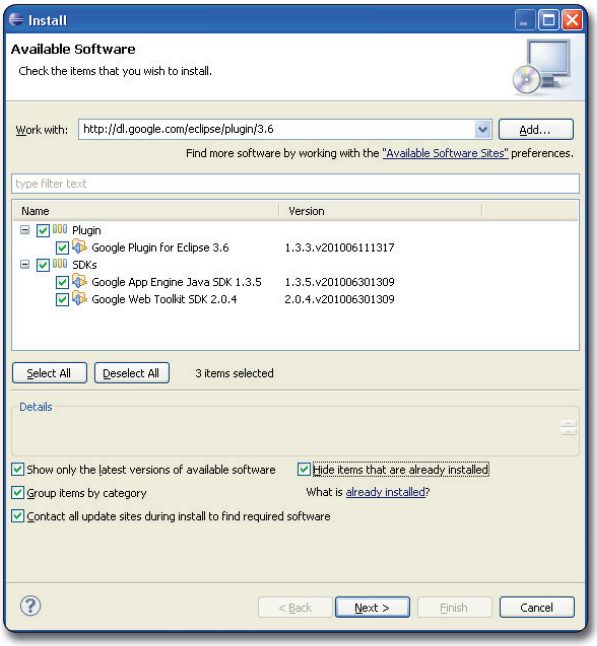

Figura 4. Instalação do plugin da plataforma *App Engine* para Eclipse: *Plugins* e SDKs

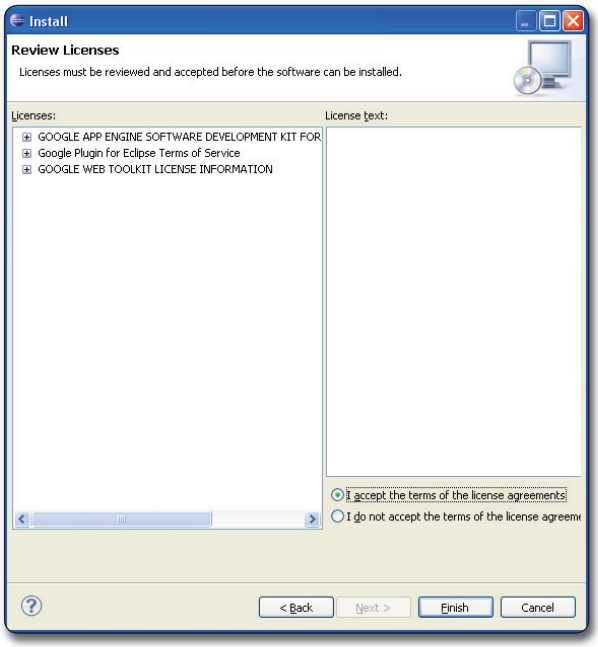

Figura 5. Instalação do plugin da plataforma *App Engine* para Eclipse: aceite dos termos de utilização

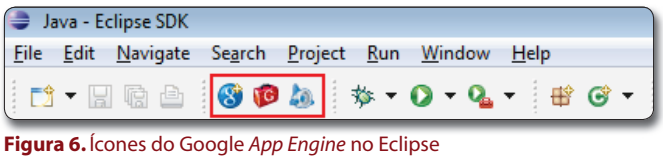

Figura 6. Ícones do Google *App Engine* no Eclipse

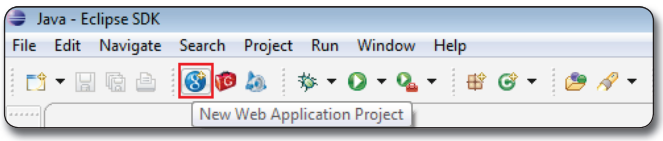

Figura 7. Criação de novo projeto

Os ícones exibidos na **Figura 6** possuem as seguintes funcionalidades, respectivamente:

- O primeiro abre o assistente de criação de projeto do Google *App Engine* para Java;
- O segundo serve para compilar um projeto que faz uso da GWT (Google Web *Toolkit*), uma ferramenta desenvolvida pela Google que facilita as tarefas de criação, reutilização e manutenção de bases de código JavaScript e componentes AJAX para construção de interfaces Web mais amigáveis e rápidas;
- O terceiro ícone, o mini-jato, serve para efetuar o upload de um projeto que faz uso da plataforma *App Engine* para a nuvem do Google.

Criação de um novo projeto Web para a aplicação

A seguir, serão descritos os passos para criação de um projeto Web para a aplicação em construção.

Primeiro, clica-se no ícone para criação de um novo projeto web (**Figura 7**).

Em seguida, deve-se definir o nome do projeto e do pacote. Após isto, desmarque a opção '*Use Google Web Toolkit*' (vide **Figura 8**), já que não serão utilizados muitos recursos de JavaScript ou AJAX, e clicar no botão 'Finish'.

A aplicação CloudDictionary deverá possuir a estrutura de arquivos e diretórios exibida na **Figura 9**, que será detalhada no decorrer deste artigo.

Criação da página principal da aplicação e da folha de estilos

No projeto criado, deve-se renomear o arquivo 'index.html' para 'index.jsp' que será a página inicial da aplicação em

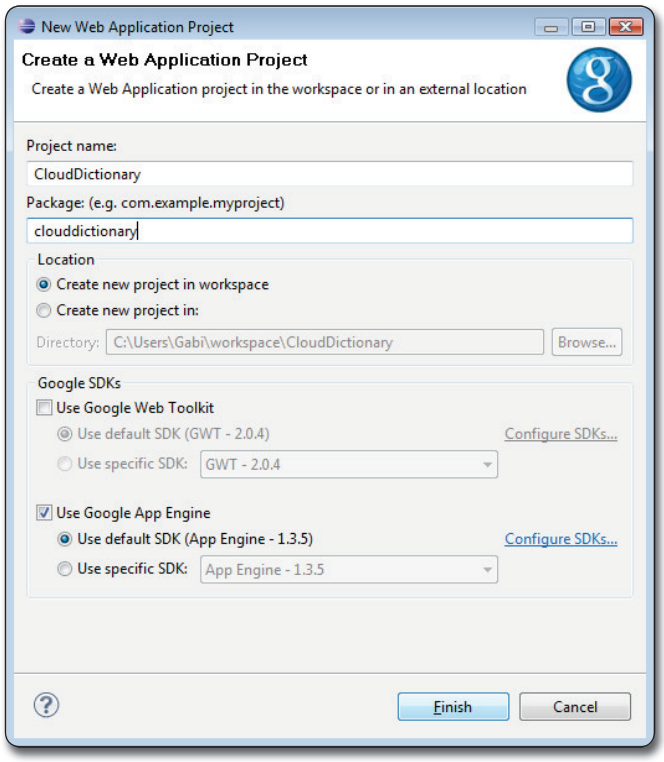

Figura 8. Dados do novo projeto

questão. Este arquivo incluirá um cabeçalho com as opções de *links* de toda a aplicação (Dicionário, Tradutor, Contato) e um texto de apresentação, vide **Listagem 1**.

Observação: Ao renomear um arquivo '.html' para '.jsp', ou ao criar um arquivo '.jsp', o Eclipse pode exibir um ícone de erro em todos os arquivos dessa extensão, conforme pode ser visto na **Figura 10**.

Esse problema pode ser solucionado alterando o caminho da JRE utilizada pelo projeto. Para isto, basta clicar no menu '*Window*', opção '*Preferences*'. Na próxima tela, clicar em '*Installed JREs*', selecionar a JRE instalada e clicar no botão '*Edit*', conforme **Figura 11**.

Listagem 1. index.jsp

```
01 <%@include file="cabecalho.html" %>
02 <div id="page">
03 <div id="content">
04 <div class="post">
05 <br />
06 <h1 class="title">
07       <a href="#"> Bem vindo ao CloudDictionary!</a>
08 </h1>
09 <br /><br />
10 <h2>Aqui você encontra:</h2>
11 <br /><br />
12 - Um dicionário atualizado da Língua Portuguesa
13 <br /><br />
14 - Um tradutor Português-Inglês e Inglês-Português
15 <br /><br /><br />
16 <h1 class="title">
17       <a href="#">Escolha a opção desejada no menu superior.</a>
18 </h1>
19 <br />
20 </div>
21 </div>
22 <div style="clear: both;"> </div>
23 </div>
24 <%@include file="rodape.html" %>
```

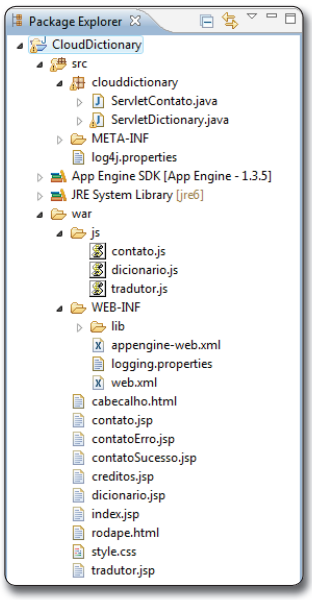

Figura 9. Estrutura da aplicação CloudDictionary

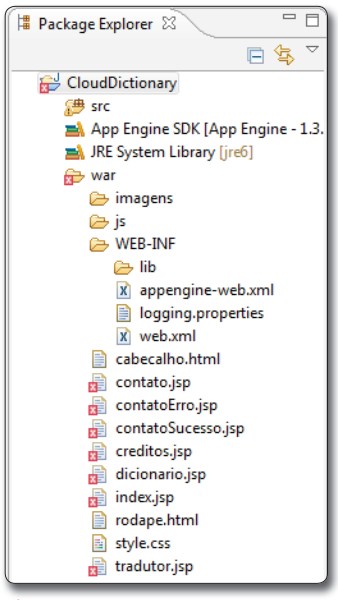

Figura 10. Erro ao renomear um arquivo '.html' para '.jsp'

Passo 3: Na próxima tela, clique no botão 'Directory...' e indique o diretório da jdk instalada, de acordo com a **Figura 12** e clique no botão 'Finish'.

Como o arquivo index.html foi renomeado, é necessária a alteração do arquivo web.xml que referencia o arquivo que será a página inicial do sistema, conforme pode ser visto na **Figura 13**.

Em seguida deve-se criar uma nova página html, com o nome 'cabecalho.html', que contém os links para todas as páginas da aplicação, e cujo código fonte encontra-se na **Listagem 2**.

Em seguida, deve-se efetuar a criação de uma nova página html, com o nome 'rodape.html', que será o rodapé padrão para todas as páginas da aplicação e cujo código fonte encontra-se na **Listagem 3**.

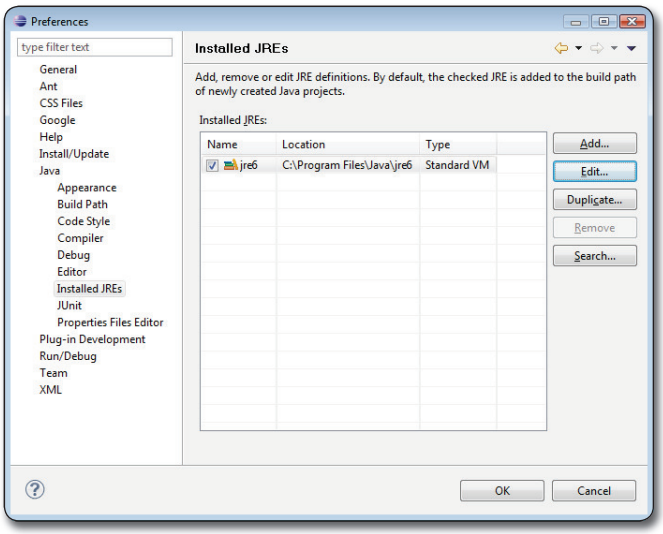

Figura 11. Alterar caminho da JRE

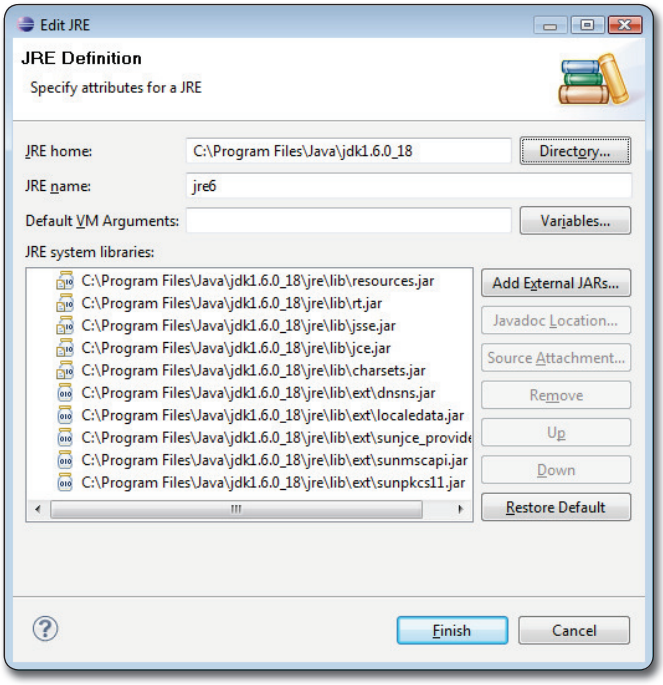

Figura 12. Definir caminho da JRE

O próximo passo será a criação do arquivo 'style.css', responsável pelos estilos que serão aplicados aos elementos da aplicação. Esta folha de estilos foi retirada do site *Free CSS Templates*, sendo necessários apenas alguns ajustes para que ela se adequasse à aplicação em questão. Alguns fragmentos podem ser visualizados na **Listagem 4**.

Criação do Dicionário e do Tradutor

Para a criação da página do dicionário, deve ser criada uma nova página jsp, nomeada 'dicionario.jsp', que inclui o cabeçalho da aplicação, um campo de *input* de texto para inserir a palavra a ser buscada e um botão de busca. Por fim, é incluído o rodapé da aplicação, conforme **Listagem 5**.

A página 'dicionario.jsp', anteriormente criada, consiste em um formulário para envio de dados a um *servlet* 'ServletDictionary.java' (ponto principal do CloudDictionary que faz uso do *URL Fetch Service* do Google *App Engine* para acessar os serviços de um dicionário externo à nuvem, o dicionário online da língua portuguesa Michaelis, da editora Melhoramentos. Para evitar que a página inteira fosse recarregada, foi usado o arquivo 'dicionario.js' (**Listagem 6**), para interligar o arquivo jsp ao *servlet*. Este arquivo deve ser criado no diretório 'js'.

Para a criação da página do tradutor deve ser criada uma nova página jsp, nomeada 'tradutor.jsp', que inclui o cabeçalho da aplicação, um campo de *input* de texto para inserir a palavra a ser traduzida, dois *radio buttons* onde escolhe-se o tipo da tradução a ser realizada e um botão de busca. Por fim, é incluído o rodapé da aplicação, conforme **Listagem 7**.

A página 'tradutor.jsp', consiste em um formulário para envio de dados ao mesmo *servlet* usado pelo formulário do arquivo 'dicionario.jsp'. Para evitar que a página inteira fosse recarregada, foi usado o arquivo 'tradutor.js' (**Listagem 8**), para interligar o arquivo jsp ao *servlet*. Este arquivo deve ser criado no diretório 'js'.

O passo seguinte é a criação do 'ServletDictionary.java'. Esse *servlet* é chamado tanto pela página de dicionário quanto pelo tradutor do CloudDictionary. A principal função consiste em efetuar uma chamada a uma URL do dicionário externo a ser usado, através do serviço *URL Fetch do Google App Engine*, conforme **Figura 14** e **Listagem 9**. Por fim, o código HTML retornado pelo dicionário externo é tratado e exibido na página da aplicação desenvolvida (no dicionário ou no tradutor).

Após a criação do servlet, é necessário que este esteja corretamente mapeado no arquivo web.xml, conforme pode ser visto na **Figura 15**.

Criação da página para Contato

Para complementar a aplicação desenvolvida, deve ser criado um novo arquivo, 'contato.jsp', com um formulário que permite o envio de mensagens ou sugestões para o e-mail do autor, que fará uso do serviço de envio de e-mails do Google *App Engine*. Este formulário pode ser visualizado na **Figura 16** e o código fonte deste encontra-se na **Listagem 10**.

Listagem 2. cabecalho.html

```
01 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
02 <html>
03 <head>
04 <meta http-equiv="content-type" content="text/html;charset=iso-8859-1">
05  <title>Gabriella Castro - Cloud Dictionary</title>
06  <link href="style.css" rel="stylesheet" type="text/css"/>
07 </head>
08 <body>
09  <div id="header-wrapper">
10   <div id="header">
11    <div id="logo">
12     <h1><a href="index.jsp"><span>Cloud</span>Dictionary</a></h1>
13    </div>
14    <div id="menu">
15     <ul>
16      <li><a href="dicionario.jsp">Dicionário</a></li>
17      <li><a href="tradutor.jsp">Tradutor</a></li>
18      <li><a href="contato.jsp">Contato</a></li>
19      <li><a href="creditos.jsp">Cr&eacute;ditos</a></li>
20     </ul>
21    </div>
22   </div>
23  </div>
24  <div id="wrapper">
```

Listagem 3. rodape.html

```
01 </div>
02  <div id="footer">
03   <p> - © 2010 - Gabriella Castro - </p>
04  </div>
05 </body>
06 </html>
```

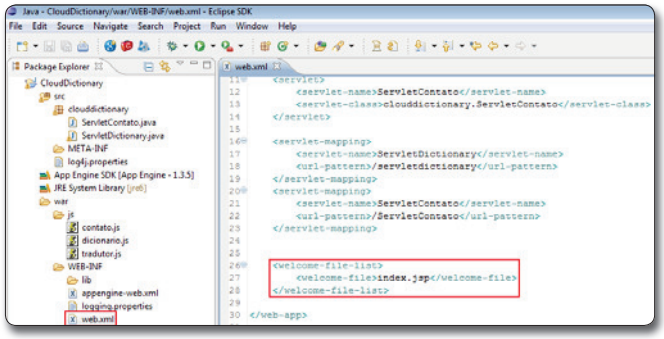

Figura 13. Alteração do arquivo web.xml

Figura 14. Serviço URL Fetch do Google App Engine

Listagem 4. style.css

```
01 /*
02 Design by Free CSS Templates
03 http://www.freecsstemplates.org
04 Released for free under a Creative Commons Attribution 2.5 License
05 */
06 body {
07   margin: 0 auto;
08   padding: 0;
09   background: #FFFFFF;
10   font-family: Arial, Helvetica, sans-serif;
11   font-size: 14px;
12   color: #3C3D3F;
13 }
14 h1, h2, h3 {
15   margin: 0;
16   padding: 0;
17   font-weight: normal;
18   color: #FF3000;
19 }
20 h1 {
21   font-size: 2em;
22 }
...
46 #wrapper {
47   padding: 0;
48 }
49 /* Header */
50 #header {
51   overflow: hidden;
52   width: 1000px;
53   height: 60px;
54   margin: 0px auto 20px auto;
55 }
56 /* Logo */
57 #logo {
58   float: left;
59   width: 300px;
60   margin: 0;
61   padding: 0;
62   color: #000000;
63 }
64 #logo h1 {
65   letter-spacing: -1px;
66   font-size: 2.8em;
67   color: #0C0C0C;
68 }
...
83 #logo a {
84   border: none;
85   background: none;
86   text-decoration: none;
87   color: #45302C;
88 }
...
121 /* Menu */
122 #menu {
123   float: right;
124   width: 600px;
125 }
126 #menu ul {
127   margin: 0px;
128   padding: 0px 0px 0px 15px;
129   list-style: none;
130 }
131 #menu li {
132   float: left;
133 }
134 #menu a {
135   display: block;
136   float: left;
137   height: 37px;
138   padding: 13px 30px 0px 30px;
139   text-decoration: none;
140   text-transform: uppercase;
141   font-family: Arial, Helvetica, sans-serif;
142   font-size: 14px;
143   font-weight: bold;
144   color: #0C0C0C;
145 }
```

Listagem 5. dicionario.jsp

```
01 <%@include file="cabecalho.html"%>
02 <script type="text/javascript" src="js/dicionario.js"></script>
03 <div id="page">
04   <h3>Dicionário</h3>
05   <div id="content">
06     <div class="post">
07       <h2 class="title"><a href="#">Digite a palavra:</a></h2>
08       <br />
09       <input type="text" id="palavra"></input>
10       <br /><br />
11       <button id="search-submit" class="botao"
12       onclick="procuraPalavra(palavra.value)">
13         Buscar
14       </button>
15       <br />
16       <div class="entry">
17         <span id="resultado"></span>
18       </div>
19     </div>
20   </div>
21   <div style="clear: both;"> </div>
22 </div>
23 <%@include file="rodape.html" %>
```

Após, deve-se criar um novo arquivo JavaScript 'contato.js'. Este arquivo deverá conter uma função para validação do formulário, conforme **Listagem 11**, de forma a impedir o envio do e-mail sem que os campos sejam corretamente preenchidos.

Para o funcionamento do formulário, deve ser criado um novo *servlet* 'ServletContato.java'. Esse *servlet* tem como função principal a utilização do serviço de envio de mensagens da plataforma. Ele realiza o envio de mensagens de acordo com os campos informados no formulário de contato para um email previamente definido (**Listagem 12** – linha 27). Uma restrição do Google *App Engine* é que esse serviço de envio de mensagens só pode partir de e-mails previamente cadastrados na área de administração da aplicação (**Figura 17**). Portanto, como o objetivo de um formulário de contato é enviar um e-mail para o responsável pela aplicação e não é possível colocar como remetente o e-mail que foi informado no campo do formulário, deve-se colocar o e-mail cadastrado na plataforma como o remetente e o e-mail da pessoa que efetuou o contato no corpo da mensagem.

Após criar o *servlet*, é necessário que ele esteja corretamente mapeado no arquivo web.xml, conforme pode ser visto na **Figura 18**.

Listagem 6. dicionario.js

```
01 var xmlhttp;
02 function procuraPalavra(palavra) {
03  xmlhttp=null;
04  if (window.XMLHttpRequest){
05   //Para IE7, Firefox, Opera, etc.
06   xmlhttp=new XMLHttpRequest();
07  }
08  else if (window.ActiveXObject) {
09   //Para IE6, IE5
10   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
11  }
12  if (xmlhttp!=null){
13   xmlhttp.onreadystatechange=status;
14   var url = "/servletdictionary?word="+palavra;
15   xmlhttp.open("GET",url,true);
16   xmlhttp.send(null);
17  }
18  else {
19   alert("Seu navegador não oferece suporte a XMLHTTP.");
20  }
21 }
22 function status()   {
23  if (xmlhttp.readyState==4){
24   // 4 = "loaded"
25   if (xmlhttp.status==200){
26    // 200 = "OK"
27    document.getElementById('resultado').innerHTML=
28    xmlhttp.responseText;
29  }
30   else{
31    alert("Erro ao acessar o dicionário:" + xmlhttp.statusText);
32   }
33  }
34 }
```

Listagem 7. tradutor.jsp

```
01 <%@include file="cabecalho.html" %>
02 <script type="text/javascript" src="js/tradutor.js"></script>
03 <div id="page">
04  <h3>Tradutor</h3>
05  <div id="content">
06   <div class="post">
07    <h2 class="title"><a href="#">Digite a palavra:</a></h2>
08    <br />
09    <input type="text" name="palavra" id="palavra" />
10    <br />
11    <input type="radio" name="chk"  value="1" checked
12    onclick="getValue(this.value)" /> Português / Inglês   
13    <input type="radio" name="chk"  value="2"
14    onclick="getValue(this.value)" /> Inglês / Português<br>
15    <input type="hidden" name="lingua" id="lingua" value="1" />
16    <br />
17    <button id="search-submit" class="botao"
18    onclick="procuraPalavra(palavra.value, lingua.value)">
19     Buscar
20    </button>
21    <br />
22    <div class="entry">
23     <span id="resultado"></span>
24    </div>
25   </div>
26  </div>
27 <div style="clear: both;"> </div>
28 </div>
29<%@include file="rodape.html" %>
```

Listagem 8. tradutor.js

```
01 var xmlhttp;
02 function procuraPalavra(palavra, lingua) {
03  xmlhttp=null;
04  if (window.XMLHttpRequest){
05   //Para IE7, Firefox, Opera, etc.
06   xmlhttp=new XMLHttpRequest();
07  }
08  else if (window.ActiveXObject) {
09   //Para IE6, IE5
10   xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
11  }
12  if (xmlhttp!=null){
13   xmlhttp.onreadystatechange=status;
14   var url = "/servletdictionary?word="+palavra+"&lang="+lingua;
15   xmlhttp.open("GET",url,true);
16   xmlhttp.send(null);
17}
18  else {
19   alert("Seu navegador não oferece suporte a XMLHTTP.");
20  }
21 }
22 function status()   {
23  if (xmlhttp.readyState==4){
24   // 4 = "loaded"
25   if (xmlhttp.status==200){
26    // 200 = "OK"
27    document.getElementById('resultado').innerHTML=
28    xmlhttp.responseText;
29  }
30  else{
31   alert("Erro ao acessar o tradutor:" + xmlhttp.statusText);
32   }
33  }
34 }
35 function getValue(radio) {
36  var obj = document.getElementById('lingua');
37  obj.value=radio;
38 }
```

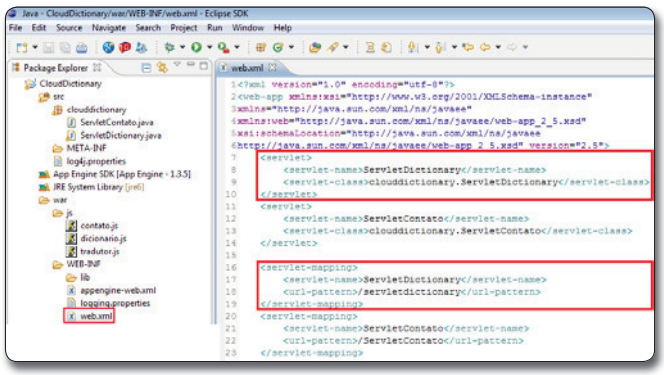

Figura 15. Alteração do arquivo web.xml – ServletDicitionary

Figura 16. Formulário de contato

Listagem 9. ServletDictionary.java

```
01 package clouddictionary;
02 import java.io.BufferedReader;
03 import java.io.IOException;
04 import java.io.InputStreamReader;
05 import java.net.URL;
06 import javax.servlet.ServletException;
07 import javax.servlet.http.*;
08 import java.util.ArrayList;
09 import java.net.URLEncoder;
10
11 @SuppressWarnings("serial")
12 public class ServletDictionary extends HttpServlet {
13 public void doGet(HttpServletRequest req, HttpServletResponse resp)
14 throws IOException {
15   String resultado = "";
16   resp.setContentType("text/plain");
17   try {
18    String palavra = req.getParameter("word");
19    String language = req.getParameter("lang");
20    if (palavra == null)
21     throw new Exception("Informe a palavra a ser buscada.");
22     palavra = palavra.trim();
23    if (palavra.length() == 0)
24     throw new Exception("Informe a palavra a ser buscada.");
25     String busca="";
26    if (language == null){
27     busca = "http://michaelis.uol.com.br/moderno/portugues/
       index.php?lingua=portugues-portugues&palavra=";
28    }
29    else{
30     if (language.equals("1")){
31      busca = "http://michaelis.uol.com.br/moderno/portugues/
        index.php?lingua=portugues-ingles&palavra=";
32     }
33     else if (language.equals("2")){
34      busca = "http://michaelis.uol.com.br/moderno/portugues/
        index.php?lingua=ingles-portugues&palavra=";
35     }
36    }
37    String a = URLEncoder.encode(palavra,"iso-8859-1");
38    busca += a;
39    URL url = new URL(busca);
40    BufferedReader reader = new BufferedReader(new InputStreamReader
      (url.openStream()));
41    StringBuffer response = new StringBuffer();
42    String line;
43    ArrayList lines = new ArrayList();
44    while ((line = reader.readLine()) != null) {
45     lines.add(line);
46    }
47    String wishedLine = (String) lines.get(284);
48    response.append(wishedLine);
49    reader.close();
50    resultado = response.toString();
51    resp.getWriter().println(resultado);
52   }
53   catch (Exception ex) {
54    resultado = "Erro: " + ex.getMessage();
55    resp.getWriter().println(resultado);
56   }
57  }
58  @Override
59  public void doPost(HttpServletRequest req, HttpServletResponse resp)
60  throws ServletException, IOException {
61   doGet(req, resp);
62  }
63 }
```

Listagem 10. contato.jsp

```
01 <%@include file="cabecalho.html" %>
02 <script type="text/javascript" src="js/contato.js"></script>
03 <div id="page">
04  <h3>Contato</h3>
05  <div id="content">
06   <div class="post">
07    <form name="form_contato" id="form_contato"
       action="ServletContato" method="POST" onSubmit=
       "return validaForm()">
08     <table id="contato" align="center">
09      <tr>
10       <td>
11        Nome:
12       </td>
13       <td>
14        <input type="text" name="nome" id="nome" size='25'/><br/>
15       </td>
16      </tr>
17      <tr>
18       <td>
19        Email:
20       </td>
21       <td>
22        <input type="text" name="email" id="email" size='25'/><br/>
23       </td>
24      </tr>
25      <tr>
26       <td>
27        Mensagem:
28       </td>
29       <td>
30        <textarea name="mensagem" id="mensagem" cols="19">
31        </textarea><br/>
32       </td>
33      </tr>
34      <tr>
35       <td colspan="2">
36        <input type="reset" class="botao" value="Limpar">
37        <input type="submit" class="botao" value="Enviar">
38       </td>
39      </tr>
40     </table>
41    </form>
42   </div>
43  </div>
44  <div style="clear: both;"> </div>
45 </div>
46 <%@include file="rodape.html" %>
```

Conforme a linha 31, da **Listagem 12**, se a mensagem for enviada, ocorrerá um redirecionamento para a página 'contatoSucesso.jsp', cujo código encontra-se na **Listagem 13**.

Conforme a linha 34, da **Listagem 12**, se a mensagem não for enviada, ocorrerá um redirecionamento para a página 'contatoErro.jsp', cujo código encontra-se na **Listagem 14**.

Executando a Aplicação

Para executar a aplicação localmente, basta clicar com o botão direito no projeto desenvolvido, selecionar a opção '*Run as*' e clicar na opção '*Web Application*', como pode ser visto na **Figura 19**.

Listagem 11. contato.js

```
01 function validaForm(){
02  d = document.form_contato;
03  if (d.nome.value == ""){
04   alert("O campo " + d.nome.name + " deve ser preenchido!");
05   d.nome.focus();
06   return false;
07  }
08  if (d.email.value == ""){
09   alert("O campo " + d.email.name + " deve ser preenchido!");
10   d.email.focus();
11   return false;
12  }
13  parte1 = d.email.value.indexOf("@");
14  parte2 = d.email.value.indexOf(".");
15  parte3 = d.email.value.length;
16  if (!(parte1 >= 3 && parte2 >= 6 && parte3 >= 9)) {
17   alert ("Email inválido!");
18   d.email.focus();
19   return false;
20  }
21  if (d.mensagem.value == ""){
22   alert ("O campo " + d.mensagem.name + " deve ser preenchido!");
23   d.telefone.focus();
24   return false;
25  }
26  return true;
27 }
```

Através do navegador pode-se acessar a aplicação. A **Figura 20** exibe a página inicial do CloudDictionary sendo utilizado localmente.

Efetue o upload da aplicação para a nuvem do Google

Inicialmente, caso o usuário da plataforma *App Engine* não possua uma conta vinculada ao Google, você deverá criá-la. Caso contrário, basta efetuar o *login* no *link* de acesso à conta do Google *App Engine*.

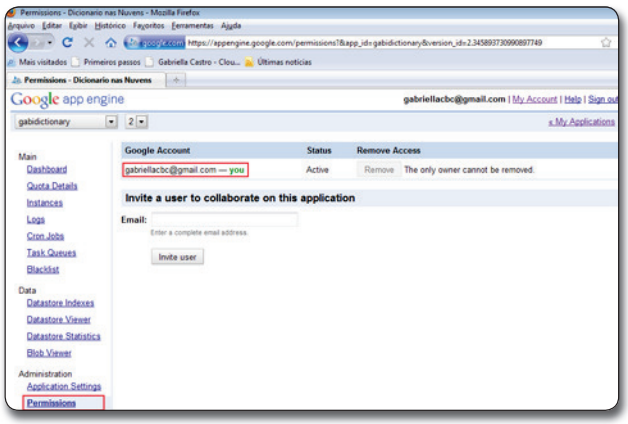

Figura 17. Cadastro de e-mails autorizados para enviar mensagens

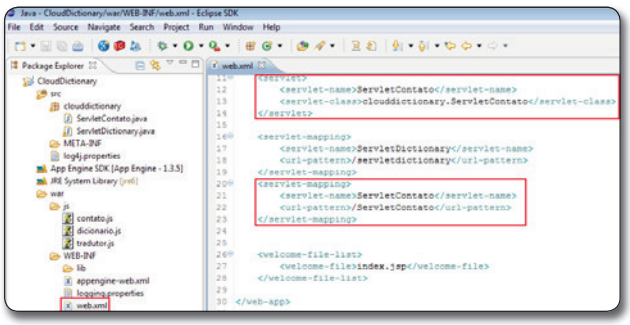

Figura 18. Alteração do arquivo web.xml – ServletContato

Listagem 12. ServletContato.java

```
01  package clouddictionary;
02  import java.io.IOException;
03  import java.util.Properties;
04  import javax.mail.Message;
05  import javax.mail.Session;
06  import javax.mail.Transport;
07  import javax.mail.internet.InternetAddress;
08  import javax.mail.internet.MimeMessage;
09  import javax.servlet.ServletException;
10  import javax.servlet.http.*;
11  @SuppressWarnings("serial")
12  public class ServletContato extends HttpServlet {
13   public void doPost(HttpServletRequest req, HttpServletResponse resp)
14   throws ServletException, IOException {
15    resp.setContentType("text/plain");
16    try {
17     String nome = req.getParameter("nome");
18     String email = req.getParameter("email");
19     String msgm = req.getParameter("mensagem");
20     String mensagem = "Nome:" + nome + "\n" +
21                       "Email: " + email + "\n" +
22                       "Mensagem: " + msgm;
23     Properties props = new Properties();
24     Session session = Session.getDefaultInstance(props, null);
25     Message msg = new MimeMessage(session);
26     msg.setFrom(new InternetAddress("gabriellacbc@gmail.com"));
27     msg.addRecipient(Message.RecipientType.TO, new InternetAddress
       ("gabriellacbc@gmail.com"));
28     msg.setSubject("CloudDictionaty - Contato");
29     msg.setText(mensagem);
30     Transport.send(msg);
31     resp.sendRedirect("contatoSucesso.jsp");
32    }
33    catch (Exception ex) {
34     resp.sendRedirect("contatoErro.jsp");
35    }
36   }
37   @Override
38   public void doGet(HttpServletRequest req, HttpServletResponse resp)
39   throws ServletException, IOException {
40    doPost(req, resp);
41   }
42  }
```

Em seguida, uma nova aplicação deverá ser criada. Para isto basta clicar no botão '*Create Application*', conforme **Figura 21**. Uma conta gratuita permite a criação de até dez aplicações.

Você deverá então informar o identificador e o título da aplicação. Neste exemplo foram usados como identificador 'gabidictionary' e título 'Dicionário nas Nuvens'. Após isto, deve-se clicar no botão '*Create Application*', conforme **Figura 22**.

Na IDE Eclipse, deve-se selecionar o projeto desenvolvido e clicar no ícone do Google *App Engine*, exibido em destaque na **Figura 23**.

Conforme exibido na **Figura 24**, deve-se informar o e-mail e a senha da conta do Google que foi usado no primeiro passo. Como é o primeiro *upload* desta aplicação, deve-se clicar no link '*App Engine Project settings*'.

Na tela aberta, conforme **Figura 25**, deve-se informar no campo *Application ID* o mesmo identificador da aplicação que foi usado anteriormente e, no campo seguinte, a versão da aplicação.

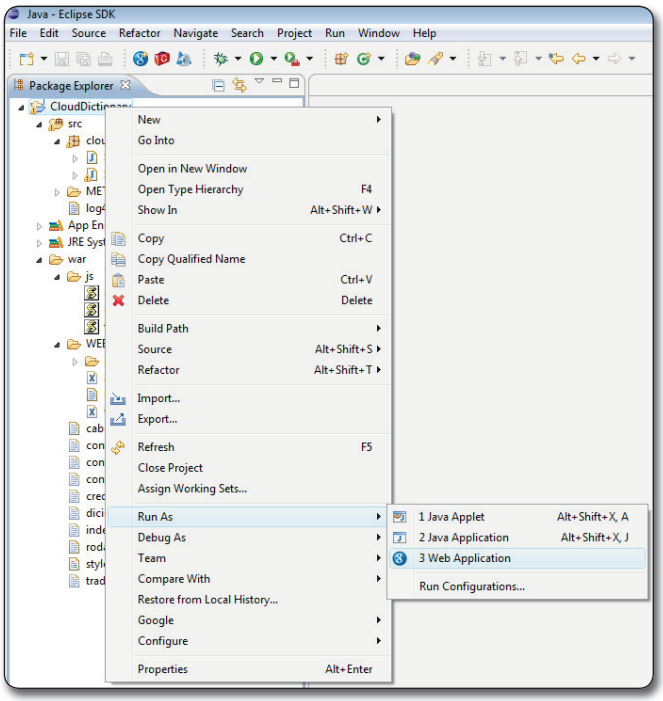

Figura 19. Rodar a aplicação no servidor local

Figura 20. Página inicial do CloudDictionary

Listagem 13. contatoSucesso.jsp

```
01 <%@include file="cabecalho.html"%>
02 <div id="page">
03  <div id="content">
04   <div class="post">
05    <h1 class='title'>Mensagem enviada!</h1>
06   </div>
07  </div>
08  <div style="clear: both;"> </div>
09 </div>
10 <%@include file="rodape.html"%>
```

Listagem 14. contatoErro.jsp

```
01 <%@include file="cabecalho.html"%>
02 <div id="page">
03  <div id="content">
04   <div class="post">
05    <h1 class='title'>
06     Erro ao enviar mensagem!<br/>
07     Tente novamente!
08    </h1>
09   </div>
10  </div>
11  <div style="clear: both;"> </div>
12 </div>
13 <%@include file="rodape.html"%>
```

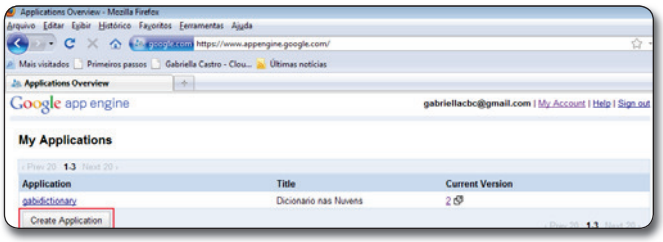

Figura 21. Criar aplicação

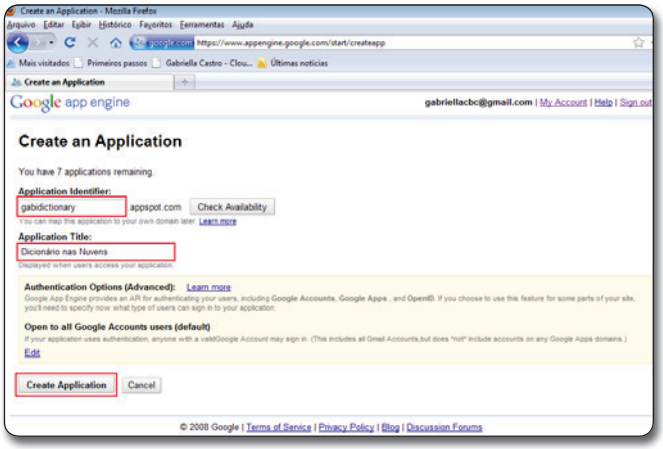

Figura 22. Dados da aplicação

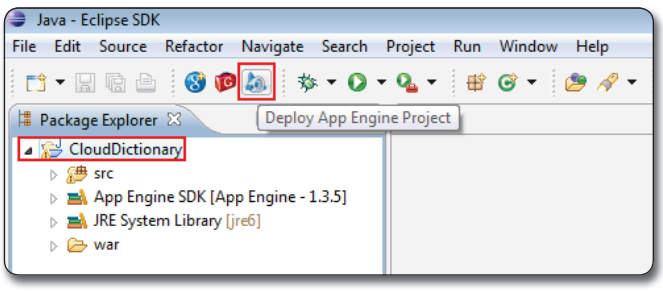

Figura 23. Deploy App Engine Project

Clica-se então no botão '*Deploy*', conforme exibido na **Figura 24**, e aguarde o término do *upload* da aplicação, até que a mensagem da **Figura 26** seja exibida.

Após os passos anteriores, a aplicação já poderá ser acessada através de qualquer navegador web, no endereço <identificador_da_aplicação>.appspot.com.br.

O Google *App Engine* fornece também um painel de controle para monitoramento das aplicações que se encontram na nuvem do Google, exibido na **Figura 27**. O painel de controle pode ser acessado ao clicar no nome da aplicação criada, que pode ser visualizado na **Figura 21**.

Conclusão

Este artigo abordou a criação de uma aplicação real que se encontra em funcionamento na nuvem do Google e faz uso de sua plataforma, a *App Engine*, mostrando que é viável e que a plataforma, mesmo sendo gratuita, é de grande utilidade e funciona nos moldes propostos pelo paradigma da Computação em Nuvem. Vale ressaltar que atualmente a plataforma tem como desvantagem a limitação quanto ao uso de linguagens de programação (permite apenas Phyton e Java), porém oferece o uso de APIs bem definidas e alto grau de escalabilidade. Outra grande vantagem é o fato de que os desenvolvedores que fazem uso da plataforma não precisam preocupar-se, em um momento inicial, com questões como hospedagem, armazenamento e crescimento das aplicações. A plataforma mostrou-se bastante útil para aplicações simples e que não requerem um alto nível de segurança em seus processos e que são voltadas a um número crescente de usuários, além de disponibilizar uma documentação apropriada e de qualidade fornecida pelo próprio Google.

Como uma forma de aprimorar o que foi aprendido neste artigo, uma dica seria implementar a mesma aplicação em outras plataformas oferecidas em nuvem pela Microsoft ou pela Amazon, por exemplo.

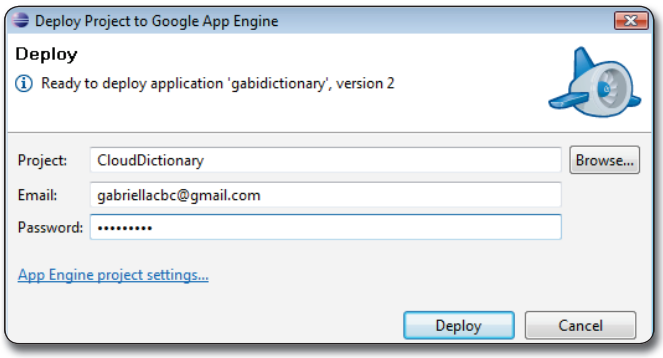

Figura 24. Dados de acesso à conta do Google App Engine

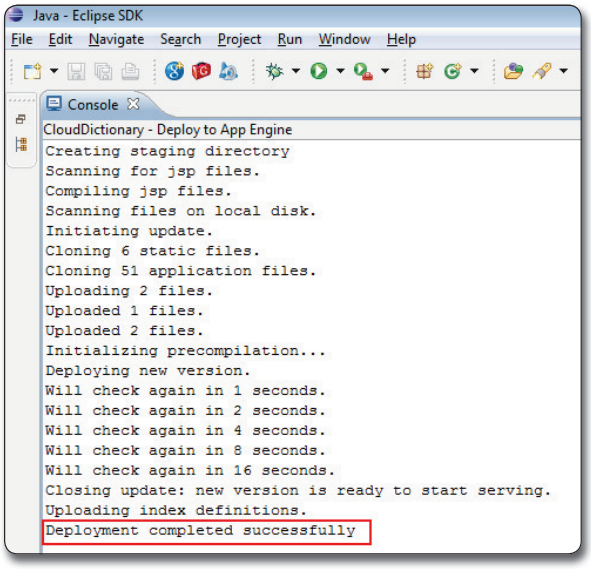

Figura 26. Mensagem de término do upload

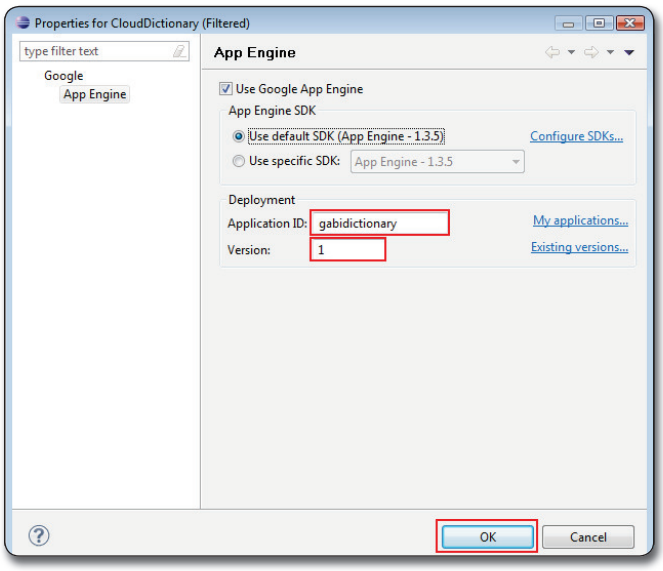

Figura 25. Application ID e versão da aplicação

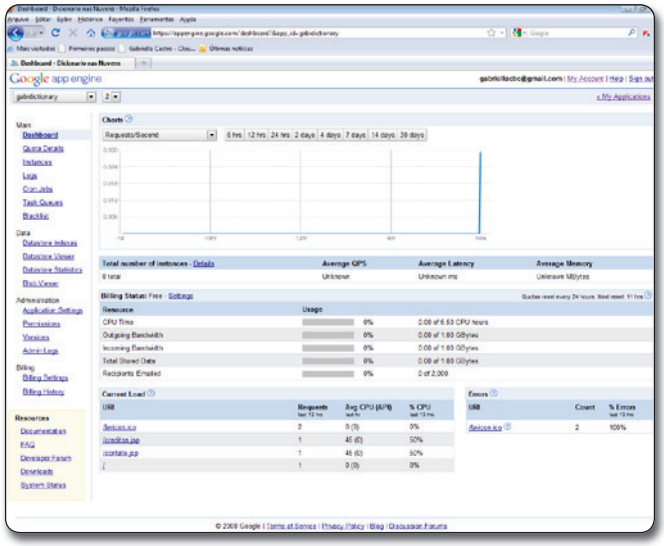

Figura 27. Painel de monitoramento de uma aplicação no Google App Engine

Referências

COSTA, Gabriella Castro Barbosa. CloudDictionary, 2010. Disponível em: <http://gabidictionary.appspot.com>.

Eclipse, SDK. Downloads, 2010. Disponível em: <http://www.eclipse.org/downloads/>.

GOOGLE, App Engine. Downloads, 2010a. Disponível em: <http://code.google.com/intl/pt-BR/appengine/downloads.html>.

GOOGLE, Application Engine. Google Code, 2010b. Disponível em: <http://code.google.com/intl/pt-BR/appengine/>.

GOOGLE. Bigtable: A Distributed Storage System for Structured Data, 2010c. Disponível em: <http://labs.google.com/papers/bigtable.html>.

GOOGLE. Google Trends, 2010d. Disponível em: <http://www.google.com/trends?q=cloud+computing>.

GOOGLE. Google Web Toolkit, 2010e. Disponível em: <http://code.google.com/intl/pt-BR/webtoolkit/>.

GOOGLE. Top ten advantages of Google's cloud, 2010f. Disponível em: <http://www.google.com/apps/intl/en/business/cloud.html>.

IRANI, Romin K. Google App Engine Java Experiments. [S.l.]: Romin K. Irani, 2010.

MELL, Peter; GRANCE, Tim. The NIST Definition of Cloud Computing, 2009. Disponível em: <http://csrc.nist.gov/groups/SNS/cloud-computing/>.

MICHAELIS. Dicionário Online, 2009. Disponível em: <http://michaelis.uol.com.br/>.

MILLER, Michael. Cloud Computing: Web-Based Applications That Change the Way You Work and Collaborate Online. Indianapolis: Que, 2009.

RITTINGHOUSE, John; RANSOME, W; JAMES, F. Cloud Computing. [S.l.]: CRC Press, 2009.

TAURION, Cezar. Cloud computing: computação em nuvem: transformando o mundo da tecnologia da informação. Rio de Janeiro: Brasport, 2009.

VELTE, Antony T; VELTE, Toby J; ELSENPETER, Robert. Cloud Computing: A Practical Approach. [S. l.]: McGraw-Hill, 2010.

Links

Michaelis Dicionário Online
http://michaelis.uol.com.br/

Limites de uso dos recursos do Google App Engine
http://code.google.com/intl/pt-BR/appengine/docs/quotas.html

Free CSS Templates
http://www.freecsstemplates.org

Google Application Engine - Acesso à conta
http://www.appengine.google.com

Dê seu feedback sobre esta edição!

A Engenharia de Software Magazine tem que ser feita ao seu gosto.
Para isso, precisamos saber o que você, leitor, acha da revista!
Dê seu voto sobre este artigo, através do link:

www.devmedia.com.br/esmag/feedback