

Como usar a API do Google Maps

# Distância segura

Saiba como incorporar mapas interativos em suas páginas web

POR ALBERTO PLANAS

**O** *Google Maps* foi um dos serviços mais inovadores da Internet em 2005. Essa foi uma boa notícia também para desenvolvedores, já que sua API (*Application Programming Interface*) é aberta. Baseada em *JavaScript* (ou *ECMAScript* [1]), ela permite a criação de aplicativos que usam os mapas e imagens de satélite do Google. As páginas abrem tranqüilamente nos navegadores, sem a necessidade de nenhuma infraestrutura extra no servidor web. Neste artigo, vamos criar um aplicativo de exemplo que calcula a distância de um trajeto marcado em um mapa.

## Chave

A API do Google Maps está disponível gratuitamente, desde que se respeite algumas regras [2]. Por exemplo, o aplicativo precisa ser gratuito para o público e não pode exceder um certo número de consultas por dia aos servidores do Google. Também não pode esconder a marca Google.

Para usar a API, é preciso antes obter uma chave [3]. Para isso, é preciso uma conta no Google (um *Gmail* já basta ou, então, é possível criar uma conta). Cada chave é associada a uma URL. Se você não tem espaço para hospedar

seu aplicativo, mas tem um servidor *Apache* local, é possível registrar o endereço <http://localhost>.

Tome cuidado, contudo, na hora de indicar uma URL: caso ela contenha erros ou for mal digitada, não será possível acessar a API. É importante fornecer o endereço completo, incluindo portas e diretórios. Por exemplo, em nossos testes – com *Apache* na porta 8080 – criamos um diretório chamado “maps”. É aqui que nosso site de exemplo será hospedado. O endereço que fornecemos foi:

```
http://localhost:8080/maps/
```

É possível criar quantas chaves forem necessárias. Uma vez que os termos de uso tenham sido aceitos, você recebe uma longa cadeia de caracteres alfanuméricos. Essa chave libera a API do Maps em seu aplicativo.

A documentação oficial dessa API [4] recomenda o uso de XHTML no lugar de HTML convencional. A razão para isso é a maior portabilidade dos documentos XHTML. Na **listagem 1** temos um exemplo de documento que acessa a API do Google Maps. O formato XHTML é declarado através do DOCTYPE, na **linha 1**. As **linhas 2, 5 e 9** permitem que o navegador *Internet Explorer* exiba corretamente os efeitos da biblioteca JavaScript do Google Maps (particularmente, os efeitos de trajetos, como será explicado depois).

Para incluir o arquivo de JavaScript contendo a chave da API, use um comando como o da **linha 10** na **listagem 1**. Temos que alterar o trecho `...&key=XXXXX`, incluindo a chave fornecida pelo Google. Curiosamente, todo o XHTML desse primeiro exemplo está entre as **linhas 27 e 29**. Nesse ponto, após carregar o documento HTML, o navegador precisa executar a função JavaScript `onLoad`, definida na **linha 14**. Na **linha 28**, temos um `div` de 500 x 500 pixels, com um identificador `map`, que iremos descrever mais tarde.

A função `onLoad` inicia o mapa. Como já mencionamos, toda a API é escrita em JavaScript. Esse código vai rodar no navegador do visitante. Como nem todos os navegadores implementam a mesma versão e funcionalidade do JavaScript,

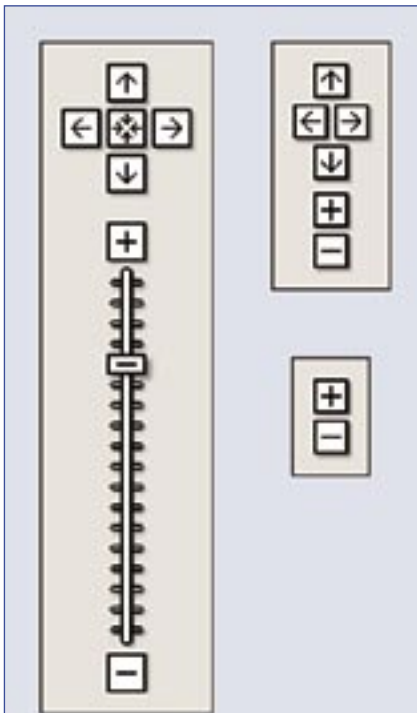
### Listagem 1: teste1.html

```

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/2
  TR/xhtml1/DTD/xhtml1-strict.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml" xmlns:v="urn:schemas-microsoft-2
  com:vm1">
3.   <head>
4.     <title>Exemplo 1 - teste1.html</title>
5.     <style type="text/css">
6.       v\:* {
7.         behavior:url(#default#VML);
8.       }
9.     </style>
10.    <script src="http://maps.google.com/maps?file=api&v=1&key=XXXXX"2
  type="text/javascript"></script>
11.    <script type="text/javascript">
12.      //
13.
14.      function onLoad() {
15.        if (GBrowserIsCompatible()) {
16.          var map = new GMap(document.getElementById("map"));
17.          map.addControl(new GSmallMapControl());
18.          map.addControl(new GMapTypeControl());
19.          map.addControl(new GScaleControl());
20.          map.centerAndZoom(new GPoint(-122.1419, 37.4419), 4);
21.        }
22.      }
23.
24.      //]]&gt;
25.    &lt;/script&gt;
26.
27.    &lt;body onload="onLoad()"&gt;
28.      &lt;div id="map" style="width: 500px; height: 500px"&gt;&lt;/div&gt;
29.    &lt;/body&gt;
30. &lt;/html&gt;
</pre>
</div>
<div data-bbox="692 132 981 343" data-label="Section-Header">
<h1>Um evento para mostrar a força do software livre no segmento desktop</h1>
</div>
<div data-bbox="718 363 956 454" data-label="Text">
<p>Participe,<br/>Divulgue,<br/>Comente</p>
</div>
<div data-bbox="680 454 983 550" data-label="Image">
<img alt="Logo of Desktop Livre featuring a penguin holding a laptop and a mouse."/>
</div>
<div data-bbox="705 560 972 632" data-label="Text">
<p>16, 17 e 18<br/>março de 2006</p>
</div>
<div data-bbox="704 643 836 660" data-label="Text">
<p>Local: UNISAL</p>
</div>
<div data-bbox="704 663 986 742" data-label="Text">
<p>Centro Universitário Salesiano<br/>de São Paulo - Unidade Lorena<br/>Rua Dam Bosco, 284 - Centro<br/>Telefone - (12) 3159-2033</p>
</div>
<div data-bbox="699 756 796 827" data-label="Image">
<img alt="Image of a computer monitor."/>
</div>
<div data-bbox="896 801 968 848" data-label="Image">
<img alt="Image of a computer mouse."/>
</div>
<div data-bbox="708 829 972 873" data-label="Text">
<p>Inscrições no site<br/><a href="http://www.desktoplivre.org.br">www.desktoplivre.org.br</a></p>
</div>
<div data-bbox="708 897 807 914" data-label="Text">
<p>Patrocínio:</p>
</div>
<div data-bbox="708 917 936 977" data-label="Text">
<p>Alternativa Linux • 2MI<br/>Marhost • Tudo Soluções<br/>Unisal • CPTEC/INPE • ITI</p>
</div>
<div data-bbox="499 958 650 973" data-label="Page-Footer">
<p><a href="http://www.linuxmagazine.com.br">www.linuxmagazine.com.br</a></p>
</div>
```

pode haver incompatibilidades. Para garantir que o aplicativo rode em todos os navegadores oficialmente compatíveis, vamos usar a função `GBrowserIsCompatible` (linha 15). Se o usuário estiver usando o *Firefox*, *Safari*, *Opera* ou *Internet Explorer 5.5* (ou superior) não teremos nenhum tipo de problema. Na próxima linha, vamos criar um objeto do tipo *GMap*.

Vamos passar ao “construtor” o objeto `div` identificado por `map`. Esse objeto HTML será usado pelo GMap para inserir um mapa com o tamanho associado de sua `tag`. O GMap oferece uma interface cuja documentação pode ser consultada em [3] (há diversos tipos de “construtores” para esse objeto). Já na linha 17 começamos a usar o método `addControl()`. Usaremos esse método para adicionar diversos controles ao mapa, que nos permitirão modifi-



**Figura 1:** O grande controle da esquerda corresponde a um `GLargeMapControl`. O de tamanho médio na direita é um `GSmallMapControl`. O menor é um `GSmallZoomControl`.

car seu comportamento. Poderemos rolar o mapa, simplesmente pressionando o botão esquerdo do mouse e arrastando o cursor. Podemos usar também um componente que permite mudar o nível de zoom.

Esse componente é aquele que adicionamos na linha 17 da listagem 1. Ele corresponde ao controle no canto superior direito da figura 1. Podemos “brincar” um pouco com o código, mudando-o para:

```
map.addControl(new
GLargeMapControl());
```

Desse modo, podemos incluir controles de zoom e posição separados. Há outros dois tipos de controle que podemos adicionar: um seletor de tipo de mapa e um controle de escala, tanto em milhas quanto quilômetros. Desses dois, o primeiro é o mais importante. Ele foi incluído na linha 18 da listagem 1. Há três tipos de mapas: normal, satélite e híbrido (uma mistura dos dois primeiros). Esses tipos de mapas estão exemplificados entre as figuras 2 e 4. No entanto, mapas com nomes de ruas e endereços, atualmente, só estão disponíveis para os Estados Unidos, Inglaterra e Japão.

Na linha 20 centramos a imagem, especificando latitude, longitude e nível de zoom. Para esse exemplo, escolhemos as coordenadas de Palo Alto, na Califórnia. É preciso um pouco de cuidado na hora de indicar as coordenadas do ponto onde vamos centrar o mapa. Primeiro, temos que indicar a longitude e depois a latitude (no geral, coordenadas costumam ser passadas na ordem inversa). Após essa introdução, vamos adicionar a parte do programa que calcula as distâncias (listagem 2).



**Figura 2:** O mapa com as ruas de uma região de Palo Alto, no estado da Califórnia.



**Figura 3:** A foto de satélite dessa mesma área, mostrando as casas e terrenos vazios.



**Figura 4:** O modo híbrido mistura os dois tipos. É possível ver os nomes das ruas na própria foto.

## Esfera

Qualquer ponto da Terra pode ser localizado usando-se dois números: latitude e longitude. Mas, se soubermos as coordenadas de dois pontos na superfície de uma esfera, ainda não poderemos calcular a distância entre esses dois locais.

Precisamos de mais uma informação: o raio da esfera. Se considerarmos que a Terra tem, na média (o planeta não é uma esfera perfeita), um raio de 6378 km, podemos usar geometria esférica para calcular distâncias baseadas em latitudes e longitudes [5]. Primeiro precisamos converter latitudes e longitudes de graus para radianos. Então, aplicamos a fórmula:

```
d = 6378.7 * acos(seno(lat1)
* seno(lat2) + coseno(lat1) *
coseno(lat2) * coseno(lon2 - lon1))
```

Essa equação não requer muita discussão, já que seu uso pode ser examinado na função `calcDistancia`, da **listagem 2**.

A **listagem 2** é ligeiramente diferente do primeiro exemplo. A primeira diferença é que, na chamada ao construtor `GMap`, estamos fornecendo um parâmetro extra (**linha 21**). Esse parâmetro é uma matriz de elemento único. Nós a usamos para indicar que só queremos o modo de exibição do tipo satélite. A API permite a associação de funções em uma lista pré-definida de eventos. Dessa maneira, podemos alterar o comportamento do sistema em resposta a um clique, uma rolagem ou à adição de uma marca, por exemplo. Há uma lista completa de eventos na documentação oficial. Nosso exemplo requer a captura de dois eventos. Um deles é a rolagem. Após esse evento, o aplicativo precisa redesenhar o mapa na nova localidade.

O outro evento é um clique no mapa, para que possamos definir o trajeto em que o cálculo de distância será feito. A API do Google permite diversas maneiras de capturar eventos. Vamos fazer isso da seguinte maneira:

```
GEvent.addListener(
(map, 'click', function(
(overlay, point) {
  map.recenterOrPanToLatLng(
(point);
});
```

Quando usamos o método `addListener()` na classe `GEvent`, devemos indicar o evento que queremos capturar e a função que irá manipulá-lo. Nesse caso, estamos capturando o clique do mouse no mapa (evento `click`).

A função específica para esse evento pode receber dois parâmetros: o *overlay*

## Aproveite sua liberdade! Faça Linux! Formações em até 12 vezes sem juros!



Novos cursos totalmente preparatórios para a certificação LPI



[www.green.com.br](http://www.green.com.br)



### Formações para Administradores LINUX

Torne-se um profissional especializado em administração de redes Linux, com conhecimentos em várias distribuições.

#### Formação LINUX Specialist - 5 cursos

Fundamentos + Sistemas I + Sistemas II +  
Redes I (+ Apache) + Redes II (+ Samba 3)

#### Formação LINUX Total - 7 cursos

Fundamentos + Sistemas I + Sistemas II + Redes I (+ Apache) +  
Redes II (+ Samba 3) + Firewall + Ferramentas e Serviços

\* Consulte condições detalhadas



Ganhe as provas  
para tirar a sua  
Certificação



Ganhe Suporte  
Técnico direto com a  
Mandriva Conectiva

**BIS**

Repetição de  
cursos sem ônus

## Listagem 2: distancia.html

```

1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2. <html xmlns="http://www.w3.org/1999/xhtml" xmlns:
v="urn:schemas-microsoft-com:vml">
3.   <head>
4.     <title>Calculando distancias</title>
5.     <style type="text/css">
6.       v\:* {
7.         behavior:url(#default#VML);
8.       }
9.     </style>
10.    <script src="http://maps.google.com/maps?file=api&v
=1&key=XXXXX" type="text/javascript"></script>
11.    <script type="text/javascript">
12.      //<![CDATA[
13.
14.        // Pontos do trajeto (GMarker)
15.        var points = new Array();
16.        // Ultima linha desenhada
17.        var polyLine;
18.
19.        function onLoad() {
20.          if (GBrowserIsCompatible()) {
21.            var map = new GMap(document.
getElementById("map"), [G_SATELLITE_TYPE]);
22.            map.addControl(new GSmallMapControl());
23.            map.addControl(new GScaleControl());
24.
25.            GEvent.addListener(map, 'moveend', function() {
26.              var center = map.getCenterLatLng();
27.              var latLngStr = '(' + center.y + ', ' +
center.x + ')';
28.              document.getElementById("latlong").innerHTML
= latLngStr;
29.            });
30.
31.            GEvent.addListener(map, 'click',
function(overlay, point) {
32.              if (overlay) {
33.                removeOverlay(map, points, overlay);
34.              } else if (point) {
35.                addOverlay(map, points, new
GMarker(point));
36.              }
37.
38.              polyLine = drawLine(map, points, polyLine);
39.
40.              var distance = calcDistancia(points);
41.              document.getElementById("distance").innerHTML
= distance + " Km";
42.            });
43.
44.            map.centerAndZoom(new GPoint(-4.48333,
36.66667), 4);
45.          }
46.
47.          function drawLine(map, points, lastLine) {
48.            var p = new Array();
49.            for (var i = 0; i < points.length; i++) {
50.              p.push(new GPoint(points[i].point.x,
points[i].point.y));
51.            }
52.            var newLine = new GPolyline(p);
53.
54.            if (lastLine) {
55.              map.removeOverlay(lastLine);
56.            }
57.            map.addOverlay(newLine);
58.
59.            return newLine;
60.          }
61.
62.          function addOverlay(map, points, overlay) {
63.            map.addOverlay(overlay);
64.            points.push(overlay);
65.          }
66.
67.          function removeOverlay(map, points, overlay) {
68.            map.removeOverlay(overlay);
69.            var oi = -1;
70.            for (var i = 0; i < points.length; i++) {
71.              if (points[i] == overlay) {
72.                oi = i;
73.                break;
74.              }
75.            }
76.            points.splice(oi, 1);
77.          }
78.
79.          function calcDistancia(points) {
80.            var distance = 0.0;
81.            var p1 = points[0];
82.            for (var i = 1; i < points.length; i++) {
83.              var p2 = points[i];
84.              var lat1 = p1.point.y * Math.PI / 180.0;
85.              var lon1 = p1.point.x * Math.PI / 180.0;
86.              var lat2 = p2.point.y * Math.PI / 180.0;
87.              var lon2 = p2.point.x * Math.PI / 180.0;
88.              distance += 6378.7 * Math.acos(Math.sin(lat1)
* Math.sin(lat2) + Math.cos(lat1) * Math.cos(lat2) *
Math.cos(lon2 - lon1));
89.              p1 = p2;
90.            }
91.
92.            return distance;
93.          }
94.        }
95.      //]]>
96.    </script>
97.
98.
99.    <body onload="onLoad()">
100.     <div id="map" style="width: 500px; height:
500px"></div>
101.     <div id="latlong"></div>
102.     <div id="distance"></div>
103.   </body>
104. </html>

```

(ou marca onde clicamos) e o ponto contendo as coordenadas do clique.

Na **linha 25**, capturamos o evento `mousedown`, que será produzido cada vez que terminarmos uma rolagem do mapa. Um importante evento é capturado na **linha 35** da **listagem 2**, quando acrescentamos nosso próprio gerenciador de cliques. Assim permitimos que o visitante coloque diversas marcas, que serão armazenadas em uma matriz. Se clicarmos de novo nessas marcas, podemos removê-las tanto do mapa quanto da matriz. Nesse caso simples, vamos definir um trajeto que podemos traçar com uma linha azul. Nosso cálculo de distância será baseado nesse trajeto.

Tanto marcas quanto linhas são chamadas de *overlays* na documentação oficial do Google Maps. Cada overlay é um objeto sobreposto ao mapa. Embora possamos definir ícones personalizados nas marcas (vide a documentação), podemos também usar diretamente o conjunto padrão fornecido. Uma marca pode ser criada e posicionada no mapa da seguinte maneira:

```
var m = new Gmarker (new GPoint(lon1, 2
lat1));
map.addOverlay(m);
```

Devemos criar uma marca em coordenadas específicas. Depois, vamos acrescentá-la ao mapa usando o método `addOverlay()` da classe `GMap`. Esse é o mesmo método que usaremos para desenhar o trajeto, mas em vez de acrescentar um objeto do tipo `GMarker`, usaremos um `GPolyline`.

```
var p = new Array;
p.push(new GPoint(lon1, lat1));
p.push(new GPoint(lon2, lat2));
...
map.addOverlay(new GPolyline(p));
```

Isso é apenas o que as funções `addOverlay()` e `drawLine()` fazem em nosso código. Um exemplo de um trajeto desenhado é o caminho mostrado na **figura 5**. Uma vez que o caminho (ou parte dele) é terminado, podemos calcular a distância usando a fórmula já citada.

## Conclusão

Usando não mais que quatro objetos e oito métodos diferentes, criamos um aplicativo capaz de calcular o comprimento de um caminho desenhado interativamente em um mapa aberto em um navegador. A API do Google Maps torna fácil a criação de aplicativos inovadores e interessantes que, de outra maneira, exigiriam profundas noções de programação, além de conhecimento elevado de matemática e navegação global.

A API fornece outro grupo de objetos para “acesso assíncrono a dados XML por JavaScript”, tecnologia mais conhecida como Ajax [6]. Esse conjunto de objetos permite o armazenamento de uma grande quantidade de dados em um banco de dados, para que eles possam ser “pintados” instantaneamente na medida em que o usuário vai navegando pelo mapa. Muitos usuários dessa API estão desenvolvendo aplicativos interessantes – veja os curiosos exemplos [8], [9] e [10]. Também há objetos para a geração de sinais, que são mostrados após cliques (ou outros eventos pré-definidos) em locais determinados. Esse sinais são úteis para a associação de comentários em mapas, como notas sobre algum monumento ou explicações sobre cruzamentos confusos.

Vale lembrar que a API do Google Maps ainda está em estágio beta de desenvolvimento – ou seja, seu funcionamento está sujeito a mudanças repentinas. É possível monitorar a



**Figura 5:** Um caminho de 0,77 km desenhado no mapa. Podemos adicionar e remover as marcas de balões vermelhos da maneira que quisermos para desenhar outros caminhos.

evolução dessas mudanças, os novos recursos e a troca de experiências entre seus usuários no grupo de discussão do Google Maps [7]. Caso desenvolva algum aplicativo baseado nessa API, não se esqueça de anunciá-lo ao grupo para que todos possamos tê-lo como possível fonte de inspiração. ■

## INFORMAÇÕES

- [1] ECMA-262: [www.ecma-international.org/publications/standards/Ecma-262.htm](http://www.ecma-international.org/publications/standards/Ecma-262.htm)
- [2] Termos de uso: [www.google.com/apis/maps/terms.html](http://www.google.com/apis/maps/terms.html)
- [3] API do Google Maps: [www.google.com/apis/maps/](http://www.google.com/apis/maps/)
- [4] Documentação da API: [www.google.com/apis/maps/documentation/](http://www.google.com/apis/maps/documentation/)
- [5] Latitudes e longitudes: [www.meridianworlddata.com/Distance-Calculation.asp](http://www.meridianworlddata.com/Distance-Calculation.asp)
- [6] Ajax: [en.wikipedia.org/wiki/AJAX](http://en.wikipedia.org/wiki/AJAX)
- [7] Grupo de discussão do Google Maps: [groups-beta.google.com/group/Google-Maps-API](http://groups-beta.google.com/group/Google-Maps-API)
- [8] Monumentos em Paris: [www.kahunablog.de/gmaps.php?map=paris](http://www.kahunablog.de/gmaps.php?map=paris)
- [9] WikiMap: [www.wikiyblog.com/Map/Guest/Home](http://www.wikiyblog.com/Map/Guest/Home)
- [10] Tráfego na Inglaterra: [www.gtraffic.info](http://www.gtraffic.info)