Recuperación de Información y Recomendaciones en la Web

Facultad de Ingeniería, UdelaR

Curso 2018

Sistema de Información de eventos musicales y culturales georreferenciados

Autores:
Alejandro Guggeri
4.555.126-6
Andrés Perez
4.833.274-6
Sabrina Sellanes
4.649.172-8
Raquel Sosa
3.168.079-4

Profesora:
Libertad Tansini

Grupo 4

Diciembre 2018

Índice

1 Introducción

Dada la gran diversidad de sitios web que existen con información de espectáculos en los diferentes puntos del país, es interesante brindarle a los usuarios mecanismos de búsqueda rápida utilizando diferentes tipos de filtros. En general los sitios tiene la información no estructurada y las búsquedas sobre los mismos son muy limitadas. Por ejemplo, el único uso de la ubicación suele ser al llegar al detalle de un espectáculo para orientar al usuario.

El objetivo principal del trabajo es tomar información provista, en primera instancia, por el sitio cartelera.com.uy, y proveer en un portal búsquedas relacionadas a los eventos encontrados pero con nuevos filtros, por ejemplo: realizar búsqueda georeferenciadas. Esta funcionalidad puede ser de gran utilidad para los usuarios, ya que puede resolver consultas del estilo: çuáles son los espectáculos que se realizarán cerca de la zona en donde estoy?".

Al analizar el sitio cartelera.com.uy, vimos que se muestran los distintos tipos de eventos agrupados por cine, música, teatro, on demand, TV cable, arte y ebooks, pudiendo acceder a los mismos mediante enlaces o búsquedas en texto plano. A partir de esto resulta interesante enriquecer esta búsqueda con nuevos filtros, y por otra parte buscar integrar información pertinente sobre los espectáculos, como por ejemplo información acerca de los artistas a partir de fuentes confiables, como puede ser la página oficial del mismo.

El presente proyecto se propone obtener la información de la cartelera, cargarla en una base de datos y definir una interfaz de consulta que permita varios tipos de consulta. Para esto se propone una arquitectura basada en el producto elasticsearch como almacenamiento y motor de búsquedas. También se incluye un módulo para la carga de datos y un cliente Web que le permita al usuario ingresar consultas de forma amigable y le devuelva los resultados.

En las siguientes secciones se define el problema en detalle, se presenta el enfoque de la solución, el diseño de la misma, se comentan algunos aspectos de la implementación, las funcionalidades brindadas y forma de uso, la evaluación del sistema desarrollado, conclusiones y trabajos a futuro.

2 Problema

2.1 Contexto y Motivación

Se propone acercar a los usuarios información sobre eventos considerando información de los artistas, la fecha y el lugar dónde se presentan. En particular se desea poder informar a los usuarios sobre los eventos que tiene cerca (geográficamente).

2.2 Tipos de consultas de Usuarios

Se desean contestar consultas de usuario del estilo:

- "¿Qué bandas tocan en Montevideo hoy?"
- "¿Dónde tocan El Cuarteto de Nos?"
- "¿Qué eventos tengo este fin de semana en Centro o Ciudad Vieja?"

2.3 Requerimientos Iniciales

- Obtener información: En una primera instancia pensamos obtener información de eventos de la página cartelera.com.uy. El sistema debe contar con un módulo en el que se realizará web scraping para obtener los datos de la web. Además en dicho módulo se deberá realizar un proceso de normalización de la información.
- En cuanto a la búsqueda geográfica, se georrefenciarán las direcciones de los lugares obtenidos con un servicio que provee la Intendencia[1]. Para zonas fuera de Montevideo se evaluará el uso de un servicio que brinda Google [2].
- Se deberá proveer un mecanismo para realizar búsquedas según ubicación geográfica (el usuario podrá seleccionar una zona de interés en el mapa), nombre de banda, estilo musical y fecha en que tocan. Para realizar este requerimiento se deberá utilizar elasticsearch como motor de búsqueda.
- Se deberá proveer un mecanismo que permita almacenar los datos georeferenciados de Montevideo.
- Se deberá brindar una interfaz web donde el usuario podrá realizar las consultas y ver los resultados obtenidos. Se espera que dicha interfaz incluya un mapa para las consultas por zonas de interés.

[1] api.montevideo.gub.uy/ubicacionRest/direcciones/posicion
[2] https://developers.google.com/maps/documentation/javascript/examples/geocoding-simple

3 Enfoque de la solución

Para la solución final se espera contar con información espacial de base que permita realizar cruzamientos espaciales. Algunas capas de interés serían: países, departamentos, ciudades, barrios.

Para la información de los eventos se considera información de internet de "Cartelera"[3]. También se consideran otras posibles fuentes como Tickantel o RedUTS. Se estudiará si se puede extraer información de los artistas de wikipedia o de sus páginas oficiales.
Para relacionar la información de los eventos con el territorio, se utiliza la dirección de las salas de los eventos. Se espera obtener la misma junto con los datos del evento y georreferenciarlas con un servicio externo de geocodificación.

Se propone usar elasticsearch[4] como componente central de la solución para la gestión de la información y resolución de las consultas.

[3] http://www.cartelera.com.uy
[4] https://www.elastic.co

4 Diseño

4.1 Tecnologías

Las tecnologías utilizadas durante el desarrollo del proyecto fueron:

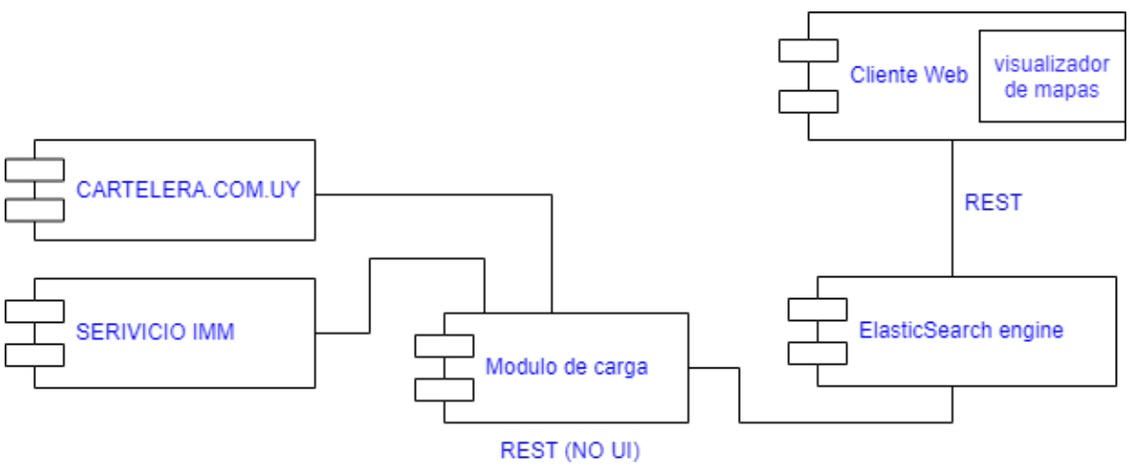
- Apache Tomcat[5], para correr el módulo de carga de datos.
- API REST Java[6] para la comunicación entre diferentes componentes.
- Jsoup[7] para web scraping para obtener datos.
- Eclipse[8] y Maven[9] como herramientas de desarrollo.
- ElasticSearch como componente para guardar y buscar los datos.
- GeoServer[10] - Servidor de Mapas - utilizado solo durante el desarrollo.
- Openlayers[11] - Visualizador de mapas incluido en la interfaz web.
- AngularJS[12] y .NET para la aplicación web.

4.2 Arquitectura del Sistema

Los principales componentes de la arquitectura del sistema son:

- Módulo de carga: Aplicación Java - WAR.
- Módulo de almacenamiento y motor de consultas: ElasticSearch
- Cliente Web que incluye el Visualizador de Mapas

A continuación se muestra el diagrama de arquitectura del sistema, a la izquierda se presentan los servicios externos con los que el sistema se comunica.



[5]http://tomcat.apache.org/
[6]https://www.arquitecturajava.com/rest-json-y-java/
[7]https://jsoup.org/
[8]https://www.eclipse.org/
[9]https://maven.apache.org/
[10]http://geoserver.org/
[11]https://openlayers.org/
[12]https://angularjs.org/

4.3 ElasticSearch

Elasticsearch es un servidor de búsqueda, que provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON. Elasticsearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache.
Utiliza Query DSL (Lenguaje de Dominio Específico) para realizar las consultas a los documentos indexados. Es un lenguaje sumamente flexible y de gran alcance, además de simple, que permite conocer y explorar los datos de la mejor manera. Al ser utilizado a través de una interfaz de tipo JSON, las consultas son muy sencillas de leer y, lo más importante, de depurar.
Además se pueden realizar consultas geoespaciales en base a elementos vectoriales (geometrías), como por ejemplo: obtener la distancia entre dos puntos geográficos, calcular el área de un polígono, intersección de un punto y un polígono, intersección entre polígonos, etc.
Como ElasticSearch tiene interfaz REST se acceder a cargar y consultar datos con pedidos HTTP de tipo POST, PUT, GET y DELETE. Durante el desarrollo y las pruebas se usó tanto Kibana[13] como PostMan[14] para hacer pruebas de conceptos, validar la carga y probar consultas durante el desarrollo. Como se partió el proyecto centrado en eventos musicales se definió una estructura para los índices de la carga de datos como se vé en la siguiente figura:

```
{
"titulo": "tituloConcierto",
    "subititulo": "subtituloConcierto",
    "artista": "artistaConcierto",
    "descripcion": "descConcierto",
    "fechaHora": "dd/MM/AAAA hh:mm",
    "sala": "salaConcierto",
    "localidades": "localidadConcierto",
    "direccionSala": "dirConcierto",
    "location": {
        "lat": coordXDir,
        "lon": coordYDir,
    }
}
```

4.4 Módulo de carga

Este módulo es el encargado de consultar los datos de la página Cartelera, consultar contra un servicio de la Intendencia de Montevideo las direcciones encontradas para el lugar de cada evento y enviar esta información a elasticsearch. El módulo de carga expone una interfaz Rest para que se pueda correr la carga a demanda. Dicha interfaz expone un único método: "/jsoupWeb/cargarDatos" del tipo HTTP POST. Este módulo está empaquetado en un WAR y se corre en un servidor de aplicaciones TOMCAT. Usa las librerías JSOUP y JERSEY[15] como implementación de JAX-RS.

[13] https://www.elastic.co/products/kibana
[14] https://www.getpostman.com/
[15] https://jersey.github.io/

4.5 Cliente Web

Este cliente está implementado con AngularJS y .NET. Tiene un área de campos de texto para que los usuarios puedan ingresar los valores de los atributos para las consultas. También muestra un mapa base provisto por OpenStreetMap[16] en un componente OpenLayers. Con este componente se pueden realizar búsquedas geoespaciales, marcando un área de referencia y obteniendo información acerca de los eventos que están dentro de esa ubicación. En este caso las tecnologías fueron elegidas porque algunos integrantes del grupo ya tenían experiencia previa en su uso. Como OpenLayers es un visualizador geográfico desarrollado en JavaScript era más directo su integración en la web desarrollada en Angular.

[16] https://www.openstreetmap.org/

5 Implementación

5.1 ElasticSearch

En primer lugar se realizaron los POST correspondientes al registro de índices, para luego poder ejecutar las consultas correspondientes. Las consultas ejecutadas en ElasticSearch se construyeron a partir de la documentación brindada en el sitio de la herramienta. Se destaca que tanto la carga de información, la consulta y las respuestas que brinda Elasticsearch son el formato JSON.

A continuación se presenta, a modo de ejemplo, la estructura de una consulta con un componente espacial, en este caso se define un área de consulta rectangular considerando 2 vértices opuestos:

```
QUERY
{
    "query":{
        "bool": {
            "must": {
                "match_all": {}
            },
            "filter": {
                "geo_shape": {
                    "location": {
                        "shape": {
                            "type": "envelope",
                            "coordinates" : [[-56.0, -34.0], [-57.0, -34.0]]
                        },
                        "relation": "WITHIN"
                    }
                }
            }
        }
    }
}
```

Consulta para obtener los conciertos que hay en el polígono especificado

En la siguiente imagen se aprecia un ejemplo del JSON que se recibe de respuesta a este tipo de consultas, dónde aparece un evento con título "Belmontez que tiene coordenadas ubicadas en Montevideo.

```
{
    "took": 313,
    "timed_out": false,
    "_shards": {
        "total": 5,
        "successful": 5,
        "skipped": 0,
        "failed": 0
    },
    "hits": {
        "total": 1,
        "max_score": 1,
        "hits": [
            {
                "_index": "indicegeo",
                "_type": "tipo",
                "_id": "GNTvo2YBHGuCa8IKW9fJ",
                "_score": 1,
                "_source": {
                    "titulo": "Belmonte",
                    "subtitulo": "null",
                    "artista": "Federico Veiroj",
                    "descripcion": "A Belmonte (43), interesado en retratar al ser humano...",
                    "fechahora": "15/11/2018 22:20",
                    "sala": "18 de Julio 930 casi Rio Branco",
                    "localidades": "",
                    "direccionSala": "18 de Julio 930 casi Rio Branco",
                    "location": {
                        "lat": -56.1963038155006,
                        "lon": -34.90638242185157
                    }
                }
            }
        ]
    }
}
```


5.2 Componente de Carga

La implementación de la funcionalidad expuesta en este componente consiste en recorrer todo el html encontrado en cartelera.com.uy. Desde aquí se obtienen los distintos links a donde conseguir información referente a los tipos de eventos expuestos en la página, música, películas entre otros. Para cada tipo de evento se consulta por todos los eventos disponibles y para cada evento se obtiene la información pertinente, consultando además la sala en donde se encuentra. Al obtener la dirección de la sala como un String, se parsea la misma y se consulta contra un servicio de la intendencia que a partir de ciertos datos de entrada devuelve un conjunto de coordenadas.

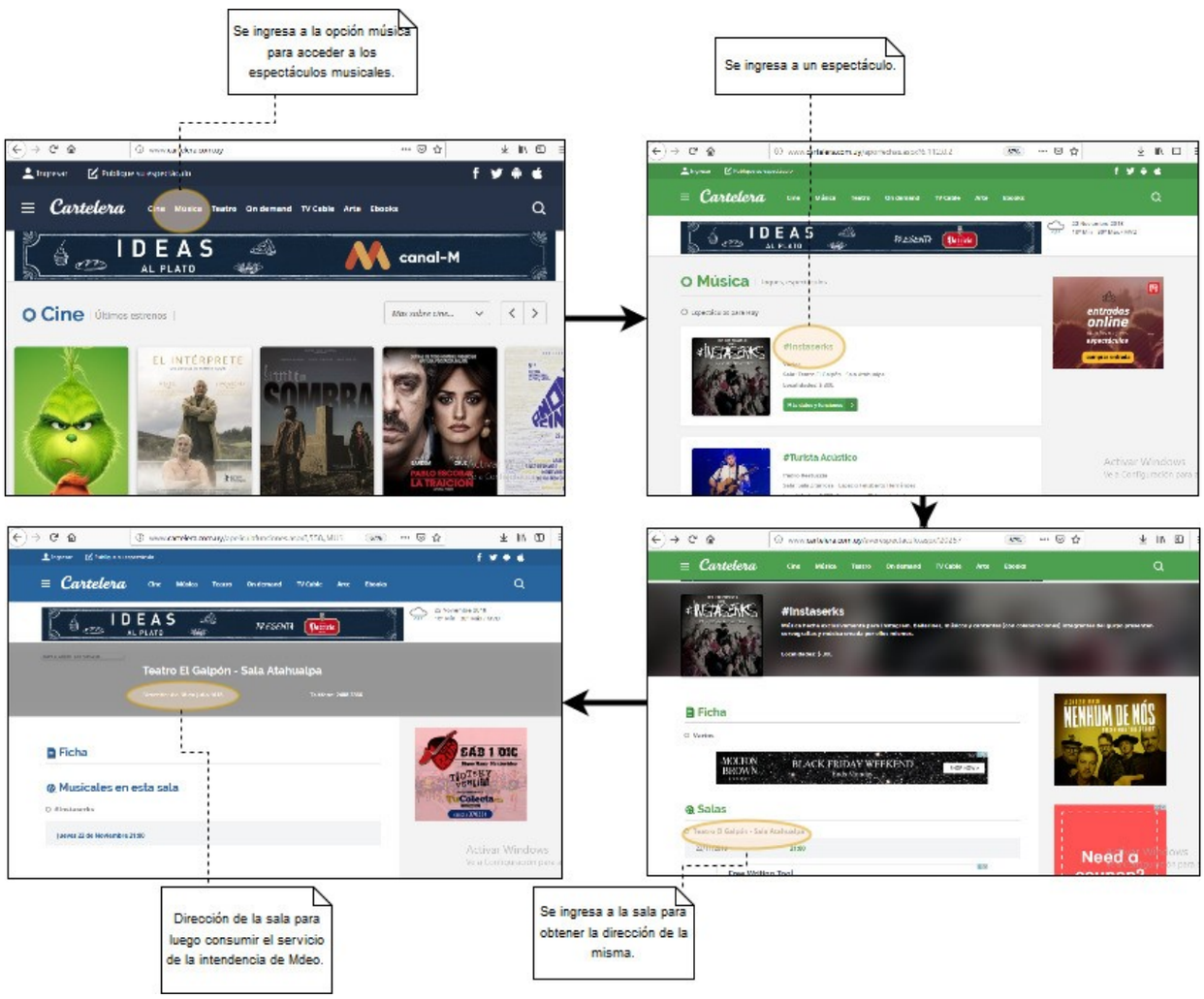
La mayor dificultad en esta parte estuvo compuesta por el parseo de la dirección. Los servicios expuestos por la intendencia son tres y algunos dependen de otros, el primero a usar es el que se le pasa un nombre de una calle y devuelve un número que referencia a esa calle. Los otros dos mapean un cruce de calles (enviando dos números que representan cada uno una calle, obtenidos en la consulta antes mencionada) o una dirección (enviando también dos números que representan el primero, una calle obtenida en la consulta antes mencionada, y el segundo un número de puerta de dicha calle). Lo difícil fue obtener la información de la calle y el número, ya que existen varias combinaciones, a modo de ejemplo, en los datos se encontró: Mercedes 1805, Buenos Aires s/n esquina Juncal, Maldonado 1150, esq. Héctor Gutiérrez Ruiz, Andes y Mercedes. Por otro lado también se encontraron direcciones del interior, en cuyo caso se descartaron por no proveer, el sistema de la intendencia, información al respecto.

Luego de obtener las coordenadas de cada evento, se tuvo que convertir las mis-

mas del sistema EPSG:32721 (provisto por la intendencia) al sistema de referencias EPSG:4326 qué es el que maneja Elasticsearch. Para esto se usó la implementación Java de la biblioteca PROJ4[17]

Otro aspecto que se destaca en este módulo, es que se decidió darle prioridad a la velocidad de las consultas, para esto se presentó un único concepto, el de evento. Este evento se "aplana" por la dirección de evento y horario. De esta forma, por cada uno de estos conceptos se realiza un request POST, al host que publica Elasticsearch y se carga cada uno de estos elementos.

A continuación se presenta el flujo de navegación de cartelera.com.uy para la obtención de sus datos.



[17]https://trac.osgeo.org/proj4j/

5.3 Cliente Web

El cliente Web realizado tiene la finalidad de proveer una interfaz gráfica al usuario para que pueda utilizar el sistema implementado. En si, la web consiste en una serie de filtros (inputs) y de un mapa. Existen dos modalidades de consultas independientes, la primera es a través de la serie de filtros disponibles que abarcan los campos de : Titulo, Subtitulo, Descripción, Artista, Fecha y Hora, Sala, Dirección y Localidades, mientras que la segunda se realiza mediante el dibujo de un polígono en el mapa.

Cuando un usuario realiza una búsqueda por filtros se realiza la siguiente consulta, similar a las anteriores:

```
{
    from: 0,
    size: 100,
    query: {
        bool:
        {
            should: []
        }
    }
}
```

Los campos *from* y *size* se utilizan para determinar la cantidad de registros máximos que se devuelven.
Cabe agregar que para cada filtro se agrega un JSON al arreglo *should* que sigue la siguiente estructura:

```
{
    wildcard: {
            Titulo: "*" + valor_Titulo + "*"
            }
}
```

El tag especial *Wildcard* y los * se utilizan para que ElasticSearch realice la búsqueda por substring.

En cambio, cuando se realiza una consulta geográfica se realiza la siguiente consulta:

```
{
    from: 0,
    size: 100,
    query: {
            bool: {
                    must: {
                        match_all : { }
                    },
                    filter : {
                            geo_polygon: {
                                location: {
                                        points: []
                                }
                            }
                    }
            }
    }
}
```

En donde para cada vértice del polígono se traduce de (X,Y) al siguiente objeto, el cual se inserta en el array points.

Para cada resultado devuelto por las consultas (ya sea por filtro o geográfica) se verifica si se encontraban las coordenadas del evento en el resultado (ya que podían estar no definidas), en el caso de estarlo se marcaba en el mapa la ubicación. Esto se logra mediante la creación de una capa adicional de puntos, en la cual se agregan todas las coordenadas de los resultado que luego se añade al mapa.

```
{
    lon: X,
    lat: Y
}
```

Por último, la implementación de la web trajo una serie de complicaciones que estuvieron dadas por la interacción del mapa de OpenLayer con el framework AngularJS y la interpretación de los JSON devueltos por ElasticSearch.

6 Funcionalidades y uso

El sistema propuesto permite realizar consultas sobre los datos cargados, a través de una interfaz web que tiene por un lado campos de texto libre y por otro lado un mapa. En los campos de texto libre se puede ingresar palabras que se usarán en la búsqueda de acuerdo a lo que se carga en cada campo: Título, Subtítulo, Artista, Descripción, Dirección (texto), Sala, FechaHora y Localidades.
No es necesario llenar todos los campos, sino que se puede consultar por uno solo de ellos. El texto ingresado se buscará dentro del campo correspondiente de la estructura en elasticsearch como un substring. Esto es si un usuario busca Artista "Tra" se le devolverán conciertos de "La Trampa". El sistema también unifica el texto para evitar problemas de mayúsculas y minúsculas.
Para usar la consulta por área de interés hay que seleccionarlo en la sección "Herramientas de Mapa" y dibujar el polígono deseado sobre el mapa. Se van haciendo clicks para marcar los diferentes puntos del polígono y para terminar de definirlo se usa el doble click. Esto enviará a elastichsearch el área deseada para buscar los eventos que quedan dentro de esa zona. Para enviar la consulta se utiliza el botón Buscar que se aprecia en la imagen:

Las consultas pueden ser solo por textos o geográficas. Luego de ejecutar una consulta los resultados de los detalles de los eventos recuperados se muestran en una tabla debajo del área de consulta. Adicionalmente, en el mapa se ponen marcadores para mostrar la ubicación de los eventos encontrados.

Titulo	Subtitulo	Artista	Descripcion	Sala	Dirección	Fecha y Hora	Localidades
Venom - Cartelera		Ruben Fleischer[, Woody Harrelson, Jenny Slate, Riz Ahmed]	Eddie Brock (Tom Hardy) es un astuto periodista que está investigando una empresa llamada Fundación Vida. Esta fundación, dirigida por el eminente científico Carlton Drake (Riz Ahmed), está ejecutando secretamente experimentos ilegales en seres humanos y realizando pruebas con formas de vida extraterrestres y amorfas conocidas como simbiontes.	Movie Montevideo	18 de Julio 1710 esq. Magallanes	27/11/2018 16:05	

7 Evaluación y resultados

Con el proceso de carga de datos se obtuvieron unos 4000 registros en promedio. Durante las pruebas se realizaron cargas en diferentes días, ya que la cartelera de eventos va variando permanentemente. Para estas cargas, los tiempos que se obtuvieron en promedio fueron de 10 minutos. En este tiempo se incluye la obtención de los datos de cartelera, y también las consultas respectivas contra el servicio de la intendencia de Montevideo para la obtención de las coordenadas, su transformación y el almacenamiento de todo en elasticsearch.

Como el servicio de georeferenciación es de la Intendencia de Montevideo, se cargaron muchos datos de eventos del interior, pero para esos registros no se guardan coordenadas. Para que no aparecieran sumados a los resultados de las búsquedas espaciales hubo que ponerles un valor por defecto que nos asegurara que para una búsqueda en Montevideo no aparecieran.

Otro aspecto a considerar es la introducción de errores por las transformaciones de coordenadas espaciales. En el caso de las coordenadas brindadas por el servicio de la Intendencia, vienen en el sistema de referencias UTM 21S y son transformadas usando la biblioteca PROJ4 de Java a coordenadas de la forma latitud-longitud. En el caso del mapa, se usa el sistema Latitud-Longitud pero el mapa tiene unidades en metros. Esto hace que a las coordenadas del polígono de consulta también haya que transformarlas a los grados que corresponden (esta transformación se hizo en forma directa con una cuenta aproximada).

Un resultado interesante es que aunque inicialmente se pensaba trabajar solamente con datos de eventos musicales, en el sistema final se cargan todos los eventos que tiene el sitio cartelera: esto incluye Cine, Música y Teatro. Otro elemento que se pensaba considerar era el origen de los artistas, pero en un análisis de los datos vimos que esa información muy pocas veces estaba disponible y generalmente aparece en el área de descripción en un campo de texto libre. Los usuarios podrán consultar ese campo pero el sistema no tiene la inteligencia para tener ese dato claramente identificado.
Por otro lado, de acuerdo al planteo inicial se agregó la información de la sala del evento, que incluye los cines, teatros, bares, etc.

8 Conclusiones

Se logró desarrollar un sistema completo que consulta información sobre eventos del sitio Cartelera, la almacena en elasticsearch y ofrece una interfaz amigable para poder consultarla, incluyendo la posibilidad de consultar por área de interés. Resultó interesante el uso de nuevas herramientas que no conocíamos, tales como jsoup para la carga de datos y elasticsearch para el guardado de los mismos. Ambas resultaron de fácil uso y ayudaron mucho para lograr la solución final presentada.

Más allá del aprendizaje de nuevas tecnologías y la integración de diversas tecnologías, uno de los más grandes desafíos del proyecto fue el trabajo con datos no muy estructurados. En varias ocasiones se necesitó normalizar o transformar los datos. Al momento de querer procesar las direcciones de las salas de los eventos nos vimos enfrentados a varias formas de escribirlas, como se detalló en la sección que describe la implementación del módulo de carga. Incluso para las direcciones que se lograban georeferenciar se hizo necesario transformar las coordenadas para luego poder cargarlas en elasticsearch y que se pudieran consultar. Reconocemos la complejidad de trabajar con datos no pensados para la consulta automática aunque no llegamos a integrar datos de diversas fuentes.

Consideramos que se lograron los objetivos del proyecto obteniendo un buen prototipo de un sistema de consulta en base a recuperación de información de la Web. Algo novedoso de este proyecto fue la integración de consultas por zonas geográficas (basada en polígonos arbitrarios definidos por el usuario). De esta forma se cubre que los usuarios puedan buscar eventos en un área de interés o en zonas cercanas a su casa.

9 Trabajo Futuro

Como trabajos a futuros se considera interesante abarcar algunos otros aspectos en cuanto a los datos. En principio, obtener alguna otra información del evento, por ejemplo, información de la página oficial del artista o el ranking de https://www.imdb.com/ en caso de una pelicula. Esto se planteó como una área a explorar en el proyecto, pero debido a las dificultades del trabajo con la fuente inicial no se pudo dedicar tiempo a esta exploración.

Por otra parte resulta atractivo que esta solución se pueda aplicar para otros proveedores de eventos como por ejemplo tickantel. Enriqueciendo de esta forma no solo la búsqueda, pero sino también la variedad de contenidos y detalles a mostrar. Por ejemplo, en Cartelera la información de precios de las entradas es algo muy vago, en tickantel es una información mucho más precisa.

Finalmente para llevar el sistema a producción sería necesario que la carga de datos se realice periódicamente, de forma de que los mismos estén siempre actualizados. Como la carga es automática, solamente se necesitaría hostear toda la aplicación en un servidor online y programar una tarea que actualice los datos cada noche.