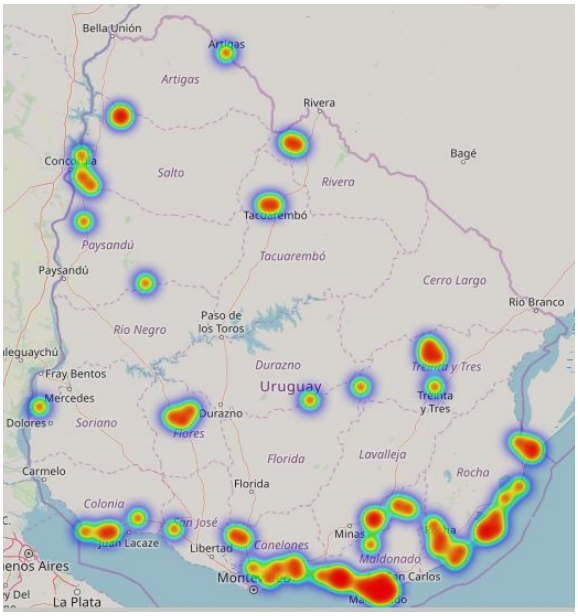
WebIR

Tamara Suárez
Matías Dornel
Bruno Sena

Visualización de precios en hoteles

24 de Noviembre del 2018

Introducción



En la actualidad no existe una herramienta sencilla para que los usuarios de sitios web de hotelería decidan el destino que querrían visitar en sus próximas vacaciones en relación a su presupuesto. Este informe incluye información sobre el proceso de construcción de la aplicación construida desde la ingeniería y estructuración de datos a la presentación para usuarios finales.

Problema

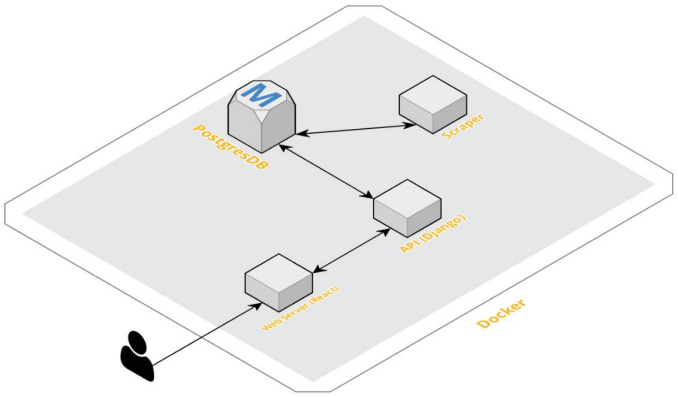
El problema nace de que no hay una forma sencilla de visualizar localidades en oferta de venta/alquiler en sitios como booking.com. Ya que la forma estándar de búsqueda de hospedajes se basa en el uso de filtros por texto y no en la información geográfica de los mismos. Además, es difícil para el usuario poder discernir qué áreas de una localidad cuenta con los precios más bajos (o más altos, en caso de que así lo desee).

Índice

Enfoque de la solución

Las solución al problema consiste en un diseño de un mapa de calor basado en los precios de los hoteles, hospedajes y apartamentos ofrecidos en Booking.com permitiendo el uso de filtros de cantidad de huéspedes, tipo de alojamiento, puntuación. Se extraen datos no estructurados del sitio Booking.com a través de selectores CSS, luego se procesa para ser limpiados y normalizados. Finalmente, se muestra en un mapa incluido en una aplicación web, que permite la interacción de los usuarios con este último.

Diseño



- Scraper: Para extraer los datos de los alojamientos disponibles en el sitio Booking.com se utilizará la biblioteca Scrapy en un entorno Python para procesar los pedidos HTTP al sitio web (utilizando las herramientas de parsing que la biblioteca integra). Este componente extraerá los datos del sitio periódicamente y los almacenará en una base de datos.
- Base de datos: Se utilizará un manejador de bases de datos para persistir aquellos que se extraigan del sitio web (esto a su vez permite almacenar datos históricos del sitio).
- API: Para consultar los datos obtenidos se expondrán un conjunto de endpoints contraídos en el framework Django que posibilitará el uso de filtros.
- Web Server: Se construirá un servidor web que provea de la aplicación que permite al usuario visualizar el mapa y controlar los filtros utilizados.
- Docker: Para facilitar el desarrollo y garantizar la correcta ejecución de la aplicación en distintos equipos se utilizarán contenedores Docker.

Implementación

La implementación de nuestro sistema consiste en dos grandes componentes, por un lado el backend y por el otro el frontend.

Backend

Django

Django es un framework de aplicaciones web gratuito y de código abierto escrito en Python. Un framework web es un conjunto de componentes que ayudan a desarrollar sitios web más fácil y rápidamente.

La meta fundamental de Django es facilitar la creación de sitios web complejos. Django pone énfasis en el re-uso, la conectividad y extensibilidad de componentes, el desarrollo rápido y el principio No te repitas (DRY, del inglés Don't Repeat Yourself). Python es usado en todas las partes del framework, incluso en configuraciones, archivos, y en los modelos de datos.

Para dar soporte al backend, es necesario extraer y recopilar datos desde Booking, para ello se usó Scrapy

¿Qué es Scrapy?

Scrapy es una plataforma colaborativa de código libre que corre en Python para extraer datos de páginas web, usado para una serie de aplicaciones como minería de datos, procesamiento de información o registro histórico.

Scrapy tiene las siguientes características:

- Rápida y poderosa: Escribes las reglas para extraer los datos y Scrapy hace el resto

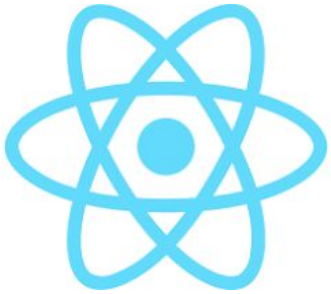
- Fácilmente extensible: Dada su configuración, puede generar nueva funcionalidad sin tener que modificar el código fuente.
- Portable y Pythonico: Está escrito en Python y puede correr en Linux, Windows, Mac y BSD.

¿Qué comprende Scrapy?

- Dado que es un framework, Scrapy tiene una serie de herramientas poderosas para hacer el "scraping" o extraer información de webs de manera fácil y eficiente. Estas herramientas comprenden:
- Soporte para extraer y seleccionar datos de fuentes HTML/XML usando selectores CSS y expresiones XPath, con métodos de ayuda para extraer usando expresiones regulares.
- Una consola interactiva en Python para ensayar los CSS y expresiones XPath para extraer datos, muy útil cuando se construyen métodos propios.
- Soporte para exportar los registros en formatos múltiples como JSON, CSV y XML.
- Soporte para manejar declaraciones foráneas, no estándares y códigos rotos.
- Fuerte extensibilidad, ya que te permite conectar tu propia funcionalidad usando señales, extensiones y pipelines.

Frontend

Por otro lado, para la implementación de la interfaz de usuario (frontend) se utilizó React.



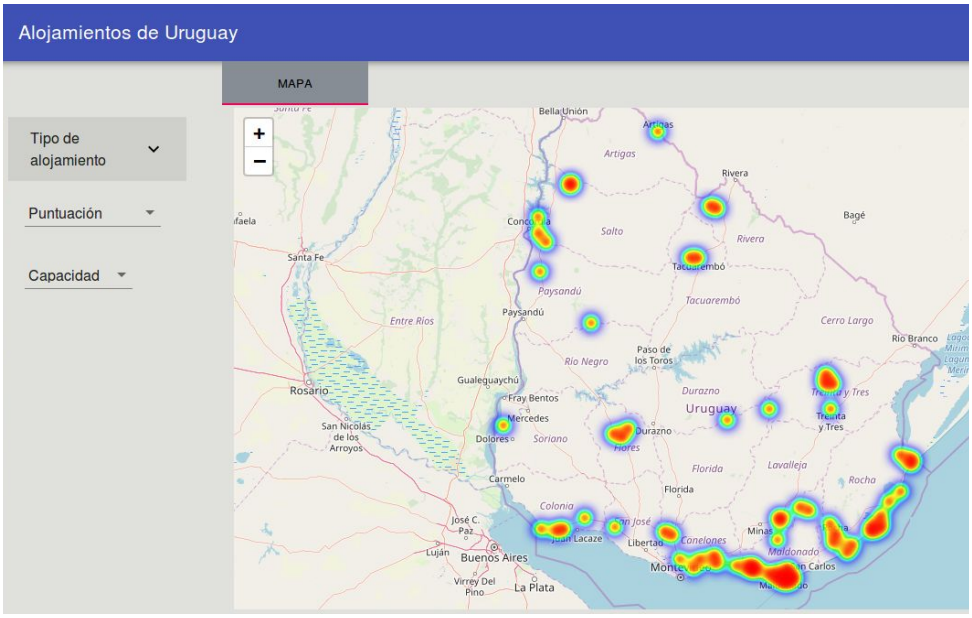
React

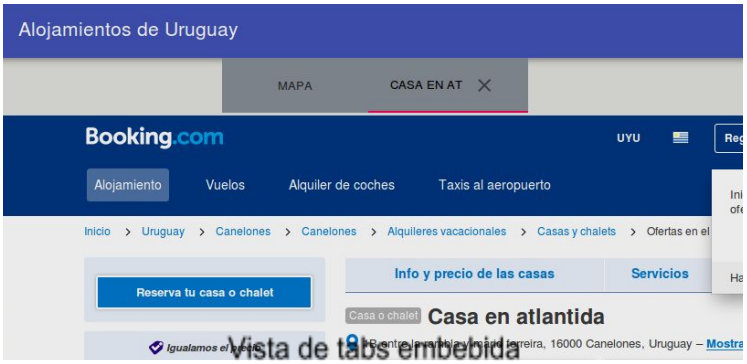
React (también llamada React.js o ReactJS) es una biblioteca javascript de código abierto diseñada para crear interfaces de usuario. Es mantenido por Facebook y la comunidad de software libre.

React intenta ayudar a los desarrolladores a construir aplicaciones que usan datos que cambian todo el tiempo. React sólo maneja la interfaz de usuario en una aplicación, es la Vista en un contexto en el que se use el patrón MVC (Modelo-Vista-Controlador) o MVVM (Modelo-vista-modelo de vista). También puede ser utilizado con las extensiones de React-based que se encargan de las partes no-UI (que no forman parte de la interfaz de usuario) de una aplicación web.

Funcionalidades y uso

La funcionalidad principal de la aplicación es la visualización de un mapa de calor en donde los puntos con mayor intensidad (más cercanos al color rojo) muestran las zonas con mayor concentración de hoteles y costo. Por otro lado, se permite que el usuario acceda al uso de filtros de tipo de alojamiento (casa, hotel, hostal, etc), puntuación de los usuarios en booking.com y elección de la capacidad mínima que desean obtener. Además, la aplicación permite que el usuario pueda acceder a un hotel en particular en la zona seleccionada por el puntero.

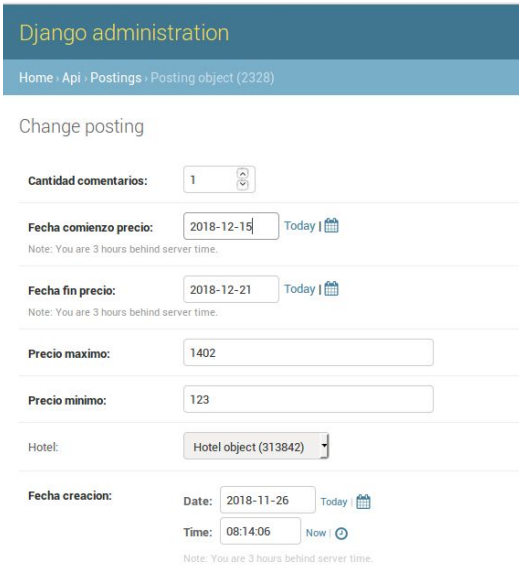




Por otro lado se ofrece una API REST (un patrón de diseño de APIs en el cual cada path de la ruta HTTP corresponde a un conjunto de objetos). En este caso, la API ofrecida muestra los distintos anuncios extraídos junto a la información del hotel (puede existir más de un anuncio que corresponda a un mismo) y los servicios ofrecidos.

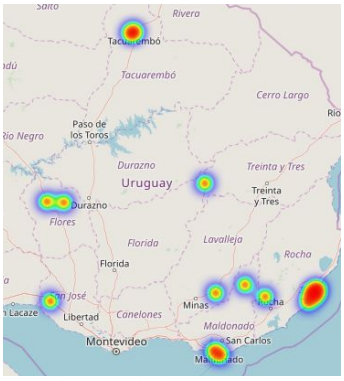
Esta API además, permite la configuración de las queries realizadas a través de parámetros en el request GET, además de proveer la vista individual de uno de los anuncios en particular.

Adicionalmente, se cuenta con un administrador para visualizar los datos conseguidos, además de permitir la edición de los mismos.



Vista de administrador

Evaluación y resultados



Hospedajes rurales

Se obtuvieron muy buenos resultados en cuanto a la visualización de datos, que era el resultado esperado. Como puede observarse en el ejemplo, los puntos geográficos en los que los alojamientos tienen un costo mayor (por ejemplo, Punta del Este) son cubiertos por valores más intensos. Además, puede apreciarse como las estancias turísticas tienen un valor más elevado.

En cuanto a rendimiento, puede notarse que el sitio puede tomar un tiempo elevado de respuesta al utilizar filtros, esto se debe principalmente a configuraciones opcionales que facilitan el desarrollo y al hecho de que todos los puntos (alojamientos) obtenidos deben ser agregados en el frontend para conseguir el mapa de calor.

Pudo observarse un rendimiento reducido de la aplicación para una cantidad relativamente baja de puntos (3500), que principalmente se corresponde al uso de una API de bajo rendimiento. La solución a este problema pasa por utilizar APIs que implementen mejores algoritmos de renderizado, sin embargo, estas son pagas (por ejemplo, la API de Google Maps).

Conclusiones

En cuanto a conclusiones finales, se debe destacar el éxito en la extracción de datos del sitio web, el problema original consistía en mostrar los datos de una forma amigable y entendible para elegir hoteles con conciencia del presupuesto del usuario. Sin embargo, algunos problemas encontrados no pudieron ser solucionados, el principal de ellos tiene que ver con el rendimiento de la aplicación. Este probablemente pueda ser solucionado utilizando soluciones que involucren renderizado del documento HTML desde el servidor, o la utilización de bibliotecas con mejor rendimiento.

Por otro lado, pudo comprobarse la diferencia en rendimiento obtenida al utilizar scrapers de código HTML y el uso APIs proveídas por los sitios web, aunque en este caso al realizar la extracción en períodos aislados de tiempo no es necesario obtener los mejores rendimientos. Por otra parte, pudo comprobarse la diferencia de tiempo al utilizar un navegador web en comparación a una aplicación local aunque es de gran utilidad contar con una aplicación web ya que puede ser fácilmente distribuida entre los usuarios. Finalmente, el uso de Docker permite la distribución sencilla de la aplicación entre desarrolladores/administradores.

Trabajo Futuro

Una gran posibilidad de ampliación de las funcionalidades de la aplicación consiste en el agregado de más filtros para ampliar el poder de búsqueda en los hoteles del sitio. Estos incluyen: cantidad de estrellas, cantidad de comentarios, servicios ofrecidos, entre otros. Estos datos se encuentran actualmente disponibles aunque no se construyeron los componentes de frontend necesarios para seleccionarlos.

Referencias

1. ReactJs: https://reactjs.org/
2. Scrapy: https://scrapy.org/
3. Docker: https://www.docker.com/
4. Booking: https://www.booking.com/
5. Django: https://www.djangoproject.com/
6. DjangoREST: https://www.django-rest-framework.org/
7. REST: https://www.restapitutorial.com/