Universidade Federal de Pelotas
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Ciência da Computação

Disciplina de Web Semântica e Ontologias

Wildlife Quiz

Integrantes

Erick Moreira
Leonardo Rodrigues
Matheus Neiverth
Pedro Barros

Pelotas, 2018

1. Introdução

O projeto visa trazer a partir de uma ontologia informações sobre animais para crianças, de modo que um *software* busque fotos de animais e pergunte a criança qual o nome de um certo animal.

A partir da base de conhecimento *DBPedia*[1] foi desenvolvido uma solução *Web* que automaticamente faz uma busca na ontologia sobre animais. Essa busca é enviada a interface *Web*, e o sistema seleciona em tempo de execução 3 fotos de 3 animais distintos criando assim 3 opções de respostas, e pede ao usuário para indicar entre as 3 opções qual é o animal correto. Ao receber a resposta do usuário o sistema avalia se é a resposta correta, e se for a correta, uma sub-seção é mostrada logo abaixo da resposta, com uma breve descrição do animal para a criança conhece-lo. Demais informações sobre o animal podem ser encontradas em um *link* "ver mais", que leva a página do animal na *Wikipedia*.

Detalhes de informações sobre a *query* enviada a *DBpedia* e as tecnologias de desenvolvimento da aplicação, estão detalhadas nas subseções logo abaixo.

1.1 *DBPedia* e a *Query*

A base de conhecimento empregada na aplicação é a *DBPedia*. Através dela uma *query* de consulta é enviada via método *Get*, com parâmetros para receber a resposta da *query* em *JSON*.

A *DBpedia* tem como objetivo extrair conteúdo de forma estruturada da *Wikipedia*. Essas informações são armazenadas em uma base de conhecimento, na forma *Open Linked Data*. É possível enviar consultas a essa base de dados a partir de uma *API*.

A *query* enviada, figura abaixo, seleciona animais de certas classes específicas, para tentar filtrar os animais não muito conhecidos ou amigáveis, e traz junto algumas informações que precisamos, como por exemplo, a *thumbnail*, o link para a *Wikipedia* e um resumo do recurso pesquisado.

.

```
select distinct *
where {
?animal dbo:kingdom dbr:Animal;
        dbo:phylum ?filo ;
        dbo:class ?classe;
        foaf:name ?nome ;
        rdfs:label ?label;
        dbo:abstract ?abstract;
        rdfs:comment ?comment;
        foaf:isPrimaryTopicOf ?link_wikipedia;
        dbo:thumbnail ?thumbnail

filter(lang(?label) = "pt" && lang(?abstract) = "pt" && lang(?comment) = "pt")
filter( ?classe = dbr:Chondrichthyes || ?classe = dbr:Actinopterygii || ?classe = dbr:Osteichthyes ||
        ?classe = dbr:Amphibia || ?classe = dbr:Sauropsida || ?classe = dbr:Reptilia ||
        ?classe = dbr:Aves || ?classe = dbr:Mammalia )

 } order by ?classe'
```

Figura 1. Código *SPARQL* enviado ao *endpoint* da *DBpedia*.

1.2 *NodeJS* e o Desenvolvimento da Aplicação

Para o desenvolvimento da aplicação foi utilizado a ferramenta NodeJS[2]. *NodeJS* é um Framework em JavaScript, que faz uso do interpretador *V8 JavaScript Engine*[3], desenvolvido pelo *Google* em *C++* para o *Google Chrome*. A ideia por trás do *NodeJS* é permitir um rápido desenvolvimento de aplicações Web com alta escalabilidade.

Nosso *software* utiliza uma biblioteca para esta ferramenta que nos fornece a capacidade de realizarmos pesquisas à base de dados da *DBPedia* através de uma *query SPARQL[4]*. Apenas escrevemos uma *string* com a *query* desejada e a biblioteca nos retorna os resultados em um arquivo do tipo *json*, que é convertido em um objeto em memória do tipo *Hash*. Estes dados, então, são manipulados e exibidos ao usuário através da aplicação.

2. Desenvolvimento do Projeto

O projeto durante seu desenvolvimento passou por um momento delicado: a troca da ontologia da *BBC Wildlife*[5] pelo *endpoint* da *DBpedia*. Gostaríamos de fazer o projeto usando a ontologia da BBC porém dois problemas surgiram - a *BBC* não disponibiliza mais a ontologia *Wildlife* de forma pública, e por segundo, encontramos uma cópia da ontologia *WIldlife* em um *github*[6]. Ao testar essa ontologia usando o *Protegé*[7], descobrimos também que ela usa uma quantidade relativamente alta de memória e em consultas não simplistas, muitas vezes o programa trava. Essas foram as motivações para a troca da ontologia da *BBC* pela *DBpedia* - o que nos gera outros problemas descritos nas próximas sessões.

No projeto usando a *DBpedia*, passamos por essas barreiras, pois existem muitos dados disponíveis e acessíveis facilmente por uma consulta *SPARQL*. Sem a necessidade de termos os dados localmente, apenas enviamos uma solicitação e recebemos a resposta. Assim, escrevemos a consulta ilustrada na figura 1 e conseguimos acesso aos dados facilmente.

Optamos por utilizar implementar um servidor *Web* utilizando *NodeJS* pela facilidade que a ferramenta oferece para prototipar aplicações e também, por encontrarmos a biblioteca *dbpedia-sparql-client*[8] que facilitava a consulta à base de dados da *DBPedia*. Com a utilização desta biblioteca, conseguimos acesso aos dados e partimos para a exibição dos dados para o usuário e a formatação do jogo.

Dos dados recebidos da pesquisa, selecionamos aleatoriamente três animais que serão utilizados como objeto de pergunta ao usuário e mostramos as imagens coletadas da base de conhecimento. Destes 3 animais, um é sorteado para ser o objeto em foco. Dele pegamos o campo *label* e o campo nome da base de dados e mostramos ao jogador, com o objetivo que o mesmo ligue o nome (e o *label)* a uma das três imagens, clicando no botão no respectivo *card* - funcionalidade de responder ao *quiz*. Se o jogador falha em responder a questão corretamente, ele pode continuar tentando até escolher a certa, para adquirir o conhecimento, mas, no entanto, com pontuação menor de quem acertou em menos tentativas.

Ao acertar a questão, o usuário recebe um resumo e um comentário sobre o animal em questão, além de um *link* para o artigo da *Wikipedia* com mais informações sobre o recurso. A aplicação então fornece uma opção para jogar novamente. Mais detalhes na sessão de testes.

Para construir o *layout* da página, utilizamos o *Material Design Lite[7]*, uma biblioteca de autoria do *Google* para o *front-end*, para facilitar alguns estilos, como o dos *cards e o header*.

Ainda durante a implementação, foram encontrados problemas relacionados a consistência dos dados da *DBpedia* - que mais a frente estão melhor documentados. Inclusive, a consulta SPARQL foi editada algumas vezes para tentar sob a nossa intuição, melhorar a qualidade do *quiz*.

Por último, existe uma funcionalidade não totalmente implementada, que é o *rank* de melhores jogadores. Por padrão definido na consulta em código, buscamos em um banco de dados no *Firebase*[8], os 10 jogadores mais bem pontuados. A parte não implementada dessa funcionalidade é que ao jogar uma vez e querer jogar novamente, o usuário solicita um novo jogo - e nessa solicitação seria necessário passar para o próximo *reload* da página via *JavaScript*, a atual pontuação do jogador, para assim ir de forma cumulativa somando sua pontuação a cada novo jogo. A parte de inserir a pontuação no *Firebase* está pronta e funcional, faltaria receber a pontuação atualizada da página, e não uma pontuação *hardcoded*.

O código do projeto encontra-se no github[10].

3. Testes de Uso

Após desenvolvido o *quiz*, foi possível jogar e obter os resultados de sua execução. Na Figura 1, a tela principal do *quiz*, onde podemos ver a pergunta feita ao jogador e as opções disponíveis a ele. No *header* é possível escolher entre jogar e consultar o *rank* com a pontuação dos jogadores.

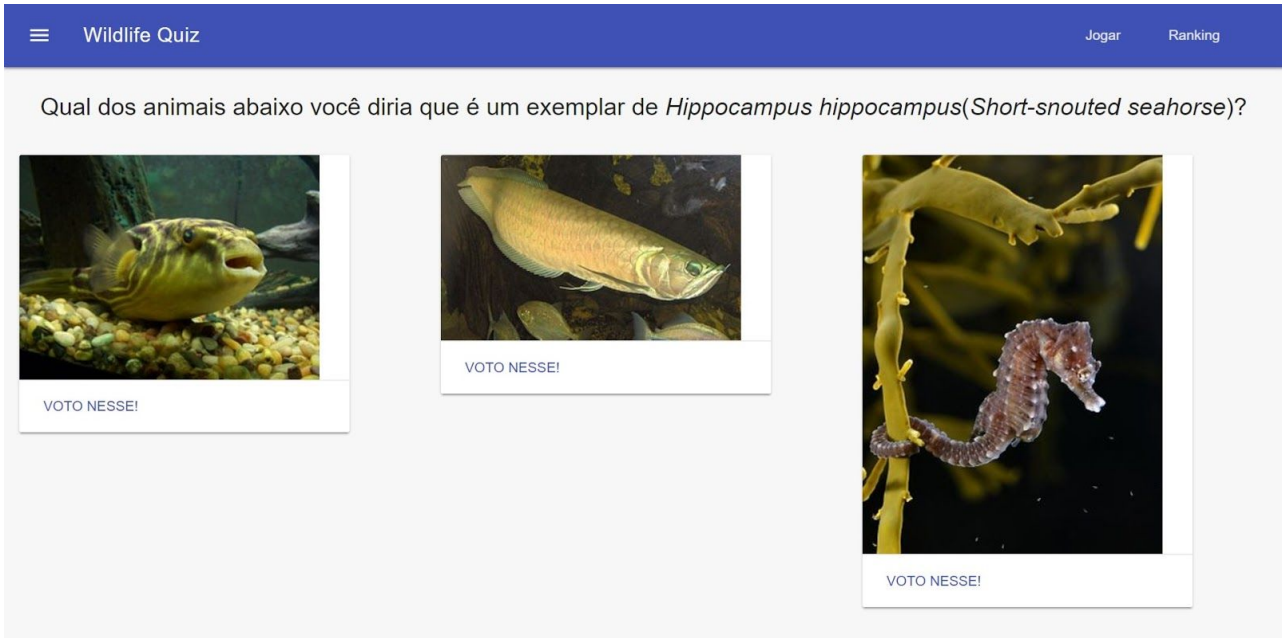

Figura 2. Pergunta e opções do Quiz.

Já na Figura 2, é possível ver um card com o campo *abstract* e *comment* da ontologia, que é mostrado quando o jogador acerta a pergunta proposta no *quiz*. Há também a opção para pedir uma nova pergunta, ou, ir ler mais sobre o animal na página da *Wikipedia*.

Figura 3. *Abstract* e *comments* da ontologia são mostrado após o usuário escolher a opção correta.

Após jogar o *quiz* algumas vezes, notou-se que alguns dados da ontologia são bem inconsistentes. Fotos de animais com nomes diferentes, mas que possuem a mesma foto em comum, campos sem informação, entre outros, são alguns problemas que foram encontrados. Além disso, talvez destacar palavras chaves (critério a definir) no *card* que apresente as informações sobre o animal, poderia atrair mais interesse ao texto do *card*.

4. Considerações Finais

Durante a codificação do trabalho, descobriu-se que em *NodeJS* as chamadas ao banco de dados são feitas de forma assíncrona não bloqueantes, ou seja, durante uma *callback* ao banco de dados o próximo trecho de código logo após a requisição ao banco continua sendo executado, perdendo assim a “noção temporal” que o programador tem no código. Isso pode levar a problemas graves pelo programador, como por exemplo ao tentar retornar a consulta do banco sendo que a consulta ainda não está pronta. Devido a isso, a funcionalidade de *rankear os usuários pela sua pontuação não está funcional - falta computar o somatório de pontos a cada novo jogo.*

Os pontos positivos desse trabalho são a apresentação de imagens de animais e a descrição e comentários após responder ao quiz - as vezes acaba vindo informação pertinente sobre o animal.

Esses pontos positivos podem ser realçados se a *query* for trabalhada de forma *offline*, isto é, tratar os dados para mostrar somente animais com boas fotos por exemplo. (a *DBpedia* possui algumas fotos de péssima qualidade, em alguns casos, e também fotos que tentam representar animais mas que não são de fato do animal). Essa melhoria poderia ser feito na *query* também, no entanto, diminui nosso domínio de resultados e acaba também por excluir bons candidatos ao *quiz* - além de não resolver alguns problemas, como fotos que não são sobre o animal mas que retornam na *query*.

Já os pontos negativos sobre o trabalho, muitos deles se remetem a qualidade das informações que estão na *DBpedia*. Muita informação está repetida, desorganizada, ou não presente - como o nome popular do animal por exemplo. Em alguns casos, a informação filtrada em "*pt-br*" acaba trazendo informação em inglês.

Um outro ponto negativo, esse durante a execução do trabalho em si, foi a mudança de ontologia por falta de capacidade de realizar consultas sobre a primeira ontologia escolhida. O trabalho não pôde ser realizado sobre a ontologia *BBC Wildlife*, porém, utilizando a *DBpedia* foi possível buscar dados semelhantes aos propostos inicialmente, e dar continuidade ao projeto mantendo a mesma ideia.

Para uma versão que poderia ser de fato empregada no mundo real, o tratamento dos dados que retornam da *query* seria essencial para um *quiz* de qualidade. Essa tarefa teria que ser realizada em grande parte na mão (no caso de fotos que não representam o animal), mas a questão de campos nulos oriundos da *query* poderiam ser detectado via *script*, e assim, esse item seria removido do *quiz*.

5. Referências Bibliográficas

1. DBpedia. Disponível em: https://dbpedia.org. Acesso em 16/07/2018.
2. NodeJS. Disponível em: https://nodejs.org/. Acesso em 16/07/2018.
3. v8. Disponível em: https://developers.google.com/v8/. Acesso em 16/07/2018.
4. Dbpedia-sparql-client. Disponível em: https://www.npmjs.com/package/dbpedia-sparql-client. Acesso em 16/07/2018.
5. BBC Ontology. Disponível em: https://www.bbc.co.uk/ontologies/wo. Acesso em 16/07/2018.
6. GITHUB. Disponível em: https://github.com/rdmpage/bbc-wildlife/. Acesso em 16/07/2018.
7. Protegé. Disponível em: https://protege.stanford.edu/. Acesso em 16/07/2018.
8. MDL. Disponível em: https://getmdl.io/. Acesso em 16/07/2018.
9. Firebase. Disponível em: https://firebase.google.com. Acesso em 16/07/2018.
10. Wildlife Quiz. Disponível em: https://github.com/mrneiverth/wso. Acesso em 16/07/2018.