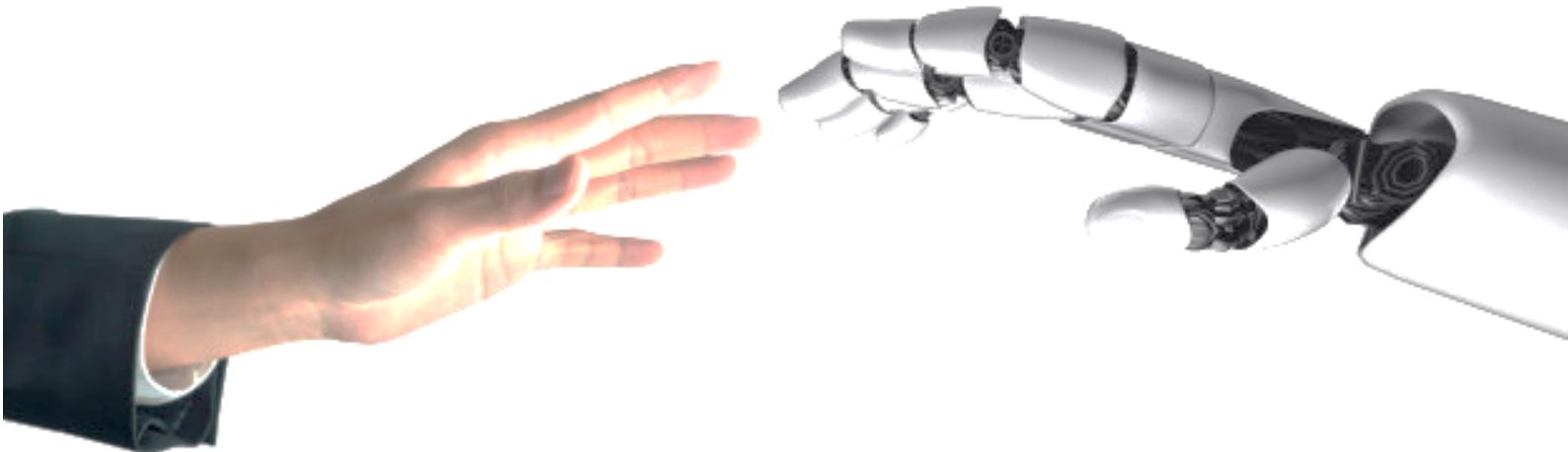




Inteligência Artificial - 2020/1



TRABALHO 1

“Para garantir que uma ontologia seja construída com qualidade é necessário definir o domínio de conhecimento com objetividade, descrevendo o conhecimento essencial ao domínio e definindo um vocabulário que evite interpretações ambíguas”

(GRUBER, 1993).

Agnes Almeida Preuss
Carolina Carvalhosa Simões
Rafael dos Santos de Oliveira Lima
Thalles Cotta Fontainha

Parte 1: Encontrar e descrever um tutorial para a criação de ontologias no Protegé

Tutorial usado pelo grupo:

<https://www.youtube.com/watch?v=zWlgROKbrJk>

Descrição do Tutorial:

O objetivo do tutorial é a criação de uma ontologia sobre a alimentação dos animais carnívoros e herbívoros. O instrutor inicia o vídeo criando um novo projeto, indicando em sua máquina a pasta que o arquivo será salvo e seu formato. Em seguida ele cria as classes, as subclasses e instâncias na aba *Entities*.

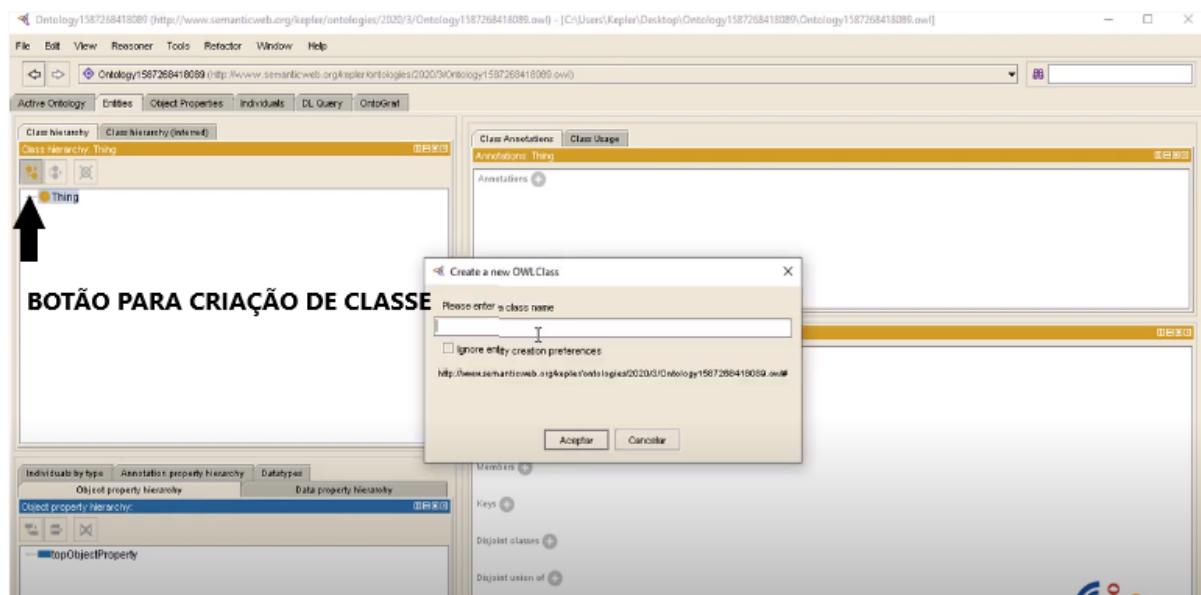


Figura 1: A imagem mostra o ícone para criar classes, subclasses e instâncias.

As classes criadas foram “Animais” e “Plantas”, sendo que animais possuem as subclasses “Herbívoros” e “Carnívoros”. Dentro de “Carnívoros” existem as instâncias “Tigre” e “Leão” e em “Herbívoros” existem “Girafa” e “Cavalo”. Além disso, foi feito o relacionamento das classes criadas, clicando na classe e em *Disjoint Classes*, onde se definiu que animais não podem ser plantas e vice versa.

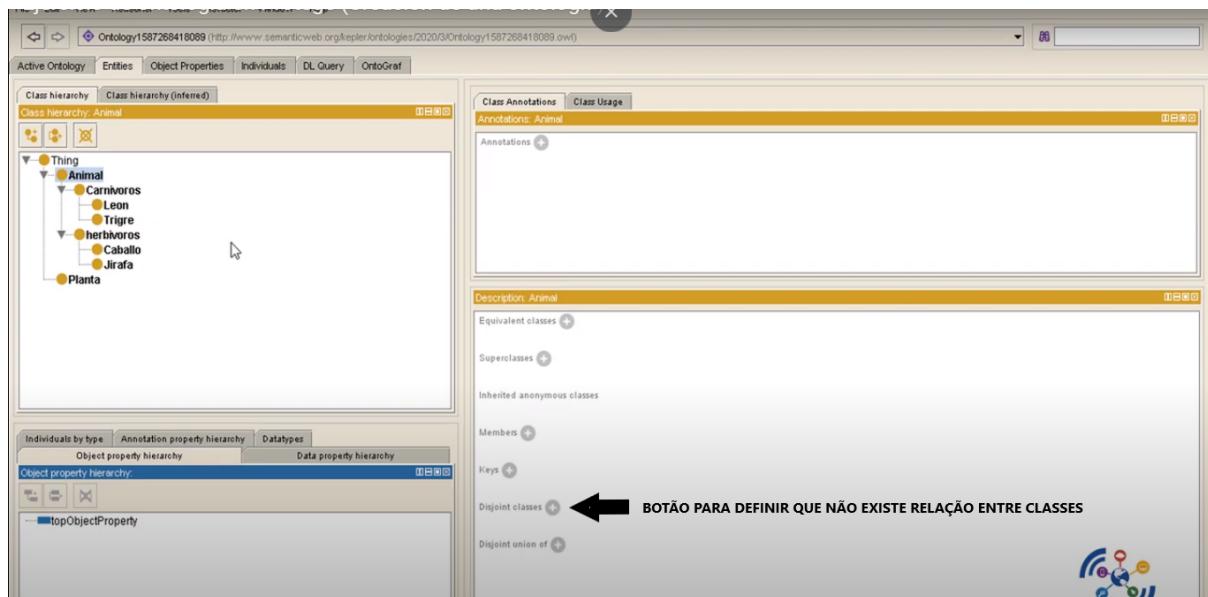


Figura 2: A Imagem mostra a hierarquia das classes e onde se faz o relacionamento das mesmas.

Após isso, foram criadas as relações “Come” e “É comido por”, na aba *Object Properties*. A primeira relação é associada às subclasses “Carnívoros” e “Herbívoros”, clicando em *Domains*, e ao que eles comem, “Animais” e “Plantas” respectivamente, clicando em *Range*. Depois é definido a relação inversa “É comido por”, marcando *Symmetric* e clicando em *Inverse*, dessa forma definimos “quem come o que” e “por quem é comido”. A segunda relação segue a mesma ideia, mas com o foco em “quem é comido”.

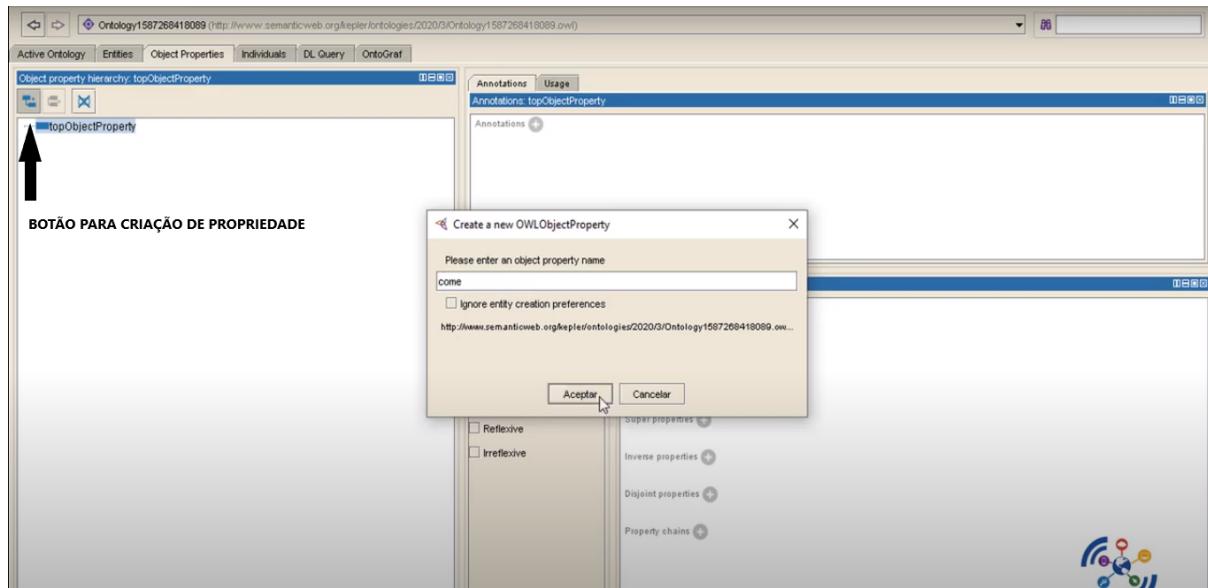


Figura 3: A imagem ilustra como é criado as relações.

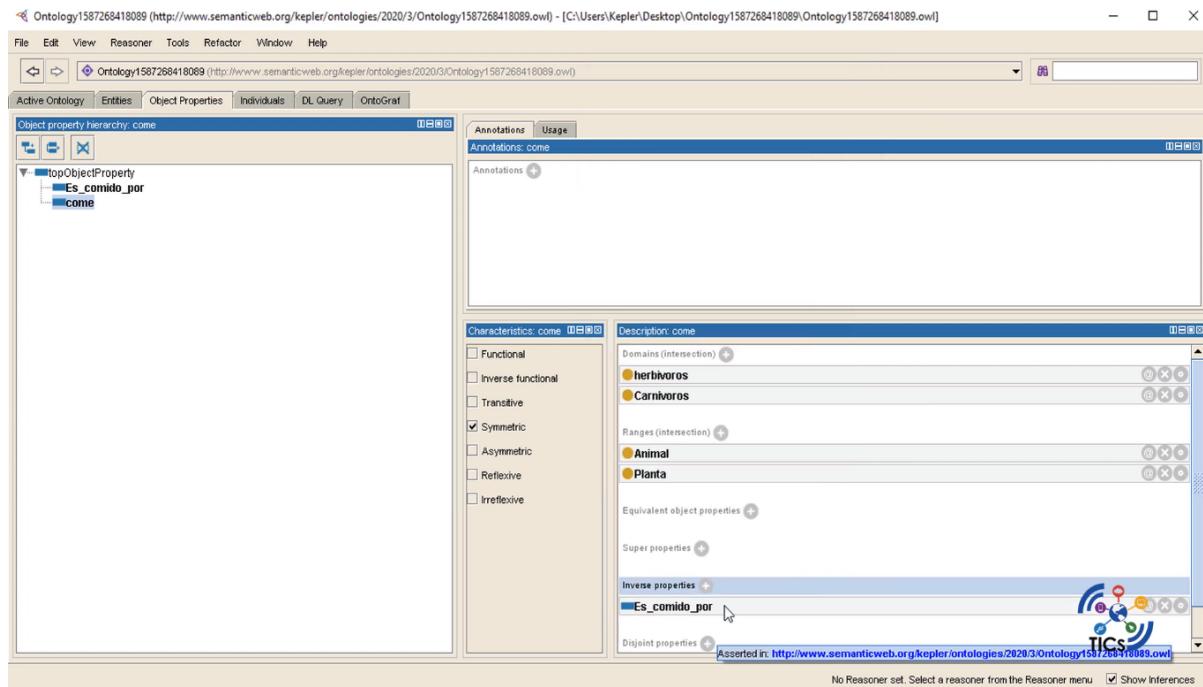


Figura 4: A imagem mostra as relação “Come”.

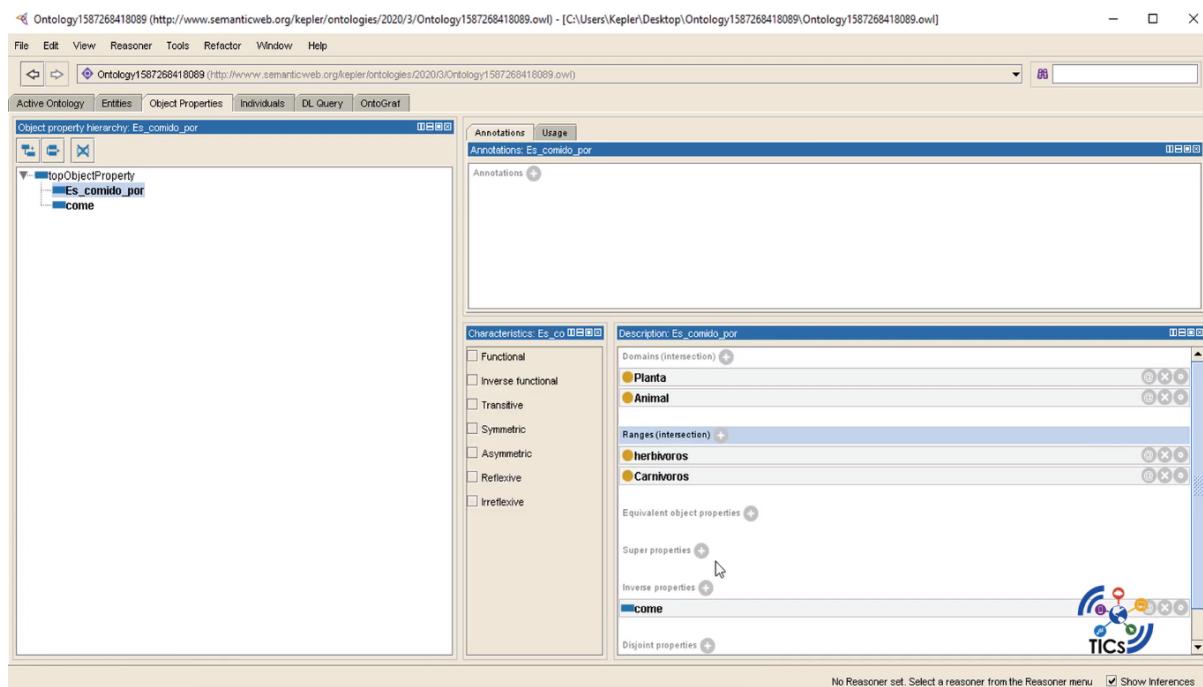


Figura 5: A imagem mostra a relação “É comido por”.

Por último o instrutor do vídeo que extraímos o nosso tutorial faz a restrições das classes para que elas coincidam com as relações. Para isso ele retorna a aba *Entites*, acessa *Classe Usage*, onde pode-se ver todas as relações, clica na classe “Animal”, depois em *Equivalent Classes* e *Object Restriction Creator*, e coloca que qualquer “Animal” pode ser comido pela subclasse “Carnívoro” e marca como restrição a opção *only*. O mesmo processo é feito para: Carnívoros somente comem animais; herbívoros somente comem planta; planta é comida por herbívoros.

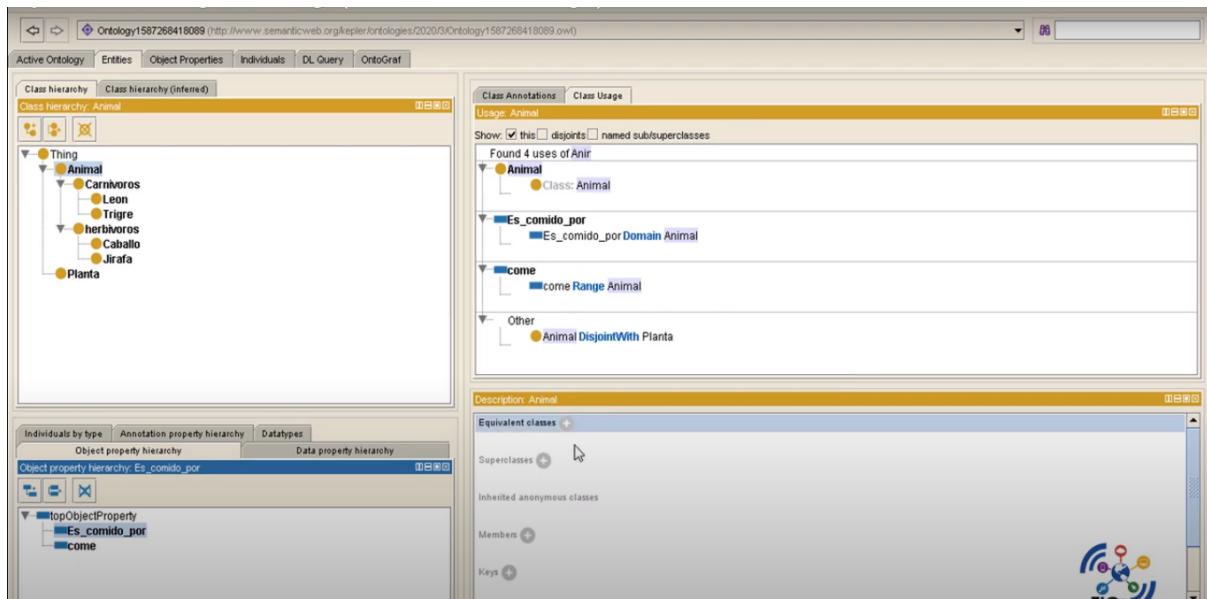


Figura 6: A imagem mostra a aba Classe Usage, onde se vê todas as relações.

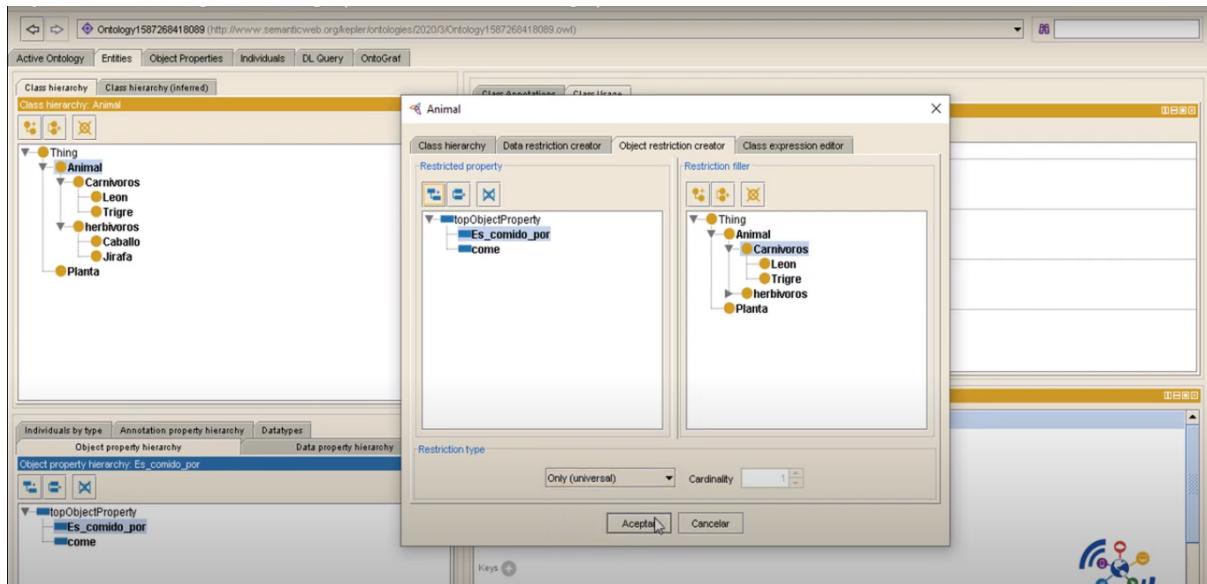


Figura 7: A imagem mostra a aba Object Restriction Creator.

Por fim, o tutor visualiza na aba *Ontograf*, selecionando cada classe, subclasse e instância, todo o domínio criado.

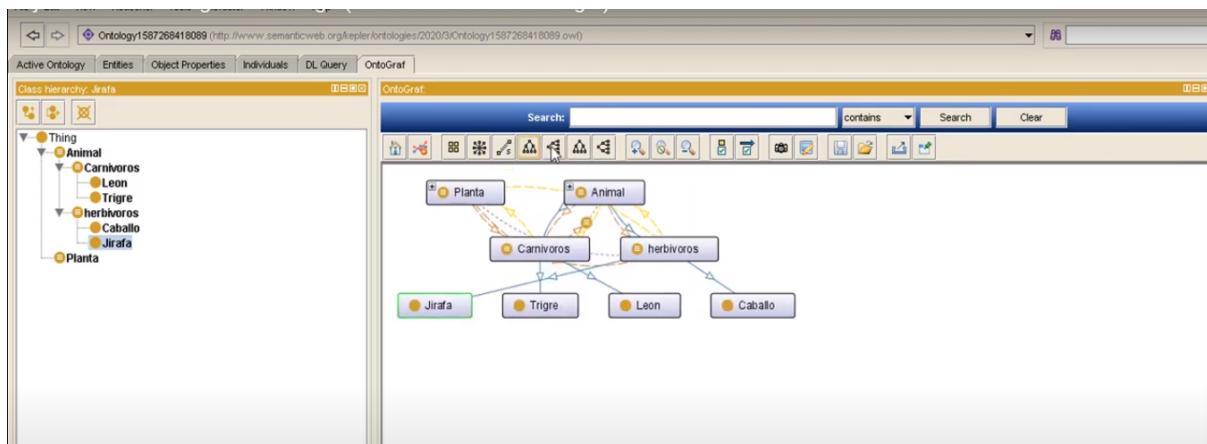


Figura 8: A imagem mostra a ontologia completa.

Parte 2: Criar uma ontologia no Protegé

O tutorial citado na parte um usa a ferramenta *Protegé* instalável, no entanto, o grupo optou por usar <https://webprotege.stanford.edu/>, por se tratar de uma ferramenta de ontologia online, gratuito e possibilitar que todos os membros do grupo participem do projeto ao mesmo tempo.

A Motivação para criação da ontologia:

Devido ao período atípico que vivemos, fomos forçados a ficar em casa e consequentemente a buscar formas de nos entreter. Conversando em grupo reparamos que a maioria dos integrantes acabou voltando a jogar *Pokémon*, criado por Satoshi Tajiri, desenvolvido pela Game Freak e publicado pela Nintendo no Japão.

A história é centrada em monstros ficcionais chamados "*Pokémons*", que são capturados na natureza por jogadores e são treinados para lutarem entre si, como se fosse um esporte. O jogo ficou tão popular que acabou atravessando oceanos e ganhando várias vertentes, como animação e quadrinhos, transformando-a em uma franquia altamente lucrativa no mercado. Somente o game para celular, *Pokémon Go*, gerou US\$ 894 milhões em receitas em seu lançamento em 2016.

Devido ao exposto e uma certa nostalgia, optamos por criar uma ontologia envolvendo o jogo. O projeto visa um usuário que queira montar um time de *pokémons*, sendo assim, ele precisa saber qual é o melhor tipo para cada combate.

Apresentar o Modelo do Domínio, contendo as relações entre os itens:

Para o melhor entendimento do trabalho, achamos pertinente fazer uma breve explicação de como é o “mundo pokémon”. Cada monstro possui um ou mais tipos (água, fogo, planta, etc), ataques, fraquezas e vantagens. Quando ele ataca o resultado pode ser *No Effect*, *Super Effective* e *Not Very Effective*.

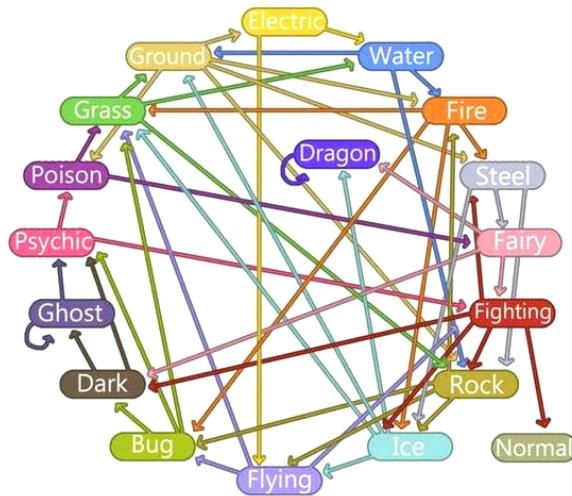


Figura 9: A imagem mostra todas as relações *Super Effective* conforme o tipo de *Pokémon*.

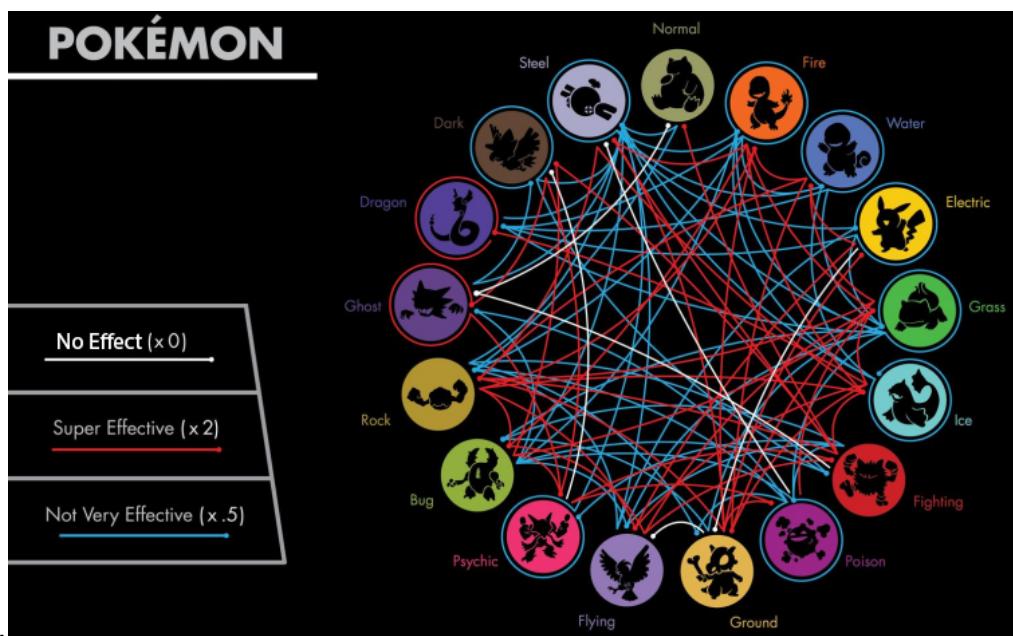


Figura 10: A imagem mostra todas as relações de *No Effect*, *Super Effective*, e *Not Very Effective* de acordo com o tipo de *Pokémon*.

Defender	Normal	Fire	Water	Grass	Electric	Ice	Fighting	Poison	Ground	Flying	Psychic	Bug	Rock	Ghost	Dragon	Dark	Steel	Fairy
Attacker	Normal																	
Normal																		
Fire	½	½	2	2												½	2	
Water	2	½	½						2	2	2	2						
Grass	½	2	½						½	2	½	½	2					
Electric	2	½	½	½					0	2								
Ice	½	½	2	½					2	2						2	½	
Fighting	2				2				½	½	½	½	2	0		2	2	½
Poison				2					½	½					½	0	2	
Ground	2		½	2				2	0			½	2			2		
Flying			2	½			2			2		2	½					
Psychic				2	2					½					0	½		
Bug	½		2			½	½	2	½	2					½	2	½	½
Rock	2			2	½	½	½	2	2	2						½		
Ghost	0									2			2		2	½		
Dragon													2		½	0		
Dark									½			2		2	½	½		
Steel		½	½	½	2							2			½	½	2	
Fairy	½				2	½						2	2	½				

Figura 11: A imagem mostra todas as relações de *No Effect*(0), *Super Effective*(2), e *Not Very Effective* (½) de acordo com o tipo do *Pokémon* de forma numérica.

Para simplificar e não deixar o trabalho tão poluído quanto a figura 10, optamos por não cadastrar todos os tipos de *pokémons*, somente 9 (classe “TYPE”). Além disso, cada monstro só pode ter um tipo, ataques deste mesmo tipo e mantemos as relações de “NO_EFFECT”, “SUPER_EFFECTIVE” e “NOT_VERY_EFFECTIVE” e adicionando outras relações: “EVOLUTION” e “IS_TYPE”.

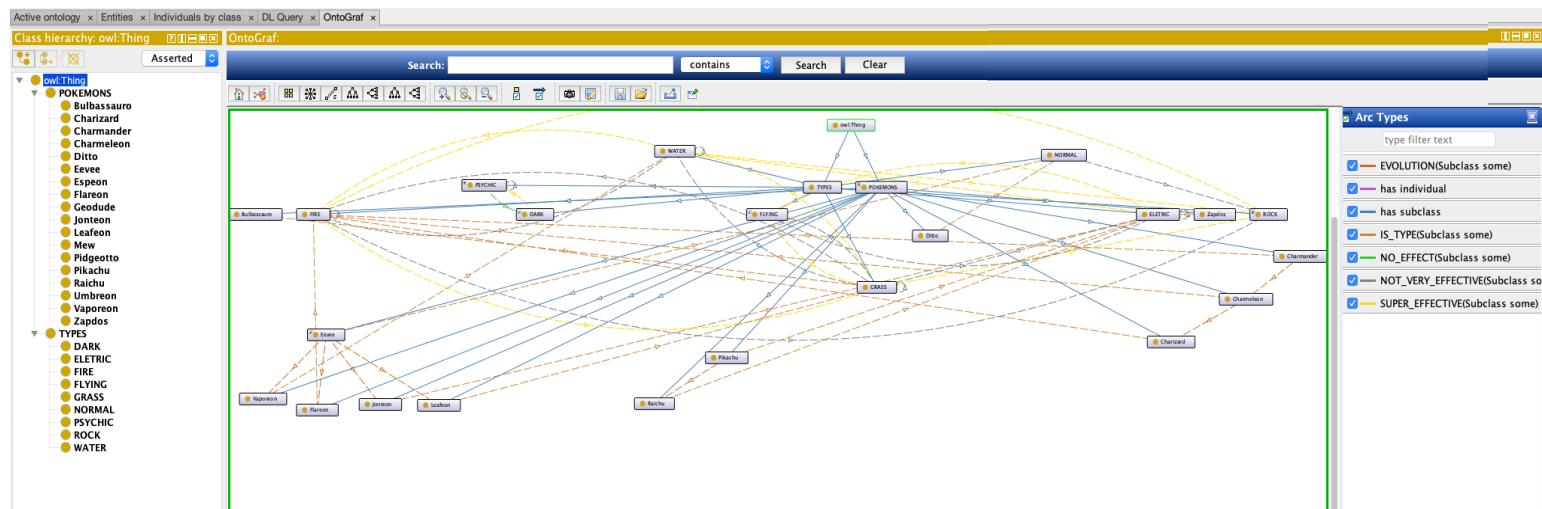


Figura 12: A imagem ilustra o domínio da ontologia construída.

Para uma melhor visualização, pegamos alguns exemplos que mostram classes, subclasses e relações:

- Subclasses de “POKEMONS”

○ Charmander:

Este *pokémon* é do tipo fogo e possui evolução. Portanto, na Figura 13.1, podemos ver as evoluções do mesmo e descobrir quais são os tipos mais fortes e mais fracos contra ele.

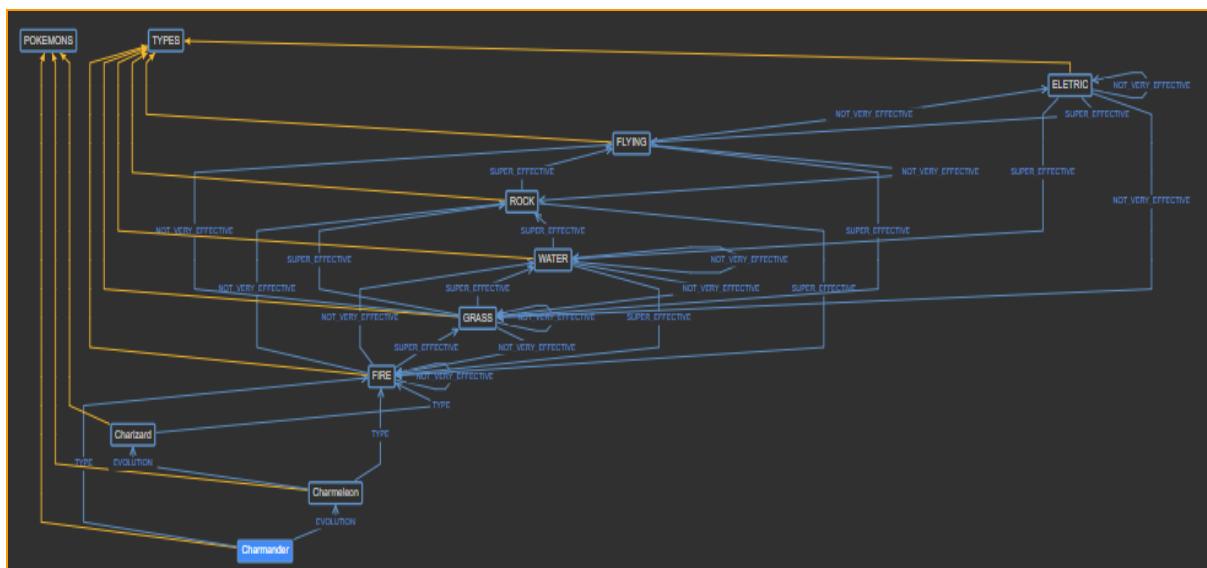


Figura 13.1: A imagem mostra o gráfico de entidade da subclasse “Charmander”.

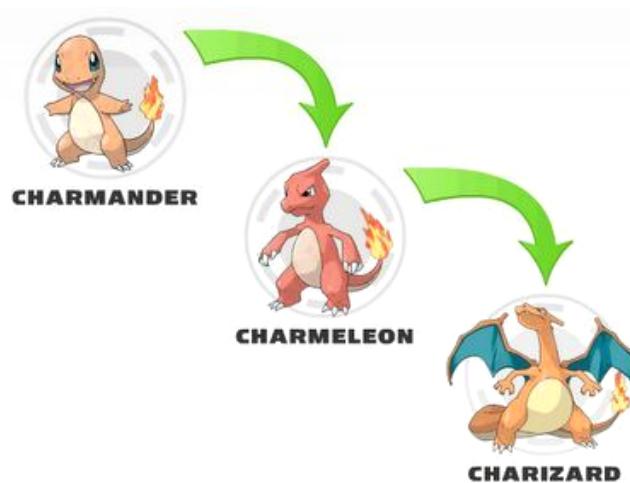


Figura 13.2: A imagem mostra a linha evolutiva do Charmander.

- Eevee:

Este *pokémon* é conhecido por ter uma composição genética instável, que sofre mutações repentinas devido ao seu ambiente. Sendo assim, a radiação de várias pedras diferentes fazem com que ele evolua para tipos diferentes.

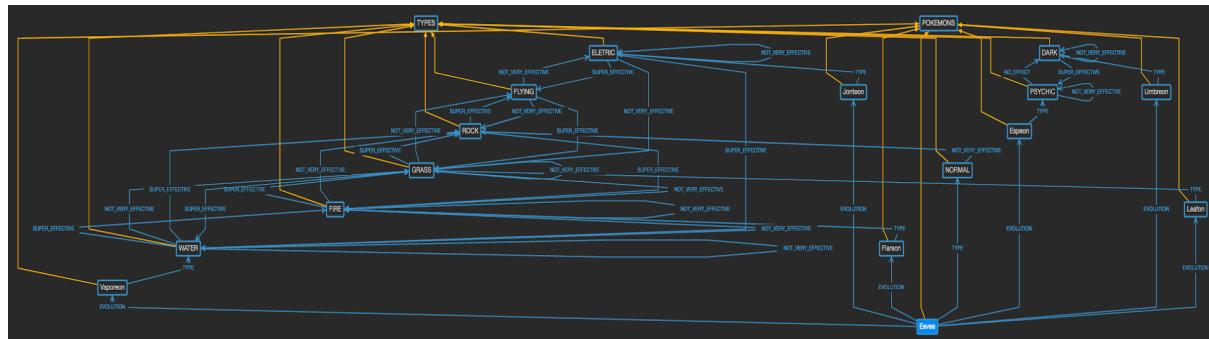


Figura 14.1: A imagem mostra o gráfico de entidade da subclasse “Eevee”.

<p>A imagem mostra a linha evolutiva de <i>Eevee</i> em sete estágios: <i>Vaporeon</i> (água), <i>Jolteon</i> (elétrico), <i>Flareon</i> (fogo), <i>Espeon</i> (psíquico), <i>Umbreon</i> (escuro), <i>Glaceon</i> (gelado) e <i>Leafeon</i> (folha).</p>	<p>Annotations</p> <ul style="list-style-type: none"> ● rdfs:label Eevee Escreva nome de Escreva o valor <p>Classes</p> <ul style="list-style-type: none"> ● POKEMONS <p>Escreva nome de classe</p> <p>Relationships</p> <ul style="list-style-type: none"> □ EVOLUTION Espeon □ EVOLUTION Flareon □ EVOLUTION Jolteon □ EVOLUTION Leafon □ EVOLUTION Umbreon □ EVOLUTION Vaporeon □ TYPE NORMAL
<p>Figura 14.2: A imagem mostra a linha evolutiva do Eevee.</p>	<p>Figura 14.3: A Imagem mostra a relação evolutiva no Eeve na ontologia.</p>

- Subclasse de “TYPE”

Através das subclasses de “TYPE” temos a visão de qual tipo é melhor que outro através dos resultados de ataque: “NO_EFFECT”, “SUPER_EFFECTIVE” e “NOT_VERY_EFFECTIVE”.

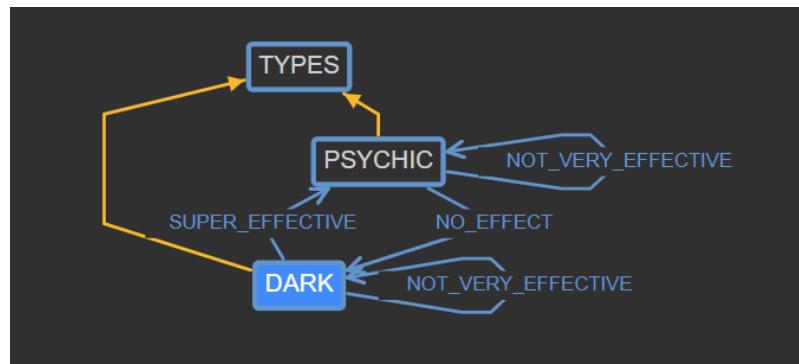


Figura 15: Imagem mostra o gráfico de entidade da subclasse “DARK”.

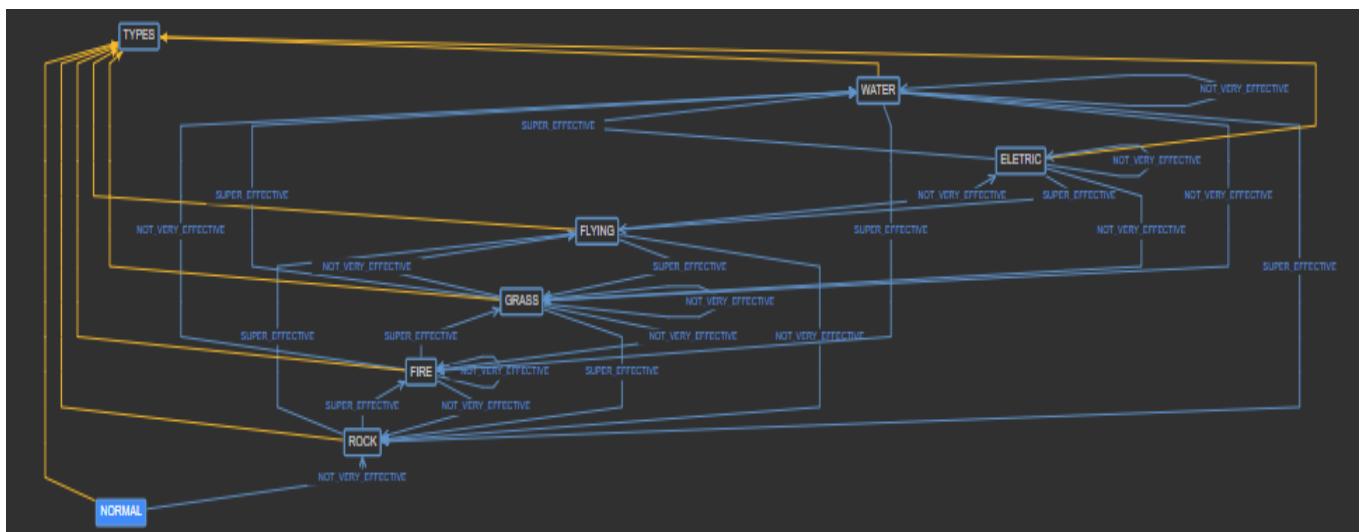


Figura 16: Imagem mostra o gráfico de entidade da subclasse “NORMAL”.

Apresentar imagens da ontologia implementada;

- Relações

Criamos 5 relações: “EVOLUTION”, “IS_TYPE”, “NO_EFFECT”, “SUPER_EFFECTIVE” e “NOT_VERY_EFFECTIVE”.

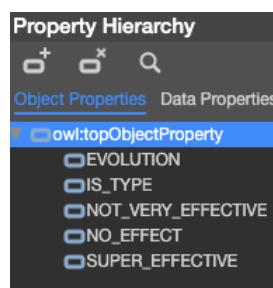


Figura 17: A imagem mostra todas as relações criadas.

- Classes e Subclasses

Criamos 2 classes: “*TYPES*” e “*POKEMONS*”. As subclasses de “*POKEMONS*” são: “*Bubassauro*”, “*Charizard*”, “*Charmander*”, “*Charmeleon*”, “*Ditto*”, “*Eevee*”, “*Espeon*”, “*Flareon*”, “*Geodude*”, “*Jonteon*”, “*Leafon*”, “*Mew*”, “*Pidgeotto*”, “*Pikachu*”, “*Raichu*”, “*Umbreon*”, “*Vaporeon*” e “*Zapdos*”. As de “*TYPE*” são: “*DARK*”, “*ELETRIC*”, “*FIRE*”, “*FLYING*”, “*GRASS*”, “*NORMAL*”, “*PSYCHIC*”, “*ROCK*” e “*WATER*”.

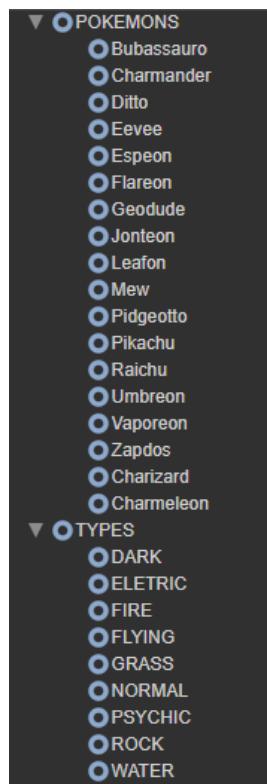


Figura 18: A imagem mostra as classes “*POKEMONS*” e “*TYPES*” com suas respectivas subclasses.

Apresentar imagens de 4 telas com consultas à ontologia:

Fizemos consultas para saber quais são as subclasses de “*POKEMONS*” e “*TYPE*”. Além disso quais *pokemóns* cadastrados tem evolução e quais possuem um tipo.

Query

Is a subclass of **POKEMONS** Direct

+ Execute

18 results

- Bubasauro
- Charizard
- Charmander
- Charmeleon
- Ditto
- Eevee
- Espeon
- Flareon
- Geodude
- Joneon
- Leafon
- Mew
- Pidgeotto
- Pikachu
- Raichu
- Umbreon
- Vaporeon
- Zapdos

Query

Is a subclass of **TYPES** Direct

+ Execute

9 results

- DARK
- ELECTRIC
- FIRE
- FLYING
- GRASS
- NORMAL
- PSYCHIC
- ROCK
- WATER

Figura 19: A imagem mostra uma consulta para saber quais são as subclasses da classe “POKEMONS”.

Figura 20: A imagem mostra uma consulta para saber quais são as subclasses da classe “TYPES”.

Query

Has a relationship on **EVOLUTION** that has any value

+ Execute

4 results

- Charmander
- Charmeleon
- Eevee
- Pikachu

Query

Has a relationship on **IS_TYPE** that has any value

+ Execute

18 results

- Bubasauro
- Charizard
- Charmander
- Charmeleon
- Ditto
- Eevee
- Espeon
- Flareon
- Geodude
- Joneon
- Leafon
- Mew
- Pidgeotto
- Pikachu
- Raichu
- Umbreon
- Vaporeon
- Zapdos

Figura 21: A imagem mostra quais *pokémons* possuem evolução.

Figura 22: A imagem mostra todos os *pokémons* que possuem tipo,no caso são todos os cadastrados.

Comentários sobre os resultados obtidos:

A experiência em criar uma ontologia foi muito interessante. Em um primeiro momento encontramos um pouco de dificuldade, mas com o tema bem definido achamos divertido e nos empolgamos, inclusive criamos mais relações do que o planejamento inicial.

Outro ponto observado foi na questão da ferramenta usada. Optamos por usar uma ferramenta online, para que todos pudessem usar ao mesmo tempo, mas não conseguimos exibir nela o domínio completo. Foi necessário usar a ferramenta no desktop, por isso a figura 12 não segue os padrões das outras.

Por fim, concluímos que nosso objetivo de auxiliar um usuário a saber qual tipo de *pokémon* tem um desempenho melhor contra outro foi alcançado. Navegando pelas hierarquias conseguimos ter essa resposta, mas, como pode-se ver nos exemplos, o gráfico pode ficar muito poluído, principalmente quando navegamos através dos *pokémons*.

Referências:

<https://www.youtube.com/watch?v=zWlgROKbrJk>

https://pokemonnewscenter.com.br/pokemon_go_2019/

<https://pt.wikipedia.org/wiki/Pok%C3%A9mon>

https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-99362010000100010

<https://br.pinterest.com/pin/863565297279087096/>

[https://pt.wikipedia.org/wiki/Pok%C3%A9mon_\(s%C3%AArie_de_jogos_eletr%C3%B3nicos\)](https://pt.wikipedia.org/wiki/Pok%C3%A9mon_(s%C3%AArie_de_jogos_eletr%C3%B3nicos))

https://pt.wikipedia.org/wiki/Tom_Gruber