Aplicación Web para la búsqueda de programación

15/12/2015

Daniel Filgueiras – Jairo Bonanata - Waldemar Lopez – Johanna Milena Amado

Grupo: 19

Facultad de Ingeniería

Universidad de la República Oriental del Uruguay
Docente: Libertad Tansini

Contenido

Introducción 3
Descripción del problema 3
Solución propuesta 3
Implementación 5
Arquitectura propuesta 5
Parsers 6
Configuraciones de Solr 8
Procesamiento para campos de texto 9
SolrNet 10
Configuraciones de relevancia 11
Configuración de la búsqueda 11
Módulo de Consulta 13
Casos Interesantes 16
Manual del Usuario 20
Herramientas y Tecnologías utilizadas 22
Características destacadas de Solr 24
Bibliografía 25

Introducción

El presente trabajo tiene como objetivo contribuir a la búsqueda rápida de diferentes programaciones que nos brindan los proveedores de cable. El mismo se llevó a cabo a través de una aplicación web; la cual proporcionara a los usuarios una información completa del programa de interés, ya sean películas, deportes, documentales, series infantiles, musicales, series de televisión o de variedades.

Descripción del problema

La vida cotidiana de los habitantes de la ciudad de Montevideo no sólo se basa en el trabajo diario sino también en los momentos de esparcimiento. Algunas actividades recreativas hacen parte de ellos, incluyendo la televisión. La ciudad maneja diversos proveedores de cable, sin embargo, algunos de los paquetes del mercado no involucran un acceso fácil y dinámico a la programación. Para ello, el usuario debe ingresar a la página Web y navegar a través de una compleja red de información para acceder a su programación favorita y planear su día de recreación. Esto involucra una cantidad de tiempo innecesaria para el cliente.

Hay que encontrar una respuesta a este inconveniente, de manera que sea posible realizar búsquedas más eficientes y confiables. Es por esta razón que nuestro proyecto optimiza el tiempo de búsqueda proporcionándole al usuario no solo la posibilidad de encontrar en un tiempo muy corto su respuesta sino también la opción de hallar una información más detallada; brindándole una interfaz gráfica agradable, sencilla, fácil de manejar y con todos los detalles actualizados, como por ejemplo los horarios, descripción y en qué canales y contratando qué proveedores podrá ver su programación favorita.

Solución propuesta

El objetivo es proporcionar una aplicación web amigable, intuitiva, sencilla y de fácil uso para el usuario; presentando de forma clara la búsqueda y la presentación de los resultados.

Para que sea más fácil para el usuario se definen distintos parámetros de búsqueda, como buscar por palabras clave dentro del título o la descripción de un programa, así como filtrar por fechas, géneros y proveedores de cable.

Para poder llevar a cabo este proyecto se implementó una aplicación web que busca entradas de programación de los diferentes proveedores de cable, a través de un motor de búsquedas. Para ello es necesario descargar y homogeneizar la información de los proveedores de cable transformándola a un esquema común para poder tratarla uniformemente desde nuestra aplicación. Este esquema común será un archivo XML que arma el manejador con la información recibida de cada

parser[1] (uno por cada proveedor). Luego de definido este esquema común se almacenan e indizan los datos en un motor de búsqueda. Finalmente desde una aplicación se realizan consultas a dicho motor de búsqueda para desplegar los resultados al usuario.

[1] Script o programa que permite descargar y procesar la información de entrada a partir de las páginas web de los proveedores de cable.

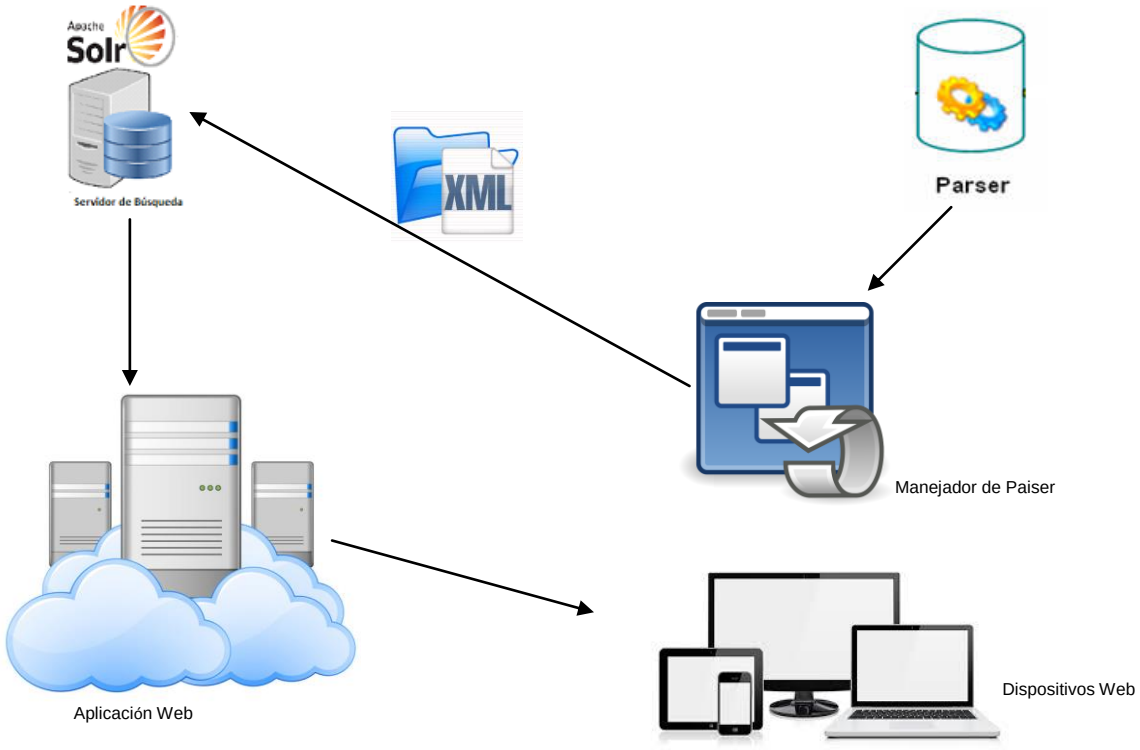
Implementación

El objetivo es proporcionar una aplicación web amigable, intuitiva, sencilla y de fácil uso para nuestros usuarios; presentando de forma clara la búsqueda y la presentación de los resultados. Para esto se realizaron los siguientes pasos:

- Parser para cada uno cada de los proveedores (Nuevo Siglo [1], MonteCable [2] y TCC [3]).
- Manejador que llama a cada parser y con el resultado armar un XML común para ser indexado en Solr [4]. Este manejador funciona como un script que se ejecuta una vez por día para actualizar la programación de los proveedores.
- Almacenamiento e indexado de los documentos recibidos en Solr.
- Una aplicación Web para consultar programación para los usuarios. Esta aplicación consulta a Solr por los programas ingresados por el usuario y devuelve los resultados obtenidos de Solr.
- La aplicación Web permitirá solamente consultas por título, descripción, por género y rangos de fechas.

Arquitectura propuesta

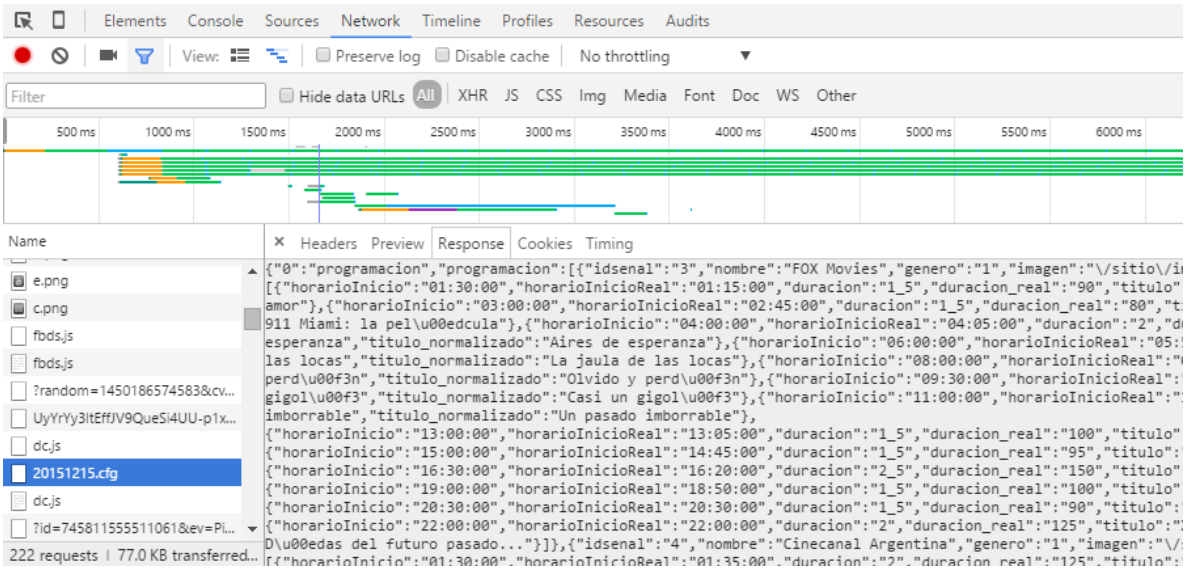
A continuación se presenta la arquitectura que se estableció para la realización de este trabajo.



Parsers

Los parsers se implementaron utilizando Python [5] y las bibliotecas Scrapy [6] y requests [7] provistas para Python.

Los parser para Nuevo Siglo y MonteCable presentaron la dificultad de que los proveedores de cable utilizan contenido dinámico, por lo que seguir los enlaces dentro de las páginas no fue una solución viable. Entonces se analizó el flujo de información que utilizaban las páginas web y se identificaron varias URLs a través de las cuales con una petición HTTP se puede obtener un archivo json que contiene casi toda la información de la programación. Además de esta con una clave que contienen dichos archivos json se accede a la descripción de cada uno de los programas, así como también a las URLs de las imágenes que sirven como entrada para la aplicación de usuario. Para ayudarnos en esta tarea se utilizó la inspección de elementos de Google Chrome [8] cómo se muestra en la imagen:



En esta imagen se ve identificado claramente un archivo en formato json al cual se le hace la petición, en el siguiente ejemplo de código se puede ver la petición para recuperar este archivo json para Nuevo Siglo:

```
headers = {'content-type': 'application/json', 'charset':'utf-8'}
today = datetime.datetime.now()
year = today.year
month = today.month
day = today.day
today =  str(year) + str(month) + str(day)
base_url = 'http://www.nuevosiglo.com.uy/sitio/files/programacion/'
grilla_url = today + '.cfg'
url = base_url + grilla_url
r = requests.get(url, headers=headers)
```

Luego de recibido, el contenido de la petición se procesa con Python que mapea este contenido a diccionario en forma transparente.

Por otro lado el parser para TCC se pudo implementar de una forma más ortodoxa, pues la página web de TCC está basada en contenido HTML estático, por esta razón se utilizó Scrapy. En el siguiente ejemplo de código se muestra cómo Scrapy recibe la URL por la cual comenzar (start_urls) a recorrer el sitio para descargar el contenido. Luego en el método **parse** con una expresión XPATH se va recorriendo toda la información de la página inicial, es decir los canales. Luego se genera un request HTTP para cada uno de estos canales para obtener la información completa de cada canal. Para cada canal se define además “el método callback”, es decir el método que se llama recursivamente para cada uno de los canales. Finalmente para cada canal se procesa cada elemento de la grilla obteniéndose la información detallada y guardándola en un archivo XML.

```
class TccSpider(Spider):

    name = 'tcc'
    allowed_domains = ['tcc.com.uy']
    start_urls = ["http://www.tcc.com.uy/signal-grid/"]
    base_url = "http://www.tcc.com.uy"


    remove_tags_re = re.compile('</?.*?>', re.IGNORECASE | re.UNICODE | re.DOTALL)
    obtener_dia_re = re.compile('[0-9]*?/[0-9]*?/[0-9]*', re.IGNORECASE | re.UNICODE | re.DOTALL)
    obteber_hora_re = re.compile('\d+?:\d+')

    def parse(self, response):
        for link in response.xpath('//ul[@id="menu1"]/li/a'):
            url = self.base_url + link.xpath('./@link').extract()[0]
            # print url
            nombre = link.xpath('./text()').extract()[0]
            # print nombre
            yield scrapy.Request(url, meta={'nombre_canal': nombre}, callback=self.parse_canal)
```

Ejemplo de código de Spider para TCC.

Otro desafío que presentó esta etapa fue la de llevar el género a un formato común, cada proveedor de cable utiliza un código distinto para cada género, por lo que fue necesario identificar dichos códigos y llevarlos a un formato común.

Luego de obtenida la información de la programación se indexan los datos en Solr.

Configuraciones de Solr

Para indexar los documentos se definió un esquema en Solr, este esquema define los campos que tiene cada documento y cómo debe ser tratado el mismo. Cada documento corresponde a un programa específico a una hora dada, en un canal dado y en un proveedor de cable dado.

Los campos que contiene cada documento son los siguientes:

- **Título**: Título del programa.
- **Proveedor**: El proveedor de cable que emite el programa.
- **Canal**: El canal que emite el programa.
- **Género**: Género al que pertenece el programa (deportes, cine, variedades, etc.).
- **Fecha de Inicio**: Fecha y hora a la que comienza la transmisión.
- **Fech de Finalización**: Fecha y hora a la que finaliza la transmisión.
  Descripción: Descripción del programa.
- **URL imagen**: URL auxiliar que sirve para mejorar la experiencia de usuario mostrando una imagen.

Es importante aclarar que Solr maneja cada campo por separado, manteniendo índices separados para los distintos campos.

Hay que destacar que no todos los campos del documento son tratados de la misma forma, por ejemplo el campo *URL imagen* está almacenado en Solr, pero no está indexado, pues no se realizarán búsquedas sobre el mismo. Por otro lado como el resto de los campos son utilizados para filtrar, buscar u ordenar; deben estar indexados.

Los campos indexados son tratados de diversas maneras, por ejemplo los campos *Fecha de Inicio* y *Fecha de Finalización* son tratados como un tipo fecha, el índice utilizado para este tipo nos permite realizar búsquedas por rangos de fechas o buscar por una fecha exacta, se utiliza un índice simple, sin necesidad de un índice invertido para estos campos.

Los campos *Canal, Proveedor* y *Género* son tratados como cadenas de caracteres por los cuales simplemente se filtra.

Procesamiento para campos de texto

Finalmente llegamos a los campos que explotan las funcionalidades de Solr y de tener un índice invertido; los mismos son *Título* y *Descripción.* Estos campos son tratados como campos de texto de Solr, es decir que se utiliza un índice invertido para los mismos, para ello como se algunas de las herramientas de construcción de índices vistas en el curso:

La primera es dividir el texto en tokens, para ello se utiliza el Tokenizer por defecto que implementa Solr, el mismo divide en tokens utilizando los espacios en blanco y signos de puntuación, para ambos campos se utilizó este Tokenizer.

Luego se procede a aplicar filtros más agresivos para mejorar la recuperación. Primero se eliminan las palabras comunes o “stop words” para el español a partir de una lista predefinida de palabras comunes. Luego de eliminadas las palabras comunes se procede a aplicar un stemmer, el mismo lleva a las palabras a su raíz. El stemmer se aplica solamente a la descripción, dado que si lo aplicamos al título, el mismo se vería afectado de forma no adecuada.

SolrNet

Para comunicar la aplicación web con el servidor de búsquedas de Solr, utilizamos el conector SolrNet que es un cliente de Solr para la plataforma .NET. SolrNet es un proyecto Open Source disponible en GitHub. Puede consultar la documentación detallada en [9].

A continuación mostramos el código más relevante en el uso de SolrNet actuando como interfaz de acceso al servidor Solr con el contenido Indexado de las diferentes grillas de programación de los múltiples proveedores.

Mapping .Net Class <=> Documento Solr

```
public class GridElement
{
    [SolrField("fecha_inicio")]
    public DateTime DateTime { get; set; }

    [SolrField("fecha_fin")]
    public DateTime DateTimeEnd { get; set; }

    [SolrField("titulo")]
    public string Title { get; set;}

    [SolrField("descripcion")]
    public string Description { get; set; }

    [SolrField("canal")]
    public string ChannelName { get; set; }

    [SolrField("proveedor")]
    public string ProviderName { get; set; }

    [SolrField("url_imagen")]
    public string ImgUrl { get; set; }

    [SolrField("genero")]
    public string Genre { get; set; }
}
```

La clase GridElement es una clase POCO (plain old class object) escrita en C# .NET. En esta clase se definen un conjunto de propiedades como *Title*, *Description*, *DateTimeEnd*, etc y para cada una se utiliza una "anotation" para indicar a SolrNet a qué campo del esquema Solr se corresponde. Por ejemplo: la propiedad "*Title*" de la clase C# se corresponde con el campo "*titulo*" del esquema Solr, la propiedad "*DateTimeEnd*" con el campo "*fecha_fin*", etc.

Este enfoque de mapping usando anotaciones permite que SolrNet pueda convertir los resultados de consultas sobre Solr en colecciones de objetos instanciados automáticamente por SolrNet como instancias de la clase GridElement.

Configuraciones de relevancia

Para recuperar la información en forma adecuada se probaron dos métodos distintos para la relevancia:

El primero es utilizar una búsqueda binaria. Para ello se construye una disyunción con las palabras que aparecen el texto introducido por el usuario y se chequea contra el título y la descripción. Y combinando las palabras clave se arma un vector de palabras que combinado con la medida TF-IDF permite definir una relevancia para cada campo, luego se ponderan las relevancias de ambos campos.

El segundo es utilizar un parser de consultas más sofisticado llamado dismax, el mismo toma el texto pasado en la consulta y construye su propia consulta. Para ello se definen campos de consulta, en estos campos es que se busca el texto y se calcula una relevancia basado en TF-IDF y en el vector de palabras de las consultas y documentos. Para mejorar las búsquedas se puede definir para cada uno de los campos de consulta un peso de relevancia, potenciando así los campos que se consideren más relevantes.

Configuración de la búsqueda

A continuación mostramos el código que implementa la configuración de búsqueda, filtrado, ordenación y paginación desde la aplicación web contra el servidor Solr usando el adaptador SolrNet con sus construcciones primitivas orientadas a objetos y fluent API. De esta forma, el programador puede despreocuparse de la comunicación con el servidor de búsqueda y de los protocolos y formatos de la comunicación subyacentes (restful services sobre http)

```
public SearchResult Search(string query, string[] providers, DateTime dtFrom, DateTime dtTo, int resultsPerPage, int pageNumber, string genre)
{
    new SolrBaseRepository.Instance<GridElement>().Start();
    var solr = ServiceLocator.Current.GetInstance<ISolrOperations<GridElement>>();

    var providerFilter = new SolrQueryInList("proveedor", providers);
    var rangeFilter = new SolrQueryByRange<DateTime>("fecha_inicio", dtFrom, dtTo);

    List<ISolrQuery> filters = new List<ISolrQuery>(){providerFilter, rangeFilter};

    if(!string.IsNullOrWhiteSpace(genre))
        filters.Add(new SolrQueryByField("genero", genre));

    SolrMultipleCriteriaQuery filterQuery = new SolrMultipleCriteriaQuery(filters, "AND");

    var options = new QueryOptions
    {
        FilterQueries = new List<ISolrQuery>(){filterQuery},
        Rows = resultsPerPage,
        Start = (pageNumber - 1) * resultsPerPage,
        OrderBy = new List<SortOrder>() {
                                        new SortOrder("fecha_inicio", Order.ASC),
                                        new SortOrder("fecha_fin", Order.ASC),
                                        new SortOrder("proveedor", Order.ASC),
                                        new SortOrder("canal", Order.ASC)
                                    }
    };

    SolrQueryResults<GridElement> results;
    if (string.IsNullOrWhiteSpace(query))
    {
        results = solr.Query("*:*", options);
    }
    else
    {
        var searchQuery = new SolrMultipleCriteriaQuery(new List<ISolrQuery>(){
            new SolrQueryByField("titulo", query),
            new SolrQueryByField("descripcion", query),
        }, "OR");

         results = solr.Query(searchQuery, options);
    }

    var searchResults = new SearchResult
                        {
                            Result = results,
                            QueryTime = results.Header.QTime,
                            TotalHits = results.Count
                        };

    return searchResults;
}
```

Inicialización de la conexión

Configuración de los filtros

Configuración de la paginación y el órden de los resultados

Dependiendo de si el texto es vacío o no se ejecuta la búsqueda usando comodín *:* (todos) o no. En caso de haber un término de búsqueda el mismo se busca sobre los campos "titulo" o "descripcion". Finalmente se ejecuta la búsqueda y se carga el resultado en la variable result como una colección de GridElement

Se configura el resultado y se retorna

Módulo de Consulta

A continuación se presenta la pantalla que les aparece a nuestros usuarios al momento de realizar su consulta, ya sea por categoría ó por proveedor.

1. Se coloca la palabra o frase a buscar y demás opciones

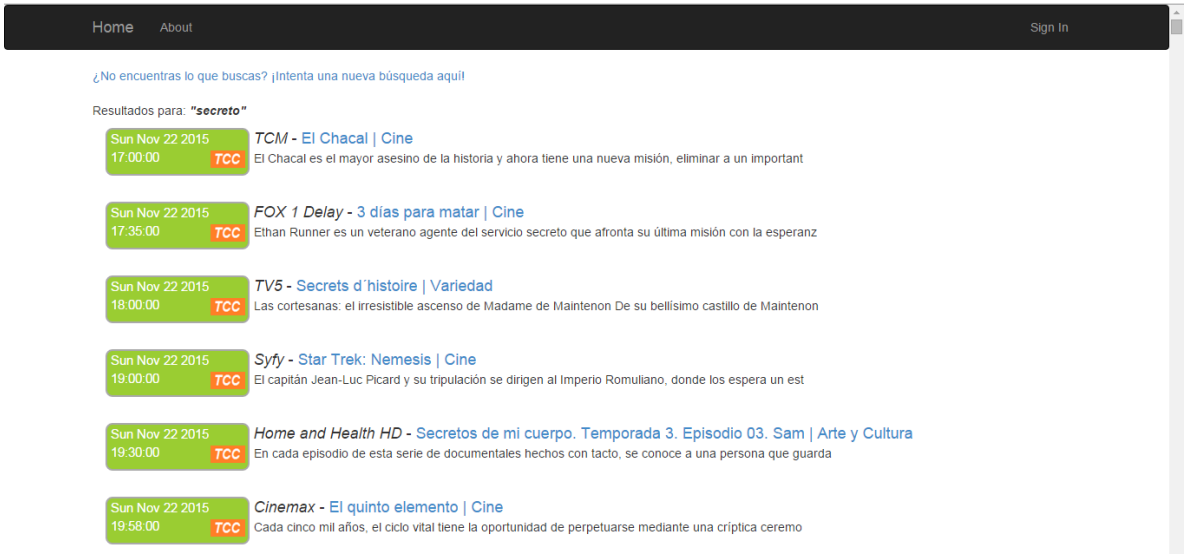
2. Se despliegan las opciones de búsqueda con respecto a la palabra o frase, indicando el proveedor de cable y el canal por el cual se emitirá

3. Al dar clic en el titulo de color azul, se despliega la descripción.

De manera adicional a la búsqueda detallada mediante palabras clave, el programa también funciona de manera similar a un motor de búsqueda en el cual se comparan las palabras introducidas para la consulta con la información disponible en la base de datos del contenido, como sinopsis, títulos, etc. Esto permite proporcionar información relacionada con las palabras de búsqueda sin tener que ingresar datos exactos.

Para demostrar lo anterior se escribe la palabra clave "Secreto"

Estas son las opciones que mostraría el programa



El programa despliega toda la información que tiene en la cual se encuentra la palabra "Secreto".

Se da clic en el enlace del interés del cliente y el sistema abre una pequeña descripción del link que se seleccionó.


Home
About
Sign In
¿No encuentras lo que buscas? ¡Intenta una
Resultados para: "secreto"
Sun Nov 22 2015 17:00:00 TCC
TCM - El Ch
El Chacal es el
Sun Nov 22 2015 17:35:00 TCC
FOX 1 Dela
Ethan Runner
Sun Nov 22 2015 18:00:00 TCC
TV5 - Secre
Las cortesanas
Sun Nov 22 2015 19:00:00 TCC
Syfy - Star
El capitán Jean
Sun Nov 22 2015 19:30:00 TCC
Home and Health HD - Secretos de mi cuerpo. Temporada 3. Episodio 03. Sam | Arte y Cultura
En cada episodio de esta serie de documentales hechos con tacto, se conoce a una persona que guarda
Sun Nov 22 2015 19:58:00 TCC
Cinemax - El quinto elemento | Cine
Cada cinco mil años, el ciclo vital tiene la oportunidad de perpetuarse mediante una críptica ceremo
3 días para matar | Cine
×
Ethan Runner es un veterano agente del servicio secreto que afronta su última misión con la esperanza de retirarse para vivir con su hija adolescente, a la que apenas conoce, antes de que sea demasiado tarde para ejercer como padre.
Sun Nov 22 2015 17:35:00
Cerrar

Casos Interesantes

Un caso interesante es por ejemplo cuando queremos buscar un programa de fútbol, en este caso realizamos la consulta “futbol”.

En este ejemplo se ve claramente que buscar un programa de fútbol, en este caso realizamos la consulta “futbol”, si bien en estos casos no hay programas con fútbol en el título sí los hay con la palabra fútbol en la descripción.

Otro ejemplo es al buscar la frase “sala de emergencias”

En este caso se ve claramente que aparecen programas con la palabra emergencias en el título. Otro caso es al buscar tirador si ponemos el filtro por deportes no recupera nada, si bien la película tirador existe.

Se puede ordenar por relevancia o por fecha y horario: En el ejemplo anterior cuando se busca futbol, se ordena por relevancia y no por fecha.

Conclusiones

Se desarrolló una aplicación web para resolver un problema existente, dicha aplicación presentó varios desafíos:

En primer lugar se debió descargar las grillas de las programaciones de los distintos proveedores de cable, esto presentó ciertas dificultades y hubo que adaptar distintas tecnologías para poder realizar esta tarea. En segunda instancia se debió indizar dichos programas en el motor de búsqueda Apache Solr, el mismo permitió resolver los problemas básicos de recuperación de información en forma sencilla una vez se tiene el conocimiento de cómo utilizarlo. Por otro lado se resolvió el problema de integrar las distintas fuentes al realizar las consultas desde una interfaz web.

Se probaron distintos modelos para ordenar los resultados, ya sea configurando la relevancia o por fecha.

Finalmente la aplicación web probó ser útil para el propósito que fue concebida permitiendo una forma fácil de acceder a la programación provista en Montevideo por los distintos proveedores de cable.

Trabajos Futuros

A futuro se pueden agregar otros filtros de búsqueda (duración por ejemplo).

Otra mejora que excede el alcance inicial del proyecto es la de incluir información complementaria sobre series, películas y eventos extraída de distintos sitios web cómo pueden ser IMDB [10] (base de datos de películas y series) o de Wikipedia.

También se puede manejar una persistencia para los usuarios, por ejemplo utilizando una base de datos SQL o MongoDB [11] en caso de que se requiera escalar. Con esta persistencia se puede mejorar la experiencia del usuario manejando recomendaciones y sugerencias, así como personalizar la búsqueda en base a preferencias de cada usuario.

Otro punto a mejorar es utilizar una lista de sinónimos cómo por ejemplo WordNet [12], para resolver problemas de sinónimos como por ejemplo football y fútbol.

Manual del Usuario

La siguiente descripción se realiza para el modulo de consulta del usuario.

1. Ingresa a la página

2. Aparece una pantalla en la cual se escribe el nombre de la consulta que se desea realizar, se selecciona la fecha. Si se desea se puede seleccionar la categoría y el proveedor de cable.

Se escribe el nombre de la consulta que se quiere realizar.

3. Por último se da clic en.... ¡Encuentra lo que quieres ver!

4. Se desplegara la información que se solicito con una breve descripción.

Un ejemplo de búsqueda sería el siguiente:


Home
About
Sign In
¿No encuentras lo que buscas? ¡Intenta una nueva búsqueda aquí!
Resultados para: "barcelona"
Sun Nov 22 2015 18:30:00 TCC
ESPN 3 HD - ESPN Compact. La Liga 2015-2016. Betis vs Atletico Madrid (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol
Sun Nov 22 2015 18:30:00 TCC
ESPN3 - ESPN Compact. La Liga 2015-2016. Betis vs Atletico Madrid (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol
Sun Nov 22 2015 19:30:00 TCC
ESPN 3 HD - ESPN Compact. La Liga 2015-2016. Villarreal vs Eibar (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol
Sun Nov 22 2015 19:30:00 TCC
ESPN3 - ESPN Compact. La Liga 2015-2016. Villarreal vs Eibar (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol
Sun Nov 22 2015 22:00:00 TCC
ESPN 3 HD - ESPN Compact. La Liga 2015-2016. Betis vs Atletico Madrid (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol
Sun Nov 22 2015 22:00:00 TCC
ESPN3 - ESPN Compact. La Liga 2015-2016. Betis vs Atletico Madrid (Grabado) | Deporte
La Primera División de España 2015/16 será la 85ª edición de la Primera División de España de fútbol

Herramientas y Tecnologías utilizadas

A continuación se listan las herramientas que se utilizaron para llevar a cabo este proyecto, con una breve descripción.

- **Parser: Python**.

El módulo analizador proporciona una interfaz para Python. El propósito principal de esta interfaz es permitir que el código Python pueda editar el árbol de análisis sintáctico de una expresión Python y crear código ejecutable a partir de esto. También es más rápido.

- **Manejador: Python**.

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 2.1.1, e incompatible en ciertas versiones anteriores.

- **Motor de búsqueda: Solr.**

Es un motor de búsqueda de código abierto basado en la biblioteca Java del proyecto Lucene, con APIs en XML/HTTP y JSON, resaltado de resultados, búsqueda por facetas, caché, y una interfaz para su administración. Corre sobre un contenedor de servlets Java como Apache Tomcat.

- **App Web: .NET**

.NET es un framework de Microsoft que hace un énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permita un rápido desarrollo de aplicaciones. Basado en ella, la empresa intenta desarrollar una estrategia horizontal que integre todos sus productos, desde el sistema operativo hasta las herramientas de mercado.

.NET podría considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos Web, como competencia a la plataforma Java de Oracle Corporation y a los diversos framework de desarrollo web basados en PHP. Su propuesta es ofrecer una manera rápida y económica, a la vez que segura y robusta, de desarrollar aplicaciones –o como la misma plataforma las denomina, soluciones– permitiendo una integración más rápida y ágil entre empresas y un acceso más simple y universal a todo tipo de información desde cualquier tipo de dispositivo.

- **Visual Studio**.

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta múltiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby, PHP; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc., a lo cual sumarle las nuevas capacidades online bajo Windows Azure en forma del editor Monaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas, etc.

- **Control de versiones: BitBucket con SourceTree**.

Bitbucket es un servicio de alojamiento basado en web, para los proyectos que utilizan el sistema de control de revisiones Mercurial y Git. Bitbucket ofrece planes comerciales y gratuitos. Se ofrece cuentas gratuitas con un número ilimitado de repositorios privados (que puede tener hasta cinco usuarios en el caso de cuentas gratuitas) desde septiembre de 2010, los repositorios privados no se muestran en las páginas de perfil - si un usuario sólo tiene depósitos privados, el sitio web dará el mensaje *"Este usuario no tiene repositorios"*. El servicio está escrito en Python.

Source Tree es un potente GUI (Graphical User Interface – Interfaz Gráfica de Usuario) para gestionar todos los repositorios ya sean Git o Mercurial. Con Source Tree se puede crear, clonar, hacer commit, push, pull, merge y algunas cosas más de una forma bastante fácil. Desarrollado por Atlassian e inicialmente solo para Mac, también cuenta con su versión para Windows.

Características destacadas de Solr

Algunas de las características más destacadas de Solr son:

- Servidor con interfaz tipo REST (interacción vía HTTP, XML, JASON, CSV, etc.)
- Esquema de datos configurable.
- Utiliza varios caches para agilizar las búsquedas.
- Interface Web de administración.
- Navegación de resultados por facetas.
- Escalable a varios servidores para búsquedas distribuidas.
- Módulos de importación de datos desde bases de datos, e-mail y archivos de texto enriquecido (PDF, Word, RTF).
- Análisis de texto (Tokenización, normalización, etc.)

Bibliografía

[1] [En línea]. Available: http://www.nuevosiglo.com.uy/sitio/. [Último acceso: 15 12 2015].

[2] [En línea]. Available: https://www.montecable.com/. [Último acceso: 15 12 2015].

[3] [En línea]. Available: http://www.tcc.com.uy/. [Último acceso: 15 12 2015].

[4] [En línea]. Available: http://lucene.apache.org/solr/. [Último acceso: 15 12 2015].

[5] [En línea]. Available: https://www.python.org/. [Último acceso: 15 12 2015].

[6] [En línea]. Available: http://scrapy.org/. [Último acceso: 15 12 2015].

[7] [En línea]. Available: http://docs.python-requests.org/en/latest/. [Último acceso: 15 12 2015].

[8] [En línea]. Available: https://support.google.com/dfp_premium/answer/4497389?hl=es. [Último acceso: 15 12 2015].

[9] [En línea]. Available: https://github.com/mausch/SolrNet/tree/master/Documentation.

[10] [En línea]. Available: http://www.imdb.com/. [Último acceso: 15 12 2015].

[11] [En línea]. Available: https://www.mongodb.org/. [Último acceso: 15 12 2015].

[12] [En línea]. Available: https://wordnet.princeton.edu/. [Último acceso: 15 12 2015].