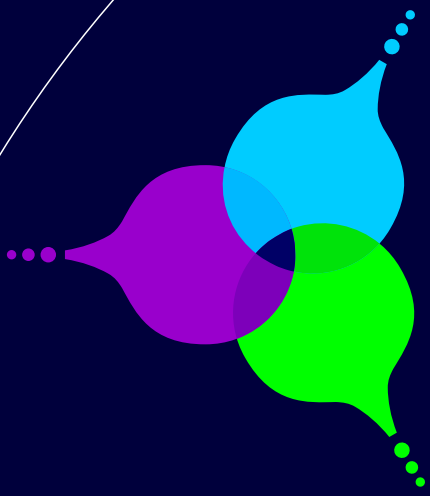


GUIA DE
WEB SEMÂNTICA

nic.br cgi.br

Embaixada Britânica
Brasília

GREAT
BRITAIN

GOVERNO DO ESTADO
SÃO PAULO

GUIA DE
WEB SEMÂNTICA

Este Guia é parte integrante do projeto de cooperação entre o Governo do Estado de São Paulo e o Reino Unido

SPUK

GREAT BRITAIN

Improving business environment through transparency in São Paulo State

Melhoria do ambiente de negócios por meio da transparência no Estado de São Paulo

Parceiros

ceweb.br nic.br cgi.br

SEADE
Fundação Sistema Estadual de Análise de Dados

Fundap

GREAT BRITAIN

Embaixada Britânica Brasília

GOVERNO DO ESTADO SÃO PAULO
Secretaria de Governo

Ano 2015

Improving business environment through transparency in São Paulo State

Melhoria do ambiente de negócios por meio da transparência no Estado de São Paulo

REALIZAÇÃO

GOVERNO DO ESTADO DE SÃO PAULO

Secretaria de Governo
- Subsecretaria de Parcerias e Inovação

Casa Civil
- Assessoria Especial para Assuntos Internacionais

Fundação do Desenvolvimento Administrativo - Fundap
Fundação Sistema Estadual de Análise de Dados – Seade
Conselho de Transparência da Administração Público

GOVERNO DO REINO UNIDO

Embaixada Britânica – Brasília

NÚCLEO DE INFORMAÇÃO E COORDENAÇÃO DO PONTO Br – NIC.br

Centro de Estudos sobre Tecnologia Web – CeWeb.br

AUTOR

Carlos Laufer

COORDENAÇÃO

Geral:
Roberto Agune - iGovSP
Vagner Diniz – CeWeb.br

Executiva e Editorial:
Caroline Burle dos Santos Guimarães - CeWeb.br
Helena Pchevuzinske - iGovSP
Sergio Pinto Bolliger - iGovSP

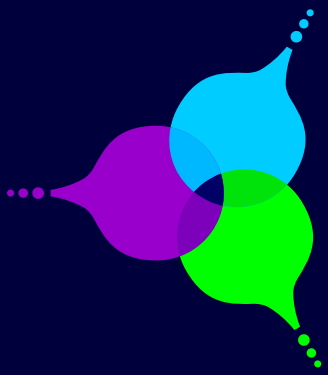
IDEALIZAÇÃO

Alvaro Gregório - iGovSP

DESIGN

Alcione de Godoy - iGovSP - e-books
Deblyn Pereira Prado - NIC.br - HTML
Ricardo Hurmus - Bulerías Arte & Design - ilustrações

SUMÁRIO



INTRODUÇÃO

Capítulo 1

A Web Semântica é uma teia de informações construída de forma a ser facilmente processável por máquinas em uma escala global. A ideia geral é a de criar uma maneira eficiente para representar dados na World Wide Web de forma a construir um banco global de dados conectados.

Este guia tem como objetivo fornecer uma base dos conceitos da Web Semântica, além de exemplos de alguns ambientes de desenvolvimento e implementação de aplicações para a manipulação de dados publicados na Web. O documento foi redigido de forma a fazer o leitor compreender esse novo universo em torno da enorme quantidade de dados publicados diariamente na Web, e a ideia de como descrever essas informações de forma que tanto as pessoas como as máquinas possam compreendê-las. O leitor com menos conhecimentos técnicos pode se concentrar apenas nos capítulos 2 e 3, que têm o objetivo de traçar um panorama e desenhar esse momento da Web. Os capítulos subsequentes entram em mais detalhes sobre a infraestrutura técnica proposta para a descrição dos dados publicados.

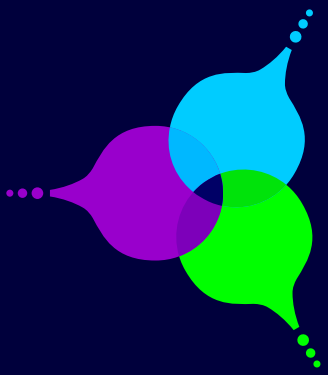
O capítulo 2 apresenta os conceitos que definem a World Wide Web, mostrando a evolução pela qual a Web tem passado, desde a sua invenção, começando como uma Web de Documentos, passando por uma ampliação na direção de uma plataforma universal de execução de aplicações, até chegar à extensão atual como uma Web de Dados. Além disso, nesse capítulo são introduzidos os conceitos de semântica e metadados. O capítulo seguinte apresenta o ecossistema que abrange os Dados na Web, onde são apresentados os conceitos de papéis desempenhados por diferentes tipos de participantes desse ambiente, assim como o ciclo de vida dos dados e a arquitetura subjacente a esse ecossistema.

A partir do capítulo 4 o guia começa uma descrição mais técnica, apresentando as tecnologias e os padrões que definem a Web Semântica, seguido do capítulo que apresenta as bases do conceito de Dados Conectados, com seu princípios, seu esquema de classificação e alguns exemplos. O Capítulo 6 apresenta uma parte fundamental para que se alcance o objetivo de fazer com que um significado pretendido pelo publicador dos dados seja o mesmo que o significado entendido pelo consumidor dos dados: o uso de vocabulários que possuam uma semântica bem definida. Nesse capítulo são apresentados alguns dos vocabulários de referência mais utilizados para a definição semântica dos dados e para a definição de ontologias de domínios específicos.

Finalmente, é apresentada uma conclusão do guia, seguida de um apêndice

que apresenta alguns exemplos de softwares atualmente utilizados em ambientes de desenvolvimento para a geração de aplicações que manipulam Dados Conectados.

O objetivo deste guia não é se aprofundar tecnicamente em cada um dos itens apresentados mas, sim, trazer o leitor para esse novo universo de dados. O leitor interessado no aprofundamento técnico tem ao seu dispor um conjunto de mais de 100 links para especificações e documentos relacionados. Todas as referências deste guia estão disponíveis na Web. Boa Viagem!



WWW E SEMÂNTICA

Capítulo 2

Hoje em dia, para uma grande parcela da população mundial, a World Wide Web é algo tão presente quanto a televisão. A Web é um local onde as pessoas consultam informações, fazem compras, trabalham, conversam, estabelecem relacionamentos, etc. A Web é algo cotidiano, uma coisa óbvia. Qual a necessidade de explicar o que é a Web para uma criança de 10 anos nascida em uma grande metrópole? Para quem nasceu no século 21, a Web é a Web, assim como a luz é a luz. Apesar das enormes transformações efetuadas no modo de vida das pessoas, a partir da utilização da Web, ela ainda é muito recente, em constante e intensa transformação. Mas o que é a Web?

2.1 WEB DE DOCUMENTOS

Em termos técnicos, a Web é um sistema que foi inventado por Tim Berners-Lee e Robert Cailliau [1], em 1989, com o objetivo de utilizar a Internet para a consulta e a atualização de itens de informação (documentos) organizados em uma estrutura hipertextual. Hipertexto é um texto estruturado, composto por um conjunto interligado de itens de informação (nós) contendo, por exemplo, textos, imagens, vídeos, etc. A arquitetura desse sistema foi criada com base no conceito cliente-servidor, onde uma aplicação (um cliente Web) requisita um documento (um recurso) a uma outra aplicação (um servidor Web) informando a identificação desse documento.

Esse sistema é composto por três elementos básicos, definidos por meio de padrões internacionais:

- URL (Unifor m Resource Locator) [2] é o identificador dos documentos, dos nós da estrutura. Por exemplo “http://pt.wikipedia.org/wiki/World_Wide_Web” identifica o documento que descreve a Web no site da Wikipedia em português.

- HTML (Hypertext Markup Language) [3] é a linguagem de marcação para a descrição dos documentos. Ela foi inicialmente projetada como uma linguagem para a descrição de documentos científicos, mas adaptações efetuadas ao longo do tempo permitiram que ela seja utilizada para descrever diversos outros tipos de documentos.

- HTTP (Hypertext Transfer Protocol) [4] é o protocolo de comunicação para acesso aos documentos. Esse protocolo estabelece as regras da comunicação entre duas aplicações, um cliente e um servidor, de forma a

permitir a requisição de um conteúdo e o respectivo resultado retornado por essa requisição, por exemplo, uma página HTML.

O sistema funciona da seguinte maneira: uma aplicação-cliente, por exemplo, um navegador Web, requisita um documento ao servidor, identificando esse documento por meio de uma URL, e informando que esse documento deve ser preferencialmente descrito em HTML. Essa URL contém, também, a identificação do servidor que será responsável por gerenciar a requisição do documento. A comunicação entre o cliente e o servidor se estabelece por meio do protocolo HTTP. Uma vez feita a requisição de um documento pelo navegador, o servidor identificado pela URL devolve o documento desejado na forma de uma cadeia de caracteres formatada na linguagem HTML. Uma vez recebido o documento em HTML, o navegador é responsável por interpretar o código HTML e exibir as informações na tela. A ideia original desse sistema era a de que além de visualizar as informações contidas em um determinado documento, o usuário também pudesse alterar as informações desses documentos, algo semelhante ao que ocorre hoje em sites como a Wikipedia.

Além desses três elementos, um outro elemento importante nesse sistema é o *link* que pode ser estabelecido entre os diversos documentos, um *hyperlink* em um hipertexto. Os *links* fazem a interligação entre os diversos nós da estrutura. Um *link* pode ser incluído dentro do código de uma determinada página, utilizando um *tag* específico da linguagem HTML (o *tag* <a href="*url-do-link*">), onde é possível especificar uma URL que aponta para um outro documento. Dessa forma temos uma rede, uma teia, uma Web de documentos interligados, um hipertexto. O documento, quando visualizado pelas pessoas em um navegador, é referenciado como sendo uma página da Web.

Resumindo, cada documento da Web pode ser acessado por meio de uma URL, podendo esse documento estar interligado a outros documentos por meio de *links*: um conjunto de documentos interligados por *links*. Essa é a Web de documentos.

O conteúdo de uma página exibida por um navegador pode conter diversas informações diferentes que serão utilizadas pelas pessoas. Podemos ter uma página, por exemplo, com informações sobre diversos produtos em promoção de uma determinada loja que vende produtos eletrodomésticos. Essas informações são descritas de forma a ser entendidas por seres humanos. O que em uma página Web é entendido por máquinas?

O navegador, ou outras aplicações que são executadas na Internet, não conseguem identificar o que está descrito nessas páginas da mesma forma que as pessoas conseguem entender. O navegador não entende o significado do texto contido nas páginas. O que um navegador consegue entender é a semântica da linguagem HTML, que contém formas de indicar, por exemplo, que uma determinada frase é um título, um *tag* específico da linguagem HTML, o *tag* <h1>, por exemplo. Portanto, podemos dizer que uma determinada frase, "A Web de Documentos", é um título: "<h1>A Web de Documentos</h1>". O que o navegador consegue entender é a relação entre o texto "A Web de Documentos" e o *tag* <h1>, e a partir daí ter uma forma de exibir esse título dentro de um determinado estilo. Para o navegador é indiferente qual é o texto contido entre os *tags* <h1> e </h1>. Ele não entende o texto. Pode ser qualquer texto. O navegador não extrai nenhuma informação do texto em si.

2.2 WEB PROGRAMÁVEL

O ambiente inicial da Web era composto, basicamente, por um universo de documentos estáticos mantidos em arquivos nos servidores Web, que eram requisitados pelos navegadores para ser exibidos aos usuários. Apesar de uma URL poder simplesmente apontar para um arquivo, também é possível que um servidor Web faça algo além de identificar o arquivo e enviá-lo de volta ao cliente. Ele pode executar um código de programa e, a partir dessa computação, retornar um conteúdo gerado dinamicamente por esse programa. Nesse caso, a página HTML retornada pelo servidor é a saída do programa executado a partir da requisição identificada pela URL.

Cada servidor Web executa um software específico para atendimento às requisições HTTP. Geralmente, um servidor HTTP tem um diretório, uma pasta, que é designada como uma coleção de documentos, arquivos que podem ser enviados em resposta às requisições dos navegadores. Esses arquivos são identificados a partir das URLs associadas às requisições.

Common Gateway Interface (CGI) é um método padrão usado para gerar conteúdo dinâmico em páginas e aplicativos da Web. Ele fornece uma interface entre o servidor Web e programas que geram o conteúdo dinâmico da Web. Esses programas são geralmente escritos em uma linguagem de script, mas podem ser escritos em qualquer linguagem de programação.

A Web evoluiu de um espaço simples para a exibição de páginas contidas em documentos estáticos, para um espaço onde diversos tipos de aplicações utilizam os navegadores como plataformas para a execução de programas. Hoje em dia, é possível fazer compras, executar procedimentos bancários, enviar mensagens, e uma infinidade de outras aplicações, a partir da utilização dos navegadores. Diversos ambientes de programação surgiram com o intuito de facilitar a criação e a execução dessas aplicações, como, por exemplo, ASP, ASP. NET, JSPJava, PHP, Perl, Python, Ruby on Rails, etc.

Aplicações são geralmente estruturadas em blocos lógicos chamados de camadas, onde para cada camada é atribuído um papel. A estrutura mais comum nas aplicações Web é a de três camadas: apresentação, lógica de negócios e armazenamento. Um navegador da Web é a primeira camada (apresentação). Um motor, usando alguma tecnologia de conteúdo dinâmico da Web (JSPJava, Python, etc.), é a camada intermediária (lógica de negócios). Um banco de dados é o terceiro nível (armazenamento). O navegador envia solicitações para a camada intermediária, que executa os serviços fazendo consultas e atualizações no banco de dados, gerando, então, a resposta na forma de uma interface de usuário.

A disseminação das aplicações Web demandou a necessidade de comunicação entre aplicações, de forma a possibilitar a troca de dados e serviços, o que fez surgir a ideia de Web Services e Web APIs, como implementações do conceito de componentes de software no ambiente Web. Web Services fornecem uma forma padrão de interoperação entre aplicações de software diferentes. É um sistema de software projetado para suportar interações máquina-máquina interoperáveis através de uma rede. Ele tem uma interface descrita em um formato processável por máquina (WSDL). Outros sistemas interagem com o Web Service usando mensagens descritas na sua interface, usando um protocolo específico (SOAP). Web APIs têm uma definição menos restritiva em relação à formatação dos dados na comunicação entre as aplicações, e utilizam um outro protocolo de comunicação (REST).

A ideia geral dessas duas tecnologias é possibilitar que aplicações possam fornecer serviços a ser consumidos por outras aplicações, o que resulta numa outra camada dentro do ambiente da Web. Nessa camada existe um tráfego de dados sendo trocados por aplicações, que podem ser manipulados, combinados e transformados, dependendo das tarefas ofertadas por cada aplicação Web, para, então, ser apresentados aos usuários.

Como forma de ilustrar a ideia de componentes, podemos pegar o exemplo de uma loja que faz vendas pela Internet e que precisa informar ao usuário qual o valor do frete dos produtos comprados. Em geral, o frete é feito por uma empresa terceirizada. Para que o site de vendas possa informar esse valor, a aplicação do site pode obter esses dados a partir de um serviço (Web Service ou Web API) oferecido pela empresa que fará o frete, e exibir esse valor na interface apresentada ao usuário. Esse tráfego de dados, entre a aplicação do site de vendas e o serviço Web oferecido pela empresa de frete, é invisível para o comprador dos produtos. O comprador só percebe a comunicação entre ele e o próprio *site* de vendas.

Um dos problemas que se apresentam nessa arquitetura de componentes Web é a de que o significado e o formato dos dados não seguem nenhum padrão, e são especificados por cada um dos serviços da forma que estes acharem mais conveniente. Dessa forma, uma aplicação que queira combinar dados de diversos componentes precisa saber cada uma das definições, e, no caso de querer fazer alguma integração desses dados, saber como interpretar as diferentes definições, de forma a identificar as semelhanças e as diferenças entre os diversos dados retornados pelos múltiplos componentes. Para cada novo serviço que se queira utilizar, é preciso entender a sua semântica e a forma como ela é descrita. A interoperabilidade semântica de diferentes serviços e seus dados tem que ser realizada de forma manual. O site do jornal The New York Times oferece um conjunto de mais de 10 diferentes APIs [5] para acesso a seus dados, cada uma com sua própria especificação e formato de dados. O site ProgrammableWeb [6] contém um catálogo com milhares de aplicações disponíveis com especificações das mais variadas e heterogêneas, e ilustra bem a diversidade existente no mundo da Web Programável.

2.3 WEB DE DADOS

Páginas da Web, exibidas por um navegador, contêm um conjunto de informações que são consumidas por pessoas. São textos, fotos, vídeos, etc., dispostos na página, de forma que uma pessoa possa extrair um significado dessas informações. Essas informações agrupam, em geral, um conjunto de dados que têm algum relacionamento entre si e que, por algum motivo, faz sentido apresentá-los em uma única página, um documento único. A partir da requisição de uma URL, o servidor Web identifica quais os dados serão retornados para o navegador.

Em uma apresentação de Tim Berners-Lee [7] no "Open, Linked Data for a Global Community", ele usa um pacote de salgadinhos como exemplo da diversidade de informações que existe na embalagem de um produto: fatos nutricionais, composição química, selos de qualidade, identificação por código de barras, formas de contato com a empresa fabricante, etc. Essas informações são descritas em vocabulários específicos que demandam um conhecimento prévio para o seu entendimento. Por exemplo, é preciso saber ler uma tabela de fatos nutricionais, entender que a tabela é referente a uma porção definida e que são listados percentuais relativos às necessidades diárias ideais para o consumo humano. Nessa embalagem, existem diversos dados diferentes agrupados em um único documento. Na realidade, podem existir muito mais informações disponíveis relativas a esse produto, mas por uma decisão que envolve diversos critérios, que podem incluir espaço disponível na embalagem, grau de importância da informação, etc., apenas algumas informações são listadas na embalagem. Muitas vezes, existe um endereço da Web impresso na embalagem que aponta para um local onde podem ser obtidas informações adicionais.

Considerando um exemplo da Web, em uma página de uma loja de vendas podem ser exibidas informações de diversos produtos diferentes, o endereço físico da loja, um telefone de atendimento ao cliente, etc. Todos esses dados são expostos em um único documento que é apresentado ao usuário, utilizando textos e recursos gráficos (cor, tipo e tamanho de letra, etc.) em uma disposição espacial dentro da página, de forma a comunicar as informações ao usuário. É um processo de comunicação. Um processo que entende que o receptor dessa mensagem é um ser humano. Como vimos anteriormente, o modelo inicial da Web entende essa rede como um conjunto de documentos interligados em uma estrutura de hipertexto. Quando é feita uma requisição a um servidor Web, este identifica qual o conjunto de dados que serão agrupados em uma determinada página. O acesso direto aos dados de forma individual não é permitido.

E se ao invés de considerarmos esses documentos como blocos estanques de dados, pensássemos em uma Web que pudesse permitir o acesso individual a todos os dados que são agrupados nessas páginas e, além dessa Web de Documentos, pudéssemos ter acesso a uma Web de Dados, onde cada um dos nós da Web não fosse mais, necessariamente, um documento, mas um determinado dado específico, um determinado recurso. Dessa forma, teríamos acesso a uma camada de granularidade mais fina da Web, e diferentes

desenvolvedores poderiam criar aplicações que agrupassem esses dados de diferentes formas. No caso do exemplo do pacote de salgadinhos, uma determinada aplicação poderia listar um outro conjunto de dados relacionados ao produto sob uma perspectiva que achasse mais conveniente, de acordo com um outro critério, por exemplo, quanto a uma alimentação para hipertensos.

Máquinas de busca são uma das aplicações mais populares da Web. A Google se transformou em uma das maiores empresas do planeta a partir da necessidade de conectar os dados publicados com os possíveis consumidores desses dados: um intermediário no processo de comunicação. Para prover esse trabalho, as páginas da Web são varridas e analisadas por robôs para que a Google possa montar um banco de dados que consiga responder à procura requisitada por uma pessoa, da forma mais precisa possível. Os motores de busca têm uma compreensão limitada do que está sendo discutido nessas páginas Web. Considerando que existem diversos dados distribuídos em cada página, são necessários algoritmos que tentam extrair esses dados a partir de uma informação formatada para seres humanos. Como identificar todos esses dados sem uma indicação específica que possa ser entendida por máquinas? Como formatar informações que possam ser úteis para que máquinas entendam o significado das informações contidas em uma página Web? Como criar novas formas de distribuição desses dados?

As seções a seguir esclarecem as noções de semântica e metadados que são utilizadas para se alcançar a ideia de uma Web de Dados entendida por seres humanos e por máquinas.

2.4 SEMÂNTICA

Neste guia estamos interessados em esclarecer a ideia de definir uma camada semântica ao modelo inicial da Web de Documentos. Como vimos anteriormente, a ideia inicial da Web foi a de servir como uma forma de navegação entre documentos dispostos em uma estrutura de hipertexto. Esses documentos são exibidos aos usuários por aplicações que interpretam a linguagem HTML. O conteúdo das páginas é visto pelas máquinas de uma forma apenas sintática. A interpretação da informação, propriamente dita, é feita pelas pessoas que visualizam as páginas. Qual a semântica das informações exibidas? Qual é o significado dessas informações?

Semântica, em linguística, é o estudo do significado que é utilizado para

entender a expressão humana por meio da linguagem. Nós conseguimos entender o significado, por exemplo, desta frase, entendendo o significado de cada uma das palavras e as relações entre essas palavras dentro da frase. Entendemos, também, os sinais de pontuação como a vírgula, o ponto, etc., e a função deles no texto. Além disso, podemos entender imagens e códigos de cores e uma diversidade de signos codificados de diferentes formas. Uma informação pode, por exemplo, ser exibida em caracteres grandes e na cor vermelha, para indicar a necessidade de atenção com aquele texto. Tudo isso depende de diversos fatores, incluindo a cultura da pessoa que recebe a informação. O vermelho, por exemplo, tem significado diferente em culturas asiáticas.

Modelos de comunicação são modelos conceituais utilizados para explicar o processo de comunicação humana (figura 2.1). O primeiro modelo importante para a comunicação veio em 1949 e foi concebido por Claude Shannon e Warren Weaver, dos Laboratórios Bell. A comunicação é o processo de transferência de informações de uma parte (transmissor) para outra (receptor). O modelo inicial de Shannon e Weaver consistia de três partes principais: transmissor, canal e receptor. Em um modelo simples, a informação (por exemplo, uma mensagem em linguagem natural) é enviada de alguma forma (como língua falada) a partir de um transmissor/remetente/codificador para um receptor/destinatário/decodificador, por meio de um canal . Essa concepção de comunicação comum vê a comunicação como um meio de enviar e receber informações.

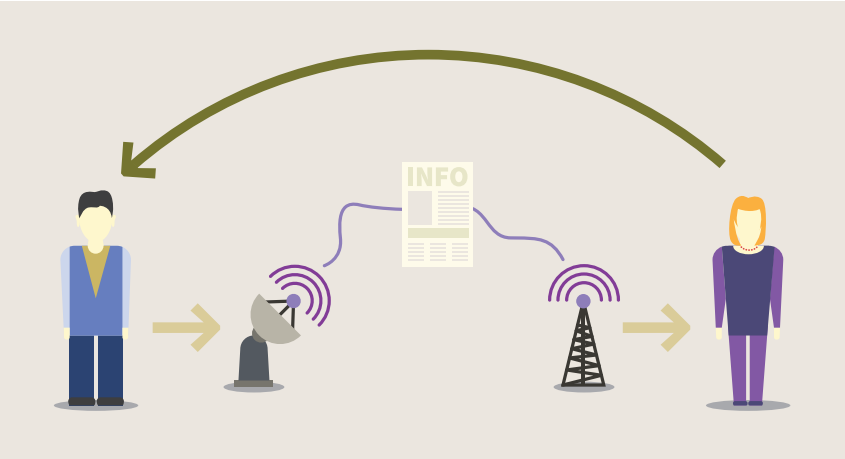


Figura 2.1 – Comunicação entre pessoas

As figuras 2.2 e 2.3 ilustram como uma informação embutida dentro de um modelo mental de uma pessoa é codificada para depois ser transmitida por meio de um canal e, então, ser decodificada e mapeada para o modelo mental de uma outra pessoa.

Neste guia estamos tratando da introdução de semântica para máquinas que acessam as informações. Como uma máquina pode interpretar o conteúdo de uma página Web? Como fazer para codificar uma informação de forma a fazer o mapeamento correto entre o significado pretendido e o significado entendido?

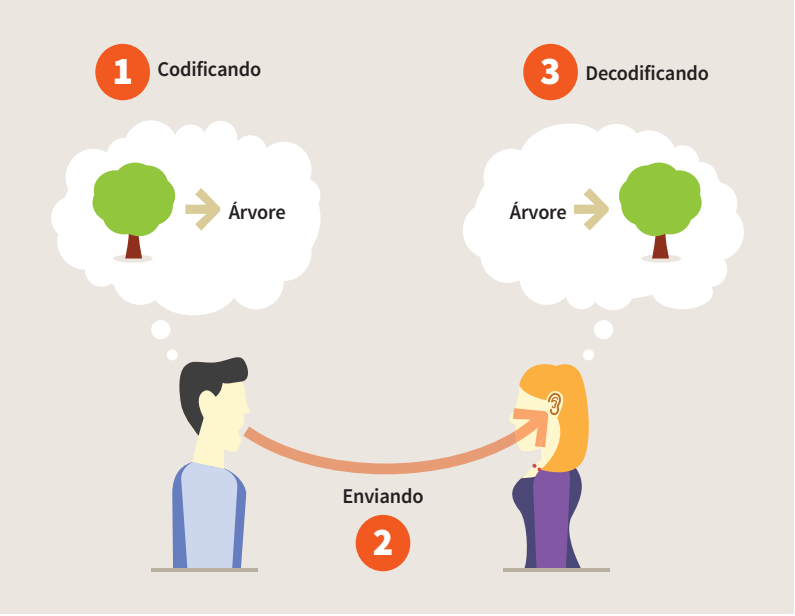


Figura 2.2 – Modelo mental

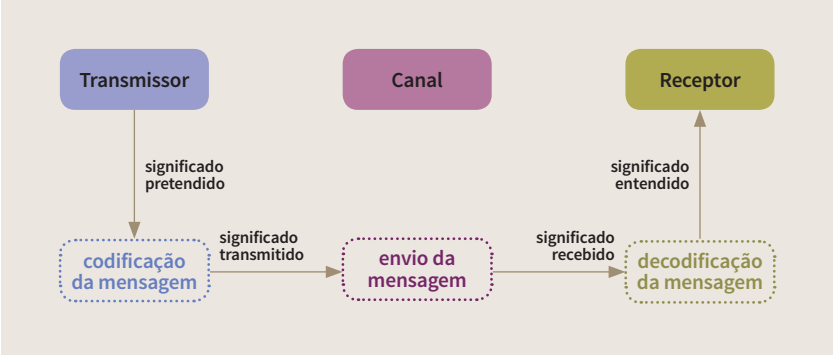


Figura 2.3 – A transmissão do significado

Tomemos o exemplo dos *sites* de comparação de preços. Assim que surgiram os sites de venda na Web, que anunciavam um conjunto de produtos em suas páginas, com informações que incluíam a descrição do produto, imagens,

preço, etc., surgiu a ideia da construção de sites que pudessem apresentar ao usuário a comparação de preços de um mesmo produto, ofertado em diferentes sites de vendas. Sites de comparação de preços podem ser considerados os primeiros exemplos de Web Semântica, mas os primeiros sistemas desenvolvidos nesses sites utilizavam softwares específicos (scrapers) para extrair informações estruturadas sobre os produtos a partir de páginas da Web. A informação é extraída a partir da identificação de padrões sintáticos dentro das páginas HTML.

HTML é uma linguagem de marcação de texto que define, basicamente, a estrutura de um texto. Em conjunto com uma outra linguagem (CSS), que define o estilo do texto, o navegador combina a estrutura e o estilo, e exibe as informações para as pessoas. Para que um scraper entenda o significado de um texto, é preciso procurar padrões que de alguma forma indiquem um significado. Vejamos um exemplo de um site de compras brasileiro, onde podemos identificar um padrão <tipo-preço>: uma cadeia de caracteres da forma "xxx,xx" ao lado dos caracteres "R$". Não existe dentro da linguagem HTML um *tag* que indique explicitamente que essa cadeia de caracteres seja um preço. Nós humanos reconhecemos facilmente que essa informação é um preço. Num exemplo um pouco mais sofisticado, podemos ter o seguinte texto "Google Chromecast HDMI Streaming De: R$ 249,00 Por: R$ 192,72", ao lado de uma imagem. Podemos escrever um código que procure pelas palavras "De" e "Por", próximas de números que podem ser identificados no padrão <tipo-preço>, e entender que é uma promoção de produto. Definimos dessa forma um padrão <tipo-promoção>. E assim por diante. É fácil perceber que mudanças na disposição e no agrupamento das informações podem requerer uma reprogramação do scraper. A construção de um scraper requer extensa programação e é um sistema bastante instável, uma vez que pode haver a necessidade de reprogramação toda vez que uma loja online muda a estrutura de suas informações.

Como vimos na seção 2.3 (Web de Dados), a ideia é que possamos identificar de forma individual, legível por máquinas, cada um dos dados agrupados nas páginas Web. Para isso é necessário que coloquemos informações extras sobre esses dados, dentro do código HTML, informações que serão consumidas por máquinas. Essas informações sobre os dados são chamadas de metadados.

2.5 METADADOS

Metadados são dados sobre dados. Eles fornecem informações adicionais sobre os dados, para ajudar desenvolvedores de aplicações e usuários finais a entender melhor o significado dos dados publicados, o seu conteúdo, a sua estrutura. Metadados são também utilizados para esclarecer outras questões relacionadas ao conjunto de dados como, por exemplo, a licença de uso, a empresa/organização que gerou os dados, a qualidade dos dados, a proveniência, como fazer o acesso, a frequência de atualização do conjunto de informações, etc. Os metadados têm como objetivo ajudar o processo de comunicação entre os publicadores e os consumidores de dados, para que os consumidores entendam todas as questões pertinentes para a utilização desses dados.

Metadados podem ser utilizados para auxiliar tarefas como, por exemplo, a descoberta e a reutilização do conjunto de dados, e podem ser atribuídos de acordo com diferentes granularidades, que vão desde uma única propriedade de um recurso (uma coluna de uma tabela) a um conjunto de dados completo, ou todos os conjuntos de dados de uma determinada empresa/organização. Um exemplo bem simples de metadados, de uso bastante corriqueiro, são os nomes das colunas de uma tabela colocadas na primeira linha de um arquivo em formato CSV. A função desses metadados é permitir que um leitor dos dados desse arquivo CSV entenda o significado de cada um dos campos, dos dados, de cada linha.

Até os dias de hoje, a Web se desenvolveu mais rapidamente como um meio de transmissão de documentos para as pessoas, ao invés de dados e informações que possam ser processados automaticamente. Metadados devem estar disponíveis em formas legíveis tanto para seres humanos quanto para máquinas. É importante proporcionar ambas as formas de metadados, a fim de alcançar os seres humanos e as aplicações. No caso de metadados legíveis por máquinas, a utilização de vocabulários de referência deve ser incentivada, como uma forma de reforçar uma semântica comum.

No exemplo da tabela CSV, é possível perceber a dificuldade que pode ocorrer quando não se utiliza um vocabulário comum de referência para descrever um metadado. Cada organização, cada pessoa, pode utilizar um termo diferente para designar o nome das colunas de uma tabela, que pode ser entendido dentro de uma determinada empresa, mas pode ter um significado ambíguo para pessoas diferentes em empresas diferentes, e, muitas vezes, dentro de

uma mesma empresa. Alguns vocabulários de referência, de uso mais popular, são apresentados no capítulo 6. Por exemplo, os dados poderiam ter a sua proveniência descrita usando PROV-O, uma recomendação W3C que fornece um conjunto de classes, propriedades e restrições que podem ser usadas para representar e trocar informações de procedência geradas em sistemas diferentes e sob diferentes contextos.

Metadados podem ser de diferentes tipos. Esses tipos podem ser classificados em diferentes taxonomias, agrupados por diferentes critérios. Por exemplo, uma taxonomia específica poderia definir metadados segundo características descritivas, estruturais e administrativas. Metadados descritivos servem para identificar um conjunto de dados, metadados estruturais servem para entender o formato em que o conjunto de dados é distribuído e metadados administrativos servem para fornecer informações sobre versão, frequência de atualização, etc. Uma outra taxonomia poderia definir tipos de metadados com um esquema que considerasse as tarefas em que os metadados são utilizados, por exemplo, a descoberta e a reutilização de dados.

Metadados podem estar embutidos em páginas Web, mesclados (dentro do código HTML) com as informações a ser apresentadas aos usuários. Dessa forma, parte do código HTML é direcionada ao consumo humano e outra parte é direcionada ao consumo por máquinas. A seção 4.5 apresenta tecnologias utilizadas para embutir metadados em páginas Web. Além disso, metadados podem estar armazenados em catálogos que mantêm informações de dados publicados na Web. Os metadados podem também ser consumidos a partir de implementações que utilizam as tecnologias ligadas a Web Semântica e a Dados Conectados, apresentados nos capítulos 4 e 5.

Uma das primeiras formas de se incluir metadados em uma página Web foi por meio da utilização do *tag* <meta> da linguagem HTML. Esse *tag* têm dois atributos onde é possível se definir um nome e um conteúdo. Um dos primeiros usos do *tag* <meta> foi como forma de comunicação entre os publicadores das páginas e os robôs das máquinas de busca que fazem a varredura dessas páginas. Esses robôs leem as páginas com o intuito de gerar um índice que sirva de base para as respostas às requisições de busca dos usuários. Uma das diversas informações que o *tag* <meta> pode comunicar aos robôs é se uma página deve ou não ser incluída nos índices das máquinas de

busca. Dessa forma é possível que um publicador possa informar aos robôs que não deseja aparecer em resultados de busca:

```
<meta name="robots" content="noindex" />
```

Um outro uso muito intenso do *tag* <meta> foi feito para que as máquinas de busca tivessem uma forma de indexar as páginas segundo um conjunto de palavras-chave definidas pelo publicador. Por exemplo:

```
<meta name="keywords" content="guia, web semântica" />
```

Como humanos, e entendedores da língua inglesa, podemos intuir que existe um conjunto de palavras-chave associadas à página onde esse *tag* foi incluído. Porém, a semântica do *tag* <meta> não define nenhuma interpretação específica para os atributos “name” e “content”. Essas informações são interpretadas pelas aplicações que leem as páginas de acordo com uma semântica que se estabeleceu com o uso. No caso do exemplo, um *tag* <meta> com ‘name=”keywords”’ é interpretado pelos robôs das máquinas de busca indicando que o campo “content” terá uma lista de palavras-chave que serão associadas ao conteúdo da página. Para que essas convenções, como a de que “keywords” significa um conjunto de palavras-chave, é necessário que sejam utilizados vocabulários de referência que tenham o seu significado compreendido pelo publicador, de forma que a aplicação que consome a página possa ter o significado entendido igual ao significado pretendido.

Essa forma de incluir metadados tem uma expressividade bastante limitada e vem sendo estendida a partir da criação de padrões para a definição de metadados e por um conjunto de vocabulários de referência, que são descritos nas próximas seções desse guia.

ECOSSISTEMA DE DADOS NA WEB

Capítulo 3

3.1 ATORES E PAPÉIS

A Web de Documentos é basicamente utilizada por dois tipos de usuários: publicadores e consumidores. O primeiro navegador criado por Tim Berners-Lee permitia não só o acesso a um documento, por meio da especificação de uma URL, como também permitia que o usuário editasse a informação e a gravasse, nos moldes da Wikipedia. Dessa forma, uma mesma pessoa, um mesmo ator desse ecossistema, podia ter dois papéis: consumidor e publicador de dados. Porém, como regra geral, temos esses dois papéis desempenhados por grupos de pessoas diferentes: um conjunto de pessoas com a tarefa de publicar documentos e um conjunto de pessoas com o interesse em consumir esses documentos. Esses dois papéis, por sua vez, podem ser especializados em conjuntos de papéis mais específicos.

Para ilustrar a diversidade de papéis em uma ambiente de publicação de dados, vamos utilizar uma analogia entre a Web e o mundo dos livros. Nesse mundo existem dois tipos básicos de papéis: os escritores e os leitores. Para que um leitor possa ter acesso a um texto criado por um escritor, é preciso que esse escritor publique o livro. Teremos então um leque de novos papéis, todos agrupados em um grande papel, que podemos denominar publicadores. Entre os publicadores teremos, além do próprio escritor, pessoas responsáveis por garantir que, a partir de um texto criado por uma pessoa, esse texto possa ser impresso e distribuído a fim de chegar aos leitores. São necessárias pessoas que cuidem da infraestrutura de publicação, da divulgação, da distribuição, designers para a definição da capa, do tipo de papel, da fonte, formatos de publicação, etc., até chegar às livrarias, para a venda ao leitor, o consumidor do livro. É fácil perceber que existem diferentes atores desempenhando diferentes papéis com diferentes competências ao longo desse processo. Do lado do leitor, o consumidor, poderíamos também identificar dois tipos de leitores: um leitor apenas interessado em consumir as informações do livro e um leitor que poderia também ser um publicador. Este estaria interessado em gerar uma nova publicação a partir das informações consumidas no livro.

No mundo da Web, também podemos considerar esses dois grandes grupos: os publicadores e os consumidores de dados. Do lado dos publicadores podemos listar uma série de diferentes atores com diferentes papéis, relacionados à publicação de informações que muitas vezes são definidas por regras ou procedimentos que fogem ao escopo das funções do criador ou do publicador efetivo dos dados. Esse conjunto de papéis pode envolver diferentes atores

dentro de uma empresa ou de um órgão de governo, onde diferentes setores podem ser responsáveis pela definição das variadas informações e processos. Podemos citar, entre outros, a definição de licenças, a criação de regras de definição do formato das URIs (um conceito mais abrangente que URL), a escolha dos formatos dos dados e das plataformas para a distribuição das informações, qual o conjunto obrigatório de metadados e documentos, a definição de estratégias para garantir a persistência, a preservação e o arquivamento dos dados, etc.

Do lado do consumidor dos dados podemos identificar pessoas interessadas no consumo direto dos dados (usuários finais), e pessoas interessadas em transformar os dados e agregar algum valor para, então, fazer a publicação de um outro conjunto de dados. Esse segundo grupo de pessoas é formado por desenvolvedores de aplicações intermediárias entre os usuários e esse novo conjunto de dados. Muitas vezes os usuários finais fazem a agregação, de forma manual, de diversos conjuntos diferentes de dados para a obtenção de um outro resultado desejado, em uma tarefa não atendida por nenhuma aplicação específica. A combinação de dados é algo infinito e, dependendo da demanda de um tipo específico de combinação, pode ser útil que se desenvolva uma aplicação particular que possa automatizar esse processo e facilitar o trabalho dos usuários finais.

Na seção seguinte, ciclo de vida dos dados na Web, é possível perceber o conjunto de diferentes papéis e competências necessárias para a realização das diversas tarefas relacionadas à publicação dos dados.

3.2 CICLO DE VIDA

Ciclo de vida de dados é o processo de gerenciamento de informações desde a sua fase de seleção até a fase de arquivamento. Esse processo envolve a participação de diferentes profissionais, com competências específicas referentes a cada uma das fases. Como vimos na seção anterior, são diversos papéis que são especializados dentro do grupo de publicadores e consumidores de dados.

Existem diversos modelos que visam capturar todas as fases do ciclo de vida e o relacionamento entre elas. Algumas das fases extraídas desses modelos são listadas a seguir.

• Planejamento − é a fase inicial onde se faz o planejamento da publicação dos dados, o que inclui a seleção dos dados a ser publicados, a identificação de equipes e órgãos que devem participar do processo e as opções de coleta e plataformas de publicação.

• Estruturação − é fase onde é definida a estrutura dos dados a ser distribuídos. Por exemplo, no caso de tabelas, quais os campos que serão expostos e as características desses campos. No caso da utilização de Web Semântica e Dados Conectados, que ontologias serão utilizadas para as instâncias dos dados (seção 5). É também nessa fase que uma nova ontologia será definida, caso seja necessária.

• Criação e Coleta − é a fase de aquisição dos dados propriamente ditos, que incluem dados a ser criados e dados já existentes, obtidos a partir de planilhas, bancos de dados, APIs, Web Services, etc.

• Refinamento e Enriquecimento (Transformação e Integração) − é a fase onde os dados são trabalhados de forma a melhorar a sua qualidade, filtrando possíveis incorreções, agregando ou separando informações e fazendo eventuais conexões com dados relativos a outras bases

• Formatação − é a fase onde os dados a ser publicados são formatados de acordo com a plataforma escolhida para publicação

• Descrição (Metadados, Documentação) − é a fase onde são definidos, criados e coletados os metadados e os documentos que devem ser agregados aos dados, de forma a auxiliar o entendimento das informações.

• Publicação − é a fase em que os dados são efetivamente colocados na Web, na plataforma escolhida para publicação

• Consumo − é a fase onde os dados são consumidos por usuários finais ou por desenvolvedores de aplicações

• Realimentação − é a fase onde são recolhidas informações relacionadas à utilização dos dados, que podem vir, por exemplo, dos consumidores dos dados ou das plataformas de distribuição.

• Arquivamento − é a fase onde os dados são retirados da Web.

3.3 ARQUITETURA

Na fase inicial da Web, a tarefa básica requisitada a um servidor Web era a de obtenção de uma página estática (identificada na cadeia de caracteres da URL), armazenada em um arquivo, e codificada em HTML. Com a expansão do desenvolvimento de aplicações na Web Programável, essa URL passou a ser utilizada para requisitar a realização de outros tipos de tarefas, como, por exemplo, um pedido de informações sobre preços de um produto em diferentes sites de vendas. Essas informações são construídas de forma dinâmica pela aplicação executada no servidor Web, que pode compilar os dados sobre os preços desse produto a partir das diferentes formas de coleta de dados listadas acima. Em um outro exemplo, a tarefa requisitada poderia ser a transferência de uma quantia em dinheiro de uma conta bancária para outra. Nesse caso, a tarefa requisitada não seria uma tarefa de obtenção de uma página de dados. Seu retorno, provavelmente, sinalizaria o sucesso ou o fracasso na realização da transferência de fundos.

A estrutura de funcionamento da Web é baseada no modelo cliente-servidor, onde temos uma aplicação requisitando a realização de alguma tarefa para uma outra aplicação. O acesso aos dados do lado do servidor é sempre intermediado por uma aplicação. Mesmo a requisição de uma página HTML estática em um servidor, é exposta por uma aplicação do lado do servidor que atende à requisição efetuada pela aplicação cliente. Portanto, qualquer dado que é consumido por uma aplicação executada no lado do cliente, um navegador, por exemplo, tem uma aplicação como intermediário, que tem acesso aos dados e o retorna para o requisitante.

Entre os muitos avanços e tecnologias que têm sido agregados ao funcionamento da Web, três são interessantes em ser ressaltadas, no histórico da manipulação de dados: JavaScript, XMLHttpRequest e Ajax. Como já foi dito, no início da Web todos os dados contidos em uma página exibida por um navegador eram retornados no código HTML resultante do processamento efetuado pelo servidor Web após a requisição de uma URL. Não existia nenhum tipo de programação executada no lado do cliente Web, do navegador, a não ser a interpretação do código HTML para a exibição da página ao usuário. Uma vez que uma página tivesse sido carregada e exibida, qualquer tipo de processamento de dados que fosse necessário tinha que ser feito pelo servidor, fruto de uma nova requisição. Por exemplo, um erro em um campo de formulário só era detectado após o envio dos dados para o

servidor e o retorno de uma nova página completa, muitas vezes praticamente igual à página anterior, apenas com uma mensagem adicional informando os erros detectados.

Novas requisições de páginas introduziam todo um tempo de retardo devido à necessidade de uso da rede para uma nova comunicação e transferência dos dados. A introdução da capacidade de executar código do lado do cliente, por meio de uma linguagem inicialmente denominada LiveScript, logo renomeada JavaScript, permitiu que uma série de manipulações pudessem ser resolvidas sem haver a necessidade de uma requisição ao servidor. Com a capacidade de processar código do lado do cliente, procedimentos como, por exemplo, a consistência de campos de formulários passaram a ser executados sem a necessidade de novas requisições. Com a sofisticação das novas especificações do código HTML, a parte do design das páginas também passou por uma mudança, com uma forma mais rica de apresentação, com a ideia de páginas dinâmicas em HTML (DHTML), onde o código JavaScript embutido nas páginas Web passou a manipular a estrutura dos dados apresentados.

Mesmo assim, a programação executada do lado do cliente só podia manipular os dados retornados pelo servidor após a requisição de uma URL. Se fossem necessários mais dados, era preciso requisitar outra página Web pois o código JavaScript executado dentro de uma página não podia se comunicar com o mundo exterior. XMLHttpRequest mudou essa limitação, permitindo que o código JavaScript contido nas páginas Web de um navegador pudesse ter acesso a mais dados do servidor, sempre que necessário.

A Google rapidamente percebeu o potencial dessas novas tecnologias, e aplicações como o Gmail e Google Maps tiraram proveito disso para construir interfaces de usuário mais ricas e parecidas com uma aplicação, uma aplicação Web ao invés de uma página Web. Com o Gmail, por exemplo, a aplicação está continuamente verificando junto ao servidor se há novos e-mails. Se houver, a página é atualizada sem que haja a necessidade de carregar uma nova página. Da mesma forma, o Google Maps permite que uma pessoa inspecione um mapa, e somente as partes necessárias são requisitadas ao servidor.

Esta nova técnica foi denominada Ajax (Asynchronous Javascript And XML) (figura 3.1) em um artigo de Jesse James Garrett [8], e o termo foi imediatamente adotado. A técnica passou a ser usada de forma ampla e surgiram diversos toolkits em JavaScript que tornaram seu uso ainda mais fácil e intuitivo.

Esse modelo de construção de uma aplicação a partir da junção de dados pode ser vista como sendo a de uma aplicação fazendo a requisição dos dados a partir da especificação de URLs e tendo como retorno, não mais um código HTML, mas um conjunto de dados que serão manipulados pela aplicação e apresentados ao usuário na forma adequada, dependendo da tarefa requerida pelo usuário. Podemos replicar esse modelo para a construção de aplicações dentro do ambiente de dados publicados na Web, incluindo os dados publicados segundo os conceitos da Web Semântica

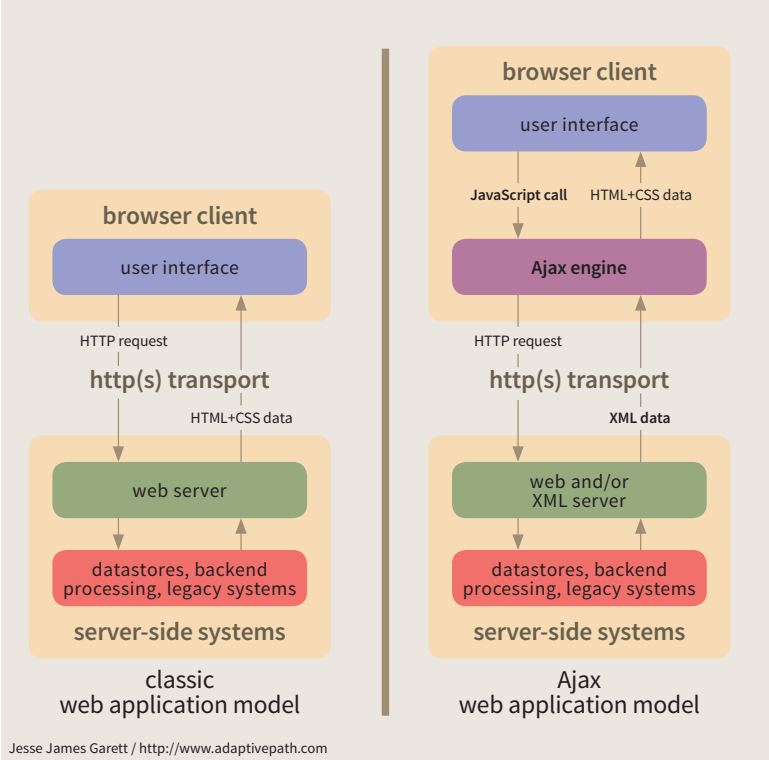


Figura 3.1 – Modelo de aplicações utilizando Ajax

A Web pode ser vista como um conjunto de camadas de dados e camadas de aplicações executadas a partir de requisições identificadas por URLs. Podemos considerar que uma requisição a um servidor é uma requisição de realização de uma tarefa, passando uma URL como informação de entrada a ser interpretada para a execução dessa tarefa e para o retorno dos resultados. Os dados resultantes de uma requisição são sempre expostos por alguma aplicação, que pode manipular dados provenientes de arquivos, de bancos de dados, dados extraídos de páginas da Web, dados resultantes de chamadas a Web Services ou chamadas a Web APIs. No caso mais simples, e que deu

início à Web, a aplicação do lado do servidor é apenas um servidor de páginas que estão armazenadas em uma estrutura de diretório. Em uma configuração mais dinâmica, essa aplicação pode executar um código que acesse dados de um banco de dados e os retorne formatados em HTML. Num cenário mais sofisticado, essa aplicação pode se comportar como cliente de outras aplicações, e retornar dados resultantes da manipulação específica que faz desses dados, obtidos a partir de requisições efetuadas a outras aplicações (Web Services e Web APIs).

Sob a perspectiva das máquinas, podemos desenhar a Web de Dados como uma camada de dados consumidos por uma camada de aplicações. Os dados gerados por essas aplicações, por sua vez, podem ser utilizados como fontes de dados por outras aplicações. Teremos, então, uma nova camada de dados (gerados por aplicações) a ser consumidos por uma nova camada de aplicações. E assim por diante. Temos, assim, um universo de camadas de dados e de aplicações que podem construir novas camadas de forma infinita, com cada aplicação oferecendo um conjunto específico de tarefas. As interpretações dos dados e de seus possíveis significados são de responsabilidade das aplicações, resultantes das relações que estas fazem a partir do consumo do universo de dados dispostos nas diversas camadas.

Sob a perspectiva das pessoas, as aplicações devem prover interfaces onde os dados resultantes das tarefas requisitadas sejam apresentados de forma a ter a sua compreensão facilitada, de competência de designers de interface. Por exemplo, no caso de conjuntos extensos de dados, recursos como a apresentação visual de gráficos e tabelas em diferentes formatos podem vir a auxiliar essa compreensão. Não faz parte do escopo desse guia explorar as diversas formas de expor os dados nas interfaces de usuários.

Temos então uma arquitetura onde existem dados publicados de diferentes formas e um conjunto de aplicações que oferecem um conjunto de tarefas e manipulam esse conjunto de dados para gerar seus resultados. A distribuição desses dados é heterogênea e comporta o acesso a dados contidos em páginas Web, downloads de arquivos armazenados em estruturas de diretórios nos servidores, dados retornados por Web Services e Web APIs. Não existe uma padronização em relação à estrutura e a semântica dos dados distribuídos. Essas questões são resolvidas pelas próprias aplicações. O entendimento da estrutura e da semântica dos dados fica embutido nas aplicações.

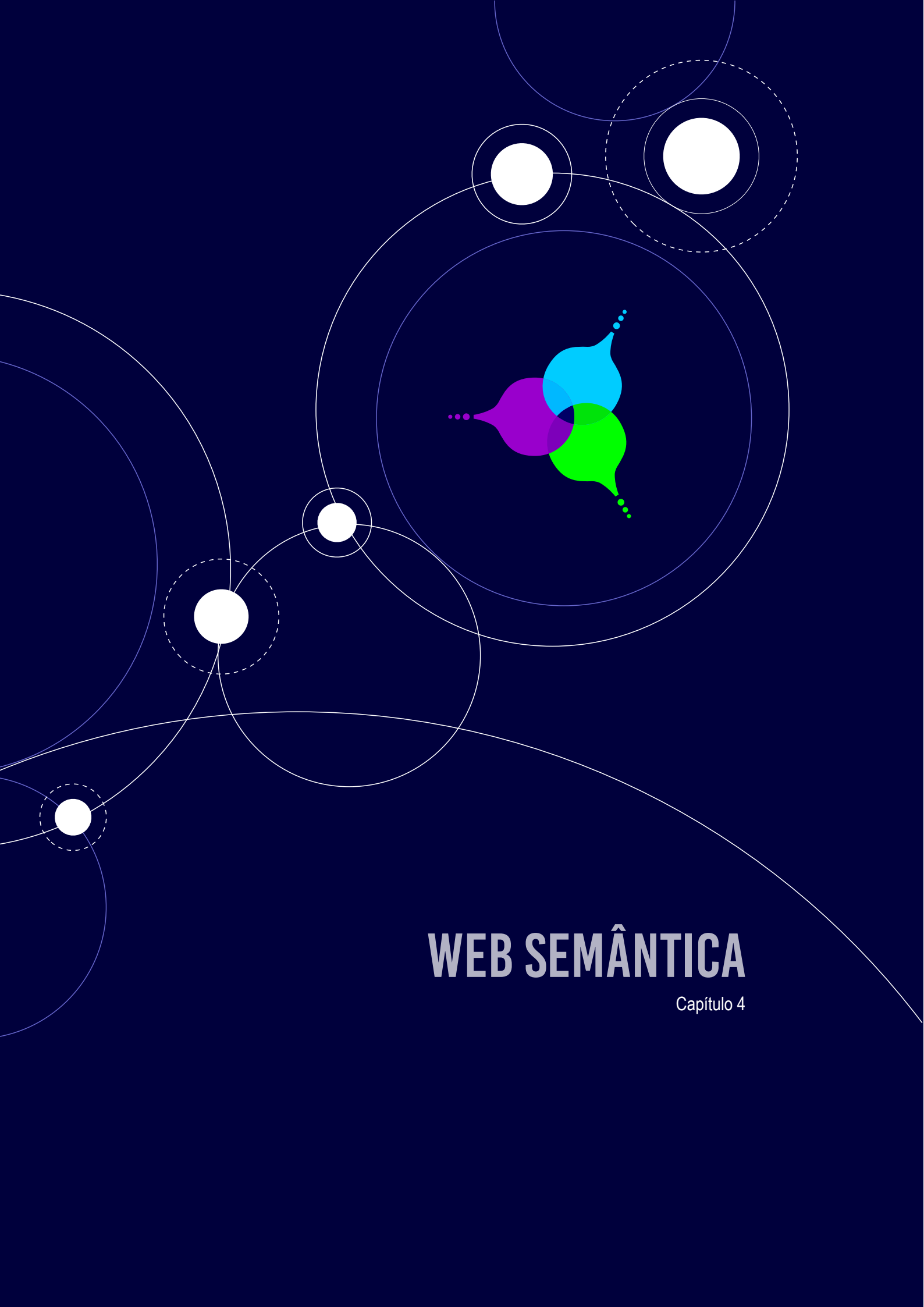
A ideia da Web Semântica é definir um modelo padrão para a representação de dados, combinado com um conjunto de vocabulários de uso comum, permitindo que a semântica dos dados possa ser anexada aos dados, de forma a construir um ambiente mais homogêneo nas camadas de dados a facilitar o trabalho dos desenvolvedores de aplicações.

Como veremos na seção seguinte, a Web Semântica (Dados Conectados) acrescenta a essa arquitetura duas novas formas de distribuição dos dados:

- Dereferenciação de Recursos − os dados, recursos representados no modelo de dados da Web Semântica, são requisitados diretamente por meio de uma URI (seção 4.1.2).

- Endpoint SPARQL − um protocolo de acesso e uma linguagem de consulta que permite uma requisição de um conjunto de recursos a partir da especificação de uma consulta em uma base de dados.

Atualmente, o ambiente da Web, como é de sua característica intrínseca, tem uma estrutura heterogênea em relação à publicação e distribuição dos dados. A parte mais importante em relação à abertura de dados é que estes sejam publicados, mesmo que em formatos proprietários e que exijam mais trabalho dos desenvolvedores e dos consumidores de dados. O que os criadores da Web Semântica almejam é que essas diferentes formas possam convergir com o passar do tempo de modo a construir uma Web de Dados, um banco de dados mundial de informações conectadas. Sobre essa Web de Dados, as aplicações poderão construir o seu conjunto de tarefas a fim de atender as demandas dos usuários. Na seção 5.2 são apresentadas cinco fases nesse caminho da abertura de dados.



WEB SEMÂNTICA

Capítulo 4

Em 2001, Tim Berners-Lee, James Hendler e Ora Lassila publicaram um artigo [9] na revista Scientific American onde lançam as bases da Web Semântica. Nos capítulos anteriores deste guia vimos como a Web de Documentos evoluiu para uma Web Programável, e a crescente oferta de aplicações que executam tarefas que manipulam dados publicados das mais diversas formas nesse ecossistema de dados na Web. A ideia de agregar semântica a esses dados visa a facilitar o entendimento e a interoperabilidade dos dados nesse universo de informações heterogêneas, publicadas nos mais diferentes formatos e com diferentes protocolos de acesso.

Os blocos básicos que definem a Web Semântica são:

- um modelo de dados padrão;
- um conjunto de vocabulários de referência;
- um protocolo padrão de consulta.

A Web Semântica busca facilitar o processo de comunicação entre os diversos participantes do ecossistema, de forma a criar um modelo mental comum, minimizando a possibilidade de ambiguidades, e facilitando, assim, o trabalho necessário para o desenvolvimento de aplicações que manipulem as diversas fontes de dados.

As seções a seguir têm como objetivo apresentar um panorama geral das diferentes tecnologias utilizadas no ambiente da Web Semântica, e suas características em termos de conceitos e o papel de cada uma delas no ecossistema de dados publicados na Web. O objetivo dessas seções não é o detalhamento de cada uma dessas tecnologias, que têm especificações bem definidas e uma série de materiais que as descrevem de uma maneira formal (referenciadas com links neste guia).

4.1 RDF

A Web, originalmente construída para o consumo humano, fornece uma infraestrutura simples e universal para a troca de diversos tipos de informações. Entretanto, existe uma grande dificuldade para que aplicações processem os dados expostos nos documentos sem que seja agregada uma informação específica, codificada em um formato adequado. A solução proposta é usar metadados para descrever os dados publicados na Web, que ajudem a localizar

e a processar informações, fornecendo descrições sobre a estrutura, o conteúdo e outras informações relacionadas (licença de uso, direitos de propriedade intelectual, proveniência, etc.). Para isso, é necessário que se tenha uma base para o processamento dos metadados, de forma a permitir a interoperabilidade dessas descrições entre diferentes aplicações, apoiadas em padrões sobre a sintaxe e a semântica dos metadados, além de um conjunto de vocabulários comuns e formas de acesso padronizadas.

4.1.1. MODELO DE DADOS

Resources Description Framework (RDF) [10] é um arcabouço para representar informações na Web. RDF permite fazer afirmações sobre recursos. Recursos são quaisquer coisas, tanto concretas quanto abstratas. Uma determinada empresa, uma pessoa, uma página Web são considerados recursos. Um sentimento, uma cor, também são recursos.

O formato das afirmações é simples. Uma afirmação RDF consiste de três elementos (uma tripla) e tem a seguinte estrutura: <sujeito> <predicado> <objeto>. Uma afirmação RDF expressa uma relação entre dois recursos. O sujeito e o objeto representam os dois recursos sendo relacionados; o predicado representa a natureza desta relação, que é formulada de modo direcional (do sujeito para o objeto) e é chamada em RDF de propriedade. Um objeto pode também ser um literal, definindo uma propriedade para um recurso.

Como forma de ilustrar a ideia da definição dos dados por meio de um conjunto de triplas, vamos fazer uso de um modelo de dados conhecido: o modelo relacional e o seu conjunto de tabelas relacionadas [11]. Vamos supor uma tabela onde existam informações sobre livros (figura 4.1). Cada linha da tabela tem informações sobre um determinado livro (figura 4.2). Cada um desses livros é um recurso. Cada coluna da tabela define um tipo de propriedade relacionada aos livros (figura 4.3). Cada célula da tabela define uma tripla (figura 4.4).

A tabela contém as informações dos recursos do tipo livro:

isbn	título	autor	editora_id	páginas
9788535912388	Gabriela, Cravo e Canela	Jorge Amado	1243	424
...	...	...	...	...
9788501067340	Vidas Secas	Graciliano Ramos	3244	176
...	...	...	...	...
9788535921199	Antologia Poética	Carlos Drummond de Andrade	1243	344

Figura 4.1 – Informações sobre livros

As linhas representam os recursos:

isbn	título	autor	editora_id	páginas
9788535912388	Gabriela, Cravo e Canela	Jorge Amado	1243	424
...	...	...	...	...
9788501067340	Vidas Secas	Graciliano Ramos	3244	176
...	...	...	...	...
9788535921199	Antologia Poética	Carlos Drummond de Andrade	1243	344

Figura 4.2 – Informações sobre um recurso

As colunas representam as propriedades dos recursos:

isbn	título	autor	editora_id	páginas
9788535912388	Gabriela, Cravo e Canela	Jorge Amado	1243	424
...	...	...	...	...
9788501067340	Vidas Secas	Graciliano Ramos	3244	176
...	...	...	...	...
9788535921199	Antologia Poética	Carlos Drummond de Andrade	1243	344

Figura 4.3 – Propriedades dos recursos

Cada célula da tabela define uma propriedade (coluna) de um recurso (linha).

isbn	título	autor	editora_id	páginas
9788535912388	Gabriela, Cravo e Canela	Jorge Amado	1243	424
...	...	...	...	...
9788501067340	Vidas Secas	Graciliano Ramos	3244	176
...	...	...	...	...
9788535921199	Antologia Poética	Carlos Drummond de Andrade	1243	344

Figura 4.4 – Tripla de um recurso

Uma tripla pode ser representada como uma espécie de grafo dirigido (um grafo RDF), do sujeito para o objeto:

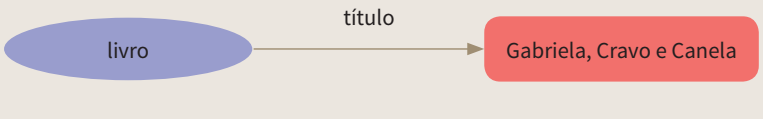


Figura 4.5 – Grafo RDF da tripla de um livro

Generalizando, temos:

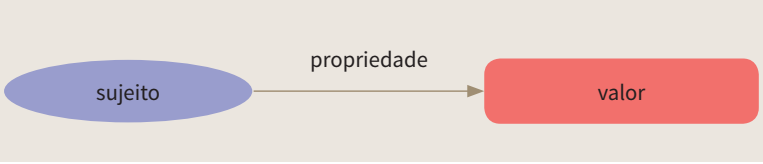


Figura 4.6 – Grafo RDF genérico de uma tripla

A figura 4.8 apresenta o grafo RDF das triplas correspondentes a três propriedades de um mesmo recurso da tabela da figura 4.7:

isbn	título	autor	editora_id	páginas
9788535912388	Gabriela, Cravo e Canela	Jorge Amado	1243	424
...	...	...	...	...
9788501067340	Vidas Secas	Graciliano Ramos	3244	176
...	...	...	...	...
9788535921199	Antologia Poética	Carlos Drummond de Andrade	1243	344

Figura 4.7 – Triplas de um recurso

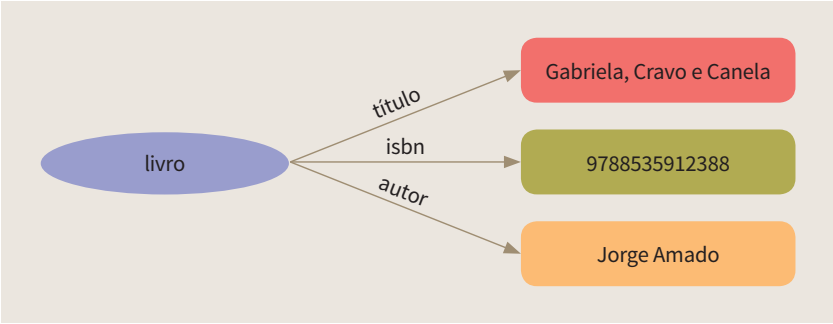


Figura 4.8 – Grafo RDF com três triplas de um mesmo recurso

No caso das triplas acima, todas elas têm um valor literal. Mas um recurso também pode estabelecer uma relação com outro recurso, o que em um esquema de tabelas relacionais seria representado por uma outra tabela, e o uso de chaves estrangeiras. No nosso exemplo, poderíamos ter uma tabela de editoras (publicadores) (figura 4.9).

id	nomes
...	...
1243	Companhia das Letras
...	...
3244	Grupo Editorial Record
...	...

Figura 4.9 – Informações sobre editoras (publicadores)

A figura 4.10 apresenta o grafo RDF considerando a relação de um livro com o seu publicador.

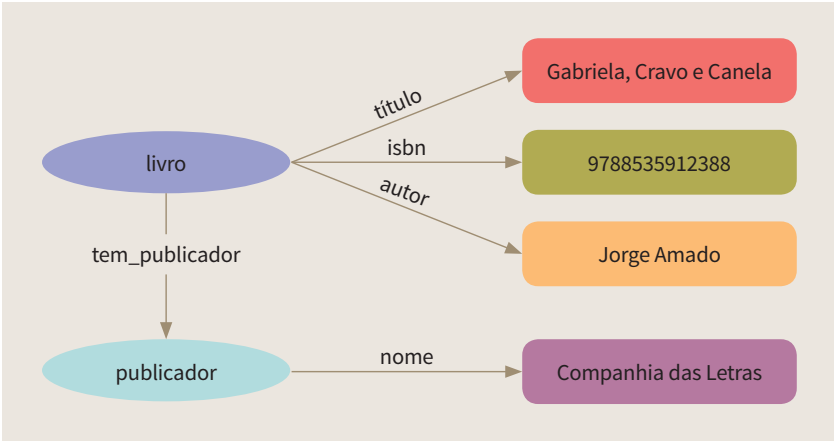


Figura 4.10 – Grafo RDF relacionando recursos

A figura 4.11 apresenta o grafo RDF considerando dois livros do mesmo publicador.

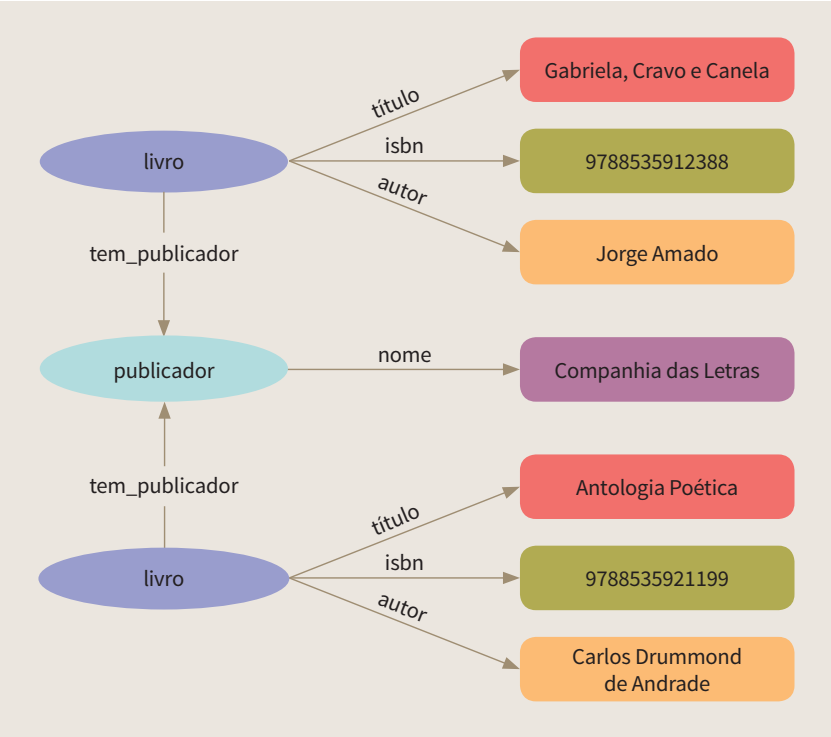


Figura 4.11 – Grafo RDF de vários recursos

4.1.2. URIs

Para completar esse modelo de dados, precisamos ter uma maneira de identificar cada um dos recursos e cada uma das propriedades de forma única e universal, de modo a ter uma semântica global e não apenas particular, algo que seja compreendido não apenas dentro de uma empresa ou organização. O criador de uma tabela, dentro de uma determinada organização, atribui nomes às propriedades de forma particular, como por exemplo, a propriedade "título". Ela poderia ter sido definida como "title", "nome", "nome da obra", "ttl", etc. A princípio, cada organização saberia entender o significado da propriedade a partir de alguma documentação interna da empresa, norma de nomenclatura, cultura de uso dentro da empresa, etc. Mesmo assim, sabemos que muitas vezes isso não ocorre, e pode haver uma ambiguidade nesse significado. Da mesma forma, se considerarmos cada um dos recursos, muitas vezes eles são identificados por meio de identificadores que são únicos, chaves primárias, mas tem o escopo apenas dentro da tabela onde estão inseridos, e dentro do

banco de dados no qual estão armazenados. Um mesmo id, por exemplo, "1243", pode ser utilizado para identificar um publicador em uma tabela, e uma livraria em outra tabela. Eles não são identificadores únicos em termos universais. Eles são identificadores individuais dentro das tabelas de um determinado banco de dados. E mesmo quando, eventualmente sejam únicos em diversas tabelas de uma organização, eles têm que ser únicos considerando todo e qualquer recurso definido por qualquer entidade, e não somente dentro de uma determinada empresa.

Em RDF a forma de identificar tanto os recursos quanto as propriedades, de forma única e universal, se faz por meio da utilização de URIs, mais especificamente, HTTP URIs [12]. URIs são uma forma mais abrangente de URL, pois não estão necessariamente ligadas à localização do recurso. Elas têm o mesmo formato de uma URL, mas são utilizadas com o intuito de identificar as coisas, enquanto uma URL identifica um endereço para a recuperação de uma informação, um documento. Uma pessoa, por exemplo, pode ser identificada por uma URI (figura 4.12).

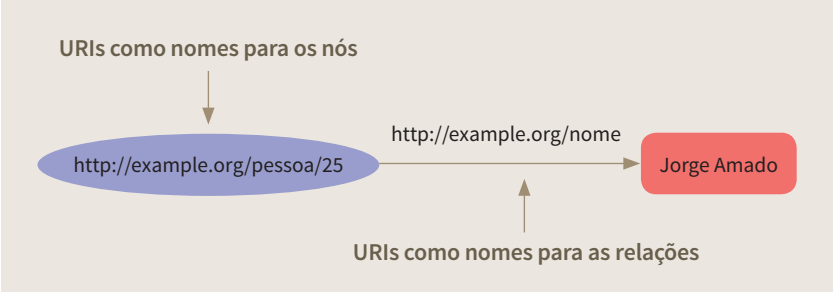


Figura 4.12 – Uso de URIs

Para se ter uma semântica mais conhecida das propriedades podemos utilizar vocabulários de referência, para a definição de propriedades de domínios específicos. Veremos alguns desses vocabulários no capítulo 6. Como exemplo, podemos citar a propriedade "nome", que em geral é definida a partir de uma propriedade do vocabulário FOAF [13], utilizado para a definição de propriedades sobre pessoas. A URI "http://xmlns.com/foaf/0.1/name" identifica a propriedade "nome" (figura 4.13).

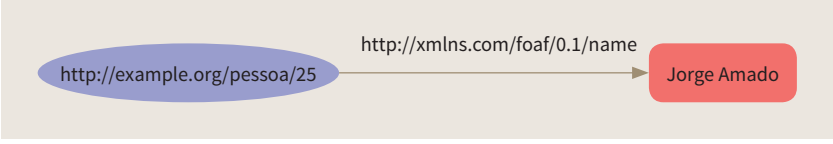


Figura 4.13 – Exemplo de uso de foaf:name

Uma questão muito importante quando da publicação de dados está relacionada ao esquema de identificação dos recursos, o esquema de URIs. Essa é uma discussão em curso no ambiente dos pesquisadores e publicadores de dados na Web Semântica. Existem diversos artigos que têm orientações quanto a esquemas propícios para a definição de URIs. Alguns defendem que a identificação de uma URI deve ser totalmente opaca, ou seja, não deve haver nenhuma informação na URI que possa ser interpretada em relação ao recurso que ela identifica, algo semelhante a uma chave numérica primária de uma tabela relacional. No exemplo de pessoa da figura 4.7, temos de alguma forma essa ideia. Outros defendem que certos esquemas podem especificar algumas informações nas URIs, de forma a fazer com que um usuário possa intuir uma outra URI relacionada, sem que seja necessário recorrer a alguma outra forma de busca. Por exemplo, uma URI com informações sobre o orçamento federal pode conter na sua formação o ano relativo às informações que se deseja: "http://orcamento.dados.gov.br/doc/2013/ItemDespesa". Caso o usuário queira as informações sobre um outro ano, basta substituir o ano na URI.

Um outro cuidado importante que se deve ter quando se define um esquema de nomeação de URIs é garantir que esse esquema seja persistente, ou, pelo menos, dure o máximo de tempo possível. E, caso alguma URI se torne obsoleta, prover uma maneira de informar ao usuário esse fato e como seria possível recuperar essa informação, às vezes disponível em algum outro local identificado por uma outra URI. Existem diversos esquemas que buscam garantir a persistência das URIs, entre eles o DOI System [14] e o PersId Initiative [15].

Essa discussão é bastante abrangente e não está no escopo deste guia esgotar esse assunto. Caso o leitor queira mais informações, os seguintes artigos, entre outros, trazem dados importantes para a decisão quanto ao esquema de definição de URIs: "Cool URIs for the Semantic Web" [16], "Cool URIs don't change" [17].

4.1.3. SERIALIZAÇÕES

RDF é um modelo de dados abstrato, não importando como ele seja representado, desde que a representação permaneça fiel às suas propriedades abstratas. Existem diversas representações sintáticas para o modelo RDF, algumas mais adequadas para o processamento de máquina, outras mais

legíveis para as pessoas.

Entre as notações mais utilizadas podemos citar RDF/XML [18], Turtle [19], N-Triples [20] e JSON-LD [21]. A primeira notação a ser utilizada foi RDF/XML, padronizada pelo W3C, e seu uso inicial tinha como vantagem o fato de as linguagens de programação terem mais suporte para XML. Além disso, também é possível fazer uso de namespaces XML para evitar o uso de URIs completas, o que torna as URIs menos extensas. Porém, a leitura da codificação em RDF/XML é bastante difícil para humanos.

Nesta seção, como forma de ilustrar as serializações, será apresentada uma descrição geral da notação Turtle (Terse RDF Triple Language), que é de mais fácil leitura para as pessoas. A partir do seu entendimento, é possível compreender a ideia das serializações. Os detalhes da sintaxe de cada uma das notações estão descritos nas referências de cada uma das especificações. Para o entendimento das outras notações basta compreender a sintaxe específica de cada uma, pois os conceitos abstratos de todas elas são os mesmos: os conceitos do modelo de dados RDF. Além disso, também será apresentada uma descrição geral de JSON-LD, que tem tido uma grande aceitação na comunidade de desenvolvedores de software.

No exemplo de Turtle, será utilizado o grafo RDF da figura 4.8. Além do vocabulário FOAF, citado anteriormente, utilizaremos três propriedades do vocabulário Dublin Core [22], para descrever o isbn (International Standard Book Number), o criador da obra, e para relacionar a obra com o publicador.

A sintaxe Turtle é formada por sentenças com os três elementos que definem uma afirmação em RDF. Essas sentenças são finalizadas pelo sinal de pontuação ".". A figura 4.14 apresenta as quatro triplas da obra "Gabriela, Cravo e Canela" representadas em Turtle.

```
<http://example.org/#gabriela-cravo-canela>
<http://purl.org/dc/elements/1.1/identifier>
"9788535912388" .

<http://example.org/#gabriela-cravo-canela>
<http://xmlns.com/foaf/0.1/name>
"Gabriela, Cravo e Canela" .

<http://example.org/#gabriela-cravo-canela>
<http://purl.org/dc/elements/1.1/creator>
"Jorge Amado" .
```

Figura 4.14 – Triplas em Turtle da obra "Gabriela, Cravo e Canela"

Como forma de tornar menos extensa a descrição em Turtle, é possível agrupar definições de múltiplas propriedades de um mesmo recurso, informado a URI do recurso uma única vez e utilizando um sinal ";" entre as diversas definições de propriedades (figura 4.15).

```
<http://example.org/#gabriela-cravo-canela>
<http://purl.org/dc/elements/1.1/identifier>
"9788535912388" ;
<http://xmlns.com/foaf/0.1/name>
"Gabriela, Cravo e Canela" ;
<http://purl.org/dc/elements/1.1/creator>
"Jorge Amado" .
```

Figura 4.15 – Sintaxe abreviada de múltiplas propriedades de um mesmo recurso

Como forma de tornar ainda mais legível a definição das triplas, é possível definir um conjunto de prefixos, que são correspondentes aos namespaces dos vocabulários utilizados (figura 4.16).

```
base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<#gabriela-cravo-canela>
    dc:identifier "9788535912388" ;
    foaf:name "Gabriela, Cravo e Canela" ;
    dc:creator "Jorge Amado" .
```

Figura 4.16 – Utilização de *namespaces* em Turtle

A figura 4.17 apresenta a serialização em Turtle do grafo da figura 4.11.

```
base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<#gabriela-cravo-canela>
    dc:identifier "9788535912388" ;
    foaf:name "Gabriela, Cravo e Canela" ;
    dc:creator "Jorge Amado" ;
    dc:publisher <#companhia-das-letras> .

<#antologia-poetica>
    dc:identifier "9788535921199" ;
    foaf:name "Antologia Poética" ;
    dc:creator "Carlos Drummond de Andrade" ;
    dc:publisher <#companhia-das-letras> .

<#companhia-das-letras>
    foaf:name "Companhia das Letras".
```

Figura 4.17 – Serialização em Turtle do grafo da figura 4.11

Uma serialização de RDF que tem tido um crescente uso na Web Semântica é JSON-LD (Javascript Object Notation for Linked Data), um formato de serialização de triplas RDF que tem como objetivo ser uma forma de representação mais legível para as pessoas, utilizando o modelo JSON de conjuntos de pares atributo-valor para representar o conjunto de triplas.

Um do elementos centrais de JSON-LD é a ideia de contexto. Quando duas pessoas se comunicam em um ambiente compartilhado, existe um contexto de conhecimento mútuo que permite que os indivíduos usem termos abreviados, um vocabulário próprio, para se comunicar mais rapidamente, mas sem perder a precisão. Um contexto em JSON-LD funciona da mesma forma. Ele permite que duas aplicações usem termos abreviados, termos particulares, para se comunicar com mais eficiência, mas sem perder a precisão. Essa foi uma necessidade percebida pelos desenvolvedores da especificação de JSON-LD (uma recomendação do W3C), resultante da dificuldade de entendimento das longas URIs que representam os recursos RDF. De uma certa forma é um retorno a um vocabulário mais simplificado, mais particular, compartilhado por uma comunidade de um determinado domínio de aplicações.

As figuras 4.18 a 4.21 ilustram a passagem de uma representação de três pares de atributos-valor em JSON, sem uma referência semântica universal,

até a especificação de um código JSON-LD, onde os mesmos três pares têm associados a si um contexto que define a semântica universal dos termos.

A figura 4.18 apresenta três propriedades de uma pessoa, em formato JSON:

- "name" – nome (literal)
- "homepage" – página pessoal (URL)
- "image" – imagem (URL)

```
{
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Figura 4.18 – Definição de dados em JSON

A figura 4.19 substitui os termos (do vocabulário particular da comunidade) pelas URIs das propriedades dos vocabulários de referência. Por exemplo, "name" é substituído pela URI "http://schema.org/name". Além disso, é informado se o valor da propriedade do atributo é um outro recurso (@id). A introdução das URIs torna o texto menos legível.

```
{
 "http://schema.org/name": "Manu Sporny",
 "http://schema.org/url": { "@id": "http://manu.sporny.org/" },
 "http://schema.org/image": {"@id": "http://manu.sporny.org/images/manu.png"}
}
```

Figura 4.19 – Inclusão de URIs de propriedades no código JSON

A figura 4.20 apresenta a definição de um contexto onde os nomes abreviados da figura 4.18 são associados às URIs introduzidas na figura 4.19. A associação das abreviações com as definições semânticas universais (contexto), ficam separada das definições dos pares atributo-valor. Isso permite que as abreviações voltem a ser utilizadas. O contexto funciona como um mapeamento entre o vocabulário particular e o vocabulário de referência.

```
{
 "@context":
 {
  "name": "http://schema.org/name",
  "image": {
    "@id": "http://schema.org/image",
    "@type": "@id"
  },
  "homepage": {
    "@id": "http://schema.org/url",
    "@type": "@id"
  }
 }
 "name": "Manu Sporny",
 "homepage": "http://manu.sporny.org/",
 "image": "http://manu.sporny.org/images/manu.png"
}
```

Figura 4.20 – Definição de um contexto em JSON-LD

Os contextos podem ser definidos em conjunto com os pares atributo-valor ou podem ser referenciados por URIs. Dessa forma, um determinado contexto conhecido de uma comunidade pode ser informado no início da definição dos dados e o vocabulário abreviado pode ser utilizado. O contexto da figura 4.20 poderia ser considerado como o contexto da definição de uma pessoa. A figura 4.21 ilustra como o código usaria as abreviações incluindo uma referência a um contexto definido externamente.

```
{
  "@context": "http://json-ld.org/contexts/person.jsonld",
  "name": "Manu Sporny",
  "homepage": "http://manu.sporny.org/",
  "image": "http://manu.sporny.org/images/manu.png"
}
```

Figura 4.21 – Definição de dados em JSON-LD

4.2 RDFS

O modelo de dados RDF fornece uma maneira de fazer afirmações sobre recursos, mas não faz suposições sobre a semântica dos recursos identificados pelas URIs. Não existe nenhum construtor em RDF que especifique que um

recurso seja uma propriedade, ou seja um livro, uma pessoa, etc. Na prática, RDF é normalmente usado em combinação com vocabulários que fornecem informações semânticas sobre esses recursos.

Resource Description Framework Schema (RDFS) [23] é um vocabulário que estende RDF e introduz uma camada que especifica algumas características que agregam semântica a dados definidos em RDF. RDFS tem construtores que permitem, por exemplo, especificar que determinadas URIs indicam propriedades de recursos, ou que determinados recursos identificados por URIs pertencem a uma determinada classe. Utilizando o exemplo da figura 4.13, por meio do RDFS é possível especificar que o recurso identificado pela URI "http://xmlns.com/foaf/0.1/name" é uma propriedade. Ou que o recurso identificado pela URI "http://example.org/#gabriela-cravo-canela" pertence à classe "Livro".

RDFS usa a noção de classe para especificar categorias que podem ser usadas para classificar os recursos. A relação entre uma instância e sua classe é indicada através da propriedade "type" de RDF. Com RDFS pode-se criar hierarquias de classes e subclasses, e hierarquias de propriedades e subpropriedades. Restrições de tipos podem ser definidas sobre os sujeitos e os objetos das triplas, por meio da especificação de domínios e contradomínios para cada um dos tipos. No exemplo anterior dos livros, poderíamos declarar que o domínio da relação "tem_publicador" é do tipo "Livro" e que o contradomínio é do tipo "Publicador".

É muito importante ressaltar que a ideia de tipos, domínios e contradomínios de uma propriedade são diferentes daqueles do modelo orientado a objetos (OO). No modelo OO, uma classe abstrai as propriedades e os métodos de um determinado conjunto de instâncias. Se um objeto é declarado como sendo de um determinado tipo (uma classe), é assumido que ele possui um conjunto de propriedades que são associadas àquele tipo. No caso do RDFS, a declaração de uma propriedade não restringe o seu uso por qualquer recurso, qualquer que seja o seu tipo. A declaração de propriedades é feita de forma separada e independente das declarações de classes. A declaração de um domínio e um contradomínio permite apenas que se faça uma inferência de um possível tipo de um recurso, mesmo que esse tipo não esteja declarado explicitamente por meio de uma tripla. No dia a dia, os humanos fazem isso constantemente. A partir de certas propriedades observadas de objetos, pessoas, etc., os humanos inferem os tipos aos quais esses recursos podem

pertencer. É uma forma comum de inferência para seres humanos..

Voltando ao RDFS, no exemplo dos livros, vamos supor que sejam declaradas as classes “Livro” e “Publicador” e que o domínio da relação “tem_publicador” é de recursos da classe “Livro” e que o contradomínio é de recursos da classe “Publicador”. Mesmo que não se declare uma tripla onde se afirme explicitamente que o recurso “http://example.org/#gabriela-cravo-canela” é da classe “Livro” e que o recurso “http://example.org/#companhia-das-letras” é da classe “Publicador”, se for declarado que existe uma relação “tem_publicador” entre esses dois recursos, as classes dos recursos serão inferidas pela declaração do domínio e do contradomínio da propriedade. Dessa forma é possível se concluir mais triplas dos que as efetivamente declaradas.

A inferência é uma das características introduzidas pelo uso das tecnologias da Web Semântica. Inferências permitem que se façam declarações menos extensas das informações. Por exemplo, em um conjunto de triplas sobre pessoas, não é necessário que se declare explicitamente as triplas com as relações do tipo “tio”, que podem ser inferidas a partir de outras relações. RDFS introduz alguns poucos tipos de inferências, que são expandidas de forma mais ampla pelo uso de OWL, como será visto na seção 4.3.

Um outro tipo de inferência introduzido por RDFS é relativo à declaração de subclasses e de subpropriedades. Se um determinado recurso é do tipo de uma determinada subclasse, será inferido que esse recurso também é do tipo da classe correspondente. O mesmo se dá em relação às subpropriedades.

A figura 4.22 ilustra a introdução de alguns construtores de RDFS no exemplo da figura 4.17. Para a definição da classe “Livro”, utilizamos a ontologia “bibo” [24], relativa à definição de informações bibliográficas. A palavra-chave “a” é uma abreviação para o predicado “rdf:type”, que indica a classe de um recurso.

```
base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix bibo: <http://purl.org/ontology/bibo/> .

<#gabriela-cravo-canela>
    a bibo:Book ;
    dc:identifier "9788535912388" ;
    foaf:name "Gabriela, Cravo e Canela" ;
    dc:creator "Jorge Amado" ;
    dc:publisher <#companhia-das-letras> .

<#vidas-secas>
    a bibo:Book ;
    dc:identifier "9788501067340" ;
    foaf:name "Vidas Secas" ;
    dc:creator "Graciliano Ramos" ;
    dc:publisher <#grupo-record> .

<#antologia-poetica>
    a bibo:Book ;
    dc:identifier "9788535921199" ;
    foaf:name "Antologia Poética" ;
    dc:creator "Carlos Drummond de Andrade" ;
    dc:publisher <#companhia-das-letras> .

<#companhia-das-letras>
    a foaf:Organization ;
    foaf:name "Companhia das Letras".

<#grupo-record>
    a foaf:Organization ;
    foaf:name "Grupo Editorial Record" .
```

Figura 4.22 – Inclusão de informações de tipos de recursos (classes)

4.3 OWL

RDF especifica um modelo de dados que representa informações por meio de um conjunto de triplas que definem propriedades de recursos, e relacionamentos entre esses diversos recursos. RDFS estende RDF possibilitando a definição de hierarquias de classes, hierarquias de propriedades e a definição de domínios e contradomínios para as propriedades, permitindo assim um primeiro conjunto de restrições sobre as triplas definidas, além de inferências que deduzem triplas não declaradas de forma explícita.

Por que definir restrições? Uma das formas de se entender o que é semântica é pensar que o que faz com que diferentes pessoas possam entender o mesmo significado de algum conteúdo é restringir o número de diferentes interpretações possíveis sobre aquele conteúdo. O ideal seria que houvesse apenas uma interpretação possível.

Ontology Web Language [25] é uma linguagem que estende RDF e RDFS e oferece um conjunto muito mais amplo de tipos de restrições ao conjunto de triplas definidas. Além disso, são oferecidos diversos construtores que permitem, entre outros, a construção de classes complexas a partir de outras definições de classes, e encadeamento de propriedades. Uma das principais bases do OWL é Description Logics (DLs) [26], uma família de linguagens de representação de conhecimento amplamente utilizadas na modelagem de ontologias. Uma ontologia é uma especificação de um conceito dentro de um determinado domínio de interesse. Como seu nome sugere, DLs são lógicas e possuem uma semântica formal: uma especificação precisa do significado. Esta semântica formal permite que os seres humanos e sistemas de computador possam intercambiar ontologias DL sem ambiguidade quanto ao seu significado, e também torna possível usar dedução lógica para inferir informações adicionais dos fatos expostos explicitamente em uma ontologia.

Ontologias fornecem meios para modelar as relações entre as entidades em um domínio de interesse. Em OWL existem três tipos de entidades

- Instâncias – representam os recursos (são também chamados de indivíduos).
- Classes – definem conjuntos de instâncias, de indivíduos.
- Propriedades – representam relações binárias entre duas instâncias (*object property*) ou entre uma instância e um literal (*datatype property*).

Uma ontologia especificada em OWL consiste de um conjunto de afirmações (axiomas), separadas em três grupos: Tbox (*Terminological*), Abox (*Assertion*) e Rbox (Role).

A Tbox descreve relacionamentos entre classes. Por exemplo, a afirmação que a classe “Person” é equivalente a classe “Human” significa que as duas classes possuem o mesmo conjunto de indivíduos. A Abox captura conhecimento sobre os indivíduos, ou seja, as classes as quais eles pertencem e como eles se relacionam entre si. Por exemplo, podemos fazer uma afirmação que o indivíduo “Mary” é da classe “Person”, isto é, “Mary” é uma instância da classe “Person”. Se juntarmos com o exemplo, da Tbox, em que a classe “Person” é definida como equivalente à classe “Human”, pode-se inferir que “Mary” também é uma instância da classe “Human”:

```
:Person owl:equivalentClass :Human .
:Mary   rdf:type :Person .
:Mary   rdf:type :Human . (inferência)
```

Na Tbox também são definidos relacionamentos entre os indivíduos. Por exemplo, podemos definir que “Mary” é esposa de “John”:

```
:John :hasWife :Mary .
```

A Rbox contém as afirmações sobre as propriedades, ou seja, metapropriedades como, por exemplo, transitividade, simetria, etc. Essas afirmações podem permitir que inferências sejam feitas sobre a base de triplas definidas explicitamente:

```
:hasAncestor  rdf:type  owl:TransitiveProperty .
:John hasAncestor :Phil .
:Phil hasAncestor :Peter .
:John hasAncestor :Peter . (inferência)
```

Os itens a seguir oferecem um resumo dos diversos construtores contidos na especificação da linguagem OWL:

- Classes são definidas em RDFS por meio da propriedade “type” de RDF:

```
:Person rdf:type rdfs:Class .
:Woman rdf:type rdfs:Class .
:Woman rdfs:subClassOf :Person .
```

• A classe de uma determinada instância pode ser declarada explicitamente, ou pode ser inferida como, por exemplo, a partir da definição de subclasses ou de domínio e contradomínio:

```
:Mary rdf:type owl:NamedIndividual .
:Mary rdf:type :Woman .
:Mary rdf:type :Person . (inferência)
```

• É possível definir que dois nomes de indivíduos representam o mesmo indivíduo, ou seja, o grafo RDF do recurso será o somatório das afirmações sobre cada um dos indivíduos:

```
:Mary owl:sameAs otherOntology:MaryBrown .
```

• Classes equivalentes possuem o mesmo conjunto de instâncias:

```
:Person owl:equivalentClass :Human .
:Mary   rdf:type :Person .
:Mary   rdf:type :Human . (inferência)
```

• Classes disjuntas indicam que uma instância que pertença a uma delas não pertence a outra.

```
[]  rdf:type     owl:AllDisjointClasses ;
    owl:members  ( :Woman  :Man ) .
```

• Uma propriedade pode ter uma valor literal (*datatype property*) ou estabelecer um relacionamento entre duas instâncias (*object property*).

```
:hasAge rdf:type owl:DatatypeProperty .
:John :hasAge 52 .
:hasWife rdf:type owl:ObjectProperty .
:John :hasWife :Mary .
```

• Classes complexas podem ser construídas a partir de outras classes, por meio de interseção, união, complemento e enumeração:

```
:Mother  owl:equivalentClass  [
   rdf:type            owl:Class ;
   owl:intersectionOf  ( :Woman :Parent )
 ] .
:Parent  owl:equivalentClass  [
   rdf:type            owl:Class ;
   owl:unionOf         ( :Mother :Father )
 ] .
:ChildlessPerson  owl:equivalentClass  [
   rdf:type            owl:Class ;
  owl:intersectionOf  ( :Person  [ owl:complementOf
:Parent ] )
 ] .
:Beatles  owl:equivalentClass  [
   rdf:type            owl:Class ;
   owl:oneOf           ( :George  :Ringo  :John  :Paul )
 ] .
```

• Propriedades podem ter uma cardinalidade que indica o número (máximo, mínimo ou exato) de triplas:

```
:John  rdf:type  [
   rdf:type            owl:Restriction ;
   owl:maxCardinality  "4"^^xsd:nonNegativeInteger ;
   owl:onProperty      :hasChild
] .
:John  rdf:type  [
   rdf:type            owl:Restriction ;
   owl:minCardinality  "2"^^xsd:nonNegativeInteger ;
   owl:onProperty      :hasChild
] .
:John  rdf:type  [
   rdf:type            owl:Restriction ;
   owl:cardinality     "5"^^xsd:nonNegativeInteger ;
   owl:onProperty      :hasChild
] .
```

• Propriedades podem ser transitivas :

```
:hasAncestor  rdf:type  owl:TransitiveProperty .
:John hasAncestor :Phil .
:Phil hasAncestor :Peter .
:John hasAncestor :Peter . (inferência)
```

• Propriedades podem ser inversas:

```
:hasParent owl:inverseOf :hasChild .
:John hasChild :Paul .
:Paul hasParent :John . (inferência)
```

• Propriedades podem ser simétricas (a inversa da propriedade é a mesma propriedade, por exemplo, cônjuge):

```
:hasSpouse  rdf:type  owl:SymmetricProperty .
:John hasSpouse :Mary .
:Mary hasSpouse :John . (inferência)
```

• Propriedades podem ser assimétricas (a propriedade inversa não pode ser a mesma propriedade, por exemplo, uma pessoa não pode ser filha do seu filho):

```
:hasChild  rdf:type  owl:AsymmetricProperty .
```

• Propriedades podem ser definidas como encadeamentos de outras propriedades (avô é o pai do pai):

```
:hasGrandparent  owl:propertyChainAxiom  ( :hasParent  :hasParent ) .
:John  hasParent  :Phil .
:Phil  hasParent  :Peter .
:Peter hasParent  :Paul .
:John hasGrandparent :Peter . (inferência)
:Phil hasGrandparent :Paul .  (inferência)
```

• Propriedades podem ser reflexivas (uma pessoa é parente de si própria):

```
:hasRelative  rdf:type  owl:ReflexiveProperty .
```

• Propriedades podem ser irreflexivas (uma pessoa não pode ser pai de si mesma):

```
:parentOf  rdf:type  owl:IrreflexiveProperty .
```

• Propriedades podem ser funcionais (só existe um elemento no contradomínio para um elemento do domínio, por exemplo, mãe):

```
:hasMother  rdf:type  owl:FunctionalProperty .
```

• Propriedades podem ser inversamente funcionais (só existe um elemento no domínio para um elemento do contradomínio, por exemplo, filho):

```
:hasChild  rdf:type  owl:InverseFunctionalProperty .
```

4.4 SPARQL

Os dados são representados na Web Semântica utilizando o modelo de dados conceitual de RDF, em conjunto com as extensões de RDFS e OWL. Esses dados podem estar armazenados em, por exemplo, um banco de dados de triplas (*triple store*), um banco de dados relacional com um esquema de mapeamento para rdf, etc.

SPARQL (SPARQL Protocol and Query Language) [27] é a linguagem de consulta da Web Semântica. Podemos fazer uma analogia entre SPARQL e a linguagem SQL de consulta a bancos de dados relacionais, considerando que SPARQL tem uma sintaxe adequada a consultas a dados representados como um conjunto de triplas RDF.

Por exemplo, considere um banco de triplas com o seguinte conteúdo:

```
<http://example.org/book/book1>  <http://purl.org/dc/elements/1.1/title>
  "RDF Tutorial" .
<http://example.org/book/book2>  <http://purl.org/dc/elements/1.1/title>
  "SPARQL Tutorial" .
```

A consulta

```
SELECT ?title
WHERE
{
  <http://example.org/book/book2>
  <http://purl.org/dc/elements/1.1/title>
  ?title .
}
```

retornaria:

title
"SPARQL Tutorial"

As variáveis SPARQL começam com um caractere “?” e podem ser definidas em qualquer uma das três posições de uma tripla (sujeito, predicado, objeto) no conjunto de dados RDF. Os padrões de triplas da cláusula SELECT têm a mesma forma de triplas normais, exceto que qualquer uma das três partes da tripla pode ser substituída por uma variável. A cláusula SELECT retorna uma tabela de variáveis com os valores que satisfazem a consulta.

Em um outro exemplo, considere um banco de triplas com o seguinte conteúdo:

```
base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .

<#gabriela-cravo-canela>
    dc:identifier "9788535912388" ;
    foaf:name "Gabriela, Cravo e Canela" ;
    dc:creator "Jorge Amado" ;
    dc:publisher <#companhia-das-letras> .

<#vidas-secas>
    dc:identifier "9788501067340" ;
    foaf:name "Vidas Secas" ;
    dc:creator "Graciliano Ramos" ;
    dc:publisher <#grupo-record> .

<#antologia-poetica>
    dc:identifier "9788535921199" ;
    foaf:name "Antologia Poética" ;
    dc:creator "Carlos Drummond de Andrade" ;
    dc:publisher <#companhia-das-letras> .

<#companhia-das-letras>
    foaf:name "Companhia das Letras".

<#grupo-record>
    foaf:name "Grupo Editorial Record" .
```

A consulta:

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX ex: <http://example.org/>
SELECT ?name ?creator
WHERE
{ ?book dc:publisher ex:companhia-das-letras .
  ?book foaf:name ?name .
  ?book dc:creator ?creator .
}
```

retornaria:

NAME	creator
"Gabriela, Cravo e Canela"	"Jorge Amado"
"Antologia Poética"	"Carlos Drummond de Andrade"

SPARQL admite uma série de filtros e operadores que permitem fazer consultas complexas ao conjunto de triplas armazenadas. Diversos bancos de

dados de triplas oferecem pontos de acesso via Web (URLs), que aceitam o protocolo SPARQL e sua linguagem de consulta. Esses pontos de acesso são denominados SPARQL *endpoints*. Um SPARQL endpoint aceita consultas e retorna os resultados via HTTP. *Endpoints* genéricos executam consultas em qualquer dado RDF com acesso possível pela Web (especificado como parâmetro). *Endpoints* específicos executam consultas apenas em conjuntos de dados particulares, fixados pela aplicação. As listas de alguns SPARQL endpoints existentes na Web são apresentadas em SPARQLES [28], W3C SPARQL endpoints [29] e Mondeca [30]. Um endpoint genérico é oferecido por Openlink [31], onde é possível fazer consultas a partir da especificação da localização do grafo RDF que contém o conjunto de triplas a ser inspecionado.

A cláusula SELECT aceita que seja especificado o grafo RDF que será consultado, por meio da utilização da palavra-chave FROM. No exemplo a seguir (figuras 4.23 e 4.24), são retornadas as homepages das pessoas conhecidas ("foaf:knows") por Tim Berners-Lee, no grafo RDF localizado em http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf

```
PREFIX foaf:  <http://xmlns.com/foaf/0.1/>
PREFIX card: <http://www.w3.org/People/Berners-Lee/card#>
SELECT ?homepage
FROM <http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
WHERE {
    card:i foaf:knows ?known .
    ?known foaf:homepage ?homepage . }
```

Figura 4.23 – Utilização de FROM em SPARQL

homepage
http://purl.org/net/eric/
http://www.johnseelybrown.com/
...

Figura 4.24 – Resultados da utilização de FROM em SPARQL

As consultas SPARQL não retornam conjuntos de triplas. O resultado é um conjunto de tuplas (como linhas de uma tabela), que pode vir em diferentes

formatos como, por exemplo, HTML, XML, JSON, CSV, etc. Esse formato é especificado como um dos parâmetros da consulta. A figura 4.25 apresenta o resultado em JSON da consulta anterior:

```
{ "head": { "link": [], "vars": ["homepage"] },
  "results": { "distinct": false, "ordered": true, "bindings": [
   { "homepage": { "type": "uri", "value": "http://purl.org/net/eric/" }},
   { "homepage": { "type": "uri", "value": "http://www.johnseelybrown.com/" }} ,
    ... ] } }
```

Figura 4.25 – Resultado em JSON de consulta SPARQL

Além do comando SELECT, SPARQL aceita os comandos DESCRIBE, CONSTRUCT e ASK. O comando DESCRIBE retorna um grafo RDF simples contendo os dados RDF sobre os recursos especificados no comando. Os dados retornados são determinados pelo processador de consulta SPARQL e não pelo cliente que fez a consulta. Podem ser retornadas triplas onde o recurso aparece como sujeito, predicado ou objeto, triplas onde o recurso aparece como sujeito ou objeto, ou triplas onde o recurso aparece apenas como sujeito.

O acesso às triplas de um recurso também pode ser feito por meio de uma dereferenciação (a partir da definição de uma URI), que é um dos princípios de Dados Conectados na Web (seção 5.1). O resultado de uma dereferenciação, assim como o resultado de um comando DESCRIBE, retorna um conjunto de triplas de recursos. O *site* DBpedia contém triplas extraídas das caixas de informações de artigos da Wikipedia. A dereferenciação da URI da DBpedia que identifica Tim Berners-Lee pode ser visualizada em http://dbpedia.org/page/Tim_Berners-Lee [32]. Essa página é retornada originalmente em HTML mas é possível obter o seu resultado em outros formatos, como por exemplo, em JSON [33].

A figura 4.26 apresenta um exemplo de comando DESCRIBE em uma consulta no endpoint da DBpedia [34]. A figura 4.27 apresenta parte dos resultados. É possível observar que são listadas triplas onde o recurso (":Tim_Berners-Lee") aparece como sujeito e como objeto.

```
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
DESCRIBE ?timbl WHERE {
  ?timbl dbpedia-owl:alias "TimBL" .
}
```

Figura 4.26 – Comando DESCRIBE de SPARQL

s	p	o
:Tim_Berners-Lee	dc:description	"British computer scientist, best known as the inventor of the World Wide Web"
:Tim_Berners-Lee	dbpedia2:occupation	:Computer_scientist
:Tim_Berners-Lee	foaf:name	"Sir Tim Berners-Lee"
:Tim_Berners-Lee	foaf:name	"Tim Berners-Lee"
:Tim_Berners-Lee	foaf:name	"Berners-Lee, Tim"
:Tim_Berners-Lee	foaf:surname	"Berners-Lee"
:Tim_Berners-Lee	rdfs:label	"Tim Berners-Lee"
:Tim_Berners-Lee	dbpedia:ontology/birthYear	"1955+02:00"
:Tim_Berners-Lee	dbpedia2:birthPlace	"United Kingdom"
:Tim_Berners-Lee	dbpedia2:nationality	"British"
:Tim_Berners-Lee	dbpedia:ontology/employer	:Massachusetts_Institute_of_Technology
:Tim_Berners-Lee	dbpedia:ontology/award	:Royal_Society
:Tim_Berners-Lee	owl:sameAs	<http://eo.dbpedia.org/resource/Tim_Berners-Lee>
...		
:Conway_Berners-Lee	dbpedia2:children	:Tim_Berners-Lee
:ENQUIRE	dbpedia2:inventor	:Tim_Berners-Lee
:Libwww	dbpedia2:author	:Tim_Berners-Lee
...		

Figura 4.27 – Resultados de comando DESCRIBE de SPARQL

De forma análoga à SQL, existe a possibilidade de manipular o conjunto de resultados retornados por uma consulta SPARQL:

- LIMIT – limita o número de linhas.
- DISTINCT – remove linhas duplicadas.
- ORDER – ordena o resultado.
- OFFSET – permite paginação.
- FILTER – aplica filtros sobre os valores procurados nas propriedades.

SPARQL oferece, também, uma série de funções pré-construídas utilizadas na especificação das consultas, que incluem operadores lógicos (“!”, “&&”, “||”), operadores matemáticos (“+”, “-”, “*”, “/”), operadores comparativos (“=,” “>”,

"<", ...), operadores de teste ("isLiteral", isURI", ...), funções de manipulação de cadeias de caracteres ("STRLEN", "SUBSTR", "UCASE", "CONCAT", ...), etc.

CONSTRUCT é um comando da linguagem SPARQL, que permite que a partir de um resultado de pesquisa seja construído um conjunto de triplas. A figura 4.28 apresenta um exemplo de comando CONSTRUCT em uma consulta no endpoint da DBpedia. A figura 4.29 apresenta os resultados.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>
CONSTRUCT { ?timbl foaf:name ?name }
WHERE {
  ?timbl dbpedia-owl:alias "TimBL" .
  ?timbl foaf:name ?name .
}
```

Figura 4.28 – Comando CONSTRUCT de SPARQL

s	p	o
:Tim_Berners-Lee	foaf:name	"Sir Tim Berners-Lee"
:Tim_Berners-Lee	foaf:name	"Tim Berners-Lee"
:Tim_Berners-Lee	foaf:name	"Berners-Lee, Tim"

Figura 4.29 – Resultados de comando CONSTRUCT de SPARQL

Além da especificação da linguagem de consulta em si, existem diversos outros padrões relacionados ao SPARQL, definidos pelo W3C, que especificam, por exemplo, uma linguagem para a manipulação e atualização de grafos RDF (SPARQL 1.1 Update [35]), consultas federadas em diversos grafos RDF (SPARQL 1.1 Federated Query [36]), etc.

4.5 METADADOS EMBUTIDOS EM PÁGINAS

Nas seções anteriores apresentamos o modelo de dados conceitual do RDF, com seu conjunto de triplas de recursos identificados por URIs, além da

apresentação de extensões por meio de RDFS e OWL, que permitem a construção de ontologias mais complexas, com um maior número de restrições e possibilidades de inferências. Apresentamos, também, a linguagem SPARQL para consulta a grafos RDF.

Essas tecnologias estão mais direcionadas para grafos que estão armazenados nos servidores Web. Existe, porém, um universo de dados sendo descritos semanticamente, espalhados em páginas HTML, que contêm uma mescla do conteúdo a ser consumido pelas pessoas com o conteúdo a ser consumido pelas aplicações. Existe uma crescente inclusão de dados semânticos nessas páginas, por criadores de páginas interessados em atender às demandas das aplicações. A inclusão de dados semânticos em um site de vendas, por exemplo, pode auxiliar a máquina de busca da Google a expor de forma mais precisa e objetiva os resultados da procura por um determinado produto.

A seção 2.5 sobre metadados, apresentou a forma inicial de introdução de metadados em uma página Web, por meio do *tag* <meta> de HTML. Esses metadados são relacionados à página Web, como um todo. Eles podem, por exemplo, prover informações sobre o autor da página, palavras-chave, descrição, data de publicação, etc. Nas seções seguintes apresentaremos três formatos utilizados para a inclusão de metadados em páginas Web, que podem ser aplicados, também, na descrição individual dos dados contidos nas páginas.

O objetivo dessas seções é descrever de forma sucinta essas três técnicas de embutir dados em páginas da Web. O importante é compreender que essa é uma das formas de se definir metadados para a construção de uma Web de Dados mais semântica, complementar ao que foi apresentado nas seções anteriores.

4.5.1 MICROFORMATO

O microformato [37] foi a primeira iniciativa no sentido de agregar informações extras ao código HTML, de forma que fosse possível incluir tipos específicos de dados (entidades), como por exemplo, pessoas, produtos, eventos, etc. Cada um desses tipos possui um conjunto particular de propriedades e uma sintaxe específica. Por exemplo, uma pessoa pode ter as propriedades nome, endereço, empresa, função, e-mail, etc.

A interpretação de um código HTML, para fins de exibição de dados para o

usuário, ignora qualquer *tag* desconhecido da especificação HTML. Em geral, microformatos utilizam os atributos de classe em tags HTML (frequentemente ou) para atribuir nomes para entidades e suas propriedades.

O exemplo das figuras 4.30 e 4.31 ilustra como mesclar informações que serão filtradas pelo navegador para a exibição da página e informações que serão filtradas pelas aplicações para o entendimento do conteúdo.

```
<div>
   <img src="www.example.com/bobsmith.jpg" />
   <strong>Bob Smith</strong>
   Senior editor at ACME Reviews
   200 Main St
   Desertville, AZ 12345
</div>
```

Figura 4.30 – Código HTML sem informações de microformato

```
<div class="vcard">
   <img class="photo" src="www.example.com/bobsmith.jpg" />
   <strong class="fn">Bob Smith</strong>
   <span class="title">Senior editor</span> at
   <span class="org">ACME Reviews</span>
   <span class="adr">
      <span class="street-address">200 Main St</span>
      <span class="locality">Desertville</span>,
      <span class="region">AZ</span>
      <span class="postal-code">12345</span>
   </span>
</div>
```

Figura 4.31 – Código HTML mesclado com informações de microformato

A seguir são apresentadas algumas das propriedades que podem ser definidas para uma entidade do tipo pessoa :

- fn – nome completo. obrigatório
- n – nome estruturado:
    - given-name – primeiro nome
    - additional-name – nome do meio
    - family-name – sobrenome

- title – função
- org – companhia, organização
- photo – foto, ícone, avatar
- email – e-mail
- tel – telefone
- adr – endereço estruturado:
    - street-address – rua
    - locality – cidade
    - region – estado
    - postal-code – código postal
    - country-name – país

Cada microformato específico tem um conjunto particular de propriedades (vocabulário). Microformatos são fáceis de utilizar devido a sua simplicidade, mas oferecem pouca capacidade de extensão, não tendo uma forma padrão de representação para os vocabulários. Além disso, só é possível especificar metadados para um pequeno conjunto de tipos de dados. Microformatos2 [38] busca evoluir o conceito de microformatos e criar uma sintaxe comum independente de vocabulários e uma padronização maior na nomenclatura das entidades e das propriedades.

4.5.2 RDFA

Assim como microdados, RDFa [39] permite que metadados sejam embutidos em páginas Web, mas utiliza uma sintaxe mais genérica para especificar as triplas RDF, independente do tipo de dados ao qual o recurso pertence e do vocabulário utilizado. As informações das triplas são especificadas na forma de atributos dentro dos tags do HTML:

- vocab – URI do vocabulário (namespace)
- about, src, href e resource – URI do recurso
- typeof – tipo do recurso
- rel – relação com outro recurso

- rev – relação inversa com outro recurso
- property – nome da propriedade (o valor da propriedade é o texto que aparece entre os tags onde é definido o atributo)
- content – substitui o valor da propriedade que aparece entre os tags onde é definido o atributo
- datatype – tipo do dado da propriedade

Como forma de ilustrar alguns desses atributos, serão apresentados dois exemplos (com suas figuras) extraídos da especificação RDFa [40] do W3C. Na Figura 4.32 são definidas triplas referentes a dois recursos. As figuras 4.33 e 4.34 apresentam os grafos RDF dessas triplas e ajudam a entender a forma como elas são definidas por meio do RDFa embutido no código HTML.

```
<body vocab="http://purl.org/dc/terms/">
   ...
   <div resource="/alice/posts/trouble_with_bob">
      <h2 property="title">The trouble with Bob</h2>
      <p>Date: <span property="created">2011-09-10</span></
p>
      <h3 property="creator">Alice</h3>
      ...
   </div>
   ...
   <div resource="/alice/posts/jos_barbecue">
      <h2 property="title">Jo's Barbecue</h2>
      <p>Date: <span property="created">2011-09-14</span></
p>
      <h3 property="creator">Eve</h3>
      ...
   </div>
   ...
</body>
```

Figura 4.32 – RDFa embutido em páginas Web

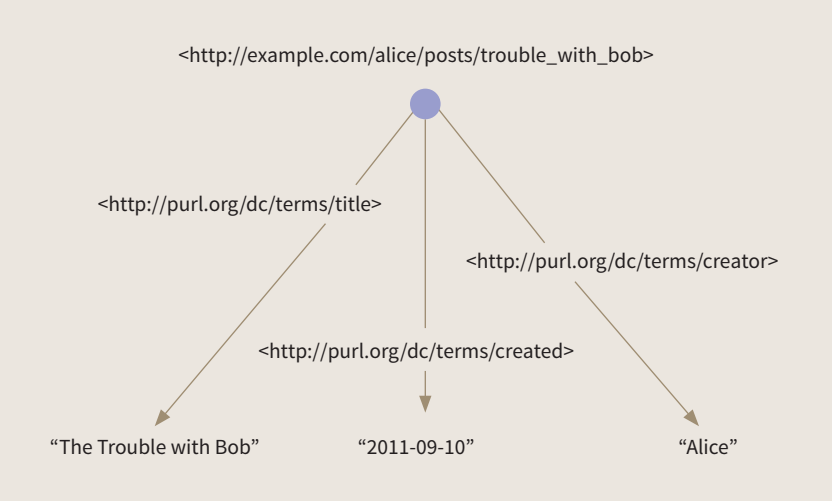


Figura 4.33 – Triplas embutidas em páginas Web (1)

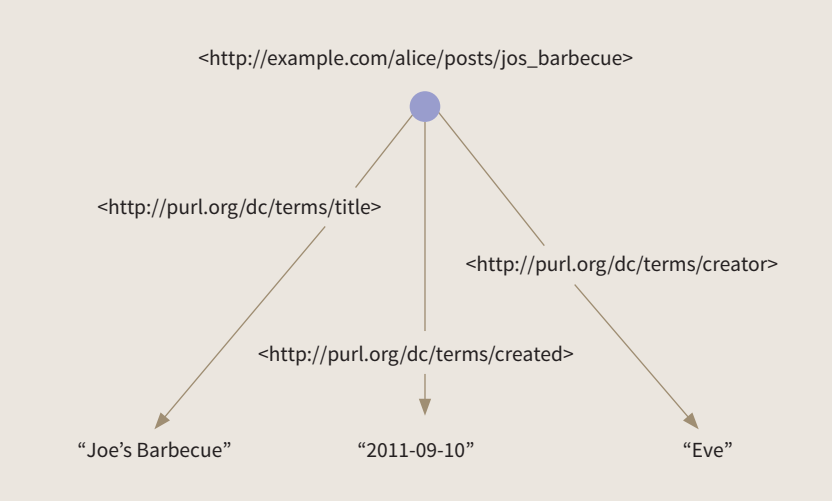


Figura 4.34 – Triplas embutidas em páginas Web (2)

O exemplo da Figura 4.35 apresenta o uso do atributo "rel", que define uma relação entre dois recursos, no caso, a relação "foaf:knows" (alguém conhecido de uma determinada pessoa). No exemplo, os recursos que são objetos dessas triplas são do tipo "foaf:Person" e para cada um deles é definido um "foaf:name" e uma "foaf:homepage". O que o navegador exibirá para o usuário serão os *links* para as homepages de Bob, Eve e Ana.

```
<div vocab="http://xmlns.com/foaf/0.1/" resource="#me">
  <ul rel="knows">
    <li resource="http://example.com/bob/#me" typeof="Person">
      <a property="homepage" href="http://example.com/bob/">
      <span property="name">Bob</span></a>
    </li>
    <li resource="http://example.com/eve/#me" typeof="Person">
      <a property="homepage" href="http://example.com/eve/">
      <span property="name">Eve</span></a>
    </li>
    <li resource="http://example.com/ana/#me" typeof="Person">
      <a property="homepage" href="http://example.com/ana/">
      <span property="name">Ana</span></a>
    </li>
  </ul>
</div>
```

Figura 4.35 – Exemplo de relação entre recursos codificado em RDFa

4.5.3 MICRODADOS

Microdados [41] são uma forma mais recente de embutir metadados em páginas Web. Ela tem tido uma crescente utilização, sendo uma das formas preferenciais das máquinas de busca como, por exemplo, a Google. Apesar de poder utilizar qualquer vocabulário para a definição de seus recursos, os criadores de páginas que utilizam microdados fazem muito uso do conjunto de vocabulários definidos pela iniciativa Schema.org (seção 6.4).

Da mesma forma que RDFa, a sintaxe de microdados não depende dos tipos de dados dos recursos definidos, nem do vocabulário utilizado para a sua descrição. A forma de embutir as informações também é por meio da definição de atributos nos tags HTML. O modelo de microdados consiste em grupos de pares nome-valor, conhecidos como itens. Cada item pode ter um tipo, um identificador global e uma lista de pares nome-valor. Cada nome do par nome-valor é conhecido como uma propriedade, e cada propriedade tem um ou mais valores. Cada valor é um item ou um literal. Os atributos básicos de microdados são:

- itemscope – define o recurso
- itemid – define uma URI para o recurso
- itemtype – define o tipo do recurso (vocabulário)
- itemprop – define uma propriedade do recurso

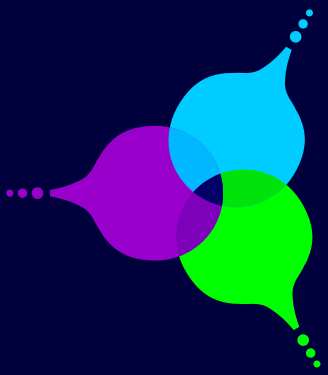
A figura 4.36 apresenta um exemplo de microdados para uma pessoa, utilizando o tipo de dados “Person” [42] do Schema.org, que tem uma série de propriedades definidas como, por exemplo, “name”, “image’, “jobTitle”, “telephone”, “email”, “colleague”, entre outras, todas com uma semântica bem definida. No exemplo é possível perceber que a propriedade “address” pode ser uma propriedade que aponta para um outro recurso (“itemscope”), que tem um outro tipo de dados, “PostalAddress”.

Da mesma forma que em RDF, o modelo de dados de microdados em si não entra no mérito do significado de cada um dos vocabulários. A sintaxe para o seu uso como metadado é única, independente de vocabulários, e a escolha e o uso adequado de cada um deles, e de suas propriedades, requer um estudo específico por parte do usuário.

A semântica agregada aos dados é materializada pela utilização das sintaxes apresentadas (RDF, RDFS, OWL, microformatos, RDFa, microdados) em conjunto com ontologias próprias, desenvolvidas para casos específicos, e os vocabulários de referência, que se estabelecem pela cultura de uso. No capítulo 6 são apresentados exemplos de alguns dos vocabulários mais utilizados.

```
<div itemscope itemtype="http://schema.org/Person">
  <span itemprop="name">Jane Doe</span>
  <img src="janedoe.jpg" itemprop="image" alt="Photo of Jane Joe"/>
  <span itemprop="jobTitle">Professor</span>
  <div itemprop="address"
       itemscope itemtype="http://schema.org/PostalAddress">
       <span itemprop="streetAddress">
          20341 Whitworth Institute
          405 N. Whitworth
       </span>
       <span itemprop="addressLocality">Seattle</span>,
       <span itemprop="addressRegion">WA</span>
       <span itemprop="postalCode">98052</span>
  </div>
  <span itemprop="telephone">(425) 123-4567</span>
  <a href="mailto:jane-doe@xyz.edu" itemprop="email">
    jane-doe@xyz.edu</a>
  Jane's home page:
  <a href="http://www.janedoe.com" itemprop="url">janedoe.com</a>
  Graduate students:
  <a href="http://www.xyz.edu/students/alicejones.html"
     itemprop="colleague">Alice Jones</a>
  <a href="http://www.xyz.edu/students/bobsmith.html"
     itemprop="colleague">Bob Smith</a>
</div>
```

Figura 4.36 – Microdados embutidos em uma página Web



DADOS CONECTADOS

Capítulo 5

Nos capítulos anteriores deste guia vimos a evolução da Web de Documentos na direção de uma Web de Dados, e as tecnologias envolvidas em agregar semântica de forma a auxiliar a manipulação dessas informações por parte das aplicações. O horizonte que se deseja alcançar é o de um banco de dados global, onde um conjunto crescente de informações possa ser acessado por um conjunto diversificado de aplicações com os mais diferentes propósitos.

Dados são publicados na Web por diferentes pessoas e estão armazenados em diferentes repositórios espalhados pelo mundo. Para facilitar a construção desse banco de dados global é preciso que se estabeleça uma forma padrão de conexão entre esses dados. Neste capítulo serão apresentados os princípios do que se denominou Dados Conectados. A aplicação desse conceito se expande de forma veloz e a sua evolução costuma ser ilustrada pela figura de uma nuvem, a nuvem da LOD (Linked Open Data) [43], que mostra um conjunto de nós que representam conjuntos de dados abertos, e *links* que representam as conexões estabelecidas entre esses conjuntos de dados. As figuras 5.1 e 5.2 mostram a nuvem em 2007 e 2014, respectivamente.

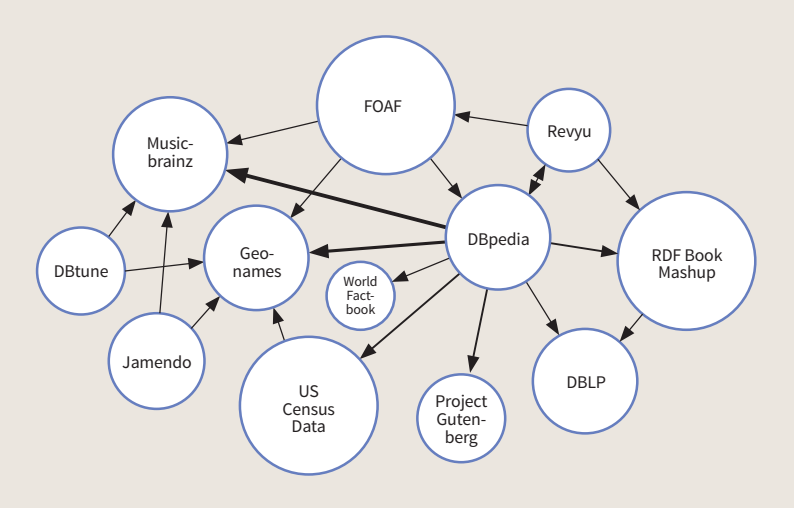


Figura 5.1 – nuvem da LOD em 2007

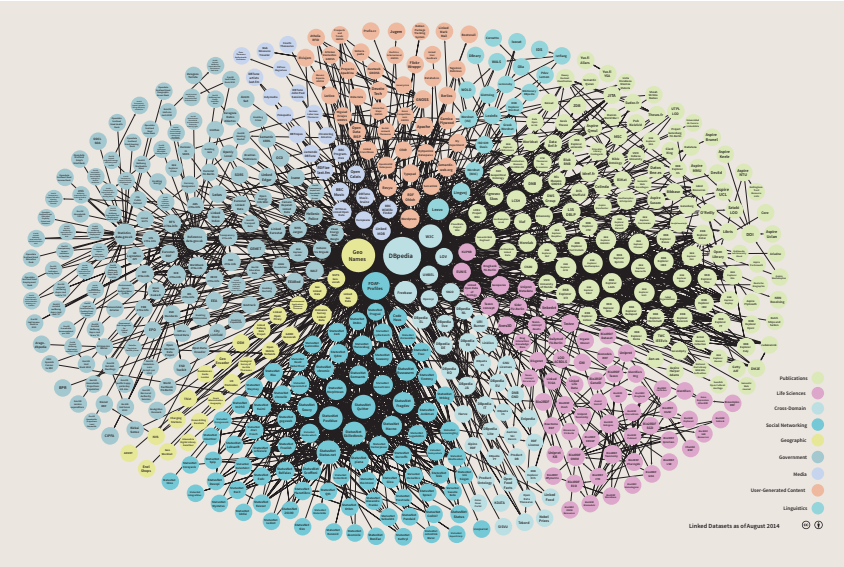


Figura 5.1 – nuvem da LOD em 2007

É fácil perceber a grande expansão da nuvem e a importância do conjunto de dados da DBpedia (nó central da nuvem de 2014, com o maior número de conexões), que contém dados representados em RDF relativos às informações contidas nas caixas de informações dos artigos da Wikipedia.

5.1 OS 4 PRINCÍPIOS

A ideia de se agregar semântica aos dados não é suficiente para a criação de um banco de dados global. Para tal, é preciso que esses dados sejam conectados. Em 2006, Tim Berners-Lee definiu uma lista de 4 princípios [44] para a conexão desses dados:

- Use URIs para identificar as coisas (recursos).
- Use HTTP URIs, de forma a possibilitar que as pessoas possam procurar essas coisas na Web.
- Quando alguém procurar por uma URI, forneça informações relevantes utilizando os padrões (RDF, SPARQL).
- Inclua *links* para outras URIs, de forma a possibilitar que mais coisas possam ser descobertas.

Como vimos anteriormente, as URIs são utilizadas para a identificação de recursos, que podem representar qualquer coisa, concreta ou abstrata. O modelo RDF define uma forma de se representar o conhecimento por meio de triplas que podem definir valores literais para as propriedades, ou, então, estabelecer relacionamentos entre os diversos recursos, as diversas coisas.

O fato de utilizarmos HTTP URIs, permite que essa identificação seja utilizada para uma requisição via Web. O que o terceiro princípio estabelece é que uma vez feita uma requisição utilizando uma dessas URIs (dereferenciação), o servidor responsável por atender a esse pedido deve retornar algum tipo de informação relacionada ao recurso identificado pela URI. Como vimos na seção que apresenta os conceitos de SPARQL, o retorno de uma requisição desse tipo poderia ser algo do tipo do comando DESCRIBE, onde são retornadas as triplas onde o recurso identificado pela URI pode aparecer como sujeito, predicado ou objeto. Uma forma mais simples, seria ter um arquivo rdf estático com as triplas consideradas úteis. Esse princípio não estabelece o que deve ser retornado. Essa decisão é uma atribuição do servidor Web que atende a requisição.

O quarto e último princípio estabelece a conexão propriamente dita entre os dados. Ele estabelece que sempre que possível, um conjunto de dados deve fazer referência a outros conjuntos, de forma a permitir uma navegação entre os diversos nós da nuvem. Por exemplo, o site da DBpedia contém informações sobre recursos que representam pessoas, países, etc. Caso algum outro conjunto de recursos, por exemplo, o *site* da BBC Music, queira referenciar informações sobre os artistas que aparecem em suas programações, ele pode incluir *links* entre os seus recursos e os recursos da DBpedia. Ou seja, triplas contidas no conjunto de dados da BBC Music farão referência a URIs de recursos contidos em triplas da DBpedia.

5.2 AS 5 ESTRELAS

A Web de Dados é atualmente um espaço heterogêneo onde diversos tipos de informações são publicados nos mais diversos formatos e estruturas de armazenamento. Temos dados armazenados em arquivos em diversos formatos, dados armazenados em bancos de dados e retornados em páginas Web, Web Services e Web APIs, dados armazenados em bancos de triplas, *endpoints* SPARQL, etc.

Como forma de orientar a publicação de dados abertos na Web, nessa nova visão semântica, Tim Berners-Lee sugeriu um esquema de classificação de 5 estrelas [45], onde é traçado um caminho evolutivo na direção desse universo de Dados Conectados, uma Web de Dados onde todas as informações estariam conectadas de acordo com os 4 princípios.

Essa escala é composta de 5 níveis:

1. Publique a sua informação na Web (qualquer que seja o formato) sob um tipo de licença de dados abertos.

2. Publique essa informação na forma de dados estruturados (por exemplo, uma planilha Excel ao invés de uma imagem digital de uma tabela).

3. Use formatos não proprietários (por exemplo, uma tabela CSV ao invés de uma planilha Excel).

4. Use URIs para identificar as coisas, de tal forma que as pessoas possam apontar para as suas informações.

5. Faça a conexão entre os seus dados e outros dados, de forma a prover um contexto maior para as informações.

Cada um desses níveis apresenta custos e benefícios que incluem, por exemplo, simplicidade do procedimento, conversões de dados, formação de pessoal qualificado, etc., tudo isso no interesse em caminhar na direção de um conjunto de dados semanticamente melhor descritos, e conectados com outros dados. A seguir é apresentado um exemplo de como um mesmo conjunto de dados poderia ser publicado dentro de cada um dos níveis da classificação “5 Estrelas”.

A figura 5.3, apresenta um exemplo de uma publicação “1 Estrela”: uma imagem de uma tabela publicada em um arquivo pdf. Essa informação pode ser facilmente processada por humanos mas é de muito difícil entendimento por parte de uma aplicação.

Temperature forecast for Galway, Ireland	
Day	**Lowest Temperature (ºC)**
Saturday, 13 November 2010	2
Sunday, 14 November 2010	4
Monday, 15 November 2010	7

Figura 5.3 – Publicação "1 Estrela"

A figura 5.4, apresenta a mesma informação publicada em uma planilha de um arquivo Excel, o que facilitaria o processamento dessa informação por parte das aplicações, porém ainda utilizando um formato proprietário, sendo portanto o caso de uma publicação "2 Estrelas".

Temperature forecast for Galway, Ireland	
Day	**Lowest Temperature (ºC)**
Saturday, 13 November 2010	2
Sunday, 14 November 2010	4
Monday, 15 November 2010	7

Figura 5.4 – Publicação "2 Estrelas"

A figura 5.5, apresenta a mesma informação publicada em um arquivo no formato CSV (um formato não proprietário), sendo portanto o caso de uma publicação "3 Estrelas".

```
Day,Lowest Temperature (C)
"Saturday, 13 November 2010",2
"Sunday, 14 November 2010",4
"Monday, 15 November 2010",7
```

Figura 5.5 – Publicação "3 Estrelas"

A classificação “4 Estrelas” denota a passagem do nível de publicações para um estágio aderente à Web Semântica. No caso do exemplo utilizado, poderíamos associar os elementos das células às propriedades de determinados vocabulários. A figura 5.6 apresenta como uma página Web [46], poderia ser visualizada pelos usuários com as informações sobre as previsões meteorológicas dispostas em uma tabela.

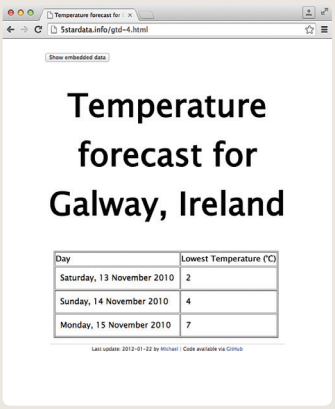


Figura 5.6 – Publicação “4 Estrelas”

A figura 5.7 apresenta parte do código HTML, com inserções de RDFa incluídas como forma de agregar informações semânticas aos dados. Um dos vocabulários definidos é o vocabulário meteo [47], que define um conjunto de propriedades para eventos meteorológicos.

```
<html  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"
       xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:meteo="http://purl.org/ns/meteo#">
<h1 property="dcterms:title">Temperature forecast for Galway,Ireland</h1>
<div id="data" about="#Galway" typeof="meteo:Place">
<table border="1px">
  <tr>
    <th>Day</th>
    <th>Lowest Temperature (&deg;C)</th>
  </tr>
  <tr rel="meteo:forecast" resource="#forecast20101113">
    <td>
      <div about="#forecast20101113">
        <span property="meteo:predicted"
              content="2010-11-13T00:00:00Z"
              datatype="xsd:dateTime">Saturday, 13 November 2010
        </span>
      </div>
    </td>
    <td rel="meteo:temperature">
      <div about="#temp20101113">
        <span property="meteo:celsius"
              datatype="xsd:decimal">2</span>
      </div>
    </td>
  </tr>
...
```

Figura 5.7 – Publicação “4 Estrelas” (código HTML + RDFa)

A figura 5.8 apresenta as triplas extraídas da página, que também podem ser visualizadas utilizando o navegador RDF Graphite [48].

```
@base <http://5stardata.info/gtd-4.html> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix meteo: <http://purl.org/ns/meteo> .
@prefix xhv: <http://www.w3.org/1999/xhtml/vocab#> .
:
  dcterm:title    "Temperature forecast for Galway, Ireland" ;
  xhv:stylesheet  <http://5stardata.info/style.css> ;
  dcterm:data     "2012-01-22"^^xsd:date ;
  dcterm:creator  <https://github.com/mhausenblas> .
:Galway
  rdf:type        meteo:Place ;
  meteo:forecast  :forecast20101113 ;
  meteo:forecast  :forecast20101114 ;
  meteo:forecast  :forecast20101115 .
:forecast20101113
  meteo:predicted    "2010-11-13T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101113 .
:forecast20101114
  meteo:predicted    "2010-11-14T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101114 .
:forecast20101115
  meteo:predicted    "2010-11-15T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101115 .
:temp20101113  meteo:celsius  "2"^^xsd:decimal .
:temp20101114  meteo:celsius  "4"^^xsd:decimal .
:temp20101115  meteo:celsius  "7"^^xsd:decimal .
```

Figura 5.8 – Publicação “4 Estrelas” (Turtle)

A classificação “5 Estrelas” denota a inserção das informações no universo de Dados Conectados. No caso do exemplo utilizado, foi feita uma conexão dos dados publicados na página com dados publicados sobre a cidade de Galway na DBPedia, utilizando a propriedade “owl:sameAs”, que indica que as duas URIs representam uma mesma coisa. Foi também criado um novo recurso (“#temp”) que faz a conexão de “temperatura” com informações definidas na DBpedia.

As figuras 5.9 e 5.10 apresentam as novas inserções, em relação ao exemplo de “4 Estrelas”.

```
<html  xmlns:xsd ="http://www.w3.org/2001/XMLSchema#"
       xmlns:dcterms="http://purl.org/dc/terms/"
       xmlns:meteo="http://purl.org/ns/meteo#">
<h1 property="dcterms:title">Temperature forecast for Galway,Ireland</h1>
<div id="data" about="#Galway" typeof="meteo:Place">
<span rel="owl:sameAs"
      resource="http://dbpedia.org/resource/Galway">
</span>
<table border="1px">
  <tr>
    <th>Day</th>
    <th>
      <div about="#temp">Lowest
        <a rel="rdfs:seeAlso"
           href="http://en.wikipedia.org/wiki/Temperature"
           resource="http://dbpedia.org/resource/Temperature">
        Temperature
        </a>
          (<span rel="owl:sameAs"
                resource="http://dbpedia.org/resource/Celsius">
                &deg;C</span>)
      </div>
    </th>
  </tr>
...
```

Figura 5.9 – Publicação “5 Estrelas” (código HTML + RDFa)

```
@base <http://5stardata.info/gtd-4.html> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix dbp: <http:/dbpedia.org/resource/> .
@prefix meteo: <ttp://purl.org/ns/meteo> .
@prefix xhv: <http://www.w3.org/1999/xhtml/vocab#> .
:
  dcterm:title    "Temperature forecast for Galway, Ireland" ;
  xhv:stylesheet  <http://5stardata.info/style.css> ;
  dcterm:data     "2012-01-22"^^xsd:date ;
  dcterm:creator  <https://github.com/mhausenblas> .
:Galway
  rdf:type        meteo:Place ;
  owl:sameAs      dbp:Galway  ;
  meteo:forecast  :forecast20101113 ;
  meteo:forecast  :forecast20101114 ;
  meteo:forecast  :forecast20101115 .
:forecast20101113
  meteo:predicted    "2010-11-13T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101113 .
:forecast20101114
  meteo:predicted    "2010-11-14T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101114 .
:forecast20101115
  meteo:predicted    "2010-11-15T00:00:00Z"^^xsd:dateTime ;
  meteo:temperature  :temp20101115 .
:temp
  owl:sameAs   dbr:Celsius ;
  rdfs:seeAlso dbr:Temperature .
:temp20101113  meteo:celsius  "2"^^xsd:decimal .
:temp20101114  meteo:celsius  "4"^^xsd:decimal .
:temp20101115  meteo:celsius  "7"^^xsd:decimal .
```

Figura 5.10 – Publicação “5 Estrelas” (Turtle)

5.3 LINKED DATA API

Um dos 4 princípios de Dados Conectados estabelece que quando é feito o acesso a uma URI de um recurso, por exemplo, uma escola, sejam retornadas

informações relevantes sobre o recurso identificado (dereferenciação). Essas informações podem estar armazenadas, por exemplo, em arquivos estáticos contendo o conjunto de triplas que o servidor Web entende que sejam relevantes como informações sobre o recurso. Porém, muitas vezes, essas informações são construídas de forma dinâmica a partir de consultas a um SPARQL *endpoint*. Isso implica que os usuários das consultas SPARQL devem possuir um entendimento do esquema dos dados e dos vocabulários utilizados, nem sempre de fácil compreensão.

A idéia da Linked Data API (LD API) [49] é fornecer uma forma fácil de acessar Dados Conectados via Web, permitindo que conjuntos de recursos sejam expostos como URIs, fáceis de consultar, podendo ser filtradas, paginadas, ordenadas, utilizando parâmetros de consulta bastante simples. A LD API suporta diversos formatos de resultados, incluindo JSON, XML, RDF/XML e Turtle.

A LD API é uma especificação para um camada intermediária de software que se apoia sobre um SPARQL *endpoint* e oferece uma Web API para acessar os dados. O mapeamento é configurado pelo administrador do servidor de dados e atende a padrões de nomenclatura, como, por exemplo, http://orcamento.dados.gov.br/doc/{ano}/ItemDespesa, onde {ano} pode ser substituído de forma a obter os resultados específicos para uma determinada data: http://orcamento.dados.gov.br/doc/2013/ItemDespesa. A figura 5.11 apresenta a consulta que seria necessária para a obtenção das triplas relativas a essa informação.

```
PREFIX loa: <http://vocab.e.gov.br/2013/09/loa#>
SELECT ?item
WHERE {
?item loa:temExercicio [loa:identificador 2013]. }  OFFSET 0 LIMIT 6
```

Figura 5.11 – Consulta SPARQL do orçamento federal

Dessa forma, podem ser especificados padrões de URIs que aceitam como parâmetros informações que são definidas na URI para acesso aos recursos. Por exemplo, o site que permite o acesso a dados do orçamento federal brasileiro [50] aceita os padrões de URIs apresentados na figura 5.12.

A LD API é uma especificação de código aberto e existem alguns produtos que a implementam, entre eles, o Elda [51] da Epimorphics.

```
/doc/{ano}/Acao
/doc/{ano}/Acao/{codigo}
/doc/{ano}/Atividade
/doc/{ano}/Atividade/{codigo}
/doc/{ano}/CategoriaEconomica
/doc/{ano}/CategoriaEconomica/{codigo}
/doc/{ano}/ElementoDespesa
/doc/{ano}/ElementoDespesa/{codigo}
/doc/{ano}/Esfera
/doc/{ano}/Esfera/{codigo}
/doc/{ano}/Exercicio
/doc/{ano}/Exercicio/{identificador}
/doc/{ano}/FonteRecursos
/doc/{ano}/FonteRecursos/{codigo}
/doc/{ano}/Funcao
/doc/{ano}/Funcao/{codigo}
/doc/{ano}/GrupoNatDespesa
/doc/{ano}/GrupoNatDespesa/{codigo}
/doc/{ano}/IdentificadorUso
/doc/{ano}/IdentificadorUso/{codigo}
/doc/{ano}/ItemDespesa
/doc/{ano}/ItemDespesa/{codigo}
/doc/{ano}/ModalidadeAplicacao
/doc/{ano}/ModalidadeAplicacao/{codigo}
/doc/{ano}/OperacaoEspecial
/doc/{ano}/OperacaoEspecial/{codigo}
/doc/{ano}/Orgao
/doc/{ano}/Orgao/{codigo}
/doc/{ano}/PlanoOrcamentario
/doc/{ano}/PlanoOrcamentario/{codigo}
/doc/{ano}/Programa
/doc/{ano}/Programa/{codigo}
/doc/{ano}/Projeto
/doc/{ano}/Projeto/{codigo}
/doc/{ano}/ResultadoPrimario
/doc/{ano}/ResultadoPrimario/{codigo}
/doc/{ano}/Subfuncao
/doc/{ano}/Subfuncao/{codigo}
/doc/{ano}/Subtitulo
/doc/{ano}/Subtitulo/{codigo}
/doc/{ano}/UnidadeOrcamentaria
/doc/{ano}/UnidadeOrcamentaria/{codigo}
```

Figura 5.12 – Padrões de URIs da Linked Data API do orçamento federal

5.4 EXEMPLOS

Nesta seção será apresentada uma seleção de exemplos de implementações de Dados Conectados.

5.4.1 DBPEDIA

A DBpedia [52] trata a Wikipedia como um banco de dados e tem como objetivo extrair informações estruturadas da Wikipedia e tornar essas informações disponíveis na Web. A DBpedia permite fazer consultas sofisticadas aos dados estruturados contidos nos artigos da Wikipedia e relacionar os diferentes conjuntos de dados na Web aos artigos da Wikipedia. Os artigos da Wikipedia consistem principalmente de texto livre, mas também contêm diversos tipos de informações estruturadas, tais como caixas de informações, informações de categorização, imagens, coordenadas geográficas e *links* para páginas da Web.

O projeto DBpedia extrai vários tipos de informações estruturadas de edições da Wikipedia em 125 idiomas e combina essas informações em uma grande base de conhecimento. Cada entidade (recurso) no conjunto de dados da DBpedia é denotado por uma URI dereferenciável, na forma "http:dbpedia.org/resource/{nome}", onde "{nome}" é derivado da URL do artigo origem da Wikipedia, que tem a forma "http://en.wikipedia.org/wiki/{nome}". Assim, cada entidade da DBpedia está conectada diretamente a um artigo da Wikipedia. Cada {nome} de entidade DBpedia retorna uma descrição de um recurso na forma de um documento Web.

Por exemplo, o recurso "http:dbpedia.org/resource/Tim_Berners-Lee" da DBpedia está relacionado ao artigo "http://en.wikipedia.org/wiki/Tim_Berners-Lee" da Wikipedia. As figuras 5.13 e 5.14 apresentam, respectivamente, a página da Wikipedia de Tim Berners-Lee e a caixa de informações, em destaque. As figuras 5.15 e 5.16 apresentam, respectivamente, a página da DBpedia de Tim Berners-Lee (recurso Tim_Berners-Lee) e o destaque de algumas das propriedades e valores extraídos da página da Wikipedia. As informações da DBpedia podem ser obtidas em diferentes formatos. A figura 5.17 apresenta parte das triplas em formato Turtle.

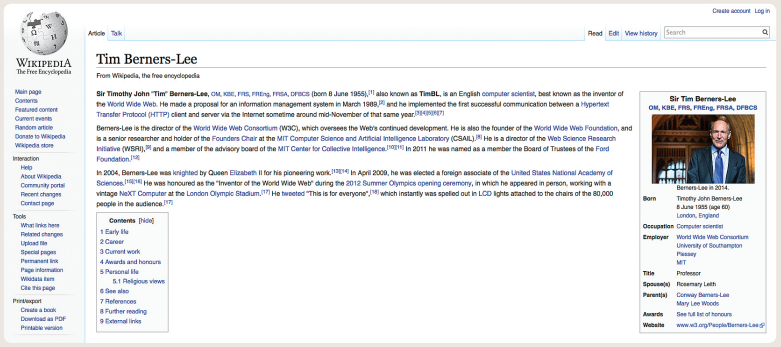


Figura 5.13 – Página de Tim Berners-Lee na Wikipedia

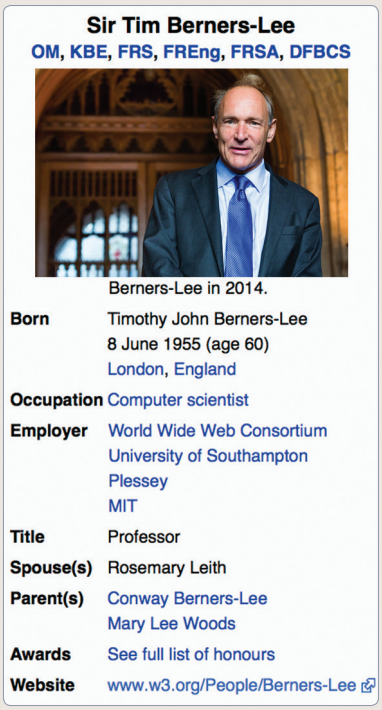


Figura 5.14 – Caixa de informações da página de Tim Berners-Lee na Wikipedia

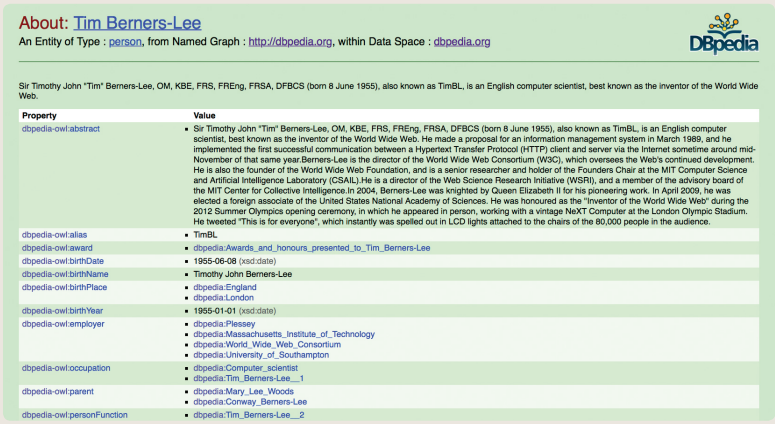


Figura 5.15 – Página de Tim Berners-Lee na DBpedia

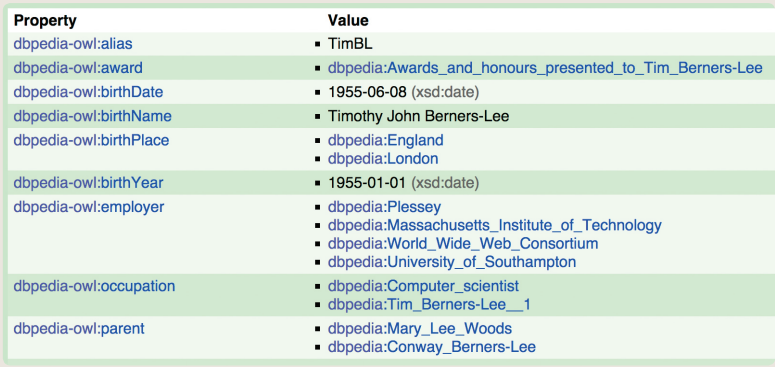


Figura 5.16 – Página de Tim Berners-Lee na DBpedia (destaque)

```
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix dbpedia:    <http://dbpedia.org/resource/> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dc:     <http://purl.org/dc/elements/1.1/> .
@prefix dbpprop:    <http://dbpedia.org/property/> .
dbpedia:Tim_Berners-Lee
  rdf:type foaf:Person ;
  foaf:name  "Sir Tim Berners-Lee"@en ;
  dbpprop:awards "*OM  \n*KBE  \n*OBE  \n*RDI  \n*FRS  \n*FREng"@en ;
  foaf:depiction
   <http://commons.wikimedia.org/wiki/Special:FilePath/Tim_Berners-Lee_ 2012.jpg>;
  dbpprop:birthName    "Timothy John Berners-Lee"@en ;
  dbpprop:dateOfBirth  "1955-06-08+02:00"^^xsd:date ;
  dbpprop:placeOfBirth "London, England"@en ;
  dbpprop:occupation   dbpedia:Computer_scientist ;
  dbpedia-owl:employer
    dbpedia:Massachusetts_Institute_of_Technology ,
    dbpedia:World_Wide_Web_Consortium ,
    dbpedia:University_of_Southampton ,
    dbpedia:Plessey ;
  dbpedia-owl:title    "Professor"@en ;
  dbpprop:partner      "Rosemary Leith"@en ;
  dbpedia-owl:parent
    dbpedia:Mary_Lee_Woods ,
    dbpedia:Conway_Berners-Lee ;
  dc:description       "British computer scientist, best known as the inventor of
the World Wide Web" ;
...
  dbpedia:Mary_Lee_Woods  dbpprop:children  dbpedia:Tim_Berners-Lee .
  dbpedia:Conway_Berners-Lee  dbpprop:children  dbpedia:Tim_Berners-Lee .
  dbpedia:World_Wide_Web_Consortium  dbpprop:leaderName
       dbpedia:Tim_Berners-Lee .
  dbpedia:World_Wide_Web  dbpprop:inventor  dbpedia:Tim_Berners-Lee .
  dbpedia:Libwww  dbpprop:author  dbpedia:Tim_Berners-Lee .
  dbpedia:ENQUIRE  dbpprop:inventor  dbpedia:Tim_Berners-Lee .
  dbpedia:WorldWideWeb  dbpedia-owl:developer  dbpedia:Tim_Berners-Lee .
```

Figura 5.17 – Página de Tim Berners-Lee na DBpedia (Turtle)

Os dados contidos na DBpedia também podem ser obtidos por meio de consultas a um SPARQL *endpoint* [53], implementado com a aplicação Virtuoso da Openlink. Os dados podem ser acessados por meio de diversas formas, incluindo o construtor de consultas Leipzig [54], o construtor de consultas interativo Openlink (iSPARQL) [55], e o explorador de consultas SNORQL [34].

Existem diversas aplicações que fazem uso das informações contidas na DBpedia, como, por exemplo, o DBpedia Spotlight [57], que a partir de um texto fornecido em estilo livre sugere *links* para os recursos da DBpedia. Tomando como exemplo o texto da figura 5.18, o Spotlight sugere o texto da figura 5.19 com diversos *links* para a DBpedia (em português), incluindo um *link* para o recurso que descreve a World Wide Web [58].

```
Sir Timothy John Berners-Lee KBE, OM, FRS (TimBL ou TBL) (Londres,
8 de junho de 19551 ) é um físico britânico, cientista da computação
e professor do MIT. É o criador da World Wide Web (Rede Mundial de
Computadores - Internet), tendo feito a primeira proposta para sua
criação a 25 de março de 1989.2 Em 25 de dezembro de 1990, com a ajuda
de Robert Cailliau e um jovem estudante do CERN, implementou a primeira
comunicação bem-sucedida entre um cliente HTTP e o servidor através da
internet.
```

Figura 5.18 – Texto livre submetido ao DBpedia Spotlight

```
Sir Timothy John Berners-Lee KBE, OM, FRS (TimBL ou TBL) (Londres,
8 de junho de 19551 ) é um físico britânico, cientista da computação
e professor do MIT. É o criador da World Wide Web (Rede Mundial de
Computadores - Internet), tendo feito a primeira proposta para sua
criação a 25 de março de 1989.2 Em 25 de dezembro de 1990, com a ajuda
de Robert Cailliau e um jovem estudante do CERN, implementou a primeira
comunicação bem-sucedida entre um cliente HTTP e o servidor através da
internet.
```

Figura 5.19 – Texto enriquecido sugerido pelo DBpedia Spotlight

5.4.2 DADOS CONECTADOS DA BIBLIOTECA DA ESPANHA

O projeto datos.bne.es [59] da Biblioteca Nacional da Espanha e do Grupo de Engenharia de Ontologias [60] da Universidade Politécnica de Madri tem como público-alvo tanto os usuários finais da biblioteca, como os desenvolvedores de software especialistas em Web Semântica. É um projeto piloto que visa propor uma abordagem e uma exploração dos dados bibliográficos diferente dos catálogos tradicionais, propondo uma experiência de navegação nova para os diferentes recursos da biblioteca, enriquecendo seus próprios dados com dados externos.

Os dados podem ser acessados a partir do portal ou de uma interface SPARQL [61]. Além disso, *dumps* completos dos dados também podem ser obtidos. Os dados definidos na aplicação seguem o modelo de dados de uma ontologia [62] desenvolvida pelo Grupo de Engenharia de Ontologias da Universidade Politécnica de Madri.

As figuras 5.20 e 5.21 apresentam, respectivamente, a página inicial do projeto e a página de busca de autores, de obras ou de temas. As figuras 5.22 e 5.23 apresentam a página resultante da busca pelo autor "Miguel de Cervantes" e a seleção da sua obra "Dom Quijote de la Mancha". A figura 5.24 apresenta um destaque da figura 5.23, onde é possível ver como as informações são apresentadas como um conjunto de valores de propriedades do recurso "Dom Quijote de la Mancha".

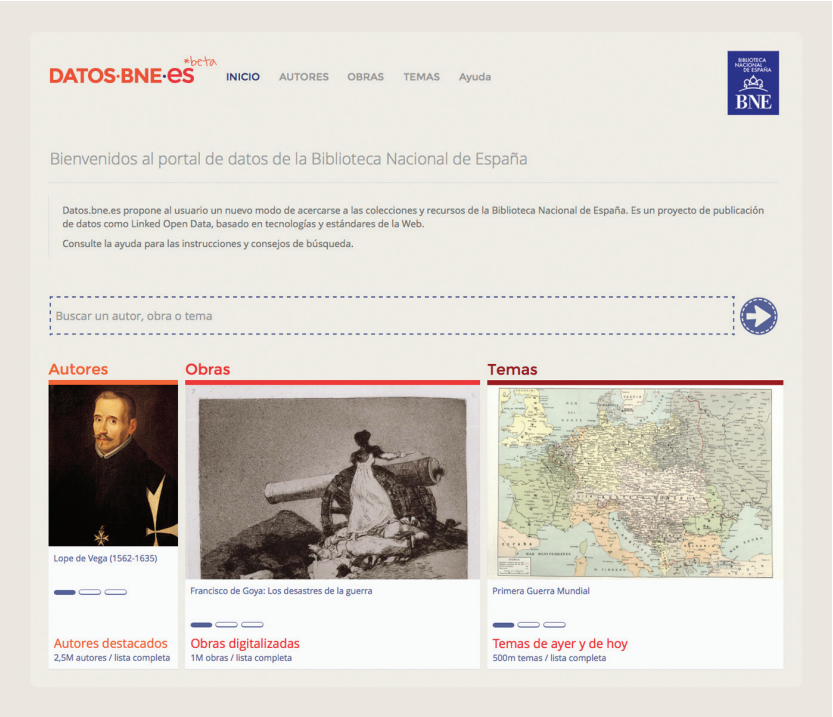


Figura 5.20 – Página inicial do *site* de dados abertos da Biblioteca da Espanha

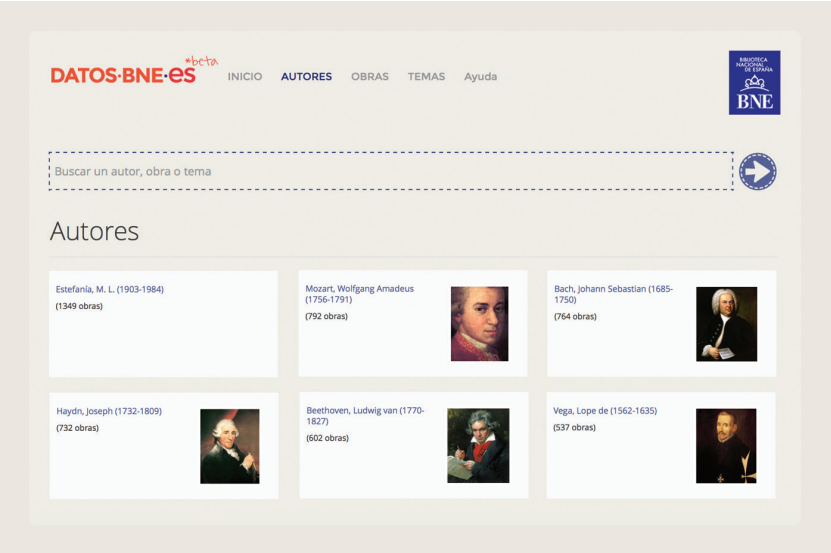


Figura 5.21 – Página de busca do *site* de dados abertos da Biblioteca da Espanha

Figura 5.22 – Página de resultado de busca pelo autor "Miguel de Cervantes"

Don Quijote de la Mancha

Acceder a la entrada en el catálogo

Título	Don Quijote de la Mancha;compuesto por Miguel de Cervantes Saavedra; [copia e ilustración de Gonzalo Bosch Bierge];
Lugar de publicación	1931-1945
Tipo de recurso	Manuscrito
Tema	Manuscritos iluminados
Descripción física o extensión	2 cajas ([VI h., p. 1-560; 561-1188])
Dimensiones	34 x 25 cm
Otras características físicas	il.
Nota	Texto enmarcado, subrayado en tinta roja o amarilla, letra de tipo gótico de Gonzalo Bosch Bierge Firmas autógrafas Juan Sedó-Peris-Mencheta (Sig: M-C; R.-Bibl. II; MC-4/36) Texto enmarcado por orlas policromadas, diferentes en todas las páginas, predominando los motivos decorativos geométricos. Iniciales iluminadas, algunas historiadas. Gran número de ilustraciones coloreadas a la aguada, algunas a página entera y a doble página como las de las p. 1, 5, 75, 144-145, 286-287, 340-341, 536-637, 691y 1076-1077. Acuarela con el busto de Miguel de Cervantes a página entera, insertado en una mandorla, a cada lado y enmarcándola, se representa a dos ángeles de pie, uno sobre la esfera del Nuevo Mundo y otro sobre la del Viejo Continente; en la parte superior y de menor tamaño los bustos del Quijote y Sancho Panza (p. 5)
Nota	La Caja 1, contiene los preliminares y los capítulos I-XXVI (VI h., p. 1-560), continuando los capítulos XXVII-LII (p. 561-1188) en la Caja 2. Paginación moderna en lápiz
Nota de contenido	Elogios autógrafos y firmados dedicados a la obra, intitulados "Hojas de oro de personalidades ilustres..." de la medicina, literatura, arte, bibliotecas, teatro, etc.: Gregorio Marañón, Ramírez Tomé, Rafael Marquina, José María Acevedo, Pascual Marquina, Pedro de Répide, Manuel Fontdevila, Joaquín Ciervo, González Marín, G. de Falla, Magariños, José María Pemán, Jacinto Benavente, B. Morales San Martín, J. Lamote de Grignón, J. Segrelles, Pilar Mancó, Manuel Abril, Juan Sedó Peris-Mencheta, Francisco Lloréns, Esteban Domenech, Francisco Rodríguez Marín, M. Santa María, Eusebi Bosch Humet, Charles Beaulieux, Henriette Berman, Federico Ruiz Morcuende, Carl H. Milam, J. Domínguez Bordona, Eduardo Ibarra Rodríguez, José Francés, J. Moreno Carbonero, Gonzalo de Reparaz, Margarita Xirgu, Francisco Vindel, Dr. Bauer, Henry Thomas, Miguel Artigas, Ezio Levi, etc. (h. III-VIv). Nota a modo de prólogo titulada "Nota del transcriptor" (p. 9-10)
Referencia bibliográfica	Biblioteca Cervantina de Juan Sedó Peris-Mencheta (Barcelona). Homenaje tributado por la Sección de Manuscritos a la de Impresos de dicha Biblioteca, con motivo de la adquisición para la misma del ejemplar número mil de ediciones del Ingenioso Hidalgo D. Quijote de la Mancha; precedido de una introducción por [D. Juan Sedó Peris-Mencheta]..., Barcelona, 1942 Plaza Escudero, Luis María. Catálogo de la colección cervantina Sedó. Barcelona, 1953
Nota sobre publicación	En p. 3: ... yo Gonzalo Bosch Bierge he podido terminar este manuscrito ... a los 14 años y seis meses de mi empresa. Deo gratias En h. II: Este manuscrito policromado, con ilustraciones y miniaturas de Gonzalo Bosch Bierge, fue empezado el día 23 de abril de 1931
Forma del contenido	Texto (visual)
Tipo de medio	sin mediación
Idioma	http://lexvo.org/id/iso639-3/spa

Descarga en otros formatos

RDF (Turtle) MARC 21 (Catálogo)

Figura 5.23 – Página de visualização da obra “Don Quijote de la Mancha”

Don Quijote de la Mancha

Acceder a la entrada en el catálogo

Título	Don Quijote de la Mancha;compuesto por Miguel de Cervantes Saavedra; [copia e ilustración de Gonzalo Bosch Bierge];
Lugar de publicación	1931-1945
Tipo de recurso	Manuscrito
Tema	Manuscritos iluminados
Descripción física o extensión	2 cajas ([VI h., p. 1-560; 561-1188])
Dimensiones	34 x 25 cm

Figura 5.24 – Página de visualização da obra “Don Quijote de la Mancha” (destaque)

5.4.3 BBC THINGS

BBC Things [63] usa tecnologias de Web Semântica que permitem o acesso a coisas que são importantes para o público do grupo BBC, tais como pessoas, lugares, organizações, competições esportivas, tópicos de estudo, etc.. A BBC tem um conjunto próprio de ontologias [64] que definem os tipos de coisas que podem ser consultados em BBC Things.

As figuras 5.25 e 5.26 apresentam a página HTML resultante de uma consulta ao tópico "Madonna" e o código exportado em Turtle. O código HTML retornado pela consulta tem as informações da triplas associadas ao recurso embutidas em RDFa.

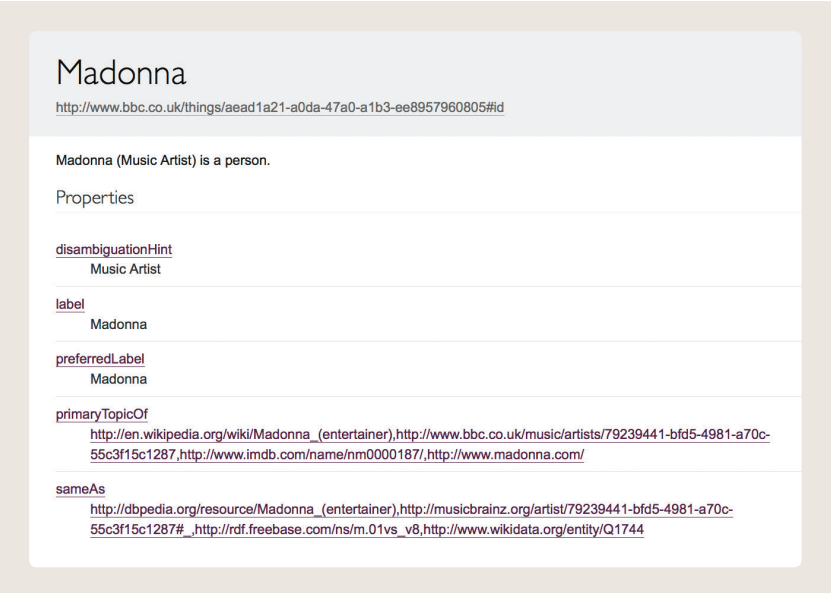


Figura 5.25 – Informações de Madonna no BBC Things (HTML)

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix core: <http://www.bbc.co.uk/ontologies/coreconcepts/> .
@prefix mo: <http://purl.org/ontology/mo/> .
@base <http://www.bbc.co.uk/things/> .
:aead1a21-a0da-47a0-a1b3-ee8957960805#id>
  a mo:MusicArtist ;
  rdfs:label "Madonna"@en-gb ;
  core:preferredLabel "Madonna"@en-gb ;
  core:disambiguationHint "Music Artist"@en-gb ;
  core:primaryTopicOf
    <http://www.bbc.co.uk/music/artists/79239441-bfd5-4981-a70c-55c3f15c1287> ,
    <http://www.madonna.com/> ,
    <http://en.wikipedia.org/wiki/Madonna_(entertainer)> ,
    <http://www.imdb.com/name/nm0000187/> ;
  core:sameAs
    <http://dbpedia.org/resource/Madonna_(entertainer)> ,
    <http://www.wikidata.org/entity/Q1744> ,
    <http://musicbrainz.org/artist/79239441-bfd5-4981-a70c-55c3f15c1287> ,
    <http://rdf.freebase.com/ns/m.01vs_v8> .
```

Figura 5.26 – Informações de Madonna no BBC Things (Turtle)

5.4.4 ORÇAMENTO FEDERAL DO GOVERNO BRASILEIRO

A Lei Orçamentária Anual (LOA) estima as receitas e fixa as despesas do Governo para o ano subseqüente, e fornece a previsão das ações planejadas pelo governo para um exercício financeiro, que corresponde ao período de 1º de janeiro a 31 de dezembro. O Orçamento anual visa concretizar os objetivos e metas propostas no Plano Plurianual (PPA), segundo as diretrizes estabelecidas pela Lei de Diretrizes Orçamentárias (LDO).

O Governo Federal publicou o orçamento federal em formato RDF (no período de 2000 a 2013), a fim de dar maior transparência e acesso aos dados, de forma que cidadãos e organizações interessados em conhecer melhor os dados do orçamento federal possam realizar pesquisas e análises de forma veloz e eficiente. A estratégia adotada no projeto "Orçamento Federal em Formato Aberto", do Governo Federal, foi criar uma ontologia [65] com base na classificação da despesa do orçamento federal, contemplando as categorias e conceitos especificados no Manual Técnico de Orçamento [66].

As informações do orçamento podem ser obtidas a partir do catálogo de dados abertos do governo federal [67]. É possível obter acesso a diversos conjuntos de dados:

- Dumps dos dados por ano (2000 a 2013).
- Acesso aos diversos padrões de URIs [68], por exemplo, a lista dos itens de despesa de 2014: http://orcamento.dados.gov.br/doc/2014/ItemDespesa.
- Consultas ao SPARQL *endpoint* [69] do projeto, por exemplo:

```
SELECT ?nomeFuncao (sum (?DotacaoInicial)) as ?soma
WHERE
{ ?item rdf:type loa:ItemDespesa .
  ?item loa:temExercicio [loa:identificador 2014] .
  ?item loa:temFuncao [rdfs:label ?nomeFuncao] .
  ?item loa:valorDotacaoInicial ?DotacaoInicial. }
GROUP BY ?nomeFuncao
```

5.4.5 BIO2RDF

Pesquisadores biológicos são frequentemente confrontados com a inevitável tarefa de ter que integrar seus resultados experimentais com resultados de outros pesquisadores. Essa tarefa geralmente envolve uma tediosa busca manual, e a assimilação de diversas coleções isoladas de dados de ciências da vida, hospedados por vários provedores independentes. Esses provedores incluem organizações tais como o Centro Nacional para Informação Biotecnológica (NCBI) e o Instituto de Bioinformática Europeu (EBI), que fornecem dezenas de conjuntos de dados enviados por usuários, assim como instituições menores, como o grupo Donaldson que publica iRefIndex3, um banco de dados de interações moleculares agregados de 13 fontes de dados diferentes.

Com milhares de bancos de dados biológicos e centenas de milhares de conjuntos de dados, a capacidade de encontrar dados relevantes é dificultada por interfaces de banco de dados não padronizadas e um número enorme de formatos de dados heterogêneos. Além disso, metadados sobre esses provedores de dados biológicos (informações do conjunto de dados, fontes dos dados, controle de versão, informações de licenciamento, data de criação, etc.) são muitas vezes difíceis de obter. Tomados em conjunto, a incapacidade de navegar facilmente pelos dados disponíveis apresenta uma barreira enorme para sua reutilização.

Bio2RDF [70] é um projeto de dados abertos que usa tecnologias da Web Semântica para tornar possível a consulta distribuída de dados integrados de ciências da vida. Desde o seu início, Bio2RDF tem feito uso de RDF e RDFS para unificar a representação de dados obtidos de diversos campos (moléculas, enzimas, doenças, etc.) e dados biológicos armazenados em formatos heterogêneos (arquivos-texto, arquivos CSV, formatos específicos de bancos de dados, XML, etc.). Uma vez convertidos para RDF, este dados biológicos podem ser consultados usando SPARQL, que pode ser utilizado para a realização de consultas federadas em diversos SPARQL *endpoints*.

Atualmente, Bio2RDF comporta 35 conjuntos de dados, entre eles o DrugBank [71], que é um banco de recursos bioinformáticos e químico-informáticos que combina dados detalhados sobre drogas (química, farmacologia, farmacêutica, etc.), com informações tais como sequência, estrutura, etc. Cada um dos conjuntos de dados possui um SPARQL *endpoint*

e utiliza conjuntos de vocabulários próprios desse domínio. Em BioPortal [72] é possível visualizar a enorme lista de vocabulários específicos para cada uma da áreas.

As Figuras 5.27 e 5.28 apresentam, respectivamente os resultados de uma consulta a "Glutahione" no site DrugBank [73] e no SPARQL endpoint [74] associado.

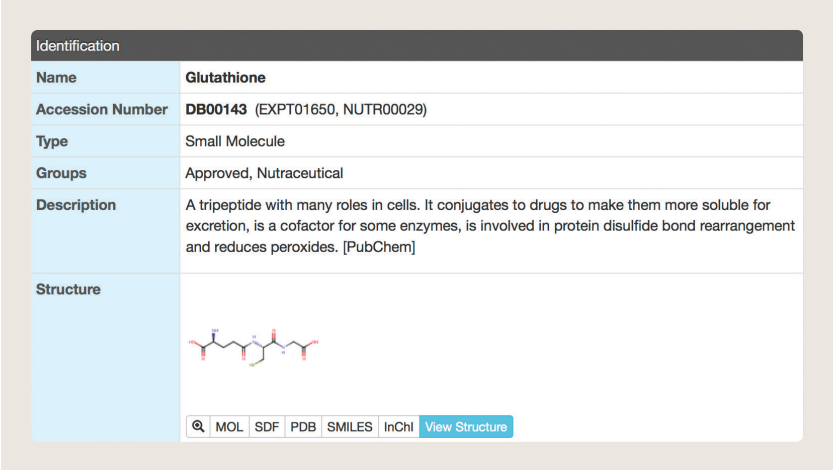


Figura 5.27 – Informações de "Glutahione" no *site* DrugBank

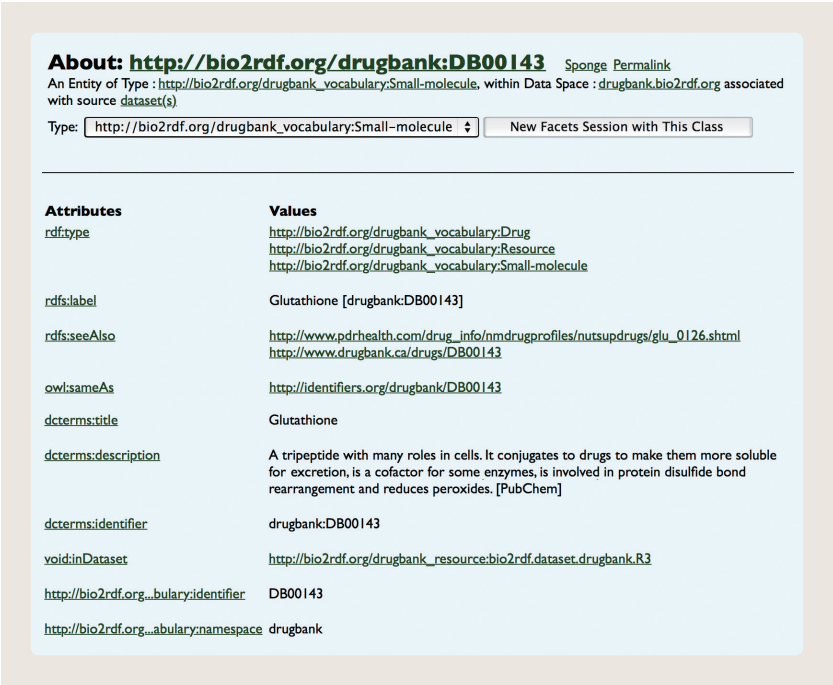
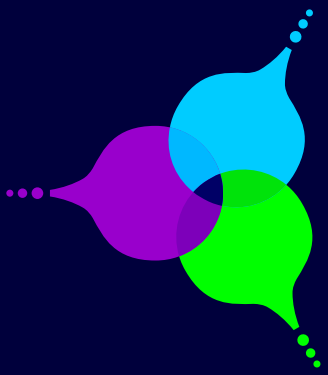


Figura 5.28 – Informações de "Glutahione"(SPARQL *endpoint*)



VOCABULÁRIOS E ONTOLOGIAS

Capítulo 6

Nos capítulos anteriores vimos as tecnologias da Web Semântica que são relacionadas à forma como sintaticamente os metadados podem ser agregados às informações. Uma parte fundamental para que se alcance o objetivo de fazer com que o significado pretendido pelo publicador dos dados seja o mesmo que o significado entendido pelo consumidor dos dados é o uso de vocabulários que possuam uma semântica bem definida. Um fator que facilita enormemente o sucesso desse objetivo é a utilização de vocabulários de referência, ou vocabulários de uso mais comum. Neste documento os termos vocabulário e ontologia são utilizados de forma intercambiável. Não existe na literatura uma separação clara dos dois conceitos, sendo que, em muitos casos, o termo vocabulário é utilizado para o caso de ontologias mais simples.

Fazendo uma analogia com a comunicação cotidiana, podemos perceber que cada grupo particular de pessoas utiliza vocabulários específicos nas suas conversas, nas suas trocas de mensagens. As pessoas se agrupam por diversos motivos: localização geográfica, relação familiar, profissional, social, e um sem número de outras situações e características. Dois médicos conversando sobre um determinado procedimento cirúrgico utilizarão termos de um vocabulário específico. Da mesma forma, a descrição dos valores nutricionais de um pacote de salgadinhos utiliza o seu próprio vocabulário. Diversos vocabulários foram criados nas conversas das redes sociais, incluindo vocabulários gráficos, como os *emoticons*.

Para que o cenário da Web Semântica fique completo é preciso que se estabeleça um conjunto de vocabulários de referência, como maneira de facilitar a comunicação dos metadados. O leitor deve estar ciente de que para cada publicação específica, deve ser feita uma busca de vocabulários existentes que possam ser utilizados. Existem alguns catálogos que podem auxiliar o usuário na busca de ontologias, entre eles o LOV [75], o BioPortal [72] e o JoinUp [76]. Nos casos em que nenhum vocabulário satisfaça as necessidades de expressividade dos metadados, uma nova ontologia pode ser criada, com o cuidado em reutilizar o maior número possível de elementos de ontologias já existentes, evitando assim a duplicação de referências diferentes aos mesmos conceitos.

Cada vocabulário é descrito por um documento apontado por uma URI. Por exemplo, o vocabulário FOAF tem a URI “http://xmlns.com/foaf/0.1/”. A URI de referência às classes e às propriedades de cada um dos vocabulários é construída a partir da concatenação da URI do vocabulário com o nome

da respectiva classe ou propriedade. Por exemplo, a propriedade “name” de FOAF tem a URI “http://xmlns.com/foaf/0.1/name”. Nas seções a seguir serão apresentados alguns dos vocabulários de referência mais populares e históricos.

6.1 DUBLIN CORE

A Dublin Core Metadata Initiative [77] (DCMI) teve o seu evento inicial em 1995 e é uma das primeiras iniciativas a pensar na definição de um vocabulário para a descrição de metadados, tendo como premissas que as descrições tenham independência em relação à sintaxe e tenham uma semântica bem definida. É um vocabulário adequado para a descrição de recursos (documentos), e começou como um conjunto básico de 15 propriedades, que tinham uma ideia análoga a dos elementos de catalogação de bibliotecas:

- title – nome do recurso.
- creator – nome do criador do recurso.
- subject – tópico do recurso.
- description – descrição do recurso, que pode ser um resumo, um sumário, etc.
- publisher – entidade responsável por tornar o recurso disponível.
- contributor – nome dos contribuidores do recurso.
- date – data associada ao recurso.
- type – tipo do recurso.
- format – formato de arquivo, meio físico de armazenamento ou dimensões do recurso.
- identifier – uma referência única ao recurso dentro de um determinado contexto.
- source – fonte que originou o recurso, como, por exemplo, o resultado de um serviço.
- language – linguagem do recurso.
- relation – relação entre dois recursos.
- coverage – cobertura temporal ou espacial do recurso, por exemplo, uma jurisdição.
- rights – direitos associados ao recurso.

Esse vocabulário tem como uma de suas utilizações a documentação de páginas da Web. A figura 6.1 apresenta um exemplo de uso do vocabulário com a utilização do *tag* <meta> de HTML. As propriedades de Dublin Core são muito reutilizadas em outras ontologias.

```
<head>
<link rel="schema.DC" href="http://purl.org/dc/elements/1.1/">
<meta name="DC.title" content="Guia de Web Semântica">
<meta name="DC.creator" content="Carlos Laufer">
<meta name="DC.subject" content="Web Semântica">
<meta name="DC.subject" content="Dados Conectados">
<meta name="DC.description" content="Introdução ao Ecossistema e às
Tecnologias de Web Semântica e Dados Conectados">
<meta name="DC.publisher" content="W3C Brasil">
<meta name="DC.date" content="21/01/2015">
</head>
```

Figura 6.1 – Exemplo de Uso do Vocabulário Dublin Core

Com a evolução do projeto, o vocabulário foi estendido com a criação de um conjunto maior de propriedades e de classes [78]. Entre as propriedades acrescentadas estão: “abstract”, “audience”, “conformsTo”, “hasVersion”, “replaces”, “requires”, etc. Entre as classes criadas estão “FileFormat”, “Jurisdiction”, “Location”, “ProvenanceStatem”, “MediaType”, etc.

6.2 FOAF

O vocabulário Friend of a Friend [79] (FOAF) surgiu no início do ano 2000 e é um vocabulário adequado para a definição de metadados sobre pessoas, seus interesse, seus relacionamentos e suas atividades. O vocabulário [13] tem um conjunto central de classes (primeira letra maiúscula) e propriedades (primeira letra minúscula):

- Agent – coisas que realizam algo, que podem ser pessoas, organizações, robôs, etc. Tem como subclasses “Person”, “Organization” e “Group”.
- Person – entidade central do vocabulário: representa as pessoas.
- name – cadeia de caracteres com um nome.
- title – forma de tratamento, como, por exemplo, “Sr.”, “Sra.”, etc.
- img – uma imagem que representa uma pessoa.
- depiction (depicts) – relaciona alguma coisa a uma imagem.

- familyName – descreve parte do nome de uma pessoa (sobrenome).
- givenName – descreve parte do nome de uma pessoa (primeiro nome).
- knows – relaciona duas pessoas.
- based_near – relação espacial entre duas coisas.
- age – a idade da pessoa.
- made (maker) – alguma coisa feita por alguém.
- primaryTopic (primaryTopicOf) – principal tópico de um documento.
- Project – um projeto.
- Organization – uma organização.
- Group – um grupo.
- Member – um membro de um grupo.
- Document – um documento.
- Image – uma imagem.

Além desse núcleo central, existe uma extensão de classes e propriedades relacionadas às características sociais da Web, tais como, “nick”, “mbox” (e-mail), “homepage”, “publications”, “account”, etc.

Uma das ideias criadas com o vocabulário FOAF foi a de que cada pessoa poderia definir um arquivo, chamado arquivo FOAF, com suas informações pessoais, por exemplo, o arquivo FOAF de Tim Berners-Lee [80]. A figura 6.2 reproduz parte das informações contidas nesse arquivo, em formato Turtle.

```
@prefix dc11: <http://purl.org/dc/elements/1.1/> .
@prefix cc: <http://creativecommons.org/ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
<http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf>
  cc:license <http://creativecommons.org/licenses/by-nc/3.0/> ;
  dc11:title "Tim Berners-Lee's FOAF file" ;
  a foaf:PersonalProfileDocument ;
  foaf:maker <http://www.w3.org/People/Berners-Lee/card#i> ;
  foaf:primaryTopic <http://www.w3.org/People/Berners-Lee/card#i> .
<http://www.w3.org/People/Berners-Lee/card#i>
  foaf:img <http://www.w3.org/Press/Stock/Berners-Lee/2001-europaeum-eighth.jpg>;
  foaf:knows
    <http://bblfish.net/people/henry/card#me> ,
    <http://danbri.org/foaf#danbri> ,
    <http://dbpedia.org/resource/John_Gage> ,
    <http://dbpedia.org/resource/John_Klensin> ,
    <http://dbpedia.org/resource/John_Markoff> ,
    <http://dbpedia.org/resource/John_Seely_Brown> ,
    <http://dbpedia.org/resource/Tim_Bray> ,
    <http://dig.csail.mit.edu/2007/wiki/people/JoeLambda#JL> ,
    <http://dig.csail.mit.edu/2007/wiki/people/RobertHoffmann#RMH> ,
    <http://norman.walsh.name/knows/who#norman-walsh> ,
    <http://www.cs.umd.edu/~hendler/2003/foaf.rdf#jhendler> ,
    <http://www.mindswap.org/2004/owl/mindswappers#Jennifer.Golbeck> ,
    <http://www.w3.org/People/Jacobs/contact.rdf#IanJacobs> .
```

Figura 6.2 – Informações do arquivo FOAF de Tim Berners-Lee (Turtle)

6.3 SKOS

Simple Knowledge Organization System (SKOS) [81] é uma recomendação do W3C que fornece uma maneira de representar esquemas de classificação, tais como vocabulários controlados, taxonomias e tesauros. Muitos desses sistemas compartilham uma estrutura semelhante e são usados em aplicações similares. SKOS captura grande parte dessas semelhanças e as torna explícitas, de forma a permitir o compartilhamento de dados entre essas diversas aplicações. Em SKOS, conceitos podem ser identificados por meio de URIs, com rótulos em uma ou mais línguas, e ser documentados com diferentes tipos de notas. Os diversos conceitos podem ser relacionados semanticamente entre si, em hierarquias e redes de associação informais, e também agrupados em esquemas conceituais. A seguir são apresentados os principais elementos de SKOS [82].

- Concept
  É o elemento fundamental de SKOS. É uma classe que define que um determinado recurso é um conceito.

```
@prefix ex: <http://example.com> .
ex:animals rdf:type skos:Concept.
```

- prefLabel, altLabel
  São rótulos que permitem fazer referência aos conceitos em linguagem natural: “prefLabel” é o rótulo preferido a ser exibido e “altLabel”é um rótulo alternativo, utilizado, por exemplo, para sinônimos.

```
ex:animals rdf:type skos:Concept ;
  skos:prefLabel "animals" ;
  skos:altLabel "creatures" .
```

- broader, narrower
  O significado de um conceito é definido não apenas pelas palavras em linguagem natural dos seus rótulos, mas, também, por suas ligações com outros conceitos do vocabulário. “broader” indica que um conceito

é mais abrangente que um outro (um conceito que engloba o outro conceito). "narrower" é o inverso de "broader".

```
ex:animals rdf:type skos:Concept ;
  skos:prefLabel "animals" ;
  skos:narrower ex:mammals .
ex:mammals rdf:type skos:Concept ;
  skos:prefLabel "mammals" ;
  skos:broader ex:animals .
```

- related
  Indica uma relação associativa entre dois conceitos.

```
ex:ornithology rdf:type skos:Concept ;
  skos:prefLabel "ornithology" .
ex:birds rdf:type skos:Concept ;
  skos:prefLabel "birds" ;
  skos:related ex:ornithology .
```

- note
  Indica uma observação a respeito do conceito. "note" indica uma nota genérica, mas existe a possibilidade de qualificar diferentes tipos de observações, utilizando "scopeNote", "historyNote", "editorialNote" e "changeNote", relativas ao escopo, história, questões editoriais, e mudanças efetuadas, respectivamente.

```
ex:microwaveFrequencies
  skos:scopeNote "Used for frequencies between 1GHz to 300Ghz"@en .
ex:tomato
  skos:changeNote "Moved from 'fruits' to 'vegetables'" .
```

- definition
  Fornece uma definição do conceito.

```
ex:documentation  rdf:type skos:Concept ;
  skos:definition "the process of storing and retrieving information
in all fields of knowledge" .
```

- ConceptScheme
  Conceitos podem ser criados e usados como entidades independentes. Entretanto, conceitos geralmente vêm em vocabulários cuidadosamente organizados, como tesauros ou esquemas de classificação. Esses esquemas podem ser representados em SKOS por meio da classe “ConceptScheme”.

```
ex:animalThesaurus rdf:type skos:ConceptScheme ;
  dct:title "Simple animal thesaurus" ;
  dct:creator ex:antoineIsaac .
ex:mammals rdf:type skos:Concept ;
  skos:inScheme ex:animalThesaurus .
ex:cows rdf:type skos:Concept;
  skos:broader ex:mammals ;
  skos:inScheme ex:animalThesaurus .
ex:fish rdf:type skos:Concept ;
  skos:inScheme ex:animalThesaurus .
```

SKOS fornece ainda meios para fazer o mapeamento entre diferentes esquemas conceituais, a partir da relação entre os diversos conceitos desses esquemas. Esse mapeamento qualifica o grau de proximidade dessas relações.

6.4 SCHEMA.ORG

Schema.org fornece uma coleção de vocabulários que podem ser utilizados para embutir metadados em páginas Web, e são entendidas pelas principais máquinas de busca: Google, Microsoft, Yandex e Yahoo!. Os metadados podem ser embutidos utilizando microdados, RDFa ou JSON-LD.

Atualmente existe mais de uma centena de vocabulários definidos em Schema.

org, segundo uma estrutura hierárquica. Cada um dos vocabulários define um tipo. “Thing“ é o tipo raiz e é relativo a qualquer item genérico. Um item do tipo “Thing” aceita propriedades como, por exemplo, “nome”, “descrição”, “imagem” e “URL”. No segundo nível da hierarquia existem dez tipos especializados, cada um com seu próprio vocabulário:

- Action
- BroadcastService
- CreativeWork
- Event
- Intangible
- MedicalEntity
- Organization
- Person
- Place
- Product

Cada um desses tipos tem suas próprias especializações, como, por exemplo,

- Organization
- Airline
- Corporation
- EducationalOrganization
- GovernmentOrganization
- LocalBusiness
- NGO
- PerformingGroup
- SportsOrganization

O site Schema.org [83] apresenta detalhadamente cada um dos vocabulários com a apresentação de exemplos e as codificações em microdados, RDFa e JSON-LD. As figuras 6.3 e 6.4 apresentam um exemplo utilizando os vocabulários para pessoas e endereços.

```
Jane Doe
<img src="janedoe.jpg" alt="Photo of Jane Joe"/>
Professor
20341 Whitworth Institute
405 Whitworth
Seattle WA 98052
(425) 123-4567
<a href="mailto:jane-doe@xyz.edu">jane-doe@illinois.edu</a>
Jane's home page:
<a href="http://www.janedoe.com">janedoe.com</a>
Graduate students:
<a href="http://www.xyz.edu/students/alicejones.html">Alice Jones</a>
<a href="http://www.xyz.edu/students/bobsmith.html">Bob Smith</a>
```

Figura 6.3 – Exemplo de utilização de vocabulários do Schema.org (sem metadados)

```
<div itemscope itemtype="http://schema.org/Person">
  <span itemprop="name">Jane Doe</span>
  <img src="janedoe.jpg" itemprop="image" alt="Photo of Jane Joe"/>
  <span itemprop="jobTitle">Professor</span>
  <div itemprop="address" itemscope
       itemtype="http://schema.org/PostalAddress">
    <span itemprop="streetAddress">
      20341 Whitworth Institute
      405 Whitworth
    </span>
    <span itemprop="addressLocality">Seattle</span>,
    <span itemprop="addressRegion">WA</span>
    <span itemprop="postalCode">98052</span>
  </div>
  <span itemprop="telephone">(425) 123-4567</span>
  <a href="mailto:jane-doe@xyz.edu"
     itemprop="email">jane-doe@xyz.edu</a>
  Jane's home page:
  <a href="http://www.janedoe.com"
     itemprop="url">janedoe.com</a>Graduate students:
  <a href="http://www.xyz.edu/students/alicejones.html"
     itemprop="colleague">Alice Jones</a>
  <a href="http://www.xyz.edu/students/bobsmith.html"
     itemprop="colleague">Bob Smith</a>
</div>
```

Figura 6.4 – Exemplo de utilização de vocabulários do Schema.org (microdados)

6.5 PROV

Quando dados são publicados, existem conjuntos de metadados que são relativos a aspectos indiretos em relação aos dados em si, como, por exemplo, qual a licença de uso dos dados, direitos do uso, etc. Não são metadados que visam esclarecer a estrutura ou a semântica da informação propriamente dita. Um tipo importante de informação sobre dados publicados refere-se à proveniência dos dados, quem gerou esses dados, como foram gerados, quais foram as fontes, etc.

Proveniência é obter informações sobre entidades, atividades e pessoas envolvidas na produção de alguma coisa, que podem ser utilizadas para as

avaliações sobre confiabilidade, qualidade, veracidade, etc. A ontologia PROV [84] de documentos define um modelo, serializações correspondentes e outras definições para permitir o intercâmbio de informações de proveniência na Web.

O modelo de proveniência definido por PROV considera três elementos básicos: entidades, atividades e agentes. Esses três elementos estão conectados por meio de um conjunto de relações. Por exemplo, "uma entidade (uma página Web, um arquivo, etc.) foi gerada por uma atividade associada a um determinado agente". A figura 6.5 apresenta um diagrama com uma relação básica entre os três elementos. O diagrama utiliza uma forma gráfica definida por PROV para a representação da ontologia.

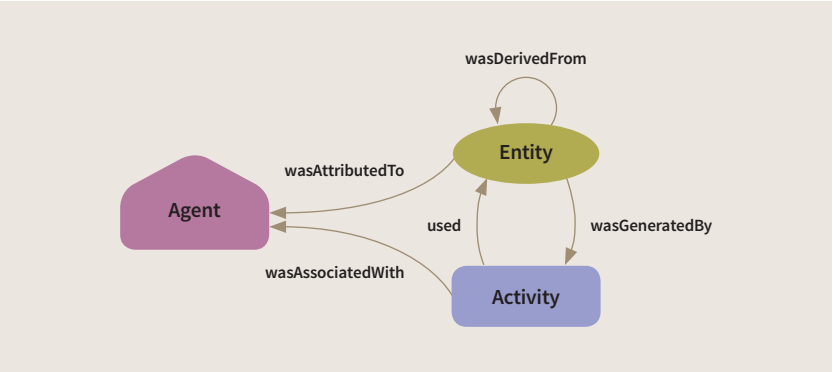


Figura 6.5 – Entidade, Atividade e Agente (PROV)

Como forma de ilustrar os princípios básicos da ontologia vamos supor a seguinte geração de dados [85]:

```
Derek, que trabalha na empresa "Chart Generators Inc", fez uma composição
de informações provenientes de um conjunto de dados e de uma lista de
regiões geográficas, e a partir do resultado dessa composição gerou uma
ilustração na forma de um gráfico.
```

Essa descrição poderia ser representada pelo diagrama da figura 6.6 e pela representação em Turtle da figura 6.7.

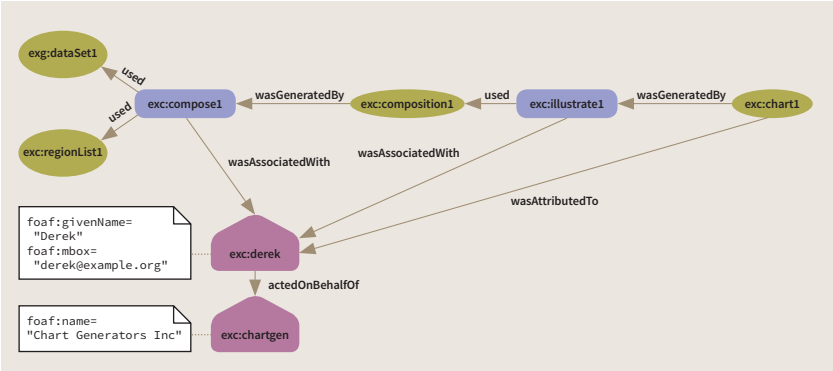


Figura 6.6 – Exemplo de PROV (diagrama)

```
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix prov: <http://www.w3.org/ns/prov#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix exc:  <http://examplec.com/> .
@prefix exg:  <http://exampleg.com/> .
exg:dataset1     a prov:Entity .
exc:regionList   a prov:Entity .
exc:composition1 a prov:Entity .
exc:chart1       a prov:Entity .
exc:compose1     a prov:Activity .
exc:illustrate1  a prov:Activity .
exc:compose1       prov:used              exg:dataset1 ;
                   prov:used              exc:regionList1 .
exc:composition1   prov:wasGeneratedBy    exc:compose1 .
exc:illustrate1    prov:used              exc:composition1 .
exc:chart1         prov:wasGeneratedBy    exc:illustrate1 .
exc:compose1       prov:wasAssociatedWith exc:derek .
exc:illustrate1    prov:wasAssociatedWith exc:derek .
exc:chart1         prov:wasAttributedTo   exc:derek .
exc:derek        a prov:Agent ,
                   prov:Person ;
                   foaf:givenName "Derek"^^xsd:string ;
                   foaf:mbox      <mailto:derek@example.org> .
exc:chartgen     a prov:Agent ,
                   a prov:Organization ;
                   foaf:name "Chart Generators Inc" .
exc:derek          prov:actedOnBehalfOf   exc:chartgen .
```

Figura 6.7 – Exemplo de PROV (Turtle)

6.6 DCAT

As máquinas de busca foram uma das primeiras aplicações que surgiram com a Web. É natural que em um ambiente onde muitas informações são publicadas exista uma forma de catalogação que permita que uma busca seja realizada. A difusão crescente da ideia de abertura e publicação de dados torna necessário que se tenha uma forma de catalogar os conjuntos de dados e suas formas de distribuição.

DCAT [86] é uma recomendação do W3C que permite a criação de catálogos com descrições de conjuntos de dados. O uso de uma forma padrão para a

descrição dos catálogos aumenta a capacidade de descoberta e permite que as aplicações possam encontrar metadados distribuídos em diferentes catálogos. Permite, também, a publicação descentralizada de catálogos e facilita a pesquisa federada de conjuntos de dados publicados em diferentes *sites*.

DCAT tem três classes principais:

- dcat:Catalog – representa o catálogo.
- dcat:Dataset – representa um conjunto de dados em um catálogo.
- dcat:Distribution – representa uma forma de acesso ao conjunto de dados, como, por exemplo, uma página da Web, um arquivo para download, um Web Service, uma Web API, um SPARQL endpoint, etc.

A figura 6.8 apresenta o diagrama de Classes de DCAT. É importante observar como a ontologia reutiliza propriedades dos vocabulários Dublin Core, FOAF e SKOS.

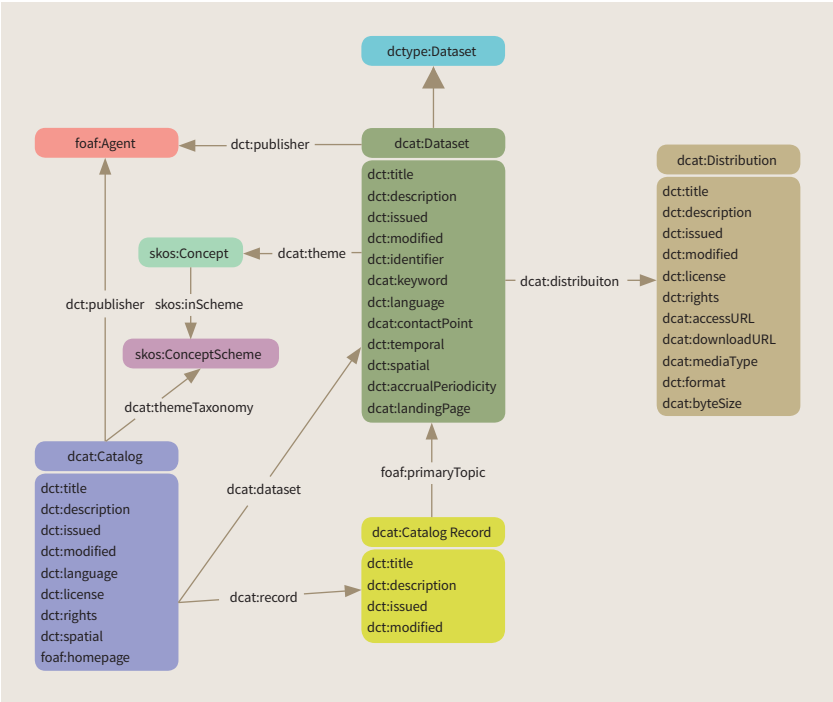


Figura 6.8 – Diagrama de classes de DCAT

A figura 6.9 apresenta um exemplo de catálogo com 2 conjuntos de dados representados em DCAT (Turtle).

```
@base <http://example.org/> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix dcat: <http://www.w3.org/ns/dcat#> .
:catalog   a          dcat:Catalog ;
    dct:title         "Imaginary Catalog" ;
    rdfs:label        "Imaginary Catalog" ;
    foaf:homepage     <http://example.org/catalog> ;
    dct:publisher     :transparency-office ;
    dct:language      <http://id.loc.gov/vocabulary/iso639-1/en>  ;
    dcat:dataset      :dataset-001 , :dataset-002  .
:transparency-office  a  foaf:Organization ;
    rdfs:label        "Transparency Office" .
:dataset-001   a      dcat:Dataset ;
    dct:title         "Imaginary dataset" ;
    dcat:keyword      "accountability" , “transparency" , "payments" ;
    dct:issued        "2011-12-05"^^xsd:date ;
    dct:modified      "2011-12-05"^^xsd:date ;
    dcat:contactPoint <http://example.org/transparency-office/contact> ;
    dct:temporal      <http://reference.data.gov.uk/id/quarter/2006-Q1> ;
    dct:spatial       <http://www.geonames.org/6695072> ;
    dct:publisher     :finance-ministry ;
    dct:language      <http://id.loc.gov/vocabulary/iso639-1/en>  ;
    dct:accrualPeriodicity <http://purl.org/linked-data/sdmx/2009/code#freq-W> ;
    dcat:distribution :dataset-001-csv .
:dataset-001-csv  a  dcat:Distribution ;
    dcat:downloadURL  <http://www.example.org/files/001.csv> ;
    dct:title         "CSV distribution of imaginary dataset 001" ;
    dcat:mediaType    "text/csv" ;
    dcat:byteSize     "5120"^^xsd:decimal .
:dataset-002  a  dcat:Dataset ;
    dcat:landingPage  <http://example.org/dataset-002.html> ;
    dcat:distribution :dataset-002-csv ,  :dataset-002-xml .
:dataset-002-csv  a  dcat:Distribution ;
    dcat:downloadURL  <http://example.org/files/dataset-002.csv> ;
    dcat:mediaType    "text/csv" .
:dataset-002-xml  a  dcat:Distribution ;
    dcat: downloadURL <http://example.org/files/dataset-002.xml> ;
    dcat:mediaType    "text/xml" .
:catalog dcat:themeTaxonomy :themes .
:themes   a   skos:ConceptScheme ;
    skos:prefLabel   "A set of domains to classify documents" .
:dataset-001 dcat:theme :accountability  .
:accountability  a  skos:Concept ;
    skos:inScheme   :themes ;
    skos:prefLabel  "Accountability" .
```

Figura 6.9 – Exemplo de Catálogo

O *site* do Open Data Institute [87] apresenta um exemplo [88] de como criar um catálogo de uma forma simples, por meio da criação de uma página Web, com *links* para o *download* de arquivos. Nesse caso, os metadados com as informações do catálogo são embutidos no código HTML, utilizando RDFa.

O projeto “show me the money” [89] fornece informações sobre o mercado de empréstimos no Reino Unido. As figuras 6.10 e 6.11 apresentam,mrespectivamente, a página do projeto onde é possível fazer o *download* dos dados [90], e parte do código HTML dessa página, com as informações do catálogo em DCAT embutidas em RDFa.

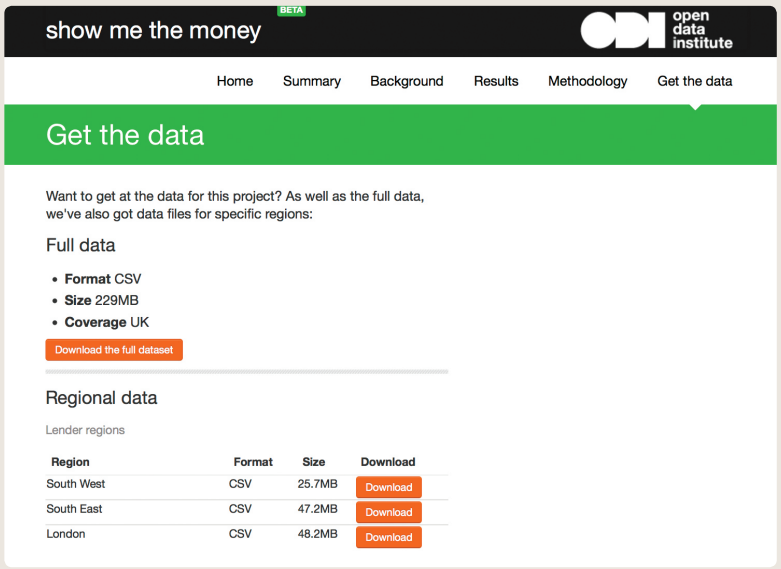
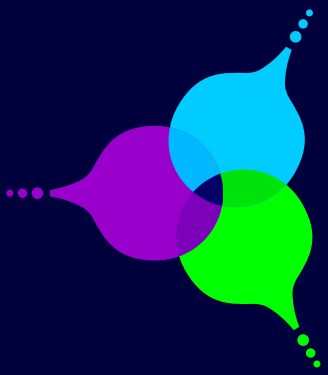


Figura 6.10 – Projeto "show me the money" (página de *download* de arquivos)

```
<!DOCTYPE html>
<html prefix="dct: http://purl.org/dc/terms/
              rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
              dcat: http://www.w3.org/ns/dcat#
              odrs: http://schema.theodi.org/odrs#">
...
<div resource='http://p2p.labs.theodi.org/download'
     typeof='dcat:Catalog'>
  <p>Want to get at the data for this project? As well as the full data,
     we've also got data files for specific regions:</p>
  <span content='http://id.loc.gov/vocabulary/iso639-1/en'
        property='dct:language'></span>
  <div property='dcat:dataset'
       resource='http://p2p.labs.theodi.org/download/#full'
       typeof='dcat:Dataset'>
    <h3 property='dct:title'>Full data</h3>
    <div property='dcat:distribution' typeof='dcat:Distribution'><ul>
      <li><strong>Format</strong><span
          content='text/csv'
          property='dcat:mediaType'>CSV</span></li>
      <li><strong>Size</strong><span
          content='240585277' datatype='xsd:decimal'
          property='dcat:byteSize'>229MB</span></li>
      <li><strong>Coverage</strong><span
          content='http://dbpedia.org/resource/United_Kingdom'
          property='dct:spatial'>UK</span</li></ul>
    <p><a class='btn btn-primary'
          href='http://4feb814f800c80231150-
8876dec7442c825b72049e4e2a169344.r56.cf3.rackcdn.com/complete.no.postcodes.
csv.zip'
          property='dcat:accessURL'>Download the full dataset</a></p>
  </div>
</div>
...
```

Figura 6.11 – Exemplo de Catálogo embutido em página Web (RDFa)



CONCLUSÃO

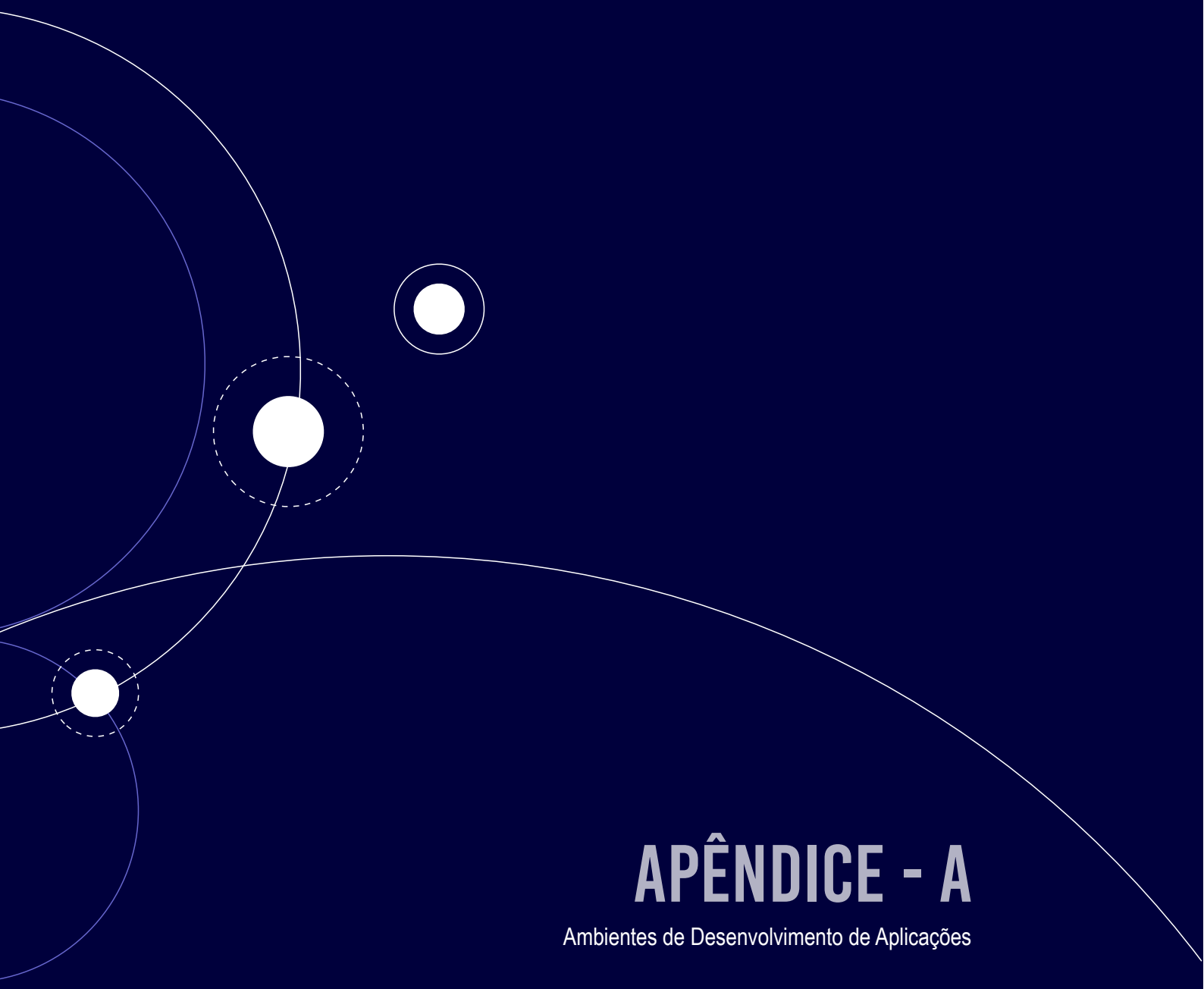
Capítulo 7

Neste guia foram apresentados os conceitos que definem a World Wide Web, a evolução pela qual ela tem passado, conceitos de semântica, metadados, o ecossistema da Web de Dados, as tecnologias da Web Semântica, as bases do conceito de Dados Conectados, e alguns dos vocabulários de referência mais utilizados. Além disso, a seguir é apresentado um apêndice com alguns exemplos de softwares atualmente utilizados em ambientes de desenvolvimento para a geração de aplicações que manipulam Dados Conectados.

Houve um tempo em que a Web já fervilhava em alguns domínios de pesquisa mas ainda não havia se popularizado e se tornado um tipo de tecnologia cotidiana. A Web Semântica encontra-se em um estágio intermediário em que alguns nichos já se aproveitam de suas potencialidades, como por exemplo, as máquinas de busca em conjunto com metadados embutidos em páginas de sites interessados em ser melhor compreendidos pelos robôs. A era do desenvolvimento das aplicações que tiram proveito desse novo modelo está apenas no seu início e será fundamental para a popularização da Web Semântica. Estamos ainda numa fase de publicação maciça de dados, cada vez mais intensa, mas com um número pequeno de aplicações. Porém, no mundo dos dispositivos móveis a importância das aplicações, ou apps, como são chamadas, é praticamente óbvia hoje em dia. Muitas pessoas quando querem pegar um táxi em uma grande metrópole utilizam de forma natural a ideia de dados publicados explorados por uma aplicação.

Este guia termina com a tradução de uma frase do livro "A Programmable Web: An Unfinished Work" [91], de Aaron Swartz, um dos grandes pesquisadores da Web e defensor da ideia de dados abertos, que tinha algumas posições críticas à visão da Web Semântica, mas também percebia as possibilidades de sucesso:

> "É fácil fazer piadas desse tipo de visão. Meu pai, quando via essas demos, costumava sempre me perguntar, 'Mas por que a sua torradeira precisa saber os preços das ações?'. E talvez, em última análise, não valham mesmo o esforço. Mas a Web Semântica é baseada em aposta, uma aposta de que uma vez que se forneça ao mundo as ferramentas que facilitem a colaboração e a comunicação, isso acarretará em possibilidades tão maravilhosas que nós dificilmente podemos imaginá-las agora. Claro, isso parece uma coisa maluca. Mas valeu a pena da última vez que eles fizeram essa aposta: acabamos com uma coisinha chamada World Wide Web. Vamos ver se eles podem fazer novamente."



APÊNDICE - A

Ambientes de Desenvolvimento de Aplicações

Este apêndice apresenta os resumos de alguns ambientes para o desenvolvimento de aplicações e para a distribuição de dados RDF. Informações mais detalhadas sobre cada um dos produtos podem ser obtidas a partir dos links fornecidos em cada resumo.

A.1 D2RQ

Atualmente, uma grande quantidade de dados encontra-se armazenada em bancos de dados relacionais. Eles são a fonte de alimentação de muitos sites, que expõem os dados desses bancos por meio de páginas na Web. A plataforma D2RQ [92] permite acessar bancos de dados relacionais como grafos RDF virtuais, em modo leitura, sem a necessidade de replicar esse banco de dados para um sistema de armazenamento nativo em RDF.

O método utiliza uma linguagem [93] para fazer o mapeamento dos dados contidos nas tabelas do banco relacional para um conjunto de triplas em RDF. Na seção 4.1, onde foram apresentados os conceitos de RDF, foi feita uma exposição inicial onde as células de uma tabela eram identificadas como triplas. As tabelas em si eram um conjunto de triplas de recursos de um determinado tipo. A linguagem da plataforma D2RQ aplica essa ideia de uma forma sofisticada permitindo que se faça uma definição apurada do conjunto de triplas.

Como forma de ilustrar esse mapeamento vamos supor um exemplo bem simples onde temos três tabelas (figuras A.1 a A.3) de um banco de dados sobre uma conferência:

- Paper – informações sobre artigos publicados.
- Person – informações sobre pessoas.
- Rel_Person_Paper – relação dos autores de um artigo.

PaperId	Title	...
1	Trusting Information Sources One Citizen at a Time	
...	...	...

Figura A.1 – Tabela “Paper”

PerId	Name	...
1	Yolanda Gil	
2	Varun Ratnakar	...
...		

Figura A.2 – Tabela “Person”

PersonId	PaperId
1	1
2	1
...	...

Figura A.3 – Tabela “Rel_Person_Paper”

A figura A.4 apresenta uma parte do mapeamento das tabelas relacionais para as triplas RDF:

- map:Database1
  Define a conexão com o banco de dados.

- map:PaperClassMap
  Define uma classe RDF, definindo qual o padrão de URIs que deverá ser utilizado para acessar um recurso desse tipo. No caso do exemplo, as triplas de um recurso associado a uma pessoa que tenha o identificador “1” na tabela “Person” será acessado pela URI “http://www.conference.org/conf2004/paper#Paper1”.

- map:paperTitle
  Define uma propriedade de uma classe, associando uma coluna de uma tabela a uma determinada propriedade de algum vocabulário. No caso do exemplo, a coluna “Title” da tabela “Paper” é relacionada a propriedade “title” do vocabulário Dublin Core.

- map:authorName
  Define também uma propriedade de uma classe, mas nesse caso a identificação da coluna é feita a partir da junção de tabelas. No caso do exemplo, para saber o nome dos autores de um artigo é preciso utilizar a tabela de relacionamentos “Rel_Person_Paper”, para identificar o

valor da propriedade. Por exemplo, o artigo "1" tem dois autores, o que implicará na criação de duas triplas com a propriedade "dc:creator" do vocabulário Dublin Core (figura A.5).

```
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
@prefix map: <file:///Users/d2r/example.ttl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
map:Database1 a d2rq:Database;
  d2rq:jdbcDSN "jdbc:mysql://localhost/iswc" ;
  d2rq:jdbcDriver "com.mysql.jdbc.Driver" ;
  d2rq:username "user ;
  d2rq:password "password" .
map:PaperClassMap a d2rq:ClassMap;
  d2rq:uriPattern
     "http://www.conference.org/conf2004/paper#Paper@@Paper.PaperID@@" ;
  d2rq:class :Paper ;
  d2rq:dataStorage map:Database1 .
map:paperTitle a d2rq:PropertyBridge ;
  d2rq:belongsToClassMap map:Paper ;
  d2rq:column "Paper.Title" ;
  d2rq:property dc:title .
map:authorName a d2rq:PropertyBridge ;
  d2rq:belongsToClassMap map:Paper ;
  d2rq:join "Paper.PaperID <= Rel_Person_Paper.PaperID" ;
  d2rq:join "Rel_Person_Paper.PersonID => Person.PerID" ;
  d2rq:datatype xsd:string ;
  d2rq:property dc:creator .
```

Figura A.4 – Exemplo de mapeamento da linguagem do D2RQ

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
http://www.conference.org/conf2004/paper#Paper1
  dc:title "Trusting Information Sources One Citizen at a Time" ;
  dc:creator "Yolanda Gil" ;
  dc:creator "Varun Ratnakar" .
```

Figura A.5 – Triplas geradas para um artigo

Além do acesso aos recursos por meio dos padrões de URIs definidos no mapeamento do banco de dados relacional, a plataforma D2RQ também oferece um SPARQL endpoint onde podem ser feitas consultas ao grafo RDF.

A.2 VIRTUOSO

OpenLink Virtuoso [94] é um servidor universal, um híbrido de Servidor de Aplicações Web e Sistema de Gerenciamento de Banco de Dados Objeto-Relacional (ORDBMS). Sua arquitetura permite a persistência de dados nos formatos relacional, RDF, XML, texto, documentos, Dados Conectados, etc. Além disso, ele também pode funcionar como um Servidor de Aplicações Web, bem como um *host* para Web Services.

Por meio de uma camada ODBC ou JDBC, o Virtuoso se conecta à maioria das plataformas de bancos de dados relacionais comerciais mais populares, incluindo Oracle, SQL Server, Progress, DB2, Sybase, CA-Ingres, Informix, etc. Uma das características principais do Virtuoso é o Sponger, que permite a transformação de dados não RDF para dados RDF em tempo de execução, similar ao grafo RDF virtual gerado pelo D2RQ. Seu objetivo é utilizar fontes de dados não RDF da Web como entrada (páginas em HTML, páginas com microformatos embutidos, dados provenientes de Web Services, Web APIs, etc.), e criar um grafo RDF como saída. Isso permite a exposição de fontes de dados não RDF na forma de Dados Conectados na Web. Para cada tipo de dado não RDF é possível instalar um cartridge que faz a extração e transformação dos dados para RDF. O produto já vem com uma grande quantidade de *cartridges* embutidos, mas o usuário pode construir um *cartridge* para um formato de dados específico.

De forma análoga ao D2RQ, o Virtuoso tem uma linguagem de mapeamento de dados relacionais para RDF [95]. A linguagem é uma extensão da linguagem de consulta SPARQL [96] e permite mapear, declarativamente, tabelas, colunas, linhas, etc. para classes, atributos, relacionamentos e instâncias definidas por esquemas RDF ou ontologias OWL. O Virtuoso também aceita o mapeamento de dados relacionais para RDF [97] utilizando R2RML [98], uma recomendação do W3C de uma linguagem para expressar os mapeamentos personalizados de bancos de dados relacionais para conjuntos de dados RDF. Além disso, o Virtuoso é um servidor de Dados Conectados que oferece um SPARQL endpoint e permite a configuração do esquema de URIs para acesso aos recursos, por meio do mapeamento das URIs para consultas SPARQL ao grafo RDF.

A.3 SESAME

Sesame [99] é um arcabouço em Java (código aberto) para armazenamento, inferência e consulta de dados RDF, extensível e configurável com relação a mecanismos de armazenamento, máquinas de inferência, formatos de arquivo RDF, linguagens de consulta e formatos de resultados de consulta. O acesso ao Sesame é feito via uma API que pode ser conectada a todas as principais soluções de armazenamento de RDF, e uma interface HTTP RESTful que oferece suporte ao protocolo SPARQL. Sesame também pode ser utilizado em conjunto com o ambiente de desenvolvimento Eclipse [100],

para o desenvolvimento de aplicações Web com bibliotecas de suporte para a manipulação de RDF.

Sesame tem duas interfaces principais de comunicação:

- Sail API (Camada de armazenamento e inferência)
  É uma API de interface de sistema, de baixo nível, para bancos de triplas RDF e máquinas de inferência. É uma forma de abstração dos detalhes de armazenamento, permitindo o uso de diversos tipos de armazenamento e de inferência. Dessa forma, diversos bancos de triplas, código aberto e comerciais, que implementam a Sail API podem ser acoplados ao Sesame.

- Repository API
  É uma API de mais alto nível utilizada na código das aplicações. Ela oferece diversos métodos para o *upload* de arquivos, consultas, e extração e manipulação de dados. Os repositórios podem ser locais ou remotos. Repositórios locais estão localizados na mesma máquina virtual Java. Repositórios remotos são utilizados segundo o modelo cliente-servidor, onde a aplicação se comunica com um servidor Sesame. Nos dois modos, a interface utilizada é a mesma, de tal forma que as aplicações podem ser desenvolvidas de forma transparente para os dois tipos de repositórios.

A figura A.6 apresenta parte de um código Java onde são incluídas triplas em um repositório. São criadas URIs para dois recursos, com duas triplas geradas para cada um deles.

```
import org.openrdf.model.vocabulary.RDF;
import org.openrdf.model.vocabulary.RDFS;
import org.openrdf.model.vocabulary.FOAF;
...
Repository rep = new SailRepository(new MemoryStore());
rep.initialize();
ValueFactory f = rep.getValueFactory();
URI alice = f.createURI("http://example.org/people/alice");
URI bob = f.createURI("http://example.org/people/bob");
Literal bobsName = f.createLiteral("Bob");
Literal alicesName = f.createLiteral("Alice");
try {
   RepositoryConnection con = rep.getConnection();
   try {
      con.add(alice, RDF.TYPE, FOAF.PERSON);
      con.add(alice, FOAF.NAME, alicesName);

      con.add(bob, RDF.TYPE, FOAF.PERSON);
      con.add(bob, FOAF.NAME, bobsName);
   }
   finally {
      con.close();
   }
}
```

Figura A.6 – Triplas geradas em um ambiente Sesame-Eclipse

A.4 JENA-FUSEKI

Jena [101] é um arcabouço em Java (código aberto) para o desenvolvimento de aplicações para Web Semântica e Dados Conectados. Ele provê uma API para extrair e inserir dados de grafos RDF, que são representados como modelos que podem ser alimentados com dados a partir de arquivos, bancos de dados, URLs, etc. Além disso, esses modelos também podem ser consultados via SPARQL. Jena oferece suporte a ontologias especificadas em OWL, com diversos raciocinadores internos, além de poder utilizar o raciocinador Pellet [102].

As figuras A.7 e A.8 apresentam, respectivamente, extratos de código da inserção de uma tripla em um modelo e o carregamento de dados em um modelo a partir da leitura de um arquivo contendo dados RDF definidos, por exemplo, em XML

```
static String personURI = "http://somewhere/JohnSmith";
static String fullName = "John Smith";
Model model = ModelFactory.createDefaultModel();
Resource johnSmith = model.createResource(personURI);
johnSmith.addProperty(VCARD.FN, fullName);
```

Figura A.7 – Tripla inserida em um modelo (Jena)

```
Model model = ModelFactory.createDefaultModel();
InputStream in = FileManager.get().open( inputFileName );
if (in == null) {
  throw new IllegalArgumentException(
   "File: " + inputFileName + " not found");
}
model.read(in, null);
```

Figura A.8 – Leitura de dados de arquivo (Jena)

Jena tem um componente denominado TDB (banco de triplas) que é utilizado para a persistência de grafos RDF, via uma API específica. O acesso via SPARQL é provido pelo Fuseki, que é um servidor que oferece suporte ao protocolo SPARQL sobre HTTP.

A.5 PUBLISHMYDATA

PublisMyData [103] é um serviço de hospedagem da Dados Conectados oferecido pela empresa Swirrl [104], localizada no Reino Unido. Ele oferece um conjunto de funcionalidades e formas configuráveis de exibição dos dados, que se propõe a ser uma forma mais atraente para o consumo de Dados Conectados. Além disso, ele oferece interfaces de acesso para desenvolvedores, como forma de facilitar a criação de aplicações sobre os dados.

A vantagem de um serviço como esse, como em geral para serviços terceirizados, é a não necessidade de ter uma instalação própria com todos os custos de infraestrutura e, principalmente, de pessoal especializado para a configuração e manutenção dos sistemas.

O serviço é utilizado, por exemplo, pela cidade de Hampshire no Reino Unido. A figura A.9 apresenta a página inicial do catálogo dos conjuntos de dados do Hampshire Hub [105].

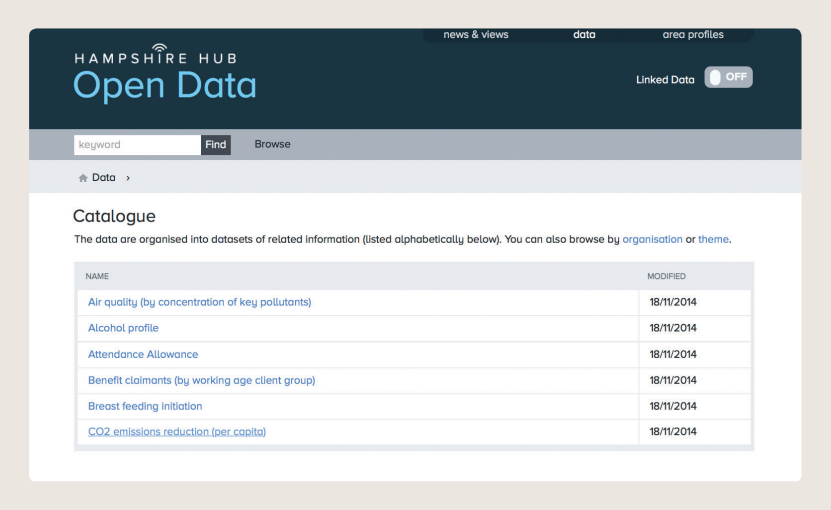


Figura A.9 – New Hampshire Hub (catálogo)

A figura A.10 apresenta a visualização do conjunto de dados "COS emissions reduction" na forma de uma planilha. Além disso, são também apresentadas informações sobre o conjunto dos dados (figura A.11), como, por exemplo, o publicador dos dados, a licença de uso, etc.

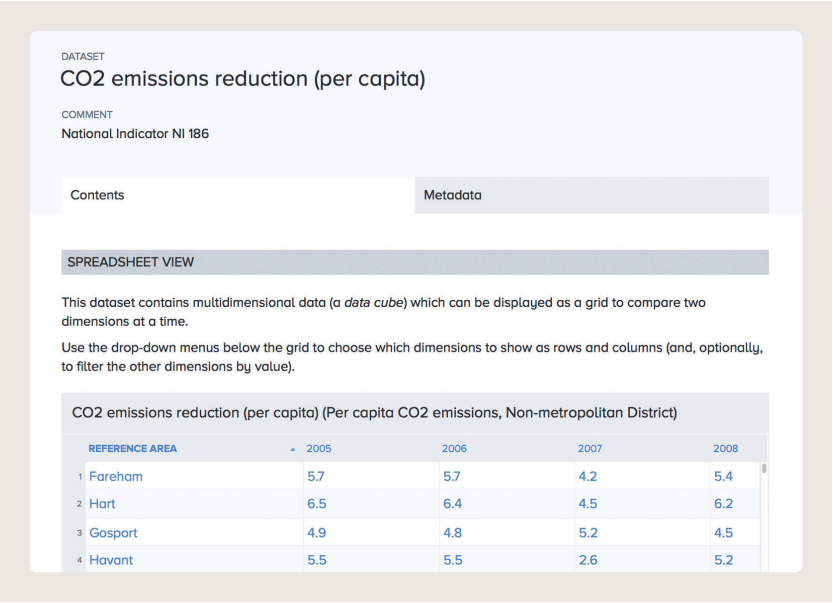


Figura A.10 – Dados do conjunto "COS emissions reduction"

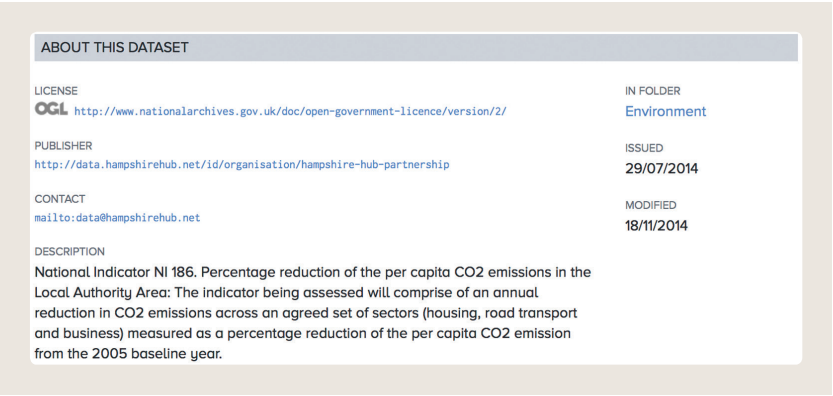


Figura A.11 – Metadados do conjunto "COS emissions reduction"

A.6 PLATAFORMA DE PUBLICAÇÃO DE DADOS CONECTADOS EPIMORPHICS

A Plataforma de Publicação de Dados Conectados Epimorphics [106] inclui uma base de triplas e um SPARQL *endpoint*, além de acesso aos recursos RDF por meio da Linked Data API. A plataforma oferece um serviço totalmente hospedado (baseado na Amazon Web Services, por exemplo) e gerenciado para a publicação de Dados Conectados, podendo, alternativamente, ser instalada em uma infraestrutura própria do cliente. Cada instância da plataforma é executada em um conjunto de máquinas dedicadas para cada cliente.

A plataforma oferece:

- Uma interface nos moldes da Linked Data API, fornecendo acesso aos dados em diferentes formatos, tanto para uso por desenvolvedores quanto para o público em geral, na forma de páginas Web.
- Um banco de triplas para armazenamento de dados RDF.
- Um SPARQL *endpoint*.
- Um gerenciador de *uploads*, para que os usuários possam carregar seus próprios dados.

A plataforma é utilizada por diversos sites do governo do Reino Unido, incluindo, environment.data.gov.uk, landregistry.data.gov.uk, entre outros.

A seguir são apresentadas algumas figuras referentes ao site de dados ambientais do governo do Reino Unido. A figura A.12 apresenta a página inicial do site. A figura A.13 apresenta um conjunto de dados referentes a qualidade da água nas diversas regiões do reino. Nessa página são listadas diversas informações incluindo aplicações desenvolvidas utilizando os dados do conjunto (figura A.14) e informações sobre a API para acesso aos dados (figura A.15). Uma das informações da API são os padrões de URI para acesso aos recursos. Na figura A.16 são apresentadas as triplas de um recurso recuperado pela URI http://environment.data.gov.uk/doc/bathing-water/ukl1702-36800, onde ukl1702-36800 é o identificador do distrito "Neath Port Talbot" localizado na região de "Wales". A seleção do *link* em "Neath Port Talbot" recupera informações sobre o distrito, armazenados na base de dados de mapeamento do Reino Unido, "Ordnance Survey" [107] (figura A.17).

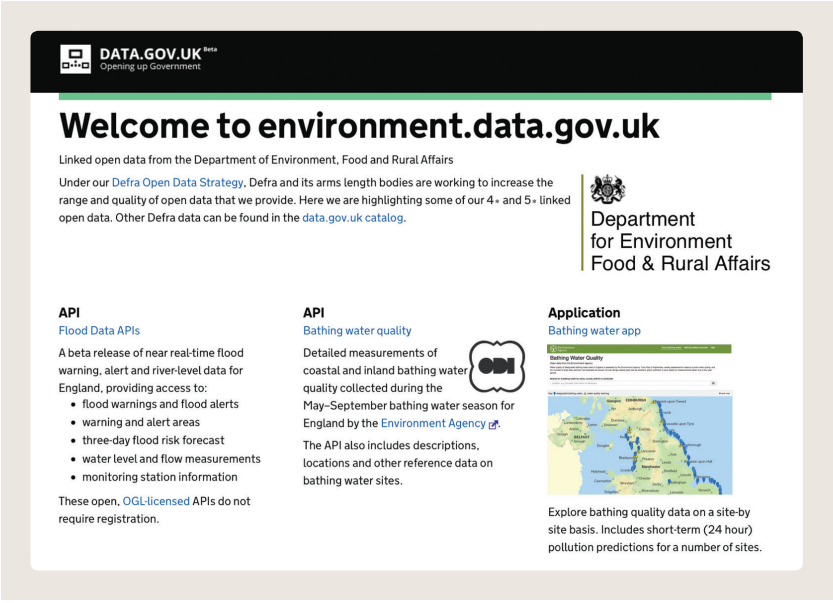


Figura A.12 – Página inicial do *site* de dados ambientais do Reino Unido

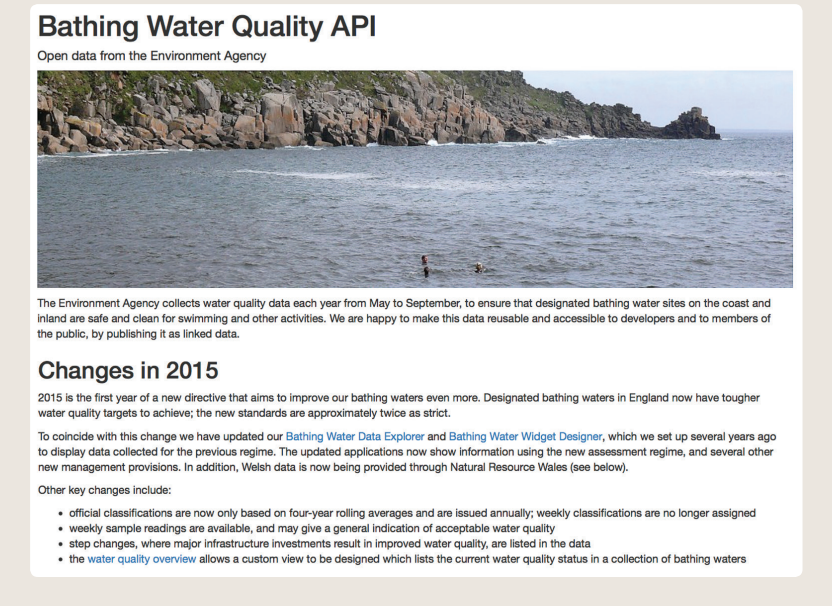


Figura A.13 – Informações sobre o conjunto de dados (qualidade da água)

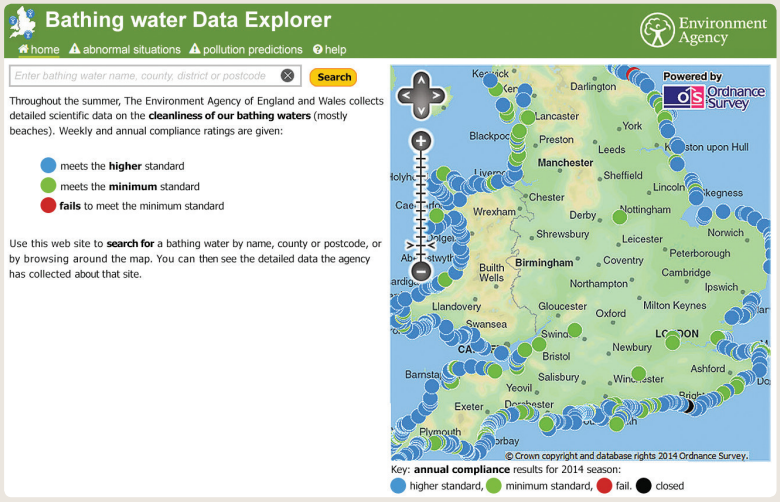


Figura A.14 – Aplicação com informações sobre a qualidade da água no Reino Unido

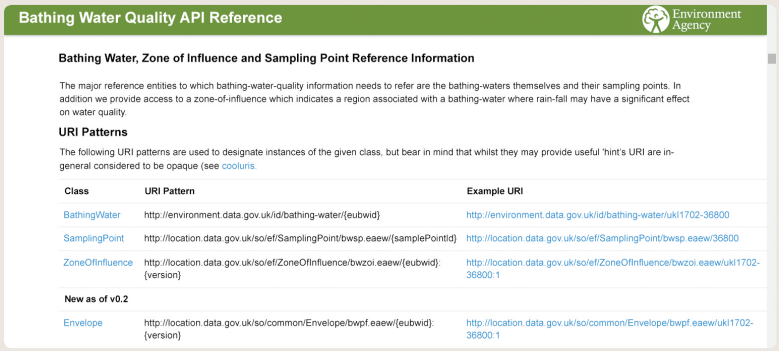


Figura A.15 – Informações sobre a API de acesso aos dados (qualidade da água)

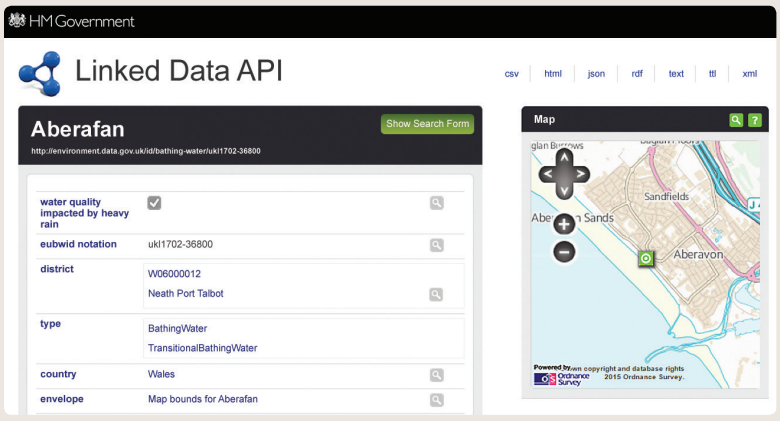


Figura A.16 – **Informações sobre a qualidade da água em "Neath Port Talbot" (LD API)**

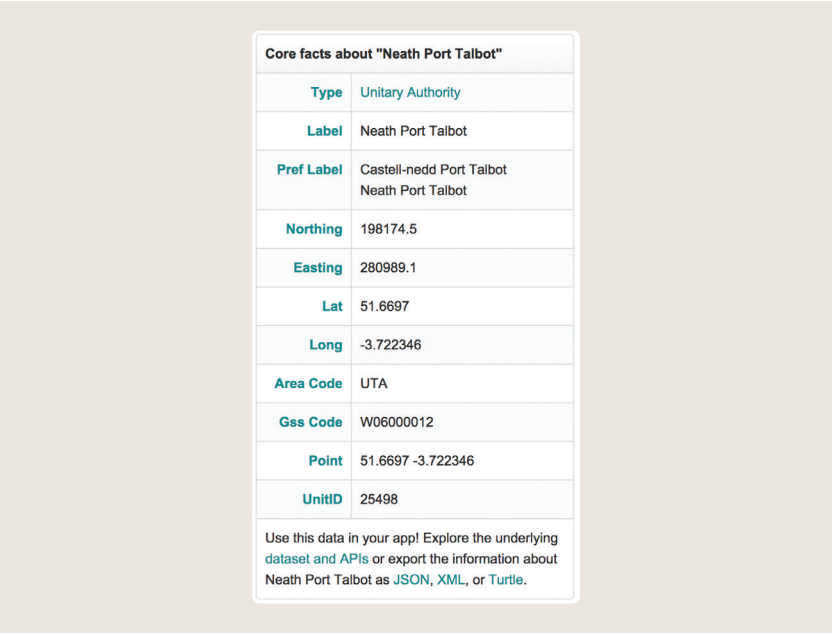


Figura A.17 – Informações sobre "Neath Port Talbot" no *site* "Ordnance Survey"

A.7 BANCOS DE TRIPLAS

Bancos de triplas são sistemas de gerenciamento para dados modelados usando RDF. Ao contrário de sistemas de gerenciamento de bancos de dados relacionais, que armazenam dados em relações (ou tabelas) e são consultados usando SQL, os bancos de triplas armazenam triplas RDF e são consultados usando SPARQL. Além disso, uma característica fundamental de muitos bancos de triplas é a possibilidade de fazer inferências. Existem dois tipos principais de bancos de triplas: os que armazenam as triplas diretamente no banco e os que fazem o armazenamento em bancos de dados relacionais e oferecem uma camada de gerenciamento de RDF. Determinados arcabouços como, por exemplo, o Sesame, têm uma interface (Sail API) que permite que uma instalação possa configurar diferentes bancos de triplas. A seguir é apresentada uma lista de alguns dos principais bancos de triplas atuais:

- Virtuoso
  Possui um banco de triplas nativo e é uma das plataformas de Dados Conectados mais utilizadas atualmente, como, por exemplo, a DBpedia. Possui duas versões, uma comercial e uma em código aberto com um número restrito de características e menor desempenho. Oferece um conjunto restrito de inferências.

- Sesame
  Arcabouço Java com um banco de triplas nativo, que permite a configuração de outros bancos de triplas que utilizem a Sail API. Não oferece, de forma nativa, um nível sofisticado de inferências.

- Jena TDB [108]
  Arcabouço Java com um banco de triplas nativo, com diversos raciocinadores internos, além de poder configurar raciocinadores externos, tais como o Pellet.

- GraphDB [109]
  Previamente denominado OWLIN, é um dos bancos de triplas mais utilizados atualmente, que permite inferências em ontologias OWL e é oferecido em três versões, uma delas gratuita. GraphDB dá suporte à Sail API do Sesame.

- 4store [110]
  É um software livre que tem como pontos fortes o desempenho, a escalabilidade e a estabilidade. Possui uma licença GNU GPL. Não oferece inferências. 4store dá suporte à Sail API do Sesame.

- Bigdata [111]
  Banco de dados de grafos, escrito em Java, de alta performance e escalabilidade. A plataforma oferece suporte ao modelo de dados RDF e SPARQL, incluindo consulta, atualização e consulta federada básica. Oferece um conjunto restrito de inferências. Possui dois tipos de licença, uma comercial e uma GNU GPLv2. Bigdata dá suporte à Sail API do Sesame.

A.8 BIBLIOTECAS

Nesta seção são listadas algumas bibliotecas que manipulam dados RDF e consultas SPARQL, para a utilização em ambientes de desenvolvimento de diferentes linguagens de programação:

- RDFLib [112]
  Biblioteca para Python com diversos parsers e diversos formatos de serializações, incluindo Turtle, RDF/XML, RDFa, Microdados e JSON-LD. Armazenamento em memória e de forma persistente

utilizando Oracle Berkeley DB. Suporta consultas e atualizações SPARQL.

- Redland [113]
  Biblioteca para C com APIs para manipular grafos RDF, triplas, URIs e literais. Armazenamento em memória e de forma persistente utilizando Oracle Berkeley DB, MySQL 3-5, PostgreSQL, Virtuoso, SQLite, entre outros. Suporta várias sintaxes para leitura e escrita de RDF como RDF/XML, N-triples, Turtle, etc., por meio da biblioteca Raptor RDF [114]. Permite consultas em SPARQL e RDQL usando a biblioteca de consulta Rasqal RDF [115].

- dotNetRDF [116]
  Biblioteca para .Net com uma API para RDF e SPARQL. Suporte para várias infraestruturas de armazenamento, bem como uma API que permite que sejam conectadas infraestruturas de terceiros, como, por exemplo, Virtuoso, 4store, Jena-Fuseki, etc.

- EasyRDF [117]
  Biblioteca para PHP com diversos parsers e diversos formatos de serializações, incluindo Turtle, RDF/JSON, N-Triples. Vem acompanhada de uma série de exemplos e oferece suporte para a visualização de gráficos usando GraphViz [118]. Permite consultas SPARQL.

- Perl RDF [119]
  Biblioteca para Perl com uma API para armazenamento, parsing e serializações de RDF. Produz um conjunto de classes para representar objetos básicos de RDF.

- rdfQuery [120]
  Biblioteca para JavaScript que permite a análise de RDFa embutido dentro de uma página, a consulta sobre os fatos que ela contém, e um raciocinador que permite inferências para produzir mais fatos. Depende da biblioteca jQuery.

REFERÊNCIAS

[1] Information Management: A Proposal - Tim Berners-Lee - March 1989
http://www.w3.org/History/1989/proposal.html

[2] Uniform Resource Locators (URL) - URI working Group - 21 March 1994
http://www.w3.org/Addressing/URL/url-spec.txt

[3] HTML5 - A vocabulary and associated APIs for HTML - W3C Recommendation 28 October 2014
http://www.w3.org/TR/html5/

[4] Hypertext Transfer Protocol (HTTP/1.1) - Internet Engineering Task Force (IETF) - June 2014
https://tools.ietf.org/html/rfc7230

[5] The New York Times - Developers APIs
http://developer.nytimes.com/docs

[6] ProgrammableWeb
http://www.programmableweb.com/

[7] Apresentação de Tim Berners-Lee no "Open, Linked Data for a Global Community"
https://www.youtube.com/watch?v=ga1aSJXCFe0

[8] Ajax: A New Approach to Web Applications - Jesse James Garrett - February 18, 2005
http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

[9] The Semantic Web - Tim Berners-Lee, James Hendler and Ora Lassila - Scientific American, May 2001
http://www.cs.umd.edu/~golbeck/LBSC690/SemanticWeb.html

[10] RDF 1.1 Primer - W3C Working Group Note 24 June 2014
http://www.w3.org/TR/rdf11-primer/

[11] The Thirty Minute Guide to RDF and Linked Data - Ian Davis
http://pt.slideshare.net/iandavis/30-minute-guide-to-rdf-and-linked-data

[12] Uniform Resource Identifier (URI): Generic Syntax
https://tools.ietf.org/html/rfc3986

[13] FOAF Vocabulary Specification - 14 January 2014
http://xmlns.com/foaf/spec/

[14] The DOI System
http://www.doi.org/

[15] The PersID initiative
http://www.persid.org/

[16] Cool URIs for the Semantic Web - W3C Interest Group Note 03 December 2008
http://www.w3.org/TR/cooluris/

[17] Cool URIs don't change
http://www.w3.org/Provider/Style/URI.html

[18] RDF 1.1 XML Syntax - W3C Recommendation 25 February 2014
http://www.w3.org/TR/rdf-syntax-grammar/

[19] RDF 1.1 Turtle - Terse RDF Triple Language - W3C Recommendation 25 February 2014
http://www.w3.org/TR/turtle/

[20] RDF 1.1 N-Triples - A line-based syntax for an RDF graph - W3C Recommendation 25 February 2014
http://www.w3.org/TR/n-triples/

[21] JSON-LD 1.0 - A JSON-based Serialization for Linked Data - W3C Recommendation 16 January 2014
http://www.w3.org/TR/json-ld/

[22] Dublin Core Metadata Element Set
http://dublincore.org/documents/dces/

[23] RDF Schema - W3C Recommendation 25 February 2014
http://www.w3.org/TR/rdf-schema/

[24] Bibliographic Ontology Specification - 4 November 2009
http://bibliontology.com/specification

[25] OWL 2 Web Ontology Language - W3C Recommendation 11 December 2012
http://www.w3.org/TR/owl2-syntax/

[26] A Description Logic Primer - Markus Krötzsch, František Simancík, Ian Horrocks - June 3, 2013
http://arxiv.org/pdf/1201.4089.pdf

[27] SPARQL 1.1 Query Language - W3C Recommendation 21 March 2013
http://www.w3.org/TR/sparql11-query/

[28] Open Knowledge Foundation SPARQL Endpoint Status
http://sparqles.okfn.org/

[29] W3C SPARQL endpoints List
http://www.w3.org/wiki/SparqlEndpoints

[30] Mondeca SPARQL Endpoint Status
http://labs.mondeca.com/sparqlEndpointsStatus/index.html

[31] OpenLink Virtuoso SPARQL Query Editor
http://demo.openlinksw.com/sparql

[32] Página de Tim Berners-Lee na DBpedia
http://dbpedia.org/page/Tim_Berners-Lee

[33] Página de Tim Berners-Lee na DBpedia em JSON
http://dbpedia.org/data/Tim_Berners-Lee.json

[34] SNORQL - SPARQL Explorer for http://dbpedia.org/sparql
http://dbpedia.org/snorql/

[35] SPARQL 1.1 Update - W3C Recommendation 21 March 2013
http://www.w3.org/TR/sparql11-update/

[36] SPARQL 1.1 Federated Query - W3C Recommendation 21 March 2013
http://www.w3.org/TR/sparql11-federated-query/

[37] Microformats
http://microformats.org/

[38] Microformats 2
http://microformats.org/wiki/microformats2

[39] RDFa Core 1.1 - W3C Recommendation 22 August 2013
http://www.w3.org/TR/rdfa-core/

[40] RDFa 1.1 Primer - Second Edition - W3C Working Group Note 22 August 2013
http://www.w3.org/TR/xhtml-rdfa-primer/

[41] HTML Microdata - W3C Working Group Note 29 October 2013
http://www.w3.org/TR/microdata/

[42] Schema.org - type Person
http://schema.org/Person

[43] The Linking Open Data cloud diagram
http://lod-cloud.net/

[44] Linked Data
http://www.w3.org/DesignIssues/LinkedData.html

[45] 5 Stars Open Data
http://5stardata.info/

[46] Exemplo de Dados Conectados “4 estrelas”
http://5stardata.info/en/examples/gtd-4/

[47] Meteo - Vocabulário para informações meteorológicas
http://inamidst.com/sw/ont/meteo

[48] Graphite RDF Browser
http://graphite.ecs.soton.ac.uk/browser/?uri=http://5stardata.info/en/examples/gtd-4/

[49] Linked Data API
https://code.google.com/p/linked-data-api/

[50] Dados Abertos do Orçamento do Governo Federal
http://orcamento.dados.gov.br/

[51] Elda - Epimorphics Linked Data API
http://www.epimorphics.com/web/tools/elda.html

[52] DBpedia
http://dbpedia.org/

[53] DBpedia Virtuoso SPARQL Query Editor
http://dbpedia.org/sparql

[54] The DBpedia query builder
http://querybuilder.dbpedia.org

[55] DBpedia - Construtor de Consultas Interativo Openlink - iSPARQL
http://dbpedia.org/isparql

[56] Aplicações da DBpedia
http://wiki.dbpedia.org/services-resources/dbpedia-applications

[57] DBpedia Spotlight
https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki

[58] DBpedia - página do recurso World Wide Web
http://pt.dbpedia.org/page/World_Wide_Web

[59] Portal de Datos de la Biblioteca Nacional de España
http://datos.bne.es

[60] Ontology Engineering Group (OEG)
http://www.oeg-upm.net/

[61] BNE - Virtuoso SPARQL Query Editor
http://datos.bne.es/sparql

[62] BNE ontology
http://datos.bne.es/def/ontology.html

[63] BBC Things
http://www.bbc.co.uk/things/

[64] BBC Ontologies
http://www.bbc.co.uk/ontologies

[65] Ontologia do Orçamento do Governo Federal Brasileiro
http://vocab.e.gov.br/2013/09/loa.owl

[66] Manual Técnico de Orçamento
http://www.orcamentofederal.gov.br/informacoes-orcamentarias/manual-tecnico/mto_2015_1a_edicao-150514.pdf

[67] Catálogo de Dados Abertos do Governo Federal
http://dados.gov.br/dataset/orcamento-federal

[68] Linked Data API do Orçamento Federal
http://orcamento.dados.gov.br/api-config

[69] Orçamento Federal - OpenLink Virtuoso SPARQL Query
http://orcamento.dados.gov.br/sparql/

[70] Bio2RDF
https://github.com/bio2rdf/bio2rdf-scripts/wiki

[71] DrugBank
http://www.drugbank.ca/

[72] BioPortal
http://bioportal.bioontology.org/

[73] Consulta a "Glutahione" no site DrugBank
http://www.drugbank.ca/drugs/DB00143

[74] Consulta a "Glutahione" no site SPARQL endpoint
http://cu.drugbank.bio2rdf.org/describe/?url=http://bio2rdf.org/drugbank:DB00143

[75] Linked Open Vocabularies (LOV)
http://lov.okfn.org/dataset/lov/

[76] JoinUp - Federated Repositories
https://joinup.ec.europa.eu/catalogue/repository

[77] Dublin Core Metadata Initiative (DCMI)
http://dublincore.org/

[78] DCMI Metadata Terms
http://dublincore.org/documents/dcmi-terms/

[79] FOAF Project
http://www.foaf-project.org/

[80] Arquivo FOAF de Tim Berners-Lee
http://dig.csail.mit.edu/2008/webdav/timbl/foaf.rdf

[81] SKOS Simple Knowledge Organization System - W3C Recommendation 18 August 2009
http://www.w3.org/TR/skos-reference/

[82] SKOS Primer - W3C Working Group Note 18 August 2009
http://www.w3.org/TR/skos-primer/

[83] Vocabulários do Schema.org
http://schema.org/docs/schemas.html

[84] PROV-O - The PROV Ontology - W3C Recommendation 30 April 2013
http://www.w3.org/TR/prov-o/

[85] PROV Model Primer - W3C Working Group Note 30 April 2013
http://www.w3.org/TR/prov-primer/

[86] Data Catalog Vocabulary (DCAT) - W3C Recommendation 16 January 2014
http://www.w3.org/TR/vocab-dcat/

[87] The Open Data Institute
http://theodi.org/

[88] Exemplo de DCAT embutido utilizando RDFa
http://theodi.org/guides/marking-up-your-dataset-with-dcat

[89] Show Me the Money Project
http://smtm.labs.theodi.org/

[90] Show Me the Money Project - Download de dados
http://smtm.labs.theodi.org/download/

[91] A Programmable Web: An Unfinished Work - Aaron Swartz
http://www.morganclaypool.com/doi/pdfplus/10.2200/S00481ED1V01Y201302WBE005

[92] The D2RQ Platform
http://d2rq.org/

[93] The D2RQ Mapping Language
http://d2rq.org/d2rq-language

[94] Openlink Virtuoso Universal Server
http://virtuoso.openlinksw.com/

[95] Mapping SQL Data to Linked Data Views
http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSSQL2RDF

[96] Virtuoso SPARQL Implementation Details
http://docs.openlinksw.com/virtuoso/rdfsparql.html

[97] Virtuoso R2RML Support
http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VirtR2RML

[98] R2RML: RDB to RDF Mapping Language - W3C Recommendation 27 September 2012
http://www.w3.org/TR/r2rml/

[99] Sesame Framework
http://rdf4j.org/

[100] Getting Started with Sesame, Maven, and Eclipse
http://rdf4j.org/sesame/tutorials/getting-started.docbook?view

[101] Apache Jena Framework
http://jena.apache.org/

[102] Pellet: OWL 2 Reasoner for Java
http://clarkparsia.com/pellet/

[103] PulishMyData
http://www.swirrl.com/publishmydata

[104] Swirrl
http://www.swirrl.com/

[105] Hampshire Hub Catalogue
http://data.hampshirehub.net/data

[106] Plataforma de Publicação de Dados Conectados Epimorphics
http://www.epimorphics.com/web/products/data-publishing

[107] Ordnance Survey - Great Britain's national mapping agency
http://data.ordnancesurvey.co.uk/doc/7000000000025498

[108] Jena TDB

http://jena.apache.org/documentation/tdb/

[109] GraphDB
http://www.ontotext.com/products/ontotext-graphdb/

[110] 4store
http://4store.org/

[111] Bigdata
http://www.bigdata.com/

[112] RDFLib
https://github.com/RDFLib

[113] Redland RDF Libraries
http://librdf.org/

[114] Raptor RDF Syntax Library
http://librdf.org/raptor/

[115] Rasqal RDF Query Library
http://librdf.org/rasqal/

[116] dotNetRDF - Semantic Web, RDF and SPARQL Library for C#/.Net
http://dotnetrdf.org/

[117] EasyRDF
http://www.easyrdf.org/

[118] Graphviz - Graph Visualization Software
http://www.graphviz.org/

[119] Perl and RDF
http://www.perlrdf.org/

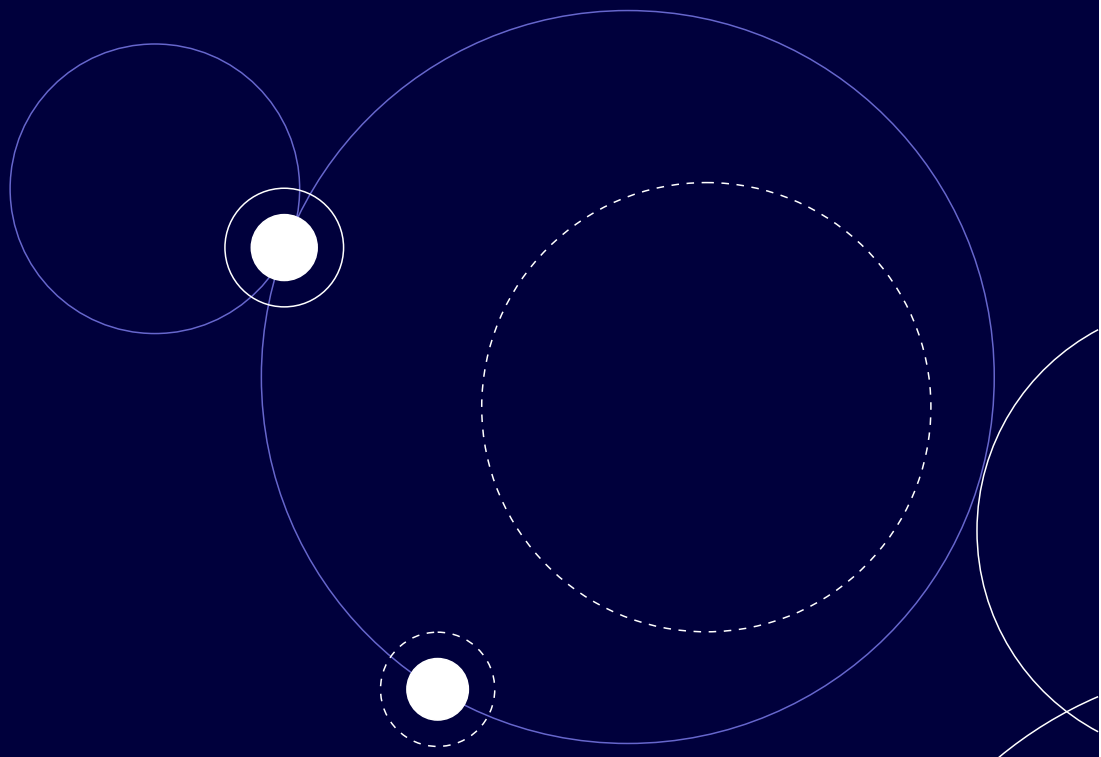
[120] rdfquery - RDF processing in your browser
https://code.google.com/p/rdfquery/

SPUK

GREAT BRITAIN

Improving business environment through transparency in São Paulo State

Melhoria do ambiente de negócios por meio da transparência no Estado de São Paulo



Ano 2015

cgi.br nic.br ceweb.br

SEADE
Fundação Sistema Estadual de Análise de Dados

Fundap

GREAT BRITAIN

Embaixada Britânica Brasília

GOVERNO DO ESTADO SÃO PAULO
Secretaria de Governo