

Escola de Engenharia da Universidade do Minho
Departamento de Informática
Mestrado Integrado em Engenharia Informática
Processamento e Representação de Conhecimento

Ontologia - Futebol

João Gomes, A74033

14 de Junho de 2019

Resumo

Trabalho realizado no âmbito da unidade curricular Processamento e Representação de Conhecimento do perfil de Processamento de Linguagens e Conhecimento do 4^o ano do Mestrado Integrado em Engenharia Informática.

O projeto consiste na modelação de uma ontologia sobre um tema a escolha, bem como na criação de uma aplicação web capaz de apresentar os dados convertidos para *RDF* e carregados no *GraphDB*.

Conteúdo

1	Introdução	2
2	Modelação	3
2.1	Classes	3
2.2	Object Properties	4
2.3	Relações	5
2.4	Data Properties	6
3	Script	7
3.1	Pedidos Efetuados	7
3.2	Sleep	7
4	Aplicação	8
4.1	API de Dados	8
4.1.1	Controllers	8
4.1.2	Rotas	10
4.2	Interface	13
5	Conclusões	15
6	Anexos	16
6.1	Script	16

Capítulo 1

Introdução

Neste projeto, foi modelada uma ontologia sobre futebol, esta ontologia a constituição de campeonatos de clubes de futebol.

Os dados que constituem esta ontologia foram retirados de uma API de dados existente do site (<https://www.football-data.org/>).

Sob esta ontologia criada, foi criada uma API de Dados que acede ao *GraphDB* e uma aplicação em *Vue.js* que acede a API de dados e mostra os mesmos.

As funcionalidades da aplicação são as seguintes: consultar lista de competições, consultar competição, consultar classificação, consultar equipa, consultar jogo, consultar jogador, consultar treinador, consultar arbitro.

Para desenvolver este projeto foi utilizado *Protegé* para a modulação, *NodeJS* para retirar os dados da API e converter em *RDF*, foi utilizado ainda para a criação de API de Dados, como Sistema de Base de Dados foi utilizado *GraphDB*. *Vue.js* foi utilizado para o desenvolvimento da Interface.

O relatório em questão é constituído pelos seguintes pontos que em seguida iremos apresentar:

- Modelação da Ontologia;
- *Script* criado para aceder e converter os dados;
- Aplicação desenvolvida API de Dados e Interface;
- Conclusões finais.

Capítulo 2

Modelação

A primeira etapa do desenvolvimento deste projeto foi a modulação depois de estudada a informação que se podia obter da API de Dados.

2.1 Classes

Começou-se por definir as Classes que iriam fazer parte da Ontologia, sendo elas:

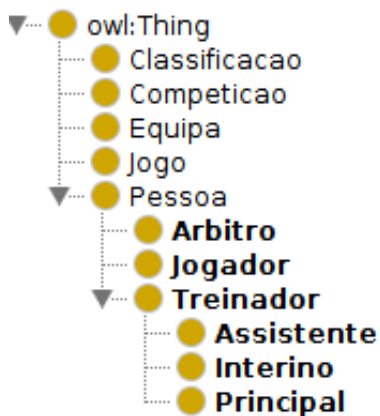


Figura 2.1: Classes

A figura anterior mostra as Classes criadas na Ontologia. Por exemplo a Classe **Treinador**, possui três subclasses sendo elas **Assistente**, **Interino**, **Principal**, esta classe é diferente de todas as outras devido a este fator e a ser também uma subclasse da classe **Pessoa**.

2.2 Object Properties

De modo a se entender como as classes se vão interligar, é necessário mostrar as *object properties* criadas. Isto são as propriedades que têm como domínio uma Classe e como contradomínio outra Classe.

Na ontologia criada temos as seguintes *object properties*:

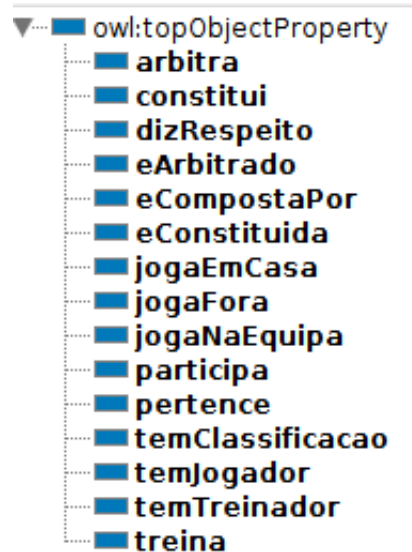


Figura 2.2: Object Properties

De seguida é explicado que classes são interligadas pelas *object properties* referidas.

2.3 Relações

- **arbitra:**

Relaciona a classe *Arbitro* e *Jogo*, isto é, pertence ao domínio *Arbitro* e tem como contradomínio *Jogo*, sendo propriedade inversa da *object Propertie* **eArbitrado**.

- **eArbitrado:**

Esta propriedade é inversa da propriedade anterior, tendo como domínio *Jogo* e contradomínio *Arbitro*.

- **constitui:**

Relaciona a classe *Jogo* com *Competicao*, ou seja apresenta domínio em *Jogo* e contradomínio em *Competicao*

- **eConstituida:**

Esta propriedade é inversa da propriedade **constitui**, logo tem domínio *Competicao* e contradomínio *Jogo*

- **dizRespeito :**

Esta propriedade relaciona a *Classificacao* com a *Equipa*, sendo o domínio *Classificacao* e contradomínio *Equipa*.

- **temClassificacao:** Propriedade inversa da anterior.

- **eCompostaPor:**

Esta propriedade relaciona a *Competicao* com a *Equipa*, sendo o domínio *Competicao* e o contradomínio a *Equipa*.

- **participa:**

Propriedade inversa da anterior.

- **jogaEmCasa:**

Esta propriedade relaciona o *Jogo* com a *Equipa*, como num jogo de futebol existem sempre duas equipas temos que ter uma propriedade para identificar quem joga em Casa e por sua vez quem joga Fora, como vamos verificar na propriedade a seguir.

- **jogaFora:**

Tal como foi dito anteriormente esta propriedade relaciona na mesma a classe *Jogo* e *Equipa* representando assim duas equipas por jogo.

- **jogaNaEquipa:**

Esta propriedade apresenta domínio em *Jogador* e contradomínio na *Equipa*.

- **temJogador:**

Propriedade inversa da anterior.

- **pertence:**

Esta propriedade tem domínio em *Classificacao* e contradomínio em *Competicao*.

- **temTreinador:**

Esta propriedade apresenta domínio em *Equipa* e contradomínio em *Treinador*.

- **treina:**

Propriedade inversa da anterior

2.4 Data Properties

Depois da análise da API de Dados e depois de escolher classes e *object properties*, escolhemos que informação devia ser armazenada, daí as *data properties* escolhidas e mostradas a seguir.



Figura 2.3: Data Properties

Capítulo 3

Script

Neste capítulo vai ser explicado de forma detalhada o *Script NodeJS* implementado.

O *Script* tem como funções fazer os pedidos a API e converter os dados em formato *JSON* para a sintaxe *RDF Turtle*, obtendo como resultado final os indivíduos da ontologia criada.

Para realizar os pedidos foi utilizado o módulo **request-promise** do npm, teve de ser utilizado ainda o módulo *sleep* devido a contrariedade de limites de pedidos a API.

De modo a converter os dados todos de forma correta teve de ser contrariada a falta de sincronismo do o *NodeJS* daí o uso do módulo de *request-promise*, que como o nome indica assenta o seu funcionamento em *promises*, impedindo assim que sejam executadas coisas assincronamente.

3.1 Pedidos Efetuados

Para obter as informações necessárias foram realizados os seguintes pedidos:

- <http://api.football-data.org/v2/competitions>;
- <http://api.football-data.org/v2/competitions/:idCompeticao/teams>;
- <http://api.football-data.org/v2/competitions/:idCompeticao/matches>;
- <http://api.football-data.org/v2/competitions/:idCompeticao/standings>;
- <http://api.football-data.org/v2/teams/:idEquipa>;

3.2 Sleep

Para combater a adversidade criada pelo limite de 10 pedidos por minuto foi feito um *sleep* de 6 segundos a cada pedido, tornando possível obter toda a informação pedida

Capítulo 4

Aplicação

Posta a fase da modelação e criação de indivíduos, faltava criar uma API de Dados capaz de interrogar o *graphDB* e obter os dados pretendidos para interface.

4.1 API de Dados

A API de dados é composta por *controllers* que executam *queries sparql*, e rotas específicas que usam esses *controllers*.

4.1.1 Controllers

De modo a ter resultados prontos a mostrar na interface, temos de realizar uma transformação no output do *GraphDB*, para tal foram criadas as seguintes funções:

```
1 normalize = function(response) {  
2     return response.results.bindings.map(obj =>  
3         Object.entries(obj)  
4             .reduce((new_obj, [k,v]) => (new_obj[k] = v.value, new_obj),  
5                 new Object()));  
6 };
```

```

1  async function execQuery(q){
2  try{
3      var encoded = encodeURIComponent(q);
4      response = await axios.get("http://localhost:7200/repositories/futebol
      " + '?query=' + encoded);
5      return(normalize(response.data));
6  }
7  catch(error) {
8      return('ERRO: ' + error)
9  }
10 }

```

A função *execQuery* faz um pedido ao repositório do *GraphDB* e executa a *querie* recebida como parâmetro, o resultado é depois transformado pela função *normalize*, removendo campos desnecessários da resposta do *GraphDB*.

Depois de criadas estas funções começaram-se a criar *queries* com base na informação pretendida, como por exemplo obter informação acerca de uma competição, como se pode ver no exemplo a seguir.

```

1  Futebol.infoCompeticao = async (id) =>{
2      const query = `PREFIX : <http://prc.di.uminho.pt/2019/futebol#>
3      select ?nome ?area where {
4          :id:nome?nome.{id} :area ?area.
5      }`
6
7      var res = await execQuery(query)
8      return res
9  }

```

4.1.2 Rotas

Com os *controllers* criados faltava definir as rotas que chamavam esses *controllers* e enviavam os dados obtidos, para este propósito foram criadas as seguintes rotas:

- **/api/competicoes:**
Obtêm todas as competições existentes.
- **/api/competicoes/:id:**
Obtm informação acerca de uma competição.
- **/api/competicoes/:id/equipas:**
Obtêm todas as equipas de uma competição.
- **/api/competicoes/:id/classificacoes:**
Obtêm a classificação de uma competição.
- **/api/competicoes/:id/jogos:**
Obtêm todos os jogos de uma competição.
- **/api/jogos:**
Obtêm todos os jogos.
- **/api/jogos/:id:**
Obtêm informação de um jogo.
- **/api/jogos/:id/arbitros:**
Obtêm todos os arbitros de um jogo.
- **/api/equipas:**
Obtêm todos as equipas.
- **/api/equipas/:id:**
Obtêm informação acerca de uma equipa.
- **/api/equipas/:id/jogadores:**
Obtêm todos os jogadores de uma equipa.
- **/api/equipas/:id/treinadores:**
Obtêm todos os treinadores de uma equipa.

- **/api/equipas/:id/jogos:**
Obtêm todos os jogos de uma equipa.
- **/api/jogadores:**
Obtêm todos os jogadores.
- **/api/jogadores/:id:**
Obtêm informação acerca de um jogador
- **/api/treinadores/principais:**
Obtêm todos os treinadores da classe *Principal*.
- **/api/treinadores/assistentes:**
Obtêm todos os treinadores da classe *Assistente*.
- **/api/treinadores/interinos:**
Obtêm todos os treinadores da classe *Interino*.
- **/api/treinadores/:id:**
Obtêm informação acerca de um treinador.
- **/api/arbitros:**
Obtêm todos os arbitros.
- **/api/arbitros/:id:**
Obtêm informação acerca de um arbitro.
- **/api/arbitros/:id/jogos:**
Obtêm todos os jogos que um arbitro participou.
- **/api/counts/competicoes:**
Obtêm número total de competições.
- **/api/counts/equipas:**
Obtêm número total de equipas.
- **/api/counts/jogos:**
Obtêm número total de jogos.
- **/api/counts/jogadores:**
Obtêm número total de jogadores.

- **/api/counts/treinadores:**
Obtêm número total de treinadores.
- **/api/counts/arbitros:**
Obtêm número total de árbitros.

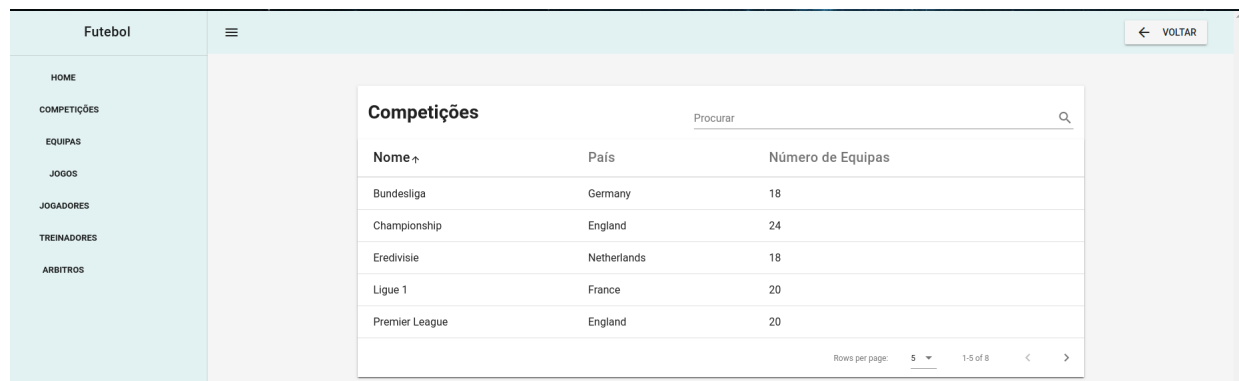
4.2 Interface

A Interface criada foi desenvolvida em *Vue.js*, de seguida irá ser mostrados alguns exemplos de páginas criadas.



Número de Competições	Número de Equipas	Número de Jogos	Número de Jogadores	Número de Treinadores	Número de Arbitros
8	158	3001	5140	210	821

Figura 4.1: Página Inicial



Nome ↕	País	Número de Equipas
Bundesliga	Germany	18
Championship	England	24
Eredivisie	Netherlands	18
Ligue 1	France	20
Premier League	England	20

Figura 4.2: Lista de Competições

Futebol

HOME

COMPETIÇÕES

EQUIPAS

JOGOS

JOGADORES

TREINADORES


ARBITROS

VOLTAR

Bundesliga

Pais

Germany



BUNDESLIGA

Classificações

Procurar

Posição

Nome

Pontos

Jogos

Vitórias

Empates

Derrotas

GM

1	FC Bayern München	78	34	24	6	4	88
2	BV Borussia 09 Dortmund	76	34	23	7	4	81
3	RB Leipzig	66	34	19	9	6	63
4	Bayer 04 Leverkusen	58	34	18	4	12	69
5	Borussia Mönchengladbach	55	34	16	7	11	55

Rows per page:

5

1-5 of 18

<

>

Figura 4.3: Parte da Página Competição

Capítulo 5

Conclusões

Dado por terminado o trabalho efetuado, é possível afirmar que o projeto cumpre todos os requisitos propostos.

As principais dificuldades encontradas ao longo do desenvolvimento do projeto, foram em primeiro lugar, a na modelação da ontologia de modo a estar correta para as fases posteriores. Em segundo lugar, a a conversão para *RDF* devido ao sincronismo que era preciso e também devido as restrições impostas pela API. Por último, a ambientação a interfaces desenvolvidas em *Vue.js*.

Como trabalho futuro poderia ser melhorada a interface, visto estar algo simples em relação ao que a ferramenta *Vue* permite, visto ser uma ferramenta muito poderosa e que precisa de uma exploração mais aprofundada.

Capítulo 6

Anexos

6.1 Script

```
1 var request = require('request-promise')
2 var sleep = require('sleep');
3
4
5 var equipas = []
6 var jogos = []
7 var arbitros = []
8 var classificacoes = []
9 var jogadores = []
10 var competicoes = []
11 var treinadores = []
12
13
14 request.get({
15   url: 'http://api.football-data.org/v2/competitions',
16   headers: {
17     'X-Auth-Token': '6f0a8db07e444773a5241eaad52f3e3e',
18     'Content-Type': 'application/json'
19   }
20 }).then(async dados1 => {
21   var obj1 = JSON.parse(dados1)
22   competicoes = obj1.competitions
23   var competicao = ""
24   await sleep.sleep(6)
```

```

25 console.log("\n### Competi es ###\n")
26 for (var i = 0; i < competicoes.length; i++) {
27     competicao = ""
28     if ((competicoes[i].id == 2002) || (competicoes[i].id == 2003) || (
        competicoes[i].id == 2014) || (competicoes[i].id == 2015) ||(
        competicoes[i].id == 2016) || (competicoes[i].id == 2017) || (
        competicoes[i].id == 2019) || (competicoes[i].id == 2021)) {
29         competicao += ":c_" + competicoes[i].id + " a owl:NamedIndividual
        ,\n"
30         competicao += "\t\t :Competicao ;\n"
31         competicao += "\t:area \"" + competicoes[i].area.name + "\";\n"
32         var name = competicoes[i].name
33         await request.get({
34             url: 'http://api.football-data.org/v2/competitions/' +
                competicoes[i].id + '/teams',
35             headers: {
36                 'X-Auth-Token': '6f0a8db07e444773a5241eaa52f3e3e',
37                 'Content-Type': 'application/json'
38             }
39         }).then(async dados2 => {
40             await sleep.sleep(6)
41             var obj2 = JSON.parse(dados2)
42             var teams = obj2.teams
43             for (var t = 0; t < teams.length; t++) {
44                 var equipa = {
45                     id: teams[t].id,
46                     anoFundacao: teams[t].founded,
47                     area: teams[t].area.name,
48                     coresClube: teams[t].clubColors,
49                     email: teams[t].email,
50                     crestUrl: teams[t].crestUrl,
51                     estadio: teams[t].venue,
52                     morada: teams[t].address,
53                     nome: teams[t].name,
54                     nomeCurto: teams[t].shortName,
55                     tla: teams[t].tla,
56                     website: teams[t].website
57                 }
58                 if (equipas.indexOf(equipa) == -1) {
59                     equipas.push(equipa)

```

```

60         }
61         competicao += "\t:eCompostaPor :e_" + teams[t].id + ";\n"
62     }
63 }).catch(e2 => console.log('Erro2: ' + e2))
64 await request.get({
65     url: 'http://api.football-data.org/v2/competitions/' +
66         competicoes[i].id + '/matches',
67     headers: {
68         'X-Auth-Token': '6f0a8db07e444773a5241eaad52f3e3e',
69         'Content-Type': 'application/json'
70     }
71 }).then(async dados3 => {
72     await sleep.sleep(6)
73     var obj3 = JSON.parse(dados3)
74     var matches = obj3.matches
75     for (var m = 0; m < matches.length; m++) {
76         var match = {
77             id: matches[m].id,
78             jogaEmCasa: matches[m].homeTeam.id,
79             jogaFora: matches[m].awayTeam.id,
80             data: matches[m].utcDate,
81             golosCasa: matches[m].score.fullTime.homeTeam,
82             golosFora: matches[m].score.fullTime.awayTeam,
83             numeroDeJogo: matches[m].matchday,
84             arbitros: matches[m].referees,
85             vencedor: matches[m].score.winner
86         }
87         if (match.numeroDeJogo == null) {
88             match.numeroDeJogo = -1
89         }
90         for (var a = 0; a < matches[m].referees.length; a++) {
91             var referee = {
92                 id: matches[m].referees[a].id,
93                 nome: matches[m].referees[a].name
94             }
95             if (arbitros.indexOf(referee) == -1) {
96                 arbitros.push(referee)
97             }
98         }
99     }
100     if (jogos.indexOf(match) == -1) {

```

```

99         jogos.push(match)
100     }
101     competicao += "\t:eConstituida :j_" + matches[m].id + ";\n"
102     }
103     competicao += "\t:nome \"" + name + "\".\n"
104 }).catch(e3 => console.log('Erro3: ' + e3))
105 console.log(competicao)
106 await request.get({
107     url: 'http://api.football-data.org/v2/competitions/' +
108         competicoes[i].id + '/standings',
109     headers: {
110         'X-Auth-Token': '6f0a8db07e444773a5241eaad52f3e3e',
111         'Content-Type': 'application/json'
112     }
113 }).then(async dados4 => {
114     await sleep.sleep(6)
115     var obj4 = JSON.parse(dados4)
116     var standings = obj4.standings[0].table
117     for (var s = 0; s < standings.length; s++) {
118         var id = standings[s].team.id
119         var standing = {
120             id: id,
121             pertence: obj4.competition.id,
122             equipa: standings[s].team.id,
123             pontos: standings[s].points,
124             posicaoTabela: standings[s].position,
125             nJogos: standings[s].playedGames,
126             nVitorias: standings[s].won,
127             nEmpates: standings[s].draw,
128             nDerrotas: standings[s].lost,
129             diferencaDeGolos: standings[s].goalDifference,
130             nGolosMarcados: standings[s].goalsFor,
131             nGolosSofridos: standings[s].goalsAgainst
132         }
133         if (classificacoes.indexOf(standing) === -1) {
134             classificacoes.push(standing)
135         }
136     }).catch(e4 => console.log('Erro4: ' + e4))

```

```

137
138     }
139
140 }
141 var uniq = {}
142 arbitros = arbitros.filter(obj => !uniq[obj.id] && (uniq[obj.id] = true));
143 for (var e = 0; e < equipas.length; e++) {
144     await request.get({
145         url: 'http://api.football-data.org/v2/teams/' + equipas[e].id,
146         headers: {
147             'X-Auth-Token': '6f0a8db07e444773a5241eaad52f3e3e',
148             'Content-Type': 'application/json'
149         }
150     }).then(async dados5 => {
151         await sleep.sleep(6)
152         var obj5 = JSON.parse(dados5)
153         var players = obj5.squad
154         for (var p = 0; p < players.length; p++) {
155             if (players[p].role === 'PLAYER') {
156                 var eq = []
157                 eq.push(equipas[e].id)
158                 var player = {
159                     id: players[p].id,
160                     jogaNaEquipa: eq,
161                     dataNascimento: players[p].dateOfBirth,
162                     nacionalidade: players[p].nationality,
163                     paisNascimento: players[p].countryOfBirth,
164                     nome: players[p].name,
165                     posicao: players[p].position
166                 }
167                 if (jogadores.indexOf(player) === -1) {
168                     jogadores.push(player)
169                 }
170                 else {
171                     jogadores[player].jogaNaEquipa.push(equipas[e].id)
172                 }
173             }
174             else {
175                 if (players[p].role === 'COACH') {
176                     var coach = {

```

```

177         id: players[p].id,
178         treina: equipas[e].id,
179         dataNascimento: players[p].dateOfBirth,
180         nacionalidade: players[p].nationality,
181         paisNascimento: players[p].countryOfBirth,
182         nome: players[p].name,
183         papel: players[p].role
184     }
185     if (treinadores.indexOf(coach) == -1) {
186         treinadores.push(coach)
187     }
188 }
189 else{
190     if(players[p].role == 'ASSISTANT.COACH'){
191         var coach = {
192             id: players[p].id,
193             treina: equipas[e].id,
194             dataNascimento: players[p].dateOfBirth,
195             nacionalidade: players[p].nationality,
196             paisNascimento: players[p].countryOfBirth,
197             nome: players[p].name,
198             papel: players[p].role
199         }
200         if (treinadores.indexOf(coach) == -1) {
201             treinadores.push(coach)
202         }
203     }
204     else{
205         if(players[p].role == 'INTERIM.COACH'){
206             var coach = {
207                 id: players[p].id,
208                 treina: equipas[e].id,
209                 dataNascimento: players[p].dateOfBirth,
210                 nacionalidade: players[p].nationality,
211                 paisNascimento: players[p].countryOfBirth,
212                 nome: players[p].name,
213                 papel: players[p].role
214             }
215             if (treinadores.indexOf(coach) == -1) {
216                 treinadores.push(coach)

```

```

217         }
218     }
219 }
220 }
221 }
222 }
223 }).catch(e5 => console.log('Erro5: ' + e5))
224
225 }
226 console.log("\n### Equipas ###\n")
227 for (var e = 0; e < equipas.length; e++) {
228     var equipa = ""
229     equipa += ":e_" + equipas[e].id + " a owl:NamedIndividual,\n"
230     equipa += "\t\t:Equipa;\n"
231     if(equipas[e].anoFundacao == null){
232         equipa += "\t:anoFundacao " + 1889 + ";\n"
233     }
234     else{
235         equipa += "\t:anoFundacao " + equipas[e].anoFundacao + ";\n"
236     }
237     equipa += "\t:area \"" + equipas[e].area + "\";\n"
238     equipa += "\t:coresClube \"" + equipas[e].coresClube + "\";\n"
239     equipa += "\t:email \"" + equipas[e].email + "\";\n"
240     equipa += "\t:nome \"" + equipas[e].nome + "\";\n"
241     equipa += "\t:website \"" + equipas[e].website + "\";\n"
242     equipa += "\t:nomeCurto \"" + equipas[e].nomeCurto + "\";\n"
243     equipa += "\t:crestUrl \"" + equipas[e].crestUrl + "\";\n"
244     equipa += "\t:estadio \"" + equipas[e].estadio + "\";\n"
245     equipa += "\t:tla \"" + equipas[e].tla + "\".\n"
246     console.log(equipa)
247 }
248 console.log("\n### Jogos ###\n")
249 for (var j = 0; j < jogos.length; j++) {
250     var jogo = ""
251     jogo += ":j_" + jogos[j].id + " a owl:NamedIndividual,\n"
252     jogo += "\t\t:Jogo;\n"
253     jogo += "\t:jogaEmCasa :e_" + jogos[j].jogaEmCasa + ";\n"
254     jogo += "\t:jogaFora :e_" + jogos[j].jogaFora + ";\n"
255     jogo += "\t:data \"" + jogos[j].data + "\";\n"
256     if (jogos[j].golosCasa == null) {

```



```

257         jogo += "\t:golesCasa  -1;\n"
258     }
259     else {
260         jogo += "\t:golesCasa  " + jogos[j].golesCasa + ";\n"
261     }
262     if (jogos[j].golesFora == null) {
263         jogo += "\t:golesFora  -1;\n"
264     }
265     else {
266         jogo += "\t:golesFora  " + jogos[j].golesFora + ";\n"
267     }
268
269     jogo += "\t:numeroDeJogo  " + jogos[j].numeroDeJogo + ";\n"
270     for (var a = 0; a < jogos[j].arbitros.length; a++) {
271         jogo += "\t:eArbitrado  :a_" + jogos[j].arbitros[a].id + ";\n"
272     }
273     jogo += "\t:vencedor  \'" + jogos[j].vencedor + "\'.\\n"
274     console.log(jogo)
275 }
276
277 console.log("\n#### Arbitros ####\n")
278 for (var a = 0; a < arbitros.length; a++) {
279     arbitro = ""
280     arbitro += ":a_" + arbitros[a].id + " a owl:NamedIndividual,\n"
281     arbitro += "\t\t:Arbitro;\n"
282     arbitro += "\t:nome  \'" + arbitros[a].nome + "\'.\\n"
283     console.log(arbitro)
284 }
285
286 console.log("\n#### Classifica es ####\n")
287 for (var c = 0; c < classificacoes.length; c++) {
288     var classificacao = ""
289     classificacao += ":cl_" + classificacoes[c].id + " a owl:
        NamedIndividual,\n"
290     classificacao += "\t\t:Classificacao;\n"
291     classificacao += "\t:pertence  :c_" + classificacoes[c].pertence + ";\n
        "
292     classificacao += "\t:dizRespeito  :e_" + classificacoes[c].equipa + ";\n
        n"
293     classificacao += "\t:nDerrotas  " + classificacoes[c].nDerrotas + ";\n"

```

```

294     classificacao += "\t:nVitorias " + classificacoes[c].nVitorias + ";\n"
295     classificacao += "\t:nEmpates " + classificacoes[c].nEmpates + ";\n"
296     classificacao += "\t:nGolosSofridos " + classificacoes[c].
        nGolosSofridos + ";\n"
297     classificacao += "\t:nGolosMarcados " + classificacoes[c].
        nGolosMarcados + ";\n"
298     classificacao += "\t:nJogos " + classificacoes[c].nJogos + ";\n"
299     classificacao += "\t:diferencaDeGolos " + classificacoes[c].
        diferencaDeGolos + ";\n"
300     classificacao += "\t:pontos " + classificacoes[c].pontos + ";\n"
301     classificacao += "\t:posicaoTabela " + classificacoes[c].posicaoTabela
        + ".\n"
302     console.log(classificacao)
303 }
304 console.log("\n### Jogadores ###\n")
305 for (var p = 0; p < jogadores.length; p++) {
306     var jogador = ""
307     jogador += ":p_" + jogadores[p].id + " a owl:NamedIndividual,\n"
308     jogador += "\t\t :Jogador;\n"
309     for (j = 0; j < jogadores[p].jogaNaEquipa.length; j++) {
310         jogador += "\t:jogaNaEquipa :e_" + jogadores[p].jogaNaEquipa[j] +
            ";\n"
311     }
312     jogador += "\t:dataNascimento \"" + jogadores[p].dataNascimento +
        "\";\n"
313     jogador += "\t:nacionalidade \"" + jogadores[p].nacionalidade + "\";\n"
314     jogador += "\t:paisNascimento \"" + jogadores[p].paisNascimento +
        "\";\n"
315     jogador += "\t:posicao \"" + jogadores[p].posicao + "\";\n"
316     jogador += "\t:nome \"" + jogadores[p].nome + "\".\n"
317     console.log(jogador)
318 }
319 console.log("\n### Treinadores ###\n")
320 for (var t = 0; t < treinadores.length; t++) {
321     var treinador = ""
322     treinador += ":t_" + treinadores[t].id + " a owl:NamedIndividual,\n"
323     if (treinadores[t].papel == 'COACH') {
324         treinador += "\t\t :Principal;\n"
325     } else {

```

```

326         if (treinadores[t].papel == 'INTERIM_COACH') {
327             treinador += "\t\t : Interino;\n"
328         }
329         else {
330             if (treinadores[t].papel == 'ASSISTANT_COACH') {
331                 treinador += "\t\t : Assistente;\n"
332             }
333         }
334     }
335     treinador += "\t:treina :e_" + treinadores[t].treina + ";\n"
336     treinador += "\t:papel \"" + treinadores[t].papel + "\";\n"
337     treinador += "\t:dataNascimento \"" + treinadores[t].dataNascimento +
338         "\";\n"
339     treinador += "\t:nacionalidade \"" + treinadores[t].nacionalidade +
340         "\";\n"
341     treinador += "\t:paisNascimento \"" + treinadores[t].paisNascimento +
342         "\";\n"
343     treinador += "\t:nome \"" + treinadores[t].nome + "\".\n"
344     console.log(treinador)
345 }
346 }).catch(e1 => console.log('Erro: ' + e1))

```