1.6 Estudo de caso: a World Wide Web

A World Wide Web [www.w3.org I, Berners-Lee 1991] é um sistema em evolução para a publicação e para o acesso a recursos e serviços pela Internet. Por meio de navegadores Web (*browsers*) comumente disponíveis, os usuários recuperam e veem documentos de muitos tipos, ouvem fluxos de áudio, assistem a fluxos de vídeo e interagem com um vasto conjunto de serviços.

A Web nasceu no centro europeu de pesquisa nuclear (CERN), na Suíça, em 1989, como um veículo para troca de documentos entre uma comunidade de físicos conectados pela Internet [Berners-Lee 1999]. Uma característica importante da Web é que ela fornece uma estrutura de *hipertexto* entre os documentos que armazena, refletindo a necessidade dos usuários de organizar seus conhecimentos. Isso significa que os documentos contêm *links* (ou *hyperlinks*) – referências para outros documentos e recursos que também estão armazenados na Web.

É fundamental para um usuário da Web que, ao encontrar determinada imagem ou texto dentro de um documento, isso seja frequentemente acompanhado de *links* para documentos e outros recursos relacionados. A estrutura de *links* pode ser arbitrariamente complexa, e o conjunto de recursos que podem ser adicionados é ilimitado – a "teia" (Web) de *links* é realmente mundial. Bush [1945] imaginou estruturas de hipertexto há mais de 50 anos; foi com o desenvolvimento da Internet que essa ideia pôde se manifestar em escala mundial.

A Web é um sistema *aberto*: ela pode ser ampliada e implementada de novas maneiras, sem perturbar a funcionalidade já existente . A operação da Web é baseada em padrões de comunicação e de documentos livremente publicados. Por exemplo, existem muitos tipos de navegadores, em muitos casos, implementados em várias plataformas; e existem muitas implementações de servidores Web. Qualquer navegador pode recuperar recursos de qualquer servidor, desde que ambos utilizem os mesmos padrões em suas implementações. Os usuários têm acesso a navegadores na maioria dos equipamentos que utilizam, desde telefones móveis até computadores de mesa.

A Web é aberta no que diz respeito aos tipos de recursos que nela podem ser publicados e compartilhados. Em sua forma mais simples, um recurso da Web é uma página ou algum outro tipo de *conteúdo* que possa ser armazenado em um arquivo e apresentado ao usuário, como arquivos de programa, arquivos de mídia e documentos em PostScript ou Portable Document Format (pdf). Se alguém inventar, digamos, um novo formato de armazenamento de imagem, então as imagens que tenham esse formato poderão ser publicadas na Web imediatamente. Os usuários necessitam de meios para ver imagens nesse novo formato, mas os navegadores são projetados de maneira a acomodar nova funcionalidade de apresentação de conteúdo, na forma de aplicativos "auxiliares" e "*plugins*".

A Web já foi além desses recursos de dados simples e hoje abrange serviços como a aquisição eletrônica de bens. Ela tem evoluído sem mudar sua arquitetura básica e é baseada em três componentes tecnológicos padrão principais:

- HTML (HyperText Markup Language), que é uma linguagem para especificar o conteúdo e o leiaute de páginas de forma que elas possam ser exibidas pelos navegadores Web.
- URLs (Uniform Resource Locators), que identificam os documentos e outros recursos armazenados como parte da Web.
- Uma arquitetura de sistema cliente-servidor, com regras padrão para interação (o protocolo HTTP – HyperText Transfer Protocol) por meio das quais os navegadores e outros clientes buscam documentos e outros recursos dos servidores Web. A

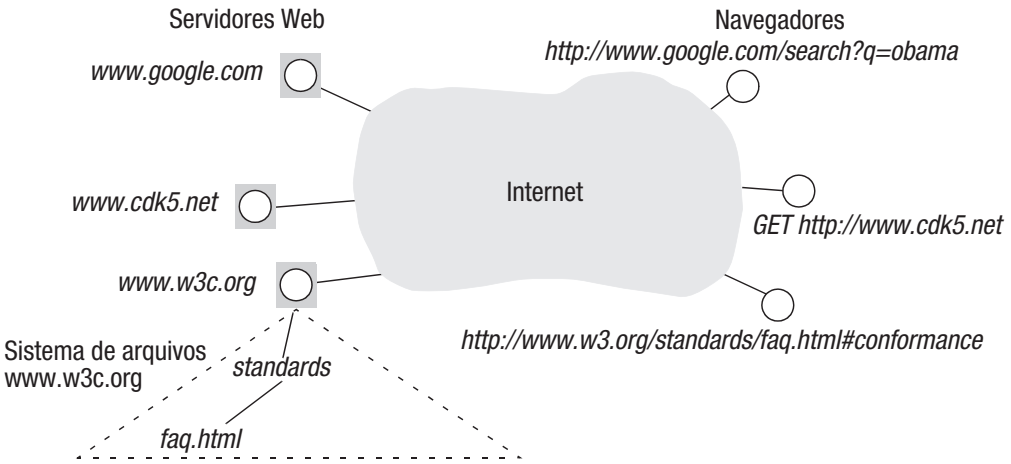

Figura 1.7 Servidores e navegadores Web.

Figura 1.7 mostra alguns navegadores realizando pedidos para servidores Web. É uma característica importante o fato de os usuários poderem localizar e gerenciar seus próprios servidores Web em qualquer parte da Internet.

Discutiremos agora cada um desses componentes e, ao fazermos isso, explicaremos o funcionamento dos navegadores e servidores Web quando um usuário busca páginas Web e clica nos *links* nelas existentes.

HTML • A HyperText Markup Language [www.w3.org II] é usada para especificar o texto e as imagens que compõem o conteúdo de uma página Web e para especificar como eles são dispostos e formatados para apresentação ao usuário. Uma página Web contém itens estruturados como cabeçalhos, parágrafos, tabelas e imagens. A HTML também é usada para especificar *links* e os recursos associados a eles.

Os usuários produzem código HTML manualmente, usando um editor de textos padrão ou um editor *wysiwyg* (*what you see is what you get*) com reconhecimento de HTML, que gera código HTML a partir de um leiaute criado graficamente. Um texto em HTML típico aparece a seguir:

```
<IMG SRC = "http://www.cdk5.net/WebExample/Images/earth.jpg">          1
<P>                                                                    2
Welcome to Earth! Visitors may also be interested in taking a look at the  3
<A HREF = "http://www.cdk5.net/WebExample/moon.html">Moon</A>.         4
</P>                                                                   5
```

Esse texto em HTML é armazenado em um arquivo que um servidor Web pode acessar – digamos que seja o arquivo *earth.html*. Um navegador recupera o conteúdo desse arquivo a partir de um servidor Web – neste caso específico, um servidor em um computador chamado *www.cdk5.net*. O navegador lê o conteúdo retornado pelo servidor e o apresenta como um texto formatado com as imagens que o compõe, disposto em uma página Web na forma que conhecemos. Apenas o navegador – não o servidor – interpreta o texto em HTML. Contudo, o servidor informa ao navegador sobre o tipo de conteúdo que está retornando, para distingui-lo de, digamos, um documento em Portable Document Format. O servidor pode deduzir o tipo de conteúdo a partir da extensão de nome de arquivo '.html'.

Note que as diretivas HTML, conhecidas como *tags*, são incluídas entre sinais de menor e maior, como em *<P>*. A linha 1 do exemplo identifica um arquivo contendo uma imagem de apresentação. Seu URL é *http://www.cdk5.net/WebExample/Images/earth.jpg*. As linhas 2 e 5 possuem as diretivas para iniciar e terminar um parágrafo, respectivamente. As linhas 3 e 4 contêm o texto a ser exibido na página Web, no formato padrão de parágrafo.

A linha 4 especifica um *link* na página Web. Ele contém a palavra *Moon*, circundada por duas tags HTML relacionadas *<A HREF...>* e *</A>*. O texto entre essas tags é o que aparece no *link* quando ele é apresentado na página Web. Por padrão, a maioria dos navegadores é configurada de modo a mostrar o texto de *links* sublinhado; portanto, o que o usuário verá nesse parágrafo será:

Welcome to Earth! Visitors may also be interested in taking a look at the Moon.

O navegador grava a associação entre o texto exibido do *link* e o URL contido na tag *<A HREF...>* – neste caso:

http://www.cdk5.net/WebExample/moon.html

Quando o usuário clica no texto, o navegador recupera o recurso identificado pelo URL correspondente e o apresenta para o usuário. No exemplo, o recurso está em um arquivo HTML que especifica uma página Web sobre a Lua.

URLs • O objetivo de um URL (Uniform Resource Locator) [www.w3.org III] é identificar um recurso. Na verdade, o termo usado em documentos que descrevem a arquitetura da Web é URI (Uniform Resource Identifier), mas, neste livro, o termo mais conhecido, URL, será usado quando não houver uma possível ambiguidade. Os navegadores examinam os URLs para acessar os recursos correspondentes. Às vezes, o usuário digita um URL no navegador. Mais comumente, o navegador pesquisa o URL correspondente quando o usuário clica em um *link*, quando seleciona um URL de sua lista de *bookmarks* ou quando o navegador busca um recurso incorporado em uma página Web, como uma imagem.

Todo URL, em sua forma completa e absoluta, tem dois componentes de nível superior:

esquema: identificador-específico-do-esquema

O primeiro componente, o "esquema", declara qual é o tipo desse URL. Os URLs são obrigados a identificar uma variedade de recursos. Por exemplo, *mailto:joe@anISP.net* identifica o endereço de *e-mail* de um usuário; *ftp://ftp.downloadIt.com/software/aProg.exe* identifica um arquivo que deve ser recuperado com o protocolo FTP (File Transfer Protocol), em vez do protocolo mais comumente usado, HTTP. Outros exemplos de esquemas são "tel" (usado para especificar um número de telefone a ser discado, o que é particularmente útil ao se navegar em um telefone celular) e "tag" (usado para identificar uma entidade arbitrária).

A Web não tem restrições com relação aos tipos de recursos que pode usar para acesso, graças aos designadores de esquema presentes nos URLs. Se alguém inventar um novo tipo de recurso, por exemplo, *widget* – talvez com seu próprio esquema de endereçamento para localizar elementos em uma janela gráfica e seu próprio protocolo para acessá-los – , então o mundo poderá começar a usar URLs da forma *widget:...* É claro que os navegadores devem ter a capacidade de usar o novo protocolo *widget*, mas isso pode ser feito pela adição de um *plugin*.

Os URLs HTTP são os mais comuns para acessar recursos usando o protocolo HTTP padrão. Um URL HTTP tem duas tarefas principais a executar: identificar qual servidor Web mantém o recurso e qual dos recursos está sendo solicitado a esse servidor. A Figura 1.7 mostra três navegadores fazendo pedidos de recursos, gerenciados por três servidores

Web. O navegador que está mais acima está fazendo uma consulta em um mecanismo de busca. O navegador do meio solicita a página padrão de outro *site*. O navegador que está mais abaixo solicita uma página Web especificada por completo, incluindo um nome de caminho relativo para o servidor. Os arquivos de determinado servidor Web são mantidos em uma ou mais subárvores (diretórios) do sistema de arquivos desse servidor, e cada recurso é identificado por um nome e um caminho relativo (*pathname*) para esse servidor.

Em geral, os URLs HTTP são da seguinte forma:

http:// nomedoservidor [:porta] [/nomedeCaminho] [?consulta][#fragmento]

onde os itens entre colchetes são opcionais. Um URL HTTP completo sempre começa com o *string http*://, seguido de um nome de servidor, expresso como um nome DNS (Domain Name System) . Opcionalmente, o nome DNS do servidor é seguido do número da porta em que o servidor recebe os pedidos (consulte o Capítulo 4) – que, por padrão, é a porta 80. Em seguida, aparece um nome de caminho opcional do recurso no servidor. Se ele estiver ausente, então a página Web padrão do servidor será solicitada. Por fim, o URL termina, também opcionalmente, com um componente de consulta – por exemplo, quando um usuário envia entradas de um formulário, como a página de consulta de um mecanismo de busca – e/ou um identificador de fragmento, que identifica um componente de um determinado recurso.

Considere os URLs a seguir:

http://www.cdk5.net
http://www.w3.org/standards/faq.html#conformance
http://www.google.com/search?q=obama

Eles podem ser decompostos, como segue:

Server DNS name	*Path name*	*Query*	*Fragment*
www.cdk5.net	(default)	(none)	(none)
www.w3.org	standards/faq.html	(none)	intro
www.google.com	search	q=obama	(none)

O primeiro URL designa a página padrão fornecida por *www.cdk5.net*. O seguinte identifica um fragmento de um arquivo HTML cujo nome de caminho é *standards/faq.html*, relativo ao servidor *www.w3.org*. O identificador do fragmento (especificado após o caractere # no URL) é *conformance* e um navegador procurará esse identificador de fragmento dentro do texto HTML, após ter baixado o arquivo inteiro. O terceiro URL especifica uma consulta para um mecanismo de busca. O caminho identifica um programa chamado de "search" e o *string* após o caractere ? codifica um *string* de consulta fornecido como argumento para esse programa. Discutiremos os URLs que identificam recursos de programa com mais detalhes a seguir, quando considerarmos os recursos mais avançados.

Publicação de um recurso: Embora a Web tenha um modelo claramente definido para acessar um recurso a partir de seu URL, os métodos exatos para publicação de recursos dependem da implementação do servidor. Em termos de mecanismos de baixo nível, o método mais simples de publicação de um recurso na Web é colocar o arquivo correspondente em um diretório que o servidor Web possa acessar. Sabendo o nome do servidor, *S*, e um nome de caminho para o arquivo, *C*, que o servidor possa reconhecer, o usuário constrói o URL como *http://S/C*. O usuário coloca esse URL em um *link* de um documento já existente ou informa esse URL para outros usuários de diversas formas como, por exemplo, por *e-mail*.

É comum tais preocupações ficarem ocultas dos usuários, quando eles geram conteúdo. Por exemplo, os *bloggers* normalmente usam ferramentas de *software*, elas próprias implementadas como páginas Web, para criar coleções de páginas de diário organizadas. As páginas de produtos do *site* de uma empresa normalmente são criadas com um *sistema de gerenciamento de conteúdo*; novamente, pela interação direta com o *site*, por meio de páginas Web administrativas. O banco de dados, ou sistema de arquivos, no qual as páginas de produtos estão baseadas é transparente.

Por fim, Huang *et al*. [2000] fornece um modelo para inserir conteúdo na Web com mínima intervenção humana. Isso é particularmente relevante quando os usuários precisam extrair conteúdo de uma variedade de equipamentos, como câmeras, para publicação em páginas Web.

HTTP • O protocolo HyperText Transfer Protocol [www.w3.org IV] define as maneiras pelas quais os navegadores e outros tipos de cliente interagem com os servidores Web. destacaremos aqui suas principais características (restringindo nossa discussão à recuperação de recursos em arquivos):

Interações requisição-resposta: o protocolo HTTP é do tipo requisição-resposta. O cliente envia uma mensagem de requisição para o servidor, contendo o URL do recurso solicitado. O servidor pesquisa o nome de caminho e, se ele existir, envia de volta para o cliente o conteúdo do recurso em uma mensagem de resposta. Caso contrário, ele envia de volta uma resposta de erro, como a conhecida mensagem *404 Not Found*. O protocolo HTTP define um conjunto reduzido de operações ou *métodos* que podem ser executados em um recurso. Os mais comuns são GET, para recuperar dados do recurso, e POST, para fornecer dados para o recurso.

Tipos de conteúdo: os navegadores não são necessariamente capazes de manipular todo tipo de conteúdo. Quando um navegador faz uma requisição, ele inclui uma lista dos tipos de conteúdo que prefere – por exemplo, em princípio, ele poderia exibir imagens no formato GIF, mas não no formato JPEG. O servidor pode levar isso em conta ao retornar conteúdo para o navegador. O servidor inclui o tipo de conteúdo na mensagem de resposta para que o navegador saiba como processá-lo. Os *strings* que denotam o tipo de conteúdo são chamados de tipos MIME e estão padronizados na RFC 1521 [Freed e Borenstein 1996]. Por exemplo, se o conteúdo é de tipo *text/html*, então um navegador interpreta o texto como HTML e o exibe; se o conteúdo é de tipo *image/GIF*, o navegador o representa como uma imagem no formato GIF; se é do tipo *application/zip*, dados compactados no formato zip, o navegador ativa um aplicativo auxiliar externo para descompactá-lo. O conjunto de ações a serem executadas pelo navegador para determinado tipo de conteúdo pode ser configurado, e os leitores devem verificar essas configurações em seus próprios navegadores.

Um recurso por requisição: os clientes especificam um recurso por requisição HTTP. Se uma página Web contém, digamos, nove imagens, o navegador emite um total de 10 requisições separadas para obter o conteúdo inteiro da página. Normalmente, os navegadores fazem vários pedidos concorrentes para reduzir o atraso global para o usuário.

Controle de acesso simplificado: por padrão, qualquer usuário com conectividade de rede para um servidor Web pode acessar qualquer um de seus recursos publicados. Se for necessário restringir o acesso a determinados recursos, isso pode ser feito configurando o servidor de modo a emitir um pedido de identificação para qualquer cliente que o solicite. Então, o usuário precisa provar que tem direito de acessar o recurso, por exemplo, digitando uma senha.

Páginas dinâmicas • Até aqui, descrevemos como os usuários podem publicar páginas Web e outros tipos de conteúdos armazenados em arquivos na Web. Entretanto, grande parte da experiência dos usuários na Web é a interação com serviços, em vez da simples recuperação de informações. Por exemplo, ao adquirir um item em uma loja *online*, o usuário frequentemente preenche um *formulário* para fornecer seus dados pessoais ou para especificar exatamente o que deseja adquirir. Um formulário Web é uma página contendo instruções para o usuário e elementos de janela para entrada de dados, como campos de texto e caixas de seleção. Quando o usuário envia o formulário (normalmente, clicando sobre um botão no próprio formulário ou pressionando a tecla *return*), o navegador envia um pedido HTTP para um servidor Web, contendo os valores inseridos pelo usuário.

Como o resultado do pedido depende da entrada do usuário, o servidor precisa *processar* a entrada do usuário. Portanto, o URL, ou seu componente inicial, designa um *programa* no servidor, e não um arquivo. Se os dados de entrada fornecidos pelo usuário forem um conjunto de parâmetros razoavelmente curto, então normalmente eles são enviados como o componente de *consulta* do URL, usando o método GET; alternativamente, eles são enviados como dados adicionais no pedido, usando o método POST. Por exemplo, um pedido contendo o URL a seguir ativa um programa chamado "search" no endereço www.google.com e especifica o *string* de consulta "obama": *http://www.google.com/search?q=obama.*

Esse programa "search" produz texto em HTML na saída, e o usuário vê uma listagem das páginas que contêm a palavra "obama". (O leitor pode inserir uma consulta em seu mecanismo de busca predileto e observar o URL exibido pelo navegador, quando o resultado for retornado.) O servidor retorna o texto em HTML gerado pelo programa, exatamente como se tivesse sido recuperado de um arquivo. Em outras palavras, a diferença entre o conteúdo estático recuperado a partir de um arquivo e o conteúdo gerado dinamicamente é transparente para o navegador.

Um programa que os servidores Web executam para gerar conteúdo para seus clientes é referido como programa CGI (Common Gateway Interface). Um programa CGI pode ter qualquer funcionalidade específica do aplicativo, desde que possa analisar os argumentos fornecidos pelo cliente e produzir conteúdo do tipo solicitado (normalmente, texto HTML). Frequentemente, o programa consulta ou atualiza um banco de dados no processamento do pedido.

Código baixado: um programa CGI é executado no servidor. Às vezes, os projetistas de serviços Web exigem que algum código relacionado ao serviço seja executado pelo navegador no computador do usuário. Em particular, código escrito em Javascript [www.netscape.com] muitas vezes é baixado com uma página contendo um formulário para proporcionar uma interação de melhor qualidade com o usuário, em vez daquela suportada pelos elementos de janela padrão do protocolo HTML. Uma página aprimorada com Javascript pode dar ao usuário retorno imediato sobre entradas inválidas (em vez de obrigar o usuário a verificar os valores no servidor, o que seria muito mais demorado).

O código Javascript pode ser usado para atualizar partes do conteúdo de uma página Web, sem a busca de uma nova versão inteira da página e sem sua reapresentação. Essas atualizações dinâmicas ocorrem devido a uma ação do usuário (como um clique em um *link* ou em um botão de opção) ou quando o navegador recebe novos dados do servidor que forneceu a página Web. Neste último caso, como a temporização da chegada de dados é desconectada de qualquer ação do usuário no próprio navegador, ela é chamada de *assíncrona*, em que é usada uma técnica conhecida como *AJAX* (*Asynchronous Javascript And XML*).

Uma alternativa para um programa em Javascript é um *applet*: um aplicativo escrito na linguagem Java que o navegador baixa e executa automaticamente ao buscar a página

Web correspondente. Os *applets* podem acessar a rede e fornecer interfaces de usuário personalizadas. Por exemplo, às vezes, os aplicativos de bate-papo são implementados como *applets* que são executados nos navegadores dos usuários, junto a um programa servidor. Nesse caso, os *applets* enviam o texto dos usuários para o servidor, o qual, por sua vez, o redistribui para os demais *applets* para serem apresentados aos outros usuários.

Serviços Web (Web services) • Até aqui, discutimos a Web do ponto de vista de um usuário operando um navegador. No entanto, outros programas, além dos navegadores, também podem ser clientes Web; na verdade, o acesso por meio de programas aos recursos da Web é muito comum.

Entretanto, sozinho, o padrão HTML é insuficiente para realizar interações por meio de programas. Há uma necessidade cada vez maior na Web de trocar dados de tipos estruturados, mas o protocolo HTML não possui essa capacidade – ele está limitado a realizar a navegação em informações. O protocolo HTML tem um conjunto estático de estruturas, como parágrafos, e elas estão restritas às maneiras de apresentar dados aos usuários. A linguagem XML (Extensible Markup Language) (consute a Seção 4.3.3) foi projetada como um modo de representar dados em formas padronizadas, estruturadas e específicas para determinado aplicativo. Em princípio, os dados expressos em XML são portáveis entre os aplicativos, pois são *autodescritivos* – eles contêm os nomes, os tipos e a estrutura dos seus elementos. Por exemplo, XML pode ser usada para descrever os recursos de dispositivos ou informações sobre usuários para muitos serviços ou aplicações diferentes. Na Web, ela é usada para fornecer e recuperar dados de recursos em operações POST e GET do protocolo HTTP, assim como de outros protocolos. No AJAX, ela é usada para fornecer dados para programas Javascript em navegadores.

Os recursos Web fornecem operações específicas para um serviço. Por exemplo, na loja eletrônica amazon.com, as operações do serviço Web incluem pedidos de livros e verificação da situação atual de um pedido. Conforme mencionamos, o protocolo HTTP fornece um conjunto reduzido de operações aplicáveis a qualquer recurso. Isso inclui principalmente os métodos GET e POST em recursos já existentes e as operações PUT e DELETE, para criar e excluir recursos Web, respectivamente. Qualquer operação em um recurso pode ser ativada com o método GET ou POST, com o conteúdo estruturado para especificar os parâmetros da operação, os resultados e as respostas a erros. A assim chamada arquitetura REST (REpresentational State Transfer) para serviços Web [Fielding 2000] adota essa estratégia com base em sua capacidade de extensão: todo recurso na Web tem um URL e responde ao mesmo conjunto de operações, embora o processamento das operações possa variar bastante de um recurso para outro. O outro lado da moeda dessa capacidade de extensão pode ser a falta de robustez no funcionamento do *software*.

Discussão sobre a Web • O sucesso fenomenal da Web baseia-se na relativa facilidade com que muitas fontes individuais e organizacionais podem publicar recursos, na conveniência de sua estrutura de hipertexto para organizar muitos tipos de informação e no fato de sua arquitetura ser um sistema aberto. Os padrões nos quais sua arquitetura está baseada são simples e foram amplamente publicados desde seu início. Eles têm permitido a integração de muitos tipos novos de recursos e serviços.

O sucesso da Web esconde alguns problemas de projeto. Primeiramente, seu modelo de hipertexto é deficiente sob alguns aspectos. Se um recurso é excluído ou movido, os assim

chamados *links* "pendentes" para esse recurso ainda podem permanecer, causando frustração para os usuários. E há o conhecido problema de usuários "perdidos no hiperespaço". Frequentemente, os usuários ficam confusos, seguindo muitos *links* distintos, referenciando páginas de um conjunto de fontes discrepantes e, em alguns casos, de confiabilidade duvidosa.

Os mecanismos de busca são uma alternativa para localizar informações na Web, mas são imperfeitos para obter o que o usuário pretende especificamente. Um modo de tratar desse problema, exemplificado no Resource Description Framework [www.w3.org V], é produzir vocabulários, sintaxe e semântica padrões para expressar metadados sobre as coisas de nosso mundo e encapsular esses metadados nos recursos Web correspondentes para acesso por meio de programas. Em vez de pesquisar palavras que ocorrem em páginas Web, os programas de busca podem então, em princípio, realizar pesquisas nos metadados para compilar listas de *links* relacionados com base na correspondência semântica. Coletivamente, esse emaranhado de recursos de metadados vinculados é conhecido como *Web semântica*.

Como uma arquitetura de sistema, a Web enfrenta problemas de escalabilidade. Os servidores Web mais populares podem ter muitos acessos (*hits*) por segundo e, como resultado, a resposta para os usuários pode ser lenta.