

Desenvolvendo um *Mashup* com a API do *Google Maps* e com os dados de uma imobiliária

Luiz Henrique Rocha e Costa

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Av. Colombo, 5.790 - zona 07 - Maringá - PR - Brasil - CEP 87020-900

riquecosta@gmail.com

Resumo: *Os Mashups vêm se destacando cada vez mais na cultura da WEB 2.0 como uma boa opção para combinar dados e conseguir um novo serviço por meio de outros já existentes. Isso ocorre principalmente porque os geradores de serviços e conteúdos disponibilizam diversas APIs para os desenvolvedores. Este trabalho propõe um Mashup que combina dados da API do Google Maps e de uma imobiliária, formando, assim, um novo serviço por meio de outros já existentes.*

Abstract: *Mashups are standing out more and more in the WEB 2.0 culture as a good way to aggregate data and get a new service by means of others that already exist. This mainly happens because content generators enable several APIs to the programmers. This article proposes a mashup that mixes data from Google Maps API and a real state office, making a new service through others that have already exist from.*

1. Introdução

Hoje a *Web* oferece uma boa estrutura para que os usuários se sintam seguros e confiantes em realizar suas transações *online*. Muitas empresas estão disponibilizando e criando novos serviços na rede para satisfazer o usuário ou ter um diferencial perante os seus concorrentes. A agregação de conteúdo na *Web* vem se tornando cada vez mais freqüente e necessária, essa tecnologia é chamada de *Mashup* e usa código de terceiros por meio de uma interface pública chamada de *API*. Os *Mashups* também podem usar como fonte de conteúdo *Web feeds* (*RSS* ou *Atom*), documentos com padrões *XML* e também do tipo *JSON*. Existem na *Web* diversas *APIs* de uso muito simples para que se possa desenvolver novos serviços combinando dados. Alguns *websites* já disponibilizam essas interfaces e os gigantes da informática como *Microsoft*, *Google*, *Yahoo* entre outros já contam com diversas *APIs* para os desenvolvedores.

O *Google Maps* é uma ferramenta para visualização de mapas na *Web*, que permite uma navegação sob demanda, tornado-o assim muito rápido. O modelo leve de programação da *Google* levou à criação de numerosos serviços de valor agregado na forma de *Mashups* que ligam o *Google Maps* a outras fontes de dados acessíveis via *Web* (O'Reilly, 2005).

Um problema comum entre as imobiliárias que oferecem o serviço de locação e venda de imóvel pela *Web* é que o usuário nem sempre consegue identificar com clareza

o endereço físico do imóvel. Neste artigo vamos explorar a *API* do *Google Maps* para que possamos marcar os imóveis de uma imobiliária em um mapa, criando assim um serviço que depende de duas fontes de dados: o *Google Maps* e uma base de dados de uma imobiliária. A *API* do *Google Maps* permite usar *JavaScript* para incorporar o *Google Maps* em uma página da *web*. Esta *API* fornece diversos utilitários para manipular mapas (como indicado na página <http://maps.google.com>) e adicionar conteúdo ao mapa por meio de diversos serviços (Google, 2009).

Desta forma, este trabalho tem o objetivo de criar um novo serviço que permita que o usuário tenha uma noção mais clara do endereço do imóvel que está procurando, sendo possível ao usuário pesquisar o imóvel direto no mapa ou consultar onde o imóvel está localizado no mapa. Assim, propomos uma ferramenta que automatiza o processo de personalização do *Google Maps*, utilizando-se do formato *XML* (por parte da imobiliária) e *JSON* (pelo *Google Maps*).

Este artigo está organizado da seguinte maneira: na Seção 2 são apresentados os tipos de fontes de dados que podem compor um *Mashup*. Na Seção 3 é apresentada a *API* do *Google Maps*. Na Seção 4 é apresentado um exemplo da aplicação. Finalmente, na Seção 5 são feitas as considerações finais sobre o trabalho.

2. Formatos de dados

A Figura 1 mostra como os dados podem ser ligados a um *Mashup* para que se possa ter um melhor entendimento sobre a sua arquitetura.

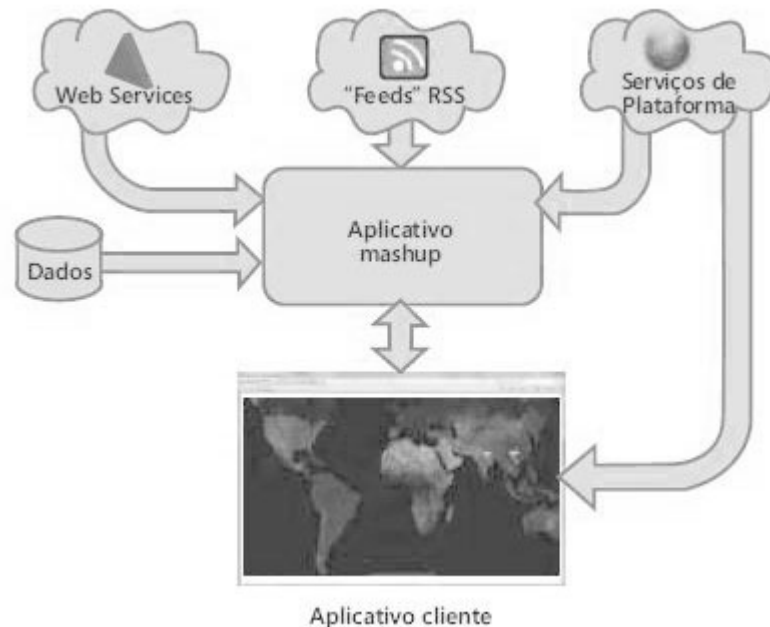


Figura 1: Arquitetura típica de um Mashup

O principal elemento de qualquer *Mashup* sem dúvida são os dados agregados e apresentados ao usuário. Embora o diagrama acima descreva os dados como um banco de dados, o conceito de *Mashup* não exige esse recurso no local, nem para o software

Mashup, nem para o cliente. Os dados podem vir dos *WebServices* nos quais são serializados geralmente para *XML* ou *JSON*.

2.1 Feeds RSS

Muitos *Mashups* se aproveitam de *feeds RSS* (*RDF Site Summary* ou *Really Simple Syndication*) como uma fonte de dados suplementar. Este formato está bem documentado e entendido com poucas variações de uma versão para a outra. Os *feeds* são muito utilizados possibilitando que um usuário da *Web* possa acompanhar novos artigos e notícias de *blogs* ou portais de conteúdo sem que ele precise visitar o *website* em si. Atualmente, a maioria dos portais de conteúdo oferece essa tecnologia. Apesar de os *feeds* usarem o formato *XML*, os *browsers* mais atuais já estão preparados para essa tecnologia e todos disponibilizam um leitor de *RSS* nativo.

2.2 XML

O *XML* (*eXtensible Markup Language*) é uma sigla que representa uma linguagem de marcação extensível. No *XML* o conjunto de *tags* a ser utilizado é ilimitado, pois o próprio usuário cria as suas *tags* e suas próprias linguagens de marcação (Ramalho e Henriques, 2003).

Pelo fato de o *XML* ter tanta flexibilidade tornou-o o principal formato de comunicação entre as fontes de dados de um *Mashup*. Documentos *XML* podem ser validados contra estruturas específicas. Essas estruturas devem prever quais elementos são encontrados nos documentos, a ordem em que estes elementos podem aparecer, a hierarquização destes elementos, o tipo de dados do conteúdo destes elementos, entre outros.

De certa forma, o *XML* acrescenta um padrão e uma semântica aos dados, por meio do uso de *tags* definidas pelo usuário. Desta forma, a busca de informações baseada nas *tags* dos documentos pode trazer resultados bem mais relevantes para a requisição do usuário. O acesso eficiente às informações pode ser facilitado quando se tem uma estrutura que descreve estas informações. A Figura 2 abaixo descreve como é um arquivo no formato *XML*.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <dadoscliente>
3    <nome>Mary Lebow</nome>
4    <endereço>
5      <rua>5 Main Street</rua>
6      <cidade zip="91912"> San Diego, CA </cidade>
7      <telefones>
8        <tel>619 332-3452</tel>
9        <tel>664 223-4667</tel>
10     </telefones>
11   </endereço>
12 </dadoscliente>

```

Figura 2: Exemplo de um arquivo no formato *XML* [Daniel Rubio-02/2007]

2.3 *WebService*

É muito comum um *Mashup* fazer chamadas a *WebServices*. Esses *WebServices* podem ser baseados em *WSDL* ou em *REST*. De acordo com a [W3C-2001] o *WSDL* (*Web Services Description Language*) é um *XML* formatado para descrever uma rede de serviços. As operações e mensagens são descritas abstratamente e uma mensagem pode definir um ponto final. Basicamente podemos dizer que o *WSDL* é a troca de informações por requisições *HTTP* do tipo *GET/POST*. Seu principal objetivo é descrever as interfaces apresentadas e apontar a localização dos serviços disponíveis em um local previsível e bem conhecido na rede, o qual permite que o cliente o acesse de maneira confiável. Por ser um documento *XML*, sua leitura se torna fácil e acessível (Reckziege, 2006).

A *REST* (*Representational State Transfer*) é vista como uma imagem do design da aplicação: uma rede de *web sites*, na qual o usuário progride com uma aplicação selecionando as ligações (transições do estado), tendo como resultado a página seguinte (que representa o estado seguinte da aplicação), a qual está sendo transferida ao usuário e apresentada para seu uso (Fielding – 2000). Na atualidade o termo *REST* pode descrever qualquer interface na *Web* que utilize *XML* e *HTTP*. Também é possível desenhar interfaces *XML/HTTP* que fazem requisições a um servidor por meio do protocolo *HTTP* e os retorna serializados no formato *XML* sem a necessidade de um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída (como o *SOAP* — *Simple Object Access Protocol*). O *REST* nada mais é do que uma página programável da *Web* que recebe dados por *HTTP GET* ou *HTTP POST*, os processa e os devolve em um arquivo do tipo *XML* ou *JSON*. Como podemos ver, a diferença básica entre o *WSDL* e o *REST* é que no *WSDL* as mensagens são descritas abstratamente e uma mensagem pode definir o ponto final, mas ambos têm em comum o fato de poderem ser acessados por comandos *GET* ou *POST* do *HTTP*.

Neste trabalho, optamos pela tecnologia *REST* para que possamos escrever um arquivo *XML* com os dados dos imóveis de uma imobiliária.

2.4 *JSON*

Atualmente existem várias alternativas ao *XML* no mercado, porém muitas delas não têm a massa crítica necessária para se tornarem conhecidas. A alternativa que mais vem se destacando neste cenário é o *JSON* (*JavaScript Object Notation*), o qual é um formato de texto para a serialização de dados estruturados. Ele pode representar quatro tipos primários (*strings*, números, booleanos e nulos) e dois tipos estruturados (objetos e vetores). Um objeto é uma coleção não ordenada de zero ou mais pares nome/valor, onde o nome é uma *string* e o valor é uma *string*, número, booleano, nulo, objeto ou vetor. Na Figura 3 podemos ver um exemplo dos dados da Figura 2 descritos no formato *JSON*.

```

1  {"dadoscliente": {"nome": "Mary Lebow",
2      "endereco": {
3          "rua": "5 Main Street",
4          "cidade": "San Diego, CA",
5          "cep": "91912",
6      },
7      "telefones": [
8          "619 332-3452",
9          "664 223-4667"
10     ]
11 }
12 }

```

Figura 3: [Rubio-2007] Arquivo de dados no formato *JSON*.

Supondo que o código acima esteja armazenado na variável "cliente", esta variável se tornaria um objeto, logo para acessar seus atributos pela linguagem *JavaScript* usariamos a seguinte notação: **cliente.nome**. Por isso o *JSON* leva uma boa vantagem sobre o formato *XML*, apesar de ser menos popular (Rubio, 2007).

3. API do Google Maps

A *API* do *Google Maps* permite usar *JavaScript* para incorporar o *Google Maps* em uma página da *web* (Google, 2009). Ela fornece diversos utilitários para manipular mapas (como indicado na página <http://maps.google.com>) e adicionar conteúdo ao mapa por meio de diversos serviços. Ela é um serviço gratuito em fase *beta* de teste e disponível para qualquer *website* que seja gratuito para os consumidores.

O *Google Maps* possui diversas funcionalidades como: navegação pelo mapa, *zoom*, traçado de rotas, medida de distâncias, realização de marcações, balões de informação, janelas de informação e etc. Além disso, também é possível alternar entre as visões de mapa, satélite e terreno. Todas essas funcionalidades estão disponíveis na sua *API*. Veja na Figura 4 um exemplo do *Google Maps*, com o mapa posicionado sobre o Brasil:



Figura 4: *Google Maps* posicionado sobre o Brasil.

A visualização do *Google Maps* é muito rápida, pois ele usa a tecnologia *tiles* (azulejos) para poder montar o mapa sob demanda assincronamente na página *web*. Dependendo da movimentação do mapa pelo usuário esses *tiles* são recarregados sem que a página *web* seja recarregada. Para que o usuário possa acessar a *API* do *Google Maps* é preciso que ele primeiro faça um cadastro e receba uma **chave de acesso** para a *API*. Sem a chave de acesso é impossível utilizar os recursos da *API*. Também se faz necessário um domínio na *Web*. Com a chave da *API* em mãos é necessário inserir o código da *API* junto com a sua chave em uma página *HTML*, que comporá um *Mashup* propriamente dito. A Figura 5 mostra como isso é feito. Nela podemos ver a **chave** que deve ser a sua chave da *API*.

```

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     <meta http-equiv="content-type" content="text/html; charset=iso-8859-1"/>
5     <title>Busca por imóveis no mapa</title>
6     <script src="http://maps.google.com/maps?file=api&v=2&key=chave" type="text/javascript">
7   </script>
8   <script type="text/javascript">

```

Figura 5: Incluindo a *API* na página de internet.

Com a *API* embutida no código *HTML* da página, por meio de *JavaScript* é possível fazer chamadas a diversos métodos. Todos os métodos, objetos, controles e eventos estão devidamente documentados no próprio *website* da *API* (veja em <http://code.google.com/intl/pt-BR/apis/maps/>).

3.1 Criando um mapa

Para criar um mapa por meio da *API* é preciso criar uma função de inicialização, a qual será chamada quando todos os objetos de modelo de documento (*DOM* — *Document Object Model*) forem carregados. O *DOM* é uma especificação da W3C, independente de plataforma e linguagem, o qual permite alterar e editar a estrutura, conteúdo e estilo de um documento eletrônico dinamicamente. Sendo assim, não se corre o risco de fazer referência a um objeto sem que o mesmo ainda não tenha sido carregado. Esta função pode ser escrita na própria página *HTML* ou pode ser escrita em um arquivo do tipo *JavaScript* e ser invocado na página *HTML*. O *HTML* é somente uma linguagem de marcação, por isso a necessidade da linguagem *JavaScript*. Em seguida basta criar e instanciar o objeto que representa o mapa e adicionar as funcionalidades que forem necessárias. Por último, é necessário fazer a chamada da função. Para criar um mapa simples é necessário fazer a chamada da *API* juntamente com sua *API Key* (como mostrado na Figura 6), usar uma função da própria *API* que verifica a compatibilidade do *browser*, em seguida criar um objeto do tipo Mapa e também indicar onde o mapa deve ser renderizado (*i.e.* em qual elemento *DOM*).

Na Figura 7 é possível ver a instanciação da variável “map” a qual está recebendo um objeto do tipo “GMap2” que deve ser renderizado no elemento “map_canvas”. Após isso, deve-se definir qual o centro do mapa, inserindo as coordenadas de latitude e longitude.

```

1 function initialize() { //Função de Inicialização
2   if (GBrowserIsCompatible()) { //Verifica se o browser é compatível (Boa prática)
3     var map = new GMap2(document.getElementById("map_canvas")); //Setando onde o mapa deverá ser exibido
4     map.setCenter(new GLatLng(37.4419, -122.1419), 13); //Setando o centro do mapa
5   }
6 }

```

Figura 6: Exemplo de criação de um mapa simples.

4. Exemplo da aplicação desenvolvida

Com o objetivo de simplificar o uso de mapas para divulgar informações de um imóvel, foi desenvolvido um *Mashup* que cria um mapa com recursos visuais e que pode ser útil não só para os usuários mas também para as empresas.

Como dito anteriormente, uma fonte de dados adotada no *Mashup* será a *API* do *Google Maps*, a qual se utiliza do formato *JSON* para troca de dados. A outra fonte de dados a ser utilizada será uma base de dados de uma imobiliária. Neste caso específico, estamos levando em consideração que a imobiliária tenha disponível em um arquivo no formato *XML* os imóveis que deverão ser marcados no mapa, juntamente com suas descrições e seus *Links* para que o usuário possa ver mais detalhes do imóvel. Neste exemplo, estaremos emulando um arquivo como se fosse uma imobiliária da cidade de Tupã-SP. No exemplo vamos colocar os atributos para cada imóvel, como endereço, título, e um *Link*. Este arquivo *XML* deverá ter alguns atributos específicos para que o *Mashup* possa interpretá-lo. A Figura 7 mostra como o arquivo *XML* deve ser formatado.

```

1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <markers>
3   <marker endereco="Rua Joisé Maria Camarinho 540 - Tupã - SP" titulo="Casa pronta para morar, aluga!" link="http://"/>
4   <marker endereco="Rua Chavantes 458 Tupã - SP" titulo="Ótima localização, vende!" link="http://"/>
5   <marker endereco="Rua Guaranis 104 Tupã - SP" titulo="Terreno todo murado/" link="http://"/>
6   <marker endereco="Alameda Cardiff 600 Tupã - SP" titulo="Oportunidade de investimento" link="http://"/>
7   <marker endereco="Rua Cherentes 988 Tupã - SP" titulo="Casa com piscina, vende" link="http://"/>
8   <marker endereco="Rua Aimorés 331 Tupã - SP" titulo="Edicula" link="http://"/>
9   <marker endereco="Rua Paiaquás 1115 Tupã - SP" titulo="Ótima localização, vende!" link="http://"/>
10  <marker endereco="Rua Piratinins 250 Tupã - SP" titulo="Oferta!!" link="http://"/>
11  <marker endereco="Rua Tabajaras 800 Tupã - SP" titulo="Ótima localização, vende!" link="http://"/>
12 </markers>

```

Figura 7: Formato do arquivo XML fornecido pela imobiliária.

O arquivo *XML*, acima descrito, deverá estar no servidor do *website* da imobiliária. No exemplo prático este arquivo *XML* estará no domínio *aflsistemas.com.br* com o nome de “*arquivo.xml*”. A aplicação *Mashup* estará no domínio *planh.com.br* com o nome de “*imóvel.html*” e por fim a *API* do *Google Maps* estará no próprio servidor do *Google*.

De posse das duas fontes de dados, é preciso fazer com que a aplicação *Mashup*, consiga fazer a “mistura” entre as duas fontes de dados, de modo que isso fique representado ao usuário de uma forma simples, acessível e confortável. Uma dificuldade encontrada na criação deste *Mashup* foi o fato de que a *API* do *Google Maps* não possibilita a leitura de um arquivo *XML* externo ao domínio que o *Mashup* está hospedado (por questões de segurança). Logo isto tornaria inviável este trabalho, já que o objetivo é misturar dados de domínios diferentes.

Para resolver este problema foi desenvolvido na linguagem *PHP* (*Personal Home Page*) um *script* que é responsável por ler os dados de um documento *XML* de outro domínio e em sua saída de dados devolver todo o conteúdo do arquivo lido, além de alterar o seu próprio *header* para *XML*, já que o arquivo se encontra na extensão *PHP*. Este processo segue o mesmo conceito de um *Proxy* e neste caso serve para emular um arquivo externo como se estivesse no mesmo servidor da aplicação *Mashup*.

Outra dificuldade encontrada no projeto foi o fato que a grande maioria das imobiliárias não cadastram em suas bases de dados a latitude e a longitude do imóvel, até mesmo por uma questão cultural, pois esses dados eram tidos como sem importância até pouco tempo atrás. Este fato influenciou muito no desenvolvimento deste projeto, tendo em vista que para se obter uma localização exata no *Google Maps* são necessários esses dois pontos.

Para descobrir essas coordenadas é necessário utilizar-se de um método que a *API* do *Google Maps* oferece que é o *GeoCoder*. Este método recebe um endereço do imóvel em questão e tenta transformá-lo em coordenadas geográficas de latitude e longitude. O problema deste método é que ele nem sempre retorna a posição exata de um endereço, mas na maioria dos casos o resultado obtido é satisfatório. Em alguns pontos do mapa esse método funciona com extrema eficácia, pode-se dizer que os pontos mais “populares” como as capitais do Brasil já estão totalmente compatíveis com o método. Como recomendação, acreditamos que as imobiliárias deveriam começar a se acostumar com a idéia de cadastrar esses tipos de dados, para estarem melhores preparadas para as tecnologias emergentes.

A aplicação desenvolvida funciona da seguinte maneira: primeiro se deve carregar a *API* do *Google Maps* com a chave obtida por meio do cadastro. Em seguida a aplicação deve ler os dados de um documento *XML*, que foi gerado pelo *script PHP* emulando um arquivo *XML* local que faz uma requisição para outro domínio, neste caso o domínio *aflsistemas.com.br*. Após isso, se faz necessário buscar por meio da *API* do *Google Maps* os pontos geográficos de latitude e longitude para o endereço de cada imóvel. De posse desses pontos a aplicação se utiliza mais uma vez da *API* para marcar no mapa os imóveis. Por último, é adicionado um evento no mapa para que, ao usuário clicar no ponto marcado, seja mostrado as informações do imóvel correspondente a aquele ponto. A Figura 8 ilustra o resultado obtido:

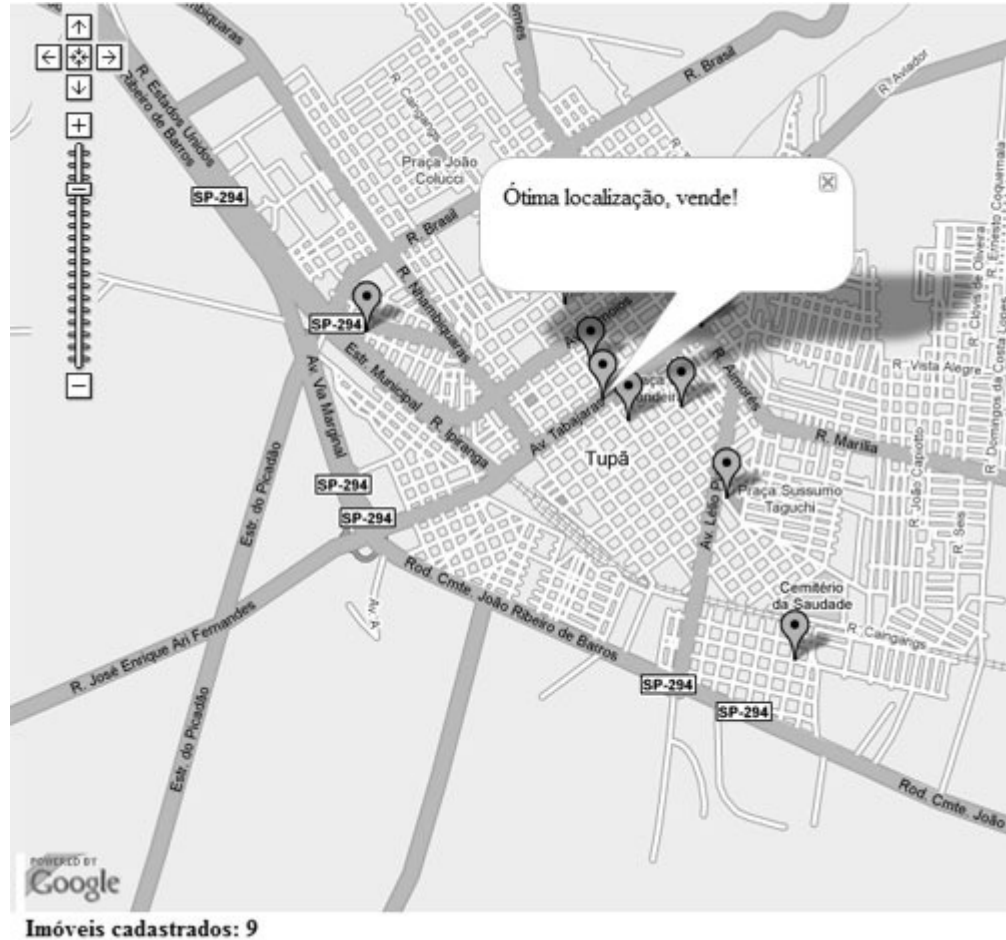


Figura 8: Aplicação desenvolvida.

5. Considerações Finais

A aplicação desenvolvida mostra-se útil por indicar aos usuários onde existem imóveis disponíveis para locação ou venda diretamente no mapa da cidade, e também qual imobiliária oferece seus serviços, sem que o usuário necessite sair de casa para fazer uma apuração mais detalhada da localização de um imóvel. Este fato é extremamente importante, principalmente do ponto de vista da usabilidade, já que gera um conforto muito grande ao usuário que não precisa manter dois *websites* abertos (Imobiliária e *Google Maps*) para que possa ter uma noção da localização. Essa é uma das vantagens da utilização do conceito de *Mashup* que tem como um ponto forte oferecer maior conforto ao usuário.

Para que se faça uma localização exata do imóvel, é possível implementar um método onde o usuário possa marcar no próprio mapa do *Google Maps*, usando a *API*, o local no qual o imóvel se encontra exatamente, cadastrando os pontos geográficos de latitude e longitude com exatidão. Além disso, também se mostra como uma ótima escolha para empresas que buscam oferecer um serviço diferenciado, inovador e com muito conforto ao usuário.

Este *Mashup* também pode compor um *website* próprio e unir diversas imobiliárias num único *website*, trazendo benefícios ainda maiores para o usuário, já que ele não precisará necessariamente entrar em diversos *webites* de imobiliárias para procurar um imóvel. Esta aplicação já está preparada para que ela possa ler diversas fontes de conteúdo (Imobiliárias), tornando muito prática e de simples implementações.

Como exemplo de outros *Mashups* que poderiam ser implementados pode-se citar um *Mashup* para marcar no mapa os postos de gasolina de uma cidade com seus respectivos preços de combustível, tendo como fonte de dados o site da ANP (Agência Nacional do Petróleo) além da *API* do *Google Maps*, sendo assim este *Mashup* forneceria para os usuários que pudessem pesquisar por preços de combustíveis dos postos que estão mais próximos de sua residência.

Os *Mashups* também podem ser combinados com outros tipos de fontes de dados, como um *Mashup* que possa unir os dados de um *website* de relacionamentos (e.g: *Orkut*) com o *Google Calendar*. Neste caso é possível marcar no *Google Calendar* todas as datas de aniversário dos seus amigos na sua rede social, podendo ser enviados avisos dos eventos como email, alerta na página do *Google Calendar* e até mensagens para o celular do tipo SMS.

Para um próximo projeto, está sendo estudada a possibilidade de montar um *Mashup* que possa unir os dados dos Boletins de Ocorrência de uma ou de várias delegacias de um município, juntamente com o *Google Maps*. Este *Mashup* se torna interessante do ponto de vista policial, pois serviria para mapear as zonas mais violentas de município, na qual ocorrem mais roubos, mais mortes, mais acidentes e etc. Se tornando muito mais fácil, rápido e confortável visualizar este tipo de informação. Este *Mashup* também poderia trazer benefícios para diversos outros setores como: Comércio, Imobiliárias, Governo e pessoas físicas.

Portanto, pode-se dizer que os *Mashups* podem ajudar muitos usuários e muitas empresas, e que com a crescente liberação de *APIs* por parte dos formadores de conteúdos teremos muito mais *Mashups* no futuro e com muito mais qualidade.

Referências Bibliográficas

- O'Reilly, T. (2005) *Padrões de design e modelos de negócios para a nova geração de software*, <http://www.oreilly.com/>, Último acesso em 10/05/2009
- Google Inc (2009) *Api do Google Maps*, <http://code.google.com/intl/pt-BR/apis/maps/>, Último acesso em 10/05/2009
- W3c, (2001) *Web Services Description Language*, <http://www.w3.org/TR/wsdl>, Último acesso em 10/05/2009
- Reckziegel, M. (2006) *Descrevendo um Web Service - WSDL* http://imasters.uol.com.br/artigo/4422/webservices/descrevendo_um_web_servi_ce_wsdl/, Último acesso em 10/05/2009
- Ramalho, J.C.L. e Henriques, P.R. (2001) *XML & XSL da Teoria à Prática*, FCA Editora de Informática

Rubio, D. (2007) *An Introduction to JSON*, <http://www.oracle.com/technology/pub/articles/dev2arch/2007/02/introduction-json.html>, Último acesso em 10/05/2009.