UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

ANTONIO DE SANTES NETO

APLICAÇÃO DE *LINKED DATA* NA WEB

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2015

ANTONIO DE SANTES NETO

APLICAÇÃO DE *LINKED DATA* NA WEB

Trabalho de Diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Everton Coimbra de Araújo.

MEDIANEIRA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

Nome do trabalho

Por

Antonio de Santes Neto

Este Trabalho de Diplomação (TD) foi apresentado às 14:30 h do dia 10 de junho de 2015 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Manutenção Industrial, da Universidade Tecnológica Federal do Paraná, Campus Medianeira. O acadêmico foi argüido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Everton Coimbra de Araújo
UTFPR – *Campus* Medianeira
(Orientador)

Prof. Evando Carlos Pessini
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Juliano Rodrigo Lamb
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Juliano Rodrigo Lamb
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

A folha de aprovação assinada encontra-se na Coordenação do Curso.

AGRADECIMENTOS

A Deus, criador de todas as coisas, por ter me permitido chegar até aqui.

A minha mãe Sarita, por sempre puxar minha orelha. Ao meu pai Carlos, por ter me ajudado sempre que precisei de apoio.

A minha namorada Andréia, por sempre me incentivar.

Aos meus ex-colegas da Universidade Tecnológica Federal do Paraná e de trabalho, pela troca de conhecimentos e pela convivência.

E aos demais que de alguma forma contribuíram para a realização deste trabalho.

RESUMO

NETO, Antonio. APLICAÇÃO DE *LINKED DATA* NA WEB. 2015. Trabalho de Conclusão de Curso, Universidade Tecnológica Federal do Paraná. Medianeira 2015.

Este trabalho apresenta um estudo sobre como os padrões *Linked Data* resolvem o problema da falta de estruturação e semântica da Web. Primeiramente aborda sobre a Web atual e sua evolução para se tornar cada vez mais semântica. Apresenta os principais conceitos relacionados aos padrões *Linked Data*, bem como a arquitetura que envolve a construção de aplicações que utilizam estes padrões. Aborda sobre como é feita a confecção de arquivos RDF para representação de dados, como também sobre a utilização do modelo SPARQL para realização de consultas RDF. Em seguida apresenta a utilização do *framework* Jena. Por fim, aplica os conceitos e tecnologias abordadas no trabalho em uma aplicação web.

Palavras-chave: Web Semântica. RDF. SPARQL. Jena.

ABSTRACT

NETO, Antonio. APLICATION OF LINKED DATA ON WEB. 2015. Trabalho de Conclusão de Curso, Universidade Tecnológica Federal do Paraná. Medianeira 2015.

This paper presents a study on how the Linked Data standards address the problem of lack of structure and the Semantic Web. First discusses about the current Web and their evolution to become increasingly semantics. It presents the main concepts related to Linked Data standards, as well as the architecture that involves building applications that use these standards. It discusses about how is the production of RDF files for data representation, but also about using SPARQL model to perform RDF queries. Then discloses the use Jena framework. Finally, apply the concepts and technologies discussed at work in a web application.

Keywords: Semantic Web. RDF. SPARQL. Jena.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAS DE SIGLAS

API	*Application Programming Interface*
FOAF	*Friend of a Friend*
HTML	*Hypertext Markup Language*
JSON	*JavaScript Object Notation*
RDF	*Resource Description Framework*
RDF-Schema	*Resource Description Framework Schema*
SPARQL	*Protocol and RDF Query Language*
URI	*Uniform Resource Identifier*
XML	*Extensible Markup Language*
W3C	*World Wide Web Consortium*
Web	*World Wide Web*

SUMÁRIO

1 INTRODUÇÃO

Atualmente a Web basicamente é composta de documentos hipermídia. Esses documentos possuem texto, imagens, vídeos, gráficos, fornecendo vários tipos de informações que pessoas podem interpretar. Portanto, pode-se dizer que a Web cumpre muito bem seu objetivo de prover um meio nos quais pessoas possam criar e consumir informações. Para exemplificar, pode-se imaginar uma pessoa que acesse um site de uma loja de roupas. Primeiramente esta pessoa olha o site e percebe que se trata de uma loja roupas, e não de outro tipo de loja. Seguindo, a pessoa visualiza algumas imagens de roupas e sabe que se trata de roupas e não de qualquer outra coisa, como calçados por exemplo. Essa percepção que a pessoa tem é fruto da capacidade que todo ser humano tem de compreender informações por meio da visão, ou seja, a pessoa vê a informação e a compreende. Nesse sentido pode-se dizer que a Web possui semântica para seres humanos, ou seja, um ser humano é capaz compreender o conteúdo da Web.

Entretanto, agentes de *software* não conseguem extrair da Web o mesmo nível de semântica que os humanos conseguem. Utilizando o exemplo anterior, onde um agente de *software* esteja vasculhando a Web em busca de informações específicas, e esse *software* acesse o mesmo site da loja de roupas. Primeiramente, tudo o que o *software* está vendo são textos e meta-dados muito abstratos em sua semântica perceptível, por exemplo: o agente de *software* sabe que um determinado elemento do site é uma imagem, mas ele não tem como saber (a menos que grandes esforços computacionais sejam aplicados) que se trata de uma imagem de uma camiseta, por exemplo. O agente de *software* sabe que o título do site é “Loja de Roupa”, porém isso não passa de dados sem semântica para ele. Deste modo fica clara a limitação do agente de *software* em relação à percepção da plenitude da semântica que o site contém. A proposta da Web Semântica é prover mecanismos que permitam adicionar mais semântica computacional no conteúdo da Web, tornando-a melhor compreendida por agentes de *software*.

A ideia da Web Semântica surgiu em 2001, quando Tim Berners-Lee, James Hendler e Ora Lassila publicaram um artigo na revista Scientific American, intitulado: “Web Semântica: um novo formato de conteúdo para a Web que tem significado para computadores vai iniciar uma revolução de novas possibilidades” (WIKIPÉDIA, 2015). Nesse artigo os autores

explanam uma nova proposta para melhorar a Web no que diz respeito à organização, conectividade e semântica do seu conteúdo. A partir disso, foram desenvolvidos vários mecanismos para tornar a Web mais semântica, os quais foram denominados *Linked Data*. Este trabalho tem como objetivo estudar a aplicação destes mecanismos, e apresentar os aspectos práticos de como publicar e consumir dados na "Web de Dados" utilizando os padrões *Linked Data*.

1.1 OBJETIVO GERAL

Demonstrar os benefícios de adotar os padrões *Linked Data* na Web para publicação de dados abertos.

1.2 OBJETIVOS ESPECÍFICOS

- Explanar a problemática da Web atual em relação a semântica dos dados.
- Explanar sobre as tecnologias relativas à aplicação do *Linked Data* na Web atual, e como aplica-las.
- Fazer uso do *Linked Data* em uma aplicação de estudo, demostrando como publicar, interligar e consumir dados neste formato.

1.3 JUSTIFICATIVA

A Web vem crescendo muito a cada dia, porém todo esse conteúdo não está organizado e interligado adequadamente. Em sua maior parte, os dados na Web ainda são organizados para serem lidos ou compreendidos por humanos e não por agentes de *software*. Para que um agente de *software* possa entender e interpretar um dado, é necessário processar a semântica envolvida naquele dado, num determinado contexto. Neste escopo, semântica diz

respeito à atribuição de significado a elementos, dados ou expressões que precisam ser interpretados numa dada situação. Por esses motivos, agentes de *software* não conseguem fazer uso de todo o potencial de informação e conhecimento que a Web possui. O HTML define a estrutura de um documento, mas não explica seu significado.

A aplicação dos conceitos de *Linked Data* será de grande utilidade para os agentes de *software* serem capazes de usufruir de todo o potencial de informação e conhecimento armazenados na Web, proporcionando uma melhor experiência para os internautas.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 LIMITAÇÕES DA WEB CLÁSSICA

Para entender melhor as mudanças que a Web semântica propõe, é importante entender como a Web clássica funciona, e quais as suas limitações.

A Web clássica é alicerçada em três principais padrões:

- HTTP (*Hypertext Transfer Protocol*) como um mecanismo universal de acesso;
- URIs (*Uniform Resource Identifiers*) como um mecanismo de identificação único;
- HTML (*Hypertext Markup Language*) como um formato padrão de representação de conteúdo.

Por meio desses padrões a Web tem possibilitado que, em larga escala, pessoas e organizações no mundo todo publiquem documentos em um único espaço global de informação, de modo que o conteúdo da Web tem se tornado vasto. O benefício de se adotar padrões e de convergir tudo em um único espaço de informação é que se torna simples criar ferramentas para interagir com a Web. Com o advento dos navegadores e dos motores de busca, se tornou fácil encontrar o que se precisa na Web. Outra característica importante da Web clássica são os *hyperlinks*. Por meio deles um documento em um servidor pode ser interligado com outro documento em outro servidor. O uso de *hyperlinks* é a maneira existente na Web clássica de relacionar diferentes documentos entre si. No entanto, a Web clássica possui muitas limitações quando se pretende atribuir semântica em seu conteúdo, e isso acontece devido à falta de estruturação dos dados. Heath e Bizer (2011) explanam esse problema.

> Um fator chave na capacidade de reutilizar dados se dá na medida em que esses dados são estruturados. Quando mais organizada e bem definida é a estrutura de dados, mais facilmente pessoas podem criar ferramentas para processá-los de forma confiável para reutilização. Enquanto a maioria dos Web sites tem algum nível de estrutura, a linguagem na qual elas foram criadas, HTML, é orientada em estruturar documentos textuais e não dados. Como os dados são espalhados no meio do texto, fica difícil para agentes de *softwares* extrair conteúdo estruturado de páginas HTML (HEATH; BIZER, 2011, p. 2, tradução própria).

2.1.1 A Limitação das *Meta-Tags*

Umas das maneiras de atribuir semântica nos documentos HTML é a utilização de meta-dados dentro destes documentos, por meio das meta-tags. *Meta-tags* só podem ser informadas dentro do elemento `<head>` e são apenas entendidas por agentes de *software*. O Quadro 1 apresenta algumas *meta-tags* disponíveis para serem utilizadas na web.

`<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">`	Define palavras-chave para motores de busca.
`<meta name="description" content="Free Web tutorials on HTML and CSS">`	Define a descrição do Web site.
`<meta name="author" content="Hege Refsnes">`	Define o autor da página.

Quadro 1 - Descrição de algumas *meta-tags*
Fonte: Autoria própria

Qualquer desenvolvedor de Web sites pode definir suas próprias *meta-tags*. Um agente de *software* pode vasculhar os documentos da Web em busca de *meta-tags* específicas na qual previamente já se conheça e, desse modo, extrair informação semântica desses documentos. No entanto, *meta-tags* são utilizadas apenas para prover um pequeno conjunto de informações, deixando o nível de semântica de todo o documento ainda muito pobre.

2.1.2 A Limitação em se Utilizar Formatos de Dados como XML e JSON

O método mais utilizado pelos Web sites para prover suas próprias informações para o resto da Web, e isso de forma mais estruturada, é fazendo uso de formatos de dados como XML e JSON. Cada Web site encapsula suas próprias informações nesses formatos e prove uma forma de acessar esses dados. Por exemplo, o Quadro 2 apresenta uma informação em XML que poderia ser retornada por um site qualquer.

```
<produto>
    <codigo>123456</codigo>
    <descricao>Arroz</descricao>
</produto>
```

Quadro 2 - Exemplo de dado em XML
Fonte: Autoria própria

O problema é que cada Web site possui seus próprios padrões de nomenclatura e significado, o número **123456** referente ao código do produto não significa nada fora do escopo do Web site que o gerou, de modo que o resto da Web precisa aprender como compreender os dados disponibilizados por cada Web site, criando assim ilhas isoladas de conteúdo semântico.

2.1.3 A Limitação dos *Microformats*

O mais próximo de uma solução para esse problema de semântica talvez seja a criação dos *microformats*[1]. *Microformats* são utilizados para informar meta-dados referentes a entidades do mundo real, como pessoas e eventos, e isso dentro dos elementos das páginas HTML. O Quadro 3 apresenta um exemplo de uso de *microformats*, onde é definido que uma *div* está diz respeito a um evento.

```
<div class="h-event">
    <h1 class="p-name">Microformats Meetup</h1>
    <p>Join us at <b class="p-adr h-adr">
      <span class="p-street-address">Some Bar</span>,
      <span class="p-locality">Someplace</span></b>
   </p>
</div>
```

Quadro 3 - Exemplo de uso de *microformats*
Fonte: Microformats (2015)

O *microformats* provê um conjunto de entidades para descrever coisas utilizando a tag *class*. Por exemplo: se um agente de *software* encontrar a marcação `class="h-event"` em

[1] *Microformats* são códigos HTML para referenciar pessoas, organizações, eventos, lugares etc. Sites utilizam *microformats* para publicar um formato padrão que é consumido por mecanismos de busca, navegadores e outros sites (Mifroformats, 2015).

algum elemento saberá que se trata de um evento. Contudo existe algumas limitações nos *microformats*, Heath e Bizer (2011) abordam isso.

> Como os *microformats* tem uma forma muito precisa de incluir dados nos documentos, aplicações podem extrair dados desses documentos de forma não ambígua. O ponto fraco dos *microformats* são que eles são restringidos para representar dados de um pequeno conjunto de diferentes entidades; ele apenas provê um pequeno conjunto de atributos que podem ser usados para descrever essas entidades, e não é possível expressar relacionamentos entre entidades, por exemplo, se uma pessoa é um professor de um evento, ou se é apenas o organizador do evento. (HEATH; BIZER, 2011, p. 2, tradução própria).

2.2 LINKED DATA

Segundo Heath e Bizer (2011) "*Linked Data* se refere a um conjunto de boas práticas para publicação e interligação de dados na Web" (HEATH; BIZER, 2011, p. 7, tradução própria). Enquanto a Web Semântica é a ideia de tornar a Web mais semântica, *Linked Data* é um conjunto de métodos para aplicar essa ideia.

Os criadores do *Linked Data* pensaram nos problemas da Web clássica e definiram mecanismos e boas práticas que possibilitam a Web Semântica. Heath e Bizer (2011, p. 7) definem essas boas práticas, os chamados *Linked Data Principles*. Estes princípios são os seguintes:

- Usar HTTP URIs para dar nomes para as coisas, e para que pessoas possam procurar essas coisas na Web.
- Dado um URI, providenciar informação que agentes de *software* possam compreender, utilizando um modelo de dados padrão chamado RDF.
- Incluir links para outras URIs, para que se possa descobrir mais coisas.

2.2.1 URIs Para dar Nome e Interligar as Coisas

Um dos princípios do *Linked Data* é que todo objeto ou conceito abstrato do mundo real tenha um identificador único na Web. Como a Web está alicerçada no protocolo HTTP, esse identificador deve ser um HTTP URI, para que assim seja possível disparar consultas

contra esses identificadores, utilizando o protocolo HTTP. Heath e Bizer (2011) ressaltam a necessidade do uso de HTTP URIs no *Linked Data*.

> O protocolo HTTP é o mecanismo universal de acesso à Web. Na Web clássica, HTTP URIs são usadas para combinar em um sistema global, identificação única e um mecanismo simples de recuperação de dados. Assim, um dos princípios do *Linked Data* advoga o uso de HTTP URIs para identificar objetos e conceitos abstratos, possibilitando que seja possível recuperar a descrição do dado referente a URI utilizando o protocolo HTTP (HEATH; BIZER, 2011, p. 8, tradução própria).

Um URI pode representar qualquer coisa, por exemplo: um URI pode representar uma pessoa real, outro pode representar um lugar, outro pode representar um time de futebol, outro pode representar um conceito abstrato, como a "amizade". O Quadro 4 apresenta exemplos de URIs.

`http://exemplo.com/joao`	URI que representa a pessoa João.
`http://exemplo.com/corinthians`	URI que representa o time de futebol Corinthians.
`http://exemplo.com/brasil`	URI que representa o país Brasil.

Quadro 4 - Exemplos de URI's
Fonte: Autoria própria

Outro princípio do *Linked Data* é a interconectividade entre os dados. Por exemplo: deve ser possível que o João seja interligado ao time do Corinthians por meio de um elemento de ligação que representa o fato de que João torce para o Corinthians. Outra ligação poderia ser com João e o país Brasil, representando que João mora no Brasil. Como tudo no *Linked Data* são URIs, no *Linked Data* um elemento de ligação também é um URI. O Quadro 5 exemplifica possíveis ligações entre João, Corinthians e Brasil.

`http://exemplo.com/joao`	`http://exemplo.com/torce_para`	`http://exemplo.com/corinthians`
`http://exemplo.com/joao`	`http://exemplo.com/mora_em`	`http://exemplo.com/brasil`

Quadro 5 - Exemplo de ligações entre URI's
Fonte: Autoria própria

Como pode ser notado no Quadro 4, diferentemente dos *hyperlinks* da Web clássica, as ligações do *Linked Data* são mais significativas, especificando também o tipo de ligação. Heath e Bizer (2011) comparam o uso de *hyperlinks* da Web clássica com o *Linked Data*, salientando que "como os *hyperlinks* na Web clássica conectam documentos em um único espaço global de informação, o *Linked Data* permite interligar dados que estão em diferentes fontes de dados, conectando assim estas fontes de dados em um único espaço global de informação" (HEATH; BIZER, 2011, p. 4, tradução própria).

2.2.2 Modelo Padrão de Representação de Dados - RDF

No *Linked Data*, essa relação entre dois recursos interligados por um outro deu origem ao modelo de representação de dados RDF (*Resource Description Framework*). O modelo RDF especifica uma forma de descrever dados na Web fazendo uso de URIs. Com esse modelo é possível representar qualquer objeto, informação, conceito abstrato ou evento. A base do modelo RDF é a interligação entre recursos identificados por URIs. Essa interligação é denominada tripla. A Figura 1 apresenta o conceito de tripla no modelo RDF.

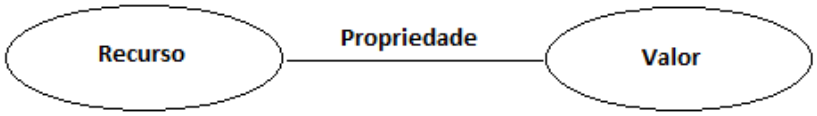

Figura 1 - Representação do conceito de tripla no *Linked Data*
Fonte: Autoria própria

Uma tripla consiste em três componentes:

- O **recurso**, que representa aquilo se pretende descrever.
- A **Propriedade**, que representa algum atributo específico do recurso.
- O **Valor**, que é o valor em si relativo a propriedade do recurso. Pode ser um valor literal, ou até mesmo outro recurso identificado por seu URI.

O recurso e o valor são interligados pela propriedade, formando uma tripla. Triplas são a base da representação semântica no *Linked Data*. Tudo é representado e interligado por meio de triplas.

2.2.3 Vocabulários RDF

No exemplo do Quadro 4 foi utilizado o URI `http://exemplo.com/joao` para representar uma pessoa, como também os URIs `http://exemplo.com/corinthians` e `http://exemplo.com/brasil` para representar o país Brasil e o time do Corinthians. No *Linked Data* um agente de *software* deve ser capaz de entender que uma URI se trata de uma pessoa, outra de um lugar, outra de um time de futebol. Para isso, esse agente de *software*

deve previamente conhecer vocabulários pré-definidos que expressam domínios de conhecimento do mundo real.

Existem vários vocabulários criados para serem utilizados na Web semântica. O Quadro 6 apresenta alguns vocabulários que já estão sendo utilizados.

RDF Schema	Utilizado para descrever recursos. É a base de todos os outros vocabulários.
FOAF (*Friend of a Friend*)	Utilizado para descrever pessoas, suas atividades e seus relacionamentos com outras pessoas.
SKOS (*Simple Knowledge Organization System*)	Utilizado para descrever conceitos, coisas.
BIBO (*Bibliographic Ontology Specification*)	Utilizado para descrever citações e referências bibliográficas (ou seja, citações, livros e artigos).

Quadro 6 - Exemplos de vocabulários utilizados na Web semântica
Fonte: Autoria própria

Todos esses vocabulários apresentados no Quadro 6, dentre outros, já estão bem difundidos, de modo que a grande maioria do conteúdo semântico da Web já está fazendo uso deles, e todos eles são construídos utilizando o modelo RDF.

Cada vocabulário contém uma variedade de URIs específicas. Por exemplo: o vocabulário RDF Schema disponibiliza o URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`, esse URI é utilizado para indicar que um recurso é de determinado tipo. Outro exemplo é o vocabulário FOAF, que disponibiliza o URI `http://xmlns.com/foaf/0.1/Person`, que é utilizado para indicar que um recurso é uma pessoa. Fazendo uso dessas URIs é possível definir uma ligação que diz que João é uma pessoa. O Quadro 7 apresenta essa ligação.

`http://exemplo.com/joao`	`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`	`http://xmlns.com/foaf/0.1/Person`

Quadro 7 - Tripla que representa João ser uma pessoa
Fonte: Autoria própria

Com o uso de vocabulários globais como o RDF Schema, um agente de *software* que encontre a descrição sobre João, saberá que se trata de uma pessoa.

2.2.4 Formatos RDF

É importante destacar que o RDF não é um formato de dados, como XML por exemplo, mas sim um modelo de dados. Vários formatos foram criados para representar o modelo RDF, como por exemplo: RDF/XML (será abordado no capítulo 4), JSON-LD[2], N-TRIPLES[3], dentre outros. Os quadros 8, 9 e 10 exemplificam um exemplo de tripla que informa que João é uma pessoa, nos formatos RDF-XML, N-TRIPLES e JSON-LD respectivamente.

RDF-XML – Utiliza XML para representar triplas

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/">

   <rdf:Description rdf:about="http://exemplo.com/joao">
  <rdf:type rdf:resourse="foaf:Person"/>
   </rdf:Description>

</rdf:RDF>
```

Quadro 8 - Exemplo de dados RDF utilizando o formato RDF-XML
Fonte: Autoria própria

N-TRIPLES – Representa cada tripla em uma única linha

```
<http://exemplo.com/joao> <http://www.w3.org/1999/02/22-rdf-syntax-
ns#type> <http://xmlns.com/foaf/0.1/Person> .
```

Quadro 9 - Exemplo de dados RDF utilizando o formato N-TRIPLES
Fonte: Autoria própria

JSON-LD – Utiliza JSON para representar as triplas

```
{
"@id": "http://exemplo.com/joao",
"@type": "http://xmlns.com/foaf/0.1/Person"
}
```

Quadro 10 - Exemplo de dados RDF utilizando o formato JSON
Fonte: Autoria própria

O formato padrão definido pela W3C é o RDF/XML. Esse formato utiliza XML para representar as triplas RDF, e será abordado em detalhes no capítulo 4.

[2] JSON-LD é um formato RDF que utiliza JSON.
[3] N-TRIPLES é um formato RDF que apresenta cada tripla em uma única linha.

2.2.5 URIs Dereferenciáveis

No *Linked Data*, qualquer URI deve ser dereferenciável[4], o que significa que clientes podem disparar consultas contra o URI utilizando o protocolo HTTP, recebendo como retorno uma descrição em formato RDF do recurso que é identificado pelo URI. Heath e Bizer (2011) salientam isso.

> Um HTTP URI deve ser dereferenciável, significando que clientes HTTP possam procurar pela URI utilizando o protocolo HTTP e recebendo como retorno uma descrição do recurso que é identificado pelo URI. Isso se aplica também para URIs que são usadas para identificar clássicos documentos HTML, como também URIs que são usadas no contexto do *Linked Data* para identificar objetos do mundo real e conceitos abstratos. Descrições de recursos são encapsuladas na forma de documentos Web. Descrições que são feitas para seres humanos interpretarem são representadas por documentos HTML. Descrições que são feitas para agentes de *software* consumirem são representadas por documentos RDF (HEATH; BIZER, 2011, p. 10, tradução própria).

Para não existir ambiguidade no *Linked Data*, um URI que representa um objeto do mundo real deve ser distinguido do URI que descreve esse objeto. O Quadro 11 ilustra isso distinguindo o URI que representa a pessoa do mundo real, e o URI que descreve essa pessoa em RDF.

`http://exemplo.com/joao`	URI que identifica o João no mundo real
`http://exemplo.com/joao.rdf`	URI que identifica o documento RDF/XML que descreve o João

Quadro 11 - Exemplo de não ambiguidade entre URI's
Fonte: Autoria própria

Heath e Bizer (2011) descreve a existência de duas estratégias para tratar esse tipo de possível ambiguidade no *Linked Data*.

> Existem duas diferentes estratégias para URIs que identificam objetos do mundo real derefenciáveis. Ambas as estratégias garantes que os objetos do mundo real e os documentos que descrevem eles não sejam confundidos, e humanos e máquinas recebam as representações apropriadas para cada um. Essas estratégias são chamadas de *303 URIs* e *Hash URIs* (HEATH; BIZER, 2011, p. 10, tradução própria).

[4] Vem do termo em inglês *dereferencing*, que significa o ato de receber uma descrição de um recurso identificado por um URI.

2.2.6 Estratégia 303 URIs

Na estratégia *303 URIs*, quando o servidor Web identifica que lhe foi solicitado um URI referente a um objeto do mundo real, ele retorna o código de redirecionamento HTTP `303 See Other` apontando para o URI referente ao documento que descreve esse objeto. Esse processo envolve os seguintes passos:

- O cliente envia uma requisição HTTP GET[5] para o servidor contra um URI que representa um objeto do mundo real. Se o cliente deseja uma descrição RDF sobre o recurso, ele envia no cabeçalho da requisição um `accept: application/rdf+xml`, informando que se deseja receber um documento RDF/XML.
- O servidor identifica que o URI se trata de um objeto do mundo real. Como o servidor não pode retornar isso, ele responde usando o código de redirecionamento HTTP `303 see other`, enviando junto com a resposta o URI do documento que descreve o objeto do mundo real em RDF/XML.
- O cliente por sua vez envia outra requisição para o URI retornado pelo servidor.
- O servidor responde com o código `200 OK` e envia para o cliente o documento RDF/XML solicitado.

Esse processo pode ser ilustrado utilizando as URIs `http://exemplo.com/joao` e `http://exemplo.com/joao.rdf` do exemplo anterior. Para obter o documento RDF que descreve o João, um agente de *software* pode se conectar com o `servidor http://exemplo.com/` e solicitar uma requisição via HTTP GET, como ilustrado no Quadro 12.

```
GET /joao HTTP/ 1 .1
Host: exemplo.com
Accept: application/rdf+xml
```

Quadro 12 - Consulta HTTP GET
Fonte: Autoria própria

[5] HTTP GET é um dos tipos de requisição do protocolo HTTP, e significa que se deseja recuperar um dado do servidor.

Como pode ser notado no Quadro 12, o cliente envia um `Accept: application/rdf+xml` indicando que se deseja receber um documento RDF. O Quadro 13 ilustra a resposta do servidor.

```
HTTP/1.1 303 See Other
Location: http://exemplo.com/joao.rdf
Vary: Accept
```

Quadro 13 - Servidor respondendo com o código: *303 See Other*
Fonte: Autoria própria

O servidor responde com o código de redirecionamento `303 See Other`, informando que o documento solicitado pelo cliente existe e se encontra em outro local. Então o cliente envia outra requisição para o local informado pelo servidor, como ilustrado no Quadro 14.

```
GET /joao.rdf HTTP/ 1 .1
Host: exemplo.com
Accept: application/rdf+xml
```

Quadro 14 - Consulta HTTP GET
Fonte: Autoria própria

O servidor por sua vez, envia o documento RDF/XML que descreve o João, como ilustrado pelo Quadro 15.

```
HTTP/1.1 200 OK
Content-Type: application/rdf+xml

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">

      <rdf:Description rdf:about="http://exemplo.com/joao">
       <rdf:type rdf:resourse="foaf:Person"/>
        <rdf:name>João</rdf:name>
       </rdf:Description>
</rdf:RDF>
```

Quadro 15 - Servidor retornando um documento RDF/XML
Fonte: Autoria própria

Utilizando a estratégia *303 URIs*, um servidor Web pode dereferenciar o URI que identifica um objeto do mundo real sem ambiguidade. No entanto, existem algumas críticas quanto ao uso dessa estratégia, as quais são abordadas por Heath e Bizer (2011).

> Uma crítica muito difundida sobre a estratégia *303 URIs* é que esse processo requer duas requisições HTTP para recuperar uma única descrição sobre um objeto do mundo real. Uma opção para evitar o uso de duas requisições é a utilização da estratégia Hash URIs (HEATH; BIZER, 2011, p. 13, tradução própria).

2.2.7 Estratégia Hash URIs

A estratégia Hash URIs faz uso da característica em que o URI pode conter uma parte especial separada da base pelo caractere *hash (#)*. Essa parte especial é chamada de fragmento identificador.

Quando um cliente envia uma requisição contra um URI que contém um fragmento separado pelo caractere *hash*, o protocolo HTTP requer que esse fragmento seja removido da requisição, ou seja, o fragmento nunca é enviado junto com a requisição. O Quadro 16 apresenta um novo URI para identificar o João, porém utilizando o caractere *hash*.

URI Base	**Fragmento Identificador**
`http://exemplo.com/joao`	`self`
URI Completa	
`http://exemplo.com/joao#self`	

Quadro 16 - Exemplo de URI com caractere *hash*
Fonte: Autoria própria

Para evitar ambiguidade, e também não precisar fazer uso de duas requisições para recuperar a descrição do objeto identificado pelo URI, poderia ser definido as seguintes URIs para identificar o objeto do mundo real e o documento que descreve o objeto, como ilustrado no Quadro 17.

`http://exemplo.com/joao#self`	URI que identifica o João no mundo real
`http://exemplo.com/joao`	URI que identifica o documento RDF/XML que contém a descrição sobre João

Quadro 17 - Exemplo de não ambiguidade entre URI's
Fonte: Autoria própria

Quando o cliente de posse de um URI como *hash* envia uma requisição HTTP GET, de acordo com as especificações do protocolo HTTP, o fragmento identificador é removido do cabeçalho da requisição, ficando como requisição apenas a base do URI. O quadro 18 ilustra uma consulta HTTP GET lançada contra o URI `http://exemplo.com/joao#self`.

```
GET /joao HTTP/ 1.1
Host: exemplo.com
Accept: application/rdf+xml
```

Quadro 18 - Requisição com URI contendo fragmento identificador
Fonte: Autoria própria

Como pode-se notar, a parte `#self` não é enviada na requisição. O servidor por sua vez, envia o documento RDF/XML que descreve o João, como ilustrado pelo Quadro 19.

```
HTTP/1.1 200 OK
Content-Type: application/rdf+xml

<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/">

    <rdf:Description rdf:about="http://exemplo.com/joao">
      <rdf:type rdf:resourse="foaf:Person"/>
       <rdf:name>João</rdf:name>
    </rdf:Description>
</rdf:RDF>
```

Quadro 19 - Resposta do servidor em RDF/XML
Fonte: Autoria própria

Como demonstrado, a estratégia URI *Hash* utiliza apenas uma requisição para dereferenciar URIs, e isso sem criar ambiguidade.

2.3 RDF/XML E RDF SCHEMA

Na Secção 2.2 foi abordado os conceitos que regem o *Linked Data*. Dentre esses conceitos está o modelo RDF de representação de dados. Nesta secção será abordado conceitos básicos e introdutórios sobre RDF/XML e em seguida será abordado sobre conceitos básicos de criação de vocabulários utilizando RDF Schema.

2.3.1 Conceitos Básicos do RDF/XML

RDF não é um formato de dados, mas sim um modelo. Existem vários formatos que implementam o modelo RDF. A seguir será abordado os conceitos básicos para criação de documentos no formato RDF/XML, que é o formato padrão definido pela W3C.

A Figura 2 apresenta um típico documento RDF/XML.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:foaf="http://xmlns.com/foaf/0.1/">
5
6    <rdf:Description rdf:about="http://exemplo.org/Leonardo_da_Vinci">
7      <foaf:name>Leonardo da Vinci</foaf:name>
8      <rdf:type rdf:resourse="foaf:Person"/>
9    </rdf:Description>
10
11 </rdf:RDF>
```

Figura 2 - Exemplo de documento RDF/XML
Fonte: Autoria própria

O documento apresentado na Figura 2 está descrevendo os seguintes fatos:

- Agora existe um dado na Web identificado pelo URI `http://exemplo.org/Leonardo_da_Vinci`.
- Esse dado possui um nome que é `"Leonardo da Vinci"`.
- Esse dado é uma pessoa.

A fim de tornar mais fácil o entendimento da sintaxe XML/RDF, a seguir é descrito o significado de cada linha deste documento:

- **Linha 1:** contém a informação de que o arquivo está no formato XML.
- **Linha 2:** é iniciada a declaração do elemento `rdf:RDF`. Esse é o elemento raiz de todo arquivo RDF.
- **Linha 3 e 4:** estão declarados dois `namespaces`, `rdf` e `foaf` (*namespaces* serão abordados na sequência).
- **Linha 6:** contém a declaração do elemento `rdf:Description`. No RDF esse elemento serve para descrever recursos. O atributo `rdf:about` indica qual recurso se está descrevendo, no caso, o recurso

`http://exemplo.org/Leonardo_da_Vinci`, que é um URI criado para referenciar a pessoa de Leonardo da Vinci.

- **Linha 7:** contém a declaração do elemento `foaf:name`. Esse elemento faz parte do vocabulário FOAF e é utilizado para atribuir nomes para os recursos, no caso, o nome é representado pelo valor literal `"Leonardo da Vinci"`.
- **Linha 8:** é indicado que o recurso se trata de uma pessoa por meio do URI abreviado `foaf:person`. Esse elemento faz parte do vocabulário FOAF.

2.3.2 *XML Namespaces*

Devido ao fato das URIs serem normalmente grandes em sua quantidade de caracteres, e serem constantemente repetidas em um mesmo arquivo RDF, é possível utilizar *XML Namespaces* para abreviar as referências feitas as URIs. Por exemplo: o elemento básico do RDF para descrever objetos possui o URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#Description`. Para não precisar utilizar esse URI em sua forma completa no documento, é recomendado criar um *namespace* utilizando a declaração `xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#`. Dessa forma o URI `http://www.w3.org/1999/02/22-rdf-syntax-ns#Description` pode ser referenciado de forma abreviada da seguinte maneira: `rdf:Description`. A utilização de *namespaces* pode ser percebida na Figura 2, analisando as linhas 3 e 4.

2.3.3 *rdf:Description*

O elemento `rdf:Description` é utilizado quando se pretende descrever um dado. Utiliza-se o atributo `rdf:about` para informar o URI que se está descrevendo. Dentro do elemento é feita toda a descrição do dado, informando as propriedades com seus respectivos valores, como ilustrado no Quadro 20.

```
<rdf:Description rdf:about="[URI]">
   [propriedade] [valor]
   [propriedade] [valor]
   [propriedade] [valor]
   ....
   ....
</rdf:Description>
```

Quadro 20 - Sintaxe do elemento *rdf:Description*
Fonte: Autoria própria

Como pode ser observado, foi criado apenas um elemento `rdf:Description` utilizado para descrever o dado identificado pelo URI `http://exemplo.org/Leonardo_da_Vinci`. Porém nada impede de no mesmo documento RDF/XML existir vários elementos `rdf:Description` identificando quando dados for necessário. A Figura 3 inclui no exemplo da Figura 2 outra descrição referente ao país Brasil, utilizando outro elemento `rdf:Description`.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:foaf="http://xmlns.com/foaf/0.1/"
5      xmlns:dbpedia="http://dbpedia.org/ontology/">
6
7    <rdf:Description rdf:about="http://exemplo.org/Leonardo_da_Vinci">
8      <foaf:name>Leonardo da Vinci</foaf:name>
9      <rdf:type rdf:resourse="foaf:Person"/>
10   </rdf:Description>
11
12   <rdf:Description rdf:about="http://exemplo.org/Brasil">
13     <foaf:name>Brasil</foaf:name>
14     <rdf:type rdf:resourse="dbpedia:country"/>
15   </rdf:Description>
16
17 </rdf:RDF>
```

Figura 3 - Exemplo de documento RDF/XML com dois elementos *rdf:Description*
Fonte: Autoria própria

2.3.4 Utilizando Propriedades Como Atributos

Outra característica do RDF/XML é que é possível informar as propriedades de um dado informando-as como atributos do elemento `rdf:Description`, desde que essa propriedade seja um *string* literal. A Figura 4 apresenta o mesmo exemplo da Figura 3, porém utilizando propriedades como atributos do elemento `rdf:Description`.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dbpedia="http://dbpedia.org/ontology/">

  <rdf:Description rdf:about="http://exemplo.org/Leonardo_da_Vinci"
          foat:name="Leonardo da Vinci">
    <rdf:type rdf:resourse="foaf:Person"/>
  </rdf:Description>

  <rdf:Description rdf:about="http://exemplo.org/Brasil"
          foaf:name="Brasil">
    <rdf:type rdf:resourse="dbpedia:country"/>
  </rdf:Description>

</rdf:RDF>
```

Figura 4 - Inserindo propriedades dentro do elemento *rdf:Description*
Fonte: Autoria própria

Como pode ser observado, a propriedade referente ao nome do Leonardo da Vinci e do Brasil estão descritas diretamente como um atributo do elemento `rdf:Description`.

2.3.5 Substituindo o Elemento *rdf:Description* Pelo Tipo

Outra forma de descrever um dado em RDF/XML é utilizando o valor da propriedade `rdf:type` como um elemento XML. A Figura 5 apresenta a descrição da pessoa Leonardo da Vinci utilizando o elemento `foaf:Person` no lugar de `rdf:Description`.

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/"
    xmlns:dbpedia="http://dbpedia.org/ontology/">

  <foaf:Person rdf:about="http://exemplo.org/Leonardo_da_Vinci">
    <foaf:name>Leonardo da Vinci</foaf:name>
  </foaf:Person>

</rdf:RDF>
```

Figura 5 - Utilizando o elemento *foaf:Person* no lugar do elemento *rdf:Description*
Fonte: Autoria própria

Desse modo não é mais preciso incluir a propriedade `rdf:type`, pois isso já e subentendido pelo elemento `foaf:Person`.

A seguir será abordado sobre criação de vocabulários RDF utilizando RDF Schema.

2.3.6 RDF Schema

O RDF Schema provê um vocabulário básico para descrever objetos e conceitos abstratos do mundo real. Para melhor compreensão, pode-se pegar o exemplo do universo conhecido. É sabido que o universo conhecido contém galáxias, e estas contêm estrelas, que contém planetas orbitando-as. Também é sabido que os seres humanos vivem no planeta terra, que por sua vez possui uma imensidão de coisas, como pessoas, animais, montanhas, árvores e casas. Utilizando o RDF Schema é possível descrever cada uma dessas coisas utilizando o conceito de classes.

2.3.7 *rdfs:Class*

Semelhante à forma que é utilizado em linguagens orientadas a objetos, no RDF Schema é possível representar o mundo real utilizando o elemento `http://www.w3.org/2000/01/rdf-schema#Class`. Por exemplo: é possível criar uma classe para representar o universo, outra para representar galáxias, outra para representar pessoas, e assim por diante.

A Figura 6 apresenta a criação de três classes que representam o universo, uma galáxia e um planeta, respectivamente.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
5
6      <rdfs:Class rdf:about="http://exemplo.org/Universo">
7          <rdfs:comment>O Universo</rdfs:comment>
8          <rdfs:label>Universo</rdfs:label>
9      </rdfs:Class>
10
11     <rdfs:Class rdf:about="http://exemplo.org/Galaxia">
12         <rdfs:comment>Uma galáxia</rdfs:comment>
13         <rdfs:label>Galáxia</rdfs:label>
14     </rdfs:Class>
15
16     <rdfs:Class rdf:about="http://exemplo.org/Planeta">
17         <rdfs:comment>Um Planeta</rdfs:comment>
18         <rdfs:label>Planeta</rdfs:label>
19     </rdfs:Class>
20
21 </rdf:RDF>
```

Figura 6 - Exemplo de descrição de classes utilizando RDF/XML
Fonte: Autoria própria

Como pode ser visto na linha 6, 11 e 16 o elemento utilizado para criar uma classe no RDF Schema é o `rdfs:Class`. O atributo `rdf:about` contém como valor um URI para ser o identificador da classe, para que outros recursos possam fazer referência a mesma. A propriedade `rdfs:comment` contém um comentário em valor literal sobre a finalidade da classe, e a propriedade `rdfs:label` contém um valor literal que serve de rótulo para a classe. Estas duas propriedades são apenas algumas dentre muitas que uma classe pode possuir.

O Quadro 21 apresenta algumas das principais classes do RDF Schema.

Classe	Descrição
`rdfs:Resource`	A classe que representa tudo. Todas as outras classes são sub-classes de rdfs:Resource.
`rdfs:Literal`	A classe de valores literais. Strings, datas, números.
`rdf:HTML`	A classe para valores HTTP literais.
`rdf:XMLLiteral`	A classe para valores XML literais.
`rdfs:Class`	A classe de classes.
`rdf:Property`	A classe das propriedades.
`rdf:List`	A classe das listas.

Quadro 21 - Algumas das principais classes do RDF Schema
Fonte: (RDF SCHEMA 1.1, 2015)

2.3.8 *rdfs:subClassOf*

No RDF Schema também é possível identificar que uma classe é subclasse de outra. No exemplo anterior foi demonstrado a criação da classe Planeta. Para tentar tornar esse exemplo mais rico em significado, pode-se tentar representar as categorias de planetas que existem. Alguns planetas são gasosos e outros são rochosos. Pode-se então definir uma classe para a categoria de planetas rochosos, a qual será uma subclasse da classe Planeta. A Figura 7 apresenta a criação dessa classe, como também o uso de subclasses no RDF Schema.

Figura 7 - Exemplo de subclasse no RDF Schema
Fonte: Autoria própria

Como pode ser observado, para identificar uma classe como subclasse de outra, basta utilizar a propriedade `rdfs:subClassOf` apontando para o recurso que representa a classe superior.

2.3.9 *rdfs:Property*

O RDF Schema se assemelha com linguagens orientadas a objetos na criação de classes, instâncias e subclasses, porém, enquanto em orientação a objetos as propriedades são criadas dentro da classe, no RDF Schema as propriedades são criadas separadas das suas classes, inclusive, uma propriedade pode ser utilizada em mais de uma classe.

A criação de propriedades é feita utilizando o elemento `rdfs:Property`. A Figura 8 mostra a criação de propriedades utilizando RDF Schema.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
5
6      <rdfs:Class rdf:about="http://exemplo.org/Documento">
7          <rdfs:comment>Representa um Documento.</rdfs:comment>
8          <rdfs:label>Documento</rdfs:label>
9      </rdfs:Class>
10
11     <rdfs:Class rdf:about="http://exemplo.org/Pessoa">
12         <rdfs:comment>Representa uma Pessoa.</rdfs:comment>
13         <rdfs:label>Pessoa</rdfs:label>
14     </rdfs:Class>
15
16     <rdf:Property rdf:about="http://exemplo.org/autor">
17         <rdfs:domain rdf:resource="http://exemplo.org/Documento"/>
18         <rdfs:range rdf:resource="http://exemplo.org/Pessoa"/>
19     </rdf:Property>
20
21 </rdf:RDF>
```

Figura 8 - Exemplo de classe e propriedade utilizando RDF Schema
Fonte: Autoria própria

Primeiramente, nas linhas 6 e 11 são criadas duas classes, `http://exemplo.org/Documento` e `http://exemplo.org/Pessoa`, respectivamente. Na linha 16 é possível analisar a declaração da propriedade `http://exemplo.org/autor`. Diferentemente de linguagens orientadas a objetos, no RDF Schema as propriedades são criadas separadas da classe. Uma atenção especial deve ser dada as propriedades `rdfs:domain` (linha 17) e `rdfs:range` (linha 18). A declaração da propriedade `rdfs:domain` define que a propriedade http://exemplo.org/autor poderá ser usada em classes e subclasses do tipo `http://exemplo.org/Documento`. No entanto, é possível utilizar mais de uma propriedade `rdfs:domain` na criação de uma propriedade, aumentando assim o escopo de onde a propriedade poderá ser utilizada. Isto permite que uma propriedade possa ser utilizada em mais de uma classe, sem que pra isso precise alterar a declaração original da classe, ou seja, nada impede que uma pessoa qualquer decida adicionar uma propriedade ao escopo de uma classe de terceiros, bastando para isso utilizar a propriedade `rdfs:domain`. Já a declaração da propriedade `rdfs:range` define que a propriedade `http://exemplo.org/autor` pode ser considerada do tipo `http://exemplo.org/Pessoa`. A propriedade `rdfs:range` também pode ser utilizada mais de uma vez, aumentando os tipos de classes que a propriedade pode ser considerada.

A Figura 9 apresenta um exemplo de uso da propriedade “autor” criada no exemplo anterior.

```
1  <?xml version="1.0" encoding="utf-8" ?>
2  <rdf:RDF
3      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
5      xmlns:exemplo="http://exemplo.org/">
6
7      <exemplo:Documento rdf:about="http://exemplo.org/Documento_12345">
8          <rdfs:comment>Documento 12345.</rdfs:comment>
9          <exemplo:autor rdf:resource="http://exemplo.org/Joao" />
10     </exemplo:Documento>
11
12 </rdf:RDF>
```

Figura 9 - Exemplo de uso de uma propriedade utilizando RDF Schema
Fonte: Autoria própria

2.4 SPARQL

No *Linked Data* o mundo real pode ser representado utilizando o modelo RDF. No entanto, a W3C também criou uma linguagem de consultas chamada SPARQL. Com ela é possível disparar consultas contra fontes de dados habilitadas para prover dados RDF.

> SPARQL pode ser usado para expressar consultas contra diversas fontes de dados, contanto que os dados estejam armazenados como RDF. SPARQL inclui a capacidade de procurar exclusivamente ou opcionalmente por padrões de triplas por meio de suas relações umas com as outras. SPARQL também suporta extensivos testes de valores e restrições de consultas em determinadas fontes de dados. O resultado de consultas SPARQL pode ser uma lista de resultados ou um documento RDF (SPARQL..., 2013).

2.4.1 Sintaxe Básica do SPARQL

O SPARQL define uma sintaxe que possibilita realizar variados tipos de consultas contra bases RDF. Basicamente uma consulta SPARQL consiste em especificar um ou mais padrões de triplas para que o servidor confira se esses padrões especificados conferem com as triplas disponíveis em sua base.

A sintaxe básica de uma consulta SPARQL é apresentada no Quadro 22.

```
SELECT [valores]
WHERE {
    [padrão de tripla] .
    [padrão de tripla] .
    [padrão de tripla] .
}
```

Quadro 22 - Sintaxe básica de uma consulta SPARQL
Fonte: Autoria própria

Dentro da cláusula `WHERE` deve ser informado um ou mais padrões de triplas. Cada padrão é separado pelo caractere ".". O servidor RDF irá comparar esses padrões com as triplas disponíveis em sua base, caso os padrões confiram com alguma tripla, os valores dessa tripla serão retornados dependendo do que foi especificado na clausula `SELECT`. Cada padrão deve ser informado seguindo a regra: recurso, propriedade e valor. Cada um desses valores pode ser um URI real ou uma variável.

Para melhor entendimento de como funciona uma consulta SPARQL, pode-se analisar o Quadro 23, onde é apresentada uma base RDF que contém duas triplas e, em seguida, é apresentado um exemplo de consulta SPARQL, como também o resultado que seria retornado dessa consulta.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:sabe   "Java" .
ex:maria  ex:sabe   "C++".
```

Consulta SPARQL

```
SELECT *
   WHERE {
       ?id ?propriedade ?valor
   }
```

Resultado da Consulta

```
id         propriedade   valor
---------------------------------
ex:joao    ex:sabe       "Java" .
ex:maria   ex:sabe       "C++"
```

Quadro 23 - Exemplo de consulta SPARQL
Fonte: Autoria própria

Como pode ser observado, a consulta contém um único padrão de triplas dentro da clausula `WHERE`. Cada um dos elementos do padrão é uma variável, que é identificada pelo caractere ? (ponto de interrogação) precedendo o nome. O resultado retornado foi todas as

triplas da base consultada. Isso acontece devido ao padrão de triplas que foi informado, pois possui todos os elementos como variáveis. O (*) da clausula `SELECT` significa que se espera que retorne todas as variáveis informadas na clausula `WHERE`. Se no lugar do (*) fosse informado a variável (?valor), somente essa variável seria retornada.

O Quadro 24 apresenta a consulta do Quadro 23 modificada para retornar apenas o identificador das pessoas que sabem “Java”.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:sabe   “Java”
ex:maria  ex:sabe   “C++”
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

SELECT ?id
   WHERE {
       ?id ex:sabe “Java”
   }
```

Resultado da Consulta

```
id
----------
ex:joao
```

Quadro 24 - Exemplo de consulta SPARQL
Fonte: Autoria própria

Como pode ser observado, antes da cláusula `SELECT` foi informado um *namespace* utilizando a palavra-chave `@PREFIX`. O funcionamento de *namespaces* no SPARQL segue a mesma lógica do uso de *namespaces* no RDF/XML. Também pode ser observado que o padrão de triplas especificado na cláusula `WHERE` contém apenas uma variável seguida de dois elementos fixos. Esse padrão diz para o servidor RDF: procure por todas as triplas que possuam a propriedade `ex:sabe` e o valor `“Java”`, e pra cada tripla encontrada, substitua a variável `?id` pelo URI que contém elas.

O próximo exemplo apresenta uma consulta que contém dois padrões de triplas na clausula `WHERE`, onde será consultado pelo nome das pessoas que sabem `“C++”`.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:sabe   "Java"
ex:joao   ex:nome   "João Carlos"
ex:maria  ex:sabe   "C++"
ex:maria  ex:nome   "Maria Miranda"
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

SELECT ?nome
   WHERE {
        ?id ex:sabe "C++" .
        ?id ex:nome ?nome
   }
```

Resultado da Consulta

```
nome
-----------------
"Maria Miranda"
```

Quadro 25 - Exemplo de consulta SPARQL
Fonte: Autoria própria

Como pode ser observado, foram incluídos dois padrões na cláusula `WHERE`. No primeiro padrão está definindo que se quer atribuir na variável (`?id`) o identificador de triplas que contém a propriedade `ex:sabe` e o valor `"C++"`. O segundo padrão define que cada identificador encontrado no primeiro padrão também deve possuir uma propriedade `ex:nome`, e se caso encontrar, o seu valor deverá ser atribuído na variável (`?nome`). A seguir será apresentado os passos executados no lado do servidor RDF ao receber essa consulta:

- O servidor procura por triplas que contenha a propriedade `ex:sabe` e o valor `"C++"`.
- Se encontrar alguma tripla, atribui o valor do identificador na variável (`?id`).
- Para cada tripla encontrada no passo 2, procura por triplas que contenham o (`?id`) seguido da propriedade `ex:nome`.
- Se encontrar, atribui o valor encontrado na variável (`?nome`).
- Retorna todas as variáveis (`?nome`) encontradas.

2.4.2 *Filter*

No SPARQL também é possível aplicar filtros mais específicos. O Quadro 26 apresenta uma consulta que busca todas as pessoas que sejam maiores de idade.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:nome   "João Carlos"
ex:joao   ex:idade  15
ex:maria  ex:nome   "Maria Miranda"
ex:maria  ex:idade  18
ex:alice  ex:nome   "Alice Carvalho"
ex:alice  ex:idade  21
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

SELECT ?nome
   WHERE {
        ?id ex:nome  ?nome .
        ?id ex:idade ?idade .
        FILTER(?idade > 18)
   }
```

Resultado da Consulta

```
nome
-----------------
"Alice Carvalho"
```

Quadro 26 - Exemplo do uso de filtro no SPARQL
Fonte: Autoria própria

Como pode-se notar, a clausula FILTER restringiu apenas pessoas cuja a idade seja maior que 18 anos. Dentro da clausula FILTER também é possível usar operadores lógicos como "`&&`" (E) , "`||`" (OU), trabalhar com datas ou filtrar por partes de um *string*, dentre outras possibilidades.

2.4.3 *Optional*

O SPARQL também disponibiliza a clausula `OPTIONAL`. Com ela é possível especificar que um ou mais padrões de triplas são desejados, porém não obrigatórios. Se o servidor RDF verificar que uma tripla não confere com o que for pedido como opcional, o mesmo não irá

invalidá-la, apenas deixará as variáveis nulas. O Quadro 27 exemplifica o uso da cláusula `OPTIONAL`, apresentando uma consulta que busca pelo nome de todas as pessoas da base, e opcionalmente o que essas pessoas sabem.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:nome   "João Carlos"
ex:joao   ex:sabe   "Java"
ex:joao   ex:sabe   "C++"
ex:maria  ex:nome   "Maria Miranda"
ex:maria  ex:idade  18
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

SELECT ?nome ?sabe
   WHERE {
        ?id  ex:nome  ?nome .
        OPTIONAL {
            ?id  ex:sabe ?sabe .
        }
   }
```

Resultado da Consulta

```
nome                sabe
-------------------------
"João Carlos"       "Java"
"João Carlos"       "C++"
"Maria Miranda"     null
```

Quadro 27 - Exemplo do uso da clausula OPCIONAL no SPARQL
Fonte: Autoria própria

Como pode ser observado, o resultado apresenta a variável (`?sabe`) da Maria com o valor nulo. Isso acontece devido ao uso da cláusula `OPTIONAL` e porque a Maria não possui nenhuma propriedade `ex:sabe` atribuída para ela. Se a cláusula `OPTIONAL` fosse removida, Maria não estaria no resultado.

2.4.4 *Union*

Quando se pretende fazer uma junção de padrões de triplas específicos utiliza-se a clausula `UNION`. O Quadro 28 apresenta uma consulta onde é solicitado o nome de todas as pessoas que sabem "Java" e todas as pessoas que sejam maiores de idade.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:nome   "João Carlos"
ex:joao   ex:sabe   "Java"
ex:maria  ex:nome   "Maria Miranda"
ex:maria  ex:idade  18
ex:alice  ex:nome   "Alice Carvalho"
ex:alice  ex:idade  21
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

SELECT ?nome
   WHERE {
      {
         ?id  ex:nome  ?nome .
         ?id  ex:sabe  "Java"
      }
      UNION
      {
         ?id  ex:nome  ?nome .
         ?id  ex:idade ?idade .
         FILTER (?idade > 18)
      }
   }
```

Resultado da Consulta

```
nome
------------------
"João Carlos"
"Alice Carvalho"
```

Quadro 28 - Exemplo do uso da clausula *UNION* no SPARQL
Fonte: Autoria própria

2.4.5 *Ask*

Além da consulta básica `SELECT`, o SPARQL disponibiliza outros tipos de consulta, uma delas é consulta `ASK`. Essa consulta serve para saber se um ou mais padrões de triplas possuem resultados. Essa consulta apenas retorna verdadeiro ou falso, como apresentado no Quadro 29.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:nome   "João Carlos"
ex:joao   ex:sabe   "Java"
ex:maria  ex:nome   "Maria Miranda"
ex:maria  ex:idade  18
ex:alice  ex:nome   "Alice Carvalho"
ex:alice  ex:idade  21
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

ASK
   WHERE {
         ?id  ex:nome  ?nome .
         ?id  ex:sabe  "Java"
   }
```

Resultado da Consulta

```
TRUE
```

Quadro 29 - Exemplo de consulta *ASK* no SPARQL
Fonte: Autoria própria

2.4.6 *Construct*

No SPARQL é possível receber a resposta de uma consulta em formato RDF. Para isso é utilizada a consulta `CONSTRUCT`, como demonstrada no Quadro 30.

Base RDF consultada

```
@prefixo ex: <http://exemplo.com/>

ex:joao   ex:nome   "João Carlos"
ex:maria  ex:idade  15
ex:maria  ex:nome   "Maria Miranda"
ex:maria  ex:idade  16
ex:alice  ex:nome   "Alice Carvalho"
ex:alice  ex:idade  21
```

Consulta SPARQL

```
PREFIX ex:<http://exemplo.org/>

CONSTRUCT {
   ?id  ex:nome  ?nome
}
WHERE {
   ?id  ex:nome  ?nome .
   ?id  ex:idade ?idade .
   FILTER (?idade > 18)
}
```

Resultado da Consulta

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://exemplo.org/">
  <rdf:Description rdf:about="http://exemplo.org/alice">
    <foaf:name>Alice Carvalho</foaf:name>
  </rdf:Description>
</rdf:RDF>
```

Quadro 30 - Exemplo do uso da consulta *CONSTRUCT*
Fonte: Autoria própria

2.4.7 *Describe*

O SPARQL também disponibiliza a consulta DESCRIBE. Esse tipo de consulta serve para solicitar toda descrição de um recurso especifico em formato RDF, como demostrado no Quadro 31.

Base RDF consultada

```
@prefixo rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefixo foaf: <http://xmlns.com/foaf/0.1/
@prefixo ex: <http://exemplo.com/>

ex:joao   rdf:type   foaf:Person
ex:joao   foaf:name  "João Carlos"
ex:joao   ex:idade   15
ex:maria  rdf:type   foaf:Person
ex:maria  foaf:name  "Maria Miranda"
ex:maria  ex:idade   16
```

Consulta SPARQL

```
PREFIX ex: <http://exemplo.org/>

CONSTRUCT ex:joao
```

Resultado da Consulta

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://exemplo.org/"
    xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <rdf:Description rdf:about="http://exemplo.org/Joao">
     <rdf:type
rdf:resource="http://xmlns.com/foaf/0.1/Person"></rdf:type>
    <foaf:name>Joao Carlos</foaf:name>
    <ex:idade>15</ex:idade>
  </rdf:Description>
</rdf:RDF>
```

Quadro 31 - Exemplo de uso da consulta *DESCRIBE*
Fonte: Autoria própria

3 MATERIAIS E MÉTODOS

3.1 INFRAESTUTURA

Para desenvolver uma aplicação Web que utilize os padrões *Linked* Data, foi utilizado Java para o *backend*, e JSP, HTML e Javascript para o *frontend*. Como container Web foi utilizado o Apache Tomcat 7x.

3.2 APACHE JENA

Apache Jena (ou apenas Jena) é um *framework* Java gratuito e open source para construção de aplicações baseadas em Web semântica e *Linked Data*. O *framework* é composto por diferentes API's, que interagem em conjunto para processar dados RDF (APACHE JENA, 2015, tradução própria).

3.2.1 Principais Classes do Jena

O Jena possui diversas classes para manipulação de arquivos RDF. Segue as principais classes do Jena.

- **`Model:`** classe que representa uma base RDF.
- **`Resource:`** classe que representa um recurso.
- **`Statement:`** classe que representa uma tripla.
- **`Query:`** classe que representa uma consulta SPARQL.
- **`QueryExection:`** classe utilizada para disparar consultas SPARQL.
- **`ResultSet:`** classe que representa uma lista de resultados retornados de uma consulta SPARQL.
- **`Reasoner:`** classe que representa motores de inferência.

3.2.2 Criando Modelos RDF

A principal classe do Jena é a classe `Model`. Essa classe é utilizada para representar uma fonte de dados RDF. Para criação de uma classe `Model` utiliza-se a classe `ModelFactory`. Essa classe por sua vez contém métodos para criação de modelos vazios, ou modelos carregados a partir de arquivos RDF.

3.2.3 Manipulando Modelos

O principal método para criação de modelos no Jena é o método `ModelFactory.createDefaultModel`. A Quadro 32 apresenta a criação de um modelo vazio.

```java
package testes;
import com.hp.hpl.jena.rdf.model.*;

public class ApacheJena {
    public static void main(String[] args) throws Exception {
        //cria um modelo vazio
        Model model = ModelFactory.createDefaultModel();

        //imprime o conteúdo do modelo utilizando o formato RDF/XML
        model.write(System.out, "RDF/XML");
    }
}
```

Saída

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
</rdf:RDF>
```

Quadro 32 - Criação de um modelo vazio utilizando o Jena
Fonte: Autoria própria

Como pode ser observado, a saída desse programa é um modelo RDF sem nenhuma tripla. Para descrever um novo recurso nesse modelo vazio utiliza-se o método `createResource`. Esse método recebe como parâmetro o URI que irá identificar o novo recurso criado, e retorna uma instância da classe `Resource`, como demonstrado no Quadro 33.

```java
package testes;
import com.hp.hpl.jena.rdf.model.*;

public class ApacheJena {
    public static void main(String[] args) throws Exception {
        //cria um modelo vazio
        Model model = ModelFactory.createDefaultModel();

        //cria um novo recurso nesse modelo
        Resource joao = model.createResource("http://exemplo.com/Joao");

        //adiciona propriedades para o recurso
        joao.addProperty(RDF.type, FOAF.Person);
        joao.addProperty(FOAF.name, "João Roberto");

        //imprime o conteúdo do modelo utilizando o formato RDF/XML
        model.write(System.out, "RDF/XML");
    }
}
```

Saída

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://xmlns.com/foaf/0.1/" >
  <rdf:Description rdf:about="http://exemplo.com/Joao">
    <j.0:name>João Roberto</j.0:name>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  </rdf:Description>
</rdf:RDF>
```

Quadro 33 - Criando recursos em um modelo no Jena
Fonte: Autoria própria

Como pode ser observado, a classe `Resource` contém o método `addProperty()`, que é utilizado para adicionar propriedades a um recurso informado a propriedade e valor desejado. Outro detalhe que pode ser reparado é o uso das classes RDF e FOAF. Essas classes contêm propriedades já conhecidas dos vocabulários RDF Schema e FOAF, respectivamente. Além dessas classes, o Jena disponibiliza várias outras, referentes a outros vocabulários já difundidos na Web.

O Quadro 34 apresenta uma descrição dos principais métodos das classes `Model` e `Resource`.

Principais métodos da classe Model	
`createResource`	Cria um novo recurso no modelo.
`getResource`	Pega um recurso já existente no modelo.
`add`	Método polimórfico que adiciona ao modelo recursos ou outros modelos.
`union`	Mescla outro modelo ao modelo atual e retorna o modelo mesclado.
`createTypedLiteral`	Cria um tipo literal nos padrões do RDF.
`createProperty`	Cria uma propriedade para ser utilizada no modelo.

`read`	Carrega dados de um arquivo ou de um InputStream no modelo.
Principais métodos da classe Resource	
`getLocalName`	Retorna o nome do recurso sem o namespace.
`getNameSpace`	Retorna o namespace do recurso.
`getModel`	Retorna o modelo no qual o recurso está incluído.
`hasProperty`	Verifica se o recurso possui alguma propriedade específica.
`removeAll`	Remove uma propriedade específica do recurso.

Quadro 34 - Principais métodos das classes *Model* e *Resource*
Fonte: Autoria própria

3.2.4 Criando Modelos a Partir de Arquivos RDF

O JENA também possibilita a leitura de modelos RDF a partir de arquivos. O Quadro 35 apresenta como carregar um modelo a partir de um arquivo RDF.

```java
package testes;
import java.io.InputStream;

public class ApacheJena {
    public static void main(String[] args) throws Exception {
        //cria um modelo vazio
        Model model = ModelFactory.createDefaultModel();

        //carrega o modelo com os dados de um arquivo que está no formato RDF/XML
        InputStream in = FileManager.get().open("c:/exemplo.rdf");
        model.read(in, "RDF/XML");

        //imprime o conteúdo do modelo utilizando o formato RDF/XML
        model.write(System.out, "RDF/XML");
    }
}
```

Saída

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:foaf="http://xmlns.com/foaf/0.1/" >
  <rdf:Description rdf:about="http://exemplo.org/Joao">
    <foaf:knows rdf:resource="http://exemplo.org/Maria"/>
    <foaf:name>Joao Roberto</foaf:name>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://exemplo.org/Maria">
    <foaf:knows rdf:resource="http://exemplo.org/Joao"/>
    <foaf:name>Maria</foaf:name>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
  </rdf:Description>
</rdf:RDF>
```

Quadro 35 - Carregando um modelo no Jena à partir de um arquivo RDF
Fonte: Autoria própria

3.2.5 Unindo Modelos

No Jena também é possível unir modelos em um só. Isso é útil quando se pretende trabalhar com diferentes bases RDF vindas de fornecedores distintos. O Quadro 36 apresenta uma maneira de unir modelos no Jena.

```java
package testes;
import com.hp.hpl.jena.rdf.model.Model;

public class ApacheJena {
    public static void main(String[] args) throws Exception {
        //cria um modelo vazio
        Model model1 = ModelFactory.createDefaultModel();
        Model model2 = ModelFactory.createDefaultModel();

        //adiciona um recurso no modelo 1
        Resource joao = model1.createResource("http://exemplo.com/Joao_Carlos");
        joao.addProperty(RDF.type, FOAF.Person);
        joao.addProperty(FOAF.name, "Joao Carlos");

        //adiciona um recurso no modelo 2
        Resource maria = model2.createResource("http://exemplo.com/Maria_Carvalho");
        maria.addProperty(RDF.type, FOAF.Person);
        maria.addProperty(FOAF.name, "Maria de Carvalho");

        //une o modelo 1 e o modelo 2, e retorna o modelo resultante
        Model modeloUnido = model1.union(model2);

        //imprime o conteúdo do modelo uniodo utilizando o formato RDF/XML
        modeloUnido.write(System.out, "RDF/XML");
    }
}
```

Saída

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:j.0="http://xmlns.com/foaf/0.1/" >
  <rdf:Description rdf:about="http://exemplo.com/Maria_Carvalho">
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <j.0:name>Maria de Carvalho</j.0:name>
  </rdf:Description>
  <rdf:Description rdf:about="http://exemplo.com/Joao_Carlos">
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <j.0:name>Joao Carlos</j.0:name>
  </rdf:Description>
</rdf:RDF>
```

Quadro 36 - Unindo modelos no Jena
Fonte: Autoria própria

3.2.6 Realizando Consultas SPARQL

No Jena também é possível realizar consultas SPARQL em modelos. Para isso o Jena disponibiliza as classes `Query`, `QueryFactory`, `QueryExecution` e `QueryExecutionFactory`. O quadro 37 apresenta os passos envolvidos na realização de uma consulta SPARQL no Jena.

```java
public class ApacheJena {
    public static void main(String[] args) throws Exception {
        //carrega um modelo desejado
        InputStream input = FileManager.get().open("c:/exemplo.rdf");
        Model model = ModelFactory.createDefaultModel().read(input, "RDF/XML");

        //consulta SPARQL
        String queryString =
                  "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
                + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"

                + " SELECT ?nome WHERE {"
                + "   ?pessoa rdf:type foaf:Person ."
                + "   ?pessoa foaf:name ?nome   ."
                + " }";

        //cria um objeto Query utilizando o método QueryFactory.create
        Query query = QueryFactory.create(queryString);
        //cria um objeto QueryExecution a partir da query e do modelo
        QueryExecution qexec = QueryExecutionFactory.create(query, model);

        try {
            //executa a consulta
            ResultSet results = qexec.execSelect();
            //itera os resultados encontrados
            for (; results.hasNext();) {
                //Processa o resultado aqui
                QuerySolution solution = results.next();
                //pega uma variavel que foi informada na consulta
                RDFNode node = solution.get("nome");
                System.out.println(node);
            }
        } finally {
            qexec.close();
        }
    }
}
```

Saída

```
João Carlos
Maria de Carvalho
```

Quadro 37 - Realizando uma consulta SPARQL no Jena
Fonte: Autoria própria

3.2.7 Realizando Consultas em uma Base RDF Remota

Também é possível realizar consultas em servidores remotos. O Quadro 38 apresenta uma consulta disparada contra base do DBPedia[6], buscando os dez primeiros pintores que encontrar, que possuam seus nomes iniciando pela letra "A".

```
 9  public class ApacheJena {
10      public static void main(String[] args) {
11          //consulta SPARQL
12          String queryString =
13                    "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
14                  + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
15
16                  + " SELECT ?uri WHERE {"
17                  + "   ?uri rdf:type <http://dbpedia.org/class/yago/Painter110391653> ."
18                  + "   ?uri foaf:name ?nome ."
19                  + "   FILTER regex(?nome, '^A', 'i')"
20                  + " }"
21                  + " LIMIT 10";
22
23          //cria a query que será enviada para o dbpedia
24          Query query = QueryFactory.create(queryString);
25          //cria o objeto QueryExecution, relacionando a query com o dbpedia
26          QueryExecution qexec =
27                  QueryExecutionFactory.sparqlService("http://dbpedia.org/sparql", query);
28
29          try {
30              ResultSet results = qexec.execSelect();
31              for (; results.hasNext();) {
32                  //processa o resultado aqui
33                  QuerySolution q = results.next();
34                  System.out.println(q.get("uri"));
35              }
36          } finally {
37              qexec.close();
38          }
39      }
40  }
```

Saída

```
http://dbpedia.org/resource/Andrew_Foley_(writer)
http://dbpedia.org/resource/Arie_Smit
http://dbpedia.org/resource/Acee_Blue_Eagle
http://dbpedia.org/resource/Afro_Basaldella
http://dbpedia.org/resource/Agustino_Da_Vaprio
http://dbpedia.org/resource/Albert_Harjo
http://dbpedia.org/resource/Albert_Kotin
http://dbpedia.org/resource/Albert_Malet_(painter)
http://dbpedia.org/resource/Alberto_Burri
http://dbpedia.org/resource/Aldro_Hibbard
```

Quadro 38 - Realizando uma consulta SPARQL no Jena em uma base remota
Fonte: Autoria própria

[6] DBpedia é um esforço colaborativo para extrair informações estruturadas da Wikipedia, tornando estas informações disponíveis na Web (DBPEDIA, 2015, tradução própria).

3.2.8 Inferência com *Jena Rules*

Outra funcionalidade importante do Jena é o seu motor de inferência, este permite inferir novos conhecimentos em um modelo, utilizando regras chamadas *Jenas Rules*.

O Jena disponibiliza três tipos de inferências:

- ***Forward Chaining*:** Conclusões, onde um conjunto de fatos levam a uma conclusão.
- ***Backward Chaining*:** fatos, parte de uma hipótese que deve ser provada através de fatos.
- ***Hybrid Chaining*:** é a junção de *Backward Chaining* com o *Forward Chaining*.

As regras do Jena Rules são descritas utilizando uma sintaxe semelhante ao SPARQL. Basicamente as regras são aplicadas da seguinte maneira: dada uma condição, se verdadeira, uma nova afirmação (tripla) é gerada.

O Quadro 39 apresenta um exemplo de uma regra onde se deseja verificar se determinada pessoa pertence a geração X, Y ou Z, se baseando no ano de nascimento das mesmas.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
xmlns:foaf="http://xmlns.com/foaf/0.1/".
xmlns:ex="http://exemplo.com/".

[geracaoX:
( ?p rdf:type foaf:Person),
( ?p ex:anoNascimento ?anoNascimento)
ge(?anoNascimento, 1960),
le(?anoNascimento, 1980)
->
( ?p ex:geracao "X" )
]

[geracaoY:
( ?p rdf:type foaf:Person),
( ?p ex:anoNascimento ?anoNascimento)
ge(?anoNascimento, 1981),
le(?anoNascimento, 1995)
->
( ?p ex:geracao "Y" )
]

[geracaoZ:
( ?p rdf:type foaf:Person),
( ?p ex:anoNascimento ?anoNascimento)
ge(?anoNascimento, 1996),
le(?anoNascimento, 2010)
->
( ?p ex:geracao "Z" )
]
```

Quadro 39 – Regra *Jena Rules* para verificar a geração de uma pessoa
Fonte: Autoria própria

Como pode ser observado no Quadro 39, foram declaradas três regras: uma para deduzir a geração X, outra para a geração Y, e outra para a geração Z. Cada regra inicia e

termina com o carctere “[” e “]”, e cada uma possui um nome que termina com o caracter “:”. Para a geração X foi definido que são as pessoas que nasceram entre os anos 1960 e 1980, para a geração Y, entre os anos 1981 e 1995, e para a geração Z, entre os anos 1996 e 2010. O Quadro 40 apresenta um arquivo RDF antes e depois de ser inferido pela regra apresentada no Quadro 39.

Antes de inferir a regra

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <foaf:Person rdf:about="http://exemplo.org/Maria">
      <foaf:name>Maria</foaf:name>
      <ex:anoNascimento>1988</anoNacimento>
  </foaf:Person>

  <foaf:Person rdf:about="http://exemplo.org/Joao">
      <foaf:name>Joao Roberto</foaf:name>
      <ex:anoNascimento>2002</anoNacimento>
  </foaf:Person>

</rdf:RDF>
```

Depois de inferior a regra

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">

  <foaf:Person rdf:about="http://exemplo.org/Maria">
      <foaf:name>Maria da Dores</foaf:name>
      <ex:anoNascimento>1988</anoNacimento>
      <ex:geracao>Y</ex:geracao>
  </foaf:Person>

  <foaf:Person rdf:about="http://exemplo.org/Joao">
      <foaf:name>Joao Roberto</foaf:name>
      <ex:anoNascimento>2002</anoNacimento>
      <ex:geracao>Z</ex:geracao>
  </foaf:Person>

</rdf:RDF>
```

Quadro 40 – Arquivo RDF antes e depois de ser inferido pela regra apresentada no Quadro 39
Fonte: Autoria própria

A Figura 10 apresenta o código Java utilizado para fazer esta inferência.

```java
public static void main(String[] args) throws Exception {
    //carrega o arquivo RDF
    InputStream in = FileManager.get().open("C:/arquivo.rdf");
    model = com.hp.hpl.jena.rdf.model.ModelFactory.createDefaultModel().read(in, null);
    //Cria um novo recurso que vai configurar o Reasoner
    Resource configuracao = model.createResource();
    //define o tipo de inferência
    configuracao.addProperty(ReasonerVocabulary.PROPruleMode, "hybrid");
    //passa o caminho do arquivo que contém as regras
    configuracao.addProperty(ReasonerVocabulary.PROPruleSet, "C:/regras.txt");
    //Cria uma instancia do Reasoner
    Reasoner reasoner = GenericRuleReasonerFactory.theInstance().create(configuracao);
    //Infere as regras no modelo, criando um novo modelo inferido
    InfModel infModel = com.hp.hpl.jena.rdf.model.ModelFactory.createInfModel(reasoner, model);
    //impre o novo modelo inferido na tela
    infModel.write(System.out);
}
```

Figura 10 – Código Java utilizado para fazer a inferência apresentada no Quadro 39
Fonte: Autoria própria

4 RESULTADOS

Para demonstrar a utilização dos padrões *Linked Data*, será apresentado o processo envolvido para construir uma aplicação Java Web que consome dados RDF utilizando o *framework* Jena.

4.1 FUNCIONAMENTO DA APLICAÇÃO

A aplicação tem como propósito disponibilizar o conteúdo da Bíblia[7] online para que usuários possam ler a Bíblia de maneira prática e rápida. Porém, além de disponibilizar a Bíblia para leitura, a aplicação também disponibilizará a leitura de comentários bíblicos. Para isso, a aplicação buscará dados em três provedores RDF fictícios. São eles:

- **biblia-rdf.com:** Fornece uma versão da bíblia em RDF/XML.
- **comentarios-scofield.com:** Fornece uma base de comentários da bíblia em RDF/XML.
- **comentarios-dlmoody.com:** Fornece uma base de comentários da bíblia em RDF/XML.

A aplicação consumirá os dados em RDF desses fornecedores utilizando Jena. Para apresentação do conteúdo da bíblia, como também dos comentários bíblicos, a aplicação utilizará consultas SPARQL. Além disso a aplicação utilizará regras de inferência do *Jena Rules* para deduzir quais versículos similares podem possuir um mesmo comentário.

[7] Bíblia é o nome dado a um conjunto de livros que são considerados sagrados por algumas religiões, principalmente pelo Cristianismo.

4.1.1 Fornecedor biblia-rdf.com

O fornecedor **biblia-rdf.com** contém uma base RDF contendo todos os versículos da bíblia em português. A aplicação Bíblia App busca os versículos da bíblia realizando consultas SPARQL nesse fornecedor. O Quadro 41 apresenta os recursos descritos nessa base.

`http://biblia-rdf.com/biblia#LivroBiblia`	Classe que representa um livro da bíblia. Todos os 66 livros da bíblia também possuem uma classe própria que é subclasse dessa. Ex.: `http://biblia-rdf.com/biblia#MATEUS`, `http://biblia-rdf.com/biblia#MARCOS`, `http://biblia-rdf.com/biblia#LUCAS` etc.
`http://biblia-rdf.com/biblia#Versiculo`	Classe que representa um único versículo da bíblia. Para cada versículo (mais de 31.000) foi criado um recurso que possui esse tipo.
`http://biblia-rdf.com/biblia#Comentario`	Casse que pode ser utilizada por qualquer aplicação externa tecer comentários nos versículos da bíblia.
`http://biblia-rdf.com/biblia#livro`	Propriedade que aponta para um livro. Criada para ser usada nas classes `http://biblia-rdf.com/biblia#Versiculo` e `http://biblia-rdf.com/biblia#Comentario`.
`http://biblia-rdf.com/biblia#ncap`	Propriedade que representa um número de capítulo. Criada para ser usada nas classes `http://biblia-rdf.com/biblia#Versiculo` e `http://biblia-rdf.com/biblia#Comentario`.
`http://biblia-rdf.com/biblia#nvers`	Propriedade que representa um número de versículo. Criada para ser usada nas classes `http://biblia-rdf.com/biblia#Versiculo` e `http://biblia-rdf.com/biblia#Comentario`.
`http://biblia-rdf.com/biblia#texto`	Propriedade que representa o conteúdo de um versículo ou de um comentário. Criada para ser usada nas classes `http://biblia-rdf.com/biblia#Versiculo` e `http://biblia-rdf.com/biblia#Comentario`.
`http://biblia-rdf.com/biblia#verseRange`	Propriedade que representa um intervalo de versículos. Criada para ser usada na classe `http://biblia-rdf.com/biblia#Comentario`.

Quadro 41 - Recursos descritos no fornecedor biblia-rdf.com
Fonte: Autoria própria

Como pode ser observado, o modelo de dados fornecido por biblia-rdf.com contém várias classes e propriedades para representar os livros, versículos e possíveis comentários da bíblia. Uma atenção especial é dada na classe chamada http://biblia-rdf.com/biblia#Comentario, que permite outras bases de dados realizar comentários em seus versículos. Os fornecedores **comentarios-scofield.com** e **comentarios-dlmoddy.com** fazem uso dessa classe para fornecer seus comentários. Nada impede de qualquer outra aplicação criar seus próprios comentários bíblicos no formato RDF utilizando as classes do modelo proposto por **biblia-rdf.com**. Dessa maneira, a aplicação pode fazer consulta em vários fornecedores de comentários bíblicos, bastando apenas estes seguirem o modelo proposto por **biblia-rdf.com**.

A Figura 11 apresenta um exemplo de um versículo descrito nessa base.

```
1  <bible:Versiculo rdf:about="http://biblia-rdf.com/biblia#Ism-26-5">
2  <bible:livro rdf:resource="http://biblia-rdf.com/biblia#I_SAMUEL" />
3  <bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">26</bible:ncap>
4  <bible:nvers rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5</bible:nvers>
5  <bible:texto>E Davi se levantou, e foi ao lugar onde Saul se tinha acampado;
   viu Davi o lugar onde se tinha deitado Saul, e Abner, filho de Ner, capitão do
   seu exército; e Saul estava deitado dentro do lugar dos carros, e o povo estava
   acampado ao redor dele.</bible:texto>
6  </bible:Versiculo>
```

Figura 11 - Exemplo de versículo descrito na base biblia-rdf.com
Fonte: Autoria própria

Como pode ser observado, as declarações iniciais do XML e dos *namespaces* foram omitidas para simplificação. A seguir é apresentada uma descrição de cada linha.

- **Linha 1:** Inicia-se a declaração do elemento `bible:Versiculo` que representa um versículo.
- **Linha 2**: Declaração da propriedade **bible:livro**. Esta propriedade está apontando para o URI que representa o livro da bíblia chamado 1ª Samuel. Isso indica que o versículo se encontra nesse livro.
- **Linha 3:** Declaração da propriedade **bible:ncap**. Esta propriedade contém como valor o número do capítulo referente ao versículo.
- **Linha 4:** Declaração da propriedade **bible:nvers**. Esta propriedade contém como valor o número do versículo a qual se refere.
- **Linha 5:** Declaração da propriedade **bible:texto**. Esta propriedade contém como valor o conteúdo do versículo.

4.1.2 Fornecedores de Comentários Bíblicos

Tanto o fornecedor **comentarios-scofield.com** quanto o fornecedor **comentarios-dlmoddy.com**, utilizam a classe `http://biblia-rdf.com/biblia#Comentario` disponibilizada pelo fornecedor **biblia-rdf.com** para criar comentários relacionados a versículos específicos da bíblia. Um comentário é gerado interligando o comentário com o livro, capítulo e intervalo de versículos que se deseja comentar. A Figura 12 apresenta um exemplo de comentário.

```
1  <bible:Comentario rdf:about="
   http://comentarios-scofield/comentarios#joao-1-1">
2      <bible:livro rdf:resource="http://biblia-rdf.com/biblia#JOAO" />
3      <bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1
       </bible:ncap>
4      <bible:verseRange>1</bible:verseRange>
5      <bible:texto>Gr. Logos (aram. Memra, uma designação de Deus nos
       targuns, i.e., as traduções aramaicas do AT). A palavra grega
       significa: 1) um pensamento ou conceito e 2) a expressão ou declaração
       desse pensamento. Portanto, como designação de Cristo, Logos é
       particularmente apropriado porque 1) nele estão encarnados todos os
       tesouros da divida sabedoria, a onisciência de Deus (1Co 1.24; Ef
       3.10,11; Cl 2.2,3) e 2) ele é, desde a eternidade, mas especialmente
       em sua encarnação, a expressão da Pessoa e do pensamento da Divindade
       (Jo 1.3-5,9,14-18; 14.9-11; Cl 2.9). A Divindade é expressa na
       essência, na Pesoa e na obra de Cristo.</bible:texto>
6      <rdfs:isDefinedBy>http://comentarios-scofield</rdfs:isDefinedBy>
7  </bible:Comentario>
```

Figura 12 - Exemplo de comentário descrito na base comentarios-scofield.com
Fonte: Autoria própria

A seguir é apresentada uma descrição de cada linha.

- **Linha 1:** Inicia-se a declaração do elemento `bible:Comentario` que representa um comentário bíblico.
- **Linha 2:** Declaração da propriedade `bible:livro`. Esta propriedade está apontando para o URI que representa o livro da bíblia chamado João. Isso indica que o comentário diz respeito a um versículo que se encontra nesse livro.
- **Linha 3:** Declaração da propriedade `bible:ncap`. Esta propriedade contém como valor o número do capítulo referente ao comentário.
- **Linha 4:** Declaração da propriedade `bible:verseRange`. Esta propriedade contém como valor um intervalo de versículo, referente ao comentário, por

exemplo: o comentário poderia ser referente ao versículo um até o versículo cinco desse mesmo capitulo, nesse caso o valor dessa propriedade seria “1-5”.

- **Linha 5:** Declaração da propriedade `bible:texto`. Esta propriedade contém como valor o comentário em si.
- **Linha 6:** Declaração da propriedade `rdfs:isDefinedBy`. Esta propriedade é utilizada para indicar quem está definindo o comentário em si.

4.1.3 Inferência de Comentários Bíblicos

Para demonstrar o uso de inferência na aplicação, será utilizado a característica de similaridade de alguns livros da Bíblia. Os livros Mateus, Marcos e Lucas contam basicamente a mesma história, sendo muito similares entre si. Um exemplo de similaridade pode ser observado analisando os versículos `Mateus 5.13` e `Lucas 14.34-35`, como apresentado no Quadro 42.

Mateus 5.13	**Lucas 14.34-35**
Vós sois o sal da terra; e se o sal for insípido, com que se há de salgar? Para nada mais presta senão para se lançar fora, e ser pisado pelos homens.	Bom é o sal; mas, se o sal degenerar, com que se há de salgar? Nem presta para a terra, nem para o monturo; lançam-no fora. Quem tem ouvidos para ouvir, ouça.

Quadro 42 – Exemplo de similaridade entre versículos
Fonte: Autoria própria

Como pode ser observado no Quadro 42, ambos os versículos descrevem um mesmo ensinamento. Com base nesse fato, pode-se dizer que se `Mateus 5.13` possui um comentário, esse mesmo comentário se torna válido para `Lucas 14.34-35`.

Conhecendo previamente quais versículos descrevem um mesmo fato ou ensinamento, foi criado três regras (ANEXO D) para o motor de inferência do Jena deduzir alguns versículos distintos que podem conter um mesmo comentário. Quando a aplicação carrega os arquivos RDF, também infere estas regras no modelo. As regras são as seguintes:

1. Se existir algum comentário para o versículo `Mateus 5.13`, este será inferido para o versículo `Lucas 14.34-35`.
2. Se existir algum comentário para o versículo `Mateus 6.21`, este será inferido para o versículo `Lucas 12.34`.

3. Se existir algum comentário para o versículo `Mateus 7.1`, este será inferido para o versículo `Lucas 6.37`.

Para simplificação, foi definida apenas estas três regras, porém existem muitos outros versículos similares na Bíblia, os quais poderiam ser criadas uma regra para cada caso.

4.1.4 Diagrama de Componentes da Aplicação Bíblia App

A Figura 13 apresenta um diagrama de componentes representando a ligação entre os fornecedores RDF e a aplicação Bíblia App.

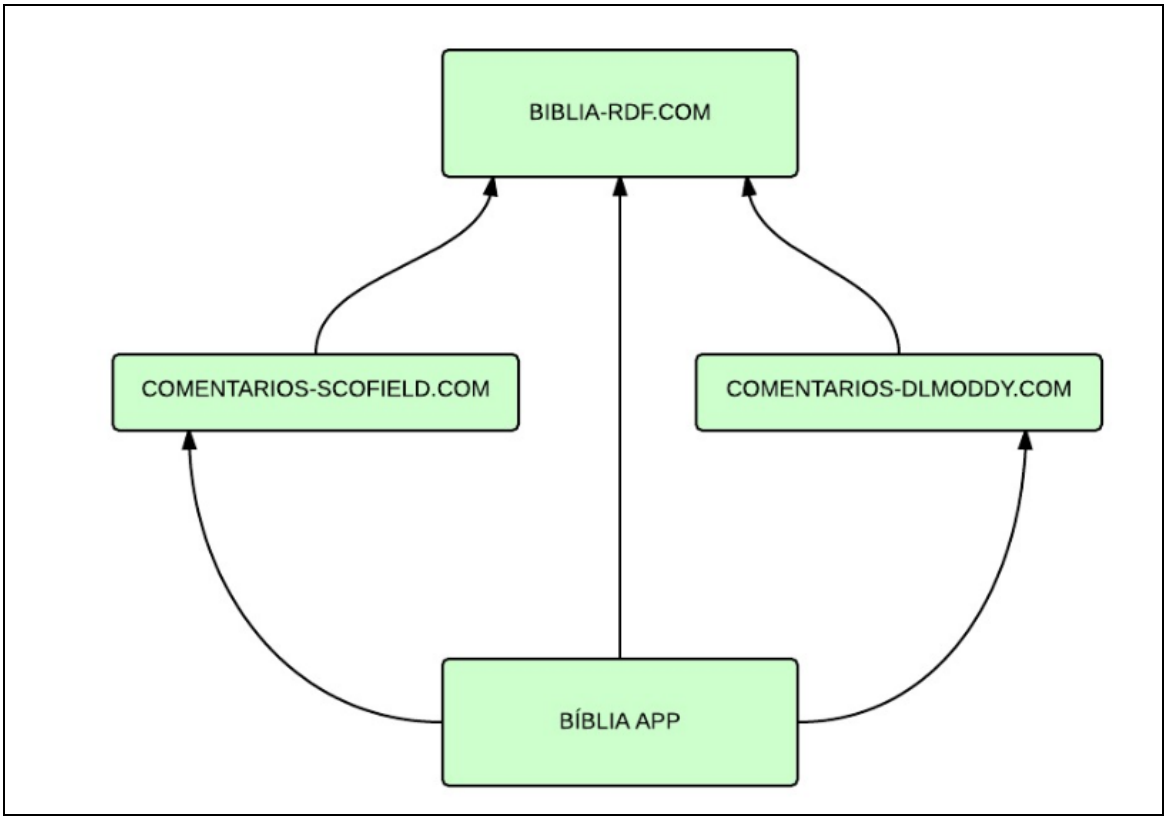

Figura 13 - Diagrama de componentes do relacionamento entre a aplicação e os fornecedores RDF
Fonte: Autoria própria

4.1.5 Acessando os Dados RDF dos Fornecedores

Para simular os dados RDF dos três fornecedores foi criado três arquivos no formato RDF/XML. São eles:

- **biblia.rdf:** Arquivo que simula a base do fornecedor biblia-rdf.com (pode ser encontrado no anexo A deste trabalho).
- **comentarios-scofield.rdf:** Arquivo que simula a base do fornecedor comentarios-scofield.com (pode ser encontrado no ANEXO B deste trabalho).
- **comentarios-dlmoddy.rdf:** Arquivo que simula a base do fornecedor comentários-dlmoddy.com (pode ser encontrado no ANEXO C deste trabalho).

A Figura 14 apresenta a classe `ModelFactory`, responsável por instanciar um modelo fazendo a união dos dados dos três fornecedores. Para acessar esse modelo, as outras classes da aplicação farão uso do método estático `ModelFactory.get`.

```
10 public class ModelFactory {
11     private static Model model;
12     public static Model get() throws Exception {
13         if (model == null) {
14             try(
15                 InputStream in1 = FileManager.get().open("C:/biblia.rdf");
16                 InputStream in2 = FileManager.get().open("C:/comentarios-scofield.rdf");
17                 InputStream in3 = FileManager.get().open("C:/comentarios-dlmoddy.rdf")) {
18 
19                 model = com.hp.hpl.jena.rdf.model.ModelFactory.createDefaultModel().read(in1, null);
20                 model = model.union(com.hp.hpl.jena.rdf.model.ModelFactory.createDefaultModel().read(in2, null));
21                 model = model.union(com.hp.hpl.jena.rdf.model.ModelFactory.createDefaultModel().read(in3, null));
22             }
23         }
24         return model;
25     }
26 }
```

Figura 14 - Classe *ModelFactory*
Fonte: Autoria própria

4.1.6 Buscando Versículos da Bíblia

Para apresentação dos versículos da bíblia, a aplicação realiza consultas SPARQL no modelo já instanciado. A aplicação apresenta para o usuário o conteúdo de um capítulo de determinado livro por vez. A Figura 15 apresenta o método responsável por buscar os versículos de determinado capítulo da bíblia.

```
public List<Versiculo> buscarCapitulo(Livro livro, int ncap) throws Exception {
    String queryString =
              "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
            + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
            + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
            + "PREFIX bible: <http://biblia-rdf.com/biblia#>"

            + " SELECT ?livro ?ncap ?nvers ?texto WHERE {"
            + "   ?a rdf:type <http://biblia-rdf.com/biblia#Versiculo> ."
            + "   ?a bible:livro <http://biblia-rdf.com/biblia#" + livro.name() + "> ."
            + "   ?a bible:ncap ?ncap ."
            + "   ?a bible:nvers ?nvers ."
            + "   ?a bible:texto ?texto ."
            + "   FILTER (?ncap = " + ncap + ") ."
            + " } ORDER BY  ?nvers";

    Query query = QueryFactory.create(queryString);
    // Cria o objeto QueryExecution
    QueryExecution qexec = QueryExecutionFactory.create(query, ModelFactory.get());
    List<Versiculo> list = new ArrayList<>();
    try {
        ResultSet results = qexec.execSelect();
        for (; results.hasNext();) {
            //Processa o resultado aqui
            QuerySolution q = results.next();
            Versiculo versiculo = new Versiculo();
            versiculo.setLivro(livro);
            versiculo.setCap(ncap);
            versiculo.setNvers(q.get("nvers").asLiteral().getInt());
            versiculo.setTexto(q.get("texto").toString());
            list.add(versiculo);
        }
    } finally {
        qexec.close();
    }
    return list;
}
```

Figura 15 - Método responsável por buscar versículos de determinado capítulo da bíblia
Fonte: Autoria própria

4.1.7 Buscando Comentários Bíblicos

Como já mencionado, a aplicação Bíblia App inclui em seu modelo RDF os dados de dois fornecedores de comentários. Esses dados são carregados no modelo Jena pela classe `ModelFactory`. A Figura 16 apresenta o método responsável por buscar todos os comentários de um versículo específico da bíblia.

```
public List<Comentario> buscarComentarios(Livro livro, int ncap, int nvers) throws Exception {
    String queryString =
              "PREFIX foaf: <http://xmlns.com/foaf/0.1/> "
            + "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>"
            + "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"
            + "PREFIX bible: <http://biblia-rdf.com/biblia#>"
            + "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>"
            + " SELECT ?texto ?verseRange ?definidoPor WHERE {"
            + "   ?a rdf:type <http://biblia-rdf.com/biblia#Comentario> ."
            + "   ?a bible:livro <http://biblia-rdf.com/biblia#" + livro.name() + "> ."
            + "   ?a bible:ncap ?ncap ."
            + "   ?a bible:verseRange ?verseRange ."
            + "   ?a bible:texto ?texto ."
            + "   OPTIONAL { ?a rdfs:isDefinedBy ?definidoPor } ."
            + "   bind( xsd:decimal(strbefore((?verseRange), \"-\" )) as ?vi) ."
            + "   bind( xsd:decimal(strafter((?verseRange), \"-\" )) as ?vf) ."
            + "   FILTER (?ncap = " + ncap + " && (xsd:decimal(?verseRange) = " + nvers
            + "           || (?vi = " + nvers + " || ?vf = " + nvers + " || (?vi < "
            +               nvers + " && ?vf > " + nvers + "))))  "
            + " }";

    Query query = QueryFactory.create(queryString);
    // Cria o objeto QueryExecution
    QueryExecution qexec = QueryExecutionFactory.create(query, ModelFactory.get());
    List<Comentario> list = new ArrayList<>();
    try {
        ResultSet results = qexec.execSelect();
        while (results.hasNext()) {
            //Processa o resultado aqui
            QuerySolution q = results.next();
            Comentario comentario = new Comentario();
            comentario.setLivro(livro);
            comentario.setCap(ncap);
            comentario.setVerseRange(q.get("verseRange").toString());
            comentario.setTexto(q.get("texto").toString());
            comentario.setDefinidoPor(q.get("definidoPor").toString());
            list.add(comentario);
        }
    } finally {
        qexec.close();
    }
    return list;
```

Figura 16 - Método responsável por buscar os comentários relacionados a determinado versículo da bíblia
Fonte: Autoria própria

4.1.8 Utilização da Aplicação

O *layout* da aplicação Bíblia App consiste em um painel à esquerda, para navegação e leitura de capítulos de determinado livro da bíblia e um painel à direita, para leitura de comentários de determinado versículo da bíblia. A aplicação foi desenvolvida para que, quando o usuário selecionar determinado versículo, o sistema busque os comentários relacionados a esse versículo apresentando-os no painel à direita.

A Figura 17 apresenta a aplicação no momento em que o usuário selecionou o versículo primeiro referente ao capítulo seis do livro de Mateus.

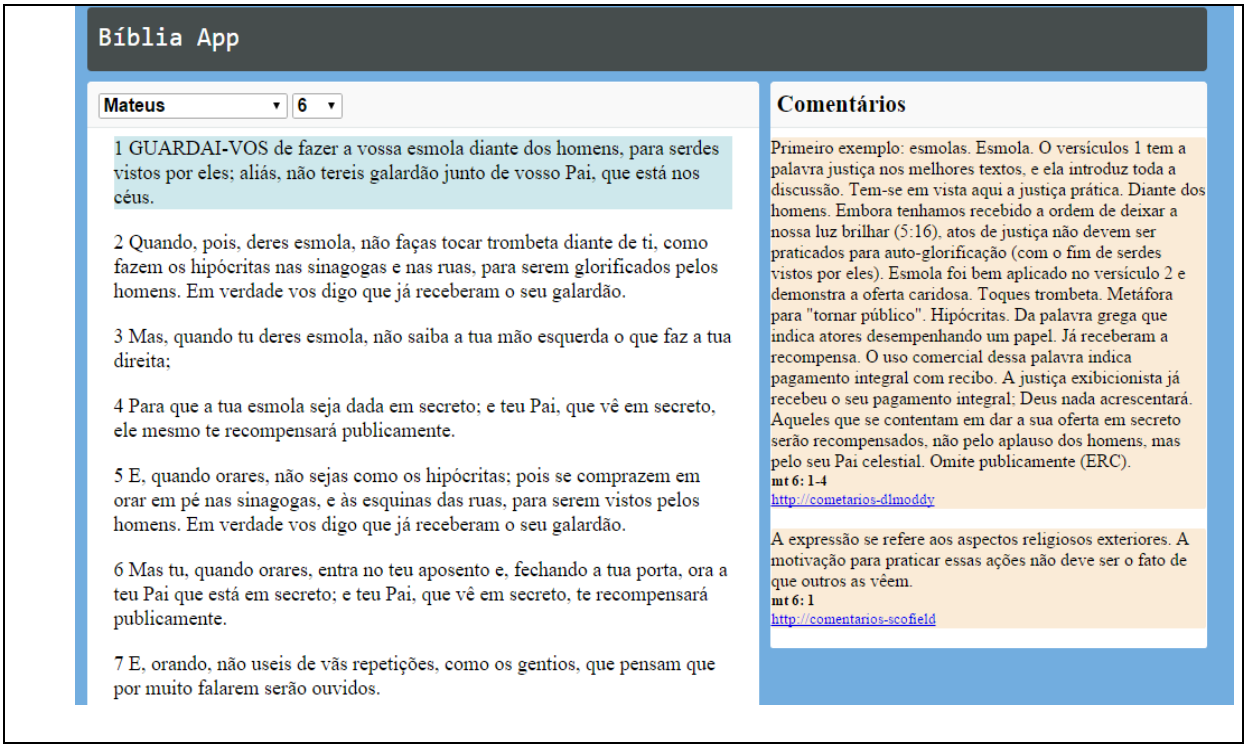

Figura 17 - *Print* da aplicação
Fonte: Autoria própria

Como pode ser observado, o painel à direita contém dois comentários para o versículo selecionado no painel à esquerda. Cada comentário é referente a um provedor de comentários diferente.

4.1.9 Visualizando Comentários Inferidos

A Figura 18 apresenta um exemplo de um versículo inferido. Como pode ser observado no ANEXOS B e C, não existe comentário para o versículo `Lucas 6.37`, no entanto este versículo foi inferido pela regra descrita no ANEXO D.

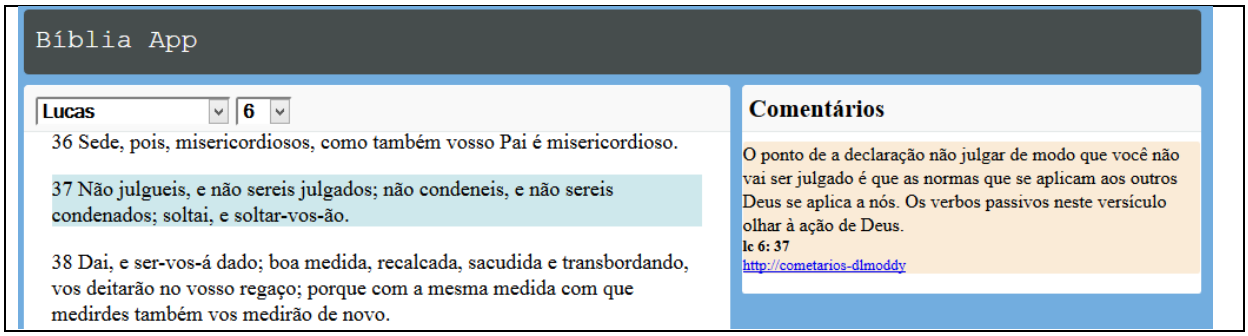

Figura 18 – Exemplo de versículo inferido.
Fonte: Autoria própria

Vale ressaltar que a aplicação Bíblia App é apenas um exemplo básico para demonstração de uma aplicação que faz uso dos padrões do *Linked Data*. A ideia principal dessa aplicação é demonstrar a interligação entre fornecedores RDF distintos, e como essa interligação pode ser facilmente consumida por outra aplicação que já conheça os padrões *Linked Data*, e também como pode ser aplicada regras de inferência nos arquivos RDF.

4.2 AVALIAÇÃO DOS RESULTADOS

Ao longo desenvolvimento da aplicação, ficou claro alguns benefícios em relação a aplicação fazer uso dos padrões *Linked Data*. Para realizar as consultas, a aplicação não precisa conhecer o banco de dados ou algum protocolo específico de seus fornecedores, basta apenas saber como conseguir os dados em formato RDF. Qualquer fornecedor de comentários bíblicos pode facilmente exportar seus dados para o formato RDF e a aplicação só precisa saber como consegui-los para incrementar sua base de comentários. Com o uso de regras de inferência, a aplicação pode enriquecer sua base de comentários, mapeando quais versículos descrevem um mesmo fato, e deduzindo quais comentários podem ser reaproveitados para outro versículo.

5 CONSIDERAÇÕES FINAIS

Ao longo do desenvolvimento desse trabalho, ficou claro que a Web semântica é o futuro da Web. Com a aplicação difundida dos padrões *Linked Data*, a experiência dos usuários da Web será cada vez mais aperfeiçoada. Agentes de *software* poderão compreender o conteúdo da Web com muito mais precisão, criando assim uma variedade de possibilidades para os desenvolvedores de aplicações Web. Com a utilização do modelo RDF, servidores Web poderão publicar seu próprio domínio de informações de forma semântica, e também poderão se interligar como outras fontes de dados RDF, criando assim uma Web de dados ligados.

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE JENA. **Getting stated with Apache Jena**. Disponível em: <https://jena.apache.org/getting_started/index.html>. Acesso em: 17 mai. 2015.

BRICKLEY, Dan; GUHA, R.V. **RDF Schema 1.1**. Disponível em <http://www.w3.org/TR/rdf-schema/>. Acesso em: 12 mai. 2015.

COSTA A., YAMATE F. **Semantic Lattes: uma ferramenta de consulta baseada em ontologias**. São Paulo: Escola Politécnica da Universidade de São Paulo, 2009.

HEATH, Tom; BIZER, Christian. ***Linked Data*: Envolving the Web into a Global Data Space**. Morgan & Claypool Publishers, 2011.

MICROFORMATS. **Welcome to the microformats wiki! Microformats Wiki**. Disponível em: <http://microformats.org/wiki/Main_Page >. Acesso em: 09 mai. 2015.

PRUD'HOMMEAUX, Eric; SEABORNE, Andy. **SPARQL Query Language for RDF**. Disponível em <http://www.w3.org/TR/rdf-sparql-query/>. Acesso em: 16 mai. 2015.

WIKIPÉDIA. **Web Semântica – Wikipédia, a enciclopédia livre.** Disponível em: < http://pt.wikipedia.org/wiki/Web_sem%C3%A2ntica>. Acesso em: 09 mai. 2015.

ANEXOS

ANEXO A – INICION DO ARQUIVO RDF BIBLIA.RDF

```
#biblia.rdf#########################################################################
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:bible="http://biblia-rdf.com/biblia#"
xml:base="http://biblia-rdf.com/biblia#">

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#LivroBiblia">
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#Versiculo">
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#Comentario">
</rdfs:Class>

<rdf:Property rdf:ID="livro">
<rdfs:label>Livro</rdfs:label>
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Versiculo"/>
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Comentario"/>
<rdfs:range rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia"/>
</rdf:Property>

<rdf:Property rdf:ID="ncap">
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Versiculo"/>
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Comentario"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Property>

<rdf:Property rdf:ID="nvers">
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Versiculo"/>
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Comentario"/>
<rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
</rdf:Property>

<rdf:Property rdf:ID="texto">
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Versiculo"/>
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Comentario"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>

<rdf:Property rdf:ID="verseRange">
<rdfs:domain rdf:resource="http://biblia-rdf.com/biblia#Comentario"/>
<rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/>
</rdf:Property>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#GENESIS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#EXODO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#LEVITICO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#NUMEROS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>
```

```
<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#DEUTERONOMIO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JOSUE">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JUIZES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#RUTE">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_SAMUEL">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_SAMUEL">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_REIS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_REIS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_CRONICAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_CRONICAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ESDRAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#NEEMIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ESTER">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#SALMOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#PROVERBIOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ECLESIASTES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>
```

```
<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#CANTARES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ISAIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JEREMIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#LAMENTACOES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#EZEQUIEL">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#DANIEL">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#OSEIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JOEL">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#AMOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#OBADIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JONAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#MIQUEIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#NAUM">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#HABACUQUE">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#SOFONIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#AGEU">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ZACARIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>
```

```
<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#MALAQUIAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#MATEUS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#MARCOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#LUCAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JOAO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ATOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#ROMANOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_CORINTIOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_CORINTIOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#GALATAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#EFESIOS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#FILIPENSES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#COLOSSENSES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_TESSALONICENSES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_TESSALONICENSES">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_TIMOTEO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_TIMOTEO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>
```

```
<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#TITO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#FILEMON">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#HEBREUS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#TIAGO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_PEDRO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_PEDRO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#I_JOAO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#II_JOAO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#III_JOAO">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#JUDAS">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<rdfs:Class rdf:about="http://biblia-rdf.com/biblia#APOCALIPSE">
<rdfs:subClassOf rdf:resource="http://biblia-rdf.com/biblia#LivroBiblia" />
</rdfs:Class>

<bible:Versiculo rdf:ID="gn-1-1">
<bible:livro rdf:resource="http://biblia-rdf.com/biblia#GENESIS" />
<bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bible:ncap>
<bible:nvers rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bible:nvers>
<bible:texto>NO princípio criou Deus os céus e a terra.</bible:texto>
</bible:Versiculo>

<bible:Versiculo rdf:ID="gn-1-2">
<bible:livro rdf:resource="http://biblia-rdf.com/biblia#GENESIS" />
<bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bible:ncap>
<bible:nvers rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2</bible:nvers>
<bible:texto>E a terra era sem forma e vazia; e havia trevas sobre a face do
abismo; e o Espírito de Deus se movia sobre a face das águas.</bible:texto>
</bible:Versiculo>
##### Restante dos versículos
```

ANEXO B – ARQUIVO COMETARIOS-SCOFIELD.RDF

```
#comentarios-scofield.rdf###########################################################
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:bible="http://biblia-rdf.com/biblia#"
xml:base="http://comentarios-scofield/comentarios#">

<bible:Comentario rdf:ID="gn-1-1">
<bible:livro rdf:resource="http://biblia-rdf.com/biblia#GENESIS" />
<bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bible:ncap>
<bible:verseRange>1</bible:verseRange>
<bible:texto>A Bíblia começa com Deus, não com argumentos filosóficos sobre usa
existência.</bible:texto>
<rdfs:isDefinedBy>http://comentarios-scofield</rdfs:isDefinedBy>
</bible:Comentario>

<bible:Comentario rdf:ID="mat-6-1">
<bible:livro rdf:resource="http://biblia-rdf.com/biblia#MATEUS" />
<bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</bible:ncap>
<bible:verseRange>1</bible:verseRange>
<bible:texto>A expressão se refere aos aspectos religiosos exteriores. A motivação
para praticar essas ações não deve ser o fato de que outros as vêem.</bible:texto>
<rdfs:isDefinedBy>http://comentarios-scofield</rdfs:isDefinedBy>
</bible:Comentario>

<bible:Comentario rdf:ID="joao-1-1">
<bible:livro rdf:resource="http://biblia-rdf.com/biblia#JOAO" />
<bible:ncap rdf:datatype="http://www.w3.org/2001/XMLSchema#int">1</bible:ncap>
<bible:verseRange>1</bible:verseRange>
<bible:texto>Gr. Logos (aram. Memra, uma designação de Deus nos targuns, i.e., as
traduções aramaicas do AT). A palavra grega significa: 1) um pensamento ou conceito
e 2) a expressão ou declaração desse pensamento. Portanto, como designação de
Cristo, Logos é particularmente apropriado porque 1) nele estão encarnados todos os
tesouros da divida sabedoria, a onisciência de Deus (1Co 1.24; Ef 3.10,11; Cl
2.2,3) e 2) ele é, desde a eternidade, mas especialmente em sua encarnação, a
expressão da Pessoa e do pensamento da Divindade (Jo 1.3-5,9,14-18; 14.9-11; Cl
2.9). A Divindade é expressa na essência, na Pesoa e na obra de
Cristo.</bible:texto>
<rdfs:isDefinedBy>http://comentarios-scofield</rdfs:isDefinedBy>
</bible:Comentario>
</rdf:RDF>
```

ANEXO C – ARQUIVO COMETARIOS-DLMODDY.RDF

```
#comentarios-dlmoddy.rdf#############################################################
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:bible="http://biblia-rdf.com/biblia#"
xml:base="http://cometarios-dlmoddy/comentarios#">

<bible:Comentario rdf:ID="mat-6-1-4">
      <bible:livro rdf:resource="http://biblia-rdf.com/biblia#MATEUS" />
      <bible:ncap
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</bible:ncap>
      <bible:verseRange>1-4</bible:verseRange>
      <bible:texto>Primeiro exemplo: esmolas. Esmola. O versículos 1 tem a palavra
justiça nos melhores textos, e ela introduz toda a discussão. Tem-se em vista aqui
a justiça prática. Diante dos homens. Embora tenhamos recebido a ordem de deixar a
nossa luz brilhar (5:16), atos de justiça não devem ser praticados para auto-
glorificação (com o fim de serdes vistos por eles). Esmola foi bem aplicado no
versículo 2 e demonstra a oferta caridosa. Toques trombeta. Metáfora para "tornar
público". Hipócritas. Da palavra grega que indica atores desempenhando um papel. Já
receberam a recompensa. O uso comercial dessa palavra indica pagamento integral com
recibo. A justiça exibicionista já recebeu o seu pagamento integral; Deus nada
acrescentará. Aqueles que se contentam em dar a sua oferta em secreto serão
recompensados, não pelo aplauso dos homens, mas pelo seu Pai celestial. Omite
publicamente (ERC).</bible:texto>
      <rdfs:isDefinedBy>http://cometarios-dlmoddy</rdfs:isDefinedBy>
</bible:Comentario>

<bible:Comentario rdf:ID="mat-5-13">
      <bible:livro rdf:resource="http://biblia-rdf.com/biblia#MATEUS" />
      <bible:ncap
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">5</bible:ncap>
      <bible:verseRange>13</bible:verseRange>
      <bible:texto>O sal foi usado como tempero ou fertilizantes ( bdag 41 sv ἅλας
a), ou como um preservativo. Se o sal deixou de ser útil, ele foi jogado fora. Com
esta ilustração Jesus advertiu sobre um discípulo que deixaram de segui-
lo.</bible:texto>
      <rdfs:isDefinedBy>http://cometarios-dlmoddy</rdfs:isDefinedBy>
</bible:Comentario>

<bible:Comentario rdf:ID="mat-6-21">
      <bible:livro rdf:resource="http://biblia-rdf.com/biblia#MATEUS" />
      <bible:ncap
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">6</bible:ncap>
      <bible:verseRange>21</bible:verseRange>
      <bible:texto>Buscando tesouro celeste significa servir aos outros e honrar a
Deus por fazê-lo.</bible:texto>
      <rdfs:isDefinedBy>http://cometarios-dlmoddy</rdfs:isDefinedBy>
</bible:Comentario>

<bible:Comentario rdf:ID="mat-7-1">
      <bible:livro rdf:resource="http://biblia-rdf.com/biblia#MATEUS" />
      <bible:ncap
rdf:datatype="http://www.w3.org/2001/XMLSchema#int">7</bible:ncap>
      <bible:verseRange>1</bible:verseRange>
      <bible:texto>O ponto de a declaração não julgar de modo que você não vai ser
julgado é que as normas que se aplicam aos outros Deus se aplica a nós. Os verbos
passivos neste versículo olhar à ação de Deus.</bible:texto>
      <rdfs:isDefinedBy>http://cometarios-dlmoddy</rdfs:isDefinedBy>
</bible:Comentario>
</rdf:RDF>
```

ANEXO D – ARQUIVO REGRAS.TXT

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix bible: <http://biblia-rdf.com/biblia#>.

#Se encontrar comentários em Mateus 5.13, infere os mesmos comentários em Lucas
14.34-35
[mateus-5-13:
( ?coment rdf:type bible:Comentario),
( ?coment bible:livro bible:MATEUS),
( ?coment bible:ncap ?ncap),
( ?coment bible:verseRange ?verse),
( ?coment bible:texto ?texto)
( ?coment rdfs:isDefinedBy ?isDefinedBy)
equal(?ncap, 5),
equal(?verse, "13"),
->
( bible:mateus-5-13-inf rdf:type bible:Comentario ),
( bible:mateus-5-13-inf bible:livro bible:LUCAS ),
( bible:mateus-5-13-inf bible:ncap 14),
( bible:mateus-5-13-inf bible:verseRange "34-35"),
( bible:mateus-5-13-inf bible:texto ?texto),
( bible:mateus-5-13-inf rdfs:isDefinedBy ?isDefinedBy)
]

#Se encontrar comentários em Mateus 6.21, infere os mesmos comentários em Lucas
12.34
[mateus-6-21:
( ?coment rdf:type bible:Comentario),
( ?coment bible:livro bible:MATEUS),
( ?coment bible:ncap ?ncap),
( ?coment bible:verseRange ?verse),
( ?coment bible:texto ?texto)
( ?coment rdfs:isDefinedBy ?isDefinedBy)
equal(?ncap, 6),
equal(?verse, "21"),
->
( bible:mateus-6-1-inf rdf:type bible:Comentario ),
( bible:mateus-6-1-inf bible:livro bible:LUCAS ),
( bible:mateus-6-1-inf bible:ncap 12),
( bible:mateus-6-1-inf bible:verseRange "34"),
( bible:mateus-6-1inf bible:texto ?texto),
( bible:mateus-6-1-inf rdfs:isDefinedBy ?isDefinedBy)
]

#Se encontrar comentários em Mateus 7.1, infere os mesmos comentários em Lucas 6.37
[mateus-7-1:
( ?coment rdf:type bible:Comentario),
( ?coment bible:livro bible:MATEUS),
( ?coment bible:ncap ?ncap),
( ?coment bible:verseRange ?verse),
( ?coment bible:texto ?texto)
( ?coment rdfs:isDefinedBy ?isDefinedBy)
equal(?ncap, 7),
equal(?verse, "1"),
->
( bible:mateus-7-1-inf rdf:type bible:Comentario ),
( bible:mateus-7-1-inf bible:livro bible:LUCAS ),
( bible:mateus-7-1-inf bible:ncap 6),
( bible:mateus-7-1-inf bible:verseRange "37"),
( bible:mateus-7-1-inf bible:texto ?texto),
( bible:mateus-7-1-inf rdfs:isDefinedBy ?isDefinedBy)
    ]
```