Entrega Final – Grupo 12

WEBIR 2023

Supervisor: Libertad Tansini

Julieta Dubra	4.972.820-7
Mauro Centurio	4.633.553-6
Ramiro Bentancor	5.126.656-8
Alexis Friss de Kereki	4.864.524-2

Indice

Introduccion

El turismo uruguayo en Argentina ha llegado, en este último año, a niveles récord. Es más, los uruguayos gastaron US$960 millones en Argentina en los primeros nueve meses del año (Parks). Argentina cuenta con un repertorio más amplio que Uruguay de espectáculos, obras de teatro, conciertos, museos y programas culturales, que además resultan económicamente accesibles a los uruguayos por la situación actual del país. Es por esto y su cercanía geográfica que muchos uruguayos deciden viajar allí para hacer turismo y compras.

Hoy en día no se cuenta con una plataforma centralizada que permita encontrar actividades culturales para hacer durante la visita. Sin embargo, viajar para asistir a un concierto o ver una obra de teatro es algo sumamente común. Se presentará entonces en este trabajo la plataforma realizada, una página web que permite encontrar distintos espectáculos y ofrece además una serie de filtros pensados para afinar la búsqueda y mostrar resultados más relevantes para el usuario.

Mientras que esto se podría hacer para todo tipo de espectáculos, y la página podría ser ampliada en un futuro, este trabajo se centra sobre todo en obras de teatro, ballet, orquestas, o cualquier tipo de show que ocurre en un teatro.

Descripción de la Solución

Para poder implementar esta plataforma, se tuvo que extraer datos de distintas fuentes de la web, para luego integrar los mismos en algo estandarizado que permita la implementación de filtros y órdenes. En este caso en particular, como no se encontraron APIs específicas que devolvieran información útil para el problema presentado, se extrajo la información de dos páginas web relevantes al problema:

- La cartelera del diario la Nación (“Cartelera de Teatro de Argentina - LA NACION”)
- La página web del teatro Colón en Buenos Aires. (“Temporada | Teatro Colón”)

Originalmente, se había elegido una fuente de datos distinta llamada Plateanet (“Todo el teatro para vos”) para la cual se llegó a implementar también un scraper. Esta opción, sin embargo, tuvo que ser dejada de lado por dos principales razones. En primer lugar, la página agregó una cola digital con un captcha que hizo que el scraper no pudiera acceder más a los datos. Además, cuando aún funcionaba, el tiempo de extracción de la información era largo, tomando alrededor de 3 horas. Aunque el segundo problema no era bloqueante en un principio, el primero sí lo fue y por lo tanto se decidió intercambiar esta fuente con la cartelera de La Nación, ya que las páginas tenían estructuras e información similar.

Para implementar la extracción de datos, entonces, se creó un worker que cuenta con dos web scrapers que extraen la información de cada espectáculo usando una librería de Node llamada Puppeteer (Puppeteer dev), que permite simular las interacciones manuales de los

usuarios con las páginas web. A medida que se extrajo la información, la misma fue llevada al siguiente formato estándar:

```
{
        title: String
         category: String
         dates: [Date],
         description: String,
         image: String,
         theater: String,
         pageURL: String,
         prices: Int
}
```

Para lograr esta estandarización, se incluye en cada scraper una etapa de preprocesamiento de los datos extraídos, para realizar algunas de estas posibles acciones:

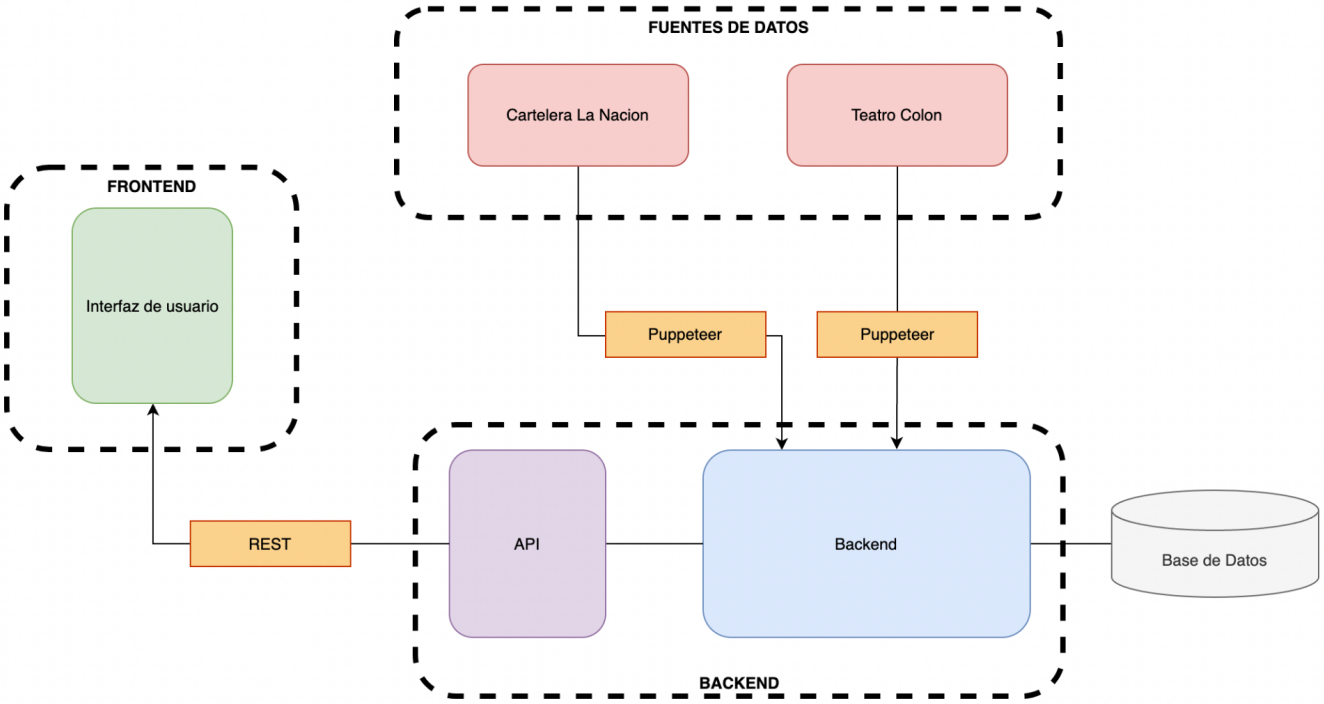
- **Modificar el tipo o formato de un campo**: esto se aplicó para las fechas y los precios, llevando los strings a los valores deseados. En particular, para las fechas, se crearon dos parsers capaces de reconocer el formato del string de cada una de las fuentes de datos.
- **Completar información faltante**: un ejemplo de esto es que las obras del Teatro Colon no tienen en ningún lado de su página el nombre del teatro, pero es posible inferir que la ubicación es el Teatro Colón por el sitio web en sí.
- **Identificar la información no disponible**: la información que se muestra para cada espectáculo puede no estar completa. En algunos casos, las fechas o los precios no están disponibles, o no se muestra una sinopsis en la página. En estos casos, la información se guardó como nula.

Para cada obra, entonces, el worker extrae los datos, hace el proceso de integración de los datos, y guarda los resultados en una base de datos SQL, para la cual se utilizó el manejador de bases de datos Postgresql. El worker corre los scrapers de forma diaria mediante un cron job, que fue configurado teniendo en cuenta dos factores principales: que tan seguido es necesario refrescar los datos para mostrar la información correcta y que tanto tiempo toma que se corran por completo ambos scrapers. Respecto al primer punto, como es poco probable que las funciones cambien mucho a lo largo de un día, se podrían refrescar los datos una o dos veces por día, preferiblemente cuando hay pocos usuarios. En este caso se configuró para que se haga una vez por día, pero esto se podría ajustar según sea necesario. Luego, respecto al segundo punto, los scrapers toman alrededor de una hora en extraer la información de las páginas. Esto se puede usar para decidir cómo se quiere implementar la llegada de nueva información, que puede estar repetida: ya sea eliminar todos los datos de la base, e ir insertando datos nuevos, o actualizar las filas ya insertadas y crear nuevas a medida que haga falta. En el caso de este trabajo, se decidió que la mejor opción sería actualizar las filas existentes en caso de que haya cambios en espectáculos ya existentes, y agregar filas nuevas

para los espectáculos nuevos cada vez que se corre el worker. Para esto, se ingresaron índices en la base de datos en ciertos campos de los espectáculos, para poder reconocer las obras entre ellas. Se implementó adicionalmente, un servicio de limpieza de la base que elimina diariamente los espectáculos de fechas ya pasadas de forma a no tener información desactualizada en la página. La ventaja de esta implementación es que se evita tener downtime en la aplicación durante las actualizaciones del worker y esto permite entonces que el worker pueda correr durante el día si llega a ser necesario.

La plataforma implementada cuenta luego con un backend implementado en Node, que ofrece una API REST, que permite obtener la información de los espectáculos, y usar los distintos filtros y órdenes implementados por el sistema. Esta API es consumida por frontend, implementado en React, que es con lo que interactúa el usuario final.

En suma, la arquitectura de la plataforma implementada se puede ver en la diagrama siguiente:



Funcionalidades principales

La primera vista que muestra la aplicación es un buscador de espectáculos al cual se puede ingresar de forma opcional un rango de fechas. En caso de no ser seleccionadas, se buscarán todos los espectáculos disponibles en la base de datos. La idea detrás de esto es que el usuario pueda buscar desde un inicio solo las fechas en la cual va estar de viaje.

Luego de seleccionar esto, el usuario es dirigido a una lista de todas las obras que están dentro del rango de fechas definido (o todas las disponibles). Las mismas están paginadas, para evitar pedidos muy grandes al backend y para mejorar la experiencia de usuario.

Si el usuario desea filtrar los resultados por título, categoría o por el teatro a visitar, puede realizarlo escribiendo sus preferencias en los campos que aparecen encima de la tabla.

Por defecto, los resultados están ordenados alfabéticamente por título de forma ascendente, pero simplemente haciendo click en los distintos cabezales de la tabla se puede cambiar la dirección y el parámetro de ordenamiento, estando disponibles: título, categoría y teatro

Por último, vale la pena mencionar que, para permitir que los usuarios utilicen la plataforma de forma cooperativa, al almacenar la información sobre los parámetros de búsqueda en la URL, es posible compartir los enlaces para que distintos usuarios obtengan los mismos resultados en sus dispositivos.

Principales desafíos

El principal desafío de la implementación de esta aplicación fue la creación de los scrapers, en particular el scraper de Plateanet que se terminó no usando. Esto es debido a que, para cada fuente de datos distinta, se tuvo que analizar la estructura de la página para poder realizar el scraper, llevando a soluciones que varían según cada página. En particular, en el caso de Plateanet, no había selectores que identificaran fácilmente cada elemento, lo que dificultó el desarrollo. Además, la página tenía tiempos de espera muy altos que llevó a que el scraping tomará alrededor de tres horas, lo que dificultaba las actualizaciones de la base. Para

solucionar esto, se implementó entonces una mejora sobre los scrapers que consiste en dividir los espectáculos a extraer en grupos y hacer que Puppeteer trabaje de forma paralela en los grupos en distintas pestañas. Esta mejora se mantuvo luego de cambiar la fuente Plateanet por La Nación, pero tiene sin embargo un problema: cuantas más pestañas se abran, más lento funciona Puppeteer en cada pestaña. Luego de realizar pruebas, se determinó que la cantidad máxima de pestañas por sesión se debería limitar a tres, y se configuró la funcionalidad para que esta cantidad se pueda ajustar usando una variable de entorno. Tener que descartar Plateanet debido a las restricciones de uso agregadas a la página también llevó a que la etapa de creación de los scrapers se alargara, aunque la implementación del scraper de La Nación fue considerablemente más simple.

Otro desafío se presentó al momento de la integración de los datos, ya que se tuvo que realizar un trabajo para uniformizar los formatos. Adicionalmente, en los casos donde había información faltante, se tuvo que determinar uno por uno si la misma se podría completar por contexto o si se tenía que dejar como no disponible.

Trabajo a futuro

El estado alcanzado del proyecto cuenta con las funcionalidades básicas para permitir que los usuarios logren encontrar espectáculos e información simple sobre los mismos. De todas formas existen mejoras a la aplicación que puedan mejorar la experiencia de usuario y que podrían aplicar más conceptos vistos en el curso.

1. Agregar más fuentes de información para espectáculos. Las páginas utilizadas como fuente logran abastecer la aplicación con un gran número de espectáculos, pero existen más páginas que, si son integradas, pueden ampliar la base de datos día a día.
2. Mejorar la información sobre un mismo espectáculo. Al existir varias páginas sobre eventos, no es difícil encontrarse con varias publicaciones sobre un mismo espectáculo en distintas páginas. Se podría trabajar en realizar una integración de datos de varios sitios para complementar la información que estos puedan llegar a tener. Tal vez alguna página se focaliza en dirección y guionistas, mientras otra tiene más información sobre el elenco arriba del escenario.
3. Diversificar a otro tipo de actividades, por ejemplo museos o institutos culturales. El teatro es uno de los principales entretenimientos en Argentina, pero existen otro tipo de eventos culturales a los que también se tiene fácil acceso (Agenda Cultural Federal).
4. Manejo de usuarios o implementación de recomendaciones. Para el usuario casual, la versión actual cumple con sus cometidos, pero para un visitante conocido se podría implementar un sistema de recomendaciones basado en elecciones previas o mismo en selección del usuario según preferencias del mismo. Agregando el manejo de usuarios en la plataforma podría incluso brindarnos más información sobre el público y sus preferencias.
5. Uso de índices invertidos para mejorar la búsqueda usando Algolia o Elastic Search. Actualmente, las búsquedas que realiza la aplicación utilizan solamente las herramientas brindadas por el gestor de base de datos (Postgresql) y el ORM utilizado (Sequelize). Integrando con algún motor de base de datos que se base en índices,

como Algolia o Elastic Search, se podría mejorar la experiencia de búsqueda y reducir en gran manera los accesos costosos (con muchos filtros y indexaciones) a la base de datos de la aplicación

Conclusiones

En términos de conclusiones, el proyecto permitió que aprendamos sobre tecnologías como Puppeteer, sus principales beneficios y desventajas. La adaptabilidad de las variaciones de las fuentes fue un punto clave lo cual requirió una investigación personalizada para cada una de ellas. La integración y gestión de los datos presentó una dificultad que supimos superar sobre todo a la hora de trabajar con información faltante. Cabe destacar, que como en este proyecto se realizaron scrapers para los cuales no se pidió permiso a los dueños del sitio original, la solución queda vulnerable a problemas sujetos a cambios inesperados en los sitios que se están scrapeando. Es entonces mejor, si existe la posibilidad, utilizar APIs públicas o sitios con los cuales se acordó que se realizará el scraping para llegar a soluciones más estables que eviten este riesgo.

Estos desafíos nos permitieron entender el curso desde un modo práctico y estar preparados para futuros proyectos similares.

Referencias

Parks, Ken. “Turistas uruguayos gastan récord de US$1.000 millones en Argentina.” *Bloomberg*, 2023, https://www.bloomberglinea.com/latinoamerica/argentina/turistas-uruguayos-gastan-record-de-us1000-millones-en-argentina/.

“Cartelera de Teatro de Argentina - LA NACION.” *La Nación*, https://www.lanacion.com.ar/cartelera-de-teatro/proximos-estrenos. Accessed 7 November 2023.

“Temporada | Teatro Colón.” *Teatro Colón* |, https://teatrocolon.org.ar/es/temporada. Accessed 7 November 2023.

Puppeteer | *Puppeteer*, https://pptr.dev/. Accessed 7 November 2023.

“Todo el teatro para vos.” *Plateanet* | *Todo el teatro para vos*, https://www.plateanet.com/lista-obras. Accessed 8 November 2023.

“Agenda Cultural Federal.” *Ministerio de Cultura*, https://www.cultura.gob.ar/agenda/. Accessed 11 November 2023.