FACULTAD DE INGENIERÍA

UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Informe Final

Recuperación de Información y Recomendaciones en la Web

Edición 2023 - Grupo 1

Integrantes	C.I
Gonzalo Abeiro	4.691.963-5
Aleksei Bochkariov	4.966.754-8
Carolina Cortés	5.040.373-7
José de León	4.971.831-7
Federico Vallcorba	5.102.078-4

Índice

Problema

En la actualidad, es común que numerosos proveedores de servicios utilicen la plataforma de redes sociales, como Instagram, como medio principal para promocionar sus productos y servicios. No obstante, se ha observado una limitación sustancial en las capacidades de búsqueda de Instagram, lo que incluye la ausencia de opciones avanzadas de filtrado y búsqueda. Esta carencia de funcionalidades robustas para la búsqueda de contenido y productos dificulta la experiencia de los

usuarios y restringe la eficacia de la promoción y la búsqueda de productos específicos en la plataforma.

Para restringir el dominio del problema, se tomarán las inmobiliarias de Uruguay como proveedores de servicios (alquileres o ventas de inmuebles). Se realizará una investigación previa para definir el conjunto de perfiles de inmobiliarias que conformarán el dominio de búsqueda (corpus) de la aplicación.

Solución planteada

Para resolver este problema, se propone el desarrollo de una plataforma web que actúe como un motor de búsqueda y filtro de inmuebles publicados en Instagram. La plataforma ofrece las siguientes funcionalidades:

- **Filtrado por barrio:** Los usuarios pueden seleccionar de un listado de barrios para buscar inmuebles disponibles en esa área.
- **Filtrado por precio:** Los usuarios pueden establecer rangos de precios para filtrar los inmuebles disponibles.
- **Filtrado por características del inmueble:** Se ofrece la opción de filtrado basadas en características de los inmuebles, como el número de dormitorios, de baños y camas. Los usuarios podrán seleccionar las características deseadas y la plataforma mostrará resultados que cumplan con esos criterios.
- **Filtrado según alquiler o venta:** Los usuarios pueden seleccionar si el inmueble a buscar debe estar en venta o en alquiler.
- **Filtrado según la cuenta de instagram:** Los usuarios pueden seleccionar en qué cuenta de Instagram desean realizar búsquedas.
- **Visualización de información básica de cada publicación:** Cada resultado de búsqueda incluirá información básica de la publicación, como precio, barrio y cuenta.
- **Visualización de información detallada:** Además de poder ver la vista previa de cada publicación, el usuario puede seleccionar una propiedad específica para obtener información más detallada, incluyendo descripciones, dimensiones, dirección exacta y detalles relevantes del inmueble.

Diseño

Para llevar a cabo la solución se definió una arquitectura en capas, especificada a continuación.

- **Capa de Presentación**: Responsable de alojar y servir las páginas web, a los distintos usuarios

- **Capa de Servicios:** Responsable de proveer distintos servicios que permitan llevar a cabo las distintas funcionalidades del sistema. Los servicios definidos en esta capa siguen el estilo arquitectónico REST.
- **Capa de Negocio:** Responsable de alojar la lógica del sistema y resolver la comunicación con la Capa de Datos y la de Servicios
- **Capa de Datos:** Componente que se encarga de almacenar los datos (publicaciones).

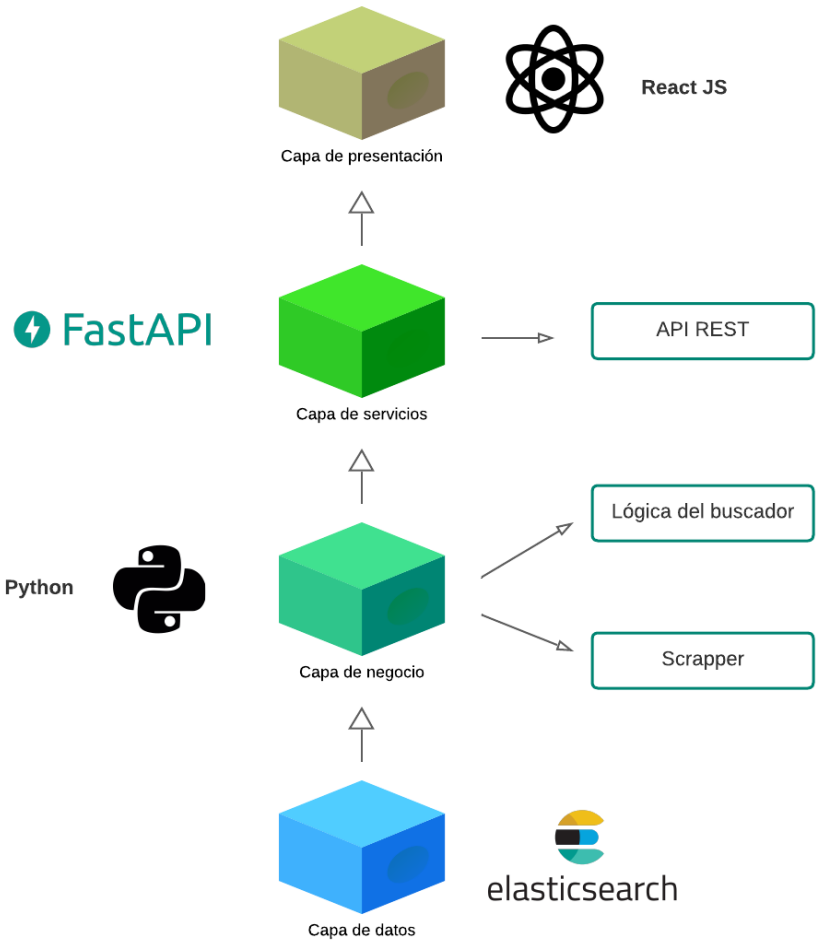

Figura 1: Arquitectura del sistema.

A continuación se mencionan las herramientas utilizadas para la implementación de cada capa:

- **Capa de presentación:** desarrollada en ReactJS [1], con la librería Material UI [2].
- **Capa de servicio y negocio:** desarrolladas en Python [3], utilizando el framework FastAPI [4] para la capa de servicios.

- **Capa de datos:** Los datos se almacenan en Elasticsearch [5], que también se utiliza como motor de búsqueda.

Implementación

Selección de inmobiliarias

Para crear nuestro corpus, elegimos algunas inmobiliarias de las cuales íbamos a tomar publicaciones. Para esto realizamos una pequeña búsqueda de inmobiliarias en la plataforma, seleccionando aquellas inmobiliarias cuyas publicaciones contenían información de los inmuebles en su descripción; ya que algunas inmobiliarias publican la información en las imágenes, por lo que en ese caso tendríamos que hacer algún tipo de procesamiento de imágenes para poder obtener esa información.

A partir de nuestra búsqueda seleccionamos las siguientes inmobiliarias:

- Remax Focus Uruguay
- Lars Inmobiliaria
- Inmobiliaria MVD
- Inmobiliaria Pointer
- LOFT Real Estate
- Sarkissian Inmobiliaria

Sin embargo, para la prueba de concepto solamente utilizamos las primeras 3.

Extracción de publicaciones

Debido a la poca flexibilidad de la API de instagram para poder obtener publicaciones de otros usuarios, es que debimos crear un Scrapper para crear nuestro conjunto de datos.

A continuación se describe el funcionamiento del scrapper para obtener las publicaciones de una inmobiliaria.

- Ingreso a la aplicación en el navegador web, utilizando una cuenta de Instagram ya existente.
- Búsqueda de la inmobiliaria e ingreso al perfil de la misma
- Dentro del perfil de esa inmobiliaria, recorre las publicaciones, quedándose con aquellas que tengan en su descripción alguna de las siguientes palabras: *alquiler*, *venta*, *baño o dormitorio*. Al considerar publicaciones con esas palabras, evitamos quedarnos con publicaciones de la inmobiliaria que no sean de la venta/alquiler de un inmueble.

El scrapper recorre 100 publicaciones de cada inmobiliaria. De cada publicación se almacena la descripción de la misma, la cuenta de instagram de la inmobiliaria y el link a la publicación. Esta información se almacena en un archivo .csv con las columnas Descripción, Link y Cuenta.
Para manipular estos datos durante la ejecución del scrapper, utilizamos la estructura de datos ***dataframe***, junto con la librería ***pandas*** de Python.

Preprocesamiento

Para facilitar el posterior filtrado de las publicaciones, en una primera instancia se realiza un preprocesamiento del archivo CSV que contiene a todas las publicaciones, con el objetivo de agregar columnas al dataset que se correspondan directamente con la información según la cual se realizará el filtrado. Para esto, se definieron expresiones regulares que permitieran obtener estos datos específicos de la descripción de cada publicación. Para esto, se utiliza la biblioteca ***re*** [6] provista por Python, que permite definir expresiones regulares y encontrar matches en los textos que se analicen. Además, se define un diccionario auxiliar "barrios", que contiene los nombres de barrios de Montevideo y Canelones.

Se definen las siguientes 5 expresiones regulares, en la sintaxis de Python:

- `patternDormitorios: (\d+|un|dos|tres|cuatro|cinco|Un|Dos|Tres|Cuatro|Cinco)\s*\w*\s*dormitorios?`
    - Esta expresión se utiliza para identificar la cantidad de dormitorios que tiene un inmueble.
- `patternBarrios: (?:' + '|'.join(map(re.escape, barrios)) + r')`
    - Esta expresión identifica el barrio donde se ubica el inmueble.
    - En esta expresión, la palabra *barrios* hace referencia a una variable que contiene una lista de todos los barrios de Montevideo.
- `patternPrecios: (USD|\$|\$US)\s*(\d{1,3}(?:[,.]\d{3})*(?:[,.]\d{2})?)`
    - Esta expresión identifica el precio del inmueble. Se consideran las posibilidades de que la moneda esté expresada en "USD", "$" o "$US".
- `patternVenta: vende|venta`
    - Esta expresión identifica si la publicación se trata de una venta del inmueble.
- `patternAlquiler: alquila|alquiler`
    - Esta expresión identifica si la publicación se trata de un alquiler del inmueble.

Utilizando estas expresiones regulares, se recorren todas las publicaciones, y se buscan coincidencias en la columna "Descripción" del Dataframe.

En el caso de los precios, se contempla la posibilidad de que la publicación presente varios precios (por ejemplo, una publicación puede presentar el precio de alquiler y el precio de gastos comunes). En estos casos, se selecciona el mayor precio. Además, en el caso de ser un precio en dólares, se

realiza la conversión a pesos uruguayos (utilizando como tasa de conversión 1 USD = $40, por simplicidad). De este procesamiento se almacenan dos columnas, “PrecioStr” que contiene todo el String que representa el precio (por ejemplo, “USD 115.000”), y la columna “PrecioInt”, que contiene únicamente el valor numérico, convertido a pesos.

Los resultados de estas coincidencias se almacenan en las columnas “Dormitorios”, “Barrio”, “PrecioStr”, “PrecioInt”, “Alquiler” y “Venta”, que se agregan al Dataframe original y se guardan como un nuevo archivo CSV con el nombre “data_procesada.csv”.

En el caso de Dormitorios, Barrio y Precio, si la expresión regular no encontró un match, se almacena el valor nulo de Python “None”. Para los casos de Alquiler y Venta, que se almacenan como valor booleanos, el valor nulo se reemplaza por “false”.

Motor de búsqueda

Además de todas las opciones de filtrado mencionadas anteriormente, la búsqueda admite un campo de texto. Este campo de texto es utilizado para la búsqueda sobre la descripción en dos modalidades: búsqueda tradicional y búsqueda semántica.

La búsqueda tradicional se basa en encontrar ocurrencias "exactas" del texto en la descripción del post. La desventaja de esta búsqueda es que no tienen por qué ocurrir las palabras exactas que se buscaron en el texto, sino que podrían aparecer sinónimos o distintas conjugaciones de las palabras. Por lo tanto, al realizar la búsqueda por ocurrencias exactas, se puede obtener un resultado vacío, a pesar de que haya textos que tengan lo buscado escrito de otra forma. Para los campos de texto, Elasticsearch permite definir un analizador, el cual potencia notoriamente el poder de búsqueda tradicional. Al analizador se le pueden pasar distintos parámetros, entre los que destacan las "stop words" y el "Stemmer". Las "stop words" son palabras comunes del idioma, que no aportan a lo que es la búsqueda, por ejemplo conjunciones, artículos o preposiciones. De esta forma, se pueden omitir, para que solamente se tengan en cuenta las palabras que son relevantes para la búsqueda. Por su parte, el "Stemmer", que en inglés significa "raíz" u "origen", lo que hace es justamente llevar las palabras a su raíz, y compararlas en su forma más básica. Por ejemplo, sin un Stemmer, las palabras lavado y lavandería no generarían una coincidencia, porque son distintas, mientras que utilizando un Stemmer sí, ya que comparten la misma raíz.

La búsqueda semántica utiliza fuertemente los vectores semánticos, conocidos como embeddings, para su funcionamiento. Estos embeddings son vectores matemáticos que representan el significado de un determinado texto. Previamente a la carga en Elasticsearch, se calculó el embedding correspondiente a cada posteo, y luego se cargó este embedding junto a la información del post en Elasticsearch. De esta forma, cada vez que el usuario hace una búsqueda por texto, se genera un embedding correspondiente al texto de búsqueda, el cual se usa para encontrar los "k" vecinos más cercanos a dicho embedding mediante el algoritmo "knn". Estos vecinos (posteos) son los que más se asemejan semánticamente a la búsqueda del usuario.

Estos dos tipos de búsqueda se combinan, de forma tal que primero se busca por el método tradicional, y luego se complementa con la búsqueda semántica en caso de que la búsqueda tradicional no haya retornado tantos resultados como los esperados.

Interfaz de usuario

A continuación se presentan algunas imágenes de la página web desarrollada.
En la Figura 2 se puede ver la página principal del buscador, donde se listan todas las publicaciones en el sistema (con paginado), mostrando información básica de la misma, como la cuenta, si está en Venta o en Alquiler, el precio, el barrio y la cantidad de dormitorios.

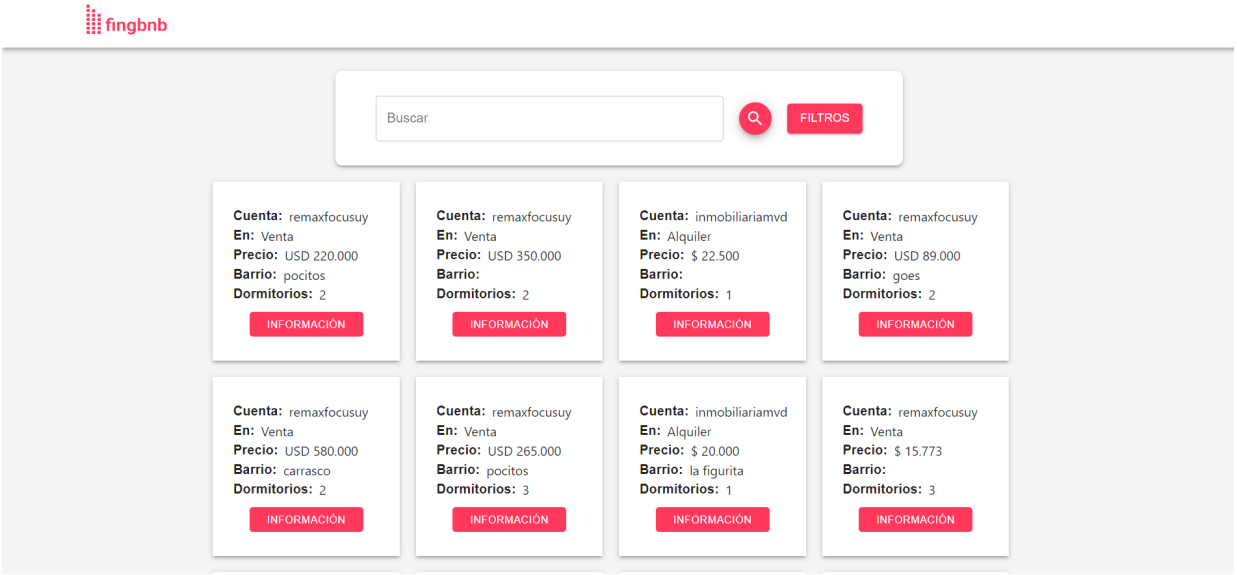

Figura 2: Página principal del sistema.

Por otro lado, si se selecciona el botón “Información” de alguna publicación se puede ver información más detallada de la misma. En particular, se muestra la descripción de la publicación de Instagram asociada a ese inmueble.

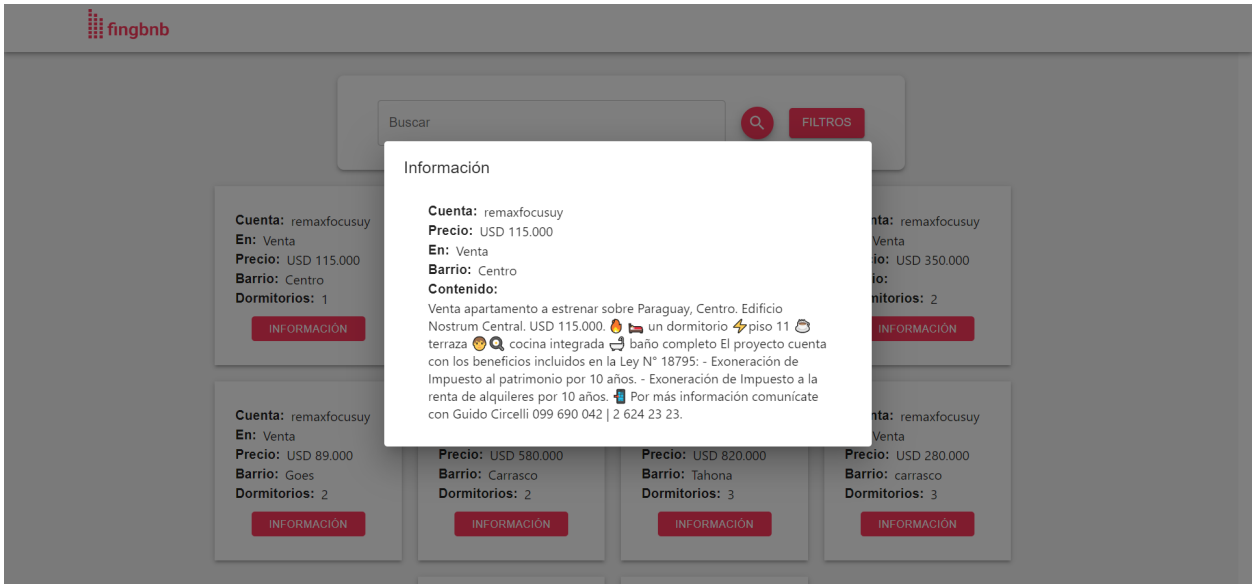

Figura 3: Información detallada de una publicación.

Al seleccionar el botón “Filtros”, al lado de la barra de búsqueda, se puede observar la pantalla de selección de filtros (Figura 4).

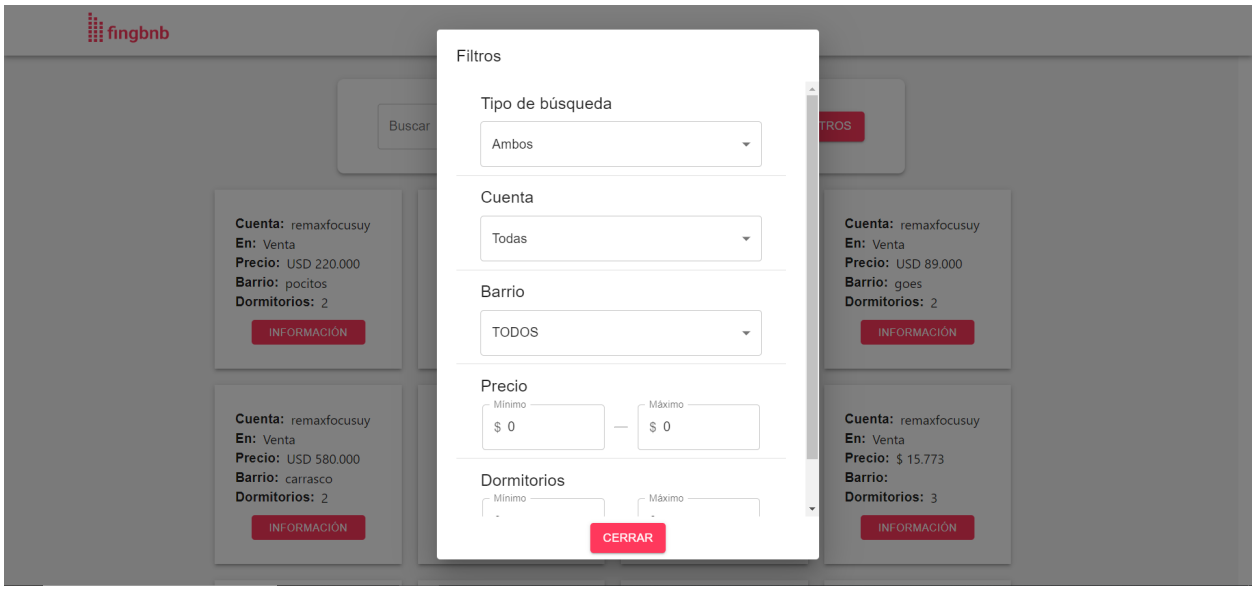

Figura 4: Selección de filtros

Por último, presentamos un ejemplo de búsqueda. En la Figura 5 se eligen filtros para el campo “Tipo de búsqueda” y “Cuenta”, seleccionado “Alquiler” y “remaxfocusuy” respectivamente.

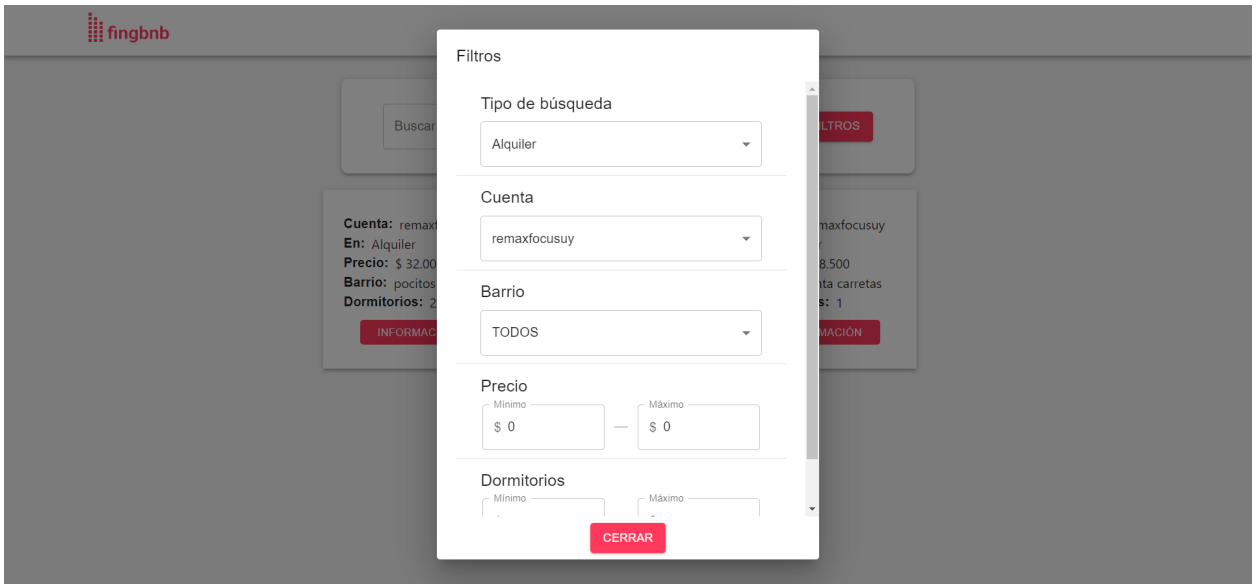

Figura 5: Ejemplo de filtros de búsqueda.

Finalmente, en la Figura 6 se pueden ver los resultados de la búsqueda, donde se obtuvieron todas las publicaciones de la inmobiliaria “remaxfocusuy” correspondientes a alquileres de inmuebles.

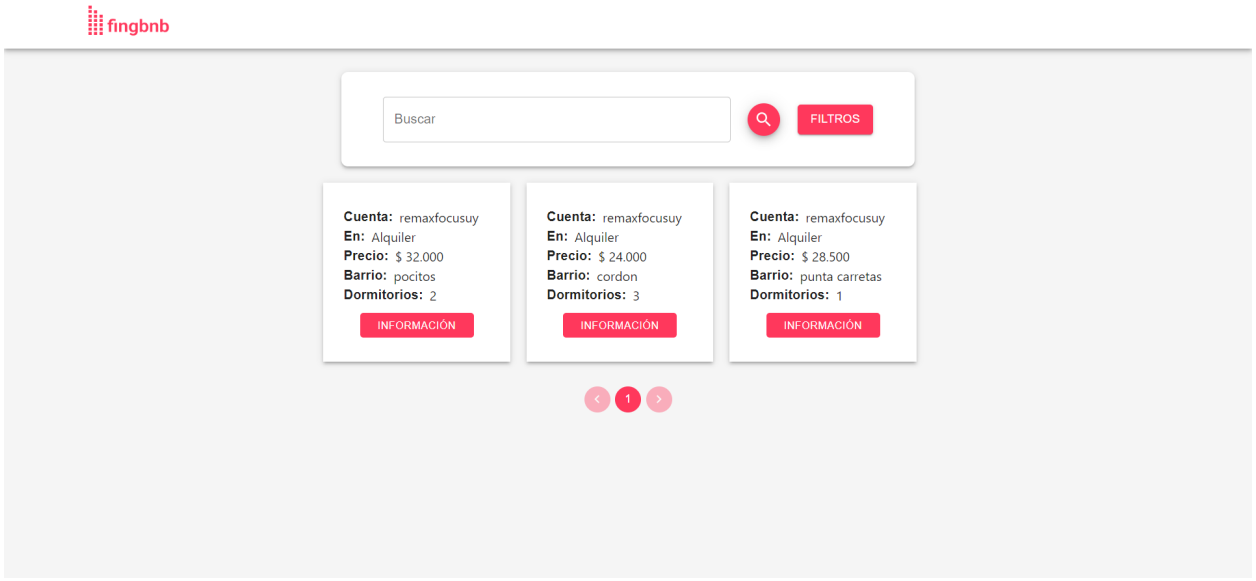

Figura 6: Resultado de búsqueda de ejemplo.

Conclusiones

Logramos desarrollar una prueba de concepto del motor de búsqueda, donde el usuario puede realizar búsquedas sobre inmuebles que pueden estar en alquiler o en venta, con ciertos filtros como precio, barrio, a partir de datos extraídos de publicaciones de Instagram.
Esta búsqueda se realiza de forma sencilla y eficiente, facilitando al usuario encontrar su inmueble ideal.

Además, el diseño de la aplicación permite una gran escalabilidad como agregar nuevas fuentes de datos y filtros sin que el sistema se vea afectado en su funcionamiento.

A partir de la prueba de concepto realizada, se puede ver el gran potencial de la red social Instagram como fuente de datos, una vez que se logra procesar su información y mostrarla mediante nuestra aplicación web.

Trabajo Futuro

Existen varias posibilidades y potenciales mejoras para realizar sobre nuestra aplicación. Una de las restricciones principales de nuestra implementación es la cantidad de inmobiliarias consideradas. Por este motivo, se puede considerar la ampliación de fuentes de datos que consume el sistema, ya sea incorporando nuevas cuentas de Instagram de las cuales consumir datos, o accediendo a servicios de otros proveedores.

Por otro lado, también se pueden implementar mejoras al rendimiento de nuestra aplicación. Por ejemplo, se podría implementar un “job” que ejecute la lógica del scraper de manera periódica. Esto evita la necesidad de ejecutar el scrapper manualmente con tal de mantener la información actualizada.

La prueba de concepto tuvo en cuenta una cantidad de filtros reducida, por lo que a futuro se puede aumentar la cantidad de filtros para tener una búsqueda más refinada y por lo tanto más eficiente, por ejemplo: cantidad de baños, aceptación de mascotas, existencia de otras instalaciones como piscina, parrillero, gimnasio, etc.

Con respecto a la lógica de filtrado, se podrían mejorar las expresiones regulares, para hacer un mejor procesamiento de lenguaje y obtener una mayor cantidad de datos de las mismas, de forma más eficiente.

Otra mejora a realizar es sobre el diseño de la interfaz de usuario, para que sea más llamativa y tenga en cuenta aspectos de usabilidad.

Por último, se puede extender el motor para que permita la realización de búsquedas sobre productos y servicios de otros rubros; ya que de momento solo contempla búsquedas sobre la venta y alquiler de inmuebles. Para hacer esta extensión se deben conseguir nuevas fuentes de datos y definir nuevos filtros que se adapten al dominio del nuevo rubro considerado.

Debido a algunas restricciones de seguridad de Instagram, no pudimos descargar las imágenes de las distintas publicaciones. Por lo tanto, como trabajo a futuro se podrían mostrar las imágenes de las distintas publicaciones.

Referencias

[1] https://react.dev/

[2] https://mui.com/

[3] https://www.python.org/

[4] https://fastapi.tiangolo.com/

[5] https://www.elastic.co/guide/en/elasticsearch/client/python-api/current/index.html

[6] https://docs.python.org/3/library/re.html