Processamento e Representação de Conhecimento (4º ano de MIEI)

Projeto Individual

Ontologia de *Magic the Gathering*

Relatório de Desenvolvimento

Eduardo Cunha
a71940

Junho 2018

Conteúdo

Capítulo 1

Introdução

No âmbito desta unidade curricular este projeto visa a criação de uma ontologia de um domínio à nossa escolha, com o auxílio do *Protege*, sendo criado um ficheiro rdf para depois ser utilizado no *GraphDb*. Aqui, depois do ficheiro importado, é possível visualizar todas as relações existentes, através de um grafo, entre os individuos previamente definidos. Posteriormente, foram definidas queries em *Sparql* que permitem a busca e filtragem de informação presente no ficheiro rdf referido anteriormente. Por fim, foi definido uma aplicação web, através do Nodejs, que permite visualizar o resultado de todas as queries Sparql definidas anteriormente, com o auxílio do módulo *sparql-client-2*.

Assim, escolhi como domínio da minha ontologia o jogo de cartas *Magic the Gathering*, que consiste num jogo onde os jogadores utilizam um baralho de cartas construído de acordo com o seu modo individual de jogo para tentar vencer o baralho adversário. O principal objetivo deste projeto era representar o mais detalhadamente possível todos os conceitos que definem este jogo, como os tipos de cartas existentes, os tipos de torneio que atualmente são organizados, etc.

Capítulo 2

Ontologia

De modo a criar a ontologia o mais completa possível, recorri a algumas fontes oficiais sobre o assunto (estas estão presentes no capítulo referências), sendo que foi-me possível retirar os principais conceitos desta atividade. Assim, decidi explorar a classe "Carta"através dos seus tipos, cores e raridades, a classe "Pessoa"com as subclasses "Jogador"e "Designer", os tipos de torneio como "Block_Construced"e "Standard"entre outras. De facto, era possível explorar ainda mais este tema, no entanto, devido às limitações da máquina onde estava a testar a ontologia, decidi reduzir as classes às mais essenciais para a representação deste tema.

2.1 Classes

Deste modo, decidi implementar as classes abaixo listadas. Foram definidas 26 classes neste projeto. Por exemplo, a classe "Carta"pode ser subdividida na subclasse "Não Permanente", o que significa que a entra jogo, as suas habilidades são usadas, e é retirada imediatamente do jogo, e na subclasse "Não-permanente"em que as cartas ficam em jogo até serem "destruídas", sendo que estas próprias classes são desenvolvidas pelas sua próprias subclasses.

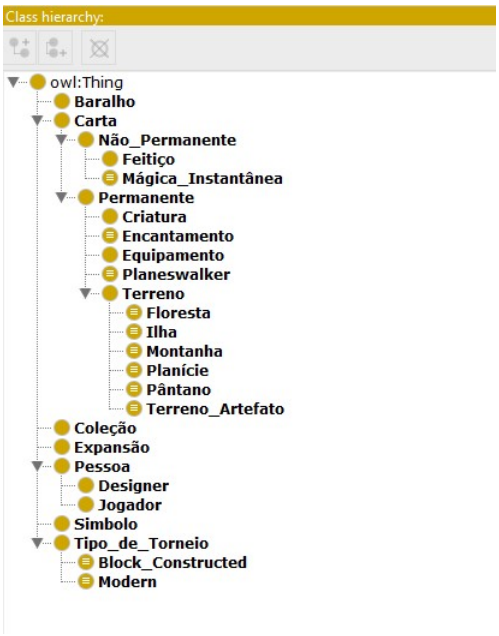


Figura 2.1: Listagem de todas as classes definidas

Também foram definidas as cardinalidades para casa classe, ou seja, as limitações que estas têm tanto em atributos como em object properties. Assim, abaixo estão demonstradas as restrinções das classes "Carta", "Floresta"e "Block_Constructed". A primeira é definida como tendo apenas uma cor, um nome e uma raridade, no entanto, como uma carta pode ser reeditada em várias vezes, pode pertencer a várias coleções, como está definido através da object property "pertence_à_coleção". Relativamente à classe, "Block_Constructed", sendo uma das regras deste tipo de torneio que os jogadores apenas podem jogar com uma coleção de cartas, foi definida a cardinalidade de "Coleção"neste tipo de torneios como exatamente 1. Por fim, sendo a classe "Floresta", uma subclasse de "Carta", também contém as suas restrições, em adição às atribuições da própria subclasse. ou seja, uma carta que seja terreno e é da cor verde, obrigatoriamente será uma "Floresta".

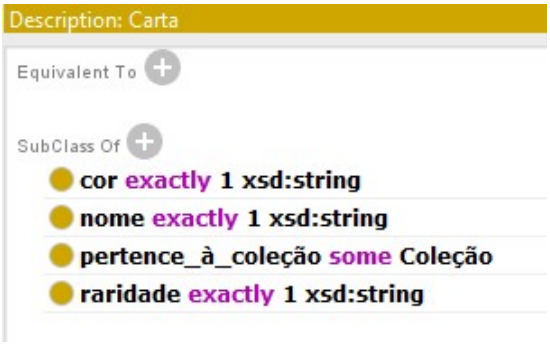


Figura 2.2: Cardinalidades da classe Carta

Figura 2.3: Cardinalidades da classe Block_Constructed

Figura 2.4: Cardinalidades da classe Floresta

2.2 Object Properties

Já com as classes criadas, era necessário criar as propriedades que fossem capazes de as relacionar. Desta forma, foram definidas as *object properties* para todas as classes criadas anteriormente. Assim estão listadas 29 object properties nesta ontologia. De modo a tentar elaborar o mais possível a ontologia, decidi implementar relações entre praticamente todas as classes, também com recurso às *Chain Subproperties*, ou seja, através de duas relações com domínios e contra-domínios "A -> B"e "B -> C", criar uma relação "A -> C", como por exemplo, um designer "desenhou"uma carta e esta pertence a uma coleção ("pertence_à_coleção"), logo esse designer também pertence a essa coleção ("designer_pertence_à_coleção").
A exposição destas relações, entre alguns dos indivíduos criados será feita num capítulo mais adiante no relatório.

Figura 2.5: Listagem de todas as relações definidas

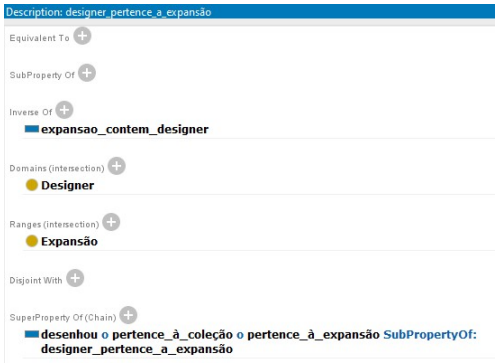


Figura 2.6: Chain Property designer_pertence_à_expansão

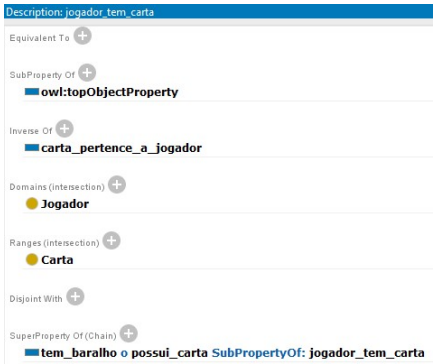


Figura 2.7: Chain Property jogador_tem_carta

2.3 Data Properties

De modo a conceder atributos às classes implementadas, utilizamos as "Data Properties"que permitem conferir às classes referidas, tipos básicos com strings e inteiros. Deste modo, muitas das classes possuem atributos únicos dessas mesmas, o que permite ao *reasoner* inferir o tipo de um indíviduo, apenas pelo reconhecimento de um atributo.Neste projeto, apenas foram implementadas data properties do tipo *xsd:string* e *xsd:integer*. Foram criados 14 data properties diferentes para esta ontologia.
Por exemplo, os atributos ataque e defesa são únicos da subclasse "Criatura", ou seja, qualquer que indivíduo que contenha essas características, é automaticamente reconhecido como uma "Criatura"(para além de todas as superclasses desta), assim como o atributo "Pontos_de_vida"funciona de forma semelhante para a subclasse "Planeswalker". Também foram definidos várias propriedades "Nome", visto que, como a um atributo tem de ser atribuído um domínio, qualquer classe com esta propriedade seria considerada do tipo desse mesmo domínio, o que estaria errado.

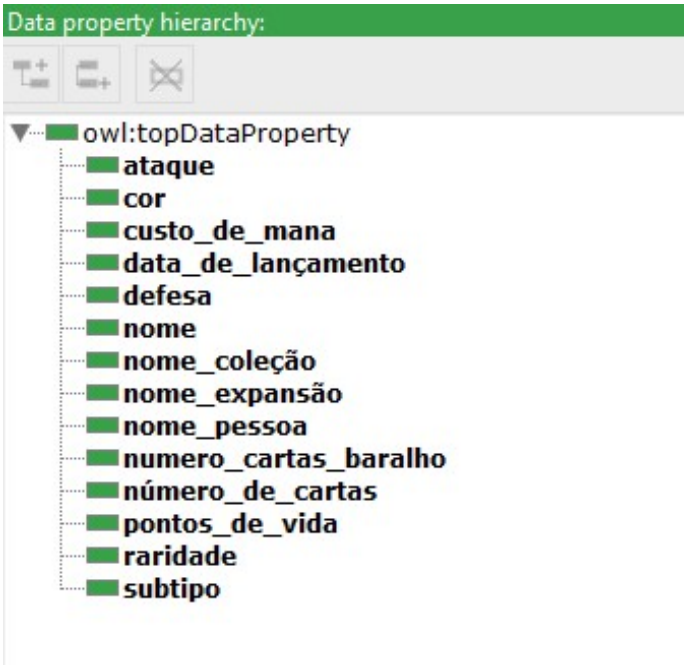


Figura 2.8: Listagem de todas os atributos definidos

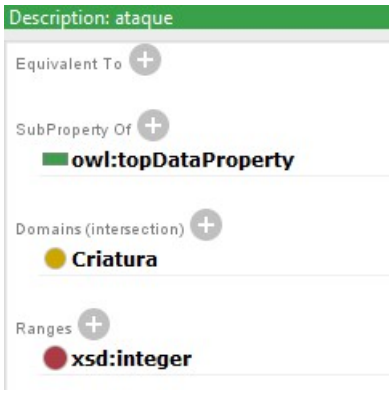


Figura 2.9: Tipo e domínio da data property Ataque

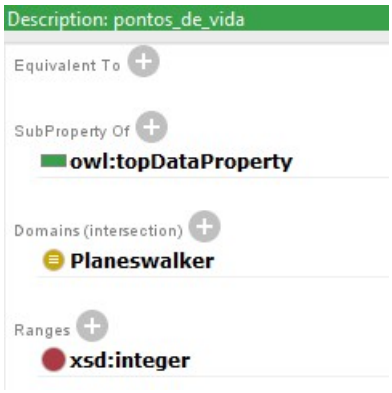


Figura 2.10: Tipo e domínio da data property pontos_de_vida

2.4 Individuals

De modo a finalizar a ontologia foi necessário popula-la. Apesar de uma elaborada pesquisa, não foi possível encontrar um dataset oficial com toda a informação referente ao jogo *Magic The Gathering*, pelo que as instâncias foram introduzidas manualmente. Foram criados indivíuos para todas as classes definidas, sendo que, grande parte destes correspondem a informação real, retirada de fontes oficiais deste jogo, sendo que esta ontologia conta com 110 indíviduos deste domínio. Abaixo estão demonstradas algumas das relações inferidas pelo *reasoner*, sendo que, neste exemplo, a única coisa que têm em comum é o "Baralho_Eduardo", sendo que, este contém várias cartas, entre elas a *Archive_Trap*, e o jogador "Eduardo_Cunha", tem o baralho "Baralho_Eduardo".

Figura 2.11: Propriedades inferidas da carta Archive_Trap

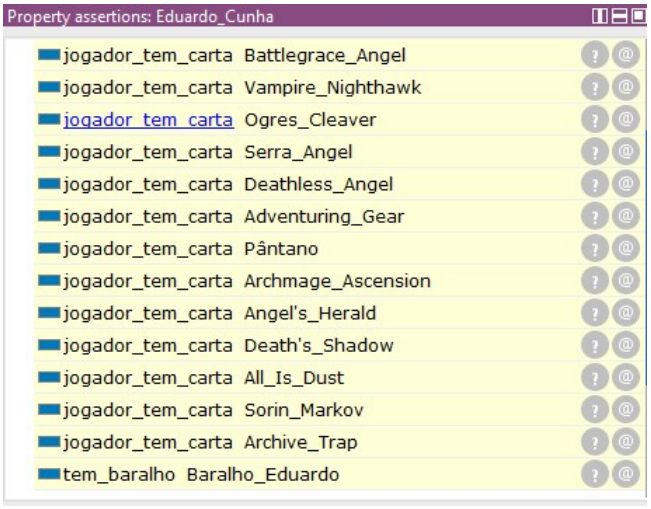


Figura 2.12: Propriedades inferidas do jogador Eduardo_Cunha

Capítulo 3

Graphdb

Depois da ontologia estar corretamente criada, foi gerado um ficheiro rdf de modo a permitir a visualização desta, e posteriormente manipular a informação nela existente, através da query language *Sparql*. Assim, abaixo está exposto um grafo que demonstra algumas das relações entre os indivíduos, partindo da entidade "Eduardo_Cunha".

Figura 3.1: Grafo ilustrativo de uma parte da ontologia

Capítulo 4

Queries Sparql

De modo a explorar toda a ontologia foram criadas várias queries em Sparql, como por exemplo, filtram as cartas quanto aos seus atributos. Inicialmente, os prefixos foram definidos da seguinte forma:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX mtg: <http://www.semanticweb.org/eduardo/ontologies/2018/5/untitled-ontology-14#>
```

1. Listar o número de cartas existentes na ontologia de acordo com a sua cor (estão a ser consideradas 6 cores).

```
select ?cor (count(distinct ?card) as ?contagens)
where{
    ?card mtg:cor ?cor.
}
GROUP BY ?cor
```

2. Apresentar o nome de todas as criaturas verdes.

```
select ?name ?color ?collection
where{
     ?c mtg:tem_carta ?card.
     ?card mtg:nome ?name.
     ?card mtg:cor ?color.
     ?c mtg:nome_coleção ?collection.
}
```

3. Listar os nomes de todas cartas, a respetiva cor e a coleção a que estas pertencem (podendo pertencer a mais do que uma coleção).

```
select ?name ?color ?collection
where{
     ?c mtg:tem_carta ?card.
     ?card mtg:nome ?name.
     ?card mtg:cor ?color.
     ?c mtg:nome_coleção ?collection.
}
```

4. Listar todos os feitiços azuis que são mítico-raros.

```
select ?card ?rarity
where{
    ?card rdf:type mtg:Feitiço.
    ?card mtg:raridade ?rarity.
    FILTER(regex(str(?rarity), "Mitica")).
}
```

5. Sendo que todas as cartas possuem uma imagem característica, esta querie lista todas as cartas desenhadas pelo designer "Jason Chan".

```
select ?card ?designer
where{
    ?designer mtg:desenhou ?card.
    FILTER(regex(str(?designer), "Jason_Chan")).
}
```

6. Apresentação do nome das cartas com o subtipo "Anjo", a respetiva cor e a coleção à qual pertencem.

```
select ?card ?subtipo ?cor
where{
?c mtg:tem_carta ?card.
    ?card mtg:nome ?name.
    ?card mtg:subtipo ?subtipo.
    FILTER(regex(str(?subtipo), "Angel")).
    ?card mtg:cor ?cor.
}
```

7. Listagem de todas as cartas permitidas no torneio "Alara"e que estão presentes no "Baralho_Eduardo".

```
select ?card ?rarity
where{
    ?col mtg:pertence_à_expansão ?exp.
    FILTER(regex(str(?exp), "Alara")).
    ?card mtg:está_no_baralho ?deck.
    FILTER(regex(str(?deck), "Baralho_Eduardo")).
    ?card mtg:pertence_à_coleção ?col.
    ?card mtg:raridade ?rarity.
}
```

9. Apresentação do nome de todas as cartas pretas presentes em qualquer baralho e o seu custo de mana.

```
select ?carta ?custo_de_Mana ?baralho
where{
    ?carta mtg:cor ?cor
    FILTER(regex(str(?cor), "Preto")).
    ?carta mtg:custo_de_mana ?custo_de_Mana.
    ?carta mtg:está_no_baralho ?baralho.
}
```

10. Exibição do nome de todas as cartas existentes da expansão Zendikar cuja raridade é mítica-rara.

```
select ?carta ?raridade ?cor ?expansão
where{
    ?expansão mtg:tem_coleção ?col.
    FILTER(regex(str(?expansão), "Exp_Zendikar")).
    ?carta mtg:raridade ?raridade.
    FILTER(regex(str(?raridade), "Mitica")).
    ?col mtg:tem_carta ?carta.
    ?carta mtg:cor ?cor.
}
```

11. Listagem de todas as cartas permitidas no "torneio_block"e que estão no baralho "Baralho_Eduardo".

```
select ?carta ?baralho ?coleção ?cor
where{
?tourney mtg:torneio_permite_coleção ?coleção
    FILTER(regex(str(?tourney), "Torneio_Block")).
    ?baralho mtg:possui_carta ?carta.
    FILTER(regex(str(?baralho), "Baralho_Eduardo")).
    ?coleção mtg:tem_carta ?carta.
?carta mtg:cor ?cor.
}
```

11. Listagem de todas as cartas permitidas no "torneio_block"e que estão no baralho "Baralho_Eduardo".

```
select ?carta ?baralho ?coleção ?cor
where{
    ?tourney mtg:torneio_permite_coleção ?coleção
    FILTER(regex(str(?tourney), "Torneio_Block")).
    ?baralho mtg:possui_carta ?carta.
    FILTER(regex(str(?baralho), "Baralho_Eduardo")).
    ?coleção mtg:tem_carta ?carta.
    ?carta mtg:cor ?cor.
}
```

12. Exibição da lista ordenada descendentemente de todas as criaturas pelo seu ataque, imprimindo o seu ataque, defesa, custo de mana e raridade,.

```
select ?carta ?ataque ?defesa ?custo_de_Mana ?raridade
where{
    ?carta mtg:ataque ?ataque.
    ?carta mtg:defesa ?defesa.
    ?carta mtg:custo_de_mana ?custo_de_Mana.
    ?carta mtg:raridade ?raridade.
}
order by DESC(?ataque)
```

Capítulo 5

Aplicação Web

Como fase final deste trabalho, foi elaborado um website, com o auxílio do *Nodejs*, nomeadamente através da sua framework *Express*, onde eram executadas todas as queries sparql demonstradas no capítulo anterior e eram imprimidas em formato html. De facto, partindo do código usado nas aulas, é possível verificar o resultado correto de todas as queries implementadas, sendo que, para a compilação do código *Sparql* foi utilizado o módulo *sparql-client-2*. Assim, na página inicial do website estão listados os objetivos de cada queries, sendo que ao clicar numa destas, a querie é executada e os seus formatos são imprimidos em forma de tabela.

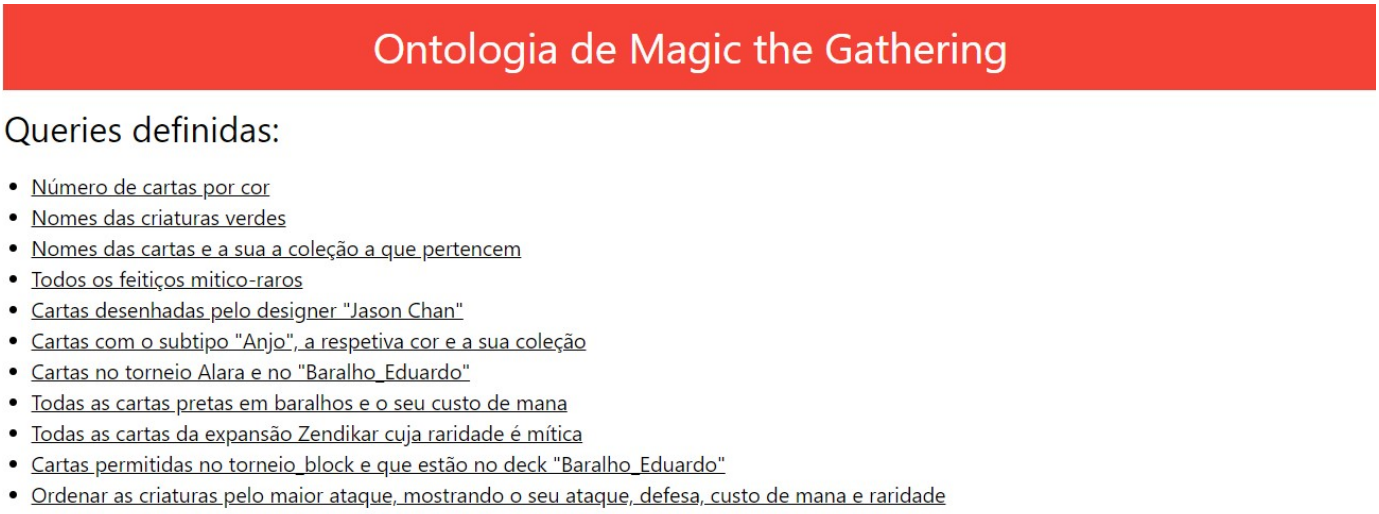


Figura 5.1: Homepage da aplicação

Figura 5.2: Resultados da query 10 na aplicação criada

Figura 5.3: Resultados da query 8 na aplicação criada

Capítulo 6

Conclusão

Após o final da elabaroção deste projeto é possível concluir que, através da aplicação dos conhecimentos adquiridos durante as aulas, nomeadamente nas ferramentas *Protege* e *GraphDB*, os objetivos que dizem respeito à criação de uma ontologia sobre o domínio *Magic the Gathering* foram obtidos com sucesso, estando todos os principais conceitos relativos a este corretamente definidos e utilizados. O principal obstáculo neste projeto foi em definir corretamente todas as classes necessárias para a criação desta ontologia, sendo que, existem várias restrições neste tipo de passatempo. Também não foi possível encontrar um dataset fiável que contivesse todas as cartas deste jogo, sendo que a quantidade de indivíduos presentes neste projeto foi abaixo do inicialmente programado.

Como trabalho futuro, gostaria de popular a ontologia com todas as cartas existentes neste jogo, de modo a explorar ainda mais as capacidades do Sparql e da ontologia criada. Da mesma forma, também gostaria de implementar ainda mais queries em Sparql relativas a este tema. Por fim, uma melhoria na aplicação web, nomeadamente a nível visual seria uma boa aposta também como trabalho futuro, e possívelmente a implementação de um navegador que permitisse a um utilizador explorar o potencial desta ontologia.

Referências

[1] http://www.magictuga.com/

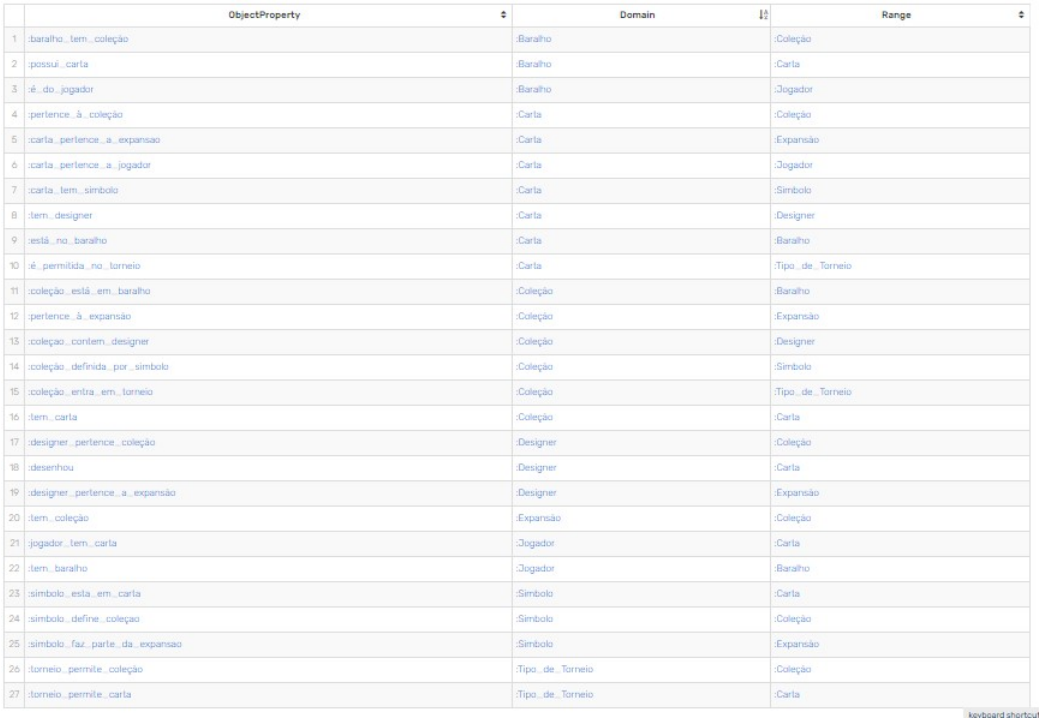
[2] https://en.wikipedia.org/wiki/Magic:_The_Gathering

[3] https://magic.wizards.com/en

Capítulo 7

Anexos

Object Properties



	ObjectProperty	Domain	Range
1	:baralho_tem_coleção	:Baralho	:Coleção
2	:possui_carta	:Baralho	:Carta
3	:é_do_jogador	:Baralho	:Jogador
4	:pertence_à_coleção	:Carta	:Coleção
5	:carta_pertence_a_expansao	:Carta	:Expansão
6	:carta_pertence_a_jogador	:Carta	:Jogador
7	:carta_tem_simbolo	:Carta	:Simbolo
8	:tem_designer	:Carta	:Designer
9	:está_no_baralho	:Carta	:Baralho
10	:é_permitida_no_torneio	:Carta	:Tipo_de_Torneio
11	:coleção_está_em_baralho	:Coleção	:Baralho
12	:pertence_à_expansão	:Coleção	:Expansão
13	:coleçao_contem_designer	:Coleção	:Designer
14	:coleção_definida_por_simbolo	:Coleção	:Simbolo
15	:coleção_entra_em_torneio	:Coleção	:Tipo_de_Torneio
16	:tem_carta	:Coleção	:Carta
17	:designer_pertence_coleção	:Designer	:Coleção
18	:desenhou	:Designer	:Carta
19	:designer_pertence_a_expansão	:Designer	:Expansão
20	:tem_coleção	:Expansão	:Coleção
21	:jogador_tem_carta	:Jogador	:Carta
22	:tem_baralho	:Jogador	:Baralho
23	:simbolo_esta_em_carta	:Simbolo	:Carta
24	:simbolo_define_coleçao	:Simbolo	:Coleção
25	:simbolo_faz_parte_da_expansao	:Simbolo	:Expansão
26	:torneio_permite_coleção	:Tipo_de_Torneio	:Coleção
27	:torneio_permite_carta	:Tipo_de_Torneio	:Carta

keyboard shortcut

Figura 7.1: Especificações de todas as object properties

Query Sparql

```
SELECT ?ObjectProperty ?Domain ?Range
where{
    ?ObjectProperty rdf:type owl:ObjectProperty.
    ?ObjectProperty rdfs:domain ?Domain.
    ?ObjectProperty rdfs:range ?Range.
}GROUP BY ?ObjectProperty ?Domain ?Range
```

Data Properties

	ObjectProperty	Domain	Range
1	:baralho_tem_coleção	:Baralho	:Coleção
2	:possui_carta	:Baralho	:Carta
3	:é_do_jogador	:Baralho	:Jogador
4	:pertence_à_coleção	:Carta	:Coleção
5	:carta_pertence_a_expansao	:Carta	:Expansão
6	:carta_pertence_a_jogador	:Carta	:Jogador
7	:carta_tem_simbolo	:Carta	:Simbolo
8	:tem_designer	:Carta	:Designer
9	:está_no_baralho	:Carta	:Baralho
10	:é_permitida_no_torneio	:Carta	:Tipo_de_Torneio
11	:coleção_está_em_baralho	:Coleção	:Baralho
12	:pertence_à_expansão	:Coleção	:Expansão
13	:coleçao_contem_designer	:Coleção	:Designer
14	:coleção_definida_por_simbolo	:Coleção	:Simbolo
15	:coleção_entra_em_torneio	:Coleção	:Tipo_de_Torneio
16	:tem_carta	:Coleção	:Carta
17	:designer_pertence_coleção	:Designer	:Coleção
18	:desenhou	:Designer	:Carta
19	:designer_pertence_a_expansão	:Designer	:Expansão
20	:tem_coleção	:Expansão	:Coleção
21	:jogador_tem_carta	:Jogador	:Carta
22	:tem_baralho	:Jogador	:Baralho
23	:simbolo_esta_em_carta	:Simbolo	:Carta
24	:simbolo_define_coleçao	:Simbolo	:Coleção
25	:simbolo_faz_parte_da_expansao	:Simbolo	:Expansão
26	:torneio_permite_coleção	:Tipo_de_Torneio	:Coleção
27	:torneio_permite_carta	:Tipo_de_Torneio	:Carta

keyboard shortcut

Figura 7.2: Especificações de todas as data properties

Query Sparql

```
SELECT ?DatatypeProperty ?Domain ?Range
where{
    ?DatatypeProperty rdf:type owl:DatatypeProperty.
    ?DatatypeProperty rdfs:domain ?Domain.
    ?DatatypeProperty rdfs:range ?Range.
}GROUP BY ?DatatypeProperty ?Domain ?Rang
```

Código javascript implementado na aplicação (Ficheiro Query.js)

```
var express = require('express');
var router = express.Router();

const SparqlClient = require('sparql-client-2')
const SPARQL = SparqlClient.SPARQL
const endpoint = 'http://localhost:7200/repositories/PRC_2018'
const myupdateEndpoint = 'http://localhost:7200/repositories/PRC_2018/statements'

var client = new SparqlClient( endpoint, {updateEndpoint: myupdateEndpoint,
                               defaultParameters: {format: 'json'}})

client.register({rdf: 'http://www.w3.org/1999/02/22-rdf-syntax-ns#',
                 mtg: 'http://www.semanticweb.org/eduardo/ontologies/2018/5/untitled-ontology-14#'})

/* GET home page. */
router.get('/1', function(req, res, next) {
    var query = "select ?cor (count(distinct ?card) as ?contagens)\n" +
                "WHERE {\n" +
                "?card mtg:cor ?cor.\n}" +
                "GROUP BY ?cor "
    client.query(query)
            .execute()
            .then(function(qres){
                console.log(JSON.stringify(qres))
                var resList = qres.results.bindings
                var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Cor </th>\n" + "\t\t<th>
Contagem </th>\n" + "\t</tr>"

                for(var i in resList){
                    var cor = resList[i].cor.value
                    var contagens = resList[i].contagens.value
                    var color = cor.slice(cor.indexOf('#')+1)
                    var contagem = contagens.slice(contagens.indexOf('#')+1)

                    dot += '\t<tr>\n' + '\t\t<th>' + cor + '</th>\n' + '\t\t<th>' + contagem + '</th>\n'
+ '\t</tr>'
                }
                    dot += "</table>"
                res.render("result", {renderingCode: dot})
            })
            .catch((error)=>{
                res.render("error", {error:error})
            })
})

router.get('/2', (req, res, next)=>{
    var did = req.params.did
    var query = "select ?name ?color\n " +
                "where{\n " +
                "?c mtg:tem_carta ?card.\n" +
                "?card mtg:nome ?name.\n" +
                "?card mtg:ataque ?at.\n" +
```

```
                "?card mtg:cor ?color.\n" +
                "FILTER(regex(str(?color), \"Verde\")).\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dname = resList[0].name.value
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Nome </th>\n" + "\t\t<th>
Cor </th>\n" + "\t</tr>"

            for(var i in resList){
                var name = resList[i].name.value
                var color = resList[i].color.value
                var cor = color.slice(color.indexOf('#')+1)
                var nome = name.slice(name.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + name + '</th>\n' + '\t\t<th>' + color + '</th>\n'
+ '\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/3', (req, res, next)=>{
    var did = req.params.did
    var query = "select ?name ?color ?collection\n " +
                "where{\n " +
                "?c mtg:tem_carta ?card.\n" +
                "?card mtg:nome ?name.\n" +
                "?card mtg:cor ?color.\n" +
                "?c mtg:nome_coleção ?collection.\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Nome </th>\n" + "\t\t<th>
Cor </th>\n" + "\t\t<th> Coleção </th>\n" + "\t</tr>"

            for(var i in resList){
                var color = resList[i].color.value
                var cor = color.slice(color.indexOf('#')+1)
                var name = resList[i].name.value
                var nome = name.slice(name.indexOf('#')+1)
                var collection = resList[i].collection.value
                var colecao = collection.slice(collection.indexOf('#')+1)
```

```
                dot += '\t<tr>\n' + '\t\t<th>' + name + '</th>\n' + '\t\t<th>' + color + '</th>\n'
+ '\t\t<th>' + colecao + '</th>\n' + '\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/4', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?card ?rarity\n" +
                "where{\n" +
                "?card rdf:type mtg:Feitiço.\n" +
                "?card mtg:raridade ?rarity.\n" +
                "FILTER(regex(str(?rarity), \"Mitica\")).\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Raridade </th>\n" + "\t</tr>"

            for(var i in resList){
                var card = resList[i].card.value
                var carta = card.slice(card.indexOf('#')+1)
                var rarity = resList[i].rarity.value
                var raridade = rarity.slice(rarity.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + carta + '</th>\n' + '\t\t<th>' + rarity + '</th>\n'
+ '\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/5', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?card ?designer\n" +
                "where{\n" +
                "?designer mtg:desenhou ?card.\n" +
                "FILTER(regex(str(?designer), \"Jason_Chan\")).\n" +
                "}"

    client.query(query)
        .execute()
```

```
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Designer </th>\n" + "\t</tr>"

            for(var i in resList){
                var card = resList[i].card.value
                var carta = card.slice(card.indexOf('#')+1)
                var designer = resList[i].designer.value
                var des = designer.slice(designer.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + carta + '</th>\n' + '\t\t<th>' + des + '</th>\n' +
'\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/6', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?card ?subtipo ?cor ?colecao\n" +
                "where{\n" +
                "?c mtg:tem_carta ?card.\n" +
                "?card mtg:nome ?name.\n" +
                "?card mtg:subtipo ?subtipo.\n" +
                "FILTER(regex(str(?subtipo), \"Angel\")).\n" +
                "?card mtg:cor ?cor.\n" +
                "?colecao mtg:tem_carta ?card" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Subtipo </th>\n" + "\t\t<th> Cor </th>\n" + "\t\t<th> Coleção </th>\n" + "\t</tr>"

            for(var i in resList){
                var card = resList[i].card.value
                var carta = card.slice(card.indexOf('#')+1)
                var subtipo = resList[i].subtipo.value
                var subtype = subtipo.slice(subtipo.indexOf('#')+1)
                var cor = resList[i].cor.value
                var color = cor.slice(cor.indexOf('#')+1)
                var colecao = resList[i].colecao.value
                var collection = colecao.slice(colecao.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' +  carta + '</th>\n' + '\t\t<th>' + subtype + '</th>\n'
+ '\t\t<th>' + color + '</th>\n' + '\t\t<th>' + collection + '</th>\n' + '\t</tr>'
            }
```

```
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/7', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?card ?rarity\n" +
                "where{\n" +
                "?col mtg:pertence_à_expansão ?exp.\n" +
                "FILTER(regex(str(?exp), \"Alara\")).\n" +
                "?card mtg:está_no_baralho ?deck.\n" +
                "FILTER(regex(str(?deck), \"Baralho_Eduardo\")).\n" +
                "?card mtg:pertence_à_coleção ?col.\n" +
                "?card mtg:raridade ?rarity.\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Raridade </th>\n" + "\t</tr>"

            for(var i in resList){
                var card = resList[i].card.value
                var carta = card.slice(card.indexOf('#')+1)
                var rarity = resList[i].rarity.value
                var raridade = rarity.slice(rarity.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + carta + '</th>\n' + '\t\t<th>' + raridade + '</th>\n'
+ '\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/8', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?carta ?custo_de_Mana ?baralho\n" +
                "where{\n" +
                "?carta mtg:cor ?cor.\n" +
                "FILTER(regex(str(?cor), \"Preto\")).\n" +
                "?carta mtg:custo_de_mana ?custo_de_Mana.\n" +
                "?carta mtg:está_no_baralho ?baralho.\n" +
                "}"
```

```
    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Custo de Mana </th>\n" + "\t\t<th> Baralho </th>\n" + "\t</tr>"

            for(var i in resList){
                var carta = resList[i].carta.value
                var card = carta.slice(carta.indexOf('#')+1)
                var custo_de_Mana = resList[i].custo_de_Mana.value
                var mana_cost = custo_de_Mana.slice(custo_de_Mana.indexOf('#')+1)
                var baralho = resList[i].baralho.value
                var deck = baralho.slice(baralho.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + card + '</th>\n' + '\t\t<th>' + mana_cost + '</th>\n'
+  '\t\t<th>' + deck + '</th>\n' +'\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/9', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?carta ?raridade ?cor ?expansão\n" +
                "where{\n" +
                "?expansão mtg:tem_coleção ?col.\n" +
                "FILTER(regex(str(?expansão), \"Exp_Zendikar\")).\n" +
                "?carta mtg:raridade ?raridade.\n" +
                "FILTER(regex(str(?raridade), \"Mitica\")).\n" +
                "?col mtg:tem_carta ?carta.\n" +
                "?carta mtg:cor ?cor.\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Raridade </th>\n" + "\t\t<th> Cor </th>\n" + "\t\t<th> Expansão </th>\n" + "\t</tr>"

            for(var i in resList){
                var carta = resList[i].carta.value
                var card = carta.slice(carta.indexOf('#')+1)
                var raridade = resList[i].raridade.value
                var rarity = raridade.slice(raridade.indexOf('#')+1)
                var cor = resList[i].cor.value
                var color = cor.slice(cor.indexOf('#')+1)
                var expansão = resList[i].expansão.value
                var expansion = expansão.slice(expansão.indexOf('#')+1)
```

```
                dot += '\t<tr>\n' + '\t\t<th>' + card + '</th>\n' + '\t\t<th>' + rarity + '</th>\n'
+  '\t\t<th>' + color + '\t\t<th>' + expansion + '</th>\n' +'\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/10', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?carta ?baralho ?coleção ?cor\n" +
                "where{\n" +
                "?tourney mtg:torneio_permite_coleção ?coleção\n" +
                "FILTER(regex(str(?tourney), \"Torneio_Block\")).\n" +
                "?baralho mtg:possui_carta ?carta.\n" +
                "FILTER(regex(str(?baralho), \"Baralho_Eduardo\")).\n" +
                "?carta mtg:cor ?cor.\n" +
                "}"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Baralho </th>\n" + "\t\t<th> Coleção </th>\n" + "\t\t<th> Cor </th>\n" + "\t</tr>"

            for(var i in resList){
                var carta = resList[i].carta.value
                var card = carta.slice(carta.indexOf('#')+1)
                var cor = resList[i].cor.value
                var color = cor.slice(cor.indexOf('#')+1)
                var baralho = resList[i].baralho.value
                var deck = baralho.slice(baralho.indexOf('#')+1)
                var coleção = resList[i].coleção.value
                var collection = coleção.slice(coleção.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + card + '</th>\n' + '\t\t<th>' +  deck + '</th>\n'
+ '\t\t<th>' + collection  +'</th>\n' + '\t\t<th>' + color + '</th>\n' +'\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

router.get('/11', (req, res, next)=>{
    var cid = req.params.id
    var query = "select ?carta ?ataque ?defesa ?custo_de_Mana ?raridade\n" +
```

```
                "where{\n" +
                "?carta mtg:ataque ?ataque.\n" +
                "?carta mtg:defesa ?defesa.\n" +
                "?carta mtg:custo_de_mana ?custo_de_Mana.\n" +
                "?carta mtg:raridade ?raridade.\n" +
                "?carta mtg:cor ?cor.\n" +
                "}\n" +
                "order by DESC(?ataque)"

    client.query(query)
        .execute()
        .then(function(qres){
            var resList = qres.results.bindings
            var dot = "<table style=\"width:50%\">\n" + "\t<tr>\n" + "\t\t<th> Carta </th>\n" + "\t\t<th>
Ataque </th>\n" + "\t\t<th> Defesa </th>\n" + "\t\t<th> Custo de Mana </th>\n" + "\t\t<th> Raridade
</th>\n" + "\t</tr>"

            for(var i in resList){
                var carta = resList[i].carta.value
                var card = carta.slice(carta.indexOf('#')+1)
                var raridade = resList[i].raridade.value
                var rarity = raridade.slice(raridade.indexOf('#')+1)
                var ataque = resList[i].ataque.value
                var attack = ataque.slice(ataque.indexOf('#')+1)
                var defesa = resList[i].defesa.value
                var defense = defesa.slice(defesa.indexOf('#')+1)
                var custo_de_Mana = resList[i].custo_de_Mana.value
                var mana_cost = custo_de_Mana.slice(custo_de_Mana.indexOf('#')+1)

                dot += '\t<tr>\n' + '\t\t<th>' + card + '</th>\n' + '\t\t<th>' + attack + '</th>\n'
+ '\t\t<th>' + defense + '</th>\n' + '\t\t<th>' + mana_cost + '</th>\n' + '\t\t<th>' + rarity + '</th>\n'
+ '\t</tr>'
            }
                dot += "</table>"
            res.render("result", {renderingCode: dot})
        })
        .catch((error)=>{
            res.render("error", {error:error})
        })
})

module.exports = router;
```