Recuperación de Información y Recomendaciones en la Web

Buscador de juegos de caja

Grupo 14
Emiliano Botti - 4.670.479-1
Francisco Cabrera - 4.904.973-6
Felipe Facio - 5.042.799-9
José María Souto 5.129.801-0

Índice

Introducción

El hobby de los juegos de caja se ha vuelto cada vez más popular en estos últimos años, pero en general hay mucha gente que aún no está metida en el nicho y por lo tanto no sabe dónde comprarlos. En esta tarea buscamos brindar una plataforma que ayude a los usuarios a poder encontrar juegos (potencialmente) en varios sitios. Para el enfoque de esta solución serán sólo dos fuentes a modo de prototipo.

Enfoque de la Solución

El enfoque de la solución es la de desarrollar un sistema de búsqueda y recuperación de información, que obtenga los datos de los juegos de caja de dos páginas web uruguayas: Sparta Board Games y Montevideo Gaming House (los dos sitios más populares) y los presente al usuario de una forma intuitiva en una aplicación móvil. De este modo, un usuario en búsqueda de un juego de caja puede acceder la app y encontrar la mayoría de los juegos en venta del mercado.

Diseño

A continuación se muestra una imagen de como consistirá la arquitectura del sistema implementado:

La arquitectura consistirá de una aplicación móvil desarrollada en React Native que consuma de un servidor en Ruby on Rails, que usa la gema Kimurai para *scrappear* los productos de los sitios mencionados y ElasticSearch para realizar las búsquedas que le envía el frontend.

Frontend

El frontend fue desarrollado en React Native debido a que la tecnología tiene la facilidad de crear una aplicación tanto para iOS como para Android de forma inmediata, disminuyendo sustancialmente el tiempo de desarrollo y permitiendo a los usuarios de ambos sistemas operativos dar uso del buscador de juegos.

Parte del equipo ya tiene experiencia en esta tecnología y por lo tanto la curva de desarrollo fue chica, permitiendo mayor tiempo a la investigación e implementación de Elasticsearch y Scrapping de las fuentes de datos.

Backend

Ruby on Rails: Utilizamos el lenguaje Ruby junto al framework Rails para crear el servidor web encargado de recibir las requests de la aplicación, scrapear los sitios, procesar y persistir la data, y realizar las bùsquedas solicitadas, retornando la información a mostrar.

La elección fue por su simplicidad y robustez en el desarrollo. Además, varios integrantes contamos con la experiencia, y creemos que es la opción más segura y confiable para implementar el backend.

ElasticSearch: Se utilizó está tecnología para implementar la búsqueda ya que, luego de investigar, consideramos que es la más conveniente para está solución. Ninguno de los integrantes del grupo cuenta con experiencia, pero como mencionamos anteriormente: la elección de las otras tecnologías agiliza el proceso.

Elasticsearch nos brinda la funcionalidad de hacer el autocompletado, búsqueda de palabras enteras, cortadas, o palabras con algún error gramatical, entre otros. Es el servidor el encargado de recibir texto plano de la app y enviarselo a ElasticSearch para recibir resultados de búsqueda complejos. Esta información es procesada por el servidor y enviada al frontend en forma de JSONs, para mostrarle al usuario resultados completos en pantalla.

Implementación

Web scraping es el proceso de extracción de datos de sitios web, usualmente con el fin de presentarlos en un lugar centralizado. Para esto lo que se hace es procesar la data no estructurada, como HTML, para transformarla en datos estructurados los cuales hacen el proceso de almacenamiento y análisis posible. Nosotros realizamos scrapping de las dos páginas web que vamos a utilizar para recolectar la información: la web de Sparta Games y la web de Montevideo Gaming House.

Para conseguir la información sobre los productos de cada web se utilizó la gema anteriormente mencionada, seleccionando los elementos del documento HTML a partir de selectores CSS, e iterando sobre cada uno de los elementos listados. De allí se extrajo: el nombre, el precio, la imagen y el enlace a la página del producto. Además, se mantiene la fuente en forma de ícono para que el usuario reconozca en qué tienda se encuentra el producto buscado.

Sin ahondar mucho en la implementación, algunos selectores CSS de ejemplo fueron:

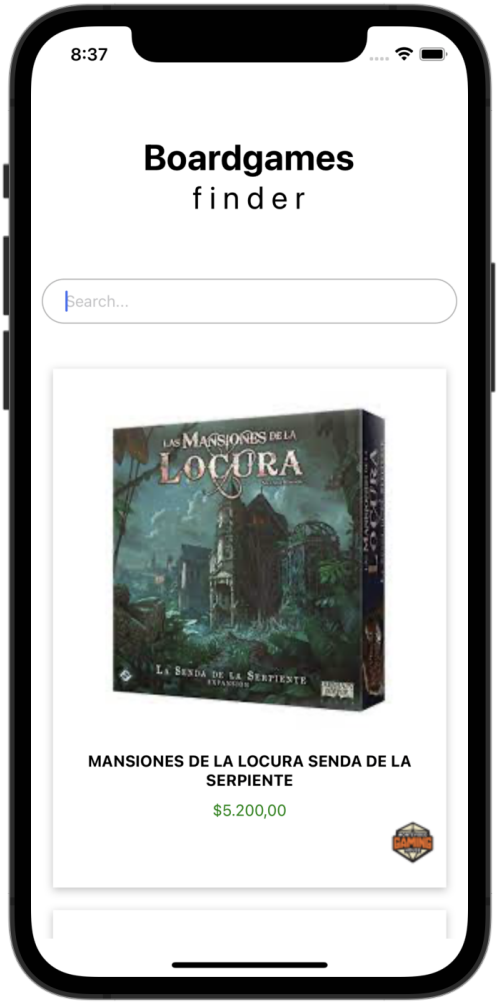
- **product-block-inner**: Clase que identifica cada uno de los productos.
- **h3.product-name**: Nombre del producto seleccionado.
- **span.woocommerce-Price-amount.amount:** Precio sin parsear
- **img.attachment-woocommerce_thumbnail.size-woocommerce_thumbnail**: URL de imagen del producto.

Este scrapping se aplica a cada una de las páginas iterando en ellas, cambiando el índice en "**/page/[indice]**":

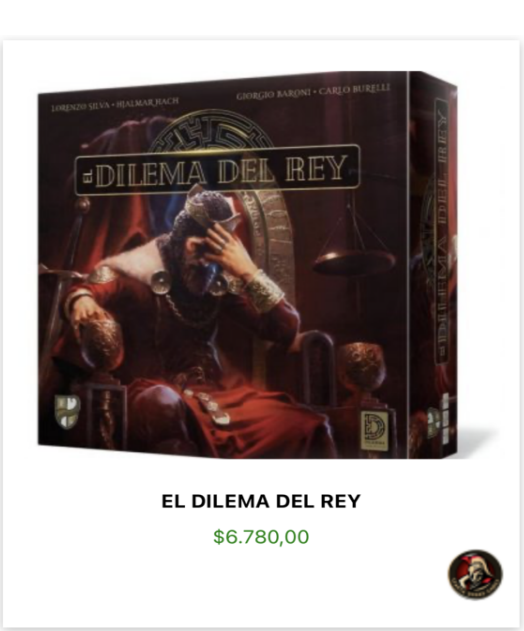
- https://spartaboardgames.com/categoria-producto/juegosdetablero/page/**2/**
- https://montevideogaminghouse.com/categoria-producto/juegos-de-mesa/page/**2**/

Funcionalidades de la app

- **Buscador de juegos** (input con placeholder *Search…*): A medida que el usuario escribe en el input, se envía una request con el texto al servidor y recibe en pantalla una lista de los juegos que corresponden con esa búsqueda.



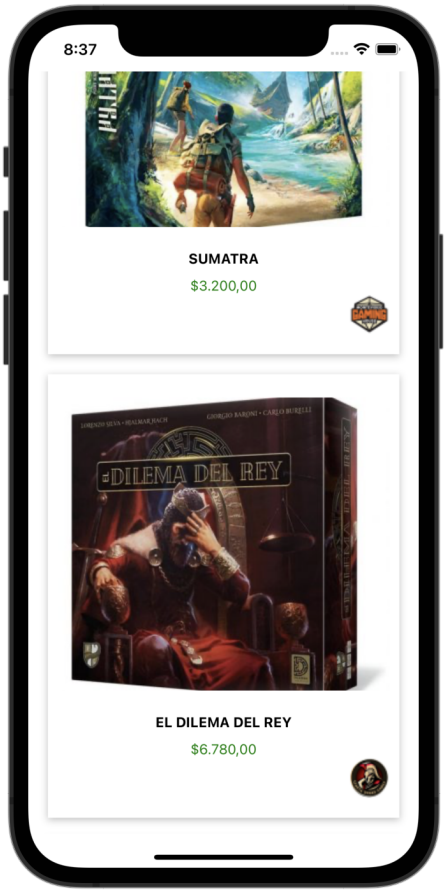
- **Navegabilidad:** Al tocar en un producto específico, se dirige al usuario a la página web del producto seleccionado en la fuente de datos correspondiente. De esta forma, el usuario podrá comprar el producto o ver más detalles en la fuente oficial.
- **Fuente de datos:** El ícono en la esquina inferior derecha del producto indica dónde el usuario puede comprarlo.

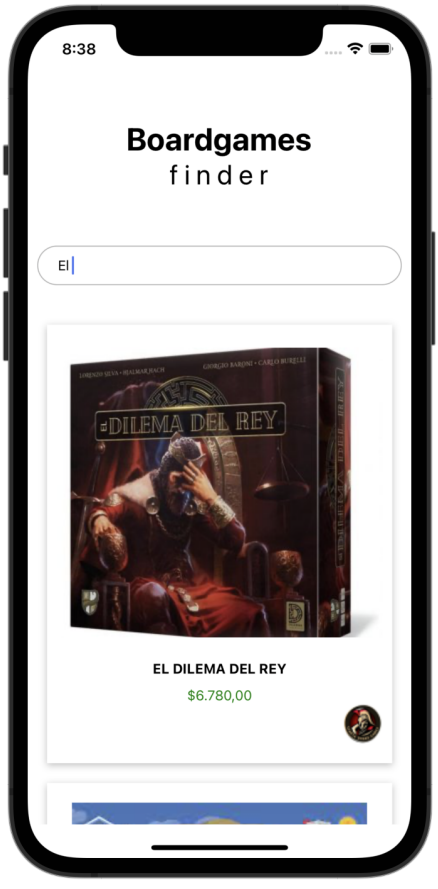


- **Información:** De cada producto se tiene el **nombre** del juego, **imagen**, y **precio** con descuento aplicado (si el producto tiene descuento en la web, le mostramos al usuario el precio final).

Aplicación


8:37
Boardgames
finder
Search...
LAS MANSIONES DE LA LOCURA
LA SENDA DE LA SERPIENTE
MANSIONES DE LA LOCURA SENDA DE LA SERPIENTE
$5.200,00


8:37
SUMATRA
$3.200,00
EL DILEMA DEL REY
EL DILEMA DEL REY
$6.780,00


8:38
Boardgames
finder
El
EL DILEMA DEL REY
EL DILEMA DEL REY
$6.780,00

Conclusiones

Este proyecto nos sirvió para terminar de comprender el contenido teórico del curso, llevándolo a la práctica y entendiendo las posibilidades reales de recuperación de información en la web, así como su almacenamiento y procesamiento, junto a las varias limitaciones y dificultades de éste.

También llegamos a la conclusión de que si bien para aplicaciones y proyectos chicos no es un gran problema, para aplicaciones de mayor tamaño y complejidad, el scrapping no es mantenible y escalable a lo largo del tiempo. Ya que si se quiere agregar nuevas fuentes de datos, se debe implementar desde el comienzo el scrapping por la falta de homogeneidad en la estructura de los datos, y si las páginas web fuente se renuevan o cambian algo de su HTML habría que volver a implementar el código.

Trabajo futuro

- Agregar más fuentes de información: como por ejemplo otras tiendas locales.
- Automatizar la extracción de información de nuevas publicaciones: de forma que se haga periódicamente y de forma incremental en el tiempo.
- Revisar validez de publicaciones creadas: se podría verificar periódicamente si una publicación está activa, según la fuente, o si se aplicó algún descuento o hubo cambio de precio.
- Extender a varios países: por ejemplo hacer scraping en Amazon, o alguna tienda internacional de juegos de caja, y comparar los precios, y se podría calcular cuánto saldría con el envío incluido.
- Utilizar alguna API que complemente los metadatos que tenemos sobre los juegos, para conseguir más información sobre cada juego y poder asignar, por ejemplo, categorías y filtrar los productos por estas categorías.
- Agregar más metadatos para hacer diferentes filtrados en las búsquedas (género, cantidad de jugadores, edades recomendadas, etc.)

Referencias

- Web Scrapping en RoR: https://medium.com/swlh/web-scraper-application-with-ruby-on-rails-864dfaae6270
- Sparta Board Games: https://spartaboardgames.com/categoria-producto/juegosdetablero
- Montevideo Gaming House: https://montevideogaminghouse.com/tienda/
- ReactJS: https://es.reactjs.org/
- React Native: https://reactnative.dev/
- ElasticSearch: https://www.elastic.co/es/
- Kimurai: https://github.com/vifreefly/kimuraframework