Proyecto Webir 2015

Grupo 1

Fabián Dávila - 4.845.223-3

Matías Irland - 4.797.039-3

Ana Clara Esponda - 4.504.058-2

Fabricio Gregorio - 4.467.976-0

Natalia Calle - 4.114.601-7

Índice

1. Introducción
2. Tema de estudio
3. Análisis de fuentes de datos
4. Arquitectura y Diseño
5. Herramientas utilizadas
6. Problemas encontrados
7. Solución planteada
    - 7.1 Obtención de datos: Gallito Luis
    - 7.2 Obtención de datos Mercado Libre
    - 7.3 Filtrado y procesamiento de la información
    - 7.4 Api Correo
    - 7.5 Presentación Web de la solución
8. Casos interesantes de prueba
9. Análisis de resultados
10. Trabajo a futuro
11. Conclusiones
12. Referencias
13. Anexo. Manual de usuario

1. Introducción

En vista de que los sitios web más importantes del mercado inmobiliario tienen grandes carencias al realizar las búsquedas y mostrar la información al usuario, nos planteamos la necesidad de crear una web que optimice las búsquedas según la zona geográfica, aprovechando los beneficios y características de la zona en la cual se encuentra el inmueble para destacar los resultados más adecuados a las necesidades del usuario.

2. Tema de estudio

Como tema de de estudio se escogieron la extracción de información de la web de forma automatizada utilizando tecnicas de Web Scraping, la cual es analizada, filtrada para luego utilizar técnicas de georeferenciación
para localizarlas en el mapa.

3. Análisis de fuentes de datos

Se realizó un fuerte análisis de distintas fuentes de datos de inmuebles, para modelar información a recolectar, realizar un análisis semántico y evaluar ventajas y desventajas de las mismas.

Entre las paginas analizadas se encuentran:

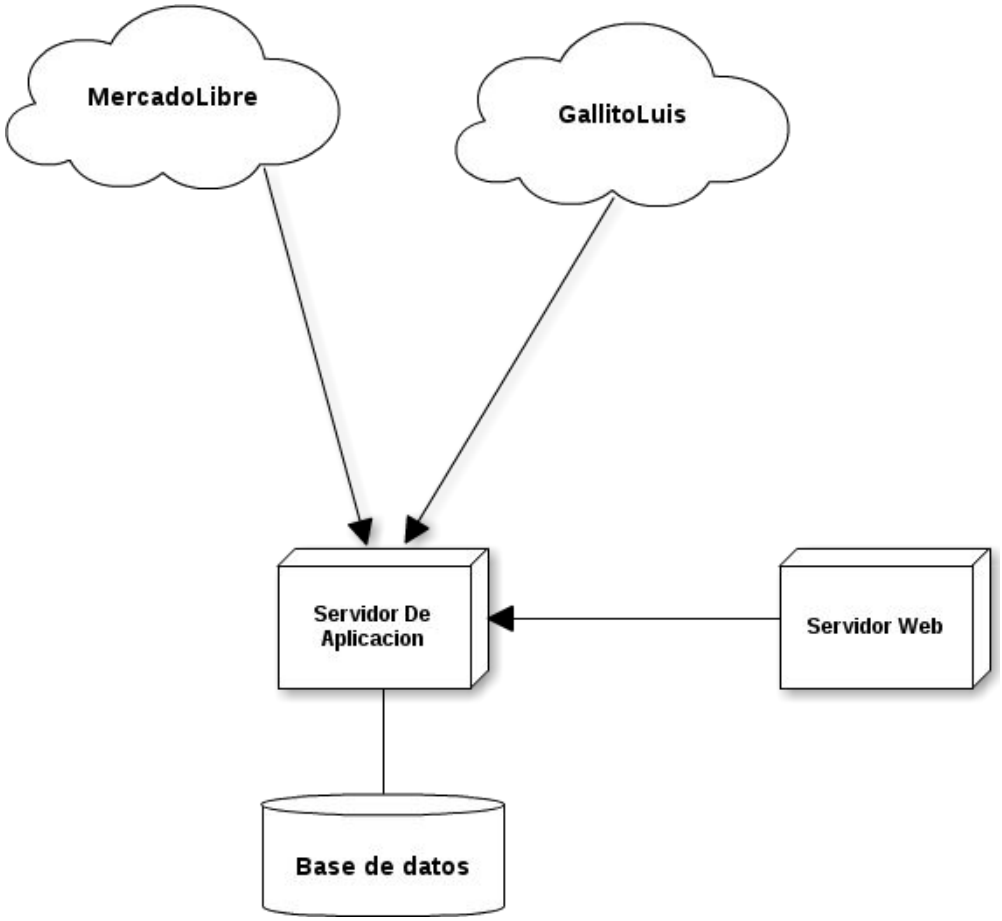
- http://www.infocasas.com.uy
- http://www.mercadolibre.com.uy/
- http://www.gallito.com.uy/inmuebles/alquiler
- http://www.buscandocasa.com/
- http://www.acsa.com.uy/index.php?p=home
- http://www.braglia.com.uy/

Se decidió extraer información de Mercado Libre y Gallito debido a que por cuestiones de tiempo no se pudo extraer de todas, y se consideró que eran las que más cantidad de inmuebles brindaban.

Estas páginas poseen la desventaja frente a ACSA o Braglia que la información de los inmuebles son ingresados por distintos usuarios, por lo cual presenta formas muy diversas de mostrar la misma. A diferencia de los inmuebles de las páginas de las inmobiliaria que estos son ingresados por un conjunto de personas que mantienen los sitios siguiendo reglas

internas a la organización, por lo cual la forma de presentar la información de los inmuebles es más universal.

4. Arquitectura y Diseño



Se plantea una arquitectura cliente servidor.
El servidor de aplicación es el encargado de recolectar datos de la web utilizando técnicas de Web Scraping, para esto se conecta directamente con los servidores de Mercado Libre y El Gallito. Para el caso de Mercado Libre consume los datos de una API publicada por ellos mientras que para el caso del gallito los datos son consumidos directamente del servidor web. Luego de recolectar los datos estos son unificados, filtrados y validados, descartando aquellos datos que no poseen una dirección válida.

El servidor de aplicación se comunica con una API del correo para solucionar los problemas de georreferenciación, utiliza un servicio que dado una dirección en formato de texto este devuelve las coordenadas del inmueble. Es de destacar que los datos recolectados son almacenados en una base de datos utilizando JPA, lo que facilita el filtrado de tuplas por alguna característica, por ejemplo el precio. Por último el servidor publica un Web Service para poder compartir los datos recolectados con el servidor Web.

El servidor web es el encargado de mostrarle al usuario de forma amigable y clara los inmuebles. Para esto consume el servicio proporcionado por el servidor de aplicación y y despliega la información en un mapa.

5. Herramientas utilizadas

Se codificó en lenguaje Java utilizando IDE Netbeans y servidor Apache Tomcat. Como software de control de versiones se optó por Bitbucket por ser gratuito para proyectos con hasta 5 participantes.

Para desplegar la información en el mapa se utilizó la librería OpenLayers 3. La misma ofrece una API para acceder a diferentes fuentes de información cartográfica como ser: Web Map Services (WMS), Mapas comerciales (ej., Google Maps, Bing, Yahoo), Web Features Services, distintos formatos vectoriales como por ejemplo iconos, etc.

Se utilizó la API de Mercado Libre para la obtención de datos, la misma es Rest y contiene las respuestas en formato JSON.

Se utilizó JPA para persistir los datos en una base de datos Derby, con el fin de poder simplificar el manejo y filtrado de los datos.

Para el almacenamiento en la nube, realización de informes y trabajo colaborativo se utilizo Google Drive.

Para la extraccion de informacion de inmuebles en el sitio www.gallito.com.uy se utilizo la herramienta Jsoup. Jsoup es una librería Java, que provee una API conveniente para manipular data utilizando los métodos DOM, css y Jquery más usados.

Como se menciona en el punto anterior, para la georeferenciación de las direcciones se usa una Api provista por el correo, que puede también ser visto como un servicio web.
Básicamente esta api lo que hace es devolver direcciones que cumplan con un criterio de búsqueda, las direcciones están compuestas por el nombre de la calle, el número de puerta, manzana, solar, localidad, departamento, código postal, ubicación geográfica (punto en SRID 4326), entre otros.
El servicio se encuentra disponible en:

http://geo.correo.com.uy/servicios/BusquedaDireccion

Los parámetros que recibe son:

- departamento (opcional): Nombre del departamento
- localidad (opcional): Nombre de la localidad
- direccion: Dirección completa.Los formatos en los que se pueden buscar direcciones son (los campos entre “[]” son opcionales):
    - calle [número] [, localidad] [, departamento]
    - calle [número] esquina calle2 [, localidad] [, departamento] (“esquina” también se puede escribir como “esq.” o “esq”)

    - [calle] manzana X solar Y [, localidad] [, departamento] (manzana también se puede escribir como man. o m. y solar como sol. o s.)
    - nombre de inmueble [, localidad] [, departamento]

La respuesta está compuesta por un documento json, una lista con elementos con una estructura como la siguiente y con los valores correspondientes en los campos que corresponda (en caso contrario tienen valor vacío).

- direccion
    - calle
        - nombre: Nombre de la calle.
    - departamento: Nombre del departamento.
    - numero: Número de puerta.
    - error: Descripcion del error en caso de exisitr
    - solar: Datos del solar.
        - nombre: Identificador del solar.
        - manzana: Manzana a la que pertenece el solar.
            - nombre: Identificador de la manzana.
            - localidad: Campo duplicado por compatibilidad hacia atrás.
    - codigoPostal: Código postal del punto encontrado.
    - porVecino: Indica si el punto fue encontrado por cercanía y no por dirección exacta.
    - porLocalidad: Indica que no se encontró la dirección y que la respuesta es una aproximación a la localidad.
    - porEsquina: Indica que la dirección se encontró por esquina y no por dirección exacta.
    - porPuntoNotable: Indica que la dirección se encontró por un nombre de inmueble.
    - porCalle: Indica que no se encontró la dirección y que la respuesta es una aproximación a la calle.
    - puntoX: Coordenada X del punto geográfico.
    - puntoY: Coordenada Y del punto geográfico.
    - srid: Sistema de referencia de las coordenadas puntoX y puntoY.
    - normalizada: Nombre de la calle normalizada.

6. Problemas encontrados

Surgieron varios problemas al unificar datos de distintas fuentes debido a que la calidad de los datos contenidos no es la mejor y el formato no se mantiene en todos los casos.

Se encontraron diferencias semánticas en las fuentes elegidas, por ejemplo en el caso de Mercado Libre se maneja el concepto de ambientes mientras que en el Gallito el de dormitorios.

A su vez se encontraron muchos datos duplicados en la misma fuente asi como direcciones mal ingresadas o abreviaciones de calles incorrectas, lo cual dificulto el parseo y la georreferenciación de las mismas.

La api de Mercado Libre utilizada tiene documentación muy escaza sobre sus métodos y cómo utilizar los filtros de búsqueda requeridos, lo cual requirió de mayores tiempos para el correcto uso de la misma.

Como capa base de OpenLayers se decidió utilizar Google Maps, surgiendo el siguiente problema: Google no publica más sus capas para ser consumidas por OpenLayers directamente, por lo que se tiene que trabajar con dos mapas superpuestos. El mapa de Google se sitúa debajo y el de OpenLayers arriba, y cada movimiento que el usuario hace en el mapa superior, es traducido al inferior.

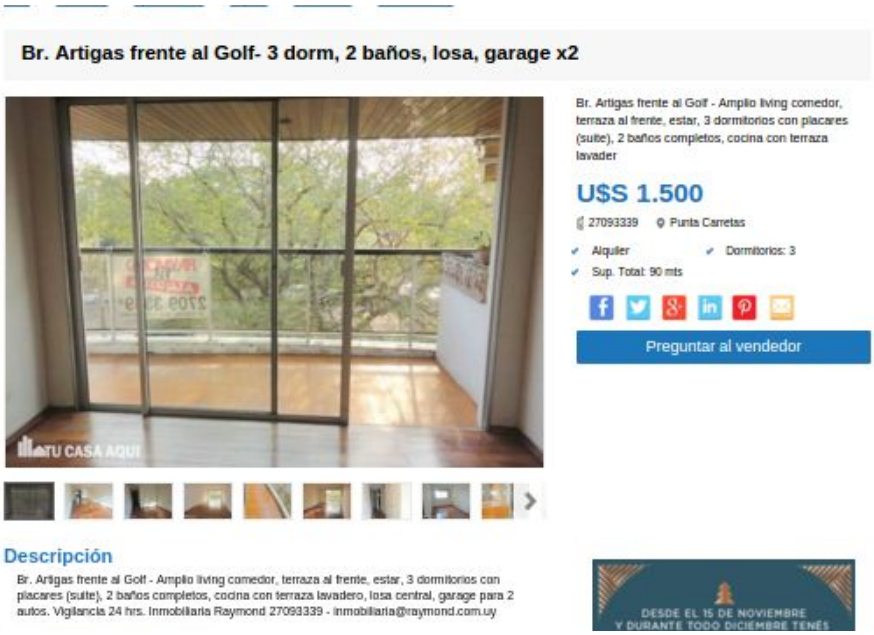
7. Solución planteada

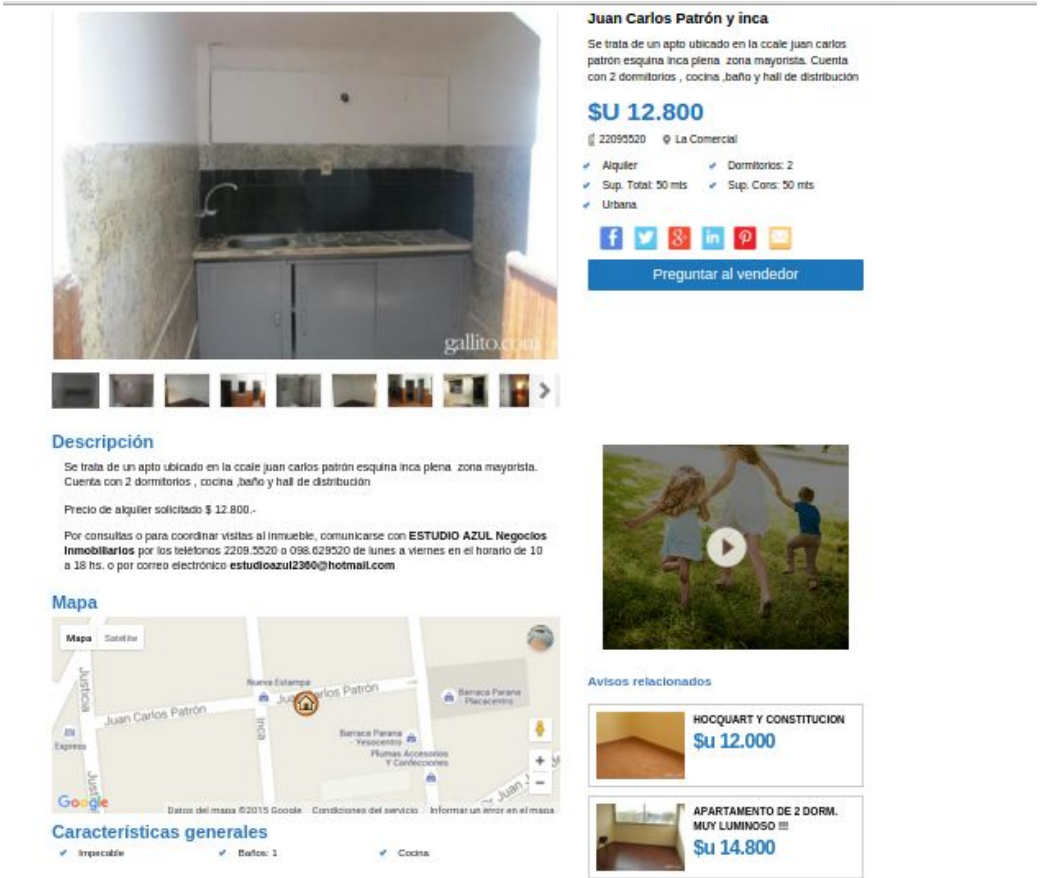
7.1 Obtención de datos: Gallito Luis

Para la obtención de información del sitio http://www.gallito.com.uy/ se utilizó la herramienta JSoup, la cual es un parser HTML que sirvió para poder extrar la información del sitio.

Esta página no posee un campo para ingresar la dirección del inmueble. Por lo cual luego de analizar distintas publicaciones se pudo observar que esta información estaba presente en los siguientes campos:

- Título
- Breve descripción
- Descripción
- Mapa





La extracción de esta información se modelo de la siguiente manera:

1. Mapa: se consumía la dirección del javascript asociado a este. Para poder realizar esto, se debió parsear el HTML a string y después acceder a la parte donde se definía el mapa, dado que los selectores JSoup no reconoce el Javascript asociado al mismo.
2. Título: Se obtuvo el título de la página utilizando el selector css `"div.titulo"`.

3. Breve Descripcion: Se obtuvo la breve descripción utilizando el selector css `"p[id=descripcion]"`
4. Descripción: Se utilizó el selector `"p[id=descripcionLarga]"`

En los pasos 2, 3, 4 luego de obtener el texto del elemento html, se utilizaba la función `AddressHelper.findAddress(texto, false);` si el resultado era null se continuaba con el siguiente paso.

7.2 Obtención de datos Mercado Libre

Para obtener los datos de Mercado Libre se analizaron dos métodos distintos.
Primeramente se realizo el web scraping con la extension de Chrome Scraper 1.7. La misma funciona ingresando expresiones regulares para cada campo que se quiere paresear, extrayendo la información a un archivo csv. Con esta herramienta se pudo obtener los datos pero resultaban de muy mala calidad, y algunos datos eran inconclusos.

El segundo método utilizado fue la api de Mercado Libre Developers, la misma funcion en modo Rest, y encapsula las respuestas en formato json. Con esta herramienta se obtuvieron mayor cantidad de datos y en mejor formato, por lo cual se decidió utilizar esta ultima opcion.

7.3 Filtrado y procesamiento de la información

Tanto para los datos de Mercado Libre como los del Gallito se utilizó la función anteriormente mencionada “findAddress”, que que tiene como objetivo encontrar la dirección en un texto dado. Para el caso del gallito esta función fue muy utilizada porque no había un campo específico donde encontrar la dirección, por lo que se utilizó en los campos que podían contener una dirección. Para el caso de Mercado Libre igualmente se utilizó la función en el campo de dirección porque existían datos erróneos o datos adicionales por ejemplo “cerca de 21 y williman”, la función filtraba y obtenía “21 de Septiembre esquina Williman”.

La función encuentra 4 tipos de direcciones en el siguiente orden.

1. [Calle] y [numero]
2. [Calle] “entre” [calle] “y” [calle]
3. [Calle] [conector y] [calle]
4. [Calle]

El conector y se componen por las palabras "y", "casi", "esq" y "esquina".
Las calles son obtenidas de la base de datos publicos [5].
Una caracteristica importante que posee la función es que acepta una tolerancia a faltas de ortografía por ejemplo la ausencia de tildes y la tolerancia a abreviaciones por ejemplo las de bulevar tales como “bvar”, "bv", "br".

La funcion basicamente sigue el orden anteriormente mencionado, buscando un conector y viendo si las palabras alrededor forman parte de una calle.
Se utilizó un conjunto de palabras vacías para asegurarnos que la calle realmente existe, por ejemplo dada la calle “21 de Setiembre”, si hay un texto que contiene la palabra “de” machea con “21 de Setiembre” pero “de” no es una calle y como pertenece al conjunto de palabras vacías esta se ignora.
Por último las calles encontradas pertenecientes a una dirección son almacenadas en una base estadística que es utilizada para mejorar la precisión de las búsquedas.

Para el caso que se busque únicamente la calle (metodo 4), se definió que se va a buscar únicamente entre las calles que ya fueron utilizadas anteriormente, es decir que se buscaron y encontraron anteriormente para otro anuncio. Para esto se utilizó la base estadística mencionada anteriormente. La razón de esto es que existen calles con nombres que son comunes en un titulo o descripcion de un anuncio, pero que no tienen por que pertenecer a la listas de palabras vacias. Por ejemplo hay un anuncio que decía que el inmueble contaba con pisos de lapacho, pero a su vez en nuestra base de calles existe una calle que se llama lapacho, por lo cual macharia con la calle. Utilizando la base de datos estadísticas, si anteriormente no se utilizó esa calle esta no sería tomada en cuenta como calle y de esta forma eliminando muchos errores. Por desgracia la mayoría de anuncios del gallito contenían únicamente el nombre de la calle como direccion, por lo que utilizando este método muchos de los anuncios no los íbamos a poder tomar en cuenta, pero asegurabamos cierto nivel de calidad y con el paso del tiempo el sistema de estadísticas iba a crecer y los anuncios se iban a poder encontrar.

7.4 Api Correo

Una vez obtenidas las direcciones, tanto desde “El gallito” como desde “Mercado Libre” y normalizadas estas estas con la función “findAddress”, se procedió a llamar a la api del correo.

Si bien esta api recibe 3 parametros (departamento,localidad y direccion) solo fue llamado con el parámetro direccion, que es en definitiva el que nosotros estamos manejando.
La api por cada llamada nos devuelve un json con una gran cantidad de información sobre la dirección, en particular de esa estructura nos interesa destacar el campo “error”, y los campos porVecino, porLocalidad, porEsquina, porPuntoNotable, porCalle y los puntos georeferenciados puntoX y puntoY.

En un comienzo solo tomamos como direccion válida, o sea direccion a la cual podemos georeferenciar a las que tenían error nulo, o a las que aproximaban por esquina, por punto notable y por vecino, dejando afuera a las aproximaciones por localidad y por calle ya que las considerábamos aproximaciones muy “brutas” de nuestra direccion a ubicar.
Luego analizando los resultados y los datos de entrada, concluimos que muchas calles son cortas, y que la api usa información de los vecinos (que claro esta, no es tan fina como la aproximación por vecino en sí, ya que esta te ubica con pocos numeros de puerta de distancia), con lo que la aproximación por calle que nos hace es suficiente para que el

usuario tenga una referencia de donde está la vivienda, con lo que dejamos fuera solo la aproximación por localidad.

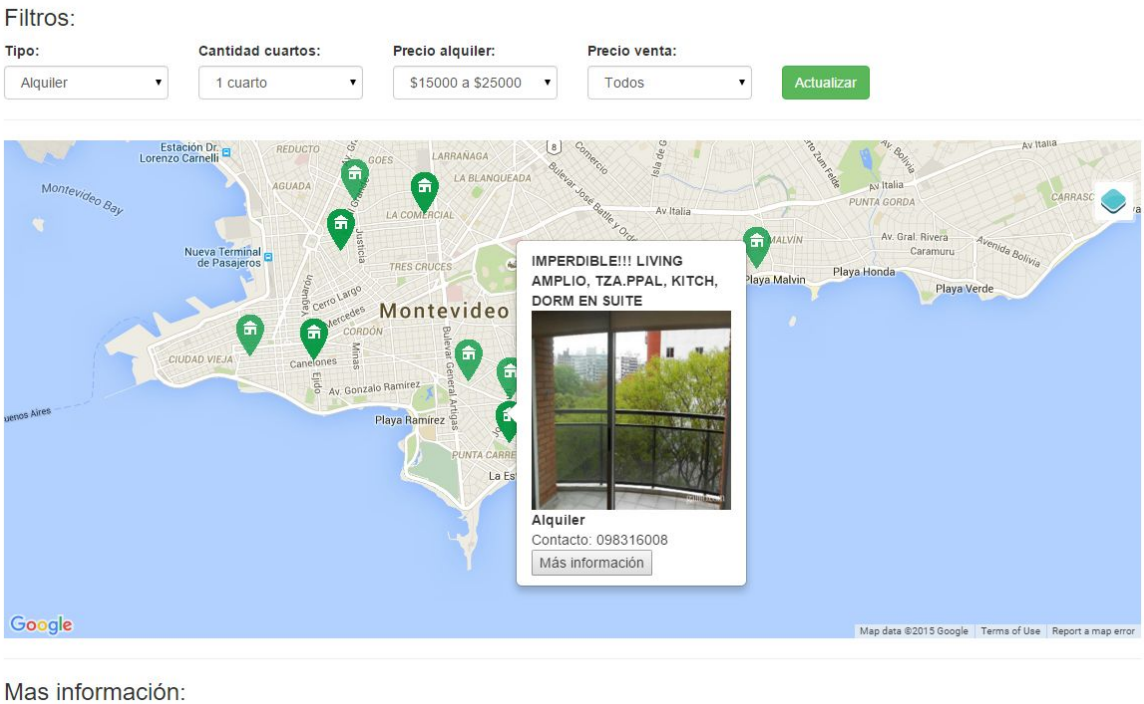
Entonces, si alguna direccion cumple con los criterios mencionados, se obtiene las coordenadas y se guardaban en la base de direcciones, quedando lista para ser consultada y georeferenciada por la aplicación web.

7.5 Presentación Web de la solución

El funcionamiento de la web es el siguiente:

1. El usuario selecciona una serie de filtros como ser cantidad de ambientes, tipo (alquiler o venta), y precio de los anteriores.
2. Se realiza la consulta a la base de datos y se despliega cada inmueble obtenido en el mapa según su longitud y latitud utilizando OpenLayers. Cada inmueble es representado por un ícono que al presionarlo despliega un popup con título, foto, si es alquiler o venta, teléfono de contacto y un botón para obtener más información.
3. Si se oprime el botón ‘Más información’, debajo del mapa se muestran más datos como ser la dirección y a futuro se pensaba mostrar una galería de fotos y un link al sitio original del cual se obtuvo el inmueble.

Originalmente tuvimos la intención de obtener las paradas de ómnibus de montevideo directamente del GeoServer de la intendencia de Montevideo y agregarlas como otra capa. Luego esto fue descartado porque el mapa de Google ya contaba con dicha información.



8. Casos interesantes de prueba

Desde el punto de vista del usuario se consideró interesante el uso de filtros combinables en los cuales se puede incluir cantidad de cuartos, tipo y precio en una misma consulta, ahorrando tiempo al usuario final. También se incluyen como trabajos a futuros otros casos interesantes al usuario como pueden ser la inclusión de más capas de informacion (paradas de omnibus, supermercados, restaurantes, etc).

9. Análisis de resultados

Se realizó un análisis del funcionamiento de la API del correo sobre el total de direcciones parseadas desde “el gallito” y desde “mercado libre.
En una primera instancia se analizó sin considerar las aproximaciones por calle, o sea solo las que tienen una precisión de metros de distancia, este resultado di{o entorno al 45 % lo cual es bastante alentador, ya que casi la mitad de las direcciones parseadas tienen una precisión exacta (metros de distancia) .
Luego, se analizó incluyendo al el filtro “por calle”, recordemos que este es un poco más bruto, ya que considera aproximar a lo largo de la calle en caso de no encontrar el punto, teniendo en cuenta los vecinos y las esquinas, con este filtro, el resultado fue casi del 95 % y tiene un margen bastante mayor que sin este filtro.

10. Trabajo a futuro

El trabajo constó de un prototipo dado que el tiempo fue acotado. Como trabajo a futuro se considera importante atacar los siguientes aspectos:

- Agregar un registro de usuario, con sistema de notificaciones, recomendaciones basadas en búsquedas anteriores del mismo.
- Soporte de calificaciones sobre los postulantes de los distintos inmuebles y usuarios finales de la aplicación.

Con respecto a la calidad y cantidad de la información se consideran grandes mejoras los siguientes puntos:

- Aumentar la cantidad de paginas a partir de las cuales se extraen datos.
- Optimizar algoritmo de búsqueda de direcciones a partir de palabras extraídas de la descripcion titulo o otros campos del inmueble.
- Disponer de más información sobre cada inmueble.

Con respecto al aprovechamiento de la georeferenciación de los inmuebles, grandes aportes pueden ser brindados por los siguientes puntos:

- Brindar consultas geográficas a los usuarios, para entre otras cosas permitir buscar por cercanía a puntos ingresados de interés al usuario.
- Agregar más capas al mapa con diferente información relevante para el usuario (ej., hospitales, escuelas, almacenes, paradas de ómnibus) que no sean ya provistas por el mapa de Google.

11. Conclusiones

Luego de finalizar el proyecto concluimos que cumplimos con los objetivos propuestos, destacándose entre ellos el aprendizaje de técnicas de Web Scraping y georeferenciación que permitieron lograr un prototipo de sistema en el tiempo dado.

Se eligió este proyecto a partir de experiencias personales, las cuales constataron la falta o diversidad de información geográfica de los inmuebles en los sitio.

Este prototipo permite al usuario tener un sitio con información unificada de diferentes fuentes, ubicadas en el mapa. El mismo puede ser de gran utilidad si se avanza en el trabajo a futuro mencionado en el punto anterior.

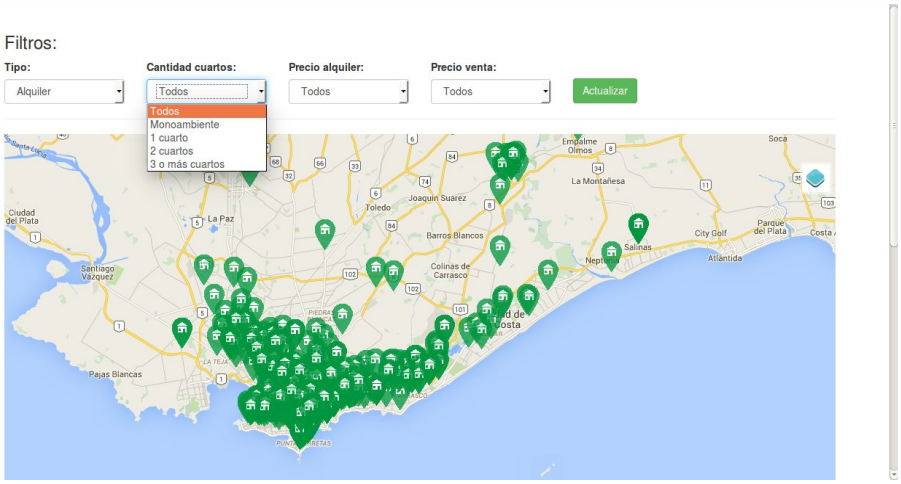
Sobre recolección de datos en la web, se puede concluir que la calidad difiere mucho en cada publicación, pudiendo impactar en la información mostrada por el sitio. Esto se debe a que ambas fuentes de datos elegidas, tienen como fuente de información de las publicaciones a usuarios.

12. Referencias

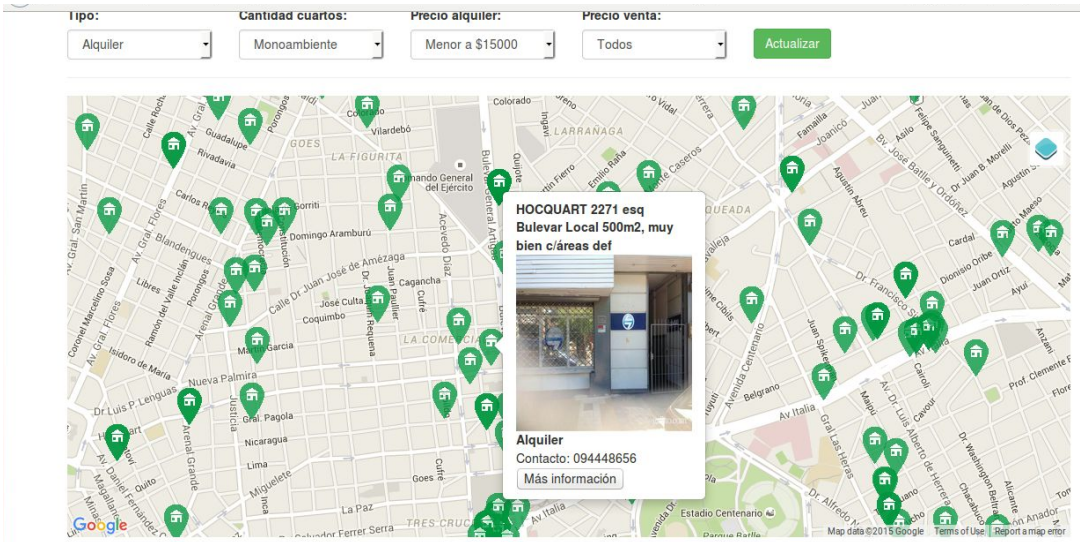
- [1] http://developers.mercadolibre.com/
- [2] http://www.mercadolibre.com.uy/inmuebles/
- [3] http://openlayers.org/
- [4] https://bitbucket.org/
- [5] https://netbeans.org/
- [6] http://tomcat.apache.org/
- [7] http://www.gallito.com.uy/
- [8] https://catalogodatos.gub.uy/
- [9] API - Correo

13. Anexo. Manual de usuario

El prototipo creado es sumamente intuitivo, una vez que se accede a la página esta cuenta con 3 secciones.
La primera es una barra de filtros, la segunda es el mapa con los inmuebles y en la tercera despliega más información sobre un inmueble particular.
Si se quiere utilizar algún tipo de filtro este se selecciona y se oprime el botón actualizar.
Los inmuebles serán actualizados automáticamente.



Navegando en el mapa se puede visualizar cualquier inmueble dando click en el marcador en el cual se desplegará la información básica del inmueble.



Si se presiona el botón “Más información” será redirigido a una sección más detallada del inmueble seleccionado.