UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Um modelo de notações customizáveis para representação gráfica de ontologias como suporte à edição por especialistas de domínio

Wander de Almeida Limeira

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Wander de Almeida Limeira

Um modelo de notações customizáveis para representação
gráfica de ontologias como suporte à edição por
especialistas de domínio

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientador: Prof. Dr. Dilvan de Abreu Moreira

USP – São Carlos
Maio de 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

L733m Limeira, Wander
Um modelo de notações customizáveis para
representação gráfica de ontologias como suporte à
edição por especialistas de domínio / Wander Limeira;
orientador Dilvan de Abreu Moreira. -- São Carlos,
2021.
120 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2021.

1. Editor de Ontologias. 2. Notações
Customizáveis. 3. Ontology Web Language. 4.
Especialistas de Domínio. 5. Sistemas de Apoio à
Decisão. I. de Abreu Moreira, Dilvan, orient. II.
Título.

Wander de Almeida Limeira

**A customizable notation model for graphical representation
of ontologies to support editing by domain experts**

Dissertation submitted to the Instituto de Ciências
Matemáticas e de Computação – ICMC-USP – in
accordance with the requirements of the Computer
and Mathematical Sciences Graduate Program, for
the degree of Master in Science. *FINAL VERSION*

Concentration Area: Computer Science and
Computational Mathematics

Advisor: Prof. Dr. Dilvan de Abreu Moreira

USP – São Carlos
May 2021

Aos meus pais, José Brás e Maria Rute.

AGRADECIMENTOS

Devo agradecer sobretudo a Deus;

A minha família pelo exemplo, cuidado e apoio. Agradeço também pela compreensão nos momentos no qual tive que dedicar integralmente a este trabalho;

A minha companheira Lilian pelo amor e pela paciência;

Aos amigos pela ajuda e incentivo no decorrer do mestrado;

A todos os professores do ICMC. Em especial ao meu orientador Dr. Dilvan Moreira por sempre me guiar de forma exemplar, com apoio e muito conhecimento. Tenho certeza que alcancei um novo patamar em minha formação;

A todos os envolvidos que de alguma forma contribuíram para esta pesquisa, sejam da USP, Embrapa ou outras instituições. Um agradecimento especial a Dra. Katia Regina Evaristo de Jesus e também ao Sérgio Alves Torquato pelo apoio durante toda a pesquisa;

A Universidade de São Paulo pela qualidade de ensino e recursos necessários ao desenvolvimento deste trabalho;

A todos vocês, o meu muito obrigado!

*“Um livro é a prova de que os seres humanos
são capazes de fazer magia.”
(Carl Sagan)*

RESUMO

LIMEIRA, W. A. **Um modelo de notações customizáveis para representação gráfica de ontologias como suporte à edição por especialistas de domínio.** 2021. 120 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Muitos estudos têm demonstrado a necessidade de melhorar as interfaces dos editores de ontologia. Esses estudos sugerem o aprimoramento da usabilidade dessas ferramentas e a necessidade de análises adicionais dos requisitos do usuário. Isso é especialmente verdadeiro quando os usuários são especialistas em domínios não relacionados à Ciência da Computação. Neste trabalho, uma nova abordagem é proposta para a interface de editores de ontologias direcionados a especialistas de domínio. O editor gráfico resultante, chamado *Knowledge Graph Web Editor* (KGWE), usa notações visuais personalizáveis pelo usuário para apresentar os elementos visuais de maneiras mais adequadas ao domínio do especialista. O objetivo foi criar interfaces mais compreensíveis para esses usuários, explorando técnicas encontradas na literatura de Visualização de Informações e Engenharia de Usabilidade. O editor salva as ontologias na *Web Ontology Language* (OWL), um padrão do W3C. Quatro experimentos de avaliação de Interface foram conduzidos com o editor, como parte de uma colaboração com pesquisadores em agricultura (sustentabilidade) da Embrapa e da Agência Paulista de Tecnologia dos Agronegócios. Eles mostraram que as interfaces do editor e as notações customizáveis foram consideradas eficientes e eficazes pelos especialistas de domínio. Esses especialistas também consideraram o editor mais utilizável do que suas ferramentas atuais.

Palavras-chave: Editor, Ontologia, OWL, Web Semântica, SAD.

ABSTRACT

LIMEIRA, W. A. **A customizable notation model for graphical representation of ontologies to support editing by domain experts**. 2021. 120 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Many studies have demonstrated the need to improve the interfaces of ontology editors. These studies suggest the improvement of the usability of these tools and the need for further analysis of user requirements. That is especially true when the users are domain experts in fields not related to computer science. In this work, a new approach is proposed for the interface of ontology editors targeted at domain experts. The resulting graphic editor, called Knowledge Graph Web Editor (KGWE), uses user-customizable visual notations to present visual elements in ways more appropriate to the expert's domain. The goal is to create more understandable interfaces for these users, exploring techniques found in information visualization and usability engineering literature. The editor saves the ontologies in the Web Ontology Language (OWL), a W3C standard. Four interface evaluation experiments were conducted using the editor, as part of a collaboration with agriculture researchers (in sustainability) from Embrapa and Agência Paulista de Tecnologia dos Agronegócios. They showed that the editor interfaces and customizable notations were considered efficient and effective by the domain experts. These experts also judged the editor as more usable than their current tools.

Keywords: Editor, Ontology, OWL, Semantic Web, DSS.

LISTA DE ILUSTRAÇÕES

Figura 1 – Grafo RDF da tripla de um livro.	37
Figura 2 – Grafo RDF com três triplas de um mesmo recurso.	37
Figura 3 – Mapa conceitual - Método de avaliação	43
Figura 4 – Arquitetura do Decisioner	44
Figura 5 – Ontologia para produção de automóveis	50
Figura 6 – Ontologia para processos biológicos	51
Figura 7 – Paleta de cores e descrição dos gêneros	54
Figura 8 – Quantidade de filmes avaliados em comparação com o conjunto de filmes do gênero	54
Figura 9 – Modelo com avaliações dos usuários por gênero	55
Figura 10 – Estrutura de usabilidade	57
Figura 11 – Representação da ferramenta WebProtégé	60
Figura 12 – Representação da ferramenta WebVOWL	62
Figura 13 – Primitivas gráficas VOWL	62
Figura 14 – Esquema de cores VOWL	63
Figura 15 – Representação da ferramenta OWLGrEd	64
Figura 16 – Representação da ferramenta OSMoSys	64
Figura 17 – Grafo de filmes	72
Figura 18 – Definições de cores sobre os filmes	73
Figura 19 – Óscares por filme	74
Figura 20 – Modelo de notações	75
Figura 21 – Arquitetura do editor	76
Figura 22 – Editor gráfico	79
Figura 23 – Configurações de cores	81
Figura 24 – Regra de cor de fundo para classe	82
Figura 25 – Configurações de tamanho	83
Figura 26 – Formas e imagens personalizadas	84
Figura 27 – Hierarquia de classes	86
Figura 28 – Visualização gráfica em três dimensões	87
Figura 29 – Aba de propriedade de dados	87
Figura 30 – Tela contendo as mudanças realizadas na ontologia	88
Figura 31 – Ajuda com vídeo auxiliar	89
Figura 32 – Aba de edição JSON-LD	90

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Consulta SPARQL para recuperar classes da ontologia	91
Código-fonte 2 – Comandos para implantação do editor em servidores Linux	119

LISTA DE TABELAS

Tabela 1 – Comparação das funcionalidades dos editores	66
Tabela 2 – Requisitos Funcionais	70
Tabela 3 – Nível de satisfação no uso do KGWE	96
Tabela 4 – Nível de relevância da funcionalidade	97
Tabela 5 – Perguntas dicotômicas sobre as funcionalidades do editor	97
Tabela 6 – Avaliação heurística realizada pelo primeiro avaliador	100
Tabela 7 – Avaliação heurística realizada pelo segundo avaliador	101
Tabela 8 – Avaliação heurística realizada pelo terceiro avaliador	101
Tabela 9 – Sumarização dos problemas identificados	101
Tabela 10 – Avaliação das tarefas conduzidas durante as entrevistas	103
Tabela 11 – Avaliação de eficiência de visualização de instâncias	105

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
APTA	Agência Paulista de Tecnologia dos Agronegócios
CID-11	Classificação Internacional de Doenças
DOM	<i>Document Object Model</i>
DSL	<i>Domain Specific Language</i>
Embrapa	Empresa Brasileira de Pesquisa Agropecuária
GMP-RAM	<i>Risk Assessment Method for Genetically Modified Plants</i>
IMDb	<i>Internet Movie Database</i>
INPI	Instituto Nacional de Propriedade Industrial
JVM	<i>Máquina Virtual Java</i>
KGWE	<i>Knowledge Graph Web Editor</i>
NPM	<i>Node Package Manager</i>
OMS	Organização Mundial da Saúde
OWL	<i>Ontology Web Language</i>
RDF	<i>Resource Description Framework</i>
RDFS	<i>RDF Schema</i>
RMI	<i>Remote Method Invocation</i>
SAD	Sistema de Apoio à Decisão
SADs	Sistemas de Apoio à Decisão
SPA	<i>Single Page Application</i>
UML	<i>Unified Modeling Language</i>
W3C	<i>World Wide Web Consortium</i>
YAML	<i>YAML Ain't Markup Language</i>

SUMÁRIO

1	INTRODUÇÃO	27
1.1	Contexto	27
1.2	Justificativa e relevância	29
1.3	Problema de pesquisa	30
1.4	Hipótese de Pesquisa	30
1.5	Objetivo da pesquisa	31
1.6	Objetivos específicos	31
1.7	Método de Avaliação e Contribuições	31
1.8	Organização do Trabalho	33
2	ONTOLOGIAS E SISTEMAS DE APOIO À DECISÃO	35
2.1	Ontologias na Agricultura	36
2.2	RDF e RDFS	37
2.3	Web Ontology Language - OWL	38
2.3.1	<i>Class Expressions</i>	39
2.3.2	<i>Object/Data Property Restrictions</i>	40
2.3.3	<i>Class Expression Axioms</i>	40
2.3.4	<i>Object/Data Property Axioms</i>	40
2.3.5	<i>Assertions</i>	41
2.4	Sistemas de Apoio à Decisão	42
2.4.1	<i>Sistema de Apoio à Decisão: SustenAgro</i>	42
2.4.2	<i>Sistema de Apoio à Decisão: Nano</i>	43
2.5	Considerações Finais	44
3	VISUALIZAÇÃO DE INFORMAÇÃO E USABILIDADE	47
3.1	Visualização de ontologias	48
3.2	Métodos de visualização	51
3.2.1	<i>Node-Link with Force-Directed Layout</i>	51
3.2.2	<i>Indented List</i>	52
3.2.3	<i>3D</i>	52
3.3	Técnicas de interação	52
3.4	Uso de cores na visualização	53
3.5	Definindo um modelo de notações gráficas	53

3.6	Usabilidade	56
3.7	Considerações Finais	56
4	FERRAMENTAS DE VISUALIZAÇÃO E EDIÇÃO DE ONTOLOGIAS	59
4.1	Ferramentas	59
4.1.1	<i>WebProtégé</i>	60
4.1.2	<i>WebVOWL</i>	61
4.1.3	<i>OWLGrEd</i>	63
4.1.4	<i>OSMoSys</i>	63
4.1.5	<i>MEMO GRAPH</i>	65
4.2	Funcionalidades dos editores	65
4.3	Outras Ferramentas	66
4.3.1	<i>Yworks - yEd - Graph Editor</i>	67
4.3.2	<i>Xmind</i>	67
4.3.3	<i>CmapTools</i>	67
4.3.4	<i>VOSviewer</i>	67
4.3.5	<i>Gephi</i>	67
4.4	Considerações Finais	68
5	KNOWLEDGE GRAPH WEB EDITOR	69
5.1	Requisitos do Editor	70
5.1.1	<i>Requisitos Não Funcionais</i>	70
5.2	KGWE nos Sistemas de Apoio à Decisão	71
5.3	Notações Gráficas	72
5.3.1	<i>Modelo de Notação</i>	74
5.4	Arquitetura da Aplicação	75
5.5	Tecnologias utilizadas no desenvolvimento	77
5.6	Componentes da Interface	78
5.6.1	<i>Graph Editor</i>	79
5.6.2	<i>Visualização de Indivíduos</i>	85
5.6.3	<i>Hierarquia de Classes</i>	85
5.6.4	<i>3D Graph View</i>	85
5.6.5	<i>Hierarquia de Propriedades de Objetos</i>	86
5.6.6	<i>Hierarquia de Propriedades de Dados</i>	86
5.6.7	<i>Versionamento, publicação e recuperação</i>	87
5.6.8	<i>Ajuda</i>	88
5.6.9	<i>Editor de JSON-LD</i>	89
5.7	Armazenamento e recuperação dos dados	90
5.7.1	<i>Repositório das notações</i>	90
5.7.2	<i>Manipulação de dados com SPARQL</i>	91

5.8	Considerações Finais	92
6	AVALIAÇÃO	93
6.1	Avaliação de protótipos	93
6.1.1	<i>Protótipo de Vídeo</i>	93
6.1.2	<i>Protótipo funcional</i>	95
6.2	Estudos de caso SustenAgro e Nano	95
6.2.1	<i>Avaliação do KGWE</i>	95
6.3	Usabilidade: Avaliação Heurística	98
6.3.1	<i>Severidade</i>	99
6.3.2	<i>Participantes</i>	99
6.3.3	<i>Cenários</i>	100
6.3.4	<i>Avaliação</i>	100
6.4	Avaliação pelo método Think-Aloud	102
6.5	Avaliação da eficiência das notações customizadas	104
6.6	Considerações Finais	105
7	CONCLUSÕES E TRABALHOS FUTUROS	107
7.1	Contribuições científicas e tecnológicas	108
7.1.1	<i>Dificuldades na pesquisa</i>	108
7.1.2	<i>Dificuldades no desenvolvimento</i>	109
7.2	Trabalhos futuros	110
7.2.1	<i>Expansão da validação do uso de notações customizadas para outros domínios;</i>	110
7.2.2	<i>Uso de notações customizáveis em outros métodos de visualização não abordados neste trabalho;</i>	110
7.2.3	<i>Expansão das regras;</i>	110
7.2.4	<i>Tornar a ferramenta disponível para desenvolvimento colaborativo de ontologias;</i>	110
7.2.5	<i>Integrar a ferramenta em bases de conhecimento existentes;</i>	111
	REFERÊNCIAS	113
	APÊNDICE A HISTÓRIAS DE USUÁRIO	117
A.1	Histórias de usuário	117
	APÊNDICE B CONFIGURAÇÕES DE IMPLANTAÇÃO	119

INTRODUÇÃO

1.1 Contexto

Gruber (1993) afirma que uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada, ou seja, a ontologia descreve os conceitos e também os relacionamentos que existem entre esses conceitos. Ontologias podem ser usadas como um instrumento computacional para representação do conhecimento de diversos domínios. Nesse sentido, as ontologias têm um papel importante na modelagem de conhecimento por especialistas, oferecendo mecanismos que possibilitam, dentre outras coisas, a interoperabilidade entre diversos sistemas.

Existem diversas linguagens formais para se construir ontologias para diferentes casos de uso, mas não existia até 2004, uma linguagem padrão aberta para representar de maneira expressiva ontologias na Web. Nesse ano, o *World Wide Web Consortium* (W3C) criou a *Ontology Web Language* (OWL) como a linguagem padrão para a criação de ontologias na Web. Com ela, é possível tornar informações, antes apenas compreensíveis por humanos, em informações passíveis de interpretação por computadores.

Diversas ferramentas dão suporte à linguagem OWL, incluindo ferramentas para deduzir informações (*reasoners*), bancos de dados (*triple-stores*) e editores. O padrão OWL deu um grande impulso para a difusão do uso de ontologias na Web. Existem editores com interfaces gráficas, como o Protégé¹, que oferecem recursos para edição de ontologias nesse formato, mas eles são voltados para um público mais técnico, com conhecimentos sobre a sintaxe e semântica da linguagem.

A linguagem OWL ajudou na padronização e difusão do uso de ontologias, mas não é uma linguagem que pode ser facilmente adotada por especialistas de domínio. O ideal, para esse tipo de usuário, seria trabalhar com conceitos mais simplificados, mas num editor que salvasse

¹ <https://protege.stanford.edu/>

as ontologias criadas usando um subconjunto da linguagem OWL (para garantir compatibilidade com as ferramentas existentes).

Especialistas e não especialistas enfrentam também diversas barreiras na criação e edição de ontologias. Modelos podem ser complexos com milhares de elementos, tais como classes, instâncias, relacionamentos e axiomas, que dificultam a compreensão dessas informações por seres humanos.

Para ajudar os usuários a analisar e compreender esses modelos, técnicas da área de Visualização de Informações podem ser utilizadas. Essa área da computação aplica técnicas de computação gráfica interativas e representações gráficas manipuláveis para auxiliar na análise e compreensão de conjuntos de dados (FREITAS *et al.*, 2001). Sendo assim, a Visualização de Informação procura auxiliar usuários a compreenderem melhor as informações contidas nesses conjuntos, para que eles possam encontrar características ou padrões de interesse nesses dados.

Técnicas encontradas na Visualização de Informações são importantes instrumentos para uso em interfaces de editores de ontologias. Com o uso de tais técnicas é possível alcançar uma melhor representação visual dos conceitos apresentados nas ontologias, garantindo assim que especialistas de domínio possam usufruir de interfaces mais compreensíveis. Essas interfaces podem alcançar um público mais diversificado, abrangendo usuários sem conhecimentos em programação ou linguagens formais para definição de ontologias.

Uma revisão sistematizada da literatura realizada por Netto e Lima (2017), indicou que mais estudos em usabilidade são necessários para a área de visualização de ontologias.

Conforme apresenta Fu, Noy e Storey (2017):

Os estudos de usabilidade e de requisitos de usuários para visualização de ontologias, necessitam ser realizados e aprofundados para maior compreensão dos benefícios, da eficiência e da eficácia das técnicas de visualização, bem como para o desenvolvimento de novas funcionalidades para as ferramentas.

Netto e Lima (2017) mencionam também que a análise dos trabalhos selecionados no portfólio bibliográfico identificou alguns campos de investigação abertos ao desenvolvimento de trabalhos futuros em visualização de ontologias, sendo os três campos abaixo, os mais importantes para este trabalho:

- Visualização de ontologias complexas, com centenas de elementos;
- Ferramentas para usuários leigos em ontologias;
- Estudos de usabilidade e de requisitos de usuários.

A visualização de ontologias complexas, ou seja, ontologias com centenas ou milhares de elementos é um problema enfrentado por muitos visualizadores e editores de ontologias. Este

trabalho traz novas abordagens para a representação de ontologias complexas, como por exemplo, a visualização da ontologia em três dimensões. Já em relação a usabilidade, os requisitos de usuários são um importante instrumento para se criar interfaces com bons níveis de eficiência e eficácia. A simplificação de conceitos presentes na OWL tende também a tornar a ferramenta mais acessível aos usuários leigos em ontologias.

1.2 Justificativa e relevância

Existe uma cooperação entre a Universidade de São Paulo e a Empresa Brasileira de Pesquisa Agropecuária (Embrapa) com a participação de pesquisadores da Agência Paulista de Tecnologia dos Agronegócios (APTA) no desenvolvimento de soluções inteligentes para a agricultura. Um dos trabalhos desenvolvidos nessa parceria é o Sistema de Apoio à Decisão (SAD) SustenAgro que permite avaliar a sustentabilidade de sistemas produtivos de cana-de-açúcar na região centro-sul do Brasil (SUAREZ, 2017). O SustenAgro se baseia no uso de ontologias para representar o conhecimento dos especialistas de domínio e em uma *Domain Specific Language* (DSL) que define o comportamento do sistema. Ele usa essas informações para produzir a interface do sistema para seus usuários finais (empregados de usinas e fazendas) e para gerar avaliações de sustentabilidade baseadas nas informações fornecidas por esses usuários. Outros SADs também estão sendo desenvolvidos no momento para outras aplicações na área de sustentabilidade.

Essa cooperação gerou a necessidade da edição das ontologias por esses especialistas de domínio. Esses especialistas usam ontologias em suas áreas, mas não têm necessariamente, experiência em criar ontologias. Editar ontologias pode ser uma tarefa bastante complexa quando essas possuem centenas de classes, instâncias, relacionamentos e axiomas. Trabalhos como o de Netto e Lima (2017) denotam a necessidade de estudos sobre a usabilidade dessas ferramentas bem como o desenvolvimento de melhores técnicas de visualização de ontologias.

Um protótipo do SustenAgro foi desenvolvido e testado por usuários da Embrapa/APTA. Para representar ontologias, ele usa um formato mais simplificado (baseado na linguagem *YAML Ain't Markup Language* (YAML)²) que é convertido para OWL. A sua ontologia foi desenvolvida pelos especialistas de domínio em sustentabilidade da Embrapa e APTA com a ajuda de pesquisadores em Computação da USP. Além dessa ontologia, o sistema usa uma DSL para especificar algumas funcionalidades do sistema. Essa DSL permite que os especialistas de domínio possam modificar partes do sistema sem a ajuda de programadores. O SustenAgro gera todas as interfaces de usuários e funcionalidades do sistema automaticamente a partir de informações na ontologia e do programa escrito na DSL.

Um editor básico de ontologias foi adicionado ao protótipo para suportar a visualização e edição online das ontologias de domínio. Tal editor foi desenvolvido para o ambiente web e

² <http://yaml.org/>

possui, como característica principal, o uso do formato de texto simplificado para ontologias baseado em YAML. Esse editor, desenvolvido em trabalhos anteriores, permite que especialistas de domínio façam pequenas modificações nas ontologias, mas eles ainda precisam contar com programadores para desenvolver a ontologia específica do SAD.

O protótipo foi bem avaliado em testes com pesquisadores de várias instituições. Mas o grupo de pesquisa teve que esperar o registro do SustenAgro no Instituto Nacional de Propriedade Industrial (INPI) para planejar um teste de campo com vários produtores e usinas (o registro era uma exigência da Embrapa para usar o programa fora da empresa). Atualmente, esse teste está sendo planejado.

O protótipo do SustenAgro demonstrou a utilidade do uso de ontologias para especificar informações num SAD e a necessidade dos especialistas de domínio poderem alterar essas ontologias diretamente, sem a ajuda de programadores.

1.3 Problema de pesquisa

Esta pesquisa lida com a visualização de informação e, mais especificamente, com a visualização de ontologias e problemas associados nas representações gráficas desses modelos visuais manipuláveis.

A pergunta desta pesquisa é: Como melhor representar uma ontologia, em um editor gráfico voltado a especialistas de domínio, utilizando formatos, dimensões, cores e tipos de elementos de interface de forma customizável?

Essa customização deve permitir que esses especialistas modifiquem as características visuais de interfaces para usar cores e formatos relacionadas ao seu próprio domínio, criando assim uma identidade visual individualizada.

1.4 Hipótese de Pesquisa

A hipótese pesquisada é que: Se um subconjunto mais simplificado (menos expressivo) de OWL, que pode ser representado graficamente, for usado em conjunto com a customização das notações dessas representações gráficas, por meio de regras, então especialistas de domínio em sustentabilidade terão uma maior capacidade de compreender e manipular os elementos de uma ontologia (classes, propriedades e instâncias) e suas interconexões.

A customização deve facilitar a identificação de padrões e problemas nas ontologias, ajudando na formação de hipóteses e na obtenção de novos *insights*.

1.5 Objetivo da pesquisa

O objetivo principal desta pesquisa foi propor e testar, através da avaliação de um protótipo, um editor gráfico de ontologias, mais eficaz para o uso de especialistas de domínio da área de sustentabilidade. Esse editor fez uso de um modelo de notações customizáveis (baseado em regras) para representação visual de ontologias no qual proporcionou interfaces mais compreensíveis, eficientes e eficazes para o uso de especialistas de domínio. O editor desenvolvido sacrificou parte da expressividade da OWL em troca da facilidade de uso de uma representação gráfica de ontologias.

Esse editor, uma vez implementado, permite que especialistas de domínio, sem conhecimentos de programação ou da linguagem OWL, possam realizar a edição interativa de ontologias. Esses especialistas conhecem e trabalham com ontologias em suas áreas de especialidade, mas não têm experiência com editores para a criação de ontologias.

1.6 Objetivos específicos

Para que fosse possível alcançar o objetivo proposto para essa pesquisa, fez-se necessário cumprir os seguintes objetivos específicos:

- Definir uma arquitetura de notações contendo definições como cores e formas dos grafos, para representação de conceitos presentes na linguagem OWL;
- Elaborar um modelo de representação junto aos especialistas de domínio, baseado na arquitetura definida, bem como no estilo de editor de ontologias desejado;
- O desenvolvimento do *Knowledge Graph Web Editor* (KGWE). Um editor gráfico de ontologias na forma de uma aplicação Web do tipo *Single Page Application* (SPA). Ele é capaz de criar ontologias para as SADs desenvolvidos atualmente pela Embrapa. Foi fornecido também o suporte necessário ao processo de implantação do editor;
- Demonstrar que o uso do editor e das notações permite aos especialistas de domínio obter ganhos na visualização e edição de ontologias.

1.7 Método de Avaliação e Contribuições

A abordagem utilizada neste trabalho foi empírica e qualitativa. O método de pesquisa utilizado foi o estudo de caso e na sua avaliação foram utilizadas as seguintes técnicas:

- Questionário estruturado online;
- Entrevista remota semiestruturada.

O editor proposto foi avaliado durante diferentes etapas de seu desenvolvimento. Foram realizadas entrevistas com os especialistas de domínio com o intuito de avaliar o grau de satisfação dos usuários. Foi avaliado também o nível de eficiência e eficácia da ferramenta, quando comparado ao modelo tradicional utilizado por esses especialistas para edição de ontologias.

As entrevistas foram realizadas em duas etapas, sendo elas:

- **Avaliação de tarefas:** nessa avaliação foram realizadas execuções completas de algumas tarefas no editor. Cada participante executou as tarefas e, durante esse processo, verbalizou seus pensamentos (técnica *Think Aloud*).
 - Ferramenta utilizada: Google Meet;
 - Tipo de registro: gravação;
 - Tempo aproximado da atividade: 1h e meia.
- **Avaliação da eficiência das notações do KGWE:** nessa avaliação, os especialistas foram orientados a avaliarem diferentes versões de uma ontologia em diferentes editores. Do mesmo modo que na avaliação anterior, o entrevistado executou um conjunto de tarefas e verbalizou suas impressões. No final, os tempos totais de cada atividade foram usados para a avaliação da eficiência de cada ferramenta nos diferentes cenários.
 - Ferramenta utilizada: Google Meet;
 - Tipo de registro: gravação;
 - Tempo aproximado da atividade: 1h.

Foram geradas diversas contribuições científicas, que serão apresentadas ao longo deste trabalho, incluindo:

- Artigo (primeiro autor) "Um Modelo de Notações Customizáveis para Representação de Ontologias" publicado 12º Seminário de Pesquisa em Ontologias no Brasil (Ontobras) no Workshop de Teses e Dissertações em Ontologias (WTDO 2019). Periódico online CEUR-WS. ISSN: 1613-0073.
- **Menção honrosa de melhor trabalho em nível de mestrado da WTDO 2019.**
- Artigo (primeiro autor) "KGWE - A customizable editor for OWL ontologies" publicado no 26º Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) no Workshop de Ferramentas e Aplicações (WFA 2020). ISSN: 2596-1683.
- Artigo (co-autor) "Desenvolvimento de ontologias e linguagem de domínio específico para análises de complexidade semântica em nanotecnologia agrícola" publicado no Workshop de Nanotecnologia Aplicada ao Agronegócio (2017). ISSN: 2175-8395.

1.8 Organização do Trabalho

O restante desta dissertação está estruturada da seguinte forma:

Capítulo 2: Apresenta definições sobre ontologias, RDF e OWL. É apresentado também os conceitos sobre os Sistemas de Apoio à Decisão, assim como os SADs Sustenagro e Nano.

Capítulo 3: Apresenta as técnicas de visualização de informação, seus mecanismos e sua importância para uma representação visual eficiente do conhecimento. Também são apresentados os conceitos ligados a visualização de ontologias, notações gráficas e usabilidade.

Capítulo 4: Apresenta as ferramentas de visualização e edição de ontologias, suas características e importância para esta pesquisa.

Capítulo 5: Apresenta o editor desenvolvido, sua arquitetura e também seus principais componentes.

Capítulo 6: Apresenta a avaliação realizada por meio dos estudos de caso e os resultados obtidos nessas avaliações.

Capítulo 7: Apresenta as conclusões desta pesquisa, as dificuldades envolvidas durante o seu desenvolvimento e também algumas sugestões de trabalhos futuros.

ONTOLOGIAS E SISTEMAS DE APOIO À DECISÃO

O termo ontologia é antigo e no sentido filosófico pode-se defini-lo como um sistema particular de categorias que representam uma certa visão de mundo ([GUARINO, 1998](#)).

Ontologias podem ser utilizadas como recurso computacional para organização e representação de conceitos dentro de um domínio e também como mecanismo para a disseminação dessas informações. Ontologias proporcionam uma forma de representar dados. Caso uma ontologia seja definida usando-se uma linguagem formal, como a OWL, esses dados podem então ser interpretados por computadores. [Gruber \(1993\)](#) define as ontologias como uma especificação formal e explícita de uma conceitualização compartilhada.

[Guarino \(1997\)](#) considera a classificação das ontologias em duas dimensões. A primeira classificação tem base no grau de detalhe usado para caracterizar uma conceitualização. A segunda dimensão considera o assunto da conceitualização, ou seja, a natureza da definição desses conceitos, classificando as ontologias como: de aplicação, de domínio, de tarefas, genéricas ou de representação.

- Ontologias de Aplicação: descrevem conceitos que dependem de um domínio específico e correspondem a funções desempenhadas por entidades de domínio durante a execução de uma determinada atividade ([GUARINO, 1997](#)).
- Ontologias de Domínio: uma ontologia de domínio é uma coleção de vocabulários e especificações da conceitualização de um determinado domínio ([GRUBER, 1993](#)). Desse modo, a ontologia de domínio representa os termos aplicados ao domínio abrangendo as particularidades genéricas daquele domínio, como exemplo, o domínio da medicina.
- Ontologias de Tarefas: descrevem o vocabulário relacionado a tarefas ou atividades genéricas, por exemplo em processos de vendas ou diagnóstico ([GUARINO, 1997](#)).

- Ontologias Genéricas: descrevem conceitos gerais, independentes de um problema ou domínio particular (GUARINO, 1997).
- Ontologias de Representação: explicam as conceituações que fundamentam os formalismos de representação de conhecimento, procurando tornar claros os compromissos ontológicos embutidos nestes formalismos (MORAIS; AMBRÓSIO, 2007).

O uso de uma ontologia de domínio em SADs permitiu, em trabalhos anteriores, explorar suas características para representação do conhecimento de especialistas de domínio, permitindo definir, classificar, relacionar e inferir conhecimento (SUAREZ, 2017).

2.1 Ontologias na Agricultura

A seguir são apresentados alguns trabalhos que também lidam com ontologias de domínio da área de sustentabilidade agrícola:

Khamparia, Pandey e Pardesi (2014) apresentam uma abordagem utilizando tecnologias da web semântica em que se busca aumentar a produtividade de agricultores, utilizando para isso consultas SPARQL nesse sistema.

Ukpe (2013) apresenta uma ontologia agrícola na Nigéria, em que se busca não apenas padronizar uma terminologia agrícola, mas fornecer informações necessárias para que os agricultores possam fortalecer a agricultura local, fazendo uso do controle de pragas, dentre outros fatores importantes.

Abt, Vigier e Schneider (2009) apresentam uma proposta de modelagem conceitual de fazendas, a fim de desenvolver sistemas de gerenciamento que lidam com maiores complexidades do domínio. O objetivo nesse caso é aprimorar o sistema produtivo agrícola, principalmente no que se refere aos aspectos relacionados à sustentabilidade.

Roussey *et al.* (2010) mencionam que o uso de ontologias tem sido realizado em diversas aplicações relacionadas à agricultura. Kumazawa *et al.* (2009) citam que, na sustentabilidade, faz-se necessário criar alternativas que permitam definir conhecimentos de forma estruturada e que as ontologias forneçam o suporte necessário para isso.

Os trabalhos citados acima evidenciam a importância do uso de ontologias nesse domínio, e comprovam a relevância deste tópico de pesquisa para a área. Isso é especialmente verdadeiro para os especialistas de domínio de sustentabilidade da Embrapa/APTA.

As próximas seções dão uma visão geral dos padrões utilizados para descrever ontologias na Web.

2.2 RDF e RDFS

Para representar ontologias de forma inteligível a computadores, é necessário o uso de uma linguagem formal. O W3C(*World Wide Web Consortium*) define um conjunto de linguagens padrão para a definição de ontologias na Web. Dentre elas, as mais básicas são o *Resource Description Framework* (RDF) e o *RDF Schema* (RDFS).

O RDF é um padrão do W3C para descrever metadados (dados sobre dados) cuja função é descrever recursos, propriedades e valores. Sua sintaxe é simples e existem diversos formatos de serialização, como o TURTLE ou JSON-LD. A serialização de dados é o processo de escrever o estado de um objeto para um fluxo de dados e também o processo de reconstruir esse fluxo de volta em um objeto (HERICKO *et al.*, 2003).

O RDF permite fazer afirmações, sendo que essas afirmações expressam uma relação entre dois recursos e possuem como estrutura três elementos (tripla): <sujeito> <predicado> <objeto> (LAUFER, 2015).

A Figura 1 mostra um grafo RDF contendo uma tripla (afirmação) sobre um livro. Já na Figura 2 são três triplas sobre esse mesmo recurso.

Figura 1 – Grafo RDF da tripla de um livro.

Fonte: Laufer (2015).

Figura 2 – Grafo RDF com três triplas de um mesmo recurso.

Fonte: Laufer (2015).

O *RDF Schema* (RDFS) é uma extensão semântica do RDF, ou melhor, um vocabulário que estende o RDF dando mais expressividade a este. Ele fornece mecanismos para descrever grupos de recursos relacionados e as relações entre esses recursos (BRICKLEY; GUHA; MC-BRIDE, 2014). Com o RDFS é possível definir ontologias de baixa ou média complexidade, mas ontologias mais sofisticadas necessitam de uma linguagem de representação com maior expressividade. Quanto mais expressiva for a linguagem da ontologia, mais precisamente o significado pretendido de um vocabulário poderá ser especificado e, conseqüentemente, conclusões mais precisas podem ser tiradas (VÖLKER, 2009).

O RDF/RDFS são importantes no contexto deste trabalho pois a OWL baseia-se nessas especificações, utilizando inclusive grande parte dos termos advindos do RDF/RDFS.

2.3 Web Ontology Language - OWL

Em 2001, o grupo de trabalho WebOnt da W3C se propôs a desenvolver uma nova linguagem para descrição de ontologias que fosse mais expressiva que RDFS e pudesse ser compartilhada na web (BECHHOFFER *et al.*, 2004b). O resultado foi a *Web Ontology Language*, que tem a sigla OWL (coruja em inglês). A linguagem OWL faz uso de lógica descritiva (*Description Logic*), um fragmento de lógica de primeira ordem, para representar explicitamente o significado de termos e as relações entre eles (BECHHOFFER *et al.*, 2004b). Por ser baseada em lógica descritiva, OWL fornece representações semânticas ainda mais expressivas que o RDFS.

Apesar dos vários benefícios extraídos da linguagem, vale ressaltar que devido a sua complexidade, a OWL é um recurso que acrescenta um certo nível de dificuldade para sua efetiva utilização. O conceito de mundo fechado já amplamente difundido pela computação em bancos de dados tradicionais é substituído pelo conceito de mundo aberto em ontologias. Isso significa que caso uma informação não esteja presente na base de dados ela não é falsa, e sim desconhecida. Este tipo de mudança de paradigma causa profunda alteração no modo que os especialistas veem os sistemas. Alguns axiomas como *Only* (owl:allValuesFrom) e *Some* (owl:someValuesFrom) são muitas vezes de difícil interpretação por usuários não especialistas em ontologias.

Na última versão da linguagem, a OWL 2, foram adicionadas algumas sub-linguagens, cada qual diferenciada em suas características de expressividade. Elas podem ser utilizadas em várias situações diferentes, em que cada perfil é adequado a um propósito específico, tendo em vista o resultado que se pretende obter.

Os perfis da OWL 2 são:

- **Full** – É destinado a usuários que desejam o máximo de expressividade, porém sem garantias computacionais. É improvável que se tenha a disposição um motor de raciocínio (*reasoner*) que seja capaz de suportar o raciocínio completo para todos os recursos do

OWL Full (BECHHOFFER *et al.*, 2004b);

- **DL** – OWL DL é destinada a usuários que desejam o máximo de expressividade juntamente com garantias computacionais de que todas as conclusões são computáveis e sobre sua decidibilidade, garantindo que todas as computações terminarão em tempo finito (BECHHOFFER *et al.*, 2004b);
- **EL** – De acordo com W3C (2012)

Esse perfil fornece um poder expressivo menor que o perfil DL, mas representa um subconjunto do OWL 2 para o qual os problemas básicos de raciocínio podem ser executados em tempo de execução polinomial relativo ao tamanho da ontologia. Esse perfil é particularmente útil em aplicações que empregam ontologias que contêm um número muito grande de propriedades e / ou classes;

- **QL** – Esse perfil da OWL2 destina-se a diversos tipos de aplicações, em especial pode-se afirmar que

OWL 2 QL é voltado para aplicações que usam grandes volumes de dados de instância, e onde a resposta à consulta é a tarefa de raciocínio mais importante. O poder expressivo do perfil é necessariamente bastante limitado, embora inclua a maioria dos principais recursos de modelos conceituais, como diagramas de classes UML e diagramas Entidade Relacionamento.(W3C, 2012);

- **RL** – Em relação a OWL 2 RL, de acordo com W3C (2012) pode-se afirmar que

Destina-se a aplicações que exigem raciocínio escalável sem sacrificar muito o poder expressivo. Esse perfil é projetado para aplicações que desejam trocar a expressividade total da linguagem para obter um maior grau de eficiência. Os sistemas de raciocínio OWL 2 RL podem ser implementados usando mecanismos de raciocínio baseados em regras. (W3C, 2012).

Para o perfil de um especialista de domínio, a OWL é uma linguagem extremamente complexa. Existem diversas expressões na linguagem, muitas delas de difícil compreensão até mesmo para um ontologista. A seguir são descritas as principais características da OWL 2.

2.3.1 Class Expressions

A OWL fornece construtores de classes lógicas. Em particular, ela fornece elementos de linguagem para os operadores lógicos "and", "or", e "not". Os termos OWL correspondentes são emprestados da teoria dos conjuntos: intersecção, união e complemento (HITZLER *et al.*, 2009).

- *Intersection*: a intersecção de duas classes consiste exatamente naqueles indivíduos que são instâncias de ambas as classes (HITZLER *et al.*, 2009);

- *Union*: a união de duas classes contém cada indivíduo que está contido em pelo menos uma dessas classes (HITZLER *et al.*, 2009);
- *Complement*: o complemento de uma classe corresponde à sua negação lógica: consiste exatamente dos objetos que não são membros da própria classe (HITZLER *et al.*, 2009).

2.3.2 Object/Data Property Restrictions

- *Universal*: é usada para descrever uma classe de indivíduos para a qual todos os indivíduos relacionados por uma propriedade devem ser instâncias de uma determinada classe (HITZLER *et al.*, 2009);
- *Existential*: define uma classe como o conjunto de todos os indivíduos que estão conectados por meio de uma propriedade particular a outro indivíduo que é uma instância de uma determinada classe (HITZLER *et al.*, 2009);
- *Exact Cardinality*: utilizado para definir o número exato de instâncias na restrição;
- *Maximum Cardinality*: especifica o número máximo de indivíduos em uma restrição;
- *Minimum Cardinality*: especifica o número mínimo de indivíduos em uma restrição.

2.3.3 Class Expression Axioms

- *Subclass*: permite modelar hierarquias de classes inteiras especificando os relacionamentos de generalização. Como regra geral, um relacionamento de subclasse entre duas classes A e B pode ser especificado se todas as instâncias da classe A são instâncias de B (HITZLER *et al.*, 2009);
- *Equivalent Classes*: duas classes são consideradas equivalentes se contiverem exatamente os mesmos indivíduos (HITZLER *et al.*, 2009);
- *Disjoint Classes*: na disjunção entre duas classes, tem-se que a participação de um indivíduo em uma classe, exclui automaticamente sua participação na outra.

2.3.4 Object/Data Property Axioms

- *Subproperty*: da mesma forma que com classes, é possível também definir uma hierarquia de propriedades;
- *Domain*: funciona como um limitador dos indivíduos onde a propriedade pode ser aplicada. A tripla é formada pela propriedade (sujeito), seguido por *rdfs:domain* (predicado) e pela classe (objeto) no qual o indivíduo deve pertencer;

- *Range*: restringe os tipos de valores que um indivíduo pode ter com base em uma classe ou um *XSD Data Type*. A tripla é formada pela propriedade (sujeito), seguido por *rdfs:range* (predicado) e pela classe ou *XSD Data Type* (objeto) que restringe o indivíduo;
- *Equivalent Properties*: pode ser usada para afirmar que duas propriedades têm a mesma extensão de propriedade (mesmos valores) (BECHHOFFER *et al.*, 2004a).

Existe uma diferença muito relevante entre a equivalência e a igualdade de uma propriedade. Propriedades equivalentes têm os mesmos "valores", ou seja, a mesma extensão de propriedade, mas podem ter diferentes significados intencionais, ou seja, denotam conceitos diferentes (BECHHOFFER *et al.*, 2004a).

- *Disjoint Properties*: duas propriedades são disjuntas se não houver dois indivíduos interligados por ambas as propriedades (HITZLER *et al.*, 2009);
- *Inverse Properties*: utilizada para definir uma relação inversa entre as propriedades;
- *Functional Property*: é uma propriedade que pode ter apenas um valor (único) y para cada instância x, ou seja, não pode haver dois valores distintos y1 e y2, de modo que os pares (x, y1) e (x, y2) sejam ambos instâncias desta propriedade (BECHHOFFER *et al.*, 2004a);
- *Symmetric Property*: é uma propriedade para a qual se mantém que se o par (x, y) é uma instância de P, então o par (y, x) também é uma instância de P (BECHHOFFER *et al.*, 2004a);
- *Transitive Property*: significa que se um par (x, y) é uma instância de P e o par (y, z) também é uma instância de P, então podemos inferir que o par (x, z) também é uma instância de P (BECHHOFFER *et al.*, 2004a).

2.3.5 Assertions

- *Individual Equality*: indica que duas referências de URI na verdade se referem à mesma coisa: os indivíduos têm a mesma "identidade" (BECHHOFFER *et al.*, 2004a);
- *Individual Inequality*: indica que duas referências de URI se referem a indivíduos diferentes (BECHHOFFER *et al.*, 2004a);
- *Class Assertion*: as classes são frequentemente usadas para denotar o conjunto de objetos compreendidos por um conceito de pensamento humano, como o conceito de pessoa ou o conceito de mulher. Consequentemente, podemos usar o mesmo tipo de afirmação para indicar que Maria é uma mulher, expressando que ela é uma instância da classe das mulheres (HITZLER *et al.*, 2009).

No âmbito computacional, a criação de ontologias tem sido beneficiada pelo desenvolvimento de linguagens formais, como a OWL, e de ferramentas sofisticadas como o Protégé, que oferecem os recursos necessários para profissionais especialistas em ontologias (NETTO; LIMA; JUNIOR, 2017).

A OWL trouxe consigo também benefícios para profissionais que não são especialistas em ontologias, mas que necessitam de uma linguagem formal para descrever conhecimento. Porém, nestes casos, os usuários enfrentam certa carência de ferramentas sofisticadas e que, ao mesmo tempo, abstraíam a complexidade de uma linguagem como a OWL.

Ontologias têm servido como um bom recurso na construção dos SADs, no qual será melhor explorado na próxima seção.

2.4 Sistemas de Apoio à Decisão

Sistemas de Apoio à Decisão (SADs) são sistemas que permitem comparar, analisar, simular e apoiar a seleção de alternativas com base na geração de cenários que envolvem variáveis relacionadas a um domínio (HEINZLE; GAUTHIER; FIALHO, 2017). Os SADs permitem aos especialistas de domínio modelar seu conhecimento.

A seguir são apresentados os dois principais sistemas de apoio à decisão em desenvolvimento conjunto com os especialistas de domínio da Embrapa/APTA.

2.4.1 Sistema de Apoio à Decisão: SustenAgro

O SAD SustenAgro é uma instanciación do *Framework Decisioner*. Ele possui uma ontologia de domínio para a avaliação de sustentabilidade de sistemas de produção de cana-de-açúcar na região centro-sul do Brasil. Essa ontologia, que também tem o nome SustenAgro, representa conceitos como: indicadores, componentes de indicadores, índices, dimensões da sustentabilidade, recomendações e o método de avaliação (SUAREZ, 2017). Para que ela fosse computável, a mesma foi definida na linguagem OWL, utilizando-se, inicialmente, o editor de ontologias Protégé. O sistema SustenAgro e sua ontologia servirão como estudo de caso para este trabalho e permitirão analisar a metodologia de desenvolvimento da ferramenta.

A Figura 3 representa um mapa conceitual dos conceitos envolvidos na avaliação de sustentabilidade. Tal avaliação recebe como entrada os indicadores selecionados pelos usuários e aplica as fórmulas de avaliação, gerando, como resultado, os índices de sustentabilidade e eficiência. Com esses dois índices, o método é capaz de estimar a sua sustentabilidade (SUAREZ, 2017). Todas essas definições estão contidas na ontologia de domínio SustenAgro e têm como objetivo suportar a avaliação de sustentabilidade no sistema produtivo de cana-de-açúcar na região centro-sul do Brasil. Todas essas definições são representadas por meio de entidades, classes, relações semânticas e axiomas presentes na ontologia.

Figura 3 – Mapa conceitual - Método de avaliação

Fonte: Suarez (2017).

Decisioner Framework

O *Decisioner Framework* permite definir e gerar SADs baseados em conhecimento. Ele é um *framework* web composto por ontologias e por uma DSL que permite definir os comportamentos e estabelecer configurações gerais de um SAD (SUAREZ, 2017). Na Figura 4 é possível ver a arquitetura do *Decisioner*.

O *Framework Decisioner* possui um editor disponível para edição da ontologia OWL. Toda informação recuperada da *triplestore* é serializada no formato YAML para edição. O formato de serialização YAML foi utilizado devido a sua simplicidade. Tal funcionalidade foi bem recebida pelos especialistas por utilizar termos mais simples para a OWL quando comparado a ferramentas como o Protégé. Porém devido às limitações de representação e da criação inicial de uma ontologia o autor cita que tais características devem ser exploradas e aprimoradas em trabalhos futuros.

2.4.2 Sistema de Apoio à Decisão: Nano

O método *Risk Assessment Method for Genetically Modified Plants* (GMP-RAM) foi desenvolvido para que fosse possível avaliar a segurança de plantas geneticamente modificadas em relação a saúde humana e ao meio ambiente (JESUS *et al.*, 2006).

Conforme descrito por BUENO e Jesus (2011), o método GMP-RAM disponibiliza dois índices como resultado, o Índice de Risco e Índice de Significância, cada um deles é composto

Figura 4 – Arquitetura do Decisioner

Fonte: [Suarez \(2017\)](#).

por fatores de ponderação, sendo eles:

- Magnitude;
- Exposição ao dano;
- Precedente;
- Extensão;
- Reversibilidade.

Por ser um método comumente utilizado e devido aos bons resultados conseguidos, foi realizada a adequação desse método para a avaliação dos impactos de nanotecnologias no meio ambiente. Vale ressaltar que o sistema de apoio à decisão Nano ainda não foi implementado.

2.5 Considerações Finais

Neste capítulo, procurou-se reunir os principais conceitos sobre as ontologias, incluindo as ontologias de domínio e também os conceitos sobre RDF e OWL.

SADs são ferramentas importantes que auxiliam os especialistas de domínio na execução de suas atividades. Trabalhos anteriores abordaram o uso desse tipo de sistema e demonstraram sua validade junto aos especialistas de domínio da Embrapa. Os SADs, em desenvolvimento

conjunto com a Embrapa/APTA, fazem uso de ontologias para que os especialistas de domínio possam descrever conhecimento.

Um editor de ontologias integrado, que permita que especialistas de domínio possam visualizar e alterar ontologias diretamente no SAD, é uma funcionalidade muito útil, proporcionando uma visualização imediata das alterações. Tal editor deve ser projetado usando conceitos de visualização e usabilidade. No próximo capítulo esses conceitos serão apresentados. Tais conceitos são muito importantes para que os sistemas construídos sejam eficientes e também eficazes.

VISUALIZAÇÃO DE INFORMAÇÃO E USABILIDADE

Com o propósito de auxiliar os usuários a melhor compreender as informações presentes em uma ontologia de forma gráfica, faz-se necessário o uso de técnicas derivadas da Visualização de Informação.

Técnicas de visualização são baseadas em mecanismos de representação visual e de interação que permitem ao usuário manipular os elementos dessa representação de modo a melhor compreender o conjunto de dados ali representado ([FREITAS *et al.*, 2001](#)).

Ainda segundo [Freitas *et al.* \(2001\)](#), a cada dia mais e mais dados ficam disponíveis para análise nos diversos campos de conhecimentos, causando uma sobrecarga de informações e gerando, com isso, diversos problemas na representação desses dados. As técnicas de visualização de informação permitem uma abstração do conjunto de informações proporcionando também uma organização desse conjunto mediante algum critério do usuário.

[Ware \(2012\)](#) apresenta alguns argumentos que denotam as vantagens da utilização de técnicas de visualização, sendo eles:

- Fornecer capacidade de compreensão de grande quantidade de dados;
- Permitir identificar propriedades que não se esperava encontrar, possibilitando novos *insights*;
- Fazer com que problemas nos dados se tornem imediatamente aparentes;
- Facilitar a compreensão dos dados em grande e pequena escala, possibilitando o descobrimento de padrões;
- Facilitar a formação de hipóteses.

A visualização de informação é definida por [Card, Mackinlay e Shneiderman \(1999\)](#) como "o uso de representações visuais interativas de dados abstratos suportadas por computador para ampliar a cognição".

Para a transformação de dados abstratos em visualizações, faz-se necessário explorar diversas representações gráficas a fim de identificar pontos em que seja possível se obter algum ganho cognitivo: ampliando a capacidade de percepção dos usuários, ajudando-os a identificar, interpretar e compreender melhor esses dados, levando até mesmo a dedução de novos conhecimentos.

Um desafio importante, em relação à visualização de ontologias, é a utilização de características capazes de adicionar dimensões extras de expressividade à apresentação. Para dar um exemplo: os seres humanos associam conceitos importantes a um tamanho maior e conceitos menos importantes a um tamanho menor. Essa peculiaridade de percepção pode ser usada para exibir classes com muitas instâncias em um tamanho maior a fim de transmitir a importância dessas classes. Outro exemplo, no contexto das visualizações, é desenhar as relações mais importantes com linhas mais espessas ([DUDÁŠ et al., 2018](#)).

Este trabalho lida com a visualização de informação e mais especificamente, com a visualização de ontologias. A seguir são apresentadas as técnicas de visualização de ontologias e sua importância para uma representação eficiente do conhecimento.

3.1 Visualização de ontologias

Segundo [Katifori et al. \(2007\)](#), uma ontologia deve ser exibida de forma que o usuário possa, sem esforço, identificar as seguintes informações:

- Classes: devem ser visualizadas de maneira inteligível;
- Instâncias: representam, na maioria dos casos, dados de real interesse. Representá-las como nós conectados a classes nem sempre é eficaz devido ao seu grande número. Como alternativa de visualização, devem ser apresentadas em uma janela separada;
- Taxonomia (relações Isa): essa é a representação das relações de herança entre as classes. O sistema deve, ao menos, fornecer uma representação hierárquica da ontologia;
- Heranças múltiplas: esses são os casos em que uma classe tem mais de um pai. Não são fáceis de representar em combinação com uma representação eficaz da taxonomia;
- Relações: outras relações entre classes, que não pertencem a taxonomia, são difíceis de serem representadas;

- Propriedades: as propriedades associadas a uma entidade são também de muita importância na representação. Essas propriedades devem ser visualizadas na ontologia ou dentro de um espaço separado.

É muito importante para um editor de ontologias apresentar elementos como classes, instâncias, relações e propriedades de forma que o usuário possa compreender facilmente todo o conjunto de informações apresentado em tela. Porém, não é uma tarefa nada fácil atender tais requisitos de interface, exigindo assim novas abordagens para representar esses elementos.

Existem diversas técnicas de visualização aplicáveis a ontologias. Por exemplo, visualizações hierárquicas são mais eficientes para taxonomias. Já outras técnicas são mais eficientes para representar grandes volumes de dados sobre classes e instâncias. O fato é que um projetista de uma ferramenta de edição de ontologias não pode usar apenas uma técnica de visualização, sob pena de criar uma ferramenta muito limitada. [Katifori et al. \(2007\)](#) apresentam seis técnicas de visualização e interação aplicáveis a representações de ontologias:

- Lista indentada;
- Árvore de links e nós;
- Zoom;
- Preenchimento de espaço;
- Foco + Contexto;
- Paisagens 3D.

Essas técnicas de visualização e interação são essenciais para o sucesso de uma aplicação visual. Mas segundo [Katifori et al. \(2007\)](#), mais pesquisas e avaliações são necessárias para acoplar a visualização e interação de forma eficaz para criar uma ferramenta útil e fácil de se utilizar.

Especialistas em domínios de diferentes áreas possuem conhecimentos diversos e utilizam meios também diversos para representar seu conhecimento em ontologias. Na [Figura 5](#) é possível ver uma ontologia de engenharia para um estudo de caso em uma fábrica na França. A meta é avaliar diferentes configurações para produção de automóveis ([COSSENTINO et al., 2010](#)).

Já na [Figura 6](#) é possível observar uma ontologia para representar processos biológicos, desenvolvida pelo consórcio *Gene Ontology Consortium* e modelada no editor de ontologias OBO-Edit ([CONSORTIUM, 2009](#)).

Essas ontologias têm em comum a representação do conhecimento de especialistas de domínio, mas divergem muito em sua representação.

Figura 5 – Ontologia para produção de automóveis

Fonte: Cossentino *et al.* (2010).

Alguns domínios veem a necessidade de representar classes com retângulos. Já outros representam classes como círculos, elipses, ou outros formatos. No editor desenvolvido durante este trabalho, o especialista poderá modelar sua interface da forma que julgar mais apropriada.

Na busca por ferramentas que tenham interfaces apropriadas ou que sejam mais fáceis de trabalhar, alguns especialistas utilizam ferramentas que não são voltadas especificamente para ontologias para modelar e representar seu conhecimento. Esse fato é relevante pois muitos especialistas são obrigados a utilizar essas ferramentas (voltadas a outros propósitos), apenas porque buscam interfaces melhores, mas acabam encontrando outras limitações inerentes ao propósito dessas ferramentas. Por exemplo, essas ferramentas não suportam importação e exportação no

Figura 6 – Ontologia para processos biológicos

Fonte: Adaptada de Consortium (2009).

formato OWL. Pode-se citar como exemplo desse tipo de ferramentas, o Gephi¹, Vosviewer², Cmaptools³, Xmind⁴, Yed⁵

3.2 Métodos de visualização

Diferentes casos de uso exigem diferentes técnicas de visualização (DUDÁŠ *et al.*, 2018). Partindo dessa premissa, neste trabalho foram pesquisadas diferentes técnicas de visualização, tendo como objetivo atender a uma gama maior de casos. As principais técnicas serão detalhadas a seguir.

3.2.1 Node-Link with Force-Directed Layout

Este método apresenta o grafo contendo os nós e suas relações. O método de *node-link* juntamente com o gráfico direcionado por força formam a principal funcionalidade de edição

¹ <<https://gephi.org/>>

² <<https://www.vosviewer.com/>>

³ <<https://cmap.ihmc.us/cmaptools/>>

⁴ <<https://www.xmind.net/>>

⁵ <<https://www.yworks.com/products/yed>>

disponível na ferramenta desenvolvida neste trabalho. A desvantagem desse tipo de visualização é lidar com o constante rearranjo dos nós e a dispersão de alguns grupos de classes em toda interface, exigindo assim constante navegação. Uma possível solução para esses casos, seria a possibilidade de salvar as posições dos nós e dar a possibilidade de se trabalhar com um subconjunto da ontologia, facilitando a visualização e busca de elementos.

3.2.2 Indented List

Tradicionalmente, ferramentas de visualização de ontologias como o Protégé, possuem este tipo de visualização.

A principal vantagem da visualização de lista indentada é sua simplicidade de implementação e representação, e sua familiaridade com o usuário, visto que o mesmo conceito é usado em vários navegadores de arquivos (KATIFORI *et al.*, 2007).

Em um estágio exploratório inicial com o protótipo navegável e também com um protótipo de vídeo, esse método de visualização foi apontado pelos especialistas de domínio como importante e essencial na ferramenta.

Seguindo esse *feedback* e visto que tais usuários já estão acostumados a utilizar esse modelo, optou-se também pela disponibilização da lista indentada no editor desenvolvido.

3.2.3 3D

O método de visualização em três dimensões é muito útil para lidar com ontologias grandes, ou seja, é um método eficiente em sistemas que apresentam milhares de classes, instâncias, propriedades, entre outros tipos de entidades.

A questão das visualizações 3D é bastante controversa (KATIFORI *et al.*, 2007). Muitas ferramentas não lidam com as complexidades adicionais inseridas por esse tipo de visualização. No campo de visualização de ontologias, estudos adicionais são necessários para avaliar os reais benefícios desse tipo de visualização, bem como mitigar problemas adicionais inseridos por esse método.

3.3 Técnicas de interação

Algumas técnicas de interação são comumente encontradas em editores gráficos para diversos tipos de dados (incluindo ontologias). Destacam-se a seguir algumas das técnicas mais populares de interação:

- *Zoom*: essa técnica permite aproximar ou distanciar o usuário de nós na visualização de ontologias;

- *Pan*: com essa técnica, é possível fazer o arrasto tanto verticalmente, como horizontalmente. Este mecanismo possibilita que o usuário não fique preso apenas em uma região da ontologia;
- *Filter*: permite a filtragem por relação de vizinhança. É possível definir qual o nível de relações que se deseja exibir a partir do nó selecionado. No editor desenvolvido pode-se também selecionar se a relação de vizinhança da filtragem é de pai para filho, de filho para pai, ou ambas.

Além dessas técnicas, outras, como o *Drag and Drop*, também podem ser usadas.

3.4 Uso de cores na visualização

O uso de cores, que não agregam informações adicionais à visualização, pode causar confusão à medida que o usuário tenta discernir seu significado. A cor é um aspecto poderoso e atraente da experiência do mundo. Ela molda a percepção humana, a interpretação e memória de tudo o que se vê. A cor, portanto, fornece uma dimensão importante na comunicação visual: quando bem usada, pode melhorar bastante a eficácia de uma mensagem, mas, quando usada mal, pode prejudicá-la substancialmente (MACDONALD, 1999).

A customização de cores tem um papel importante no presente trabalho. Ela é usada para delegar a definição de cada cor do modelo para os especialistas. Essa atribuição tem como objetivo mitigar possíveis distorções ou uso incorreto de cores pré-definidas.

3.5 Definindo um modelo de notações gráficas

Genova (2014) afirma que uma notação gráfica é um sistema de representação de elementos de determinado campo de conhecimento. O conceito de notação, neste trabalho, corresponde à representação das características presentes numa ontologia em símbolos gráficos correspondentes.

Para demonstrar de forma simplificada como deve ser realizado o processo de criação de um modelo de notações, o exemplo a seguir é um caso sobre filmes extraídos de um blog da web disponibilizado por Clara Giacalone. O conteúdo extraído vem do *Internet Movie Database* (IMDb), um banco de dados sobre filmes. Essa base possui diversos dados de classificação de filmes, inseridos com base na avaliação de milhares de usuários. A Figura 7 mostra uma tentativa de se criar um modelo de notações para filmes do IMDb com o uso de cores. É possível observar que cada gênero é representado com cores específicas, selecionadas para criar um tipo de identidade visual para cada tipo de gênero na figura.

As cores influenciam a construção de nossas percepções visuais. No dia a dia é comum utilizar o vermelho para representar o amor, o branco para a paz, assim como o preto para repre-

Figura 7 – Paleta de cores e descrição dos gêneros

Fonte: Clara Giacalone (2015).

sentar o luto. As cores podem produzir impressões, sensações e reflexos sensoriais importantes, atuando como estimulante ou perturbador na emoção, na consciência e em nossos impulsos e desejos (FARINA; PEREZ; BASTOS, 2011).

Percebe-se a tentativa do autor de exprimir essas sensações no uso de cores para cada gênero de filme.

Na Figura 8, para representar a variação da quantidade de filmes por gênero, cada quadrado presente na imagem possui agora diferentes dimensões. Drama e comédia, por exemplo, são maiores e indicam que esses gêneros possuem uma grande quantidade de filmes.

Figura 8 – Quantidade de filmes avaliados em comparação com o conjunto de filmes do gênero

Fonte: Clara Giacalone (2015).

O conceito de tamanho é também muito importante no contexto das notações. Com o uso de regras é possível definir um conjunto muito maior de representações.

Com o uso de atribuições lógicas é possível montar regras extremamente úteis nas representações. Abaixo, podem ser observados exemplos de cenários reais em que a utilização de regras poderia surtir efeitos positivos na representação de entidades de uma ontologia:

1. Representar uma classe que possua quinhentos indivíduos com tamanho exato de cinquenta *pixels*; abaixo disso representar com dez *pixels* e acima disso representar com cem;
2. Representar uma classe que seja subclasse de outra com tamanho de vinte *pixels*. Para cada subclasse adicional aumentar vinte *pixels*, ou seja, se uma classe for subclasse de outras cinco classes, ela deve possuir tamanho total de cem *pixels*. Para classes que não são subclasses de nenhuma outra representá-las com tamanho de dez *pixels*;
3. Para cada restrição de propriedade adicionada na classe adicionar cinco *pixels* na borda do elemento.

É possível se usar regras ainda mais complexas:

1. Classes que forem subclasse de Pizza e (and) conterem em seu nome o texto "queijo" e (and) possuírem mais de 100 indivíduos **então** (then) exiba a cor de fundo vermelho e (and) apresente com 100 *pixels* e (and) apresente a borda com a cor amarela.

Voltando ao exemplo dos filmes, é possível também representar a quantidade de avaliações realizadas pelos usuários classificadas em cada gênero de filme. Para tanto, buscou-se representá-las na extremidade de cada elemento, ou seja, no volume da borda. Quanto maior a borda, significa que o filme possui um número maior de avaliações de usuários. A [Figura 9](#) apresenta variações nas bordas com base nas classificações realizadas pelos usuários. A cada informação adicionada, o modelo torna-se mais rico, ao mesmo tempo que novas notações são criadas para representá-las.

Figura 9 – Modelo com avaliações dos usuários por gênero

Fonte: [Clara Giacalone \(2015\)](#).

Caso um especialista de domínio estivesse modelando essas informações, ele poderia facilmente representá-las em uma ontologia OWL e usar um esquema visual parecido. No [Capítulo 5](#) na [Figura 17](#) é mostrado um exemplo desse tipo de esquema aplicado a uma ontologia OWL sobre filmes.

3.6 Usabilidade

Segundo [García-Barriocanal, Sicilia e Sánchez-Alonso \(2005\)](#), a usabilidade dos editores de ontologia pode ser vista como uma condição essencial para a adoção das tecnologias da web semântica.

A norma NBR 9241-11 da Associação Brasileira de Normas Técnicas (ABNT) define os conceitos da avaliação de usabilidade como ([ABNT et al., 2002](#)):

- Usabilidade: medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação num contexto específico de uso;
- Eficácia: acurácia e completude com as quais usuários alcançam objetivos específicos;
- Eficiência: recursos gastos em relação à acurácia e abrangência com as quais usuários atingem objetivos;
- Satisfação: ausência do desconforto e presença de atitudes positivas para com o uso de um produto.

De modo a especificar ou medir a usabilidade, é necessário identificar os objetivos e decompor eficácia, eficiência, satisfação e os componentes do contexto de uso em subcomponentes com atributos mensuráveis e verificáveis ([ABNT et al., 2002](#)). A [Figura 10](#) mostra o relacionamento entre esses componentes.

Os requisitos de usabilidade usados neste trabalho são discutidos no [Capítulo 5](#).

3.7 Considerações Finais

Numerosos métodos de visualização de ontologias já foram propostos, no entanto, nenhum método, ou conjunto de métodos, foi aceito pela maioria da comunidade da Web Semântica como um padrão de fato até agora ([MACDONALD, 1999](#)).

Para este trabalho, a pesquisa dos fundamentos em visualização de informação é relevante perante o interesse nas técnicas de visualização de informação já desenvolvidas. Essas técnicas deverão criar representações que venham a facilitar a compreensão das estruturas em uma ontologia.

[Netto e Lima \(2017\)](#) mencionam a relevância do tema de visualização de ontologias, visto o seu potencial para melhorar tarefas como a compreensão humana do conhecimento implícito, bem como o alinhamento da informação em domínios de aplicação.

O próximo capítulo apresenta as principais ferramentas de visualização e/ou edição relacionadas a este trabalho e expõe brevemente o mecanismo de visualização adotado por elas.

Figura 10 – Estrutura de usabilidade

Fonte: [ABNT et al. \(2002\)](#).

FERRAMENTAS DE VISUALIZAÇÃO E EDIÇÃO DE ONTOLOGIAS

Vários trabalhos têm contribuído para o desenvolvimento de diversas ferramentas para visualização e edição de ontologias.

Durante o desenvolvimento dessa dissertação, foram realizadas pesquisas na literatura para identificar os trabalhos ou ferramentas que compunham o estado da arte na área. Os trabalhos considerados nesta pesquisa possuem os seguintes critérios de classificação:

- Ser um visualizador e/ou editor de ontologias;
- Ter suporte a OWL;
- Estar disponível na plataforma Web.

Além desses trabalhos, foram consideradas algumas ferramentas não voltadas à edição de ontologias, mas que foram sugeridas por especialistas da Embrapa e são usadas por eles para modelagem. O objetivo foi verificar o tipo de interface que esses especialistas estão acostumados e acham úteis.

4.1 Ferramentas

Houve a tentativa de apresentar aqui as ferramentas mais relevantes, considerando como relevância a disponibilização da ferramenta para uso e/ou que características, presentes na ferramenta, têm alguma intersecção com a proposta deste trabalho (como exemplo, a exibição de elementos OWL em formatos não convencionais).

4.1.1 WebProtégé

WebProtégé é uma ferramenta de código aberto para desenvolvimento colaborativo de ontologias no ambiente Web.

O WebProtégé foi desenvolvido pelo *Stanford Center for Biomedical Informatics Research*¹. Essa ferramenta possui características como o suporte a OWL 2, histórico de mudanças e desenvolvimento de ontologias em colaboração. A Figura 11 ilustra a ferramenta WebProtégé e seu mecanismo de visualização gráfica padrão, que representa as relações de herança do elemento selecionado. Essa ferramenta é derivada de uma versão *desktop* famosa, o editor Protégé, que é amplamente utilizado para edição de ontologias OWL no mundo todo, mas não tem todos os recursos desta.

Figura 11 – Representação da ferramenta WebProtégé

Fonte: Tudorache et al. (2013a).

O WebProtégé permite a hospedagem da ferramenta em servidores locais (em adição a sua versão online), nesses servidores usuários podem criar regras específicas para o controle de acesso. Organizações podem aplicar políticas próprias de acesso ou usar a ferramenta numa intranet.

O *backend* da aplicação foi construído com a linguagem Java e a comunicação entre as camadas é realizada por meio de APIs *Remote Method Invocation* (RMI).

Usuários podem personalizar a aparência do WebProtégé reorganizando as abas e definindo um *layout* personalizado. Essas personalizações serão aplicadas na próxima vez que o usuário abrir o projeto na ferramenta (TUDORACHE et al., 2013b).

¹ <<https://bmir.stanford.edu/>>

A aba de hierarquia de classes, presente no editor, é a principal funcionalidade de edição de uma ontologia nesta ferramenta. O WebProtégé não possui a visualização gráfica (grafo), sendo assim não foi possível extrair as características padrões para exibição da ontologia neste tipo de representação (cor padrão de classe, tamanho padrão, etc). Na aba de hierarquia de classes o especialista pode realizar diversas operações, como criar ou remover classes e subclasses, porém, a versão da interface dessa aba na ferramenta web é significativamente mais concisa que na versão *desktop*. Se por um lado a interface parece mais simples, por outro o usuário fica mais limitado a certos tipos de operações de edição para a OWL.

O WebProtégé permite a customização de suas interfaces através da criação de *portlets*. Por exemplo, o grupo do Protégé desenvolveu o iCAT, uma extensão do WebProtégé, para uso dos especialistas de domínio da Organização Mundial da Saúde (OMS) durante o desenvolvimento da 11a revisão da Classificação Internacional de Doenças (CID-11). Essa foi a primeira revisão baseada em OWL. Contudo, o iCAT é um programa em Java desenvolvido por programadores e não uma customização que pode ser feita por usuários/especialistas.

O WebProtégé é uma das ferramentas onde há uma grande intersecção com o presente trabalho, principalmente quando se olha o público alvo de ambas as ferramentas.

4.1.2 WebVOWL

A WebVOWL é uma aplicação web para visualização de ontologias que faz uso das notações definidas pela VOWL.

A notação visual para ontologias OWL, denominada VOWL, define uma linguagem visual para representação de ontologias. Os componentes da VOWL são formados por um conjunto básico de blocos que consideram formas e cores distintas para representar diferentes aspectos dos elementos da OWL (LOHMANN *et al.*, 2014).

A WebVOWL fornece representações gráficas para elementos da OWL que são combinadas com um *layout* gráfico direcionado por força (LOHMANN *et al.*, 2015b). A Figura 12 ilustra a ferramenta WebVOWL.

A ferramenta WebVOWL utiliza, para representação dos seus elementos, tecnologias como HTML5 e SVG, fazendo uso da biblioteca *javascript* D3 para renderizar os elementos em seu modelo. Originalmente, a ferramenta disponibilizava apenas a visualização de ontologias, porém, durante o desenvolvimento deste trabalho, a versão beta do editor foi incorporada à versão oficial fazendo com que a ferramenta se tornasse de fato um editor e não apenas um visualizador de ontologias.

Uma diferença importante entre o presente trabalho e o VOWL é o desenvolvimento de um modelo de notações totalmente personalizável, este tipo de abordagem não foi identificado na ferramenta WebVOWL.

Figura 12 – Representação da ferramenta WebVOWL

Fonte: Lohmann *et al.* (2015a).

Pode-se observar na Figura 13 a representação gráfica das classes, propriedades e demais entidades para representar os conceitos presentes em OWL.

Figura 13 – Primitivas gráficas VOWL

Primitive	Application
	classes
	properties
	property directions
	datatypes, property labels
	special classes/properties
text number symbol	labels, cardinalities

Fonte: Lohmann *et al.* (2014).

Na Figura 14 é possível observar o esquema de cores proposto pelo modelo.

Para se alterar as cores definidas no esquema do VOWL, é necessária uma nova compilação do software, ou seja, é necessário o envolvimento de desenvolvedores. Não é possível que usuários consigam uma melhor representação de formas e cores dos elementos de forma autônoma.

A utilização de cores fixas nas representações dos elementos pode ser uma barreira na construção cognitiva e, até mesmo, uma dificuldade para usuários daltônicos. Com representações customizáveis, usuários podem, por exemplo, definir cores em que o contraste seja mais nítido

Figura 14 – Esquema de cores VOWL

Name	Color	Application
General		classes, object properties, disjoint relations
Rdf		elements based upon RDF and RDF schema
External		external classes and properties
Deprecated		deprecated classes and properties
Datatype		datatypes, literals
Datatype property		datatype properties
Highlighting		interactively highlighted elements

Fonte: [Lohmann et al. \(2014\)](#).

para eles. É essencial que os especialistas, como usuários do sistema, possam estabelecer cores e outras características que sejam mais representativas em seu domínio.

4.1.3 OWLGrEd

O editor OWLGrEd é uma ferramenta para visualização e edição de ontologias utilizando notações gráficas baseadas em *Unified Modeling Language* (UML) ([LIEPINŠ; GRASMANIS; BOJARS, 2014](#)). A UML é uma linguagem que permite construir documentações de software, ou seja, é uma linguagem de modelagem. Apesar de suas notações serem baseadas em UML, o editor não é específico para a linguagem. Os elementos gráficos são renderizados utilizando o elemento *Canvas* do HTML5 em conjunto com a biblioteca KineticJS ([Figura 15](#)). É importante salientar que apenas a versão *desktop* possui capacidade de edição, a versão Web permite apenas a visualização de ontologias.

O OWLGrEd utiliza o conceito de notações, porém elas são fixas (não customizáveis). Embora bastante difundida na Computação, a simbologia da UML ainda é desconhecida em muitas áreas, sendo que especialistas em ontologias geralmente não possuem tal conhecimento (fazendo-se necessário um treinamento prévio). Outra possível dificuldade, é que a simbologia do UML pode não ser parecida com a utilizada em muitos domínios.

4.1.4 OSMoSys

A OSMoSys é uma interface web para visualização, exploração e navegação de dados nos formatos RDF e OWL. Ela exhibe conceitos da ontologia como classes em forma de nós e propriedades e dados como arestas, como pode ser visto na [Figura 16](#). A interface OSMoSys é apresentada através de um protótipo que usa uma estrutura de representação de conhecimento para planejamento e gerenciamento de cidades inteligentes, permitindo a integração semântica de dados urbanos heterogêneos de diversas fontes ([PSYLLIDIS, 2015](#)).

4.1.5 MEMO GRAPH

MEMO GRAPH é uma ferramenta de visualização de ontologias projetada para ser utilizada por especialistas e não especialistas em ontologias. Ela trabalha com uma filosofia de design para todos, que se estende a pessoas com deficiências. Essa ferramenta utiliza um conjunto de diretrizes de projeto, coletadas da literatura, para projetar interfaces dedicadas a pacientes com Alzheimer (GHORBEL *et al.*, 2016). As limitações que acompanham os pacientes com Alzheimer são agrupadas em sete grupos (GHORBEL *et al.*, 2016):

- Deficiências na memória;
- Deficiências cognitivas;
- Deficiências visuais;
- Deficiências auditivas;
- Deficiências de mobilidade;
- Declínio de habilidades da fala;
- Habilidades de leitura e escrita inapropriadas.

Essa ferramenta é interessante por ter sido desenvolvida especificamente para ser utilizada por qualquer tipo de usuário e não apenas ontologistas. Devido a isso, os grafos mostram uma visão simplificada das ontologias.

Ela usa imagens ou símbolos para representar determinadas classes na ontologia, mas não permite customização.

Uma avaliação da satisfação dos usuários realizada com 24 usuários experientes em ontologia e 22 pacientes com Alzheimer confirmam que a ferramenta desenvolvida é eficiente (GHORBEL *et al.*, 2016).

4.2 Funcionalidades dos editores

A Tabela 1 apresenta uma comparação entre as funcionalidades das ferramentas apresentadas na última seção.

Das ferramentas apresentadas neste capítulo, apenas a ferramenta VOWL oferece edição gráfica em sua versão beta. A OWLGrEd oferece suporte à edição apenas em sua versão *desktop*. OSMoSys e MemoGraph possuem interface gráfica, mas não editável. Já o editor WebProtégé não disponibiliza um grafo editável, a edição nesta ferramenta deve ser realizada apenas na visão de árvore hierárquica.

Tabela 1 – Comparação das funcionalidades dos editores

	WebProtégé	VOWL	OWLGrEd	OSMoSys	MemoGraph
Tecnologia					
Web	Sim	Sim	Sim	Sim	Sim
Recurso					
Suporte a OWL	Sim	Sim	Sim	Sim	Sim
Representação					
Grafo com edição	Não	Sim	Não	Não	Não
Grafo com propriedades	Não	Sim	Sim	Não	Não
Funcionalidade					
Drag and Drop	Não	Não	Não	Não	Não
Customização de cor	Não	Não	Não	Não	Não
Customização de tamanho	Não	Não	Não	Não	Não
Regras para customização	Não	Não	Não	Não	Não

Uma característica importante, é que nenhuma das ferramentas analisadas possui função de *Drag and Drop* (função esta encontrada em diversos editores não voltados para ontologias OWL).

É possível observar que apenas as ferramentas VOWL e OWLGrEd apresentam propriedades OWL no grafo e nenhuma das ferramentas analisadas possui customização de cor e tamanho dos elementos visuais e nem possibilitam a criação de regras de exibição.

Por fim, alguns editores, como o VOWL, utilizam um conjunto de formas e cores para representação de ontologias, mas se baseiam em modelos fixos de representação. Uma das principais hipóteses discutidas neste trabalho foi que o uso de representações customizadas e baseadas em regras definidas pelos próprios especialistas de domínio devem beneficiar a visualização desses modelos.

4.3 Outras Ferramentas

As ferramentas a seguir não são voltadas para ontologias, mas muitas vezes fazem parte do cotidiano de um ontologista. O objetivo foi verificar o tipo de interface que esses especialistas estão acostumados e acham úteis.

As ferramentas das próximas subseções foram identificadas após reunião com um ontologista da Embrapa Informática Agropecuária e em conversas com especialistas da Embrapa e APTA Regional.

Algumas dessas ferramentas são pagas e muitas vezes estão disponíveis apenas em versões *desktop* para *download*.

4.3.1 Yworks - yEd - Graph Editor

O yEd - Graph Editor² é uma ferramenta para construção de diagramas fácil de utilizar. Essa ferramenta possui um *pallette* com diversos elementos para arrastar por meio da funcionalidade de *drag and drop*. Possui também *zoom* do diagrama, customização de tamanho e cor do elemento (manualmente, sem uso de regras), dentre outras funções.

4.3.2 Xmind

A ferramenta Xmind³ é conhecida por disponibilizar a criação de um mapa mental. A ideia é que o especialista possa explorar sua criatividade de forma rápida, sem ser barrado por problemas de complexidade de algumas ferramentas. A ferramenta possui vários tipos de *layout*.

4.3.3 CmapTools

A ideia do Cmaptools⁴ é a construção de mapas conceituais. Essa ferramenta permite criar entidades e conectá-las, sendo possível nomear essas conexões. Dessa forma, o CmapTools é útil para representar uma ontologia como um grafo.

4.3.4 VOSviewer

O VOSviewer⁵ é uma ferramenta preparada para construir ou visualizar redes bibliométricas. O VOSviewer trabalha com grandes volumes de dados. Possui uma interface que possibilita diversos ajustes na apresentação dos elementos, estando disponíveis também várias abas com diferentes mecanismos de visualização.

4.3.5 Gephi

O Gephi⁶ é um software de código aberto e gratuito e, assim como o VOSviewer, também trabalha com grandes volumes de informação. Seu foco é a apresentação de gráficos genéricos e de redes. O Gephi é uma ferramenta muito útil para análise de dados por cientistas de diversas áreas.

Ontologistas utilizam essas ferramentas, sem suporte à edição de ontologias, pois muitas delas possuem características que eles consideram relevantes e que não são encontradas em

² <<https://www.yworks.com/products/yed>>

³ <<https://www.xmind.net/>>

⁴ <<https://cmap.ihmc.us/cmaptools/>>

⁵ <<https://www.vosviewer.com/>>

⁶ <<https://gephi.org/>>

editores de ontologia para OWL. Principalmente em questões de usabilidade. Características como rapidez na criação da ontologia e personalização da interface foram citadas, assim como a funcionalidade de *drag and drop*. Sendo assim, é importante considerar tais características no desenvolvimento de um novo editor de ontologias.

4.4 Considerações Finais

Neste capítulo foram apresentados brevemente os visualizadores e editores de ontologias considerados mais relevantes. As ferramentas apresentadas serviram como base no estudo das características de um editor Web de ontologias para OWL.

No próximo capítulo será possível ver, em mais detalhes, as funcionalidades presentes no editor KGWE. Será possível também entender um pouco mais sobre as tecnologias empregadas no seu desenvolvimento, seus requisitos e sua arquitetura.

KNOWLEDGE GRAPH WEB EDITOR

O editor de ontologias *Knowledge Graph Web Editor (KGWE)* foi desenvolvido para que fosse possível validar o modelo de notações proposto neste trabalho. Com o propósito de torná-lo graficamente mais simples e ao mesmo tempo usar representações mais ricas para elementos de ontologias, foram desenvolvidos vários recursos com base em diversas técnicas de representação de informação.

O desenvolvimento do editor foi baseado em histórias de usuários. As histórias de usuários foram criadas com base em informações obtidas por meio de protótipos desenvolvidos no início deste trabalho.

O KGWE trabalha sempre em conjunto com os SADs existentes. Ao solicitar alterações em uma ontologia existente no SAD, o especialista de domínio será redirecionado para o editor KGWE.

As customizações de interface são sempre salvas a nível de usuário, ou seja, se um especialista criar um modelo de notações, esse modelo ficará visível somente para seu usuário, não concorrendo com customizações realizadas por outros usuários dentro do editor. Porém, alterações realizadas em uma ontologia são sempre persistidas na mesma base de dados, independente do usuário, ou seja, neste caso as modificações realizadas por diferentes usuários afetam os demais utilizadores da ferramenta para uma determinada ontologia. Isso ocorre devido ao processo de publicação da ontologia nos SADs. A ontologia pode ser desenvolvida de modo colaborativo pelos especialistas, porém uma única versão deve ser publicada no sistema de destino, ou seja, no sistema de apoio à decisão que originou a edição.

O KGWE apresenta diversas abas que permitem a edição e/ou visualização da ontologia, mas apenas uma delas permite a customização. Caso o especialista deseje customizar a representação do grafo, deve-se então utilizar a aba "Graph Editor".

Neste capítulo, é descrito o processo de análise de requisitos da ferramenta, apresentado o projeto da sua arquitetura, as tecnologias empregadas no desenvolvimento e também os principais

componentes da aplicação.

5.1 Requisitos do Editor

Para o levantamento de requisitos foram utilizados dois protótipos, sendo um protótipo de vídeo e um protótipo funcional. Com base nesses protótipos vários questionamentos foram realizados aos especialistas de domínio para criação das histórias de usuário (*User Stories*). Mais detalhes podem ser encontrados no [Capítulo 6](#).

As histórias de usuário são especificações contendo poucas sentenças na linguagem cotidiana do usuário do sistema que captura o que um usuário necessita fazer como parte de sua função de trabalho. O uso dessas histórias é comum em metodologias ágeis de desenvolvimento para capturar o "quem", "o quê" (o desejo do usuário) e "para que" de um requisito em uma forma concisa e simples. Essas histórias podem ser encontradas no apêndice A.1. A [Tabela 2](#) mostra resumidamente os principais requisitos funcionais do editor.

Tabela 2 – Requisitos Funcionais

Id	Descrição
RF01	A aplicação deve exibir aba que permite a edição dos nós em duas dimensões.
RF02	A aplicação deve exibir aba que permite a edição dos nós pela árvore hierárquica.
RF03	A aplicação deve exibir aba que permite a visualização dos nós em três dimensões.
RF04	A aplicação deve permitir a customização da cor de fundo de classes genéricas ou de uma classe específica através de regras.
RF05	A aplicação deve permitir a customização da cor de relação(link) de classes genéricas.
RF06	A aplicação deve permitir a customização da cor da borda de classes genéricas ou de uma classe específica através de regras.
RF07	A aplicação deve permitir a customização da cor de rótulo de classes genéricas.
RF08	A aplicação deve permitir a customização de tamanho do raio de classes genéricas ou de uma classe específica através de regras.
RF09	A aplicação deve permitir a customização de tamanho da borda de classes genéricas ou de uma classe específica através de regras.
RF10	A aplicação deve permitir a customização de espessura da relação(link) de classes genéricas.
RF11	A aplicação deve possibilitar o uso de imagens personalizadas para representar classes genéricas ou para representar uma classe específica através de regras.

5.1.1 Requisitos Não Funcionais

Os requisitos não funcionais do editor foram elaborados levando em consideração padrões encontrados em outras ferramentas. Esses requisitos possibilitam dentre outras coisas, uma ferramenta com melhor desempenho, usabilidade e disponibilidade. Abaixo estão em destaque os principais requisitos não funcionais do editor:

1. O sistema deverá ser disponibilizado na plataforma Web;
2. O editor, ao exibir grandes ontologias, ou seja, ontologias com dezenas de milhares de nós, deverá gerenciar o uso da memória de tal forma que não comprometa o desempenho geral do sistema. Em outras palavras, o editor deverá evitar o desperdício na alocação de recursos desnecessários em um determinado momento de tempo, assim gerenciando o uso desses recursos, alocando-os e desalocando-os conforme a sua efetiva necessidade;
3. A aba gráfica deverá apresentar desempenho de carregamento satisfatório. Para ontologias de até 1000 nós é esperado um tempo de carregamento na ordem de no máximo 5 segundos, porém, em cenários reais, espera-se que o tempo de carregamento fique entre 0,5 e 2 segundos (com uma boa conexão, bom tráfego de rede e sem sobrecarga do servidor). Para ontologias maiores, o tempo não deverá ultrapassar 10 segundos;
4. Requisições REST genéricas deverão possuir tempo máximo de resposta de 5 segundos, com *feedback* interativo ao usuário;
5. O sistema deve possuir uma duplicação de *containers* em servidores distintos juntamente a um *load balancer*, garantindo assim sua disponibilidade ao usuário;
6. A aplicação deverá realizar um controle de usuários para que os dados sejam consistentes, ou seja, para que o dado de um determinado usuário não reflita indevidamente em dados de outros usuários, garantindo assim sua integridade.

5.2 KGWE nos Sistemas de Apoio à Decisão

O editor KGWE é uma ferramenta importante no ciclo de criação, manutenção e utilização de SADs baseados em ontologias. A sua função, dentro desses SADs, é suprir a necessidade de criação e edição das ontologias por especialistas de domínio.

O KGWE pode ser integrado a SADs para criar e/ou editar suas ontologias. Alterações realizadas na ontologia pelos usuários no KGWE devem refletir imediatamente no SAD após a realização da publicação das alterações no editor. A comunicação entre o sistema de apoio à decisão e o KGWE ocorre por meio de APIs.

Contudo, no estágio atual do desenvolvimento desses SADs, ainda não foi possível implementar a sua integração com o KGWE. Ainda existem questões que precisam ser resolvidas, como a possível perda de dados (instâncias) em alterações complexas na estrutura de uma ontologia preexistente.

5.3 Notações Gráficas

Uma das hipóteses, discutida neste trabalho, foi que o uso de notações gráficas customizadas e baseadas em regras (definidas pelos próprios especialistas de domínio) deve beneficiar a visualização.

O KGWE possui interfaces simples, mas efetivas para modelar essas notações. Vale ressaltar também que esse mecanismo de customização não está preso a uma área específica do conhecimento, podendo ser aplicado em qualquer tipo de modelagem.

No [Capítulo 3](#) foi apresentado um exemplo de notações sobre filmes. A [Figura 17](#) mostra como foi possível atingir o mesmo nível de representação de informações que o exemplo dos filmes do IMDb, criando as regras diretamente no KGWE.

Figura 17 – Grafo de filmes

Fonte: Elaborada pelo autor.

Percebe-se na [Figura 17](#) que o mesmo nível de abstração de informações foi alcançado pois os elementos possuem:

- Nome para identificar os elementos ou nós;
- Cores que os diferenciam um dos outros;
- Tamanho variável de acordo com sua classe (total de filmes por gênero);
- Bordas representando informações extras da classe.

Para alcançar esse modelo de notação, vários ajustes de cores e tamanhos foram realizados, dentre eles:

- Ajuste de cor de fundo de classe de acordo com o gênero do filme;
- Ajuste de cor na borda de classe;
- Ajuste de cor de classe padrão;
- Ajuste de tamanho de classe de acordo com o número total de indivíduos que a classe possui;
- Ajustes de tamanho de bordas de classes para representar a quantidade de avaliações que cada filme possui.

Os ajustes realizados neste exemplo poderiam ser facilmente criados por usuários finais do editor, ou seja, pelos especialistas de domínio.

As cores dos rótulos e das ligações dos elementos foram mantidas com o valor padrão. A Figura 18 mostra com mais detalhes como foram realizados os ajustes de cores na ferramenta.

Figura 18 – Definições de cores sobre os filmes

Fonte: Elaborada pelo autor.

Procurou-se ir um pouco além do conteúdo encontrado no exemplo dos filmes. Algumas classes adicionais foram criadas para representar a relação entre gênero e filmes. Também foram cadastradas instâncias de classes que representam os prêmios obtidos no Oscar por cada filme. A Figura 19 mostra todos os indivíduos contidos nas classes "Titanic" e "Gone With The Wind". "Titanic" possui 11 óscares, enquanto "Gone With The Wind" possui 10. Com base nessas informações, foi criada uma regra adicional que diz que classes com 10 instâncias ou mais serão representadas com 60 *pixels*. Isto as tornam maiores que classes genéricas, que possuem 40 *pixels*. Além disso, existe ainda uma outra regra que define as classes que são subclasses de "Romance". De acordo com a regra, essas classes devem conter a mesma cor da classe pai, ou seja, devem herdar a cor da classe "Romance".

Figura 19 – Óscares por filme

IRI: http://semantic.icmc.usp.br/#Titanic	IRI: http://semantic.icmc.usp.br/#GoneWithTheWind
Individuals	Individuals
http://semantic.icmc.usp.br/#MelhoresEfeitosVisuais	http://semantic.icmc.usp.br/#OscarTecnicoCientifico
http://semantic.icmc.usp.br/#MelhorTrilhaSonoraDramatica	http://semantic.icmc.usp.br/#OscarHonorario
http://semantic.icmc.usp.br/#MelhorSom	http://semantic.icmc.usp.br/#MelhorRoteiro
http://semantic.icmc.usp.br/#MelhorFotografia	http://semantic.icmc.usp.br/#MelhorFotografia
http://semantic.icmc.usp.br/#MelhorFilme	http://semantic.icmc.usp.br/#MelhorFilme
http://semantic.icmc.usp.br/#MelhorFigurino	http://semantic.icmc.usp.br/#MelhorEdicao
http://semantic.icmc.usp.br/#MelhorEdicaoDeSom	http://semantic.icmc.usp.br/#MelhorDirecaoDeArte
http://semantic.icmc.usp.br/#MelhorEdicao	http://semantic.icmc.usp.br/#MelhorDirecao
http://semantic.icmc.usp.br/#MelhorDirecaoDeArte	http://semantic.icmc.usp.br/#MelhorAtrizCoadjuvante
http://semantic.icmc.usp.br/#MelhorDirecao	http://semantic.icmc.usp.br/#MelhorAtriz
http://semantic.icmc.usp.br/#MelhorCancaoOriginal	

Fonte: Elaborada pelo autor.

A Figura 17 mostra como ficou esse modelo representado graficamente pelo editor. É possível perceber que os filmes "Titanic" e "Gone With The Wind" são maiores que o filme "Pretty Woman", justamente por essas classes terem 10 ou mais óscares e que todas as subclasses de "Romance" possuem a mesma cor.

5.3.1 Modelo de Notação

Durante o processo de criação/edição de ontologias, especialistas de domínio têm a possibilidade de alterar as notações gráficas utilizadas. Ao final do processo, eles têm, a sua disposição, um modelo de notações (Figura 20) específico para as ontologias utilizadas. Esse modelo é um resumo de todas as definições e ajustes realizados pelos especialistas nas ontologias. Ele servirá como modelo de referência, tanto para os próprios especialistas, como para o compartilhamento entre outros profissionais.

O modelo de notações é dividido em grupos, sendo eles:

- Representações gráficas das classes: aqui é exibido qual tipo de representação será utilizada pelas classes genéricas das ontologias. Podem ser usados círculos, retângulos ou mesmo imagens. Classes customizadas podem ser representadas por símbolos personalizados;
- Esquema de cores das classes: aqui estão as regras que definem as cores de fundo de uma classe, de sua borda ou relação;
- Esquema de tamanho das classes: as regras contidas neste grupo são correspondentes aos ajustes de tamanho realizados, podendo estes ser realizados em círculos, retângulos, bordas, dentre outros. Assim como nos demais grupos, as regras são sempre aplicadas com base em condições pré-determinadas.

Regras criadas diretamente sobre classes tem precedência sobre regras criadas para subclasses, ou seja, se ambas afirmações forem válidas para uma determinada classe na ontologia, o parâmetro definido pela primeira regra de maior precedência será aplicado.

Caso o usuário deseje obter uma cor personalizada de classe (não disponível na seleção de cores do editor), o mesmo poderá definir a cor usando seu código hexadecimal diretamente na interface.

Para regras que utilizam o número de instâncias de classe para aplicar um determinado tamanho ou cor, é necessário informar o valor da condição ("maior ou igual a" e "menor ou igual a").

Figura 20 – Modelo de notações

Graph Settings | COLORS | SIZE | SHAPES / IMAGES | **OTHER SETTINGS** | X

Rect Padding
Image Padding
Link Arrow Padding
Font Properties
Force-directed graph properties
Replace link text
Notation Specifications
General Settings

Graphical representations of classes

OWL	Type	Representation	Description
owl:Class	Circle		Classes in the ontology are represented by circles

Class color scheme

OWL	Color Rule	Precedence	Component Color	Color code	Rule	Value
owl:Class	Background	1		#ffc400	Class equals =>	SustainabilityIndicator
owl:Class	Background	4		#2979ff	Class equals =>	All Classes
owl:Class	Border	1		#00e676	Class equals =>	SustainabilityIndicator
owl:Class	Border	1		#ff3d00	Class equals =>	All Classes
owl:Class	Label	1		white	Class equals =>	All Classes
owl:Class	Relation (link)	1		#76ff03	Class equals =>	All Classes

Class size scheme

OWL	Size Rule	Precedence	Size	Rule	Value
owl:Class	Radius	1	70px	Class equals =>	EconomicIndicator
owl:Class	Circle Border	2	5px	Subclass equals =>	EconomicIndicator

PRINT EXPORT

Fonte: Elaborada pelo autor.

5.4 Arquitetura da Aplicação

A arquitetura do KGWE é composta pelos componentes apresentados na [Figura 21](#). A interação entre as camadas se inicia quando o usuário realiza alterações em uma ontologia no KGWE. A comunicação avança através das camadas subjacentes até que as alterações sejam persistidas na base de dados.

Figura 21 – Arquitetura do editor

Fonte: Elaborada pelo autor.

A seguir, cada componente da arquitetura é detalhado:

1. **User:** representa ontologistas, especialistas de domínio ou responsáveis de forma geral pela ontologia;
2. **KGWE:** interface utilizada pelo usuário da ferramenta. Representa o *frontend* da aplicação. Componente responsável por transmitir e requisitar dados do *backend*;
3. **Controller e Model:** responsáveis pela lógica da aplicação. São os componentes que lidam diretamente com os dados alterados. Sua função é salvar esses dados em repositórios de dados (*triple store*), bem como se comunicar com outras APIs para delegar alguma tarefa;
4. **Triple Store (Data Layer):** Banco de dados no qual armazena a versão de trabalho atual da ontologia, ou seja, a versão que ainda não foi publicada oficialmente no SAD. Nesse banco de dados encontra-se também todas as configurações de interface (customizações) realizadas pelos especialistas de domínio;

5. **EUL**: representa a instanciação de um SAD. Componente responsável por receber uma ontologia alterada e persisti-la em sua base local para uso do sistema;
6. **Triple Store (Communication Layer)**: Banco de dados no qual armazena a versão da ontologia publicada no SAD, assim como os dados gerados pelos usuários finais da aplicação.

Vale ressaltar que a versão de trabalho da ontologia, ou seja, a versão no qual será modificada por um determinado período de tempo antes de sua publicação oficial, é salva na *triplestore* do KGWE. Depois de publicado, essas alterações serão refletidas na *triplestore* do SAD em uso.

O GitHub mantém todo histórico de alterações das ontologias, permitindo comparar a versão atual de trabalho com versões publicadas anteriormente na ferramenta.

5.5 Tecnologias utilizadas no desenvolvimento

O KGWE foi projetado para ser acessado por navegadores convencionais, como o Google Chrome, Firefox, Safari, Opera, entre outros. Por ser um sistema *WEB*, o frontend da aplicação é construído com HTML, CSS e JavaScript, porém com o suporte do *framework* Kotlin JS¹.

Foi utilizado um *wrapper*² do Kotlin para o node.js que simplifica configurações de construção da aplicação com o empacotador de módulo JavaScript *webpack*. O gerenciador de pacotes, o *Node Package Manager* (NPM), do node.js possui uma vasta quantidade de bibliotecas JavaScript disponíveis e possui suporte à instalação e gerenciamento de dependências.

Pensando na performance de renderização, utilizou-se a biblioteca REACT³ para construção dos componentes de interface. O REACT traz um poderoso algoritmo de manipulação da árvore *Document Object Model* (DOM) dos navegadores, este mecanismo utilizado é denominado Virtual DOM. O REACT é uma ferramenta muito produtiva e amplamente utilizada para desenvolvimento Web.

A biblioteca de componentes gráficos Material-UI⁴ é usada no KGWE. Ela é a biblioteca de componentes mais popular para REACT. Ela adota as regras de design do *Material Design*, proporcionando boa experiência de navegação ao usuário. O Material-UI foi utilizado para construir as interfaces da aplicação, provendo componentes tais como botões, campos de texto, rótulos, listas, dentre diversos outros componentes.

¹ <<https://kotlinlang.org/docs/reference/js-overview.html/>>

² <<https://github.com/JetBrains/create-react-kotlin-app/>>

³ <<https://reactjs.org/>>

⁴ <<https://material-ui.com/>>

O principal componente do editor, a aba do editor gráfico, utiliza a biblioteca JavaScript `d3.js`⁵ para renderizar os nós e suas ligações. O `d3` é uma biblioteca para manipulação e visualização de dados. Neste trabalho, o `d3` auxilia na criação de nós gráficos com recursos do HTML e CSS.

Para representar a estrutura de dados em um espaço tridimensional, utilizando um layout interativo direcionado por força, foi incorporado ao projeto a biblioteca `3d-force-graph`⁶ disponível no GitHub.

Para facilitar a construção da interface, utilizou-se também o *framework* `bootstrap`⁷. Ele permite reduzir a necessidade de código em CSS, disponibilizando um sistema de GRID de fácil utilização e diversos outros utilitários, como exemplo, o *Flex Layout*.

O *backend* do KGWE possui diversas APIs no padrão REST que disponibilizam funcionalidades, como a recuperação de classes da ontologia, inserção de um novo nó, entre outros recursos. O *backend* é executado sobre a *Máquina Virtual Java* (JVM) e faz uso do *framework* `Spring`⁸ utilizando mecanismos de injeção de dependências e outros facilitadores como o `Spring Boot`. As APIs fazem operações de gravação e consultas em base de dados de triplas (*triplestore*). O `Graphdb`⁹ foi utilizado como *triplestore* neste projeto. Sua interface gráfica permite executar consultas SPARQL diretamente pelo navegador, onde é possível também explorar os dados por uma visualização do grafo disponível. O protocolo de comunicação entre o *backend* da aplicação e a *triplestore* é o definido pela W3C, o SPARQL 1.1 Protocol.

Vale mencionar também que a IDE IntelliJ IDEA¹⁰, da JetBrains foi utilizada durante todo o desenvolvimento deste projeto.

5.6 Componentes da Interface

A seguir serão apresentados os principais componentes da interface do editor KGWE. Eles foram inspirados nas representações gráficas usadas pelo editor Protégé, com exceção da aba "Graph Editor". Esses componentes não são representações fiéis das funcionalidades contidas no Protégé e sim uma tentativa de simplificação dessas funcionalidades. Lembrando que o público alvo do KGWE são especialistas de domínio e não editores de OWL, e por isso, não precisam de todas as funcionalidades do Protégé. Ao mesmo tempo, tentou-se adotar características modernas, adotando conceitos mais recentes presentes no desenvolvimento web atual.

⁵ <<https://d3js.org/>>

⁶ <<https://github.com/vasturiano/3d-force-graph>>

⁷ <<https://getbootstrap.com/>>

⁸ <<https://spring.io/>>

⁹ <<http://graphdb.ontotext.com/>>

¹⁰ <<https://www.jetbrains.com/idea/>>

Colors

A seleção de cores na visualização de dados não é apenas uma escolha estética, é uma ferramenta crucial para transmitir informações. As cores selecionadas corretamente transmitem os dados subjacentes com precisão, em contraste com muitos esquemas de cores, comumente usados na visualização, que distorcem os relacionamentos entre os valores dos dados (SIMMON, 2013).

O KGWE permite ao especialista de domínio definir cores customizadas para representar entidades em uma ontologia. Na aba de cores, o especialista tem a disposição uma ferramenta importante, onde é possível customizar a cor de vários elementos do grafo, tais como:

- Cor de fundo de uma classe genérica;
- Cor da relação entre as classes;
- Cor dos rótulos;
- Cor da borda dos elementos.

Na [Figura 23](#) é apresentada a tela em que o usuário pode realizar as configurações de cores para qualquer ontologia. Todas essas características juntas tornam a representação mais robusta, propiciando ao especialista de domínio representações que possibilitam identificar mais rapidamente os elementos no grafo e até mesmo possibilitando a obtenção de novos *insights*.

Com a customização de cores, o especialista pode obter um ajuste mais preciso para representar características particulares do seu domínio. Um bom exemplo poderia ser a definição de uma cor característica para um determinado grupo de classes, facilitando sua identificação em ontologias mais complexas.

A aba de cores possui uma interface aprimorada, permitindo a seleção de cores por meio de caixas coloridas. É possível obter um ajuste mais fino arrastando horizontalmente alguns controles deslizantes, a tonalidade das cores sofrerá modificações instantaneamente.

O KGWE possui um paleta de cores para facilitar a visualização e seleção, mas, se mesmo assim o usuário não ficar satisfeito com as cores disponibilizadas, basta selecionar a opção de cor customizada e definir a cor manualmente por nome (em inglês) ou código hexadecimal. As cores reconhecidas diretamente por nome estão contidas nas definições do CSS, possuindo 140 cores no total. Já na definição direta por valor hexadecimal, é possível obter uma gama gigantesca de possibilidades. Com o padrão RGB (sRGB) o usuário tem disponível uma gama de mais de 16 milhões de cores.

Figura 23 – Configurações de cores

Fonte: Elaborada pelo autor.

Colors Rules

Até aqui, foi abordado apenas o uso de cores aplicadas a modelos genéricos, tais como a cor de fundo de classes em geral. Porém, no editor é possível ainda aplicar uma determinada cor para uma classe específica no grafo. Para isso, o editor possui a funcionalidade de regras que facilitam esses ajustes de representação. Com ela, o especialista pode, por exemplo, dizer que a classe "Pizza Mussarela" deve ter a cor amarela, enquanto que a classe "Pizza Calabresa" deve ser representada pela cor vermelha. Esse cenário é factível dentro das configurações de regras de cores da ferramenta.

Para se criar uma regra de cor, primeiramente é necessário estabelecer o tipo de elemento que se deseja aplicar a regra, sendo:

- Class - Indica que a cor será aplicada a classe que possuir o valor indicado;
- SubClass Of - Indica que a cor será aplicada a todas as classes que são subclasses onde a classe pai possui o valor indicado;
- Number Of Individuals - Indica que a cor será aplicada onde o número de indivíduos atinja o padrão estabelecido.

Caso o usuário selecione a opção "Class" ou "SubClass Of" na caixa de seleção, antes de selecionar a cor é necessário informar o nome da classe na qual pretende-se aplicar a regra.

Caso o usuário selecione a opção "Number Of Individuals" na caixa de seleção, antes de selecionar a cor é necessário informar o intervalo no qual a regra será aplicada. Ou seja, caso o usuário deseje aplicar uma determinada cor para classes que tenham entre 150 e 300 indivíduos,

o mesmo deverá preencher os campos "maior ou igual a" com valor "150" e o campo "menor ou igual a" com o valor "300".

Preenchido os valores, é possível realizar a seleção de cor do mesmo modo que foi mostrado anteriormente. A [Figura 24](#) mostra a janela de criação de regras.

Figura 24 – Regra de cor de fundo para classe

Fonte: Elaborada pelo autor.

Caso o especialista tenha definido uma regra que indique que classes com mais de 10 e menos de 100 indivíduos serão representados pela cor azul, então toda e qualquer classe que satisfaça essa condição será assim representada.

As regras de cores podem ser criadas tanto para a cor de fundo dos elementos, como para a borda dos elementos, baseando-se nos mesmos critérios de personalização.

Size

Outra função muito importante, para que fosse possível validar a hipótese apresentada neste trabalho, é a funcionalidade de tamanho dos elementos. Na interface, mostrada na [Figura 25](#), é possível definir de forma customizada o tamanho dos elementos renderizados.

Nas configurações de tamanho, o usuário poderá customizar os seguintes atributos:

- Diâmetro de um círculo (definido pelo raio);
- Borda de um círculo;
- Altura de um retângulo;
- Largura de um retângulo;

Figura 25 – Configurações de tamanho

Fonte: Elaborada pelo autor.

- Borda de um retângulo;
- Altura de uma imagem (em casos de uso de imagens personalizadas);
- Largura de uma imagem (em casos de uso de imagens personalizadas);
- Largura de um link (relação).

Os itens acima são sempre aplicados de forma genérica, ou seja, se classes são representadas por círculos e o usuário informa que deseja que círculos possuam 80 *pixels*, então todo círculo será renderizado com o tamanho especificado. Regras customizadas serão detalhadas logo a seguir.

Regras de Tamanho

O especialista de domínio pode ter o interesse de representar determinadas classes com tamanho diferenciado, sejam elas maiores ou menores que o tamanho padrão especificado para classes genéricas. Esses ajustes customizados podem ser realizados tanto em tamanho de círculos, retângulos e imagens, como também na borda de cada um desses elementos (exceto imagens, pois imagens personalizadas não são representadas com bordas no editor). No mesmo exemplo anterior, pode-se ter duas classes que representam "Pizza Grande" e "Pizza Pequena" e definir que "Pizza Grande" deve ser representada por círculos com diâmetro de 100 *pixels* e "Pizza Pequena" deve possuir metade disso, 50 *pixels*.

Para um exemplo simples, o ganho visual parece ser bem limitado, mas a medida em que se utilizam ontologias maiores e em casos onde há necessidade de representar muitos dados, o

especialista de domínio pode criar sua própria "arquitetura de informação" adicionando diferentes mecanismos visuais de representação.

Shape/Images

Esta funcionalidade permite que classes sejam representadas com imagens personalizadas. Usando essa funcionalidade, o especialista não fica mais preso a representações fixas, como círculos e retângulos, assim a variabilidade na representação cresce exponencialmente. Além de poder definir a representação de uma classe genérica com uma imagem personalizada, o especialista pode ainda criar regras pelo nome da classe e até mesmo para subclasses. No exemplo da ontologia de pizzas, pode-se usar imagens reais para cada elemento, como pode ser visto na [Figura 26](#).

Figura 26 – Formas e imagens personalizadas

Fonte: Elaborada pelo autor.

Other Settings

Em outras configurações, o especialista de domínio poderá realizar um ajuste fino na apresentação gráfica da ontologia. Ajustes de espaçamento, tipo de fonte, ajustes dos parâmetros

do gráfico direcionado por força, substituição de texto dos links automaticamente, entre outras configurações.

5.6.2 *Visualização de Indivíduos*

Nesta aba, é possível visualizar todos os indivíduos presentes na ontologia em uma única interface. Selecionando um indivíduo, é possível visualizar suas informações básicas, como seu IRI e suas anotações, ou realizar operações mais complexas como a definição de valores para propriedades. O usuário pode remover um indivíduo da lista ou criar um novo indivíduo, informando inclusive se o mesmo pertence a alguma classe.

Assim como outras abas do editor, o usuário pode publicar as alterações realizadas na ontologia, refletindo suas mudanças imediatamente no sistema de apoio à decisão.

5.6.3 *Hierarquia de Classes*

A aba de hierarquia de classes ([Figura 27](#)), apresenta uma visão clássica de árvore indentada para uma ontologia. Usuários de sistemas como o Protégé possuem uma certa familiaridade com este tipo de representação. Dentro dessa aba, o usuário tem a sua disposição as seguintes funcionalidades:

- Criar classe;
- Criar subclasse;
- Criar indivíduos;
- Criar label;
- Deletar nó;
- Alterar valor da label;
- Alterar relação de subclasse;
- Visualizar instâncias.

5.6.4 *3D Graph View*

Esta aba representa também uma visão muito importante dentro do editor. Existem atualmente pouquíssimos editores que trabalham com esse tipo de visualização. A ideia por trás desse tipo de representação é justamente a abertura de novas possibilidades para que o especialista de domínio se sinta à vontade para explorar novos modelos com uma interface de fácil utilização. A visão em três dimensões possui uma característica que representa um

Figura 27 – Hierarquia de classes

Fonte: Elaborada pelo autor.

diferencial em relação às demais visões. Com ela é possível fazer um melhor aproveitamento do espaço disponível em tela, ou seja, por conta das suas várias dimensões os nós são melhores distribuídos, e uma ontologia de tamanho complexo pode ser representada no mesmo espaço que uma ontologia simples em apenas duas dimensões.

A visualização em 3D, como pode ser visto na [Figura 28](#), permite ampliar ou reduzir a visão por meio de *zoom* dinâmico, bem como girar o modelo por completo, tanto na vertical como na horizontal, facilitando bastante a leitura dos nós.

5.6.5 Hierarquia de Propriedades de Objetos

Para visualizar as propriedades de objetos presentes numa ontologia, o usuário deve ir até a aba "Object Property Hierarchy". Aqui é possível visualizar todas as propriedades de objeto da ontologia. Clicando sobre uma propriedade, as informações do item selecionado podem ser visualizadas e editadas no painel lateral direito.

5.6.6 Hierarquia de Propriedades de Dados

Aqui estão disponíveis as propriedades de dados. Ao selecionar um item, informações como "sub-propriedade" e suas características (se é uma propriedade funcional ou não) serão exibidas como pode ser visto na [Figura 29](#). Tais informações são passíveis de edição.

Figura 28 – Visualização gráfica em três dimensões

Fonte: Elaborada pelo autor.

Figura 29 – Aba de propriedade de dados

Fonte: Elaborada pelo autor.

5.6.7 Versionamento, publicação e recuperação

Durante o desenvolvimento deste projeto, verificou-se a necessidade de realizar o versionamento das ontologias de modo a garantir que os seguintes dois requisitos fossem atendidos:

- Recuperabilidade: sabe-se que o *backup* de dados de um servidor web é fundamental para garantir a recuperação das informações de forma íntegra em casos de eventuais panes,

desastres ou ataques. O *backup* das informações da ontologia é feito de forma automática em cada publicação nos servidores do GITHUB;

- Rastreabilidade: isso permite o acompanhamento das versões publicadas. As correções de problemas ocorridos durante todo ciclo de vida da ontologia podem ser visualizadas facilmente na ferramenta, onde é possível indicar tanto o nome do usuário que realizou a alteração, como também a mensagem utilizada para identificar aquela publicação.

O usuário pode verificar dentro da própria ferramenta quais alterações foram realizadas em versões publicadas anteriormente. A [Figura 30](#) mostra como é a interface que apresenta as diferenças identificadas entre a versão publicada selecionada e a versão anterior.

Figura 30 – Tela contendo as mudanças realizadas na ontologia

Fonte: Elaborada pelo autor.

O uso do versionamento GIT, a possibilidade de construção de ontologias de forma colaborativa por vários usuários, bem como a visualização e recuperação de publicações anteriores conferem ao KGWE um grande diferencial quando comparado a outros editores disponíveis.

5.6.8 Ajuda

O item de ajuda tem como objetivo auxiliar o usuário no entendimento do uso de todas as funções do sistema. Nele, o usuário irá encontrar textos informativos com a finalidade de descrever passo a passo cada funcionalidade disponível dentro do editor. Há também diversos vídeos que acompanham os textos e mostram detalhadamente como é realizada a edição de uma ontologia de forma prática, conforme é mostrado na [Figura 31](#). A ajuda/documentação é um importante mecanismo que auxilia a usabilidade da ferramenta.

Figura 31 – Ajuda com vídeo auxiliar

Fonte: Elaborada pelo autor.

5.6.9 Editor de JSON-LD

Uma outra visão da ontologia pode ser obtida pela aba "JSON-LD Editor". O formato JSON-LD é muito utilizado dentro do contexto da web semântica. É um formato de serialização que facilita o entendimento por desenvolvedores, já que é baseado no formato JSON tradicional. Essa aba permite a edição das informações diretamente em cada linha, como pode ser visto na [Figura 32](#).

Essa aba não é voltada aos usuários finais do sistema, mas a desenvolvedores que precisam interagir num nível mais baixo com a ontologia. Ela provavelmente não será disponibilizada na versão final do programa.

No editor de JSON-LD é possível editar várias informações da ontologia. O JSON-LD não é uma aba para criar novas classes, instâncias ou propriedades. A principal função aqui é realizar ajustes rápidos em rótulos ou nome de classes que possuam informações incorretas na base de dados.

Figura 32 – Aba de edição JSON-LD

Fonte: Elaborada pelo autor.

5.7 Armazenamento e recuperação dos dados

Todo o armazenamento de dados é realizado em uma *triplestore*. Para este projeto, optou-se pela utilização do programa GraphDB, porém outras *triplestores*, como o Blazegraph, podem ser utilizadas, já que o padrão de comunicação (protocolo) é o mesmo entre essas soluções.

5.7.1 Repositório das notações

Ao instalar a aplicação, por padrão é criado um repositório denominado KGWE. Esse repositório é o responsável por manter as informações de customização do usuário.

Como estratégia, foi criada uma ontologia OWL simples para mapear cada informação passível de customização. Dessa maneira, as alterações realizadas pelo usuário nas notações e regras associadas são armazenadas na triplestore e recuperadas sob demanda pela aplicação.

Futuramente, pode-se expandir essa ontologia a fim de abranger também outras características customizáveis que hoje não estão presentes no KGWE.

5.7.2 Manipulação de dados com SPARQL

Qualquer alteração, realizada em uma ontologia pelo especialista de domínio, deve se refletir em sua respectiva base de dados imediatamente após a efetivação da alteração. As operações gráficas em uma ontologia devem resultar em operações de inserção, alteração, remoção e recuperação de dados na triplestore. Essas operações são possíveis devido ao uso da linguagem SPARQL em sua versão 1.1¹¹ definida pelo consórcio internacional W3C.

O exemplo a seguir ([Código-fonte 1](#)), mostra a recuperação de classes da ontologia para serem exibidas graficamente na ferramenta.

Código-fonte 1 – Consulta SPARQL para recuperar classes da ontologia

```
1: PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2: PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3: PREFIX owl: <http://www.w3.org/2002/07/owl#>
4: PREFIX ui: <http://purl.org/biodiv/semanticUI#>
5: CONSTRUCT {
6:   ?parent #rdf:type ?type;
7:           rdfs:label ?parentLabel.
8:   ?child #rdf:type ?childType;
9:          rdfs:label ?childLabel;
10:          rdfs:subClassOf ?parent.
11: }
12: WHERE {
13:   ?parent rdfs:subClassOf param1;
14:          a ?type;
15:          rdfs:label ?parentLabel.
16:   ?child rdfs:subClassOf ?parent;
17:          a ?childType;
18:          rdfs:label ?childLabel.
19:   # Get ride of (i) class is a subclass of itself and (ii)
      blank nodes.
20:   FILTER (?parent != ?child && !isBlank(?child) && !isBlank(?
      parent))
21:
22:   # This filter gets rid of classes that are not immediate
      subclasses.
23:   # It gets rid of children classes that are also grandchildren
      classes
24:   FILTER NOT EXISTS {
25:     ?child2 rdfs:subClassOf ?parent.
```

¹¹ <https://www.w3.org/TR/sparql11-overview/>

```
26:      ?child rdfs:subClassOf ?child2 .
27:      FILTER (?parent != ?child2 && ?child != ?child2) # ?? ?child
      = ?grand2)
28:  }
29: }
```

Os parâmetros dentro da *query* são substituídos automaticamente pelos valores recebidos na requisição do editor (rodando no navegador do usuário). Na sequência, a *query* é executada e os resultados repassados para o cliente que solicitou a operação.

Existem dezenas de artefatos SPARQL dentro do projeto KGWE, sendo que alguns são apenas para consultas, já outros possuem instruções para inserir, remover ou alterar dados da ontologia.

5.8 Considerações Finais

Este capítulo apresentou a ferramenta KGWE, o propósito deste editor, sua ligação com os SADS e suas funcionalidades projetadas e implementadas durante todo o ciclo deste projeto de pesquisa.

Em outros editores gráficos de OWL, classes são comumente representadas por elipses ou círculos, com cores e bordas sem muitas variações. O KGWE possui uma ferramenta poderosa de edição dessas representações. Seus mecanismos de visualização, aliados a customização dos elementos por meio de regras, pode tornar a apresentação dos elementos de uma ontologia mais intuitiva e mais próxima a representações comuns na área de domínio do especialista.

Porém, ainda existem algumas lacunas a serem exploradas no KGWE para atender a customização de todos os elementos em OWL. Neste trabalho foi abordada apenas a customização de representações de classes dentro da ontologia, porém outros elementos visuais, como propriedades das classes, também podem ser explorados.

O próximo capítulo traz os resultados das avaliações realizadas na ferramenta.

AVALIAÇÃO

Desde o início deste projeto, buscou-se identificar as reais necessidades dos especialistas no que se refere à edição das ontologias. Para isso, várias avaliações foram feitas durante o desenvolvimento do KGWE para se detectar possíveis necessidades e verificar se os critérios adotados durante os ciclos de desenvolvimento estavam sendo bem aceitos pelos especialistas, atendendo plenamente às suas necessidades. Este capítulo apresenta essas avaliações e seus resultados.

6.1 Avaliação de protótipos

Nesta etapa foram levantadas características pertinentes às representações visuais que o editor de ontologias proposto deveria possuir.

6.1.1 *Protótipo de Vídeo*

Um protótipo de vídeo foi construído a fim de se obter, por meio de um questionário, um modelo de arquitetura de software inicial, ou seja, esses usuários foram questionados sobre o que eles esperavam de um editor de ontologias. Para isso, foi criado um vídeo explicativo sobre as várias funcionalidades presentes no *mockup*. A avaliação foi realizada com 2 especialistas de domínio entre os meses de março e abril de 2018. A apresentação do vídeo mostrando as funcionalidades de um protótipo de software com uma narrativa bem organizada, tornou mais simples o entendimento do problema e ajudou na concepção de novas ideias ou sugestões.

Durante o vídeo, foram realizados diversos questionamentos a esses usuários. Durante essa avaliação, foi solicitado também que os 2 participantes utilizassem o editor Protégé para que fosse possível comparar as funcionalidades do protótipo desenvolvido. Esse protótipo serviu como um estudo comparativo inicial em relação as funcionalidades disponibilizadas pelo editor Protégé (utilizado para criar a versão inicial da ontologia Sustenagro).

Ao final do processo, os participantes foram orientados a responderem um questionário online com 11 perguntas referente ao vídeo, sendo:

1. No modo de edição visual, ao clicar em um item (uma classe por exemplo), o editor deverá exibir informações desse elemento no painel lateral direito. Essa disposição é adequada?
2. Referente a questão anterior, caso tenha em mente um outro fluxo de exibição para essas informações, faça essa anotação:
3. As relações entre entidades devem possuir flexibilização das *labels*? (por exemplo, ao invés de owl:subclass of, o próprio usuário poderia definir a *label* que considerar mais apropriada.
4. Na edição das informações do menu lateral direito, quais informações são essenciais e quais não são? (pode-se considerar como modelo o Protégé)
5. Itens do menu de configuração e disposição dos elementos apresentados no vídeo (*mockup*) estão adequados?
6. Referente a questão anterior, caso tenha alguma observação:
7. Quais "telas" são importantes no Protégé e quais não são?
8. Quais características do Protégé deveriam ser alteradas e melhoradas?
9. Quais características são importantes, mas estão ausentes no Protégé?
10. Características visuais como tamanho dos elementos, descrições dos campos, posição e cores destes elementos estão adequados no Protégé? (mencione aspectos de melhoria para estes itens)
11. Quais outras características que não foram mencionadas você considera relevante?

Com as respostas obtidas, durante a avaliação do protótipo de vídeo, foi possível verificar quais telas eram importantes para o editor, e quais não eram. Foi possível também verificar quais características os especialistas consideravam importantes para um editor de ontologias. Os especialistas foram questionados também sobre quais características consideravam importantes na ferramenta Protégé (que também foi apresentada no vídeo), e quais funcionalidades necessitavam de melhorias. As respostas ajudaram a construir uma ferramenta mais assertiva com as expectativas desses usuários.

6.1.2 Protótipo funcional

Posteriormente, foi realizada uma reunião no dia 12/04/2018 entre a equipe da USP e os especialistas em sustentabilidade e ontologias da Embrapa, na Embrapa Informática Agropecuária em Campinas, São Paulo, onde foi apresentado um protótipo inicial da ferramenta.

O protótipo desenvolvido era um protótipo navegável funcional, ou seja, as funcionalidades de interface estavam disponíveis para utilização no navegador, como em uma versão final, porém os dados ou alterações realizadas pelo usuário não eram salvos em um banco de dados. Esse protótipo era composto de um *frontend* da aplicação construído usando-se HTML, CSS e Javascript. Ele incorporava sugestões de funcionalidades fruto das interações com o protótipo de vídeo.

Devido ao pouco tempo disponível neste encontro, procurou-se realizar um *brainstorming* com os especialistas a fim de explorar suas ideias, conhecer melhor as ferramentas que habitualmente utilizavam e também entender as suas necessidades de desenvolvimento de funcionalidades inovadoras. Após a reunião, foi agendado uma nova discussão em Maio de 2018 para que os especialistas pudessem colaborar com *feedbacks* a respeito da versão do editor.

Com essa versão do protótipo, foi possível validar que o caminho adotado no desenvolvimento estava correto em relação às funcionalidades esperadas para o editor. Foi possível também avaliar quais tipos de bibliotecas de software seriam necessárias para a continuidade do desenvolvimento do editor.

6.2 Estudos de caso SustenAgro e Nano

Uma vez que o editor KGWE se tornou completamente funcional, foi conduzido um estudo de caso usando as duas ontologias desenvolvidas em projetos de cooperação com a Embrapa, o SustenAgro e Nano (as ontologias têm o mesmo nome que seus respectivos projetos). Esses estudos de caso foram realizados com especialistas de domínio em sustentabilidade da Embrapa/APTA. Os especialistas foram orientados a utilizarem a ferramenta em um cenário real de edição para cada uma das ontologias e preencherem um questionário.

A versão do editor utilizada para esse estudo de caso contempla todas as funcionalidades planejadas desde o início do desenvolvimento, bem como novas funcionalidades sugeridas durante as avaliações dos protótipos de vídeo e funcional. É importante mencionar que a avaliação do estudo de caso foi realizada usando apenas o editor KGWE, mas que, futuramente, se pretende integrar o editor aos SADs SustenAgro e Nano.

6.2.1 Avaliação do KGWE

Esta avaliação foi realizada com dois especialistas em sustentabilidade da Embrapa/APTA, que trabalhavam nas ontologias SustenAgro ou Nano, entre os meses de setembro e

novembro de 2020.

Foi disponibilizado um link para a ferramenta para que cada especialista avaliasse as funcionalidades do editor e depois respondesse um questionário disponibilizado online. Para esta avaliação não foi disponibilizado nenhum roteiro predeterminado, os especialistas estavam livres para utilizarem todas funcionalidades disponibilizadas pela ferramenta. Essa tarefa foi realizada sem o acompanhamento de nenhum instrutor/entrevistador. O intuito de solicitar esse tipo de atividade, como navegar na ferramenta e realizar alterações em classes e instâncias como em uma versão final da ferramenta, foi para que esses especialistas obtivessem insumos suficientes para responderem os questionários posteriormente disponibilizados.

Com a finalidade de avaliar a satisfação de cada um dos especialistas de domínio no uso da ferramenta foram realizadas um total de três perguntas utilizando uma escala de quatro pontos, sendo:

- **(0):** Nada satisfeito(a);
- **(1):** Moderadamente satisfeito(a);
- **(2):** Muito satisfeito(a);
- **(3):** Extremamente satisfeito(a).

Para essa avaliação foram realizadas um total de 3 perguntas. A [Tabela 3](#) apresenta, para cada uma das escalas apresentadas anteriormente (colunas), uma sumarização da quantidade de especialistas que escolheram esta opção do questionário.

Tabela 3 – Nível de satisfação no uso do KGWE

Pergunta	Nível de satisfação			
	(0)	(1)	(2)	(3)
Satisfação para a funcionalidade de customização de cores?				2
Satisfação referente a usabilidade da ferramenta?			1	1
Satisfação para a funcionalidade de customização de tamanho dos elementos?			1	1

Para que fosse possível identificar **quais funcionalidades os especialistas de domínio consideravam mais importantes no editor** foram apresentadas doze funcionalidades existentes na ferramenta a fim de se obter o grau de relevância de cada uma delas. Elas também utilizaram uma escala de quatro pontos, onde:

- **(0):** Nada relevante;
- **(1):** Moderadamente relevante;

- **(2)**: Muito relevante;
- **(3)**: Extremamente relevante.

A [Tabela 4](#) apresenta, para cada uma das escalas apresentadas anteriormente (colunas), uma sumarização da quantidade de especialistas que escolheram cada opção do questionário.

Tabela 4 – Nível de relevância da funcionalidade

Funcionalidade	Relevância			
	(0)	(1)	(2)	(3)
Background Color (class)				2
Relation Color				2
Class Label Color				2
Class Border Color				2
Background Color Rules				2
Border Color Rules				2
Circle Radio Width			1	1
Circle Border Width			1	1
Circle Radius Rules			1	1
Circle Border Width Rules			1	1
Stroke Width			1	1
Stroke Width Rules			1	1

Por fim, os especialistas responderam duas questões que ajudaram a determinar se o KGWE efetivamente atendia alguns critérios estabelecidos durante o desenvolvimento, como a tentativa de simplificação da interface, e também para determinar se o editor proposto proporcionou interfaces mais eficientes e eficazes.

Tabela 5 – Perguntas dicotômicas sobre as funcionalidades do editor

Pergunta	Sim	Não
De modo geral, considera as representações visuais do KGWE mais eficientes e eficazes do que editores que você tradicionalmente utilizaria para edição das ontologias abordadas nos estudos de caso?	2	
Os conceitos presentes na ferramenta (OWL) foram simplificados quando comparados aos editores que você tradicionalmente utilizaria para edição das ontologias abordadas nos estudos de caso?	2	

Com base nas respostas obtidas nessa avaliação, foi possível observar que os especialistas estavam satisfeitos ou extremamente satisfeitos com as funcionalidades de customização de cores e tamanhos dos elementos, assim como para a usabilidade da ferramenta. Verificou-se também que, de acordo com os especialistas de domínio, as representações visuais disponibilizadas pelo KGWE são mais eficientes e também eficazes que outras ferramentas utilizadas anteriormente.

Por fim, constatou-se que os conceitos OWL disponibilizados no KGWE foram simplificados quando comparados a editores anteriores utilizados para edição dessas ontologias.

6.3 Usabilidade: Avaliação Heurística

Medir a usabilidade de editores de ontologias serve para ajudar no esforço contínuo de melhoria de suas interfaces. Na avaliação heurística, um pequeno conjunto de avaliadores examina a interface e julga sua conformidade com um conjunto de princípios estabelecidos de usabilidade (as "heurísticas") (NIELSEN, 1994).

A avaliação heurística foi realizada por meio das 10 Heurísticas de usabilidade de Nielsen (NIELSEN, 1995) em novembro de 2020. O método consiste em localizar problemas de usabilidade nas interfaces, classificá-los e atribuir um grau de severidade a eles. Os problemas identificados pelos avaliadores são categorizados em cada uma das heurísticas utilizadas, sendo elas:

1. **Visibilidade do status do sistema:** o sistema deve sempre manter os usuários informados sobre o que está acontecendo por meio de *feedback* apropriado dentro de um tempo razoável;
2. **Correspondência entre o sistema e o mundo real:** o sistema deve falar a linguagem do usuário, com palavras, frases e conceitos familiares ao usuário, em vez de termos orientados ao sistema. Deve seguir as convenções do mundo real, fazendo as informações aparecerem em uma ordem natural e lógica;
3. **Controle e liberdade do usuário:** os usuários podem escolher funções do sistema por engano e precisarão ter uma saída de emergência clara para deixar o estado indesejado sem precisar passar por um extenso diálogo. Prover funções "desfazer" e "refazer";
4. **Consistência e padrões:** os usuários não devem adivinhar que palavras, situações ou ações diferentes significam a mesma coisa. Siga as convenções da plataforma;
5. **Prevenção de erros:** melhor do que boas mensagens de erro, é um design cuidadoso que evita a ocorrência de um problema. Elimine condições propensas a erros ou verifique-as e apresente aos usuários uma opção de confirmação antes que eles se comprometam com a ação;
6. **Reconhecimento em vez de lembrança:** minimize a carga de memória do usuário, tornando objetos, ações e opções visíveis. As instruções de uso do sistema devem ser visíveis ou facilmente recuperáveis sempre que necessário;

7. **Flexibilidade e eficiência de uso:** permite que os usuários adaptem ações frequentes. Provê atalhos e aceleradores para aumentar a velocidade de interação para usuários experientes;
8. **Estética e design minimalista:** os diálogos não devem conter informações irrelevantes ou raramente necessárias. Qualquer unidade extra de informação em um diálogo concorre com as unidades relevantes de informação e diminui sua visibilidade relativa;
9. **Ajude os usuários a reconhecer, diagnosticar e recuperar-se de erros:** as mensagens de erro devem ser expressas em linguagem simples (sem códigos), indicar com precisão o problema e sugerir construtivamente uma solução;
10. **Ajuda e documentação:** embora seja melhor um sistema que possa ser usado sem documentação, é necessário fornecer ajuda e documentação. Essas informações devem ser fáceis de encontrar, focadas na tarefa do usuário e não muito extensas.

6.3.1 Severidade

Além de encontrar e classificar os problemas durante a inspeção da interface, os avaliadores devem também classificá-los quanto a sua severidade. Segundo Nielsen (1994), esses problemas podem ser classificados por severidade conforme a lista a seguir:

0. **Sem Importância:** não afeta o uso do sistema;
1. **Cosmético:** não precisa ser corrigido, a menos que haja tempo extra disponível;
2. **Simples** - este tipo de correção deve receber baixa prioridade;
3. **Grave** - importante corrigir, por isso deve ser dada alta prioridade;
4. **Catastrófico** - deve ser corrigido antes do produto ser disponibilizado ao usuário.

Com base nessa classificação, os avaliadores foram orientados a classificarem cada um dos problemas encontrados quanto ao seu nível de severidade.

6.3.2 Participantes

Alunos de pós-graduação com experiência em computação foram usados como avaliadores. O perfil básico desses avaliadores inclui:

- Experiência em usabilidade;
- Experiência em modelagem de ontologias em outros editores (como o Protégé, yEd, Xmind, etc).

Segundo Nielsen (1994), de três a cinco avaliadores são suficientes para identificar grande parte dos problemas. Esta avaliação foi realizada por 3 avaliadores.

6.3.3 Cenários

A avaliação foi dividida em dois cenários, um para cada objetivo geral de avaliação. O intuito geral foi avaliar o uso da interface hierárquica de classes, que usa uma lista indentada, disponibilizada pela KGWE, juntamente com o modo gráfico da ferramenta (grafo direcionado por força):

- **Cenário 1:** na hierarquia de classes, crie uma classe, crie uma subclasse, atribua um *label*, insira um indivíduo e remova um nó existente;
- **Cenário 2:** na interface gráfica, crie uma classe, crie uma subclasse, atribua um *label*, insira um indivíduo e remova um nó existente.

As avaliações de cada voluntário foram coletadas, consolidadas e analisadas. Os resultados serão apresentados na próxima seção.

6.3.4 Avaliação

Os resultados das avaliações heurísticas de Nielsen, realizadas pelos 3 avaliadores, podem ser conferidos a seguir. Os problemas identificados pelos avaliadores foram associados a cada uma das heurísticas e relacionados ao seu respectivo grau de severidade.

Tabela 6 – Avaliação heurística realizada pelo primeiro avaliador

Avaliador 1		
Problema	Heurística	Severidade
Ao salvar uma classe, aparece um erro na tela, apesar de a classe ter sido criada	5	4
Não é claro como criar uma classe na interface. Sugestão: Botão fixo como na hierarquia	6	3
Criação de classes em hierarquia e na interface são diferentes	4	2
Muito espaço entre as classes na interface, tive que dar muito zoom-in e zoom-out para localizar a classe que tinha criado	3	3
Classe não foi deletada/excluída	3	4

Tabela 7 – Avaliação heurística realizada pelo segundo avaliador

Avaliador 2		
Problema	Heurística	Severidade
Ao clicar em criar uma subclasse de <i>Thing</i> , na aba de hierarquia, preenchendo o campo, e o <i>label</i> e selecionando PT, aparece erro. Porém verificando depois, a classe foi criada. Talvez seja bom trazer uma tela diferente para erro, alertando um usuário mais leigo	5	4
Ao criar uma classe via GRAPH EDITOR é atualizada a interface, porém não volta onde o usuário estava. Se ele criou uma classe, talvez seja ideal após o carregamento, aparecer em tela a classe criada, para que ele possa terminar as ações (criar <i>data properties</i> , indivíduos e etc). Dessa forma, após o <i>refresh</i> , o usuário pode ficar meio perdido, e ter que procurar onde está a classe criada	6	1

Tabela 8 – Avaliação heurística realizada pelo terceiro avaliador

Avaliador 3		
Problema	Heurística	Severidade
Object Property Hierarchy apresentou tempo maior que 8 segundos para carregamento das informações	1	2
Desfazer e refazer operações não está funcionando na aba gráfica	3	3
Documentação está incompleta	10	0

Tabela 9 – Sumarização dos problemas identificados

Avaliador	Severidade				
	Nível 0	Nível 1	Nível 2	Nível 3	Nível 4
1	0	0	1	2	2
2	0	1	0	0	1
3	1	0	1	1	0
Total	1	1	2	3	3
Média (%)	10	10	20	30	30

De acordo com o resultado dessa avaliação, foi possível identificar vários problemas que não estavam realmente aparentes. Esses problemas impactavam negativamente a usabilidade da ferramenta. A execução dessas avaliações por avaliadores experientes permitiu identificar problemas importantes para a usabilidade da ferramenta.

A heurística mais violada foi a de controle e liberdade do usuário. Dar opção de desfazer e refazer é um passo importante e que precisa ser aprimorado no editor. Não menos importante,

é preciso que em situações onde ocorram erros indesejados, o usuário não fique preso a situação sem saber qual ação tomar. Melhorias nesse sentido estão sendo desenvolvidas na ferramenta.

Apesar da alta severidade em boa parte dos problemas apresentados, muitos deles já foram corrigidos e os demais estão em uma fila para futura correção.

6.4 Avaliação pelo método Think-Aloud

Think aloud (DAVEY, 1983) é um método de avaliação de interfaces no qual os participantes falam em voz alta quaisquer pensamentos que apareçam em sua mente enquanto completam uma tarefa usando a interface (CHARTERS, 2003). Ao final da avaliação, geralmente é feito um *debriefing* dos participantes onde eles podem comentar sobre quaisquer dificuldades, pontos positivos, etc. Esses comentários podem destacar pontos importantes para os usuários que podem levar a *insights* importantes sobre a interface.

Em geral, a pesquisa na literatura sobre o método *Think-Aloud* mostra sua forte base teórica e confirma seu valor como uma forma de explorar os processos de pensamento dos indivíduos (CHARTERS, 2003).

Para esta avaliação, dois especialistas de domínio da Embrapa/APTA foram orientados a realizar cinco tarefas básicas no editor, usando a ontologia SustenAgro, e falar em voz alta seus pensamentos durante o processo. Essas observações foram coletadas e são apresentadas na Tabela 10 juntamente com as tarefas realizadas. Também foram adicionados comentários obtidos ao final de cada experiência durante o *debriefing* do participante.

Essas avaliações foram conduzidas entre os meses de setembro e novembro de 2020. Os encontros foram conduzidos totalmente de forma online por meio de reuniões agendadas devido a pandemia.

Tabela 10 – Avaliação das tarefas conduzidas durante as entrevistas

Tarefa	Comentário
Criar uma nova classe	<ul style="list-style-type: none"> Melhorar a busca da classe. Difícil encontrá-la após a criação.
Criar uma nova instância	<ul style="list-style-type: none"> Problema no <i>refresh</i> da hierarquia de classes. Precisou forçar a atualização. Apresentar um <i>tooltip</i> com a <i>annotation</i> do indivíduo deve melhorar a visualização.
Adicionar um novo <i>label</i> na classe	<ul style="list-style-type: none"> Limitar a criação de uma <i>label</i> por linguagem (PT ou EN).
Criar uma nova subclasse	<ul style="list-style-type: none"> Informar que uma determinada URI já existe na ontologia e impedir a criação de uma nova classe ou subclasse.
Mudar a cor de fundo de uma classe	<ul style="list-style-type: none"> No lugar de escrever o nome da classe, seria mais interessante apresentar a hierarquia para o usuário selecionar a classe diretamente.

Em adição ao método *Think-Aloud*, também se pediu aos especialistas de domínio que escolhessem, com base em uma lista pré-determinada, quais tipos de notações, baseados em regras customizadas, eram essenciais para o KGWE. A lista a seguir contém todos os itens que os especialistas consideraram importantes:

- Cor de fundo de todas as classes;
- Cor de fundo de uma classe específica (por nome);
- Cor de fundo de todas subclasses de uma classe específica (por nome);
- Cor de fundo de uma classe pelo número de indivíduos que ela possui;
- Cor das relações (link);
- Cor das *labels*;
- Cor das bordas de classes;
- Cor da borda de uma classe específica (por nome);

- Cor da borda de todas subclasses de uma classe específica (por nome);
- Cor da borda de uma classe pelo número de indivíduos que ela possui;
- Tamanho padrão de todas as classes;
- Espessura padrão da borda de todas as classes;
- Tamanho de uma classe específica (por nome);
- Espessura da borda de uma classe específica (por nome);
- Tamanho de uma classe pelo número de indivíduos que ela possui;
- Espessura da borda de uma classe pelo número de indivíduos que ela possui.

Com os resultados dessa avaliação, foi possível verificar que os especialistas sentem necessidade de melhorias em algumas funcionalidades, como por exemplo na seleção de classes na tela de customização de cores. Essas melhorias estão sendo implementadas ou estão em uma lista para implementação futura na ferramenta.

6.5 Avaliação da eficiência das notações customizadas

Para avaliar a eficiência do uso de notações customizadas para visualização de instâncias no editor, foi proposto um experimento envolvendo três versões distintas de uma mesma ontologia, a Nano, e três diferentes editores para avaliação de diferentes cenários. Cada ontologia possui exatamente as mesmas classes, tendo ao todo 86 classes. O que difere entre essas ontologias é que somente uma dessas classes possui dois indivíduos, as demais classes da ontologia possuem apenas um indivíduo.

Cada uma das três diferentes versões da ontologia foram submetidas para avaliação dos especialistas, cada uma em um editor diferente, sendo eles o KGWE, WebVOWL e WebProtégé.

Previamente, os especialistas foram informados sobre o funcionamento da representação de indivíduos em cada uma das ferramentas. Para o KGWE, o especialista foi orientado a criar sua própria representação de classe para o cenário proposto, ou seja, era necessário criar uma regra customizada que indicaria visualmente que classes, com um número de duas instâncias, seriam apresentados com tamanho maior ou cor diferenciada.

Os especialistas não sabiam previamente o nome da classe que continha os dois indivíduos na ontologia.

Durante a conversa, os especialistas de domínio foram orientados então a identificar qual classe na ontologia possuía exatamente dois indivíduos.

Toda a conversação foi gravada e analisada posteriormente para obtenção do tempo total de cada atividade. Vale ressaltar que o tempo de configuração (criação da regra customizada) não foi contabilizado no tempo medido, pois o tempo total deveria refletir apenas a busca da informação na ontologia.

Os resultados contendo a média dos tempos podem ser visualizados na [Tabela 11](#).

Tabela 11 – Avaliação de eficiência de visualização de instâncias

Editor	Tempo médio (min:seg)
KGWE	0:09
WebVOWL	4:21
WebProtégé	n/a

Não foi possível obter o tempo para o editor WebProtégé, pois a ferramenta se mostrou bastante limitada para o cenário proposto. Realizar uma pesquisa para cada classe presente na ontologia, a fim de se conhecer o número de indivíduos de cada uma, seria uma tarefa absurdamente desgastante em uma ontologia com centenas de classes.

Outras técnicas poderiam ser utilizadas para essa avaliação, como o rastreamento ocular, do inglês *Eye tracking*. Esse é um método para estudar a atenção visual do usuário. Com ele, é possível determinar em que área ele fixa sua atenção, por quanto tempo e a sequência que ele segue na exploração visual ([OKAMOTO; FURQUIM, 2020](#)). Nesta avaliação, utilizou-se apenas a métrica apresentada, com aferição do tempo total da atividade.

O resultado dessa avaliação mostrou que o KGWE proporcionou um método visual de busca muito eficiente para instâncias. Isso evidencia que alguns dos editores disponíveis na Web não fornecem alternativas eficientes de visualização em determinados cenários. É óbvio que se você pode destacar uma configuração de interesse, como uma classe que tem 2 instâncias, será muito mais eficiente encontrar exemplos dela. O problema é que essa informação não está disponível durante a criação dos editores, mas está disponível durante seu uso. Como apenas o KGWE permite esse tipo de configuração pelos usuários, ele tem uma grande vantagem em relação aos outros editores.

6.6 Considerações Finais

A [Tabela 3](#) mostra que os especialistas de domínio estão muito satisfeitos ou extremamente satisfeitos com o uso do KGWE e que as representações visuais do editor são mais eficientes e eficazes quando comparadas a ferramentas que tradicionalmente esses especialistas utilizavam. Foi possível validar também que o editor proporcionou interfaces em que os conceitos da OWL foram simplificados quando comparados a ferramentas anteriores de trabalho.

A avaliação heurística permitiu identificar diversos problemas não encontrados em testes funcionais realizados durante o desenvolvimento da ferramenta ou mesmo posteriormente, durante as avaliações com os especialistas. A correção desses problemas proporciona aos usuários uma ferramenta mais robusta e com melhor usabilidade.

Com o método *Think aloud* os especialistas puderam contribuir com diversas sugestões de melhorias. Os especialistas apontaram também alguns erros que necessitavam de ajustes na interface do editor.

A avaliação da eficiência usando-se notações mostrou que a ferramenta possui um método visual de busca muito eficiente para instâncias da ontologia.

Ainda há espaço para muitas melhorias no KGWE, pois novas abordagens em diferentes tipos de regras customizadas podem ser adotadas futuramente.

O próximo capítulo traz as conclusões sobre este trabalho, as dificuldades envolvidas na pesquisa e no desenvolvimento da ferramenta e algumas sugestões de trabalhos futuros.

CONCLUSÕES E TRABALHOS FUTUROS

O objetivo deste trabalho foi propor e testar, através da avaliação de um protótipo, um editor gráfico de ontologias mais eficaz para o uso de especialistas de domínio da área de sustentabilidade, através do uso de um modelo de notações customizáveis (baseado em regras) para a representação visual de ontologias.

Para que fosse possível alcançar esse objetivo, foi (i) definida uma arquitetura de notações para representação de conceitos presentes na linguagem OWL, (ii) elaborado um modelo de representação junto aos especialistas de domínio, (iii) desenvolvido um protótipo do editor, o KGWE(*Knowledge Graph Web Editor*) e (iv) realizada uma série de avaliações para que fosse possível testar a adequação do modelo proposto. O modelo proposto mostrou-se capaz de contribuir para a visualização e edição de ontologias por especialistas de domínio.

Vários experimentos foram conduzidos com o auxílio de técnicas de engenharia de usabilidade. No total 4 avaliações diferentes foram realizadas na ferramenta desenvolvida.

As avaliações dos protótipos de vídeo e funcional ajudaram a desenvolver os requisitos iniciais do editor. Os estudos de caso Sustenagro e Nano mostraram que os especialistas estavam mais satisfeitos com a nova versão da ferramenta quando comparada ao editor que esses especialistas utilizavam anteriormente. Nos estudos de caso foi possível também avaliar o grau de eficiência e eficácia do editor. Os especialistas consideraram a ferramenta mais eficiente e também eficaz que as ferramentas que eles utilizavam anteriormente.

Para disponibilizar uma ferramenta com melhor usabilidade, foi realizada uma avaliação heurística que permitiu identificar diversos problemas, que muitas vezes não estavam aparentes que não foram identificados em testes funcionais durante o desenvolvimento do *software*. Com ajuda do *Think-Aloud*, várias sugestões de melhorias foram coletadas e desenvolvidas. Essas contribuições permitiram desenvolver uma ferramenta mais útil. Nesta etapa foram detectados também alguns erros que levaram a ajustes na ferramenta.

Por fim, a avaliação da eficiência de notações customizadas mostrou reais ganhos em cenários de visualização de ontologias. Essa avaliação mostrou que as notações customizáveis baseadas em regras tornaram as representações dos elementos mais eficientes.

Essas avaliações permitiram validar que os conceitos mais simplificados da interface, em combinação com o uso de notações, foram bem recebidos pelos especialistas de domínio e tornaram seu trabalho mais efetivo.

7.1 Contribuições científicas e tecnológicas

Este trabalho gerou diversas contribuições tanto de pesquisa como tecnológicas. Como contribuições científicas, o presente trabalho tem como resultados principais:

- Um modelo de notações customizáveis baseado em regras para visualização de ontologias;
- Um protótipo funcional que implementa esse modelo de notações. O editor KGWE - Knowledge Graph Web Editor foi a maior contribuição tecnológica desta pesquisa;
- Artigo (primeiro autor) "Um Modelo de Notações Customizáveis para Representação de Ontologias" publicado no 12º Seminário de Pesquisa em Ontologias no Brasil (Ontobras) no Workshop de Teses e Dissertações em Ontologias (WTDO).

Os anais da conferência foram publicados no periódico online CEUR-WS. ISSN: 1613-0073;

- Artigo (primeiro autor) "KGWE - A customizable editor for OWL ontologies" publicado no 26º Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia) no Workshop de Ferramentas e Aplicações (WFA).

Anais Estendidos do WebMedia'2020. ISSN: 2596-1683;

- Artigo (co-autor) "Desenvolvimento de ontologias e linguagem de domínio específico para análises de complexidade semântica em nanotecnologia agrícola" publicado no Workshop de Nanotecnologia Aplicada ao Agronegócio. ISSN: 2175-8395.

Este trabalho contribuiu também para fortalecer as relações de pesquisa entre a USP e a Embrapa/APTA. Os especialistas de domínio se mostraram bastante motivados com nossas contribuições. Um agradecimento especial à toda equipe de pesquisadores da Embrapa e da APTA.

7.1.1 Dificuldades na pesquisa

Com o objetivo de não tornar essa pesquisa excessivamente ampla, buscou-se realizar alguns cortes nos objetivos. Por exemplo, a validação da hipótese foi feita utilizando-se um

público mais específico, neste caso os especialistas de domínio em sustentabilidade. Como o grupo já tem uma colaboração com uma equipe da área de sustentabilidade da Embrapa, isso facilitou o trabalho.

Também pode ser relatado aqui a dificuldade na busca de trabalhos fortemente relacionados com o tema da pesquisa.

Não é uma tarefa fácil recrutar especialistas de qualquer domínio para experimentos. Eles são poucos em número e muito ocupados. Mesmo com o uso de especialistas da Embrapa e APTA, houve uma carência de especialistas disponíveis para experimentos, principalmente durante o início da pesquisa.

Por fim, a maior dificuldade foi a pandemia da COVID-19 iniciada meses antes do término dessa pesquisa. Ela mudou toda a dinâmica de trabalho na USP, na Embrapa, APTA e na empresa onde trabalho (este trabalho foi feito em tempo parcial). Todos os envolvidos tiveram que se adaptar. Os encontros semanais, que eram presenciais, foram substituídos por contatos realizados de forma totalmente *online*. A continuação das reuniões, realizadas com os especialistas, também só foi possível devido a alternativas remotas de comunicação. Tudo isso impactou e atrasou o trabalho em meses. Tínhamos planejado envolver mais especialistas, mas a maioria deles estavam, compreensivelmente, completamente ocupados em se adaptar às novas condições de trabalho.

7.1.2 Dificuldades no desenvolvimento

No início do projeto, identificou-se um impasse na definição da linguagem para construção da aplicação, pois tinha-se como premissa o uso de uma linguagem que fosse poderosa e ao mesmo tempo flexível. Inicialmente, uma versão limitada do editor foi construída utilizando puramente o JavaScript, mas ela não atendia aos requisitos básicos do projeto. Foram então iniciados testes com a linguagem RUST. Após algumas semanas, ficou nítido a falta de suporte para aplicações Web dessa linguagem. Estudos com Clojure também foram realizados, mas ela também foi descartada. Como havia um requisito para que a linguagem do *backend* fosse baseada na JVM, a linguagem Kotlin foi escolhida, já que pode ser compilada tanto para *bytecodes* Java quanto para JavaScript. Isso permitiu o uso de uma só linguagem no desenvolvimento.

Para validar a hipótese deste trabalho, foi necessária a construção do editor KGWE, porém, construir todas as funcionalidades necessárias para o uso do editor, exigiu também muito tempo. Foi necessário mais de um ano de desenvolvimento para chegar a uma versão completa o suficiente para testar as funcionalidades planejadas.

Utilizando-se o CLOC¹, um software que analisa e faz uma contabilização de código desenvolvido, foi possível verificar que, considerando-se somente os artefatos de classes em Kotlin, existiam um total de 200 arquivos com cerca de 19 mil linhas de codificação. O projeto

¹ <<https://github.com/AIDanial/cloc/>>

ainda usou outros artefatos de código, como arquivos HTML, CSS, SPARQL entre outros. Esses dados demonstram a extensão do projeto e a difícil tarefa de concluí-lo em um curto período de tempo.

7.2 Trabalhos futuros

Durante todo o desenvolvimento deste projeto, foram identificados novas funcionalidades e possíveis objetos de novas pesquisas na área de visualização de ontologias. Os principais resultaram nas seguintes sugestões de trabalhos futuros.

7.2.1 *Expansão da validação do uso de notações customizadas para outros domínios;*

Estudos podem ser realizados para averiguar o uso de notações customizadas para outros domínios de conhecimento e outros tipos de editores gráficos (além de editores de ontologias). Resultados dessas pesquisas poderão sugerir modificações relevantes para ferramentas de suas áreas.

7.2.2 *Uso de notações customizáveis em outros métodos de visualização não abordados neste trabalho;*

Existem diversos métodos de visualização não explorados neste trabalho. O estudo da eficácia desses métodos é pertinente para a customização da exibição de elementos presentes em uma ontologia.

7.2.3 *Expansão das regras;*

O editor KGWE poderia disponibilizar um conjunto maior de tipos de regras. Elas poderiam usar sentenças customizadas envolvendo propriedades, classes e instâncias de ontologias, e modificar características visuais, tais como cor, forma e dimensão dos elementos. Neste trabalho, utilizou-se o modo de igualdade para comparar valores literais, porém, seriam bastante úteis outros mecanismos de *matching*, como, por exemplo, se uma determinada classe contém parte de um texto informado.

7.2.4 *Tornar a ferramenta disponível para desenvolvimento colaborativo de ontologias;*

Atualmente, o editor KGWE não é capaz de trabalhar com toda a expressividade da linguagem OWL. Ele pode ser expandido para incluir a mesma capacidade que a ferramenta Protégé, e pode ser utilizada para editar um grande número de ontologias em OWL. Isso

proporcionaria uma validação ampla da ferramenta com grande potencial de obtenção de *feedback*. Também contribuiria para o fortalecimento de diversas comunidades e seus interessados, visto que há uma evidente carência de ferramentas desse tipo.

7.2.5 *Integrar a ferramenta em bases de conhecimento existentes;*

Uma vez que o KGWE pudesse editar um grande número de ontologias em OWL, seria importante integrá-lo a bases de conhecimentos públicas, como o BioPortal. Isto tornaria a ferramenta ainda mais relevante e funcional.

REFERÊNCIAS

ABNT, A.; DEFESA, E.; GLOBAL, N.; DEFESA, E.; ZUCCARO, P.; DEFESA, E. Requisitos ergonômicos para trabalho de escritórios com computadores parte 11—orientações sobre usabilidade. **AAB d. N. Técnicas (Ed.). Brasil: ABNT—Associação Brasileira de Normas Técnicas**, 2002. Citado nas páginas 56 e 57.

ABT, V.; VIGIER, F.; SCHNEIDER, M. Enterprise business modelling languages applied to farm enterprise: A case study for ideo, gaudi, and ams languages. In: **Advances in modeling agricultural systems**. [S.l.]: Springer, 2009. p. 167–191. Citado na página 36.

BECHHOFFER, S.; HARMELEN, F. V.; HENDLER, J.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A. *et al.* Owl web ontology language reference. **W3C recommendation**, v. 10, n. 02, 2004. Citado na página 41.

BECHHOFFER, S.; HARMELEN, F. van; HENDLER, J.; HORROCKS, I.; MCGUINNESS, D.; PATEL-SCHNEIDER, P.; STEIN, L. A. **OWL Web Ontology Language Reference**. [S.l.], 2004. See <<http://www.w3.org/TR/owl-ref/>>. Citado nas páginas 38 e 39.

BRICKLEY, D.; GUHA, R. V.; MCBRIDE, B. Rdf schema 1.1. **W3C recommendation**, v. 25, p. 2004–2014, 2014. Citado na página 38.

BUENO, C. d. C.; JESUS, K. R. E. de. Emprego e adaptação do método gmp-ram para avaliação dos riscos das nanotecnologias. In: IN: WORKSHOP DA REDE DE NANOTECNOLOGIA APLICADA AO AGRONEGÓCIO, 6., 2012, FORTALEZA. ANAIS... SÃO CARLOS: EMBRAPA INSTRUMENTAÇÃO; FORTALEZA: EMBRAPA AGROINDÚSTRIA TROPICAL, 2012. **Embrapa Meio Ambiente-Artigo em anais de congresso (ALICE)**. [S.l.], 2011. Citado na página 43.

CARD, S. K.; MACKINLAY, J. D.; SHNEIDERMAN, B. (Ed.). **Readings in Information Visualization: Using Vision to Think**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. ISBN 1-55860-533-9. Citado na página 48.

CHARTERS, E. The use of think-aloud methods in qualitative research an introduction to think-aloud methods. **Brock Education: A Journal of Educational Research and Practice**, v. 12, n. 2, 2003. Citado na página 102.

CLARA GIACALONE. **IMDb Interaction between Genres and Ratings Datasets**. 2015. Disponível em: <<https://claragiacaloneportfolio.wordpress.com/2015/01/09/imdb-website/>>. Acesso em: 22 dezembro. 2019. Citado nas páginas 53, 54 e 55.

CONSORTIUM, G. O. The gene ontology in 2010: extensions and refinements. **Nucleic acids research**, Oxford University Press, v. 38, n. suppl_1, p. D331–D335, 2009. Citado nas páginas 49 e 51.

COSENTINO, M.; GAUD, N.; HILAIRE, V.; GALLAND, S.; KOUKAM, A. Aspects: an agent-oriented software process for engineering complex systems. **Autonomous Agents and Multi-Agent Systems**, Springer, v. 20, n. 2, p. 260–304, 2010. Citado nas páginas 49 e 50.

- DAVEY, B. Think aloud: Modeling the cognitive processes of reading comprehension. **Journal of Reading**, JSTOR, v. 27, n. 1, p. 44–47, 1983. Citado na página 102.
- DUDÁŠ, M.; LOHMANN, S.; SVÁTEK, V.; PAVLOV, D. Ontology visualization methods and tools: a survey of the state of the art. **The Knowledge Engineering Review**, Cambridge University Press, v. 33, 2018. Citado nas páginas 48 e 51.
- FARINA, M.; PEREZ, C.; BASTOS, D. **Psicodinâmica das cores em comunicação**. [S.l.]: Editora Blucher, 2011. Citado na página 54.
- FREITAS, C. M. D. S.; CHUBACHI, O. M.; LUZZARDI, P. R. G.; CAVA, R. A. Introdução à visualização de informações. **Revista de informática teórica e aplicada. Porto Alegre. Vol. 8, n. 2 (out. 2001), p. 143-158**, 2001. Citado nas páginas 28 e 47.
- FU, B.; NOY, N. F.; STOREY, M.-A. Eye tracking the user experience—an evaluation of ontology visualization techniques. **Semantic Web**, IOS Press, v. 8, n. 1, p. 23–41, 2017. Citado na página 28.
- GARCÍA-BARRIOCANAL, E.; SICILIA, M. A.; SÁNCHEZ-ALONSO, S. Usability evaluation of ontology editors. **KO KNOWLEDGE ORGANIZATION**, Nomos Verlagsgesellschaft mbH & Co. KG, v. 32, n. 1, p. 1–9, 2005. Citado na página 56.
- GENOVA, C. **Origami: dobras, contas e encantos**. [S.l.]: Escrituras Editora e Distribuidora de Livros Ltda., 2014. Citado na página 53.
- GHORBEL, F.; ELLOUZE, N.; MÉTAIS, E.; HAMDI, F.; GARGOURI, F.; HERRADI, N. Memo graph: An ontology visualization tool for everyone. **Procedia Computer Science**, Elsevier, v. 96, p. 265–274, 2016. Citado na página 65.
- GRUBER, T. R. A translation approach to portable ontology specifications. **Knowl. Acquis.**, Academic Press Ltd., London, UK, UK, v. 5, n. 2, p. 199–220, jun. 1993. ISSN 1042-8143. Disponível em: <<http://dx.doi.org/10.1006/knac.1993.1008>>. Citado nas páginas 27 e 35.
- GUARINO, N. Understanding, building and using ontologies. **Int. J. Hum.-Comput. Stud.**, Academic Press, Inc., Duluth, MN, USA, v. 46, n. 2-3, p. 293–310, mar. 1997. ISSN 1071-5819. Disponível em: <<http://dx.doi.org/10.1006/ijhc.1996.0091>>. Citado nas páginas 35 e 36.
- _____. **Formal Ontology in Information Systems: Proceedings of the 1st International Conference June 6-8, 1998, Trento, Italy**. 1st. ed. Amsterdam, The Netherlands, The Netherlands: IOS Press, 1998. ISBN 9051993994. Citado na página 35.
- HEINZLE, R.; GAUTHIER, F. A. O.; FIALHO, F. A. P. Semântica nos sistemas de apoio a decisão: o estado da arte. **Revista da UNIFEDE**, v. 1, n. 8, p. 225–248, 2017. Citado na página 42.
- HERICKO, M.; JURIC, M. B.; ROZMAN, I.; BELOGLAVEC, S.; ZIVKOVIC, A. Object serialization analysis and comparison in java and .net. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 38, n. 8, p. 44–54, ago. 2003. ISSN 0362-1340. Disponível em: <<http://doi.acm.org/10.1145/944579.944589>>. Citado na página 37.
- HITZLER, P.; KRÖTZSCH, M.; PARSIA, B.; PATEL-SCHNEIDER, P. F.; RUDOLPH, S. *et al.* Owl 2 web ontology language primer. **W3C recommendation**, World Wide Web Consortium (W3C), v. 27, n. 1, p. 123, 2009. Citado nas páginas 39, 40 e 41.

JESUS, K. R. E. de; LANNA, A. C.; VIEIRA, F. D.; ABREU, A. L. de; LIMA, D. U. de. A proposed risk assessment method for genetically modified plants. **Applied Biosafety**, SAGE Publications Sage CA: Los Angeles, CA, v. 11, n. 3, p. 127–137, 2006. Citado na página 43.

KATIFORI, A.; HALATSIS, C.; LEPOURAS, G.; VASSILAKIS, C.; GIANNOPOULOU, E. Ontology visualization methods—a survey. **ACM Comput. Surv.**, ACM, New York, NY, USA, v. 39, n. 4, nov. 2007. ISSN 0360-0300. Disponível em: <<http://doi.acm.org/10.1145/1287620.1287621>>. Citado nas páginas 48, 49 e 52.

KHAMPARIA, A.; PANDEY, B.; PARDESI, V. Performance analysis on agriculture ontology using sparql query system. In: IEEE. **Data Mining and Intelligent Computing (ICDMIC), 2014 International Conference on**. [S.l.], 2014. p. 1–5. Citado na página 36.

KUMAZAWA, T.; SAITO, O.; KOZAKI, K.; MATSUI, T.; MIZOGUCHI, R. Toward knowledge structuring of sustainability science based on ontology engineering. **Sustainability Science**, Springer, v. 4, n. 1, p. 99, 2009. Citado na página 36.

LAUFER, C. Guia de web semantica. **Sao Paulo**, 2015. Citado na página 37.

LIEPINŠ, R.; GRASMANIS, M.; BOJARS, U. Owlged ontology visualizer. In: **Proceedings of the 2014 International Conference on Developers**. [S.l.: s.n.], 2014. v. 1268, p. 37–42. Citado nas páginas 63 e 64.

LOHMANN, S.; LINK, V.; MARBACH, E.; NEGRU, S. **WebVOWL**. 2015. Disponível em: <<http://vowl.visualdataweb.org/webvowl.html>>. Citado na página 62.

_____. WebVOWL: Web-based visualization of ontologies. In: **Proceedings of EKAW 2014 Satellite Events**. [S.l.]: Springer, 2015. (LNAI, v. 8982), p. 154–158. Citado na página 61.

LOHMANN, S.; NEGRU, S.; HAAG, F.; ERTL, T. Vowl 2: User-oriented visualization of ontologies. In: SPRINGER. **International Conference on Knowledge Engineering and Knowledge Management**. [S.l.], 2014. p. 266–281. Citado nas páginas 61, 62 e 63.

MACDONALD, L. W. Using color effectively in computer graphics. **IEEE Computer Graphics and Applications**, IEEE, v. 19, n. 4, p. 20–35, 1999. Citado nas páginas 53 e 56.

MORAIS, E. A. M.; AMBRÓSIO, A. P. L. Ontologias: conceitos, usos, tipos, metodologias, ferramentas e linguagens. **Relatório Técnico—RT-INF-001/07, dez**, 2007. Citado na página 36.

NETTO, C. M.; LIMA, G. Â. Visualização de ontologias: estudos e perspectivas. **Informação & Sociedade**, Universidade Federal da Paraíba-Programa de Pós-Graduação em Ciência da Informação, v. 27, n. 3, 2017. Citado nas páginas 28, 29 e 56.

NETTO, C. M.; LIMA, G. A. B. d. O.; JUNIOR, I. P. Modelos de visualização de informação para ontologias de domínio. 2017. Citado na página 42.

NIELSEN, J. **Usability engineering**. [S.l.]: Morgan Kaufmann, 1994. Citado nas páginas 98, 99 e 100.

_____. 10 usability heuristics for user interface design. **Nielsen Norman Group**, v. 1, n. 1, 1995. Citado na página 98.

OKAMOTO, A. K.; FURQUIM, F. V. **Eye tracking: o que é, como funciona e suas aplicações**. 2020. Disponível em: <<http://www.each.usp.br/petsi/jornal/?p=2774>>. Acesso em: 08 maio. 2021. Citado na página 105.

- PSYLLIDIS, A. Osmosys: a web interface for graph-based rdf data visualization and ontology browsing. In: SPRINGER. **International Conference on Web Engineering**. [S.l.], 2015. p. 679–682. Citado nas páginas 63 e 64.
- ROUSSEY, C.; SOULIGNAC, V.; CHAMPOMIER, J.; ABT, V.; CHANET, J. Ontologies in Agriculture. In: **AgEng 2010, International Conference on Agricultural Engineering**. Clermont-Ferrand, France: Cemagref, 2010. p. p. – p. Disponível em: <<https://hal.archives-ouvertes.fr/hal-00523508>>. Citado na página 36.
- SIMMON, R. Use of color in data visualization. **Nasa**, https://earthobservatory.nasa.gov/resources/blogs/intro_to_color_for_visualization.pdf, v. 18, n. 5, p. 2017, 2013. Citado na página 80.
- SUAREZ, J. F. G. **Ontologias e DSLs na geração de sistemas de apoio à decisão, caso de estudo SustenAgro**. Dissertação (Mestrado) — Universidade de São Paulo, 2017. Citado nas páginas 29, 36, 42, 43 e 44.
- TUDORACHE, T.; NYULAS, C.; NOY, N. F.; MUSEN, M. A. **WebProtege**. 2013. Disponível em: <<https://protegewiki.stanford.edu/wiki/WebProtege>>. Citado na página 60.
- _____. Webprotégé: A collaborative ontology editor and knowledge acquisition tool for the web. **Semant. web**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 4, n. 1, p. 89–99, jan. 2013. ISSN 1570-0844. Disponível em: <<http://dl.acm.org/citation.cfm?id=2595053.2595061>>. Citado na página 60.
- UKPE, E. Agriculture ontology for sustainable development in nigeria. **IOsR Journal of Computer Engineering (IOsR-JCE)**, v. 14, n. 5, p. 57–59, 2013. Citado na página 36.
- VÖLKER, J. **Learning expressive ontologies**. [S.l.]: IOS Press, 2009. v. 2. Citado na página 38.
- W3C. **OWL 2 Web Ontology Language Profiles (Second Edition)**. 2012. [Http://www.w3.org/TR/owl2-profiles/](http://www.w3.org/TR/owl2-profiles/). Disponível em: <<http://www.w3.org/TR/owl2-profiles/>>. Citado na página 39.
- WARE, C. **Information visualization: perception for design**. [S.l.]: Elsevier, 2012. Citado na página 47.

HISTÓRIAS DE USUÁRIO

A.1 Histórias de usuário

Este anexo trás as principais histórias de usuário construídas durante o desenvolvimento da ferramenta:

0. **QUEM** um especialista de domínio,
DESEJO acessar a URL do editor apontando para a ontologia de interesse,
PARA visualizar e editar a ontologia,
ENTÃO a aplicação exibe as abas que permitirão a edição pelos nós em duas dimensões,
OU pela árvore hierárquica,
OU pela aba de visualização em três dimensões,
OU ainda por meio do editor JSON-LD.
1. **QUEM** um especialista de domínio,
DESEJO customizar a cor de uma classe genérica ou específica através de regras,
PARA que se possa obter uma representação mais fiel da ontologia para seu domínio,
ENTÃO a aplicação deve permitir editar a cor de fundo de uma classe,
OU editar a cor de uma relação entre as classes,
OU editar a cor do rótulo,
OU editar a cor da borda de uma classe.
2. **QUEM** um especialista de domínio,
DESEJO customizar o tamanho de uma classe genérica ou específica através de regras,

- PARA** que se possa obter uma melhor representação de tamanhos dos nós,
ENTÃO a aplicação deve permitir editar o raio de uma classe para círculos,
OU editar o tamanho da borda de uma classe,
OU editar a largura e altura para retângulos e imagens,
OU editar a largura dos links entre as classes.
3. **QUEM** um especialista de domínio,
DESEJO customizar a representação de uma classe com imagens personalizadas,
PARA que imagens diversas possam ser utilizadas na representação dos nós da ontologia,
ENTÃO a aplicação deve permitir editar um nó genérico ou específico, informando a URL da imagem;
4. **DADO** que se realizou a criação de uma regra customizada,
E a aplicação solicita o tipo do elemento,
QUANDO verifico as opções da lista do menu de seleção,
ENTÃO a ferramenta exibe as seguintes opções: “Classe”, “Subclasse de”, “Número de indivíduos”,
PARA que a regra criada seja aplicada a uma classe específica,
OU para todos as subclasses do valor informado (nome da classe),
OU para todos os nós cujo número de instâncias seja maior ou igual a um valor informado.
5. **DADO** que se realizou a edição de uma ontologia na aba do editor gráfico,
E se clicou em uma classe,
QUANDO se verificam as informações desta classe no painel lateral direito,
ENTÃO a ferramenta deve exibir informações como: IRI, nó pai, descrição da classe e instâncias.
6. **DADO** que se realizou a edição de uma ontologia na aba do editor gráfico,
E se clicou com o botão direito em uma classe,
QUANDO se verificavam as opções da lista do menu de opções (menu de contexto),
ENTÃO a ferramenta deve exibir as seguintes opções: criar subclasses, criar indivíduo, criar rótulo, remover nó e alterar relações de vizinhança para exibição.
7. **DADO** que se realiza a edição de uma ontologia na aba do editor gráfico,
E se clica com o botão direito em uma classe,
QUANDO se realiza alguma alteração das informações da classe,
ENTÃO essas alterações irão refletir imediatamente no grafo.

CONFIGURAÇÕES DE IMPLANTAÇÃO

Código-fonte 2 – Comandos para implantação do editor em servidores Linux

```
1:
2: [Container 1 - OntologyEditor Backend]
3: ::: warning sudo apt-get update sudo apt-get install default-
   jre sudo apt-get install default-jdk
4:
5: sudo apt-get install git
6:
7: git clone https://github.com/DilvanLab/OntologyEditor.git Usúá
   rio: Senha:
8:
9: cd OntologyEditor/api/ ./gradle build
10:
11: cd OntologyEditor/api/build/libs --./root/OntologyEditor/api/
   build/libs/ontology-editor-api-0.0.1-SNAPSHOT.jar
12:
13: java -jar -Dspring.profiles.active=prod ontology-editor-api
   -0.0.1-SNAPSHOT.jar :::
14:
15: [Container 2 - OntologyEditor GraphDB]
16: ::: warning sudo apt-get update sudo apt-get install default-
   jre sudo apt-get install default-jdk
17:
18: --Download as a stand-alone server wget "http://go.pardot.com/e
   /45622/38-graphdb-free-8-8-1-dist-zip/6t61z2/1428512475?h=
   CtR2lSzWFK9eYPCCnGbOngvm4c2ccrITI5qvrgGPAAI"
19:
```

```
20: apt install unzip unzip 1428512475?h=
    Ctr2lSzWFK9eYPCCnGb0ngvm4c2ccrITI5qvrgGPAAI.2
    --1428512475....é o nome do arquivo do graphdb no qual foi
    realizado o download cd graphdb-free-8.8.1/ cd bin chmod +x
    ./graphdb ./graphdb -d :::

21:

22: [Container 3 - OntologyEditor Frontend]

23: ::: warning sudo apt-get update sudo apt-get install build-
    essential libssl-dev sudo apt-get install nodejs sudo apt-
    get install npm sudo apt-get install git sudo apt-get
    install default-jre sudo apt-get install default-jdk

24:

25: git clone https://github.com/DilvanLab/OntologyEditor.git Usua-
    rio: Senha:

26:

27: cd OntologyEditor/ npm install curl -sL https://deb.nodesource.
    com/setup_10.x | sudo bash - sudo apt install nodejs npm
    cache clean --force rm -rf node_modules/.cache

28:

29: export JAVA_OPTS="-Xmx4096m" npm start

30:

31: Note: If you prefer, an HTTP server can be installed and using
    the npm run build command to generate a production version
    of the code and deploy. :::
```