CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U nº 198, de 14/10/2016
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Matheus Rodrigues Leal

UTILIZAÇÃO DE UMA ONTOLOGIA DE DOMÍNIO PARA A ORGANIZAÇÃO DE INFORMAÇÕES DOS PERFIS DE PROFESSOR E ALUNO DE UM PORTAL ACADÊMICO

Palmas – TO

2017

Matheus Rodrigues Leal

UTILIZAÇÃO DE UMA ONTOLOGIA DE DOMÍNIO PARA A ORGANIZAÇÃO DE INFORMAÇÕES DOS PERFIS DE PROFESSOR E ALUNO DE UM PORTAL ACADÊMICO

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Prof. M.e Parcilene Fernandes de Brito.

Palmas – TO

2017

Matheus Rodrigues Leal

UTILIZAÇÃO DE UMA ONTOLOGIA DE DOMÍNIO PARA A ORGANIZAÇÃO DE INFORMAÇÕES DOS PERFIS DE PROFESSOR E ALUNO DE UM PORTAL ACADÊMICO

Projeto de Pesquisa elaborado e apresentado como requisito parcial para aprovação na disciplina de Trabalho de Conclusão de Curso II (TCC II) do curso de bacharel em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Prof. M.e Parcilene Fernandes de Brito.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Parcilene Fernandes de Brito

Orientadora

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2017

RESUMO

LEAL, Matheus Rodrigues. **Utilização de uma ontologia de domínio para a organização de informações dos perfis de professor e aluno de um portal acadêmico**. 2017. XX f. Trabalho de Conclusão de Curso (Graduação) – Curso de Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas/TO, 2017.

Este trabalho objetiva o desenvolvimento de uma ontologia de domínio para auxiliar no processo de organização semântica das informações relacionadas aos perfis de professor e aluno de um portal acadêmico de uma instituição de ensino superior. São apresentados alguns conceitos relacionados à Web Semântica, dando ênfase às ontologias e suas linguagens utilizadas para a representação de conhecimento. Além disso, são apresentados alguns trabalhos relacionados ao contexto que fazem o uso de técnicas de mapeamento de modelos de banco de dados relacionais em ontologias, que auxiliam no processo de construção e preenchimento das informações de uma ontologia. Em conjunto com a utilização dessas técnicas, a metodologia do trabalho se baseia em um procedimento estruturado pela universidade de Stanford para a criação de ontologias. Para divulgar os resultados do trabalho, será desenvolvida uma área de apresentação das informações relacionadas aos perfis de professor e aluno no portal acadêmico do Centro Universitário Luterano de Palmas.

PALAVRAS-CHAVE: Ontologia, Organização de informações, Portal Acadêmico.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

BMIR – *Biomedical Informatics Research*

CAPES – Coordenação de Aperfeiçoamento de Pessoa de Nível Superior

CEULP – Centro Universitário Luterano de Palmas

DAML – *DARPA Agent Markup Language*

FOAF – *Friend of a Friend*

MEC – Ministério da Educação

OIL – *Ontology Inference Layer*

OWL – *Web Ontology Language*

RDF – *Resource Description Framework*

RDFS – *Resource Description Framework Schema*

RIF – *Rule Interchange Format*

SQL – *Structured Query Language*

URI – *Uniform Resource Identifier*

URL – *Uniform Resource Locator*

W3C – *World Wide Web Consortium*

XML – *Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

No início da década de 2000, Tim Berners-Lee descreveu sua visão acerca de uma web diferente da que é mundialmente conhecida e utilizada hoje em dia. Esta visão de uma nova web foi descrita como "uma versão totalmente estruturada da web atual, tanto semântica quanto sintaticamente" (ZHANG, 2007, pág. 9). Esta versão foi batizada de Web Semântica e, posteriormente, foi devidamente aceita e regulamentada pela *World Wide Web Consortium* (W3C). Ela foi definida como um ambiente web estruturado por meio de padrões de linguagem, integrado com várias fontes de informação distintas e colaborativo, de forma que fosse gerenciado pelos próprios usuários. A Web Semântica enfatiza a organização das informações, com o intuito de proporcionar um melhor entendimento dos conceitos apresentados. Para garantir este nível de organização, foram estabelecidas algumas ferramentas e tecnologias para a publicação de conteúdo na Web Semântica. As estruturas definidas como modelos para representação do conhecimento são as ontologias, que geralmente estão relacionadas a um domínio específico.

"Ontologias de domínio objetivam a redução (ou eliminação) de qualquer tipo de confusão conceitual ou terminológica entre os membros de uma comunidade virtual de usuários" (NAVIGLI e VELARDI, 2004, pág. 151). De acordo com essa definição, é possível observar que a utilização de ontologias de domínio pode proporcionar uma série de benefícios. Segundo Gavrilova, Gorovoy e Petrashen (2009), alguns desses benefícios são:

- completude;
- precisão;
- adequação cognitiva;
- eliminação de excessos e contradições; e
- diminuição da complexidade do contexto.

A **completude** se refere à capacidade de representação de informações suficientes para a interpretação de mais diversas características de um domínio. A **precisão** se refere à confiabilidade das informações apresentadas, devido à possibilidade de definir uma série de restrições relacionadas aos objetos e propriedades do domínio. **Adequação cognitiva** refere-se ao estabelecimento de um consenso acerca das definições dos conceitos dentro de um domínio. A **eliminação de excessos e contradições** está relacionada ao fato de as ontologias auxiliarem na eliminação de redundâncias e ambiguidade. A di**minuição da complexidade do contexto** se refere à possibilidade de visualização do domínio em diversas escalas, de forma mais ampla ou mais específica, de acordo com a preferência do usuário. Além disso,

Alalwan, Zedan e Siewe (2009) também citam mais um conjunto de benefícios trazidos pelas ontologias:

- adição de características semânticas aos dados, de forma que possam ser interpretadas mais facilmente por computadores;
- compartilhamento de uma única perspectiva semântica acerca das informações apresentadas;
- separação entre o conhecimento do domínio e o conhecimento operacional; e
- realização de suposições explícitas acerca do domínio com base nas restrições estabelecidas entre os dados.

Portanto, pode-se observar que a utilização de uma ontologia de domínio é capaz de auxiliar no entendimento das informações por parte dos usuários.

"Portais acadêmicos são uma das ferramentas mais importantes para a comunicação entre as instituições e o público. Devido ao grande volume de informação, tais ambientes virtuais devem carregar características de boa usabilidade e navegabilidade" (VIEIRA, 2014, pág. 2).

A maioria dos portais acadêmicos de instituições de ensino superior têm o objetivo de apresentar para a comunidade a estrutura física da instituição, os cursos oferecidos, o corpo docente que os compõem, as matrizes curriculares dos cursos, as atividades e eventos desenvolvidos, dentre outras. No entanto, devido a esse grande volume de dados, a organização das informações no que tange a semântica acaba sendo comprometida, pois não há uma definição clara de seus conceitos e relações. Um dos problemas gerados a partir dessa situação é a falta de conexão entre as informações apresentadas, que pode implicar na dificuldade de entendimento e identificação de características específicas. Por exemplo, uma página apresenta uma lista dos professores vinculados a cada um dos cursos da instituição e outra página, de um determinado curso, também apresenta a lista de seus respectivos professores. No entanto, as informações presentes em ambas as páginas apresentam algumas divergências.

Em um Portal Acadêmico de uma Instituição de Ensino Superior, os perfis de professor e aluno estão relacionados a uma série de fatores, por exemplo, cursos, disciplinas, turmas, grupos de pesquisa, atividades extensionistas etc. Portanto, a organização desses elementos a partir da definição de seus conceitos e relações pode tornar possível uma apresentação de informações de forma mais clara e intuitiva. Além disso, pode tornar possível a identificação de relações que não são tão explícitas, como grupos de pesquisa ou projetos de extensão que

possuem professores e alunos de múltiplos cursos, o que evidencia a interdisciplinaridade, por exemplo.

No entanto, o desenvolvimento de uma ontologia para auxiliar no processo de organização de um portal acadêmico pode ser um processo inviável, ao considerar alguns aspectos. Uma vez que o portal acadêmico já esteja em funcionamento, é possível que as informações disponibilizadas por meio dele estejam armazenadas em uma ferramenta que, na maioria das vezes, consiste em um banco de dados relacional. Logo, ao desenvolver uma ontologia para o domínio, todas as informações existentes nesse banco de dados relacional precisariam ser inseridas em uma nova ferramenta, que é responsável pelo gerenciamento da ontologia. Esse processo pode requerer tempo, esforço e gastos consideráveis, dependendo da quantidade de informações existentes inicialmente no banco de dados vinculado ao portal acadêmico. Além disso, seria necessário modificar a implementação do portal acadêmico para substituir a utilização da base de dados relacional pela ontologia.

Com base nisso, o problema que este trabalho procura resolver está relacionado à criação de uma ontologia para representar o perfil de professor e aluno de uma instituição de ensino superior e apresentar as informações resultantes em uma página web. Levantando como hipótese que, se houver um entendimento dos conceitos e relações presentes no modelo de banco de dados relacional que compõe o contexto de um portal acadêmico, então é possível criar uma ontologia de domínio relacionada ao perfil de professor e aluno e implementar um mecanismo para a apresentação das informações.

Portanto, além do desenvolvimento de uma ontologia de domínio relacionada aos perfis de professor e aluno, este trabalho objetiva representar a ontologia usando um padrão de representação da Web Semântica. Além disso, para a apresentação dos resultados, este trabalho também tem como objetivo desenvolver um protótipo de tela, baseada na utilização de ontologias, para a apresentação das informações relacionadas aos perfis de professor e aluno.

Este trabalho é estruturado da seguinte forma: seção 1, Introdução, apresenta o problema, hipótese, objetivos e justificativa que permeiam o trabalho; seção 2, Referencial Teórico, aborda os conceitos relacionados ao desenvolvimento do trabalho, identifica outros contextos de aplicação desses conceitos e discute alguns trabalhos relacionados à temática utilizada; seção 3, Metodologia, enumera e fornece uma breve introdução aos materiais e tecnologias utilizadas, apresenta e explica a metodologia escolhida para o desenvolvimento do trabalho, que consiste em uma adaptação da proposta de Noy e McGuinness (2001), da Universidade de Stanford; seção 4, Cronograma, apresenta como foram distribuídas as tarefas relacionadas ao

desenvolvimento do trabalho no decorrer do tempo; e, posteriormente, são listadas as referências que serviram como base para a construção do trabalho.

2 REFERENCIAL TEÓRICO

Esta seção define e descreve os principais conceitos relacionados a este trabalho, como: Web Semântica; Ontologias e suas respectivas formas de modelagem e implementação; e trabalhos relacionados que utilizaram abordagens diferentes para solucionar problemas semelhantes ao deste trabalho.

2.1 WEB SEMÂNTICA

A Web Semântica, conforme a organização *World Wide Web Consortium* (W3C) (2013), está relacionada a duas coisas: formatos comuns de integração e combinação de dados oriundos de fontes distintas e, também, uma linguagem capaz de definir como os dados se relacionam com os objetos do mundo real. Assim, permite que os usuários vejam todas as informações relacionadas a um contexto, independentemente da distinção entre as fontes de dados.

Além da definição oficial da W3C, alguns autores complementam o conceito de Web Semântica. Ding et al. (2004) a definem como um meio que oferece uma solução promissora para a publicação de informações e serviços na web de maneira aprimorada, pois faz uso de descrições em um formato no qual as máquinas são capazes de processar e entender as informações mais facilmente. Essa solução auxilia os agentes web na realização de uma série de tarefas para seus usuários, como a descoberta e integração de informações e a negociação e composição de serviços.

Os mesmos autores também definem a Web Semântica como um *framework* que permite que dados e conhecimento sejam publicados, compartilhados e reutilizados na web desvinculados de aplicações, empresas ou da comunidade.

Decker et al. (2000) descrevem a Web Semântica como a web da próxima geração, que objetiva permitir o desenvolvimento de serviços inteligentes, como intermediadores de informações, agentes de busca e filtros de informações, todos capazes de oferecer funcionalidade e interoperabilidade melhores que os serviços autônomos atuais. Porém, este ambiente só pode ser criado após o estabelecimento de certos níveis de interoperabilidade entre os dados. O autor afirma que somente a padronização dos documentos com relação a sua sintaxe não é o suficiente, ela também precisa ser aplicada aos conteúdos semânticos dos documentos.

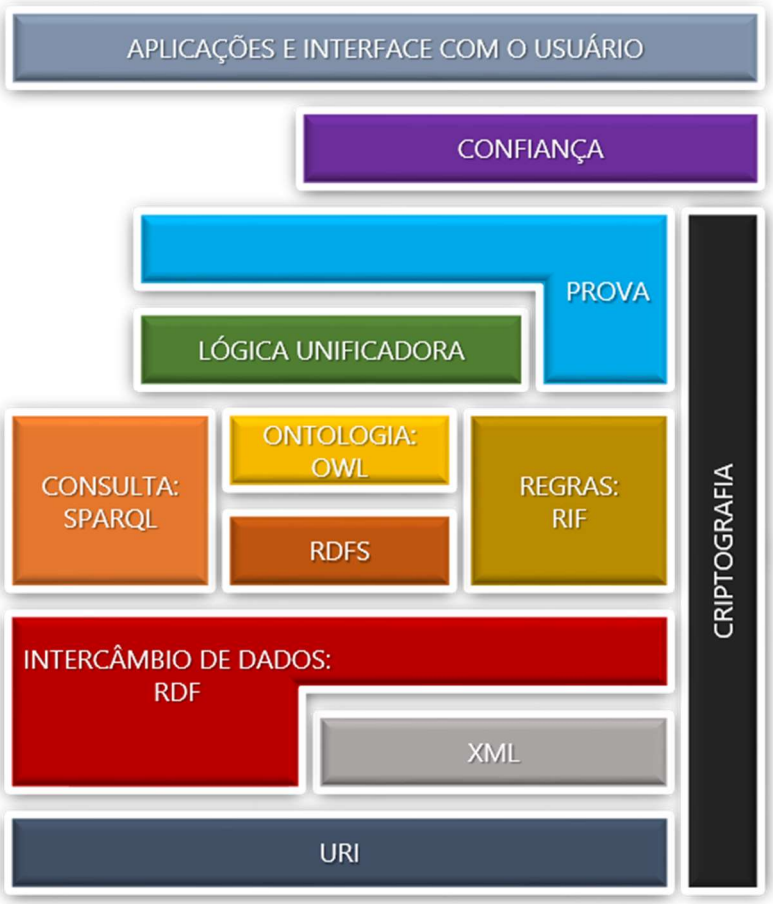
Ao mencionar a palavra “semântica”, Allemang & Hendler (2011) a definem como “a representação da relação entre um termo em uma sentença e a entidade a qual o termo se refere”, também enfatizando que este é o conceito que motiva o desenvolvimento da Web Semântica. Porém, ao contrário de focar no estudo de símbolos ou signos e suas relações com o que eles representam (campo de estudo da Semiótica), a Web Semântica trabalha com a modelagem dessas relações, ou seja, se símbolos se referem a coisas, o objetivo é construir modelos a partir destes símbolos que sejam capazes de auxiliar as pessoas a capturar, entender e comunicar o que há de conhecimento acerca dessas relações.

Ainda de acordo com os autores, toda unidade de informação que possua um valor semântico (ex.: Michio Kaku, Harvard, os elefantes da África ou a Teoria das Cordas) são denominadas Recursos. O grande diferencial da Web Semântica é a capacidade de conectar estes recursos, ao invés de conectar documentos diferentes sobre estes recursos. Na web atual pode haver um documento sobre Michio Kaku que apresente um *link* para um documento sobre a Teoria das Cordas, mas não há a noção de uma entidade que represente cada um desses recursos, os usuários estão restritos às informações contidas nos documentos.

“Na web tradicional as relações funcionam por meio de *hyperlinks* entre páginas, e na Web Semântica, entre informações” (CAMPBELL e MCNEIL, 2010, pág. 2). Na web tradicional, os *hyperlinks* têm o objetivo de conectar a página atual com uma página alvo, e essa relação não é nomeada, ou seja, o sentido ou significado dela deve ser inferido pelo leitor. Já na Web Semântica, as relações são estabelecidas entre recursos. Além disso, elas também são nomeadas, ou seja, as definições de cada relação são explícitas com o intuito de auxiliar no entendimento de cada uma delas.

Para que o funcionamento desse intercâmbio automático de dados possa ocorrer, e assim tornar possível a existência da Web Semântica, foi definida uma estrutura que engloba uma série de padrões, processos, tarefas e tecnologias que devem trabalhar juntas para que este resultado possa ser alcançado. A Figura 1 faz uma representação dessa estrutura em forma de camadas.

Figura 1 - Estrutura da Web Semântica em camadas



Fonte: (CAMPBELL e MCNEIL, 2010, adaptado)

De acordo com Ding et al (2005), as duas camadas mais inferiores da estrutura apresentada na Figura 1 (URI e XML) representam o alicerce da Web Semântica, sendo que a *eXtensible Markup Language* (XML) é utilizada para a sintaxe dos documentos e os *Uniform Resource Identifiers* (URI) são utilizados para nomear e identificar os recursos apresentados, ou seja, as próprias informações. As camadas intermediárias (*Resource Description Framework* - RDF, SPARQL, *Resource Description Framework Schema* - RDFS, *Web Ontology Language* - OWL e *Rule Interchange Format* - RIF) são responsáveis pela representação dos conceitos, propriedades e indivíduos relacionados ao contexto abordado. As camadas superiores da estrutura (Lógica Unificadora, Prova, Confiança e Aplicação), ainda em processo de desenvolvimento e consolidação, têm o objetivo de estender as características semânticas das informações no intuito de apresentar inferências, regras, provas e demonstrar a confiabilidade dos resultados. Além disso, existe a camada de Criptografia, que permeia toda a estrutura, cujo objetivo é proporcionar segurança aos dados apresentados.

As seções a seguir contêm explanações acerca de algumas das camadas dessa estrutura, que serão utilizadas para o desenvolvimento deste trabalho.

2.2 ONTOLOGIA

"Ontologia consiste em uma especificação explícita de uma conceitualização" (GRUBER, 1995, pág. 1). Posteriormente em seu trabalho, o autor define conceitualização como uma visão abstrata e simplificada do domínio que alguém deseja representar por algum motivo. Além disso, afirma que toda base de conhecimento, sistema ou agente baseado em conhecimento estão comprometidos com alguma forma de conceitualização, seja ela implícita ou explícita. Isso implica dizer que ontologia é uma forma de representação de um domínio específico por meio da descrição das características que o compõem.

Com base no trabalho de Gruber, Borst (1996) fez uma adaptação do conceito de ontologia para uma especificação formal de uma conceitualização compartilhada. Dessa forma, adicionando os conceitos de formalidade e compartilhamento. A ideia de formalidade indica que mesmo que ontologias mudem de acordo com o domínio no qual estão inseridas, elas devem seguir um padrão, ou seja, um conjunto de regras para a sua construção.

A ideia de compartilhamento dentro do conceito de ontologia faz referência a dois pontos. Primeiro, que tanto a descrição das características do domínio de uma ontologia quanto a forma de descrevê-las não são centralizadas, ou seja, podem ser definidas de maneira conjunta. O outro ponto é que ontologias não são desenvolvidas vinculadas a uma forma de apresentação, ou seja, a partir de sua definição, elas podem ser exploradas e apresentadas, ou compartilhadas, de diversas maneiras, dependendo da aplicação.

Os trabalhos de Alalwan, Zedan e Siewe (2009) e Telnarova (2010) corroboram com Borst ao enfatizar, respectivamente, as ideias de formalidade e compartilhamento. "Além da formalidade, ontologias também podem apresentar um caráter hierárquico, de modo a trabalhar com conceitos que englobam outros conceitos" (ALALWAN, ZEDAN e SIEWE, 2009, pág. 22). Enquanto isso, Telnarova (2010) afirma que ontologias funcionam como uma base de conhecimento, que podem ser reutilizadas para diversas finalidades.

Além dos pontos apresentados, Yahia, Mokhtar e Ahmed (2012) mencionam outro aspecto importante relacionado à ideia de formalidade presente no conceito de ontologia. Os autores afirmam que o caráter formal implica dizer que uma ontologia deve ser escrita de maneira que máquinas possam entendê-la. Dessa forma, o computador pode ser utilizado para auxiliar na análise do valor semântico dos dados, realizando inferências e recomendações

dentro do contexto da ontologia. Logo, é possível concluir que ontologias são representações de domínios reais por meio de padrões com o intuito de permitir que máquinas sejam capazes de entender e processar as informações desses domínios.

"Ontologia consiste em uma descrição formal de conceitos em um domínio de discurso (classes), propriedades de cada conceito que descrevem suas características ou atributos (*slots*), e restrições relacionadas a estes *slots* (facetas)" (NOY e MCGUINNESS, 2001, pág. 3). Uma ontologia, juntamente com uma série de instâncias de classes, dá origem a uma base de conhecimento.

Ainda de acordo com as autoras, as classes são o foco da maior parte das ontologias, por exemplo, uma classe `Funcionário` representa todos os funcionários dentro de um contexto. Cada funcionário é identificado como uma instância dessa classe. As classes podem ter subclasses, que trabalham com conceitos mais específicos que sua respectiva superclasse. Por exemplo, a classe `Professor` pode ser considerada uma subclasse de `Funcionário` dentro do contexto deste trabalho. Os *slots*, são propriedades das classes que são comuns a todas as suas instâncias. Ao dizer que o nome de um funcionário é `Matheus Rodrigues Leal` e sua idade é `21 anos`, duas propriedades de uma instância da classe `Funcionário` estão sendo apresentadas. Com relação às propriedades, as subclasses herdam as existentes em sua superclasse e adicionam suas especificidades, logo, uma instância da classe `Professor` também teria as propriedades `Nome` e `Idade`, além de algumas próprias, como `Titulação` ou `Área de atuação`.

Nas próximas seções, são apresentadas algumas das linguagens que compõem os padrões utilizados para a criação e manuseio de ontologias.

2.2.1 Tecnologias para implementação e manipulação de ontologias

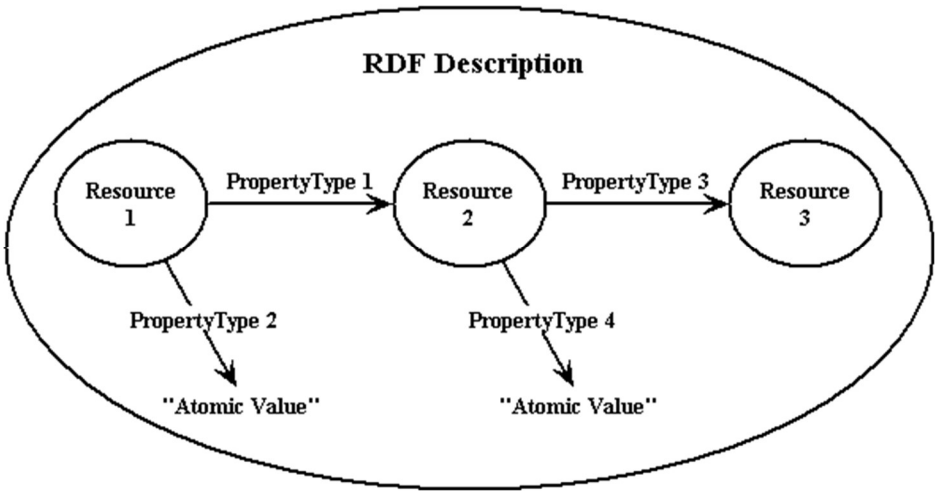
Assim como Alalwan, Zedan e Siewe (2009) e Decker et al. (2000) afirmam a importância das ontologias para o contexto da Web Semântica, eles também concordam que para que um ambiente como esse seja construído da forma como é concebido, uma série de padrões devem ser definidos e seguidos, tanto para a estrutura sintática da implementação, quanto para a semântica das informações que serão apresentadas. Dessa forma, existem uma série de tecnologias associadas ao conceito de ontologia que objetivam permitir que as informações sejam associadas a um contexto específico, e estruturá-las de uma maneira que seja mais compreensível para os motores de busca. As seções a seguir apresentam algumas delas.

2.2.1.1 Resource Description Framework – RDF

"*Resource Description Framework* (RDF) é uma tecnologia desenvolvida para padronizar a definição e o uso de metadados, ou seja, conjunto de dados que objetivam descrever outros dados e relações entre eles" (DECKER et al., 2000, pág. 5). A unidade básica do RDF é denominada Tripla Objeto-Atributo-Valor ou, fazendo menção à estrutura formal da gramática de alguns idiomas como o português e o inglês, Tripla Sujeito-Predicado-Objeto.

Ao formalizar a definição das triplas do RDF utilizando a denominação Objeto-Atributo-Valor, o resultado é a relação $A(O,V)$. Essa relação expressa que um objeto O possui um atributo A com valor V. Outra maneira de representar essa relação é através de uma ligação identificada entre dois nós: $[O] - A \rightarrow [V]$. A Figura 2 demonstra o resultado visual de uma descrição RDF em forma de grafo.

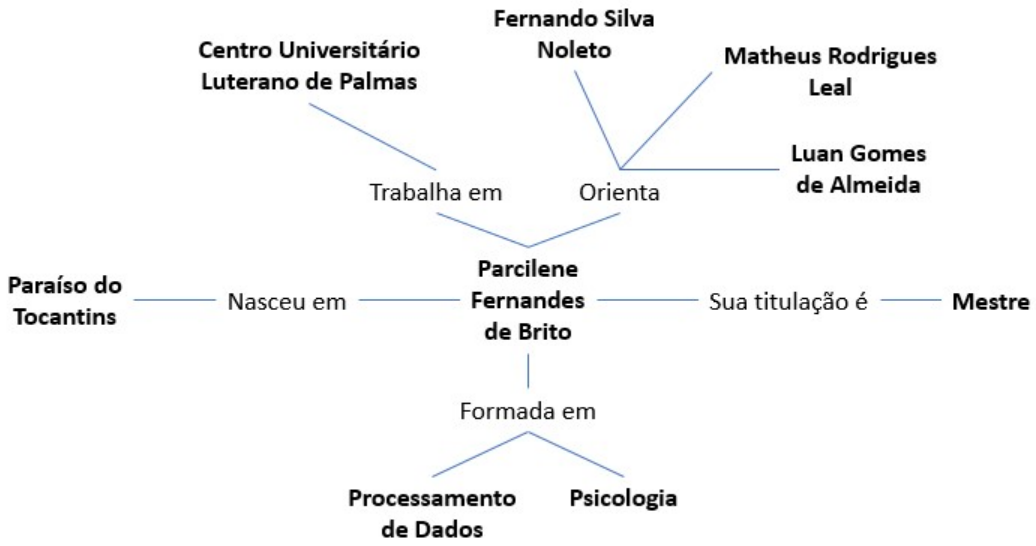
Figura 2 - Representação visual da estrutura criada pelo RDF



Fonte: (MILLER, E. 1998)

Na imagem, é possível visualizar a afirmação apresentada na seção anterior, que a Web Semântica trabalha com relações nomeadas entre recursos, as *PropertyTypes*, com o intuito de deixar explícitas as informações conhecidas acerca das relações. Além disso, essa representação também demonstra a flexibilidade dessa estrutura, pois uma vez que o objeto O, os *Resources*, e o valor V, *Resources* ou *Atomic Values*, são representados igualmente como nós, isso significa que um mesmo dado pode ser um objeto em uma relação e um valor em outra. A seguir, na Figura 3, há um exemplo de grafo RDF dentro de um contexto específico, relacionado à arte.

Figura 3 - Exemplo de grafo RDF com os perfis de professor e aluno

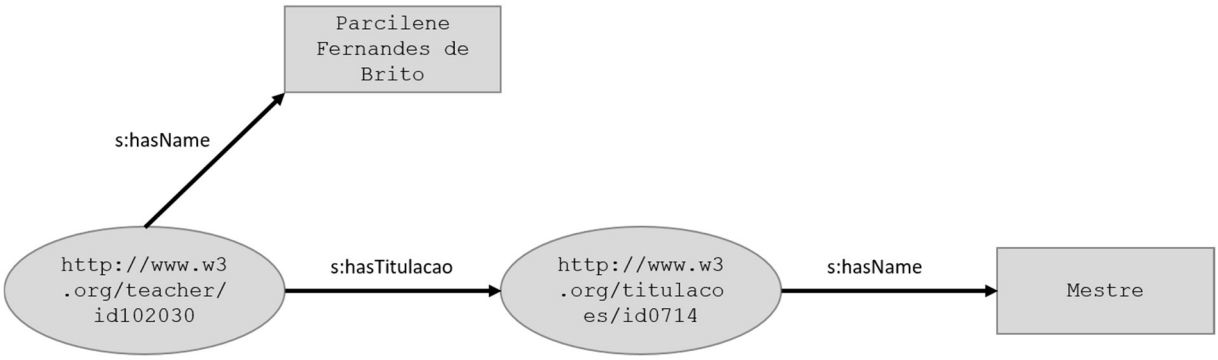


A Figura 3 facilita o entendimento dos elementos do RDF, principalmente, dos atributos, que representam as relações entre os recursos. É possível perceber que os atributos podem representar literalmente qualquer característica dos recursos do grafo. Dessa forma, é possível estender o formato de representação sempre que necessário, ao adicionar mais informações.

Porém, uma questão que vem à tona ao visualizar uma representação como a da Figura 3, é a possibilidade de ambiguidade. Como certificar que o recurso `Mestre` que está sendo utilizado realmente se refere a uma titulação acadêmica? Ou como evitar que informações relacionadas a outro `Matheus Rodrigues Leal` não sejam incluídas a este mesmo grafo? Como estes recursos existem em contextos diferentes, a identificação do contexto específico realmente é um problema a ser considerado.

A solução para este problema foi a mesma utilizada como solução para uma questão semelhante na Web tradicional, a utilização de um padrão de identificação. Na Web tradicional, são utilizadas as *Uniform Resource Locators* (URL), enquanto para a Web Semântica foi definido o termo *Uniform Resource Identifier* (URI), que tem o objetivo de apontar para um recurso, e não para uma página web específica. Ao apontar para um recurso, são exibidas todas as informações conhecidas que já foram adicionadas a ele, ou seja, todas as suas relações com outros recursos. Dessa forma, a Figura 4 apresenta o resultado de um grafo RDF utilizando as URIs.

Figura 4 - Exemplo de grafo RDF utilizando URIs



Utilizando a representação dos recursos por meio de URIs (Figura 4), os únicos valores explícitos que permanecem no grafo são valores literais. Além disso, antes dos nomes dos atributos, é adicionado um termo seguido de dois pontos que é utilizado para representar o contexto no qual ele está inserido (`s:hasName` e `s:hasTitulacao`), pois a ambiguidade também pode afetar o significado dos nomes utilizados pelos atributos.

2.2.1.2 Web Ontology Language – OWL

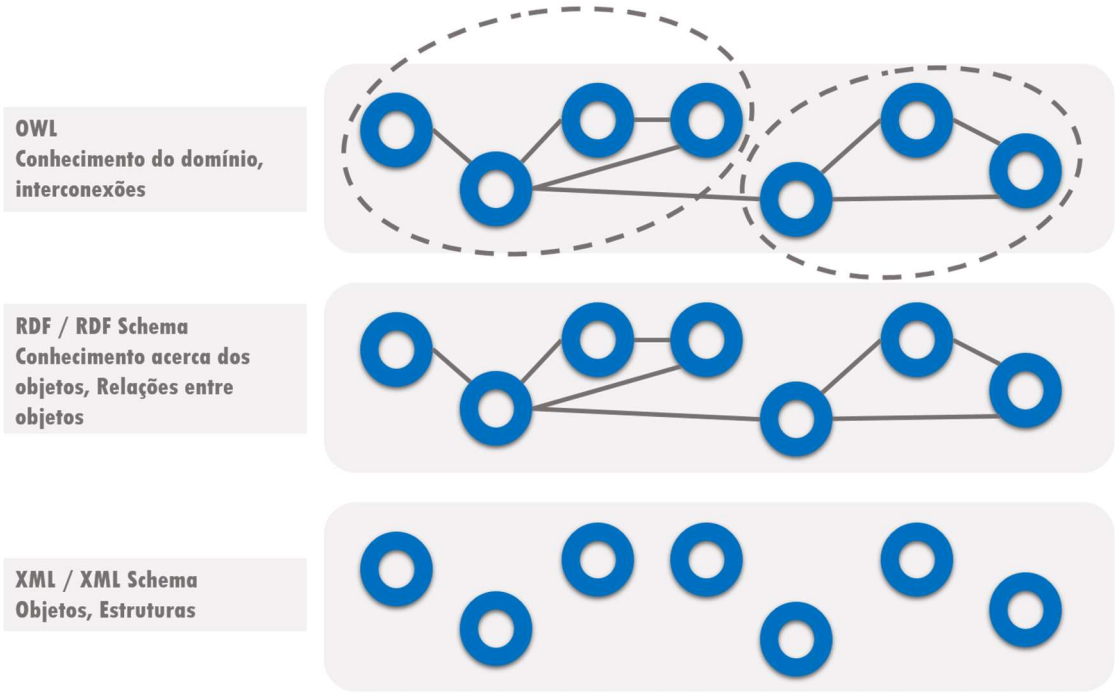
Ontologias, como explicado anteriormente, são estruturas que objetivam representar o conhecimento referente a um domínio específico. Porém, apesar de suas particularidades no que tange à semântica, as ontologias compartilham padrões sintáticos de representação, ou seja, linguagens específicas, assim como linguagens de programação. As linguagens de representação de ontologias mais conhecidas são a *Resource Description Framework Schema* (RDFS), *Ontology Inference Layer* (OIL), *DARPA Agent Markup Language* (DAML+OIL) e a *Web Ontology Language* (OWL) (W3C, 2004). Para este trabalho será utilizada a linguagem OWL.

"OWL é uma linguagem computacional baseada em lógica para a representação de conhecimento, que permite que tal conhecimento seja explorado por meio de programas de computador" (W3C, 2004). Exemplos de tarefas que podem ser realizadas em documentos OWL são: verificação da consistência do conhecimento representado e a apresentação de conhecimentos implícitos de forma explícita. A verificação da consistência significa responder à seguinte pergunta: dada uma informação em uma ontologia, alguma outra informação existente apresenta algum tipo de contradição a ela? A apresentação de conhecimentos implícitos consiste na possibilidade de descoberta de novas informações a partir da organização das informações existentes e suas respectivas relações. Essas e outras

tarefas são desempenhadas por meio da realização de consultas utilizando a linguagem específica para consultas em ontologias, SPARQL, que será apresentada na seção 2.2.1.3.

No contexto de representação do conhecimento voltado para a Web Semântica, a OWL é a linguagem utilizada para representar as informações de mais alto nível acerca de um domínio, assim como apresentado na Figura 5.

Figura 5 - Níveis de representação do conhecimento



Fonte: (LINCKELS, S. 2014, adaptada)

De acordo com a Figura 5, existem três níveis complementares de representação do conhecimento dentro do contexto de Web Semântica. A linguagem XML e sua extensão XML *Schema* é utilizada para representar o conhecimento em seu nível mais baixo, de forma que o objetivo dessa representação é somente estrutural, ou seja, as estruturas isoladas dos objetos e entidades que fazem parte do domínio no qual a ontologia está inserida. A linguagem RDF e sua extensão RDF *Schema* é utilizada para representar o conhecimento no nível intermediário. Por meio delas são inseridas as informações acerca das relações conhecidas entre os objetos e entidades representadas. A linguagem OWL é utilizada para representar o conhecimento no nível mais elevado, ou seja, as interconexões existentes entre os objetos e entidades e as características específica do domínio, como restrições, propriedades, atributos e as instâncias dos objetos e entidades.

Por exemplo, dentro do contexto deste trabalho, no nível mais baixo de representação do conhecimento, seriam definidas as classes e suas respectivas propriedades (`Aluno`,

`Professor`, `Turma` etc). No nível intermediário de representação do conhecimento, seriam definidas as relações entre as classes do nível anterior, com o intuito de deixar explícito o conhecimento acerca das entidades existentes (`Professores` são vinculados a `Turmas`, `Turma` tem `Alunos`, `Alunos` participam de `Projetos` etc). No nível mais alto de representação do conhecimento, são definidas as restrições relacionadas ao domínio, ou seja, as características específicas das classes, propriedades e relações existentes (Um `Professor` pode ter várias `Turmas`, Um `Projeto` pode estar relacionado a vários `Departamentos`, Um `Aluno` só deve estar vinculado a um `Curso` por vez etc).

Segundo as descrições da W3C (2004), um documento OWL é composto por *namespaces*, declarações de classes, propriedades de classes e indivíduos ou instâncias de classes.

Namespaces são componentes ontológicos cujo objetivo é fornecer uma indicação precisa do vocabulário utilizado no documento, para evitar ambiguidades. Por padrão, a declaração de *namespaces* ocorre no início do documento, semelhante a importações de pacotes e bibliotecas em documentos de linguagens de programação tradicionais. Além disso, são representadas dentro da *tag* `rdf:RDF`. A Figura 6 apresenta um exemplo de declaração de *namespaces* em um documento OWL.

Figura 6 - Declaração de *Namespaces*

```
1  <rdf:RDF
2      xmlns="http://www.w3.org/TR/2004/REC-owl-guide-20040210/portal#"
3      xmlns:portal="http://www.w3.org/TR/2004/REC-owl-guide-20040210/portal#"
4      xml:base="http://www.w3.org/TR/2004/REC-owl-guide-20040210/portal#"
5      xmlns:professores="http://www.w3.org/TR/2004/REC-owl-guide-20040210/professores#"
6      xmlns:alunos="http://www.w3.org/TR/2004/REC-owl-guide-20040210/alunos#"
7      xmlns:owl="http://www.w3.org/2002/07/owl#"
8      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
9      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
10     xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
```

O trecho de código da Figura 6 se refere às definições do vocabulário que será trabalhado no documento. A linha 2 representa a declaração do *namespace* padrão do documento, ou seja, todos os termos apresentados posteriormente no documento que não vierem acompanhados por um *namespace*, serão automaticamente relacionados a este. Na linha 3 é definido o nome para esse *namespace* padrão, nesse caso é o nome `portal`. Na linha 4 é realizada a ligação entre o *namespace* padrão e sua respectiva URI.

As linhas 5 e 6 funcionam como importações de ontologias previamente existentes que serão utilizadas no documento. No exemplo apresentado, são importadas duas ontologias que são referenciadas por meio dos *namespaces* `professores` e `alunos`.

A partir da linha 7 as declarações se referem à importação dos recursos padrões da Web Semântica, são eles: OWL, RDF, RDF *Schema* e XML *Schema*. Dessa forma, as propriedades e estruturas definidas por esses padrões podem ser utilizados dentro do documento.

Outro componente natural de documentos OWL corresponde à declaração de classes. Na linguagem OWL, a declaração de classes é realizada de acordo com a Figura 7.

Figura 7 - Declaração de classes

```
1   <owl:Class rdf:ID="Professor"/>
2   <owl:Class rdf:ID="Aluno"/>
3   <owl:Class rdf:ID="Projeto"/>
```

Na Figura 7 há a declaração de três classes: `Professor, Aluno e Projeto`. A partir dessa declaração, essas classes podem ser referenciadas no documento por meio do comando `rdf:resource`. Além disso, elas podem ser referenciadas por outra ontologia que fizer a importação do *namespace* `portal`, utilizado para nomear este exemplo. Outra forma de referenciar uma classe é por meio do comando `rdf:about`. Este comando permite que novas informações sejam inseridas à descrição da classe sem que a descrição existente seja afetada. Além disso, esse comando também pode referenciar classes de outras ontologias, o que permite a ideia de colaboração existente nos princípios da Web Semântica.

Na linguagem OWL, por padrão, todas as classes existentes são subclasses da classe `owl:Thing`. Essa relação é utilizada para que seja possível representar relacionamentos que não especificam uma única classe, ou seja, podem ser realizados entre qualquer tipo de classe. No entanto, não é necessário deixar explícito no código a relação entre as classes criadas de acordo com o contexto e a classe `Thing`, pois isso já é definido automaticamente pela linguagem. Porém, para estabelecer a relação de subclasse entre as demais classes, existe o comando `rdf:subClassOf`, que é definido como uma propriedade. O objetivo dessa propriedade é a criação e o relacionamento de novas classes mais específicas com classes mais genéricas. Por exemplo, a classe `Projeto`, definida anteriormente, pode estar relacionada a uma classe mais específica denominada `ProjetoPesquisa`. Nesse caso, a nova classe seria uma subclasse de `Projeto`. A Figura 8 apresenta um exemplo de criação de uma subclasse na linguagem OWL.

Figura 8 - Criação de subclasse

```
1   <owl:Class rdf:ID="ProjetoPesquisa">
2   |  <rdfs:subClassOf rdf:resource="#Projeto" />
3   |  ...
4   </owl:Class>
```

Da mesma forma que é possível relacionar uma classe mais específica a uma mais genérica, também é possível fazer o contrário. Para isso, é utilizado o comando `rdf:type`, que tem como objetivo dizer que uma casse faz parte de outra maior. A Figura 9 apresenta um exemplo do uso desse comando.

Figura 9 - Relacionamento de subclasse com superclasse

```
1   <owl:Class rdf:ID="ProjetoExtensao" />
2
3   <owl:Class rdf:about="#ProjetoExtensao">
4      <rdf:type rdf:resource="#Projeto"/>
5   </owl:Class>
```

De acordo com a Figura 9, é possível notar que a estrutura de utilização dessa propriedade é semelhante à utilização da propriedade `subClassOf`. Isso ocorre pelo fato de ambos estarem relacionados com o mesmo tipo de elementos, as classes.

Para descrever as características das classes, existem as propriedades. Existem dois tipos de propriedades utilizadas na linguagem OWL, são elas: *Object Properties* (Propriedades de Objeto) e *Datatype Properties* (Propriedades de Tipos de Dados). Elas se diferenciam pelo tipo de elementos que cada uma é capaz de relacionar. As Propriedades de Objeto relacionam instâncias de classes a instâncias de outras classes, enquanto as Propriedades de Tipos de Dados relacionam instâncias de classes a valores literais, como valores numéricos ou textuais.

Ao definir uma propriedade de objeto, os campos que identificam a instância de classe origem e a instância de classe destino do relacionamento são denominados domínio (*domain*) e alcance (*range*), respectivamente. A Figura 10 apresenta um exemplo de definição de uma propriedade de objeto.

Figura 10 - Definição de propriedade de objeto

```
1   <owl:ObjectProperty rdf:ID="responsavelPor">
2     <rdfs:domain rdf:resource="#Professor"/>
3     <rdfs:range rdf:resource="#Projeto"/>
4   </owl:ObjectProperty>
```

Na Figura 10 é definida a propriedade de objeto `responsavelPor`, que tem como domínio a classe `Professor` e como alcance a classe `Projeto`. Isso significa que todas as instâncias da classe `Professor` terão a propriedade `responsavelPor`, que indica por qual projeto ele é responsável.

Ao definir uma propriedade de tipos de dados, os campos domínio e alcance também precisam ser especificados. No entanto, o campo alcance passa a indicar um tipo de dado

literal que o valor da propriedade terá, como: *string*, *float*, inteiros, tempo, data, booleano etc. A Figura 11 apresenta um exemplo de definição de uma propriedade de tipos de dados.

Figura 11 - Definição de propriedade de tipos de dados

```
<owl:DatatypeProperty rdf:ID="cargaHoraria">
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range  rdf:resource="&xsd;positiveInteger"/>
</owl:DatatypeProperty>
```

Na Figura 11, é definida a propriedade de tipos de dados `cargaHoraria`, que tem como domínio a classe `Professor` e como alcance um identificador que se refere aos números inteiros positivos. Isso significa que o valor atribuído a essa propriedade para todas as instâncias da classe `Professor` deve ser um número maior que zero e não decimal.

Além disso, as propriedades também podem ter algumas características específicas, que fornecem algumas informações sobre os recursos que são relacionados por meio delas. De acordo com a WC3 (2004), as características existentes para as propriedades são:

- *Transitive Property*: "ao assumir $x, y\ e\ z$ como recursos, uma propriedade é classificada como *Transitive Property* se $P(x,y) \wedge P(y,z) \rightarrow P(x,z)$". Por exemplo, se um professor faz parte de um projeto de extensão e este projeto faz parte dos projetos de um determinado departamento da instituição, logo o professor faz parte dos projetos de extensão deste departamento. Um exemplo de definição de *Transitive Property* em linguagem OWL é apresentada na Figura 12.

Figura 12 - Exemplo de *Transitive Property*

```
<owl:ObjectProperty rdf:ID="fazParte">
  <rdf:type rdf:resource="&owl;TransitiveProperty" />
  <rdfs:domain rdf:resource="&owl;Thing" />
  <rdfs:range rdf:resource="&owl;Thing" />
</owl:ObjectProperty>

<Professor rdf:ID="Parcilene">
  <fazParte rdf:resource="#Projeto_SentimentALL" />
</Professor>

<Projeto rdf:ID="Projeto_SentimentALL">
  <fazParte rdf:resource="#Projetos_Computação" />
</Projeto>
```

- *Symmetric Property*: "ao assumir $x\ e\ y$ como recursos, uma propriedade é classificada como *Symmetric Property* se $P(x,y) = P(y,x)$". Por exemplo, ao atribuir uma relação `colegaDeTurma` entre duas instâncias da classe `Aluno`, ela permanece com

o mesmo significado independente de qual seja o domínio e qual seja o alcance da propriedade. Um exemplo de *Symmetric Property* é apresentado na Figura 13.

Figura 13 - Exemplo de *Symmetric Property*

```
<owl:ObjectProperty rdf:ID="colegaDeTurma">
  <rdf:type rdf:resource="&owl;SymmetricProperty" />
  <rdfs:domain rdf:resource="#Aluno" />
  <rdfs:range rdf:resource="#Aluno" />
</owl:ObjectProperty>

<Aluno rdf:ID="Matheus">
  <colegaDeTurma rdf:resource="#Fernando" />
</Aluno>

<Aluno rdf:ID="Fernando">
  <colegaDeTurma rdf:resource="#Matheus" />
</Aluno>
```

- *Functional Property*: "ao assumir $x, y \, e \, z$ como recursos, uma propriedade é classificada como *Functional Property* se $P(x, y) \wedge P(x, z) \rightarrow y = z$". Essa característica é atribuída a propriedades de tipos de dados. Por exemplo, ao definir a propriedade `temMaeBiologica`, sabe-se que todas as instâncias que possuírem esta propriedade, só poderão ter um valor relacionado a ela, pois as pessoas só possuem uma mãe biológica. Dessa forma, se uma mesma instância for representada com valores distintos para esta propriedade, assume-se que estes valores se referem ao mesmo objeto. Por exemplo, se a propriedade `temMaeBiologica` do aluno `Matheus` é preenchida com o valor `Solange` em um determinado local, e com o valor `Solange Rodrigues` em outro, pode-se afirmar que `Solange` e `Solange Rodrigues` se referem ao mesmo indivíduo. O código OWL que representa este exemplo é apresentado na Figura 14.

Figura 14 - Exemplo de *Functional Property*

```
<owl:ObjectProperty rdf:ID="temMaeBiologica">
  <rdf:type    rdf:resource="&owl;FunctionalProperty" />
  <rdfs:domain rdf:resource="#Pessoa" />
  <rdfs:range  rdf:resource="#Pessoa" />
</owl:ObjectProperty>

<Pessoa rdf:ID="Matheus">
  <temMaeBiologica rdf:resource="Solange">
</Pessoa>

<Pessoa rdf:ID="Matheus">
  <temMaeBiologica rdf:resource="Solange Rodrigues">
</Pessoa>
```

- *InverseOf*: "ao assumir $P1\ e\ P2$ como propriedades, elas são classificadas como *InverseOf* se $P1(x, y) = P2(y, x)$". Por exemplo, ao assumir uma propriedade `responsavelPor`, entre `Professor` e `Projeto`, uma propriedade *InverseOf* a ela seria `temResponsavel`, que indica qual professor é responsável por um determinado projeto. O código OWL que representa este exemplo é apresentado na Figura 15.

Figura 15 - Exemplo de *InverseOf*

```
<owl:ObjectProperty rdf:ID="responsavelPor">
  <rdfs:domain rdf:resource="#Professor" />
  <rdfs:range  rdf:resource="#Projeto" />
</owl:ObjectProperty>

<owl:ObjectProperty rdf:ID="temResponsavel">
  <owl:inverseOf rdf:resource="#responsavelPor" />
</owl:ObjectProperty>
```

- *Inverse Functional Property* inversa: "ao assumir $x, y\ e\ z$ como recursos, uma propriedade é classificada como *Inverse Functional Property* se $P(y, x) \wedge P(z, x) \rightarrow y = z$". Essa característica é atribuída a propriedades de objeto e representa uma propriedade com característica *InverseOf* de uma *Functional Property*. Utilizando o exemplo da *Functional Property* como base, a *Inverse Functional Property* equivalente seria a propriedade `temFilhoBiologico`. Nessa situação, se o indivíduo `Solange` tem o valor `Matheus` para essa propriedade em um determinado local, e o valor `Matheus Rodrigues` em outro, assume-se que `Matheus` e `Matheus Rodrigues` se referem ao mesmo indivíduo. O código OWL que representa este exemplo é apresentado na Figura 16.

Figura 16 - Exemplo de *Inverse Functional Property*

```
<owl:ObjectProperty rdf:ID="temFilhoBiologico">
  <rdf:type    rdf:resource="&owl;InverseFunctionalProperty" />
  <rdfs:domain rdf:resource="#Pessoa" />
  <rdfs:range  rdf:resource="#Pessoa" />
</owl:ObjectProperty>

<Pessoa rdf:ID="Solange">
  <temFilhoBiologico rdf:resource="Matheus">
</Pessoa>

<Pessoa rdf:ID="Solange Rodrigues">
  <temFilhoBiologico rdf:resource="Matheus">
</Pessoa>
```

Além dessas cinco características, a ferramenta Protégé, utilizada nesse trabalho, disponibiliza mais três: *Asymmetric, Reflexive* e *Irreflexive*. A seguir são apresentados os conceitos relacionados a estas características de acordo com (HORRIDGE et. al., 2011, pág. 32-33).

- *Asymmetric Property*: "se P é uma *Asymmetric Property* e relaciona o indivíduo a ao indivíduo b, então b não pode ser relacionado ao indivíduo a por meio da propriedade P". Por exemplo, se o indivíduo `Matheus` está relacionado ao indivíduo `Solange` por meio da *Asymmetric Property* `EhFilhoDe`, então o indivíduo `Solange` não pode estar relacionado ao indivíduo `Matheus` por meio da mesma propriedade. Isso indica que `Matheus` é filho de `Solange` e `Solange` não é filho de `Matheus`.
- *Reflexive Property*: "uma propriedade P é classificada como *Reflexive Property* se um indivíduo a pode se relacionar com ele mesmo por meio de P". Por exemplo, ao definir `Conhece` como uma *Reflexive Property*, é possível estabelecer a relação `Matheus Conhece Matheus`, que indica que o indivíduo `Matheus` tem conhecimento de si mesmo.
- *Irreflexive Property*: "se P é uma *Irreflexive Property*, isso significa que ao relacionar o indivíduo a ao indivíduo b por meio de P, estes indivíduos são obrigatoriamente diferentes". Por exemplo, ao definir `EhMaeDe` como uma *Irreflexive Property*, o indivíduo `Solange` pode ser relacionado ao indivíduo `Matheus`, porém não pode ser relacionado com si mesmo, o que indica que `Solange` não pode ser mãe de si mesma.

Por fim, o último componente que compõe um documento OWL de acordo com a W3C (2004) são as instâncias de classes. Instâncias representam membros de classes, ou seja, um

exemplo de elemento que pode ser criado a partir do preenchimento das propriedades de uma classe. A Figura 17 apresenta como uma instância de classe é criada na linguagem OWL.

Figura 17 - Criação de instâncias de classes

```
<Professor rdf:ID="Parcilene">
  <responsavelPor rdf:resource="#analiseSentimento" />
</Professor>
```

A Figura 17 representa a criação da instância `Parcilene`, indicada como membro da classe `Professor`. Como só havia uma propriedade dessa classe definida anteriormente no exemplo apresentado, essa instância tem informações somente dessa propriedade.

2.2.1.3 SPARQL

Da mesma forma que existem tecnologias e padrões de representação e especificação de informações relacionados ao contexto de Web Semântica, também há uma tecnologia direcionada para a consulta de dados em documentos que seguem esse formato. “SPARQL é uma linguagem que permite a construção de consultas para a aplicação em diversas fontes de dados, uma vez que estes dados estejam nativamente armazenados em formato RDF ou que possam ser convertidos por meio de um software intermediário” (W3C, 2001).

O modo de funcionamento da linguagem é por meio da busca em grafos, sendo que os resultados podem ser apresentados em forma de conjuntos de dados, subgrafos do domínio original ou, por padrão, em tabela de dados. Além disso, a linguagem suporta a realização de testes e revisões em estruturas extensíveis, ou seja, não há a necessidade de reformular a consulta ou estagnar o desenvolvimento de uma base dados.

De acordo com os exemplos apresentados na versão mais recente da documentação oficial da linguagem, produzida pela W3C (2013), uma consulta utilizando a linguagem SPARQL pode ser realizada por meio do código apresentado na Figura 18.

Figura 18 - Exemplo de consulta SPARQL

```
SELECT * WHERE
{
  <http://exemplo.org/professores>
  <http://exemplo.org/relacoes>
  ?name
}
```

Na primeira linha da consulta são especificados os termos das triplas RDF que devem aparecer no resultado, ou seja, as colunas da tabela. Ao utilizar o asterisco, todas as colunas

da tabela são apresentadas. A cláusula `WHERE` é onde são especificados os parâmetros da consulta. Caso a consulta deva considerar algum termo específico, ele deve ser representado por meio de sua URI, como é o caso da entidade `Professor` e da relação entre a entidade e o valor correspondente, apresentado no exemplo como `Relacoes`. É possível substituir um ou mais termos da tripla pela estrutura: ponto de interrogação sucedido do nome do termo (*namespace*). Dessa forma, serão apresentados todos os resultados que se encaixam no respectivo *namespace*.

A estrutura padrão de apresentação dos dados é em forma de tabela. A Tabela 1 representa um possível conjunto de resultados para a consulta utilizada como exemplo anteriormente.

Tabela 1 - Exemplo de Resultado SPARQL

X	Y	Z
“Parcilene”	<http://exemplo.org/professores>	<http://exemplo.org/relacoes
“Fabiano”	<http://exemplo.org/professores>	<http://exemplo.org/relacoes
“Jackson”	<http://exemplo.org/professores>	<http://exemplo.org/relacoes

A Tabela 1 apresenta um conjunto de variáveis ($x, y\ e\ z$) vinculadas a uma coluna da tabela. No exemplo apresentado, levando em consideração a estrutura sujeito-objeto-predicado para representar as triplas RDF, a variável x representa o objeto, a variável y representa o sujeito e a variável z representa o predicado. A quantidade de linhas na tabela corresponde à quantidade de instâncias na ontologia que correspondem aos critérios especificados na consulta SPARQL.

De acordo com a W3C (2008), a consulta utilizada como exemplo anteriormente é classificada como uma consulta *SELECT*. A linguagem SPARQL possui mais três tipos de consulta, são elas: consultas *ASK*, *CONSTRUCT* e *DESCRIBE*. As consultas *ASK* são utilizadas quando o objetivo é verificar se existem dados correspondentes a uma determinada consulta ou não, sem que esses dados sejam apresentados, ou seja, os resultados para essas consultas são valores booleanos *true* ou *false*. Um exemplo desse tipo de consulta é apresentado na Figura 19.

Figura 19 - Exemplo de consulta *ASK*

```
PREFIX portal: <http://w3.ulbra-to/portal#>
ASK
{
  <http://w3/ulbra-to/portal/professores/fabiano> prop:length ?turmas
  <http://w3/ulbra-to/portal/professores/jackson> prop:length ?turmas
  FILTER(portal:fabiano ?turmas > portal:jackson ?turmas)
}
```

Essa consulta busca a quantidade de turmas relacionadas ao Professor Fabiano e a quantidade de turmas relacionadas ao Professor Jackson. Após isso, procura saber se o Professor Fabiano possui mais turmas que o Professor Jackson.

As consultas *CONSTRUCT* são utilizadas quando o objetivo é obter grafos RDF como resultado, ao invés de uma tabela de dados como nas consultas *SELECT*. A Figura 20 apresenta um exemplo desse tipo de consultas.

Figura 20 - Exemplo de consulta *CONSTRUCT*

```
PREFIX portal: <http://w3.ulbra-to/portal/cursos#>
CONSTRUCT {
  ?X cursos:nome ?name
  ?X cursos:ch ?ch
  ?X cursos:curriculo ?curriculo
}
FROM <http://w3.ulbra-to/portal/professores#>
WHERE {
  OPTIONAL { ?X professores:name ?name . FILTER isLiteral(?name) }
  OPTIONAL { ?X professores:email ?email . FILTER isLiteral(?email) }
}
```

Essa consulta objetiva a construção de um grafo RDF que reúne os cursos e os professores da instituição e apresenta as relações entre eles.

As consultas *DESCRIBE* são utilizadas quando o objetivo é obter a descrição completa de um determinado recurso presente em uma tripla RDF, ou seja, a lista de todas as suas propriedades. Um exemplo desse tipo de consulta é apresentado na Figura 21.

Figura 21 - Exemplo de consulta *DESCRIBE*

```
PREFIX portal: <http://w3.ulbra-to/portal/cursos#>
DESCRIBE ?curso WHERE {
  ?curso cursos:nome "Sistemas de Informação"
}
```

Essa consulta solicita a descrição do recurso presente no *namespace* `cursos` cujo nome é `Sistemas de Informação`.

Além dos tipos de consulta, que influenciam no formato do resultado, a W3C (2008) define outros recursos da linguagem: os modificadores de solução, os filtros SPARQL e os padrões de grafos. Eles são utilizados com o intuito de auxiliar na obtenção de resultados específicos a partir das consultas. Os modificadores de solução e suas respectivas funcionalidades são:

- *Order Modifier*: permite a reorganização dos resultados em uma nova ordem, ascendente ou descendente, utilizando um termo do resultado como parâmetro;
- *Projection Modifier*: permite a seleção e apresentação de um conjunto específico de variáveis dentre as existentes no resultado;
- *Distinct Modifier*: permite a apresentação de somente ocorrências únicas existentes no conjunto de resultados;
- *Reduced Modifier*: permite a eliminação de algumas ocorrências do conjunto de resultados;
- *Offset Modifier*: permite a indicação do ponto inicial de onde a consulta deve começar a busca por resultados;
- *Limit Modifier*: permite o estabelecimento do número máximo de resultados que o conjunto pode ter.

A utilização desses modificadores consiste em acrescentá-los ao final das consultas, como é apresentado na Figura 22.

Figura 22 - Exemplo de utilização dos modificadores de solução

```
1  PREFIX portal: <http://w3.ulbra-to/portal/alunos#>
2  SELECT ?aluno WHERE {
3    ?aluno alunos:idade 17
4  }
5  LIMIT 100
```

Esta consulta busca pela lista de alunos da instituição com idade igual a 17 anos e restringe a quantidade de resultados para 100, utilizando o *Limit Modifier*.

Em conjunto com os modificadores de solução, existem os filtros SPARQL, “que auxiliam na eliminação de ocorrências que podem interferir na verificação dos resultados” (FEIGENBAUM, 2013, pág. 10). Estes filtros são representados por meio de operadores lógicos e matemáticos e palavras reservadas da linguagem. Além disso, são separados em classes de acordo com suas funcionalidades, sendo elas: *Logical Filters*, *Math Filters, Existence Filters,* SPARQL *Tests, Accessors Filters* e *Miscellaneous Filters*. A Tabela 2 apresenta a forma de utilização desses filtros

Tabela 2 - Tipos de filtros da linguagem SPARQL

Categoria	Funções/Operadores	Exemplos
Logical	!, &&, | |, =, !=, <, <=, >, >=	?hasPermit | | ?age < 25
Math	+, -, *, /	?decimal * 10 ?minPercent
Existence	EXISTS, NOT EXISTS	NOT EXISTS { ?p professor:email ?email }
SPARQL Tests	isURI, isBlank, isLiteral, bound	isURI(?person) | | !bound(?person)
Accessors	str, lang, datatype	lang(?title) = "pt"
Miscellaneous	sameTerm, langMatches, regex	regex(?ssn, "\\d{5}-\\d{4}")

Fonte: (FEIGENBAUM, E. 2013, adaptado)

Outra característica da linguagem SPARQL é a possibilidade de combinar duas ou mais consultas em estruturas denominadas padrões de grafo, ou *Graph Patterns*. A W3C (2008) descreve que a forma como essas combinações são realizadas pode criar uma série de estruturas distintas:

- *Basic Graph Patterns*: são consultas que utilizam um conjunto de triplas para buscar por resultados correspondentes;
- *Group Graph Patterns*: são consultas que utilizam um conjunto de padrões de grafo para buscar por resultados correspondentes;
- *Optional Graph Patterns*: são consultas que utilizam um conjunto de padrões de grafo como opção para estender os resultados;
- *Alternative Graph Patterns*: são consultas que utilizam dois ou mais conjuntos de padrões de grafo como alternativas para buscar por resultados correspondentes;

A utilização desses padrões de grafos consiste em acrescentá-los entre duas consultas, conforme apresentado na Figura 23.

Figura 23 - Exemplo de utilização dos padrões de grafo

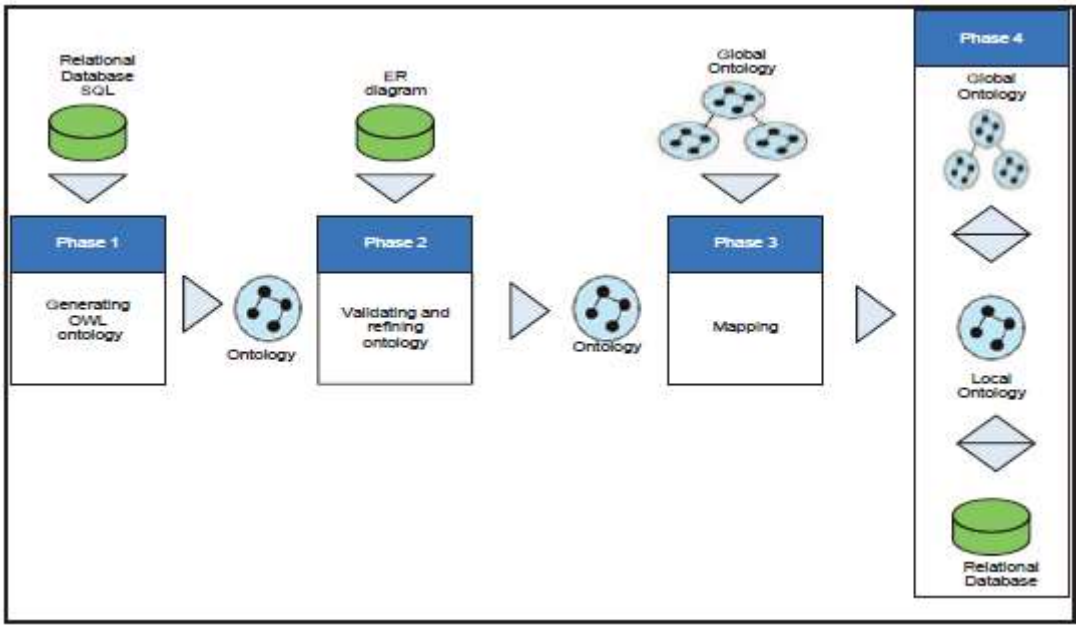
```
1  PREFIX portal: <http://w3.ulbra-to/portal/alunos#>
2  PREFIX portal: <http://w3.ulbra-to/portal/cursos#>
3  SELECT ?aluno WHERE {
4    { ?aluno alunos:idade 17 }
5    UNION
6    { ?curso cursos:nome ?nome }
7  }
```

Esta consulta objetiva reunir os resultados dos alunos da instituição que possuem idade igual a 17 anos e a lista dos nomes dos cursos da instituição.

2.3 TRABALHOS CORRELATOS

O trabalho de Alalwan, Zedan e Siewe (2009), intitulado "*Generating OWL Ontology for Database Integration"*, consiste na implementação de um protótipo de *software* capaz de realizar a transformação automática de um modelo de banco de dados relacional para uma ontologia OWL. No trabalho, os autores desenvolveram a metodologia apresentada na Figura 19.

Figura 24 - Processo geração de ontologia a partir de um modelo relacional



Fonte: (ALALWAN, N. ZEDAN, H. e SIEWE, F. 2009)

De acordo com a Figura 19, o processo foi dividido em quatro fases, sendo elas:

- geração de uma ontologia OWL a partir de instruções SQL;
- validação e refinamento da ontologia produzida, por meio de uma comparação com o modelo relacional do banco de dados;
- mapeamento da ontologia produzida (ontologia local) para uma ontologia global; e
- integração da ontologia global por meio da conexão com o banco de dados.

Nesse trabalho, os autores realizaram somente as duas primeiras fases da metodologia, e indicaram que o desenvolvimento das fases restantes seria realizado em trabalhos futuros.

Para a realização da primeira fase, geração de uma ontologia OWL a partir de instruções SQL, foram definidas uma série de regras lógicas e matemáticas, denominadas predicados. Essa definição compõe a primeira etapa desse processo, e é sucedida de mais três etapas que utilizam esses predicados como recurso, são elas: definição de classes e tipos de dados de suas

propriedades; criação das propriedades das classes; e migração das instâncias existentes no banco de dados para a ontologia.

A realização da segunda fase, validação e refinamento da ontologia produzida, ocorreu de forma manual, pois o processo ainda não havia sido automatizado juntamente à ferramenta de geração da ontologia. Para a etapa de validação, foi realizada uma comparação entre o modelo relacional do banco de dados e a ontologia produzida, de acordo com os seguintes critérios:

- o número de entidades com relações de cardinalidade com valor n (com $n > 2$) no modelo relacional deve ser igual ao número de classes na ontologia;
- relações binárias com atributos adicionais no modelo relacional devem ser contabilizadas como entidades;
- a relação "É uma" entre duas entidades no modelo relacional equivale à relação *subClassOf* entre as respectivas classes derivadas na ontologia;
- para cada relação um para muitos entre duas entidades no modelo relacional, devem existir duas propriedades inversas nos objetos da relação na ontologia, sendo uma propriedade em cada objeto;
- para cada relação muitos-para-muitos no modelo relacional, devem existir duas propriedades inversas nos objetos da relação na ontologia, porém esses objetos não fazem parte de nenhuma classe; e
- Se houver alguma relação entre mais de duas entidades no modelo relacional, devem existir duas propriedades inversas nos objetos da relação na ontologia para cada chave estrangeira existente.

O resultado obtido após a realização da comparação entre os documentos se mostrou favorável. "Ao analisar os resultados da comparação, é possível verificar que o sistema desenvolvido foi capaz de identificar corretamente as características do modelo relacional e criar os elementos equivalentes para a ontologia de acordo com as regras definidas" (ALALWAN, N. ZEDAN, H. e SIEWE, F. 2009, pág. 31).

No entanto, segundo os autores, foi necessária a realização de uma etapa de refinamento para que a ontologia resultante representasse adequadamente o domínio. Para a etapa de refinamento da ontologia foram identificadas algumas características semânticas que não haviam sido trabalhadas pelo sistema, tornando necessário sua adição à ontologia de forma manual. São elas:

- se alguma entidade possuir um atributo composto, o atributo principal deve ser representado na ontologia como uma classe, e os seus sub-atributos devem ter

propriedades relacionadas a ela. Por exemplo, na entidade `Funcionário` do modelo relacional, o atributo `Nome` possuía os sub-atributos `Primeiro Nome`, `Nome do Meio` e `Último Nome`. Dessa forma, foi criada a classe "`Nome`" e os sub-atributos foram ligados a ela por meio da relação `Tem_Nome`. Além disso, é definida a cardinalidade um para um para essa relação a fim de garantir que elas apareçam apenas uma vez.

- as restrições de cardinalidade para as relações entre as entidades também são adicionadas manualmente, pois não é possível inferir essas informações a partir das instruções SQL.

Após a realização dessas etapas e uma nova comparação entre o modelo relacional e a ontologia produzida, foi constatado que o sistema se mostrou eficaz no processo, pois conseguiu identificar todas as características descritas nas regras definidas pelos autores e gerar uma ontologia equivalente ao domínio.

Os procedimentos realizados pelos autores contribuem para este trabalho por meio das regras definidas para que o sistema identifique determinadas características em modelos relacionais e gere uma ontologia equivalente. Essas regras servem de auxílio para a identificação das informações necessárias para a criação de uma ontologia dentro de um contexto, mesmo que de forma manual. A maneira como as regras são apresentadas, em conjunto com exemplos práticos de aplicação de cada uma delas, fornece os subsídios necessários para a reutilização do processo em outros contextos, pois os autores especificam as informações utilizadas como parâmetro e seus respectivos resultados em cada uma das etapas.

Telnarova (2010), por meio do trabalho intitulado "*Relational database as a source of ontology creation*", apresenta e discute as tarefas relacionadas ao mapeamento de dados de bancos de dados relacionais para ontologias, ou a alimentação de informações em ontologias (criação de instâncias) a partir dos dados de um banco de dados relacional. Inicialmente, a autora descreve quatro áreas nas quais os trabalhos relacionados a esta temática podem ser divididos.

A primeira área abrange trabalhos relacionados à criação de um novo esquema de banco de dados baseado no esquema do banco original e no mapeamento desse esquema em uma ontologia por meio de engenharia reversa. Este novo esquema é comumente chamado de modelo médio, que representa um esquema de banco de dados relacional após a adição de características semânticas.

A segunda área abrange trabalhos cujo objetivo seja desenvolver um método baseado no preenchimento de instâncias ontológicas utilizando dados de modelos relacionais como fontes externas. Geralmente é construída uma interface XML entre a ontologia e a base de dados.

A terceira área engloba trabalhos que objetivam desenvolver um método que cria uma ontologia a partir de um esquema de um modelo relacional. Nesse processo, o objetivo é a criação de uma ontologia com RDFS. Porém, os resultados semânticos fornecidos pelos trabalhos relacionados a este método geralmente apresentam a falta de características da ontologia, como a capacidade de inferência de informações.

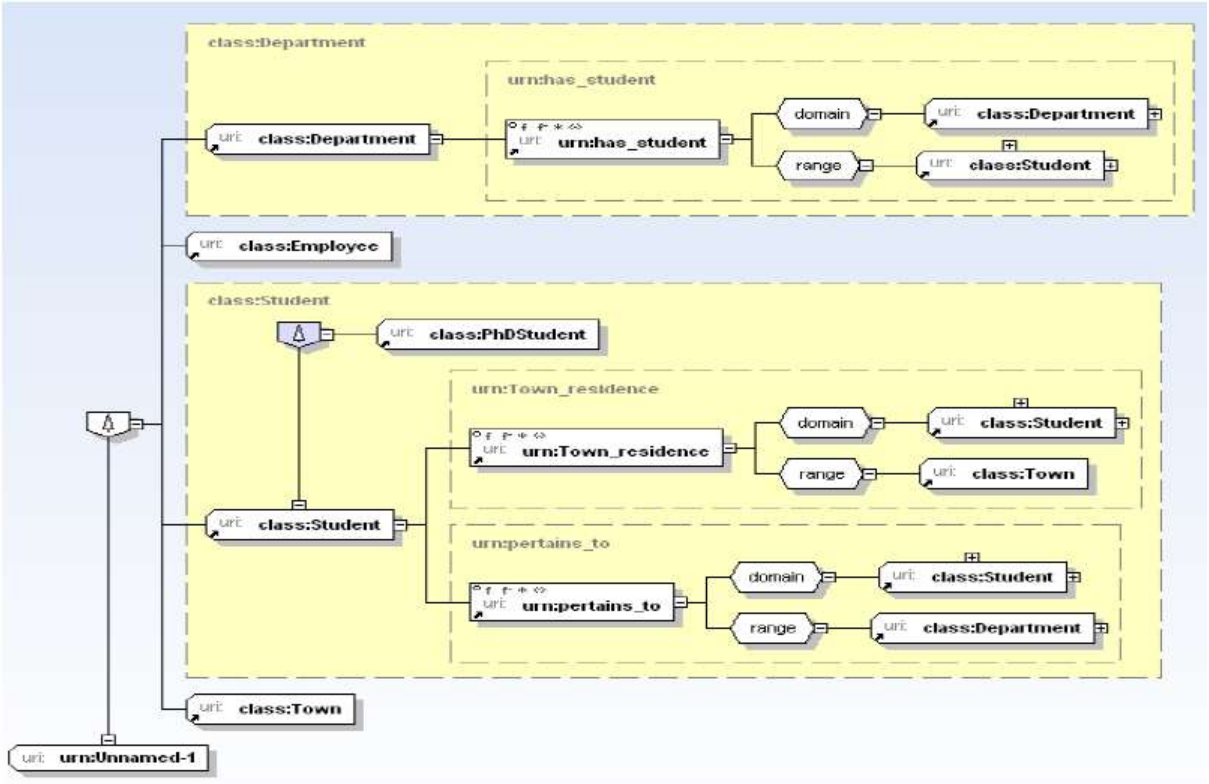
A quarta área está relacionada a trabalhos que visam ao desenvolvimento de um método que cria uma ontologia a partir de um esquema de um modelo relacional, manual ou automaticamente. Porém, visando a construção de uma ontologia OWL.

Após a descrição das áreas, Telnarova (2010) afirma que o trabalho desenvolvido está relacionado à quarta área e, posteriormente, apresenta o passo a passo dos métodos e técnicas utilizadas para o seu desenvolvimento. A maior contribuição do trabalho está na definição, apresentação e exemplificação do conjunto de regras utilizadas para o processo. Essas regras abrangem as seguintes tarefas:

- criação das classes da ontologia;
- criação das propriedades das classes;
- criação da hierarquia das classes;
- criação das cardinalidades; e
- criação de instâncias para a ontologia.

Após a definição desse conjunto de regras e a exemplificação de algumas delas com base em um modelo relacional simples, a autora apresenta um fragmento da ontologia resultante do processo de mapeamento na Figura 20.

Figura 25 - Fragmento da ontologia resultante do processo de mapeamento



Fonte: (TELNAROVA, Z. 2010)

Ao comparar os resultados esperados a partir da aplicação de cada uma das regras com o modelo ontológico apresentado na Figura 20, a autora afirmou que o processo realizado foi um sucesso, capaz de contemplar todos os pontos necessários para a criação de uma ontologia a partir de um modelo relacional de banco de dados.

O trabalho de Telnarova (2010) é de grande utilidade para o desenvolvimento deste trabalho. Assim como classificado pela autora, este trabalho também se encaixa na quarta área de abrangência de trabalhos que utilizam modelos relacionais como parâmetro para a construção de uma ontologia OWL, seja manual ou automaticamente. Além disso, as regras apresentadas servem de base para o desenvolvimento do trabalho, principalmente as Regras relacionadas à criação de instâncias para a ontologia, que definem quais características devem ser levadas em consideração para que um registro armazenado em um banco de dados relacional possa ser mapeado em uma instância de uma ontologia. As outras categorias de regras servem de auxílio, pois definem formalmente a maneira como deve ser avaliado um modelo relacional de banco de dados com o intuito de criar uma ontologia de domínio adequada.

O trabalho de Dadjoo e Kheirkhah (2015), intitulado "*An approach for transforming of relational databases to OWL ontology*", tem como objetivo apresentar os resultados de uma abordagem de transformação de um modelo relacional de banco de dados em uma ontologia por meio de um sistema. Esta abordagem foi dividida em três etapas:

- extração de metadados;
- criação de uma representação intermediária em grafo; e
- criação de uma ontologia com base em uma estrutura de grafo.

Na primeira etapa do desenvolvimento do trabalho, foi utilizado como entrada para o sistema, o código de representação do modelo relacional, escrito em linguagem SQL *Data Definition Language*. A partir disso, são extraídos alguns metadados referentes à quantidade de tabelas e relacionamentos existentes, assim como suas respectivas cardinalidades.

Na segunda etapa, o sistema utiliza os metadados extraídos para construir um grafo direcionado e nomeado, a fim de acrescentar algumas características conceituais nativas da estrutura de grafos, como um formato mais adequado de visualização das entidades e relações presentes no domínio.

Para concluir, a terceira etapa consiste na utilização da estrutura de grafo definida anteriormente para criar uma ontologia. O modelo ontológico desenvolvido busca contemplar todas as características necessárias para a definição de uma ontologia, inclusive a estrutura de hierarquias e as regras e limitações relacionadas às tabelas e às colunas do modelo inicial. Além dessas etapas, os autores adicionaram uma funcionalidade ao sistema, que apresenta um diferencial em relação a outros trabalhos dessa área. O sistema desenvolvido também é capaz de identificar as *triggers* (gatilhos) definidas no modelo relacional e criar uma classe de evento equivalente. Esta classe é capaz de identificar o evento responsável por ativar a *trigger* e desempenhar o mesmo papel, inserindo as informações necessárias na entidade ontológica adequada. Ao analisar os resultados dos modelos utilizados como exemplo, Dadjoo e Kheirkhah (2015) concluíram que "devido a utilização de mais componentes envolvidos no processo de transformação, este método obteve sucesso ao apresentar informações semânticas mais completas na ontologia resultante". Além disso, a conversão intermediária para uma estrutura de grafo permite que o sistema desempenhe suas tarefas independentemente da implementação inicial do banco de dados e também do sistema de gerenciamento de banco de dados utilizado.

Assim como os outros trabalhos apresentados, o trabalho de Dadjoo e Kheirkhah (2015) também objetiva o mapeamento de modelo relacional de banco de dados em uma ontologia. Porém, o seu diferencial consiste em uma etapa intermediária adicional, que é a criação de um

grafo a partir dos metadados extraídos do modelo relacional. No artigo, os autores descrevem de forma detalhada as regras utilizadas para a criação do grafo, de forma que é possível seguir estas etapas com a finalidade de verificar o mapeamento realizado nesse trabalho.

Dessa forma, após a análise dos trabalhos relacionados, a Tabela 3 apresenta, de maneira condensada, um comparativo que destaca as principais características de cada um.

Tabela 3 – Comparativo dos trabalhos relacionados

Autores	Entrada	Processamento	Tipo	Observações
Alalwan, Zedan e Siewe (2009)	Código SQL	Regras lógicas e matemáticas	Automático	Alguns elementos foram adicionados manualmente
Telnarova (2010)	Diagrama relacional	Regras lógicas e matemáticas	Manual	Transforma os dados do banco de dados em instâncias da ontologia
Dadjoo e Kheirkhah (2015)	Código SQL	Transformação em grafo	Automático	Identifica eventos (*triggers)* no código SQL e cria eventos equivalentes para a ontologia

Ao analisar a Tabela 3, é possível observar que, mesmo os trabalhos seguindo uma abordagem um pouco diferente, seja no conteúdo utilizado como entrada, na forma como esse conteúdo é processado ou no tipo de processamento, os resultados são semelhantes e, de acordo com os autores, válidos. Além disso, as observações presentes na tabela indicam a importância de se entender o domínio do problema, para que os devidos ajustes no processo possam ser realizados com a finalidade de alcançar um resultado relevante para a situação, pois todos os trabalhos precisaram de uma característica específica para que seus respectivos problemas pudessem ser solucionados adequadamente.

3 METODOLOGIA

3.1 MATERIAIS

Para o desenvolvimento do trabalho, foram utilizados os seguintes materiais e tecnologias:

- *Web Ontology Language* (OWL): Linguagem de representação de ontologias;
- *Resource Description Framework* (RDF): Padrão W3C para descrição de recursos;
- SPARQL: Linguagem de consulta para RDF;
- Protégé: Software de suporte à modelagem de ontologias;
- Python: Linguagem de programação de alto nível;
- *HyperText Markup Language* (HTML): padrão de marcação de hipertexto da W3C, usado para criação de páginas web;
- *Cascading Style Sheets* (CSS): padrão de estilo da W3C, usado para adicionar estilos a páginas web;
- *JQuery*: biblioteca web utilizada para manipulação de recursos gráficos em páginas HTML.

De acordo com o *Stanford Center for Biomedical Informatics Research* (BMIR) (2007), Protégé é um editor e um *framework* para a construção de sistemas inteligentes. É utilizado por uma grande comunidade de usuários acadêmicos, governamentais e corporativos que desenvolvem soluções orientadas ao conhecimento em diversas áreas, como biomedicina, *e-commerce* e modelagem organizacional. “A arquitetura do Protégé pode ser adaptada para ser utilizada em aplicações simples ou complexas que são baseadas em ontologias” (HORRIDGE et. al., 2011, pág. 10). A saída do *software* pode ser integrada com sistemas de regras ou sistemas solucionadores de problemas para a construção de sistemas inteligentes.

Segundo Rossum (2008), Python é uma linguagem de programação de fácil aprendizagem. Possui estruturas de dados de alto nível e uma abordagem de programação orientada a objetos simples, porém efetiva. Sua sintaxe dinâmica, em conjunto com sua natureza interpretada e grande quantidade de funcionalidades, fazem dela uma linguagem adequada para o desenvolvimento em várias áreas.

De acordo com a W3C (2017), HTML é uma linguagem ou idioma de marcação para criar páginas da web. Consiste em uma estrutura de marcação caracterizada por blocos e *tags*. Estas *tags* representam um determinado conteúdo especificado pelo título da etiqueta.

Segundo a W3C (2017), CSS é um mecanismo para adicionar estilo (fontes, cores etc) aos documentos da Web. “Conhecido atualmente como folha de estilo, o CSS tem como objetivo descrever como os elementos do HTML serão exibidos” (W3SCHOOLS, 2017).

JQuery, de acordo com a W3Schools (2017), é uma biblioteca de *JavaScript* rápida, leve e rica em recursos. Ela trabalha com manipulação de elementos HTML, manipulação de eventos, animação etc. Consiste em uma biblioteca simples e compatível com diversas versões dos navegadores atuais.

Para utilizar estes materiais para o desenvolvimento deste trabalho, foi necessário estudar e compreender o seu funcionamento, para que pudessem ser utilizados de forma eficaz. Com relação às linguagens utilizadas para a construção de ontologias (RDF e OWL), foi necessário compreender suas sintaxes e suas respectivas responsabilidades, de forma que fosse possível identificar e analisar os recursos criados no arquivo OWL gerado a partir da ferramenta Protégé. Com relação à linguagem utilizada para a realização de consultas em ontologias (SPARQL), foi necessário compreender sua sintaxe para a realização de consultas nas triplas da ontologia com o intuito de recuperar os dados a serem apresentados. Com relação ao software de suporte à modelagem de ontologias (Protégé), foi necessário estudar o funcionamento dos componentes de sua interface que são responsáveis pelas diversas etapas de construção de uma ontologia, como a criação de classes, propriedades de objeto, propriedade de tipos de dados, instâncias, restrições etc. Com relação à biblioteca Python para manipulação de ontologias (RDFLib), foi necessário estudar sua documentação para que fosse possível importar a ontologia modelada no Protégé para uma aplicação Python, assim permitindo a análise e o tratamento dos dados de acordo com os recursos da linguagem de programação.

3.2 PROCEDIMENTOS

Para a modelagem e implementação da ontologia, foi utilizada uma adaptação da metodologia proposta por Noy e McGuinness (2001), apresentada na Figura 26.

Figura 26 - Metodologia do Trabalho

A seguir é apresentada uma descrição das etapas da metodologia:

1. definição das classes e sua hierarquia: consistiu em identificar os termos do domínio que possuem uma existência independente. A partir disso, esses termos se tornaram classes da ontologia e foram organizados em uma taxonomia hierárquica.
2. definição das propriedades das classes: etapa em que foram definidas as relações existentes entre as classes definidas na etapa anterior.
3. definição das características das propriedades: etapa em que as regras e padrões das propriedades foram definido. Além disso, foram identificadas as classificações das propriedades de acordo com suas características específicas no domínio, ou seja, *Transitive Property, Symmetric Property, Functional Property, InverseOf* e *InverseFunctional Property*.
4. criação de instâncias das classes: consistiu na criação de um conjunto de indivíduos pertencentes às classes definidas anteriormente e o estabelecimento das relações entre eles.
5. realização e verificação de inferências: etapa em que foi utilizado o motor de inferência da ferramenta Protégé para a identificação de relações implícitas na ontologia, levando em consideração as classes e relações existentes. Além disso, as inferências foram verificadas a fim de identificar os benefícios resultantes da semântica adicionada ao domínio.
6. recuperação dos dados da ontologia: consistiu na importação da ontologia construída na ferramenta Protégé para uma aplicação Python com a utilização da biblioteca RDFLib.
7. protótipo para apresentação das informações: etapa em que foram desenvolvidos os protótipos de telas para serem apresentados no portal acadêmico.

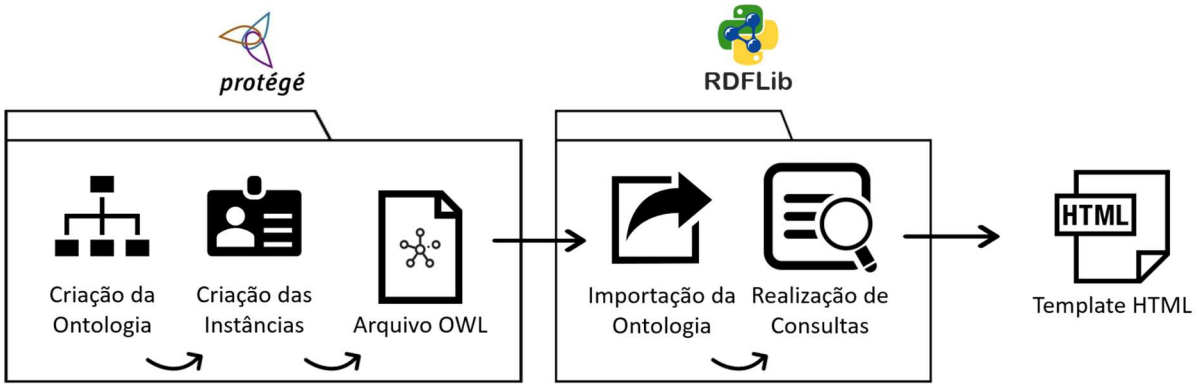
4 RESULTADOS

Esta seção apresenta e descreve a estrutura e os procedimentos realizados para o desenvolvimento do processo.

4.1 ESTRUTURA

A **Erro! Fonte de referência não encontrada.** apresenta, de forma gráfica, a estrutura do processo desenvolvido nesse trabalho.

Figura 27 - Estrutura do Processo



Conforme a Figura 27, o processo foi dividido em dois módulos, que podem ser caracterizados por suas saídas, que são um arquivo OWL e um arquivo JSON, respectivamente.

O primeiro módulo refere-se aos procedimentos realizados na ferramenta Protégé e são descritos a seguir.

- etapa de Criação da Ontologia: foram definidas as classes, propriedades das classes e características das propriedades das classes referentes ao domínio do trabalho. A interface do Protégé permitiu que essas tarefas fossem realizadas de forma visual, sem a necessidade de manipular diretamente o código OWL ou RDF necessário para isso.
- etapa de Criação de Instâncias: ainda realizada na ferramenta Protégé, consistiu na criação de alguns indivíduos referentes às classes anteriormente definidas, com o intuito de verificar os resultados do enriquecimento semântico de um domínio a partir da realização de inferências. Essas inferências foram realizadas pelo mecanismo de inferências (*Reasoner*) da própria ferramenta, cujos detalhes são mais explicados nas seções a seguir.

- etapa de Criação do Arquivo OWL: após a realização desses procedimentos, foi utilizada a funcionalidade de exportação da ontologia para um arquivo OWL, presente da ferramenta Protégé. Este arquivo possui o código que descreve todos os elementos criados por meio da ferramenta, porém eles ainda não estão estruturados em formato de triplas RDF, que é o formato desejável para se trabalhar com os dados de uma ontologia.

Com base nisso, o segundo módulo refere-se à importação da ontologia para uma aplicação Python para gerar as triplas RDF e recuperar os dados da ontologia, cujo recurso utilizado para a realização dessas tarefas foi a biblioteca Python RDFLib.

- etapa de Importação da Ontologia: foi possível importar a ontologia para a aplicação passando a referência do local onde o arquivo OWL se encontrava na máquina. Para trabalhar com a ontologia no código, a biblioteca fornece uma estrutura de dados denominada *Graph* (Grafo) que ao receber um código OWL, gera e armazena as triplas RDF equivalentes.
- etapa de Realização de Consultas: para realizar as consultas aos dados, a biblioteca disponibiliza duas alternativas: um *plugin* que reconhece e executa consultas SPARQL na estrutura de grafo e um conjunto de métodos Python que recebem os parâmetros de consulta no formato de triplas RDF e retornam os resultados encontrados, de forma que não é necessário lidar diretamente com a linguagem SPARQL. O método escolhido para realizar as consultas de dados foi o descrito na segunda alternativa. Assim, as consultas foram realizadas com o intuito de recuperar todas as informações acerca dos indivíduos pertencentes aos perfis de Professor e Aluno, por exemplo, todas as triplas RDF que estejam relacionadas ao indivíduo Matheus.
- etapa de Criação de Template HTML: levando em consideração esses dados, foram construídos *Templates* HTML para apresentar as informações de cada um dos perfis, de Professor e Aluno. No entanto, esse processo foi realizado de forma manual.

4.2 DESENVOLVIMENTO

Esta seção tem como objetivo descrever os resultados obtidos em cada uma das etapas do desenvolvimento do trabalho. As etapas foram divididas em: seção 4.2.1, Criação da Ontologia, apresenta os resultados da execução da adaptação da metodologia de Noy e McGuinness (2001), da Universidade de Stanford; seção 4.2.2, Realização e Verificação de

Inferências, apresenta e discorre sobre as relações inferidas pelo mecanismo da ferramenta Protégé acerca das características definidas na ontologia; seção 4.2.3, Recuperação dos Dados da Ontologia, discorre sobre o processo de carregamento dos dados da ontologia para uma aplicação Python.

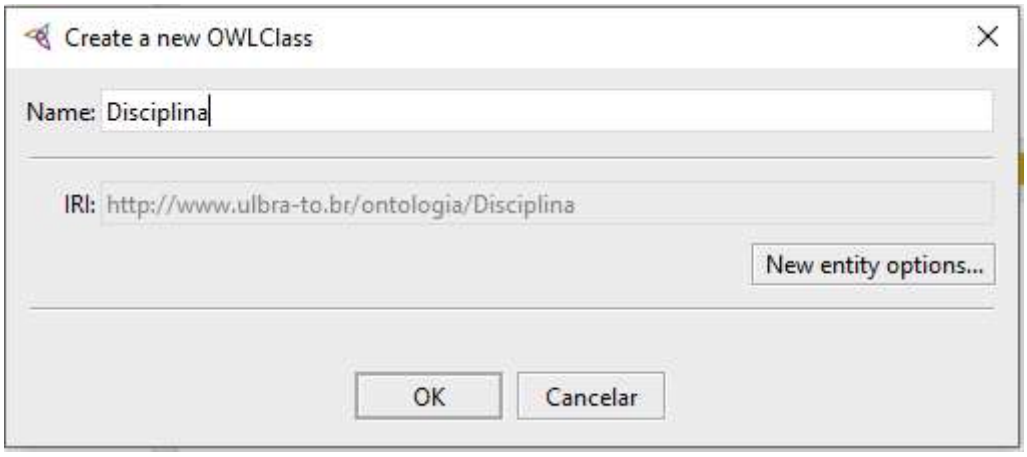
4.2.1 Criação da Ontologia

Esta seção descreve os procedimentos realizados para a criação da ontologia de domínio utilizada nesse trabalho, de acordo com uma adaptação da metodologia proposta por Noy e McGuinness (2001), da Universidade de Stanford. O *software* utilizado para a realização desses procedimentos foi a ferramenta Protégé, versão 5.2.0.

4.2.1.1 Definição das classes e sua hierarquia

A definição da hierarquia utilizada levou em consideração a identificação dos papéis das classes de acordo com a seguinte regra: Se a classe A é superclasse da classe B, então toda instância de B é também uma instância de A. Para a realização dessa etapa existem três abordagens: *top-down*, *bottom-up* e híbrida, que são, respectivamente, partir dos conceitos genéricos para os mais específicos, partir dos específicos para os mais genéricos, e definir o conceitos das extremidades e adicionar especificações e generalizações até chegar a um ponto de encontro. A abordagem utilizada foi a híbrida, que se iniciou com a definição das classes `Pessoa` e `Curso` sendo, posteriormente, definidas as outras classes relacionadas ao domínio do trabalho. Para criar uma classe na ontologia utilizando a ferramenta Protégé é necessário selecionar a funcionalidade e definir um nome para ela, conforme apresentado na Figura 28.

Figura 28 - Criação de Classe

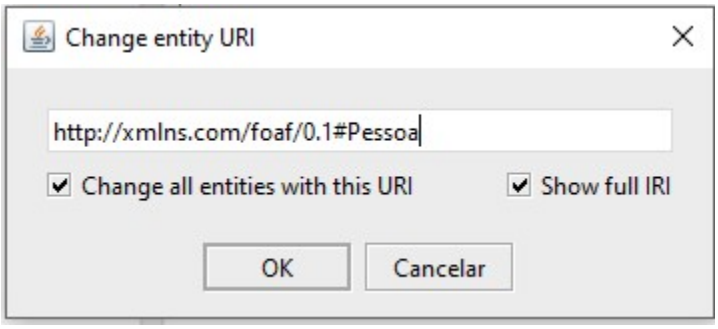


Na Figura 28é possível visualizar que, ao indicar o nome do elemento, é apresentada uma prévia da URI (*Uniform Resource Identifier*) completa que representará esse elemento dentro da ontologia. Dessa forma, a partir do momento em que a ontologia for compartilhada na internet, os seus elementos e relações também se tornam disponíveis por meio desse identificador.

A ferramenta Protégé não fornece uma funcionalidade que permita importar uma classe de outra ontologia para ser utilizada no desenvolvimento do trabalho, como a FOAF (*Friend of a Friend*), por exemplo. Para isso, é necessário importar a ontologia completa e acrescentar os elementos da nova ontologia a ela. Porém, é possível referenciar uma ontologia a partir da indicação de sua URI no momento da criação de uma classe. Dessa forma, ao exportar a ontologia para um arquivo OWL, a ferramenta indica no código que há uma referência para essa ontologia externa, que passa a funcionar como uma importação a partir do momento em que o código for compartilhado na internet.

Com base nisso, a classe `Pessoa` foi criada referenciando a classe `Pessoa` da ontologia FOAF, conforme apresentado na Figura 29.

Figura 29 - Referência da Classe Pessoa à Ontologia FOAF



Dessa forma, uma vez que esta ontologia seja compartilhada na internet, qualquer pessoa poderá ter acesso às subclasses e propriedades relacionadas a essa classe ao referenciar o elemento desejado presente na ontologia FOAF. A Figura 30 apresenta o código gerado pela ferramenta Protégé para indicar que a classe `Pessoa` faz referência à ontologia FOAF.

Figura 30 - Código OWL de Referência à Ontologia FOAF

```
622     <!-- http://xmlns.com/foaf/0.1#Pessoa -->
623
624     <owl:Class rdf:about="http://xmlns.com/foaf/0.1#Pessoa"/>
```

O código demonstrado na Figura 30 garante que a classe Pessoa e todas as suas características definidas na ontologia sejam adicionadas à ontologia FOAF, pois a utilização

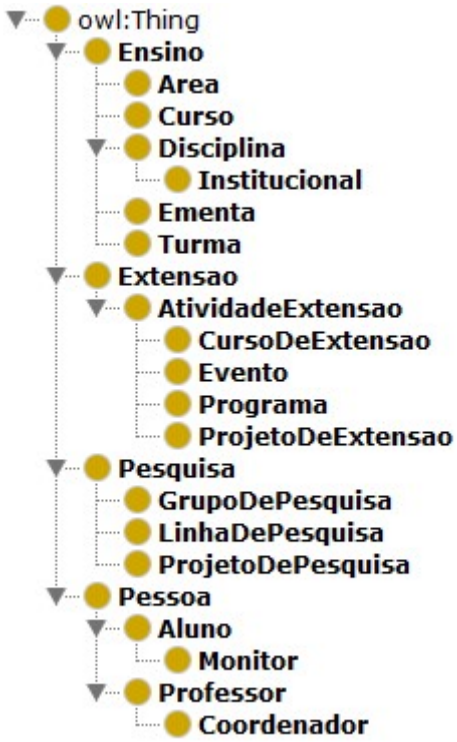
da *tag* `rdf:about` tem o objetivo de indicar que um determinado recurso (classe `Pessoa`) faz referência a um recurso existente em uma outra ontologia (FOAF).

Além da classe Pessoa, utilizada para se referir a Professor e Aluno, também foram criadas outras classes que representam contextos dentro do domínio. A Figura 31 apresenta a lista de classes que representam os contextos levados em consideração para a construção da ontologia.

Figura 31 - Lista de Classes

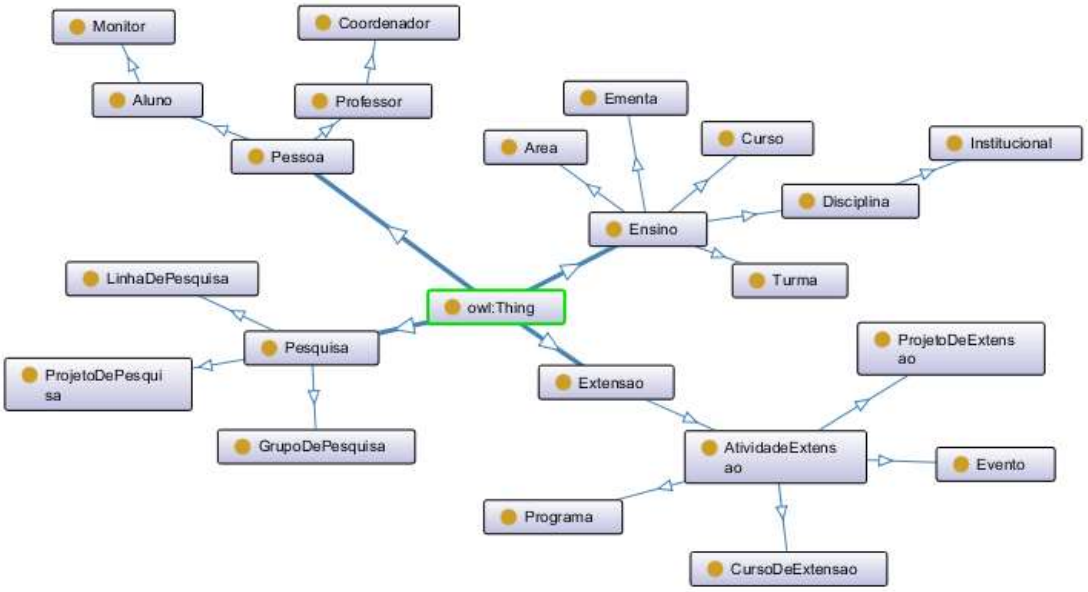
As classes apresentadas na Figura 31 foram escolhidas pelo fato de representarem os pilares fundamentais de uma instituição de ensino, sendo elas `Ensino`, `Extensao`, `Pesquisa` e `Pessoa`. Por esse motivo, é possível definir várias entidades e relações inseridas nesses contextos. A Figura 32 apresenta a lista completa das classes definidas, indicando as entidades pertencentes a cada um dos contextos citados anteriormente.

Figura 32 - Lista Completa de Classes



Para a definição das classes no formato apresentado na Figura 32, foi necessário estabelecer uma hierarquia entre elas. A Figura 33 apresenta de forma gráfica a hierarquia de classes estabelecida.

Figura 33 - Hierarquia de Classes



A Figura 33 permite a visualização da ontologia de domínio em uma estrutura de grafo direcionado, deixando explícita a hierarquia de classes estabelecida. É importante ressaltar que as classes que representam os contextos (`Ensino`, `Pesquisa`, `Extensao` e `Pessoa`), são utilizadas somente para controle e organização dos elementos subordinados a elas, ou seja, essas classes não possuem indivíduos diretamente relacionados (são classes abstratas).

4.2.1.2 Definição das propriedades das classes

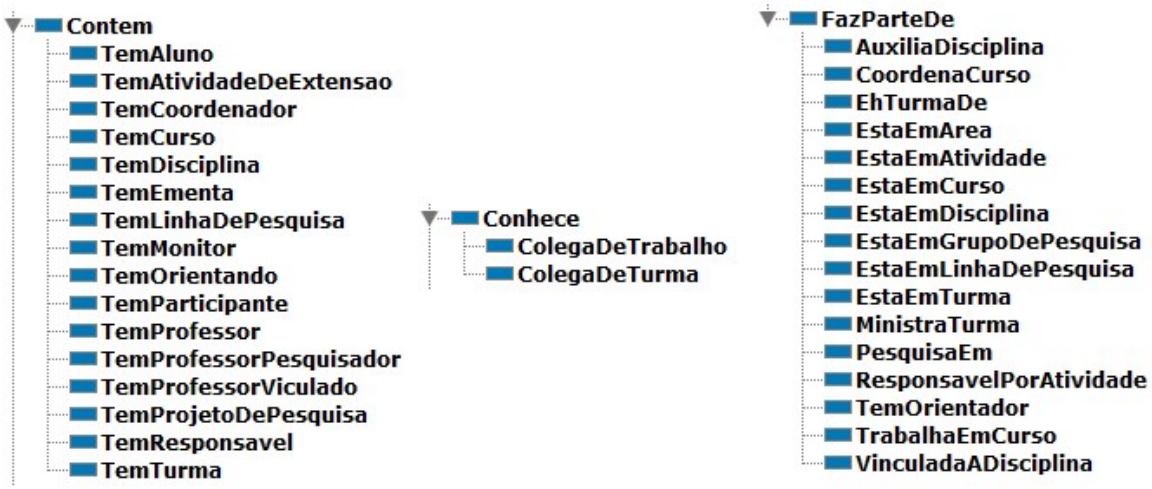
Essa seção descreve a etapa de criação das propriedades de objeto (*Object Properties*) na ontologia. Uma vez que as classes e a hierarquia foram definidas, a próxima etapa foi a de definição das propriedades das classes, ou seja, as demais formas de relacionamento entre elas, além da relação de subclasse. A criação de propriedades segue um processo semelhante ao de criação de classes, demonstrado na Figura 28.

Na linguagem OWL as propriedades, assim como as classes, também podem possuir relações de herança. Essa funcionalidade permite a definição de propriedades mais abrangentes, que relacionam classes mais genéricas e, posteriormente, a definição de propriedades filhas, que representam relações mais específicas entre as classes. A Figura 34 apresenta as propriedades de objeto mais abrangentes criadas.

Figura 34 - Lista de Propriedades de Objeto

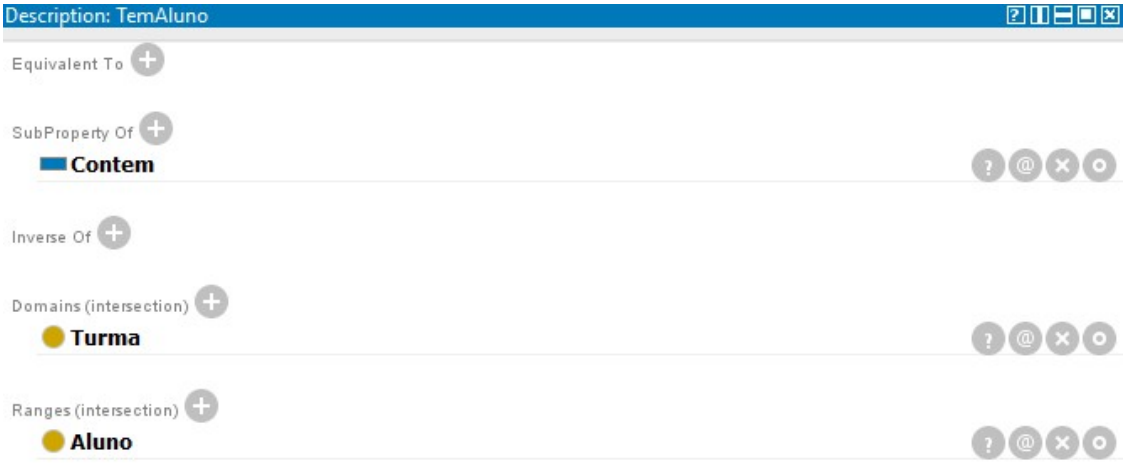
As propriedades apresentadas na Figura 34, por serem mais abrangentes, representam relações entre classes genéricas, ou seja, possuem seus respectivos domínios (*Domain*) e alcance (*Range*) preenchidos com essas classes. No entanto, estas propriedades foram definidas com o mesmo propósito das classes de contexto citadas anteriormente, para controle e organização dos elementos subordinados a elas, ou seja, elas não são utilizadas diretamente para relacionar instâncias de classes. As propriedades utilizadas para este fim são apresentadas na Figura 35.

Figura 35 - Lista Completa de Propriedades de Objeto



Como apresentado na Figura 35, cada uma das propriedades genéricas possui um conjunto de propriedades filhas. Esse conjunto de propriedades permite o estabelecimento de relações entre todas as classes definidas anteriormente. Após a criação das propriedades, é necessário definir quais as classes que são relacionadas por meio delas. A Figura 36 apresenta a funcionalidade que permite que esse processo seja realizado na ferramenta Protégé.

Figura 36 - Definição das Classes Relacionadas por uma Propriedade



A Figura 36 apresenta a funcionalidade que permite definir o domínio (*Domain*) e alcance (*Range*) das propriedades criadas anteriormente. O exemplo demonstrado na figura refere-se à propriedade `TemAluno`, indicado no topo da janela, que estabelece uma relação em as classes `Turma` e `Aluno`.

A Tabela 4 apresenta os respectivos domínios e alcances definidos manualmente para cada uma das propriedades.

Tabela 4 - Lista de Domínios e Alcances de cada Propriedade

Domínio	Propriedade	Alcance
Pessoa	Conhece	Pessoa
Professor	ColegaDeTrabalho	Professor
Aluno	ColegaDeTurma	Aluno
Thing	Contem	*Thing*
Disciplina	TemAluno	Aluno
Curso	TemAtividadeExtensao	AtividadeExtensao
Curso	TemCoordenador	Coordenador
Area	TemCurso	Curso
Curso	TemDisciplina	Disciplina
Disciplina	TemEmenta	Ementa
GrupoDePesquisa	TemLinhaDePesquisa	LinhaDePesquisa
Disciplina	TemMonitor	Monitor
Professor	TemOrientando	Aluno
AtividadeExtensao	TemParticipante	Pessoa
Disciplina	TemProfessor	Professor
ProjetoDePesquisa	TemProfessorPesquisador	Professor
Curso	TemProfessorVinculado	Professor
LinhaDePesquisa	TemProjetoDePesquisa	ProjetoDePesquisa
AtividadeExtensao	TemResponsavel	Professor
Disciplina	TemTurma	Turma

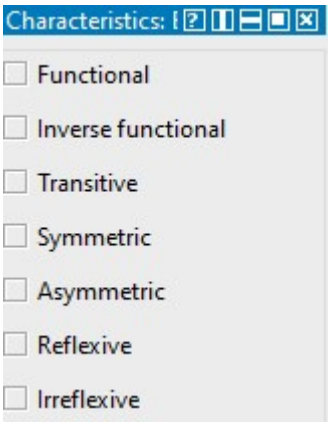
A Tabela 4 especifica quais classes se relacionam por meio de cada uma das propriedades definidas para a ontologia, indicando seus respectivos domínios e alcances, que podem ser considerados como origem e destino da relação, respectivamente.

4.2.1.3 Definição das características das propriedades

Esta seção descreve o processo de classificação das propriedades de objeto de acordo com as características apresentadas na seção 2.2.1 do Referencial Teórico, sendo elas: *Functional, Inverse Functional, Transitive, Symmetric InverseOf*, *Asymmetric*, *Reflexive* e *Irreflexive*.

Para definir as características das propriedades na ferramenta Protégé, é necessário selecioná-las individualmente para cada propriedade, conforme apresentado na Figura 37.

Figura 37 - Definição de Características das Propriedades



Esse processo foi realizado em todas as propriedades criadas. Assim, a Tabela 5 apresenta a distribuição das propriedades de objeto de acordo com suas características.

Tabela 5 - Distribuição das Propriedades de Objeto por Característica

Característica	Propriedades
Functional	TemCoordenador, TemEmenta, EhTurmaDe, EstaEmArea
Inverse Functional	EstaEmDisciplina
Transitive	Contem, FazParteDe
Symmetric	Conhece, ColegaDeTrabalho, ColegaDeTurma
InverseOf	TemAluno, TemAtividadeExtensao, TemCoordenador, TemCurso, TemDisciplina, TemEmenta, TemLinhaDePesquisa, TemMonitor, TemOrientando, TemParticipante, TemProfessor, TemProfessorPesquisador, TemProfessorVinculado, TemProjetoDePesquisa, TemResponsavel, TemTurma, AuxiliaDisciplina, CoordenaCurso, EhTurmaDe, EstaEmArea, EstaEmAtividade, EstaEmCurso, EstaEmDisciplina, EstaEmGrupoDePesquisa, EstaEmLinhaDePesquisa, EstaEmTurma,

Característica	Propriedades
	MinistraTurma, PesquisaEm, ResponsavelPorAtividade, TemOrientador, TrabalhaEmCurso, VinculadaADisciplina
Asymmetric	
Reflexive	
Irreflexive	Conhece, ColegaDeTurma

Para cada propriedade definida como subpropriedade de `Contem`, foi criada uma propriedade inversa como subpropriedade de `FazParteDe`. Por esse motivo, todas elas são classificadas como *InverseOf*.

As propriedades *Symmetric*, *Asymmetric*, *Reflexive*, *Irreflexive* só podem existir quando os elementos relacionados por elas são da mesma classe. Além disso, cada par dessas propriedades funciona de forma excludente. Logo, como as únicas propriedades que relacionam elementos de uma mesma classe (`Conhece`, `ColegaDeTrabalho`, `ColegaDeTurma`) já foram definidas como *Symmetric* e *Irreflexive*, não há nenhuma propriedade definida como *Asymmetric* ou *Reflexive*.

4.2.1.4 Criação das instâncias das classes

Esta é a etapa em que os indivíduos das classes apresentadas anteriormente foram criados e suas respectivas relações foram estabelecidas. Na ferramenta Protégé, por padrão, um indivíduo é criado independente de uma classe ou relação, somente com um nome para identificação. O processo de criação de instâncias também é semelhante ao processo de criação de classes, apresentado anteriormente.

Depois de criada a instância, a classe à qual ela se refere é adicionada por meio da funcionalidade apresentada na Figura 38.

Figura 38 - Definição da Classe da Instância

A instância `GestaoTecnologica1`, utilizada para representar o processo de criação de instâncias, foi definida como indivíduo da classe `Disciplina`, conforme apresentado na Figura 38.

Após a definição da classe da instância, as relações que ela mantém com os outros elementos já criados são adicionadas por meio da funcionalidade apresentada na Figura 39.

Figura 39 - Definição das Relações da Instância

A Figura 39 apresenta uma relação da instância `GestaoTecnologica1`, que, ao ser convertida para o formato de tripla RDF, se refere a `GestaoTecnologica1 TemTurma 0300`. Vale ressaltar que, para estabelecer essa relação, foi necessário criar a instância `0300` e defini-la como indivíduo da classe `Turma` utilizando o processo apresentado anteriormente nas Figuras 38 e 39.

Dessa forma, o conjunto de instâncias criada seguindo o mesmo processo é apresentado na Figura 40.

Figura 40 - Lista de Instâncias

Como as instâncias são criadas de forma independente a qualquer classe ou elemento, a organização utilizada pela ferramenta é a ordem alfabética, como pode ser visualizada na Figura 40.

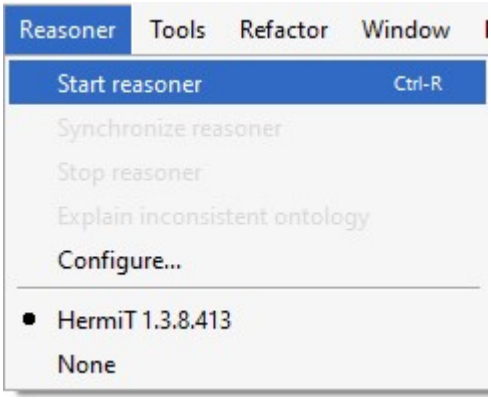
4.2.2 Realização e verificação de inferências

A definição das características das propriedades, conforme apresentado na seção 4.2.2.3, permite que a ferramenta Protégé seja capaz de inferir algumas informações acerca do domínio por meio de seu mecanismo de inferência (*Reasoner*) nativo, denominado HermiT. Enquanto o *Reasoner* permanecer desativado, a ferramenta apresenta somente as relações que foram manualmente definidas. A Figura 41 permite a visualização de um exemplo disso a partir do indivíduo `0300`, que é uma instância de `Turma`.

Figura 41 - Relações Definidas Manualmente

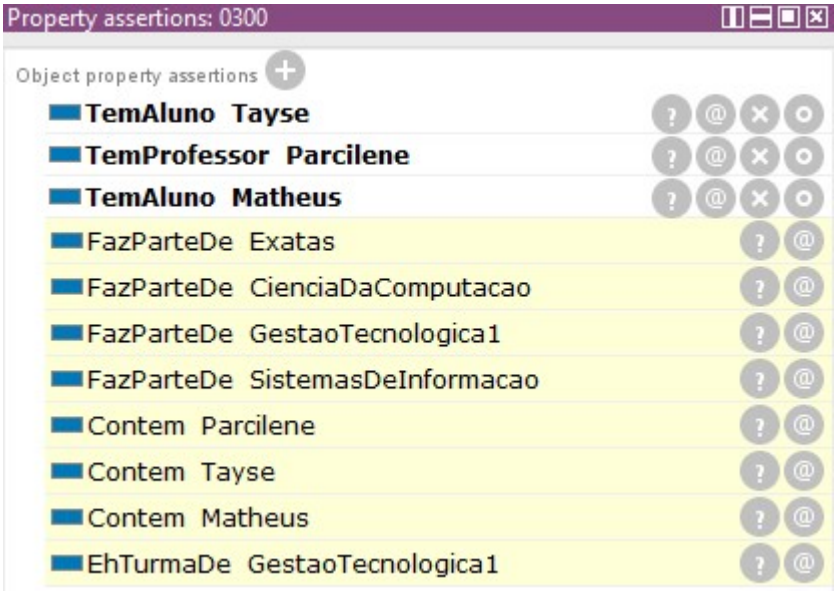
A Figura 41 mostra que somente três relações foram definidas diretamente a partir do indivíduo `0300`, porém no indivíduo `GestaoTecnologica1`, que é uma instância da classe `Disciplina`, existe mais uma relação que está diretamente ligada ao indivíduo `0300`, que é representada pela tripla RDF `GestaoTecnologica1 TemTurma 0300`. No entanto, existe uma série de outras relações que podem ser inferidas a partir dessas quatro. Esse processo é realizado automaticamente pela ferramenta a partir da ativação do mecanismo de inferência. Para ativar o mecanismo, basta selecionar a opção *Reasoner* e depois *Start Reasoner*, que aparecem no menu superior da ferramenta, conforme apresentado na Figura 42.

Figura 42 - Ativação do Mecanismo de Inferência



Uma vez que o *Reasoner* é ativado, conforme a Figura 42, as relações que podem ser inferidas, a partir das outras relações existentes e de suas respectivas características, passam a aparecer juntamente com as relações definidas manualmente, porém com uma formatação diferente para destacar. A Figura 43 demonstra o resultado desse processo.

Figura 43 - Resultado do Processo de Inferência do *Reasoner*



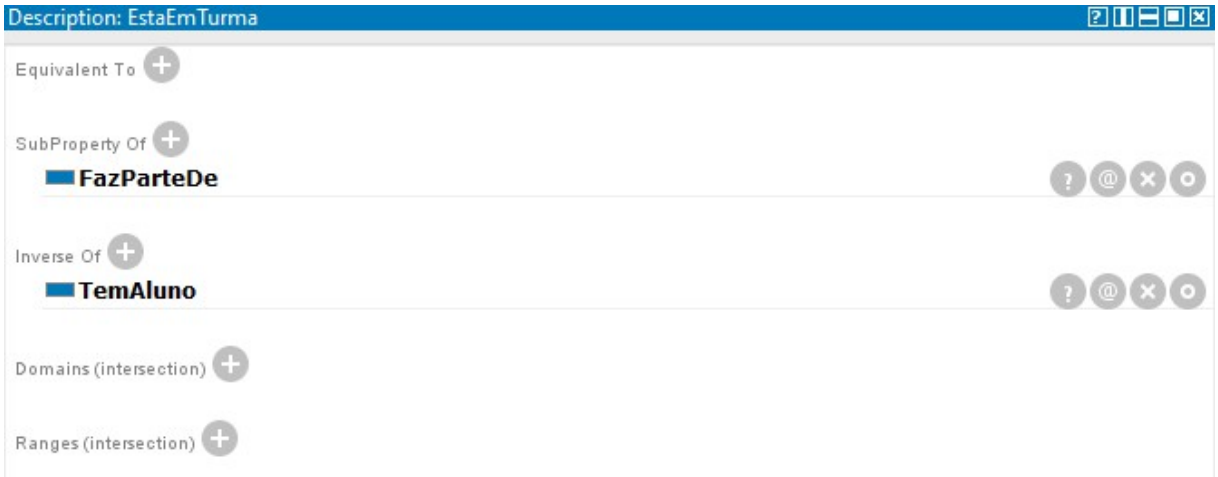
Os resultados apresentados na Figura 43 só foram alcançados a partir da adição de semântica ao contexto, que é uma das características da utilização de ontologias para a representação de conhecimento. Como todas as relações envolvidas nesse contexto são *Transitive* e *InverseOf*, é possível chegar a esses resultados a partir da análise delas. Por exemplo, existem as propriedades `TemCurso`, e `TemDisciplina`, que relacionam uma `Area` a um `Curso` e um `Curso` a uma `Disciplina`, respectivamente. Ao analisá-las em conjunto com a propriedade `TemTurma`, citada anteriormente, o *Reasoner* é capaz de inferir

a área da disciplina em que uma turma está inserida, mesmo que não existam ligações diretas entre esses elementos. Isso pode ser verificado por meio da relação `FazParteDe Exatas`. Além disso, como ambos os indivíduos de `Curso`, `SistemasDeInformacao` e `CienciaDaComputacao`, possuem a relação `TemDisciplina` com o indivíduo `GestaoTecnologica1`, o *Reasoner* também é capaz de inferir que a turma `0300 FazParteDe SistemasDeInformacao` e `CienciaDacomputacao`.

Além das inferências realizadas acerca das instâncias, também foram realizadas algumas inferências diretamente nas propriedades, de acordo com as suas características. Por exemplo, a propriedade `FazParteDe` foi definida como *InverseOf* da propriedade `Contem`, assim como cada propriedade filha de `FazParteDe` também foi definida como *InverseOf* de alguma propriedade filha de `Contem` (como `TemAluno` e `EstaEmTurma`). Por esse motivo, não foi necessário definir as classes relacionadas (domínio e alcance) por meio de nenhuma das propriedades filhas de `FazParteDe`, pois o mecanismo de inferência foi capaz de identifica-los automaticamente.

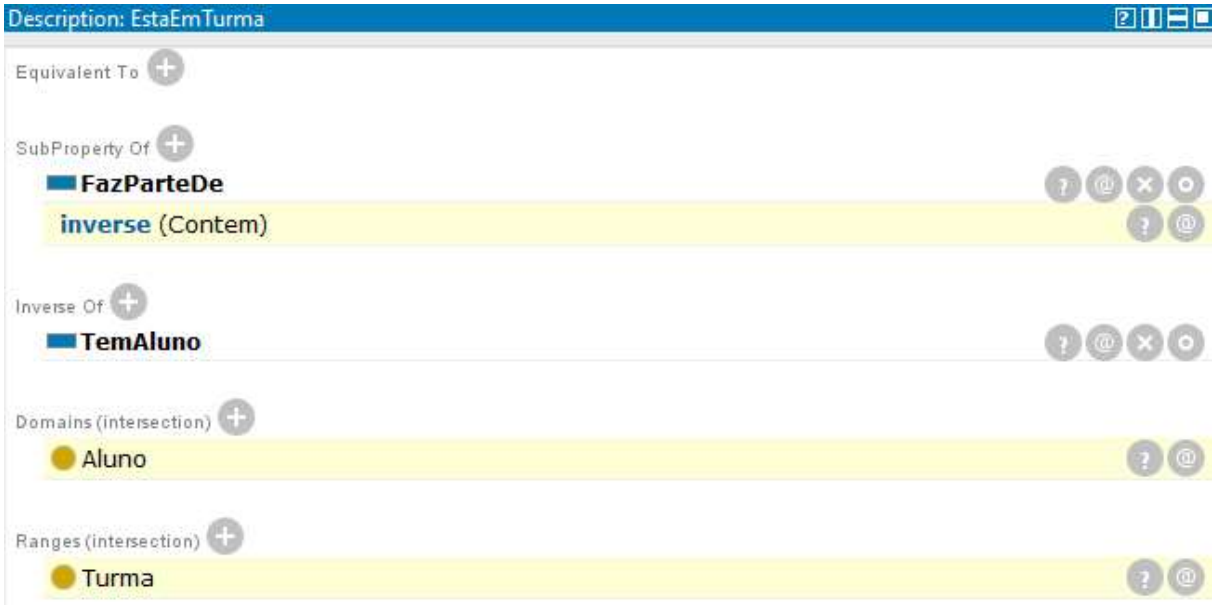
A Figura 44 apresenta a tela de definição de domínio e alcance da propriedade `EstaEmTurma` enquanto o *Reasoner* encontra-se desativado.

Figura 44 - Classes Relacionadas pela Propriedade EstaEmTurma



Na Figura 44, é possível verificar que não há classes definidas manualmente como domínio ou alcance. Também pode-se observar que ela foi definida como *InverseOf* da propriedade `TemAluno`. A Figura 45 demonstra o comportamento da tela após a ativação do *Reasoner*.

Figura 45 - Inferência das Classes Relacionadas pela Propriedade EstaEmTurma



Levando em consideração as informações definidas na propriedade `TemAluno`, o mecanismo de inferência foi capaz de identificar o domínio e o alcance da propriedade `EstaEmTurma`, conforme apresentado na Figura 45. Este mesmo resultado foi obtido em todas as outras propriedades filhas da propriedade `FazParteDe`, diminuindo a realização de tarefas de forma manual a partir da adição de semântica ao contexto. Após isso, esses resultados foram exportados para um arquivo OWL para ser utilizado nas etapas seguintes.

4.2.3 Recuperação dos dados da ontologia

Esta seção descreve os procedimentos e apresenta os resultados da etapa de recuperação dos dados da ontologia, que consiste na importação da ontologia construída na ferramenta Protégé para uma aplicação Python. Para realizar esse processo, foi utiliza a biblioteca Python RDFLib, que possui funcionalidades que auxiliam na utilização das linguagens RDF, OWL e SPARQL.

Inicialmente, antes de importar a ontologia, foi necessário declarar uma instância da estrutura de dados fornecida pela biblioteca para interpretar o código OWL da ontologia. Essa estrutura de dados é denominada *Graph* (Grafo). Posteriormente, foi utilizado o método `parse()` dessa estrutura para indicar a referência do local onde o arquivo OWL estava armazenado na máquina. A Figura 46 demonstra o código utilizado para a realização destes passos.

Figura 46 - Utilização da Estrutura *Graph* da Biblioteca RDFLib

```
4   g = Graph()
5   g.parse("Ontologia.owl")
```

Nesse caso, como o arquivo OWL estava localizado no mesmo diretório da aplicação Python, foi necessário indicar somente o nome do arquivo. Ao realizar estes passos, as triplas RDF já foram geradas e armazenadas na instância da estrutura de grafo (variável `g`). Isso pode ser verificado ao utilizar um comando de impressão para imprimir o conteúdo da variável `g` por meio de um laço de repetição. Ao utilizar a função `len()` para medir o comprimento do conteúdo da variável, foi obtido o resultado 362, que indica que foram geradas trezentas e sessenta e duas triplas RDF relacionadas à ontologia descrita no arquivo OWL.

Uma vez concluída essa etapa, o próximo passo foi a vinculação do *namespace* da ontologia a uma variável, com o intuito de simplificar o processo de realização de consultas. Este passo consistiu na utilização da função `Namespace()` da biblioteca RDFLib para armazenar a *URI* da ontologia em uma variável, que passa a ser um apelido. A Figura 47 demonstra o código utilizado para a realização desse procedimento.

Figura 47 – Vinculação da URI da Ontologia a uma Variável

```
9   TCC = Namespace('http://www.ulbra-to.br/ontologia#')
10  g.bind('tcc', TCC)
```

A função `bind()`, apresentada abaixo da função `Namespace()` na Figura 47, é utilizada para realizar um processo semelhante, porém ao invés de vincular a *URI* a uma variável, ela é vinculada a uma *string*, nesse caso, a *string* `tcc`. A realização desse processo permite que, ao utilizar a *string* `tcc` na definição de uma consulta, por exemplo, essa *string* passa a ter o valor da *URI* da ontologia.

A partir disso, a próxima etapa foi a realização das consultas. O comando utilizado para a realização das consultas é apresentado na Figura 48.

Figura 48 - Comando para Realização de Consultas

```
12  for s,p,o in g.triples( (None, None, TCC.Matheus) ):
13      print(s, p, o)
```

Este comando consiste na varredura dos resultados da função `triples()` por meio da utilização de um laço de repetição com três argumentos. A função `triples()` tem como objetivo retornar todas as triplas RDF que se adequem à estrutura dos valores passados como

parâmetro para a função. Os parâmetros da função representam o *subject* (sujeito), *predicate* (predicado) e *object* (objeto) de uma tripla RDF, respectivamente. Este é o mesmo significado dos argumentos utilizados no laço de repetição. Dessa forma, a consulta apresentada na Figura 48 tem como objetivo obter todas as triplas RDF que tenham o indivíduo `Matheus` como objeto. Os resultados obtidos com a realização dessas consultas foram utilizados para a construção do protótipo de tela para apresentação das informações

4.2.4 Protótipo para apresentação das informações

Esta seção tem como objetivo apresentar e descrever as telas construídas como protótipo para apresentação das informações. Durante o processo de construção das telas, buscou-se enfatizar a apresentação de informações relevantes para os perfis de Professor e Aluno dentro do contexto de uma instituição de ensino superior. A Figura 49 apresenta o protótipo de tela construído para apresentar as informações do perfil do aluno.

Figura 49 - Tela do Perfil de Aluno

As informações apresentadas na Figura 49 são resultantes das consultas que envolvem o indivíduo `Matheus Rodrigues Leal`, da classe `Aluno`. A descrição dos campos destacados na tela é apresentada a seguir.

- na seção mais à esquerda (A) há uma área para a apresentação de algumas informações pessoais do indivíduo, como foto, nome completo e um texto de descrição.
- a caixa de informações do curso (B) indica o curso em que o aluno está matriculado.
- a seção de turmas (C) apresenta as turmas em que o aluno está matriculado no semestre atual.
- a relação de envolvimento em grupos de pesquisa (D) apresenta os grupos de pesquisa nos quais o aluno está inserido. Esta informação foi obtida por meio das propriedades *Transitive* e *InverseOf* existentes entre os elementos Grupo de Pesquisa, Linha de Pesquisa, Projeto de Pesquisa, Professor Orientador e Aluno.
- a seção de projetos (E) apresenta os projetos de pesquisa em que o aluno já trabalhou. Esta informação foi obtida por meio das propriedades *Transitive* e *InverseOf* entre os elementos Projeto de Pesquisa, Professor Orientador e Aluno.
- a caixa de informações das atividades de extensão (F) apresenta as atividades em que o aluno esteve envolvido, de forma que as que estão destacadas em verde representam as atividades em que o aluno atuou como palestrante ou instrutor.
- a seção de colegas de classe (G) apresenta uma relação dos outros alunos que estavam matriculados nas mesmas turmas em que o aluno em questão esteve. Estas informações foram obtidas por meio da propriedade *Symmetric* existente entre os indivíduos da classe Aluno.
- a última seção, das atividades interdisciplinares (H), apresenta uma relação de todas as atividades que possuem outros cursos envolvidos, além do curso do aluno. No início de cada atividade há uma indicação de sua categoria, que pode ser Ensino, Pesquisa ou Extensão. Estas informações foram obtidas por meio das propriedades *Transitive* entre os elementos Turma, Atividade de Extensão, Aluno, Projeto de Pesquisa e Curso.

Além das informações explícitas na tela, os elementos representados em formato de botão (elementos das seções de B a F) possuem algumas outras informações que são exibidas por meio de componentes denominados *tooltips*, que funcionam quando o usuário passa o *mouse* sobre o elemento específico na tela. A Figura 50 demonstra o funcionamento desse componente ao apresentar as informações do elemento Curso para um perfil de Aluno.

Figura 50 - Informações no Componente *Tooltip*

As informações apresentadas no componente, conforme a Figura 50, são obtidas a partir das relações acerca do elemento selecionado e do perfil do usuário, ou seja, elementos de classes diferentes apresentam informações diferentes, que também podem variar de acordo com o perfil. A Figura 51 apresenta a relação das informações exibidas na *tooltip* de cada elemento, de acordo com o perfil do usuário.

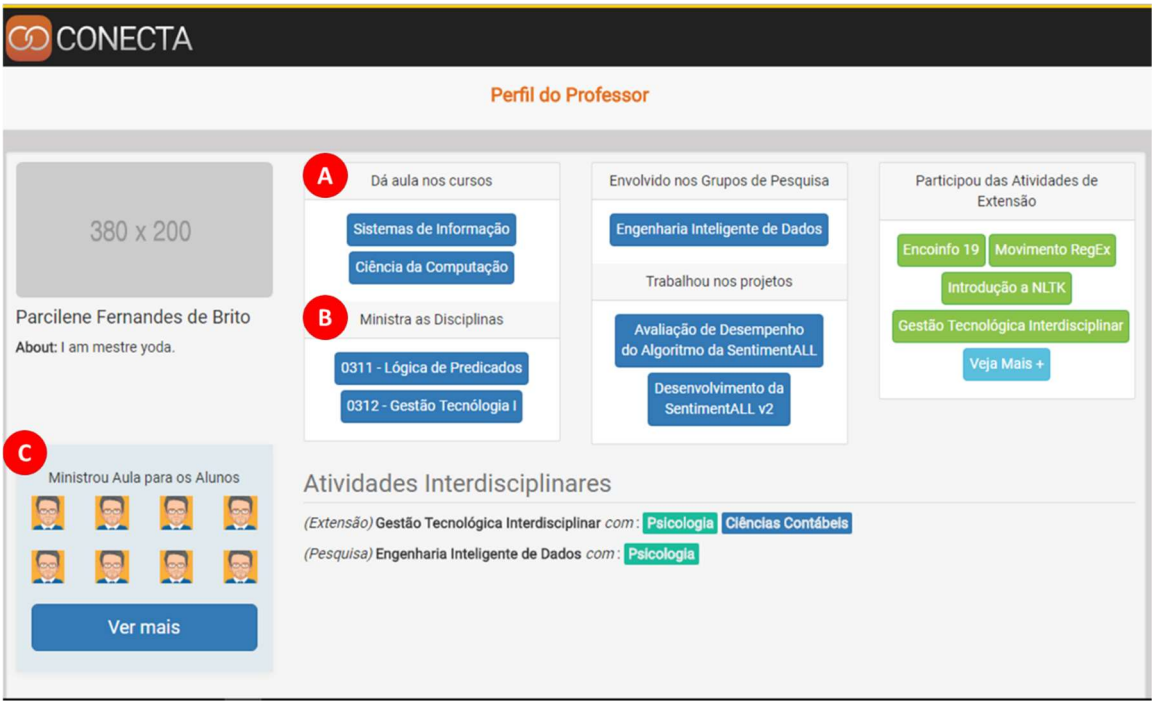
Figura 51 - Informações Exibidas na *Tooltip* de cada Elemento por Perfil

Elemento	Aluno	Professor
Curso	Área, Coordenador, Turno, Carga Horária	Área, Coordenador, Turno, Carga Horária
Turma	Professor, Horário, Sala	Horário, Sala, Qtd. de Alunos
Grupo de Pesquisa	Professores, Outros Alunos, Cursos, Qtd. Projetos, Qtd. Linhas de Pesquisa	Professores, Alunos, Cursos, Qtd. Projetos, Qtd. Linhas de Pesquisa
Projeto de Pesquisa	Orientador, Ano, Linha de Pesquisa	Aluno, Ano, Linha de Pesquisa
Atividade de Extensão	Tipo, Cursos, Semestre, Carga Horária, Participantes	Tipo, Cursos, Semestre, Carga Horária, Participantes

A alteração nas informações apresentadas em alguns elementos foi realizada levando em consideração a relevância de cada uma delas para o perfil do usuário. Por exemplo, nas informações do elemento Turma exibidas para um Aluno, há o item “Professor”, no entanto, este item não está presente no conjunto de informações do mesmo elemento apresentado para um perfil de Professor, pois todas as turmas listadas para ele consideram o fato de que ele é o próprio professor da turma.

Devido às diferenças na forma de apresentação de algumas informações, foi desenvolvido um protótipo de tela diferente para o perfil de Professor, que é apresentado na Figura.

Figura 52 - Tela do Perfil de Professor



Na Figura 52, é possível identificar alterações nas seções de curso (A), turmas (B) e alunos (C). Na tela do perfil de Professor, a seção de curso (A) apresenta os cursos em que o professor ministra alguma aula. A seção de turmas (B) apresenta as turmas que ele ministra no semestre atual. Enquanto a seção de alunos (C) apresenta a relação de alunos que se matricularam em alguma das turmas que ele já ministrou.

Dessa forma, estes protótipos de telas têm como objetivo apresentar para os usuários os resultados obtidos com realização dos processos desenvolvidos neste trabalho. A apresentação de um conjunto de informações reunidas em uma única tela possibilita que o usuário tenha uma visão mais ampla das relações que ele possui no contexto da instituição de ensino. Por exemplo, a participação em atividades interdisciplinares, a relação de alunos envolvidos no mesmo projeto de pesquisa e os cursos envolvidos nas atividades de extensão em que ele participou.

5 CONSIDERAÇÕES FINAIS

A utilização de ontologias e dos conceitos de Web Semântica como forma de representação do conhecimento tem como um de seus objetivos enriquecer a semântica de um dado contexto. Este enriquecimento é capaz de permitir a identificação de informações antes implícitas, que resultam do cruzamento de várias outras informações. Para verificar esta proposta, este trabalho objetivou a aplicação destas técnicas dentro do contexto de um portal acadêmico de uma instituição de ensino superior.

A primeira etapa para o desenvolvimento deste trabalho foi a criação de uma ontologia capaz de representar as relações existentes entre os perfis de Professor e Aluno dentro do contexto de uma instituição de ensino superior. A realização dessa etapa teve uma complexidade considerável por uma série de fatores. Primeiramente, porque a quantidade de relações existentes entre os elementos é grande e as especificidades delas variam bastante dependendo da fonte das informações. Por exemplo, foram identificadas algumas divergências ao buscar informações no site da CAPES (Coordenação de Aperfeiçoamento de Pessoa de Nível Superior), do MEC (Ministério da Educação) e na Coordenação de Pesquisa da própria instituição acerca de algumas relações no âmbito de Pesquisa, como a própria hierarquia das entidades existentes nesse contexto. A solução para essas questões foi o estabelecimento de um consenso em conjunto com a orientadora e especialista do domínio deste trabalho, Professora Parcilene Fernandes de Brito.

Outra dificuldade relacionada ao desenvolvimento da ontologia foi identificada na etapa de definição das características das propriedades das classes. Durante o estudo dos conceitos das características na literatura, foi encontrada uma baixa variedade de exemplos para serem utilizados como base. Para algumas características mais simples, como a *Transitive*, isso não representou grandes problemas, porém para características mais específicas dos conceitos de Web Semântica, como *Functional* e *Inverse Functional*, foi mais complicado encontrar uma aplicação para elas dentro do contexto.

No entanto, apesar das dificuldades encontradas, o desenvolvimento da ontologia foi concluído e apresentou resultados satisfatórios, que foram descritos na seção 4.2.2 Realização e verificação de inferências.

Com relação ao processo de recuperação dos dados da ontologia, a biblioteca RDFLib foi de grande auxílio. A disponibilização de métodos e estruturas que permitiram a manipulação dos dados por meio da sintaxe da linguagem Python permitiu que essa atividade fosse

realizada de forma mais simples. No entanto, a compreensão das linguagens OWL, RDF e SPARQL ainda se fez necessária, para que os resultados obtidos pelas consultas com a biblioteca RDFLib pudessem ser verificados.

Como trabalhos futuros, existem alguns pontos que podem ser explorados para aprimorar o processo desenvolvido, principalmente no que tange à automação de algumas tarefas. O primeiro ponto seria a estruturação dos resultados obtidos no processo de recuperação dos dados da ontologia em um arquivo JSON, para que as telas dos perfis fossem preenchidas com conteúdo de forma automática. Este arquivo JSON poderia ser organizado da seguinte forma: dois *arrays*, um representando uma lista de Professores e outro representando uma lista de Alunos, de forma que a identificação de cada indivíduo e seus atributos sejam seus respectivos nomes.

Outra atividade que poderia ser desenvolvida é a construção da ontologia de forma automática, por meio de um processo de mapeamento de um modelo de banco de dados relacional para um modelo ontológico, conforme alguns exemplos apresentados na seção 2.3 Trabalhos Correlatos. A implementação desse processo iria contribuir para uma redução considerável de esforço manual gasto nas tarefas relacionadas à criação da ontologia, assim podendo ser redirecionado para a identificação de formas de enriquecer semanticamente o domínio. Além disso, dependendo da forma como fosse implementado, poderia se tornar um mecanismo adaptável a diferentes contextos.

Outro ponto que poderia ser levado em consideração para o desenvolvimento de trabalhos futuros é a construção de uma ferramenta para a criação de instâncias da ontologia de forma automática, com base nos dados armazenados em um banco de dados relacional. Esta solução poderia ser melhor aproveitada se utilizada juntamente com a solução descrita no parágrafo anterior, pois ambas iriam compor uma ferramenta de transformação completa de um modelo relacional de banco de dados para um modelo ontológico, de forma totalmente automática. Além dessa transformação, também seria necessário utilizar um *Reasoner* para realizar as inferências no modelo ontológico. Para isso, poderia ser utilizado algum dos mecanismos de inferência conhecidos, como o HermiT ou Pellet, ambos suportados pela ferramenta Protégé. Entretanto, se necessário, também poderia ser implementado um mecanismo próprio, específico para o domínio em que a solução seria aplicada.

Além disso, outro ponto que poderia ser explorado é a forma de apresentação das informações resultantes dos processos descritos anteriormente. O estudo de técnicas de visualização da informação com o intuito de identificar alternativas mais eficientes poderia contribuir nesse sentido, assim como a utilização de técnicas de *User Experience*.

REFERÊNCIAS

ALALWAN, N., ZEDAN, H., SIEWE, F. **Generating OWL Ontology for Database Integration**. Software Technology Research Laboratory. De Montfort University. Leicester, United Kingdom. Third International Conference on Advances in Semantic Processing. 2009.

ALLEMANG, D., HENDLER, J. **Semantic Web for the Working Ontologist**: Modeling in RDFS and OWL. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 2011.

BECHHOFER, S. et al. **OWL Web Ontology Language Reference**. Technical report. Web Ontology Working Group. World Wide Web Consortium – W3C. 2004. Disponível em: <https://www.w3.org/TR/owl-ref/>. Acesso em: maio de 2017.

BMIR. **Protégé** "Protégé Presentation". Stanford Center for Biomedical Informatics Research. Stanford University. 2007. Disponível em: <http://protege.stanford.edu/> Acesso em: maio de 2017.

BORST, P. **Construction of Engineering Ontologies for Knowledge Sharing and Reuse**. PhD thesis, Tweente University, 1997.

CAMPBELL, L., M., MACNEILL, S. **The Semantic Web, Linked and Open Data**. Centre for Educational Technology & Interoperability Standards - JISC CETIS. Creative Commons Attribution 3.0 License. 2010.

DADJOO, M., KHEIRKHAH, E. **An Approach for Transforming of Relational Databases to OWL Ontology**. International Journal of Web & Semantic Technology (IJWesT). Vol 6, No 1. 2015.

DAN, C. et al. **DAML+OIL Reference**. World Wide Web Consortium. 2001. Disponível em: <https://www.w3.org/TR/daml+oil-reference>. Acesso em: maio de 2017.

DECKER, S. et al. **The Semantic Web:** The Roles of XML and RDF. Institute of Eletrical and Eletronics Engineers (IEEE) – Internet Computing. 2000.

DING, L. et al. **How the Semantic Web is being used:** An analysis of FOAF documents. University of Maryland, Baltimore County. Baltimore MD USA. 2005.

DING, L. **Search on the Semantic Web**. University of Maryland, Baltimore County. Baltimore MD USA. 2005.

FEIGENBAUM, L. **SPARQL by Example: The Cheat Sheet**. VP Technology & Standards. Cambridge Semantics. W3C SPARQL Working Group. 2013. Disponível em: <http://www.iro.umontreal.ca/~lapalme/ift6281/sparql-1_1-cheat-sheet.pdf>. Acesso em: junho de 2017.

GAVRILOVA, T., GOROVOY, V., PETRASHEN, E. **Ontology-Based Conceptual Domain Modeling for Educational Portal**. Saint-Petesburg State University. Volkhovsky. St. Petesburg, Russia. Ninth IEEE International Conference on Advanced Learning Technologies. 2009.

GRUBER, T. R. **Toward principles for the design of ontologies used for knowledge sharing**. International Journal of Human-Computer Studies, 43(5-6):907--928, 1995.

HORRIDGE, M. et al. **A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools**. University of Manchester. Edition 1.3. 2011.

LINCKELS, S. **Semantic Web – OWL**. Bachelor in Informatics. Faculty of Science, Technology and Communication (FSTC). University of Luxembourg. 2014.

MILLER, E. **An Introduction to the Resource Description Framework**. Online Computer Library Center, Inc. D-Lib Magazine. ISSN 1082-9873. 1998.

NAVIGLI, R. e VELARDI, P. **Learning Domain Ontologies from Document Warehouses and Dedicated Websites**. Computational Linguistics. Volume 30. Issue 2. Publisher: MIT Press. 2004.

NOY, N. F., MCGUINNESS, D. L. **Ontology Development 101:** A Guide to Creating Your First Ontology. Stanford University. Stanford, California. 2001.

ROSSUM, G. V. **Python 3000 and You**. All Things Pythonic. Python Community. 2008. Disponível em: < https://www.artima.com/weblogs/viewpost.jsp?thread=227041> Acesso em: Setembro de 2017.

TELNAROVA, Z. **Relational Database as a Source of Ontology Creation**. University of Ostrava. Czech Republic. Proceedings of the International Multiconference on Computer Science and Information Technology. p 135-139. 2010.

VIEIRA, E. R. **Estudo Sobre Nomenclatura na Navegação de Portais Universitários Brasileiros**. 11º Congresso Brasileiro de Pesquisa e Desenvolvimento em Design. Nº 4, Volume 1. 2014.

W3C. **OWL** "OWL Web Ontology Language Reference". 2004. Disponível em: <http://www.w3.org/TR/owl-ref/>. Acesso em: maio de 2017.

W3C. **OWL** "OWL Web Ontology Language Guide". 2004. Disponível em: <https://www.w3.org/TR/owl-ref/#ref-owl-guide>. Acesso em: junho de 2017.

W3C. **RDF Vocabulary Description Language 1.0:** RDF Schema. Technical Report. RDF Core Working Group. World Wide Web Consortium – W3C. 2004. Disponível em: < https://www.w3.org/TR/2004/REC-rdf-schema-20040210/>. Acesso em: maio de 2017.

W3C. **Semantic Web** "W3C Semantic Web Activity". 2013. Disponível em: <https://www.w3.org/2001/sw/>. Acesso em: abril de 2017.

W3C. **SPARQL** "SPARQL 1.1 Overview". 2013. Disponível em: <https://www.w3.org/TR/sparql11-overview/>. Acesso em: maio de 2017.

W3C. **SPARQL** "SPARQL Query Language for RDF". 2008. Disponível em: <https://www.w3.org/TR/rdf-sparql-query/>. Acesso em: junho de 2017.

W3C. **HTML** “HTML 5”. 2017. Disponível em: < https://www.w3.org/html/>. Acesso em: setembro de 2017.

W3C. **CSS** “Página inicial do Cascading Style Sheets”. 2017. Disponível em: < https://www.w3.org/css/>. Acesso em: setembro de 2017.

W3SCHOOLS. **Tutorial CSS**. 2017. Disponível em: <https://www.w3schools.com/css/>. Acesso em: setembro de 2017.

W3SCHOOLS. **jQuery Tutorial**. 2017. Disponível em: <https://www.w3schools.com/jquery/>. Acesso em: setembro de 2017.

YAHIA, N., MOKHTAR, S., A. AHMED, A. **Automatic Generation of OWL Ontology from XML Data Source**. Computers and Systems Department, Eletronic Research Institute. Cairo, Egypt. International Journal of Computer Science Issues, Vol. 9, Issue 2, No 2. 2012.

ZHANG, J. **Ontology and the Semantic Web**. Proceedings of the North American Symposium on Knowledge Organization. Vol 1. 2007.

APÊNDICES

ANEXOS