Agenda deportiva centralizada

Informe final

Recuperación de Información y Recomendaciones en la Web

Grupo 8
Elias Cuttica 5048043-8
Andrés Pereiro 2956171-2
Maximiliano Montiglio 5040026-8
Juan Manuel Serralta 4603460-5

Introducción

Dada la gran cantidad de eventos deportivos que son transmitidos por diferentes medios de comunicación diariamente y sobre todo los fines de semana, se hace difícil poder armar un calendario de transmisiones deportivas que contengan los intereses de las diferentes personas. Estos intereses deben contemplar los gustos en diferentes tipos de deportes, medio de transmisión, ubicación del usuario que los consulta.
Si bien hay muchas fuentes web las cuales pueden ser consultadas para tener esta información, no es fácil contemplar todos los intereses antes descritos y muchas veces no todas las fuentes contienen toda la información que se desea encontrar.

El proyecto "Agenda deportiva centralizada" consiste en una aplicación web que se encarga de recolectar información deportiva de distintas fuentes y centralizar esa información con el fin de permitir realizar búsquedas de forma sencilla en base a filtros o por medio de un buscador general.

Problema

El problema principal consiste en integrar distintas fuentes de datos relacionadas a los deportes y extraer de ellas el calendario de transmisión de los eventos a los efectos de poder efectuar consultas centralizadas y aumentar la probabilidad de que la información consultada sea de utilidad.

Cada fuente de datos maneja diferentes formas de nombrar las entidades (canales, equipos, tipo de deporte, etc.) relacionadas a este tema. Cada extractor debe normalizar la información extraída para que sean más certeras las búsquedas de información.
También es importante que los usuarios tengan a su disposición la posibilidad de filtrar la información por distintas características, haciendo búsquedas sencillas. Para esto se debe resolver el problema de proveer una interfaz web lo suficientemente potente para poder hacer búsquedas con ayudas contextuales sobre gran cantidad de datos y que los tiempos de respuesta sean los adecuados.

Enfoque de la solución

Para la solución a este problema se plantea extraer los datos desde las fuentes seleccionadas teniendo un extractor específico para cada fuente.
Cada extractor normaliza los datos pasando cada tipo de dato a un único lenguaje común o tipo de dato común a los efectos de poder hacer más fáciles las consultas. Por ejemplo se utiliza una estructura común para las fechas o para palabras de diccionario como fútbol, golf, etc.
Luego de estudiar las diferentes fuentes de datos, se ve que tienen un patrón común en los datos que estas manejan. Entonces se define una estructura de datos específica para este tipo de información con todos los datos relevantes que se pueden obtener de estas fuentes a los efectos de que sean tanto informativas como datos a ser utilizadas en búsquedas.
Se tiene un repositorio cor un ladoentralizado para alojar estos datos. El repositorio cuenta con algún mecanismo de consulta para poder extraer estos datos y una interface amigable por la cual se puedan hacer estas consultas.
Por último se provee de una aplicación web facilite a los usuarios a hacer las consultas que deseen proveyendo ayudas contextuales filtradas de los datos.
Además la solución construida permite filtrar los eventos según las siguientes características de los mismos:

- Deporte
- Canal
- Fuerte

Fuentes de datos

Si bien se investigaron varios sitios de deportes para entender la información provista por estos, nos quedamos con dos que son Marca[1] y Olé[2].
Las razones en la elección de estos sitios son, calidad en los datos, y la posibilidad de poder utilizar diferentes técnicas de recuperación de información ya que en el caso de Marca usamos Scrapy y en el caso de Olé provee un api JSON.

Diseño, Implementación

El diseño del sistema consiste en:

- Interfaz web desarrollada en React.
- Recolectores de datos desarrollados en Python.
- Indizador de datos para proveer ayuda en la búsqueda de términos.
- Servidor de Elasticsearch.

Para normalizar los datos recolectados se utilizaron estructuras de diccionarios que contienen la misma palabra escrita de distinta manera. Los diccionarios permiten normalizar los datos evitando que se ingrese en el sistema información repetida y además que se muestre en la web la misma palabra escrita de distinta forma , por ejemplo si al extraer los datos, aparecen algunas las palabras como: Football, football, soccer, etc, se transforma a la palabra "Fútbol".
También se normalizo el horario del evento a un formato común (UTC 0) ya que al recolectar información de distintos sitios, estos pueden contener el mismo evento pero en distinta zona horaria.
Para evitar que en el sistema se ingrese información repetida se llevó a cabo el siguiente procedimiento:

1. Se normalizan los datos para llevarlo a una forma común. En la extracciòn se tienen diccionarios para traducir las palabras claves desde el estilo de la fuente a un estilo comùn que luego es usado para guardar en el repositorio centralizado. Cada diccionario es particular a la fuente de datos utilizada.
2. Antes de ingresar los datos se realiza una consulta por los campos sport, teams y date, si la consulta no retorna ningún resultado se ingresa, sino no. Al utilizar elasticsearch permite que esta búsqueda se ejecute de forma muy rápida ya que elasticsearch es una plataforma near-realtime, permitiendo una gran velocidad al momento de realizar búsquedas.

La consulta realizada es la siguiente:

```
"query": {
                    "bool":{
                       "must" : [
                          { "match": {"sport": datos['sport'] } },
                          { "match": {"teams": datos['teams'] } },
```

```
                    { "match": {"date": datos['date'] } }
                ]
            }
        }
    }
```

Para guardar documentos en Elasticsearch primero se debe crear un índice en dónde se define la estructura de los documentos JSON que se van a guardar.
La estructura del índice utilizado para almacenar los datos recolectados es:

```
{
  "mappings": {
    "data": {
      "properties": {
        "date": {
            "type": "date"
        },
        "sport": {
            "type": "keyword"
        },
        "competition": {
            "type": "keyword"
        },
        "teams": {
            "type": "keyword"
        },
        "channel":{
            "type": "keyword"
        },
        "source":{
            "type": "keyword"
        }
      }
    }
  }
}
```

En esta estructura se guarda la fecha del evento en UTC 0, como se describió anteriormente, el deporte al cual pertenece, tipo de competición, equipos involucrados si es que aplica, canal o fuente por el cual se transmite y la fuente de datos de la cual se extrajo.

Todos los tipos de datos están como keyword ya que se permite realizar búsquedas por cualquiera de estos campos.

También se implementó un índice para los filtros por los cuales se pueden buscar:

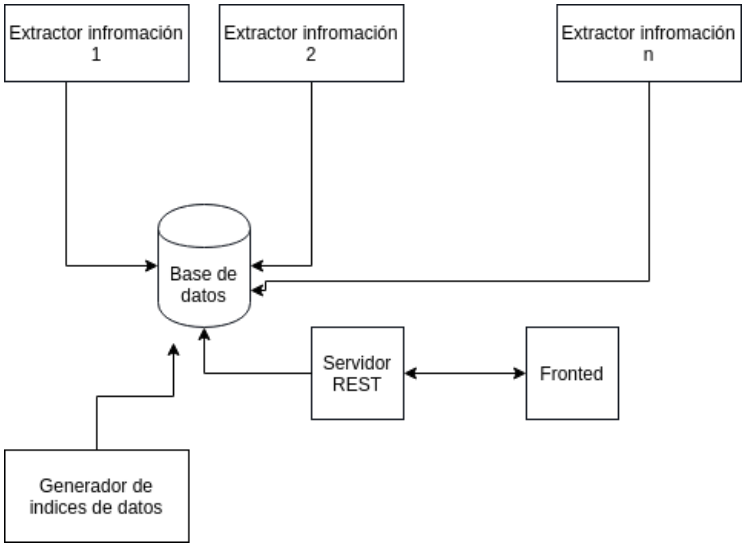
```
{
  "mappings": {
    "data": {
      "properties": {
        "type": {
            "type": "[webir] Entrega 2 Grupo XX"
        },
        "value": {
            "type": "keyword"
        },
        "quantity": {
            "type": "float"
        }
      }
    }
  }
}
```

Estos datos son utilizados para poblar el menú contextual de búsqueda y para ayuda de autocompletar.
El campo type describe cuál es el campo de datos por el cual se busca, por ejemplo "sport".
El campo value describe uno de los valores encontrados en este campo, por ejemplo "Fútbol".
Por último, el campo quantity describe cuántas veces se encuentra ese valor de este tipo en los datos y es utilizado para ordenar la ayuda contextual ubicando los más encontrado primeros.

Arquitectura

La arquitectura planteada consiste de tener un componente extractor de datos por cada fuente de datos. Cada extractor es el encargado de normalizar los datos y validar la duplicidad de los mismos. Cada extractor se comunica con un repositorio de datos centralizado que expone un API Rest de consulta a través del cual se hacen consulta de los datos desde el Frontend.
Existe también un componente encargado de consultar los datos de los eventos y generar fuente de datos para las ayudas contextuales.

La arquitectura es descrita en el diagrama siguiente:



Herramientas

A continuación se describen las herramientas utilizadas para la elaboración de este proyecto. Para realizar el scrapping de los datos se utilizó Scrapy, el cual es un framework de python que se utiliza para recolectar datos de páginas web.
Utilizamos Elasticsearch para almacenar los datos recolectados y hacer las consultas. Además utilizamos una biblioteca de python que nos permitió integrar Scrapy con Elasticsearch facilmente.
Por último se utilizó React para la elaboración del front-end de la aplicación web.

Elasticsearch[3]: Es un motor de búsqueda basada en lucene, orientado a documentos con formato JSON permitiendo indexar gran cantidad de datos. Provee API Rest potente para ingresar datos y hacer consultas.

Scrapy[4]: Framework open source para extraer los datos de sitios web. De una manera rápida, simple y extensible.

Python[5]: Es un lenguaje de programación que dada la gran cantidad de bibliotecas que posee nos permitió integrar de forma sencilla los distintos componentes.

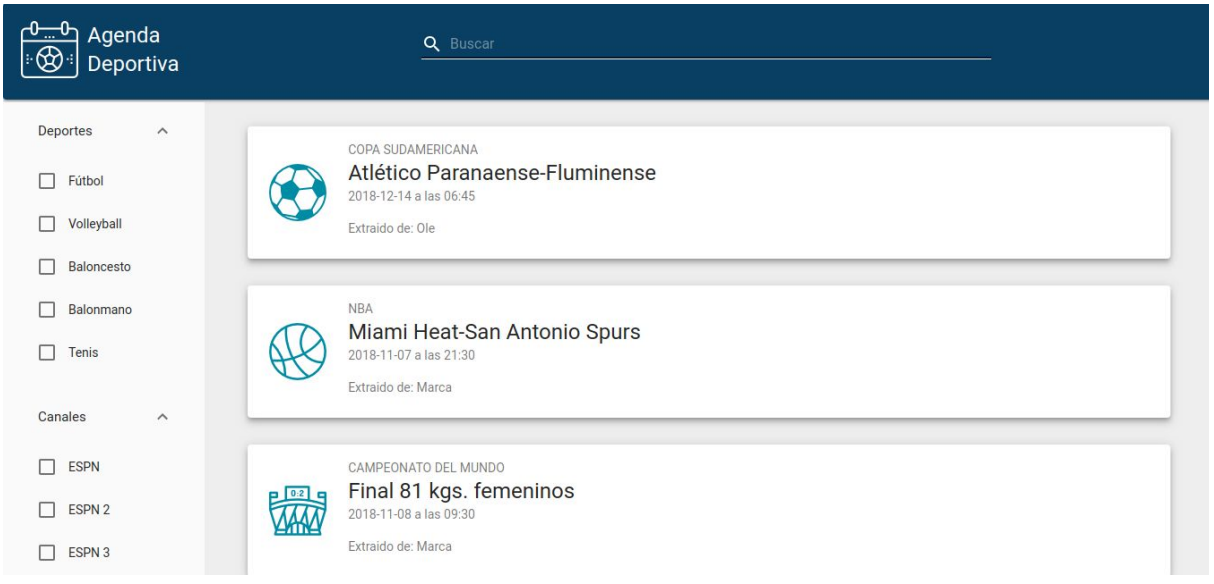
React[6]: Es un framework javascript que ayuda a construir aplicaciones web de una manera más ordenada, basándose en dividir la interfaz en componentes favoreciendo la reusabilidad

Next.js[7]: Es un framework que ya trae resuelta muchas de las configuraciones necesarias para hacer andar servidores Node con React

Funcionalidades y uso

La aplicación web contiene las siguientes funcionalidades:
Búsqueda de eventos: la aplicación contiene un buscador que permite buscar tanto por el nombre de equipo, deporte o canal. Además permite autocompletar las búsquedas, haciendo así más amigable y fácil el uso para el usuario final.
Filtros de búsqueda: Se cuentan con filtros por deporte, canales y fuente de datos, esto permite listar los eventos deportivos según el filtro elegido. Estos filtros se cargan a partir del índice de filtros creado a partir de los datos.
Listado de eventos: al iniciar el sistema se listan los últimos eventos deportivos según la fecha, luego el listado cambia según los filtros que se apliquen o la búsqueda que se realice.

A continuación se muestra una imagen ilustrativa de la web:



Evaluación y resultados

En general los resultados fueron buenos y acordes con el alcance del curso.
Las herramientas de scraping y elasticsearch fueron fáciles de integrar usando python ya que se disponían de bibliotecas que permitieron realizar las tareas de forma fácil, por lo tanto no se presentaron mayores problemas con el uso de las mismas.
Elasticsearch es un repositorio de datos robusto que provee un API muy completa y potente lo cual la hace muy fácil de utilizar y facilita su integración tanto para el ingreso de datos como para la consulta.

Conclusiones

Fueron relevadas diferentes fuentes de datos a los efectos de resolver nuestra problemática y se eligieron las dos utilizadas debido a que cumplen con la condición de ser una buena muestra como ejemplo, por un lado, de sitio web al cual aplicar la técnica de web scraping y, por otro lado, una fuente de datos que provee un API JSON que posee una estructura de datos más definida.
La elección de las fuentes de datos es esencial ya que se tiene que tener en cuenta la calidad de los mismos y el esfuerzo que hay que poner para la extracción, normalización y actualización de los datos en el sistema centralizado.

Las herramientas utilizadas fueron de mucha utilidad y facilitaron al procesamiento de los datos y posterior consulta.
Por un lado el uso de Scrapy para el procesamiento de sitios Web tiene una integración con Elasticsearch utilizando python lo cual permitió la integración casi nativa de ambas herramientas.
A su vez python posee una librería para el manejo del API rest de Elasticsearch lo cual facilita también el manejo de documentos JSON e integración con este tanto para el ingreso de documentos como para la consulta de los mismos.
La elección de Elasticsearch tuvo dos puntos de vista fundamentales, por un lado el hecho de tener un potente mecanismo de indexación de documentos utilizando un tipo de documento el cual se ha convertido en casi un estándar y liviano como es JSON. Por otro lado el hecho de poseer una interfaz REST con un potente lenguaje de consulta y de rápida

respuesta. Estos factores hacen que la integración con cualquier sistema sea muy sencillo y cumple con los criterios de aceptación que se buscaban.

Se podría decir que la elección de las diferentes herramientas que cumplen diferentes funciones (extracción, almacenamiento, etc.) es fundamental para que la construcción del sistema sea lo más sencillo posible y poder concentrar los esfuerzos en el diseño de la solución y no en poder hacer que las diferentes herramientas interactúen entre ellas. O sea que sean una ayuda y no un fin.
Con respecto al diseño se intentó que los diferentes componentes se ajusten entre sí lo más naturalmente posibles sin necesidad de forzar su interacción así como al diseño de la estructura de los datos.

Como conclusiones sobre el problema planteado podríamos decir que cumplimos con el objetivo de la centralización de la información, proveyendo una interfaz sencilla y con gran poder de consulta sobre los datos almacenados.

Trabajo Futuro

Como trabajo a futuro vale la pena destacar:

- Integrar más fuentes de datos. La solución actual solo implementa dos fuentes de datos pero se contempla que la cantidad de fuentes no sea una restricción dada las características de la estructura de datos.
- Manejo de gran cantidad de datos. Una de las razones por las cuales se elige Elasticsearch es el manejo de gran cantidad de datos. Sería interesante ver el comportamiento del mismo a medida que se tiene información histórica.
- Usuarios logueados para conocer sus preferencias como la ubicación y así lograr entregar información más certera. Por ejemplo canales solo que correspondan a su ubicación.
- Agregar repetición de transmisiones a la información. No solo "en vivo". Los datos que se obtienen solo tienen transmisión en vivo pero se podría obtener fuentes que den información sobre retransmisión de eventos a los efectos de que un usuario pueda agendar y ver posteriormente un evento que coincida con otro que quiere ver.
- Poder vincular los eventos con una agenda, por ejemplo Google Calendar, según las preferencias del usuario o porque el usuario

selecciona desde el Frontend.Google Calendar, según las preferencias del usuario o porque el usuario selecciona desde el Frontend.

Referencias

[1] https://www.marca.com/
[2] https://www.ole.com.ar/wg-agenda-deportiva/json/agenda.json
[3] www.elastic.co
[4] www.scrapy.org
[5] www.python.org
[6] www.reactjs.org
[7] nextjs.org