Popcorn - Aplicação XML

Afonso Caldas
ei10051t@fe.up.pt

Rui Monteiro
ei10086@fe.up.pt

Vítor Magano
ei100241@fe.up.pt

9 de Junho de 2014

Resumo

De forma a explorar as capacidades das linguagens de anotação optou-se por implementar uma aplicação que, tendo como base as diversas APIs já existentes como RottenTomatoes e Trackt.tv, utiliza os recursos da base de dados XML nativa eXistdb para oferecer ao utilizador várias funcionalidades como a pesquisa rápida de filmes por vários parâmetros e a sugestão inteligente de filmes.

1 Introdução

O objetivo deste trabalho é o de explorar as linguagens de anotação e as tecnologias associadas e de uma base de dados XML nativa como a *eXistdb* [1]. Assim, devido ao nosso interesse no mundo do cinema e no conhecimento de algumas aplicações existentes relacionadas com esta área, surge a ideia de criar uma aplicação que se apoia em informações disponíveis sobre filmes. A aplicação proposta consiste num website que permite a pesquisa, avaliação e recomendação de filmes. Toda a informação sobre os filmes é obtida a partir das APIs *RottenTomatoes* e *Trackt.tv*. O *RottenTomatoes* e o *track.tv* são duas plataformas *online* com uma massa de utilizadores consideravelmente grande e com uma base de dados de filmes e de séries bastante completa.

Com recurso à API do *Rottentomatoes*, iremos criar o nosso próprio modelo de dados para desenvolver uma aplicação que forneça informações sobre os filmes existentes e faça recomendações baseadas nos gostos e visualizações do utilizador. Além disso, de forma a aumentar a complexidade da aplicação, irá ser feita a ligação com a API da plataforma *Trackt.tv*, o que permite importar desta o historial de visualizações do utilizador caso este possua uma conta. Desta forma é possível simplificar um processo que seria bastante trabalhoso para o utilizador.

Neste relatório irão ser descritos os objetivos do tema proposto, um estudo da arte, os requisitos de utilizador, a arquitetura planeada, o processo de transformação dos dados provenientes das APIs para a base de dados *eXistdb* e os detalhes de implementação da aplicação.

2 Objetivos

Os principais objetivos da aplicação passam por permitir ao utilizador a pesquisa fácil de filmes, através de vários parâmetros como título, sinopse, género, atores envolvidos, data de lançamento e visualizar informações mais detalhadas como sinopse ou *trailer* do filme que pretender.

Irá, também, ter um sistema de sugestões automáticas de filmes que possam ser do interesse utilizador, tendo em conta os filmes que visualizou e avaliou na aplicação. Além disso, será possível ao utilizador integrar a sua lista de filmes vistos e avaliados na plataforma *Track.tv*.

3 Estado da Arte

Existem já, na Web, algumas plataformas que forcenem um serviço com aspectos semelhantes à desta aplicação. De seguida, abordar-se-ão os principais.

3.1 IMDb

A Plataforma online sobre filmes mais popular, com 51 milhões de utilizadores registados. Permite obter todas as informações e notícias sobre filmes, séries e jogos. A pontuação de um filme resulta da votação de 0 a 10 feita pelos utilizadores [2].

Vantagens: possui a base de dados de filmes e séries mais completa de todas as disponibilizadas online.

Desvantagens: a pontuação de filmes recentes normalmente é inflacionada; não disponibiliza uma API pública gratuita.

3.2 Rotten Tomatoes

Semelhante a imdb mas mais focado em críticas profissionais.

Vantagens: credibilidade das críticas feitas aos filmes.

Desvantagens: apenas permite aos utilizadores dizer se gostam ou não de um filme; API apenas permite 10 requests por segundo [3].

3.3 trakt.tv

Uma plataforma social online que permite ao utilizador manter um registo das séries que acompanha e dos filmes que já viu. Ao seguir uma série, o utilizador do trakt vai ser notificado sempre que saiam novos episódios e é-lhe indicado qual o próximo episódio que deve ver dessa série, consoante o seu registo. Incorpora um sistema de comentários e de classificações e oferece ao utilizador a possibilidade de seguir outros utilizadores com um perfil de visualizações semelhante. Faz sugestões de novos filmes ou séries que o utilizador pode ter interesse em ver [4].

Vantagens: listagem dos filmes já vistos; existência de uma watchlist; possibilidade de acompanhar as atualizações dos amigos; partilha de opiniões muitas vezes importante na seleção dos filmes a ver; API flexível [5].

Desvantagens: publicidade no plano gratuito; algumas funcionalidades só disponíveis no plano pago e mais influência em votações de utilizadores que beneficiem deste plano.

3.4 Letterboxd

Rede social de filmes focada nos amantes de cinema. Cria uma comunidade que visa a partilha de opiniões acerca dos filmes e classificação dos mesmos. Permite ainda ao utilizador organizar a sua biblioteca de filmes num conjunto de listas que podem ser criadas conforme a necessidade. Dispõem ainda de funcionalidades extra através do pagamento de uma anuidade, desde importação de dados do imdb até à integração da watchlist com aplicações externas [6].

Vantagens: Semelhante a trackt.tv; sistemas de listas bastante útil.

Desvantagens: impossibilidade de exportação de dados; página inicial com demasiada informação, que pode não interessar a toda a gente, tornando a curva de aprendizagem mais acentuada.

3.5 Suggest Me Movie

Motor de sugestão de filmes, permite aplicar alguns filtros como data de lançamento, género, pontuação no IMDB, e sugere um filme que cumpra esses requisitos [7].

Vantagens: sugestão rápida de filmes ao utilizador.

Desvantagens: apenas mostra um filme de cada vez e filtros são limitados; não tem em conta filmes vistos pelo utilizador.

4 Requisitos de utilizador

Ator	ID	Nome	Descrição
Visitante	UC01	Registo	Criar conta de utilizador
Visitante	UC02	Login	Autenticação na aplicação
Visitante e Utilizador	UC03	Pesquisa	Pesquisa de filmes através de vários filtros
Visitante e Utilizador	UC04	Detalhes do filme	Ver informações de um filme
Utilizador	UC05	Avaliação de filme	Avaliação de um filme segundo uma nota de 1 a 5
Utilizador	UC06	Avaliação de ator	Avaliação de um ator segundo uma nota de 1 a 5
Utilizador	UC07	Avaliação de realizador	Avaliação de um realizador segundo uma nota de 1 a 5
Utilizador	UC08	Sugestão	Recomendação automática de filmes segundo o historial do utilizador
Utilizador	UC09	Watched	Lista de filmes vistos
Utilizador	UC10	Importar avaliações	Importar avaliações de trakt.tv

Tabela 1: *Requisitos de utilizador*

5 Arquitetura

A aplicação é implementada através da *framework NodeJS* pois permite a criação de um servidor de forma simples e existem imensas bibliotecas disponíveis que facilitam a manipulação de documentos XML e a interação entre os componentes da aplicação, incluindo uma desenvolvida pelo próprio autor do *eXistdb*. No

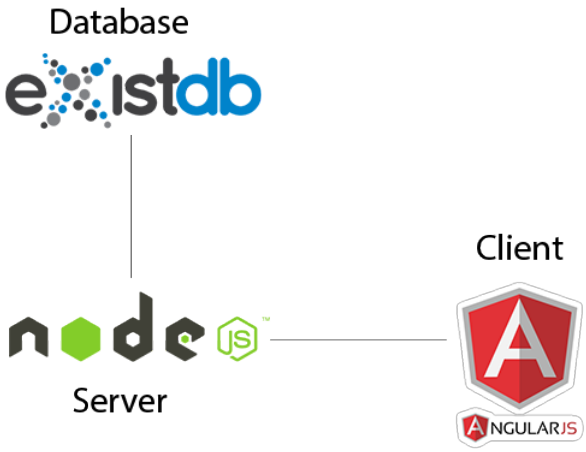

Figura 1: *Arquitetura da aplicação*

lado do cliente é utilizada a *framework AngularJS* que tem elevada compatibilidade com o *NodeJS* e permite a apresentação de vistas dinâmicas a partir dos dados obtidos do servidor. Além disso, esta é também uma oportunidade de explorar estas duas ferramentas que têm vindo a ganhar popularidade. O *eXistdb* foi escolhido por ser uma base de dados XML nativa open source com todas as funcionalidades necessárias para explorar as capacidades das tecnologias XML como *Full-Text Search*, XSLT ou XQUERY.

A comunicação do servidor com a base de dados é feito através da REST API do *eXistdb*. Para fazer *upload* de um recurso XML é utilizado um pedido POST e para o remover é necessário um pedido DELETE. Para fazer *queries* ou alterações nos recursos existentes há três alternativas:

- Pedido GET, utilizando o parâmetro "query=[XPath]"para acrescentar uma condição XPATH
- Pedido POST, com uma expressão em XUPDATE [8] ou em XQUERY
- Pedido GET/POST de uma *stored query*, ou seja, um recurso XQUERY guardado na base de dados, podendo este ser dependente dos parâmetros ou do conteúdo do pedido

6 Detalhes de implementação

6.1 Obtenção dos dados

A informação sobre os filmes é obtida a partir de um processo inicial de obtenção dos dados a partir de APIs externas, a conversão para o formato a ser utilizado pela aplicação e interação com o *eXistdb*. Este processo tem os seguintes passos:

1. Pedido GET de uma listagem de filmes do *RottenTomatoes*. Optou-se por começar pelos 50 filmes mais alugados.
2. Para cada filme é feito um pedido GET para obter todas as informações sobre este.
    - Devido a limitações da API do *RottenTomatoes* a maior parte dos filmes não tem sinopse pelo que foi necessário acrescentar um pedido extra para obter uma a partir da API do *Track.tv* .
    - Este passo poderá ser posteriormente aumentado para adicionar outras informações como *trailers*, *reviews*, etc.
3. Pedido de informações dos filmes similares aos 50 inicialmente obtidos, até chegar aos número de filmes pretendidos.
4. Conversão dos filmes de JSON para XML.
5. Pedido POST para fazer upload de cada filme para a base de dados.
6. Transformação do documento XML através de XSLT para corrigir os erros originados pela conversão e para modificar para o modelo de dados a ser utilizado pela aplicação.
7. Validação do documento XML final através de um XSD Schema.

Todos os documentos utilizados podem ser consultados na pasta "exis"contida na pasta "sources"enviada em conjunto com este relatório.

6.2 Pesquisa de filmes

Para obter uma listagem de filmes é feito um pedido GET para uma *stored query (getMovies.xq)* que pode receber um filtro. É utilizada a seguinte expressão XPATH:

```
$movies//movie[descendant::*/text()[contains(lower-case(.),lower-case($filter))]]
```

para realizar uma *Full-Text Search* e retornar todos os filmes que contenham esse filtro nos seus elementos. Caso não seja definido um filtro são retornados todos os filmes. Além disso, pode também receber um tipo de filtro, dando a opção de fazer a pesquisa *Full-Text* apenas por título, sinopse, ator ou realizador.

Existem também outras pesquisas implementadas como filmes mais votados *(getMostRated.xq)*, filmes com melhor votação média *getBestRated.xq* e melhores filmes por género *(getGenreMovies.xq)*.

6.3 Adicionar utilizador

Além do documento com filmes, é necessário guardar também as informações sobre os utilizadores e as suas avaliações de filmes. Para adicionar um utilizador é também utilizada uma *stored query (addUser.xq)* que adiciona um utilizador se não existir nenhum com o mesmo nome de utilizador.

6.4 Adicionar avaliação

Para guardar uma avaliação feita por um utilizador é utilizada a linguagem XML Update, suportada pelo *eXistdb*, que permite fazer modificações a um documento XML como *append, replace, delete*, etc. É apresentado um exemplo no anexo.

6.5 Sincronização com trakt.tv

Caso o utilzador tenha uma conta no *trakt.tv*, pode importar as avaliações que fez nesse website inserindo os dados de *login*da sua conta. De

forma a reconhecer os filmes a ser importados é utilizado o seu "id"no IMDB *(addTraktRating.xq)*.

6.6 Recomendação de filmes

A recomendação de filmes é feita também tirando partido da API do *trakt.tv*. Caso o utilizador tenha sincronizado a sua conta anteriormente, esta é utilizada para obter recomendações baseadas no seu historial de filmes vistos. Caso este não tenha uma conta no *trakt.tv* é utilizada uma conta própria da aplicação, apagando todos os seus filmes vistos e acrescentando os que o utilizador avaliou na nossa aplicação.

7 Conclusão

A implementação desta aplicação permitiu perceber as potencialidades das tecnologias XML, verificando-se com agrado que foi possível utilizar diversas destas tecnologias como XSD, XSLT, XPATH, XQUERY e XUPDATE. Este processo demonstrou de que forma estas podem ser utilizadas em aplicações WEB. O tema escolhido apresenta elevada complexidade devido à diversidade de APIs a ser utilizadas, com diferentes formatos de dados, e à extensa informação que é necessário manipular para implementar a aplicação proposta inicialmente, no entanto através das tecnologias utilizadas esta tarefa ficou facilitada. Devido à elevada compatibilidade do *AngularJs* com o formato JSON, foi necessário converter o XML recebido dos pedidos feitos à base de dados *eXistdb*, o que procova algum *overhead* e complica a implementação. A utilização de outra *framework* ou inclusão de uma extensão poderia resolver este problema.

Referências

[1] existdb. eXistdb — Documentation. http://exist-db.org/exist/apps/doc/. Acedido em Maio, 2014.

[2] Wikipedia. Wikipedia — IMDb. http://en.wikipedia.org/wiki/Internet_Movie_Database/. Acedido em Março, 2014.

[3] Rotten Tomatoes. Rotten Tomatoes — Api Overview. http://developer.rottentomatoes.com/docs/. Acedido em Março, 2014.

[4] trakt.tv. trakt.tv — About. https://trakt.tv/about/. Acedido em Março, 2014.

[5] trakt.tv. trakt.tv — Api Docs. https://trakt.tv/api-docs/. Acedido em Março, 2014.

[6] Letterboxd. Letterboxd — Frequently Asked Questions. http://letterboxd.com/about/frequent-questions/. Acedido em Março, 2014.

[7] Suggest Me Movie. Suggest Me Movie. http://www.suggestmemovie.com/. Acedido em Março, 2014.

[8] xupdate. xupdate — Project. http://xmldb-org.sourceforge.net/xupdate//. Acedido em Maio, 2014.

8 Anexo

8.1 RottenTomatoes movie list


```
{
  "total": 2,
  "movies": [{
    "id": "770672122",
    "title": "Toy Story 3",
    "year": 2010,
    "mpaa_rating": "G",
    "runtime": 103,
    "critics_consensus": "Deftly
       blending comedy, adventure,
        and honest emotion, Toy
       Story 3 is a rare second
       sequel that really works.",
    "release_dates": {
      "theater": "2010-06-18",
      "dvd": "2010-11-02"
    },
    "ratings": {
      "critics_rating": "Certified
          Fresh",
      "critics_score": 99,
      "audience_rating": "Upright
         ",
      "audience_score": 91
    },
    "synopsis": "Pixar returns to
       their first success with
       Toy Story 3. The movie
       begins with Andy leaving
       for college and donating
       his beloved toys -- 
       including Woody (Tom Hanks)
        and Buzz (Tim Allen) -- to
        a daycare. While the crew
       meets new friends,
       including Ken (Michael
       Keaton), they soon grow to
       hate their new surroundings
        and plan an escape. The
       film was directed by Lee
       Unkrich from a script co-
       authored by Little Miss
       Sunshine scribe Michael
       Arndt. ~ Perry Seibert,
       Rovi",
    "posters": {
      "thumbnail": "http://
         content6.flixster.com/
         movie/11/13/43/11134356
         _mob.jpg",
      "profile": "http://content6.
         flixster.com/movie
         /11/13/43/11134356_pro.
         jpg",
      "detailed": "http://content6
         .flixster.com/movie
         /11/13/43/11134356_det.
         jpg",
      "original": "http://content6
         .flixster.com/movie
         /11/13/43/11134356_ori.
         jpg"
    },
    "abridged_cast": [
      {
        "name": "Tom Hanks",
        "characters": ["Woody"]
      },
      {
        "name": "Tim Allen",
        "characters": ["Buzz
           Lightyear"]
      },
      {
        "name": "Joan Cusack",
        "characters": ["Jessie the
            Cowgirl"]
      },
      {
        "name": "Don Rickles",
        "characters": ["Mr. Potato
            Head"]
      },
      {
        "name": "Wallace Shawn",
        "characters": ["Rex"]
      }
    ],
    "alternate_ids": {"imdb":
       "0435761"},
    "links": {
```

```
      "self": "http://api.
         rottentomatoes.com/api/
         public/v1.0/movies
         /770672122.json",
      "alternate": "http://www.
         rottentomatoes.com/m/
         toy_story_3/",
      "cast": "http://api.
         rottentomatoes.com/api/
         public/v1.0/movies
         /770672122/cast.json",
      "clips": "http://api.
         rottentomatoes.com/api/
         public/v1.0/movies
         /770672122/clips.json",
      "reviews": "http://api.
         rottentomatoes.com/api/
         public/v1.0/movies
         /770672122/reviews.json",
      "similar": "http://api.
         rottentomatoes.com/api/
         public/v1.0/movies
         /770672122/similar.json"
    }
  }],
  "links": {
    "self": "http://api.
       rottentomatoes.com/api/
       public/v1.0/movies.json?q=
       Toy+Story+3&page_limit=1&
       page=1",
    "next": "http://api.
       rottentomatoes.com/api/
       public/v1.0/movies.json?q=
       Toy+Story+3&page_limit=1&
       page=2"
  },
  "link_template": "http://api.
     rottentomatoes.com/api/public
     /v1.0/movies.json?q={search-
     term}&page_limit={results-per
     -page}&page={page-number}"
}
```

8.2 movies.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="movies">
        <movies>
            <xsl:apply-templates select="movie"/>
        </movies>
    </xsl:template>
    <xsl:template match="movie">
        <movie>
            <xsl:apply-templates select="*"/>
        </movie>
    </xsl:template>
    <xsl:template match="genres">
        <genres>
            <xsl:for-each select="movie">
                <genre>
                    <xsl:value-of select="."/>
                </genre>
            </xsl:for-each>
        </genres>
    </xsl:template>
    <xsl:template match="release_dates">
        <release_date>
            <xsl:value-of select="theater"/>
        </release_date>
    </xsl:template>
    <xsl:template match="posters">
        <poster>
            <xsl:value-of select="detailed"/>
        </poster>
    </xsl:template>
    <xsl:template match="abridged_cast">
        <cast>
            <xsl:for-each select="movie">
                <actor>
                    <xsl:copy-of select="name"/>
                    <xsl:copy-of select="id"/>
                    <characters>
                        <xsl:for-each select="characters/movie">
                            <character>
                                <xsl:value-of select="."/>
                            </character>
                        </xsl:for-each>
                    </characters>
                </actor>
            </xsl:for-each>
        </cast>
```

```
    </xsl:template>
    <xsl:template match="abridged_directors">
        <directors>
            <xsl:for-each select="movie">
                <director>
                    <xsl:copy-of select="name"/>
                </director>
            </xsl:for-each>
        </directors>
    </xsl:template>
    <xsl:template match="alternate_ids">
        <xsl:copy-of select="imdb"/>
    </xsl:template>
    <xsl:template match="similar">
        <similar_movies>
            <xsl:for-each select="movie">
                <similar_movie>
                    <xsl:copy-of select="node()"/>
                </similar_movie>
            </xsl:for-each>
        </similar_movies>
    </xsl:template>
    <xsl:template match="*">
        <xsl:copy-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

8.3 movies.xml

```
<movie>
        <id>770672122</id>
        <title>Toy Story 3</title>
        <year>2010</year>
        <genres>
            <genre>Animation</genre>
            <genre>Kids &amp;amp; Family</genre>
            <genre>Science Fiction &amp;amp; Fantasy</genre>
            <genre>Comedy</genre>
        </genres>
        <mpaa_rating>G</mpaa_rating>
        <runtime>103</runtime>
        <critics_consensus>Deftly blending comedy, adventure, and honest emotion, T
        <release_date>2010-06-18</release_date>
        <ratings>
            <critics_rating>Certified Fresh</critics_rating>
            <critics_score>99</critics_score>
            <audience_rating>Upright</audience_rating>
            <audience_score>89</audience_score>
        </ratings>
```

```
<synopsis>Pixar returns to their first success with Toy Story 3. The movie
<poster>http://content6.flixster.com/movie/11/13/43/11134356_det.jpg</poste
<cast>
    <actor>
        <name>Tom Hanks</name>
        <id>162655641</id>
        <characters>
            <character>Woody</character>
        </characters>
    </actor>
    <actor>
        <name>Tim Allen</name>
        <id>162655909</id>
        <characters>
            <character>Buzz Lightyear</character>
        </characters>
    </actor>
    <actor>
        <name>Joan Cusack</name>
        <id>162655020</id>
        <characters>
            <character>Jessie the Cowgirl</character>
        </characters>
    </actor>
    <actor>
        <name>Ned Beatty</name>
        <id>162672460</id>
        <characters>
            <character>Lots-o'-Huggin' Bear</character>
            <character>Lotso</character>
        </characters>
    </actor>
    <actor>
        <name>Don Rickles</name>
        <id>341817905</id>
        <characters>
            <character>Mr. Potato Head</character>
        </characters>
    </actor>
</cast>
<directors>
    <director>
        <name>Lee Unkrich</name>
    </director>
</directors>
<studio>Walt Disney Pictures</studio>
<imdb>0435761</imdb>
<links>
```

```
            <self>http://api.rottentomatoes.com/api/public/v1.0/movies/770672122.js
            <alternate>http://www.rottentomatoes.com/m/toy_story_3/</alternate>
            <cast>http://api.rottentomatoes.com/api/public/v1.0/movies/770672122/c
            <clips>http://api.rottentomatoes.com/api/public/v1.0/movies/770672122/
            <reviews>http://api.rottentomatoes.com/api/public/v1.0/movies/77067212
            <similar>http://api.rottentomatoes.com/api/public/v1.0/movies/77067212
        </links>
        <similar_movies>
            <similar_movie>
                <id>770671912</id>
                <title>Up</title>
                <poster>http://content7.flixster.com/movie/10/89/43/10894361_det.jp
            </similar_movie>
            <similar_movie>
                <id>9414</id>
                <title>Toy Story 2</title>
                <poster>http://content6.flixster.com/movie/10/93/63/10936392_det.jp
            </similar_movie>
            <similar_movie>
                <id>9559</id>
                <title>Toy Story</title>
                <poster>http://content7.flixster.com/movie/10/93/63/10936393_det.jp
            </similar_movie>
            <similar_movie>
                <id>9382</id>
                <title>Monsters, Inc.</title>
                <poster>http://content6.flixster.com/movie/11/16/65/11166524_det.jp
            </similar_movie>
            <similar_movie>
                <id>9749</id>
                <title>A Bug's␣Life</title>
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣<poster>http://content9.flixster.com/movie/10/93/89/10938947_det.jp
␣␣␣␣␣␣␣␣␣␣␣␣</similar_movie>
␣␣␣␣␣␣␣␣</similar_movies>
␣␣␣␣</movie>
```

8.4 users.xml

```
<users>
    <user>
        <name>Teste</name>
        <password>teste</password>
        <ratings>
            <rating>
                <movie>
                    <id>771250003</id>
                </movie>
                <date>10-4-2014</date>
                <grade>4</grade>
            </rating>
            <rating>
                <actor>
                    <id>770800260</id>
                </actor>
                <date>10-4-2014</date>
                <grade>4</grade>
            </rating>
            <rating>
                <movie>
                    <id>771250004</id>
                </movie>
                <date>11-4-2014</date>
                <grade>4</grade>
            </rating>
        </ratings>
    </user>
</users>
```

8.5 Adicionar avaliação (Exemplo)

```
<?xml version="1.0" encoding="UTF-8"?>
<xu:modifications version="1.0" xmlns:xu="http://www.xmldb.org/xupdate">
  <xu:remove select="//user[name="Teste"]//rating[movie[id="771250004"]]"/>
  <xu:append select="//user[name="Teste"]/ratings">
    <xu:element name="rating">
      <movie>
        <id>771250004</id>
      </movie>
      <date>11-4-2014</date>
      <grade>4</grade>
    </xu:element>
  </xu:append>
</xu:modifications>
```