Web Site preparado para a Web Semântica: Semântica de Eventos

João Rodrigues, Diogo Laginha

jarod@student.dei.uc.pt, dmachado@student.dei.uc.pt

**Resumo:** *EventSemantic* é um site de eventos musicais a realizar em Portugal. À semelhança do que a LastFM [2] oferece, esta iniciativa pretende apresentar e transformar a informação de eventos de música nacionais, em dados de um nível superior. Técnicas de recolha de informação e de Web Semântica foram usadas, de modo a concluir este trabalho com sucesso.

**Palavras-chave:** Eventos, Música, Semântica, Portugal

1. Introdução

Este projecto realizado no âmbito da Web Semântica, nasceu de uma lacuna ignorada pelos actuais sites web de eventos nacionais. Estes, para além de possuírem conteúdos limitados, apresentam um modelo de navegação unidimensional pouco eficaz. Ademais, os métodos de pesquisa são deveras elementares e crus, uma vez que a informação disponível contem pouco significado e valor.

*EventSemantic* tem então como objectivo melhorar os serviços actualmente disponíveis. Assim sendo, foi desenvolvido uma aplicação web de pesquisa e navegação de eventos nacionais de música, recorrendo a técnicas de Web Semântica, onde será igualmente incorporado um sistema de sugestões de eventos.

2. Descrição da arquitectura e design do sistema

No nosso sistema, independentemente da página em que o utilizador se encontra, estará sempre visível uma barra superior representando o menu principal e uma caixa de texto associada a um botão onde se pode inserir uma pesquisa semântica.



Figura 1 – Aparência da barra superior do sistema

Como podemos ver na figura 1, a opção *Home* do menu é a que vem seleccionada de raiz. Portanto, sempre que um utilizador acede ao *url* da página, aparece um ecrã de boas-vindas com informações úteis e conselhos sobre a utilização desta. Por outro lado, quando a opção *Contact* é seleccionada, o sistema não faz mais do que mostrar uma página com informações sobre como contactar os colaboradores e programadores da página.

Quando a opção *Events* é escolhida, o utilizador é reencaminhado para a página de *browsing*, onde pode filtrar os eventos consoante determinadas características. Do lado direito estão disponíveis dois menus onde os eventos podem ser filtrados, respectivamente, por categoria e por zona geográfica ao passo que mais a cima, debaixo do menu principal, aparece um sistema de *tabs* que permite a filtragem por data.

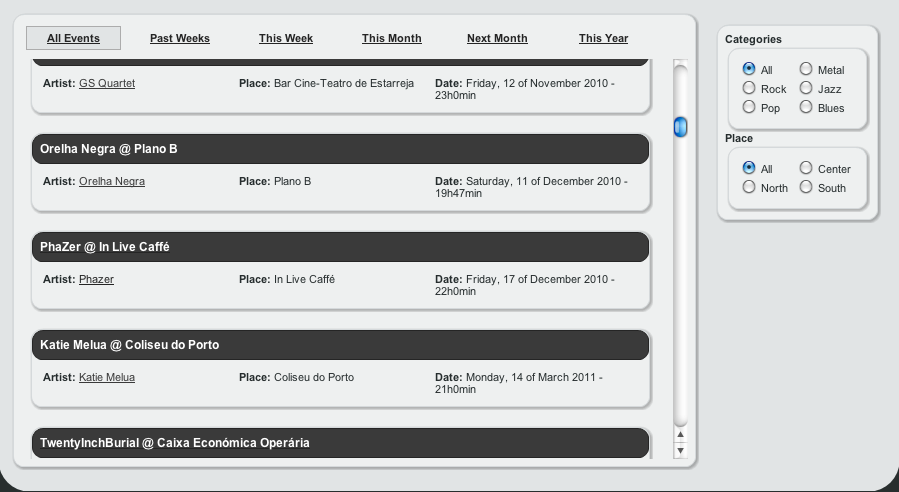


Figura 2 – Aparência dos menus de *browsing* e listagem de eventos por *scroll*

Assim, por cada vez que o utilizador escolher uma determinada característica de um dos menus, a página é automaticamente actualizada mostrando apenas os eventos que partilham as três características que estão actualmente seleccionadas em cada um dos menus de filtragem. Como podemos ver na figura 2, as características que estão seleccionadas de raiz são *todas as datas*, *todas as categorias* e *todas as zonas geográficas*, pelo que todos os eventos da base de dados vão estar listados de inicio.

Tanto para mostrar os eventos resultantes do *browsing* como os resultantes da pesquisa, optou-se por um sistema de *scroll* dentro de um bloco. Esta abordagem faz com que o utilizador possa fazer *scroll* sobre os eventos sem nunca perder de vista tanto os menus de filtragem como a barra do menu principal com caixa de pesquisa.

Como em alguns casos vão ser listados muitos eventos, optou-se por se mostrar apenas as informações mais importantes de cada um na listagem, como o nome, artistas, localização e data, fazendo assim com que a informação ficasse mais legível para o utilizador. Se este, posteriormente, desejar visualizar informação mais detalhada sobre um determinado evento, poderá sempre clicar no seu nome e será redireccionado para a sua página.

Na página de cada evento, para além de ser disponibilizada informação mais detalhada sobre o mesmo, são também disponibilizados *links* para eventos sugeridos pelo sistema na parte direita do ecrã. Além disso, tanto na página de cada evento como na página de listagem de eventos, estão disponíveis *links* para a página de cada artista, onde se encontra informação mais detalhada sobre este, como por exemplo em que eventos participa, que géneros musicais o caracterizam e que álbuns já lançou.

3. Ontologia usada

A nossa aplicação tem por base uma única ontologia (MusicEvents) que está definida no ficheiro Projecto\_WS.owl na pasta Protege e que tem a seguinte estrutura de classes:

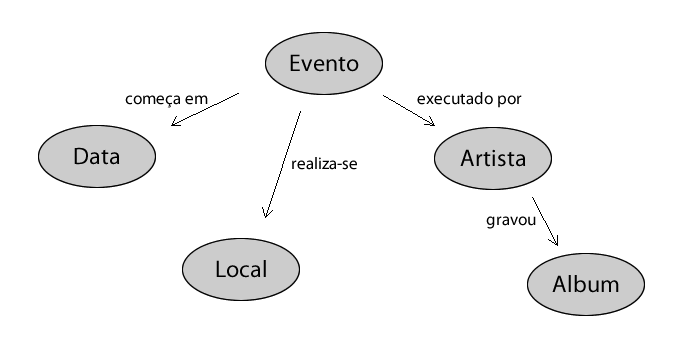


Figura 3 – Estrutura de classes da ontologia utilizada

Quanto às propriedades específicas de cada classe, estas estão definidas na seguinte tabela:

Tabela 1 – Propriedades de cada classe da ontologia utilizada

|  |  |
| --- | --- |
| **Classe** | **Propriedades** |
| *Evento* | nome, descrição |
| *Data* | nome, nome do dia, número do dia, hora, minutos, segundos, nome do mês, número do mês, número da semana, ano, *timestamp* |
| *Local* | Nome, morada, código postal, localidade, país, latitude, longitude |
| *Artista* | nome, género, sumário, descrição |
| *Album* | nome, editora |

4. Descrição dos módulos mais importantes do sistema

4.1. create\_rdf.py

Este módulo, apesar de não fazer parte da aplicação web, é bastante importante na medida em que quando é executado carrega a base de dados com triplos. A sua tarefa é processar toda a informação obtida sobre os vários eventos, que se encontra espalhada por ficheiros que foram construídos previamente através de *APIs* e de técnicas de *screen scrapping*. Essa informação encontra-se em formato *xml* ou *json*, pelo que este módulo vai funcionar como uma espécie de *parser* que simplesmente a interpreta e cria triplos estabelecendo relações entre entidades.

Existem também alguns tipos de informação que não estão disponíveis nos ficheiros mas que este módulo os consegue obter através de processamento, como são o caso do número da semana e do *timestamp* do evento, informações que podem ser obtidas através do dia e da hora a que este ocorre. Destaca-se também neste módulo o uso de expressões regulares de modo a extrair e filtrar um texto descritivo e um sumário que nos é fornecido sobre cada evento, eliminando assim o “lixo” que o *xml* fornecido contém.

4.2. get\_rdf.py

oal

4.3. views.py

5. Estrutura e organização do código

Na pasta *Scripts* poderão ser encontrados todos os *scripts python* que foram implementados de modo a extrair o máximo de informação possível sobre eventos do site <http://www.lastfm.com>. Este foi o site escolhido devido ao facto de ser completo e de possuir uma boa *API* que, apesar de ter alguns métodos interessantes, não nos permitiu retirar toda a informação que desejaríamos do site, daí termos também recorrido a técnicas de *screen scrapping* para recolha de mais informação*.*

Na pasta *event\_semantics* encontram-se todos os ficheiros correspondentes à aplicação web. Esta aplicação foi implementada em *Django* [3], uma *framework* para aplicações web desenvolvida em *python,* e todo o código segue a organização e estrutura convencionais desta.

Portanto, sempre que há um pedido *http* ao servidor, o ficheiro urls.py vai analisar o url e, caso este seja válido, vai accionar a respectiva função de processamento. Resumindo, para cada url válido existe sempre uma função de processamento específica (*view*), que estará implementada no ficheiro views.py.

De seguida, a *view* que foi accionada vai ser executada, devolvendo no final um *template* (um ficheiro *html* com código *Django)* com a informação que foi computada associada a este (ex: o resultado de uma *query*). Este *template* é automaticamente executado sendo produzido no final um ficheiro *xhtml+rdfa* que vai ser devolvido ao utilizador.

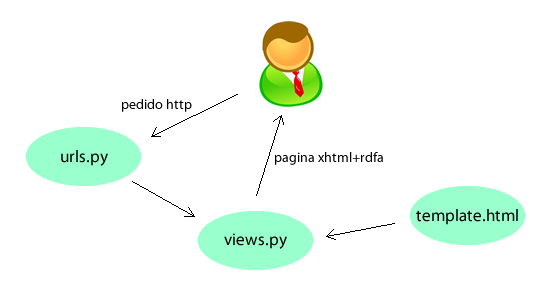


Figura 4 – Estrutura do código da aplicação web

Como todos *templates* vão repetir partes do *html* de outros (ex: menus), optou-se por implementar um ficheiro base.html com todo o *html* que é comum a todos os *templates*. Deste modo cada *template* só vai ter *html* específico e importa da base todo o *html* comum fazendo assim com que o código seja mais limpo.

Este ficheiro base.html, bem como os restantes *templates* e todos os ficheiros de processamento da aplicação web (urls, views, etc) encontram-se dentro da pasta *events* ao passo que a pasta *media* contem todos os ficheiros *javascript*, *css* e imagens.

6. Conclusão

O forato de página deve ser o A4, com as margens superior de 6cm, inferior de 5cm e esquerda e direita de 4,4cm. Referir horas extra para cada um dos passos do trabalho.

7. Referências

O formato de página deve ser o A4, com as margens superior de 6cm, inferior de 5cm e esquerda e direita de 4,4cm.