Facultad de Ingeniería
Julio Herrera y Reissig 565.
Montevideo, Uruguay.

Noviembre 2018

Proyecto Recuperación de Información y Recomendaciones en la Web

Mathías Esteban, 4.957.703-8, mathiasesteban.me@gmail.com
Bruno Serra, 5.234.956-7, brunoserra1965@gmail.com.uy
Martín Lago, 4.814.287-0, martinlago0901@gmail.com
Santiago Rodriguez, 5.154.441-3, santi_1409@hotmail.com
Nicolás Lantean, 4.858.325-0, nico.11@hotmail.com

Profesora: Libertad Tansini

Índice

Índice 1

Introducción 2

Problema 2

Enfoque de la solución 2

Diseño 2

Implementación 4

Herramientas: 4

Descarga de imágenes 4

Publicaciones del Perfil 4

Historias del Perfil 5

Extracción de texto 6

Identificación de fechas 6

Funcionalidades y uso 7

WEB 7

Django 8

Evaluación y resultados 9

Conclusiones y trabajo futuro 10

Referencias 11

Introducción

Son decenas de eventos los que suceden en lugares bailables por fin de semana en Montevideo, la información suele encontrarse dispersa en las redes sociales de cada local, agrupar esta información es de utilidad para que los usuarios tengan un único punto de acceso a esta.

Problema

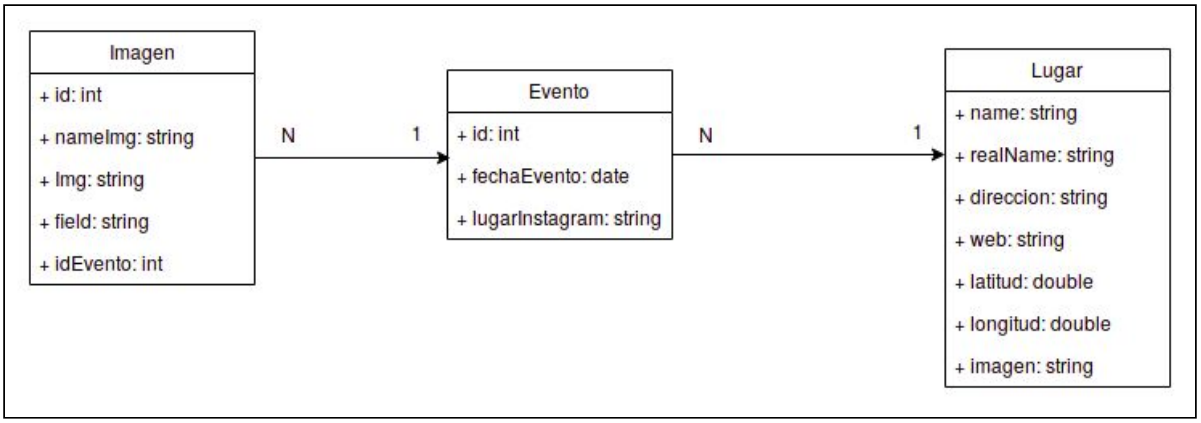
Se busca desarrollar una aplicación web que brinde una lista de los eventos a realizarse en Montevideo a través de información recopilada de las fotos publicadas por los organizadores en la red social Instagram.
El problema se puede subdividir en etapas, la primera consiste en la obtención de la información, lo que implica la descarga de imágenes de la red social. La segunda etapa consiste en la extracción de la información textual que aparece en las imágenes. Luego se debe identificar los datos relevantes asociados a los eventos en el texto extraído, siendo este el principal desafío. Finalmente el despliegue de la información procesada al usuario.

Enfoque de la solución

Tras investigar el formato de las imágenes publicadas por los organizadores se decidió desarrollar una solución que tiene como objetivo identificar las fechas de los eventos en las imágenes, implementando un sistema de información que almacene los eventos encontrados y mediante una interfaz web puedan ser consultados por los usuarios con la capacidad de aplicar filtros de fecha.

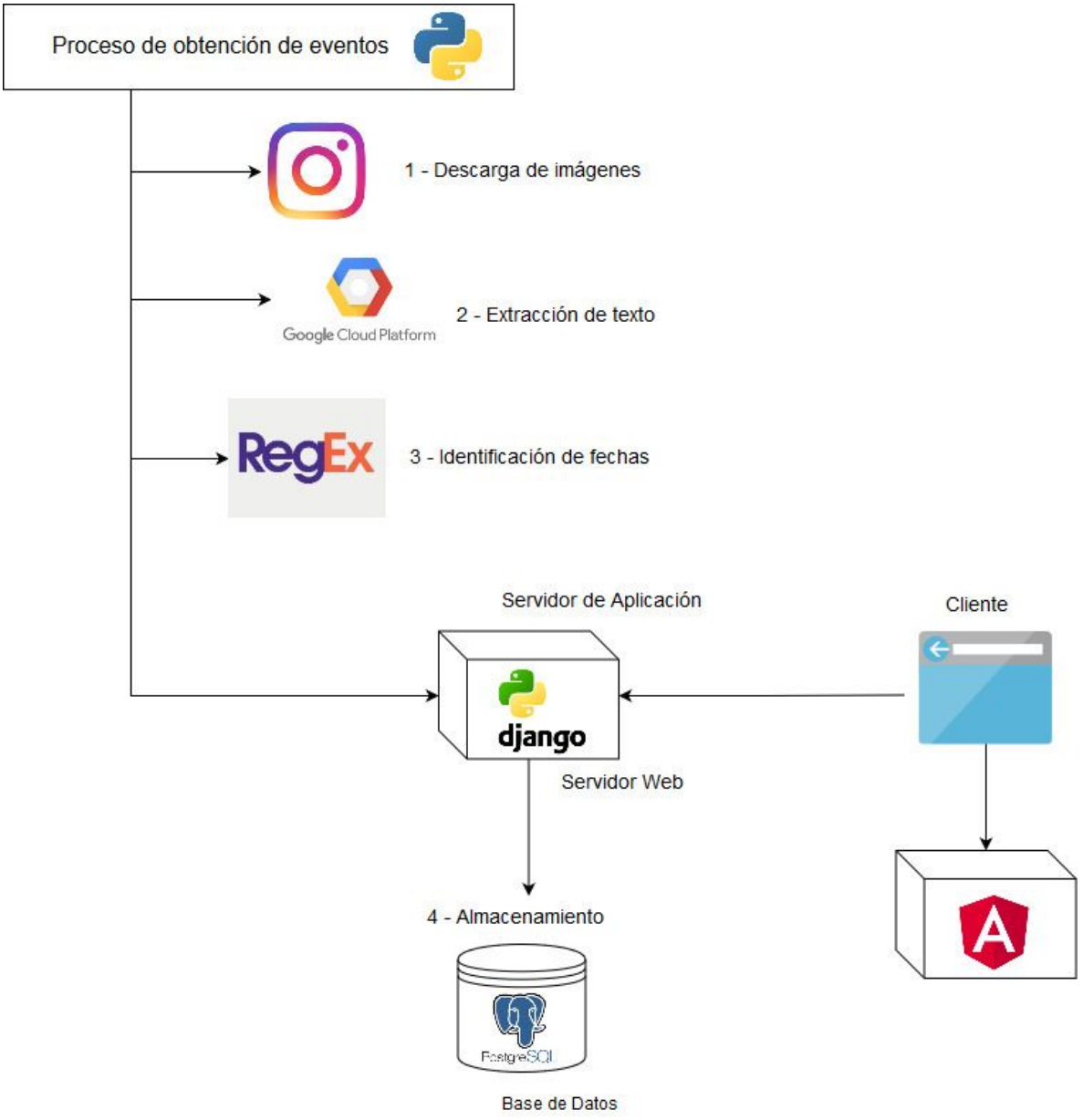
Diseño

A continuación se muestra un diagrama de las principales clases que modelan la realidad:



Para cada lugar registrado en la aplicación, se debe obtener a partir de su cuenta de instagram las imágenes asociadas a los eventos que se desarrollarán, múltiples eventos pueden realizarse en un determinado lugar, así como también puede suceder que dos o más de las imágenes publicadas en un mismo perfil estén asociadas al mismo evento, siendo responsabilidad de la lógica identificar estos casos.

En la siguiente imágen se puede ver un diagrama de la arquitectura general del sistema.



El usuario mediante una aplicación web pública desarrollada en Angular consume una API Rest desarrollada en Python con el framework Django, la cual provee los datos almacenados en una base PostgreSQL en ambiente dockerizado. El último elemento a destacar es el módulo python de creación de eventos, cuya lógica obtiene las imágenes, extrae el texto e identifica las fechas y por medio de la api ofrecida por django almacena en la base de datos la información obtenida.

Implementación

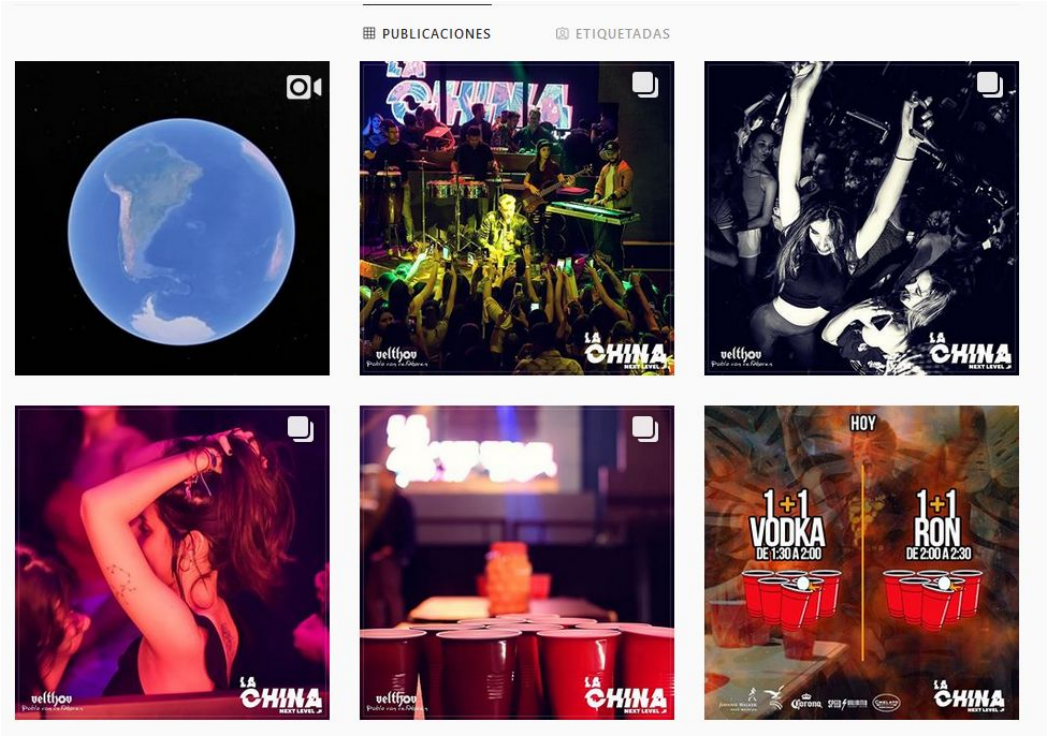
Herramientas:

- Versión 3.6 del lenguaje Python para el desarrollo de backend [1]
- El framework Django y su extensión Django-RestFramework en sus versiones 2.1 y 3.9 respectivamente. [2] [3]
- Base de datos relacional PostgreSQL 10.6 [4]
- Framework para desarrollo web Angular [5]
- Herramientas de virtualización Docker y Docker-Compose [6]
- Biblioteca Google-Cloud-Vision para reconocimiento de texto en imágenes [7]

Descarga de imágenes

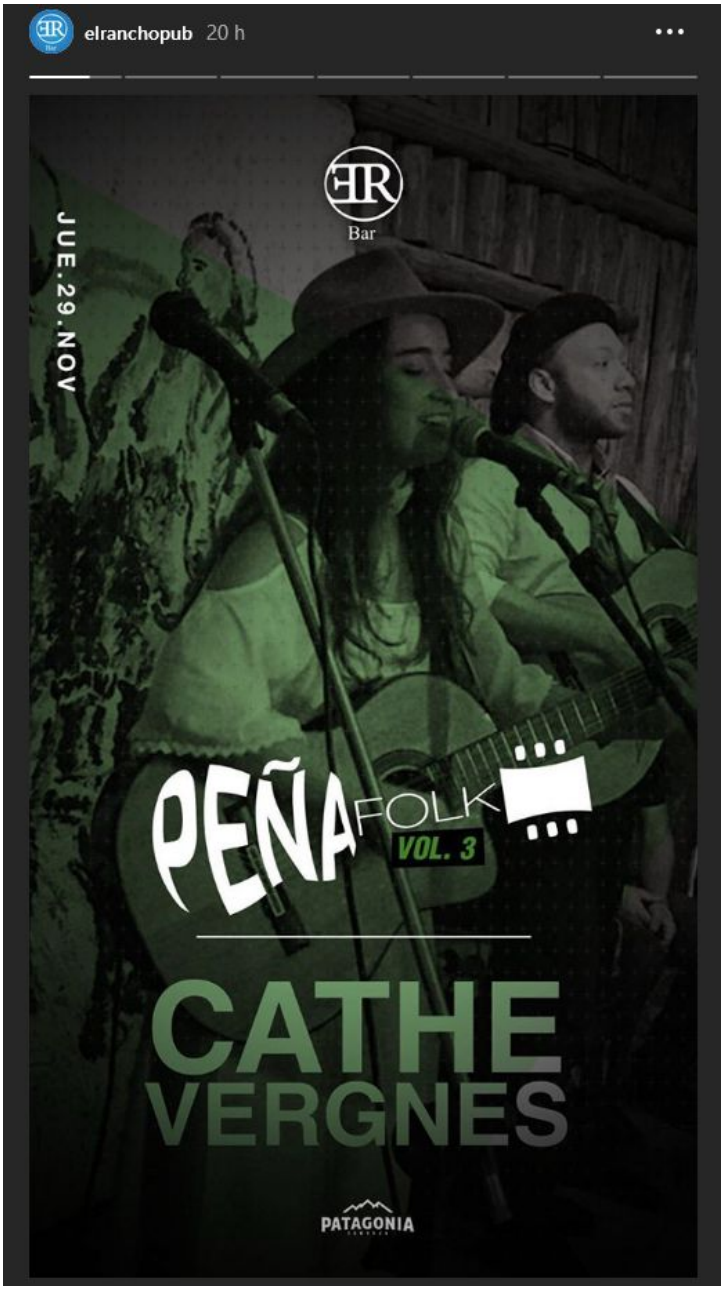
Para realizar la descarga de imágenes se implementó un módulo en python que realiza web scraping en la red social instagram para la obtención de imágenes de distintos perfiles.
Luego de investigar la API REST que brinda la red social nos dimos cuenta que la misma permitía extraer imágenes de tan solo un perfil, en específico, de la cuenta que registramos como desarrollador, para poder extraer imagenes de mas de un perfil a la vez es necesario abonar una membresía, esto nos obligó a implementar el script antes mencionado.
El algoritmo se divide en dos partes, descarga de publicaciones e historias.

Publicaciones del Perfil



La descarga de las imágenes de las publicaciones se realizó por medio de web scraping, se realiza una request a la url del perfil (http://www.instagram.com/[nombrePerfil]/) que se quiere obtener las publicaciones, la misma responde con un cuerpo HTML a través del cual se navega obteniendo las URLs de las imágenes incrustadas en el mismo.
Como las imágenes tienen un nombre único (GUID) definido por la red social, se obtiene el nombre de la URL y se invoca a la API de DJANGO para comprobar que la imagen no fue procesada en iteraciones previas del script, si no lo fue, se realiza una request HTTP GET a la URL de la imagen y se almacena la imagen descargada codificada en base64 en un lote, el cual posteriormente será analizado para realizar la extracción del texto.

Historias del Perfil



Par poder obtener las historias publicadas por los usuarios se intentó realizar web scraping igual que en la parte anterior pero no se tuvo éxito por los permisos que solicitaba la red social. Se tuvo que realizar un paso intermedio y desarrollar distintas funcionalidades para almacenar credenciales que instagram genera por medio de requests HTTP, las cuales luego son enviadas para simular el inicio de una sesión en la red social y lograr nuestro cometido.
Dichas credenciales tienen una vida útil de aproximadamente 10 minutos, se realizan chequeos para solicitarlas nuevamente.
Se creó una cuenta de instagram para utilizar en el script. Con sus credenciales se realizan los mismos pasos que en la descarga de las publicaciones ya que podemos obtener el cuerpo HTML de las mismas.

Extracción de texto

Una vez descargadas las imágenes y almacenadas en el conjunto mencionado en la sección previa, el siguiente paso es extraer todo el texto que se encuentre escrito en ella para su posterior análisis.
Para esta tarea se utilizó la biblioteca Cloud-Vision de Google la cual requiere un registro previo y en su versión gratuita tiene un tope de pedidos mas que suficiente para el alcance de este proyecto.
A esta tarea se la denomina reconocimiento óptico de caracteres y se logra mediante la aplicación de técnicas de machine learning en grandes conjuntos de entrenamiento sobre una amplia gama de lenguajes y caligrafías.

Específicamente desde el punto de vista de la implementación, por cada imagen a estudiar se debe realizar un pedido a la API antes mencionada, la cual retorna el texto detectado.
Además se realizó análisis al texto retornado por la API de Google, la cual reconoce emojis y estos caracteres no son soportados por el encoder utilizado en Python, para resolver este problema, se creó una función que a través de una expresión regular sustituye los caracteres asociados a emojis por un string vacío.

Identificación de fechas

Luego de extraer el texto de las imágenes, el siguiente paso es obtener del mismo la fecha del “evento”. Para esto hacemos uso de expresiones regulares las cuales son capaces de “reconocer” distintos tipos de formatos de fechas utilizados en las distintas imágenes.
Luego de realizar un análisis exhaustivo en las imágenes de diferentes lugares bailables se logró identificar un grupo selecto de patrones, el cual se repite en la mayoría de las imágenes.

Dentro de estos patrones se encuentran fechas que contienen el mes de forma escrita y el día (por ejemplo, “21 de septiembre”) o fechas que contienen el día y el mes en números (por ejemplo “21/9”) entre otros.

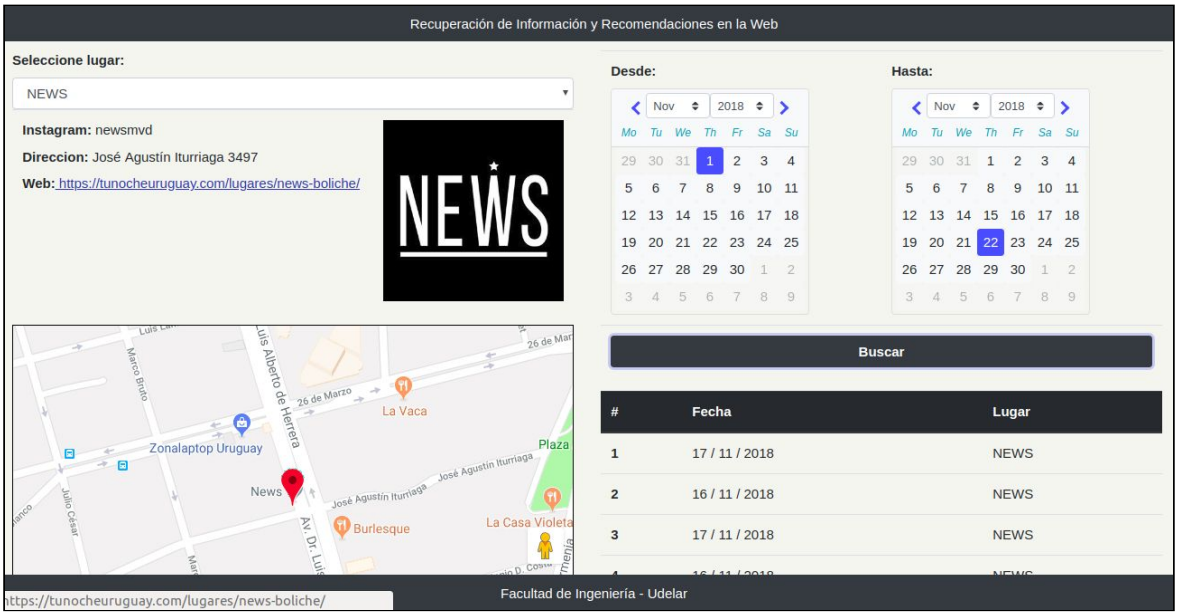
La función creada para identificar fechas retorna el tipo de datos **Date** si alguna de las expresiones regulares coincide o **None** en caso contrario.

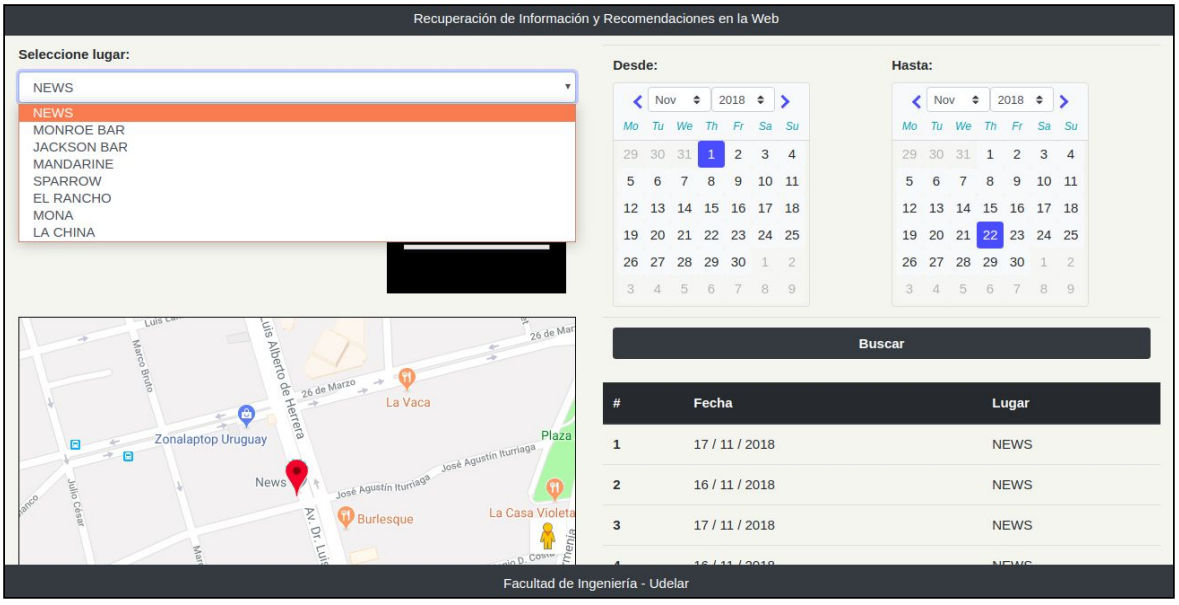
- ➔ Si la función retorna una fecha válida, se procede a invocar a la API DJANGO para comprobar si el evento encontrado existe ya en nuestra base de datos, si el evento se encuentra ya instanciado se retorna su identificador y se asocia la imagen al mismo, en caso contrario se procede a crear un nuevo evento con la fecha e imagen obtenida.
- ➔ Si la función retorna **None** se almacena la imagen en la base de datos para que no sea procesada en posteriores iteraciones, logrando de este modo ahorrar tanto tiempo como recursos del sistema.

Funcionalidades y uso

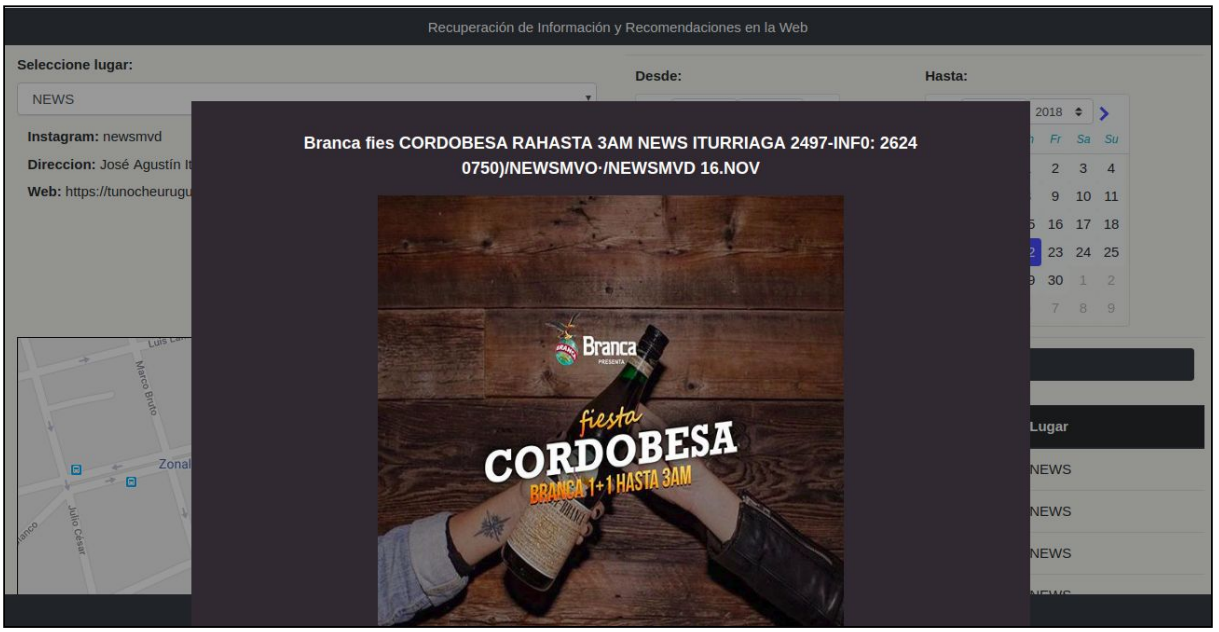
WEB

En esta sección presentaremos de manera visual la interacción del usuario con la aplicación web y las funcionalidades que esta ofrece. La interfaz se desarrolló siguiendo un diseño SPA (Single-Page-Application), es decir que interactúa con el usuario de manera dinámica sin tener que redireccionar a nuevas páginas en el navegador. La página con la cual el usuario interactúa muestra toda la información de manera compacta permitiéndole seleccionar el lugar que desee y ver sus datos, instagram, dirección, ubicación georeferenciada, web y por supuesto, los eventos asociados (con la posibilidad de filtrarlos según la fecha).





Por las características de nuestro proyecto, es interesante poder trazar el proceso de extracción de información, por lo que se agrega en la web la posibilidad de ver la imágen original descargada y el texto extraído que tras ser parseado por las expresiones regulares generaron una determinada fecha.



Django

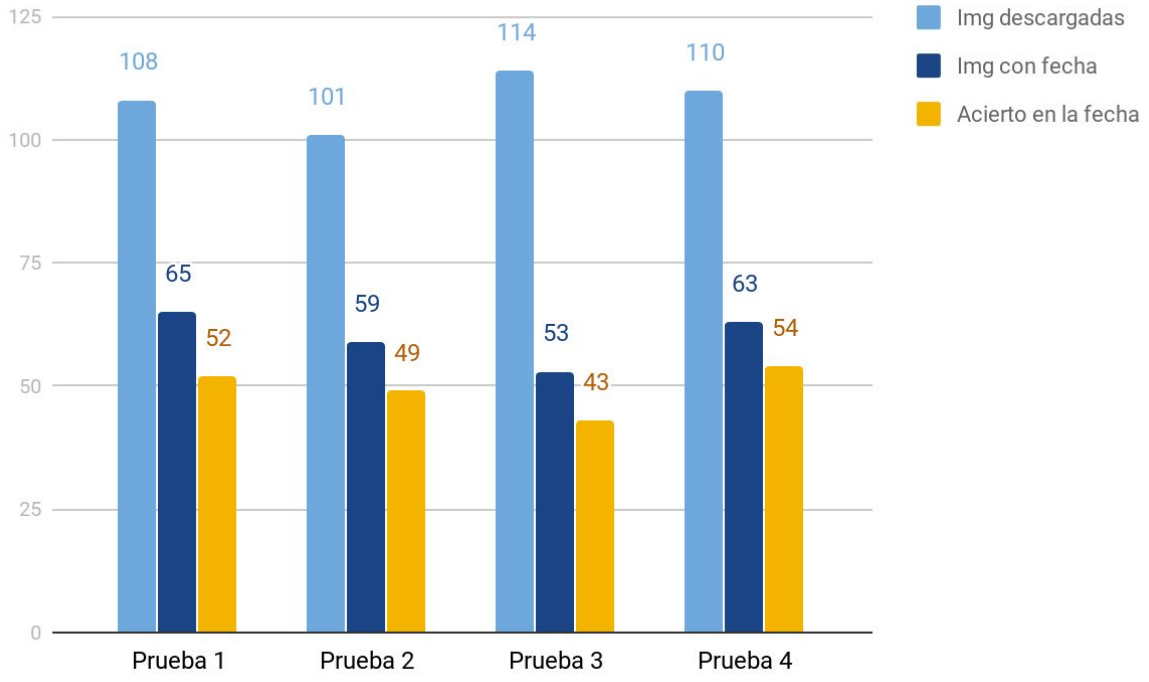
En esta sección se especifican las funcionalidades creadas en Django, se crearon los siguientes métodos **REST** para acceder a la base de datos:

1. getInstagrams/(**GET**): Retorna los datos de los locales en la base de datos.
2. getEventosFecha/(**GET**): Retorna los eventos en un rango de fechas especificado.
3. getEventos/(**GET**): Retorna todos los eventos.
4. getEvento/(**GET**): Retorna un evento con un identificador dado.
5. saveEvento/(**POST**): Almacena el evento especificado en los datos de la Request.
6. saveImagenEvento/(**POST**): Almacena en la base la imagen asociada al evento indicado en los datos de la Request.
7. getImgExist/(**GET**): Retorna true o false, dependiendo si la imagen existe en la base de datos.
8. getIdEventoFecha/(**GET**): Retorna el id del evento asociado a la fecha y lugar enviados en los datos de la Request, -1 si no existe.
9. saveLugarInstagram/(**POST**): Almacena en la base el lugar enviado en los datos de la Request.

Cabe mencionar que los datos son transferidos en formato JSON.

Evaluación y resultados

El hecho de utilizar expresiones regulares como mecanismo para identificar fechas es poco escalable y muy rígido definiendo de forma manual todas las posibles representaciones que puedan llegar a ocurrir en el texto, es necesario tener en cuenta no solo los formatos estándar y los diferentes idiomas sino también los posibles errores ortográficos y caracteres basura que puedan surgir en la extracción de texto.
Sin embargo, se debe tener en cuenta el contexto de la aplicación, esta busca las fechas en imágenes de lugares preestablecidos, los organizadores de los eventos suelen utilizar el mismo formato para escribir fechas en todas sus publicaciones, por lo que, una vez realizado el trabajo para un determinado lugar, el conjunto de reglas definido permanece invariante y detecta de manera eficaz la información objetivo. Esto se observó claramente durante el proceso de desarrollo donde el conjunto de expresiones regulares se fue estabilizando hacia el fin del proyecto logrando para un determinado conjunto de 10 lugares descargando aproximadamente de a 12 imágenes por local, una tasa de aciertos del 82.5%.



Otro aspecto a mencionar es el tiempo de ejecución, por el complejo manejo de imágenes y la interacción con la plataforma Cloud Vision el script de identificación de fechas tiene una ejecución prolongada, esto no es un problema ya que el mismo puede ejecutar de manera paralela al resto del sistema sin interrumpir su funcionamiento.

Conclusiones y trabajo futuro

Se desarrolló un sistema de información de pequeño porte pero de gran peso en cuanto a recuperación de información. Desde tres puntos de vista, se logró la recuperación de imágenes en redes sociales, la recuperación de texto en imágenes y la recuperación de información particular en texto logrando altos porcentajes de acierto.
El trabajo permitió constatar la gran dificultad que conlleva extraer información estructurada desde lenguaje natural, siendo esta parte del proyecto la de mayor complejidad. (Aclarar que la extracción de texto en imágenes se delegó a la biblioteca Cloud Vision y su complejidad es alta).
Como posibles lineamientos futuros podemos incluir continuar expandiendo el conjunto de reglas, profundizar en el testeo de la aplicación, plantear nuevos escenarios donde la lógica desarrollada pueda ser útil, e incluir de manera local la extracción del texto para reducir los tiempos de ejecución.

Referencias

[1] Python: https://www.python.org/

[2] Django: https://www.djangoproject.com/

[3] Django-RestFramework: https://www.django-rest-framework.org/

[4] PostgreSQL: https://www.postgresql.org/

[5] Angular: https://angular.io/

[6] Docker: https://www.docker.com/

[7] Docker-Compose: https://docs.docker.com/compose/

[8] Google Cloud Vision: https://cloud.google.com/vision/