UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO LATO SENSU
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO WEB

HEBER MASSAMI NUNOMURA

WEB INTENTS: descoberta de serviços pelo lado cliente e modelo de execução

MONOGRAFIA DE ESPECIALIZAÇÃO

LONDRINA
2014

HEBER MASSAMI NUNOMURA

WEB INTENTS: descoberta de serviços pelo lado cliente e modelo de execução

Monografia de especialização apresentada no Câmpus Londrina da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de “Especialista em Desenvolvimento Web”.

Orientador: Prof. Msc. Thiago Prado de Campos

LONDRINA
2013

TERMO DE APROVAÇÃO

Título da Monografia

WEB INTENTS: DESCOBERTA DE SERVIÇOS PELO LADO CLIENTE E MODELO DE EXECUÇÃO

por

HEBER MASSAMI NUNOMURA

Esta monografia foi apresentada às 14h00 do dia **07** de **fevereiro** de **2014** como requisito parcial para a obtenção do título de ESPECIALISTA EM DESENVOLVIMENTO WEB. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho _____. (aprovado, aprovado com restrições ou reprovado)

_____
Prof. Msc. André dos Santos Domingues
(UTFPR)

_____
Prof. Esp. André Frederico Lucas da Silva
(UTFPR)

_____
Prof. Me. Thiago Prado de Campos
(UTFPR)

Visto da coordenação:

_____
Prof. Me. Thiago Prado de Campos
Coordenador da esp. em Desenvolvimento Web

_____
Prof. Dr. Walmir Eno Pottker
Coordenador de Pós-Graduação Lato Senso

A Folha de Aprovação preenchida e assinada encontra-se na Coordenação do Curso

Aos meus queridos pais e irmã
Dedico esta monografia

AGRADECIMENTO

À meu orientador Thiago Prado de Campos, por seu grande apoio e dedicação sempre me oferecido.

RESUMO

NUNOMURA, Heber. ***Web* Intents**: **descoberta de serviços pelo lado cliente e modelo de execução**. 2013. 42 fls. de Conclusão de Curso (Curso de Especialização em Desenvolvimento *Web*), Diretoria de Pesquisa e Pós-Graduação, Universidade Tecnológica Federal do Paraná. Londrina, 2013.

Nos últimos anos temos visto nas páginas *Web* a proliferação dos botões de compartilhamento como “like”, “tweet”, “g+”, dentre outros. Estamos acompanhando, talvez, o início de uma nova tecnologia que permitirá a integração entre as páginas *Web* e provedores de serviço. Existe um problema na forma como é feita a integração entre as páginas *Web* e os provedores de serviço. Atualmente, para se utilizar os botões de compartilhamento de um provedor de serviço é necessário usar o código de programação específico deste mesmo provedor. Em programação, existe a recomendação que devemos manter o código com baixo acoplamento e alta coesão. O fato de dependermos de código externo evidencia o aumento do acoplamento no código fonte. *Web Intents* e outras tecnologias similares tentam diminuir essa dependência fornecendo uma interface padrão que poderá ser usada por todos os provedores de serviço. O presente trabalho apresenta o *Web Intents* e tecnologias similares que tratam da interação entre páginas *Web* e provedores de serviço. Também apresenta e discute as visões diferentes que pessoas e empresas envolvidas em soluções deste tipo têm sobre como a tecnologia deve funcionar.

Palavras chave: *Web Intents*. API de integração. Descoberta de serviços *Web*.

ABSTRACT

NUNOMURA , Heber . Web Intents : the future of Web development. 2013. 42 pages. End of Course (Web Development Specialization Course), Research and Graduate Studies Board, Federal Technological University of Paraná. Londrina , 2013.

Over the last years, we have seen the proliferation of sharing buttons in Web pages, such as "like", "tweet", "g+", among others. We could be watching the beginning of a new technology that will allow integration between Web pages and service providers. There is an issue concerning how the integration between Web pages and service providers is done. Nowadays, in order to use the sharing buttons of a service provider, you need to use the code from this particular service provider. In programming, there is a recommendation that we should keep the code with low coupling and high cohesion. The fact that we rely on external code demonstrates an increase in coupling the source code. *Web Intents* and other similar technologies try to reduce this dependency by providing a standard interface that can be used by all service providers. This paper presents the *Web Intents* and similar technologies that address the interaction between Web pages and service providers. This also presents different views and discusses different views of people and companies involved in the creation of this standard on how this technology should work.

Keywords: Web Intents. Integration API. Web service discovery

LISTA DE ILUSTRAÇÔES

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

API.....Application Programming Interface
OWL-S…….. Semantic Markup for Web Services
REST……. Representational State Transfer
RESTful…...Termo usado para se referir a aplicações Web que implementam REST
SOA…….....Service-Oriented Architecture
SOAP…….Simple Object Access Protocol
TCP.....Transmission Control Protocol
TI.....Tecnologia da Informação
UDDI.....Universal Description, Discovery and Integration
UI……..……..User Interface
WADL……..…….Web Application Description Language
WCP…….…..Windows Communication Foundation
WSDL…….…… Web Services Description Language

SUMÁRIO

Conteúdo

1 INTRODUÇÃO

A *World Wide Web* (www) oferece uma infinidade de serviços para ajudar usuários em suas tarefas diárias. Algumas dessas tarefas incluem edição de fotos, compartilhamento de arquivos, adição de eventos em calendários pessoais, etc. Na maioria das vezes, os desenvolvedores estão sobrecarregados com a decisão de selecionar um subconjunto adequado de serviços para integrar com suas aplicações *Web*. Infelizmente, o modo de programação atual das aplicações *Web* não permite deixar escolhas em aberto para o usuário e também não permite que aplicações possam ser usadas de forma independente e desacopladas do provedor de serviços.

O *Web Intents* foi proposto para resolver isso, facilitando interações entre aplicações *Web* e provedores de serviço. Atualmente as soluções de colaboração da *Web* impõem um custo (tempo e estudo) não trivial para cada API suportada, enquanto que o *Web Intents* permite aos desenvolvedores consolidar custos em uma única API. As alterações posteriores, como adição ou remoção de provedores de serviço suportados não impacta no custo de manutenção de uma página da *Web* com *Web Intents*, pois a interface de um serviço poderá ser usada por outro serviço sem nenhuma alteração.

Web Intents é um framework que funciona do lado cliente que permite o compartilhamento de recursos e comunicação entre diferentes aplicações *Web*. Para utilizar este framework, o provedor de serviço precisa associar cada recursos com as ações especificadas no *Web Intents* (por ex. compartilhar); o usuário por sua vez, evoca uma ação e escolhe o provedor de serviço de sua preferência.

Neste trabalho apresentamos como funciona o *Web Intents* e fazemos uma comparação com outras soluções similares que tratam da comunicação entre aplicativos *Web*. No capítulo 2 é apresentado conceito sobre Serviços *Web* e cenários onde o *Web Intents* poderia ser útil. No capítulo 3 é descrito o *Framework Web Intents*, suas características, especificações e funcionamento. No capítulo 4 é descrito algumas opções alternativas ao *Web Intents*. No capítulo 5 é feito uma análise das tecnologias apresentadas juntamente com as considerações finais.

2 CONCEITOS SOBRE SERVIÇOS E CENÁRIOS

O *Web Intents* foi projetado para facilitar a comunicação entre a aplicação e os serviços *Web* disponíveis. Os desenvolvedores que criam aplicativos *Web* poderiam facilmente incluir funcionalidades de outros serviços da *Web*, sem precisar investir tempo e recurso para codificar a integração com outros serviços *Web*. Considerando que o principal objetivo do *Web Intents* é a integração com os serviços *Web* e a descoberta de serviços, segue uma breve explanação sobre o tema.

2.1 SERVIÇOS

O conceito de serviço tem sido amplamente discutido, mas não existe um acordo universal já que sua definição pode ser expressa de forma diferente em cada contexto. Em *Service-Oriented Architecture* (SOA), serviços são definidos como funcionalidades de negócio que são construídos como componentes de software que podem ser reusados para diferentes propósitos (BELL, 2008).

Na documentação do OWL-S (*Semantic Markup for Web Services*) (W3C, 2004), serviços significam página da *Web* que permitem ao usuário obter ações e mudar o estado anterior. Em sistema operacionais, como Windows ou Android, um serviço é um componente de uma aplicação que executa operações de longo prazo, em segundo plano (*background*) e sem mostrar a interface da aplicação para o usuário (GUIA DO DESENVOLVEDOR ANDROID, 2013).

Mesmo com essa variedade de definições, uma característica fundamental dos serviços é que eles fornecem algumas funções e facilitam o usuário na realização de tarefas. A computação como serviço, mesmo depois de muitos anos de desenvolvimento, ainda é um campo emergente, promissor e amplo, atraindo interesse de pesquisadores e desenvolvedores de várias áreas. Abrange tecnologias como *Web* Services, SOA, *Cloud Computing*, consultoria em metodologia de negócios e utilidades, modelagem de processos de negócios, transformação e integração. Embora parte dessas tecnologias tenha sido aplicada com sucesso, a maior parte ainda não é madura o suficiente e são necessários grandes esforços em pesquisa.

2.2 SERVIÇOS *WEB*

Serviços *Web* ou *Web services* são serviços que se comunicam uns com os outros através da *Web*. Ao contrário de serviços de *desktop*, um serviço *Web* pode

ser publicado *online* utilizando-se linguagens de descrição de interface padrão e estar disponível através de protocolos uniformes. (*19TH INTERNATIONAL CONFERENCE ON WEB SERVICES*, 2012) Em tal padrão e protocolo incluem-se as linguagens *Web Application Description Language* (WADL), a *Web Services Description Language* (WSDL) e a *Web Ontology Language of the Semantic Web* (OWL-S) para a descrição do serviço, o padrão *Universal Description*, *Discovery and Integration* (UDDI) para descoberta de serviços e o protocolo SOAP (*Simple Object Access Protocol*) para a transmissão de mensagens.

2.2.1 Tipos de *Web* services

Richardson (RICHARDSON; RUBY, 2007, p. 13) classifica o *Web* service atual em três categorias: RESTFul, RPC-style e REST-RPC.

RESTful *Web* services cumpre rigorosamente os princípios e restrições especificadas por Fielding (FIELDING, 2000, p. 116). A RESTful *Web* service utiliza completamente os métodos (GET, POST, PUT, DELETE) especificados no HTTP. Em outras palavras, a comunicação entre o RESTful *Web* service e seu cliente é construída sobre simples mensagens HTTP. Como exemplo, podemos citar aplicações desenvolvidas em Ruby on Rails (*Framework* para desenvolvimento *Web*). O Ruby on Rails utiliza como padrão para desenvolvimento o REST. Caso o desenvolvedor tenha seguido o padrão de desenvolvimento do Ruby on Rails, o aplicativo será RESTful, utilizando-se métodos e rotas do REST.

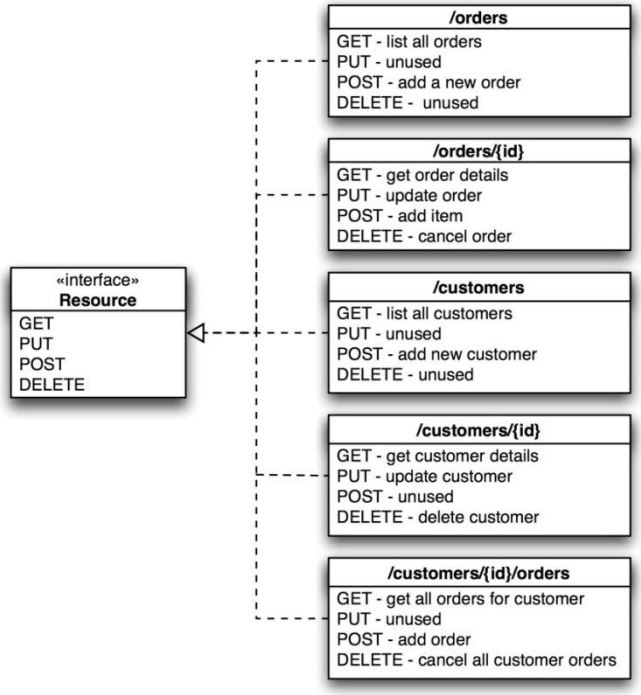

Figura 1 - Métodos e rotas do REST

RPC-style *Web* services, chamados de “Big *Web* services” por Richardson (RICHARDSON; RUBY, 2007, p. 299), podem definir métodos quanto necessário. Além disso, este serviço costuma adotar envelopes especiais ao invés de usar HTTP para transmissão de mensagens. Por exemplo, *Web* services baseados em SOAP pertencem a este domínio. Como exemplo, podemos citar o Windows Communication Foundation (WCF). O WCF é um conjunto de APIs usada para construir aplicações no framework .NET da Microsoft. O WCF inclui conexões predefinidas para os protocolos mais comuns como SOAP sobre HTTP, SOAP sobre TCP (*Transmission Control Protocol*), SOAP sobre *Message Queues*, etc. A interação entre o ponto final WCF e o cliente e feito usando o envelope SOAP (WIKIPEDIA.ORG).

REST-RPC é uma classe de arquitetura híbrida das arquiteturas RESTful e RPC. *Web* services dessa categoria geralmente usa mensagens HTTP como meio de comunicação, mas podem criar novos métodos além dos usados no HTTP. Esses novos métodos e seus parâmetros são freqüentemente incorporados dentro de URIs ou em outros campos da solicitação da mensagem HTTP (ZHENG; SHEN; GHNNIWA, 2013). Alguns dos exemplos mais conhecidos de serviços REST-RPC se incluem:

- A API do del.icio.us (https://delicious.com/)(BROWN, 2012)
- A API *Web* "REST" do Flickr (http://www.flickr.com)

A API do Flickr solicita aos cliente o uso do HTTP GET mesmo quando eles querem modificar os dados. Para excluir uma foto você faz uma solicitação GET para uma URI que inclui method=flickr.photos.delete. (RICHARDSON; RUBY, 2007)

Muitas aplicações Web são planejadas para usar o REST como sendo a arquitetura de desenvolvimento, mas durante a execução do modelo acabam se utilizando de chamadas para métodos RPC ou utilizam os métodos HTTP de forma incorreta. Essas aplicações podem ser consideradas como hibridas, pois não são REST nem RPC, mas uma mistura das duas arquiteturas.

2.2.2 Descoberta de Serviços

Como o número de serviços *Web* aumenta por causa do baixo custo dos recursos técnicos, o problema de localizar os serviços *Web* se torna proeminente, pois é necessário buscar os serviços de interesse em meio um grande conjunto de serviços disponíveis.

A maior parte mecanismos de descoberta de serviços usa o algoritmo convencional para encontrar compatibilidade com base em atributos. Esse mecanismo de busca fica aquém de capturar a semântica para o serviço de descoberta e geralmente não satisfaz ou satisfaz parcialmente a necessidade do usuário (LAMPARTER; SCHNIZLER, 2006, p. 1679-1683).

Devido ao mecanismo de busca ser deficiente, o serviço desejado muitas vezes não é encontrado e o usuário poderá ter que buscar termos diferentes de pesquisa para alcançar algum resultado. Assim, encontrar os serviços da *Web* apropriados de acordo com a necessidade do usuário é um desafio.

Um pré-requisito para a descoberta de serviços é a obtenção de uma coleção de *Web* services disponíveis. Uma maneira simples de buscar o serviço é a partir de registros de serviços públicos, como registro de negócios UDDI. No entanto, registros públicos UDDI estão desatualizados e perderam o apoio de gigantes da TI. Assim, nos próximos anos o crescimento do registro de *Web* services existirá principalmente fora do UDDI.

Outra maneira de obter informações de *Web* services disponíveis é usar uma ferramenta de busca, como o Google, e procurar por “<termo> filetype:wsdl” já que o Google indexa as definições do WSDL. Também é possível usar a página da *Web* xmethods (http://xmethods.org) especializada em rastreamento de serviços WSDL.

Catálogos centralizados de API como o ProgrammableWeb (http://programmableweb.com) permitem categorizar e documentar os serviços, assim como era nos primeiros dias da *World Wide Web* antes dos motores de busca. Ainda assim, a fim de descobrir os serviços, é necessária intervenção manual.

Apesar dos esforços, as deficiências existentes na modelagem atual dos serviços *Web* (*Web* Service) minam o futuro da descoberta de serviços e integração. Primeiro, porque a atual descrição das linguagens do *Web* service, por exemplo, o WSDL, usam nomes característicos de programação, o que é pouco didático para descrever um serviço, pois os nomes das operações vem da perspectiva dos profissionais da programação. Segundo porque, diferente de páginas HTML, as linguagem descritivas quase não tem construtores como no hipertexto que adiciona links através das páginas HTML para constituir redes. Essa qualidade da *Web* é amplamente utilizada pelos rastreadores (crawlers) para colecionar e indexar páginas *Web* (ZHENG; SHEN; GHNNIWA, 2013). E por último, mas não menos importante, o motivo que dificulta a descoberta de serviços e sua integração é que, nenhuma linguagem de descrição de serviço domina o mercado, o que significa que nenhum *Web service*, especialmente RESTfull *Web* services, tenham arquivos de descrição publicados online, e mesmo aqueles que tem arquivo de descrição, podem estar escritos em formatos diferentes o que coloca obstáculos na descoberta de serviços

2.3 CENÁRIO DO PROBLEMA

Nos últimos anos houve uma rápida proliferação dos botões nas páginas *Web* que permitem ao usuário realizar alguma ação relacionada à página da *Web* em que está no momento ou enviar dados de uma página da *Web* para outra ou para uma aplicação. Botões como “*Blog this*”, “*Digg*”, “*Read later*”, “*Follow*”, “*Like*”, “*Share*”, “*Tweet*”, “+1”. (ÇELIK, 2011)

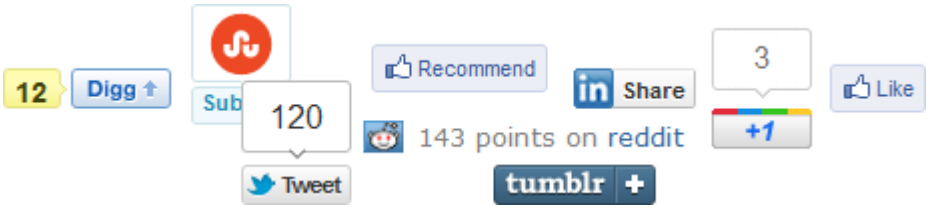

Figura 2 - Botões de compartilhamento

Em geral estes botões executam ações em algum serviço disponível na Web. Para incluir um botão e torná-lo funcional é necessário na maioria das vezes embutir scripts do site do serviço externo e programar uma chamada para suas APIs, mas isto causa alguns problemas:

2.3.1 Problemas de privacidade

Os botões de redes sociais usadas em muitas página da *Web* freqüentemente se utilizam de *cookies* de rastreamento, significando que o serviço da rede social são informados sobre o conteúdo que os usuários logados estão vendo, mesmo se nenhuma ação seja executada (*like*, *retweet*, +1). Dependendo da natureza do conteúdo da página da *Web*, pode haver uma violação de padrões éticos de privacidade. Por exemplo, informações de visitas a páginas da *Web* com conteúdo sexual, saúde ou políticos podem ser levar as empresas responsáveis pelos *cookies* a fazerem inferências indesejadas que violam a privacidade do usuário.

2.3.2 Problemas de acomplamento

Muitas aplicações como Twitter, Dropbox, Facebook tem suas próprias APIs e se você quiser compartilhar informações com essas aplicações ou fazer uso de seus serviços você precisará usar as APIs especificas de cada um.

O botão “*like*” do Facebook, por exemplo, faz uso de APIs específicos da rede social para compartilhar as informações. Se alguma interface dessa API mudar, uma nova versão do botão precisará ser criada.

2.3.3 Problema de excesso de opções ou da não existência do serviço

Existem atualmente centenas de serviços de compartilhamento de páginas na Internet. Os sites poderiam oferecer todas as opções a seus usuários, mas isso tornaria o visual do site poluído e retardaria o tempo de carregamento das páginas, já que para cada serviço incluído no site seria necessário um tempo extra para o carregamento deste serviço, mesmo quando o serviço não é usado (HALES, 2013).

Também existe o problema de cada serviço de compartilhamento ter suas interfaces com o site alteradas, o que poderia resultar em perda de comunicação ou prejudicar alguma interação com o provedor de serviços. Além disso, novos serviços surgem diariamente na Internet e, para manter o site

Por outro lado, se o site optar por usar apenas os botões dos principais serviços de compartilhamento, o site pode estar privando o usuário de usar o serviço de sua preferência.

2.4 O EXEMPLO DO ANDROID

O *Web Intents* foi inspirado no modelo Intents do Sistema Operacional Android. Neste modelo, o sistema operacional monitora quando certos eventos acontecem e executa a ação correspondente ao evento disparado. Coordenar o processo de comunicação entre os aplicativos, sistema operacional e o usuário é a função do Intents.

Um exemplo simples de como funciona o Intents do Android é quando se usa a opção “Compartilhar página” do navegador do Android. Ao clicar nessa opção é mostrada uma lista de aplicativos que podem ser usadas pelo usuário para compartilhar a página em que ele está no momento.

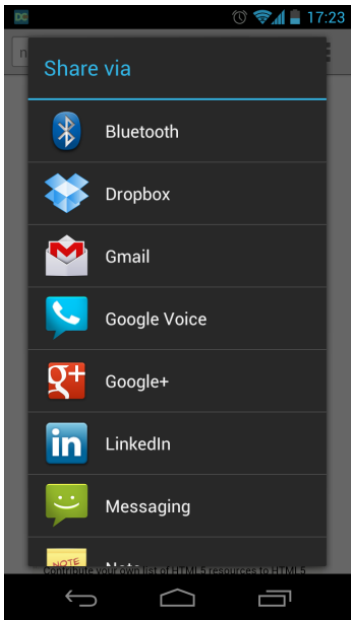

Figura 3 - Exemplo do Intents do Android

No Intents do Android, o aplicativo cliente solicita uma ação genérica, por exemplo: “compartilhar”. O usuário recebe uma lista de aplicações que existe

registrado para a ação solicitada. Ao selecionar a aplicação, o programa será executado e os dados entre a aplicação de origem e a aplicação de destino serão enviados sem que seja necessária qualquer outra intervenção do usuário.

No Android, ao se instalar um aplicativo, o aplicativo registra no sistema operacional quais “intenções” ele poderá responder. Quando a ação (intenção) for solicitada pelo usuário, o aplicativo aparecerá como uma das opções disponíveis. O sistema operacional não precisa saber como funciona o aplicativo nem precisa integrar-se com a API de nenhum aplicativo. O *Web Intents* facilita a troca de informações entre o aplicativo e o sistema operacional ao oferecer uma API padrão para os eventos do sistema.

É desejável que páginas ou aplicações *Web* tivessem o mesmo comportamento que o Android para situações como a que descrevemos acima. Neste sentido, um novo padrão começou a ser rascunhado por um grupo de trabalho no W3C. Este padrão foi intitulado *Web Intents*. O *Web Intents* foi projetado para ser primeiramente uma API simples e fácil de usar. Com ela será possível conectar as aplicações *Web* com as aplicações de serviço com poucas linhas de código, sendo que o navegador cuidará do resto do trabalho. Assim como no Android, o *Web Intents* contará com um conjunto inicial de ações (editar, visualizar, compartilhar, etc.) que provavelmente cobrirá a maior parte dos casos de uso na *Web* atual; porém, com o crescimento da *Web* e o aumento de funcionalidade das página da *Web*, novas ações poderão ser incorporadas ao padrão.

2.5 POSSÍVEIS CASOS DE USO

O *Web Intents* representa o próximo passo lógico no modelo atual, ao invés de agregar dados de múltiplas fontes, o Web Intents permite processar dados de vários serviços que pertencem a diferentes entidades. Os possíveis casos de uso são: integrar pesquisas em serviços de terceiros, edição de imagens, serviços de autenticação, serviços de compartilhamento, salvar arquivos em nuvem, enviar arquivos da nuvem direto para provedores de serviço, encurtador de URLs, entre outros.

Por exemplo, se você estiver usando um serviço de armazenamento em nuvem como o Dropbox e quiser editar um arquivo Word, você poderia escolher entre o Word Online da Microsoft ou poderia usar o editor do Docs do Google. Ao

final da edição poderia salvar esse arquivo de volta ao Dropbox. Todas essas interações entre provedores de serviço seriam feitas sem que o usuário precisasse mexer com o sistema de arquivos. É como se o arquivo pudesse estar sempre na nuvem sem a necessidade trazer o arquivo para o computador local.

3 WEB INTENTS

O *Web Intents* é um sistema RPC extremamente leve entre aplicativos, modelado a partir do sistema Android. Uma intenção é uma ação iniciada pelo usuário e delegada para ser executada em um provedor de serviços. Uma Intenção fornece uma sintaxe declarativa que permite que fornecedores de serviço listem as Intenções que eles são capazes de lidar, o Agente de Usuário (*User Agent*, “o navegador”) permite que o usuário selecione qual provedor de serviço usar e o provedor do serviço executa a ação da Intenção, possivelmente usando os dados passados pela intenção como parâmetros de entrada para a execução do serviço. O provedor de serviços pode ou não retornar dados como saída para o cliente. (BILLOCK; HAWKINS; KINLAN, 2012)

O *Web Intents* permite rica integração entre serviços e aplicações *Web*. As aplicações disponíveis na *Web* têm uma necessidade de transmitir dados entre si para completar suas funções. Ao possibilitar que aplicações *Web* se integrem com serviços de terceiros sem necessidade de acesso a APIs especificas de cada serviço, o *Web Intents* mantém o baixo acoplamento e facilita a troca de informações entre os aplicativos (KASARKOD, 2011).

3.1 COMO O WEB INTENTS FUNCIONA?

A especificação do *Web Intents* permite que uma aplicação *Web* solicite a execução de uma ação abstrata, por exemplo, o compartilhamento ou a visualização de um recurso e permite que provedores de serviços se registrem para lidar com essas ações. O navegador funciona como um corretor, conectando anonimamente uma aplicação *Web* com o serviço compatível com o que foi escolhido pelo usuário anteriormente (LYLE; NILSSON; ISBERG, 2013).

Digamos, por exemplo, que uma pessoa queira editar uma foto que foi enviada para um serviço de hospedagem que não tenha a função para editar a foto. Esse serviço de hospedagem poderia usar o *Web Intents* e permitir ao usuário

escolher um editor de fotos online para realizar esse trabalho. O mecanismo de *Web Intents* permite a integração da aplicação com os serviços de edição de imagens com um esforço bastante reduzido.

Com o *Web Intents* os serviços são explicitamente registrados e associados a intenções, que, basicamente, representam ações como editar, visualizar, compartilhar etc., incluindo parâmetros relevantes para cada ação. Em tempo de execução, é possível escolher qual dos serviços registrados responderá por uma determinada ação. O serviço escolhido recebe a responsabilidade pela execução, utilizando os dados fornecidos pela aplicação de origem (KASARKOD, 2011).

O ciclo de vida do *Web Intents* é primeiro, o usuário solicitar uma ação para ser executada. Os dados da “intenção” são passados para o navegador, que permite ao usuário selecionar qual dos serviços registrados podem ser executados para a ação escolhida. Então o serviço é executado e é mostrada uma interface gráfica que executa a ação especificada pela intenção. Finalmente o serviço pode retornar dados para a página que originou o *Web Intents*.

3.2 CICLO DE VIDA DO WEB INTENTS

1. O provedor de serviços inclui alguns códigos de programação na página *Web* para incluir a ação que o serviço oferece no navegador do usuário.


```
<!-- codigo para fazer a instalação do webintents no navegador do usuário -->
<html>
  <head>
     <script src="http://Web Intents.org/Web Intents.min.js"></script>
  </head>
  <body>
     <intent
        action="http://Web Intents.org/share"
        type="*"
        href="servidor.html"
        />
  </body>
</html>
```

Figura 4 - Código *Web Intents* no provedor de serviços

2. Quando o usuário encontra alguma página da qual gostaria de compartilhar, o usuário clica em um botão genérico de compartilhamento e serão mostradas as opções de provedores de serviço.

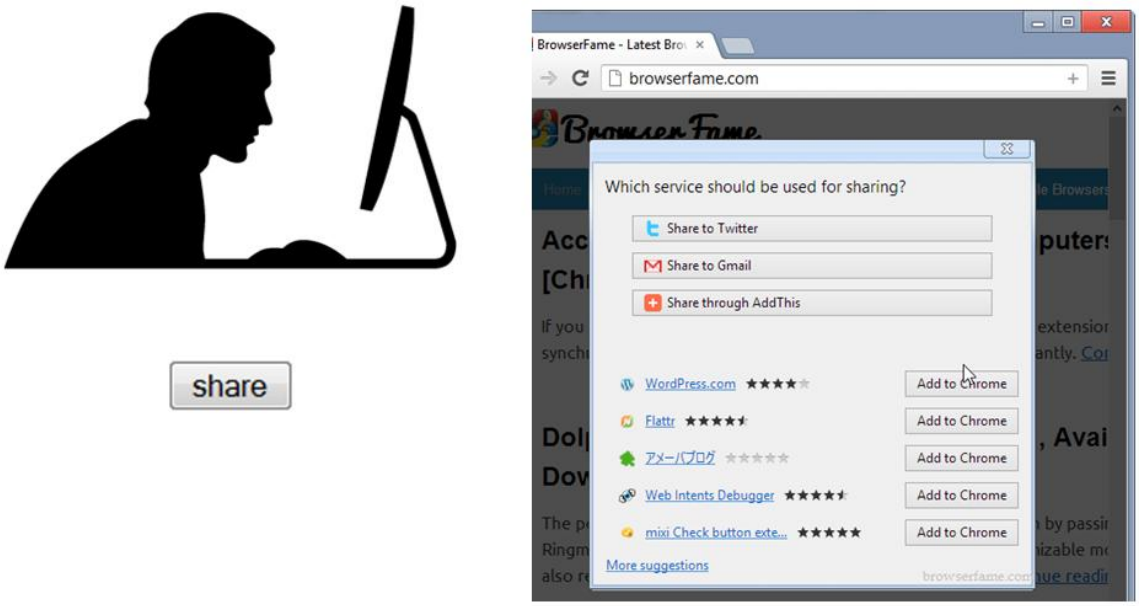

Figura 5 - Janela com as opções do *Web Intents*

3. O servidor da página *Web* irá se comunicar com o provedor de serviços e o provedor de serviço irá executar a ação solicitada pelo usuário. Durante a execução da ação o provedor de serviços poderá solicitar algumas informações ao usuário como autenticação. No fim da execução, o provedor de serviços poderá retornar alguma informação para a página *Web* que o chamou.

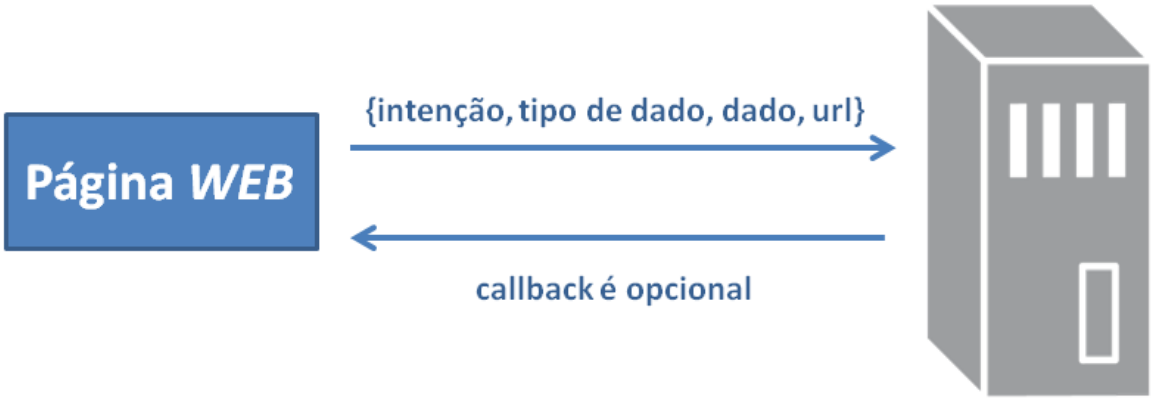

Figura 6 - Comunicação entre a página *Web* e o provedor de serviços usando *Web Intents*

O processo de registro de serviços ainda está indefinido, uma das propostas é que o processo seja automático. Por exemplo, quando o usuário entra em um serviço como http://twitter.com, nos bastidores, o Twitter vai registrar seus serviços

no navegador do usuário. Então, se o usuário for para o https://www.tumblr.com e escrever um post no blog, o Tumblr também vai registrar seus serviços no navegador do usuário. Dessa forma o navegador vai gradualmente construir uma lista de serviços de acordo com a utilização dos serviços.

3.2.1 Lado Cliente

Um desenvolvedor *Web* poderia usar o recurso de compartilhar a página *Web* para redes sociais através do *Web Intents* usando o seguinte código:

```
<html>
   <head>
      <script src="http://Web Intents.org/Web Intents.min.js"></script>
      <script type="text/javascript">
         function addListeners(){
           document.getElementById('btn-compartilhar').addEventListener("click",btnclick,false);
         function btnclick(){
           var intent = new Intent(
                                  "http://Web Intents.org/share",
                                  "image/*",
                                  document.URL
                                  );
            window.navigator.startActivity(intent, pickOnSuccess, pickOnError);
            function pickOnSuccess(data) {
              alert("Compartilhado com sucesso");
            }
            function pickOnError(data) {
              alert("Erro ao fazer o compartilhamento");
            }
         }
         }
         window.onload = addListeners;
      </script>
   </head>
   <body>
      <button id="btn-compartilhar">Compartilhar</button>
   </body>
</html>
```

Figura 7 - Código fonte para usar o *Web Intents* em uma página *Web*

Esse código delega o compartilhamento para o provedor de serviços e este provedor de serviço dá andamento à ação de compartilhar. Após a execução da ação, a página recebe a resposta do provedor, informando se o compartilhamento foi feito com sucesso ou não.

Observe que a integração não é específica de um único serviço *Web*. Qualquer provedor de serviços pode oferecer a integração com a aplicação. O usuário pode ter muitos serviços de compartilhamento registrados no navegador e escolher a opção que melhor agrade em cada situação.

3.2.2 Lado Servidor

Do lado do provedor de serviços, será necessário criar duas páginas: uma para a instalação do serviço no navegador do usuário e outra que será para executar a ação do *Web Intents*.

Na página de instalação do *Web Intents* é necessário constar a informação sobre qual ação do *Web Intents* o provedor de serviço dará suporte, os tipos de dados que serão recebidos como parâmetro e o endereço que o serviço usará para executar a ação.

```
<html>
   <head>
      <script src="http://Web Intents.org/Web Intents.min.js"></script>
   </head>
   <body>
      <intent
         action="http://Web Intents.org/share"
         type="*"
         href="servidor.html"
         />
   </body>
</html>
```

Figura 8 - Código fonte para usar no provedor de serviços para instalação do serviço

No exemplo da figura 8, a ação a ser instalada no navegador é o compartilhar (*share*).

Na página onde a ação do *Web Intents* é executada, é necessário preparar a página para receber parâmetros de entrada e, em alguns casos, retornar dados após a execução do serviço:

```
<html>
   <head>
      <script src="http://Web Intents.org/Web Intents.min.js"></script>
      <script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
      <script>
         $(document).ready(function() {
           var urlToShare = intent.data;
           var twitterURL = "http://twitter.com/intent/tweet"
                          + "?via=utfpr"
                          + "&text=Veja isso! - " + urlToShare
                          + " - Compartilhado usando Web intents";
           $("#twitterlink").attr("href", twitterURL);
           $.getScript("http://platform.twitter.com/widgets.js", function () {
               function handleTweetEvent(event) {
                   intent.postResult("Link successfully sent to twitter!");
                   // retornando dados para a aplicação que chamou
               }
               twttr.events.bind('tweet', handleTweetEvent);
           });
         });
      </script>
   </head>
   <body>
      <a id="twitterlink">Compartilhar no Twitter</a>
   </body>
</html>
```

Figura 9 - Código fonte para usar no provedor de serviços para execução do serviço

3.3 JUSTIFICATIVA PARA A UTILIZAÇÃO DO WEB INTENTS

A gama de benefícios vai desde coisas pequenas como redução de botões de compartilhamento em página da *Web* até aumentar as opções de provedores de serviços, já que os desenvolvedores não mais precisarão programar pontos de integração para o serviço diretamente. Isso significa que em vez que colocarem alguns poucos serviços amplamente adotados, os provedores de serviço de nicho poderão se encaixar no eco-sistema junto com as demais marcas mais conhecidas (WILSON, 2011).

3.4 INSTALAÇÃO DO WEB INTENTS

John J. Barton argumenta que a especificação *Web Intents* é muito vaga para ser implementada e deixa sem respostas algumas questões específicas como o método como o usuário viria a acumular o conjunto de opções para lidar com qualquer uma das intenções. Em um exemplo de edição de imagens, quando o usuário clicar "Editar" deve existir uma aplicação registrada para lidar com a intenção Editar senão o comando não seria completado. "Então vocês estão contando com os usuários para ir cadastrando uma lista de serviços uteis antes que eles precisem do serviço?" (BARTON, 2012)

Billock respondeu que existem três possibilidades. Primeiro, que os serviços de busca poderiam indexar as páginas da *Web* que oferecem as intenções e construir registros de aplicações compatíveis com o que for sugerido. Segundo, as *Web Store*s como o Chrome *Web Store* poderiam destacar as intenções de sua lista de aplicações. Ou terceiro, as página da *Web* que usam o *Web* Intent (como a galeria de imagens do exemplo) poderiam simplesmente incluir links para os serviços compatíveis com o que eles recomendam (BILLOCK, 2012).

3.5 SITUAÇÃO ATUAL DO WEB INTENTS

O estado do documento que descreve as especificações do *Web Intents* no W3C está definido como sendo uma “Nota do grupo de trabalho”, sendo que os membros desse grupo de trabalho concordaram em não avançar a especificação para um documento de recomendação.

O suporte ao *Web Intents* foi removido antes do lançamento do beta da versão 24 do navegador Chrome. De acordo com Google, o *Web Intents* foi colocado na versão alfa apenas para testar em condições reais e ver o que poderia ser melhorado.

“Nós lançamos o suporte experimental para o *Web Intents* no Chrome com a intenção de obter dados e feedback sobre a API e informar sobre seu desenvolvimento” explicou Greg Billock do Google, na lista de discussão do *Web Intents* no W3C. (BILLOCK, 2012)

“Agora que nós conseguimos dados suficientes, nós desabilitamos o suporte experimental para o *Web Intents* na versão 24” ele complementou. Billock diz que a remoção é temporária, embora pareça como um retorno do Google ao começo, pelo menos em algumas áreas.

> Após revisar cuidadosamente as características (do *Web Intents*), nós identificamos uma quantidade de áreas no desenvolvimento da API e na experiência do usuário. Nós iremos usar os resultados desse experimento a fim de assegurar se estamos abordando o problema de comunicação entre aplicativos *Web* da melhor maneira possível. (BILLOCK, 2012).

4 ALTERNATIVAS AO WEB INTENTS

4.1 WEB ACTIVITIES

O *Web Activities* é a API da Mozilla com as mesmas características e funcionalidades parecidas com a do *Web Intents*, onde o principal objetivo é mesmo: deixar de lado o modelo atual, onde a página da *Web* e o provedor de serviços precisam estar unidos através de APIs específicas, em favor de um modelo onde os provedores de serviço e as página da *Web* utilizam uma mesma API para se comunicarem e, através dessa API, permitir categorizar serviços de acordo com as ações que os provedores de serviço oferecem.

Embora o Google (*Web Intents*) e a Mozilla (*Web Activities*) compartilhem da mesma visão, suas implementações e objetivos finais são ligeiramente diferentes. A equipe de desenvolvimento da Mozilla é mais interessado em ter uma API mais simples que permita uma aplicação delegar uma atividade para outra aplicação com regras bem claras a respeito como é feito o inicio da atividade, como a atividade se sucede e a interface do usuário durante o todo o processo. Já a equipe de desenvolvimento do Chrome é mais focado em fazer uma API mais abrangente, que possa delegar atividades para outra aplicações e estabelecer um canal de comunicação entre o aplicativo e o provedor de serviço.

Mounir Lamouri, engenheiro de software da Google e antigo engenheiro da plataforma da Mozilla, propôs um no grupo de discussão, um rascunho para a especificação do *Web Activities*. O *Web Activities* é uma proposta com escopo mais limitado que o *Web Intents*. Discute-se apenas três ações (compartilhar, editar e escolher), mas, também, afirma que qualquer string não-vazia deve ser considerada uma ação válida. No entanto, em sua proposta, Lamouri argumenta que o *Web Intents* vai longe demais ao tentar cobrir a comunicação em duplo sentido entre as aplicações. Esse grau de interoperabilidade deve ser feito com APIs específicas e não gerenciado por um mecanismo para propósito geral como o *Web Activities* ou *Web Intents*. (LAMOURI, 2012)

Na proposta de Lamouri (LAMOURI, 2012), o aplicativo declara sua habilidade de lidar com um *Web Activities* através da propriedade *Activities* {} no arquivo de manifesto (ao invés de ser uma nova tag HTML), uma escolha que outro desenvolvedores da Mozilla parecem concordar. Tantek Çelik, um dos que trabalha

com os padrões *Web* na Mozilla, explica: “O objetivo em basear a descoberta em aplicativos *Web* é o presumir que, ao instalar o aplicativo, o processo estará centrado no usuário, sendo mais seguro, aumentando a privacidade e sendo um processo entendível”. (COBBETT, 2012)

O estado atual do desenvolvimento do *Web Activities*, que é experimental e ainda espera pela proposta de rascunho da especificação, usa o Window.postMessage, mas também, está em um estágio inicial de desenvolvimento. Lamouri disse em um email que o plano de Mozilla é apresentar o *Web Activities* para o Grupo de Trabalho de Aplicações *Web* da W3C quando a API estiver finalizada (LAMOURI, 2012).

4.2 WINDOW.POSTMESSAGE

As aplicações *Web* já possuem um mecanismo para se comunicar com outras aplicações *Web* através do navegador: a interface window.postMessage. O window.postMessage faz parte da API do HTML5 e permite enviar dados entre duas janelas/frames através de domínios diferentes. (WALSH, 2010)

O window.postMessage é simples, produtivo e é facil de construir novas idéias sobre ele. O window.postMessage não se importa com tipos mime, caixa de diálogo, callbacks. Ele simplesmente é um canal autenticado de mensagens.

A única razão pela qual o window.postMessage não pode ser usada no lugar do *Web Intents* é que o despachante (página da *Web*) e o destinatário (provedor de serviços) são, na maioria dos casos, bastante acoplados. O despachante precisa especificar o receptor, o que significa que o usuário não poderia escolher um serviço substituto de sua preferência.

Ben Adida, engenheiro da Square e antigo diretor de engenharia da Mozilla, sugeriu que o window.postMessage fosse aprimorado e passasse a ser de baixo acoplamento, permitindo assim que o usuário pudesse escolher e combinar despachantes com destinatários. Ben Adida, diz o seguinte: “Dado que a abstração do window.postMessage tem sido bem-sucedido e útil, o modo “correto” de avançar é ajustá-lo, minimamente, não redesenhar uma pilha diferente” (ADIDA; BILLOCK, 2012).

Uma das propostas para o window.postMessage funcionar com baixo acoplamento é usar um protocolo personalizado como alvo dos canais window.postMessage. Assim, quando uma grande portal de notícias quer compartilhar um artigo, em vez de usar o window.postMessage (ou linkar) para http://twitter.com, poderia usar uma URL (*Uniform Resource Locator*) um pouco diferente como share://twitter.com. O navegador poderia substituir a URL pelo provedor de serviço preferencial do usuário. A diferença é que o prefixo poderia dar ao usuário a chance de intervir e dizer “por favor, este serviço”.

4.3 WEB ACTIONS

O *Web Actions* são ações que o usuário executa em uma página da *Web* e a ação se estende a outra página da *Web* (WALTERS, 2012). O *Web Actions* não é uma tecnologia especifica, mas uma variedade de tecnologias, como:

1. Um *hyperlink*, botão, iFrame ou outro elemento para indicar a possibilidade de usar uma ação especifica;
2. Marcação declarativa (link ou formulário de submissão) para executar a ação;
3. JavaScript (que use técnicas AJAX) para melhorar de forma discreta a interação dos elementos visíveis fornecendo uma interação alinhada e responsiva;
4. Uma janela pop-up de confirmação, para fornecer uma informação adicional;
5. A execução de uma ação em outra página da *Web* ou outra aplicação.

O *Web Actions* consiste de várias partes de software com baixo acoplamento que permitem ao usuário especificar serviços para agente do usuário (navegador) e este delegar ações para o provedor de serviços escolhido. Um bom exemplo de *Web Actions* é o “Twitter’s *Web Intents*” que faz uso de marcação semântica mínima e bem formatada.

O *Web Actions* é proposta de usar a tecnologia que temos hoje ou com pequenas adaptações, usando as melhoras práticas de programação *Web*, como uso de marcação semântica, da não necessidade de scripts, da manutenção da privacidade do usuário, do desenvolvimento de página da *Web* de carregamento rápido e sem uso excessivo de memória.

O *Web Actions* é de baixo acoplamento, pois a ligação com o provedor de serviços é feito através de links comuns <a href> e não através de marcação não semântica como e script.

Uma característica interessante do *Web Actions* é que a tecnologia segue os conceitos do padrão REST, já que a URL da requisição do *Web* Action pode ser feita apenas usando a operação GET, mas sempre obtendo a UI (*User Interface* - Interface do Usuário) de onde outras requisições poderão ser feitas, sem existir abuso na utilização dos verbos HTTP (WALTERS, 2012).

4.3.1 Marcação "ACTION"

Para ser possível agrupar ações de provedores de serviços em um único elemento, como botão ou *hyperlink*, foi sugerido a marcação action:

```
<action do="post" with="permalink">
  <a href=twitter>..</a>
  <a href=pinterest>...</a>
  <a href=tumblr>...</a>
</action>
```

Figura 10 - Marcação do *Web Actions*

Onde o "do" aceita um verbo (podem ser múltiplos verbos). Os verbos mais comuns, de acordo com Tantek Çelik (ÇELIK, 2011), seriam:

- Depois (*later*): "Eu não tenho tempo ou é muito longo". Readability e Instapaper são exemplos;
- Salvar (*save*): "Isso é interessante e eu gostaria de guardar isso". Parecido com o "depois", mas com a motivação de colecionar, categorizar e saber que em algum lugar será possível achar isso novamente, uma vez que pode ser útil algum dia. É uma forma de favoritar (Bookmarking);
- Apoiar (*props*): "Isso é legal. Eu gosto disso" - Quando o item é digno de merece reconhecimento Digg, Like, +1 são alguns exemplos;
- Compartilhar (*share*): "Isso é muito bom que eu gostaria que compartilhar. Meus amigos vão querer ver isso". Exemplos: Tweet, Blog This e Tumblr;
- Seguir (*follow*): "Eu gostaria de ouvir mais deste autor". Enquanto que o botão do Twitter "*follow*" é um exemplo óbvio, os RSS (assinar meu feed) também entra nessa categoria.

O parâmetro “*with*” é a URL que suportará a ação do verbo.

Dentro do <action> estariam os elementos que a página da *Web* usaria como fallback para navegadores que não suportem o *Web Actions*, assim a codificação seria equivalente ao que existe hoje.

4.3.2 Suporte dos Navegadores

Se o navegador tiver suporte ou se existir algum complemento do navegador que permita usar a tag “action” então será possível clicar nos botões de ações do *Web Actions* em qualquer página da *Web*. O “*Web* Action Hero Toolbelt” é uma extensão que permite usar o *Web Actions* em Firefox, Safari, Chrome e Opera.

4.3.3 Modo de funcionamento

O *Web* Action é configurado no agente do usuário (navegador) onde o usuário escolhe os provedores de serviços de acordo com o sua preferência. Quando o usuário clicar no botão do *Web Actions* será mostrado as opções de acordo com o que foi configurado no navegador.

4.3.4 Ciclo de vida do Web Actions

1. O desenvolvedor da página *Web* coloca código *Web Actions* para que a página *Web* possa responder esse tipo de solicitação (figura 11)

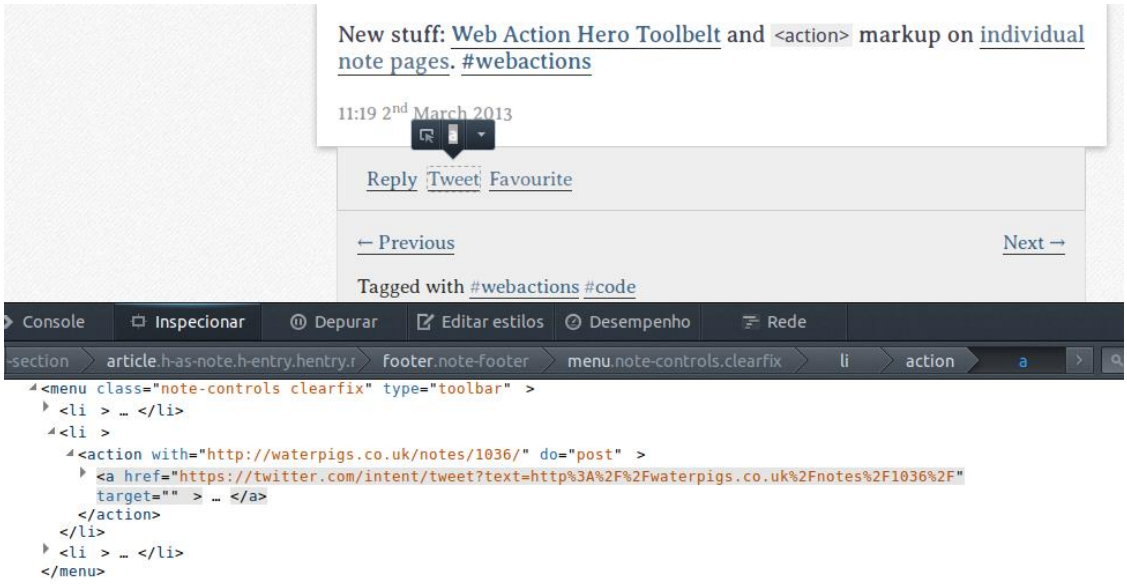

Figura 11 - Página *Web* com código *Web Actions*

2. O usuário configura no seu navegador a ação e quais serão os provedores de serviço que responderão essa ação (figura 12).

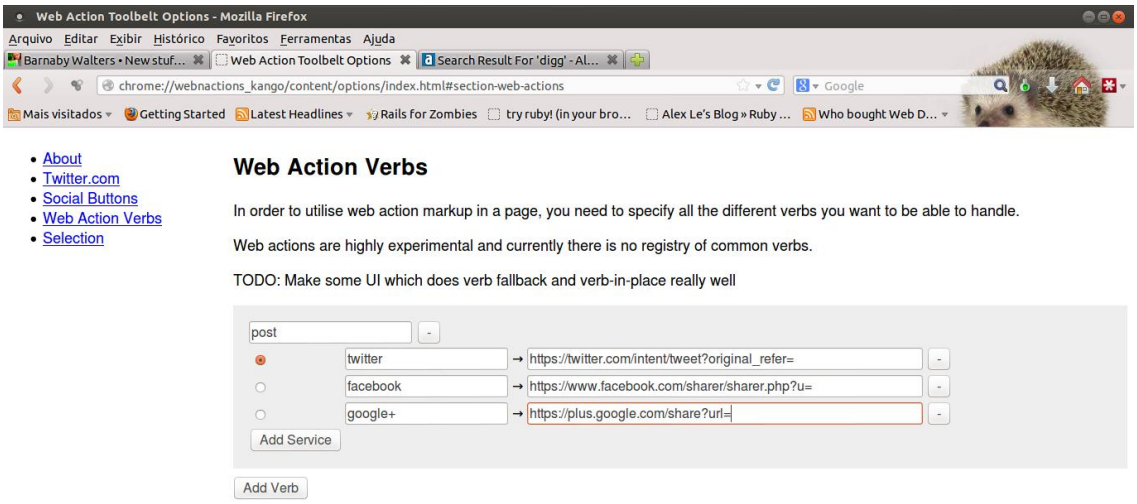

Figura 12 - Configuração do *Web Actions* no navegador do usuário

3. Quando o usuário estiver em uma página *Web* o usuário poderá usar o botão do *Web Actions*, caso a página *Web* ofereça essa opção (figura 13).

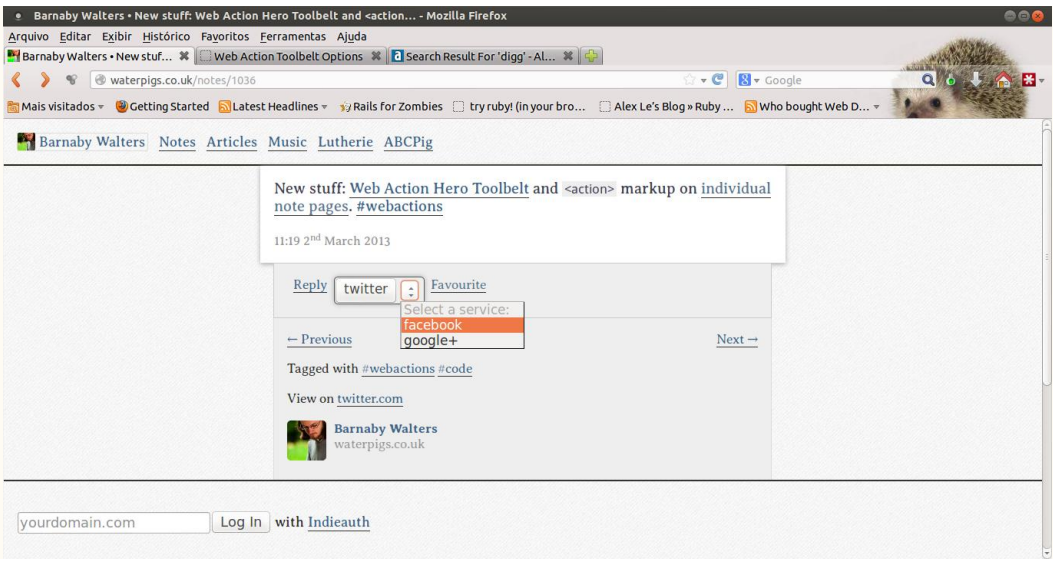

Figura 13 - O *Web Actions* monstrando as opções de compartilhamento

5 ANÁLISE E CONSIDERAÇÕES FINAIS

Na lista de discussão e blog, os desenvolvedores deixam claro que, apesar da lista de intenções padrões, eles não têm a intenção de definir rigorosamente um conjunto fixo de intenções aceitáveis. Essa posição levou a algumas perguntas sobre o que exatamente deve ser tratado como uma intenção e o que não deve.

Uma das questões recorrentes é por que uma intenção - que pode passar qualquer estrutura de dados através de um frame do navegador para outro - é melhor do que outros métodos existentes para a comunicação *cross-site*, como window.postMessage, que não exigem a definição de novos elementos HTML e propriedades de DOM. Ben Adida da Mozilla escreveu que a combinação window.postMessage e um pequeno conjunto de esquemas URI (tais como share://) iria resolver os mesmos problemas de uma forma mais simples.

A resposta mais citada é que o *Web Intents* implementa operações simples e genéricas entre aplicações *Web* mutualmente desconhecidos, enquanto que Window.postMessage é usado em APIs mais abrangentes e completas entre páginas da *Web*s que são projetados de antemão para trabalhar juntos. Mas não existe uma distinção clara entre os dois casos de utilização.

Ben Adida acredita que o *Web Intents*, como proposto, é excessivamente complexo. Assim como o *Web Activities*. Ele argumenta que não necessitamos de tipos de mime, novos elementos HTML ou novas propriedades DOM. Precisamos apenas aprimorar o que já existe (Window.postMessage e registerProtocolHandler) e usar algumas bibliotecas JavaScript para tornar mais fácil o trabalho dos desenvolvedores. (ADIDA; BILLOCK, 2012)

Greg Billock da Google, no entanto, tem uma opinião diferente. Ele argumenta que o principal problema não é encarado. O problema não é estabelecer a comunicação do transporte de pacotes. É na semântica das mensagens trocadas. “É possível configurar o transporte inter-páginas com nada além de iFrames, entradas /etc/hosts e manipulação DOM, mas não é uma grande surpresa que um eco-sistema ainda não tenha surgido em torno dessa viabilidade bruta de comunicação” (ADIDA; BILLOCK, 2012).

Alex Russel, engenheiro de software da Google, acrescenta que, se a tecnologia a ser usada for apenas de baixo nível (Window.postMessage e registerProtocolHandler) a quantidade de pessoas a fazer uso efetivo desses recursos será drasticamente reduzido. Também destruirá a capacidade dos motores de busca de encontrarem a semântica do *Web Intents*, retardando o avanço da *Web* ao dificultar a descoberta de serviços que poderiam ser usados pelas páginas da *Web*. (ADIDA; BILLOCK, 2012)

5.1 CÓDIGO FONTE PARA COMPARAÇÃO DAS TECNOLOGIAS

5.1.1 Código do provedor de serviço para instalação do serviço

5.1.1.1 Web Intents

```
<intent
   action="http://webintents.org/share"
   href="http://provedor.com/share.html"
   type="text/uri-list"
   title="Provedor" />
```

5.1.1.2 Web Activities

```
// através de Arquivo de Manifesto no formato JSON
'mymanifest.webapp':
{
  name: "SharingApp",
  icons: {
    32: "/icons/32.png"
  },
  services: {
    "http://webactivities.org/share": {
       type: [ "application/url", "text/*", "image/*" ]
       path: "/services/share.html"
    }
  }
}
```

5.1.1.3 Window.postMessage

```
navigator.registerProtocolHandler("share",
                                  "https://www.provedor.com/?uri=%s",
                                  "Provedor");
```

5.1.1.4 Web Actions

Não necessita que o provedor de serviços instale nada no navegador. O usuário cadastra no navegador quais provedores de serviço deseja usar para cada tipo de ação.

5.1.2 Código para incluir na página Web para chamar uma ação

5.1.2.1 Web Intents

```
var intent = new Intent({
   "action": "http://webintents.org/share",
   "type": "text/uri-list",
]);
navigator.startActivity(intent, onSuccess, onError);
function onSuccess(data) {
   doSomethingWithURL(data);
}
function onError(data) {
   alert("Erro ao compartilhar!");
}
```

5.1.2.2 Web Activities

```
shareActivity = new Activity("http://webactivities.org/share",
                             "text/html", {url: location.href});
navigator.apps.startActivity(
   shareActivity,
   function (successResult) {
      alert('Thanks for sharing us!');
   },
   function (errorResult) {
      alert('Sharing failed :( ' + errorResult));
   }
);
```

5.1.2.3 Window.postMessage

```
<a href="share://www.manualdousuario.net" target="_blank">Compartilhar</a>
```

5.1.2.4 Web Actions

```
<action do="share" with="permalink">
   <a href=twitter>..</a>
   <a href=pinterest>...</a>
   <a href=facebook>...</a>
</action>
```

5.1.3 Código do provedor de serviços para aceitar as solicitações

5.1.3.1 Web Intents

```
<html>
   <head>
      <script src="http://Web Intents.org/Web Intents.min.js"></script>
      <script src="http://code.jquery.com/jquery-2.0.3.min.js"></script>
      <script>
         $(document).ready(function() {
           var urlToShare = intent.data;
           // do something with urlToShare
         intent.postResult("Link successfully sent to...");
         });
      </script>
   </head>
   <body>
   </body>
</html>
```

5.1.3.2 Web Activities

```
navigator.apps.services.registerHandler(
   'http://webactivities.org/share',
   function(activity, credential)
   {
      my_ajax.post_share({
      url: activity.data.url,
      title: activity.data.title,
      comment: activity.data.comment,
      credential: credential,
      success: function() {
         activity.postResult({status:"ok"});
      },
      error: function(statusCode) {
         if (statusCode == 403) {
            activity.postException(activity.CREDENTIAL_FAILURE);
         } else {
            activity.postException(activity.FAILURE);
         }
      }
    });
  }
);
```

5.1.3.3 Window.postMessage

```
<?php
   $value = "";
   if ( isset ( $_GET["value"] ) ) {
     $value = $_GET["value"];
   }
 ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
   <head>
      <title>Web Protocol Handler Sample</title>
   </head>
   <body>
      <h1>Web Protocol Handler Sample - Handler</h1>
      <!-- do something with $value -->
      <?php echo(htmlspecialchars($value, ENT_QUOTES, 'UTF-8')); ?>
   </body>
</html>
```

5.1.3.4 Web Actions

O provedor de serviços que lida com as solicitações do Web Actions recebe os parâmetros como uma página Web comum, através do GET do REST.

```
<?php
   $value = "";
   if ( isset ( $_GET["value"] ) ) {
     $value = $_GET["value"];
   }
 ?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="en">
   <head>
      <title>Web Protocol Handler Sample</title>
   </head>
   <body>
      <h1>Web Protocol Handler Sample - Handler</h1>
      <!-- do something with $value -->
      <?php echo(htmlspecialchars($value, ENT_QUOTES, 'UTF-8')); ?>
   </body>
</html>
```

5.2 SEM O APOIO DOS PRINCIPAIS PROVEDORES DE SERVIÇO

Segundo Glenn Jones, existe um valor oculto nos botões das redes sociais que permite que empresas possam rastrear o histórico do navegador. Os dados obtidos têm grande valor monetário para as empresas envolvidas e não existindo mais o Javascript nas páginas poderia desabilitar o rastreamento dos usuários. Sendo assim, o histórico do navegador poderia desencorajar as empresas a fornecerem interfaces para novas tecnologias. Esta é um bom motivo para usuários a usarem a tecnologia, mas poderia retardar sua adoção por parte das grandes provedores de serviço (COBBETT, 2012).

O sucesso que o *Web Intents* possa vir a apresentar passa necessariamente pela vontade da indústria, pela partilha e abertura dos processos de normatização e das próprias especificações dai resultantes. Parte significativa desse processo foi desenvolvida no âmbito do W3C, através de representantes da Google. O problema é que não existe um consenso em torno do *Web Intents* nem de outra tecnologia sobre como deverá ser feita a integração entre aplicativos *Web*. A Google e a Mozilla tentaram trabalhar em conjunto para unificar as propostas do *Web Intents* e o *Web Activities*, mas devido à diferentes objetivos a proposta não foi levada adiante. O Web Activities está sendo usado para integrar os serviços do Firefox OS, enquanto que o Web Intents está sendo repensado pelos engenheiros da Google.

Apesar desse percalço, o *Web Intents* ou alguma tecnologia nessa área de integração de aplicativos Web provavelmente terá seu lugar no mercado, pois o modelo atual de comunicação entre aplicativos é deficitário ao não permitir código independente do provedor de serviço. Recursos como permitir ao usuário escolher o serviço, a possibilidade de interfaces de usuário mais limpas e ao mesmo tempo

códigos não acoplados são justificativas suficiente para a existência e popularização dessa tecnologia.

Por que não nos deixar o Facebook/Twitter dominar? Os usuários realmente querem escolha?

> Sim, as estatísticas AddThis mostram uma cota de 67% do mercado nas mãos de Facebook e Twitter. Mesmo com uma diferença existe centenas de outros serviços que se encaixam no padrão de design do *Web Intents*. Talvez seja a incapacidade de fornecer escolha que define os padrões de uso corrente e não o contrário. (JONES, 2011)

As páginas *Web* atuais possuem um punhado de botões dos maiores provedores de serviços porque eles não têm meios de saber quais são os serviços que um usuário individual prefere. Se uma página *Web* pudesse entregar botão personalizado com base na escolha de um usuário, poderia haver um ganho substancial na utilidade desses botões e aumento do tráfego entre páginas Web e provedores de serviço. Essa mudança pode até não afetar o tráfego dos grandes provedores de serviços, mas em vez disso, aumentar o tráfego dos menores. (JONES, 2011).

5.3 TRABALHOS FUTUROS

O Google parece ter deixado o *Web Intents* em segundo plano mesmo sabendo que a tecnologia poderia ajudar a aprimorar o sistema Chrome OS que a mesma desenvolve. Achar os motivos que levaram o Google a deixar a tecnologia seria uma questão oportuna para identificar novas idéias ou tendências.

A fim de verificar a segurança do *Web Intents* seria conveniente um estudo sobre como o framework trata questões de confidencialidade e integridade dos usuários. No *Intents* do Android existe algumas vulnerabilidades como ataque *hijacking*[1]. Como o *Web Intents* partilha das mesmas funcionalidades do *Intents* do Android é possível que existam ameaças parecidas.

O *Web Intents*, O *Web Activities*, o Window.PostMessage e o *Web Actions* tem mais características que as apresentadas aqui nesse trabalho. Seria interessante detalhar tais características e evoluções de cada proposta. Também acompanhar

[1] O termo hijacking refere-se à exploração de uma sessão válida de um computador para conseguir acesso não autorizado a informações de um sistema de computador. (WIKIPEDIA, 2014)

novas tecnologias que venham a se destacar na área de integração de aplicativos Web e descoberta de serviços.

6 REFERÊNCIAS

19TH iNTERNATIONAL CONFERENCE ON WEB SERVICES, 2012. **Services Computing: The Foundation Discipline of the Modern Services Industry**. Disponível em: <http://conferences.computer.org/icws/2012>. Acesso em: 28 nov 2013

ADIDA, Ben; BILLOCK, Greg. **A simpler, webbier approach to Web Intents (or Activities)**. Disponível em: <http://benlog.com/2012/02/09/a-simpler-webbier-approach-to-web-intents-or-activities/>. Acesso em 28 nov 2013.

BARTON, John. **A simpler, webbier approach to Web Intents**. Lista de discussão sobre Web Intents. Disponível em: <http://lists.w3.org/Archives/Public/public-web-intents/2012Feb/0036.html> . Acesso em 30 nov 2013.

BELL, Michael; **Service-oriented modeling: service analysis, design**. Estados Unidos; Editora John Wiley & Sons, Inc. Hoboken, New Jersey. p. 282.

BILLOCK, Greg. **A simpler, webbier approach to Web Intents**. Lista de discussão sobre Web Intents. Disponível em: <http://lists.w3.org/Archives/Public/public-web-intents/2012Feb/0038.html>. Acesso em 28 nov 2013.

BILLOCK, Greg. **Status of web intents in Chrome**. Disponível em: <http://lists.w3.org/Archives/Public/public-web-intents/2012Nov/0000.html>. Acesso em 30 de nov de 2013.

BILLOCK, Greg; HAWKINS, James; KINLAN, Paul. **Web Intents**. W3C Working Draft. <http://www.w3.org/TR/2012/WD-Web-intents-20120626/>. Acesso em: 28 nov 2013

BROWN, Nicholas Watkins. **What does ReST really mean?** Disponível em: <http://standardout.wordpress.com/2012/04/14/what-does-rest-really-mean>. Acesso em: 28 nov 2013

ÇELIK, Tantek. **Web Actions: Identifying a New Building Block For The Web**. Disponível em: <http://tantek.com/2011/220/b1/Web-Actions-a-new-building-block>. Acesso em: 28 nov 2013

COBBETT, Richard. **Web Intents: the future of web apps**. Issue 229 of .Net Magazine. Disponível em: <http://tektile.com.ph/web-intents-the-future-of-web-apps>. Acesso em: 28 nov 2013.

FIELDING, Roy Thomas, **Architectural Styles and Design of Network-based Software Architectures**, PhD dissertation, University of California, Irvine, 2000

HALES, Wesley, **Sharing Content with Web Intents**. Disponível em: <http://wesleyhales.com/blog/2012/07/11/Sharing-With-Web-Intents-Today>.Acesso em 17 fev 2014.

GUIA DO DESENVOLVEDOR ANDROID, 2013. **Services**. Disponível em: <http://developer.android.com/guide/components/services.html>. Acesso em: 28 nov 2013

JONES, Glenn. **Choosing the Right Words**. Disponível em: <http://glennjones.net/articles/2011-10-04-choosing-the-right-words-web-intents>. Acesso em 30 de nov de 2013.

KASARKOD, Jeevak. **Web Intents do Google: um novo mecanismo para integração entre aplicações Web** <http://www.infoq.com/br/news/2011/08/web-intents-chrome>. Acesso em: 28 nov 2013

LAMOURI, Mounir. **Web Activities: counter-proposal to Web Intents**. Lista de discussão sobre Web Intents. Disponível em: < http://lists.w3.org/Archives/Public/public-web-intents/2012Jun/0061.html>. Acesso em 28 nov 2013.

LAMOURI, Mounir. **Web Activities v0**. Disponível em: <https://groups.google.com/forum/#!msg/mozilla.dev.Webapi/4KCxWBSkgqs/qJR1QnxzNcUJ>. Acesso em: 28 nov 2013

LAMPARTER, S; SCHNIZLER, B. **Trading Services in Ontology-driven Markets**, em 2006 ACM symposium on Applied computing, Dijon, France, 2006, p. 1679 – 1683

LYLE, John; NILSSON, Claes; ISBERG, Anders FAILY, Shamal. **Extending the web to support personal network services**. Disponível em: <http://www.cs.ox.ac.uk/files/5273/sac2013.pdf>. Acesso em: 28 nov 2013

RICHARDSON, Leonard; RUBY, Sam, **Restfull Web Services: Web services for the real world**, Sebastopol. CA. O'Reilly Media, 2007.

W3C, 2004. **OWL-S: Semantic Markup for Web Services**. W3C. Disponível em: <http://www.w3.org/Submission/OWL-S/>. Acesso em: 28 nov 2013

WALSH, David. HTML5's window.postMessage API. Disponível em: <http://davidwalsh.name/window-postmessage>. Acesso em 22 fev 2014.

WALTERS, Barnaby. **Web Actions**. Disponível em: <http://waterpigs.co.uk/articles/web-actions>. Acesso em: 28 nov 2013.

WIKIPEDIA.ORG. **Windows Communication Foundation**. Disponível em: <http://en.wikipedia.org/wiki/Windows_Communication_Foundation>. Acesso em: 28 nov 2013

WIKIPEDIA.ORG. **Session hijacking**. Disponível em: <http://pt.wikipedia.org/wiki/Session_hijacking>. Acesso em 23 fev 2013

WILSON, Scott. **Web Intents**. Disponível em: <http://scottbw.wordpress.com/2011/12/02/Web-intents/>. Acesso em: 28 nov 2013

ZHENG,Cheng; SHEN, Weiming; GHENNIWA Hamada H., **An Intents-based Approach for Service Discovery and Integration**. Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design. University of Western Ontario, Canada. 2013.