Danillo Ricardo Celino

FrameWeb-LD: Uma abordagem baseada em ontologias para a Integração de Sistemas de Informação Web e a Web Semântica

Vitória, ES

2017

Danillo Ricardo Celino

FrameWeb-LD: Uma abordagem baseada em ontologias para a Integração de Sistemas de Informação Web e a Web Semântica

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2017

Danillo Ricardo Celino

FrameWeb-LD: Uma abordagem baseada em ontologias para a Integração de Sistemas de Informação Web e a Web Semântica/ Danillo Ricardo Celino. – Vitória, ES, 2017-

98 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Dissertação de Mestrado – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Programa de Pós-Graduação em Informática, 2017.

1. Web Semântica. 2. Ontologias. 3. Engenharia Web, frameworks, web semântica, dados ligados, ontologias. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. FrameWeb-LD: Uma abordagem baseada em ontologias para a Integração de Sistemas de Informação Web e a Web Semântica

CDU 02:141:005.7

Danillo Ricardo Celino

FrameWeb-LD: Uma abordagem baseada em ontologias para a Integração de Sistemas de Informação Web e a Web Semântica

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Trabalho aprovado. Vitória, ES, 20 de novembro de 2017:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. João Paulo A. Almeida, PhD
Universidade Federal do Espírito Santo

Prof. Frank Augusto Siqueira, PhD
Universidade Federal de Santa Catarina

Vitória, ES
2017

A Deus, meu Pai, por me abençoar em tudo e principalmente em minhas limitações. Obrigado Senhor, por me ouvir, sempre estar comigo e sacrificar seu Filho, Jesus Cristo, único Intercessor e Salvador.

Agradecimentos

Agradeço a Deus por ter me dado coragem para conquistar mais essa vitória. Aos meus pais, Neilton e Ercília, pelo grande apoio, sacrifício, dedicação ... essa vitória é tão de vocês quanto minha! Aos meus irmãos, Daniel e Darlan, por sempre me alegrarem e nunca me deixarem desistir. Ao mestre, orientador e, acima de tudo, amigo Vitor Souza, por me guiar nessa jornada e pelo enorme incentivo. Um agradecimento especial Beatriz Martins e Bruno Borloni por me ajudarem nesse projeto. Ao meus amigos da UFES ao pessoal do NEMO, pelos momentos de correria, e por me tirarem tantas dúvidas.

Ao Programa de Pós-Graduação em Informática (PPGI) da Universidade Federal do Espírito Santo (UFES) pela estrutura acadêmica durante o desenvolvimento do trabalho. Aos Professores do PPGI/UFES por tudo que aprendi nas disciplinas do programa. Oportunidade verdadeira para ampliar meus conhecimentos e crescer profissionalmente.

Gostaria de dividir toda a alegria com vocês. Nunca teria chegado até aqui se todos vocês não estivessem presentes em minha vida!

"Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes". (Marthin L. King)

Resumo

Com a enorme quantidade de dados disponível na Web, tecnologias de Dados Ligados vem sendo propostas para tentar alcançar a visão da Web Semântica, permitindo o acesso, descoberta e combinação eficiente dos dados disponíveis. Tais dados devem ser publicados em forma de dados estruturados e vinculados a vocabulários conhecidos, para que possam ser entendidos por agentes de *software*. Além disso, os modelos conceituais abstratos por trás desses dados, ou seja, suas ontologias, também podem ter uma grande influência na adoção de conjuntos de dados vinculados e seus vocabulários.

Em 2007, foi proposto um método de Engenharia Web para o projeto e desenvolvimento de aplicações Web baseadas em *frameworks*, chamado ***FrameWeb***, e a sua posterior integração com a Web Semântica por meio de uma extensão chamada ***S-FrameWeb***. Dados os resultados produzidos na literatura nesta área, como ontologias bem fundamentadas, e os últimos avanços das tecnologias de Dados Ligados, propomos como evolução do ***S-FrameWeb*** o ***FrameWeb-LD***, uma abordagem para a integração de Sistemas de Informação Web com a Web Semântica. Nossa proposta usa linguagens e métodos bem fundamentados para a modelagem de ontologias e auxilia desenvolvedores na publicação de dados e serviços da sua aplicação na Web de Dados, por meio de um processo sistemático que traz para os modelos de projeto a integração dos dados do sistema com vocabulários da Web Semântica e de uma ferramenta que gera automaticamente boa parte do código-fonte relativo à publicação de Dados Ligados.

Palavras-chaves: Engenharia Web; Frameworks; Web Semântica; Dados Ligados; Ontologias.

Abstract

With the enormous amount of data available on the Web, Linked Data technologies have been proposed to try and achieve the vision of the Semantic Web, allowing the efficient access, discovery and combination of the available data. Such data should be published in a structured way and bound to known vocabularies, so they can be understood by software agents. Moreover, the abstract conceptual models behind this data, i.e., their ontologies, can also have a great influence in the adoption of a Linked Data set and its vocabularies.

In 2007, a Web Engineering method for the design and development of Web applications based on frameworks, named ***FrameWeb***, was proposed, along with an extension of the method, called ***S-FrameWeb***, that proposed the subsequent integration of the application's data with the Semantic Web. Given the advances of the literature in this area of research, such as well-founded ontologies and the evolution of Linked Data technologies, we propose an evolution of ***S-FrameWeb*** called ***FrameWeb-LD***, an approach for the integration of Web-based Information Systems with the Semantic Web. Our proposal uses well-founded languages and methods for the construction of ontologies and aids developers in publishing the application's data and services in the Web of Data, by offering a systematic process that brings to architectural design models how the data from the system is integrated with Semantic Web vocabularies and a tool that generates automatically most of the source code related to Linked Data publishing.

Keywords: Web Engineering; Frameworks; Semantic Web; Linked Data; Ontologies.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DAO	Data Access Object
IRI	Internationalized Resource Identifier
Java EE	Java Enterprise Edition
MDD	Model-Driven Development
MOF	Meta Object Facility
OntoUML	Ontology UML
ORM	Object/Relational Mapping
OWL	Web Ontology Language
OWL-S	OWL Services
RDF	Resource Description Framework
RDFS	RDF Schema
SOAP	Simple Object Access Protocol
UDDI	Universal Description, Discovery and Integration
UFO	Unified Foundational Ontology
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WebE	Web Engineering
WIS	Web Information System
WSDL	Web Services Description Language
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

Sumário

1 Introdução

Esse capítulo apresenta o contexto, as motivações, a metodologia e a organização estrutural adotada no desenvolvimento desse trabalho.

1.1 Contexto

Atualmente, é notório e considerável o aumento no volume de dados disponíveis na Web, como informações geográficas, resultados de pesquisas científicas, dados governamentais, dentre outros. Muitas organizações e até indivíduos decidiram compartilhar dados abertamente por meio da Web. Então surge a demanda por eficiência no acesso aos dados que permita sua fácil reutilização, descoberta facilitada de dados relevantes dentre a gama de fontes de dados e a integração de dados provenientes de diversas fontes (HEATH; BIZER, 2011).

A visão da Web Semântica, descrita pela primeira vez por Berners-Lee, Hendler e Lassila (2001) em seu artigo seminal, "*The Semantic Web*", propôs aproveitar a arquitetura da Web para vincular dados, ou seja, ligar dados, adicionando semântica a esses links. De acordo com os autores, disponibilizando dados na Web em um formato processável por máquina permitiria a criação de agentes de software que poderiam nos ajudar em tarefas repetitivas, impraticáveis ou mesmo impossíveis de realizar hoje em dia. Essas tarefas são prejudicadas pela quantidade de dados disponíveis na Internet, sem descrição semântica, criando problemas para fornecer acesso aos dados para que possam ser mais facilmente reutilizados, para permitir a descoberta de dados relevantes dentro da multidão de conjuntos de dados, ou até para integrar dados de diferentes fontes de dados anteriormente desconhecidas (HEATH; BIZER, 2011). Por isso, para que alcancemos esses objetivos, Berners-Lee (2006) sugere que primeiramente publiquemos todos esses dados na forma de dados ligados (*Linked Data*).

Neste contexto, a Web não é a única fonte e inspiração para as tecnologias que suportam a Web Semântica. Ressalta-se a construção de modelos abstratos (Modelagem Conceitual e Engenharia de Ontologias), a computação com conhecimento (Lógica e Inteligência Artificial) também são envolvidos no processo de construção da Web de Dados (*Web of Data*) (HITZLER; KRöTZSCH; RUDOLPH, 2009).

As ontologias em especial têm um papel fundamental no desenvolvimento da Web Semântica. Definindo um significado comum aos dados publicados em várias fontes de dados, facilitando assim a integração. No estado-da-arte, existem diversas pesquisas sobre linguagens e métodos para ontologias. Dentre elas, destacam-se a ontologia de fundamen-

tação denominada UFO e a linguagem de modelagem conceitual OntoUML (GUIZZARDI, 2005), assim como métodos sistemáticos para construção de ontologias, destacando SABiO (FALBO, 2014).

Desde do nascimento da Engenharia Web (WebE), muitos métodos foram criados para a construção de Sistemas de Informação Web (*Web-based Information Systems* ou WIS), como por exemplo OOWS (*Object Oriented Web Solution*) (PASTOR; FONS; PELECHANO, 2003), UWE (*UML-based Web Engineering*) (KOCH; KRAUS, 2002), dentre outros. Entretanto, poucos realizam a integração com a Web de Dados. O método ***FrameWeb*** (SOUZA; FALBO, 2007) e a sua extensão ***S-FrameWeb*** (SOUZA et al., 2007) para criação de WISs semânticos, auxiliando desenvolvedores que utilizam o método para o desenvolvimento de WIS a integrá-los mais facilmente à Web de Dados. No entanto, tal método não considera diversos avanços nas áreas de modelagem conceitual e *Linked Data*. Por isso, é importante a busca e atualização de métodos, com linguagens e ferramentas mais atuais para integração com a Web de Dados.

1.2 Motivação

Apesar da Web Semântica e Web de Dados serem amplamente discutidas na literatura científica e aplicadas em outros tantos meios, existem poucos métodos que relacionam construção de Sistemas de Informação Web (*Web Information System - WIS*) com a Web de Dados. Neste contexto, identificamos as seguintes oportunidades de pesquisa:

- Não existem na literatura arquiteturas ou métodos para desenvolvimento de WIS baseados em ontologias bem fundamentadas;
- Poucos métodos integram WIS com a Web de Dados com uso de ferramentas que automatizam partes do processo;
- Existem diversas abordagens para a criação de *Web Services* Semânticos, mas poucas delas são integradas a métodos de desenvolvimentos de WISs.

Nesse contexto, foi realizada uma revisão da literatura sobre Dados Ligados, métodos de desenvolvimentos de WIS integrados a Web Semântica e ontologias de fundamentação para descrição de dados. Feito isso, propusemos um método que vai ao encontro dos desafios elencados acima, o qual denominamos ***FrameWeb-LD***, por ser baseado no método ***FrameWeb***, porém com foco em *Linked Data*. O método propõe ser uma evolução de ***S-FrameWeb***, proposto por Souza et al. (2007) como uma extensão de ***FrameWeb*** para a Web Semântica.

1.3 Objetivos

O objetivo geral deste trabalho é propor uma evolução do método ***S-FrameWeb*** (SOUZA et al., 2007) de modo que o WIS seja baseado em ontologias bem fundamentadas, atualizá-lo em relação aos últimos avanços das tecnologias ligadas à Web de Dados e incluir procedimentos para o desenvolvimento de *Web Services* Semânticos. Tal objetivo pode ser desmembrado nas seguintes metas:

- Propor um processo sistemático para o desenvolvimento WISs de modo a guiar o desenvolvedor no uso dos modelos propostos;
- Utilizar OntoUML e ferramentas para construção de ontologias, integrando-as ao processo sistemático para o desenvolvimento de WISs preparados para a Web Semântica;
- Desenvolver ferramentas que auxiliem o desenvolvedor a conduzir o processo sistemático com vistas à integração do WIS com a Web de Dados;
- Realizar experimentos que validem as propostas, demonstrando sua utilidade e viabilidade.

Este trabalho é uma pequena contribuição em relação a própria visão da Web Semântica. Podemos, no entanto, nos beneficiar do conceito de dados ligados, mesmo que essa visão nunca seja alcançada.

1.4 Metodologia

Esse trabalho foi conduzido da seguinte forma:

- **Revisão da Literatura**: foi realizada uma revisão da literatura, buscando artigos relacionados ao tema para conhecer o estado da arte sobre Web Semântica, Web de Dados e métodos sistemáticos de construção de WIS nesse contexto. Realizamos um estudo aprofundado do método ***S-FrameWeb***. Foram realizadas buscas por ferramentas de automação de atividades relacionadas a publicação e consumo de dados. Também foi realizada a leitura do livro *Linked Data: Evolving the Web into a Global Data Space* (HEATH; BIZER, 2011).
- **Evolução do método *S-FrameWeb***: após a revisão da literatura, elaboramos a proposta de evolução de ***S-FrameWeb***, denominada ***FrameWeb-LD***, integrando atividades e tecnologias avançadas para ligação de dados, propondo também uso de *Web services* semânticos, baseando o método na ontologia bem fundamentada UFO

e na linguagem OntoUML, bem como utilizando ferramentas que produzam (semi-)automaticamente artefatos de código baseados nos modelos construídos ao longo do processo. Tais ferramentas usam transformação de banco de dados amplamente utilizados no mercado para o formato da Web Semântica, facilitando a tarefa de integração com *Linked Data*.

- **Avaliação do método**: avaliamos o método, seus modelos, suas ferramentas e atividades a partir de projetos efetuados por alunos de pós-graduação com o método criado, a partir dos modelos por eles produzidos, utilizando as ferramentas construídas.

- **Escrita da Dissertação**: os resultados obtidos nas etapas anteriores foram descritos nessa dissertação de mestrado. O trabalho realizado para essa dissertação também deu origem à publicação (CELINO et al., 2016), na qual a evolução do método ***FrameWeb*** é apresentada.

1.5 Organização da Dissertação

Nesse primeiro capítulo da dissertação foram apresentadas as ideias gerais deste trabalho, foram descritos o contexto de aplicação, os objetivos e a metodologia adotada. Além dessa introdução este documento organiza-se da seguinte forma:

- **Capítulo 2 - Ontologia**: descreve os principais conceitos relacionados à ontologia de fundamentação UFO (GUIZZARDI, 2005) e ao método de engenharia de ontologias SABiO (FALBO, 2014).

- **Capítulo 3 - Web Semântica e FrameWeb**: apresenta e discute as principais metodologias e o estado-da-arte sobre Web Semântica, Dados Ligados. São descritos também o método ***FrameWeb*** (SOUZA; FALBO, 2007) e ***S-FrameWeb*** (SOUZA et al., 2007) para evolução do mesmo. Na segunda parte, são apresentados também os métodos encontrados na revisão da literatura realizada para comparação com trabalhos relacionados.

- **Capítulo 4 - FrameWeb-LD**: detalha todo o processo do método ***FrameWeb-LD***, bem como a utilização da linguagem OntoUML e ferramentas para modelagem conceitual, a adição de atividades de projeto para dados ligados e *Web Services* semânticos e a geração de ferramentas para auxiliar desenvolvedores na ligação de dados.

- **Capítulo 5 - Avaliação do Método**: descreve o processo de avaliação dos modelos e ferramentas geradas no processo em WIS experimentais.

- **Capítulo 6 - Conclusões**: descreve as conclusões obtidas na realização desse trabalho. As principais contribuições e possíveis trabalhos futuros e pontos para a continuação dessa pesquisa são apresentados.

2 Ontologias

Nesse capítulo serão apresentados os conceitos principais sobre ontologia para o embasamento teórico para a criação e desenvolvimento da dissertação. É dividido em duas seções: a Seção 2.1 apresenta UFO, uma ontologia de fundamentação usada como base para construção de outras ontologias; e a Seção 2.2 apresenta o método SABiO, abordagem utilizada para a construção de ontologias neste trabalho.

Gruber (1995) define uma ontologia como uma especificação de uma conceituação, ou seja, uma descrição de conceitos e relações de um determinado domínio e suas definições, propriedades e restrições na forma de axiomas. Borst (1997) entende ontologias como uma especificação formal de uma conceituação compartilhada, sendo assim úteis para apoiar a especificação e implementação de qualquer sistema de informação complexo. Nas sub-áreas da Ciência da Computação, como Engenharia de Software ou Web Semântica, foco desse trabalho, ontologias são artefatos computacionais, representam conhecimento e modelos conceitual de domínio. A Web Semântica depende fortemente de ontologias formais para estruturar dados para compreensão abrangente de agentes de software (MAEDCHE; STAAB, 2001).

Guarino et al. (1998) classifica ontologias de acordo com sua especificidade, conforme ilustra a Figura 1:

- **Ontologias de Fundamentação (ou de alto nível)**: descrevem conceitos gerais

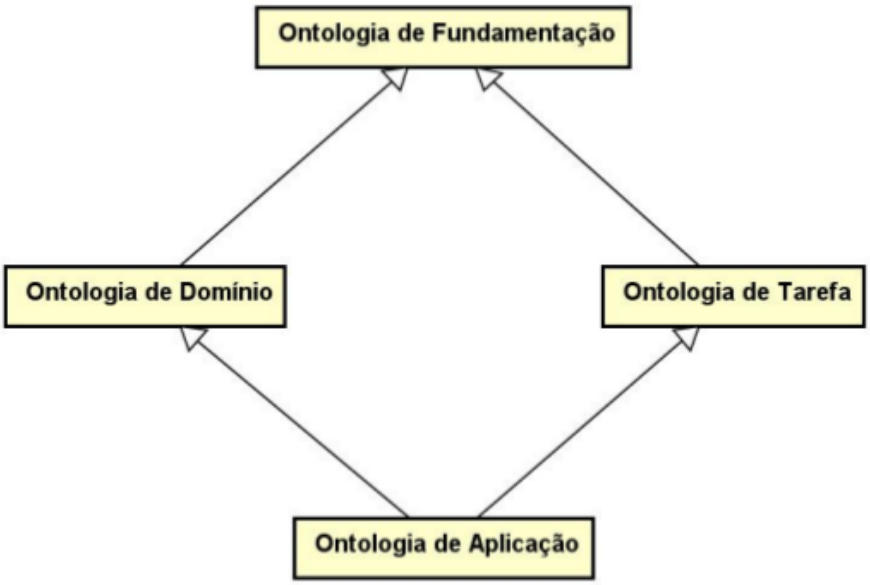

Figura 1 – Relação entre os tipos de ontologias (GUARINO et al., 1998)

como objetos, espaço, tempo, eventos e entre outros, o que as torna independentes de um domínio ou problema.

- **Ontologias de Domínio**: são ontologias que descrevem um vocabulário para um determinado domínio, como, por exemplo, Engenharia de Requisitos de Software ou Odontologia, e utilizam conceitos especializados de conceitos presentes em ontologias de fundamentação.
- **Ontologias de Tarefa**: descrevem tarefas ou atividades genéricas, por exemplo, compra e venda. Seus conceitos são especializados de conceitos em ontologias de fundamentação.
- **Ontologias de Aplicação**: descrevem conceitos que dependem de domínios e tarefas específicas, sendo assim especializados de duas ontologias relacionadas.

Outra classificação importante neste contexto é a que divide ontologias entre modelos de referência e modelos operacionais:

- **Ontologias de Referência** se referem a um tipo de modelo conceitual que representa um domínio formal, especializando conceitos de uma ontologia de fundamentação, sendo também um vocabulário não ambíguo de um determinado domínio (FALBO; BERTOLLO, 2009).
- **Ontologias Operacionais** são versões implementadas de ontologias de referência capazes de serem interpretadas e utilizadas por agentes de software. Não são focadas na adequação da representação, mas sim em garantir propriedades computacionais (FALBO, 2014).

O foco desse trabalho é na construção de ontologias de referência e operacionais quanto a artefatos. Quanto a função, o foco é nas ontologias de aplicação e domínio, segundo a classificação de Guarino et al. (1998).

2.1 UFO

UFO (*Unified Foundational Ontology*) é uma ontologia de fundamentação que provê um sistema de categorias básicas e relações que vem sendo desenvolvida baseada em um número de teorias das áreas de Ontologia Formal, Lógica Filosófica, Linguística, Filosofia de Linguagem e Psicologia Cognitiva (GUIZZARDI, 2005).

UFO é formada por três partes:

- **UFO-A** ontologia de objetos (*endurants*) analisa objetos em aspectos estruturais, tipos, partes e seus papéis.

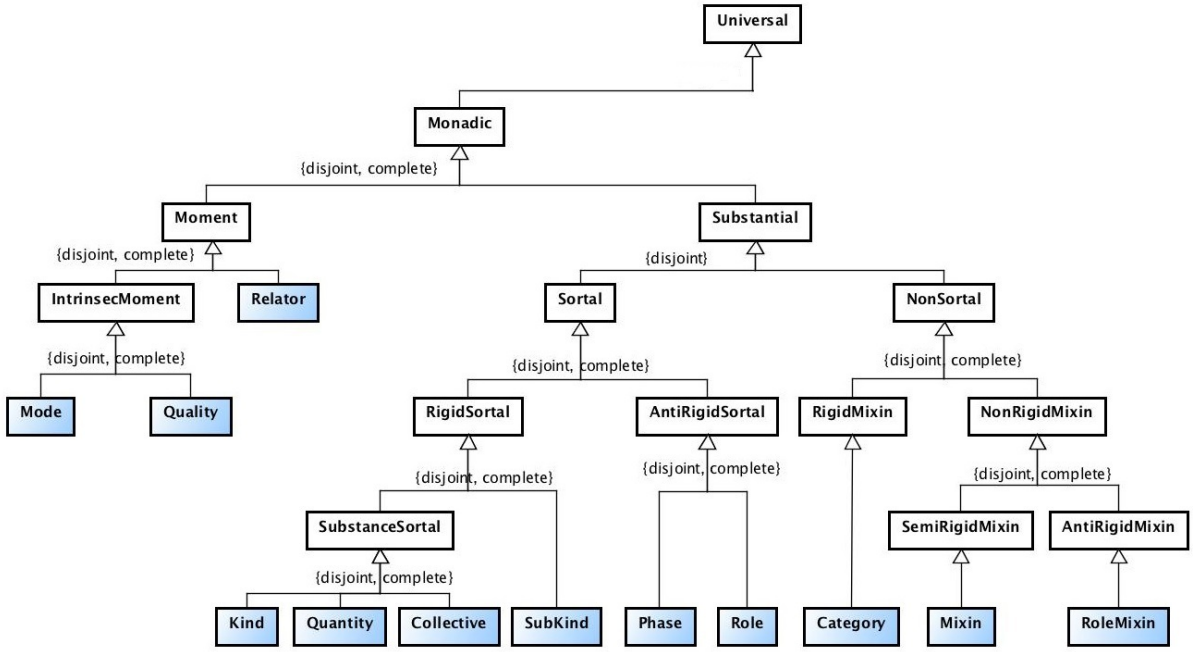

Figura 2 – Especializações de *Monadics* do fragmento da UFO-A

- **UFO-B** ontologia de eventos (*perdurants*) tratando de eventos, relacionamentos entre eventos, a participação de objetos e as propriedades temporais das entidades.

- **UFO-C** uma ontologia de entidades sociais, representando agentes, compromissos e intenções. UFO-C é construída a partir de UFO-A e UFO-B, ou seja, especializa entidades dessas duas outras partes de UFO.

Importante destacar que o foco desse projeto é em UFO-A. UFO faz uma distinção formal entre indivíduos e universais. Indivíduos (*Individuals*) são entidades que possuem uma identidade única, enquanto Universais (*Universals*) são padrões abstratos de características que podem ser encontradas em um número de diferentes indivíduos. Universais se especializam em *Monadics*, que representam Universais que não são responsáveis por unir outras entidades, e *Relations*, que são responsáveis por unir entidades.

A Figura 2 apresenta as especializações de *Monadic*, que especializam em Substanciais (*Substantials*) e Momentos (*Moments*). Substanciais são indivíduos existencialmente independentes (uma pessoa, um carro). Momentos, por sua vez, são indivíduos inerentes a outros indivíduos, sendo deles dependentes (a dor de cabeça de uma pessoa, a cor de um carro), podem ser *Intrinsic Moments* sendo dependentes de um único indivíduo, ou *Relators* de vários indivíduos.

Substantial tem dois tipos: *Sortal* e *Non-Sortal*. Vemos na Figura 2 que *Sortal* é especialização de *Monadic* e que tem ou herda um princípio de identidade. *Rigid Sortal*, são *Sortals* cujas características são aplicadas a todas as suas instâncias. Já *AntiRigid Sortal* são *Sortals* cujos padrões característicos não se aplicam a todas as suas instâncias.

Tabela 1 – Tabela sobre entidades UFO-A *Substantial Sortals* (FONSECA, 2015)

Entidade	**Descrição**
Kind	*Kind* são *Substance Sortals* cujas instâncias são complexos funcionais (*functional complexes*), que incluem instâncias naturais (como Pessoa, Cachorro, Árvore) e de artefatos (Cadeira, Carro, Televisão).
SubKind	Um *SubKind* é um tipo relacionalmente independente de um *Substance Sortal* que carrega o princípio de identidade herdado , como um Homem pode ser *SubKind* de Pessoa.
Role	Papéis (*Role*) representam especializações de Sortais que têm como condição de especialização uma relação.
Phase	Fases (*Phase*) representam, *AntiRigid Sortal*, no qual há sempre um único supertipo comum que é necessariamente um *Sortal*, como fases da vida (criança, adolescente, jovem, etc).
Collective	*Collective* são *Substance Sortals* cujas instâncias são coletivos, ou seja, coleções de complexos que têm estrutura uniforme, como Grupo de Pessoas.
Quantity	*Quantity* são *Substance Sortals* cujas instâncias são quantidades, como Ouro ou Areia.

Rigid Sortal tem um tipo específico, o *Substance Sortal*, corresponde a objetos rígidos, relacionalmente independentes e que proveem um princípio de identidade.

Destacamos também *Non-Sortal*, que não são capazes de prover uma identidade a suas instâncias, tendo dois tipos *Rigid Mixins*, que agregam propriedades essenciais e comuns a diferentes *Sortals*, e suas características são aplicadas a todas as suas instâncias; e *Non-Rigid Mixin*, subdivididos em de dois tipos: *Anti-Rigid Mixins*, que nem todas as características da agregação estão em suas instâncias, representando papéis que podem ser desempenhados por tipos diferentes e disjuntos entre si. *SemiRigid Mixins*, que são propriedades essenciais para algumas instâncias e acidentais a outras, funcionando como tipos *Rigid Mixin* e para outros, como *Anti-Rigid Mixins*.

Os tipos “folhas” da hierarquia mostrada na Figura 2, são apresentados em tabelas, originalmente apresentadas em (FONSECA, 2015). A Tabela 1 mostra entidades sobre *Substantials Sortals*; a Tabela 2 sobre *Substancials Non-Sortals* e a Tabela 3 mostra entidades de *Moments*.

UFO também apresenta relacionamentos entre entidades (*Relation*), que possui três tipos: *Material Relations*, que possuem uma estrutura material própria e que são derivadas de *Relators*; *Formal Relation*, que são relações entre entidades sem intervenção de outro indivíduo, podendo ser definidas por propriedades intrínsecas dos indivíduos; e *Meronymic*, que representa relação de paridade entre indivíduos. A Figura 3 apresenta os tipos de relações e a Tabela 4 detalha instâncias dos tipos citados acima.

Usando um fragmento do metamodelo UML 2.0, Guizzardi (2005) criou OntoUML, uma linguagem ontologicamente bem-fundamentada baseada em UFO. Nesse projeto

Tabela 2 – Tabela sobre entidades UFO-A *Substantial Non-Sortals* (FONSECA, 2015)

Entidade	Descrição
Category	Um *Category* agrega propriedades essenciais que são comuns a diferentes *Substance Sortals*. Por exemplo, uma Entidade Racional como categoria que generaliza Pessoa e Agente Inteligente.
RoleMixin	Um *RoleMixin* representa uma agregação de propriedades que são comuns a diferentes papéis (*Roles*), por exemplo *RoleMixin* Cliente que incluí *Roles* Cliente Privado e Cliente Corporativo.
Mixin	Um *Mixin* representa propriedades essenciais para algumas das instâncias e acidentais para outros (*SemiRigid*). Um exemplo é o *Mixin* Sentável, que representa uma propriedade que pode ser considerada essencial para uma cadeira, mas acidental para uma pedra.

Tabela 3 – Tabela sobre entidades UFO-A *Moments* (FONSECA, 2015)

Entidade	Descrição
Relator	*Relator* é uma entidade dependente de pelo menos duas entidades distintas, ou seja, instanciação de propriedades relacionais, como casamentos, beijos, apertos de mão, entre outras.
Mode	*Mode* é existencialmente dependente de exatamente uma entidade. Exemplos incluem habilidades, pensamentos, crenças.
Quality	*Quality* é definido como uma instância com o poder qualificar uma entidade, como um valor em um espaço de qualidade geométrica.

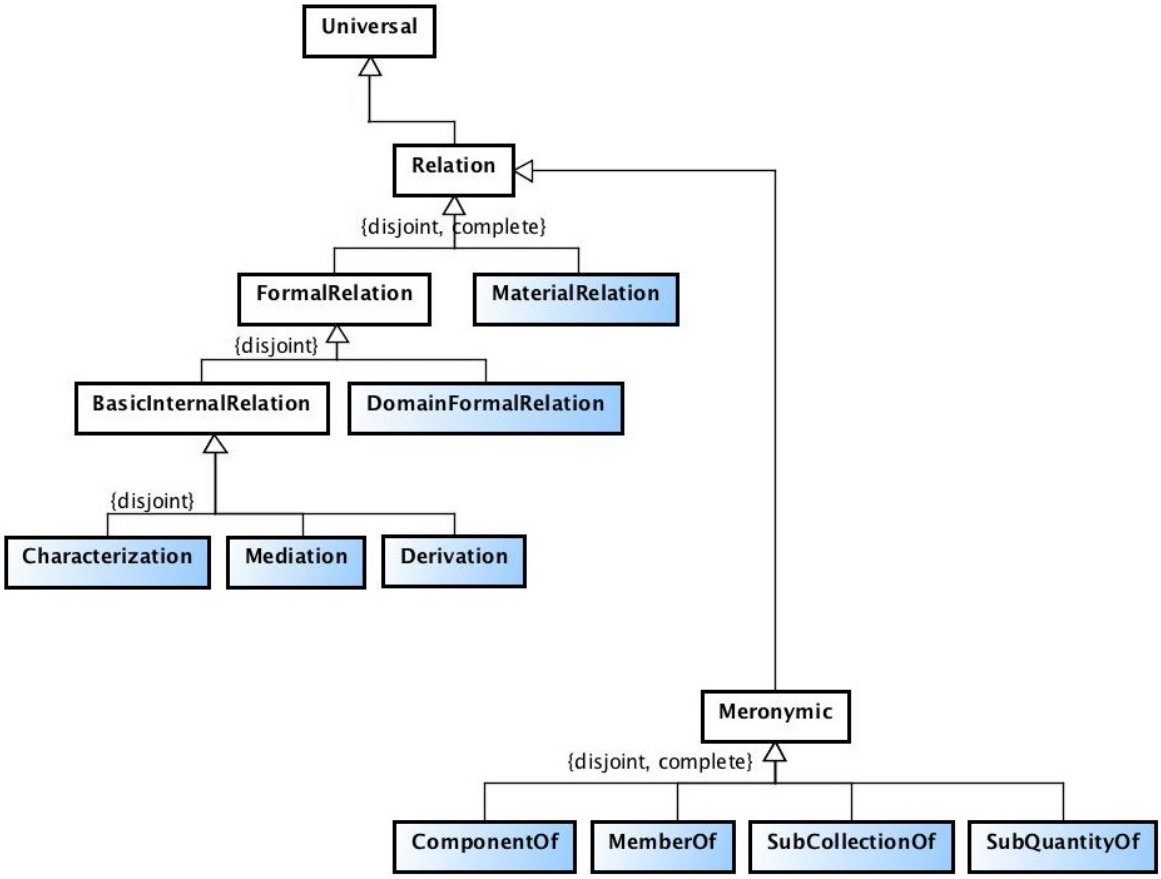

Figura 3 – Especializações de *Relations* do fragmento da UFO-A.

Tabela 4 – Tabela sobre entidades UFO-A *Relation*.

Entidade	Descrição
Mediation	*Mediation* é uma relação formal que ocorre entre *Relators* e *Endurants* a qual media. Por exemplo, o casamento media o papel de Marido e Esposa.
Characterization	*Characterization* é uma relação formal que ocorre entre um *Mode* e um *Endurant* que este *Mode* em questão caracteriza. Por exemplo, o Objetivo e Capacidade Privada caracteriza um Agente.
Derivation	*Derivation* é relação formal entre uma *MaterialRelation* e o *Relator* que ela deriva.
MaterialRelation	Uma associação *Material* são relações que tem estrutura material por conta própria, isto é, uma relação que é induzida por um *Relator*. Exemplos incluem estudantes estudam na Universidade, o paciente é tratado em unidade médica.
DomainFormalRelation	Uma associação *Formal* são relações que acontecem diretamente entre duas ou mais entidades, sem qualquer outro indivíduo entre elas e são usadas para comparação entre elas.
ComponentOf	*ComponentOf* é uma relação entre complexos funcionais, por exemplo um motor é um componente funcional de um carro.
MemberOf	Uma associação *MemberOf* é feita por *Collectives* e seus objetos, como uma pessoa é parte de um grupo.
SubCollectionOf	*SubCollectionOf* é uma associação entre *Collectives*, por exemplo o norte da Floresta Amazônica é parte da Floresta Amazônica.
SubQuantityOf	Uma associação *SubQuantityOf* dá-se entre *Quantities*, como Álcool é parte de Vinho.

usaremos essa linguagem para construção de ontologias de referência.

Albuquerque e Guizzardi (2013) apresentam extensões a UFO para melhorar a ontologia, trazendo fundamentos relativos aos espaços de valor, empregando a noção de espaços de referência semântica. Apresentando uma aplicação concreta desta teoria, aplicam as extensões propostas a UFO usando classes *Datatypes* UML.

A Figura 4 representa um fragmento das extensões. Na figura pode-se observar *ReferenceStructure*, que podem ser definidas como escalas de qualidade. *ReferenceStructures* podem ser subdivididas em *MeasurementStrucuture* e *NominalStructure*. O *MeasurementStrucuture* são frequentemente indicado por um conjunto de símbolos, como (1, 2, 3, 4, 5, 6, 7, 8, 9, 0), podendo ser dividido de várias maneiras, portanto, originando diferentes escalas. O *NominalStructure* são também conjunto de símbolos como o alfabeto português. Em outras palavras, *MeasurementStrucuture* agem como escalas baseadas em estruturas de qualidade, enquanto *NominalStructure* representam estruturas de nomeação. *MeasurementStrucuture*

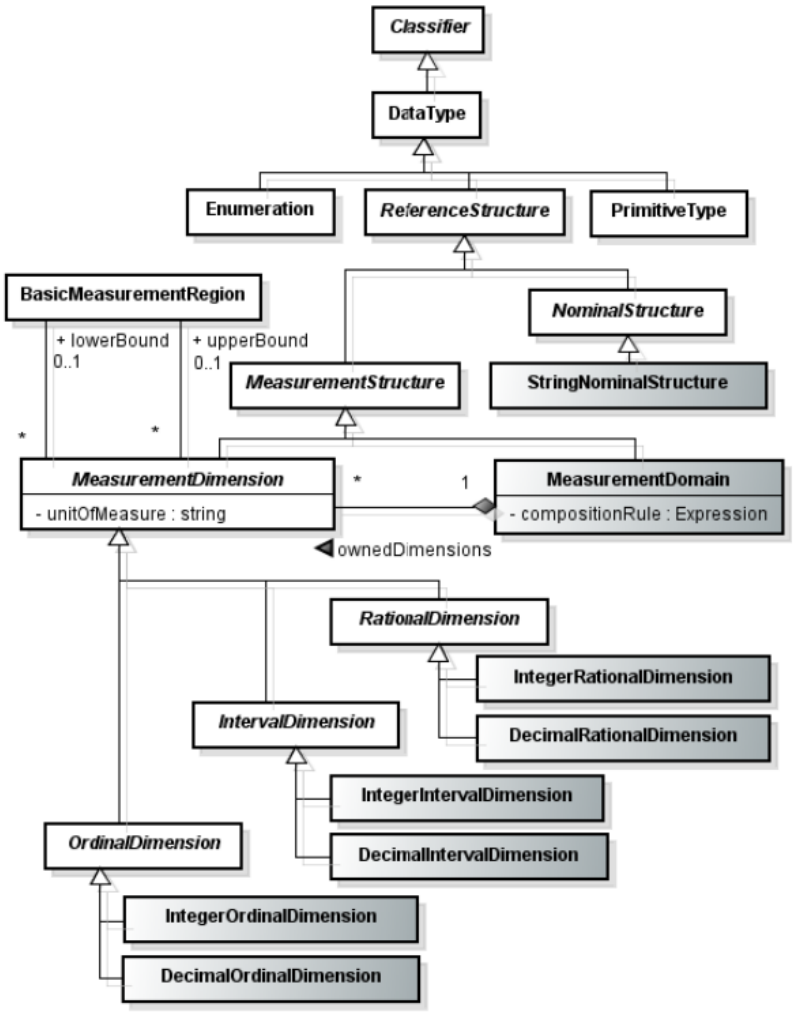

Figura 4 – Um fragmento da extensão OntoUML sobre estruturas de referência (ALBUQUERQUE; GUIZZARDI, 2013).

é subdividida em duas partes: *MeasurementDimension*, que trata da dimensão da medição da qualidade, e *MeasurementDomain*, estruturas de medição multi-dimensionais.

MeasurementDimension é subdividida em três partes: *OrdinalDimension*, estruturas baseadas na noção de escala de ordenação, *IntervalDimension*, estruturas com base em intervalos (comprimento da unidade de medida) e *RationalDimension*, que são estruturas com base em intervalos e que podem ser convertidos para outros, como multiplicação ou divisão de números.

A Tabela 5 apresenta detalhadamente as entidades usadas em OntoUML com relação a espaços de valor, e a relação entre elas, definindo também os conceitos de qualidade: *PerceivableQuality*, *NonPerceivableQuality* e *NominalQuality*.

Tabela 5 – Tabela da extensão da OntoUML para representar universal de qualidade e espaços de qualidade.

Entidade	Descrição
IntegerOrdinalDimension/ DecimalOrdinalDimension	*OrdinalDimension* usadas para representar estruturas de qualidade ordenada.
IntegerIntervalDimension/ DecimalIntervalDimension	*IntervalDimension* usado para representar estratégias de qualidade baseadas em intervalos, tais como escalas Celsius e Fahrenheit.
IntegerRationalDimension/ DecimalRationalDimension	*RationalDimension* que podem ser convertidos para outros, por multiplicação / divisão de uma constante, como as escalas de metro/ decímetro/ centímetro.
MeasurementDomain	Estruturas de medição multidimensionais, compostas por dimensões de medição.
StringNominalStructure	Estruturas nominais que permitem referência a substantivos.
MeasurementEnumeration	*MeasurementStructures* com um conjunto predefinido de referências baseado em um *ReferenceStructures*.
Structuration	Uma relação binária que associa estruturas de qualidade que são abstraídas como estruturas de referência e qualidades universais.
PerceivableQuality	Uma qualidade universal que é originada partir de observação e medição.
NonPerceivableQuality	Uma qualidade universal que é originada nos processos de concepção.
NominalQuality	Um qualidade universal que é originada em convenções sociais.

2.2 SABiO

Para construir as ontologias de referências propostas nesse projeto, foi utilizado o método SABiO (*Systematic Approach to Build Ontologies*) (FALBO, 2014), uma abordagem baseada nos processos da Engenharia de Software, amplamente aceita pela comunidade, tendo foco na construção de ontologias de domínio. SABiO é composto por cinco fases e cinco processos de apoio, como na Figura 5:

- **Identificação de Proposta e Levantamento de Requisitos**: fase para a identificação do propósito e a utilização da ontologia. Nessa fase são definidas as questões de competência que a ontologia deve responder.
- **Captura e Formalização da Ontologia**: nesta fase os conceitos, relações, restrições e axiomas para a conceituação da ontologia são identificados e organizados. Nessa fase também é feita uma representação em linguagem formal da conceituação capturada pela ontologia. SABiO sugere linguagem gráfica para facilitar a conexão com interessados e engenheiros de ontologia, resultando em uma ontologia de referência de domínio.

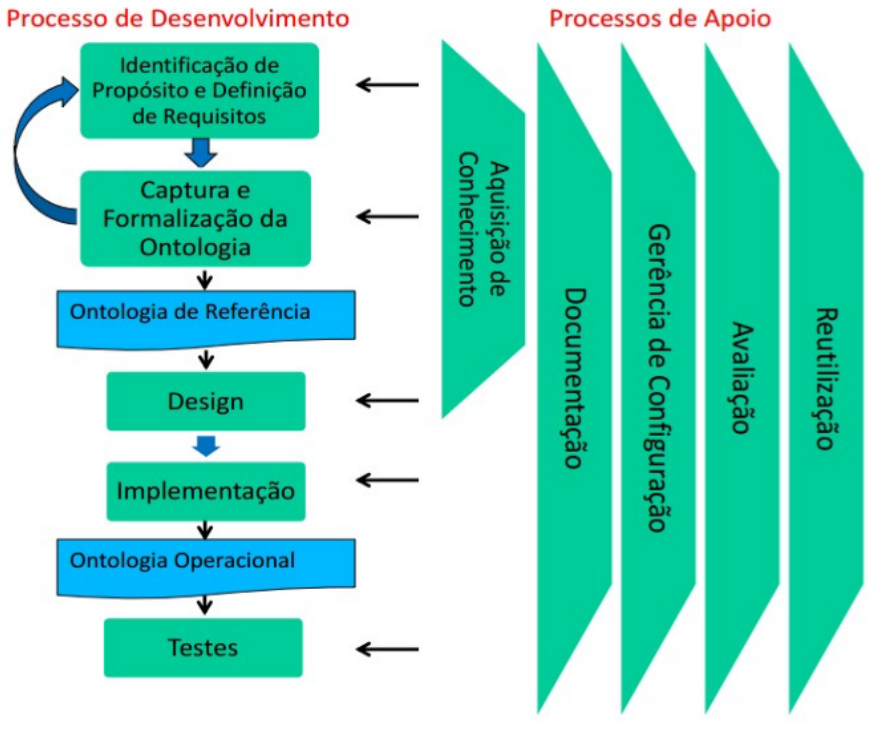

Figura 5 – Processo de Desenvolvimento de ontologias por SABiO (FALBO, 2014)

- ***Design***: nessa fase, a especificação conceitual da ontologia de referência é transformada em uma especificação de *design* para uma ontologia operacional, levando em consideração as questões arquiteturais e técnicas para implementação.

- **Implementação**: essa fase trata da codificação da ontologia para uma linguagem operacional escolhida na fase de *design* para ser capaz de ser interpretada por agentes de software.

- **Teste**: a fase de testes é a ultima fase proposta por SABiO, realizando a verificação e validação de forma objetiva e dinâmica a ontologia operacional usando conjunto de casos de teste e comparar os resultados com o comportamento esperado pelas questões de competência. SABiO sugere que os testes sejam feitos primeiramente por sub-ontologias que compõem a ontologia de referência e subindo à medida que as sub-ontologias vão sendo integradas até testar a ontologia final.

SABiO sugere ainda cinco processos de suporte: Aquisição de Conhecimento, Documentação, Gerência de Configuração, Avaliação e Reuso, mostrados na Figura 5, que são desenvolvidos em paralelo ao processo de desenvolvimento da ontologia.

2.3 Conclusão do Capítulo

Neste capítulo foram apresentados os conceitos teóricos para o desenvolvimento de ontologias deste trabalho. Primeiramente foi apresentada conceitos sobre ontologias, onde demonstramos os vários níveis ontológicos e apresentamos a ontologia de fundamentação UFO, a linguagem OntoUML e o método SABiO. OntoUML é uma linguagem ontologicamente bem fundamentada, de modo que os modelos criados nesta linguagem sejam consistentes com a Ontologia UFO.

Importante destacar que o foco desse trabalho são os conceitos da UFO-A para a construção de ontologias de domínio como base de sistemas de informação Web.

3 Web Semântica e FrameWeb

Nesse capítulo serão apresentados os conceitos do estado-da-arte sobre Web Semântica e dados ligados (*linked data*) para o embasamento teórico necessário à criação e ao desenvolvimento desta dissertação. Também será apresentado o método ***FrameWeb*** para desenvolvimento de Sistemas de Informação Web (*Web-based Information Systems*, ou WISs), usado como base para a nossa pesquisa.

O capítulo é dividido da seguinte forma: a Seção 3.1 apresenta conceitos sobre Web Semântica e ferramentas para aplicação da semântica na Web e dados ligados. A Seção 3.2 apresenta ***FrameWeb***, o método que foi estendido para o desenvolvimento da proposta desta dissertação. Por fim, a Seção 3.3 apresenta trabalhos relacionados ao dessa dissertação.

3.1 A Web Semântica

Desde que a *World Wide Web* (WWW, ou Web) foi criada em 1989 nos laboratórios da CERN, ela rapidamente evoluiu para um local de informações que é composto de bilhões de páginas interligadas. Estas páginas nos mostram a informação sobre uma enorme variedade de tópicos e áreas de conhecimento, entretanto sem semântica, ou seja, apenas apresenta a informação e não a interpreta, essa parte ficando a cargo de agentes humanos.

A Web Semântica (*Semantic Web*) é uma proposta de evolução da Web atual e tem por objetivo proporcionar que agentes de software possam ser capazes de interpretar o conteúdo da Web, ajudando assim humanos em suas atividades. Isso significa criar uma plataforma comum que permita o compartilhamento e a reutilização dos dados por meio das fronteiras de aplicações, empresas e comunidades, podendo ser processados automaticamente tanto por ferramentas quanto manualmente, também revelando novos relacionamentos possíveis entre as porções de dados.

O FAQ disponível em (W3C, 2009) exemplifica os benefícios da Web Semântica: eu posso ver meus extratos bancários na Web, também minhas fotografias, e ainda meus compromissos em um calendário. Mas posso ver minhas fotos em um calendário para ver o que eu estava fazendo quando eu as tirei? Posso ver linhas de extrato bancário em um calendário? Por que não? Por que não termos uma rede de dados? Por que os dados são controlados por aplicativos, e cada aplicativo os mantém para si mesmo?

Para superar essas limitações e até responder essas perguntas, a Web Semântica visa tornar o conteúdo da Web semântico para a máquina. Berners-Lee, Hendler e Lassila

(2001) definem a Web Semântica como "uma extensão da Web atual na qual as informações têm um significado bem definido, permitindo que os computadores e as pessoas trabalhem em cooperação".

Segundo Tauberer (2006) em seu artigo "*What is RDF*", o termo semântica na Web Semântica não significa que os computadores realmente serão capazes de entender o significado das palavras e termos, mas que o conhecimento poderá ser mecanicamente manipulado por computadores, visando trazer maior facilidade aos humanos.

A visão da Web Semântica é estender os princípios da Web de documentos para dados, para que os dados possam ser acessados de maneira geral em uma arquitetura genérica. Sendo assim relacionados uns com os outros exatamente como documentos, permitindo que sejam compartilhados e reutilizados em todas aplicações, seja de empresas ou da comunidade em geral, para serem processados automaticamente por ferramentas ou manualmente.

Para alcançar os objetivos descritos acima, o mais importante é ser capaz de definir e descrever as relações entre dados (isto é, recursos) na Web, o que pode ser feito por *hiperlinks*, como já se faz entre documentos na Web. A diferença importante é que, na Web Semântica, esses relacionamentos podem ser estabelecidos entre quaisquer dois recursos, independente da página, e a relação em si também é nomeada (W3C, 2009). Isso pode ser feito criando metadados em RDF (*Resource Description Framework*), descrito na subseção a seguir, que serve como um dos blocos fundamentais da Web Semântica, fornecendo uma definição formal de descrição de dados.

3.1.1 RDF

RDF (*Resource Description Framework*) é um modelo para representar dados e metadados da Web. É uma tecnologia aprovada pela W3C (*World Wide Web Consortium*),[1] tendo como um dos principais objetivos criar um modelo simples para armazenamento de informações. Modelos RDF são compostos por três elementos básicos, ou seja, triplas (CONSORTIUM et al., 2014):

- Sujeito: um recurso qualquer na Web, identificado por uma URI (*Uniform Resource Identifier*);
- Predicado: também identificado por uma URI, representa uma propriedade, algo que relaciona o sujeito com o objeto;
- Objeto: consiste no valor da propriedade, ou seja, do predicado. Pode ser uma URI (relação entre dois recursos) ou um valor literal.

[1] <http://www.w3.com>.

O modelo de dados em RDF é projetado para a representação integrada de informações que se originam de diversas fontes como ligação de nós através de arcos direcionados (formando triplas: nó-arco-nó) e pretende ser empregado como uma linguagem genérica, capaz de modelar os dados usados na Web. Detalhes sobre o modelo RDF podem se encontrados em (CONSORTIUM et al., 2014). Abaixo, exemplificamos o uso do modelo de dados de RDF:

Danillo Ricardo Celino tem nickname drcelino

Sujeito Predicado Objeto

Podemos entender assim: um sujeito de uma tripla é o URI que identifica o recurso; o objeto pode ser um valor literal simples, uma palavra, número, data, ou até um URI de outro recurso que está de alguma forma relacionado com o sujeito; e o predicado indica que tipo de relação existe entre sujeito e objeto, por exemplo, este é o nome, data de nascimento ou o empregador. O predicado também é identificado por um URI. Os URIs de predicados vêm de vocabulários, coleções de URIs que podem ser usados para representar informações sobre um determinado domínio.

Para publicar um grafo RDF na Web, ele deve primeiro ser serializado usando uma sintaxe RDF, ou seja, organizar as triplas em um grafo RDF usando uma sintaxe específica para guardar em um arquivo. A sintaxe padronizada pelo W3C e é amplamente utilizada para publicar dados vinculados na Web é a RDF/XML. A Listagem 3.1 mostra o exemplo anterior em formato RDF/XML.

Listagem 3.1 – Exemplo de dados em RDF serializados em formato RDF/XML.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3  xmlns:foaf="http://xmlns.com/foaf/0.1/">
4
5      <rdf:Description rdf:about="http://www.example.com/people/danillo_celino">
6        <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
7         <foaf:bick>drcelino</foaf:nick>
8      </rdf:Description>
9    </rdf:RDF>
```

Existem outras formas de serializar o RDF, RDFa (ADIDA et al., 2008), que é um formato de serialização que incorpora triplas em RDF em documentos HTML, o que significa que o conteúdo existente na página pode ser marcado com RDFa modificando o código HTML, mostrando os dados estruturados para a Web. Pode-se ainda serializar em em triplas usando Turtle (PRUD'HOMMEAUX et al., 2013), N-Triples (CYGANIAK; HARTH; HOGAN, 2008) e JSON-LD (SPORNY et al., 2014).

RDF fornece uma representação de metadados, suas propriedades e seus relacionamentos. RDF não descreve um mecanismo para definição dos recursos, suas propriedades

e seus relacionamentos. Esta é a função exercida pelo *RDF Schema* (RDFS), isto é, um esquema que permite a criação de classes e propriedades, inclusive descrições dessas classes, combinações possíveis de classes, propriedades e valores e restrições entre relacionamentos. RDFS é descrito na subseção 3.1.2 a seguir.

3.1.2 RDFS

O RDFS (*RDF Schema*) (BRICKLEY; GUHA; MCBRIDE, 2014) fornece um vocabulário especial, introduzindo o conceito de classe (*rdfs:Class*) e propriedade (*rdfs:Property*), além de propriedades RDF para modelar hierarquias de classe (*rdfs:subClassOf*), hierarquias de propriedade (*rdfs:subPropertyOf*), bem como definir domínio e imagem de uma relação (*rdfs:domain*, *rdfs:range*). Essas propriedades principais de RDFS permitem descrever e incorporar semântica de vocabulários definidos pelo usuário no próprio RDF.

A Figura 6 específica a relação entre os conceitos principais de RDFS. Na figura, as classes são representadas pelas ovais nas cores verdes. Vale resaltar que *rdfs:Property* e *rdfs:Class* são sublclasses de *rdfs:Resource*, que representa qualquer coisa que tenha um IRI (*Internationalized Resource Identifier*), generalização de URI, fornecendo meios para identificar de maneira única os recursos da Web Semântica. As propriedades são representadas pelos retângulos laranja, sendo todas instâncias (*rdf:type*) de *rdfs:Property*. Uma oval amarela representa *rdfs:Literal*, instância de *rdfs:Class* que representa a classe de valores literais, como strings e números inteiros.

É importante resaltar que ao passar do tempo RDFS precisaria de mais representação de algumas restrições, assim surge OWL, descrito na Subseção 3.1.3.

3.1.3 OWL

OWL (*Web Ontology Language*) (GROUP et al., 2009) estende o RDFS e permite expressar outras definições de esquema em RDF, por exemplo, permitindo expressar igualdade de indivíduos (*owl:sameAs*), equivalência ou disjunção de propriedades e classes (*owl:equivalenteClass, owl:equivalenteProperty, owl:disjointWith, owl:propertyDisjointWith*), ou definições de classe complexas.

Silva (2016) destaca que OWL inclui um termo para classe *owl:Class*, instância de *rdfs:Class*, sendo *owl:Class* especialização *rdfs:Class* e *rdfs:Class* especialização de *owl:Class*, o que os torna equivalentes. OWL apresenta uma superclasse *owl:Thing* para *rdfs:Resource* que classifica tudo. Apresenta também especialização de *rdf:Property*, uma para distinguir instâncias de *owl:DatatypeProperty*, que conecta um recurso a um literal, e outra especialização para instâncias de *owl:ObjectProperty*, que conecta um recurso a outro recurso. A Figura 7 representa um fragmento do metamodelo de OWL e explica essas principais relações com RDFS.

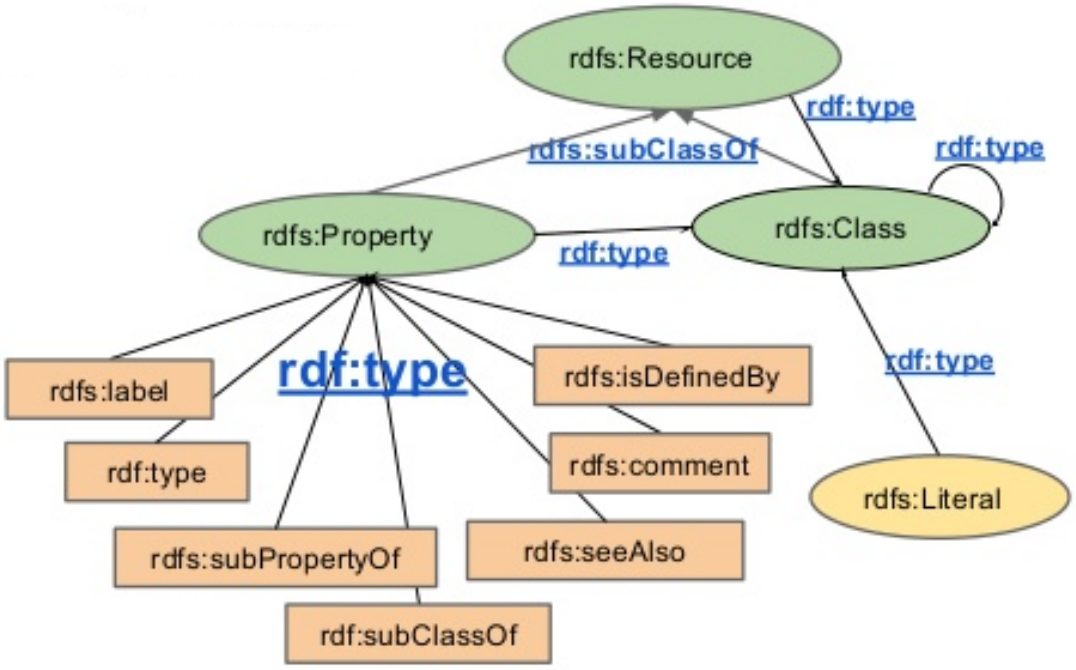

Figura 6 – Fragmento do vocabulário RDFS

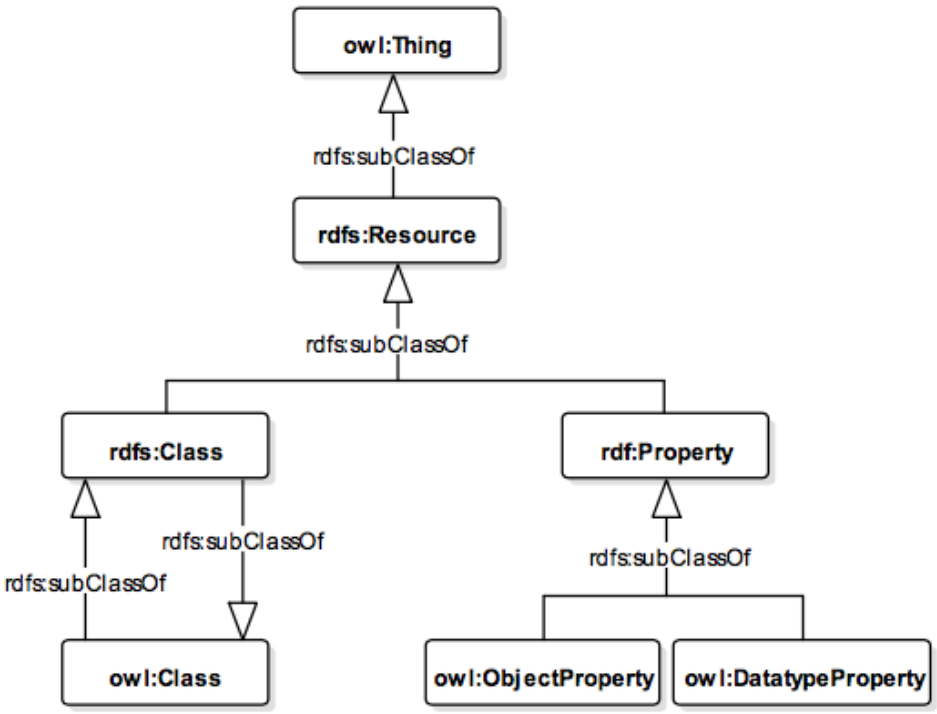

Figura 7 – Fragmento do metamodelo OWL extraído de (SILVA, 2016)

Importante destacar que com base em linguagens como OWL e RDFS usados para metadados foram criados vocabulários de referências de determinados domínios, como **DBpedia** (AUER et al., 2007), que extrai conteúdo estruturado da Wikipedia,[2] **FOAF** (*friend-of-a-friend*) (BRICKLEY; MILLER, 2007), que é adequado para a definição de metadados sobre pessoas, seus interesses, seus relacionamentos e suas atividades, ***Dublin Core*** (WEIBEL et al., 1998), vocabulário para documentos, **SKOS** (*Simple Knowledge Organization System*) (ISAAC; SUMMERS, 2009), vocabulário para esquemas de classificação, **DBLP** (*Digital Bibliography and Library Project*) (LEY, 2005), descrevendo pesquisadores e publicações acadêmicas, dentre muitos outros.

3.1.4 Dados Ligados

Heath e Bizer (2011) resumem os conceitos de Dados Ligados (*Linked Data - LD*) como simplesmente usar a Web para criar *links* tipados entre dados de diferentes fontes. Estes podem ser tão diversos como bases de dados mantidas por duas organizações em diferentes locais geográficos, ou simplesmente sistemas heterogêneos em uma organização que, historicamente, não facilitou a interoperabilidade ao nível dos dados.

Dados ligados também descrevem um conjunto de melhores práticas para consumo e publicação de dados estruturados na Web. Essas melhores práticas são conhecidas como princípios de *Linked Data*, Tim Berners-Lee, quando descreve a arquitetura da Web para dados ligados (BIZER; HEATH; BERNERS-LEE, 2009), apresenta os seguintes princípios:

- Use URIs como nomes para as coisas;
- Use URIs HTTP, para que as pessoas possam procurar esses nomes;
- Quando alguém procurar um URI, forneça também informações úteis, usando os padrões (RDF(S), OWL);
- Incluir links para outros URIs, para que eles possam descobrir mais coisas.

Sendo assim, a ideia básica de Dados Ligados é adicionar à arquitetura geral da Web a tarefa de compartilhar dados estruturados em escala global. Para entender esses princípios de Dados Ligados é importante entender a arquitetura do documento Web.

Heath e Bizer (2011) descrevem a arquitetura do documento Web sendo construída sobre um pequeno conjunto de padrões simples: *Uniform Resource Identifiers* (URIs) como mecanismo único de identificação, o *Hypertext Transfer Protocol* (HTTP) como mecanismo de acesso universal, e a *Hypertext Markup Language* (HTML) como um formato de conteúdo amplamente utilizado. A Web é baseada na idéia de definir hiperlinks entre documentos da Web guardados em diferentes servidores Web.

[2] pt.wikipedia.com

Os *hiperlinks* permitem que os usuários naveguem entre diferentes servidores, permitindo que os agentes de software de pesquisa rastreiem a Web e ofereçam recursos usando o conteúdo rastreado, sendo assim conectando conteúdos de diferentes servidores em um único espaço global de informações. Os Dados Ligados utilizam esta mesma arquitetura, porém interligando coisas (recursos) e não páginas, formando um espaço global de dados.

Como ferramenta para procurar *links*, W3C sugere o uso da linguagem SPARQL. O *SPARQL Protocol and RDF Query Language* (SPARQL) é, essencialmente, uma linguagem de consulta de grafos RDF. Como descrito em (HARRIS; SEABORNE; PRUD'HOMMEAUX, 2013), a maioria das formas de consulta SPARQL contém um conjunto de padrões de triplas chamado grafo básico padrão. Esses padrões de triplas são como triplas de RDF, exceto que cada sujeito, predicado e objeto pode ser uma variável. Um grafo básico padrão corresponde a um subgrafo dos dados RDF quando os termos RDF desse subgrafo podem ser substituídos pelas variáveis.

Por exemplo, usando um vocabulário como FOAF, pode-se criar uma consulta SPARQL para encontrar o nome de uma pessoa a partir de um grafo de dados fornecido. A consulta consiste em duas partes: o item SELECT, que identifica as variáveis a serem exibidas nos resultados da consulta, e o item WHERE, que fornece o grafo básico padrão para coincidir com o grafo de dados. O grafo básico padrão neste exemplo consiste em um único padrão da tripla com uma única variável (`?Nome`) na posição do objeto.

Sendo assim, temos o dado RDF descrito em *turtle*:

```
1 <http://www.example.com/people/danillo_celino> <http://xmlns.com/foaf/0.1/nick>
    "drcelino" .
```

Temos a consulta SPARQL:

```
1 PREFIX foaf:   <http://xmlns.com/foaf/0.1/>
2 SELECT ?Nickname
3 WHERE { ?x foaf:nick ?Nickname }
```

Retornando como resultado na consulta do dado mostrado na Tabela 6:

Tabela 6 – Tabela resultado da consulta SPARQL

Nickname
"drcelino"

Para ajudar usuários (humanos ou agentes de software) a consultarem uma base de conhecimento usando a linguagem SPARQL, foram criados SPARQL *endpoints*. SPARQL *endpoint* (QUILITZ; LESER, 2008) é um serviço de protocolo SPARQL, do qual os resultados normalmente são retornados em um ou mais formatos processáveis por máquina com uma interface amigável para uma base de conhecimento.

Existem *triple stores*, ou seja, bases de dados onde se persistem triplas RDF, muito

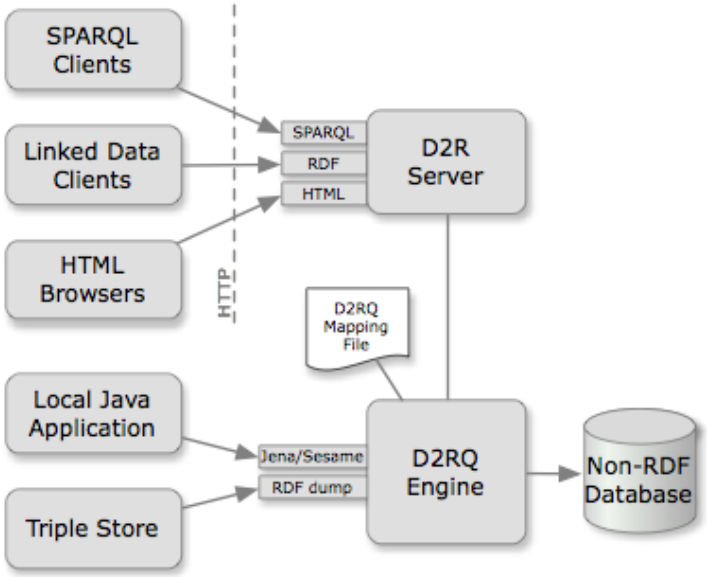

Figura 8 – Arquitetura D2RQ (BIZER; CYGANIAK, 2013).

difundidas na indústria, como Virtuoso (ERLING; MIKHAILOV, 2009) e D2RQ (BIZER; CYGANIAK, 2007), os quais também são acompanhados por SPARQL *endpoints*. Focamos nesse projeto em D2RQ. D2RQ (*Database to* RDF) é um sistema para acessar bancos de dados relacionais como grafos RDF virtuais para somente leitura (BIZER; CYGANIAK, 2007). Oferece acesso RDF ao conteúdo de bancos de dados relacionais sem ter que replicá-lo em um *triple store* RDF. Usando D2RQ podemos:

- Consultar um banco de dados não RDF usando SPARQL;
- Acessar o conteúdo do banco de dados com dados vinculados na Web;
- Acessar informações em um banco de dados não RDF usando uma API de software (ex.: Apache Jena API[3]).

D2RQ tem uma máquina que gera a partir de um banco de dados relacional arquivos em RDF (*triple store*) utilizando um arquivo de mapeamento escrito em *turtle* para Tabelas e Colunas, criando um vocabulário novo ou se baseando em um fornecido. D2RQ também tem um servidor (D2RQ Server) que apresenta esses dados tanto para humanos quanto para agentes de software, gerando automaticamente um SPARQL *endpoint* para consultas RDF. A Figura 8 mostra a arquitetura de D2RQ de forma detalhada. D2RQ se torna assim uma ferramenta que facilita a criação de *triple store* usando banco de dados relacionais, tecnologia amplamente utilizada para armazenamento de dados de sistemas de informação.

A Figura 9 demonstra por meio de uma pilha tecnológica qual parte da arquitetura da Web Semântica as linguagens se encaixam, mostrando também que o conceito de Dados

[3] <https://jena.apache.org>.

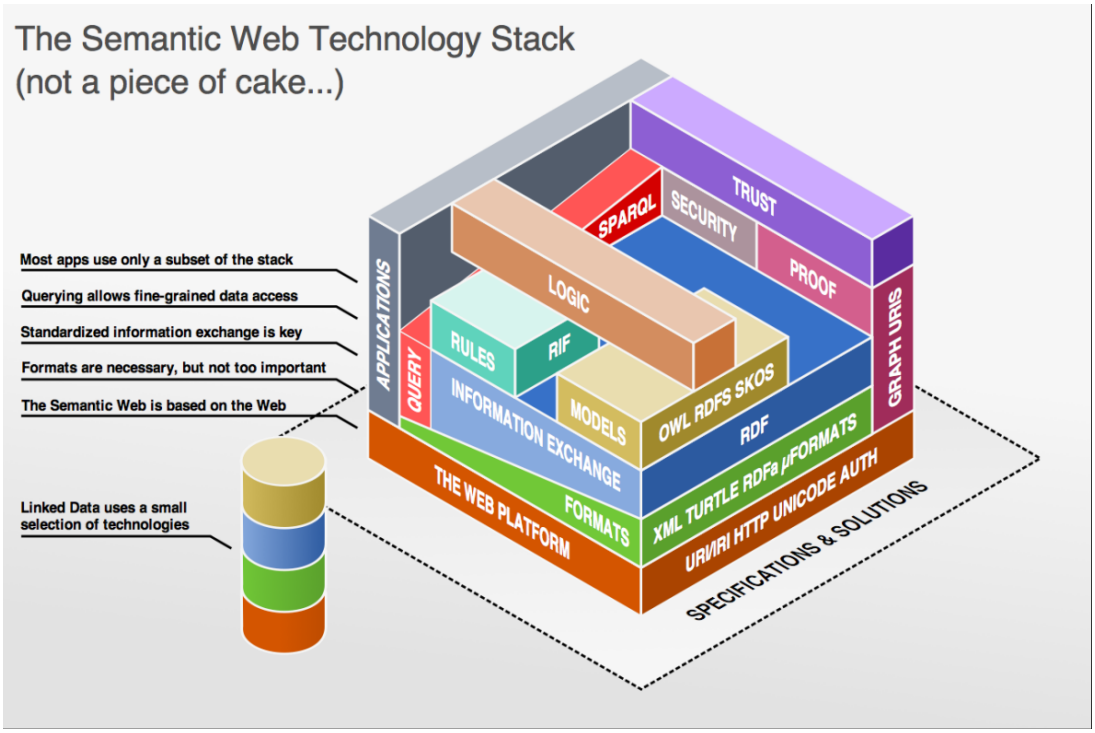

Figura 9 – Pilha tecnológica da Web Semântica (NOWACK, 2009)

Ligados usa essas tecnologias descritas anteriormente para operar, demonstrando sua intrínseca relação com a Web Semântica.

3.1.5 Serviços Web Semânticos

É cada vez mais comum fornecer serviços na Web, ou seja, um programa de software identificado por uma URI, que pode ser acessado via Internet por meio de sua interface exposta. Para ser um serviço, é necessário que a aplicação Web possa receber requisições de outras aplicações, sendo essas Web ou não, e gerar respostas possíveis de serem interpretadas.

De acordo com o W3C (2004), um *Web Service* é um software projetado para suportar interação máquina-a-máquina interoperáveis sobre uma rede. Utilizando uma interface de formato processável. Em termos práticos, *Web Service* se refere a uma arquitetura de comunicação entre softwares que sejam da mesma plataforma ou não. A característica marcante dessa arquitetura é que a comunicação é sempre realizada em rede e deve estar sempre disponível. Para a implementação de *Web Services*, os protocolos SOAP (*Simple Object Data Protocol*) e REST (*Representational State Transfer*) são as opções mais utilizadas atualmente.

Box et al. (2000) define SOAP como um protocolo leve para troca de informações em um ambiente descentralizado e distribuído, um protocolo baseado em XML que consiste em três partes: um envelope que define uma estrutura para descrever o que está na mensagem

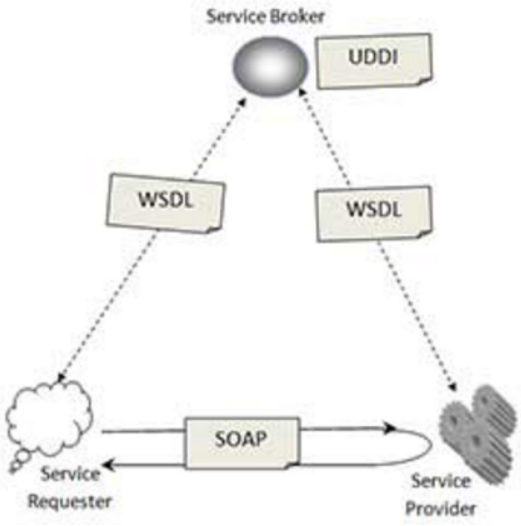

Figura 10 – Arquitetura SOAP (BOX et al., 2000)

e como processá-la, um conjunto de regras de codificação para expressar instâncias de tipos de dados e uma convenção para representação de procedimentos de envio e respostas.

No início da década de 2000, o protocolo SOAP teve grande importância, quando passou a ser uma recomendação da W3C para desenvolvimento de serviços Web, sendo o padrão mais implementado na época e deixando um legado de sistemas e integrações que perdura até hoje.

A Figura 10 representa os protocolos de serviço Web e a ligação entre eles para sua execução. Importante salientar que a linguagem XML é usada para estruturar os dados nas mensagens recebidas ou enviadas. Na figura, percebemos que nas chamadas às operações, os parâmetros de entrada ou saída são codificadas no protocolo SOAP[4] (*Simple Object Access Protocol*). Os serviços são descritos usando a linguagem WSDL[5] (*Web Services Description Language*). O processo de publicação ou descoberta de *Web Services* utiliza o protocolo UDDI[6] (*Universal Description Discovery and Integration*).

O REST, por sua vez, foi desenvolvido juntamente com o protocolo HTTP, com o objetivo de estabelecer um protocolo para comunicação de objetos e serviços e criar serviços que poderiam ser acessados por qualquer tipo de sistema. Fielding e Taylor (2000), em seu trabalho acadêmico *Architectural Styles and the Design of Network-based Software Architectures*, na Universidade da Califórnia, estabelecem os princípios de REST.

Na Figura 11, a *AppClient* envia uma requisição (HTTP *Request*) contendo as informações necessárias para executar uma determinada operação para a *AppServer*. A *AppServer* processa a requisição e devolve para a *AppClient* uma resposta (HTTP *Response*)

[4] <http://www.w3.org/TR/soap/>.
[5] <http://www.w3schools.com/xml/xml_wsdl.asp>.
[6] <http://www.tutorialspoint.com/uddi/uddi_overview.htm>.

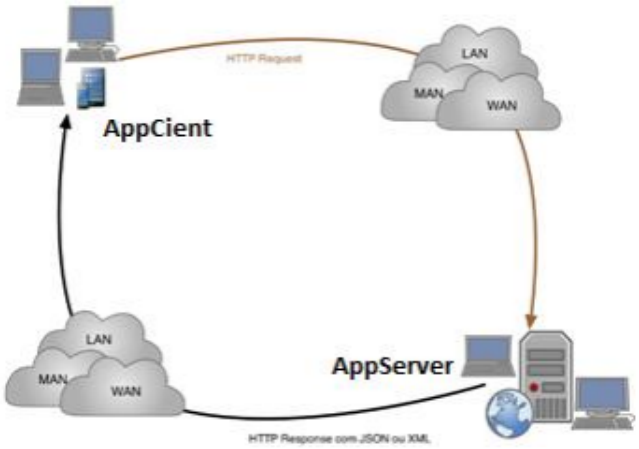

Figura 11 – Arquitetura REST (FERREIRA[1]; MOTA[1], 2014)

contendo um arquivo XML ou uma *String* JSON (*Java Script Object Notation*) contendo desde uma simples mensagem a um conjunto de informações complexas.

O JSON é um formato leve para intercâmbio de dados. Como podemos observar na arquitetura REST, não existe um descritor de funcionamento do serviço. A requisição realizada pela *AppClient* parte do principio que a mesma conhece o que deve ser enviado para *AppServer*, facilitando assim o processo de implementação. JSON é um texto que representa um objeto, no formato de pares `chave: valor`.

Web services, porém englobam apenas aspectos sintáticos, não envolvendo o significado das mensagens. O WSDL, por exemplo, pode especificar as operações disponíveis por meio de um serviço Web e a estrutura dos dados enviados e recebidos, mas não pode especificar o significado semântico dos dados ou restrições semânticas nos dados. A princípio é necessária a intervenção humana para se criar uma composição de serviços.

Serviços Web Semânticos (*Semantic Web Services*) (MCILRAITH; SON; ZENG, 2001) trazem uma luz a esse problema, fornecendo uma outra camada no topo da infraestrutura dos *Web Services* possibilitando a descoberta, composição e execução de funcionalidades descrevendo semanticamente os serviços. Existem quatro abordagens principais de *frameworks* que têm impulsionado o desenvolvimento do Serviços Web Semânticos: SAWSDL,[7] WSMO,[8] METEOR-S[9] e OWL-S.[10] Aqui, nos concentramos neste último.

OWL-S (*Web Ontology Language Service*) é uma linguagem, baseada em OWL, com o objetivo de oferecer descrição semântica de serviços na Web, usando anotações, fornecendo um vocabulário padronizado para descrição de serviços, que pode ser utilizado com outros aspectos da OWL.

[7] <https://www.w3.org/TR/sawsdl/>.
[8] <http://www.wsmo.org/>.
[9] <http://www.lsdis.cs.uga.edu/projects/meteor-s/>.
[10] <http://www.w3.org/Submission/OWL-S/>.

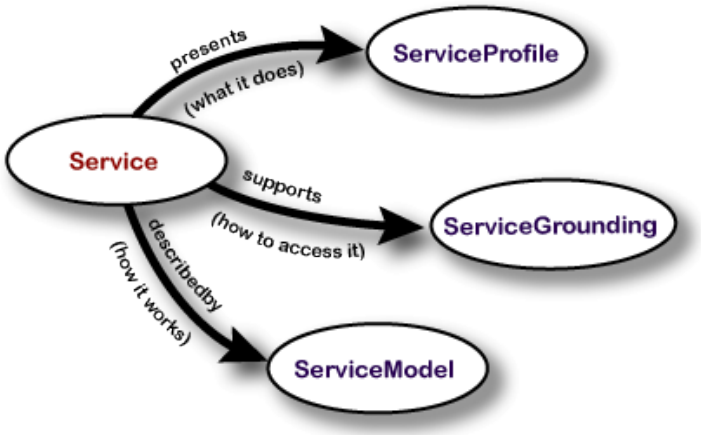

Figura 12 – Descrição da OWL-S (BURSTEIN et al., 2004)

Foi escolhido OWL-S pois fornece invocação automática de *web service*, sem a necessidade de se ter um programa específico para isso. Visando possibilitar a um agente de software ler automaticamente pela descrição das entradas e saídas de *web service*. OWL-S também fornece, em uma tarefa complexa, envolver a invocação coordenada de vários *web services*, tendo base em uma descrição de alto nível do objetivo, assistindo a composição e interoperação dos serviços permitindo a execução automática dessas tarefas.

OWL-S propõe a definição de uma ontologia principal, denominada *Service*, e outras três sub-ontologias: *Profile*, *Process* e *Grounding*.

Estas três sub-ontologias descrevem, respectivamente, o que o serviço provê ao seu consumidor, como o serviço pode ser usado e como interagir com o serviço. A ontologia principal, *Service*, tem como única função ligar as três ontologias, definindo classes abstratas que posteriormente serão especializadas pelas sub-ontologias (BURSTEIN et al., 2004).

Como pode ser visto na Figura 12, a ontologia *Service* define a classe *Service* e outras três classes: *ServiceProfile*, *ServiceModel* e *ServiceGrounding*. Sendo assim:

- As especializações de *ServiceModel* descrevem como um cliente pode usar o serviço e como ele funciona.
- As especializações de *ServiceProfile* descrevem as funcionalidades disponibilizadas pelo serviço.
- As especializações concretas de *ServiceGrounding* devem especificar os detalhes de como um agente de software acessa o serviço, descrevendo detalhes dos protocolos de comunicação utilizados e os formatos das mensagens.

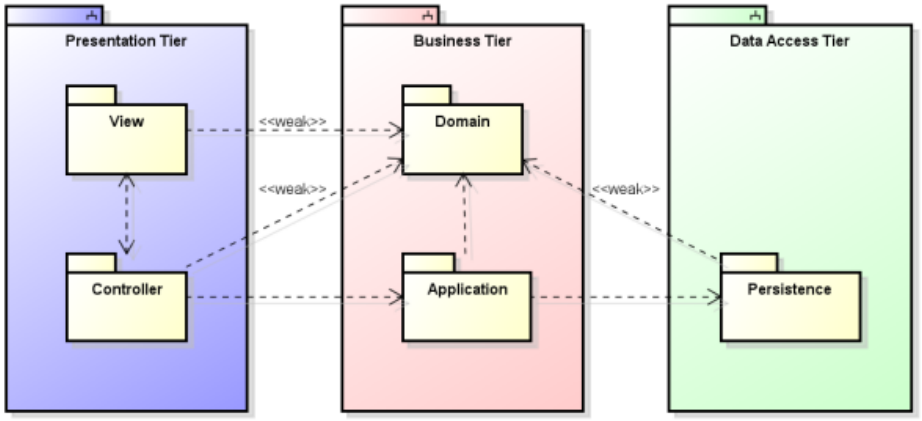

Figura 13 – Arquitetura ***FrameWeb*** (SOUZA; FALBO, 2007)

3.2 FrameWeb

FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA; FALBO, 2007) é um método de Engenharia Web direcionado à fase de projeto e voltado para aplicações que fazem uso de *frameworks*. A arquitetura de ***FrameWeb*** é baseada no padrão Camada de Serviço (*Service Layer*) (FOWLER, 2002) como descrito na Figura 13. A arquitetura propõe 3 camadas: a camada de negócio (*Business Tier*), responsável pelas funcionalidades relacionadas às regras de negócio da aplicação, camada da apresentação (*Presentation Tier*), responsável pelas interface com o usuário e a camada de acesso a dados (*Data Access Tier*), responsável pela persistência. A camada de apresentação acessa as funcionalidades pelo padrão arquitetural MVC (*Model-View-Controller*) (VLISSIDES et al., 1995) que fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação, sendo três pacotes:

- ***Model***: Contendo o pacote *Domain* (Domínio) onde estão os elementos do domínio do problema do Sistema de Informação Web (*Web-based Information System* - WIS) específico, o pacote *Application* (Aplicação), onde se definem parte dos casos de uso e o pacote *Persistence* (Persistência) responsável pelo armazenamento e recuperação dos dados;
- ***View***: O pacote *View* (Visão) contendo os elementos de comunicação com o usuário;
- ***Controller***: O pacote *Controller* (Controle) contendo os elementos que executam e respondem às requisições.

O método ***FrameWeb*** define uma linguagem de modelagem baseada no metamodelo da UML, propondo e tendo como base quatro tipos de modelos de projeto baseados no diagrama de classes da UML, que representam os componentes de WIS e dos *frameworks*

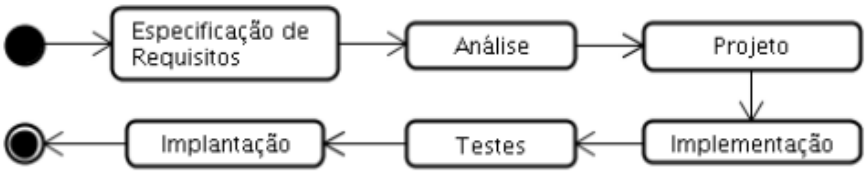

Figura 14 – processo de desenvolvimento de software simples para o ***FrameWeb*** (SOUZA; FALBO, 2007)

propostos por ***FrameWeb***. Cada modelo representa as informações levantadas durante o processo de desenvolvimento envolvendo as camadas da arquitetura de software. São eles:

- **Modelo de Entidades** (*Entity Model*): modela elementos do domínio do problema e da aplicação Web e o mapeamento desses objetos para a persistência, referindo-se ao pacote de domínio;
- **Modelo de Persistência** (*Persistence Model*): modela as classes responsáveis pelas operações de persistência, referente ao pacote de persistência, utilizando o padrão de projeto *Data Access Object* ou DAO (PATTERNS, 2010);
- **Modelo de Navegação** (*Navigation Model*): refere-se aos componentes dos pacotes de visão e controle, que juntos formam a interface com o usuário;
- **Modelo de Aplicação** (*Application Model*): modela os serviços contidos no pacote de aplicação, responsáveis pela implementação dos casos de uso e as dependências entre este pacote e outros, como controle e persistência.

A Figura 14 apresenta de forma simplificada um possível processo para o desenvolvimento de WIS usando ***FrameWeb***. Apesar de incluir sugestões para várias fases do processo de software ilustrado na figura, as contribuições do método concentram-se na fase de projeto.

Ao transcorrer do tempo, várias tecnologias e novos *frameworks* foram surgindo para construção de WIS. Martins e Souza (2015), então, adéquam o ***FrameWeb*** para facilitar futuras atualizações, ao produzir metamodelos que definem uma linguagem específica de domínio para ***FrameWeb***, incluindo nestes metamodelos pontos de extensão, que permitam novos *frameworks* sejam integrados ao método.

O fragmento do Metamodelo de Entidades representado na Figura 15 apresenta a metaclasse das entidades do domínio WIS (`DomainClass`), que pode possuir operações (`DomainMethod`) e pertence ao pacote das entidades WIS (`DomainPackage`), os quais seguem o padrão UML em relação a classes.

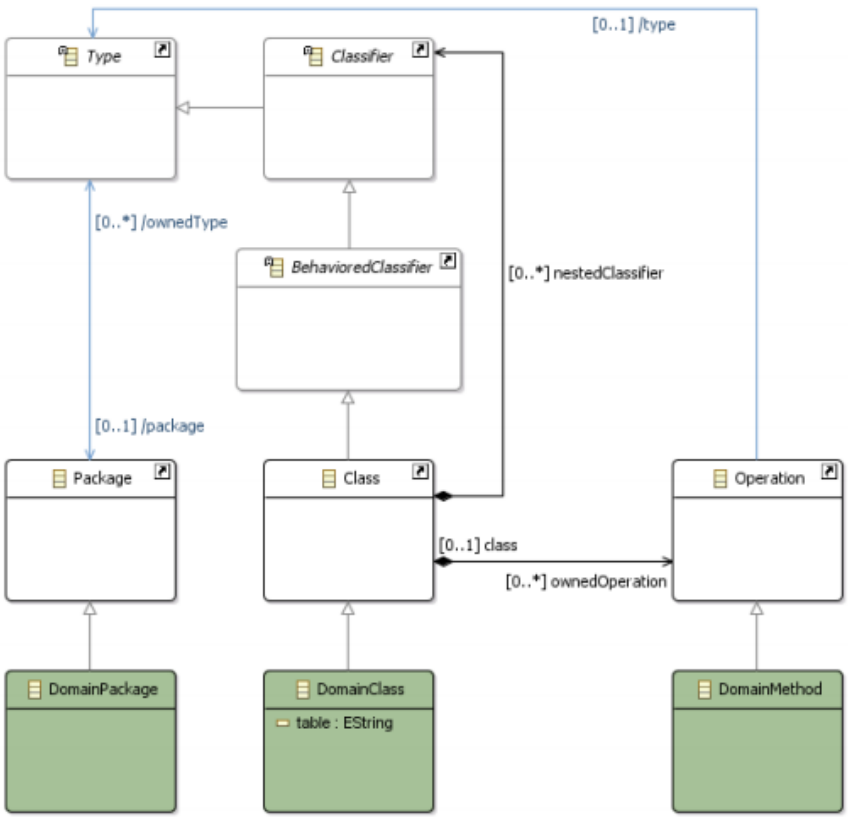

Figura 15 – Fragmento do metamodelo de Entidades: Classes (MARTINS, 2016)

No metamodelo de Entidades também temos `DomainAssociation` que em todas as suas instâncias é obrigatório que tenham pelo menos duas propriedades (`Property`) que assumam o papel de membros desta associação, sendo que essas propriedades podem ser *DomainAttribute* e *DomainProperty*, como na Figura 16. Assim como descrito no metamodelo UML, o modelo de entidades modela generalizações (`DomainGeneralization`) e conjunto de generalizações (`DomainGeneralizationSet`) entre elementos do modelo.

No Metamodelo de Entidades as restrições `DomainConstraint` são usadas nas associações entre conceitos do domínio do WIS para fornecer uma semântica específica detalhadas na Figura 17. As restrições também são aplicadas a `DomainAtribute` de forma semelhante às de associação, sendo que estas não estão detalhadas, mas foram implementadas (MARTINS, 2016).

Destacamos também que no metamodelo de Entidades do ***FrameWeb*** todos os elementos de entidades estão contidos na classe `EntityModel`, significando que todos seus elementos representam o domínio do problema.

Originalmente, os modelos ***FrameWeb*** foram criados usando qualquer editor UML, para representar o perfil UML proposto pelo método. Essa abordagem era limitada para modeladores, o fato de que os componentes do modelo não estão relacionados com uma sintaxe específica proposta pelo método, impede o processamento e interpretação por

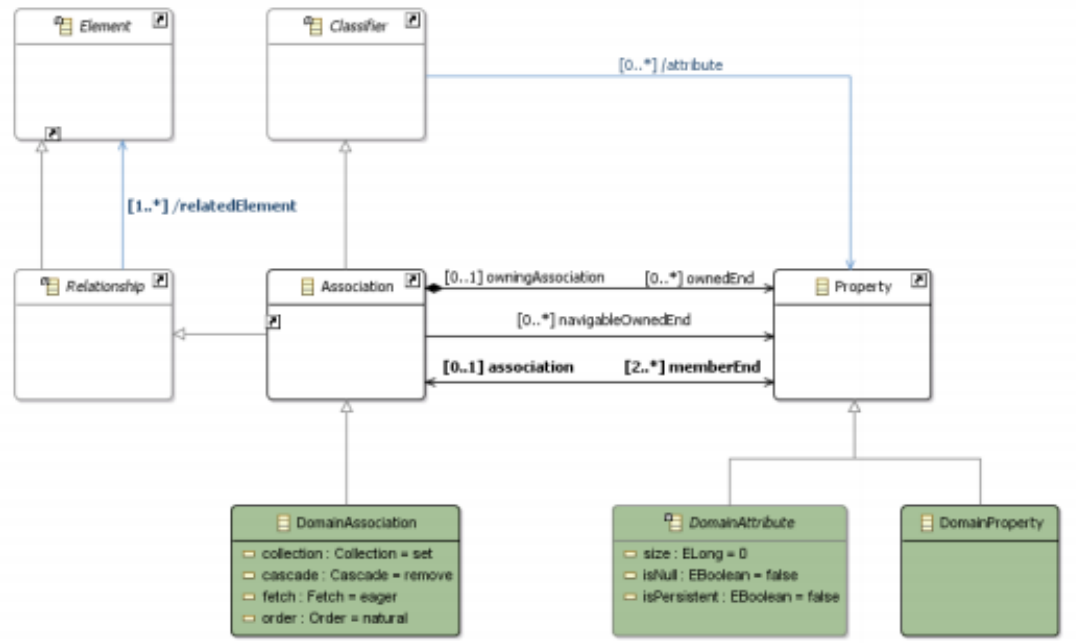

Figura 16 – Fragmento do metamodelo de Entidades: Associações (MARTINS, 2016)

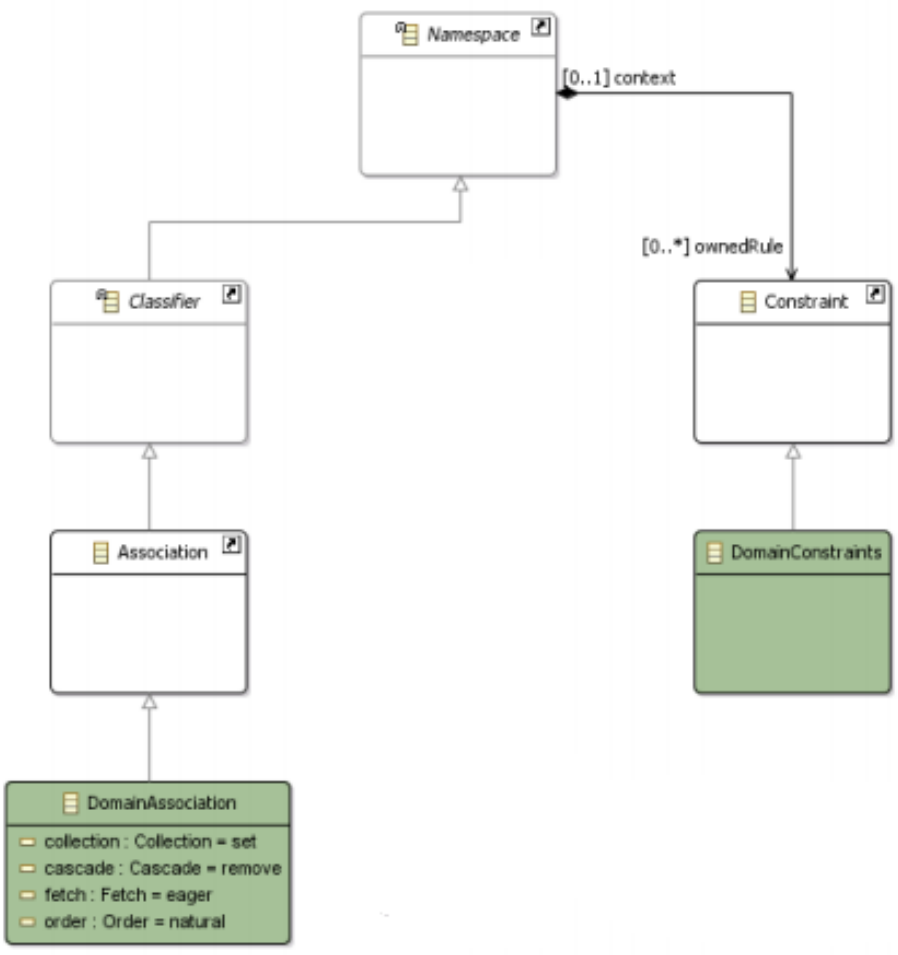

Figura 17 – Fragmento do metamodelo de Entidades: Restrições (MARTINS, 2016)

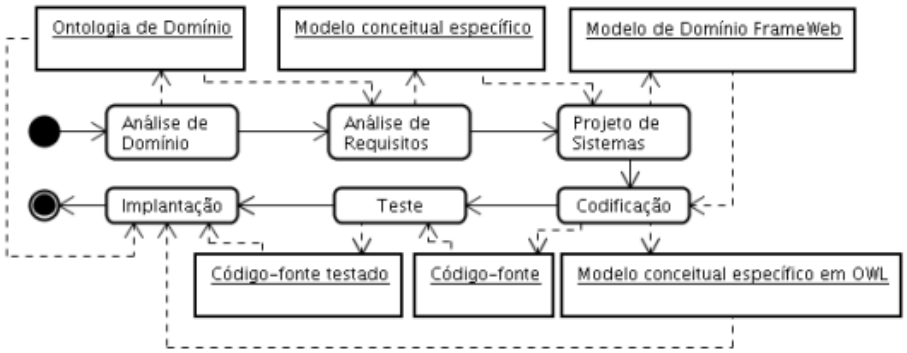

Figura 18 – processo de desenvolvimento de software sugerido para o ***S-FrameWeb*** (SOUZA; FALBO, 2007)

ferramentas para fornecer recursos úteis como, por exemplo, geração de código. Usando técnicas e ferramentas de Desenvolvimento Dirigido a Modelos (*Model-Driven Development* - MDD) (PASTOR et al., 2008) e os meta-modelos ***FrameWeb*** apresentados anteriormente serviram como base para o desenvolvimento de uma ferramenta chamada *FrameWeb Editor* (CAMPOS; SOUZA, 2017), que fornece um editor gráfico para o criação de modelos válidos neste método.

Esses modelos são então usados como entrada para se gerar código em linguagem de programação para facilitar o desenvolvimento de WIS. O *FrameWeb Editor* suporta a criação dos quatro modelos básicos propostos pelo método: Entidade, Persistência, Aplicação e Navegação, além de um diagrama de nível de projeto que agrega estas quatro partes, bem como as definições de *frameworks* importados para o projeto.

3.2.1 S-FrameWeb

Para alcançar a visão da Web Semântica, WISs precisariam fornecer seus dados de modo que possam ser processáveis por máquina, conforme discutido na Seção 3.1. Neste contexto, Souza et al. (2007) propõem ***S-FrameWeb*** (*Semantic FrameWeb*): extensão do ***FrameWeb*** que visa apoiar o desenvolvimento de WISs Semânticos, sugerindo diretrizes para desenvolvimento de aplicações Web com semântica associada, objetivando auxiliar desenvolvedores a produzirem anotações semânticas dinâmicas para que seus WISs possam ser interpretados por agentes de software.

S-FrameWeb propõe extensões no ***FrameWeb*** em quatro pontos demonstrados como processo de desenvolvimento na Figura 18:

1. Uma atividade de Análise de Domínio, no início do projeto gerando uma ontologia para o domínio ou reutilizando uma já existente que represente o domínio do WIS;

2. Nas atividades de Especificação de Requisitos e Análise são criados modelos conceituais do domínio, baseados na ontologia de domínio desenvolvida na Análise de

Domínio;

3. Na fase de Projeto, o Modelo de Entidades de ***FrameWeb*** recebe anotações semânticas baseadas na ontologia de domínio;

4. Na Implementação, o *framework* MVC utilizado deve ser estendido substituindo as respostas dirigidas a humanos por documentos passíveis de serem interpretados por agentes de software, sendo capaz de decidir qual resposta utilizar por meio de um parâmetro na requisição HTTP.

Entretanto, ***S-FrameWeb*** não prescreve um método sistemático para construção de ontologias, sendo assim difícil controlar e mensurar a qualidade de ontologias sendo simples ou complexas. Além disso ***S-FrameWeb***, por ter sido proposto em 2007, acaba propondo o uso de linguagens e tecnologias já ultrapassadas como ODM (*Ontology Definition MetaModel*), o que nos motivou a propor uma revisão e atualização do ***S-FrameWeb***, que rebatizamos de ***FrameWeb-LD***.

3.3 Trabalhos Relacionados

Existem diversos trabalhos sobre dados ligados (*Linked Data* - LD) na Web Semântica. O nosso interesse, no entanto, é em abordagens que, como a nossa, se propõem a integrar dados de WIS na Web de Dados, ou seja, conectá-los com vocabulários conhecidos de dados ligados.

3.3.1 Hera

Hera (HOUBEN et al., 2003) é um método de *design* para WISs semânticos. É focado em sistemas de informação que usam tecnologias da Internet para recuperar informações de diferentes fontes na Web e entregá-las aos usuários com base nas suas preferências. Proporciona uma arquitetura com três camadas:

- **Semântica:** especifica o conteúdo dos dados em termos de um modelo conceitual;
- **Aplicação:** especifica a visualização hipermídia, representa estruturas de navegação e adaptações de usuários;
- **Apresentação:** especifica detalhes necessários para produzir a visão, para se comunicar com usuário.

A arquitetura está detalhada na Figura 19. Embora temos pontos convergentes, baseamos o domínio em ontologias bem fundamentadas para descrever o modelo conceitual do WIS, enquanto os modelos conceituais da Hera são baseados em ontologias operacionais (OWL, RDF(S)).

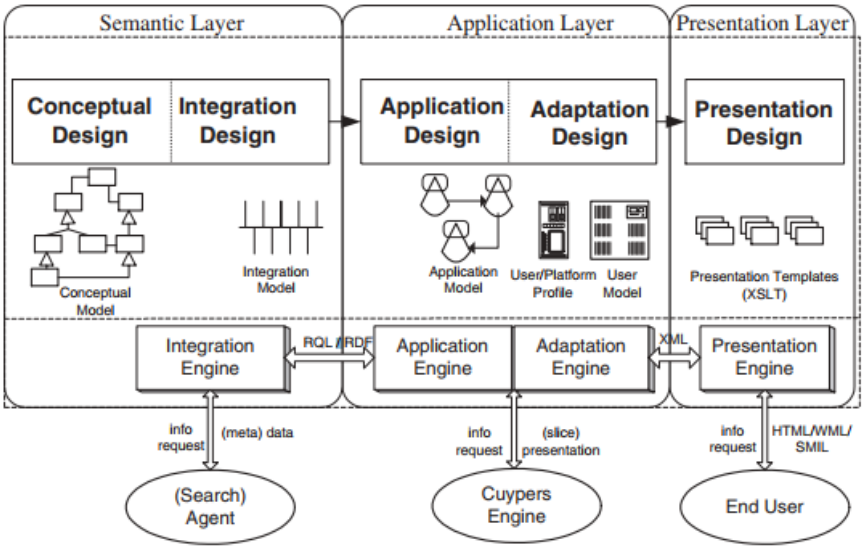

Figura 19 – Arquitetura do método Hera (HOUBEN et al., 2003).

3.3.2 OntoWeaver

O OntoWeaver (LEI; MOTTA; DOMINGUE, 2005) é uma abordagem orientada a ontologias para criar e manter aplicativos Web personalizados, ou seja, cujos conteúdos são apresentados de acordo com a necessidade e preferências de seus usuários, bem como o tipo de dispositivo que está sendo usado para acessar o aplicativo.

A natureza declarativa da especificação da aplicação da Web permite que um *designer* gerencie e mantenha no nível conceitual os aplicativos. O modelo de conhecimento interno do OntoWeaver é baseado em quadros (*frameworks*) e compatível com OCML[11] (*Operational Conceptual Modelling Language*), linguagem focada em modelos de conhecimento. A Figura 20 apresenta detalhadamente a arquitetura do OntoWeaver apresentando artefatos, recursos humanos e bases específicas usadas na arquitetura.

Destacamos também o OntoWeaver-S (LEI; MOTTA; DOMINGUE, 2004) que fornece suporte no nível de *design* de sites que podem acessar *Web Services*. Obviamente, como OntoWeaver, OntoWeaver-S também é uma estrutura de *design* para Web, que fornece modelos explícitos para prover suporte de alto nível para o *design* de sites. OntoWeaver-S adapta a ontologia de visão do site, que descreve a navegação, estruturas e interfaces de usuário, para também descrever serviços disponíveis no site.

O modelo do OntoWeaver é baseado em *frameworks* e compatível com OCML, cujo foco reside na construção de modelos de conhecimento, enquanto que nesse projeto propomos o uso de uma linguagem ontologicamente bem fundamentada baseada na UML, a mais conhecida linguagem de modelagem para o desenvolvimento de sistemas. Facilitando a construção de modelos de domínio.

[11] <http://technologies.kmi.open.ac.uk/ocml/>

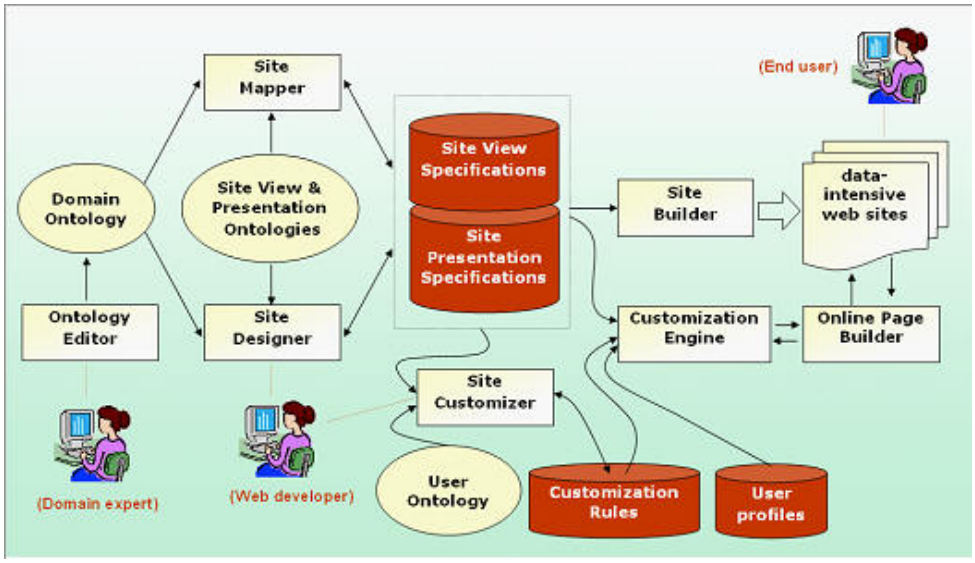

Figura 20 – Arquitetura do método OntoWeaver (LEI; MOTTA; DOMINGUE, 2005).

3.3.3 JOINT

JOINT (*Java Ontology Integrated Toolkit*) (HOLANDA et al., 2013) é um *kit* de ferramentas que suporta o desenvolvimento de aplicações baseadas em ontologias por meio da integração de tecnologias RDF e tecnologias orientadas a objeto. O JOINT propõe o uso de um *triplestore* em vez de um banco de dados relacional, gerando código que integra os dados RDF em uma aplicação Java.

JOINT usa a API Java RDF Sesame[12] e a biblioteca Java para RDF Alibaba.[13] Sesame é uma API Java de código aberto com suporte para inferência e consulta em RDFS. Oferece um grande leque de ferramentas aos desenvolvedores para aproveitar o poder do RDF e RDFS. Abibaba, por sua vez, é uma biblioteca de ferramentas para desenvolver aplicativos complexos de armazenamento RDF. É uma coleção de módulos que fornecem abstrações RDF simplificadas para acelerar o desenvolvimento e facilitar a manutenção de aplicativos.

Segue uma descrição geral das camadas da Arquitetura do JOINT, detalhada na Figura 21:

- **API do Sesame**: nessa camada se encontram os métodos necessários para conexão com as ferramentas de armazenamento de triplas RDF;
- **Operações em Repositório**: essa camada reúne serviços pertinentes à manipulação de repositórios do Sesame;
- **Módulo de Raciocínio**: esse módulo tem como função inferir novos dados em um

[12] https://www.w3.org/2001/sw/wiki/Sesame
[13] https://bitbucket.org/openrdf/alibaba

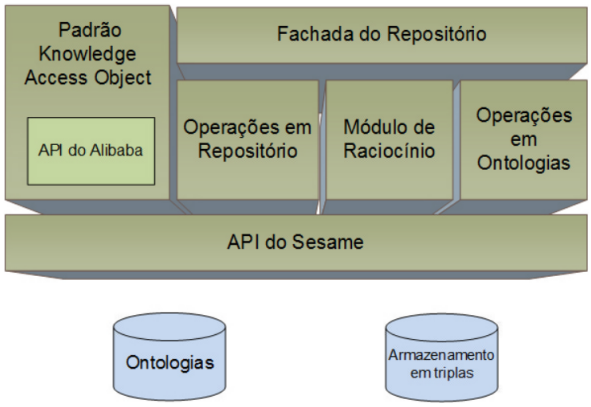

Figura 21 – Arquitetura do método JOINT.

repositório por meio da execução de regras;

- **Operações em Ontologias**: essa camada é responsável por agregar funcionalidades efetuadas em ontologia, como por exemplo: inserção e remoção de uma ontologia em um repositório e a geração de código Java a partir da API do Alibaba;

- **Padrão KAO**: o *Knowledge Object Access* (KAO) é um padrão de persistência semelhante ao *Data Access Object* (DAO), com a diferença de que o KAO não só trabalha com dados, mas trabalha com informações das ontologias. O padrão KAO tem como objetivo fornecer uma abstração do mecanismo de persistência utilizado. A grande diferença entre o KAO e o DAO, é que ao usar o padrão KAO, o desenvolvedor irá criar classes concretas KAO para cada ontologia que ele deseja acessar e manipular.

JOINT usa um repositório de ontologias e um banco de triplas. Nossa proposta mantém o banco de dados relacional, que é uma escolha popular para a arquitetura de WISs, adicionando uma camada de mapeamento Relacional para RDF, assim aumentando recursos em dados ligados.

Outras publicações também propõem métodos sistemáticos para a construção de WISs, mas com foco em preocupações específicas, como multimídia (LIMA; SCHWABE, 2003), portais semânticos (STOJANOVIC et al., 2001) ou a integração de APIs da Web (HAUSENBLAS, 2009).

3.4 Conclusão do Capítulo

Este capítulo apresentou uma visão geral da Web Semântica, revisando alguns conceitos relacionados, como ontologias, linguagens de representação, dados ligados e o método ***FrameWeb***, destacando os benefícios de ligar dados e encontrar informações na Web e também disponibilizar informações na Web em formato passível de análise por agen-

tes de software. Fundamentamos então, nossa discussão sobre este assunto, apresentando uma revisão dos conceitos e trabalhos desta área.

Apresentamos também projetos que se relacionam com o trabalho proposto nessa dissertação. A quantidade de propostas para a área de Engenharia Web com métodos relacionados a Web Semântica e dados ligados não é bastante extensa, o que demonstra que essa área ainda está bastante aberta à pesquisa científica.

Desta maneira, apresentamos nossa proposta para construção de WISs Semânticos como uma extensão do ***FrameWeb***, chamada ***FrameWeb-LD***, no Capítulo 4. ***FrameWeb-LD*** centra-se no uso de ontologias bem fundamentadas e ferramentas modernas, apoiando a publicação de dados ligados em RDF e descrevendo *Web Services* semânticos.

Nossa proposta sugere, então, uma abordagem para construção de WISs preparados para consumir e publicar dados na Web, fomentando a Web de Dados, mantendo o banco de dados relacional, que é uma escolha popular para a arquitetura de WISs, adicionando um mapeamento relacional para RDF uma camada acima dele, diferentemente das abordagens existentes (cf. Seção 3.3). Esperamos, desta forma, contribuir a longo prazo com o fomento da Web Semântica.

4 FrameWeb-LD

Nesse capítulo é apresentada a proposta principal desenvolvida nesse trabalho, uma nova extensão para ***FrameWeb***, chamada ***FrameWeb-LD*** (*FrameWeb for Linked Data*) (CELINO et al., 2016), substituindo a antiga extensão ***S-FrameWeb***. Com o objetivo de auxiliar desenvolvedores a fazerem com que seus Sistemas de Informação Web (WISs) gerem anotações dinâmicas para interpretação por agentes da Web Semântica e integrá-los a Web de Dados, propomos neste trabalho algumas extensões para o método apresentado no Capítulo 3 que auxiliam o desenvolvedor a construir WISs preparados para este novo paradigma da World Wide Web. Em resumo, as contribuições deste trabalho são:

1. Utilizar uma abordagem sistemática para a construção de ontologias, o método SABiO (seção 2.2), ontologicamente bem fundamentado na linguagem OntoUML (seção 2.1), para captura e formalização de ontologias, usando ferramentas como o OLED[1] ou Menthor,[2] que podem ajudar os analistas na construção dos modelos em OntoUML e incluem recursos para gerar as ontologias operacionais OWL;

2. Seguir as melhores práticas para *Linked Data* (LD, seção 3.1.4), usando ferramentas como D2RQ, que serve como uma camada para mapear banco de dados relacionais (banco de dados mais usado em WISs) para LD, fornecendo também uma ferramenta para a geração automática de um arquivo de mapeamento entre um banco de dados relacional e uma *TripleStore*, usado no servidor D2RQ;

3. Uso de Serviços Web semânticos (seção 3.1.5), usando uma linguagem de descrição padrão OWL-S, por meio de ferramentas como OWL-S *Editor*[3] e OWLComposer,[4] que auxiliam na descrição do serviço semântico.

A Figura 22 demonstra uma visão geral do processo de desenvolvimento proposto pelo ***FrameWeb-LD***. No fluxograma, as linhas tracejadas representam um fluxo de informações, enquanto as linhas sólidas demonstram a sequência de atividades, além das informações habituais entre as atividades sequenciais. O processo é dividido nas seguintes fases: Análise (seção 4.2), Projeto (seção 4.3), Implementação (seção 4.4), Teste e Implantação (seção 4.5). A Tabela 7 apresenta os artefatos gerados nas fases de Análise, Projeto e Implementação.

[1] <https://nemo.inf.ufes.br/projects/oled/>.
[2] <http://www.menthor.net>.
[3] <http://owlseditor.semwebcentral.org/index.shtml>.
[4] <https://github.com/FORMAS/OWL-S-Composer>.

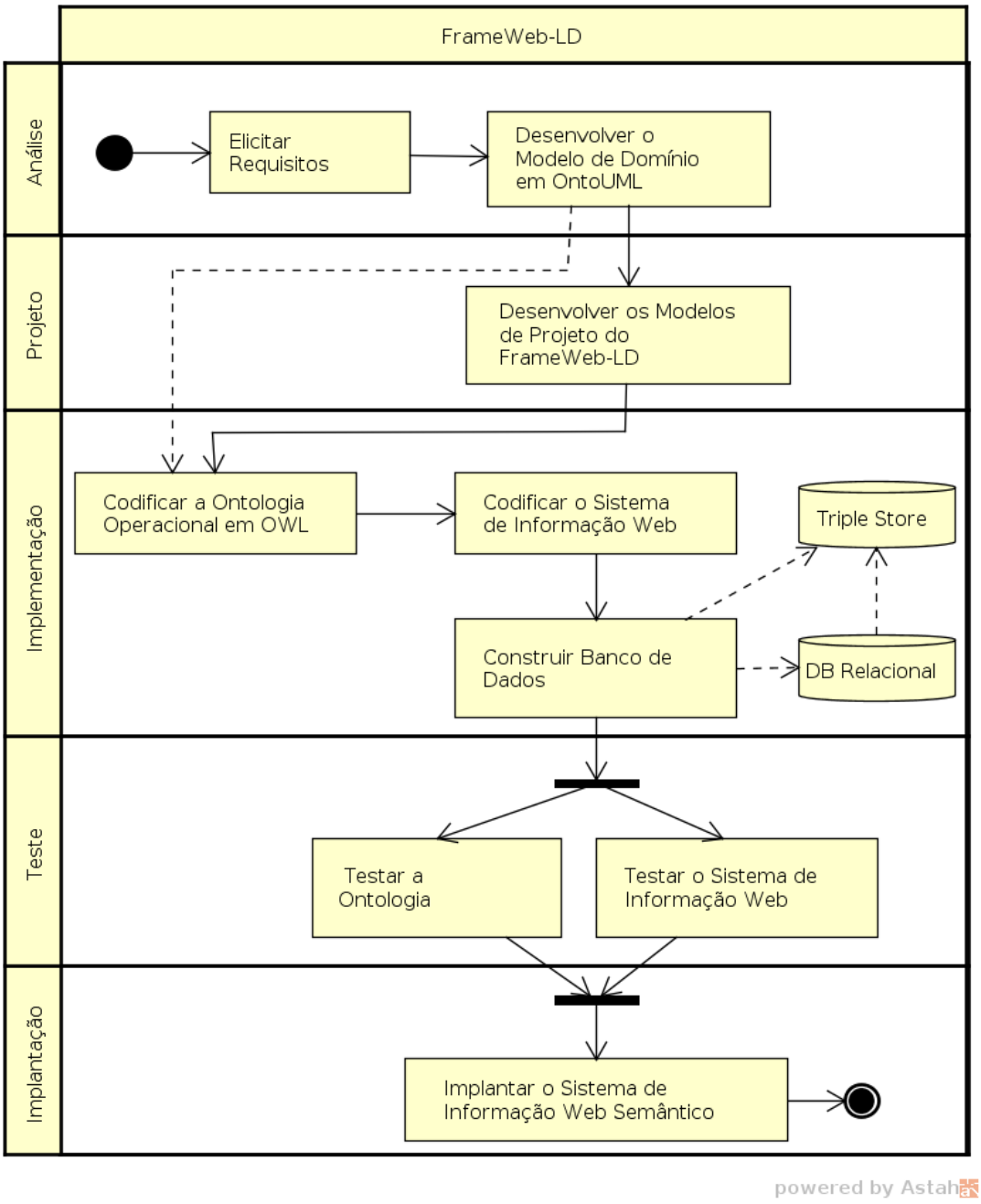

Figura 22 – Visão geral da proposta metodológica de ***FrameWeb-LD***.

Tabela 7 – Artefatos gerados de ***FrameWeb-LD***

Fase	Atefato	Descrição
Análise	Ontologia em OntoUML	Conceitos específicos do domínio, modelados em OntoUML.
Projeto	Modelo de Entidades e de Aplicação ***FrameWeb-LD***	Modelo de entidades específico, estendido do modelo OntoUML da fase anterior com adições relacionadas a mapeamento para dados ligados e modelo de aplicação para o Serviços Web Semânticos.
Implementação	Ontologia em OWL e arquivo de mapeamento *turtle*	Representação em OWL e do mapeamento de dados relacionais para dados ligados.

Destacamos também que o processo proposto não prescreve um ciclo de vida específico do desenvolvimento do WIS. Sugerimos, no entanto, o uso de métodos de ciclo de vida incrementais ou iterativos.

O recurso humano usado foi o de Desenvolvedor Web Geral para representar os atores que realizam as atividades no processo de desenvolvimento de software, podendo ser engenheiros de requisitos, arquitetos de software, programadores, entre outros, dependendo da fase em questão.

FrameWeb-LD é apresentado em detalhes neste capítulo. Antes, porém, para ilustrar seu uso, apresentamos brevemente, na Seção 4.1, o **Sistema de Credenciamento e Classificação de Docentes do Programa de Pós-Graduação em Informática (C2D)**, desenvolvido por Vettler (2016) sob nossa orientação e utilizado como exemplo nas seções seguintes.

4.1 Sistema de Credenciamento e Classificação de Docentes do PPGI (C2D)

O sistema **C2D** tem como propósito realizar as principais atividades relacionadas ao credenciamento e gerenciamento de docentes do Programa de Pós-Graduação em Informática (PPGI) da Universidade Federal do Espírito Santo (UFES), controlando pontuação de publicações necessárias para entrar e se manter no programa como docente. Programas de Pós-Graduação são avaliados pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) de acordo com sua produção técnica, ou seja, publicação de artigos científicos em veículos (conferências e periódicos) qualificados. Por esta razão, os programas de pós-graduação (PPGs) elaboram critérios para que docentes sejam credenciados no programa, se mantenham dentro do programa, alcancem diferentes níveis dentro do programa, etc. Todos estes critérios giram em torno das publicações dos professores e da

qualificação que a CAPES faz dos veículos onde estes artigos são publicados, chamada de Qualis.

O **C2D** deve considerar, inicialmente, os critérios estabelecidos pelo PPGI/UFES, mas sendo também genérico o suficiente para permitir atualização destes critérios (caso mudem no futuro), podendo, assim, também ser aproveitado por outros PPGs. Para efeito de descrição de minimundo, no entanto, consideram-se os critérios de publicação atuais do PPGI.

À época da construção do **C2D**, o PPGI definia 4 critérios: (a) ingresso no programa; (b) credenciamento e recredenciamento no programa; (c) categorização como colaborador ou permanente; e (d) habilitação para orientação no doutorado. Tais critérios envolvem vários aspectos, como: ter plano de trabalho aprovado, ter lecionado disciplinas, ter concluído orientações, ter vínculo funcional-administrativo e, por fim, atender a requisitos mínimos de publicação.

O ator que realiza diversos casos de usos citados acima do **C2D** é a *Secretaria do PPGI* como mostrado na Figura 23. O **C2D** também deve publicar os dados de sua base de dados em formatos de dados interligados do W3C (RDF/OWL), obtendo sempre que possível informações adicionais em bases de dados interligados existentes (ex.: dados de conferências na base do DBLP). Como **C2D** publica seus dados, estes estão disponíveis para consumo por um agente de software por exemplo. Na Figura 23, o ator *Agente da Web Semântica* realiza o caso de uso de consumo de dados interligados.

O desenvolvimento do **C2D** usando ***FrameWeb-LD***, detalhando a construção dos modelos e uso de ferramentas para facilitar essas tarefas é detalhado nas seções a seguir, com o objetivo de ilustrar as propostas deste trabalho.

4.2 Análise

O maior propósito de um WIS é solucionar um problema num domínio específico, como governo, serviços ou educação e deixá-lo disponível na Web para que sejam facilmente acessados. É importante se aprofundar no domínio geral antes do domínio específico. A atividade de Análise de Domínio, apresentada por Falbo, Guizzardi e Duarte (2002), cria uma ontologia para o domínio do problema completo como, por exemplo, publicações de pesquisadores associados a programas de pós-graduação em geral, para que o conhecimento aprendido possa ser reutilizado em outros domínios específicos (ex.: no PPGI/UFES).

Por isso é importante descrever formalmente o domínio no qual o sistema se encontra. ***FrameWeb-LD*** inclui no processo de desenvolvimento de WISs uma fase de Análise de Domínio, atividade que se dedica à descrição do domínio. No Capítulo 2, destacamos que atualmente para representação de domínio são utilizadas as ontologias. A atividade de

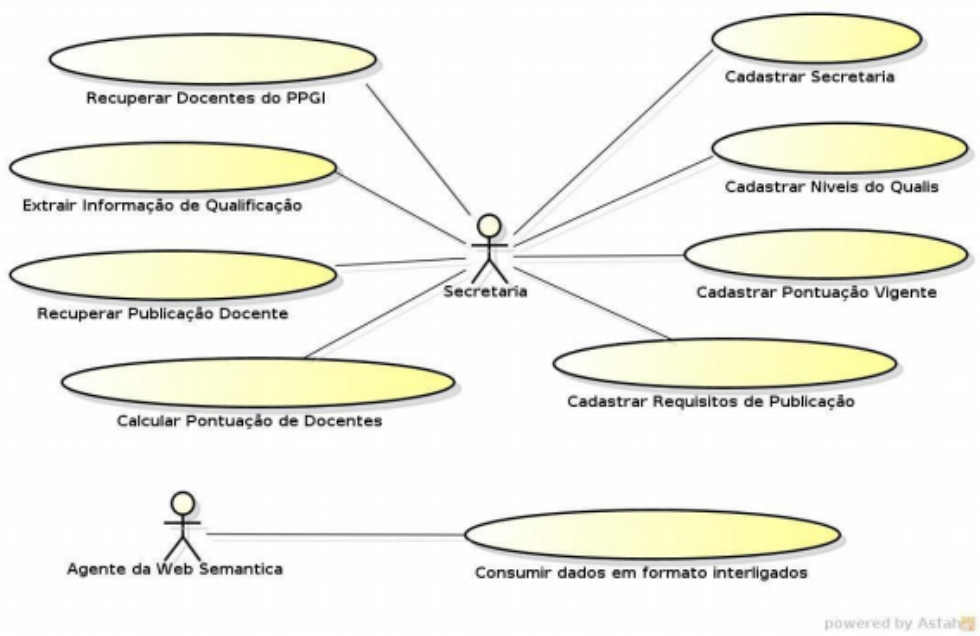

Figura 23 – Diagrama de Caso de Uso do sistema **C2D**.

Análise de Domínio produz um modelo de domínio que pode ser, portanto, uma Ontologia de Domínio ou de Aplicação (herdada de uma Ontologia de Domínio).

A análise se concentra na estrutura interna do WIS, definindo, assim, o que o WIS deve ter internamente para tratar os requisitos elicitados, tendo como objetivo, em última instância, a construção de um modelo que representa uma parte do mundo, uma abstração da realidade relevante para o WIS. Por usarmos o método SABiO durante esta fase, o analista realiza as duas primeiras etapas definidas por este processo conforme Figura 22:

1. Elicitação de requisitos (atividade de extração ou reconhecimento de requisitos);
2. Desenvolvimento do Modelo de Domínio em OntoUML.

O artefato gerado dessa fase é o modelo de domínio em OntoUML, gerando assim uma ontologia de referência. O modelo produzido em OntoUML nesta fase deve ser utilizado como base para a construção do modelos nas próximas fase do ***FrameWeb-LD*** como demonstrado na Figura 22.

Em paralelo com as atividades de Análise de Requisitos comuns (por exemplo, a captura de requisitos funcionais e não funcionais), temos a captura das Questões de Competência (QCs) proposta por SABiO (vide Seção 2.2). No caso de nosso exemplo ilustrativo, o C2D, o foco é no Programa de Pós-Graduação em Informática (PPGI). Temos, então, as seguintes QCs para este exemplo:

QC1. O que é um pesquisador para o PPGI?

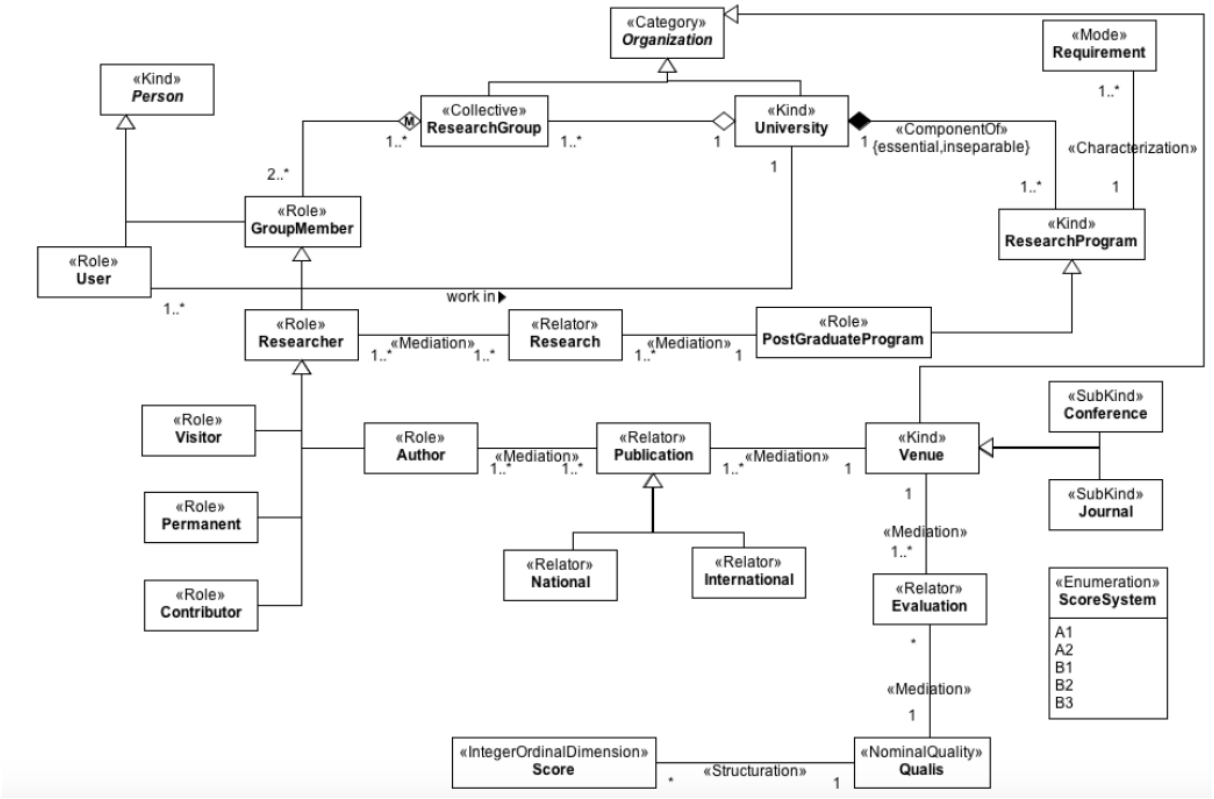

Figura 24 – Modelo conceitual em OntoUML para o C2D.

QC2. Quais os papéis possíveis de um pesquisador?

QC3. Como um Pesquisador é avaliado?

QC4. Qual a origem de uma publicação?

QC5. Quais os tipos de publicação?

QC6. Quais os tipos de veículos de publicação?

QC7. Como é o sistema de qualificação usado pelo PPGI?

QC8. Como é o sistema de pontuação do PPGI?

Essas questões de competência fornecem requisitos para uma ontologia em OntoUML para **C2D**, mostrada na Figura 24. Essa ontologia, no entanto, representa o domínio específico do PPGI/UFES.

Este modelo conceitual mostra elementos do domínio, como o programa de pós-graduação, seus pesquisadores, suas publicações e como eles são marcados no sistema de pontuação. Cada classe tem um estereótipo que determina a sua relação com os conceitos da ontologia de fundamentação UFO apresentada na Seção 2.1.

Esse modelo é usado como base para o modelo ***FrameWeb*** (com elementos adicionais, tais como atributos, enumerações, etc.), apresentado na seção a seguir.

4.3 Projeto

O maior objetivo da fase de projeto, ou *design*, é criar soluções para os problemas modelados na fase de Análise de Requisitos, adicionando tecnologia às necessidades essenciais apresentadas como requisitos pelo usuário. Nesta fase, o projetista ou *designer* deve produzir o modelos ***FrameWeb*** descritos no Capítulo 3, que são os Modelos de Entidades, de Aplicação, de Navegação e de Persistência. A extensão ***FrameWeb-LD*** propõe alterações e adições aos dois primeiros modelos na lista, o de Entidades e o de Aplicação, descritos a seguir.

4.3.1 Modelo de Entidades

O Modelo de Entidades é baseado na ontologia feita como modelo conceitual realizado na fase anterior, mas com detalhes adicionados sobre a implementação. ***FrameWeb*** adiciona anotações de mapeamento Objeto/Relacional (*Object/Relational Mapping*, ou ORM), para classes de domínio, além de outros detalhes habituais de implementação (tipos e dados de atributos, navegabilidade de associações, etc.). Seguindo o mesmo padrão, ***FrameWeb-LD*** acrescenta mapeamento para dados ligados em RDF neste modelo. O modelo resultante serve também como ontologia de especificação de projeto da fase de projeto (*design*) do método SABiO.

Os meta-modelos definidos em ***FrameWeb*** demonstrados pelas figuras 15, 16 e 17 (Subseção 3.2) foram estendidos para permitir a inclusão de anotações em RDF, que especificam como os dados do WIS se referem a vocabulários externos ao WIS e bem conhecidos da Web Semântica, com o objetivo de integrá-los à Web de Dados.

Um fragmento desse meta-modelo é mostrado na Figura 25. As classes brancas na figura vêm do meta-modelo da UML, modelo estendido pelo ***FrameWeb***, que inclui a classe `FrameWebModel`, em azul, e `DomainAttribute`, em verde. As classes restantes, em amarelo, são as extensões do meta-modelo ***FrameWeb-LD***, baseadas na especificação da sintaxe OWL 2.0. A meta-classe `VocabularyModel` possui como instâncias as classes incluídas em modelos ***FrameWeb-LD*** e que contém anotações relativas a vocabulários *Linked Data*. Esse modelo pode importar `Axioms` e `Annotations` de vocabulários externos, dada a URI de cada vocabulário. Então, o modelo é anotado via `VocabularyAssociation`, `VocabularyProperty` ou `VocabularyConstraint`.

Para realização da ferramenta que usa o meta-modelo para gerar código de mapeamento para D2RQ, foram acrescentadas as classes apresentadas na Figura 26. Seguindo o padrão do meta-modelo de Entidade apresentado na Subseção 3.2 (figuras 15, 16 e 17), acrescentamos as classes de vocabulário: `VocabularyClass`; classes de Entidade: `VocabularyEntity`, tendo como subclasses `ObjectProperty`, `DataProperty` e `Annotation`, que são também classes de associação, i.e., derivam de `VocabularyAssociation`,

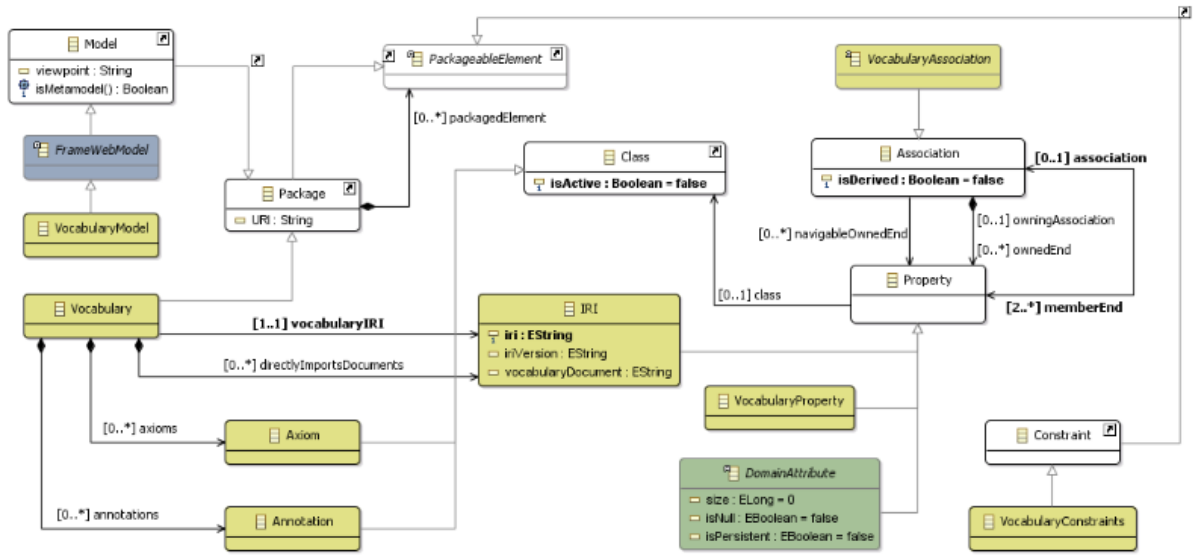

Figura 25 – Meta-modelo proposto ***FrameWeb-LD***.

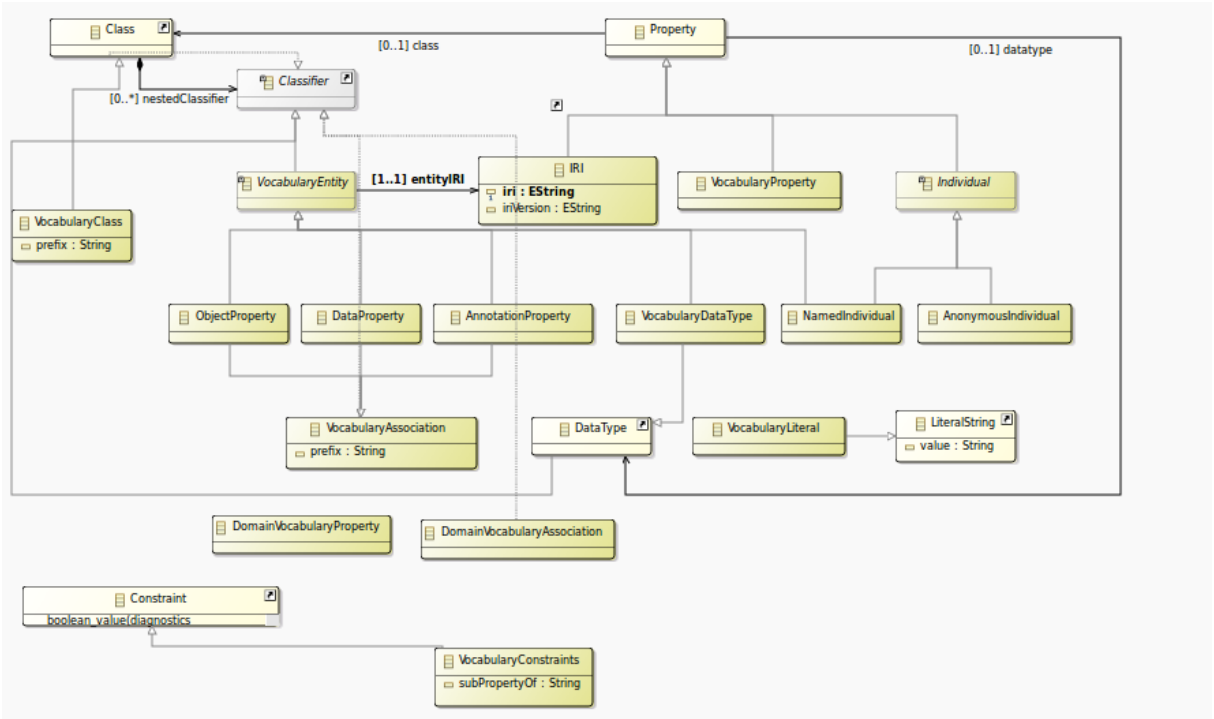

Figura 26 – Meta-modelo de Domínio ***FrameWeb-LD***.

além de outras duas subclasses: `VocabularyDataType`, para tipos de dados de vocabulários e classes de OWL 2.0, e `NamedIndividual`, que é uma classe usada para declarar o nome (em contraste com o anonimato) individual. Para declarar indivíduos anônimos, utiliza-se `AnonymousIndividual`. Adicionamos também `VocabularyLiteral` para adição de literais nos modelos.

A Figura 27 ilustra as extensões ao meta-modelo propostas acima. A figura apresenta um Modelo de Entidades para o **C2D**, construído no *FrameWeb Editor*, ferramenta CASE

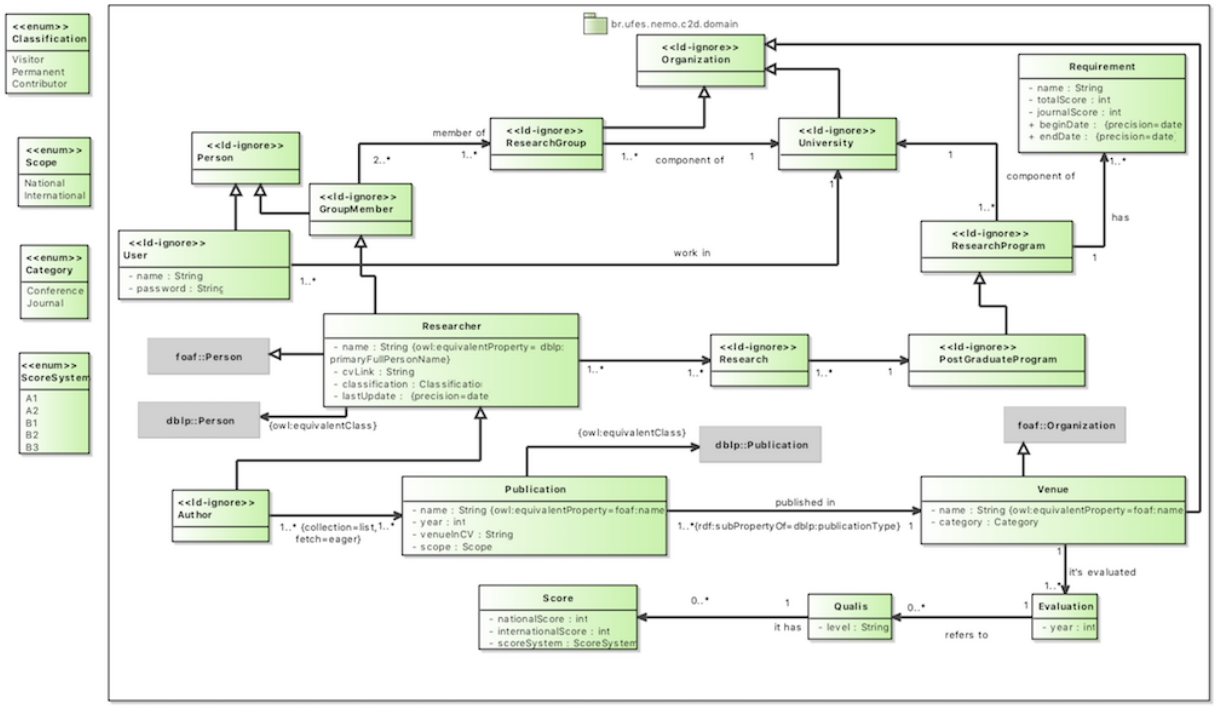

Figura 27 – Modelo de Entidades em ***FrameWeb-LD*** para o C2D.

baseada nos meta-modelos ***FrameWeb*** e ***FrameWeb-LD***, proposta por Campos e Souza (2017). São ilustradas as seguintes extensões:

1. O meta-modelo ***FrameWeb*** associa identificadores de vocabulários com as suas respectivas URIs, assim como o cabeçalho de um documento RDF faz. No exemplo do **C2D**, alguns conceitos estão sendo associados com `foaf` <http://xmlns.com/foaf/0.1/> (*friend-of-a-friend*) e com `dblp` <http://dblp.rkbexplorer.com/id/> (*DBLP Computer Science Bibliography dataset*);

2. Classes de vocabulários externos são mostradas no diagrama, usando seus IDs de vocabulário com um `namespace` UML, tal como em RDF. Tais classes podem ser relacionadas com as classes do WIS por meio de associações UML, navegáveis em direção à classe externa, representando uma tripla RDF: a classe do WIS é o sujeito, o objeto é o vocabulário externo e o predicado é especificado como uma restrição. No meta-modelo ***FrameWeb-LD*** classes de vocabulários são `VocabularyClass` (ver Figura 26). No exemplo, temos a ligação `owl:equivalentClass` (`VocabularyAssociation`) entre `Researcher` e `dblp:Person`;

3. As relações `rdfs:subClassOf` são representadas como herança UML entre uma classe do WIS e uma classe do vocabulário externo, como no exemplo: `Researcher rdfs:subClassOf foaf:Person`. No meta-modelo ***FrameWeb*** essa relação é representada por generalização de domínio (`GeneralizationDomain`);

4. Triplas relacionadas a atributos das classes do WIS também podem ser representadas utilizando restrições na forma `predicado = objeto`. No exemplo, temos que `Researcher.name` é `owl:equivalentProperty` de `dblp:primaryFullPersonName`, No meta-modelo essas triplas representam restrições de atributos, ou seja, mais um atributo da meta-classe (`DomainAttribute`) (ver Figura 16);

5. Restrições em associações das classes do nosso WIS estabelecem relações entre as propriedades do objeto (na mesma forma que restrições em atributos estabelecem relações entre as propriedades de dados). No exemplo, a associação entre a publicação (`Publication`) e o veículo de publicação (`Venue`) é feita com `rdfs:subPropertyOf dblp:publicationType`. Assim como pra atributos, essas restrições no meta-modelo são representadas por `VocabularyConstraint` (ver Figura 26);

6. Por fim, os dados de todas as classes devem ser publicados como dados ligados, a menos que o estereótipo `ld-ignore` seja usado, podendo também excluir atributos específicos de classes. No exemplo, a classe de usuário (`User`) é excluída dos dados que serão publicados juntamente com o WIS.

Seguindo a abordagem ***FrameWeb***, Modelos de Entidade em ***FrameWeb-LD*** fornecem instruções claras sobre como publicar dados ligados do nosso WIS nas próximas fases do processo.

4.3.2 Modelo de Aplicação

O modelo de Aplicação é centrado em classes que implementam serviços do WIS, que são colocados à disposição para agentes humanos (usuários) por meio de uma interface Web mediada pela estrutura MVC. Para agentes de software, entretanto, exite uma maneira mais apropriada de acessar esses serviços que são os Serviços Web Semânticos. Portanto, ***FrameWeb-LD*** propõe o estereótipo `«semanticwebservice»` para ser usado em classes de aplicação como um todo ou apenas alguns de seus métodos.

A Figura 28 mostra parte de um modelo de **C2D** que especifica que os métodos da classe de serviço responsável pelo cálculo das pontuações para pesquisadores do programa de pós-graduação em um determinado ano devem ser disponibilizados como Serviços Web Semânticos. Tal como acontece com o Modelo de Entidades, as informações contidas neste diagrama são utilizadas nas fases seguintes do processo para orientar a implementação do WIS.

4.4 Implementação

FrameWeb-LD propõe três atividades para esta fase: Codificar a Ontologia Operacional em OWL (o equivalente a fase Implementação em SABiO), Codificar o

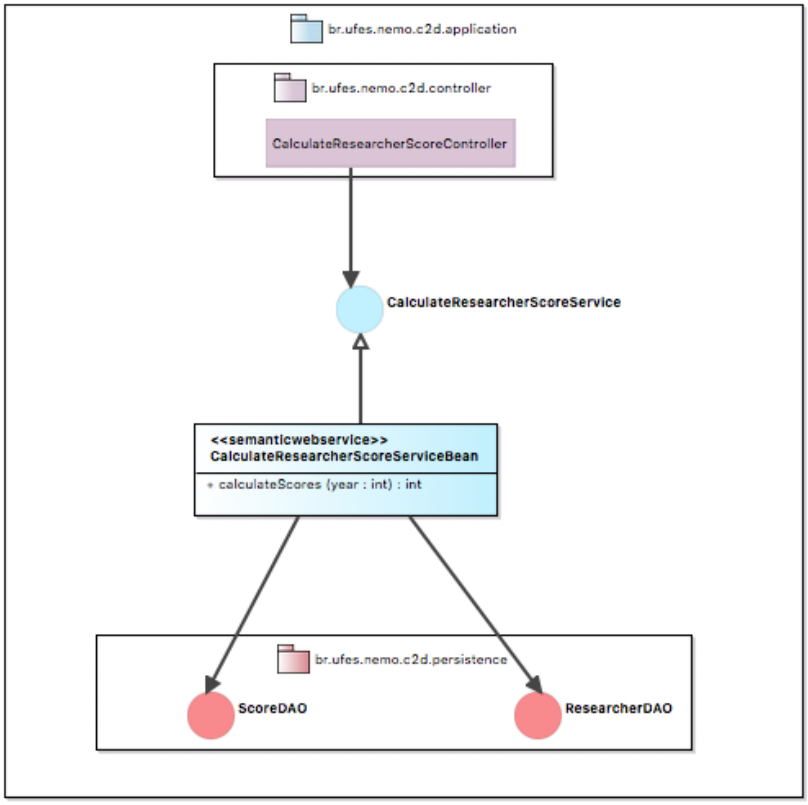

Figura 28 – Modelo de Aplicação ***FrameWeb-LD*** para o C2D.

Sistema de Informação Web e Construir Banco de Dados. A primeira atividade pode ser automatizada por ferramentas como OLED ou Menthor Editor (citados no começo desse capítulo), que podem gerar ontologias operacionais OWL a partir de modelos OntoUML.

O arquivo OWL gerado é lido como base para o vocabulário (esquema em RDF) do WIS, mas precisa ser completado com os relacionamentos com vocabulários externos, representados anteriormente como anotações no Modelo de Entidades (por exemplo, Figura 27). Para facilitar esta tarefa, construímos um protótipo de ferramenta de gerador de código chamada ReMaT (***Re****lational Database* ***Ma****pping to* ***T****riple Store*), que lê um Modelo de Entidades ***FrameWeb-LD***, completando o vocabulário lido do modelo para o WIS.

A ferramenta encontra-se disponível para instalação e uso na página do repositório do projeto, hospedada em <https://github.com/nemo-ufes/FrameWeb> com o nome de ReMaT. Um exemplo é mostrado na Listagem 4.1. Neste trecho do arquivo OWL gerado pelo Menthor Editor, ReMaT adiciona as relações entre classes `rdfs:subClassOf` e `owl:equivalentClass` representadas nas linhas 15 e 16 da Listagem 4.1, mapeando respectivamente a generalização entre `foaf::Person` e `Researcher` e a associação de equivalência entre `dblp::Person` e `Researcher`, mostradas na Figura 27.

Listagem 4.1 – Parte da ontologia operacional em OWL gerada em Menthor, com ligações a vocabulários externos adicionadas por ReMaT.

```
1
2     <owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl/c2d.owl#Researcher">
3         <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              Researcher</rdfs:label>
4         <owl:equivalentClass>
5             <owl:Class>
6                 <owl:intersectionOf rdf:parseType="Collection">
7                     <owl:Restriction>
8                         <owl:onProperty rdf:resource="http://dev.nemo.inf.ufes.br
                              /owl/c2d.owl#research"/>
9                         <owl:someValuesFrom rdf:resource="http://dev.nemo.inf.
                              ufes.br/owl/c2d.owl#Research"/>
10                    </owl:Restriction>
11                </owl:intersectionOf>
12            </owl:Class>
13        </owl:equivalentClass>
14        <rdfs:subClassOf rdf:resource="http://dev.nemo.inf.ufes.br/owl/c2d.owl#
              GroupMember"/>
15  <rdfs:subClassOf rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
16  <owl:equivalentClass rdf:resource="http://dblp.uni-trier.de/rdf/schema
        -2015-01-26#Person"/>
17    </owl:Class>
18
19
20    <owl:Class rdf:about="http://dev.nemo.inf.ufes.br/owl/c2d.owl#Publication">
21        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              Publication</rdfs:label>
22        <owl:equivalentClass>
23            <owl:Class>
24                <owl:intersectionOf rdf:parseType="Collection">
25                    <owl:Restriction>
26                        <owl:onProperty rdf:resource="http://dev.nemo.inf.ufes.br
                              /owl/c2d.owl#author"/>
27                        <owl:someValuesFrom rdf:resource="http://dev.nemo.inf.
                              ufes.br/owl/c2d.owl#Author"/>
28                    </owl:Restriction>
29                    <owl:Restriction>
30                        <owl:onProperty rdf:resource="http://dev.nemo.inf.ufes.br
                              /owl/c2d.owl#venue"/>
31                        <owl:onClass rdf:resource="http://dev.nemo.inf.ufes.br/
                              owl/c2d.owl#Venue"/>
32                        <owl:qualifiedCardinality rdf:datatype="http://www.w3.org
                              /2001/XMLSchema#nonNegativeInteger">1</
                              owl:qualifiedCardinality>
33                    </owl:Restriction>
34                </owl:intersectionOf>
35            </owl:Class>
36        </owl:equivalentClass>
37        <rdfs:subClassOf rdf:resource="http://www.menthor.net/ontouml#Relator"/>
38  <owl:equivalentClass rdf:resource="http://dblp.uni-trier.de/rdf/schema
        -2015-01-26#Publication"/>
39    </owl:Class>
```

A Listagem 4.2 apresenta um fragmento do arquivo de mapeamento D2RQ (de banco de dados relacionais para *triple store*), utilizando a sintaxe *turtle*, gerado automaticamente por ReMaT. D2RQ gera automaticamente esse arquivo baseado em um banco de dados relacional, mas sem relacionamentos com outros vocabulários. ReMaT, porém, com base nos modelos ***FrameWeb-LD***, gera o arquivo de mapeamento com ligações com outros vocabulários, como é mostrado nas linhas 22 a 49, a ligação `rdfs:subClassOf` com *foaf:Person* e a ligação `owl:equivalentClass` com *dblp:Person*.

Listagem 4.2 – Fragmento do arquivo de mapeamento gerado por ReMaT para D2RQ.

```
1  @prefix map: </Users/danillo.celino/git/ReMaT/rematgenerator-master/remat.
       generator/mapping-c2d.ttl#> .
2  @prefix db: <> .
3  @prefix c2d: <http://dev.nemo.inf.ufes.br/owl/c2d.owl#> .
4  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
5  @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
6  @prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
7  @prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
8  @prefix jdbc: <http://d2rq.org/terms/jdbc/> .
9  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
10 @prefix dblp: <http://dblp.uni-trier.de/rdf/schema-2015-01-26#> .
11 @prefix owl: <http://www.w3.org/2002/07/owl#> .
12
13 map:database a d2rq:Database;
14   d2rq:jdbcDriver "com.mysql.jdbc.Driver";
15   d2rq:jdbcDSN "jdbc:mysql://localhost:3306/c2d";
16   d2rq:username "root";
17   d2rq:password "root";
18   jdbc:autoReconnect "true";
19   jdbc:zeroDateTimeBehavior "convertToNull";
20   .
21
22 map:Researcher a d2rq:ClassMap;
23   d2rq:dataStorage map:database;
24   d2rq:uriPattern "Researcher/@@Researcher.idReseacher@@";
25   d2rq:class c2d:Researcher;
26   d2rq:classDefinitionLabel "Researcher";
27   d2rq:additionalClassDefinitionProperty map:additionalClassPropertyResearcher1;
28   d2rq:additionalProperty map:additionalPropertyResearcher1;
29   d2rq:additionalClassDefinitionProperty map:additionalClassPropertyResearcher2;
30   d2rq:additionalProperty map:additionalPropertyResearcher2;
31   .
32
33 map:additionalClassPropertyResearcher1 a d2rq:AdditionalProperty;
34   d2rq:propertyName rdfs:subClassOf;
35   d2rq:propertyValue foaf:Person;
36   .
37 map:additionalPropertyResearcher1 a d2rq:AdditionalProperty;
38   d2rq:propertyName rdfs:subClassOf;
39   d2rq:propertyValue foaf:Person;
40   .
41
42 map:additionalClassPropertyResearcher2 a d2rq:AdditionalProperty;
43   d2rq:propertyName owl:equivalentClass;
```

```
44   d2rq:propertyValue dblp:Person;
45   .
46 map:additionalPropertyResearcher2 a d2rq:AdditionalProperty;
47   d2rq:propertyName owl:equivalentClass;
48   d2rq:propertyValue dblp:Person;
49   .
50
51 map:Researcher_name a d2rq:PropertyBridge;
52   d2rq:belongsToClassMap map:Researcher;
53   d2rq:property c2d:Researcher_name;
54   d2rq:propertyDefinitionLabel "Researcher name";
55   d2rq:column "Researcher.name";
56   d2rq:additionalPropertyDefinitionProperty
         map:additionalPropertyResearcher_name1;
57   .
58 map:additionalPropertyResearcher_name1 a d2rq:AdditionalProperty;
59   d2rq:propertyName owl:equivalentProperty;
60   d2rq:propertyValue dblp:primaryFullPersonName;
61   .
62 map:Researcher_cvLink a d2rq:PropertyBridge;
63   d2rq:belongsToClassMap map:Researcher;
64   d2rq:property c2d:Researcher_cvLink;
65   d2rq:propertyDefinitionLabel "Researcher cvLink";
66   d2rq:column "Researcher.cvLink";
67   .
68 map:Researcher_Classification a d2rq:PropertyBridge;
69   d2rq:belongsToClassMap map:Researcher;
70   d2rq:property c2d:Researcher_Classification;
71   d2rq:propertyDefinitionLabel "Researcher Classification";
72   d2rq:column "Researcher.Classification";
73   .
74 map:Researcher_lastUpdateDate a d2rq:PropertyBridge;
75   d2rq:belongsToClassMap map:Researcher;
76   d2rq:property c2d:Researcher_lastUpdateDate;
77   d2rq:propertyDefinitionLabel "Researcher lastUpdateDate";
78   d2rq:column "Researcher.lastUpdateDate";
79   .
```

Destacamos também que ReMaT requisita o endereço da descrição do vocabulário, ou seja, o endereço onde se encontra o arquivo OWL gerado por Menthor. Esse endereço é colocado no início do arquivo de mapeamento, assim como outros vocabulários usados no mapeamento como mostrado na Listagem 4.2. ReMaT também necessita de informações para gerar o arquivo de mapeamento. Tais informações são demostradas na Figura 29 que representa a interface inicial do ReMaT.

A codificação do WIS é feita seguindo o método ***FrameWeb*** e tendo como base o conteúdo dos modelos propostos juntamente com os *frameworks* escolhidos para criar o aplicativo.

FrameWeb-LD propõe o uso de *Web Services* Semânticos com o WIS com base em modelos de aplicação. Destacamos também que IDEs como Eclipse podem gerar uma descrição WSDL para os serviços Web a partir dos métodos das classes de aplicação

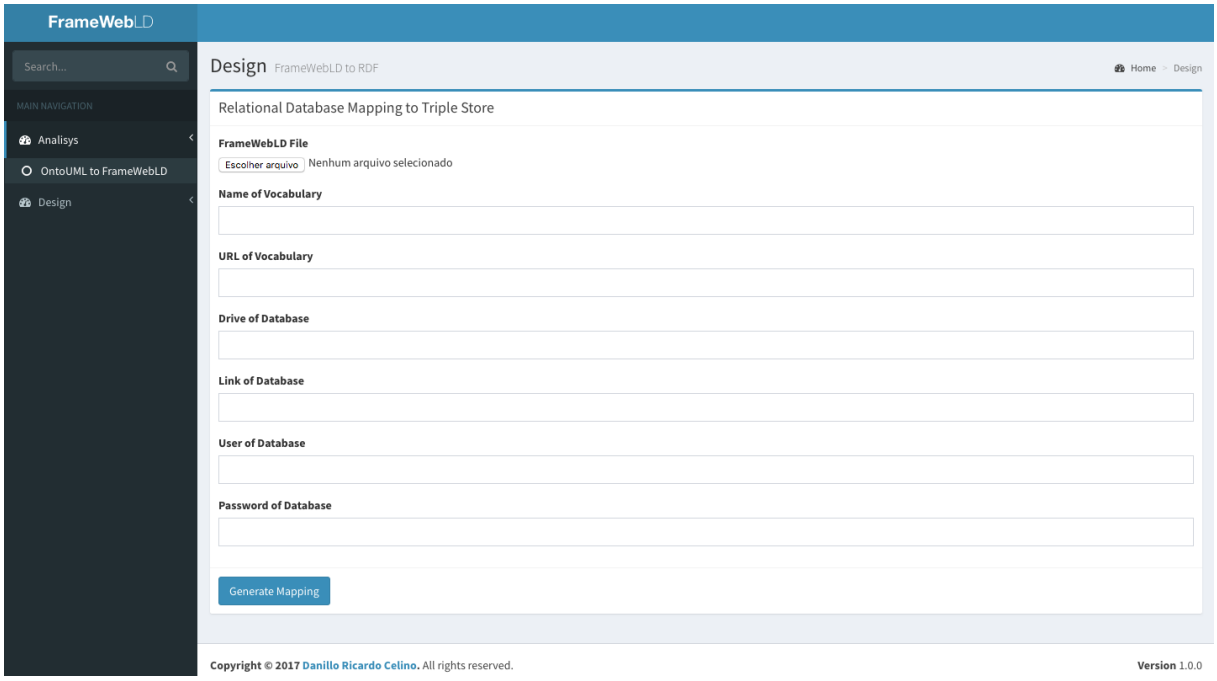

Figura 29 – Interface inicial da ferramenta ReMaT.

implementados, o que serve como arquivo de entrada para uma ferramenta que gera anotação semântica ao serviço utilizando a linguagem OWL-S (vide Seção 3.1.5), chamada OWL-S Editor, produzindo assim uma ontologia OWL-S do serviço. Um exemplo é mostrado na Listagem 4.3 sobre o serviço Web ilustrado anteriormente na Figura 28.

Listagem 4.3 – Descrição do Serviço Web Semântico gerado pelo OWL-S *Editor*

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <rdf:RDF
3    xml:base="http://dev.nemo.inf.ufes.br/owl-s/c2d/CalculateReseacherScores/
         _Service.owl#"
4    xmlns:owl="http://jamsci.servehttp.com/owlsedit/owl.rdf#"
5    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
6    xmlns:rdfs="http://jamsci.servehttp.com/owlsedit/rdf-schema.rdf#" xmlns:service
         ="http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/owls11/
         Service.owl#">
7    <owl:Ontology rdf:about="">
8      <owl:versionInfo>Version 1.0</owl:versionInfo>
9      <rdfs:comment>Service Ontology to Calculate Reseacher Scores</rdfs:comment>
10     <owl:imports rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns"/>
11     <owl:imports rdf:resource="http://jamsci.servehttp.com/owlsedit/owl.rdf"/>
12     <owl:imports rdf:resource="http://jamsci.servehttp.com/owlsedit/rdf-schema.
           rdf"/>
13     <owl:imports rdf:resource="http://staff.um.edu.mt/cabe2/supervising/
           undergraduate/owlseditFYP/owls11/Service.owl"/>
14   </owl:Ontology>
15   <service:Service rdf:ID="_Service">
16     <service:presents rdf:resource="http://dev.nemo.inf.ufes.br/owl-s/c2d/
           CalculateReseacherScores/_Profile#_Profile"/>
17     <service:describedBy rdf:resource="http://dev.nemo.inf.ufes.br/owl-s/c2d/
           CalculateReseacherScores/_ProcessModel#_ProcessModel"/>
18     <service:supports rdf:resource="http://dev.nemo.inf.ufes.br/owl-s/c2d/
```

```
        CalculateReseacherScores/_Grounding#_Grounding"/>
19    </service:Service>
20 </rdf:RDF>
```

Destacamos que um agente de software poderá ter acesso à descrição do serviço a partir desse arquivo em OWL, que entre as linhas 15 a 19 da Listagem 4.3 divide a descrição do serviço em outros 3 arquivos OWL, disponíveis no WIS.

4.5 Teste e Implantação

A fase de testes deve ser conduzida de forma a verificar a eficácia do WIS completo. A fase de teste do ***FrameWeb-LD*** consiste em testar a ontologia e o WIS propriamente dito. ***FrameWeb-LD*** não fornece contribuições específicas nesse sentido. A ontologia deve ser testada de acordo com o SABiO (validando as questões de competência, verificando a ontologia operacional, etc.) e o WIS seguindo métodos de teste apropriados de Engenharia Web / Sofware. Consideramos de grande importância, no futuro, analisar e adequar os métodos de teste já dinfudidos na literatura a ***FrameWeb-LD***.

A implantação é feita como de costume para o WIS compilando e testando o código-fonte (não diferente do ***FrameWeb***), com a adição da ontologia em OWL e do servidor D2RQ, que precisa ser executado ao lado do servidor Web que hospeda o WIS.

4.6 Conclusão do Capítulo

FrameWeb-LD busca atualizar ***S-FrameWeb***, acrescentando atividades e ferramentas para construção de WISs voltados tanto para Web Semântica e Web de Dados. Por meio do desenvolvimento de um modelo de domínio bem fundamentado construído em OntoUML, durante a fase de análise. Na fase de Projeto a criação do Modelo de Entidades e de Aplicação em ***FrameWeb-LD*** utilizando um novo perfil UML com adições referente a dados ligados e especificação das classes que proverão Serviços Web Semânticos. Por fim, na fase de Implementação, a geração da ontologia em OWL, do mapeamento relacional para triplas em RDF/*turtle* e dos Serviços Web Semânticos descritos em OWL-S.

Da mesma forma que ***S-FrameWeb*** promove agilidade no projeto e implementação de WISs semânticos, ***FrameWeb-LD*** além de trazer agilidade também na fase de análise traz benefícios aos desenvolvedores para ligar os dados específicos do WIS, descrevendo-os e ligando-os com vocabulários externos. Tais benefícios incluem:

- Atividade de desenvolvimento do modelo de Domínio em OntoUML, uma linguagem baseada na ontologia de fundamentação UFO;
- Transformação do diagrama ontológico para artefatos operacionais em OWL;

Figura 30 – Tela inicial do sistema **C2D**

- Transformação automática do modelo de entidades ***FrameWeb-LD*** em um arquivo de mapeamento de banco de dados relacionais a uma *TripleStore* (D2RQ).
- Implementação dos Serviços Web Semânticos, sendo descritos em OWL-S.

O **C2D**, utilizado durante todo o capítulo para ilustrar o método, foi o projeto piloto de ***FrameWeb-LD***, desenvolvido por uma estudante de iniciação científica para um treinamento sobre Engenharia Web e ***FrameWeb-LD*** utilizando *frameworks* que compõem a plataforma Java EE (EJB, JPA, CDI, JSF e PrimeFaces). O desenvolvimento de forma geral ocorreu de forma satisfatória, com a maioria dos módulos entregues dentro do prazo. Considerando o fato dela já conhecer a linguagem Java, mas com pouca experiência com desenvolvimento para a Web, e conhecimento de dados ligados e ontologias, consideramos o resultado bastante satisfatório.

A Figura 30 demonstra a tela inicial do sistema **C2D** com menus e apresentação do sistema e a Figura 31 apresenta a tela mais importante do sistema, a de cadastramento de pesquisadores. Finalmente, a Figura 32 apresenta dados ligados fornecidos pelo D2RQ após o mapeamento feito por ReMaT com base nos modelos ***FrameWeb-LD***.

É importante mencionar, porém, que a utilização de ***FrameWeb-LD*** para construção dos modelos de projeto apresentados neste capítulo não exige do projetista a utilização de outros métodos, modelos ou linguagens para descrever aspectos do projeto ou aspectos específicos de dados ligados. O nosso foco foi em dar uma semântica na fase de análise com OntoUML descrevendo um vocabulário e então facilitar a ligação dele com outros vocabulários na fase de projeto.

Este trabalho definiu então uma versão de ***FrameWeb***, como sendo um método para projeto WIS semânticos com dados ligados. Entretanto há muito espaço para evolução do método, realizando teste por exemplo com outros domínios. O Capítulo 5 apresenta uma avaliação do método, dos modelos e das ferramentas propostas.

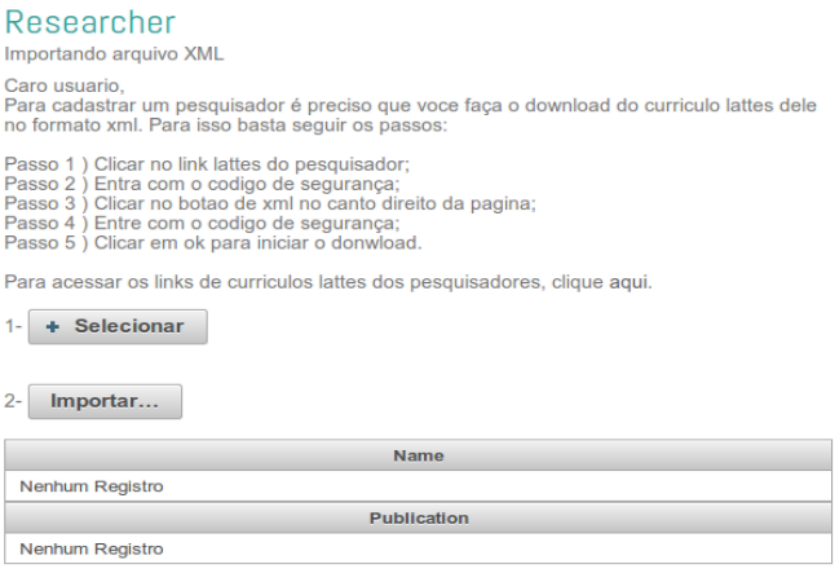

Figura 31 – Tela de cadastro de Pesquisador do sistema **C2D**

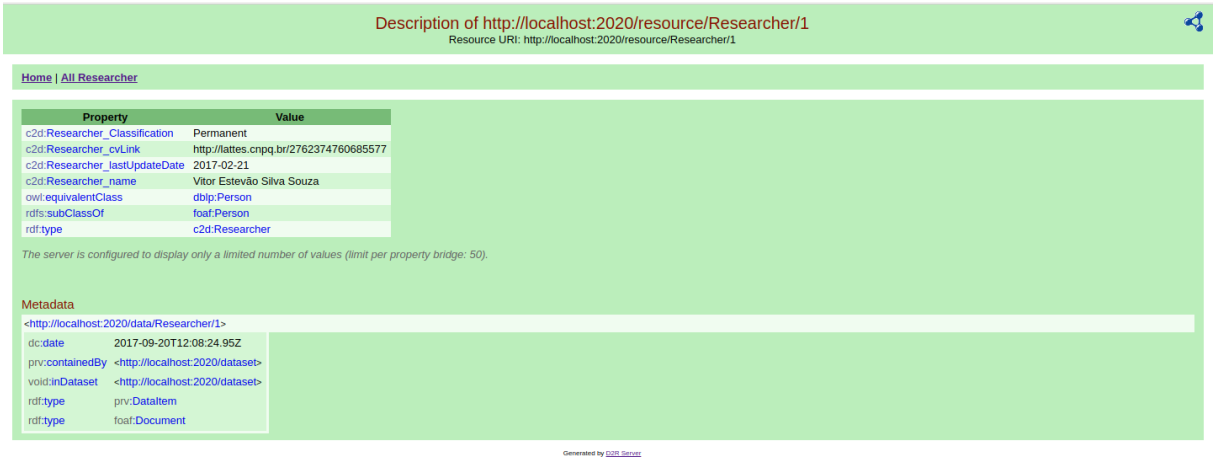

Figura 32 – Tela apresentando um Pesquisador e as ligações com outros vocabulários gerados por D2RQ

5 Avaliação

Neste capítulo, apresentamos uma avaliação preliminar do ***FrameWeb-LD*** com alunos matriculados no curso de Desenvolvimento Web e Web Semântica do Programa de Pós-Graduação em Informática da UFES. Os alunos em grupos desenvolveram WISs utilizando métodos e ferramentas para publicação ou consumo de dados ligados. No final do primeiro semestre de 2015, cada grupo produziu um relatório, documentando seus WISs com Modelos ***FrameWeb*** e ***S-FrameWeb***.

Ao todo, foram produzidos 10 relatórios, dos quais analisamos e adaptamos dois WIS documentados em ***S-FrameWeb*** aos modelos propostos para ***FrameWeb-LD*** usando, em seguida, ReMaT para inclusão das ligações com vocabulários externos na ontologia OWL e no mapeamento D2RQ.

Nesse capítulo apresentaremos 4 sistemas propostos pelos alunos:

- *Medic Appointment* um WIS para marcar consultas médicas;
- *Social Music*, WIS para cadastro de preferências musicais de usuários;
- *State Expenditures*, WIS tratando de despesas públicas estaduais;
- *Ambulance Dispatch* um WIS para despacho de ambulâncias.

A última seção descreve as conclusões obtidas na realização desses experimentos para avaliação da proposta.

5.1 Medic Appointment

Medic Appointment é um WIS sobre consultas médicas, para acompanhar pacientes, diagnósticos e prescrições. Além de ser ligado com o vabulário *FOAF* já mencionado anteriomente, *Medic Appointment* conecta ao vocabulário *MESH*[1] (*Medical Subjects Headings*) do Serviço Americano de Saúde, uma terminologia tratando de assuntos médicos. A Figura 33 apresenta o modelo de Entidades do *Medic Appointment* em ***FrameWeb-LD***.

A partir do modelo de Entidades ***FrameWeb-LD***, o ReMaT gerou automaticamente o arquivo de mapeamento entre o banco de dados relacional e uma *TripleStore* RDF que será usado por D2RQ e gera as adições no vocabulário em OWL. Para exemplificar o uso do ReMaT desse WIS usaremos a classe `Diagnostic`. A Listagem 5.1 mostra o arquivo de mapeamento da classe `Diagnostic`. Na linha 16 observamos a referência da

[1] <http://www.nlm.nih.gov/mesh/>.

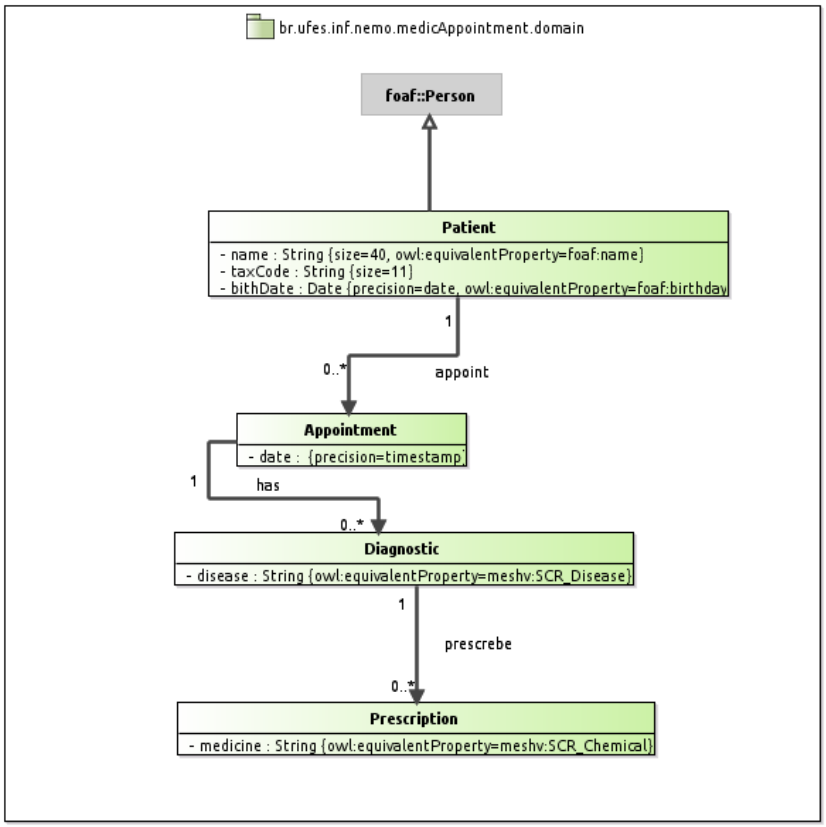

Figura 33 – Modelo de Entidades ***FrameWeb-LD*** do WIS *Medic Appointment*

adição no atributo `disease` de uma propriedade `meshv:SCR_Disease` e nas linhas 18 a 20 obtemos a efetiva adição da propriedade do vocabulário. Na Listagem 5.2, as adições feitas na ontologia em OWL no atributo `disease` são observadas na linha 8.

Listagem 5.1 – Fragmento do arquivo de mapeamento gerado por ReMaT do WIS *MedicAppointment*.

```
1 @prefix map: </Users/danillo.celino/git/ReMaT/rematgenerator-master/remat.
    generator/mapping-medic.ttl#> .
2 @prefix medic: <www.nemo.inf.ufes.br/medic> .
3 @prefix meshv: <http://id.nlm.nih.gov/mesh/vocab#>.
4
5
6 map:Diagnostic a d2rq:ClassMap;
7   d2rq:dataStorage map:database;
8   d2rq:class medic:Diagnostic;
9   d2rq:classDefinitionLabel "Diagnostic";
10  .
11 map:Diagnostic_disease a d2rq:PropertyBridge;
12   d2rq:belongsToClassMap map:Diagnostic;
13   d2rq:property medic:Diagnostic_disease;
14   d2rq:propertyDefinitionLabel "Diagnostic disease";
15   d2rq:column "Diagnostic.disease";
16   d2rq:additionalPropertyDefinitionProperty
       map:additionalPropertyDiagnostic_disease1;
17   .
```

```
18 map:additionalPropertyDiagnostic_disease1 a d2rq:AdditionalProperty;
19   d2rq:propertyName owl:equivalentProperty;
20   d2rq:propertyValue meshv:SCR_Disease;
21   .
```

Listagem 5.2 – Fragmento do arquivo da ontologia em OWL do WIS *MedicAppointment*.

```
1
2
3
4     <!-- http://www.nemo.inf.ufes.br/medicAppointment#disease -->
5
6     <owl:DatatypeProperty rdf:about="http://www.nemo.inf.ufes.br/medicAppointment
          #disease">
7         <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              disease</rdfs:label>
8   <owl:equivalentProperty rdf:resource="http://id.nlm.nih.gov/mesh/vocab#
        SCR_Disease"/>
9
10        <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/medicAppointment#
              Diagnostic"/>
11        <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
12    </owl:DatatypeProperty>
13
14
15
16    <!-- http://www.nemo.inf.ufes.br/medicAppointment#Diagnostic -->
17
18    <owl:Class rdf:about="http://www.nemo.inf.ufes.br/medicAppointment#Diagnostic
          ">
19        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              Diagnostic</rdfs:label>
20        <owl:equivalentClass>
21            <owl:Class>
22                <owl:intersectionOf rdf:parseType="Collection">
23                    <owl:Restriction>
24                        <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                              /medicAppointment#disease"/>
25                        <owl:qualifiedCardinality rdf:datatype="http://www.w3.org
                              /2001/XMLSchema#nonNegativeInteger">1</
                              owl:qualifiedCardinality>
26                        <owl:onDataRange rdf:resource="http://www.w3.org/2001/
                              XMLSchema#string"/>
27                    </owl:Restriction>
28                </owl:intersectionOf>
29            </owl:Class>
30        </owl:equivalentClass>
31        <rdfs:subClassOf rdf:resource="http://www.menthor.net/ontouml#
              FunctionalComplex"/>
32    </owl:Class>
```

5.2 Social Music

O WIS *SocialMusic* é um sistema no qual as pessoas podem declarar seus interesses musicais por artistas / bandas, seus álbuns ou faixas específicas. *SocialMusic* utilizou o vocabulário *Music Ontology*[2] uma Ontologia Musical, descrevendo conceitos sobre a música relacionado à indústria fonográfica, além do vocabulário *FOAF*. Representado na Figura 34 encontra-se o modelo de Entidades ***FrameWeb-LD***. Para exemplificar trataremos a classe `Album` e seu atributo `name`.

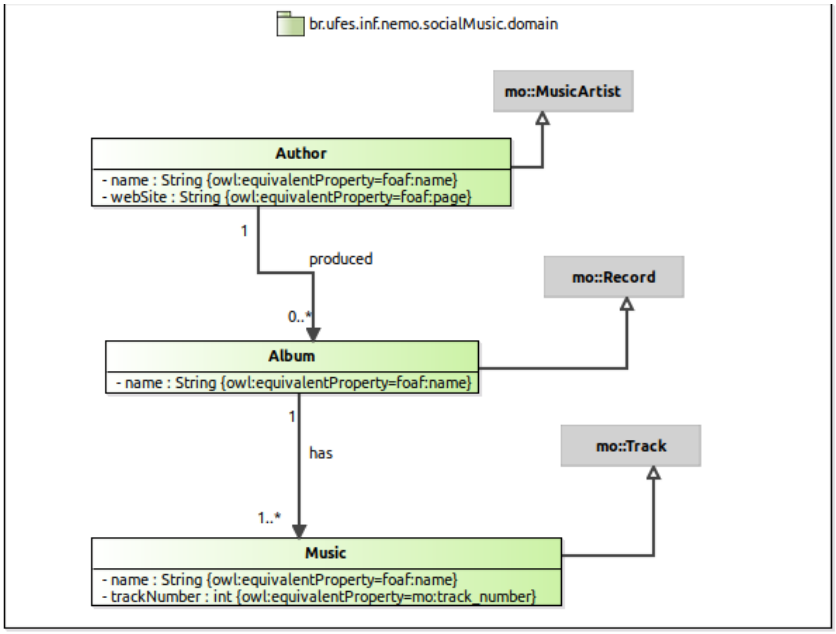

Figura 34 – Modelo de Entidades ***FrameWeb-LD*** do WIS *Social Music*

A Listagem 5.3 mostra o arquivo de mapeamento da classe `Album`. Na linha 5 observamos a referência da adição na classe, demostrando ser sub-classe de `mo:Record` e nas linhas 11 a 14 obtemos a efetiva adição da propriedade. Na Listagem 5.4, as adições feitas na ontologia em OWL na classe `Album` são observadas na linha 15, enquanto na linha 8 é apresentada a adição realizada na propriedade `name` da classe `Album`.

Listagem 5.3 – Fragmento do arquivo de mapeamento gerado por ReMaT do WIS *SocialMusic*.

```
1 @prefix map: </Users/danillo.celino/git/ReMaT/rematgenerator-master/remat.
    generator/mapping-music.ttl#> .
2 @prefix music: <www.nemo.inf.ufes.br/socilamusic> .
3 @prefix mo: <http://purl.org/ontology/mo/>.
4
5 map:Album a d2rq:ClassMap;
6   d2rq:dataStorage map:database;
7   d2rq:class music:Album;
8   d2rq:classDefinitionLabel "Album";
```

[2] <http://musicontology.com/specification/>.

```
9   d2rq:additionalPropertyDefinitionClass map:additionalPropertyAlbum1;
10  .
11 map:additionalPropertyAlbum1 a d2rq:AdditionalProperty;
12  d2rq:propertyName rdfs:subClassOf;
13  d2rq:propertyValue mo:Record;
14  .
15
16 map:Album_name a d2rq:PropertyBridge;
17  d2rq:belongsToClassMap map:Album;
18  d2rq:property music:Album_name;
19  d2rq:propertyDefinitionLabel "Album name";
20  d2rq:column "Album.name";
21  d2rq:additionalPropertyDefinitionProperty map:additionalPropertyAlbum_name1;
22  .
23 map:additionalPropertyAlbum_name1 a d2rq:AdditionalProperty;
24  d2rq:propertyName owl:equivalentProperty;
25  d2rq:propertyValue foaf:name;
26  .
```

Listagem 5.4 – Fragmento do arquivo da ontologia em OWL do WIS *SocialMusic*.

```
1
2       <!-- http://www.nemo.inf.ufes.br/music#Album_name -->
3
4     <owl:DatatypeProperty rdf:about="http://www.menthor.net/myontology#name">
5         <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">name</
              rdfs:label>
6         <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/music#Album"/>
7         <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
8   <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/#name" />
9     </owl:DatatypeProperty>
10
11
12
13    <!--http://www.nemo.inf.ufes.br/music#Album -->
14
15    <owl:Class rdf:about="http://www.nemo.inf.ufes.br/music#Album">
16        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Album<
              /rdfs:label>
17        <owl:equivalentClass>
18            <owl:Class>
19                <owl:intersectionOf rdf:parseType="Collection">
20                    <owl:Restriction>
21                        <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                              /music#Album_name"/>
22                        <owl:qualifiedCardinality rdf:datatype="http://www.w3.org
                              /2001/XMLSchema#nonNegativeInteger">1</
                              owl:qualifiedCardinality>
23                        <owl:onDataRange rdf:resource="http://www.w3.org/2001/
                              XMLSchema#string"/>
24                    </owl:Restriction>
25                </owl:intersectionOf>
26            </owl:Class>
27        </owl:equivalentClass>
28        <rdfs:subClassOf rdf:resource="http://www.menthor.net/ontouml#
              FunctionalComplex"/>
```

```
29      <rdfs:subClassOf rdf:resource="http://purl.org/ontology/mo#Record"/>
30      </owl:Class>
```

5.3 State Expenditures

O WIS *State Expenditures* é uma aplicação Web que disponibiliza as informações sobre a Despesa Pública Estadual de forma simplificada. *State Expenditures* utiliza o vocabulário *DBpedia*,[3] um projeto cujo objetivo é extrair conteúdo estruturado das informações da Wikipédia.[4] Representado na Figura 35 encontra-se o modelo de Entidades ***FrameWeb-LD*** para o WIS. Para exemplificar trataremos a classe `City` e seus atributos `name` e `cnpj`.

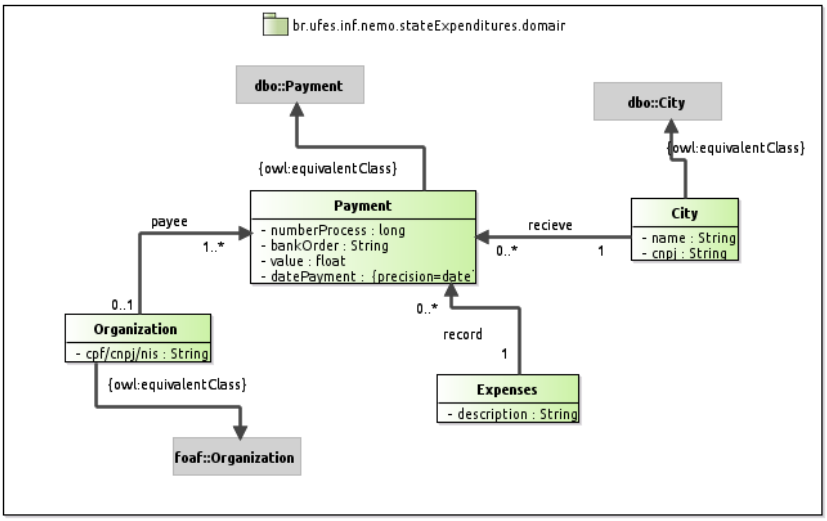

Figura 35 – Modelo de Entidades ***FrameWeb-LD*** do WIS *State Expenditures*

A Listagem 5.5 mostra o arquivo de mapeamento da classe `City`. Na linha 5 é feita a referência da adição na classe, demostrando ser subclasse de `dbo:City` e nas linhas 12 a 15 obtemos a efetiva adição da propriedade. A Listagem 5.6, as adições feitas na ontologia em OWL na classe `City` são feitas na linha 20 referente à sua herança.

Listagem 5.5 – Fragmento do arquivo de mapeamento gerado por ReMaT do WIS *State Expenditures*.

```
1 @prefix map: </Users/danillo.celino/git/ReMaT/rematgenerator-master/remat.
      generator/mapping-stateExpenditures.ttl#> .
2 @prefix stateExpenditures: <www.nemo.inf.ufes.br/stateExpenditures> .
3 @prefix dbo: <http://dbpedia.org/ontology/>.
4
5 map:City a d2rq:ClassMap;
6   d2rq:dataStorage map:database;
```

[3] <http://dbpedia.org/ontology/>.
[4] <https://www.wikipedia.org/>

```
7   d2rq:class stateExpenditures:City;
8   d2rq:classDefinitionLabel "City";
9   d2rq:additionalClassDefinitionProperty
        map:additionalClassDefinitionPropertyCity1;
10  d2rq:additionalProperty map:additionalPropertyCity1;
11  .
12 map:additionalClassDefinitionPropertyCity1 a d2rq:AdditionalProperty;
13  d2rq:propertyName rdfs:subClassOf;
14  d2rq:propertyValue dbo:City;
15  .
16 map:additionalPropertyCity1 a d2rq:AdditionalProperty;
17  d2rq:propertyName rdfs:subClassOf;
18  d2rq:propertyValue dbo:City;
19  .
20
21 map:City_name a d2rq:PropertyBridge;
22  d2rq:belongsToClassMap map:City;
23  d2rq:property stateExpenditures:City_name;
24  d2rq:propertyDefinitionLabel "City name";
25  d2rq:column "City.name";
26  .
27 map:City_cnpj a d2rq:PropertyBridge;
28  d2rq:belongsToClassMap map:City;
29  d2rq:property stateExpenditures:City_cnpj;
30  d2rq:propertyDefinitionLabel "City cnpj";
31  d2rq:column "City.cnpj";
32  .
```

Listagem 5.6 – Fragmento do arquivo da ontologia em OWL do WIS *State Expenditures*.

```
1   <!-- http://www.nemo.inf.ufes.br/statesExpendirutes#cnpj -->
2
3   <owl:DatatypeProperty rdf:about="http://www.nemo.inf.ufes.br/
        statesExpendirutes#cnpj">
4       <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">cnpj</
            rdfs:label>
5       <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/statesExpendirutes
            #City"/>
6       <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
7   </owl:DatatypeProperty>
8
9   <!-- http://www.nemo.inf.ufes.br/statesExpendirutes#name -->
10
11  <owl:DatatypeProperty rdf:about="http://www.nemo.inf.ufes.br/
        statesExpendirutes#name">
12      <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">name</
            rdfs:label>
13      <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/statesExpendirutes
            #City"/>
14      <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
15  </owl:DatatypeProperty>
16
17
18  <!-- http://www.nemo.inf.ufes.br/statesExpendirutes#City -->
19
20  <owl:Class rdf:about="http://www.nemo.inf.ufes.br/statesExpendirutes#City">
```

```
21          <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">City</
                rdfs:label>
22          <rdfs:subClassOf>
23              <owl:Class>
24                  <owl:intersectionOf rdf:parseType="Collection">
25                      <rdf:Description rdf:about="http://www.nemo.inf.ufes.br/
                            statesExpendirutes#Organization"/>
26                      <rdf:Description rdf:about="http://www.menthor.net/ontouml#
                            FunctionalComplex"/>
27                      <owl:Restriction>
28                          <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                                /statesExpendirutes#cnpj"/>
29                          <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
30                      </owl:Restriction>
31                      <owl:Restriction>
32                          <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                                /statesExpendirutes#name"/>
33                          <owl:maxCardinality rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#nonNegativeInteger">1</owl:maxCardinality>
34                      </owl:Restriction>
35                  </owl:intersectionOf>
36              </owl:Class>
37          </rdfs:subClassOf>
38      <rdfs:subClassOf rdf:resource="http://dbpedia.org/ontology#City"/>
39        </owl:Class>
```

5.4 Ambulance Dispatch

O WIS *Ambulance Dispatch* é um WIS que disponibiliza ambulâncias de forma automatizada através de um chamado. *Ambulance Dispatch* utiliza o vocabulário *LinkedGeoData*[5] é um esforço para adicionar uma dimensão espacial à Web dos dados. *LinkedGeoData* usa as informações coletadas pelo *OpenStreetMap*,[6] um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo.

Na Figura 36 encontra-se o modelo de Entidades ***FrameWeb-LD*** para a aplicação. Para exemplificar trataremos a classe `Hospital` e seus atributos `name` e `address`.

A Listagem 5.7 mostra o arquivo de mapeamento da classe `Hospital`. Na linha 6 é feita a referência, para demonstrar equivalência com a classe `lgdo:Hospital` e nas linhas 13 a 16 obtemos a efetiva adição da propriedade. Para o atributo `address` a ligação com outro vocabulário é feita nas linhas 47 a 52. Na Listagem 5.8, as adições feitas na ontologia em OWL na classe `Hospital` são feitas na linha 25 referente à propriedade `owl:equivalentClass`, e a ligação do atributo `address` ao atributo `vcard:adr` é realizada na linha 4.

[5] <http://linkedgeodata.org/ontology>
[6] <https://www.openstreetmap.org/>

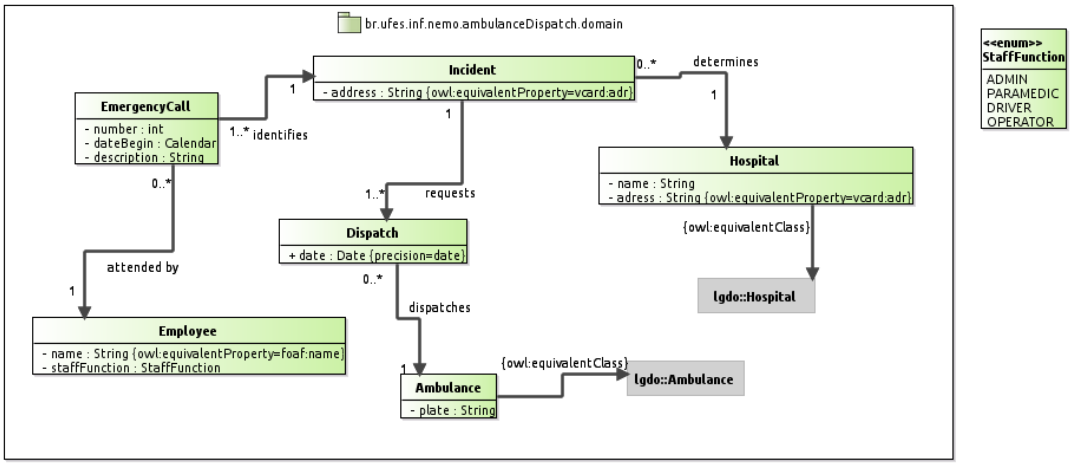

Figura 36 – Modelo de Entidades ***FrameWeb-LD*** do WIS *Ambulance Dispatch*

Listagem 5.7 – Fragmento do arquivo de mapeamento gerado por ReMaT do WIS *Ambulance Dispatch.*

```
@prefix map: </Users/danillo.celino/git/ReMaT/rematgenerator-master/remat.
    generator/mapping-ambulanceDispatch.ttl#> .
@prefix ambulanceDispatch: <www.nemo.inf.ufes.br/ambulanceDispatch> .
@prefix lgdo: <http://linkedgeodata.org/ontology/>.
@prefix vcard: <http://www.w3.org/2006/vcard/ns#>.

map:City a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:class ambulanceDispatch:Hospital;
  d2rq:classDefinitionLabel "Hospital";
  d2rq:additionalClassDefinitionProperty
      map:additionalClassDefinitionPropertyHospital1;
  d2rq:additionalProperty map:additionalPropertyHospital1;
  .
map:additionalClassDefinitionPropertyHospital1 a d2rq:AdditionalProperty;
  d2rq:propertyName owl:equivalentClass;
  d2rq:propertyValue lgdo:Hospital;
  .
map:additionalPropertyHospital1 a d2rq:AdditionalProperty;
  d2rq:propertyName owl:equivalentClass;
  d2rq:propertyValue lgdo:Hospital;
  .

map:Hospital_name a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Hospital;
  d2rq:property ambulanceDispatch:Hospital_name;
  d2rq:propertyDefinitionLabel "Hospital name";
  d2rq:column "Hospital.name";
  .
map:Hospital_address a d2rq:PropertyBridge;
  d2rq:belongsToClassMap map:Hospital;
  d2rq:property ambulanceDispatch:Hospital_address;
  d2rq:propertyDefinitionLabel "Hospital address";
  d2rq:column "Hospital.address";
  d2rq:additionalPropertyDefinitionProperty
```

```
      map:additionalPropertyDefinitionPropertyHospital_address1;
34  .
35 map:additionalPropertyDefinitionPropertyHospital_address1 a
    d2rq:AdditionalProperty;
36  d2rq:propertyName owl:equivalentProperty;
37  d2rq:propertyValue vcard:adr;
38  .
```

Listagem 5.8 – Fragmento do arquivo da ontologia em OWL do WIS *Ambulance Dispatch.*

```
1
2     <!-- http://www.nemo.inf.ufes.br/ambulanceDispatch#address -->
3
4     <owl:DatatypeProperty rdf:about="http://www.nemo.inf.ufes.br/
          ambulanceDispatch#address">
5         <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              address</rdfs:label>
6         <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/ambulanceDispatch#
              Hospital"/>
7         <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
8   <owl:equivalentProperty rdf:resource="http://www.w3.org/2006/vcard/ns#adr"/>
9     </owl:DatatypeProperty>
10
11
12
13    <!-- http://www.nemo.inf.ufes.br/ambulanceDispatch#name -->
14
15    <owl:DatatypeProperty rdf:about="http://www.nemo.inf.ufes.br/
          ambulanceDispatch#name">
16        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">name</
              rdfs:label>
17        <rdfs:domain rdf:resource="http://www.nemo.inf.ufes.br/ambulanceDispatch#
              Hospital"/>
18        <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-schema#Literal"/>
19    </owl:DatatypeProperty>
20
21
22
23    <!-- http://www.nemo.inf.ufes.br/ambulanceDispatch#Hospital -->
24
25    <owl:Class rdf:about="http://www.nemo.inf.ufes.br/ambulanceDispatch#Hospital"
          >
26        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
              Hospital</rdfs:label>
27        <owl:equivalentClass>
28            <owl:Class>
29                <owl:intersectionOf rdf:parseType="Collection">
30                    <owl:Restriction>
31                        <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                              /ambulanceDispatch#address"/>
32                        <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                              XMLSchema#nonNegativeInteger">1</owl:cardinality>
33                    </owl:Restriction>
34                    <owl:Restriction>
35                        <owl:onProperty rdf:resource="http://www.nemo.inf.ufes.br
                              /ambulanceDispatch#name"/>
```

```
36                          <owl:cardinality rdf:datatype="http://www.w3.org/2001/
                                XMLSchema#nonNegativeInteger">1</owl:cardinality>
37                      </owl:Restriction>
38                  </owl:intersectionOf>
39              </owl:Class>
40          </owl:equivalentClass>
41  <owl:equivalentClass rdf:resource="http://linkedgeodata.org/ontology#Hospital"/
        >
42          <rdfs:subClassOf>
43              <owl:Class>
44                  <owl:intersectionOf rdf:parseType="Collection">
45                      <rdf:Description rdf:about="http://www.nemo.inf.ufes.br/
                            ambulanceDispatch#Organization"/>
46                      <rdf:Description rdf:about="http://www.menthor.net/ontouml#
                            FunctionalComplex"/>
47                  </owl:intersectionOf>
48              </owl:Class>
49          </rdfs:subClassOf>
50      </owl:Class>
```

5.5 Conclusões do Capítulo

Os projetos do curso de Desenvolvimento Web e Web Semântica tem propósitos educacionais, ou seja, não são sistemas com domínios complexos. Reconhecemos, portanto, que uma abordagem maior, com abrangência de diversos domínios específicos com experimentos sistemáticos são necessários para avaliar melhor a nossa proposta. Vale salientar que não foi realizada a fase de análise do ***FrameWeb-LD***, ou seja, o modelo de OntoUML não foi realizado. Entretanto o modelo de análise proposto por ***S-FrameWeb*** foi realizado e com base nele os alunos realizaram o modelo de Entidades.

Se tratando de avaliação de sistemas, é importante salientar ainda que os modelos apresentados nesta seção foram verificados por meio do menu “validate”, da ferramenta Eclipse, ilustrado na Figura 37. A figura mostra um dos modelos criados com o *FrameWeb Editor*, porém aberto no editor padrão do *Eclipse Modeling Framework* (EMF). A partir do metamodelo de Entidades, o EMF realiza uma avaliação do modelo em questão quanto às restrições impostas pelo metamodelo.

No âmbito da Engenharia de Software, a **validação** identifica se um sistema atende aos seus requisitos originais impostos por parte do cliente, enquanto a **verificação** diz respeito à correção do modelo em relação à sintaxe e rastreabilidade a modelos anteriores. Portanto, apesar do menu chamar-se *validate* (validar), ele realiza, de fato, uma verificação.

Apesar da simplicidade do protótipo ReMaT em seus aspectos gráficos (como observado na Figura 29), a ferramenta cumpre com sua finalidade de automaticamente gerar um arquivo de mapeamento para D2RQ, facilitando a tarefa de publicação de dados e criação de *triple store* aos desenvolvedores.

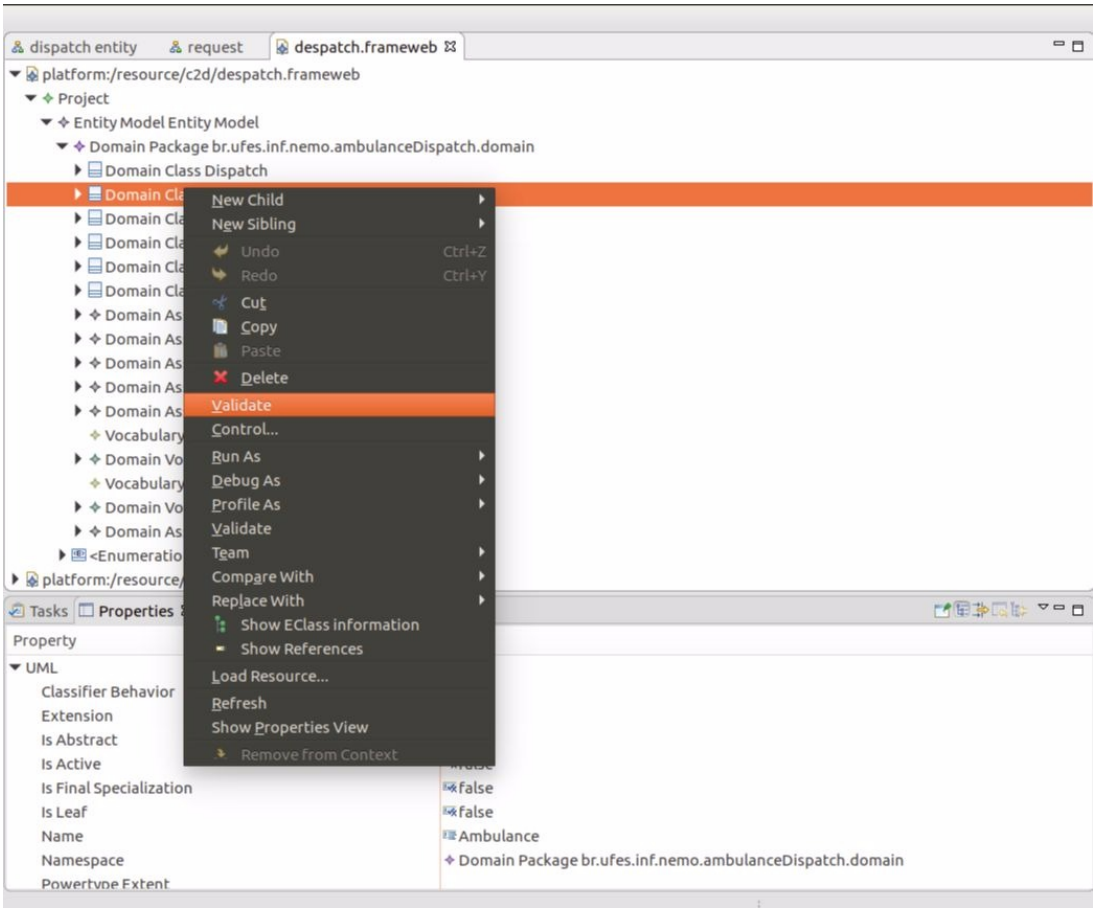

Figura 37 – Menu *validate* dos elementos do modelo ***FrameWeb***

Importante destacar, enfim, que não foram realizados questionários aos alunos da disciplina que desenvolveram os sistemas relatados neste capítulo, sendo, portanto, nossas as conclusões acima.

6 Conclusões

Neste trabalho é apresentado ***FrameWeb-LD***, uma extensão do método ***FrameWeb***, para a integração de WISs na Web Semântica e Web de Dados baseado em ***FrameWeb*** e proposto como uma evolução de ***S-FrameWeb***. Utilizando um processo sistemático e uma linguagem bem fundamentada para a construção de ontologias, a abordagem proposta gera dados ligados a outros vocabulários e serviços Web semânticos para WIS com base em modelos de *design*.

A idéia principal de ***FrameWeb-LD*** é diminuir o peso sobre os desenvolvedores para publicar dados ligados em WIS e então promover a adoção de tecnologias de dados ligados. Isso é feito tomando a especificação do esquema de dados em um nível mais alto de abstração (modelos de *design*) e automatizando a maioria das etapas necessárias para a geração de dados ligados com base nele.

Neste capítulo apresentamos uma avaliação geral do trabalho (Seção 6.1) e as perspectivas futuras (Seção 6.2).

6.1 Avaliação geral do trabalho

Pesquisas em geral tem pontos positivos e negativos. Para analisar os pontos fortes e fracos do projeto é importante revisitar as propostas reunidas nesta dissertação. São contribuições deste trabalho:

- Diretrizes gerais, passo a passo, para a construção dos modelos de projeto citados, com uso de ferramentas para auxílio do desenvolvimento, baseadas nos meta-modelos propostos;
- Utilização de modelos da linguagem de modelagem bem fundamentada OntoUML, e os utilizando como insumo para atividades da fase de projeto;
- Ferramenta de geração automática de arquivo de mapeamento entre banco de dados relacionais e *triple store* – o ReMaT;
- Procedimentos para utilização e descrição semântica de *Web Services*, incluindo modelo de projeto de aplicação pra tal.

Consideramos os seguintes pontos fortes deste trabalho:

- ***FrameWeb-LD*** fornece liberdade à equipe de desenvolvimento em relação ao

processo, deixando-a livre para utilizar o processo de software que melhor se encaixa em cada situação, propondo diretrizes com o objetivo de agilizar o processo;

- A divisão em camadas na arquitetura básica proposta por ***FrameWeb*** integrado a *frameworks*, propõe facilitar os processos de implementação e manutenção, com boas práticas de programação do WIS, sendo essa tarefa menos trabalhosa;

- A inclusão de procedimentos para criação de WISs semânticos contribui para o crescimento da Web de Dados, fornecendo descrição dos dados e facilitando a sua publicação;

- Fornece procedimentos para publicação facilitada de dados ligados a partir de banco de dados relacionais, amplamente utilizados por WISs;

- Promove o uso de *Web Services* semânticos, facilitado por uso de ferramentas de geração de *Web Services* e também de descrição desses;

- O uso da OntoUML como linguagem de desenvolvimento do modelo de domínio fundamenta a descrição do WIS, mas também gera uma ontologia de referência e ontologias operacionais que descrevem um vocabulário específico podem ser facilmente geradas por ferramentas, como OLED e Menthor;

- A utilização de ferramentas de interpretação do modelo ***FrameWeb-LD***, promove a Web de Dados facilitando uma tarefa complexa.

Nesta dissertação muitas questões não foram abordadas com profundidade, temos então pontos fracos da proposta. Sendo eles:

- Poucos experimentos foram conduzidos e nem todo o processo proposto foi exercitado nestes experimentos, sendo assim conclusões mais precisas e fortes como agilidade proporcionada não foram abordadas;

- A fase de testes e outras atividades de verificação e validação não foram aprofundadas nesse trabalho. Por isso, é de grande importância que sejam analisadas no contexto do WIS semânticos e Dados Ligados;

- A ferramenta ReMaT precisa ser desenvolvida para incluir a automação de mais etapas do processo (por exemplo, integrando a descrição do *Web Service* semântico com o vocabulário gerado em OWL para o WIS);

- A proposta é baseada no D2RQ. O método não foi testado em outras *triple stores* para testar o uso de arquivos de mapeamentos de banco de dados relacionais.

6.2 Perspectivas futuras

Esta dissertação é apenas um estopim para integração de WIS à Web Semântica e Dados Ligados, sendo facilitado com uso de ferramentas. Ao analisarmos os pontos fracos deste trabalho percebemos diversas perspectivas de projetos futuros que, abordando esses itens, complementam o presente trabalho e promovem a evolução de ***FrameWeb-LD***:

- Os modelos OntoUML atualmente incluem muitas restrições na ontologia operacional OWL gerada devido à sua natureza bem fundamentada, com o objetivo de garantir consistência. Implementar na prática tais restrições no WIS é uma tarefa muito desafiadora que precisa ser investigada;
- Discutir a atividade de testes no contexto de ***FrameWeb-LD***;
- Expandir o ReMaT para integração com ferramentas gráficas para criação dos diagramas propostos;
- Conduzir experimentos formais sobre o método para avaliar sua eficácia para organizações;
- Integrar automaticamente o vocabulário gerado pelo modelo OntoUML em OWL com ReMaT;
- Integrar ReMaT com a descrição semântica de *Web Services*.

Destacamos que o desenvolvimento de ***FrameWeb-LD*** é um trabalho contínuo, com limitações que devem ser abordadas no futuro. Também pretendemos investigar maneiras de descobrir (semi-)automaticamente os vocabulários relevantes para o desenvolvimento de um WIS, ajudando os desenvolvedores a estabelecer mais e melhores conexões entre o vocabulário do WIS e os externos da Web de Dados.

Referências

ADIDA, B. et al. Rdfa in xhtml: Syntax and processing. *Recommendation, W3C*, v. 7, 2008. Citado na página 41.

ALBUQUERQUE, A.; GUIZZARDI, G. An ontological foundation for conceptual modeling datatypes based on semantic reference spaces. In: IEEE. *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*. [S.l.], 2013. p. 1–12. Citado 3 vezes nas páginas 15, 34 e 35.

AUER, S. et al. Dbpedia: A nucleus for a web of open data. *The semantic web*, Springer, p. 722–735, 2007. Citado na página 44.

BERNERS-LEE, T. *Linked Data - Design Issues, <http://www.w3.org/DesignIssues/LinkedData.html> (last access: May 7th, 2015)*. 2006. Citado na página 23.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. *Scientific American*, v. 284, n. 5, p. 34–43, 2001. Citado 2 vezes nas páginas 23 e 40.

BIZER, C.; CYGANIAK, R. D2rq-lessons learned. In: *W3C Workshop on RDF Access to Relational Databases*. [S.l.: s.n.], 2007. p. 35. Citado na página 46.

BIZER, C.; CYGANIAK, R. *D2RQ–Accessing Relational Databases as Virtual RDF Graphs*. 2013. Citado 2 vezes nas páginas 15 e 46.

BIZER, C.; HEATH, T.; BERNERS-LEE, T. Linked data-the story so far. *Semantic services, interoperability and web applications: emerging concepts*, p. 205–227, 2009. Citado na página 44.

BORST, W. N. *Construction of engineering ontologies for knowledge sharing and reuse*. [S.l.]: Universiteit Twente, 1997. Citado na página 29.

BOX, D. et al. *Simple object access protocol (SOAP) 1.1*. 2000. Citado 3 vezes nas páginas 15, 47 e 48.

BRICKLEY, D.; GUHA, R. V.; MCBRIDE, B. Rdf schema 1.1. *W3C recommendation*, v. 25, p. 2004–2014, 2014. Citado na página 42.

BRICKLEY, D.; MILLER, L. *FOAF vocabulary specification 0.91*. [S.l.]: Tech. rep. ILRT Bristol, Nov. 2007. ur l: http://xmlns. com/foaf/spec/20071002. html, 2007. Citado na página 44.

BURSTEIN, M. et al. Owl-s: Semantic markup for web services. *W3C Member Submission*, 2004. Citado 2 vezes nas páginas 15 e 50.

CAMPOS, S. L.; SOUZA, V. E. S. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In: *XVI Workshop de Ferramentas e Aplicações, XXIII Simpósio Brasileiro de Sistemas Multimídia e Web*. [S.l.]: ACM, 2017. Citado 2 vezes nas páginas 55 e 69.

CELINO, D. R. et al. A framework-based approach for the integration of web-based

information systems on the semantic web. In: ACM. *Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web*. [S.l.], 2016. p. 231–238. Citado 2 vezes nas páginas 26 e 61.

CONSORTIUM, W. W. W. et al. Rdf 1.1 concepts and abstract syntax. World Wide Web Consortium, 2014. Citado 2 vezes nas páginas 40 e 41.

CYGANIAK, R.; HARTH, A.; HOGAN, A. N-quads: Extending n-triples with context. *W3C Recommendation*, 2008. Citado na página 41.

ERLING, O.; MIKHAILOV, I. Rdf support in the virtuoso dbms. In: *Networked Knowledge-Networked Media*. [S.l.]: Springer, 2009. p. 7–24. Citado na página 46.

FALBO, R. A. SABiO: Systematic Approach for Building Ontologies. In: GUIZZARDI, G. et al. (Ed.). *Proc. of the Proceedings of the 1st Joint Workshop ONTO.COM / ODISE on Ontologies in Conceptual Modeling and Information Systems Engineering*. [S.l.]: CEUR, 2014. Citado 6 vezes nas páginas 15, 24, 26, 30, 36 e 37.

FALBO, R. A.; GUIZZARDI, G.; DUARTE, K. C. An ontological approach to domain engineering. In: *Proc. of the 14th International Conference on Software Engineering and Knowledge Engineering*. [S.l.]: ACM, 2002. p. 351–358. Citado na página 64.

FALBO, R. D. A.; BERTOLLO, G. A software process ontology as a common vocabulary about software processes. *International Journal of Business Process Integration and Management*, Inderscience Publishers, v. 4, n. 4, p. 239–250, 2009. Citado na página 30.

FERREIRA[1], C. d. F.; MOTA[1], R. D. Comparando aplicação web service rest e soap. 2014. Citado 2 vezes nas páginas 15 e 49.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7. Citado na página 48.

FONSECA, L. B. R. da. *OntoUML Specification 1.0*. 2015. Disponível em: <http://www.ontology.com.br/ontouml/spec/>. Citado 3 vezes nas páginas 17, 32 e 33.

FOWLER, M. *Patterns of enterprise application architecture*. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2002. Citado na página 51.

GROUP, W. O. W. et al. {OWL} 2 web ontology language document overview. 2009. Citado na página 42.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, Elsevier, v. 43, n. 5-6, p. 907–928, 1995. Citado na página 29.

GUARINO, N. et al. Formal ontology and information systems. In: *Proceedings of FOIS*. [S.l.: s.n.], 1998. v. 98, n. 1998, p. 81–97. Citado 3 vezes nas páginas 15, 29 e 30.

GUIZZARDI, G. *Ontological Foundations for Structural Conceptual Models*. Tese (PhD Thesis) — University of Twente, The Netherlands, 2005. Citado 4 vezes nas páginas 24, 26, 30 e 32.

HARRIS, S.; SEABORNE, A.; PRUD'HOMMEAUX, E. Sparql 1.1 query language. *W3C*

recommendation, v. 21, n. 10, 2013. Citado na página 45.

HAUSENBLAS, M. Exploiting linked data to build web applications. *IEEE Internet Computing*, IEEE Computer Society, v. 13, n. 4, p. 68, 2009. Citado na página 59.

HEATH, T.; BIZER, C. *Linked Data: Evolving the Web into a Global Data Space*. [S.l.]: Morgan & Claypool Publishers, 2011. (Synthesis Lectures on the Semantic Web: Theory and Technology). Citado 3 vezes nas páginas 23, 25 e 44.

HITZLER, P.; KRöTZSCH, M.; RUDOLPH, S. *Foundations of Semantic Web Technologies*. [S.l.]: CRC Press, 2009. Citado na página 23.

HOLANDA, O. et al. JOINT: Java ontology integrated toolkit. *Expert Systems with Applications*, Elsevier, v. 40, n. 16, p. 6469–6477, 2013. Citado na página 58.

HOUBEN, G.-J. et al. Hera: Development of Semantic Web information systems. In: *Web Engineering*. [S.l.]: Springer, 2003. p. 529–538. Citado 3 vezes nas páginas 15, 56 e 57.

ISAAC, A.; SUMMERS, E. Skos simple knowledge organization system. *Primer, World Wide Web Consortium (W3C)*, 2009. Citado na página 44.

KOCH, N.; KRAUS, A. The expressive power of uml-based web engineering. In: CYTED. *Second International Workshop on Web-oriented Software Technology (IWWOST02)*. [S.l.], 2002. v. 16. Citado na página 24.

LEI, M. Y.; MOTTA, P. E.; DOMINGUE, D. J. Ontoweaver-s: Integrating web services into data-intensive web sites. 2004. Citado na página 57.

LEI, Y.; MOTTA, E.; DOMINGUE, J. OntoWeaver: An ontology-based approach to the design of data-intensive web sites. *Journal of Web Engineering*, Rinton Press, v. 4, n. 3, p. 244, 2005. Citado 3 vezes nas páginas 15, 57 e 58.

LEY, M. Dblp computer science bibliography. University of Trier, 2005. Citado na página 44.

LIMA, F.; SCHWABE, D. Modeling applications for the Semantic Web. In: *Web Engineering*. [S.l.]: Springer, 2003. p. 417–426. Citado na página 59.

MAEDCHE, A.; STAAB, S. Ontology learning for the semantic web. *IEEE Intelligent systems*, IEEE, v. 16, n. 2, p. 72–79, 2001. Citado na página 29.

MARTINS, B. F.; SOUZA, V. E. S. A model-driven approach for the design of web information systems based on frameworks. In: ACM. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.], 2015. p. 41–48. Citado na página 52.

MARTINS, B. F. S. *Uma abordagem dirigida a modelos para o projeto de Sistemas de Informação Web com base no método FrameWeb*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2016. Citado 3 vezes nas páginas 15, 53 e 54.

MCILRAITH, S. A.; SON, T. C.; ZENG, H. Semantic web services. *IEEE intelligent systems*, IEEE, v. 16, n. 2, p. 46–53, 2001. Citado na página 49.

NOWACK, B. *The Semantic Web Technology Stack (not a piece of cake...). Linked Data Developer*. 2009. Disponível em: <http://linkeddatadeveloper.com/Projects/Linked-Data/media/fig11.2.png.> Citado 2 vezes nas páginas 15 e 47.

PASTOR, O. et al. Model-driven development. *Informatik-Spektrum*, Springer, v. 31, n. 5, p. 394–407, 2008. Citado na página 55.

PASTOR, O.; FONS, J.; PELECHANO, V. Oows: A method to develop web applications from web-oriented conceptual models. In: *International Workshop on Web Oriented Software Technology (IWWOST)*. [S.l.: s.n.], 2003. p. 65–70. Citado na página 24.

PATTERNS, D. *Data Access Object*. [S.l.]: Retrieved, 2010. Citado na página 52.

PRUD'HOMMEAUX, E. et al. Turtle–terse rdf triple language. *Candidate Recommendation, W3C*, 2013. Citado na página 41.

QUILITZ, B.; LESER, U. Querying distributed rdf data sources with sparql. In: SPRINGER. *European Semantic Web Conference*. [S.l.], 2008. p. 524–538. Citado na página 45.

SILVA, F. B. Representation of multi-level domains on the web. 2016. Citado 3 vezes nas páginas 15, 42 e 43.

SOUZA, V. E. S.; FALBO, R. de A. Frameweb: a framework-based design method for web engineering. In: ACM. *Proceedings of the 2007 Euro American conference on Telematics and information systems*. [S.l.], 2007. p. 3. Citado 6 vezes nas páginas 15, 24, 26, 51, 52 e 55.

SOUZA, V. E. S. et al. S-FrameWeb–a Framework-based Design Method for Web Engineering with Semantic Web Support. In: *Proceedings of the International Workshop on Web Information Systems Modeling (Trondheim, Norway)*. [S.l.: s.n.], 2007. p. 55–66. Citado 4 vezes nas páginas 24, 25, 26 e 55.

SPORNY, M. et al. Json-ld 1.0. *W3C Recommendation*, v. 16, 2014. Citado na página 41.

STOJANOVIC, N. et al. SEAL: a framework for developing SEmantic PortALs. In: ACM. *Proceedings of the 1st International Conference on Knowledge Capture*. [S.l.], 2001. p. 155–162. Citado na página 59.

TAUBERER, J. What is rdf. In: *XML. com*. [S.l.: s.n.], 2006. v. 26. Citado na página 40.

VETTLER, L. *Desenvolvimento de uma Aplicação para a Web Semântica utilizando o método FrameWeb-LD*. [S.l.]: UFES, 2016. Citado na página 63.

VLISSIDES, J. et al. Design patterns: Elements of reusable object-oriented software. *Reading: Addison-Wesley*, v. 49, n. 120, p. 11, 1995. Citado na página 51.

W3C. *Web Services Architecture*. 2004. Disponível em: <https://www.w3.org/TR/ws-arch/>. Citado na página 47.

W3C. *W3C Semantic Web Frequently Asked Questions*. 2009. Disponível em: <http://www.w3.org/RDF/FAQ>. Citado 2 vezes nas páginas 39 e 40.

WEIBEL, S. et al. *Dublin core metadata for resource discovery*. [S.l.], 1998. Citado na página 44.