DESCARGA MASIVA DE IMÁGENES DESDE LA WEB Y APLICACIÓN DE ETIQUETADO DE IMÁGENES

Edna Lizeth Ocampo Londoño

Johan Sebastián Ordoñez Rocha

Xiomara Estefany Palacios Montoya

Ingeniería de Sistemas

Director: MSc. Mauricio Arias Correa

INSTITUTO TECNOLÓGICO METROPOLITANO

Junio 27 de 2016

RESUMEN

A partir de la necesidad de una herramienta de software para image-mining que permitiera a los investigadores del laboratorio de Visión y Fotónica de ITM, descargar y etiquetar imágenes, para realizar un posterior proceso de clasificación automática, se desarrolló el presente trabajo.

La necesidad fue satisfecha por medio del desarrollo de un software que permite descargar colecciones grandes de imágenes desde la web y posteriormente extraer el conocimiento asociado. Para el desarrollo se utilizó el lenguaje de programación Python para acceder a varios motores de búsqueda y repositorios para descarga de imágenes.

El uso del software requiere de la priorización de motores de búsqueda; permite obtener aproximadamente cuatro mil (4000) imágenes realizando la consulta general, pero si se realizan consultas más específicas se obtienen aproximadamente entre mil quinientas (1500) y dos mil (2000) imágenes.

De manera similar, se desarrolló un aplicativo de etiquetado de imágenes por medio del lenguaje de programación asp.net, aplicando la arquitectura cliente-servidor, el cual permite cargar un grupo de imágenes aleatoriamente desde una carpeta, para que un usuario no experto, asigne –según su criterio- etiquetas a los objetos que componen la imagen.

Tanto la herramienta de software para descarga masiva de imágenes desde la web, como el aplicativo para etiquetado de imágenes, fueron evaluados por los investigadores de la línea de Visión Artificial y Fotónica del ITM, obteniendo resultados satisfactorios.

***Palabras clave*:** Descarga de imágenes; web scraping; etiquetado de imágenes; web; cliente-servidor; metadatos.

RECONOCIMIENTOS

Debemos agradecer en primer lugar al docente Mauricio Arias Correa quien nos brindó su acompañamiento y tutoría en el desarrollo de este proyecto.

También debemos agradecer al laboratorio de visión y fotónica, a sus docentes que nos permitieron formar parte y dejar nuestro aporte a su valiosa labor en la institución.

A nuestros padres y amigos que nos brindaron la confianza y el apoyo para realizar nuestros sueños que ya empezamos a alcanzar.

A todos nuestros compañeros de pregrado por su acompañamiento y apoyo en los momentos más difíciles y alegres de nuestra carrera.

A todos aquellos cuyos nombres no podemos listar, pero cuya presencia en nuestras vidas ha sido vital para llegar hasta aquí.

A todos ellos, muchas gracias.

ACRÓNIMOS

CBIR Content Base in Image Retrieval

IIS Internet Information Services

IBM International Business Machines Corporation

SABRES Sistema operativo que controla reservas de vuelos.

SQL Es un lenguaje estándar para acceder y manipular bases de datos.

XML Extensible Markup Language.

API Application program **i**nterface

HTTP Hypertext Transfer Protocol.

URL Uniform resource locator.

.**TXT** Extensión de un archivo de texto "textfile".

ITM Instituto Tecnológico Metropolitano

TABLA DE CONTENIDO

1. INTRODUCCIÓN

El Instituto Tecnológico Metropolitano de Medellín (ITM por sus siglas en español), es una Institución Universitaria de orientación, vocación y tradición tecnológica, de carácter público y del orden municipal. El área de Investigación de la Facultad de Ingeniería está orientada a la administración y gestión de la producción científica y tecnológica obtenida de la articulación de los investigadores, docentes, estudiantes-investigadores y los laboratorios especializados Parque i.

La Facultad cuenta con dos (2) Grupos de Investigación (*Materiales Avanzados y Energía - MATYER y Automática, Electrónica y Ciencias Computacionales - AECC*). El grupo de investigación en Automática, Electrónica y Ciencias Computacionales del Instituto Tecnológico Metropolitano, está a la fecha (junio de 2016), clasificado en categoría A1, la máxima categoría otorgada por Colciencias a los grupos de investigación en Colombia. De acuerdo al ranking Sapiens 2014, ocupa el décimo cuarto lugar entre los grupos de investigación con mayor producción científica a nivel nacional y actualmente tiene categoría A1 en Colciencias, siendo su línea de *Visión Artificial y Fotónica*, una línea de investigación relevante en el alcance de dichos logros.

La línea de Visión Artificial y Fotónica centra sus actividades de investigación en el estudio teórico y el desarrollo experimental de diferentes técnicas de sensado de variables físicas. Entre los dispositivos de adquisición de señales, se cuenta con un amplio conjunto de cámaras de visión para adquirir imágenes en diferentes zonas del espectro electromagnético, a saber: cámaras RGB, cámaras infrarrojas, cámaras térmicas y cámaras RGB-D.

La detección de objetos y personas en imágenes, es un área de trabajo creciente en investigación, debido a las diferentes alternativas que existen en cuanto a técnicas métodos, que pueden ser utilizados para dicho propósito.

De igual manera la clasificación de características que componen una imagen, puede hacer uso de diferentes técnicas y métodos, en particular de aquellos que son entrenados para hacer clasificación automática a partir de valores de entrada y de salida. Dichos valores están constituidos –entre otros- por etiquetas que se agregan a las imágenes. Por tanto desarrollar un sistema de clasificación automática de características en imágenes, requiere inicialmente de dos actividades previas: componer un conjunto de imágenes de interés y realizar el etiquetado de las mismas.

Un conjunto de imágenes de interés puede crearse a partir de descargas de imágenes desde la web, pero el proceso manual es lento y engorroso. Por otro lado, el etiquetado de imágenes requiere de muchas horas/hombre, que podrían ser proporcionadas por usuarios no expertos, usuarios de una intranet.

Justificación

El desarrollo de un aplicativo para descarga automática de imágenes a partir de un tema de interés, así como el desarrollo de un módulo de etiquetado de las imágenes descargadas, se constituye en un aporte para las investigaciones en el tema de clasificación de imágenes, debido a la automatización del proceso.

Problema

Los aplicativos comerciales no permiten descargar un volumen de imágenes, alrededor de un tema de interés, con las características ideales para conformar un conjunto de imágenes de entrenamiento. Las imágenes descargadas por dicho medio, se repiten, no tienen metadatos asociados que llenen requerimientos para procesos de clasificación de imágenes y la actividad de realizar descargar a partir de diversos aplicativos requiere de muchas horas/hombre, convirtiéndose el método manual de descarga en un proceso inefectivo. Por otro lado, el etiquetado de un conjunto grande de imágenes (100,000 o más), no es

eficiente, cuando se requiere de muchas personas trabajando desde computadores stand-alone realizando la actividad.

OBJETIVO GENERAL

Desarrollar herramientas de software para la descarga masiva de imágenes desde la web y el etiquetado de sus componentes, como apoyo a actividades de la línea de investigación en visión artificial y fotónica del ITM.

OBJETIVOS ESPECÍFICOS

- Desarrollar un aplicativo para la descarga masiva de imágenes desde la web
- Implementar un módulo para el etiquetado de objetos de imágenes como servicio web.
- Evaluar el desempeño del aplicativo y del módulo como elementos independientes y de forma relacionada.

ORGANIZACIÓN DEL TRABAJO

El contenido de este trabajo está estructurado de la siguiente manera:

En la sección 2, se presenta el estado del arte relacionado con la técnica de mayor relevancia utilizada para dar solución a las necesidades planteadas.

La metodología se presenta en la sección 3. Allí se hace claridad acerca de los métodos y técnicas utilizados para el desarrollo de la herramienta de descarga de imágenes y del aplicativo de etiquetado. En esta misma sección, se puede apreciar el proceso de desarrollo de ambos aplicativos de software. Los resultados donde se presentan las generalidades de las aplicaciones.

Las conclusiones de las técnicas y sus implicaciones, funcionalidades de las aplicaciones, sugerencias y trabajo futuro, se presentan en la sección 4.

En el apéndice (sección no numerada que aparece después de las referencias), se puede apreciar el código desarrollado.

2. MARCO TEÓRICO

(Vargiu & Urru, 2012) utilizan las técnicas de scraping para actividades tales como: el etiquetado de acceso como miembros de objetos; averiguar etiquetas cuyo nombre, contenido o atributos coincidan con criterios de selección; además para hacer etiquetado que permita acceder a los atributos utilizando una sintaxis similar a la del diccionario. El scraping se lleva a cabo por medio del uso de librerías de Python; el extractor de anuncios se basa en la Web scraping a través de dos bibliotecas especializadas: HTMLParser y BeautifulSoup. HTMLParser define una clase HTMLParser que sirve como base para analizar archivos de texto con formato en HTML y XHTML. La clase es instanciada sin argumentos y su ejemplo es alimentado por datos HTML, llama a funciones de controlador de etiquetas cuando comienzan y terminan. BeautifulSoup no es un verdadero analizador de HTML, pero utiliza expresiones regulares para bucear a través de la sopa de la etiqueta. Las principales características de BeautifulSoup son: obtiene un árbol de análisis sintáctico que da tanto sentido como el documento original, proporciona algunos métodos simples e idiomas Pythonic para navegación, búsqueda y modificación del árbol de análisis, convirtiendo automáticamente los documentos entrantes a Unicode y documentos salientes a UTF-8. El módulo extractor de anuncios da como salida un conjunto ordenado de las banderas extraídas, junto con el URL correspondiente y sus descripciones, este conjunto será analizado por el selector de anuncios que selecciona las tres banderas que se insertarán en la página web original de selección de anuncios obteniendo como resultado un sistema de publicidad Web basado en filtrado colaborativo destinado a encontrar los anuncios más relevantes para una página web.

En (Polidoro et al., 2015), utilizan las técnicas de web scraping para recolectar datos electrónicos sobre productos de uso doméstico y precios del pasaje para Italian HICP compilation. La modernización de la recolección de datos se compone principalmente de

tres características principales: ampliar el uso de dispositivos electrónicos para recoger información sobre los precios en el campo, acceder a los datos del escáner como fuente para las estimaciones de inflación, y la ampliación de la adopción de web scraping, técnicas para scrape-data desde la web para HICP compilation. El Instituto Nacional de Estadística italiano (ISTAT) está trabajando activamente en tres de estas características para la modernización de la recolección de datos. Específicamente, un grupo de expertos en estadística y TI ha probado el uso de técnicas de web en la encuesta de precios al consumidor se centra la atención en dos grupos de productos: "el consumo electrónico" (mercancías) y "pasajes aéreos" (servicios). Procedimientos Web scraping se han desarrollado y probado para estos dos grupos de productos. El resultado obtenido es la recolección de datos de diferentes sitios web para el estudio estadístico del precio al consumidor italiano.

Con respecto a la implementación de técnicas de (VT) e Inteligencia Competitiva (IC) utilizando algoritmos de minería de la WEB para examinar documentos de dominio público; (Eckerta et al., 2016), presentan las ventajas de la obtención de dicha información durante el proceso de toma de decisiones estratégicas en la cadena de producción y comercialización del té.

En (Penman, Baldwin, & Martinez, 2009) el web scraping es usado convencionalmente para la extracción de datos desde la web mediante el análisis de estas. Site Scraper es un desarrollo pensado en hacer frente al problema que se presenta a la hora de trabajar en la extracción de datos desde la web utilizando web scraping, cuyo funcionamiento se encuentra ligado a la estructura de las páginas web a intervenir y es por ello que su funcionamiento se puede ver limitado al momento en que se le aplican cambios o actualizaciones como lo son el adicionar contenido o aplicar un nuevo diseño a la página, y esto hace que trabajar con estos datos sea una tarea tediosa de revisar manualmente los cambios para aplicar de forma efectiva nuevamente el web scraping.

El resultado fue que Site Scraper -a partir de un conjunto de direcciones URLs- realizara un aprendizaje de forma automática apoyado en Xpath ("XPath", 2016) y lograr que cuando un sitio web cambie, pero su contenido se mantiene constante es decir su cambio es a nivel de actualización o estilo de la página Site Scraper es capaz o conservar automáticamente su modelo sin intervención humana.

Para lograr dicho desarrollo aplicaron Python y su biblioteca beautifulsoup ("Beautiful Soup", 2016) que es utilizada para parsear (analizar sintácticamente) documentos HTML creando un árbol con todos los elementos del documento, también incluye en el desarrollo:

- Chickenfoot ("Chickenfoot", 2016) que es una extensión de Firefox que apoya la interacción con JavaScript que apoya el desarrollo en cuanto si la página web a intervenir necesita JavaScript para su funcionamiento
- Piggy Bank (Blass, 2016),que es una extensión para Firefox que recopila información de las páginas web y la guarda
- Scrubyt que es una biblioteca de Ruby ("Acerca de Ruby", 2016), que dada una cadena de ejemplo es capaz encontrar coincidencias de dicha cadena en la página web y extraer todo lo similar.
- WWW-Mechanize, biblioteca de Perl utilizada para simular la sesión de un navegador.
- TemplateMaker, biblioteca de Python que realiza comparaciones entre los códigos HTML de las páginas con el fin de identificar que es contenido dinámico y cuál es estático y generar un modelo para realizar el scrape.

3. METODOLOGÍA

Para el desarrollo del trabajo de investigación se crearon dos aplicaciones, la primera fue implementada en Python 2.7, la cual consiste en descargar imágenes de manera masiva desde la web utilizando la técnica de web scraping(Baron Penman, Baldwin, & Martinez, 2009); con la segunda se etiquetará objetos de las imágenes provistas por un servidor, implementando una arquitectura Cliente-Servidor("Cliente-servidor", 2016) y desarrollada en Visual Studio 2013 con el lenguaje de programación ASP.NET c# y framework.

3.1. APLICACIÓN DESCARGA MASIVA DE IMÁGENES DESDE LA WEB

Para crear una base de datos con una amplia colección de imágenes de forma manual, se requiere de un trabajo arduo, demandando esfuerzo, paciencia y mucho tiempo, por lo que se generó en el laboratorio de visión y fotónica del Instituto Tecnológico Metropolitano - ITM- la necesidad de desarrollar un aplicativo capaz de crear grandes volúmenes de imágenes para la extracción del conocimiento o el Image Mining(Hema & Annasaro, 2013). Con el fin de hacer posible la construcción del aplicativo de descarga de imágenes desde la web, es necesario conocer el funcionamiento de los motores de búsqueda(Oller Gómez, 2003), tal como se indicó anteriormente, es por esto que se utilizan los “bots”(Oller Gómez, 2003), los cuales son programas que tiene la capacidad de imitar el comportamiento humano, y su funcionamiento se basa en recorrer las páginas web a través de sus enlaces como lo haría un usuario común en la World Wide Web("World Wide Web", 2016), para recolectar información y retornarla a su servidor; los motores priorizan la búsqueda en las páginas nuevas o que están en constante actualización.

Basados en el funcionamiento de los motores de búsqueda, se indagó sobre las herramientas que ellos proveen para el manejo de su información y que podrían ser útiles

para dar solución a la necesidad planteada, llegando así a lo que se conoce como API's("Interfaz de programación de aplicaciones", 2016).

Las API's proveen al usuario una serie de librerías, que permiten obtener el resultado de la consulta en un formato manipulable (xml, JSON, HTML, etc), así se puede extraer las URL´s de las imágenes y a través de otro procesamiento descargarlas.

De la búsqueda de las API's disponibles se encontró:

Tabla I. Revisión de API's

API	**DESCRIPCIÓN**
JSON/ATOM custom search API	"Permite desarrollar sitios y aplicaciones web para recuperar y mostrar los resultados del Motor personalizado custom search de manera programmable"("Custom Search JSON/Atom API", 2016)
Flickr API	"Está compuesta por un grupo de métodos a los que se puede llamar y algunos extremos API. Para realizar una acción usando la API de Flickr, debes seleccionar una convención de llamada, enviar una solicitud a su extremo y especificar un método y algunos argumentos, y recibirás una respuesta con formato". ("Flickr Services", 2016)
BOSS SEARCH API	Proporciona un rico conjunto de herramientas que los desarrolladores y los empresarios pueden utilizar para construir motores de búsqueda personalizados y experiencias innovadoras. La API FUE descontinuada en el mes de marzo del 2016. ("BOSS Search API - Yahoo Developer Network", 2016)
Yandex.XML	Este servicio permite enviar consultas al motor de búsqueda Yandex y obtener respuestas en formato XML. Puede ser utilizado para la

	creación de búsqueda a través de un sitio, un grupo de sitios, o la totalidad de Internet. ("Yandex.XML - Yandex Technologies", 2016)

Se obtuvieron los siguientes resultados después de probar las diferentes API's mencionadas en la Tabla I:

- JSON/ATOM custom search API permite obtener las urls de una búsqueda, limitando la cantidad de resultados, dejando disponible cien (100) resultados y la cantidad de búsquedas que se pueden realizar por día, siendo cien (100) búsquedas.
- Flickr API permite obtener una gran cantidad de urls.
- BOSS SEARCH API fue descontinuada
- YANDEX.XML solo trae resultados web

Del anterior análisis, se llega a la conclusión que las API's no fueron óptimas para la solución del problema debido a las restricciones impuestas por las mismas, pero dieron un aporte importante, dejando como hipótesis, la idea de cómo obtener los resultados de las búsquedas.

Indagando acerca del proceso que realizan las APi's para obtener las URL's con el fin de simular ese mismo proceso, se encontró una técnica denominada Web Scraping, la cual permite obtener el contenido de una página en formato manipulable.

Con la idea de la técnica a implementar para obtener las urls de las imágenes, se investigó acerca de lenguajes que proveen librerías capaces de facilitar el uso de dicha técnica y se consideró viable python 2.7.

3.1.1. DESARROLLO APLICACIÓN DESCARGA MASIVA DE IMÁGENES DESDE LA WEB

Se indagaron diferentes motores de búsqueda para determinar según sus características, a las que se les aplicaría la técnica antes descrita.

Tabla II. Revisión Motores de búsqueda

Motor	Resultados	IMP	Enlace
Google	Motor de búsqueda #1 y buena cantidad de resultados de acuerdo a la búsqueda	SI	https://www.google.com/imghp?hl=es-419
Yahoo	De los motores de búsqueda más implementados y buena cantidad de resultados de acuerdo a la búsqueda	SI	https://co.images.search.yahoo.com
Yandex	De los motores de búsqueda más implementados y buena cantidad de resultados de acuerdo a la búsqueda	SI	https://www.yandex.com/images
Dogpile	Trae buenos resultados de búsqueda, resultados similares a yahoo	SI	http://www.dogpile.com/?qc=images
Ecosia	Trae buenos resultados de búsqueda, resultados similares a yahoo	SI	https://www.ecosia.org/images?
Foter	Buscador de imágenes para flickr	SI	http://foter.com
Baidu	Descartado por idioma – japonés y chino	NO	http://www.baidu.com

Bing	Descartado por resultados no acordes a la búsqueda	NO	https://www.bing.com/?scope=images&FORM=Z9LH1
DuckDuck Go	Descartado por pocos resultados que además se pueden obtener de otros motores más completos	NO	https://duckduckgo.com/?q=fuego&iax=1&ia=images
Exalead	Descartado por pocos resultados no acordes a la búsqueda	NO	https://www.exalead.com/search/image
GigaBlast	No tiene búsqueda de imágenes	NO	https://www.gigablast.com/
Munax	Descartado por dificultad en su uso	NO	http://www.munax.com/
Qwant	Descartado por pocos resultados que además se pueden obtener de otros motores más completos	NO	https://www.qwant.com/images
Sogou	Descartado por idioma – chino	NO	-
Soso.com	Descartado por idioma – chino	NO	http://www.sogou.com/?rfrom=soso
Youdao	Descartado por idioma- chino	NO	http://www.youdao.com/
Faroo	Descartado pues no tiene búsqueda de imágenes	NO	http://www.faroo.com/

Yacy	Descartado por malos resultados	NO	-
Excite	Powered by dogpile	NO	http://msxml.excite.com/search/images
HotBot	Descartado porque no tiene búsqueda de imágenes	NO	http://www.hotbot.com/

La implementación de la aplicación consta de 3 partes claves:

- Proveer la url de los diferentes motores de búsqueda, con el fin de identificar los parámetros variables, como lo son: la consulta y el paginado.
- Extracción de las urls de las imágenes mediante el uso de expresiones regulares.
- Proceso de descarga.

Las pruebas realizadas se ejecutaron con el navegador Google Chrome en los sistemas operativos Windows 8.1 y Windows 10

Para el primer ítem, basta con capturar la url al momento de realizar una búsqueda.

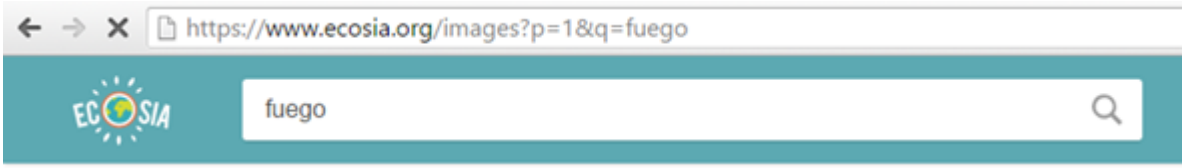

Figura 1. Extracción Url Ecosia

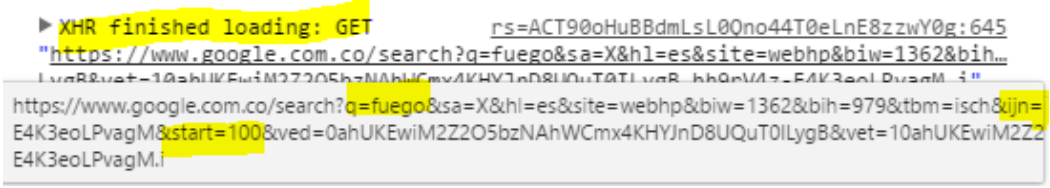

Figura 2. Extracción Url Google Infinite Scroll

Se identifican las variables, donde ***p*** es el paginado y ***q*** es la búsqueda para este motor. En el caso en que la página del motor use ***Infinite Scroll***, la extracción se realiza mediante una inspección de la página. Por medio de la herramienta provista por Chrome se busca la

pestaña de consola, se habilita la opción de Log XMLHttpRequest y se captura la url provista por el XHR.

Cabe anotar que este proceso de extracción de url es diferente para cada motor, lo que implica que se deban examinar detalladamente cada uno, con el fin de determinar los parámetros necesarios para la búsqueda, como lo es la consulta y el paginado que será controlado por un ciclo, ocurriendo lo mismo aplica para cada expresión regular.

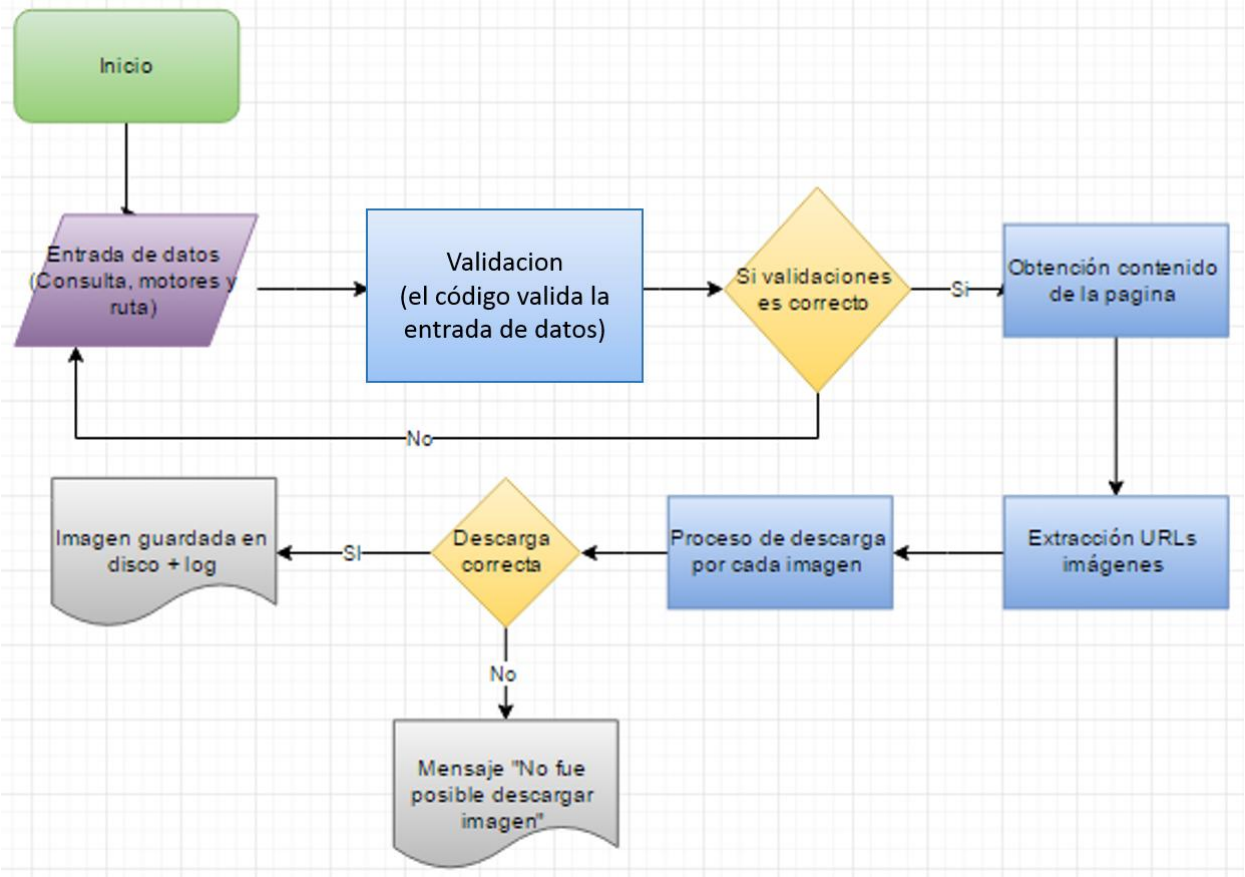

Figura 3. Diagrama de flujo aplicativo descarga de imágenes

Teniendo la url, se obtiene el contenido de dicha página por medio de la librería provista por python Urllib2(“urllib2 — extensible library for opening URLs — Python 2.7.12rc1 documentation", 2016), seguido por extraer las urls de las imágenes que arroja la búsqueda utilizando expresiones regulares("Expresiones regulares", 2016).

Después de haber obtenido dichas direcciones se procede con la descarga física de las imágenes.

Con la funcionalidad completa se implementó una interfaz usando la librería tkInter("Tkinter — Python interface to Tcl/Tk — Python 2.7.12rc1 documentation", 2016), donde el usuario puede seleccionar por ejemplo: diversos motores de búsqueda, estimados de imágenes a descargar y la ruta donde desea almacenar el contenido; además de guardar un pequeño log donde quedan consignados datos generales de la búsqueda.

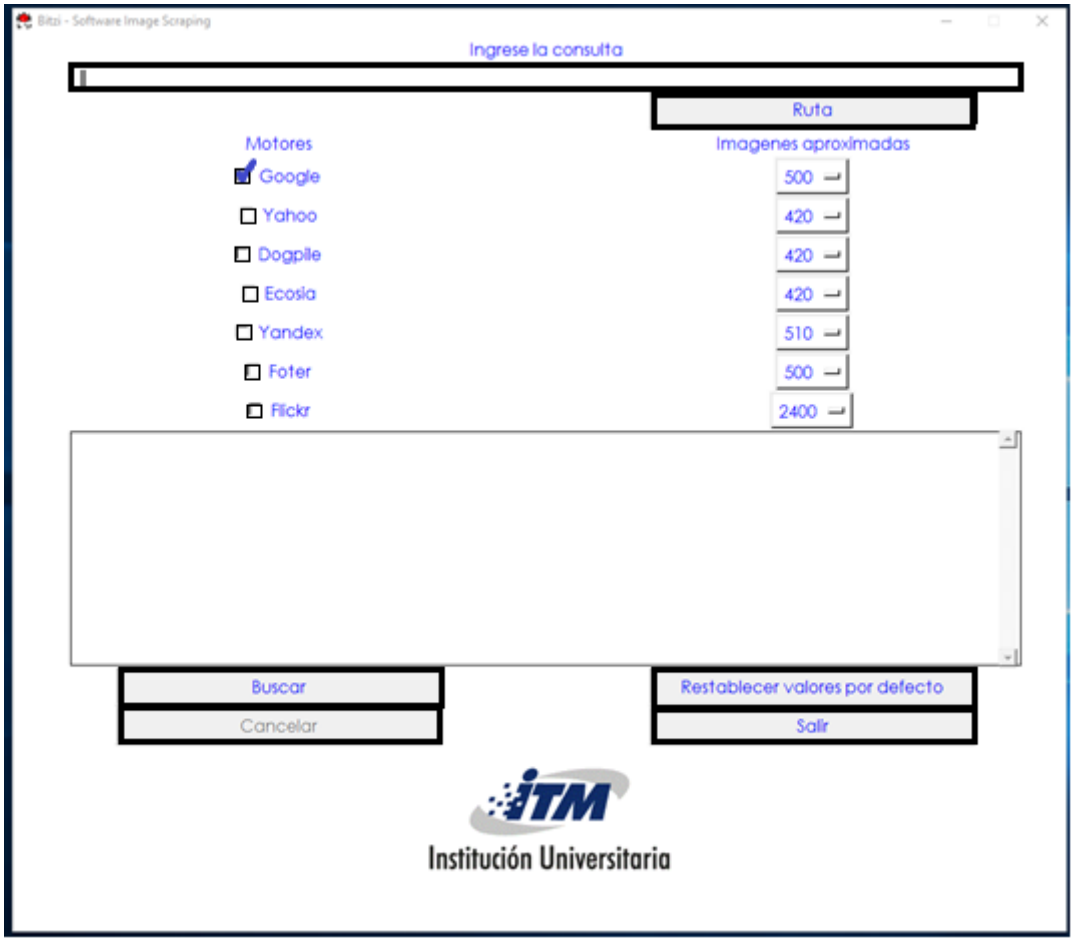

Figura 4. Interfaz aplicación descarga masiva de imágenes

3.2. APLICACIÓN ETIQUETADO DE IMÁGENES

Es de gran importancia estratégica obtener gran cantidad de información, a la cual es posible acceder por medio de la minería de datos y otras técnicas de inteligencia artificial,

es por esto, que se procede con la creación del aplicativo de etiquetado de imágenes con el fin de crear metadatos("Metadato", 2016), es decir, datos sobre datos que están estructurados de forma que ayudan a la identificación, descripción, clasificación y localización del contenido de cada una de las imágenes seleccionadas previamente.

Para el efectivo funcionamiento del aplicativo se implementó el modelo arquitectural de cliente- servidor en el cual las cargas son repartidas entre dos roles: un servidor que contiene toda la lógica de negocios y un cliente que consumirá los servicios provistos por el servidor.

El servidor se encarga de:

- Carga de imágenes aleatorias: el servidor busca una imagen de una ruta específica o en sus subdirectorios, selecciona una, lee sus bytes y los convierte enBase64 para ser agregados como url de un control ImageButton que será desplegado al cliente. Además se almacena en un control HiddenField el nombre de la imagen y en otro la fecha y hora de la solicitud.
- Almacenamiento de etiquetas: Se captura el valor de un control HiddenField que contiene la descripción del punto seleccionado junto con su respectiva coordenada. Luego se procede a almacenar las etiquetas en un archivo plano con el siguiente formato (“Fecha y hora”+”nombre de la imagen”.txt) y para ello es necesario consultar los controles HiddenField capturados en la carga de imágenes. El uso de los HiddenField es necesario para evitar la sobreescritura al momento de escribir los archivos planos debido a los múltiples clientes que pueden hacer solicitud.

El cliente se encarga de:

- Solicitar una nueva imagen.
- Encargarse del proceso de etiquetado que consiste en dar click sobre un objeto de la imagen y asignar la descripción de dicho punto. Repetir el procedimiento las veces que sea necesario.

Nota: Al dar click sobre la imagen previamente cargada, un script de JavaScript despliega una ventana donde el usuario ingresa la descripción del punto seleccionado y es almacenado en un control HiddenField.

Para el uso final de la aplicación es necesario publicarla para que efectivamente los clientes puedan acceder a ella, dicho procedimiento hace uso de IIS 6.0 y se le asigna un puerto de escucha que debe ser habilitado en el firewall con el fin de permitir las conexiones.

4. RESULTADOS Y DISCUSIÓN

4.1. APLICACIÓN DESCARGA MASIVA DE IMÁGENES DESDE LA WEB

Como producto para el requerimiento se creó una aplicación desarrollada en python 2.7 con una interfaz gráfica que permite al usuario ingresar una consulta y escoger de una lista de motores de búsqueda de cuales desea incluir resultados con la cantidad aproximada por cada uno de ellos, adicional a esto le permite escoger la ruta de destino en su equipo donde las imágenes obtenidas serán guardadas.

El aplicativo le informa al usuario durante la descarga que imágenes fueron obtenidas con éxito o si ocurrió algún error y al final del proceso se le informa el resumen general de la operación, totalizando las urls escaneadas, descargadas, descargadas por motor y no descargadas.

En la ruta asignada por el usuario donde se almacenan las imágenes se entrega un archivo plano con la información general de la búsqueda (consulta realizada, hora y fecha), la información de cada una de las urls que se descargó y los totales que se informan en la interfaz al final del proceso de descarga.

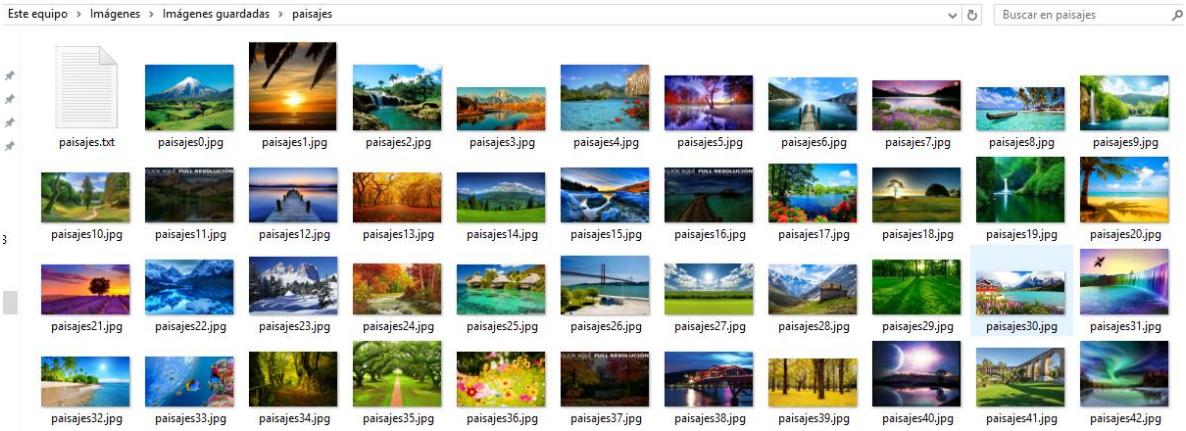

Figura 5. Contenido donde se realizó descarga.

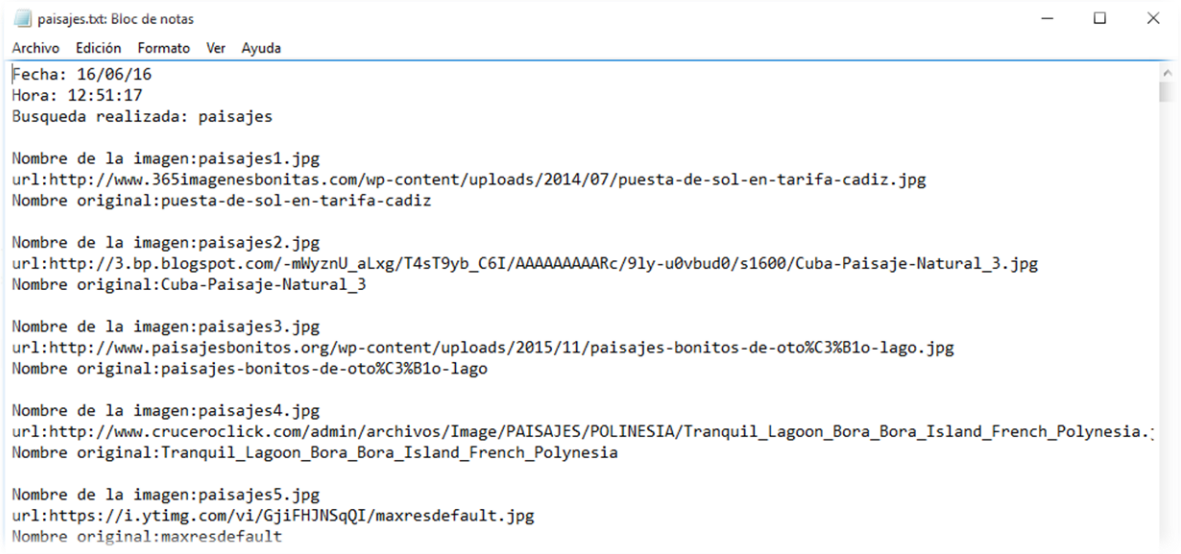

Figura 6. Metadatos de la descarga.

4.2. APLICACIÓN: ETIQUETADO DE IMÁGENES

Se desarrolló un aplicativo web en Visual Studio 2013 con framework 3.5, el cual permite al usuario ingresar a una dirección web, solicitar una imagen que le será entregada aleatoriamente por el servidor, para que posteriormente el usuario pueda proceder a realizar el proceso de etiquetado, el cual consiste en identificar múltiples objetos en la imagen, dar click sobre ellos y asignarles una descripción en el cuadro de diálogo desplegado por la aplicación.

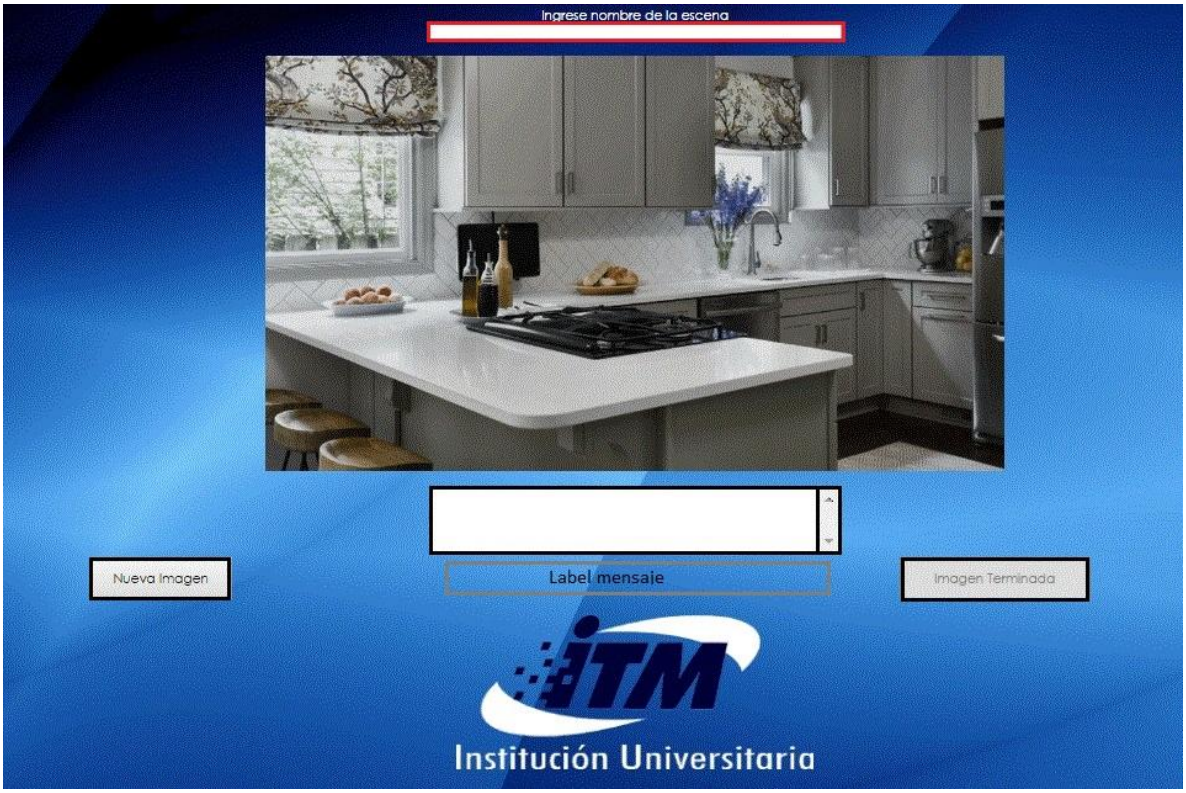

Figura 7. Interfaz gráfica del aplicativo para etiquetado de imágenes.

El servidor tiene alojadas las imágenes que utilizará la aplicación en una ruta específica, cada vez que el usuario ingresa una etiqueta esta se escribe junto a su coordenada y se muestra en el listbox que está ubicado debajo de la imagen, al darle click al botón Imagen terminada, se captura la etiqueta general de la escena y el contenido del listbox, que se almacena en el documento de texto y se bloquea la imagen de manera que no se puedan ingresar más etiquetas, además de permitirle al usuario solicitar una nueva.

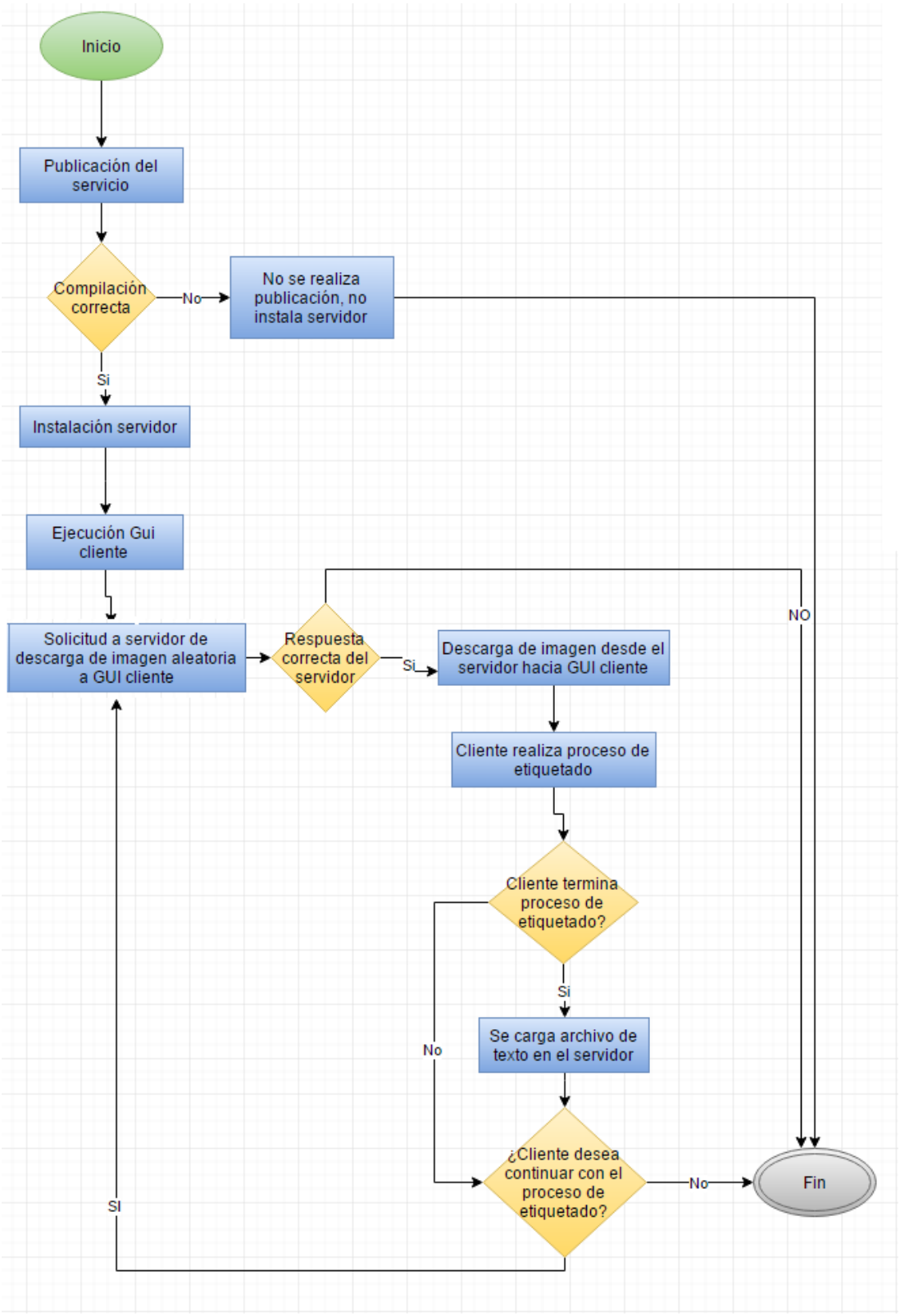

Figura 8. Diagrama de flujo aplicativo etiquetado.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

- La aplicación de descarga de imágenes permite obtener cerca de cuatro (4000) mil imágenes utilizando todos los criterios de búsqueda cuando la consulta realizada es general, para el caso de una consulta más específica los resultados se limitan (1500 - 2000 imágenes) debido al uso de repositorios como fuente de obtención de imágenes.
- Se recomienda al momento de hacer consultas específicas evitar el uso de los repositorios y priorizar el uso de los motores búsqueda tales como google, yahoo, dogpile, ecosia y yandex.
- Al hacer uso de web scraping se está sujeto a:
    - Al ser un proceso automatizado puede ser detectado como un bot por alguno de los motores de búsqueda utilizados y por ello el motor deniega la solicitud y no arroja resultados.
    - Debido a que el web scraping trabaja con la información que puede obtener de las páginas de los motores de búsqueda, al momento que dichas páginas sean actualizadas y/o modificadas, el proceso para la obtención de imágenes puede verse comprometido y no funcionar correctamente.
- Se recomienda el uso de proxy o ip dinámica al momento de usar el aplicativo con el fin de minimizar el riesgo de que se detecte el proceso automatizado.
- Se debe evitar el uso de caracteres especiales en las consultas con el fin de que la búsqueda sea más eficaz.

- El realizar una consulta muy específica puede provocar que los resultados sean escasos y no ser lo esperado.
- La velocidad de descarga de las imágenes está sujeta a la velocidad de la red.
- La aplicación de etiquetado provee al cliente imágenes en su tamaño real debido a la necesidad de obtener su coordenada en un punto específico lo cual puede ser incómodo con las imágenes de gran resolución en relación al dispositivo en el que se estén visualizando.
- Para trabajo futuro en relación al aplicativo de etiquetado se piensa:
    - En la incorporación de un login con el fin de que las personas que deseen utilizar el aplicativo deban identificarse y evitar que se creen registros basura.
    - La implementación de una marca en los puntos que ya fueron etiquetados con el fin de ayudar de una manera gráfica al usuario a tener un control de lo que está etiquetando.
    - Controlar la recarga de la página al etiquetar un punto.

REFERENCIAS

- Vargiu, E., & Urru, M. (2012). Exploiting web scraping in a collaborative filtering-based approach to web advertising. Artificial Intelligence Research, 2(1), p44.
- Polidoro, F., Giannini, R., Conte, R. L., Mosca, S., & Rossetti, F. (2015). Web scraping techniques to collect data on consumer electronics and airfares for Italian HICP compilation. Statistical Journal of the IAOS, 31(2), 165-176.
- Eckert, K., Favret, F., Barboza, M., Witzke, L. M., & Alvarenga, V. M. (2016, May). Modelos de análisis de información para la toma de decisiones estratégicas del sector tealero. In XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina).
- Penman, R. B., Baldwin, T., & Martinez, D. (2009). Web scraping made simple with sitescraper.
- Beautiful Soup. (2016). Es.wikipedia.org. Retrieved 28 June 2016, from https://es.wikipedia.org/wiki/Beautiful_Soup
- Chickenfoot. (2016). Groups.csail.mit.edu. Retrieved 28 June 2016, from http://groups.csail.mit.edu/uid/chickenfoot/
- Blass, S. (2016). Extending Firefox like a piggy bank. Network World. Retrieved 28 June 2016, from http://www.networkworld.com/article/2314768/software/extending-firefox-like-a-piggy-bank.html
- Acerca de Ruby. (2016). Ruby-lang.org. Retrieved 28 June 2016, from https://www.ruby-lang.org/es/about/
- XPath. (2016). Es.wikipedia.or
- Cliente-servidor. (2016). Es.wikipedia.org. Retrieved 23 June 2016, from https://es.wikipedia.org/wiki/Cliente-servidor

- Hema, & Annasaro,. (2013). A SURVEY IN NEED OF IMAGE MINING TECHNIQUES. International Journal Of Advanced Research In Computer And Communication Engineering, 2.
- Oller Gómez, J. (2003). Elementos teórico-prácticos útiles para comprender el uso de los motores de búsqueda en Internet. Acimed, 11(6), 0-0. Recuperado en 23 de junio de 2016, de http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352003000600007&lng=es&tlng=es.
- World Wide Web. (2016). Es.wikipedia.org. Retrieved 23 June 2016, from https://es.wikipedia.org/wiki/World_Wide_Web
- Interfaz de programación de aplicaciones. (2016). Es.wikipedia.org. Retrieved 23 June 2016, from https://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones
- Custom Search JSON/Atom API. (2016). Google Developers. Retrieved 6 May 2016, from https://developers.google.com/custom-search/json-api/v1/overview
- Flickr Services. (2016). Flickr.com. Retrieved 6 May 2016, from https://www.flickr.com/services/api/misc.overview.html
- –BOSS Search API - Yahoo Developer Network. (2016). Developer.yahoo.com. Retrieved 6 May 2016, from https://developer.yahoo.com/boss/search/
- –Yandex.XML — Yandex Technologies. (2016). Tech.yandex.com. Retrieved 6 May 2016, from https://tech.yandex.com/xml/
- urllib2 — extensible library for opening URLs — Python 2.7.12rc1 documentation. (2016). Docs.python.org. Retrieved 23 June 2016, from https://docs.python.org/2/library/urllib2.html
- Expresiones regulares. (2016). DesarrolloWeb.com. Retrieved 23 June 2016, from http://www.desarrolloweb.com/articulos/2033.php
- Tkinter — Python interface to Tcl/Tk — Python 2.7.12rc1 documentation. (2016). Docs.python.org. Retrieved 23 June 2016, from https://docs.python.org/2/library/tkinter.html

- Metadato. (2016). Es.wikipedia.org. Retrieved 23 June 2016, from https://es.wikipedia.org/wiki/Metadato#Metadatos_en_la_inform.C3.A1tica

APÉNDICE

Apéndice A Bitzi software image mining

- Librerías implementadas en el código

```
##Librerias utilizadas
import urllib2 #Para hacer hacer las peticiones y para abrir cada una de la urls
import re #Para controlar expresiones regulares
import os #Para manejo de carpetas
from Tkinter import *   #Para la Interfaz
import tkFileDialog     #Para Abrir un browse de la ruta a guardar
import tkFont
import threading    #Para manejar hilos de ejecucuón
import time     #Para obtener datos como la fecha y hora
import sys
##Fin Librerias
```

- Variables globales

```
#variables globales
x = 0   #Contador global para imagenes descargadas
totalUrls = []  #Arreglo que contiene todas las url obtenidas sin repetir
hdr = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/50.0.2661.102 Safari/537.36',
       #Aplicacion que funciona como cliente en un protocolo de red
    'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
    #Determina el tipo de contenido o MIME que se espera de la respuesta
    'Accept-Charset': 'ISO-8859-1,utf-8;q=0.7,*;q=0.3',     #Determina el set de caracteres aceptable en la respuesta
    'Accept-Encoding': 'none',  #Determina la codificación (compresión) que se espera de la respuesta
    'Accept-Language': 'es-ES,es;q=0.8',    #Determina el idioma aceptado para la respuesta
    'Connection': 'keep-alive'}
consulta = ''
swCancelar = 0 #Si swCancelar = 0 se ejecuta con normalidad el proceso, si es swCancelar = 1, se termina
urlsRepetidas = 0   #Contador urls repetidas
cPerdidas = 0   #Contador imagenes perdidas
#Fin variables globales
```

- Inicio de los métodos que hacen parte del funcionamiento de la interfaz

  Find_data_file: conjunto de instrucciones necesarias al momento de realizar el empaquetado de la solución.

```
#Métodos Interfaz
def find_data_file(filename):
    if getattr(sys, 'frozen', False):
        # The application is frozen
        datadir = os.path.dirname(sys.executable)
    else:
        # The application is not frozen
        # Change this bit to match where you store your data files:
        datadir = os.path.dirname(__file__)

    return os.path.join(datadir, filename)
```

- Validaciones

```
def validaciones():     ##Valida que los parametros necesarios para la consulta esten ingresados
    if not cons.get():  ##Si el textBox de consulta esta vacio
        mensaje.insert(END, 'Debe Ingresar la consulta ')
        return 0
    if not ruta.get():  ##Si no seleccionó una ruta
        mensaje.insert(END, 'Debe selecionnar una carpeta para guardar las imagenes ')
        return 0

    contador = 0

    if chkGoogle.get() == 1:
        contador = contador+1
    if chkYahoo.get() == 1:
        contador = contador+1
    if chkEcosia.get() == 1:
        contador = contador+1
    if chkDogpile.get() == 1:
        contador = contador+1
    if chkYandex.get() == 1:
        contador = contador+1
    if chkFoter.get() == 1:
        contador = contador+1
    if chkFlickr.get() == 1:
        contador = contador+1

    if contador == 0: ##Si ningún CheckBox esta seleccionado
        mensaje.insert(END, 'Debe selecionnar un motor ')
        return 0

    return 1
```

- Bloqueo de controles en la interfaz

```
def bloqueo():
##Bloquea los controles cuando se esta haciendo una busqueda para prevenir errores por otra solicitud
    cons.config(state=DISABLED)
    btnRuta.config(state=DISABLED)
    btnBuscar.config(state=DISABLED)
    btnRestablecer.config(state=DISABLED)
    btnCancelar.config(state=ACTIVE)
    CH1.config(state=DISABLED)
    CH2.config(state=DISABLED)
    CH3.config(state=DISABLED)
    CH4.config(state=DISABLED)
    CH5.config(state=DISABLED)
    CH6.config(state=DISABLED)
    CH7.config(state=DISABLED)
    w1.config(state=DISABLED)
    w2.config(state=DISABLED)
    w3.config(state=DISABLED)
    w4.config(state=DISABLED)
    w5.config(state=DISABLED)
    w6.config(state=DISABLED)
    w7.config(state=DISABLED)
```

- Desbloqueo de los controles de la interfaz

```
def desbloqueo():   ##Desbloquea los controles permitiendo una nueva búsqueda
    cons.config(state=NORMAL)
    btnRuta.config(state=ACTIVE)
    btnBuscar.config(state=ACTIVE)
    btnRestablecer.config(state=ACTIVE)
    btnCancelar.config(state=DISABLED)
    CH1.config(state=NORMAL)
    CH2.config(state=NORMAL)
    CH3.config(state=NORMAL)
    CH4.config(state=NORMAL)
    CH5.config(state=NORMAL)
    CH6.config(state=NORMAL)
    CH7.config(state=NORMAL)
    w1.config(state=ACTIVE)
    w2.config(state=ACTIVE)
    w3.config(state=ACTIVE)
    w4.config(state=ACTIVE)
    w5.config(state=ACTIVE)
    w6.config(state=ACTIVE)
    w7.config(state=ACTIVE)
```

- Definición de los botones de la interfaz Salir, Cancelar y Reestablecer

```
def salir():    ##Cierra la interfaz
    root.destroy()  #Destruye la ventana principal

def cancelar():     ##Cambia el swCancelar = 1
    global swCancelar
    swCancelar = 1

def restablecer():  ##Restablece los controles a los valores por defecto
    cons.delete(0, END)
    ruta.set('')

    parGoogle.set('500')
    parYahoo.set('420')
    parDogpile.set('420')
    parEcosia.set('420')
    parYandex.set('510')
    parFoter.set('500')
    parFlickr.set('2400')
    CH1.select()
    CH2.deselect()
    CH3.deselect()
    CH4.deselect()
    CH5.deselect()
    CH6.deselect()
    CH7.deselect()

    mensaje.delete(0, END)
```

- Método enlazado al botón buscar de la interfaz

```
def buscar():   #Método llamado por el boton btnBuscar
    global consulta
    global x
    global totalUrls
    global swCancelar
    global cPerdidas
    global urlsRepetidas
    #Se restablecen las variables globales
    swCancelar = 0
    consulta = ''
    x=0
    totalUrls = []
    cPerdidas = 0
    urlsRepetidas = 0
    if validaciones() != 0:     #Si todas las validaciones son correctas, se continua con el proceso normal
        bloqueo()
        mensaje.insert(END, 'Se esta buscando: '+cons.get())
        consulta = cons.get() #Almacena la consulta dada por el usuario
        consulta.strip()    #Elimina espacios al inicio y al final de la consulta
        consulta = consulta.replace(' ', '+')   #Reemplaza espacios por + para evitar conflictos en las consultas

        fichero = ruta.get() #Almacena el nombre o la ruta de la carpeta para guardar las imagenes
        fichero = fichero + '/' + consulta + ' ' + time.strftime("%d-%m-%y_%I_%M")
        ruta.set(fichero)
        #Creamos o ingresamos en el directorio
        directorio = os.path.join(fichero)
        if not os.path.isdir(directorio):
            os.mkdir(directorio)
        os.chdir(directorio)

        #Se crea otro hilo de ejecucion para que el proceso de descarga no interfiera con el proceso de interfaz
        t = threading.Thread(target=motores) #El nuevo hilo apunta al método motores
        t.start() #Se inicia el nuevo hilo
```

- Código enlazado al botón ruta de la interfaz

```
def hallarRuta():   #Abre el cuadro de dialogo para el usuario seleccione la ruta donde se guardaran las imagenes
    ruta.set(tkFileDialog.askdirectory())
```

- Método motores

```
def motores():  #Llama al proceso central y al finalizar despliega los totales y los agrega a los metadatos
    global totalUrls
    global x
    global swCancelar
    global urlsRepetidas
    global cPerdidas

    #Contadores de descargas por motores
    m1 = 0
    m2 = 0
    m3 = 0
    m4 = 0
    m5 = 0
    m6 = 0
    m7 = 0

    #Si los motores esta seleccionados, se inicia el proceso central que retorna
    #un entero con la cantidad de imagenes descargadas
    if chkGoogle.get() == 1 and swCancelar == 0:
        m1 = proceso('1', int(parGoogle.get())/100)
    if chkYahoo.get() == 1 and swCancelar == 0:
        m2 = proceso('2', int(parYahoo.get())/35)
    if chkEcosia.get() == 1 and swCancelar == 0:
        m3 = proceso('3', int(parEcosia.get())/35)
    if chkDogpile.get() == 1 and swCancelar == 0:
        m4 = proceso('4', int(parDogpile.get())/30)
    if chkYandex.get() == 1 and swCancelar == 0:
        m5 = proceso('5', int(parYandex.get())/30)
    if chkFoter.get() == 1 and swCancelar == 0:
        m6 = proceso('6', int(parFoter.get())/50)
    if chkFlickr.get() == 1 and swCancelar == 0:
        m7 = proceso('7', int(parFlickr.get())/30)

    if swCancelar == 1: #El proceso finalizó porque el usuario cancelo
        mensaje.insert(END, 'Proceso Cancelado')
    else:   #El proceso finalizó correctamente
        mensaje.insert(END, 'Proceso Finalizado')

    #Se imprimen los totales en el listBox
    mensaje.insert(END, 'Total Urls: ' + str(len(totalUrls) + urlsRepetidas))
    mensaje.insert(END, 'Total Imagenes Descargadas: ' + str(x))
    mensaje.insert(END, 'Total Imagenes Descargadas Google: ' + str(m1))
    mensaje.insert(END, 'Total Imagenes Descargadas Yahoo: ' + str(m2))
    mensaje.insert(END, 'Total Imagenes Descargadas Ecosia: ' + str(m3))
    mensaje.insert(END, 'Total Imagenes Descargadas Dogpile: ' + str(m4))
    mensaje.insert(END, 'Total Imagenes Descargadas Yandex: ' + str(m5))
    mensaje.insert(END, 'Total Imagenes Descargadas Foter: ' + str(m6))
    mensaje.insert(END, 'Total Imagenes Descargadas Flickr: ' + str(m7))
    mensaje.insert(END, 'Total Urls Repetidas: ' + str(urlsRepetidas))
    mensaje.insert(END, 'Total Imagenes no descargadas: ' + str(cPerdidas))

    try:    #Se escriben los totales en metadatos
        a = file(consulta + '.txt', 'a')
        a.write('\nTotal Urls: ')
        a.write(str(len(totalUrls) + urlsRepetidas ))
        a.write('\nTotal Imagenes Descargadas: ')
        a.write(str(x))
        a.write('\nTotal Imagenes Descargadas Google: ')
        a.write(str(m1))
        a.write('\nTotal Imagenes Descargadas Yahoo: ')
        a.write(str(m2))
        a.write('\nTotal Imagenes Descargadas Ecosia: ')
        a.write(str(m3))
        a.write('\nTotal Imagenes Descargadas Dogpile: ')
        a.write(str(m4))
        a.write('\nTotal Imagenes Descargadas Yandex: ')
        a.write(str(m5))
        a.write('\nTotal Imagenes Descargadas Foter: ')
        a.write(str(m6))
        a.write('\nTotal Imagenes Descargadas Flickr: ')
        a.write(str(m7))
        a.write('\nTotal Urls Repetidas: ')
        a.write(str(urlsRepetidas))
        a.write('\nTotal Imagenes no descargadas: ')
        a.write(str(cPerdidas))
        a.write('\nHora finalización Proceso')
        a.write(time.strftime("%I:%M:%S"))
        a.close()
    except:
        mensaje.insert(END, 'Algo fallo con los metadatos')
        a.close()

    desbloqueo()
```

- Metodos internos

  Método que realiza la descarga de las imágenes

```
#Métodos Internos
def download(url, NOMBRE):  #Método encargado de la descarga de imagenes dada una url y el nombre
    #Llamamos las variables locales
    global x #contador global
    global hdr #Headers
    global cPerdidas

    try:

        req = urllib2.Request(url, headers=hdr)  #Hacemos la petici{on
        furl = urllib2.urlopen(req, timeout=30)  #Abrimos la url
        f = file(NOMBRE,'wb+')  #Creamos un nuevo archivo en el directorio actual
        f.write(furl.read())    #Escribimos en el archivo el contenido de lo que trae la url
        f.close()               #Cerramos el archivo
        metadatos(NOMBRE, url)  #Llamamos al metodo Metadatos
        mensaje.insert(END, "Descarga completada -->" + NOMBRE)
        return 1

    except:
        cPerdidas = cPerdidas + 1
        mensaje.insert(END, 'No fue posible descargar la imagen')
        return 0
```

- Método que crea el documento de texto

```
def metadatos (NOMBRE, url):  #Método encargado crear y escribir en un archivo plano la informacion perteneciente
    al origen de la imagen(lugar de descarga y nombre)
    #Este metodo recibe el nombre de la imagen local y la url de donde se descargo
    global x
    global consulta
    try:
        a = file(consulta + '.txt', 'a')    #Creamos o abrimos un archivo txt donde se puede escribir al final
        if x == 0:
            a.write('Fecha: ')
            a.write(time.strftime("%d/%m/%y"))
            a.write('\n')
            a.write('Hora: ')
            a.write(time.strftime("%I:%M:%S"))
            a.write('\n')
            a.write('Busqueda realizada: ')
            a.write(consulta)
            a.write('\n')
            a.write('\n')
            a.close()
        else:
            a.write('Nombre de la imagen:')
            a.write(NOMBRE)
            a.write('\nurl:')
            a.write(url)
            a.write('\nNombre original:')
            a.write(url[url.rfind('/',0,url.find(extension(NOMBRE)))+1 : url.find(extension(NOMBRE)) ])
            #El nombre original lo capturamos encontrando la informacion que hay entre el formato y el / que le antecede
            a.write('\n')
            a.write('\n')
            a.close()   #Cerramos el archivo
    except:
        a.close()
        mensaje.insert(END, 'Algo fallo con los metadatos')
```

- Método que encargado de hacer los reemplazos y dejar normalizadas las urls para su uso

```
def reemplazos(url):    #Método encargado de hacer reemplazos de HTML URL Encoding
    a = url
    a = a.replace('%2525252525','%')
    a = a.replace('%25252525','%')
    a = a.replace('%252525','%')
    a = a.replace('%2525','%')
    a = a.replace('%25','%')
    a = a.replace('%20',' ')
    a = a.replace('%23','#')
    a = a.replace('%3a',':')
    a = a.replace('%3A',':')
    a = a.replace('%3f','?')
    a = a.replace('%3F','?')
    a = a.replace('%2f','/')
    a = a.replace('%2F','/')
    a = a.replace('%26','&')
    a = a.replace('%3c','<')
    a = a.replace('%3C','<')
    a = a.replace('%7c','|')
    a = a.replace('%7C','|')
    a = a.replace('%25','%')
    a = a.replace('%3e','>')
    a = a.replace('%3E','>')
    a = a.replace('%3b',';')
    a = a.replace('%3B',';')
    a = a.replace('%3d','=')
    a = a.replace('%3D','=')
    a = a.replace('\u003','=')
    a = a.replace(';ru=','')
    a = a.replace('_m','_c')
    a = a.strip(' ')

    return a    #Retorna la Url con los reemplazos realizados
```

- Método para hallar la extensión de la imagen a descargar

```
def extension(url):     #Método encargado de hallar la extension de la url
    ##Extensiones a Trabajar = ['.jpg','.png','.gif','.jpeg']
    url = url.lower()   #Colocamos la Url en minuscula para estandizar la búsqueda
    #El método find retorna un indice cuando encunetra la cadena especificada, en caso contrario retorna -1
    #si url.find retorna algo diferente a -1, el formato es la cadena especificada
    if url.find(".jpg") != -1:
        ext = ".jpg"
    elif url.find(".png") !=-1:
        ext = ".png"
    elif url.find(".gif") != -1:
        ext = ".gif"
    else:
        ext = ".jpeg"
    return ext  #Retornamos la extensión
```

- Proceso central

```
#Proceso central, simulamos las busqueda que se harian en los diferentes motores,
#capturamos el contenido html y por medio de html scrape y el uso de expresiones regulares capturamos las urls de las imagenes a dercargar
def proceso(var, num):  #var es un indice que indica el motor y num el numero de solicitudes que se van a hacer a los motores(simulando el paginado o el infinite scroll)

    #Llamamos a las variables globales
    global totalUrls    #Arreglo que almacena todas las urls que se van capturando
    global x            #Contador global
    global hdr          #Headers
    global consulta     #Query a buscar
    global swCancelar
    global urlsRepetidas

    cMotor = 0  #Contador Imagenes descargadas por motor
    for j in range(num):    #Controla el paginado
        if swCancelar == 1:
            break
        #los diferentes motores
        #1 para google
        #2 para yahoo
        #3 para ecosia
        #4 para dogpile
        #5 para yandex
        #6 para Foter
        #7 para Flickr

        #Dejamos lista la url que contiene el query y va a simular la busqueda en los motores
        if var == '1':      #Google
            #q indica la busqueda, ijn indica el paginado y start el index la primera imagen--------google tiene capacidad de 100 urls por paginado
            url = "https://www.google.com.co/search?q=" + consulta + "&hl=es&site=imghp&biw=1023&bih=979&tbm=isch&ijn=" + str(j) +
            "&ei=ZsgZV5ThN8aumQH-u6qABA&start=" + str((j*100)) +
            "&ved=0ahUKEwiUrJHR0qHMAhVGVyYKHf6dCkAQuT0ILigB&vet=10ahUKEwiUrJHR0qHMAhVGVyYKHf6dCkAQuT0ILigB.ZsgZV5ThN8aumQH-u6qABA.i"
        elif var =='2':     #Yahoo
            #p indica la busqueda, b indica el index de la primera imagen y iid=Y es el paginado--------Yahoo tiene la capacidad de aproximadamente
            #40-60 urls por paginado(el index tiene que hacer aumentos de 60)
            url = "https://co.images.search.yahoo.com/search/images?fr=yfp-t-726&fr2=piv-web&o=js&p=" + consulta +
            "&tab=organic&tmpl=&nost=1&b=" + str((j*60+1)) + "&iid=Y." + str(j) + "&ig=628b8fd11b4547718f00000000d82ccd&rand=1461527127355"
        elif var == '3':    #Ecosia
            #p es el paginado y q es la busqueda----------Ecosia tiene la capacidad de 50 urls por paginado
            url= "https://www.ecosia.org/images?aspect=&f=false&p="+str(j)+"&q="+consulta
        elif var == '4':    #Dogpile
            #qsi es el index de la primera imagen y q la busqueda---------Dogpile tiene la capacidad de 35 urls por paginado
            url = "http://www.dogpile.com/info.dogpl/search/images?qsi=" + str(j*35+1) + "&q=" + consulta + "&fcoid=4&fcop=results-bottom&fpid=2"
        elif var =='5':     #Yandex
            #p es el paginado y text es la consulta------------Yandex tiene la capacidad de 28 urls aproximadamente
            url = "https://www.yandex.com/images/search?p=" + str(j) + "&text=" + consulta
        elif var == '6':    #Foter
            #q es la busqueda y page el paginado---------Foter tiene la capacidad de 50 urls por paginado
            url ="http://foter.com/search/instant/?qm" +consulta.replace('+','%20') + "&page="+str(j+1)
        else: #Flickr, utilzando api, se debe solcitar una llave(api_key)
            #text es la consulta page el pagina----------flickr tiene la capacidad de aproximadamente 30-50 urls por paginado
            url = "https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key=33962d4190957d9fabdf691515ac175e&text=" + consulta.replace('+','%20') +
            "&extras=url_o&sort=relevance&per_page=50&page=" + str(j+1)

        try:
            req = urllib2.Request(url, headers=hdr)     #Envia los Headers para poder abrir la url
            urlContent = urllib2.urlopen(req).read()    #Almacena el contenido html de la url dada
            #Con expresiones regulares extraemos las urls de interes del contenido html
            if var == '1':
                imgUrls = (re.findall('","ou":"(.*?)",', urlContent))
            elif var == '2':
                imgUrls = (re.findall('&imgurl=(.*?)&rurl', urlContent))
            elif var == '3':
                aux = re.findall('a href="(.*?)"', urlContent)
                imgUrls = []
                swE = 0
                for h in range(len(aux)):
                    if swE==0:
                        imgUrls.append(aux[h])
                        swE = 1
                    else:
                        swE=0
            elif var == '4':
                aux = re.findall('ru%3d(.*?)%26', urlContent)
                imgUrls = []
                swE = 0
                for h in range(len(aux)):
                    if swE==0:
                        imgUrls.append(aux[h])
                        swE = 1
                    else:
                        swE=0  #Se aumenta la h debido a que el motor trae 2 veces la misma url
            elif var == '5':
                imgUrls = (re.findall('"img_href":"(.*?)",', urlContent))
            elif var == '6':
                imgUrls = re.findall('src="(.*?)"', urlContent)
            else:
                imgUrls = (re.findall('url_o":"(.*?)"', urlContent))

            for i in range(len(imgUrls)):   #Recorremos las urls extraidas
                if swCancelar == 1:
                    break

                imgUrls[i]= reemplazos(imgUrls[i])  #Llamamos el metodo de reemplazos para limpiar la url y dejarla util para el uso de urllib

                if totalUrls.count(imgUrls[i]) != 0:    #Si la url ya esta fue descargada, no la vuelva a descargar
                    sw = 1
                    urlsRepetidas = urlsRepetidas + 1
                else:
                    sw = 0  #switch para controlar que no descargue 2 veces la misma url

                if sw == 0:     #Proceda con la descarga si la url no se repite


                    if not imgUrls[i].lower().startswith('http://') and not imgUrls[i].lower().startswith('https://'):
                    #Validacion para determinar si la url tiene http o https
                        imgUrls[i] = 'http://%s'%imgUrls[i]     #En caso de no ser asi, lo concatenamos

                    totalUrls.append(imgUrls[i])    #Agregue al arreglo que contiene todas las urls la nueva

                    mensaje.insert(END, imgUrls[i])

                    formato = extension(imgUrls[i]) #Extraemos la extension de la imagen
                    nombreImagen = consulta.replace('"','') + str(x) + formato  #Se le asigna un nombre a la imagen a descargar(Nombre como quedara grabada en disco)
                    mensaje.insert(END, nombreImagen)
                    aumento = download(imgUrls[i], nombreImagen)  #Descarga la imagen
                    x = x+aumento     #Aumente el contador global
                    cMotor = cMotor + aumento

        except:
            mensaje.insert(END, 'Algo fallo con una url')

    return cMotor #Retorna el numero de imagenes descargadas por el motor

#Fin métodos internos
```

- Interfaz

```
#Main
try:
    ##interfaz
    #-----------------------------NOTA-------------------------------------------------
    #A todos los controladores se les debe pasar commo primer parametro la variable root(ventana)

    #-------------------------------------------------------------------------------
    root = Tk() #inicializa la ventana
    root.title('Bitzi - Software Image Scraping')
    root.geometry('1000x850')    #Asignamos tamaño a la ventana
    root.resizable(0,0)
    fuente = tkFont.Font(family='Century Gothic',size='12')
    color1='white'  #color background
    color2='blue'   #color font
    #0B0B61
    root.configure(background=color1) #toda la ventana queda del color1
    icono = find_data_file('icono.ico')
    root.iconbitmap(icono)


    Label(root, text='Ingrese la consulta', width=30, bg=color1, fg=color2, font=fuente).grid(row=0, columnspan=2) #Label consulta
    cons = Entry(root, width=100, relief=SOLID, font=fuente)   #textbox para ingresar consulta
    cons.grid(row=1, columnspan=2)  #Añadir al grid
    #campos de grid
    ruta= StringVar() #almacenará la ruta para guardar las imagenes
    Label(root, textvariable=ruta, width=50, bg=color1, fg=color2).grid(row=2) #label ruta
    btnRuta = Button(root, text='Ruta', command=hallarRuta, width=30, fg=color2, relief=RIDGE, font=fuente)
    btnRuta.grid(row=2, column=1)
    Label(root, text='Motores',width=50, bg=color1, fg=color2, font=fuente).grid(row=3, column=0)
    Label(root, text='Imagenes aproximadas', width=50, bg=color1, fg=color2, font=fuente).grid(row=3, column=1)

    #declaracion variables chk que tendran el estado de los checkbox de los navegadores
    chkGoogle =IntVar()
    CH1 = Checkbutton(root, text='Google', variable=chkGoogle, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH1.grid(row=5)
    chkYahoo =IntVar()
    CH2 = Checkbutton(root, text='Yahoo', variable=chkYahoo, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH2.grid(row=6)
    chkDogpile = IntVar()
    CH3 = Checkbutton(root, text='Dogpile', variable=chkDogpile, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH3.grid(row=7)
    chkEcosia = IntVar()
    CH4 = Checkbutton(root, text='Ecosia', variable=chkEcosia, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH4.grid(row=8)
    chkYandex = IntVar()
    CH5 = Checkbutton(root, text='Yandex', variable= chkYandex, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH5.grid(row=9)
    chkFoter = IntVar()
    CH6 = Checkbutton(root, text='Foter', variable= chkFoter, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH6.grid(row=10)
    chkFlickr = IntVar()
    CH7 = Checkbutton(root, text='Flickr', variable= chkFlickr, bg=color1, fg=color2, selectcolor=color1, font=fuente)
    CH7.grid(row=11)

    CH1.select()

    #declaración variables par para almacenar cantidad de páginas a consultar
    parGoogle = StringVar(root)
    parGoogle.set('500') #Se le asigna valor por defecto
    w1 = OptionMenu(root,parGoogle, '100', '200', '300', '400', '500', '600')   #Se especifican las opciones del menu
    w1.config(bg=color1, fg=color2, font=fuente)
    w1.grid(row=5, column=1)    #Se añade al grid

    parYahoo = StringVar(root)
    parYahoo.set('420')
    w2 = OptionMenu(root, parYahoo , '35', '70', '105', '140', '175', '210', '245',
                    '280', '315', '350', '385', '420', '455')
    w2.config(bg=color1, fg=color2, font=fuente)
    w2.grid(row=6, column=1)

    parDogpile = StringVar(root)
    parDogpile.set('420')
    w3 = OptionMenu(root, parDogpile , '30', '60', '90', '120', '150', '180', '210',
                    '240', '270', '300', '330', '360', '390', '420', '450')
    w3.config(bg=color1, fg=color2, font=fuente)
    w3.grid(row=7, column=1)

    parEcosia = StringVar(root)
    parEcosia.set('420')
    w4 = OptionMenu(root,parEcosia , '35', '70', '105', '140', '175', '210', '245',
                    '280', '315', '350', '385', '420', '455')
    w4.config(bg=color1, fg=color2, font=fuente)
    w4.grid(row=8, column=1)

    parYandex = StringVar(root)
    parYandex.set('510')
    w5 = OptionMenu(root, parYandex, '30', '60', '90', '120', '150', '180', '210',
                    '240', '270', '300', '330', '360', '390', '420', '450', '480',
                    '510', '540', '570')
    w5.config(bg=color1, fg=color2, font=fuente)
    w5.grid(row=9, column=1)

    parFoter = StringVar(root)
    parFoter.set('500')
    w6 = OptionMenu(root, parFoter , '50', '100', '150', '200', '250', '300', '350',
                    '400', '450', '500')
    w6.config(bg=color1, fg=color2, font=fuente)
    w6.grid(row=10, column=1)

    parFlickr = StringVar(root)
    parFlickr.set('2400')
    w7 = OptionMenu(root, parFlickr , '150', '300', '450', '600', '750', '900', '1050',
                    '1200', '1350', '1500', '1650','1800','1950','2100', '2250', '2400')
    w7.config(bg=color1, fg=color2, font=fuente)
    w7.grid(row=11, column=1)
```

```
    mensaje = Listbox(root, width=100, relief=SOLID, font=fuente) #Cuadro de texto que contiene los mensajes del programa
    mensaje.grid(row=13, columnspan=2)
    scrollbar = Scrollbar(root)     #Se añade scrollbar para el cuadro de texto
    scrollbar.grid(row=13, column=1, sticky='ns', ipadx=186) #el parametro ns rellena de norte a sur el grid
    mensaje.config(yscrollcommand=scrollbar.set)
    scrollbar.config(command=mensaje.yview)

    btnBuscar = Button(root, text='Buscar', command=buscar, width=30, fg=color2, relief=RIDGE, font=fuente)     #Boton de buscar
    btnBuscar.grid(row= 15)
    btnRestablecer = Button(root, text='Restablecer valores por defecto', command=restablecer, width=30, fg=color2, relief=RIDGE, font=fuente)
    #Boton de Restablecer
    btnRestablecer.grid(row=15, column=1)
    btnSalir = Button(root, text='Salir', command=salir, width=30, fg=color2, relief=RIDGE, font=fuente)    #Boton de Salir
    btnSalir.grid(row=16, column=1)
    btnCancelar = Button(root, text='Cancelar', command=cancelar, width=30, fg=color2, relief=RIDGE, font=fuente)   #Boton de cancelar
    btnCancelar.grid(row=16)
    btnCancelar.config(state=DISABLED)

    rutaImagen = find_data_file('itm-logo.GIF')
    img = PhotoImage(file=rutaImagen)
    labelLogo = Label(image = img, bd=0)
    labelLogo.grid(row=17, columnspan=2)

    root.mainloop()  #Mantiene la ventana abierta
except:
##    mensaje.insert(END, 'Algo fallo con un motor')
  print 'Algo fallo con la interfaz'
##    pass
   # sys.exit(1)
```

Apéndice B Etiquetado de imágenes

- **Interfaz**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="webEtiquetado._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Etiquetado de imágenes</title>
    <link rel="shortcut icon" href="images/iconoEtiqueta.ico"/>
    <style type="text/css">
        .auto-style2 {
            text-align: center;
        }
        .auto-style3 {
            font-family: "Century Gothic";
            font-size: medium;
        }
    </style>
</head>
<body style="background-image: url('images/fondo2.jpg'); background-repeat: no-repeat; background-size:cover ">
    <script type="text/javascript">
        function etiqueta(callback) {
            var a = prompt("Ingrese etiqueta");
            document.getElementById('<% = HiddenField1.ClientID%>').value = a;
            callback();
        }
    </script>
    <form id="form1" runat="server">
        <div>

            <table align="center" width ="100%">
                <tr>
                    <td colspan="2" style="text-align: center">
        <asp:Label ID="Label1" runat="server" Text="Ingrese nombre de la escena" ForeColor="White" style="font-size: medium; font-family: 'Century Gothic'"></asp:Label>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">

        <asp:TextBox ID="TxtEtiqueta" runat="server" Width="500px"></asp:TextBox>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2"> </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">

        <asp:ImageButton ID="ImageButton1" runat="server" OnClientClick="etiqueta()" OnClick="ImageButton1_Click" Enabled="False" />

                    </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2"> </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">

                        <asp:ListBox ID="lbCoordenadas" runat="server" Width="500px" CssClass="auto-style3"></asp:ListBox>

                    </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">
                        <asp:Label ID="lblMensaje" runat="server" ForeColor="White" CssClass="auto-style3"></asp:Label>
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2">
        <asp:Button ID="btnCargarImagen" runat="server" OnClick="btnCargarImagen_Click" Text="Nueva Imagen" CssClass="auto-style3" Height="51px" Width="173px" />
                    </td>
                    <td style="text-align: center">
                        <asp:Button ID="btnTerminado" runat="server" style="text-align: center" Text="Imagen Terminada" Enabled="False" OnClick="btnTerminado_Click" BorderStyle="Ridge"
                    </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">
                         </td>
                </tr>
                <tr>
                    <td class="auto-style2" colspan="2">
                        <asp:Image ID="Image1" runat="server" ImageUrl="~/images/ITM.png" Height="187px" Width="436px" />
                    </td>
                </tr>
            </table>
            <asp:HiddenField ID="HiddenField1" runat="server" />

            <asp:HiddenField ID="hfNombre" runat="server" />
            <asp:HiddenField ID="hfRuta" runat="server" />
            <asp:HiddenField ID="hffecha" runat="server" />
```

- **Librerías y variables globales.**

```
//Librerías
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.IO;
using System.Globalization;

namespace webEtiquetado
{
    public partial class _Default : System.Web.UI.Page
    {
        #region "Variables Globales"

        //Variables que se necesitan para guardar el archivo de texto con las etiquetas
        string nombreImagen;
        string fecha;
        string ruta;

        #endregion
```

- **Métodos públicos.**

```
        #region "Métodos Públicos"

        public void escritura(string copiar) // Método encargado de copiar el contenido del archivo
        {
            try
            {
                using (StreamWriter file = //Se usa el comando using para evitar que el StreamWriter quede abierto, Se crea una instancia del ultimo
                new StreamWriter(ruta, true))
                {
                    file.Write(copiar); //Se copia en la última línea del documento
                    //file.Close();
                }
            }
            catch (Exception)
            {
                lblMensaje.Text = "No se puedo escribir correctamente en el archivo";
                throw;
            }
        }

        public bool esImagen(string archivo) //Método encargado de verificar los formatos válidos
        {
            string extension = "";
            extension = Path.GetExtension(archivo);// Se halla la extensión del archivo
            extension = extension.ToLower();
            if (extension == ".jpg" || extension == ".jpeg" || extension == ".png" || extension == ".gif" || extension == ".wmf" || extension == ".bmp")
            {
                return true;
            }
            else
            {
                return false;
            }
        }
```

```
public string imgAleatoria() //Método encargado de devolver la ruta de una imagen aleatoria
{
    string rutaImagen = "";
    string ruta = "C:\\Users\\Johan\\Pictures\\Saved Pictures"; //Carpeta donde se alojan las imagenes que serán cargadas
    //string ruta = "D:\\Imágenes\\imagenes tesis";  //Carpeta donde se alojan las imagenes que serán cargadas
    int c = 0; //Contador de archivos contenidos en la carpeta
    int random;

    try
    {
        var archivos = Directory.GetFileSystemEntries(ruta); //Almacena los nombres de todo lo que hay en la ruta
        c = archivos.Length;
        random = new Random().Next(0, c);

        while (!esImagen(archivos[random])) //Mientras el nombre del archivo no tenga los formatos validos entra
        {
            if (Directory.Exists(archivos[random]))//Si el nombre del arhivo es una carpeta entra
            {
                ruta = archivos[random];//Cambia la ruta al nuevo directorio
                archivos = Directory.GetFileSystemEntries(ruta);

                c = archivos.Length;
                random = new Random().Next(0, c);
            }
            else //Si es un archivo, se genera nuevo random
            {
                random = new Random().Next(0, c);
            }
        }
        rutaImagen = archivos[random];

    }
    catch (Exception)
    {
        lblMensaje.Text = "Algo fallo con el directorio de las imágenes";
        throw;
    }
    return rutaImagen;

}
```

- **Código controles**

```
protected void Page_Load(object sender, EventArgs e)
{
    //Se llena las variables globales con los valores de los hidden fields
    nombreImagen = hfNombre.Value;
    fecha = hffecha.Value;
    ruta = hfRuta.Value;
}

protected void btnCargarImagen_Click(object sender, EventArgs e)    //Método encargado de cargar una imagen en el image Button
{
    byte[] buff = null;     //Array de bytes para leer la imagen
    string imagen64 = "";   //String que contiene la imagen en base64
    string urlImagen = "";  //Ruta de la imagen
    lblMensaje.Text = "";
    TxtEtiqueta.Text = "";

    ImageButton1.Enabled = true;
    //urlImagen = "D:\\Imágenes\\Johan\\2013-07-587471.jpg";
    urlImagen = imgAleatoria(); //Obtenemos la ruta de una imagen aleatoria a cargar

    try
    {
        buff = File.ReadAllBytes(urlImagen); //Se lee los bytes de la imagen
        imagen64 = Convert.ToBase64String(buff);    //Se convierten los bytes a Base64
        ImageButton1.ImageUrl = "" + imagen64; //Se carga el imageButton con la imagen
        nombreImagen = urlImagen.Substring(urlImagen.LastIndexOf("\\") + 1);    //Se halla el nombre de la imagen
        hfNombre.Value = nombreImagen;  //Se guarda el nombre de la imagen en un hidden field con la finalidad de conservar el valor cuando haya un page Load
        //lblMensaje.Text = nombreImagen;
        CultureInfo esp = new CultureInfo("es-ES"); //Se crea una instancia de culture info "es-Es" para hallar la hora en dicho formato
        fecha = DateTime.Now.ToString(esp).Replace(":", "-").Replace("/", "-");//Se halla la fecha, y se reemplaza los : y / por guiones por motivos posteriores
        hffecha.Value = fecha; //Se guarda la fecha  en un hidden field con la finalidad de conservar el valor cuando haya un page Load
    }
    catch (Exception)
    {
        lblMensaje.Text = "Algo salio mal al cargar la imagen";
    }

}
```

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e) //Método encargado de caputar coordenadas y etiquetas
{
    string etiqueta = "";
    string coordenadaX = "";
    string coordenadaY = "";
    string lines = "";

    etiqueta = HiddenField1.Value; //Se captura el valor de la etiqueta
    if (!String.IsNullOrEmpty(etiqueta))
    {
        btnTerminado.Enabled = true;
        btnCargarImagen.Enabled = false;

        ruta = "C:\\Users\\Johan\\Pictures\\Etiquetas"; //Ruta donde se guarda los archivos de texto con las coordenadas capturadas

        try
        {
            //se cambia el formato a txt
            nombreImagen = nombreImagen.Replace(".jpg", ".txt");
            nombreImagen = nombreImagen.Replace(".jpeg", ".txt");
            nombreImagen = nombreImagen.Replace(".png", ".txt");
            nombreImagen = nombreImagen.Replace(".gif", ".txt");
            nombreImagen = nombreImagen.Replace(".wmf", ".txt");
            nombreImagen = nombreImagen.Replace(".bmp", ".txt");
            ruta = ruta + "\\" + fecha + " " + nombreImagen; //Se guarda la ruta del archivo plano correspondiente a la imagen actual

            hfRuta.Value = ruta; //Se guarda la ruta del archivo plano en un hidden field con la finalidad de conservar el valor cuando haya un page Load

            coordenadaX = e.X.ToString(); //Se captura el valor de la coordenada X
            coordenadaY = e.Y.ToString(); //Se captura el valor de la coordenda Y
            lines = "(" + coordenadaX + ", " + coordenadaY + ") - " + etiqueta;
            //escritura(lines); //Se llama al metodo encargado de copiar el contenido del archivo
            lbCoordenadas.Items.Add(lines);

        }
        catch (Exception)
        {
            lblMensaje.Text = "Algo salio mal";
        }
    }
}
```

```
protected void btnTerminado_Click(object sender, EventArgs e) //Captura la etiqueta de la escena final y desbloquea el  control para cargar una nueva imagen
{
    string strContenido = "";

    ImageButton1.Enabled = false;
    btnTerminado.Enabled = false;
    btnCargarImagen.Enabled = true;
    strContenido = "Nombre de la imagen: " + nombreImagen;
    strContenido = strContenido + "\r\nEscena general de la Imagen: " + TxtEtiqueta.Text;
    strContenido = strContenido + "\r\nCoordenada    Etiqueta\r\n";
    foreach (var item in lbCoordenadas.Items)
    {
        strContenido = strContenido + item + "\r\n";
    }
    strContenido = strContenido.Replace("-", "");
    escritura(strContenido);  //Se llama al metodo encargado de copiar el contenido del archivo
    lblMensaje.Text = "Imagen etiquetada con exito";
    hfNombre.Value = "";
    hffecha.Value = "";
    hfRuta.Value = "";
    TxtEtiqueta.Text = "";
    lbCoordenadas.Items.Clear();
}
```

- **Script en Javascript**

```
<script type="text/javascript">
    function etiqueta(callback) {
        var a = prompt("Ingrese etiqueta");
        document.getElementById('<% = HiddenField1.ClientID%>').value = a;
        callback();
    }
</script>
```

FIRMA ESTUDIANTES Edna Lizeth Ocampo Londoño

Johan Ordoñez

Xiomara E. Palacios M.

FIRMA ASESOR 

FECHA ENTREGA: Junio 29 de 2016

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD _____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____