UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Ronaldo Amaral Santos

ANOTAÇÃO SEMÂNTICA DE CONTEÚDO NO APOIO A PRODUÇÃO DE OBJETOS DE APRENDIZAGEM. UM ESTUDO DE CASO NO AMBIENTE MOODLE.

CAMPOS DOS GOYTACAZES, RJ
Novembro de 2014.

UNIVERSIDADE CANDIDO MENDES - UCAM
PROGRAMA DE PÓS-GRADUAÇÃO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL
CURSO DE MESTRADO EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL

Ronaldo Amaral Santos

ANOTAÇÃO SEMÂNTICA DE CONTEÚDO NO APOIO A PRODUÇÃO DE OBJETOS DE APRENDIZAGEM. UM ESTUDO DE CASO NO AMBIENTE MOODLE.

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional da Universidade Candido Mendes – Campos/RJ para obtenção do grau de MESTRE EM PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Orientadora: Prof.ª Georgia Regina Rodrigues Gomes, D.Sc.
Co-orientador: Prof. Mark Douglas de Azevedo Jacynho, D.Sc.

CAMPOS DOS GOYTACAZES, RJ
Novembro de 2014.

RONALDO AMARAL SANTOS

ANOTAÇÃO SEMÂNTICA DE CONTEÚDO NO APOIO A PRODUÇÃO DE OBJETOS DE APRENDIZAGEM. UM ESTUDO DE CASO NO AMBIENTE MOODLE.

Dissertação apresentada ao Programa de Pós-Graduação em Pesquisa Operacional e Inteligência Computacional da Universidade Candido Mendes – Campos/RJ para obtenção do grau de Mestre em PESQUISA OPERACIONAL E INTELIGÊNCIA COMPUTACIONAL.

Aprovado em 19 de dezembro de 2014

BANCA EXAMINADORA:

_____

Prof.ª Georgia Regina Rodrigues Gomes, D.Sc.- Orientadora
Universidade Candido Mendes
Universidade Federal Fluminense

_____

Prof. Mark Douglas de Azevedo Jacyntho , D.Sc.
Universidade Candido Mendes

_____

Prof. Ítalo de Oliveira Matias, D.Sc.
Universidade Candido Mendes

_____

Prof. Breno Fabrício Terra Azevedo, D.Sc.
Instituto Federal Fluminense

CAMPOS DOS GOYTACAZES, RJ
2014

AGRADECIMENTOS

Agradeço a Deus, em primeiro lugar, por todas as oportunidades que tive na vida, dentre as quais, a oportunidade de continuar os estudos.

Agradeço aos meus pais Donaldo e Fátima pela compreensão nas ausências e por todo o incentivo que sempre me deram de continuar estudando.

Agradeço a minha esposa Danusa por todo o incentivo desde o começo do mestrado, pelas palavras de carinho, paciência e compreensão em todos os momentos, principalmente nos momentos em que tivemos que abdicar de passeios, lazer e até mesmo de visitas aos familiares.

Agradeço aos meus orientadores Georgia e Mark Douglas pela disponibilidade, confiança e por me guiarem nesta pesquisa de forma que eu conseguisse chegar até o final.

Agradeço aos meus colegas de turma pela troca de conhecimentos e pela amizade durante o Mestrado.

Agradeço ao Instituto Federal Fluminense por incentivar e custear essa capacitação, tão importante na nossa região.

Agradeço aos professores da Candido Mendes pela disponibilidade e pelo convívio amistoso durante esses três anos.

RESUMO

ANOTAÇÃO SEMÂNTICA DE CONTEÚDO NO APOIO A PRODUÇÃO DE OBJETOS DE APRENDIZAGEM. UM ESTUDO DE CASO NO AMBIENTE MOODLE.

Na Educação à distância – EAD, um dos elementos vitais é a disponibilização de materiais de aprendizagem relevantes e de valor. Nos últimos anos, muitas aplicações educacionais baseadas na *Web* têm sido desenvolvidas, mas alguns desafios ainda existem, dentre os quais a pesquisa por materiais e objetos de aprendizagem mais inteligentes e eficientes. Para tal, pode-se utilizar-se dos conceitos introduzidos com a *Web* Semântica, onde metadados estruturados inteligíveis por máquina são adicionados à *Web* de forma a agregar significado à informação. A agregação de valor semântico aos documentos é uma forma de organizar o processo de publicação, recuperação e enriquecimento da informação, haja vista que possibilita a recuperação da informação contida nos documentos de forma precisa e eficaz, facilitando sua reutilização. Portanto, o objetivo deste trabalho é propor uma arquitetura orientada a serviço para anotação semântica de conteúdo, que automatiza a extração de conceitos em objetos de aprendizagem, utilizando técnicas de mineração de texto. Foi desenvolvido um protótipo para demonstrar o funcionamento da arquitetura e realizado um estudo de caso aplicado ao ambiente de aprendizagem Moodle. Com este trabalho, espera-se construir novas formas de descoberta de conhecimento e reuso de informação, a partir de dados disponíveis em ambientes de aprendizagem, por meio da descrição estruturada em RDF dos objetos de aprendizagem. Além disso, interligar conceitos a fonte de dados na *Web* de Dados (*Web of Linked Data*), enriquecendo a base de conhecimento do ambiente de aprendizagem com *mashup* semântico entre estes conceitos e recursos pré-existentes da *Web* de dados, possibilitando, um aumento da produtividade no processo de ensino-aprendizagem.

PALAVRAS-CHAVE: Anotação semântica, Ensino a distância (EAD), Objeto de aprendizagem, Linked data.

ABSTRACT

SEMANTIC ANNOTATION CONTENT IN SUPPORTING THE PRODUCTION OF LEARNING OBJECTS. A CASE STUDY IN THE ENVIRONMENT MOODLE.

In Distance Education - Distance Learning, one of the vital elements is the provision of relevant learning materials and value. In recent years, many educational *Web*-based applications have been developed, but there are still some challenges, among which the search for material objects more intelligent and efficient learning. For this, one can utilize the concepts introduced with the Semantic *Web*, where machine-understandable structured metadata are added to the *Web* in order to add meaning to the information. The aggregation of semantic value to documents is a way to organize the publishing process, recovery and enrichment of information, considering that enables the retrieval of information contained in the documents so accurate and effective, facilitating their reuse. Therefore, the aim of this work is to propose a service oriented architecture for semantic annotation of content, which automates the extraction of concepts in learning objects, using text mining techniques. A prototype was developed to demonstrate the operation of the architecture and conducted a case study applied to the learning environment Moodle. This work is expected to build new forms of knowledge discovery and reuse of information from data available in learning environments, through structured RDF description of learning objects. Furthermore, linking concepts to the data source in the Data *Web* (*Web* of Linked Data), enriching the knowledge base of the learning environment with semantic mashup between these concepts and pre-existing features of *Web* data, enabling increased productivity in the teaching-learning process.

KEYWORDS: Semantic annotation, Distance learning (ODL), Learning object, Linked data

LISTA DE FIGURAS

LISTA DE TABELA E QUADROS

ABREVIATURAS E SIGLAS

API	Application Programming Interface
DLS	Desambiguação Lexical de Sentido
DCMI	Dublin Core Metadata Initiative
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifiers
JSON	JavaScript Object Notation
LCMS	Learning Content Management System
LO	Learning Object
LOD	Linked Open Data
LOM	Learning Object Metadata
MUTO	Modular Unified Tagging Ontology
ORM	Object RDF Mapper
OWL	*Web* Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
REST	Transferência do Estado Representativo
SAIL	Storage And Inference Layer
SeRQL	Sesame RDF Query Language

SOA	Service-oriented architecture
SGBD	Sistemas de Gerenciamento de Banco de Dados
SPARQL	SPARQL Protocol and RDF Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

Com o crescimento da modalidade de ensino a distância – EAD, aliado à evolução das Tecnologias da Informação (TI), tornou-se necessário incentivar essa modalidade de ensino através da pesquisa por novas tecnologias para suporte ao ensino de EAD. A Internet impulsiona esse movimento, pois facilita o compartilhamento e processamento de documentos eletrônicos, independente de sistema operacional, hardware ou dispositivo (GOMES, 2006).

Um dos elementos vitais no ensino a distância é a disponibilização de materiais de aprendizagem relevantes e de valor. É sempre recomendável que seja valorizado o reaproveitamento e a interoperabilidade entre diferentes plataformas na produção de conteúdos e recursos didáticos de qualidade. Estes itens representam um custo elevado no processo de ensino e aprendizagem (MOURA, 2005).

Segundo Araujo (2003), a *Web* está se tornando uma grande biblioteca virtual, onde a informação sobre qualquer assunto está disponível, a qualquer hora e em qualquer lugar, com ou sem custo, criando oportunidades em várias áreas do conhecimento humano, dentre as quais a Educação. Porém, as informações na *Web* não são estruturadas e organizadas, as máquinas não podem “compreender” e nem “interpretar” o significado das informações. Embora muitas aplicações educacionais baseadas na *Web* tenham sido desenvolvidas nos últimos anos, alguns problemas nesta área não foram resolvidos, entre os quais está a pesquisa de materiais e objetos de aprendizagem mais inteligentes e eficientes.

Em (BERNERS-LEE, et al., 2001) foi proposta uma extensão da *Web* convencional, onde metadados estruturados inteligíveis por máquina são adicionados à *Web*, de forma que computadores possam entender o significado da informação publicada e, portanto, executar automaticamente, em larga escala, a

tarefas que são executadas manualmente. Esta nova *Web* que agrega significado a informação é denominada *Web* Semântica.

Como parte do movimento da *Web* Semântica, encontra-se em (BERNERS-LEE, 2006) a definição do conceito de *Linked Data* ou Dados Ligados, que são um conjunto de diretrizes para publicar e conectar dados estruturados na *Web*, formando a chamada *Web of Linked Data* (*Web* de Dados). A *Web* de dados é voltada para processamento por máquinas, visando serviços de busca mais eficientes, integração automática de dados e, ainda, inferência automática de dados, gerando novos dados.

A agregação de valor semântico aos documentos, proposta pela *Web* semântica, é uma forma de organizar o processo de publicação, recuperação e enriquecimento da informação. As anotações semânticas, ou seja, as associações das expressões relevantes de trechos de textos ou metadados descrevendo os documentos a conceitos e instâncias descritas em um domínio podem permitir que a recuperação da informação contida nos documentos seja realizada com maior precisão e eficácia, facilitando a sua reutilização. Muito embora, outros problemas precisam ser solucionados para alcançar este objetivo, dentre eles, a definição de métodos e ferramentas para automatizar o processo de anotação semântica.

Boa parte da automatização do processo de anotação semântica está relacionada com a descoberta de conceitos em conteúdos textuais. Para Gomes (2006), através da análise de textos é possível a descoberta de conceitos, classificações automatizadas e sumarizações para documentos não estruturados. Para que esta descoberta possa ser automatizada são utilizadas técnicas de Mineração de texto (*Text Mining* ou *Knowledge Discovery from Texts* - KDT) que pode ser definida como o nome dado às técnicas de análise e extração de dados a partir de textos, frases ou apenas palavras. Uma das áreas de conhecimento que compõem a mineração de textos é o Processamento de Linguagem Natural (PLN), essencial para o reconhecimento de textos em linguagem natural. PLN é uma área de pesquisa cujo objetivo é estudar o desenvolvimento de técnicas e ferramentas que analisam, reconhecem ou geram textos em linguagens humanas ou linguagens naturais. Para Lopes (2011), PLN é uma área com grandes desafios, pois a linguagem natural é rica em ambiguidades, diferentemente das linguagens formais que são definidas evitando a ambiguidade.

1.1. MOTIVAÇÃO

O presente trabalho está motivado pelo contexto da Educação a Distância e sua constante evolução e adaptação às novas exigências de desenvolvimento tecnológico, onde se faz necessária a construção de novas formas de descoberta de conhecimento e reuso de informação, a partir de dados disponíveis em ambientes de aprendizagem.

Com isso, será apresentada uma proposta de arquitetura orientada a serviço para otimizar e enriquecer os recursos empregados na Educação a Distância, com foco nos objetos de aprendizagem, através da utilização de técnicas de descoberta automática de conceitos e anotação semântica de conteúdo em um ambiente de aprendizagem.

Serão estudadas técnicas, ferramentas e metodologias no desenvolvimento de um *Web service* para extração automática de conceitos a fim de apoiar a anotação semântica automatizada de conteúdo, integrada a um ambiente de aprendizagem virtual.

1.2. OBJETIVOS

1.2.1. Objetivo Geral

O objetivo geral deste trabalho é propor uma arquitetura orientada a serviço, aplicada no desenvolvimento de um protótipo de *Web service* que automatize a extração de conceitos em objetos de aprendizagem, utilizando técnicas de mineração de texto e realizando a anotação semântica automatizada destes conteúdos, a partir de um ambiente de aprendizagem.

Estas anotações são utilizadas para a descrição explícita dos objetos de aprendizagem, por meio de metadados estruturados em *Resource Description Framework* (RDF), instanciando ontologias de referência e em conformidade com os princípios *Linked Data*, enunciados em Berners-Lee (2006).

1.2.2. Objetivos Específicos

Dentre os objetivos deste trabalho, pode-se citar:

- Definir uma arquitetura orientada a serviços aplicada no desenvolvimento de um *Web service* para anotação semântica de conteúdo;
- Definir método e técnica para extração de conceito em conteúdo textual não estruturado; Buscar e selecionar ferramentas para anotação semântica de conteúdo;
- Realizar anotação semântica no escopo de objetos de aprendizagem;
- Integrar *Web service* ao ambiente de aprendizagem Moodle;
- Descrever o enriquecimento dos objetos de aprendizagem através das anotações semânticas;
- Disponibilizar em formato RDF os objetos de aprendizagem anotados;
- Criar repositório integrado ao ambiente de aprendizagem para armazenamento e recuperação dos objetos de aprendizagem anotados;
- Criar relacionamento*,* ou seja, o *marshup* semântico entre os conceitos presentes nos objetos de aprendizagem e os conceitos presentes em recursos pré-existentes da *Web* de dados;

1.3. PROPOSTA DO TRABALHO

Neste trabalho será proposta uma arquitetura orientada a serviços, aplicada no desenvolvido de um protótipo de *Web service* para anotação semântica de conteúdo, que automatiza a extração de conceitos em objetos de aprendizagem, utilizando técnicas de mineração de texto, e armazenamento dos relacionamentos semânticos em um repositório RDF, realizando o *marshup* semântico entre os conceitos presentes nos objetos de aprendizagem e os conceitos presentes em recursos pré-existentes da *Web* de dados.

Os serviços serão integrados, como estudo de caso, ao ambiente de aprendizagem Moodle, por meio de *plugins* clientes do *Web service*, que permitem a anotação semântica e buscas dos objetos de aprendizagem diretamente na plataforma, de forma transparente para o usuário.

Para a realização do *mashup* semântico, dentre as fontes de dados presentes na *Web* de Dados, será utilizada a DBpedia (Wikipedia em RDF) (LEHMANN et al., 2014) (DBPEDIA, 2014), por dois motivos, a saber: ser uma fonte de dados central de referência e por descrever múltiplos domínios de conhecimento (*crossdomain*).

A arquitetura proposta do *Web service* visa possibilitar integrações futuras com outros ambientes de aprendizagem.

A base do *Web service* será composta pelos seguintes módulos:

- Extração de Conceitos – responsável por extrair conceitos de conteúdo textual não estruturado utilizando técnicas de mineração de texto.
- Anotação Semântica – responsável pela transformação dos conceitos extraídos em anotações semânticas e a ligação destes conceitos com outras fontes de dados da *Web* de Dados.
- Armazenamento – responsável por persistir os dados e as anotações semânticas.
- Consulta – responsável por disponibilizar uma interface de recuperação ao conteúdo anotado.

O ambiente de aprendizagem Moodle será customizado através da instalação e desenvolvimento de módulos que permitam a comunicação com os diversos serviços da arquitetura, possibilitando uma visão integradora dos serviços.

1.4.CONTRIBUIÇÕES ESPERADAS

O presente trabalho busca proporcionar as seguintes contribuições:

- Disponibilizar um *Web service*, baseado em uma arquitetura orientada a serviço para otimizar e enriquecer os recursos empregados na Educação a Distância;
- Demonstrar através da utilização de técnicas de mineração de texto a efetividade na extração de conceitos em determinado domínio;
- Interligar conceitos a diversas fontes de dados na *Web*, enriquecendo os objetos de aprendizagem, possibilitando novas formas de descoberta de conhecimento e reuso de informação;
- Demonstrar como é possível melhorar a recuperação de objetos de

aprendizagem, através das técnicas de anotação semânticas de conteúdo;

- Construir novas formas de acesso a dados disponíveis em ambientes de aprendizagem.

1.5. ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em seis capítulos. O Capítulo 1 contém a introdução, que aborda a motivação, os objetivos e as principais contribuições esperadas.

O Capítulo 2 apresenta a fundamentação, onde são abordados os conceitos teóricos relacionados ao trabalho desenvolvido.

O Capítulo 3 apresenta a arquitetura proposta, aplicada no desenvolvimento do protótipo de *Web service*.

O Capítulo 4 apresenta o protótipo de *Web service* desenvolvido, baseado na arquitetura proposta para extração de conceitos, anotação semântica, consulta e armazenamento em repositório RDF. Também é apresentada a implementação do estudo de caso no ambiente de aprendizagem *Moodle*.

O Capítulo 5 apresenta os trabalhos relacionados que trazem o uso de tecnologias da *Web* semântica para o enriquecimento de objetos de aprendizagem em ambientes de aprendizagem.

O Capítulo 6 apresenta as considerações finais, as conclusões obtidas com esse estudo, as contribuições trazidas e o indicativo de trabalhos futuros.

2. FUNDAMENTAÇÃO

Neste capítulo, são abordados conceitos importantes para um melhor entendimento desta dissertação. São apresentadas definições de padrões de metadados, objetos de aprendizagem reutilizáveis, padrões de metadados para objetos de aprendizagem, mineração de texto, *Web* semântica, *Web* de dados, anotação semântica de conteúdo e arquitetura orientada a serviço.

2.1. PADRÕES DE METADADOS

Os metadados ou dados sobre dados fornecem informações sobre um determinado recurso. São dados descritivos que podem informar sobre o título, autor, data, publicação, palavras-chave, descrição, localização de recursos, seus objetivos e características, mostrando como, quando e por quem o recurso foi armazenado e como está formatado, promovendo a interoperabilidade, identificação, compartilhamento, integração, utilização/reutilização, gerenciamento e recuperação dos mesmos de maneira mais eficiente. (GOMES et al., 2005)

Os padrões de metadados presentes no instrumental de software são considerados as linguagens de marcação que são capazes de identificar cada entidade informacional digna de significado presente nos documentos (BAX, 2001). Etimologicamente, quer dizer "dado sobre dado", dado que descreve a essência, atributos e contexto de emergência de um recurso e caracteriza suas relações, facilitando seu acesso e uso potencial.

2.2. OBJETOS DE APRENDIZAGEM REUTILIZÁVEIS (RLOS)

O conceito dominante de objeto de aprendizagem (LO) é qualquer recurso digital usado com objetivo educacional. Tais objetos têm que formar blocos de informações e estar inseridos em um determinado ambiente de aprendizagem, precisando apresentar as seguintes características: reusabilidade, adaptabilidade, granularidade, acessibilidade, durabilidade e interoperabilidade (SILVA, CAFÉ e CATAPAN, 2010).

Um objeto de aprendizagem reutilizável é definido como qualquer entidade, digital ou não, que possa ser utilizada para aprendizagem, educação ou treinamento, ou mesmo qualquer recurso digital que possa ser reutilizado no apoio à aprendizagem com suporte de tecnologias (IEEE, 2011).

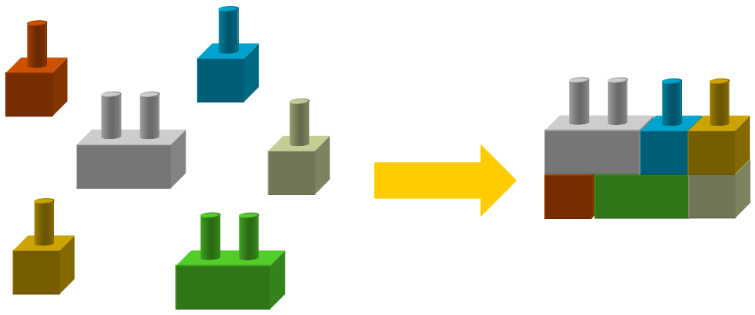

Figura 1: Blocos de lego representando um LO fragmentado,
Fonte: adaptado de Tarouco (2010).

Como metáfora educativa, costuma-se comparar os objetos de aprendizagem (LO) com blocos de LEGO para melhor explicar sua definição, onde cada peça pode se juntar a outras peças e compor novos objetos. O mesmo ocorre com os objetos de aprendizagem que após serem fragmentados, criam várias partes com possibilidade de reutilização em outros ambientes. A Figura 1 ilustra os blocos de LEGO comparados com um LO fragmentado.

Um objeto de aprendizagem pode ser utilizado em diferentes contextos, pois sua característica principal é a capacidade de reuso. No entanto, para que ele possa ser recuperado e reutilizado, é necessário que esteja corretamente indexado (preenchimento dos metadados) e armazenado em um repositório onde possa ser

devidamente recuperado. (VARLAMIS e APOSTOLAKIS, 2006)

Como exemplos de objetos de aprendizagem, pode-se citar: gráficos, figuras, capítulos de um livro, uma animação, um apêndice de outro livro, entre outros (GIRARDI, 2004).

São grandes os desafios em se trabalhar com objetos de aprendizagem devido ao seu grau de granularidade. A granularidade de um LO está relacionada ao tamanho do objeto. Um curso on-line completo não exigiria grande esforço para ser reutilizado, mas como objeto de aprendizagem teria uma granularidade muito baixa, dificultando sua reutilização em diferentes contextos. Neste caso, se um objeto contiver apenas uma imagem estaria com granularidade alta, oferecendo mais oportunidade de reutilização. No entanto, quanto maior a granularidade, maior o trabalho para se organizar um curso completo (WILEY, 2000).

2.3. PADRÕES DE METADADOS PARA OBJETOS DE APRENDIZAGEM

Os metadados são responsáveis por tornar os objetos de aprendizagem acessíveis. Eles promovem a identificação e possibilitam o compartilhamento, a utilização, a reutilização, o gerenciamento e a recuperação dos LO de maneira mais eficiente. Os metadados atuam como organizadores e facilitadores na recuperação dos objetos de aprendizagem (SILVA, CAFÉ; CATAPAN, 2010).

Os metadados de objetos de aprendizagem (LOs) são as informações sobre os dados que compõem os LOs, sejam eles físicos ou digitais. Eles descrevem as suas principais características que serão utilizadas para sua catalogação em repositórios de objetos de aprendizagem reutilizáveis (RLOs) (TAROUCO e SCHMITT, 2010).

2.3.1. Learning Object Metadata (LOM)

O padrão de metadados *Learning Object Metadada* (LOM) foi representado pelo IEEE *Learning Technology Standards Committee* (LTSC) para facilitar a busca, aquisição, avaliação e utilização de Objetos de Aprendizagem para instanciação por aprendizes e instrutores ou processos automáticos de *software*. (IEEE, 2013)

Facilitando o comportamento e troca de LO, permitindo o desenvolvimento de repositórios, levando em consideração a diversidade cultural, contextos lingüísticos nos quais os Objetos de Aprendizagem e seus metadados são reutilizados. Este é um dos metadados mais utilizados para descrição de Objetos de Aprendizagem (GOMES et al., 2005) É considerado um padrão aberto e internacionalmente reconhecido por facilitar a busca, avaliação, construção e uso de LO, provendo um modelo de dados normalmente codificado em XML, que segue a norma IEEE Std 1484.12.1 – 2002. (IEEE, 2002)

Os elementos de dados que descrevem um LO, conforme o padrão LOM, são agrupados em nove categorias listadas no Quadro 1.

Características	**Elementos**
GERAIS	Reúnem as características gerais sobre o objeto de aprendizagem tais como identificador (catálogo, entrada), título, idioma, descrição, palavra-chave, cobertura, estrutura, nível de agregação
CICLO DE VIDA	Descrevem a evolução, o estado atual e as diversas contribuições, tais como: versão, status, contribuintes (papel, entidade, data).
META-METADADOS	Descrevem os metatados que estão sendo utilizados, tais como: identificador (catálogo, entrada), contribuintes (papel, entidade e data), esquema de metadados, linguagem.
TÉCNICAS	Reúnem aspectos técnicos necessários para utilizar o objeto de aprendizagem, bem como suas características próprias, tais como formato, tamanho, localização, requisitos, comentários sobre instalação, requisitos para outras plataformas, duração
EDUCACIONAIS	Descrevem aspectos educacionais e pedagógicos associados, tais como tipo de interatividade, tipo de recurso de interatividade, tipo de recurso de aprendizagem, nível de interatividade, densidade semântica, papel do usuário final, contexto, faixa etária, dificuldade, tempo previsto para aprendizado, descrição e linguagem.
DIREITOS	Relatam condições de uso e aspectos de propriedade intelectual, tais como custo, direito de cópia e outras restrições, descrição.
RELAÇÕES COM OUTROS RECURSOS	Descrevem como este objeto de aprendizagem está relacionado com outros objetos de aprendizagem, tais como tipo e recurso (identificador - catálogo e entrada e descrição).
OBSERVAÇÕES	Reúnem comentários sobre o uso educacional do objeto de aprendizagem e dados sobre a autoria dos comentários, tais como entidade, data e descrição.
CLASSIFICAÇÃO	Descrevem como um objeto de aprendizagem enquadra-se em um sistema de classificação particular, tais como propósito, caminho taxonômico (identificador e entrada), descrição e palavra-chave

Quadro 1: Elemento de Dados Desvcritos pleo Padrão LOM

A árvore completa dos elementos de dados do padrão LOM e seus atributos estão ilustrados na Figura 2.

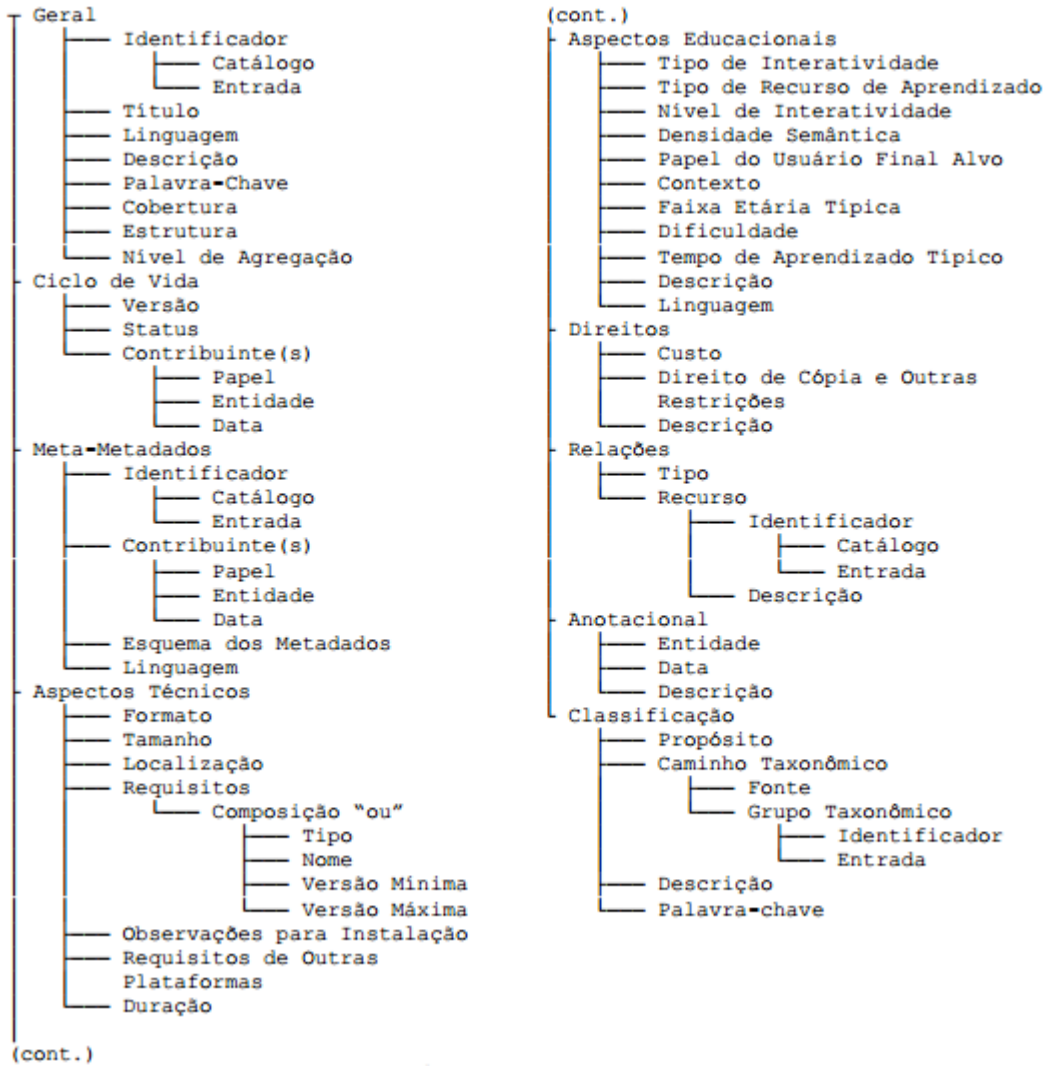

Figura 2: Árvore completa do padrão LOM. (GHELMAN, 2006)

2.3.2.Dublin Core

O *Dublin Core Metadata Element Set* ou simplesmente *Dublin Core* (DCMIa, 2012) é um vocabulário de quinze propriedades para uso na descrição de recursos. O nome *Dublin Core* teve origem em um workshop realizado na cidade de Dublin, Ohio e “core” é devido os elementos serem amplos e genéricos, utilizados para descrever uma ampla gama de recursos (DCMIa, 2012).

Os quinze elementos do *Dublin Core* são partes de um conjunto maior de vocabulários de metadados e especificações técnicas mantidas pelo *Dublin Core Metadata Initiative* (DCMI) e destina-se a ser utilizado por usuários comuns. Espera-

se que os usuários da catalogação sejam capazes de utilizar o *Dublin Core* para descrição de recursos, tornando suas coleções mais visíveis às ferramentas de busca e sistemas de recuperação. É utilizado para padronizar metadados de bibliotecas digitais e objetos de aprendizagem.

O conjunto completo de vocabulários, *DCMI Metadata Terms* (DCMIb, 2012), inclui um conjunto de classes de recursos, esquemas de codificação de vocabulários, sintaxe e esquemas de codificação. (DCMIa, 2012)

Os elementos de dados que descrevem um LO, conforme o padrão Dublin Core estão listadas no Quadro 2.

Atributos	Descrição
IDENTIFICADOR	Identificação não ambígua do recurso dentro de um dado contexto.
COLABORADOR	Entidade responsável pela contribuição ao conteúdo do recurso
COBERTURA	Extensão ou cobertura espaço-temporal do conteúdo do recurso.
CRIADOR	Entidade principal responsável pela elaboração do conteúdo do recurso.
DATA	Data associada a um evento no ciclo de vida do recurso.
DESCRIÇÃO	Descrição sobre o conteúdo do recurso.
FORMATO	Manifestação física ou digital do recurso.
LINGUAGEM	Idioma do conteúdo intelectual do recurso.
PUBLICADOR	É a instituição responsável pela difusão do recurso.
RELAÇÃO	Uma referência para um outro recurso que tenha dado origem ao presente recurso.
DIREITOS	Informações sobre os direitos do recurso e de seu uso.
FONTE	Uma referência para um outro recurso que tenha dado origem ao presente recurso.
ASSUNTO	Assunto referente ao conteúdo do recurso.
TÍTULO	Título dado ao recurso.
TIPO DE RECURSO	A natureza ou gênero do conteúdo do recurso.

Quadro 2: Elementos de Dados Descritos pelo Padrão Dublin Core (DCMIa, 2012)

Todos os elementos relacionados no Quadro 2 são opcionais ao descrever um recurso, não havendo restrição quanto número de vezes que cada elemento pode ser utilizado, nem mesmo quanto ao tamanho do valor de cada elemento.

Desde janeiro de 2008, o *Dublin Core* inclui domínios formais e faixas de

definições de suas propriedades, seguindo as boas práticas da *Web* Semântica. Portanto, para não afetar a conformidade das implementações existentes anteriores a está nova definição foi mantida uma implementação simples do *Dublin Core* e criadas quinze novas propriedades com nomes idênticos aos do *Dublin Core Metadata Element Set* versão 1.1 com o *namespace dcterms* definido em http://purl.org/dc/terms/. (DCMI, 2012)

Estas quinze novas propriedades foram definidas como subpropriedades das propriedades correspondentes de *Dublin Core Metadata Element Set* versão 1.1, e domínios e faixas atribuídos, conforme especificado em *DCMI Metadata Terms*. (DCMIb, 2012)

Em boa parte das utilizações, o conjunto de descritores do Dublin Core fica embutido no próprio documento descrito (HTML, XML – *Extensible Markup Language* e outros), porém dependendo do recurso, os metadados (metainformação) podem ser encontrados separados do recurso catalogado.

Dentre as principais características do padrão de metadados Dublin Core pode-se destacar a simplicidade na descrição dos recursos, escopo internacional e extensibilidade (o que permite sua adaptação às necessidades adicionais de descrição) e o entendimento semântico universal dos elementos (SOUZA et al., 2000).

2.4.MINERAÇÃO DE TEXTOS

Mineração de texto (*Text Mining* ou *Knowledge Discovery from Texts* - KDT) pode ser definido com o nome dado as técnicas de análise e extração de dados a partir de textos, frases ou apenas palavras (Gomes, 2006). Lopes (2004) afirma que Mineração de Textos pode também ser definido como um conjunto de técnicas e processos que se prestam a descobrir conhecimento inovador nos textos.

Hearst (2003), descreve o processo de Mineração de Texto como sendo a descoberta pelo computador de informação nova e previamente desconhecida, através da extração automática de informação de várias fontes de texto.

Para Gomes (2006), através da análise de textos é possível a descoberta de conceitos, classificações automatizadas e sumarizações para documentos não

estruturados.

Em Monteiro et al. (2006) são descritas as três principais etapas do *Text Mining*: a etapa de Pré-Processamento ou preparação dos dados, a etapa de Análise dos Dados e Extração do Conhecimento, também conhecida como Etapa de Processamento de Textos e a etapa de Pós-processamento ou Avaliação das Descobertas.

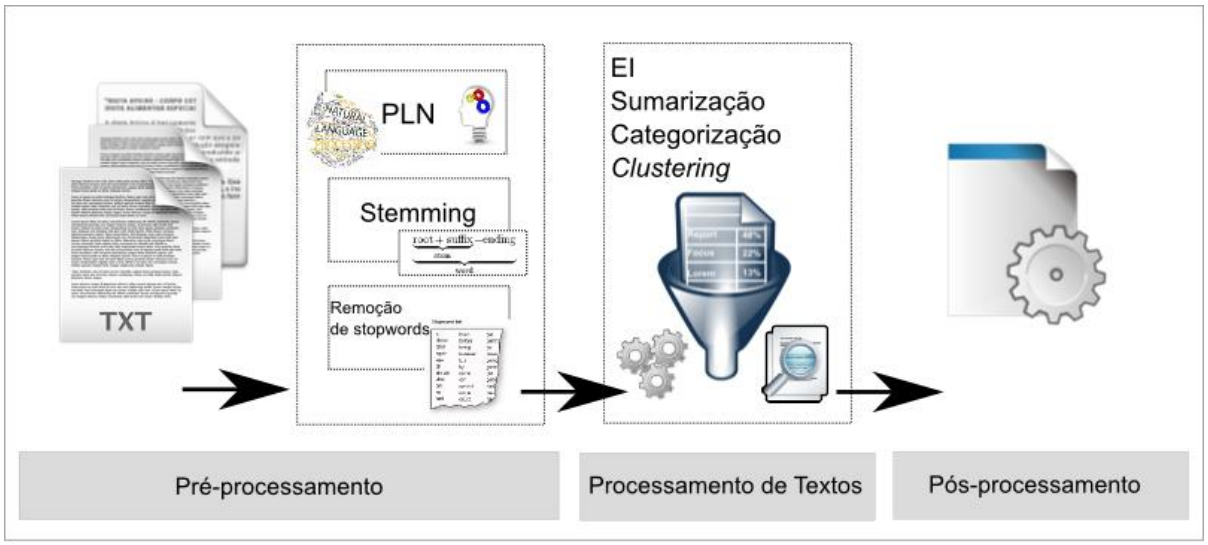

Figura 3: Etapas do processo de Mineração de Texto.

A Figura 3 demonstra o processo de mineração de texto que se inicia na etapa de Pré-processamento, é envolve a aplicação das técnicas de Processamento de Linguagem Natural PLN, *Stemming*, ou seja, o processo de redução ao radical original da palavra derivada ou flexionada, e remoção de *Stopword,* que consiste no processo de remoção de palavras que não possuem nenhum valor semântico, sendo somente úteis para a compreensão geral do texto. A segunda etapa é o Processamento de Textos, onde podem ser aplicados métodos de Extração de informação, Sumarização, Categorização e Clusterização. A última etapa, Pós-processamento ou Avaliação das Descobertas, são apresentados os resultados obtidos na fase anterior, baseado nas regras definidas para a base de conhecimento.

2.4.1. Extração de Informação

Extração de Informação (EI) é um processo de remoção de informação estruturada a partir de grandes volumes de dados. EI pode ser vista como qualquer método que filtre informações de um grande volume de texto. (GRISHMAN, 1997)

Lopes (2004) afirma que Extração de Informação é uma área de pesquisa que conjuga *Text Mining* e Processamento de Linguagem Natural (PLN), realizando o processo de extrair informação pré-definida sobre objetos e relacionamentos entre eles a partir de documentos textuais.

A busca do conhecimento a partir de bases de dados textuais não estruturadas exige a compreensão de dados armazenados (GOMES, 2006). Automatizar esta atividade é tão complexo como construir um sistema para compreender a linguagem natural (MOULIN, 1992).

2.5. WEB SEMÂNTICA

Segundo Araujo (2003), a *Web* está se tornando uma grande biblioteca virtual, onde a informação sobre qualquer assunto está disponível a qualquer hora e em qualquer lugar, com ou sem custo, criando oportunidades em várias áreas do conhecimento humano, dentre as quais a Educação. Porém, as informações na *Web* não são estruturadas e organizadas, as máquinas não podem “compreender” e nem “interpretar” o significado das informações.

Com crescimento da *Web*, torna-se imprescindível a busca por tecnologias capazes de manipular grandes quantidades de informação de diferentes níveis de complexidades de forma automática

Em Berners-Lee et al. (2001), foi proposta uma extensão da *Web* convencional, onde metadados estruturados inteligíveis por máquina são adicionados à *Web*, de forma que computadores possam entender o significado da informação publicada e, portanto, executar automaticamente, em larga escala, as tarefas que são executadas manualmente. Esta nova *Web* que agrega significado a informação é denominada *Web* Semântica.

“A *Web* Semântica não é uma *Web* separada, mas uma extensão da atual, na qual a informação é utilizada com significado bem definido, aumentando a capacidade dos computadores para trabalharem em cooperação com as pessoas”

(BERNERS-LEE et al., 2001).

Para Araujo (2003), é uma possibilidade de ter dados, na *Web*, conectados e com significados definidos, de modo a ser usados pelos computadores. Por exemplo, se em determinada página *Web* existir a palavra “banco” será possível distinguir se ela significa um “assento” ou um “estabelecimento comercial”. Neste contexto, pode-se perceber que a *Web* Semântica pode facilitar e melhorar a recuperação de informações relevantes, já que a própria máquina, dotada de ferramentas inteligentes, pode identificar o conteúdo de um site, por associação e dedução automática, antes de trazê-lo ao usuário como resultado de uma pesquisa (PICKLER, 2006).

Para que as informações possam ser compreendidas tanto por humanos quanto por computadores, Bernes-Lee et al. (2001) propõem os padrões da *Web* Semântica, representado na Figura 4, onde é definida uma arquitetura em camadas.

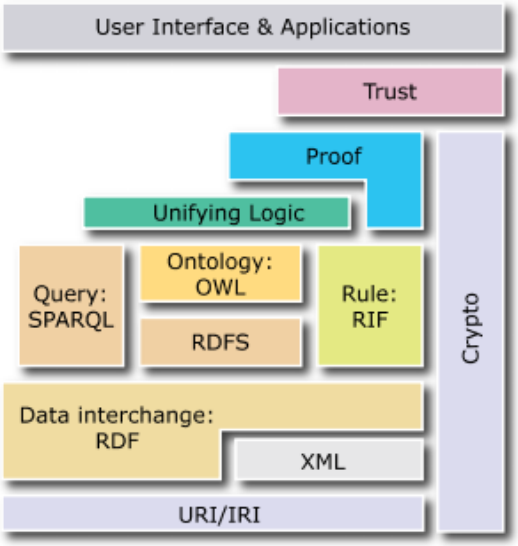

Figura 4: Arquitetura em camadas da *Web* Semântica (HAWKE et al., 2013).

Esta arquitetura define a representação sintática, estrutural, semântica e lógica de informações referentes aos recursos *Web*. Para atingir os propósitos da *Web* Semântica, Souza e Alvarenga (2004) observam que é necessária uma padronização de tecnologias, de linguagens e de metadados descritivos, de forma que esta possa ser consumida por outros usuários, de uma maneira automática e não ambígua.

A seguir serão apresentadas as principais camadas da *Web* Semântica.

2.5.1. URI / IRI

A primeira camada denominada URI (*Uniform Resource Identifier*) / IRI (*Internationalized Resource Identifiers*), definida na arquitetura da *Web* Semântica, permite que recursos (objetos de dados) disponíveis na *Web* sejam identificados de forma global, usando o mesmo esquema de endereços do protocolo HTTP, já consagrado na *Web* convencional. A RFC 3986[1] define URI como uma seqüência compacta de caracteres que identifica um recurso abstrato ou físico e a RCF 3987[2] descreve os padrões a serem adotados quanto a codificação de caracteres do protocolo IRI. Por exemplo, poderíamos ter um URI que identifica o autor deste trabalho (http://www.ucam-campos.br/pessoas/ronaldoamaral), bem como um URI que identifica o próprio trabalho (http://www.ucam-campos.br/dissertacao/1234).

2.5.2. XML

A camada XML (*Extensible Markup Language*) em conjunto com XML *namespace* e XML *schema*, garantem uma definição de sintaxe comum a ser usada na *Web* semântica. XML é uma linguagem de marcação para documentos contendo informação estruturada.

2.5.3. RDF

A camada RDF, apresenta a principal forma de representação na *Web* Semântica, o *Resource Description Framework* (RDF). (RDF, 2014) O RDF é um modelo de dados para representação de informação acerca de recursos em forma de grafos, baseado-se em triplas “sujeito-predicado-objeto (ou recurso-propriedade-valor). Por exemplo, para representar a informação que a pessoa identificada pelo URI "http://www.ucam-campos.br/pessoas/ronaldoamaral" é autor do trabalho

[1] Disponível em: < http://www.ietf.org/rfc/rfc3986.txt>. Acesso em 15 out 2014

[2] Disponível em: < http://www.ietf.org/rfc/rfc3987.txt>. Acesso em 15 out 2014

correspondente ao URI "http://www.ucam-campos.br/dissertacao/1234", bastaria criar um link entre estes dois URIs e associar o rótulo "é autor de" ao link. Este rótulo, na verdade, também é um URI "http://example.org/ontology/authorOf " que identifica uma propriedade definido em alguma ontologia.

Por fim, temos a tripla apresentada no grafo da Figura 5 e a descrição na sintaxe RDF/XML[3] apresentado na Figura 6.

RDF/XML é uma sintaxe baseada em XML para representação de grafos RDF. Outras notações podem ser utilizadas para representação de grafos RDF, tais como N3[4], NTriples[5] e Turtle[6].

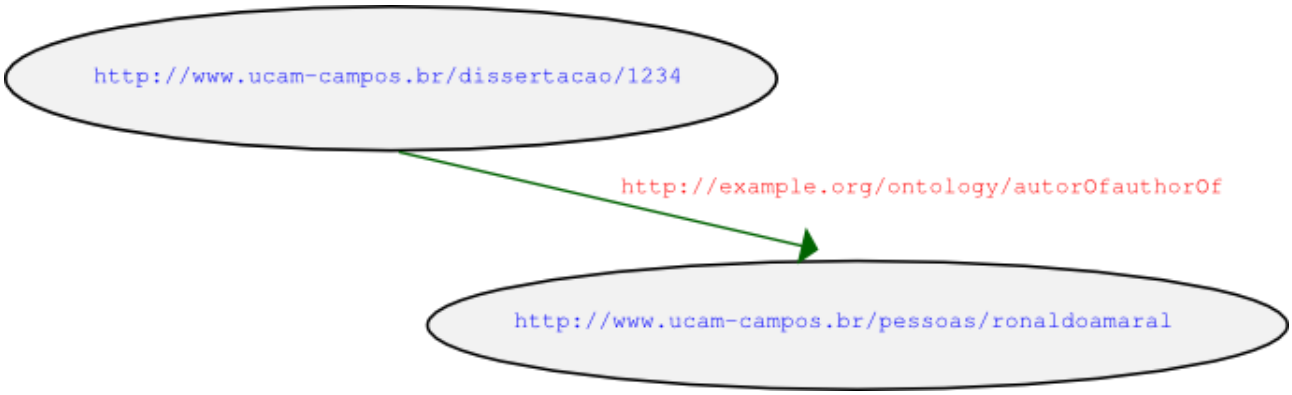

Figura 5: Representação da Tripla em Grafo.

```
1: <?xml version="1.0"?>
2: <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3:   xmlns:onto="http://example.org/ontology/autorOf">
4:   <rdf:Description rdf:about="http://www.ucam-campos.br/dissertacao/1234">
5:     <onto:authorOf rdf:resource="http://www.ucam-campos.br/pessoas/ronaldoamaral" />
6:   </rdf:Description>
7: </rdf:RDF>
```

Figura 6: Representação da tripla RDF na sintaxe RDF/XML.

2.5.4. RDFS

A camada denominada de RDFS apresenta a linguagem *RDF Schema* (RDFS), utilizada para definição de vocabulários RDF, possibilitando a definição de

[3] Disponível em: <http://www.w3.org/TR/REC-rdf-syntax/>. Acesso em 19 set 2014
[4] Disponível em: <http://www.w3.org/TeamSubmission/n3/>. Acesso em 19 set 2014
[5] Disponível em: <http://www.w3.org/2001/sw/RDFCore/ntriples/>. Acesso em 19 set 2014
[6] Disponível em: <http://www.w3.org/TeamSubmission/turtle/>. Acesso em 19 set 2014

taxonomias de recursos em termos de uma hierarquia de classes, ou seja, uma ontologia simples (*light-weight ontologies*) (ZHU & MADNICK, 2007). Esta linguagem fornece meios para definição de classes e propriedades de um domínio específico (BRICKLEY e GUHA, 2014). RDFS provê mecanismos necessários para facilitar a descrição de classes e propriedades, de modo ser possível indicar quais classes e propriedades podem ser utilizadas em conjunto. Por exemplo, para dizer que a propriedade nome (*name*) da ontologia FOAF pode ser usada para descrever uma classe Pessoa (*Person*) desta mesma ontologia.

Brickley e Guha (2014) afirmam que as classes e propriedades do *RDF Schema* são similares ao sistema de tipos nas linguagens de programação orientadas a objetos, como Java. A diferença do *RDF Schema* destes sistemas é que ao invés de definir classes em função das propriedades que suas instâncias podem ter, RDFS descrevem propriedades em função das classes de recurso aos quais se aplica, ou seja, é o conceito de domínio (*domain*) e contradomínio (*range*). Por exemplo, podemos definir a propriedade “autor” para o domínio de “Documento” e o contradomínio de “Pessoa”. Nos sistemas orientados a objetos teríamos uma classe “Livro” com um atributo “autor” do tipo “Pessoa”.

RDFS permite a organização de classes de forma hierárquica, por exemplo, a classe “Papagaio” pode ser atribuída como uma subclasse de “Ave”, que por sua vez é uma subclasse de “Animal”. Esta hierarquia permite afirmar que um recurso que é definido na classe “Papagaio” também pertence a classe “Animal”.

Segundo Bomfim (2011), as classes e propriedades RDF fornecem informações adicionais sobre os recursos que descrevem, não sendo uma restrição sobre as informações.

2.5.5. Web Ontology Language (OWL)

A camada Ontologia (*Ontology*) apresenta a linguagem *Web Ontology Language* (OWL), utilizada para definição de ontologias na *Web* (BECHHOFER, 2009). Ontologia é um termo oriundo da filosofia que se refere à ciência de descrever os tipos das entidades no mundo e como elas se relacionam. Segundo Smith et al., (2004), OWL foi projetada para representar o conhecimento rico e

complexo sobre entidades, grupos de entidades e as relações entre as entidades.

Uma ontologia OWL inclui a descrição de classes, propriedades e suas instâncias (SMITH et al., 2004). A semântica formal de OWL especifica como derivar as consequências lógicas de uma ontologia, isto é, os fatos não literalmente presentes na ontologia, mas decorrentes da sua semântica. (SMITH et al., 2004)

OWL foi projetada para ser usada por aplicações que precisam processar o conteúdo da informação, não sendo apenas uma forma de visualização da informação (MCGUINNESS & HARMELEN, 2004). Ela foi desenvolvida como uma extensão do vocabulário RDF, sendo uma derivação da linguagem DAML+OIL (BECHHOFER et al. 2004). Mcguinness & Harmelen (2004) destaca que OWL é uma revisão da linguagem de ontologia *Web* DAML+OIL, onde foram incorporadas as lições aprendidas a partir da concepção e aplicação da DAML+OIL.

A linguagem OWL possui facilidades, não encontrados em linguagens XML, RDF e RDF Schema, para expressar significado e semântica (MCGUINNESS & HARMELEN, 2004). Ela possui três sublinguagens projetadas para uso específico, sendo elas: OWL Lite, OWL DL, e OWL Full. (SMITH et al., 2004)

Para Bomfim (2011), OWL é considerada uma linguagem mais expressiva do que RDFS, pois pode expressar diversos tipos de relacionamentos entre classes e propriedades, não possíveis de serem expressos em RDFS.

A linguagem OWL encontra-se em sua segunda versão, conhecida com o OWL 2 (W3Cb, 2012) recomendada em Outubro de 2009 pela W3C como linguagem para descrição de ontologias para *Web* Semântica.

2.5.6. Consulta (SPARQL)

A camada de Consulta (*query*) apresenta SPARQL, uma linguagem de consultas para RDF (PRUD'HOMMEAUX & SEABORNE, 2008) e protocolo de acesso a dados RDF (KENDALL et al., 2008).

A linguagem de consulta SPARQL permite manipular e recuperar dados armazenados em RDF, assim como a linguagem SQL (EISENBERG et al., 2004) realiza em banco de dados relacionais. SPARQL possibilita a recuperação de dados estruturados e semiestruturados, a exploração de dados em consultas com relações

desconhecidas e uniões complexas de conjunto de dados diversos em uma única consulta. O resultado de uma consulta SPARQL pode ser um conjunto de resultados ou um grafo RDF. (W3Cc, 2013)

Uma consulta SPARQL possui a estrutura apresentada na Figura 7, sendo composta por: declarações de prefixos (PREFIX), definições de conjunto de dados (FROM), cláusulas de resultado (SELECT), padrão de consulta (WHERE) e modificadores de consulta (ORDER BY, LIMIT e DINSTINCT, por exemplo).

```
# declarações de prefixo
PREFIX foo: <http://example.com/resources/>
...
# definição de conjunto de dados
FROM ...
# cláusula de resultado
SELECT ...
# padrão de consulta
WHERE {
   ...
}
# modificadores de consulta
ORDER BY ...
```

Figura 7: Estrutura consulta SPARQL
Fonte: Elias e Holanda (2014).

A Figura 9 demonstra uma consulta SPARQL, onde é retornado o titulo do objeto de aprendizagem identificado pela URI "*http://exemplo.com.br/objetos/historia_do_brasil*" dado o grafo apresentada na Figura 8.

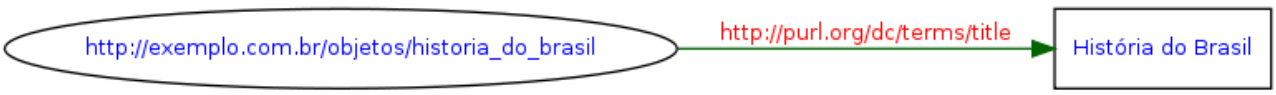

Figura 8: Grafo recurso História do Brasil.

```
1 SELECT  ?title
2 WHERE
3   { <http://exemplo.com.br/objetos/historia_do_brasil> <http://purl.org/dc/elements/1.1/title> ?title }
```

Figura 9: Exemplo de consulta SPARQL.

O resultado desta consulta é apresentado conforme Tabela 1:

Tabela 1: Resultado consulta SPARQL.

Titile
“História do Brasil”

O protocolo SPARQL consiste em duas operações HTTP, uma para operações de consulta (*query*) e uma operação para atualização (*update*) (FEIGENBAUM et al. 2013). O protocolo possui uma interface com a descrição abstrata das operações, de maneira a serem implementadas em *Web services*. Sua construção é feita no topo do protocolo HTTP, todos os requisitos para requisições (*request*) e respostas (*response*) HTTP podem ser encontradas em (FEIGENBAUM et al. 2013). A versão atual do protocolo recomendada pela W3C desde março de 2013 é a 1.1. (FEIGENBAUM et al. 2013)

Web services que implementam o protocolo SPARQL, também são conhecidos como *Sparql Endpoint*. Eles permitem que clientes do serviço, sejam estes, humanos ou máquinas, possam executar consultas a uma base de conhecimento utilizando a linguagem SPARQL. O resultado destas consultas são retornados em diferentes formatos processáveis por máquinas, por exemplo, JSON, RDF/XML, XML, o que possibilita, fornecer a um navegador RDF uma URI que referencia um arquivo RDF ou a URI de um *Sparql Endpoint*, que após o processamento da consulta requisitada, irá retornar um conjuto de dados RDF.

Sparql Endpoint são utilizados para explorar a *Web* Semântica. Por exemplo, pode citar o projeto *Linked Open Data* (LOD) que disponibiliza um conjunto de repositórios RDF interligados, acessíveis por diversos *Sparql Endpoints*. (SW, 2011)

2.6. WEB DE DADOS: LINKED DATA

Como parte do movimento da *Web* Semântica, encontra-se em Berners-Lee (2006), a definição do conceito de *Linked Data* ou Dados Ligados, que são um conjunto de diretrizes para publicar e conectar dados estruturados na *Web*, formando a chamada *Web of Linked Data* (*Web* de Dados), cuja topologia é ilustrada

na Figura 10. Cada círculo representa um conjunto de dados em RDF (*datasets*) publicado segundo os princípios *Linked Data*. Os links entre os conjuntos de dados indicam a existência de pelo menos uma tripla com o sujeito no *dataset* de origem e com o objeto no *dataset* de destino, ou seja, *mashup* semântico entre fontes de dados. É importante destacar o conjunto de dados DBpedia, apresentado no centro da Figura 10, por ser uma fonte de dados central de referência e por descrever múltiplos domínios de conhecimento (*crossdomain*).

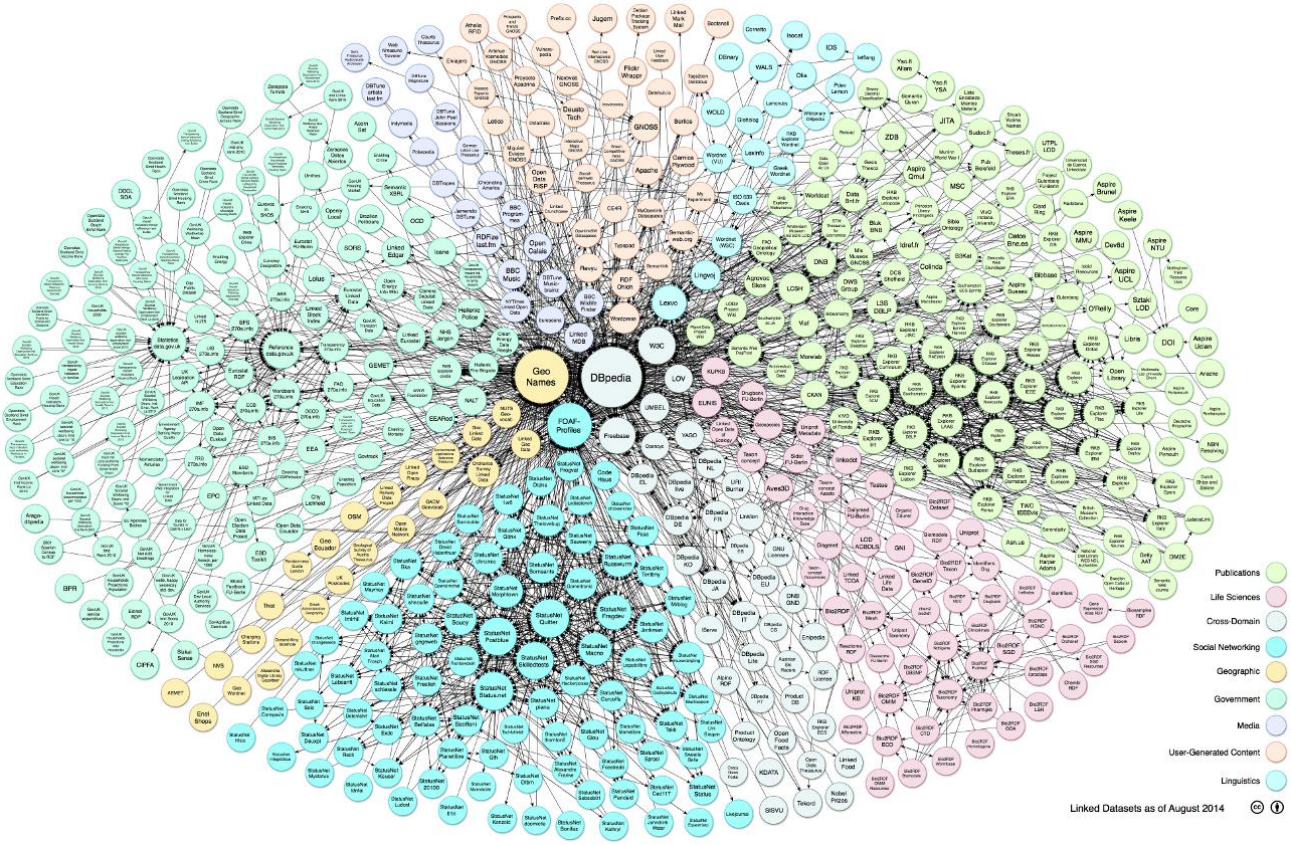

Figura 10: Topologia da Web de Dados (Linking Open Data cloud diagram)
Fonte: Cyganiak e Jentzsch (2014).

A ideia é usar a arquitetura pré-existente da *Web* não apenas para publicar e interligar documentos, mas também para publicar e interligar (relacionar) dados. Em outras palavras, publicar dados diretamente na *Web*, identificando-os por URIs e estabelecer links (relacionamentos) entres estes dados por meio de URIs que representam propriedades descritas em ontologias. Tudo isso utilizando o modelo de dados RDF, formando um único grafo global de dados mundial. Desta forma, ao

acessar um URI, a máquina obtém um arquivo RDF, contendo triplas (recurso-propriedade-valor) que descrevem o recurso correspondente ao URI e, partir destas triplas, a máquina pode navegar para outras URIs obtendo os correspondentes arquivos RDFs e, portanto, buscando mais dados e integrando-os para, por fim, fazer consultas sobre estes dados para nos auxiliar.

A *Web* de dados é voltada para processamento por máquinas, visando serviços de busca mais eficientes, integração automática de dados e, ainda, inferência automática de dados, gerando novos dados.

2.6.1: DBpedia

A DBpedia é um projeto que busca extrair e disponibilizar na *Web* informações estruturadas da enciclopédia aberta Wikipedia com o uso de tecnologias de *Web* Semântica e *Linked Data*. (LEHMANN et al., 2014)

Segundo Lehmann et al. (2014), a Wikipedia é o 6º mais popular *Website*, a enciclopédia mais amplamente utilizada e um dos melhores exemplos na criação de conteúdos colaborativos. Para Auer et al. (2007), o conteúdo da Wikipedia está em constante revisão, dando origem a uma rica fonte de dados de diferentes domínios. Lehmann et al. (2014) destaca que o modo como acontece a busca textual da Wikipedia não facilita para os usuários encontrarem as informações disponíveis.

O projeto DBpedia constrói uma base de conhecimento multilíngue por meio da extração de informações da Wikipedia em 111 idiomas (LEHMANN et al., 2014), tornando possível a realização de consultas sofisticadas e possibilitando a interligação de dados da DBpedia a outras fontes de dados da *Web* (AUER et al., 2007).

Os dados disponíveis na DBpedia são atualizados periodicamente por meio dos dados disponíveis na Wikipedia. A versão mais atual da DBpedia é a 3.9 de 09 de setembro de 2013, conta com uma base de conhecimento em sua versão em inglês, com 4 milhões de entidades, das quais 3,222 milhões estão classificadas de forma consistente em ontologia, incluído 832 mil pessoas, 639 mil lugares, 372 mil trabalhos (álbuns de música, filmes e jogos e outros), 209 mil organizações (empresas, instituições de ensino e outros), 226 mil espécies e 5.600 tipos de

doenças (THIBODEAU, 2014). Em sua totalidade nos 119 idiomas disponíveis, são 24.9 milhões de entidades, das quais 16.8 milhões são interligadas aos conceitos disponíveis em inglês.

Cada entidade presente na DBpedia possui uma URI com a sua descrição RDF, esta URI tem como base o endereço da Wikipedia. Por exemplo, as informações sobre o país Brasil estão disponíveis na página da Wikipedia "*http://pt.wikipedia.org/wiki/Brasil*" e a representação deste recurso na DBpedia está disponível na URI "*http://pt.dbpedia.org/resource/Brasil*". A descrição das entidades presentes na DBpedia incluem: *links* para outras fontes da *Web* descrevendo a entidade, descrição em vários idiomas, relacionamentos com outros recursos e classificações em três diferentes esquemas (*schemata*). (SAHNWALDT, 2013)

A DBpedia estabelece uma ontologia, criada manualmente com base nas *infoboxes* mais utilizadas na Wikipedia, a fim de representar semanticamente o conteúdo extraído. Esta ontologia consiste em 529 classes e mais de 2.333 propriedades. (SAHNWALDT, 2014)

Na Wikipedia, *infoboxes* são *templates* contidas nos artigos, apresentados em uma tabela no topo dos artigos contendo as informações mais relevantes (MORSEY et al., 2012). As *Infoboxes* são utilizadas com o intuito de sumarizar as informações comuns a diversos artigos e otimizar a navegação em artigos relacionados.

A base de dados da DBpedia permite a realização de consultas sofisticadas, com o uso da linguagem SPARQL ou por meio do *Faceted Wikipedia Search* (HAHN et al., 2010), uma interface gráfica que permite ao usuário a criação de consultas complexas.

A DBpedia atualmente é o principal *hub* de interligação da *Web of Linked Data* (*Web* de dados). As fontes de dados interligadas ao DBpedia fornecem cerca de 4,5 bilhões de dados em diferentes domínios, com uma abrangência de informações geográficas, pessoas, empresas, filmes, músicas, livros, publicações cientificas dentre outros. (KOBILAROV et al., 2009). Tim Berners-Lee (TALIS, 2008) considera a DBpedia como parte mais importante *Web of Linked Data*. Conforme pode ser observado na Figura 10 a DBpedia é o *hub* central da *Web* de dados.

2.6.2. Anotação Semântica de Conteúdo

As anotações semânticas, ou seja, as associações das expressões relevantes de trechos de textos ou metadados descrevendo os documentos a conceitos e instâncias descritas em um domínio podem permitir que a recuperação da informação contida nos documentos seja realizada com maior precisão e eficácia, facilitando a sua reutilização.

Segundo Popov (2003), anotação semântica é um esquema específico para a geração e uso de metadados, permitindo novos métodos de acesso à informação. Para Glonvezynski (2008), a anotação semântica adiciona no documento uma camada que descreve o seu conteúdo, tornando possível a recuperação da informação por agentes de software de forma mais precisa, através da associação do documento a uma ontologia.

As anotações semânticas são fundamentais no desenvolvimento da *Web* Semântica, seja na criação de novos documentos com conteúdo semântico descrito ou para prover semântica a documentos já existentes. O World Wide *Web* Consortium (W3C) recomenda que sejam usadas ontologias escritas na linguagem OWL para representação de conceitos sobre um determinado domínio de conhecimento e que as anotações semânticas, que venham a descrever a relação de recursos na *Web* e as instâncias de uma ontologia, sejam escritas utilizando o modelo RDF (BECHHOFER et al., 2004).

Na Figura 11, de Popov et al. (2003), está descrita a ideia de anotação semântica em conteúdo textual como a atribuição de links com a descrição semântica em entidades textuais. A ideia deste tipo de metadados é fornecer a ambos, classe e instância, informações sobre as entidades nos referidos documentos.

Oren (2006) destaca que existem várias ferramentas e paradigmas para a criação de anotações semânticas em recursos *Web*, sendo elas de forma manual, semi-automática ou totalmente automática.

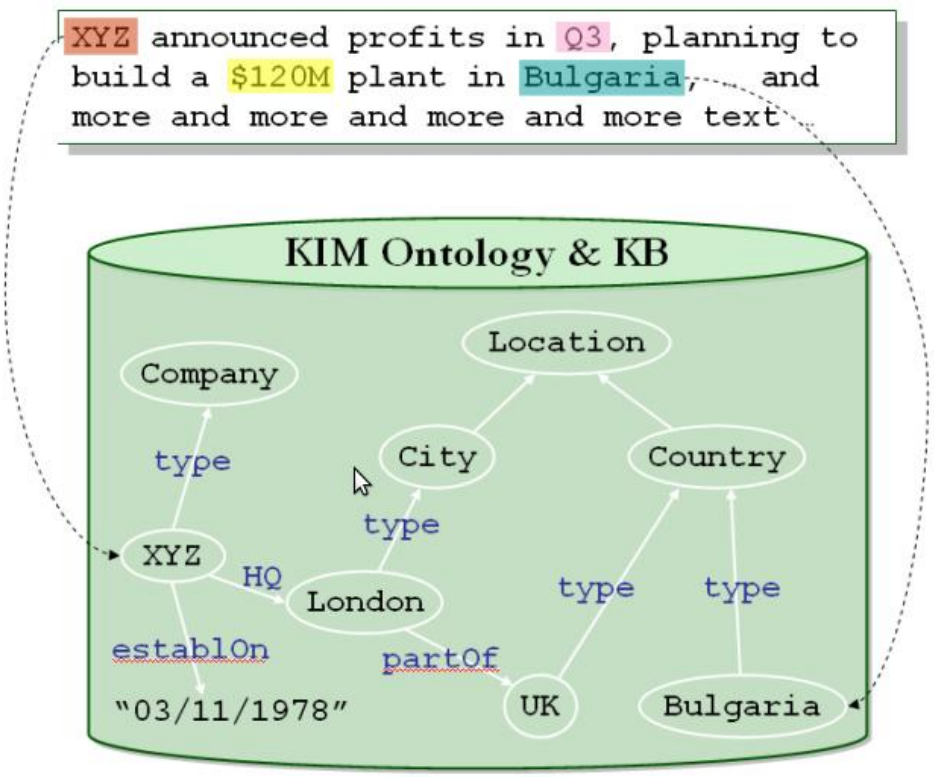

Figura 11: Anotação Semântica
Fonte: Popov et al. (2003).

2.7. WEB SERVICES

Web services ou serviços *Web* podem ser definidos como componentes independentes de aplicações disponíveis na *Web*, de modo que outras aplicações *Web* possam encontrá-los e utilizá-los. Suas características os tornam capazes de promover a flexibilidade e computação distribuída na Internet.

Sahai et al. (2002) descreve *Web services* como aplicações capazes de executar transações na Internet e serem acessados tanto pelos clientes como por outros *Web services*. Estes podem ser acessados de forma direta, permitindo sua maior utilização em aplicações de Internet.

Web services permitem uma forma de integrar sistemas distintos e modularizar serviços. Fornecem uma infraestrutura rica e estruturada de interoperabilidade entre clientes e servidores, provendo uma base por meio da qual um programa cliente em uma organização possa interagir com um servidor de outra organização, sem supervisão humana. Em particular, os serviços *Web* permitem o

desenvolvimento de aplicações complexas, fornecendo serviços que integram vários outros serviços. (COULOURIS et al., 2007)

Segundo Champion et. al. (2002), um *Web service* é um sistema de software identificado por um *Uniform Resource Identification* (URI), cujas interfaces públicas e ligações são definidos e descritos usando XML. Sua definição pode ser descoberta por outros sistemas. Estes sistemas interagem com o *Web service* através da sua definição, realizando requisições e respostas formatadas geralmente em XML que são transmitidas por protocolos de Internet, como por exemplo, o HTTP. Algumas informações adicionais precisam ser descritas para os *Web services*: a codificação, os protocolos de comunicação e o local do serviço. Estes serviços podem ser utilizados e organizados através do paradigma da Arquitetura Orientada a Serviço (SOA).

Web services fornecem uma interface comum que pode ser chamada a partir de praticamente qualquer linguagem de programação. Eles são o ponto-chave para integração de diferentes aplicações, sejam elas de diferentes plataformas, linguagens ou sistemas, uma vez que são baseados em um conjunto de padrões que as tornam independentes das tecnologias utilizadas para a sua prestação. (CASTILLO ET AL., 2013)

Para Ray e Kulchenko (2002), os *Web services* projetados e construídos com padrões abertos possibilitam que funcionalidades de aplicações sejam exportadas para outras aplicações. Segundo Martin et al. (2003), os *Web services* permitem que aplicações possam interoperar por meio de padrões *Web*.

Várias tecnologias como a SOAP, REST ou XMLRPC (XMLRPC, 2014) são utilizadas no desenvolvimento de *Web services*. As mais populares são a SOAP (GUDGIN, 2007) e REST (FIELDING, 2000), que será detalhado neste trabalho.

Para Castillo et al. (2013) seja qual for a tecnologia utilizada para implantar serviços *Web*, eles oferecem várias vantagens, como independência de linguagem e os mecanismos de distribuição. Também aumentam a interoperabilidade entre diferentes elementos de software. Por exemplo, é possível adicionar bibliotecas de comunicação sem modificar o código existente, facilitando a distribuição de código entre as equipes de trabalho.

2.7.1. Combinação de Web Services

Os *Web services* podem ter suas operações combinadas com as de outros serviços, fornecendo, assim, novas funcionalidades. Por exemplo: Algumas pessoas podem utilizar seus navegadores *Web* para fazer reserva de hotel, compra de passagens aéreas, reserva de aluguel de carro com uma seleção de sites diferentes. No entanto, se estes sites fornecerem uma interface de *Web service* padrão, é possível que um serviço de Agência de viagens utilize suas operações para prover uma combinação desses serviços ao cliente, em um único *Web site*, conforme ilustrado na Figura 12.

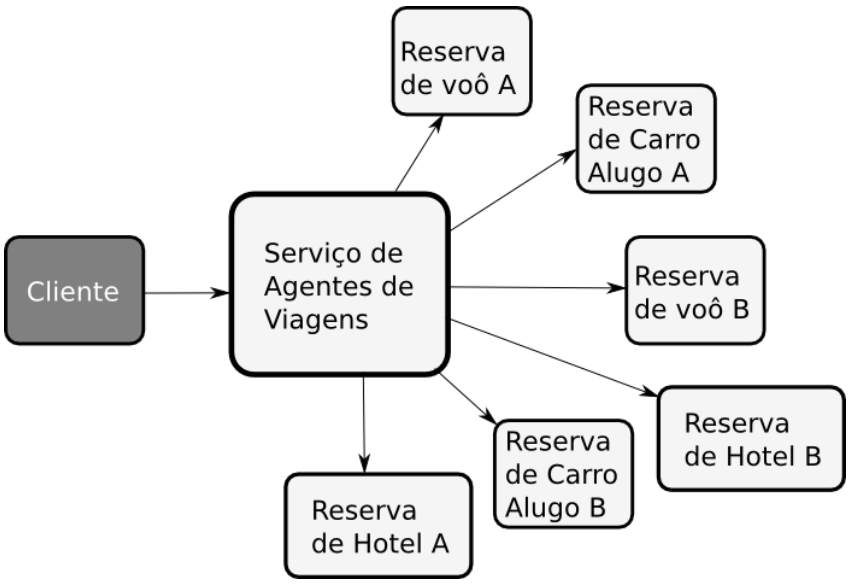

Figura 12: Combinação (Orquestração) de serviços em agência de viagens
Fonte: Adaptado de Coulouris et. al (2007).

Na Figura 12, Uma Agência de Viagens combina *Web services* para fornecer aos clientes vários serviços em um único local. Neste exemplo, as reservas são efetuadas através de uma comunicação assíncrona, onde é realizada a troca de mensagens, informações sobre data e destinos até confirmação da reserva. Enquanto que o pagamento das operações com cartão de crédito é realizado de forma síncrona, com confirmação imediata da operação.

Os *Web services* utilizam geralmente um padrão de comunicação de

requisição e resposta síncrona ou assíncrona, mas também pode utilizar um padrão de comunicação estilo evento, quando o cliente é notificado da ocorrência de um determinado evento. Por exemplo: os clientes de um serviço de diretórios podem se registrar nos eventos de interesse e então, serem notificados quando certos eventos estão para ocorrer. Um evento pode ser a chegada ou a saída de um serviço. (COULOURIS et al., 2007)

2.7.2. Arquitetura Orientada a Serviços

A arquitetura básica de *Web services* é uma arquitetura orientada a serviços (*Service-Oriented Architecture* – SOA) que se baseia na interação de três entidades e operações. As entidades desempenham funções de provedor do serviço, requisitante do serviço e registrador do serviço. A arquitetura de *Web services* relaciona vários componentes e tecnologias capazes de trocar mensagens, descrever, publicar e localizar serviços *Web* (BOOTH et. al., 2004). As operações que atuam sobre os componentes da arquitetura são: publicar (*publish*), encontrar (*find*) e interagir (*interact*). A arquitetura SOA é baseada nos princípios da computação distribuída e utiliza o paradigma *request/reply* para estabelecer a comunicação entre os sistemas clientes e os sistemas que implementam os serviços. (KODALI, 2005). A Figura 13 ilustra a arquitetura básica de *Web services*, onde as interações ocorrem entre o Provedor do Serviço (*Service Provider*) que disponibiliza um serviço, o Requisitante do Serviço (*Service Requestor*) que faz uso do serviço e o Registrador do Serviço (*Discovery Agencies*) onde os provedores publicam as descrições de seus *Web services*.

Um *Web* service é composto por dois componentes: o serviço e a descrição do serviço. (CHAMPION et. al., 2002) Um serviço pode ser chamado ou interagir com um solicitante de serviço. Pode funcionar como um solicitante, utilizando outros serviços *Web* na sua implementação.

A descrição de serviço contém os detalhes da interface e implementação do serviço (tipos de dados, operações, informações de conexão). Inclui a categorização e metadados para facilitar a descoberta e utilização pelos solicitantes. A descrição do serviço pode ser publicada diretamente para o solicitante ou para um registrador de

serviços (*Discovery Agencies*). (CHAMPION et. al., 2002)

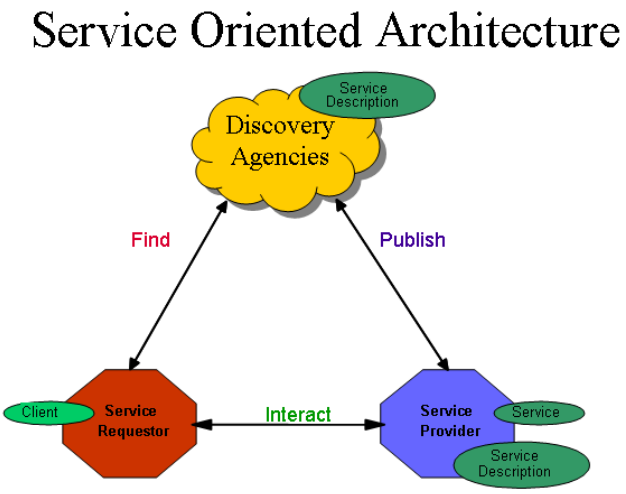

Figura 13: Arquitetura Básica de Web Services
Fonte: Champion et. al. (2002).

O provedor de serviços é o responsável por criar o *Web service* e disponibilizá-lo para que um cliente possa utilizar. Ele descreve o *Web service* em um formato padrão, que seja compreensível para qualquer cliente que precise usar o serviço e publica os detalhes sobre seu *Web service*. (CHAMPION et. al., 2002)

O requisitante do serviço, que também pode ser denominado de cliente, é qualquer entidade que utilize *Web service* fornecido por um provedor de serviços. Este conhece as funcionalidades do *Web service*, a partir da sua descrição (interface) disponibilizada pelo provedor de serviços. (CHAMPION et. al., 2002)

Um registrador de serviços ou *Discovery Agencies* é uma localização central onde os provedores de serviços podem relacionar seus *Web services*, e no qual um consumidor de serviços pode pesquisá-los. O registro dos serviços contém informações, tais como: detalhes de uma empresa, quais os serviços que ela fornece e a descrição técnica de cada um deles. (CHAMPION ET. AL., 2002)

2.7.3. REST

A Transferência de Estado Representacional (*REpresentational State Transfer* - REST), pode ser definido como um padrão arquitetural para projetos de *Web services* com foco no acesso aos recursos simples, identificados por URIs únicas, e sem estado, utilizando-se os métodos do protocolo HTTP (GET, POST, PUT e DELETE). Cada recurso pode ter uma ou mais representações (XML, JSON, Text, RDF, etc) as quais são transferidas entre o cliente e o serviço, durante a invocação ao método (KAMALELDIN e DUMINDA, 2012).

REST é um estilo de arquitetura para sistemas hipermídia distribuídos, como a World Wide *Web*. É um estilo híbrido derivado de vários estilos de arquitetura baseada em rede combinado com restrições adicionais que definem uma interface de conexão uniforme. (FIELDING, 2000)

O termo surgiu na tese de doutorado sobre a *Web*, escrita por Roy Fielding (*Architectural Styles and the Design of Network-based Software Architectures*) (FIELDING, 2000).

Em REST um conceito importante está na existência de recursos, onde qualquer informação que possa ser nomeada pode ser um recurso, ou seja, qualquer conceito que possa ser referenciado em hipertexto cabe dentro da definição de um recurso. Um recurso possui um identificador global para sua manipulação, chamados de URI (*Uniform Resource Identifier*) que identifica ou denomina um recurso na Internet. As representações de recursos são trocados através de uma interface padrão HTTP entre os componentes da rede (clientes e servidores). (FIELDING, 2000)

Várias restrições arquitetônicas são necessárias para orientar o comportamento dos componentes em REST, de maneira ser possível a construção de uma interface uniforme. Estas restrições são: identificação de recursos, manipulação de recursos através de representações, autodescrição de mensagens e hipermídia como motor de estado da aplicação.

REST utiliza um protocolo cliente/servidor sem estado, onde cada mensagem HTTP contém toda a informação necessária para compreender o pedido. Desta forma, o cliente e o servidor não necessitam armazenar o estado das comunicações entre mensagens.

Os sistemas que são desenvolvidos segundo os princípios REST são denominados *RESTful*.

3. ARQUITETURA PROPOSTA

Neste capítulo, será detalhada a arquitetura proposta neste trabalho, aplicada no desenvolvimento do protótipo *de Web service* que será apresentado no Capitulo 4.

Com esta arquitetura, será possível integrar diferentes ambientes de aprendizagem a uma interface única para anotação semântica de conteúdo. Esta arquitetura provê a integração de diversos *Web* services e ferramentas, que auxiliam no processo de anotação semântica de conteúdo, armazenamento, consulta e publicação de dados na *Web* de Dados (*Web of Linked Data*).

A fim de facilitar a integração de diferentes plataformas, foi adotado o conceito de arquitetura orientada a serviços – SOA , de modo a permitir a agregação dos diferentes serviços, através do modelo de *Web services*. (NEWCOMER, 2002). Esta integração promove a intercomunicação de aplicações através do compartilhamento de métodos e lógicas de negócios. Esta abordagem agiliza o desenvolvimento de aplicações, pois o esforço na implementação de funcionalidades existentes em outros contextos é reduzido (LINTHICUM, 2003).

Para Hohpe & Woolf (2003), com a tecnologia de *Web services* é possível aos usuários o acesso, por uma interface consistente, a diversas operações por meio de uma requisição e resposta.

A arquitetura proposta possibilita a integração com outras aplicações, sejam elas ambientes de aprendizagem, bibliotecas digitais, dentre outras aplicações educacionais. Viabiliza o reuso de informação e a interoperabilidade entre as diversas plataformas de produção de recursos didáticos, potencializando, sobremaneira, o processo de desenvolvimento de objetos de aprendizagem.

3.1. VISÃO GERAL DA ARQUITETURA

O propósito geral desta arquitetura é prover uma interface para *Web service*, simples, interoperável, abstrata, com grande capacidade de reutilização, que proporcione uma visão integradora dos diversos serviços que apoiam o processo de anotação semântica de conteúdo em ambientes de aprendizagem. A interface provê a anotação semântica e buscas dos objetos de aprendizagem diretamente no ambiente, independente da plataforma, de forma transparente para o usuário, possibilitando a integração com outras ferramentas, tais como, outros ambientes de aprendizagem, bibliotecas digitais, dentre outras aplicações educacionais.

A arquitetura é composta por módulos que realizam a interface com os diversos serviços de anotação semântica de conteúdo, armazenamento, consulta e publicação de dados. Possui ainda, uma API (Application Programming Interface) que integra todos os módulos e provê acesso aos métodos e funções, de maneira a abstrair a implementação interna de cada componente da arquitetura. Também possui clientes para a API.

Esta abordagem permite que clientes para o *Web service* sejam desenvolvidos facilmente, independente da linguagem de programação ou plataforma adotada.

A forma de comunicação cliente-servidor é implementada via Transferência de Estado Representativo (REST), um padrão arquitetural para projetos de *Web services* onde o foco está no acesso aos recursos simples, identificados por URIs únicas, e sem estado, utilizando-se os métodos (GET, POST, PUT e DELETE) do protocolo HTTP.

A Figura 14 apresenta os diversos componentes da arquitetura, que serão detalhados a seguir.

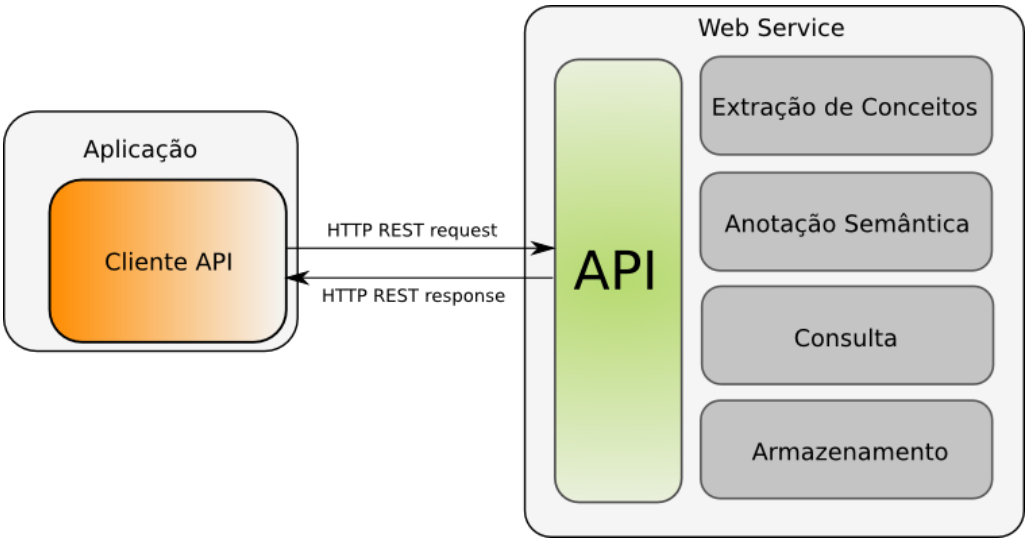

Figura 14 : Visão Geral dos Componentes da Arquitetura.

A API do *Web service*, proposto na arquitetura, permite a comunicação dos clientes com os serviços disponíveis, provendo um isolamento das ferramentas e *Web services* utilizados. Nela é implementado o controle de requisições (*request*) e respostas (*response*) aos clientes e definida a interface para os métodos e funções do *Web service*.

Os clientes da API são desenvolvidos em cada aplicação que deseja utilizar os serviços, fazendo com que ocorra um isolamento da camada lógica das aplicações clientes, tornando possível a modificação do processo de criação, busca e inserção de objetos.

O módulo de Extração de Conceitos realiza a extração de conceitos em conteúdo textual não estruturado utilizando técnicas de mineração de texto. Para tal é necessária a comunicação com os *Web services DBpedia Spotligth* (COELHO, 2013) e *DBpedia Lookup* (MENDES, 2013). A integração entre os *Web service*s é realizada por meio de um cliente REST desenvolvido de acordo com a API disponível nos serviços. A Figura 15 ilustra o processo de comunicação entre os clientes e os *Web services DBpedia Spotligth* e *DBpedia Lookup*.

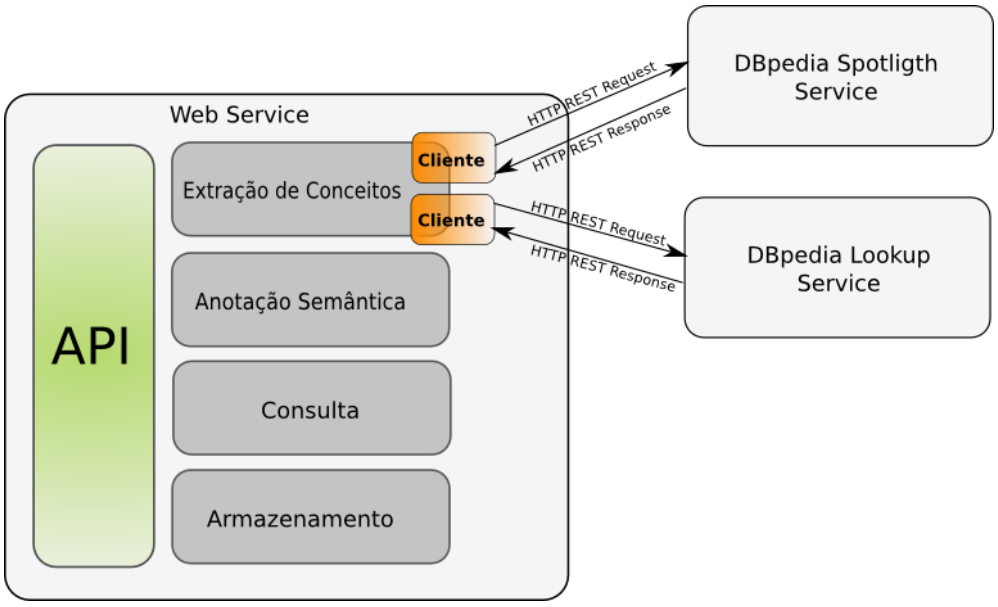

Figura 15: Comunicação entre os clientes e os Web services DBpedia Spotligth e DBpedia Lookup.

O módulo de Anotação Semântica transforma os conceitos extraídos em anotações semânticas, descritas em triplas RDF, e realiza a ligação destes conceitos com outras fontes de dados da *Web* de Dados, com o uso de ontologias específicas. A Figura 16 apresenta a relação entre o módulo e a *Web* de Dados, por meio do *hub* central DBpedia.

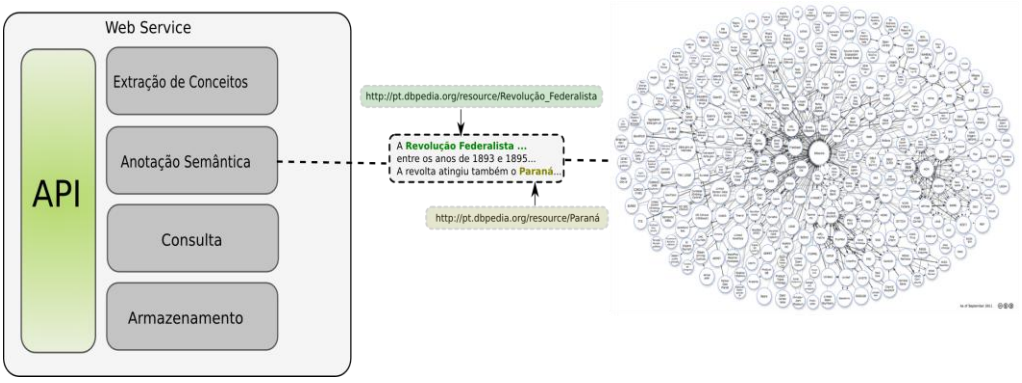

Figura 16: Relação entre o Módulo de Anotação Semântica e a Web de Dados.

Os módulos de Armazenamento e Consulta utilizam o repositório de triplas RDF (*RDF Triplestore*) *OpenRDF Sesame (*OPENRDF, 2014) para armazenamento das anotações semânticas e consultas ao repositório, por meio do protocolo *HTTP Sesame 2*, baseado em uma arquitetura REST, conforme apresentado na Figura 17.

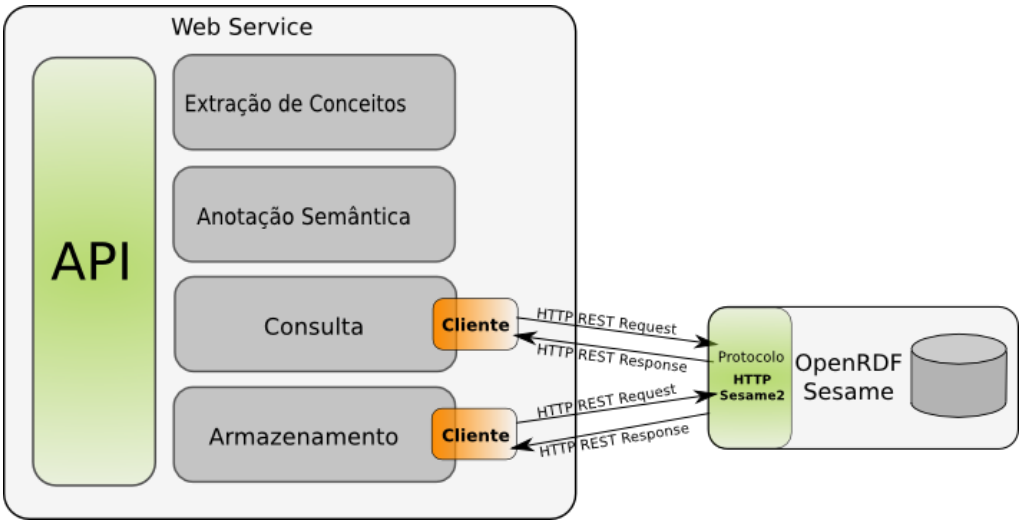

Figura 17: Comunicação dos Módulos de Armazenamento e Consulta com *OpenRDF Sesame.*

O módulo de Consulta disponibiliza uma interface de recuperação ao conteúdo anotado, por meio da tradução de consultas para a linguagem de consulta estruturada SPARQL, padrão da *Web* Semântica.

Na próxima seção será detalhado o funcionamento dos *Web services,* integrados à arquitetura proposta.

3.2. WEB SERVICES INTEGRADOS

Na arquitetura proposta, foram integrados os *Web services* DBpedia *Spotligth* e *DBpedia Lookup*, utilizados no módulo de Extração de Conceitos e o *Web service* OpenRDF Sesame, utilizado no módulo de Consulta e Armazenamento. O detalhamento é apresentado nos tópicos a seguir.

3.2.1. Dbpedia Spotligth

O *DBpedia Spotligth* é um projeto de código aberto desenvolvido para realizar a anotação automática de entidades da DBpedia presentes no texto em linguagem natural (MENDES et al., 2011). São identificados em fragmentos de textos (tais como documentos, parágrafos, frases) os recursos (entidades ou conceitos) presentes na DBpedia. A ferramenta analisa as palavras e expressões e retorna a URI do recurso correspondente na DBpedia. Na Figura 18, observa-se o resultado da anotação de

um parágrafo, onde são identificados conceitos e entidades (Revolução Federalista, político, Rio Grande do Sul, Paraná e Santa Catarina) e retornada a URI correspondente na DBpedia. Por exemplo, a entidade Santa Catarina tem a seguinte URI "http://pt.dbpedia.org/resource/Santa_Catarina" correspondente na DBpedia, que identifica o Estado de Santa Catarina, da região Sul do Brasil.

Figura 18: Ilustração do resultado de uma anotação de um parágrafo feito pelo DBpedia Spotlight.

Para Coelho (2013), a ferramenta *DBpedia Spotlight* fornece uma solução para a ligação de fontes de informações não estruturadas à nuvem da *Web* de Dados *(Linked Open Data cloud),* através da DBpedia.

O projeto foi iniciado por pesquisadores do grupo de pesquisa *Web Based Systems Group* da Universidade Freie *Universität Berlin (Free University of Berlin)* (W3C, 2011). Inicialmente o foco do projeto era a anotação de texto no idioma Inglês, no entanto, os modelos de aprendizagem do DBpedia Spotligth são baseados na Wikipedia, o que torna possível a adaptação da ferramenta para qualquer idioma que tenha edição na Wikipedia.(DAIBER et al., 2013)

A ferramenta possui recursos de identificação de palavras e expressões a serem anotadas e mecanismos de Desambiguação Lexical de Sentido (DLS) (AGIRRE & EDMONDS, 2007), de forma a prover uma interpretação não ambígua dos conceitos. O objetivo é reconhecer as entidades, a ambiguidade e o seu significado. Por exemplo, a palavra "Laranja" dependendo do contexto pode significar uma cor ou o nome de uma fruta. Mendes et al. (2011) destaca que o principal desafio na anotação é a ambiguidade, pois um recurso pode ter o mesmo nome, porém, aplicado em contextos diferentes, referem-se a recursos diferentes da

DBpedia. Ele afirma que a desambiguação para leitores humanos é feita em decisão baseada no conhecimento e no contexto em que o recurso é mencionado, o que torna a desambiguação automática um problema desafiador.

O algoritmo padrão de desambiguação é construído a partir dos conceitos de similaridade de cosseno e modificações de peso TF-IDF, utilizando o Apache Lucene[7]. O principal algoritmo utilizado para reconhecimento de *strings* é o LingPipe com a implementação Aho-Corasick. (DAIBER et al., 2013).

São passados parâmetros de configuração de suporte e confiança para o *DBpedia Spotlight* realizar o processo de anotação. O valor de confiança pode variar de 0 a 1, quanto maior o valor, mais confiável é a anotação. Portanto, o valor maior indica que será mais rigoroso o refinamento das anotações, resultando em menor número de anotações, todavia mais confiáveis. Contudo, valores menores de confiança resultam em maior número de anotações, porém susceptíveis a erros. O valor de confiança também é utilizado a fim de comparações com outros parâmetros internos, como o de “ambiguidade contextual”. O valor de suporte especifica o número mínimo de ligações internas da Wikipedia que um recurso deve ter para ser anotado, ou seja, o número mínimo de referências que uma determinada entidade deve possuir dentro da Wikipedia. Estas referências são links de páginas para uma determinada entidade. O valor pode variar de 0 até infinito, quanto maior o valor, as anotações retornam somente entidades mais populares.

O *DBpedia Spotlight* permite o uso de *whitelist* e *blacklist*, utilizadas, respectivamente, a fim de permitir ou negar determinadas palavras no processo de identificação de *strings* para anotação e consultas SPARQL para refinamento e filtro das anotações. Por exemplo, pode-se utilizar consultas SPARQL para definir que somente serão anotados conceitos ou entidades presentes no contexto de determinada ontologia ou explicitar que as anotações devem ser realizadas somente para determinados conceitos relacionados a uma entidade.

O retorno das anotações podem ser apresentados em diferentes formatos de arquivos (XML, JSON, RDF e HTML). A ferramenta pode ser utilizada de duas maneiras: por meio de um aplicativo *Web* com uma interface HTML + JavaScript,

[7] Disponível em: <http://lucene.apache.org/core/>. Acesso em: 15 out 2014.

disponível na página do projeto (Disponível em: <http://dbpedia-spotlight.github.io/demo/>. *Acess em: 12 out 2014*), ou por meio de um serviço *Web*, disponível em (Disponível em: <http://spotlight.sztaki.hu:2222/rest>. *Acess em: 12 out 2014*).

3.2.1.1. Aplicação Web

Na aplicação *Web* disponível na página do projeto, é possível um uso interativo da ferramenta. São permitidas as configurações dos parâmetros de suporte e confiança, seleção das classes de interesse contidas nas ontologias da DBpedia e filtragem por consultas SPARQL. O texto a ser anotado é inserido no formulário contido na página HTML e submetido ao *DBpedia Spotlight.* Este retorna o texto com as marcações das palavras identificadas e a associação da URIs correspondentes na DBpedia.

Na Figura 19 pode ser observado a interface HTML da ferramenta e o resultado de uma anotação.

Figura 19: Interface da Aplicação *Web* de anotação automática - *DBpedia Spotlight*.

3.2.1.2. Serviço Web

De maneira a facilitar a integração de serviços e aplicações *Web* aos recursos da *DBpedia Spotlight*, foi desenvolvido um serviço *Web* RESTful (MENDES et al., 2011). Este serviço permite o acesso aos recursos de anotação e desambiguação disponíveis na ferramenta.

A API do serviço possui quatro métodos: *Spotting, Disambiguate, Annotate* e *Candidates*. O método *Spotting* realiza a identificação dos conceitos ou entidades a serem anotados. Este é acessado pelo *endpoint* "/*spot*" e recebe como parâmetro o texto a ser anotado e o tipo algoritmo de mineração de texto implementado. Estão disponíveis para utilização os algoritmos: *LingPipeSpotter, AtLeastOneNounSelector, CoOccurrenceBasedSelector, NESpotter, KeyphraseSpotter, OpenNLPChunkerSpotter, WikiMarkupSpotter, SpotXmlParser, AhoCorasickSpotter* (MENDES et. al., 2014).

O método *Disambiguate* realiza a desambiguação das entidades e conceitos identificados para anotação. Este método recebe as entidades ou conceitos identificados e realiza a escolha do recurso da DBpedia que os representa em um determinado contexto.

O método *Annotate* realiza a identificação das entidades e conceitos, e a desambiguação. Este método escolhe dentre as entidades ou conceitos identificados, o recurso da DBpedia, dado o contexto apresentado. São passados os parâmetros de suporte e confiança, assim como filtros de consulta SPARQL, nos casos onde se necessita a criação de restrições ou refinamento dos resultados. O retorno deste método pode ser observado no fragmento de XML apresentado na Figura 20, onde é realizada a anotação de um trecho de texto.

```
<?xml version="1.0" encoding="utf-8"?>
<Annotation text="A Revolução Federalista foi...
                  ... A revolta atingiu também o Paraná e Santa Catarina."
                  confidence="0.4" support="20"
                  types="" sparql="" policy="whitelist">
  <Resources>
    <Resource URI="http://pt.dbpedia.org/resource/Revolução_Federalista"
              support="256" types="" surfaceForm="Revolução Federalista"
              offset="2" similarityScore="0.9999916145866622" percentageOfSecondRank="8.385483652903284E-6"/>
    ...
    </Resources>
</Annotation>
```

Figura 20: Fragmento de XML representando uma anotação de texto realizada pelo serviço.

O método *Candidades* retorna uma lista de entidade ou conceitos candidatos, identificados para a anotação. Esta lista forma um ranking dos candidatos e possui propriedades que ajudam na escolha do recurso que melhor representa uma entidade ou conceito.

3.2.2. Dbpedia Lookup Service

O *DBpedia Lookup* é um *Web service* que possibilita realizar buscas por URIs da DBpedia relacionadas a palavras-chave, ou seja, é possível buscar a URI de determinado recurso por meio das palavas-chave ou textos âncoras, utilizados para encontrar o recurso na Wikipedia. Por exemplo, é possível encontrar o recurso "*http://dbpedia.org/resource/United_States*" utilizando as palavras "USA", "US" ou "*United States*" (MENDES et al., 2013).

Segundo Kobilarov et al. (2009), o *DBpedia Lookup* tem como base um índice customizado do servidor Lucene que combina métricas de relevância e classificação por *string-similarity-based* do Lucene.

Os resultados da busca são ordenados utilizando os números de *inlinks* (linhas de entrada) que apontam, a partir de outras páginas da Wikipedia, para a página encontrada como resultado.

O *Web service* disponbiliza duas APIs: *Keyword Search* e *Prefix Search*.

A API *Keyword Search* é utilizada para encontrar recursos da DBpedia relacionados a uma única palavra ou sequência de palavras.

A API *Prefix Search* realiza a busca em palavras-chave não completas, ou seja, pode-se informar somente parte da palavra-chave e são retornadas as URIs de recursos relacionados. Por exemplo, é informada a palavra "Bras" e são retornados recursos relacionados, como "*http://dbpedia.org/resource/Brazil*", "*http://dbpedia.org/resource/ Brasília*", dentre outros. Segundo Mendes et al. (2013) esta API pode ser utilizada para a implementação de campos de texto com *autocomplete* (*autocomplete input boxes*).

As APIs do *Web service* recebem três parâmetros, sendo eles: *QueryString, QueryClass e MaxHits*. O parâmetro *QueryString* recebe a palavra a ser buscada, o *QueryClass* recebe a classe da ontologia da DBpedia que o resultado deve

pertencer, e *MaxHits*, o número máximo de resultados retornados.

No trecho a seguir temos um exemplo de requisição ao método *KeywordSearch* disponível no servidor "*lookup.dbpedia.org*", onde são retornados todos os recursos pertencentes a classe *"place"* da ontologia da DBpedia que contém a palavra-chave "Brasil".

```
http://lookup.dbpedia.org/api/search/KeywordSearch?QueryClass=place&
QueryString=Brasil
```

O resultado da busca é retornado por padrão no formato XML, mas também está disponível o retorno no formato JSON, sendo, neste caso, necessário informar no cabeçalho da requisição ao serviço o tipo de retorno desejado.

3.2.3. OpenRDF Sesame

Segundo Broekstra et al. (2001), OpenRDF *Sesame* é um *framework* da *Web* Semântica para armazenamento e consulta de dados e esquemas RDF. Ele é um projeto de código aberto que permite inserir, apagar, atualizar e consultar dados em RDF. Possui uma API HTTP *Restful* para suporte ao protocolo SPARQL. (OPENRDFa, 2014)

Sua estrutura de *framework* permite que sejam utilizados diversos mecanismos de armazenamento e inferência, além do que permite a extensão de seus componentes.

O armazenamento pode ser realizado em banco de dados relacionais, sistema de arquivos, em memória, dentre outros, pois o *Sesame* fornece uma API para abstração das funcionalidades do *framework*, quanto ao armazenamento utilizado. Está API é denominada SAIL.

As consultas podem ser executadas na linguagem SPARQL ou no formato próprio do framework, a linguagem SeRQL.

O *Sesame* pode ser executado como um *Web service*, funcionando como um banco de dados de triplas RDF (RDF Triplestore). Possui uma interface gráfica e um console por linhas de comandos. (FIRMINO et al., 2014)

A arquitetura do OpenRDF *Sesame* é apresentada na Figura 21, onde são demonstrados os componentes e as APIs mais importantes do *framework*. Também mostra como eles são construídos, um em cima do outro. Cada API ou componente depende diretamente da API ou componente que está abaixo na arquitetura. (OPENRDFa, 2014)

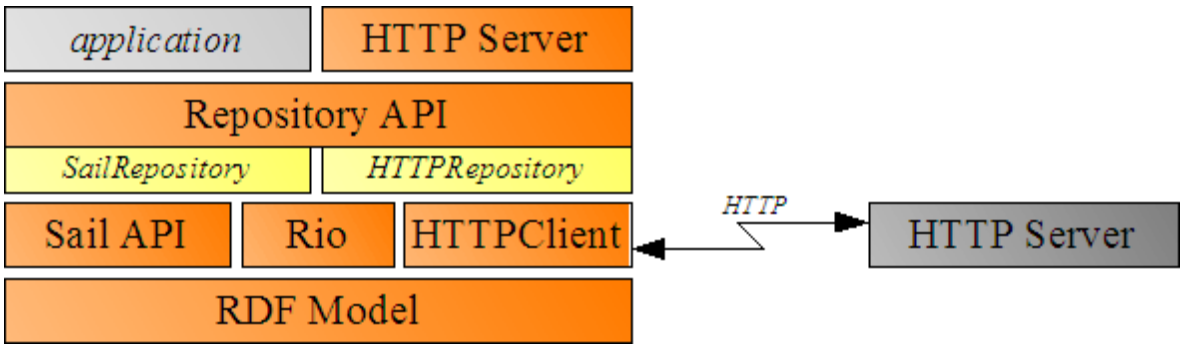

Figura 21: Arquitetura de alto nível dos componentes e APIs mais importantes do OpenRDF *Sesame*
Fonte: OpenRDFa (2014).

O componente *RDF model* é a base do *framework Sesame*, devido sua implementação ser orientada a RDF, todas as partes do *Sesame* são extensões do modelo RDF, onde é definida a interface e implementação de todas as entidades básicas do RDF, sendo elas: URI, nó em branco, literal e declarações (*statements*).

O componente *Rio*, abreviação de RDF I/O (Entrada/Saída), consiste em um conjunto de *parses* e *writers* para vários formatos de arquivos RDF. Os *parses* podem ser utilizados a fim de traduzir um arquivo RDF para um conjunto de declarações, e *writers* realizam a operação inversa, ou seja, usado para escrever arquivos RDF.

A API SAIL (*Storage And Inference Layer*) é uma API de baixo nível para armazenamento RDF e inferência. Ela abstrai os detalhes de armazenamento e inferência, permitindo que diversos tipos sistemas de armazenamento e inferência sejam utilizados. Segundo Broekstra et al. (2001), uma vantagem da arquitetura em camadas aplicada ao *Sesame* é tornar possível a implementação do armazenamento de diferentes formas sem mudar qualquer outro componente. Existem diversas implementações para a API SAIL, por exemplo, o armazenamento de dados RDF em memória (*MemoryStorage*), o *NativeStorage* que utiliza uma

estrutura especializada para armazenamento dos dados em disco e o armazenamento em banco de dados relacionais, como por exemplo o *MySQL RDF Store* e *PostgreSQL RDF Store* (OPENRDFa, 2014) que são baseados nos Sistemas de Gerenciamento de Banco de Dados (SGBD) *MySQL (*MYSQL, 2014*)* e *PostgreSQL* (POSTGRESQL, 2014).

O *Repository API* é uma API alto nível que disponibiliza diversos métodos para manipulação de dados RDF. Está API permite que sejam desenvolvidos aplicativos mais facilmente, pois oferece métodos para *upload* de arquivos, consulta, extração e manipulação de dados. Existem diversas implementações para esta API. Na Figura 21 são apresentadas duas, a *SailRepository*, utilizada para repositórios locais, que traduz chamadas para a implementação SAIL escolhida (*MemoryStorage*, *NativeStorage*, *MySQL RDF Store*, Postgre*SQL RDF Store* dentre outras), e a *HTTPRepository*, utilizada para repositórios remotos, que oferece uma comunicação cliente-servidor transparente com o *Sesame*, utilizando o protocolo HTTP. (OPENRDFb, 2014)

Na arquitetura proposta neste trabalho foi utilizado o armazenamento local *NativeStorage* acessado por meio da API *SailRepository*. Está escolha foi motivada pelo fato de ser uma implementação simples e por possuir suporte nativo a mecanismos de inferência disponível no *Sesame*. A desvantagem do armazenamento *NativeStorage*, frente a outras formas de armazenamento está na escalabilidade, uma vez que para pequenos e médios *datasets* o desempenho é satisfatório, porém em grandes *datasets* é recomendado a utilização de outra alternativa. (OPENRDFc, 2014)

No topo da arquitetura está o componente HTTP *Server*, que consiste de *Java Servlets* que implementam um protocolo para prover acesso ao repositório *Sesame* com o uso do protocolo HTTP.

O servidor *Sesame* possibilita uma comunicação baseada na arquitetura REST. A API REST disponível, também conhecida como *Sesame 2 HTTP communication protocol* é compatível com as recomendações da W3C *SPARQL Protocol for RDF* (CLARK et al., 2008), *SPARQL 1.1 Protocol for RDF* (FEIGENBAUM et al. 2013) e *SPARQL 1.1 Graph Store HTTP Protocol* (OGBUJI, 2013). A implementação destas recomendações, fazem com que o *Sesame* possua as características de um *SPARQL endpoint.*

O acesso ao servidor *Sesame* também pode ser realizado por meio do aplicativo *Web OpenRDF Workbench*, que possibilita a realização de buscas, atualizações e exploração do repositório *Sesame* por uma interface *Web* (OPENRDFa, 2014). A Figura 22 apresenta a interface a interface *Web* do *OpenRDF Workbench* na funcionalidade de listagem de repositórios existentes.

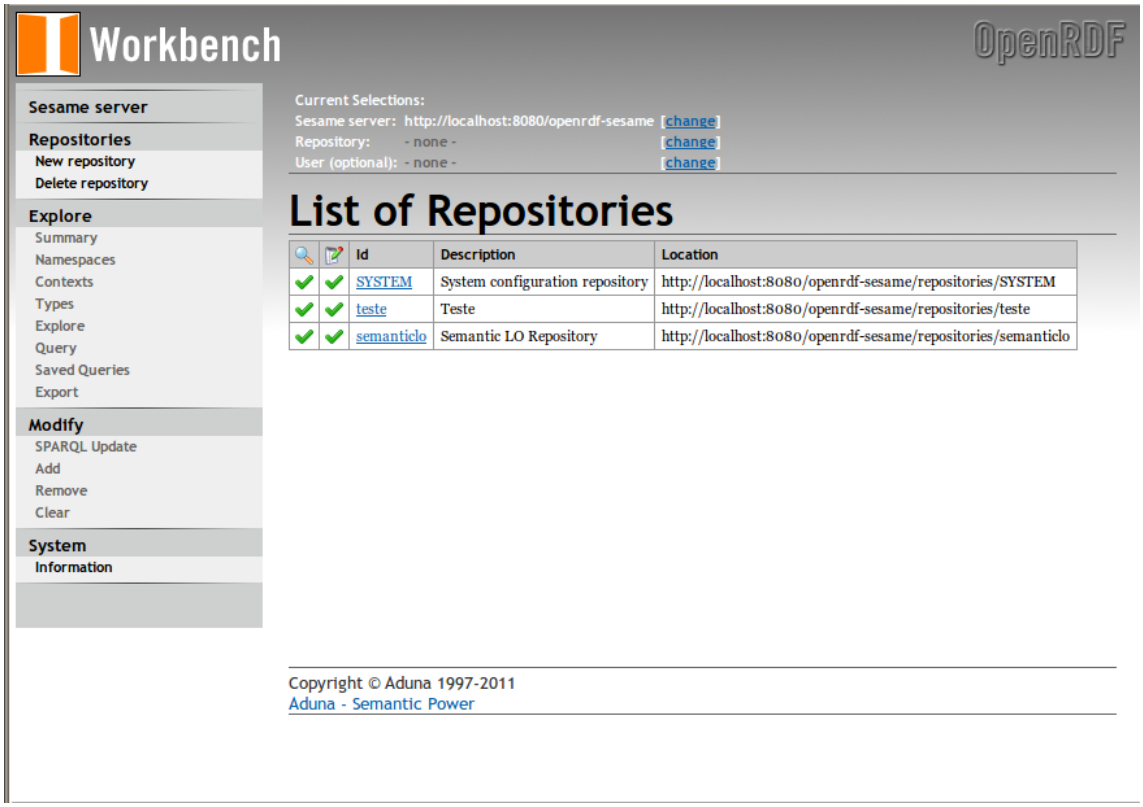

Figura 22: Aplicativo Web Sesame OpenRDF Workbench.

4.PROTÓTIPO DESENVOLVIDO E ESTUDO DE CASO

Nesta seção, será apresentado o protótipo desenvolvido baseado na Arquitetura Orientada a Serviços – SOA proposta no Capítulo 3, que visa realizar a extração de conceitos, anotação semântica, consulta e armazenamento em repositório RDF. Inicialmente, será feita a descrição do *Web service* desenvolvido e em seguida a implementação do estudo de caso no ambiente de aprendizagem. O objetivo deste estudo de caso é aplicar a utilização dos serviços construídos ao ambiente de aprendizagem Moodle na versão 2.4 e à fonte de dados do DBpedia.

4.1. PROTÓTIPO DO WEB SERVICE

O protótipo utiliza a forma de comunicação cliente-servidor implementado via Transferência de Estado Representativo (REST).

A base do *Web service* é composta pelos seguintes módulos:

- Servidor *RESTful* – responsável por prover as APIs necessárias e o controle aos acessos aos diversos módulos do serviço.
- Extração de Conceitos – responsável por extrair conceitos de conteúdo textual não estruturado utilizando técnicas de mineração de texto.
- Anotação Semântica – responsável pela transformação dos conceitos extraídos em anotações semânticas e a ligação destes conceitos com a DBpedia. Armazenamento – responsável por persistir os dados e as anotações semânticas em repositório específico para armazenamento de triplas

- RDF – RDF *Triplestore*
- Consulta – responsável por disponibilizar uma interface amigável de recuperação ao conteúdo anotado, utilizando, sem que o usuário se dê conta, de forma encapsulada, a linguagem de consulta estruturada padrão da *Web* Semântica SPARQL .

A Figura 23 apresenta uma visão geral da arquitetura do *Web service* e sua integração com os componentes do sistema, conforme apresentado no Capítulo 3.

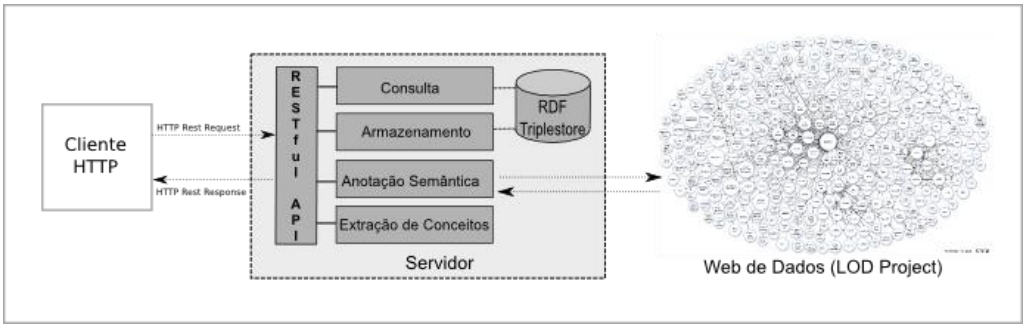

Figura 23: Visão Geral da Arquitetura do Web Service.

A seguir será detalhado o funcionamento de cada módulo do Web service

4.1.1. Servidor RESTful

O servidor RESTFul desenvolvido foi denominado de **semantic-lo**, sua implementação foi na linguagem de programação *Python* e todo o código fonte do projeto está disponível em *https://github.com/ronaldoamaral/semantic-lo,* de forma a facilitar as contribuições externas ao projeto e estimular seu desenvolvimento pela comunidade e implementação de projetos futuros.

A base do servidor foi desenvolvida com o framework *Flask* (RONACHER, 2014a), um micro-framework para desenvolvimento *Web* escrito em *Python,* com características simples, mas extensível. Seus pilares são as bibliotecas *Werkzeug* (RONACHER, 2014b) e *Jinja2* (RONACHER, 2014c).

Werkzeug é uma biblioteca para desenvolvimento de aplicações conforme a especificação WSGI (*Web Server Gateway Interface*) (EBY, 2011), que descreve como servidores Web se comunicam com aplicações Web desenvolvidas em

Python. Esta biblioteca possui total suporte a manipulação de *request* e *response* HTTP, controle de *cache*, *cookies*, *status* HTTP, *upload* de arquivos, sistema de roteamento de URL e ferramenta de *debug* (ROCHA, 2014).

Jinja2 é uma *engine* de *templates* completa para *Python*, com total suporte a *Unicode*, inspirada no sistema de *templates* do *framework Django*, porém com mais funcionalidades e com implementações mais flexíveis (HIDEKI, 2014).

Foi adicionando ao framework *Flask* a extensão *Flask-RESTful* (BURKE et al., 2014) disponível em *http://flask-restful.readthedocs.org/en/latest/* que adiciona suporte a construção de APIs REST.

O servidor disponibiliza uma API para acesso as diversas funcionalidades da aplicação. Conforme os princípios de aplicações REST, esta API é baseada na manipulação do recurso objeto de aprendizagem (LO), apresentado no Quadro 3, onde são descritas brevemente as ações da API disponível no serviço.

Método HTTP	**URI**	**Ação**
POST	http://[hostname]/los	Adiciona um novo LO
GET	http://[hostname]/los	Realiza busca nos LOs
GET	http://[hostname]/los/*{id}*	Acessa as informações do LO
DELETE	http://[hostname]/los/*{id}*	Remove um LO
PUT	http://[hostname]/los/*{id}*	Atualiza os metadados de um LO existente
GET	http://[hostname]/los/*{id}*/tags_candidates_resources	Realiza a extração de conceitos de um LO e apresenta os recursos candidatos para o *marshup*
GET	http://[hostname]/los/*{id}*/tags_resources	Recupera as marcações de um LO
PUT	http://[hostname]/los/*{id}*/tags_resources	Atualiza as marcações de um LO
GET	http://[hostname]/los/*{id}*/subjects_candidates_resources	Identifica as entidades ou conceitos presentes em palavras-chaves
GET	http://[hostname]/los/*{id}*/subjects_resources	Recupera as palavras-chaves de um LO
PUT	http://[hostname]/los/*{id}*/subjects_resources	Atualiza as palavras-chaves de um LO

Quadro 3: Descrição da API disponível no Web service.

Na Figura 24, é apresentado o diagrama de classe do servidor com o relacionamento dos recursos e métodos disponíveis na API. A seguir será detalhada cada classe, e suas funcionalidades.

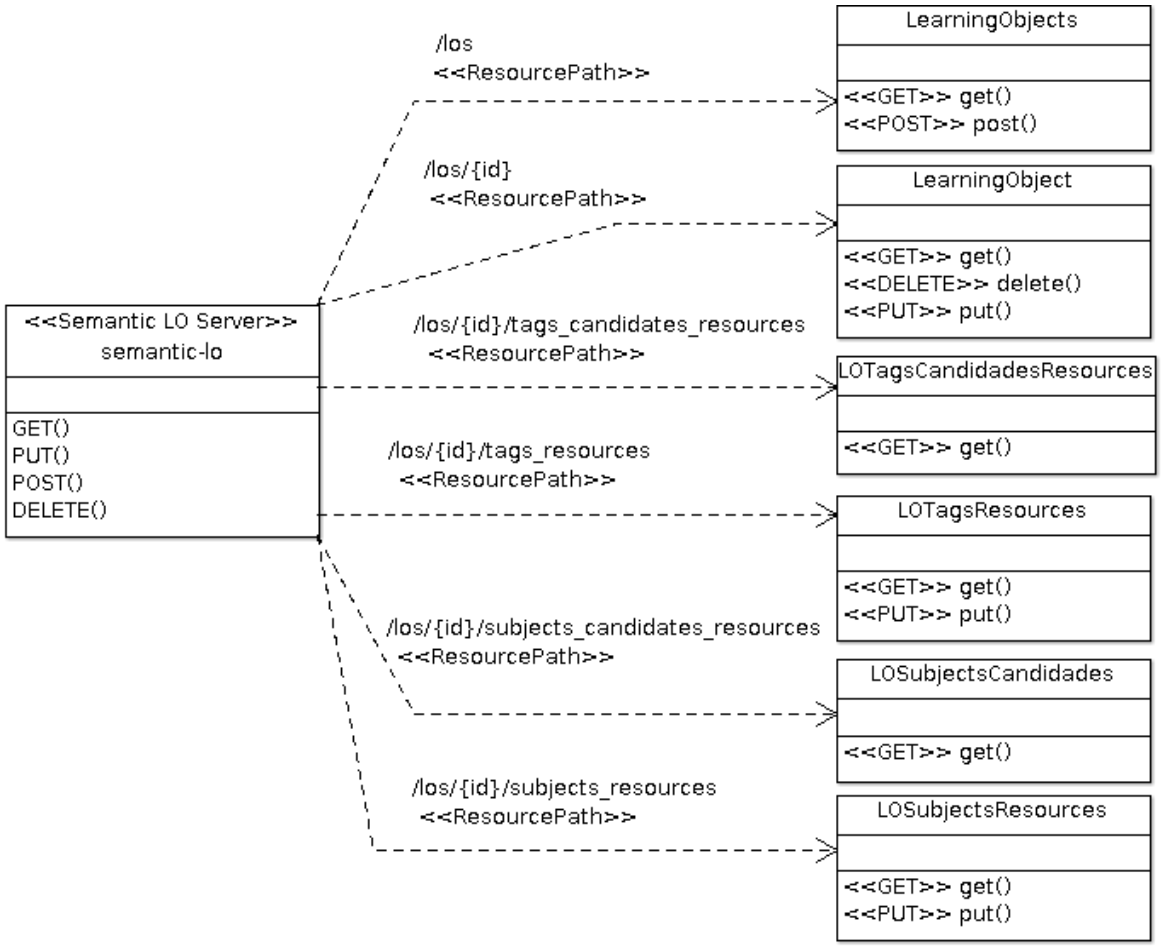

Figura 24: Diagrama de classes servidor RESTful semantic-lo.

4.1.1.1. Adição e busca de objetos de aprendizagem

A interface para adição e busca de objetos de aprendizagem é implementada pela classe *LearningObjects*, por meio dos métodos HTTP *post* e *get*. Esta classe é responsável por: disponibilizar uma interface para persistência em banco de triplas RDF (*RDF Triplestore*) da representação do objeto de aprendizagem (LO); mapeamento dos metadados por meio da integração com o módulo de anotação semântica; busca no repositório de triplas RDF por meio da integração com o módulo de consulta.

A adição de LOs é invocada pela operação HTTP *post* à URI "*http://[hostname]/los*", onde é esperado um *request* no formato JSON com os

parâmetros: titulo (*title*), identificador (*identifier*), descrição (*description*), marcações (*annotations*) e palavras-chaves (*subjects*). Estes parâmetros são definidos conforme metadados mapeados pelo módulo de anotação semântica. Na seção 4.1.3 é detalhado este mapeamento. Após o processamento dos parâmetros, é instanciada a representação do objeto de aprendizagem e associado os metadados aos parâmetros recebidos.

Após o tratamento da requisição, é retornado ao cliente, no formato JSON, o *status* da operação e a URI do LO armazenado. Na Figura 25 é demonstrado o código fonte da classe *LearningObjetcs*.

```
131 class LearningObjects(Resource, LearningObjectUtils):
132
133     def post(self):
134         """ """
135         data = {}
136         if request.json.get('title'):
137             data['title'] = request.json['title']
138         if request.json.get('identifier'):
139             data['identifier'] = request.json['identifier']
140         if request.json.get('description'):
141             data['description'] = request.json['description']
142         if request.json.get('annotations'):
143             data['annotations'] = request.json['annotations']
144         if request.json.get('subjects'):
145             data['subjects'] = request.json['subjects']
146
147         obj = LO(repository_lo[convert_to_uri(data['title'])])
148         response_data = {"status": "",
149                          "data": {"uri": obj.resUri },
150                          "message": "null" }
151         if self._setmetadata(obj, data):
152             headers = {'Location': obj.resUri}
153             response_data['status'] = "success"
154             return response_data, 201, headers
155         else:
156             response_data['status'] = "error"
157             response_data['message'] = "error adding learning object"
158             return response_data, 500
159
160     def get(self):
161         """ """
162         title = request.args.get('title', '')
163         result = search.title(title)
164         data_js = json.loads(result.read())
165         return data_js, result.getcode()
```

Figura 25: Código fonte classe LearningObjects.

A busca no repositório de triplas RDF é invocada por meio da operação HTTP *get* à URI *http://[hostname]/los*, com a passagem do metadado a ser buscado e a *string* de busca. Neste protótipo, foi implementada a busca para o metadado *title*.

Após o processamento dos parâmetros recebidos, o método *get* submete ao módulo de consulta a *query* de busca e retorna ao cliente um arquivo no formato JSON com o resultado da requisição.

4.1.1.2. Atualização, remoção e recuperação de objetos de aprendizagem

A interface para atualização, remoção e recuperação dos objetos de aprendizagem é implementada pela classe *LearningObject*, por meio dos métodos HTTP *put*, *delete* e *get*.

A atualização dos objetos é invocada pela operação HTTP *put* à URI do recurso *"http://[hostname]/los/{id}",* onde "*{id}"* representa a identificação do objeto a ser atualizado. Este recebe como parâmetros os metadados a serem atualizados e armazena a descrição RDF atualizada do objeto, conforme definido no módulo de anotação semântica.

A remoção dos objetos é invocada por meio da operação HTTP *delete* à URI do recurso a ser removido. Este realiza a operação de remoção do objeto do banco e retorna ao cliente o HTTP *status code* de sucesso ou erro da operação.

A recuperação dos objetos é invocada por meio da operação HTTP *get* ao recurso. Este método implementa a negociação de conteúdo com o cliente, sendo possível a recuperação do objeto na representação XML/RDF ou JSON-LD (SPORNY et. al., 2013), uma representação JSON para *Linked Data*.

A negociação de conteúdo é realizada com a passagem do parâmetro *accept*, no cabeçalho da requisição HTTP, onde ao ser passado o *mimetype application/rdf+xml* é retornado ao cliente a representação XML/RDF do objeto, e ao ser passado o *mimetype application/ld+json* é retornada a representação JSON-LD do objeto.

No trecho de código "*curl -H 'accept:application/rdf+xml' http://localhost:5000/los/historia-do-brasil.rdf*" é demonstrado com o uso do comando *Linux/Unix curl*[8], uma requisição ao recurso "historia-do-brasil" com a passagem do parâmetro *accept* com o valor *application/rdf+xml* de modo a retornar a representação XML/RDF do objeto. Na Figura 26 é demonstrado o retorno no

[8] Disponível em: <http://curl.haxx.se/>. Acesso em: 20 ago 2014.

formato RDF/XML do objeto acessado onde são descritas as triplas RDF do recurso conforme notação RDF/XML, com o tipo do recurso, título, identificador, descrição, palavras-chaves e as marcações associadas ao recurso.

```
<?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:sesame="http://www.openrdf.org/schema/sesame#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:fn="http://www.w3.org/2005/xpath-functions#">

<rdf:Description rdf:about="http://localhost:8080/semanticlo/resource/historia-do-brasil">
  <rdf:type rdf:resource="http://ltsc.ieee.org/rdf/lomv1p0/lom#LearningObject"/>
  <title xmlns="http://purl.org/dc/terms/">Historia do Brasil</title>
  <identifier xmlns="http://purl.org/dc/terms/">www.youtube.com/watch?v=wThhu2cqRRU</identifier>
  <description xmlns="http://purl.org/dc/terms/">Documentário com animação e locução contando a história do Brasil,resumido em 15 minutos,
a contar de1822 aos dias de hoje, abordando o desenvolvimento, politica, arte, imigração etc...!</description>
  <subject xmlns="http://purl.org/dc/terms/" rdf:resource="http://dbpedia.org/resource/Brazil"/>
  <subject xmlns="http://purl.org/dc/terms/" rdf:resource="http://dbpedia.org/resource/History"/>
</rdf:Description>

<rdf:Description rdf:about="http://localhost:8080/semanticlo/tagging/ad5f256f-3a0c-4b52-9755-43304f287e8b">
  <taggedResource xmlns="http://purl.org/muto/core#" rdf:resource="http://localhost:8080/semanticlo/resource/historia-do-brasil"/>
</rdf:Description>

</rdf:RDF>
```

Figura 26: Arquivo RDF/XML de retorno do método control via get.

No Quadro 4, são apresentadas as triplas RDF (sujeito, predicado e objeto) descritas no RDF/XML da Figura 26, onde o sujeito é representado pela URI do recurso, o predicado é a propriedade, e objeto é o valor.

Sujeito	Predicado	Objeto
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://www.w3.org/1999/02/22-rdf-syntax-ns#type	http://ltsc.ieee.org/rdf/lomv1p0/lom#LearningObject
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://purl.org/dc/terms/title	"Historia do Brasil"
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://purl.org/dc/terms/identifier	"www.youtube.com/watch?v=wThhu2cqRRU"
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://purl.org/dc/terms/description	"Documentário com animação e locução contando a história do Brasil,resumido em 15 minutos, a contar de1822 aos dias de hoje, abordando o desenvolvimento, politica, arte, imigração etc...!"
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://purl.org/dc/terms/subject	http://dbpedia.org/resource/Brazil
http://localhost:8080/semanticlo/resource/historia-do-brasil	http://purl.org/dc/terms/subject	http://dbpedia.org/resource/History
http://localhost:8080/semanticlo/tagging/ad5f256f-3a0c-4b52-9755-43304f287e8b	http://purl.org/muto/core#taggedResource	http://localhost:8080/semanticlo/resource/historia-do-brasil

Quadro 4: Triplas RDF descritas no RDF/XML da Figura 26.

Na segunda linha do Quadro 4, temos o tipo do recurso, identificado pela propriedade *type* do RDF. Na terceira linha, temos o título do recurso identificado

pela propriedade *title* do *Dublin Core*. Na quarta linha, temos o identificador do recurso, conforme propriedade *identifier* do *Dublin Core*. Na quinta linha a descrição identificada pela propriedade *description* do *Dublin Core*. Na sexta e sétima linhas são apresentadas as palavras-chaves identificadas pela propriedade *subject* do *Dublin Core*, onde é realizado o *marshup* com os recursos presentes na *DBpedia*. Neste exemplo, o recurso apresenta as palavras-chaves Brasil e Historia, associadas respectivamente com os recursos da DBpedia *http://dbpedia.org/resource/Brazil* e *http://dbpedia.org/resource/History*. Na última linha, temos a identificação das marcações, conforme ontologia específica, onde o recurso que identifica as marcações realizadas é apresentado como sujeito. Na seção 4.1.3 Anotação Semântica, serão detalhados os metadados e ontologias utilizadas neste trabalho.

4.1.1.3. Extração de conceitos candidatos

A interface de extração dos conceitos é implementada pela classe *LOTagsCandidadesResources* por meio do método HTTP *get* à URI *http://[hostname]/los/{id}/tags_resources*. Este método realiza comunicação com o módulo de extração de conceitos, conforme apresentado no trecho de código da

Figura ***27***, onde são esperados os parâmetros de configuração suporte (*support*) e confiança (*confidence*) para a extração de conceitos.

```
228 class LOTagsCandidadesResources(Resource):
229
230     def get(self, lo_id):
231         """ Realiza a extracao de conceitos das palavras contidas na descricao """
232         args = parser.parse_args()
233         if args['support']:
234           support = args['support']
235         else:
236           support = '20'
237         if args['confidence']:
238           confidence = args['confidence']
239         else:
240           confidence = '0.4'
241
242         if request.args.get('text'):
243             text = request.args.get('text')
244             result = extr_c_utils.extract_concept_candidates(text, support, confidence)
245             data_js = json.loads(result)
246             return data_js, 200
247
248         else:
249             uri = repository_lo[lo_id]
250             obj = LO(uri)
251             text = obj.description
252             result = extr_c_utils.extract_concept_candidates(text, support, confidence)
253             data_js = json.loads(result)
254             return data_js, 200
```

Figura 27: Código fonte classe *LOTagsCandidadesResource.*

A extração de conceitos é realizada no texto presente no metadado “descrição” do objeto de aprendizagem, entretanto, também é possível submeter um texto por meio do parâmetro “*text*” para a realização da extração.

Após a extração, o servidor retorna ao cliente um arquivo no formato JSON com os conceitos candidatos identificados e suas respectivas classificações, de modo a permitir a escolha por parte do usuário, dos conceitos a serem interligados (*marshup*) semanticamente com *a Web of Linked Data*.

4.1.1.4. Atualização e recuperação de marcações (tags)

A interface para atualização e recuperação das marcações (*tags*) dos objetos de aprendizagem é implementada pela classe *LOTagsResources*, por meio do método HTTP *get* e *put*.

Quando invocado pelo método HTTP *get* à URI *http://[hostname]/los/{id}/tags_resources*, é retornado ao cliente uma lista no formato JSON com as marcações (*Tags*) do recurso, contendo as triplas RDF que representam as marcações, conforme definido no módulo de anotação de semântica

A atualização é realizada pelo método HTTP *put* à URI *http://[hostname]/los/{id}/tags_resources*, passando como parâmetro uma lista no formato JSON com as marcações, contendo a identificação e URI das entidades ou conceitos a serem marcados.

4.1.1.1.5. Identificação de conceitos em palavras-chave

A interface para identificação de conceitos em palavras-chaves é implementada pela classe *LOSubjectsCandidade*, por meio do método HTTP *get* à URI *http://[hostname]/los/{id}/subjects_candidates_resources*. Este método realiza a comunicação com o módulo de Extração de Conceitos para a identificação de entidade ou conceitos presentes nas palavras-chaves, e retorna a URI dos recursos candidatos ao *marshup* semântico.

São esperados os parâmetros de configuração domínio (*domain*), ou seja, o contexto de determinada ontologia, e o número máximo de retornos (*hits*) para

identificação dos recursos.

A identificação é realizada nas palavras presentes no metadado “*subject*” do objeto de aprendizagem, todavia, também é possível submeter uma palavra por meio do parâmetro “*word*” para a realização da identificação das entidades ou conceitos.

Após a identificação, o servidor retorna ao cliente um arquivo no formato JSON contendo o nome (*label*) e URI dos recursos identificado e suas respectivas classificações.

4.1.1.6. Atualização e recuperação de palavras-chave (subjects)

A interface para atualização e recuperação das palavras-chaves (*subjects*) dos objetos de aprendizagem é implementada pela classe *LOSubjectsResources* por meio do método HTTP *get* e *put*.

Quando invocado pelo método HTTP *get* à URI *http://[hostname]/los/{id}/subjects_resources*, é retornado ao cliente uma lista no formato JSON com as palavras-chaves (*subjects*) do recurso acessado, contendo, quando disponível, a URI do recurso associado a palavra-chave.

A atualização é realizada pelo método HTTP *put* à URI *http://[hostname]/los/{id}/subjects_resources*, passando como parâmetro uma lista no formato JSON com as palavras-chaves, podendo esta conter o nome (*label*) e URI do recurso associado identificado pelo módulo de Extração de Conceitos.

4.1.2. Extração de Conceitos

A extração de conceitos é realizada por meio dos serviços *Web DBpedia Spotlight* e *DBpedia Lookup Service*. Estes são responsáveis pela identificação de termos relevantes no texto, termos estes que também são encontrados na DBpedia. Após a identificação dos termos, é apresentada uma lista de conceitos candidatos a interligação, com suas respectivas relevâncias. Com a utilização destes serviços *Web*, se torna possível a interligação de conceitos identificados nos LOs com os presentes na DBpedia.

Os serviços Web foram integrados ao protótipo desenvolvido, por meio da classe *ExtractionConceptsUtils*, onde é implementada a comunicação com as APIs REST disponíveis nos serviços.

A classe *ExtractionConceptsUtils* disponibiliza três métodos, são eles: *resources_candidates*, *subject_resources_candidates* e *extract_concept_candidates*.

Os métodos *resources_candidates* e *subject_resources_candidates* implementam a comunicação com o *web service DBpedia Lookup*, que auxilia na identificação de conceitos presentes nas palavras-chaves dos objetos de aprendizagem. Na Figura 28, é apresentado o código fonte do método *resources_candidates*, que recebe como parâmetros uma palavra a ser buscada (*word*), o domínio (*domain*), ou seja, o contexto de uma determinada ontologia, e o número máximo de retornos (*hits*). Este método realiza a busca por recurso da *DBpedia* que representa o conceito relacionado com a palavra informada por meio do método *PrefixSearch* (*http://lookup.dbpedia.org/api/search/PrefixSearch*), disponível na API do *web service DBpedia Lookup.*

```
5  class ExtractionConceptsUtils():
6
7      def resources_candidates(self, word='', domain='', hits='5'):
8          """ """
9          server_backend_api = 'http://lookup.dbpedia.org/api/search/PrefixSearch'
10         query = server_backend_api + '?' + urllib.urlencode({'QueryClass': domain, 'MaxHits': hits, 'QueryString':word.encode('utf-8')})
11         buf = cStringIO.StringIO()
12         c = pycurl.Curl()
13         c.setopt(c.URL, query)
14         c.setopt(c.WRITEFUNCTION, buf.write)
15         c.setopt(c.HTTPHEADER, ['accept: application/json', 'Accept-Charset: UTF-8'])
16         c.perform()
17         return buf.getvalue()
```

Figura 28: Código fonte método *resources_candidates.*

O retorno do método é um arquivo JSON contendo a resposta do *web service*, a identificação (URI) dos recursos encontrados na *DBpedia*, relacionados a palavra informada e uma classificação dos recursos por relevância.

Os método *subject_resources_candidates*, apresentado na *Figura 29*, recebe como parâmetros uma palavra-chave a ser buscada (*word*), o domínio (*domain*) e o número máximo de retornos (*hits*). Este realiza a busca por recurso da *DBpedia* relacionados com a palavra-chave informada, por meio do método *KeywordSearch*, disponível na API do *web service DBpedia Lookup*.

```
19    def subject_resources_candidates(self, word='', domain='', hits='5'):
20        """ """
21        server_backend_api = 'http://lookup.dbpedia.org/api/search/KeywordSearch'
22        query = server_backend_api + '?' + urllib.urlencode({'QueryClass': domain, 'MaxHits': hits, 'QueryString':word.encode('utf-8')})
23        buf = cStringIO.StringIO()
24        c = pycurl.Curl()
25        c.setopt(c.URL, query)
26        c.setopt(c.WRITEFUNCTION, buf.write)
27        c.setopt(c.HTTPHEADER, ['accept: application/json', 'Accept-Charset: UTF-8'])
28        c.perform()
29        return buf.getvalue()
30
```

Figura 29: Código fonte método *subject_resources_candidates.*

O retorno ao método é um arquivo JSON contendo a resposta do *web service*, a identificação (URI) dos recursos encontrados na *DBpedia* e sua classificação por relevância.Figura ***30***, é apresentada uma tela de inserção de palavras-chave, onde ao ser inserido parte da palavra desejada, são efetuadas chamadas ao serviço *DBpedia Lookup* de maneira ser possível a criação de um campo com sugestões automáticas (*autocomplete*) para o usuário, com as informações encontradas na DBpedia. A requisição ao *DBpedia Lookup* é realizada por meio do serviço *Semantic LO*, especificamente chamada a API REST *"/los/{id}/subjects_candidates_resources"* que por sua vez aciona o método *resources_candidates* da classe *ExtractionConceptsUtils* passando como parâmetro a palavra digitada (*QueryString*) e o número máximo de resultados esperados (*MaxHits*). O método *resources_candidates* realiza a requisição ao serviço *DBpedia Lookup* conforme URL apresentada *http://lookup.dbpedia.org/api/search/PrefixSearch?MaxHits=10&QueryString=Bra.*

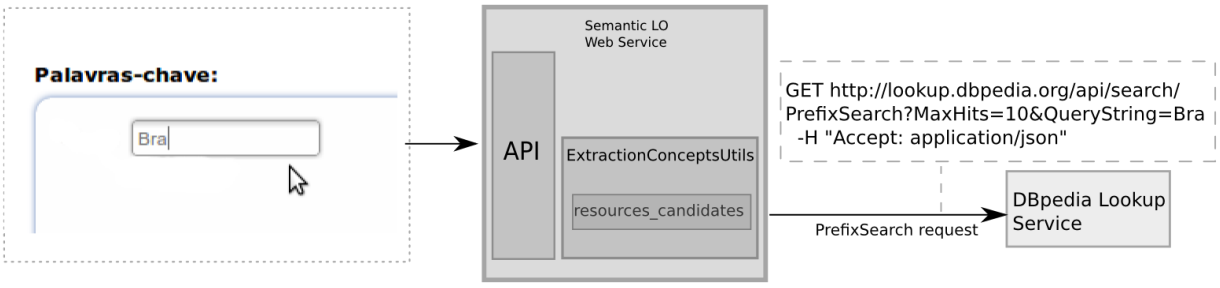

Figura 30: Exemplo de funcionamento serviço *DBpedia Lookup* integrado ao *web service.*

A Figura 31 apresenta o retorno da requisição ao serviço *DBpedia Lookup*, onde é recebido um arquivo no formato JSON contendo o resultado da busca. Este

arquivo é tratado pelo serviço pelo método *resources_candidates*, onde são extraídas as informações do nome do recurso (*label*) e URI presentes na *DBpedia*, e retornados ao cliente de maneira que possa ser escolhida pelo usuário a opção que representa a palavra a ser inserida. Após a escolha da palavra, é gerada uma referência para o recurso da *DBpedia* por meio da URI armazenada.

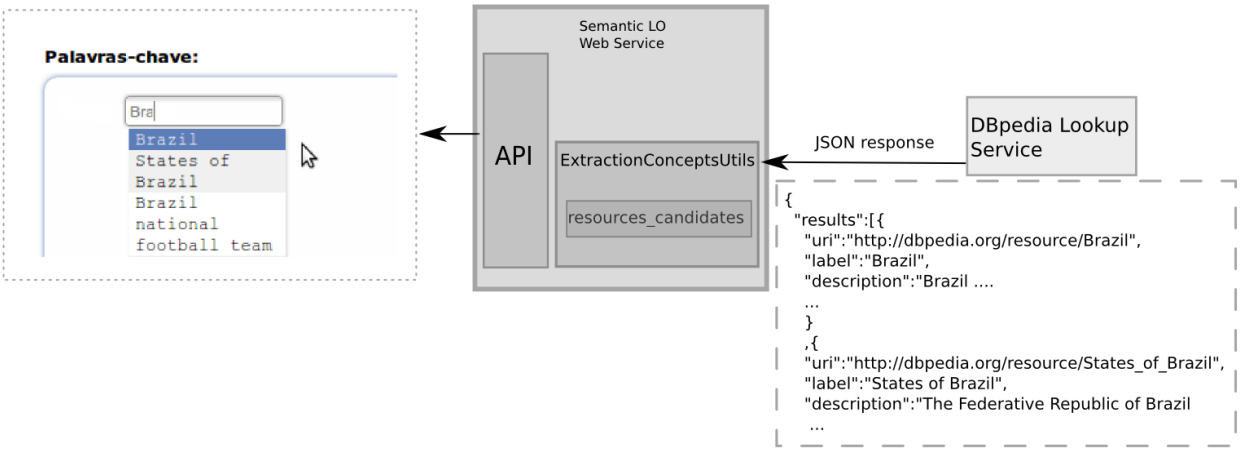

Figura 31: Retorno da requisição ao método *PrefixSearc.*

A comunicação com o web *service DBpedia Spotlight* é implementada pelo método *extract_concept_candidates*. Este recebe como parâmetro um texto (*text*) a ser extraído os conceitos e os parâmetros de configuração suporte (*support*), confiança (*confidence*) e linguagem (*language*).

O método realiza o mapeamento das APIs do *web service DBpedia Spotlight* para o idioma Português e Inglês, e configura de acordo com o parâmetro *language* recebido. O algoritmo de mineração de dados utilizado na extração dos conceitos é o *LingPipeSpotter* (implementado no *DBpedia Spotligth*), cuja implementação é baseada no algoritmo *LingPipe*, para o processamento de linguagem natural e análise textual.

O método faz uma requisição ao web *service DBpedia Spotlight* com os demais parâmetros de configuração, e realiza a chamada a API "*candidates*", conforme apresentado no código fonte demonstrado na Figura 32.

```
32
33    def extract_concept_candidates(self, text='', support='20', confidence='0.4', language='pt-br'):
34        """ """
35        servers_lang = {'pt-br': 'http://spotlight.sztaki.hu:2228/rest/candidates', \
36                        'en':'http://spotlight.dbpedia.org/rest/candidates'}
37
38        dbpedia_backend_api = servers_lang.get(language)
39        powered_by = 'no'
40        showScores = 'yes'
41        spotter = 'Default' # 'LingPipeSpotter', // one of: LingPipeSpotter,AtLeastOneNounSelector,CoOccurrenceBasedSelector
42        disambiguator = 'Default' #// one of: LingPipeSpotter,AtLeastOneNounSelector,CoOccurrenceBasedSelector
43        policy = 'whitelist'
44        types = ''
45        sparql = ''
46        data = urllib.urlencode({'text': text.encode('utf-8'), 'confidence': confidence, 'support': support, 'powered_by': powered_by,
  'showScores': showScores, 'disambiguator': disambiguator, 'spotter': spotter, 'policy' : policy, 'type' : types, 'sparql':
  sparql })
47        buf = cStringIO.StringIO()
48        c = pycurl.Curl()
49        c.setopt(c.URL, dbpedia_backend_api)
50        c.setopt(c.WRITEFUNCTION, buf.write)
51        c.setopt(c.HTTPHEADER, ['Accept:application/json', 'content-types:application/x-www-form-urlencoded'])
52        c.setopt(c.POST, 1)
53        c.setopt(c.POSTFIELDS, data)
54        c.perform()
55        return buf.getvalue()
56
```

Figura 32: Código fonte método *extract_concept_candidates*

O retorno do método é um arquivo JSON contendo o *status* da resposta *do web service*, as URIs e nomes (*label*) dos recursos candidatos identificados na *DBpedia*

Na Figura 33, pode ser observado o fluxo da identificação de conceitos e a interação entre os Web services.

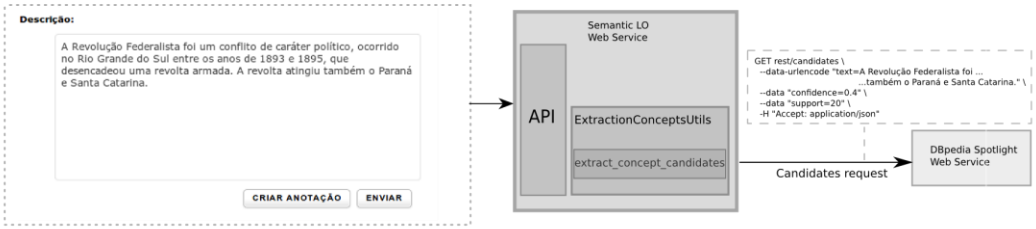

Figura 33: Fluxo da identificação de conceitos e a interação entre os Web services.

Do lado esquerdo da Figura 33, é apresentada uma interface de inserção de texto, a ser preenchida pelo usuário. Ao clicar em “Criar Anotação”, o texto é submetido ao protótipo do serviço Web “*Semantic LO*” por meio da API REST *“/los/{id}/tags_candidates_resources”,* este por sua vez, trata esta requisição e adiciona os parâmetros recebidos de suporte e confiança, realiza a chamada ao método *extract_concept_candidates*, onde é realizada a requisição ao serviço *DBpedia SpotLigth* por meio do método “*candidates”* disponível na API REST do serviço.

A Figura 34, demonstra o processo de resposta do serviço *DBpedia Spotligth*. Após o processamento e identificação dos termos candidatos, o serviço retorna ao

método *extract_concept_candidates* um arquivo no formato JSON, com uma lista de termos candidatos, as URIs identificadas na *DBpedia* e seus respectivos parâmetros de relevância.

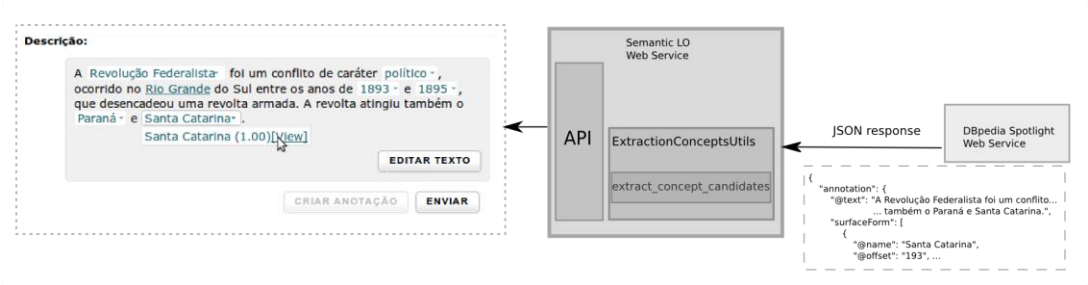

Figura 34: Processo de resposta do método *candidates* do serviço DBpedia Spotligt.

A

Figura ***35***, apresenta um trecho do arquivo JSON retornado, onde é descrita uma lista "*surfaceForm*" com os recursos identificados. A lista dos conceitos candidatos para interligação (*marshup*) é retornada ao cliente, de modo a possibilitar ao usuário a escolha dentre os termos que melhor traduz o conceito expresso no LO, possibilitando a interligação dos conceitos presentes no LO e a *DBpedia*.


```
{
    "annotation": {
        "@text": "A Revolução Federalista foi um conflito de caráter político, ocorrido no Rio Grande do Sul entre os anos de 1893 e 1895, que desencadeou uma revolta armada. A revolta atingiu também o Paraná e Santa Catarina.",
        "surfaceForm": [
            {
                "@name": "Santa Catarina",
                "@offset": "193",
                "resource": {
                    "@label": "Santa Catarina",
                    "@uri": "Santa_Catarina",
                    "@contextualScore": "0.7822950557166064",
                    "@percentageOfSecondRank": "3.095482640460671E-4",
                    "@support": "6987",
                    "@priorScore": "4.476355463754783E-4",
                    "@finalScore": "0.9996820459587104",
                    "@types": ""
                }
            },
```


Figura 35 : Trecho do arquivo JSON retornado ao método *extract_concept_candidates.*

4.1.3. Anotação Semântica

A etapa de anotação semântica é realizada pelas classes LO (*Learning Object*), Tag e Tagging, estas são responsáveis respectivamente pela representação do objeto de aprendizagem e representação das marcações (*tag* e *tagging*) com uso

da ontologia MUTO (*Modular Unified Tagging Ontology*) (LOHMANN, 2011), uma ontologia para marcação (*tagging*) e folksonomias (*folksonomies*) baseada em uma revisão completa de ontologias de marcação existentes que unifica conceitos fundamentais em um esquema consistente e extensível. Suporta diferentes formas de marcação, como marcação comum, semântica, por grupo, privada, e marcação automática (LOHMANN, 2011).

As classes LO (*Learning Object*), Tag e Tagging são implementadas em *Python* e fazem uso da biblioteca RDFAlchemy's, um ORM (Object RDF Mapper) que permite o mapeamento de objetos *python* para RDF, facilitando sua manipulação e armazenamento (COOPER & HIGGINS, 2012).

O mapeamento implementado na classe LO, apresentada na Figura 36, define os metadados do objeto de aprendizagem a serem armazenados. No desenvolvimento deste protótipo, foram utilizados os metadados do *Dublin Core*: título (*title*), identificador (*identifier*), descrição (*description*) e palavras-chaves (*subject*) conforme descritos em *DCMI Metadata Terms* (DCMIb, 2012), utilizando-se o *namespace dcterms*, seguindo as recomendações de mapeamento do padrão de metadados IEEE-LOM para RDF, baseado nos princípios *Linked Data,* conforme descrito por Rajabi et al. (2013).

São importados da classe *utils,* os *namespaces lom* e *dcterms* definidos, respectivamente, pelas URIs *http://ltsc.ieee.org/rdf/lomv1p0/lom#* e *http://purl.org/dc/terms/.* Estes *namespaces* são transformados em instâncias da classe *Namespace* definida pela biblioteca *rdflib*, uma biblioteca Python para manipulação de RDF (RDFLIB, 2014), que provê mecanismos para o gerenciamento de *namespaces*, de modo que URIs podem ser manipuladas mais facilmente como objetos Python. Para gerar a URI do *namespace* http://ltsc.ieee.org/rdf/lomv1p0/lom#/LearningObject., basta acessar uma instância de *Namespace*, conforme exemplo: “lom.LearningObject”.

A classe LO herda da classe *rdfSubject*, responsável por gerenciar todo o armazenamento da representação de uma instância em RDF. Esta classe define, por exemplo, as configurações do banco de triplas onde deverá ser armazenado o recurso. Realizando um paralelo com a representação em triplas RDF (sujeito-predicado-objeto), instâncias de objetos definidos na classe LO correspondem ao sujeito. Os atributos representam o predicado e os dados armazenados em cada

atributo são o valor ou objeto.

São importados os descritores (*descriptors*) *rdfSingle* e *rdfMultiple,* definidos pela biblioteca RDFAlchemy's, necessários para informar o tipo de campo dos metadados. O descritor *rdfSingle* é usado para definição de campo com valor literal simples. E *rdfMultiple* é usado para armazenamento de múltiplos valores. São definidos cinco atributos na classe LO correspondentes aos metadados utilizados e o tipo de recurso, sendo estes: *rdf_type* (propriedade que define o tipo de recurso), *title* (título do recurso), *identifier* (identificador do recurso), *description* (descrição do recurso), *subject* (palavras-chaves do recurso).

```
1  # -*- coding: utf-8 -*-
2
3 from utils import *
4 from rdfalchemy.descriptors import *
5 from rdfalchemy.orm import mapper
6
7 class LO(rdfSubject):
8   """ Representacao de Objeto de Aprendizagem """
9
10   rdf_type = lom.LearningObject
11   title = rdfSingle(dcterms.title)
12   identifier = rdfSingle(dcterms.identifier)
13   description = rdfSingle(dcterms.description)
14   subject = rdfMultiple(dcterms.subject)
15
16 mapper(LO)
```

Figura 36: Código fonte classe LO.

As classes Tag e Tagging são responsáveis por armazenar as informações dos conceitos extraídos, realizar as marcações (*tagging*) e o *marshup* semântico dos conceitos presentes nos objetos de aprendizagem com a DBpedia. Estas classes recebem as informações extraídas na fase de extração de conceitos e realizam o mapeamento das informações para a ontologia MUTO (LOHMANN, 2011), que é uma ontologia para marcação (*tagging*) e folksonomia, baseada em uma revisão completa sobre ontologias de *tagging* que unifica conceitos fundamentais em um vocabulário consistente.

A classe Tagging, apresentada na Figura 37, é responsável por ligar os recursos que estão sendo marcados, ou seja, anotados semanticamente, com os

respectivos marcadores (*tags*). As informações mapeadas nesta classe são: o recurso marcado (*taggedResource*), informações de data e hora de criação (*taggingCreated*) , informações de data e hora de modificação (*taggingModified*), usuário responsável pela criação (*hasCreator*), permissões de acesso (*hasAccess*), concessão de direitos de acesso (*grantAccess*) e marcadores associados (*hasTag*).

O atributo *hasTag* recebe o parâmetro especial *range_type*, apresentado na linha 18 da Figura 37, usado pala restringir o tipo de objeto ou recurso que deverá ser usado como valor para o atributo. O *namespace* da ontologia MUTO é importado da classe *utils*, referenciado como "*muto*", definido pela URI http://purl.org/muto/core#.

```
1  # -*- coding: utf-8 -*-
2
3 from utils import *
4 from rdfalchemy.descriptors import *
5 from rdfalchemy.orm import mapper
6
7 class Tagging(rdfSubject):
8   """ Representacao de Tagging utilizando ontologia MUTO """
9
10   rdf_type = muto.Tagging
11   taggedResource = rdfSingle(muto.taggedResource)
12   taggedWith = rdfSingle(muto.taggedWith)
13   taggingCreated = rdfSingle(muto.taggingCreated)
14   taggingModified = rdfSingle(muto.taggingModified)
15   hasCreator = rdfSingle(muto.hasCreator)
16   hasAccess = rdfSingle(muto.hasAccess)
17   grantAccess = rdfSingle(muto.grantAccess)
18   hasTag = rdfMultiple(muto.hasTag, range_type=muto.Tag)
19
20 mapper(Tagging)
21
```

Figura 37: Código fonte classe Tagging.

A classe Tag, apresentada na Figura 38, é responsável por mapear os marcadores (*tags*) dos conceitos extraídos dos LOs. São utilizadas as seguintes informações: indicação se a marcação foi realizada automaticamente (*autoMeaning*), marcadores associados com mesmo significado (*meaningOf*), próximo marcador (*nextTag*), marcador anterior (*previousTag*), informações de data e hora de criação (*tagCreated*), rótulo do marcador (*tagLabel*), significado da marcação, recurso associado (*tagMeaning*) e marcação associada (*tagOf*).

```python
# -*- coding: utf-8 -*-

from utils import *
from rdfalchemy.descriptors import *
from rdfalchemy.orm import mapper

class Tag(rdfSubject):
  """ Representacao de Tag utilizando ontologia MUTO """

  rdf_type = muto.Tag
  autoMeaning = rdfSingle(muto.autoMeaning)
  meaningOf = rdfSingle(muto.meaningOf)
  nextTag = rdfSingle(muto.nextTag)
  previousTag = rdfSingle(muto.previousTag)
  tagCreated = rdfSingle(muto.tagCreated)
  tagLabel = rdfSingle(muto.tagLabel)
  tagMeaning = rdfSingle(muto.tagMeaning)
  tagOf = rdfSingle(muto.tagOf)

mapper(Tag)
```

Figura 38: Código fonte classe Tag.

Na Figura 39, pode-se visualizar a descrição da *tag* Brasil, associada ao conceito Brasil, encontrado na DBpedia (Disponível em: <http://pt.dbpedia.org/resource/Brasil>. Acesso em: 20 out 2014.)

```
−<rdf:Description rdf:about="http://localhost:8080/semanticlo/tag/7f91e206-6442-400c-9028-d67a59ed84bd">
    <rdf:type rdf:resource="http://purl.org/muto/core#Tag"/>
    <tagLabel>Brasil</tagLabel>
    <tagMeaning rdf:resource="http://pt.dbpedia.org/resource/Brasil"/>
    <tagOf rdf:resource="http://localhost:8080/semanticlo/tagging/ad5f256f-3a0c-4b52-9755-43304f287e8b"/>
  </rdf:Description>
```

Figura 39: Trecho do código RDF/XML descrevendo a tag Brasil na ontologia MUTO.

4.1.4. Armazenamento

O armazenamento dos objetos de aprendizagem é realizado no banco de triplas RDF (RDF TripleStore) OpenRDF Sesame. Ele pode ser configurado quanto à forma de armazenamento (em memória ou RDF *store*) e permite a realização de consultas utilizando a linguagem SPARQL ou SeRQL.

O servidor OpenRDF Sesame foi instalado e configurando com o repositório local implementado via *NativeStore* denominado *semanticlo* e integrado como armazenamento padrão do *Web service* com o uso da biblioteca RDFAlchemy's. Esta biblioteca inclui o acesso ao OpenRDF Sesame através da classe *SesameGraph* que é uma subclasse de *SPARQLGraph* construída para prover

acesso de leitura à SPARQL *endpoint* com o uso do protocolo HTTP Sesame2 (COOPER, 2012).

Foi adicionando ao servidor uma interface para *Linked Data* denominada *Pubby*, Esta interface é um projeto desenvolvido para prover à SPARQL *endpoints* uma forma mais fácil de interligar, descobrir e consumir dados na *Web* Semântica (CYGANIAK & BIZER, 2011). Esta interface permite que navegadoras RDF, RDF *crawlers* e agentes de buscas RDF acessem os dados armazenados, sem a necessidade de consultas diretas via SPARQL. Suas características principais são: disponibilizar uma interface HTML simples que exibe os dados disponíveis sobre os recursos, manipulação e negociação de conteúdo, e a geração de URIs que possibilitem o derreferenciamento.

Na Figura 40 é apresentada a interface padrão do *Pubby,* com o acesso realizado por meio de um navegador *Web* ao recurso *http://localhost:8080/semanticlo/resource/historia-do-brasil,* onde são exibidos os dados disponíveis sobre o recurso e é possível verificar o funcionamento da negociação de conteúdo para exibição da página em HTML. São adicionados, automaticamente, metadados referentes a proveniência do recurso acessado. A *W3C Provenance Incubator Group* define proveniência de um recurso no contexto de *Linked Data* como: o conjunto de informações relacionadas às entidades, aos processos e aos agentes envolvidos na produção e disponibilização do recurso (GIL et al., 2010). Como pode ser visualizado na seção “Metadata” da Figura 40, estes metadados são utilizados para a verificação da confiabilidade dos dados e como eles devem ser integrados com outras fontes de informação, bem como, para dar crédito aos seus autores (MENDONÇA, 2013).

Segundo Mendonça et al. (2014), metadados de proveniência têm sido reconhecidos como mecanismo fundamental no apoio a avaliação de qualidade e consistência de dados e, de forma geral, como complemento aos esforços de integração e interoperabilidade.

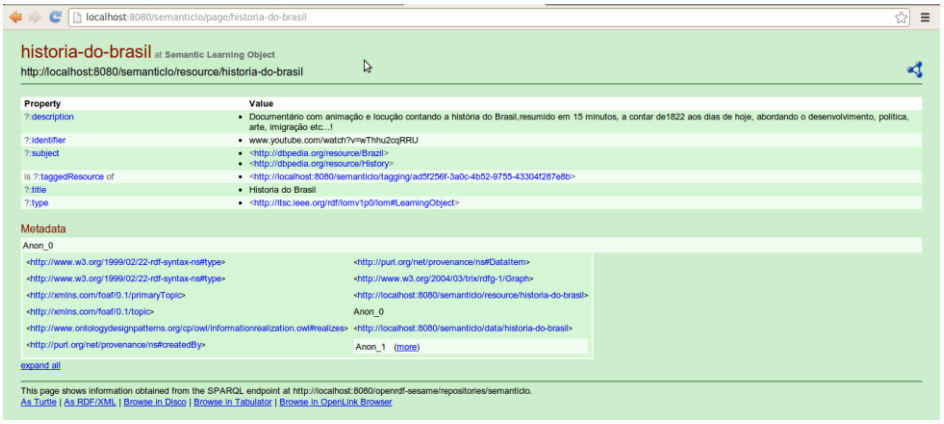

Figura 40: Acesso ao recurso “História do Brasil”.

4.1.5. Consulta

A busca no repositório de dados RDF é implementada pela classe *SearchRepository,* que recebe como parâmetro o metadado a ser buscado, realiza a tradução da *query* de busca para a linguagem SPARQL e submete ao banco de triplas RDF. Na Figura *41*, temos um exemplo de busca pelo metadado título, onde é passado o termo “História” de modo a retornar todos os LOs armazenados que contenham o termo “História” no título. O módulo de consulta realiza a tradução da busca para a linguagem SPARQL, conforme apresentado na Figura *41* e retorna ao cliente um arquivo em formato JSON com o resultado da busca.

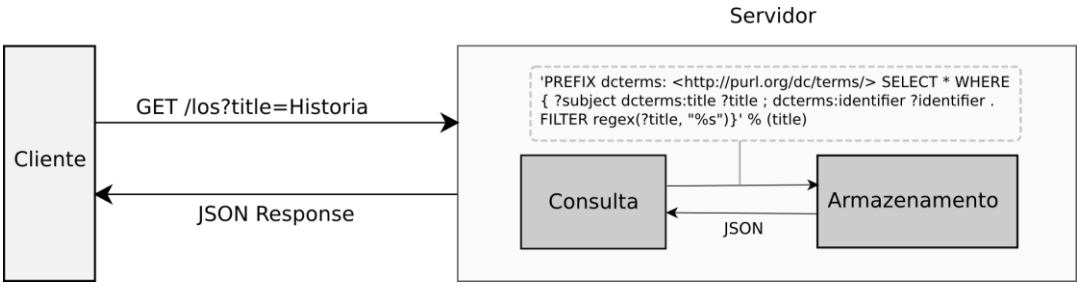

Figura 41: Tradução de Consulta para linguagem SPARQL.

A Figura 42 apresenta o retorno da busca no formato JSON, onde são apresentados todos os resultados para a *query* de busca pelo título “História”. O arquivo de retorno contém o titulo, identificador e a URI de cada resultado.

```
{
  "head" : {
    "vars" : [ "subject", "title", "identifier" ]
  },
  "results" : {
    "bindings" : [ {
      "identifier" : {
        "type" : "literal",
        "value" : "www.youtube.com/watch?v=wThhu2cqRRU"
      },
      "subject" : {
        "type" : "uri",
        "value" : "http://localhost:8080/semanticlo/resource/historia-do-brasil"
      },
      "title" : {
        "type" : "literal",
        "value" : "Historia do Brasil"
      }
    }, {
      "identifier" : {
        "type" : "literal",
        "value" : "http://www.youtube.com/watch?v=wwdiRYNLnno"
      },
      "subject" : {
        "type" : "uri",
        "value" : "http://localhost:8080/semanticlo/resource/um-resumo-da-historia-do-estados-unidos"
      },
      "title" : {
        "type" : "literal",
        "value" : "Um resumo da Historia do Estados Unidos"
      }
    } ]
  }
}
```


Figura 42: Retorno da busca pelo termo “História” em formato JSON.

4.2. ESTUDO DE CASO NO AMBIENTE MOODLE

O ambiente de aprendizagem Moodle na versão 2.4 foi customizado através da instalação e desenvolvimento de *plugins* que possibilitam a comunicação com os diversos serviços providos pelo *Web service*, possibilitando uma visão integradora dos serviços, conforme apresentado na Figura 43.

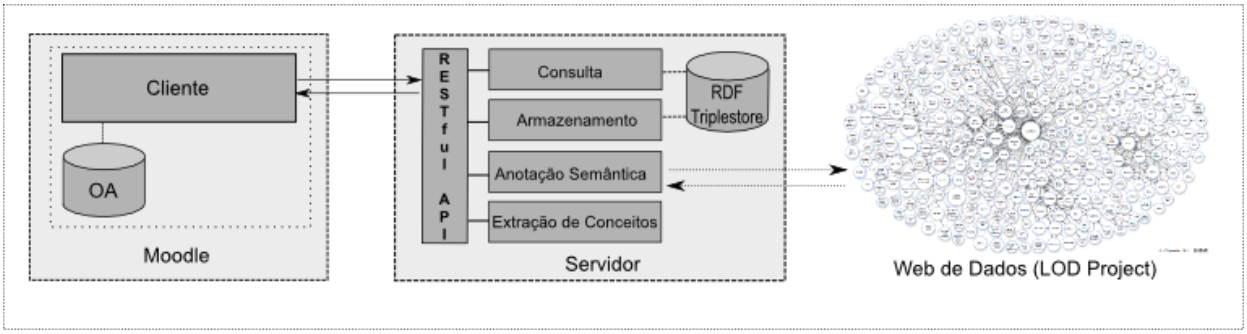

Figura 43: Visão geral da integração Moodle e Web service.

O fluxo de comunicação do sistema e a interação entre os módulos da arquitetura orientada a serviços são apresentados na Figura 44, onde é

demonstrado o fluxo da anotação semântica de conteúdo distribuído em três camadas: Aplicação, Servidor e *Web* de dados.

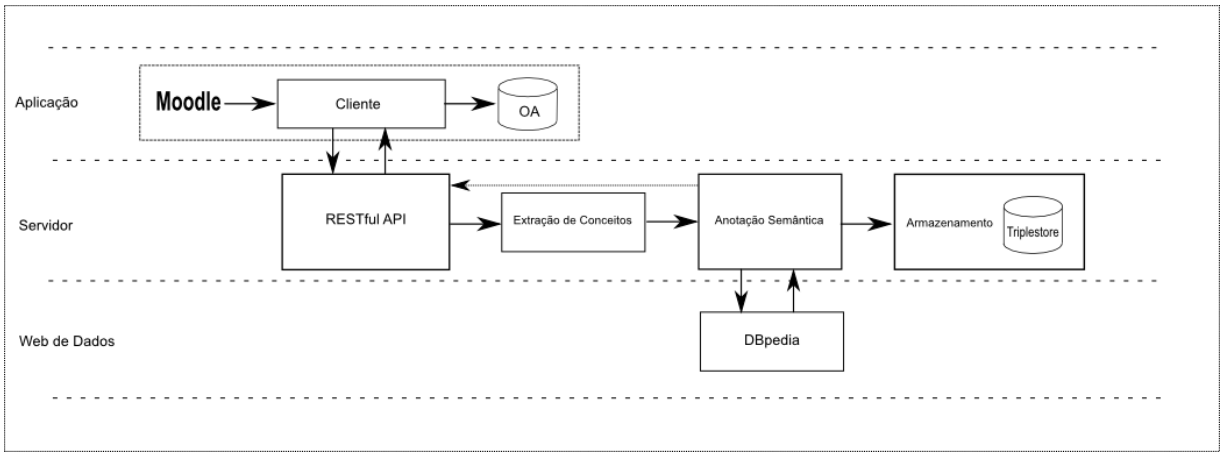

Figura 44: Fluxo da iteração entre as camadas da arquitetura proposta.

Na camada de Aplicação, encontra-se o ambiente de aprendizagem Moodle, que realiza a comunicação com a camada Servidor por meio de três *plugins* cliente específicos desenvolvidos para a *API Restfull* disponibilizada pelo *Web service*. São eles:

- **moodle-repository-semantic-lo:** Responsável pela interface de consulta e acesso aos objetos armazenados.
- **moodle-repository-semantic-lo-upload:** Responsável pela inserção de objetos de aprendizagem com conteúdo a ser armazenado no repositório local.
- **moodle-repository-semantic-lo-url:** Responsável pela inserção de objetos de aprendizagem com referência a recursos externos ao repositório.

Os *plugins* desenvolvidos têm as características de repositório externo de conteúdo para o ambiente *Moodle*, conforme descrito em *https://docs.moodle.org/dev/ Repository_plugins* (MOODLE.ORG, 2014).

Ao ser criado um objeto de aprendizagem no ambiente *Moodle*, este é processado pelo *plugin* cliente específico antes de seu armazenamento final, para que seu conteúdo seja anotado semanticamente.

A camada Servidor realiza o processamento deste objeto de aprendizagem criado e retorna o seu conteúdo anotado semanticamente.

Na primeira etapa é realizada a Extração de Conceitos, que identifica os conceitos relevantes no conteúdo, por meio de técnicas de mineração de textos e envia estas informações ao módulo de Anotação Semântica.

Na etapa de Anotação Semântica é realizado o *mashup* dos conceitos identificados e a interação com a camada de *Web* de Dados, onde é utilizado a *DBpedia* como fonte de dados. O retorno da anotação semântica é formado por triplas RDF que são armazenadas na camada Servidor, pelo módulo de Armazenamento.

Após o término do processamento do objeto de aprendizagem, o cliente do *Moodle,* da camada de Aplicação, recebe uma URI do recurso armazenado e passa a ser disponível para utilização.

Todos os recursos armazenados ficam disponíveis para consulta por meio de uma interface específica, possibilitando assim o reuso dos recursos armazenados. Esta interface realiza as buscas nos repositório em RDF através de consultas SPARQL.

A seguir, será detalhado o processo de adição e busca de recursos no ambiente *Moodle* e a interação com os *plugins* desenvolvidos.

4.2.1. Processo de adição de recurso

Os recursos, ou seja, os objetos de aprendizagem são adicionados como parte de um curso no ambiente de aprendizagem.

O usuário escolhe o tipo de recurso que será armazenado, sendo estes: referências externas (URL); arquivos; páginas, dentre outros, conforme apresentado na Figura 45, onde é demonstrada a tela de inserção de objetos de aprendizagem do ambiente *Moodle*.

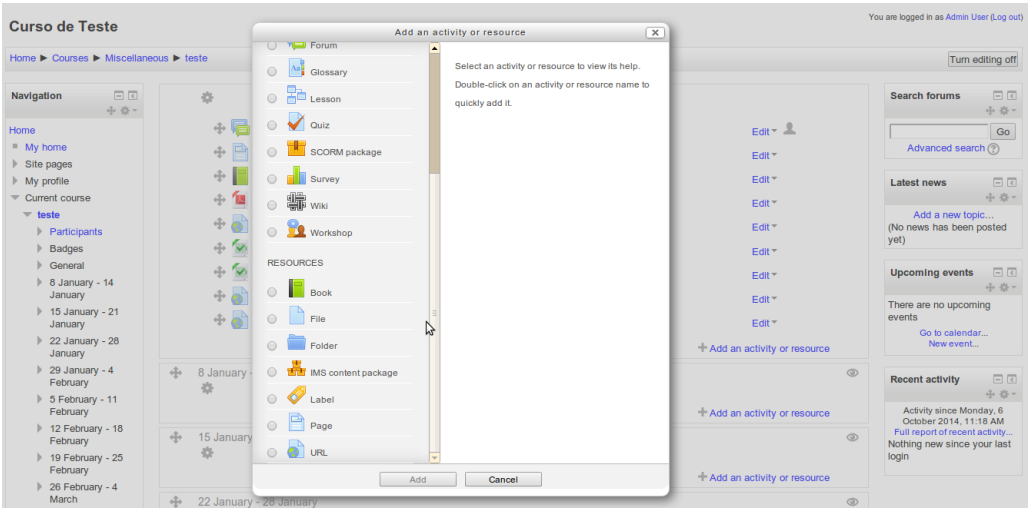

Figura 45: Tela de Inserção de Objetos de Aprendizagem do ambiente Moodle.

Após a escolha do tipo de recurso que será armazenado, o usuário é direcionado para a interface padrão do *Moodle* para inserção dos metadados que descrevem a atividade no curso, onde está sendo adicionado o recurso. Nesta etapa, o usuário pode optar por inserir um novo recurso ou reutilizar um recurso já existente nos repositórios disponíveis no ambiente.

Os *plugins* desenvolvidos trabalham como repositórios para o ambiente de aprendizagem, desta forma, o usuário pode efetuar uma busca nos objetos armazenados no repositório semântico por meio do *plugin moodle-repository-semantic-lo*, ou adicionar uma referência externa ou arquivo, por meio dos *plugin moodle-repository-semantic-lo-url* e *plugin moodle-repository-semantic-lo-upload*.

Neste estudo de caso foram customizadas as telas de inserção de recursos tipo URL e arquivo, sendo o *plugin moodle-repository-semantic-lo-url* responsável pela inserção de referência a recursos externos, e o *plugin moodle-repository-semantic-lo-upload,* responsável pela interface de *upload* de arquivos para o ambiente.

Ao ser escolhida a inserção de uma referência externa, é apresentada uma interface ao usuário, conforme Figura 46, para que seja preenchido o metadado titulo e URL.

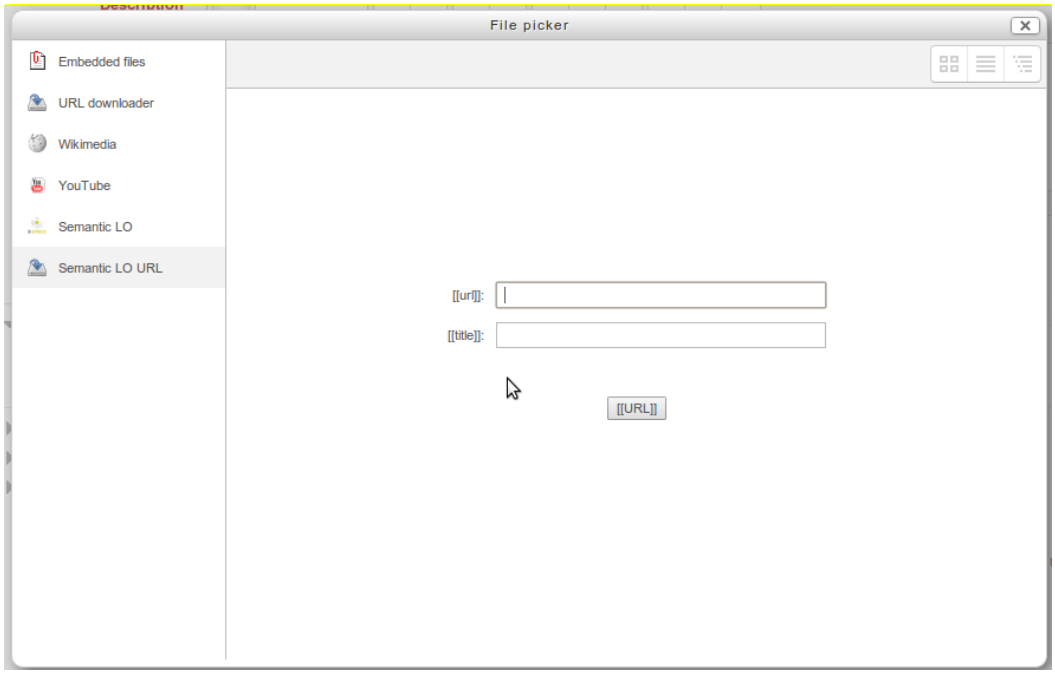

Figura 46: Tela de Inserção de referências externas.

Após o preenchimento destes metadados, é verificado se a URL é válida e o objeto é submetido ao protótipo de *Web service* desenvolvido, para que os dados sejam armazenados. O *Web service,* por sua vez, realiza o armazenamento do objeto e retorna um identificador do objeto. Após receber o retorno do serviço com o identificador e o *status* da operação, o *plugin moodle-repository-semantic-lo-url* direciona o usuário para uma interface de inserção dos metadados palavas-chaves e descrição. Nesta etapa é realizada uma comunicação com a *Web service* para a criação de um campo de preenchimento de palavras-chaves com sugestões automáticas para o usuário, conforme apresentado na Figura 47.

Esta interface estabelece uma comunicação com o *Web service*, por meio da API REST *"/los/{id}/subjects_candidates_resources"* de modo que buscas na *DBpedia* possam ser realizadas para retornar aos usuários conceitos candidatos para interligação com as palavra-chaves.

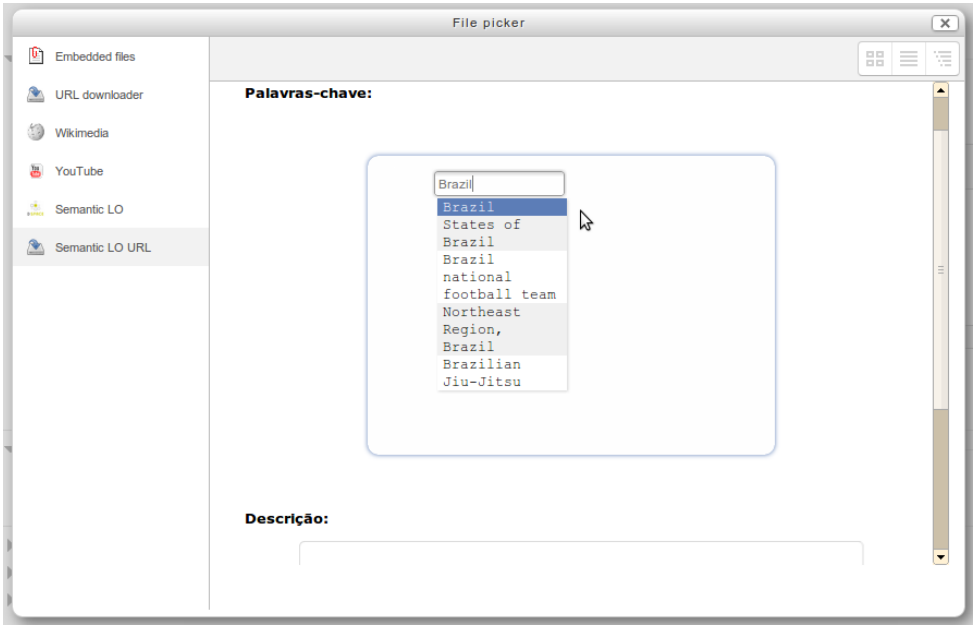

Figura 47: Tela de preenchimento do metadado palavra-chave.

No metadado descrição é realizada a identificação dos conceitos presentes no texto. Esta etapa ocorre de forma semi-automática. É apresentada para o usuário uma interface para preenchimento da descrição e uma opção de “Criar Anotação”, conforme Figura 48, onde após o preenchimento o usuário solicita a criação das anotações e o texto é submetido ao *Web service* para identificação dos conceitos presentes.

Figura 48 : Tela de inserção metadado descrição.

Após a identificação, são retornados as entidades e conceitos presentes na DBpedia, com um ranking, de maneira ser possível a escolha do conceito que melhor representa o conceito identificado no contexto do objeto de aprendizagem. Na Figura 49, é apresentado o retorno ao usuário com os conceitos identificados.

Figura 49: Marcação de conceitos identificados no metadado descrição.

Ao clicar no botão “Enviar”, os dados inseridos nos metadados palavras-chaves e descrição, bem como as marcações efetuadas, são submetidas ao *Web service* para que o objeto de aprendizagem possa ser atualizado .

O processo de adição de recurso tipo arquivo, realizado pelo *plugin moodle-repository-semantic-lo-upload*, segue as mesmas etapas do processo de inserção de recurso de referência externa. Entretanto, ao invés da inserção de uma URL externa, o usuário escolhe um arquivo para realizar o *upload* para o sistema de arquivos local do ambiente *Moodle*.

4.2.2. Busca no repositório

A busca no repositório é realizada por meio do *plugin moodle-repository-*

semantic-lo, que disponibiliza uma interface, conforme apresenta na Figura 50, onde são inseridos os metadados a serem buscados. Este *plugin* recebe os dados inseridos pelo usuário e realiza uma chamada ao *Web service,* passando os parâmetros de busca no repositório de objetos de aprendizagem. Neste estudo de caso foram implementadas buscas pelo metadado Título e Identificador (URI).

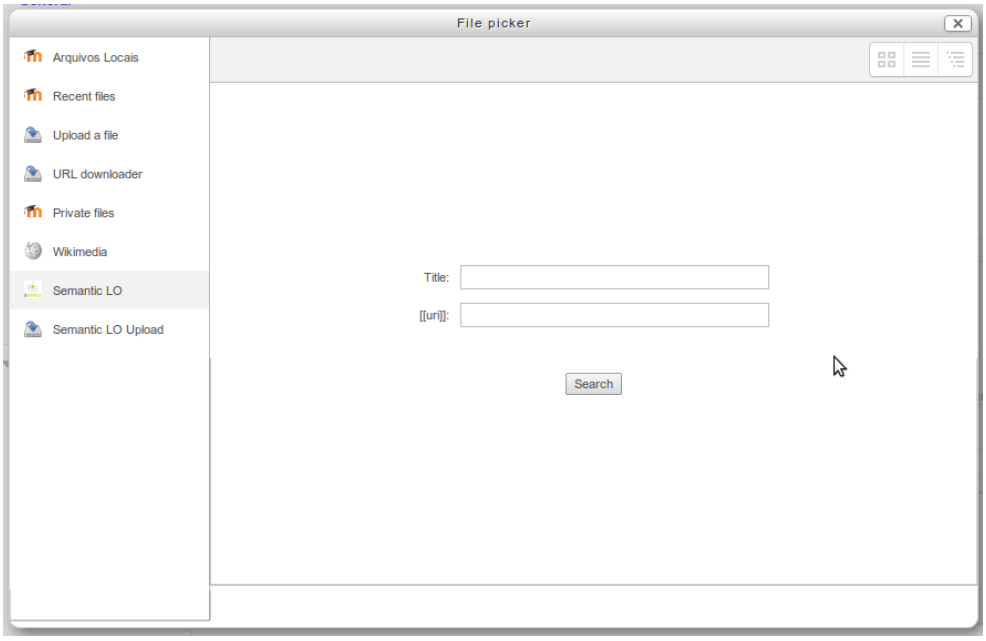

Figura 50: Interface de Busca.

O retorno da busca no repositório é apresentado ao usuário, conforme Figura 51, onde é apresentada uma miniatura com o *preview* dos objetos encontrados, bem como o título e identificador.

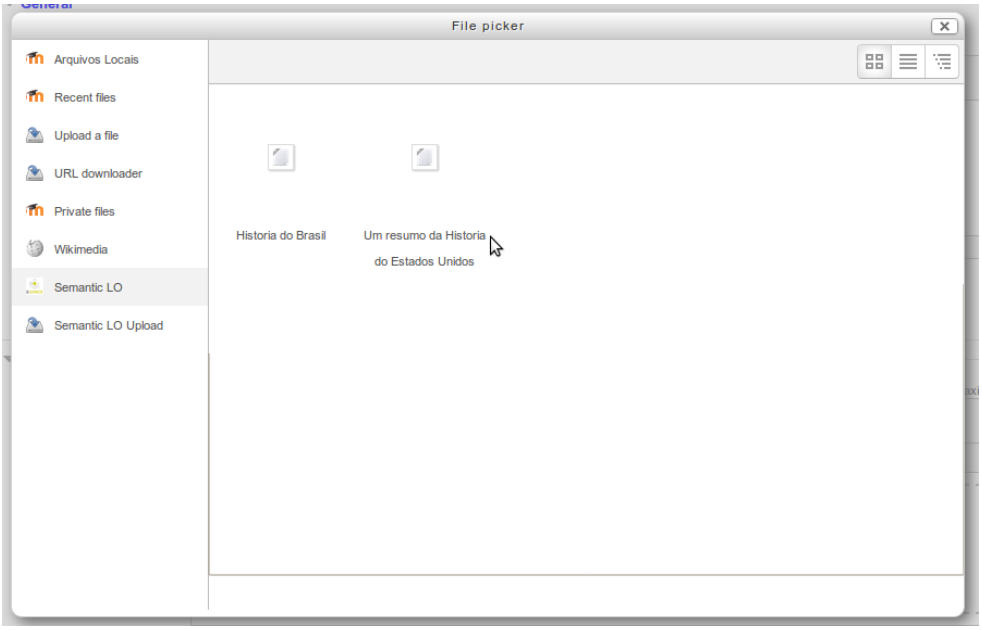

Figura 51: Interface de retorno da busca no repositório.

Ao escolher um dos objetos retornados, as referências do objeto são adicionadas automaticamente no curso ou atividades a ser utilizado.

5. TRABALHOS RELACIONADOS

Após o levantamento bibliográfico, foi verificado que existem outros trabalhos na literatura especializada que apresentam o uso de tecnologias da *Web* semântica para o enriquecimento de objetos de aprendizagem em ambientes de aprendizagem. No entanto, trazem abordagens diferentes da que foi apresentada neste trabalho. A seguir, serão realizadas algumas comparações, destacando as principais diferenças com este trabalho.

Em Doush et al. (2011) é apresentada uma proposta de framework para adição de informações semânticas em conteúdo de *e-learning* para aprendizagem de matemática, com base em dois princípios. O primeiro é a adição automática de informação semântica na criação de conteúdo matemático, utilizando a recomendação da W3C para codificação de material matemático na *Web* em RDF, denominado *MathML*("*Mathematical Markup Language*) (SANDHU, 2010). O segundo princípio é a marcação colaborativa, permitindo que os alunos destaquem e anotem partes específicas dos conteúdos, e a utilização de uma ontologia específica para classificação de conteúdo.

O objetivo do trabalho proposto por Doush et al. (2011) é agregar significado em conteúdo de educação a distância, adicionando relações entre conteúdos, e criar um ambiente para facilitar a busca pelo conteúdo, permitindo a consulta semântica de conteúdo, com a linguagem SPARQL. O ambiente de aprendizagem utilizado foi o *Moodle* e integrado a ferramenta de edição de equações matemáticas MathCast ("*MathCast, an open source equation editor*). (CHAKAN, 2010)

Doush et al. (2011) apresenta objetivos semelhantes ao apresentado nesta dissertação, porém o escopo é restrito a criação de objetos de aprendizagem para

ensino de matemática. Também é válido destacar que a arquitetura proposta nesta dissertação possibilita a integração com diferentes ambientes de aprendizagem, o que diferencia do trabalho proposto por Doush et al. (2011) que visa a integração somente com o ambiente de aprendizagem *Moodle*. Entretanto, é importante destacar a diferença do processo de marcação e anotação semântica de conteúdo, onde em Doush et al. (2011) é realizado de forma colaborativa e nesta dissertação é realizado de forma semi-automática sem a possibilidade de marcação colaborativa.

Em Bamidis et al. (2011) é apresentada uma proposta de um módulo para o LCMS *Moodle* que descreve objetos de aprendizagem de acordo com o *schema mEducator*[9], utilizado para criar e publicar recursos educacionais da área de medicina. Este *schema* também possibilita a interligação dos recursos com a *Web* de Dados (*Web of Linked Data*). Bamidis et al. (2011) afirma que existe uma necessidade de tornar mais eficaz o processo de busca e recuperação de informações e recursos na área de educação médica e saúde.

O módulo desenvolvido é baseado na ferramenta *MrCute2*[10] (*Moodle Repository Create, Upload, Tag and Embed*) que cria um repositório local para armazenamento de objetos de aprendizagem no ambiente *Moodle*. Um novo banco de dados e a interface do *MrCute* foram estendidos de maneira a implementar as classes e propriedades definidas no *schema mEducator.* A descrição semântica dos objetos de aprendizagem é realizada pelo servidor D2R[11], onde foi desenvolvido um mapeamento D2RQ[12] para a ontologia definida no *schema mEducator*. Este servidor disponibiliza uma interface para a realização de buscas com a linguagem SPARQL e também permite que os dados sejam interligados à *Web* de dados (*Web of Linked Data*) por meio de um *Sparql endpoint*.

A arquitetura e o protótipo apresentados nesta dissertação apresentam grandes vantagens quando comparados com o trabalho de Bamidis et al. (2011), uma vez que a arquitetura proposta na dissertação é flexível, possibilitando a integração com outros LCMS.

Em Dovrolis el al. (2011) é apresentado um ambiente para publicação e

[9] Disponível em: <http://linkededucation.org/meducator>. Acesso em: 20 out 2014
[10] Disponível em: <http://www.learningobjectivity.com/mrcute/>. Acesso em: 20 out 2014
[11] Disponível em: <http://d2rq.org/d2r-server/>. Acesso em: 20 out 2014
[12]Disponível em: < http://d2rq.org/d2rq-language/>. Acesso em: 20 out 2014

compartilhamento de conteúdo educacional para área médica. Dovrolis el al. (2011) afirma que muitas vezes o conteúdo educacional é compartilhado entre diferentes educadores, enriquecido e adaptado de modo ser possível o reuso em diferentes contextos. O ambiente apresentado utiliza paradigmas da *Web* Social e *Web* Semântica para publicação de recursos educacionais anotados semanticamente e interligados à *Web* de Dados (*Web of Linked Data*). O objetivo do trabalho apresentado por Dovrolis el al. (2011) é permitir que seja possível a realização de buscas e recuperação de conteúdo educacional mais relevante na área médica incluindo publicações científicas e dados clínicos.

O trabalho proposto por Dovrolis el al. (2011) e o apresentado nesta dissertação apresentam finalidades muitos semelhantes, porém em Dovrolis el al. (2011) não é realizada a integração a um ambiente de aprendizagem, nem mesmo anotações de objetos de aprendizagem de forma automática.

6. CONSIDERAÇÕES FINAIS

Na educação a distância, disponibilizar materiais de aprendizagem relevantes e de valor com foco no reaproveitamento e a interoperabilidade entre as diversas plataformas de produção de recursos didáticos torna-se um desafio. Portanto, este trabalho apresenta a utilização de técnicas de mineração de textos e uma arquitetura de *Web services* para anotação semântica de conteúdo que enriquecem os objetos de aprendizagem, possibilitando novas formas de descoberta de conhecimento e reuso de informação, construindo novas formas de acesso a dados disponíveis em ambientes de aprendizagem, por meio da descrição estruturada em RDF dos objetos de aprendizagem. Além disso, interliga conceitos a fonte de dados na *Web* de Dados, enriquecendo a base de conhecimento do ambiente de aprendizagem com *mashup* semântico entre estes conceitos e recursos pré-existentes da *Web* de dados (*Web* of Linked Data).

Por fim, possibilita, um aumento da produtividade no processo de ensino-aprendizagem, uma vez que o usuário, quer seja aprendiz, quer seja instrutor, passa a ser assistido pela máquina, de forma efetiva, na tarefa periférica de busca, integração e inferência ontológica de conhecimento, podendo, pois, focar por mais tempo na tarefa que realmente importa, que é a absorção do referido conhecimento.

6.1. CONTRIBUIÇÕES

As principais contribuições deste trabalho são:

- Definição de uma arquitetura orientada a serviço para agregação de diferentes *Web services* para o apoio a anotação semântica de conteúdo;

- Construção de um protótipo de *Web service restful* para anotação semântica de conteúdo, possibilitando a integração com diferentes plataformas e ferramentas de apoio à educação a distância;
- Construção de *plugins* para o ambiente *Moodle*, que integram as diversas funcionalidades providas pelo *Web service* desenvolvido.
- Descrição em RDF dos objetos de aprendizagem, seguindo as recomendações e princípios *Linked Data* para padrões de metadados de objetos de aprendizagem.
- Marcação de conceitos e entidades presentes nos objetos de aprendizagem conforme ontologia específica para Tagging.
- Construção de um repositório RDF de objetos de aprendizagem.
- Apresentação do uso de técnicas de mineração de texto na identificação de conceitos e entidades presentes em texto.
- Construção de novas formas de acesso aos objetos de aprendizagem, por meio da descrição estrutura em RDF dos objetos, segundo os princípios da Web de Dados.

6.2. TRABALHOS FUTUROS

Como trabalhos futuros, são apresentadas as seguintes sugestões:

- Aperfeiçoamento do módulo de busca, ampliando a possibilidades de busca e explorando os recursos da descrição semântica dos objetos de aprendizagem.
- Possibilitar a escolha de ontologia específica para o domínio do conteúdo a ser anotado, de maneira a aperfeiçoar o mecanismo de descoberta de conceitos. Por exemplo, se o objeto de aprendizagem que está sendo criado for relacionado à área de medicina, restringir a busca de conceitos a serem anotados, somente ao domínio de medicina.
- Desenvolver mecanismo de sugestão automática de conteúdo relacionado, a partir da analise do domínio das anotações, sugerir ao usuário outros conteúdos que podem auxiliar processo de aprendizagem.
- Criação de nuvem de *tags* semântica (*semantic tag cloud*), possibilitando

relacionar o conteúdo anotado.

- Integração do *Web service* à outros ambientes de aprendizagem ou repositórios de objetos de aprendizagem, potencializando o reuso e integração entre diversas plataformas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

AGIRRE, E; EDMONDS, P. Word sense disambiguation: algorithms and applications: chapter introduction. **Springer Science and Business Media Journal,** New York, v.16, n. 16, p.1-2, novembro, 2007. Disponível em: <http://www.springer.com/br>. Acesso em: 24 out 2014.

ARAUJO, Moysés de. **Educação à distância e a *web* semântica:** modelagem ontológica de materiais e objetos de aprendizagem para a plataforma, 2003. 178f, il. Tese (Doutorado em Informática) - Universidade de São Paulo, São Carlos –SP, 2004.

AUER, Sören; BIZER, Christian; KOBILAROV, Georgi; et al. DBpedia: a nucleus for a web of open data. In: SEMANTIC WEB AND ASIAN CONFERENCE ON ASIAN SEMANTIC WEB CONFERENCE, 6, 2007, Heidelberg, Berlin. **Proceedings**…. Heidelberg, Berlin: Springer-Verlag, 2007, p.722–735. Disponível em: <http://dl.acm.org/citation.cfm?id=1785162.1785216>. Acesso em: 08 ago. 2014.

BAMIDIS, P.D. et al. Federating learning management systems for medical education: Aapersuasive technologies perspective. In: INTERNATIONAL SYMPOSIUM ON COMPUTER MEDICAL SYSTEMS, 24, 2011, Baltimore. **Proccedings**..... Baltimore: CSB, 2011.p.1,6, 27-30.

BAX, Marcelo Peixoto. Introdução às linguagens de marcas**. Revista Ciência da Informação**, Brasília-DF, v.30, n.1, p.32-38, jan./abr. 2001.

BECHHOFER, S, et al. **OWL web ontology language reference.**Disponível em http://www.w3.org/TR/owl-ref/>. Acesso em: 27 nov. 2013.

_____. **OWL**: web ontology language.Disponível em:

<http://link.springer.com/referenceworkentry/ 10.1007/ 978-0-387-39940-9_1073?no-access=true>. Acesso em: 02 nov. 2014.

BERNERS-LEE, T. **Linked data. design issues about web architecture.** Disponível em <http://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 15 mai. 2013.

_____. HENDLER, J.; LASSILA, O. The semantic web. **Scientific American**. New York, v.284, p. 34-43, 2001. Disponível em: <http://www.scientificamerican.com/article/the-semantic-web/>. Acesso em: 22 maio 2014.

BOMFIM, Mauricio Henrique de Souza**. Um método e um ambiente para o desenvolvimento de aplicações na we*b* semântica**. 2011. 196 f. Dissertação (Mestrado em Informátia) - Pontifícia Universidade Católica, Rio de Janeiro, 2012.

BOOTH , David; et al. Web Services Arquiteture. In: W3C. **Working group note**, 11 February 2004. Disponível em: <http://www.w3.org/TR/ws-arch>. Acesso em: 16 set. de 2014.

BREITMAN, K.; et al. **Publicação de dados governamentais no padrão linked data**. Disponível em: <http://www.w3c.br/cursos/dados-abertos/curso/Parte-3-Modulo-1-SPARQL.pdf>. Acesso em: 15 ago. 2014.

BRICKLEY D; GUHA, R.V. **RDF schema 1.1**: W3C recommentation.Disponível em: <http://www.w3.org/TR/rdf-schema/>. Acesso em: 01 set. 2014.

BROEKSTRA, Jeen; KAMPMAN, Arjohn; Harmelen, Frank van. **Sesame**: a generic architecture for storing and querying RDF and RDF schema.Disponível em: <http://link.springer.com/chapter/10.1007%2F3-540-48005-6_7>. Acesso em: 15 ago. 2014.

BURKE, Kevin; et al. **Flask-RESTful**. Disponível em: <http://flask-restful.readthedocs.org/en/latest/>. Acesso em: 12 ago. 2014.

CASTILLO, P. A; et al. Using SOAP and REST *web* services as communication protocol for distributed evolutionary computation. **International Journal of Computers and Technology**, New York, v.10, n. 10, p. 1659-1677, Jully, 2012. . 2013. Disponível em: <http://cirworld.org/journals/index.php/ijct/article/view/2158>. Acesso em 18 abr. 2014.

CATAPAN, Araci Hack; SILVA, Edna Lúcia da; CAFÉ, Lígia Arruda. Definição de metadados para o repositório de objetos de aprendizagem da EaD - UFSC. **Encontros Bibli: Revista Eletrônica de Biblioteconomia e Ciência da informação**, São Carlos – SP, v. 15, n. 29, 2010. Disponível em: <https://periodicos.ufsc.br/index.php/eb/article/view/13701>. Acesso em: 19 out. 2014.

CLARK, K. G.; FEIGENBAUM, L.; TORRES, E. **SPARQL protocol for RDF**: W3C recommendation. (2008) Disponível em : <http://www.w3.org/TR/rdf-sparql-protocol/>. Acesso em 14 set. 2014

CHAKAM, T.; LEE, T. **Math cast**: an open source equation editor. Disponível em <http://mathcast.sourceforge.net/home.html>. Acesso em 20 set. 2014.

CHAMPION, Michael; et al. **Web services architecture** . Disponivel em: <http://www.w3.org/TR/2002/WD-ws-arch-20021114/> Acesso em: 15 set 2014.

COOPER, Philip. **RDF alchemy** : sesame endpoints. Disponível em: <http://www.openvest.com/trac/wiki/RDFAlchemy>. Acesso em: 20 ago. 2014.

_____; HIGGINS, Graham. **Welcome to RDF alchemy's documentation**. Disponível em: <http://rdfalchemy.readthedocs.org/en/latest/>. Acesso em: 15 ago. 2014.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas distribuídos**: conceitos e projeto. 4. ed. Porto Alegre: Bookman, 2007.

CYGANIAK, Richard; BIZER, Chris. **Pubby**: a Linked data frontend for SPARQL endpoints. Disponível em: <http://wifo5-03.informatik.uni-mannheim.de/pubby/>. Acesso em: 17 ago. 2014.

_____; JENTZSCH, A. **The linking open data cloud diagram.** Disponível em: < http://lod-cloud.net/>. Acesso em: 18 out. 2014.

COELHO, Sandro. **DBpedia spotlight**: github. Disponível em: <https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki/Introduction> . Acesso em: 24 ago. 2014.

DAIBER, J; et al. . Improving efficiency and accuracy in multilingual entity extraction.

In: INTERNATIONAL CONFERENCE ON SEMANTIC SYSTEMS, 9, june 6-8, 2013, Seatlle. **Proceedings...** Seatlle: I-Semantics. Disponível em: <http://jodaiber.de/doc/entity.pdf>. Acesso em: 25 de jul. 2014.

DBPedia. **The DBpedia knowledge base**. Disponível em: <http://www.dbpedia.org/About>. Acesso em: 08 nov. 2014.

DOUSH, I. A.; ALKHATEEB, F.; Al Maghayreh, E; ALSMADI, I,; SAMARAH, S. Annotations, Collaborative Tagging, and Searching Mathematics in E-Learning. International Journal of Advanced Computer Science and Applications (IJACSA) 2(4). (2011) Disponível em: <http://arxiv.org/pdf/1211.1780.pdf> Acesso em 21 Set 2014

DOVROLIS, N.; STEFANUT, T. ; DIETZE, S. ; YU, H.Q. ; VALENTINE, C. ; KALDOUDI, E., Semantic Annotation and Linking of Medical Educational Resources; 5th European IFMBE Conference, IFMBE Proceedings 37, pp. 1400–1403, 2011

DUBLIN CORE (DCMIa). **Dublin core metadata element set**: version 1.1: reference description. Disponível em: <http://dublincore.org/documents/dces/>. Acesso em: 18 out. 2014.

_____(DCMIb).**Dublin core metadata terms**. Disponível em: <http://dublincore.org/documents/dcmi-terms/>. Acesso em: 18 out. 2014.

EBY, P. **Python web server gateway interface v1.0**: PEP 333. Disponível em <http://legacy.python.org/dev/peps/pep-0333/> Acesso em 18 out 2014.

ELIAS, E; HOLANDA, O. **SPARQL**: linguagem de consulta em ontologias. Maceió: Universidade Federal de Alagoas . Disponível em: <http://www.egov.ufsc.br/portal/sites/default/files/sparqlrevisado.pdf>. Acesso em: 09 set. 2014.

EISENBERG, A; et al. **SQL**: 2003 has been. New York: Sigmod Record 2004.

FEIGENBAUM, L; et al. **W3C - SPARQL 1.1 protocol**: W3C recommendation. Disponível em: <http://www.w3.org/TR/sparql11-protocol/>. Acesso em: 20 ago. 2014.

FIRMINO, F; et al. **Tutorial sesame**. Disponível em: <http://greco.ppgi.ufrj.br/ lodbr/index.php/principal/ tutorial/sesame/>. Acesso em 14 set. 2014.

FIELDING, Roy Thomas. **Architectural styles and the design of network-based software architectures**. 2000, 200 f. Dissertation (Doctor Of Philosophy In Information And Computer Science) - University Of California, Irvine, California - Us, 2001. Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 20 set. 2014.

GHELMAN, Raphael. **Extensão de um sistema de integração de repositórios de objetos de aprendizagem visando a personalização das consultas com enfoque em acessibilidade**. 2006. 128 f. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007. Disponível em: <http://www.maxwell.vrac.puc-rio.br/Busca_etds.php?strSecao=resultado&nrSeq=9147@1 >. Acesso em: 10 de out. 2014.

GIL, YOLANDA; et al. **Provenance XG final report**. Disponível em: <http://www.w3.org/2005/Incubator/prov/XGR-prov-20101214/>. Acesso em: 21 out 2014.

GIRARDI, Reubem Alexandre Almeida. **Framework para coordenação e mediação de web services modelados como learning objects para ambientes de aprendizagem na *Web*.** 2004. 111 f. Dissertação (Mestrado em Informática) - Pontifícia Universidade Católica do Rio de Janeiro, Rio, Rio de Janeiro, 2005.

GLONVEZYNSKI, Régis Alessandro. **Modelo de anotação de documentos para a codificação do conteúdo semântico no processo de autoria**. 2008. 112 f Dissertação (Mestrado em Informática) - Universidade Federal de Santa Catarina, Florianópolis, 2009.

GOMES, Geórgia Regina Rodrigues. **Integração de repositórios de sistemas de bibliotecas digitais e de sistemas de aprendizagem**. 2006. 143 p, il. Tese (Doutorado em Informática) Pontifícia Universidade Católica do Rio de Janeiro, Rio, Rio de Janeiro, 2006.

GOMES, Sionese Rocha; et al. Objetos de aprendizagem e as limitações dos metadados atuais. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 16, 2005, Juiz de Fora. **Anais....** Juiz de Fora: SBIE /UFJF, 2006. Disponível em: http://br-ie.org/pub/index.php/sbie/article/view/406/392.

GRISHMAN, Ralph. Information extraction: techniques and challenges. In: INTERNATIONAL SUMMER SCHOOL SCIE, 9 , 1997, New York. **Proceedings...** New York : Springer-Verlag, 1998. p. 10-27.

GUDGIN, M; . **SOAP version 1.2 part 1**: messaging framework . Disponível em: <http://www.w3.org/TR/soap12-part1/> Acesso em: 09 set. 2014

HARMELEN, F. V; MCGUINNESS, D. L. **OWL**: web ontology language overview. Disponível em: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>, Acesso em: 15 abr. 2014.

HAWKE, S.; HERMAN, I.; ARCHER, P.; PRUD'HOMMEAUX, E. **Semantic web activity latest layercake diagram**. Disponível em: <http://www.w3.org/2001/sw/>. Acesso em: 13 abr. 2014.

HAHN, Rasmus; et al. Faceted Wikipedia search. In: ABRAMOWICZ, Witold (org);

HEARST, Marti **.What is text mining?** Berkeley: SIMS UC, 2003.

HIDEKI, Eric. Armin Ronacher. **Pythonistas que você devia conhecer.** Disponível em: <https://ericstk.wordpress.com/tag/jinja2/>. Acesso em: 11 ago. 2014.

HOHPE, Gregor; WOOLF, Bobby. **Enterprise integration patterns: designing, building, and deploying messaging solutions**. Toronto, Canada : Addisson Wesley, 2003.

IEEEXPLORE. **information technology** : learning technology : learning objects metadata LOM: working drafty. resource meta-data specification**.** Disponível em: <http://ltsc.ieee.org/wg12/index.html>. Acesso em 25 mar. 2014.

_____.**Standard for learning object metadata**. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1032843>. Acesso em: 19 out. 2014.

KAMALELDIN, M.; DUMINDA, W. Performance analysis of web services on mobile devices. **Procedia Computer Science**, New York, v. 10, p. 744-751, 2012.

KENDALL, G. C., FEIGENBAUM, L., TORRES, E: **SPARQL protocol for RDF**: W3C recommendation. Disponível em: <http://www.w3.org/TR/rdf-sparqlprotocol/>. Acesso em: 29 ago. 2014.

KOBILAROV, G.; et al. . Dbpedia: a linked data hub and data source for web

applications and enterprises. In: International World Wide Web Conference, 18,2009, 20-24 april, Madrid, Spain. **Proceedings….** Madrid, Spain: IWWWC, 2010. Disponível em: <http://www.*Web*semanticsjournal.org/index.php/ps/article/view/319/319>. Acesso em: 25 jul. 2014.

KOBILAROV, Georgi; SCOTT, Tom; RAIMOND, Yves; et al. Media meets semantic web: how the BBC uses DBpedia and linked data to make connections. In: EUROPEAN SEMANTIC WEB CONFERENCE ON THE SEMANTIC WEB, 6, 2009, Heidelberg, Berlin. **Proceedings**… Heidelberg, Berlin: Springer-Verlag, 2010. Disponível em: <http://dx.doi.org/10.1007/978-3-642-02121-3_53>. Acesso em: 10 jul. 2014.

KODALI, R.R. **What is service-oriented architecture?**. Disponível em : < http://www.javaworld.com/article/2071889/soahat-is-service-oriented-architecture/soa/what-is-service-oriented-architecture.html>. Acesso em 09 set. 2014.

LEHMANN, Jens; et al. Dbpedia: a large-scale, multilingual knowledge base extracted from wikipedia. **Semantic Web Journal**, Toronto, v. 20. n.19, p.70-80. April, 2014. Disponível em: <http://www.semantic-*Web*-journal.net/system/files/swj558.pdf> Acesso em 18 ago. 2014.

LINTHICUM, David S. **Next generation application integration: from simple information to *web* services**. New York: Addisson Wesley, 2003.

LOPES, Lucelene. **Extração automática de conceitos a partir de textos em língua portuguesa**. 2011. 156f. Tese (Doutorado em Educação) - Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2012.

LOPES, Maria Célia Santos. **Mineração de dados textuais utilizando técnicas de clustering para o idioma português,** 2004. 180f Tese (Doutorado em Informática) - Universidade Federal do Rio de Janeiro, 2005.

LOHMANN, Steffen. **Modular Unified Tagging Ontology** (MUTO): specification of MUTO core. Disponível em: <http://muto.socialtagging.org/core/v1.html>. Acesso em: 18 ago. 2014.

MARTIN, Joanne; et al. **Web services**: promises and compromises. Disponível em: <http://doi.acm.org/10.1145/637958.639315> . Acesso em: 18 ago. 2014.

MCGUINNESS, Deborah L; HARMELEN, Frank van. **OWL**: web ontology language overview : W3C Recommendation . Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em: 19 out. 2014.

MENDES, Pablo; JACOB, Max; HAYNES, Matt. **DBpedia lookup**. Disponível em: <https://github.com/dbpedia/lookup>. Acesso em 04 de set. 2014.

_____; DAIBER, Jo. **DBpedia spotlight**: gitHub. Disponível em: <https://github.com/dbpedia-spotlight/dbpedia-spotlight>. Acesso em: 13 nov. 2014.

_____;GARCÍA-SILVA, Andrés. DBpedia spotlight: shedding light on the web of documents. In:International In: CONFERENCE ON SEMANTIC SYSTEMS.7, 2011, New York. **Proceedings...** New York: I-Semantics, 2012. Disponível em: <http://doi.acm.org/10.1145/2063518.2063519>. Acesso em: 19 out. 2014.

MENDONÇA, Rogers Reiche de. **Uma abordagem para coleta e publicação de dados de proveniência no contexto de linked data**. 2013. 143 f. Dissertação (Mestrado em Informática) - Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2014.

_____; et al. Gerência de proveniência multigranular em linked data com a abordagem ETL4LinkedProv. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 29, 2013, 06-07 out, Curitiba. **Anais...** Curitiba: SBBD, 2014. Disponível em <http://www.inf.ufpr.br/sbbd-sbsc2014/sbbd/proceedings/artigos/pdfs/81. pdf>. Acesso em 22 out. 2014

MOODLE.ORG. **Repository plugins**: Moodle docs. Disponível em: <https://docs.moodle.org/dev/Repository_plugins>. Acesso em: 12 nov. 2014.

MONTEIRO, L. de O; GOMES, I. R; OLIVEIRA, T. Etapas do processo de mineração de textos – uma abordagem aplicada a textos em português do Brasil. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 26, 2006, Campo Grande. **Anais....** Campo Grande: UFMS/UCDB, 2007.

MORSEY, Mohamed; et al. **DBpedia and the live extraction of structured data from Wikipedia program**: electronic library and information systems. Disponível em <http://jens-lehmann.org/files/2012/program_el_dbpedia_live.pdf>. Acesso em: 10 ago. 2014.

MOULIN, Bernard; ROUSSEAU, Daniel. Automated Knowledge Acquisition from

regulatory Texts. **IEEE Expert**, Los Alamitos, v. 7, n. 2, p. 27-35, oct, 1992.

MOURA, Simone Leal de. **Uma arquitetura para integração de repositórios de objetos de aprendizagem baseada em mediadores e serviços web**. 2005. 158 f Dissertação (Mestrado em Informática) - Pontíficia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

MYSQL COMMUNITY EDITION. Disponível em: <http://www.mysql.com/products/community/>. Acesso em: 13 nov. 2014.

NEWCOMER, Eric. **Understanding web services**: XML, WSDL, SOAP, and UDDI. Toronto: Addison Wesley, 2002. 339 p.

PICKLER, M. E. V; Web semântica: ontologias como ferramentas de representação do conhecimento. **Perspectivas em Ciência da Informação**, São Paulo, v.20, n.19,p 83-90, abril-maio, 2006. Disponível em: <http://www.scielo.br/scielo.php?script=sci_serial&pid=1413-9936&lng=en&nrm=iso> Acesso em: 20 maio 2014.

OGBUJI, Chimezie. **SPARQL 1.1 graph store http protocol**: W3C recommendation. . Disponível em: <http://www.w3.org/TR/sparql11-http-rdf-update/>. Acesso em: 14 set. 2014.

OPENRDF SESAME. **Community website**. Disponível em: <http://rdf4j.org/>. Acesso em: 08 nov. 2014.

_____.(OPENRDFa). **User documentation: sesame 2.7**. Disponível em: <http://openrdf.callimachus.net/sesame/2.7/docs/users.docbook?view>. Acesso em: 14 set. 2014.

_____. (OPENRDFb). **System documentation for sesame 2**. Disponível em: <http://openrdf.callimachus.net/sesame/2.7/docs/system.docbook?view>. Acesso em: 14 set. 2014.

_____. (OPENRDFc).**Creating a repository object**. Disponível em: <http://rdf4j.org/sesame/2.7/docs/articles/repository-api/creating-repositories.docbook? view>. Acesso em: 15 out. 2014.

OREN, E. **What are semantic annotations*?*** Disponível em: <http://www.siegfried-

handschuh.net/pub/2006/whatissemannot2006.pdf>. Acesso em: 05 mar. 2014.

POPOV, B; Towards semantic web information extraction. IN; INTERNATIONAL SEMANTIC WEB CONFERENCE, 2, 2003,Sanibel Island, Florida. **Proceedings....** Sanibel Island, Florida: WC, 2004. . Disponível em: <http://gate.ac.uk/conferences/iswc2003/proceedings/popov.pdf>. Acesso em: 2 abr. 2014.

POSTGRESQL. **PostgreSQL**: the world's most advanced open source database. Disponível em: <http://www.postgresql.org/>. Acesso em: 13 nov. 2014.

PRUD'HOMMEAUX, E., SEABORNE, A: **SPARQL query language for RDF**: W3C recommendation. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Acesso em: 12 ago. 2014.

RAJABI, Enayat; et al. . **Recommendation on exposing IEEE LOM as linked data 0.9**. Disponível em: <http://data.organic-edunet.eu/ODS_LOM2LD/ODS_FirstDraft.html>. Acesso em: 18 ago. 2014.

RAY R.J., KULCHENKO P. **Programming web services with perl**. Toronot: O'Reilly: Sebastapol, 2002 .

RDF: SEMANTIC WEB STANDARDS. **Resource description framework**. Disponível em <http://www.w3.org/RDF/>. Acesso em: 08 nov. 2011.

_____. **Rdflib 4.2-dev documentation**. Disponível em: <http://rdflib.readthedocs.org/en/latest/>. Acesso em: 24 out. 2014.

RONACHER, Armin. **Flask web development one drop at a time**. (2014a). Disponível em: <http://flask.pocoo.org/>. Acesso em: 11 ago. 2014.

_____. **Jinja**. (2014c). Disponível em: <http://jinja.pocoo.org/>. Acesso em: 11 ago. 2014.

_____. **Werkzeug**: the python WSGI utility library. (2014b). Disponível em: <http://werkzeug.pocoo.org/>. Acesso em: 11 ago. 2014.

ROCHA, Bruno. **What the flask Pt-1**: introdução ao desenvolvimento *web* com

python. Disponível em: <http://pythonclub.com.br/what-the-flask-pt-1-introducao-ao-desenvolvimento-*Web*-com-python.html>. Acesso em: 11 ago. 2014.

SAHAI, A.; MACHIRAJU, V.; OUYANG, J.; WURSTER, K. Message tracking in SOAP-based *web* services. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM NOMS, 7, 2002, 15-19 April, Florence, Italy. **Proceedings....** Florence, Italy.: 2003. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.7161&rep=rep1&type=pdf>. Acesso em 20 set. 2014.

SAHNWALDT, Christopher. **The DBpedia data**. Disponível em <http://wiki.dbpedia.org/Datasets>. Acesso em: 15 ago. 2014.

_____. **The DBpedia ontology**. Disponível em <http://wiki.dbpedia.org/Ontology>. Acesso em: 21 ago. 2014.

SANDHU, P. **The math ml handbook**: mathematical markup language. Disponível em <http://www.w3.org/Math/>. Acesso em: 21 set. 2014.

SILVA, E. L. D.; CAFÉ, L.; CATAPAN, A. H. Os objetos educacionais, os metadados e os repositórios na sociedade da informação**. Revista Ciência da Informação**, Brasília, v. 39, p. 93-104, 2010. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S010019652010>.

SMITH, M. K; WELTY, C; MCGUINNESS, D. L. **OWL**: web ontology language guide. Disponível em: <http://www.w3.org/TR/2004/REC-owl-guide-20040210/>. Acesso em: 5 abr. 2014.

SOUZA, M. I. F.; VENDRUSCULO, L. G.; MELO, G. C. Metadados para a descrição de recursos de informação eletrônica: utilização do padrão Dublin Core. **Revista Ciência da Informação**, Brasília, v. 29, n. 1, p. 93-102, abr. 2000. Disponível em: <http://www.scielo.br/pdf/ci/v29n1/v29n1a10.pdf>. Acesso em: 10 jun. 2014.

SOUZA, Renato Rocha; ALVARENGA, Lídia. A web semântica e suas contribuições para a ciência da informação. **Revista Ciência da Informação**., Brasília, v. 33, n. 1, p. 132-141, jan./abr. 2004.

SPORNY, Manu; et al. **JSON-LD 1.0**. Disponível em: <http://www.w3.org/TR/json-ld/>. Acesso em: 10 nov. 2014.

SEMATIC WEB.ORG (SW). **Semantic web: sparql endpoint**. Disponível em: <http://semantic*Web*.org/wiki/SPARQL_endpoint>. Acesso em: 04 set. 2014.

TALIS, Sir Tim Berners-Lee. **Talks with talis about the semantic web**. Disponível em: <http://talis-podcasts.s3.amazonaws.com/twt20080207_TimBL.html>. Acesso em: 30 nov. 2013.

TAROUCO, Liane Margarida Rockenbach ; SCHMITT, Marcelo Augusto Rauh. Adaptação de Metadados para Repositórios de Objetos de Aprendizagem. RENOTE - **Revista Novas Tecnologias na Educação.** Porto Alegre, v. 8, n 2, 2010.

_____.Sistemas de gestão de conteúdo para objetos de aprendizagem: características desejáveis e soluções existentes**. Encontros Bibli: Revista Eletrônica de Biblioteconomia e Ciência da Informação**, Brasília, v. 15, n. 29, 2010. Disponível em: <https://periodicos.ufsc.br/index.php/eb/article/view/13700>.

TESTER, D. **RDFS vs. OWL**: Cambridge semantics. Disponível em: <http://www.cambridgesemantics.com/semantic-university/rdfs-vs.-owl>. Acesso em: 25 ago. 2014.

THIBODEAU, Ted. **The DBpedia knowledge base**. Disponível em: <http://wiki.dbpedia.org/About> Acesso em: 19 ago. 2014.

TOLKSDORF, Robert (org). **Business information systems**. Heidelberg, Berlin : Springer. Disponível em: <http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/hahn-etal-faceted-wikipedia-search-BIS2010.pdf>. Acesso em: 30 jul. 2014.

VARLAMIS I.; APOSTOLAKIS I. The present and future of standards for E-learning Technologies**. Interdisciplinary Journal of Knowledge and Learning Objects**, Toronto, v. 2, n.2, p.18-21, july-august, 2006. Disponível em: <http://www.informingscience.us/icarus/journals/ijello>. Acesso em 15 nov 2014.

WILEY, Dusavid A. **Connecting learning objects to instrutional desing theory: a definition, a metaphor, and a taxonomy**. Logan – UT: Utah State University , 2000. Disponível em: <http://reusability.org/read/chapters/wiley.doc>.

WORLD WIDE WEB CONSORTIUM (W3Ca). Disponível em:

<http://www.w3.org/2001/sw/wiki/DBpedia_Spotlight> . Acesso em: 24 ago. 2014.

_____. **OWL working group**: Web Ontology Language (OWL). Disponível em: <http://www.w3.org/2001/sw/wiki/OWL>. Acesso em: 13 fev. 2014.

_____. (W3Cb). **OWL working group**: OWL 2 web ontology language: document overview. Disponível em: <http://www.w3.org/2001/sw/wiki/OWL>. Acesso em: 13 fev. 2014.

_____. **W3C recommendation**. Disponível em: <http://www.w3.org/TR/owl2-overview/.>. Acesso em: 05 mai. 2014.

_____. (W3Cc). **SPARQL working group**: SPARQL 1.1 overview: W3C recommendation. Disponível em: <http://www.w3.org/TR/sparql11-overview/>. Acesso em: 02 set. 2014.

XMLRPC.NET. **Simple cross-platform distributed computing, based on the standards of the internet**. Disponível em: <http://xmlrpc.scripting.com/default.html> . Acesso em: 08 nov. 2014.

ZHU, Hongwei; MADNICK, Stuart E. Scalable interoperability through the use of COIN lightweight ontology. In: CONFERENCE ON ONTOLOGIES-BASED DATABASES AND INFORMATION SYSTEMS, 2, 2007, Heidelberg, Berlin. **Proceedings....** Heidelberg, Berlin: Springer-Verlag, 2008. Disponível em: <http://dl.acm.org/citation.cfm?id= 1894726.1894729>. Acesso em: 18 out. 2014