WEBIR 2018

ENTREGA FINAL

GRUPO 12

PROYECTO

¿QUÉ HACEMO?

INTEGRANTES

Mauro Mottini	**María Luciana Martínez**	**Guido Dizioli**	**Martín Pacheco**
4.693.539-0	4.421.798-2	4.838.731-5	4.638.953-7

DOCENTE

Libertad Tansini

Índice

1. Introducción

Hoy en día existe una gran cantidad de información en la web. En particular en lo que refiere a eventos, existen muchas fuentes como lo pueden ser redes sociales, páginas especializadas en la información de eventos, páginas de ventas de tickets, foros de discusión, grupos de meetup, etc. En todas las fuentes existen diversos formatos de acceso (público, privado, ubicación en sección específica, etc) y presentación de la información, en donde además, varían los datos, así como los adicionales a la información central del evento como por ejemplo: precio, medios de pago, promociones, condiciones especiales, instrucciones de acceso, entre otros.

Este estudio se enfoca en particular en la información existente a **eventos culturales en la ciudad de Montevideo**. No obstante, el problema analizado es aplicable a eventos de cualquier índole y ubicación. El conocimiento generado en torno a este análisis puede ser aplicado de forma en variados escenarios.

A medida que las empresas, grupos culturales, artistas y demás actores reconocieron internet como la principal fuente de llegada al público, comenzaron a plasmar toda su información en el mismo. Es de allí que todo tipo de negocios, tanto pequeños bares, grandes organizadores de eventos y artistas independientes deciden colocar la información relativa a sus eventos en la web.

Este canal de llegada al público contrae grandes beneficios para estos generadores de eventos, como difusión de manera gratuita y masiva, traduciendo esto a mejoras en la rentabilidad, difusión y concurrencia a los mismos.
También se observan beneficios para los usuarios, ya que fácilmente pueden acceder a la información sobre los eventos de sus lugares preferidos, e informarse de futuros eventos que talvez no conocerían.

2. Problema

Dado el gran flujo de información existente en el internet, observamos que es difícil para un usuario acceder a toda la información que estos negocios publican en internet. Esto es debido a que suele hallarse dispersa en distintos sitios web.

Para el caso de uruguay identificamos los siguientes sitios:

- Cartelera [http://www.cartelera.com.uy]
- RedUTS [http://reduts.com.uy/]
- TickAntel [https://tickantel.com.uy/]
- Agenda de Actividades (Intendencia de Montevideo) [http://www.montevideo.gub.uy/agenda-de-actividades]
- Páginas específicas de Facebook de cada negocio o usuarios específicos de cada negocio en Instagram.

Se mencionan solo algunos de los varios canales existentes, pudiendo haber otros pero que se dejan por fuera del alcance de este análisis.

La diversidad de canales presenta un problema para el usuario, dado que podría perder información, debido a que resulta tedioso para el mismo consultar todos los canales presentes, estar al tanto de los cambios. Esto genera que se reduzca la cantidad de información a la que accede, así como datos adicionales de los eventos que podrían ser de su interés.

3. Enfoque de la solución

Para ayudar a resolver este dilema que presenta el usuario, se decidió implementar un sistema web el cual unifica la información de estos canales, agrupando y organizando los eventos de los mismos, para así proveer la información al usuario de forma centralizada.

Para la solución presentada se tomaron dos de estos canales con diferentes formatos, por un lado se tomó una fuente estructurada de información, como ser TickAntel (https://tickantel.com.uy/). La segunda fuente escogida fueron páginas específicas de Facebook, las cuales presentan su información de una manera más desestructurada.

La justificación de esta elección de canales, fue demostrar como se puede integrar la información de dos sitios distintos, cuando la misma se presenta de manera muy distinta.

4. Diseño

Obtener información del canal TickAntel

Para obtener la información que se encontró en TickAntel, se decidió utilizar la estrategia de *web scraping*, mediante la cual se analiza la estructura del *HTML* de la web objetivo, extrayendo así la información necesaria de cada evento.
Se scrapearon los eventos a partir de la url https://tickantel.com.uy/inicio/buscar, agregando las diferentes categorías:

- Deportes (https://tickantel.com.uy/inicio/buscarDeportes),
- Música (https://tickantel.com.uy/inicio/buscarMusica),
- Teatro (https://tickantel.com.uy/inicio/buscarTeatro),
- Danza (https://tickantel.com.uy/inicio/buscarDanza),
- Otros (https://tickantel.com.uy/inicio/buscarOtros).

Las distintas categorías sirvieron posteriormente para clasificar los eventos.

Obtener información del canal Facebook

Para este canal, inicialmente se intentó acceder a la información mediante la *API* que provee la misma plataforma, *Graph API*. Dado los recientes sucesos en la compañía Facebook, su *API* al momento de realizar el trabajo sufrió cambios los cuales imposibilitaron acceder a toda la información buscada mediante este medio.

Fue por esta razón, que se decidió aplicar la misma técnica que para TickAntel, *scrapeando* la información de eventos de la siguiente página:
https://www.facebook.com/events/discovery/

Almacenamiento de la información

Para almacenar la información recabada, se estructuró la información siguiendo un modelo de datos único. A continuación se presenta un ejemplo para visualizar el mismo:


```
{
	"date": "06/04/2019",
	"title": "Bolshoi Ballet - La Bayadere",
	"place": "Teatro Solís-Sala Principal",
	"tag": "Danza",
	"url":
"https://tickantel.com.uy/inicio/espectaculo/40005296/espectaculo/Bolshoi%20Ballet%20-%20La%20Bayadere",
	"description": {}
}
```


Se generaron diferentes archivos de formato *JSON* con las listas de eventos obtenidas a través de las páginas *scrapeadas.* Una vez normalizada la información, se impactó la misma de manera asincrónica mediante una *API* creada por el equipo de investigación en una base de datos con la funcionalidad de un motor de búsqueda *ElasticSearch*.

Elasticsearch es un motor de búsqueda de texto completo que permite indexar y analizar en tiempo real grandes cantidades de datos de manera distribuida. Permite almacenar documentos e indexar todos los campos de estos documentos. A diferencia de otros sistemas parecidos, no necesita declarar un esquema de la información que añadimos.

Acceso del usuario a la información

Se implementó un interfaz web, mediante la cual se despliegue la información recabada, logrando así el objetivo de centralizar la información para el usuario. Es de esta manera que se le agrega un valor al usuario, una web en la que pueda observar rápidamente la información de eventos orientados a espectáculos.

5. Implementación

Web scraping

Para realizar el *scraping* de los diferentes sitios web se utilizó la librería Puppeteer[1].
Puppeteer es una librería basada sobre el ambiente NodeJS[2] que proporciona una API de alto nivel para controlar Chrome or Chromium en modo *headless*, es decir, sin la necesidad de la interacción por medio de la interfaz gráfica. En vez de interactuar con los elementos visuales como se hace normalmente, se implementan casos de uso de manera programática para la navegación de la web.

En cuanto a la implementación para el *scraping* del sitio web Facebook, se presentó una dificultad extra, la cual fue tener que loguearse a la red social. La misma fue sorteada mediante casos de uso extra en la ejecución del proceso.

API

Flask Python
Se utilizó el framework Flask de Python para la implementación de la API debido a que es minimalista y adecuado para el caso.
Se definió un endpoint público **/get_all** para obtener todos los eventos cargados en Elasticsearch. Ante la petición HTTP se realiza un búsqueda en ElasticSearch usando la librería *Elasticsearch* de python.
El otro endpoint definido para la carga de los datos es **/insert_data** que permite la carga de los datos a la base. Este endpoint es privado y solo se utiliza de manera interna para popular la base de datos con el resultado del scraping.

[1] https://github.com/GoogleChrome/puppeteer
[2] https://nodejs.org/en/

Además definió un endpoint público **/search** que permite realizar búsquedas en ElasticSearch mediante un término de búsqueda.

Elastic Search

Se decidió utilizar un servicio online gratuito que provee de una instancia de ElasticSearch llamado Bonsai, debido a que simplifica las complejidades de instalar y configurar un servidor de ElasticSearch y al mismo tiempo permite acceso remoto a todos los miembros del equipo de desarrollo.
En el caso de poner en producción la aplicación sería necesario realizar la instalación de Elasticsearch en un servidor adecuado que provea el servicio de manera ininterrumpida y sin tantas limitaciones como Bonsai, pero durante la fase de prueba de concepto es suficiente el uso de esta alternativa.

Interfaz Web

Para la interfaz web se decidió utilizar a React[3].

React es una librería escrita en el lenguaje Javascript, enfocada en la construcción de interfaces gráficas. Además de ser una tecnología *open source* (una de las principales razones por las cuales React fue elegido para construir la interfaz gráfica de este proyecto), el mismo también cuenta con herramientas que permiten la construcción de una aplicación web en poco tiempo, creando la estructura básica del proyecto, importando las librerías necesarias (siendo muy fácil agregar nuevas) y brindando vistas para comenzar la implementación.

Dado el enfoque del proyecto, resultó de gran importancia brindarle al usuario una forma amigable e intuitiva de mostrar todos los eventos que fueron recolectados. Por esta razón

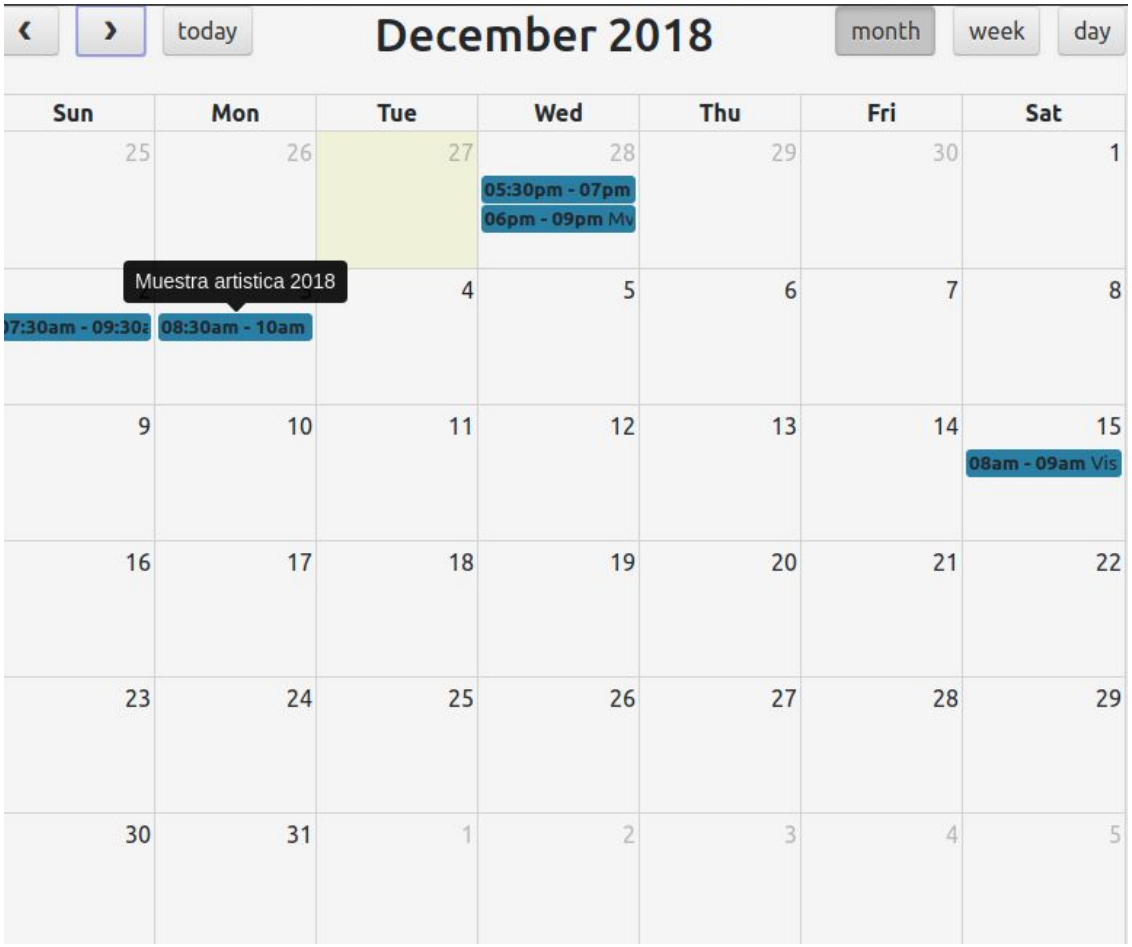
[3] https://reactjs.org/

se decidió que la mejor forma sería con un calendario donde se pudieran visualizar fácilmente todos los eventos.

Para el calendario el enfoque utilizado fue el de brindarle al usuario la mayor cantidad de opciones para que pudiera personalizar su experiencia. Por esta razón se decidió que el calendario tuviera 3 opciones de visualización: mes, semana y día. Estas 3 opciones permiten que el usuario pueda elegir entre un rango más amplio de fechas, visualizando más eventos, pero siendo menos claro para cada fecha, o reducir el rango hasta un solo día, pudiendo visualizar mejor ese día en específico. Además de lo ya mencionado se agregó un botón para que el usuario pueda ir a la fecha actual, dado que a veces (particularmente en la vista en días) puede resultar un poco confuso encontrar la fecha deseada.

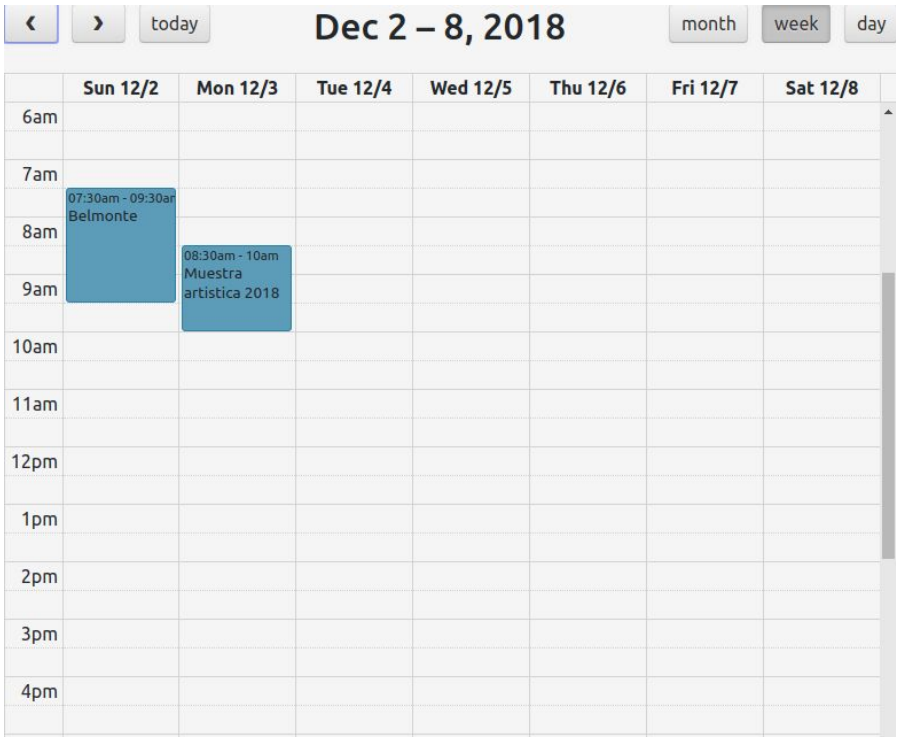
6. Evaluación y resultados

A continuación se presentan imágenes del resultado del producto final. Se puede observar cómo se cumple con los requerimientos del problema planteado, brindado al usuario información de diversas fuentes de manera fácil de entender, agregando así valor a los mismos.

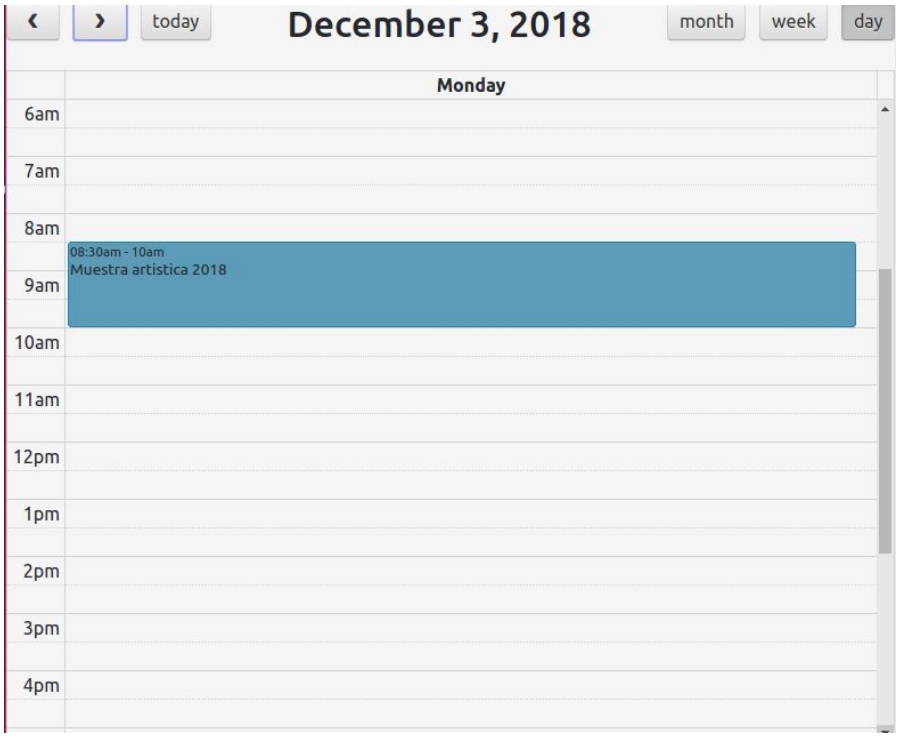


[4]

[4] Vista mensual de eventos



[5]



[6]

[5] Vista semanal de eventos
[6] Vista diaria de eventos

7. Conclusiones

Para finalizar, el equipo quedó satisfecho con el producto final. Se logró demostrar la fiabilidad de la combinación y agregación de la información sobre eventos de espectáculos, provenientes de distintas y diversas fuentes.

Por más que se presentaron obstáculos para recolectar la información, se desarrolló una solución robusta la cual no depende del proveedor de información. Mediante la técnica de *scraping*, podremos recolectar información de cualquier sitio web, comprobando que el diseño de la solución no solo es viable, sino que aplicable a una gran gama heterogénea de casos de uso más allá de eventos de espectáculos.

Por otro lado, se pudo resolver el problema planteado, combinando la información de distintos canales, y presentando la misma de una manera útil para el usuario.

Dada la infraestructura de base de datos, queda pendiente y como posible trabajo futuro, implementar una interfaz gráfica que permita al usuario realizar una *meta-búsqueda* en al que ingresando una oración, se encuentren eventos que incluyan términos de la misma. La solución presentada presenta las capacidades de realizar esta funcionalidad debido a que la información es registrada en una base de datos con un motor de búsqueda ElasticSearch.

8. Trabajos Futuros

- ❏ Obtener más información de los eventos scrapeando el evento en sí, es decir, haciendo click en cada evento con el fin obtener toda la descripción y así agregar precios, descuentos, descripción, etc.
- ❏ Agregar eventos provenientes de nuevas páginas web, como ser http://www.cartelera.com.uy/ y otras fuentes identificadas o nuevas que surjan.
- ❏ Personalizar las búsquedas de eventos en calendarios por medio de usuarios con diferentes intereses.
- ❏ Realizar recomendaciones de eventos teniendo en cuenta usuarios con intereses similares.
- ❏ Agregar la posibilidad de que los usuarios agreguen correcciones a la información de los eventos, como por ejemplo cambios de última hora en la programación o cancelación de los mismos, modificación de fechas o lugar, etc.
- ❏ Agregar información de comentarios de los usuarios sobre los eventos, con el fin de brindar información adicional sobre el mismo, desde el punto de vista de los que concurrieron.
- ❏ Automatizar la ejecución periódica el proceso de scraping de las fuentes, para esto es necesario además definir una estrategia de merging en el caso de que la información sea modificada en las fuentes entre un scraping y otro.
- ❏ Diseño gráfico de la interfaz web del usuario con el objetivo de mejorar la experiencia del usuario al utilizar la aplicación.

9. Referencias

- https://github.com/GoogleChrome/puppeteer
- https://www.toptal.com/puppeteer/headless-browser-puppeteer-tutorial
- https://hackernoon.com/tips-and-tricks-for-web-scraping-with-puppeteer-ed391a63d952
- https://codeburst.io/a-guide-to-automating-scraping-the-web-with-javascript-chrome-puppeteer-node-js-b18efb9e9921
- https://nodejs.org/en/
- https://www.elastic.co/products/elasticsearch
- http://flask.pocoo.org
- https://bonsai.io