UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

HERRAMIENTA OSINT PARA LA RECOPILACIÓN DE INFORMACIÓN DE USUARIOS EN REDES SOCIALES

OSINT TOOL FOR COLLECTING USER INFORMATION ON SOCIAL NETWORKS

Realizado por
FRANCISCO GUERRERO PÉREZ

Tutorizado por
EDUARDO GUZMÁN DE LOS RISCOS

Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE 2022

Resumen

En base al crecimiento exponencial de las redes sociales y de la enorme cantidad de información que los ciudadanos depositamos día a día en ellas, este trabajo busca aprovechar ese volumen de información que existe en la web con el fin de investigar a personas potencialmente peligrosas y analizar su actividad en la red para poder trazar un perfil básico con su información personal y las personas con las que se relaciona.

Para ello, se ha desarrollado una aplicación híbrida que ejecuta de forma asíncrona Osintgram (una herramienta que permite obtener información de Instagram sobre un usuario) y realiza una búsqueda en la página web Socialbearing.com (web que permite analizar los perfiles de Twitter de un usuario). Los datos obtenidos mediante estas fuentes son descargados, tratados y posteriormente se muestran al usuario de forma clara.

Se han empleado tecnologías como Node.Js, Electron, JavaScript o HTML, además de herramientas que como Puppeteer para la realización de Web scrapping o Node-pty junto a Xterm.js para la creación de un terminal donde realizar las peticiones a la herramienta Osintgram. Además, se ha utilizado la API de Geocoding que proporciona Google para poder visualizar en el mapa las distintas ubicaciones obtenidas durante la búsqueda de información.

En líneas generales, este trabajo persigue no sólo proporcionar una herramienta útil, cómoda y sencilla para la investigación de individuos peligrosos, sino de alguna forma demostrar que la información pública que se puede obtener de una persona en la red es mucho más reveladora y extensa de lo que se imagina. Por tanto, queremos transmitir el mensaje de que, aunque lo olvidemos, toda nuestra información privada que es subida las redes sociales puede ser obtenida mucho más fácil de lo que creemos.

Palabras clave: **Extracción de información de la web, Redes sociales, OSINT, Electron JS, NodeJS.**

Abstract

Due to the exponential growth of social media and the enormous amount of information that citizens upload each day on them, this project aims to use that information that exists on the web with the goal of investigating potentially dangerous people and analyse their activity on the network to being able to craft a basic profile with their personal information and the people they are more connected with.

For that purpose, we have developed a hybrid application which executes asynchronously Osintgram (a tool which allows us to perform analysis on Instagram account of any users) and execute a search on the website Socialbearing.com (a website for analysing on Twitter account). The data collected from these sources is then downloaded, treated and then showed to the user in a clear way.

There have been used technologies such as Node.Js, Electron, JavaScript or html, including tools like Puppeteer for the Web scrapping part or Node-pty along Xterm.js for the creation of the terminal where Osintgram commands were launched. Furthermore, the Geocoding API from Google has been used to visualize the multiple locations obtained during the search of information.

In general lines, this project not only pursued being able to provide a useful, handleable and simple tool for the investigation of dangerous individuals, but also to show that the public information of a person can be obtained in a more revelling and extensive way than we imagine. For this, we want to transmit that, even though we forget it, all our

private information that is uploaded to the social media can be obtained more easily than we think.

Keywords: Web scrapping, Social media, OSINT, Electron JS, Node JS.

Índice

1

Introducción

1.1 Motivación

El número de datos abiertos que los usuarios depositan en la red aumenta cada día con el auge de Internet y las redes sociales. En 2020, el número de usuarios alcanzaba la cifra de 3.805 millones. Se estima que al menos una persona consume 5 plataformas de redes sociales en un mes. Todo esto se traduce en una web sobrecargada con información personal y datos públicos disponibles al alcance de todo aquel que quiera aprovecharlos.

Puede parecer alarmante, pero esto también nos ofrece la oportunidad de obtener información de personas de cierto interés (delincuentes, sospechosos, etc.), en especial dentro del ámbito de la criminología y en general de la investigación policial.

A la hora de investigar a un individuo, la utilización de estos datos puede ayudar a facilitar en gran medida el proceso de obtención de información. Además, gracias a las redes sociales, también es posible analizar y estudiar el entorno donde se encuentra: lugares más visitados, personas más cercanas, etc.

1.2 Objetivos

Como objetivo principal se pretende desarrollar una herramienta software que permita obtener información de un usuario y generar un modelo de los datos que permita visualizar toda la información de ese usuario y sus posibles relaciones con otros.

Entre las funcionalidades principales que tendrá la herramienta, podemos destacar las siguientes:

- **Información básica de un usuario:** se buscará información de un individuo a través de su nombre de usuario (nombre, gustos, posts, tuits, etc.).
- **Galería de un usuario:** se mostrarán las imágenes del individuo seleccionado.
- **Redes empleadas:** las redes sociales que se emplearán son Twitter e Instagram.
- **Estudio de la geolocalización:** se mostrará un mapa con las distintas localizaciones obtenidas.
- **Análisis social:** se mostrarán los usuarios con los que más se relaciona el individuo seleccionado.

1.3 Metodología de trabajo

La metodología empleada para realizar este trabajo ha sido Scrum, con iteraciones o *sprints* de 3 semanas donde se ha desarrollado de manera secuencial la aplicación finalizando cada sprint con una demostración del producto desarrollado hasta el momento.

Cada *sprint* ha contado con las siguientes fases de trabajo: planificación de requisitos a desarrollar durante esa fase, a continuación, la implementación de los requisitos seleccionados, y por último el desarrollo de pruebas y su respectiva documentación.

1.5 Estructura de la memoria

Esta memoria se encuentra organizada según los siguientes capítulos:

Capítulo 2: Tecnologías utilizadas y estudio del arte

En este capítulo enumeramos y estudiamos las principales tecnologías y herramientas empleadas para el desarrollo de nuestra aplicación. Además, abordamos el tema central de este proyecto que es OSINT y sus distintas funcionalidades.

Capítulo 3: Especificación del sistema

A lo largo de este capítulo realizamos una especificación del sistema que hemos desarrollado con distinción entre los requisitos funcionales y los no funcionales que la componen.

Capítulo 4: Diseño

Es aquí donde se realiza un diseño del programa planteado como respuesta a los requisitos expuestos. Se mostrará un diagrama de componentes donde analizaremos la estructura del programa, y el diseño de los datos que se emplearán para analizar un usuario objetivo y sus relaciones con otros.

Capítulo 5: Implementación

En este capítulo hablaremos sobre el proceso de creación de la aplicación y explicaremos el funcionamiento y la organización de los archivos que componen nuestro programa.

Capítulo 6: Conclusiones y líneas futuras

El capítulo abordará los principales problemas y dificultades encontrados a la hora de trabajar en el proyecto, una serie de posibles mejoras y líneas futuras que podría tomar, y unas conclusiones a base de reflexión sobre el trabajo realizado.

Capítulo 7: Referencias

Se enumerarán las referencias empleadas a lo largo del desarrollo de esta memoria, realizadas en el formato APA.

Apéndice A: Manual de Instalación

En este apéndice veremos los requisitos necesarios para la ejecución de nuestra aplicación, la guía para la instalación de esta, y por último una serie de problemas comunes que pueden ocurrir a la hora de utilizarla.

Apéndice B: Manual de Usuario

En esta última sección hablaremos sobre cómo utilizar la aplicación y las distintas funcionalidades que ofrece al usuario.

2

Tecnologías utilizadas y estudio del arte

2.1 Lenguajes de programación

2.1.1 JavaScript

El lenguaje de programación JavaScript (también conocido como JS) es famoso por su utilidad y su frecuente uso a la hora de realizar comandos para las páginas web y en entornos de programación como *Node.js*, *Apache CouchDB*, y *Adobe Acrobat* (*MDN*, s. f.).

Se encuentra dentro de la categoría de lenguajes de programación orientada a objetos, concretamente como un lenguaje basado en prototipos. Presenta, además, la capacidad de soportar otros estilos de programación como la Imperativa o la declarativa.

Por último, JavaScript emplea como estándar ECMAScript (ECMA-262), estandarizado por la organización ECMA International.

2.1.2 HTML

El lenguaje HTML (también conocido como Lenguaje de Marcas de Hipertexto, o *Hypertext Markup Language*) compone una de las partes fundamentales del desarrollo web (W3C) (*MDN*, s. f.; *Wikipedia, La Enciclopedia Libre*, s. f.). Este lenguaje permite estructurar y organizar el contenido de cualquier página web. Es necesario resaltar que suele ir acompañado siempre por otras tecnologías que dotan a las páginas web de complejidad y funcionalidad, como CSS o JavaScript.

El término hipertexto, que forma parte del nombre de este lenguaje, indica que gracias a HTML las páginas web pueden establecer referencias o enlaces que conecten distintas páginas entre sí. Esta característica es lo que dota al lenguaje de su importancia, y lo que ayudó a construir lo que hoy en día conocemos como Internet.

Para finalizar, hoy en día HTML se encuentra en su versión 5, con novedades destacables que permiten el diseño de la Web 3.0 (web semántica), un mayor número de etiquetas y mejoras, y nuevas APIs para facilitar el desarrollo.

2.1.3 CSS

El lenguaje CSS (*Cascading Style Sheets*) es el utilizado para estilizar el contenido de los documentos HTML o XML. Mediante los distintos elementos que presenta el lenguaje podemos especificar cómo deben ser renderizados los componentes de una página web de forma que se adapten al estilo establecido (*MDN*, s. f.).

Por último, al igual que HTML, este lenguaje se encuentra estandarizado por la organización W3C (*World Wide Web Consortium*), lo que asegura el crecimiento de la web.

2.2 Herramientas

2.2.1 Node.js

Node.js es un entorno de ejecución de JavaScript *open-source* orientado a la ejecución de eventos asíncronos y la creación de aplicaciones web en varias plataformas como Windows, Linux, Mac OS, etc. (*Node.Js*, s. f.; *W3School*, s. f.).

Entre las ventajas que presenta este entorno de ejecución podemos señalar el manejo cómodo de la concurrencia, simplicidad a la hora de programar el *back-end* y el *front-end*, y una gran cantidad de librerías que permiten realizar un gran número de funcionalidades y con fácil instalación.

Además, Node.js permite generar páginas web dinámicas y controlar de forma sencilla el manejo de ficheros o de datos dentro de una base de datos.

2.2.2 Chromium

Chromium es un navegador *open-source* centrado en la seguridad, estabilidad y la rapidez a la hora de navegar por Internet (*Wikipedia, La Enciclopedia Libre*, s. f.; *Chromium-Projects*, s. f.). Como su nombre indica, fue desarrollado por Google, y muchos otros navegadores web como Microsoft Edge u Opera se basan en su código. Su origen tiene lugar en 2008, y desde entonces este navegador ha estado siempre en desarrollo.

En cuanto a su arquitectura, este navegador es similar al resto de navegadores. Su arquitectura entra dentro de la categoría de multi-procesos, donde un proceso principal ejecuta y crea procesos hijos para realizar subtareas.

2.2.3 Electron

Electron es el framework que hemos empleado para el desarrollo de esta aplicación. Permite desarrollar aplicaciones híbridas empleando JavaScript, HTML y CSS. Funciona mediante la combinación de Chromium y Node.js, y al igual que estos nos proporciona la capacidad de crear aplicaciones en varias plataformas mientras la base del código se mantiene en JavaScript (*Electron*, s. f.).

Entre las ventajas por las que Electron es un framework tan útil se encuentra la simplificación que otorga a procesos como: la instalación, el mantenimiento, los *crash logs*. A la hora de programar una aplicación web podemos olvidarnos del entorno en el que se ejecutará, dejando que Electron se encargue de comprobar la versión del

navegador o las características del equipo, lo que significa la labor del programador y permite centrarse en las funcionalidades de la aplicación que queremos desarrollar.

2.2.4 Osintgram

La herramienta Osintgram compone una parte fundamental de este trabajo, ya que es la encargada de recolectar, analizar y descargar toda la información disponible sobre la cuenta de Instagram de un usuario concreto. Se trata de una herramienta OSINT gratuita desarrollada en Python y con objetivos puramente académicos (*GitHub - Datalux/Osintgram*, s. f.).

Entre la información que podemos obtener con esta herramienta podemos destacar: datos básicos de un perfil, la lista de usuarios que menciona un usuario y por los que ha sido mencionado, el conjunto de fotos subidas por el usuario a Instagram y el conjunto de direcciones registradas al usuario en sus posts.

Cabe señalar, que las funcionalidades más avanzadas como son obtener las fotos y las direcciones registradas de un usuario no están disponibles para todas las cuentas de Instagram. Para ello, la cuenta debe ser pública, o en el caso de que sea privada debe ser seguida por la cuenta que le indiquemos a la aplicación a la hora de instalarla.

2.2.5 Bootstrap

El framework gratuito y de código abierto Bootstrap nos permite desarrollar de forma cómoda el *front-end* de una aplicación web (*Bootstrap*, s. f.; *TechTarget*, s. f.). Está diseñado para permitir que las páginas web que emplean sus distintos componentes y diseños reaccionen de forma dinámica a cambios en el tamaño o la orientación del dispositivo donde se proyecten.

Bootstrap está apoyado en HTML, CSS y JavaScript, y ofrece un sistema de fácil de usar basado en cuadrículas donde se pueden distribuir de forma dinámica los componentes HTML que componen nuestra página web.

Podemos destacar que el uso de este framework es muy sencillo gracias a que únicamente requiere conocimientos básicos de HTML, CSS y JavaScript, permite una configuración inicial rápida para convertir nuestro proyecto de forma fácil, y se encuentra extensamente documentado en Internet facilitando de esta forma su uso.

2.3 ¿Qué es OSINT? ¿Para qué sirve?

Las siglas de OSINT (*Open Source Intelligence*) referencian el proceso de obtener, tratar y analizar un conjunto de datos abiertos con el fin de extraer información y valor de ellos (*INCIBE-CERT*, s. f.; *Derecho de La Red*, s. f.). Podemos visualizar mejor este proceso gracias al diagrama de la Figura 2.1.

Hoy en día podemos encontrar multitud de fuentes para obtener estos datos abiertos: los medios de comunicación online, información de los canales oficiales del gobierno, datos científicos o ensayos de bibliotecas web, o el canal en el que vamos a centrarnos en este trabajo, las redes sociales.

Figura 2.1 Diagrama del proceso OSINT

Este gran conjunto de datos es empleado en ámbitos muy variados como el financiero, el militar, el márquetin, etc. Sin embargo, el más destacable dentro del contexto de

este trabajo es la ciberseguridad, con aplicaciones como el análisis de hacking ético (comúnmente conocido como *Pentesting*), la defensa frente a ataques o amenazas potenciales, o establecer la huella digital de una amenaza.

Por último, la metodología OSINT nos permite realizar búsquedas de información sobre un usuario (tanto en redes sociales como en documentos). Esta aplicación es la seleccionada para la elaboración de este trabajo, y se utiliza frecuentemente para realizar los denominados "tests de ingeniería social" (permite reconocer la información disponible de un usuario en la red para evitar los ataques de ingeniería social, que persiguen manipular a una víctima utilizando la información pública de la misma) (*INCIBE*, s. f.).

3

Especificación del sistema

3.1 Requisitos Funcionales

- **RF-01:** El usuario podrá buscar información de un individuo a través del nombre de usuario.
- **RF-02:** el usuario podrá indicar el número máximo de cuentas relacionadas con el objetivo que desea que se muestren por pantalla.
- **RF-03**: el usuario podrá abrir el directorio donde se almacenan los datos descargados por la aplicación.
- **RF-04:** El sistema mostrará el nombre del individuo.

- **RF-05:** El sistema mostrará la información del perfil del individuo (biografía, seguidores, personas seguidas, número de tuits, alcance, visualizaciones, retuits, favoritos y respuestas).
- **RF-06:** El sistema mostrará la imagen de perfil del individuo.
- **RF-07:** El sistema mostrará las ubicaciones visitadas por el individuo (localización del perfil, localización de los posts, localización de los tuits, etc).
- **RF-08:** El sistema mostrará los usuarios con los que el individuo más se relaciona.
- **RF-09:** El sistema mostrará las fotos de Instagram del individuo.
- **RF-10:** El sistema analizará los tuits del individuo y los clasificará por tipo y por sentimiento.

3.2 Requisitos No Funcionales

- **RNF-01:** El sistema empleará la página *socialbearing.com* para la obtención de tuits del individuo y el análisis de estos.
- **RNF-02:** El sistema empleará la herramienta *Osintgram* para la obtención de post y el análisis de estos.
- **RNF-03:** El sistema presentará una interfaz clara y adaptable.
- **RNF-04:** El sistema mostrará por defecto los 5 usuarios con los que más se relaciona el individuo.
- **RNF-05:** El sistema capturará y mostrará los errores por pantalla, permitiendo que la ejecución continue.
- **RNF-06:** El sistema obtendrá la información de un usuario de forma asíncrona permitiendo la actualización progresiva de los datos mostrados sin que el sistema se bloquee.
- **RNF-07**: El sistema obtendrá la información básica (datos personales) en menos de 5 segundos.

3.3 Casos de Uso

A partir de los requisitos especificados previamente, se han podido generar los casos de uso que se estudiarán a continuación. Estos se encuentran representados por el diagrama presente en la Figura 3.1.

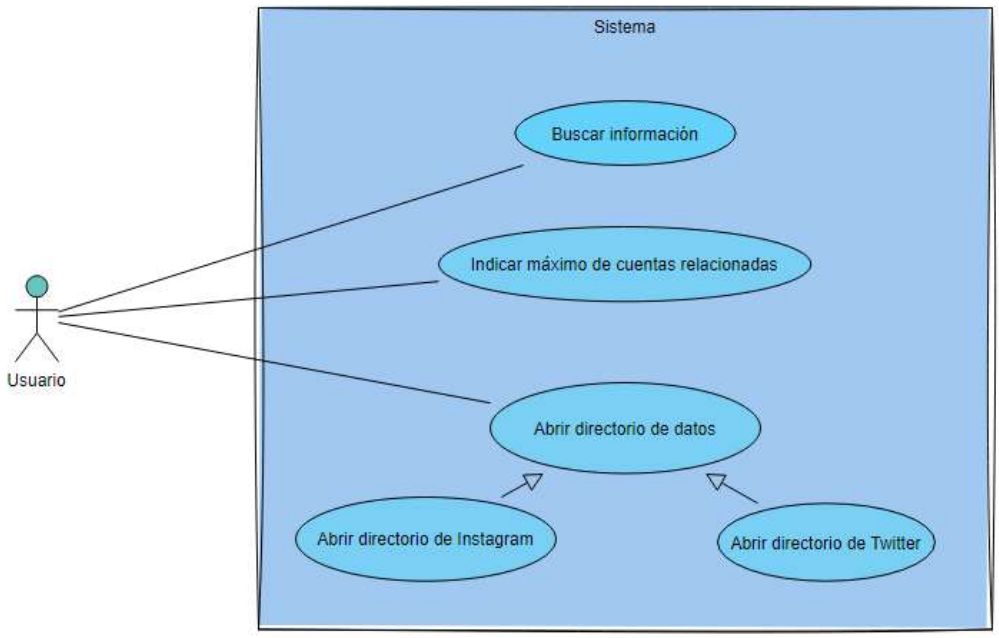

Figura 3.1 Diagrama de casos de uso

3.3.1 Buscar información

- **Descripción**: el usuario podrá buscar información de un individuo a través del nombre de usuario.
- **Pre-condición**: el usuario se encuentra en la ventana principal de la aplicación y el nombre de usuario a introducir dispone de una cuenta de Twitter e Instagram.
- **Post-condición**: la información del objetivo se muestra por la pantalla principal y se permite la búsqueda de otro usuario objetivo.
- **Secuencia normal**:
    1. El usuario introduce el nombre de usuario en el formulario y presiona el botón de buscar.
    2. El sistema bloquea el botón de búsqueda.

3. El sistema descarga, analiza y trata la información del objetivo.
4. El sistema muestra la información del objetivo y desbloquea el botón de búsqueda.

- **Excepciones**:
  - **Paso 1**: El usuario introduce el nombre de usuario en el formato incorrecto.
    - El sistema muestra un mensaje de error por pantalla
    - Se cancela el caso de uso

3.3.2 Indicar máximo de cuentas relacionadas

- **Descripción**: el usuario podrá indicar el número máximo de cuentas relacionadas con el objetivo que desea que se muestren por pantalla.
- **Pre-condición**: el usuario se encuentra en la ventana principal de la aplicación.
- **Post-condición**: la información del objetivo se muestra por la pantalla principal y se muestran como máximo el número de usuarios relacionados indicado.
- **Secuencia normal**:
  1. El usuario especifica el número máximo de cuentas relacionadas.
  2. El usuario introduce el nombre de usuario en el formulario y presiona el botón de buscar.
  3. El sistema bloquea el botón de búsqueda.
  4. El sistema descarga, analiza y trata la información del objetivo.
  5. El sistema muestra la información del objetivo (donde se encuentran como máximo el número de cuentas indicado) y desbloquea el botón de búsqueda.
- **Excepciones**:
  - **Paso 1**: El usuario introduce un número negativo.
    - El realizará la búsqueda considerando el número introducido como 0.

3.3.3 Abrir directorio de Instagram

- **Descripción**: el usuario podrá abrir el directorio donde se almacenan los datos descargados de Instagram por la aplicación.

- **Pre-condición**: el usuario se encuentra en la ventana principal de la aplicación.
- **Post-condición**: el directorio donde se almacena la información de Instagram se muestra por pantalla.
- **Secuencia normal**:
    1. El usuario presiona el botón para abrir el fichero de datos de Instagram.
    2. El sistema muestra el directorio de datos por pantalla.

3.3.4 Abrir directorio de Twitter

- **Descripción**: el usuario podrá abrir el directorio donde se almacenan los datos descargados de Twitter por la aplicación.
- **Pre-condición**: el usuario se encuentra en la ventana principal de la aplicación.
- **Post-condición**: el directorio donde se almacena la información de Twitter se muestra por pantalla.
- **Secuencia normal**:
    1. El usuario presiona el botón para abrir el fichero de datos de Twitter.
    2. El sistema muestra el directorio de datos por pantalla.

4

Diseño

4.1 Diagrama de componentes

Para el diseño de nuestro sistema hemos realizado un diagrama de componentes (Figura 4.1) con el objetivo de describir y analizar la organización y la estructura de nuestro sistema. En este se muestran todos los elementos que componen nuestro programa y la relación que existe entre los mismos.

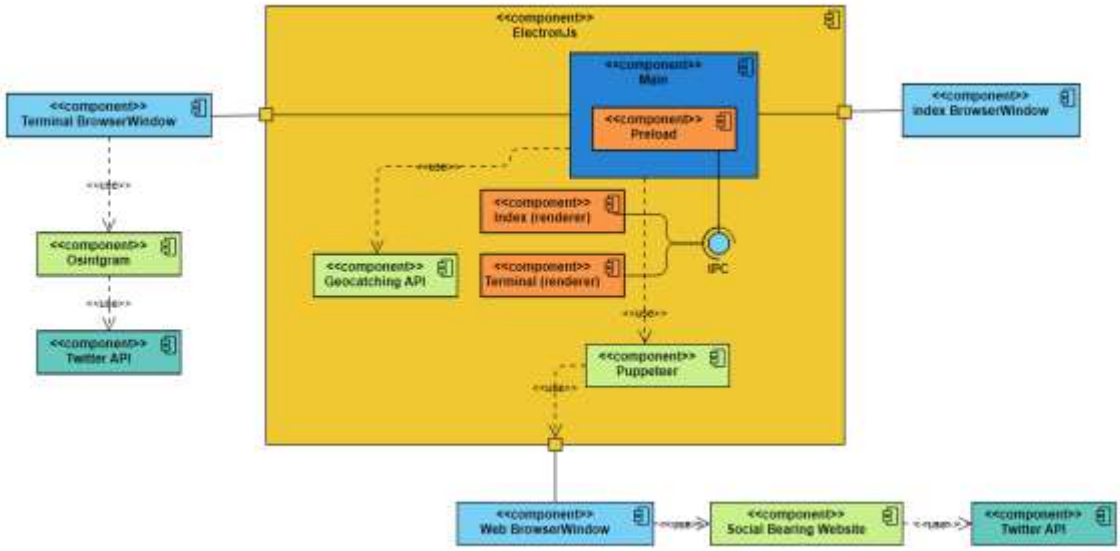

Figura 4.1 Diagrama de componentes

Vamos a estudiar cada uno de los componentes que se muestran en el diagrama de la Figura 4.1:

- **Main**: controlador principal de la aplicación, encargado de controlar el flujo principal de trabajo y de la creación de los distintos navegadores. Se comunica con los procesos *renderer* mediante mensajes (IPC) a través de canales definidos en el componente *Preload*.
- **Preload**: es el encargado de definir la comunicación entre los distintos procesos. Básicamente se encarga de exponer las APIs y sus distintos métodos para que puedan ser utilizadas en otros procesos.
- **Index (renderer)**: componente encargado de renderizar la ventana principal de la aplicación *Index BrowserWindow*. Permite modificar dinámicamente la ventana principal añadiendo la información que se obtiene del análisis de un usuario.
- **Index BrowserWindow**: la vista principal de la aplicación, donde se puede buscar un usuario y se muestran los datos obtenidos tras realizar las distintas llamadas asíncronas necesarias.
- **Terminal (renderer)**: componente encargado de renderizar la ventana del terminal *Terminal BrowserWindow***.** Permite definir el comportamiento de la terminal y modificar su estado.

- **Terminal BrowserWindow**: vista del terminal, accesible una vez se comienza la búsqueda de un usuario. Permite ejecutar el programa *Osintgram* a través de comandos de consola.
- **Geocatching API**: API proporcionada por Google que permite obtener las coordenadas de una localización dado su nombre.
- **Puppeteer**: librería de Node que proporciona una API de alto nivel para poder controlar Chrome. Permite controlar la ventana *Web BrowserWindow* para obtener los datos necesarios.
- **Web BrowserWindow**: ventana del navegador que permite acceder a la página *Social Bearing*, accesible una vez se comienza la búsqueda de un usuario. Permite descargar la información de una cuenta de Twitter.

4.2 Diseño de los datos

Como se ha comentado al comienzo de esta memoria, la aplicación a desarrollar aprovecha el gran número de datos que un usuario tiene almacenado en la red. Estos datos son obtenidos por varios canales (como *Osintgram* y *Social Bearing*) y se encuentran en formatos CSV y JSON. Sin embargo, para mayor comodidad a la hora de emplearlos dentro de nuestra aplicación los convertiremos todos al formato JSON.

Para poder trabajar con estos datos, hemos preparado una representación gráfica de cada uno de ellos. Gracias a esto, podemos analizar qué parámetros del usuario se obtienen con estas herramientas y la jerarquía en la que se organizan.

4.2.1 Info.json

Figura 4.2 Diagrama del conjunto de datos Info.json

Este conjunto de datos representa la información básica de un perfil de Instagram. Podemos obtenerlo gracias a la aplicación *Osintgram*, empleando el comando *info*. El archivo resultante tiene como nombre el nombre del usuario analizado junto al comando de la forma: *username_info.json*.

A continuación, estudiaremos cada uno de los atributos que componen el diagrama de la Figura 4.2:

- **Id**: número que identifica la cuenta de Instagram de un usuario de forma inequívoca.
- **Full_name**: nombre completo del objetivo.
- **Biography**: biografía del usuario.
- **Edge_followed_by**: número de seguidores del usuario.
- **Edge_follow**: número de personas a las que sigue el usuario.
- **Is_business_account**: valor booleano que indica si la cuenta del usuario es de empresa o no. Este atributo es condición necesaria para que existan los

atributos opcionales indicados en el diagrama, aunque no garantiza que estos estén definidos por el usuario de la cuenta.

- **Is_verified**: valor booleano que indica si la cuenta del usuario está verificada por Instagram.
- **Profile_pic_url_hd**: dirección URL donde se encuentra alojada la imagen de perfil del usuario.
- **Email**: email profesional del usuario. (Puede estar vacío)
- **Connected_fb_page**: página de Facebook asociada. (Puede estar vacío)
- **Whatsapp_number**: número de WhatsApp del usuario. (Puede estar vacío)
- **City_name**: nombre de la ciudad donde se encuentra la empresa. (Puede estar vacío)
- **Address_street**: nombre de la calle donde se encuentra la empresa. (Puede estar vacío)
- **Contact_phone_number**: número de teléfono de contacto de la empresa. (Puede estar vacío)

4.2.2 Tagged.json

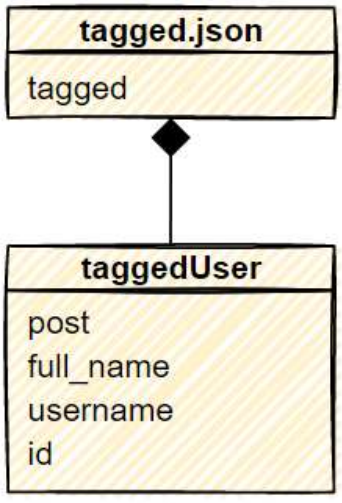

Figura 4.3 Diagrama del conjunto de datos tagged.json

El siguiente conjunto de datos representa la lista de usuarios que han sido mencionados por el objetivo en sus posts. Podemos obtenerlo gracias a la aplicación *Osintgram*, empleando el comando *tagged*. El archivo resultante tiene como nombre el nombre del usuario analizado junto al comando de la forma: *username_tagged.json*.

A continuación, estudiaremos los atributos que componen este archivo representado en la Figura 4.3:

- **Tagged**: array que contiene información básica sobre los usuarios que han sido mencionados por el objetivo. Hemos denominado a cada elemento de este Array como *taggedUser*.

Estos son los atributos que definen esta clase:

- **Post**: número de posts de Instagram donde el objetivo ha mencionado al usuario. Este atributo nos permite ordenar la lista *Tagged* para conocer a qué usuarios ha mencionado más el objetivo y de esta forma ver con qué usuarios se relaciona más.
- **Full_name**: nombre completo de la persona mencionada.
- **Username**: nombre de usuario de la persona mencionada.
- **Id**: número que identifica la cuenta de Instagram de la persona mencionada de forma inequívoca.

4.2.3 Users_who_tagged.json

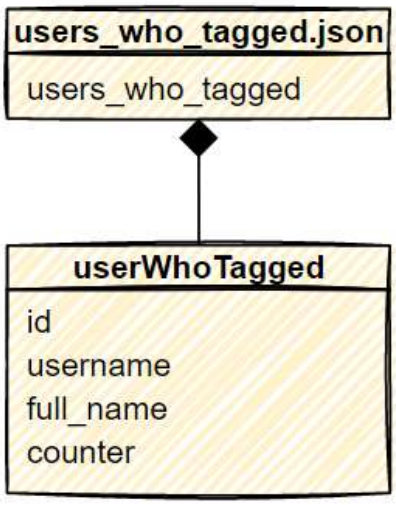

Figura 4.4 Diagrama del conjunto de datos users_who_tagged.json

El conjunto de datos de a continuación representa la lista de usuarios que mencionaron al objetivo. Podemos conseguir este dato gracias a la aplicación

Osintgram, empleando el comando *wtagged*. El archivo resultante tiene como nombre el nombre del usuario analizado junto al comando de la forma: *username_users_who_tagged.json*.

A continuación, vamos a analizar los atributos del diagrama de la Figura 4.4:

- **Users_who_tagged**: array que contiene la información básica sobre los usuarios que han mencionado al objetivo. Hemos denominado a cada elemento de este Array como *userWhoTagged*.

Esta clase está definida de la siguiente forma:

- **Id**: número que identifica la cuenta de Instagram de la persona que menciona al objetivo de forma inequívoca.
- **Username**: nombre de usuario de la persona que menciona al objetivo.
- **Full_name**: nombre completo de la persona que menciona al objetivo.
- **Counter**: número de posts de Instagram donde el objetivo ha sido mencionado por el usuario. Este atributo nos permite ordenar la lista *users_who_tagged* para conocer qué usuarios han mencionado más al objetivo y de esta forma ver qué usuarios se relacionan más con él.

4.2.4 Addrs.json

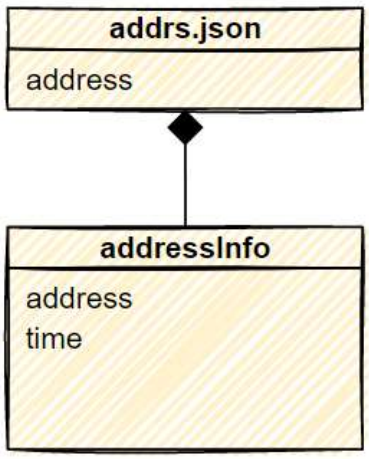

Figura 4.5 Diagrama del conjunto de datos addrs.json

Este conjunto de datos hace referencia a las distintas localizaciones que el usuario ha indicado en sus posts. Este conjunto de datos se puede conseguir gracias a la

aplicación *Osintgram*, empleando el comando *addrs*. El archivo resultante tiene como nombre el nombre del usuario analizado junto al comando de la siguiente forma: *username_addrs.json*.

Vamos a estudiar los atributos de este conjunto de datos representado en la Figura 4.5:

- **Address**: array que contiene información básica sobre las direcciones asociadas al objetivo. Hemos denominado cada elemento de este Array como *addressInfo*.

Este subconjunto está compuesto por los siguientes atributos:

- **Address**: dirección de la ubicación establecida por el objetivo. Es importante destacar que no se incluyen las coordenadas, únicamente el nombre de la ubicación. Esto supone necesario el uso de *geocoding* para poder representar en un mapa las ubicaciones obtenidas.
- **Time**: fecha y hora de la publicación del post donde se ha obtenido la dirección.

4.2.5 Tweet-report.json

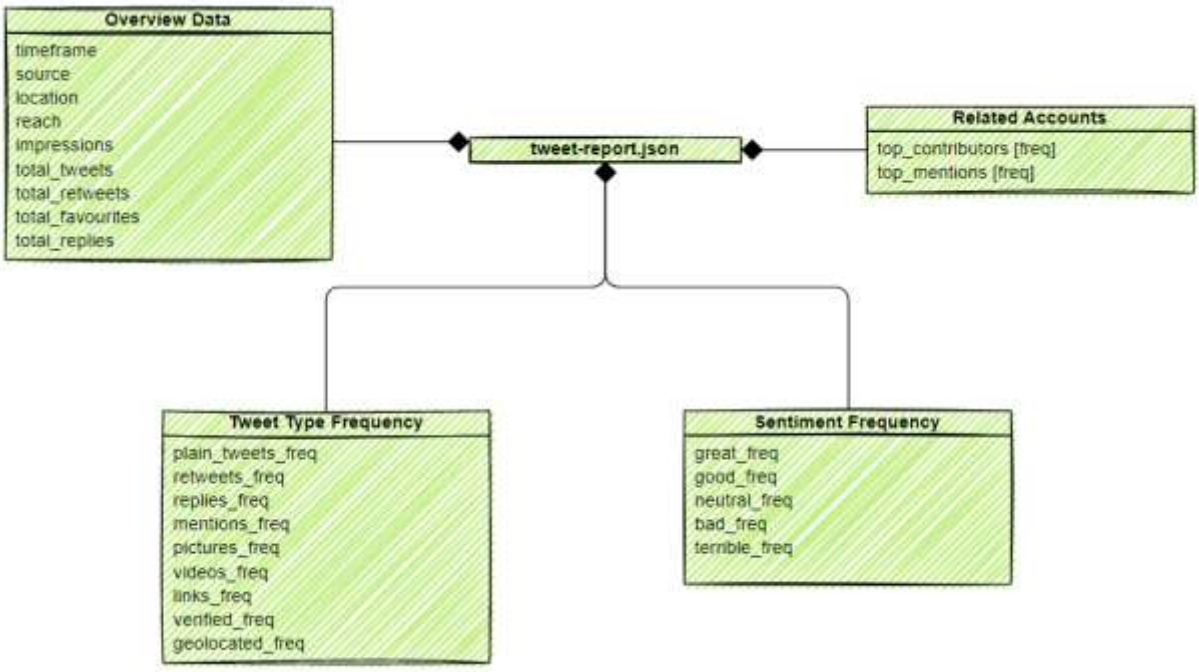

Figura 4.6 Diagrama del conjunto de datos tweet-report.json

El siguiente conjunto de datos ilustrado en la Figura 4.6 es el encargado de representar toda la información de un usuario a través de su perfil de Twitter. Estos datos se pueden conseguir gracias a la página web *Social Bearing*. El archivo obtenido, sin embargo, es un CSV, de modo que es necesario su conversión a json. Al hacerlo, obtenemos un JSON Array donde cada posición del array representa una fila del CSV. Para poder estudiar mejor la estructura, hemos dividido el archivo en cuatro componentes diferenciables. A continuación, estudiaremos cada uno de estos componentes y los atributos que los definen:

Primero veamos el componente *Overview Data*. Este se encarga de mostrar la información más básica al objetivo en lo referente a estadísticas y datos del propio perfil.

- **Source**: dispositivo principal desde donde se utiliza la cuenta de Twitter.
- **Location**: localización del perfil de Twitter.
- **Timeframe**: la franja de tiempo en la que se encuentran los tweets analizados, desde el más antiguo hasta el último.
- **Reach**: estimación del número de cuentas que han encontrado el contenido del objetivo dentro de la franja de tiempo indicada. Se basa en el número de seguidores y menciones.
- **Impressions**: estimación de las visualizaciones recibidas por el objetivo dentro de la franja de tiempo indicada. Se basa en el número de seguidores del usuario, el número de retuits, e incluye usuarios repetidos.
- **Total_tweets**: número total de tweets analizados en la franja indicada.
- **Total_retweets**: número total de retuits realizados.
- **Total_favourites**: número total de favoritos realizados.
- **Total_replies**: número total de respuestas.

A continuación, estudiemos el componente *Tweet Type Frequency*. Como su nombre indica, este se encarga de analizar los tipos de tuits del usuario y calcular la frecuencia de cada uno.

- **Plain_tweets_freq**: frecuencia de publicación de tuits con texto plano.
- **Retweets_freq**: frecuencia de retuits.

- **Replies_freq**: frecuencia de respuesta.
- **Mentions_freq**: frecuencia de mención.
- **Pictures_freq**: frecuencia de publicación de tuits con imágenes.
- **Videos_freq**: frecuencia de publicación de tuits con vídeos.
- **Links_freq**: frecuencia de publicación de tuits con enlaces.
- **Verified_freq**: frecuencia de publicación de tuits verificados.
- **Geolocated_freq**: frecuencia de publicación de tweets con localización.

Ahora veamos el componente *Sentiment Frequency*. La página web de *Social Bearing* dispone de un analizador básico de sentimientos que puntúa cada tuit en función de las palabras que lo componen, de forma que cada palabra es probada con un conjunto de palabras predefinidas positivas y negativas (*How Is Tweet Sentiment Analysis Calculated? « Social Bearing*, s. f.). Este análisis debe ser tomado como una indicación, pues como afirma el creador de la página web, siempre habrá falsos positivos, además de conceptos como la ironía o el sarcasmo que son difíciles de detectar.

- **Great_freq**: frecuencia de tuits con puntuación muy positiva.
- **Good_freq**: frecuencia de tuits con puntuación positiva.
- **Neutral_freq**: frecuencia de tuits con puntuación neutra.
- **Bad_freq**: frecuencia de tuits con puntuación negativa.
- **Terrible_freq**: frecuencia de tuits con puntuación muy negativa.

Por último, estudiemos el componente *Related Accounts*. Este es el encargado de ayudar a visualizar la relación que existe entre el objetivo y otros usuarios en la red social.

- **Top_contributors [freq]**: lista con los usuarios que más apariciones y sus respectivas frecuencias en el hilo de Twitter del objetivo. Cabe señalar que el propio objetivo puede aparecer en esta lista ya que sus propias publicaciones se tienen en consideración.
- **Top_mentions [freq]**: lista con los usuarios más mencionados por el usuario y sus respectivas frecuencias.

5

Implementación

5.1 Main.js

El archivo principal de nuestra aplicación es *Main.js*, siendo este el encargado de crear las distintas ventanas, ejecutar los procesos asíncronos de búsqueda de información y procesar los datos para mostrarlo todo por pantalla.

En esta sección estudiaremos cómo funciona y cómo se organiza este archivo, dividiendo en secciones las principales funcionalidades que se dan.

5.1.1 Creación de la ventana principal

Al iniciar la aplicación, lo primero que ejecutamos es una función para limpiar la carpeta de salida del programa *Osintgram*. Esto evita que el fichero se sobrecargue con sucesivas descargas de datos, manteniendo intactos los últimos datos que se han obtenido.

A continuación, definimos la ventana de ejecución principal de la aplicación, indicando el script *preload.js* para definir el funcionamiento interno y la vista *index.html* que define la estructura.

Por último, definimos dos *listeners* de eventos: uno para obtener el usuario cuando se pulsa el botón de buscar en la ventana principal, y otro para cuando se pulsa el botón para abrir la carpeta con la ubicación de los datos obtenidos.
En el caso de que se realice una búsqueda, primero se comprueba que el formato de entrada es válido (mostrando por pantalla si existe algún error) y si es correcto se ejecuta la búsqueda.

5.1.2 Ejecución asíncrona de búsqueda

La función llamada en caso de que el nombre de usuario introducido sea válido es la encargada de controlar los procesos asíncronos de búsqueda de información. Para ello, se crea una "promesa" con la función encargada de obtener los datos de Twitter (ejecutar una búsqueda en el navegador) y con la función encargada de obtener los datos de Instagram (ejecutar una serie de comandos en la terminal).

Al crear esta "promesa" podemos controlar cuando han finalizado los dos procesos asíncronos, y de esta forma habilitar el botón de búsqueda para realizar, si se desea, otra consulta.

5.1.3 Abrir terminal (Instagram)

Como hemos mencionado, esta función es la encargada de recopilar los datos de Instagram de un usuario mediante la aplicación *Osintgram* y el uso de un terminal de comandos. Además, manipula los datos obtenidos para poder enviarlos mediante un IPC al renderizador de la ventana principal (*index.js*).

Primero definimos la ventana encargada de mostrar la terminal, indicando la vista empleada *terminal.html*. Hablaremos sobre el funcionamiento de la terminal de forma más detallada en la sección de *terminal.js*, sin embargo, destacaremos que emplearemos IPC para analizar la respuesta de la terminal.

Este análisis de la salida del terminal es lo primero que realizamos, para establecer cuando es necesario cerrar la terminal o para mostrar por pantalla errores (en especial el de "challenge_error" porque muestra un enlace por pantalla que hay que tratar para que sea accesible, y el de "login_required" que requiere ejecutar un comando para resetear la configuración de *Osintgram*).

A continuación, ejecutamos los comandos necesarios para ejecutar *Osintgram* (y en caso de error instalamos las dependencias) y realizamos la búsqueda de información básica con el comando *info*.

Tras realizar una espera activa para que el comando se ejecute, intentamos leer el archivo JSON y enviamos su contenido al *renderer*. Es en este archivo donde se indica si el objetivo posee una cuenta de negocio (pública), necesaria para realizar las búsquedas avanzadas.

En caso de que la cuenta sea pública, ejecutamos los comandos *tagged*, *wtagged*, *photos*, y *addrs* siempre realizando una espera activa y enviando los datos al *renderer*. Sin embargo, cabe señalar que para las fotos y las direcciones el proceso es ligeramente distinto.
Al descargar las fotos, realizamos un filtrado para obtener únicamente las fotos del objetivo; cuando obtenemos las direcciones, empleamos la API de Google (Geocoding API) para obtener las coordenadas.

5.1.4 Abrir navegador (Twitter)

Esta función es la encargada de obtener los datos del usuario de Twitter mediante la página web *Social Bearing*, tratarlos y posteriormente enviarlos al *renderer* (*index.js*) para su visualización.

Lo primero que realizamos es una limpieza asíncrona del archivo "tweet-report.csv" en el caso de que exista para poder descargar el nuevo con los datos actualizados. A continuación, empleamos la librería *Puppeteer* para ejecutar y controlar Chrome con la url de la página *Social Bearing*.

El principal problema que presenta esta página es que es necesario esperar a que cargue completamente antes de descargar los datos CSV (de lo contrario el fichero resultante estará vacío), y que para poder obtener este fichero es necesario pulsar un botón de descarga de la página.

Es aquí donde *Puppeteer* destaca más: nos permite esperar a que la página cargue mediante una función, nos da libertad para establecer la ruta de descarga, y por último nos permite pulsar el botón para obtener los datos. Una vez obtenemos el archivo CSV lo transformamos a un archivo JSON Array y lo guardamos en el sistema.

El fichero JSON Array obtenido presenta como peculiaridad que la primera *key* es dinámica y depende del número de tuits analizados y el usuario objetivo, mientras que el resto de *keys* son constantes de la forma "*field2"*, "field3", etc. Es por eso que obtenemos todas las *keys*, para poder utilizar la primera y de esta forma saber el número de tuits analizados. En caso de que este número sea positivo enviamos los datos al *renderer*.

Por último, para realizar un análisis de la geolocalización es necesario obtener primero la dirección del archivo JSON Array. Como la arquitectura de este archivo no cambia nunca (podemos consultarla en la Figura 5.1), conocemos la fila donde se encuentra el primer resultado (7) y la *key* donde obtenerlo. De esta forma podemos obtener todas las direcciones mediante un bucle donde modifiquemos la fila de forma creciente hasta que obtengamos un campo vacío.

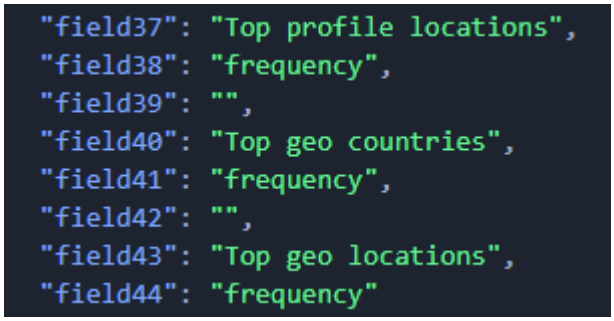

Figura 5.1 Campos necesarios para la geolocalización del archivo tweet-report.json

Una vez obtenemos una dirección que deseamos señalar en el mapa, empleamos la API de Google (Geocoding API) para obtener las coordenadas.

5.2 Index.js

Como hemos indicado con anterioridad, el archivo *Index.js* es el encargado de renderizar la página *Index.html* y de proporcionalidad dinamismo tanto para cargar los datos obtenidos como para la ejecución de componentes (los botones plegables de Instagram y Twitter).

A continuación, analizaremos las funciones que componen este archivo y el proceso empleado para transmitir los datos con otros componentes de la aplicación.

5.2.1 Búsqueda de usuario

Es necesario definir un *listener* para dar paso a la búsqueda de información de un usuario una vez se pulsa el botón de buscar. En este método obtenemos el nombre del usuario introducido y lo exponemos mediante el método definido en el archivo *preload.js*, además de almacenar la variable con el número de usuarios relacionados y plegar los botones de datos.

5.2.2 Botones y errores

Para controlar el flujo de ejecución de la aplicación y evitar acciones como ejecutar dos búsquedas simultáneas de usuarios o abrir el desplegable de datos de Twitter cuando el proceso no ha finalizado, se han empleado funciones para activar y desactivar los principales botones de la aplicación.

Además, hemos establecido métodos para añadir mensajes de error a la cabecera de la página y para limpiarlos todos cuando se reinicia la aplicación.

5.2.3 Datos básicos JSON

Como comentábamos en el apartado 5.1, al recibir los datos tanto de Twitter como de Instagram enviábamos estos al *renderer* para visualizarlo. Para ello disponemos de métodos personalizados para cada red social donde definimos los componentes dinámicos de la página *index.html* y actualizamos sus valores en función de los datos recibidos en el JSON.

Debemos señalar que en el caso de los "usuarios relacionados" generamos un enlace por cada usuario obtenido que modifica el objetivo a buscar para facilitar las búsquedas consecutivas, y este es agregado como elemento individual a una lista.
Para poder ordenar esta lista disponemos de varias funciones que ordenan los usuarios por número de menciones o número de veces mencionados.

5.2.4 Mapas

Por último, disponemos de varias funciones encargadas de controlar el mapa de la aplicación. Recibiendo como argumento un array con la información básica de una ubicación (latitud, longitud, dirección completa, etc.) podemos añadir marcadores personalizados para visualizar de forma cómoda los sitios obtenidos.

Además, podemos eliminar todos los marcadores para comenzar otra búsqueda y disponer del mapa en blanco.

5.3 Preload.js

Este archivo es el principal encargado de garantizar la comunicación entre procesos de la aplicación. Para la creación de los distintos canales se "exponen" APIs con funciones que pueden ser llamadas tanto por el proceso principal *Main.js* como por el *renderer Index.js*.

Para facilitar la comprensión de este archivo hemos definido cuatro canales: *username*, encargado únicamente de transmitir el nombre del usuario objetivo; *fileExplorer*, que comunica al proceso principal la orden de abrir el directorio indicado; *electronAPI*, como API generar para transmitir errores o variables básicas; *dataJson*, el responsable de la transmisión de datos desde el proceso principal al renderizador.

5.4 Terminal.js

El archivo *terminal.js* es el responsable de la creación y el funcionamiento de la terminal empleada para ejecutar los comandos de *Osintgram*. Para ello, empleamos

dos librerías especiales encargadas de proporcionar los métodos necesarios: *node-pty* y *xterm*.

En primer lugar, *xterm* nos permite disponer del componente "terminal" como tal en el frontend. Este componente es el que empleamos en la página *terminal.html* y permite escribir comandos en él. Es necesario resaltar que únicamente proporciona una interfaz para el usuario, y que por sí mismo *xterm* no permite ejecutar comandos.

Para poder añadirle la capacidad de ejecutar comandos en nuestro dispositivo empleamos la librería *node-pty*. Con esta librería somos capaces de ejecutar procesos y emular el funcionamiento de una terminal mediante *listeners* como explicamos en el apartado 5.1.3.

6

Conclusiones y líneas futuras

6.1 Problemas encontrados

A lo largo del desarrollo de esta aplicación nos hemos enfrentado a varios retos, algunos con un nivel de exigencia superior a otros. A continuación, comentaremos brevemente cuáles son los más destacados y los principales problemas derivados de los mismos.

Comenzaremos señalando el aprendizaje del principal *framework* empleado en el desarrollo de esta aplicación, *ElectronJs*. Pese a que presenta una magnífica guía a modo de iniciación en su página web, al principio cuesta un poco manejarse con esta

herramienta. Y no sólo eso, sino que debido a la poca experiencia previa con NodeJs también ha sido necesario realizar un aprendizaje de este entorno.

En cuanto a la herramienta *Osintgram*, sin duda ha sido el principal motivo de complicaciones de este proyecto. Es una herramienta fantástica a la hora de extraer información de Instagram, sin embargo, al tener que ejecutarse desde una terminal emulada aumenta la dificultad y complejidad de programación.

Desde el cómo utilizar la terminal, hasta cómo controlar cuando un comando ha sido ejecutado para continuar con la ejecución del siguiente, esta aplicación ha requerido de un gran número de modificaciones de código y horas de investigación.

Además, los problemas relacionados con la API de Instagram han dificultado el proceso de desarrollo y ralentizado el avance de la implementación. Estos ocurrían de forma ocasional mientras se trabajaba en la resolución de otro problema, y obligaban a detener la resolución del primero para trabajar en el segundo.

Por último, señalar la limitación de la API de Geocoding. Esta se encuentra limitada para garantizar que el servicio utilizado es gratuito, con una limitación diaria de peticiones y una finalización del servicio una vez que el periodo de prueba gratuito finalice.

6.2 Líneas futuras

Existen varias mejoras que se podrían implementar de cara a un desarrollo más largo de la aplicación. A continuación, propondremos algunas de estas:

6.2.1 Eliminación de dependencias

Las herramientas empleadas para la obtención de datos de redes sociales facilitan en gran medida el proceso de recolección de información y proporcionan métodos que ayudan a desarrollar de forma cómoda sobre estos. Sin embargo, es cierto que también suponen un problema a la hora de adaptar nuestra aplicación al desarrollo independiente de estas.

Como posible mejora, se consideraría la eliminación de las herramientas de *Osintgram* y *Social Bearing* para tratar directamente con las APIs de las redes sociales deseadas. De esta forma se controlaría mejor el proceso de obtención de datos y los posibles errores derivados del mismo, además de evitar posibles fallos que puedan surgir cuando se actualicen estas herramientas.

6.2.2 Análisis detallado de sentimientos

Como hemos mencionado con anterioridad, la página web *Social Bearing* proporciona un analizador de sentimientos basado en el *matching* de palabras. Esta técnica sirve de indicativo, pero quizás sería interesante sustituirla por otra más compleja y precisa para poder estudiar con más exactitud el contenido de los tweets de un usuario objetivo.

6.2.3 Búsquedas simultáneas de objetivos

Realizar búsquedas simultáneas de objetivos distintos supondría un aumento enorme de la efectividad de la aplicación, permitiendo investigar a un usuario objetivo y a sus usuarios relaciones al mismo tiempo.

El principal problema que plantearía esta mejora sería el consumo de recursos a la hora de ejecutar varias pantallas de forma asíncrona, por lo que sería necesario controlar y configurar los recursos asignables a esta búsqueda de datos.

6.2.4 Uso de otras redes sociales

Instagram y Twitter no son las únicas redes sociales en auge hoy en día. Como posible mejora de nuestra aplicación se podría plantear la incorporación de otras redes social, lo que permitiría manejar un volumen mayor de datos y de esta forma analizar mejor a un objetivo y sus distintas relaciones sociales.

6.2.5 Aplicación móvil

Quizás sería interesante permitir una mayor portabilidad de la aplicación desarrollando una versión capaz de funcionar en los dispositivos móviles. Como hemos

señalado, el ámbito pensado para esta aplicación es el policial y el de investigación de posibles individuos sospechosos. Dicho esto, esta investigación podría ser realizada en cualquier parte y no únicamente en sitios donde se disponga de un ordenador.

6.3 Conclusiones

Consideramos que el desarrollo de este proyecto ha concluido de forma positiva, con los objetivos establecidos cumplidos y un resultado satisfactorio. Además, trabajar con tantas herramientas y tecnologías de las aplicaciones web ha permitido adquirir un conocimiento considerable sobre estas.

Finalmente, cabe destacar que nos sentimos satisfechos con el trabajo realizado y consideramos que proporciona a cualquier interesado una buena idea sobre las capacidades que se disponen a la hora de desarrollar un proyecto de software.

Referencias

Acerca | Node.js. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://nodejs.org/es/about/

Bootstrap · The most popular HTML, CSS, and JS library in the world. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://getbootstrap.com/

Chromium (navegador) - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://es.wikipedia.org/wiki/Chromium_(navegador)

Chromium-projects. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.chromium.org/chromium-projects/

CSS | MDN. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://developer.mozilla.org/es/docs/Web/CSS

GitHub - Datalux/Osintgram: Osintgram is a OSINT tool on Instagram. It offers an interactive shell to perform analysis on Instagram account of any users by its nickname. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://github.com/Datalux/Osintgram

How is tweet sentiment analysis calculated? « Social Bearing. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://socialbearing.com/faqs/how-is-tweet-sentiment-analysis-calculated

HTML: Lenguaje de etiquetas de hipertexto | MDN. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://developer.mozilla.org/es/docs/Web/HTML

HTML5 - Wikipedia, la enciclopedia libre. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://es.wikipedia.org/wiki/HTML5

Ingeniería social | INCIBE. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.incibe.es/aprendeciberseguridad/ingenieria-social

Introducción | Electron. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.electronjs.org/es/docs/latest

Introduction to Node.js. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://nodejs.dev/en/learn/

JavaScript | MDN. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://developer.mozilla.org/es/docs/Web/JavaScript

Node.js Introduction. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.w3schools.com/nodejs/nodejs_intro.asp

OSINT - La información es poder | INCIBE-CERT. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.incibe-cert.es/blog/osint-la-informacion-es-poder

OSINT, ¿Qué es? ¿Para qué sirve? | Derecho de la Red. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://derechodelared.com/osint/

Usuarios de redes sociales en España. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.epdata.es/datos/usuarios-redes-sociales-espana-estudio-iab/382

What is a Bootstrap and how does it work? (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://www.techtarget.com/whatis/definition/bootstrap

XII Estudio de redes sociales 2021: TikTok y Twitch, al asalto de las redes clásicas. Facebook, en declive - Marketing 4 Ecommerce - Tu revista de marketing online

para e-commerce. (s. f.). Recuperado 9 de septiembre de 2022, a partir de https://marketing4ecommerce.net/xii-estudio-de-redes-sociales-2021-tiktok-y-twitch-al-asalto-del-trono-de-las-redes-clasicas-facebook-en-declive/

Apéndice A

Manual de Instalación

Requerimientos:

- Python 3 (https://www.python.org/downloads/)
- Osintgram (https://github.com/Datalux/Osintgram)

Instalación

Es importante seguir la guía de instalación de *Osintgram* e instalar la aplicación en la ruta del usuario, ya que el programa desarrollado emplea esa ruta para la ejecución del programa.

Dicho esto, es necesario modificar levemente los archivos de *Osintgram* tanto para la configuración del usuario de Instagram empleado en el análisis como para la resolución de un error producido por el propio código.

Para la configuración del usuario (indicado como el paso 6 en la guía de instalación de *Osintgram*), hemos creado un perfil básico de Instagram con las siguientes credenciales:

Nombre de usuario: frangueuma

Contraseña: frangueuma1

Es importante señalar que este perfil básico de Instagram puede bloquearse por motivos de seguridad al realizar un gran número de peticiones a la API de Instagram (como veremos en la sección de "Problemas comunes") por lo que es recomendable crear un perfil propio de Instagram. Esto además permitiría obtener información de cuentas privadas si el perfil indicado sigue al objetivo.

En cuanto al error indicado, es necesario modificar el archivo *main.py* de la carpeta de *Osintgram* y cambiar en la línea 13 "*import from gnureadline*" por "*import readline as gnureadline*".

Problemas comunes

Ya que nuestra aplicación depende de otra, y a su vez de la API de Instagram, ocasionalmente podemos sufrir problemas con la cuenta de Instagram empleada. A continuación, veremos los más comunes y cómo actuar frente a ellos:

- **Challenge required**: cuando se realizan muchas peticiones a la API de Instagram esta se bloquea y emite como respuesta un desafío para poder continuar respondiendo peticiones. Esto se traduce en un mensaje de error en nuestra petición seguido de un enlace a seguir.

  Si seguimos este enlace se nos dirigirá a la página de inicio de Instagram, donde tendremos que iniciar sesión con la cuenta **indicada en los archivos de configuración de *Osintgram*** y seguidamente introducir un número de teléfono para autentificar que no somos una máquina.

  Una vez enviado el mensaje se podrá continuar ejecutando nuestra aplicación sin problema.
- **Logged out**: de forma ocasional el archivo *settings.json* de *Osintgram* enviará este fallo (https://github.com/Datalux/Osintgram/issues/243). Este error está controlado por la aplicación, que ejecuta de forma automática la corrección. Simplemente tendremos que volver a realizar la búsqueda y funcionará correctamente.

- **Geocatching timeout**: como hemos mencionado, para realizar geocoding empleamos la API de Google. Si el usuario objetivo dispone de muchas ubicaciones en sus datos, es posible que la API reciba muchas llamadas de forma continua y se bloquee mostrando por pantalla este error.

Apéndice B

Manual de Usuario

Búsqueda de información

Para comenzar la búsqueda de información de un objetivo simplemente tenemos que introducir su nombre de usuario en el formulario indicado a continuación en la Figura B.1 (evitando emplear '@') y pulsas el botón "*Search*".

Figura B.1 Formulario de búsqueda de usuario

Esto comenzará el proceso de búsqueda, abriendo las dos ventanas para la obtención de datos de Instagram y Twitter, descargando los ficheros necesarios y actualizando de forma automática el contenido de la página.

Una vez que estén disponibles los datos básicos del usuario se desbloquearan los botones de Instagram y Twitter permitiendo abrir el desplegable con la información, como vemos en la Figura B.2 y la Figura B.3. Cabe señalar que, en el caso de los datos

de Instagram, si la cuenta es pública se dispondrá en esta categoría la información de las cuentas relacionadas (en el caso de Twitter esta información está disponible siempre).

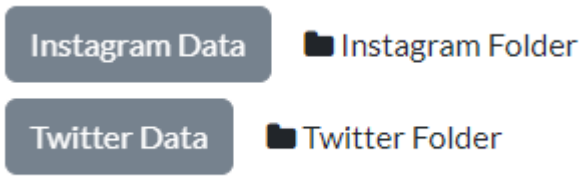

Figura B.2 Botones de información básica desactivados

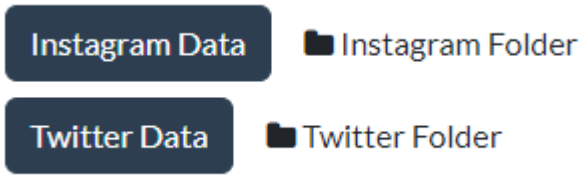

Figura B.3 Botones de información básica activados

Adicionalmente, en el caso de que el perfil de Instagram sea público, se dispondrá de la galería de fotos del usuario, mostrado en la Figura B.4. Es necesario señalar que el tiempo en el que se completa este proceso depende de la cantidad de fotos que el usuario tenga en su red social.

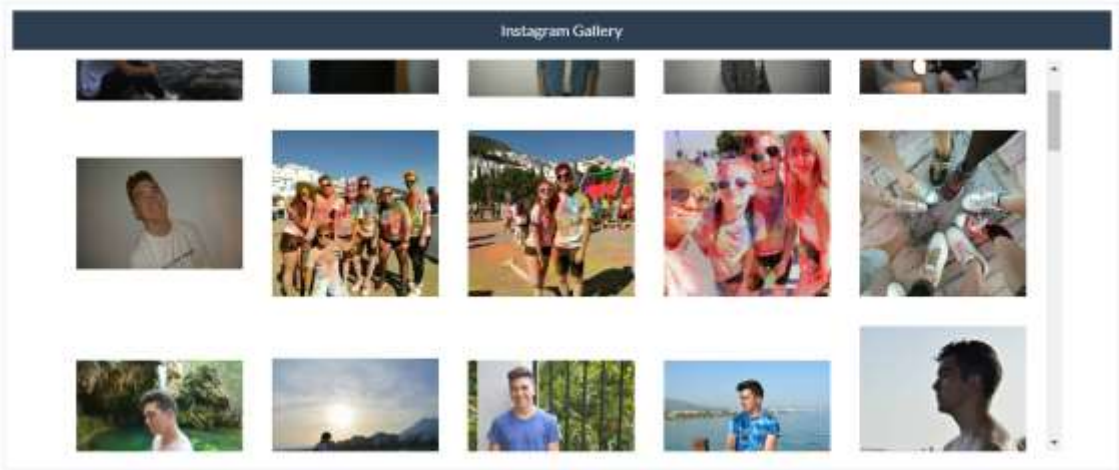

Figura B.4 Galería de Instagram

Finalmente, los datos de geolocalización podrán ser visualizados en el mapa de la parte inferior de la página, de la forma que vemos en la Figura B.5. Las distintas

localizaciones serán señaladas mediante *markers* con información sobre la localización, el lugar de visita y la fuente de datos desde donde se ha obtenido (Twitter o Instagram).

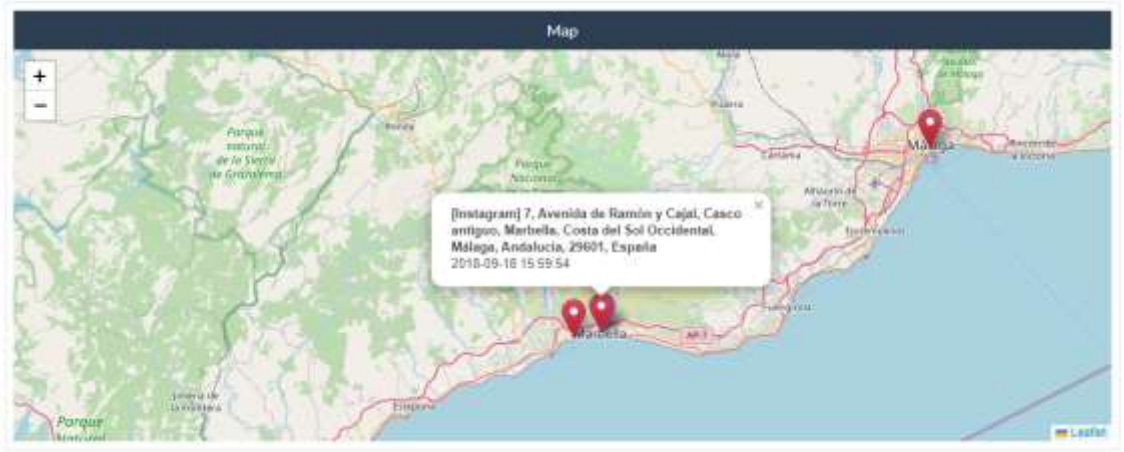

Figura B.5 Mapa con las localizaciones

Cuentas relacionadas

A la hora de obtener la lista de usuarios relacionados con el objetivo se puede indicar el número máximo de resultados que se desea mostrar por pantalla en el campo indicado por la Figura B.6. Por defecto este número es 5, pero si lo deseamos podemos modificar el valor "*Max number of related accounts*".

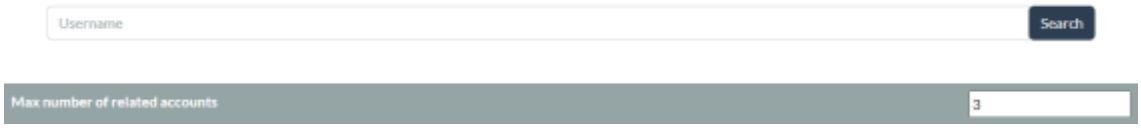

Figura B.6 Configuración de cuentas relacionadas

Una vez modificado este valor solamente es necesario iniciar la búsqueda de información de nuevo.

Como particularidad, los usuarios relacionados con el objetivo se muestran en forma de enlace, lo que permite pulsar el usuario y que automáticamente se escriba en el formulario de búsqueda de usuario. Podemos visualizar estos usuarios de la forma en la que se presentan en las Figuras B.7 (Instagram) y B.8 (Twitter). Esto facilita la

dinámica de búsquedas consecutivas de usuario, siendo necesario únicamente pulsar el botón para comenzar la búsqueda.

Figura B.7 Cuentas relacionadas Instagram

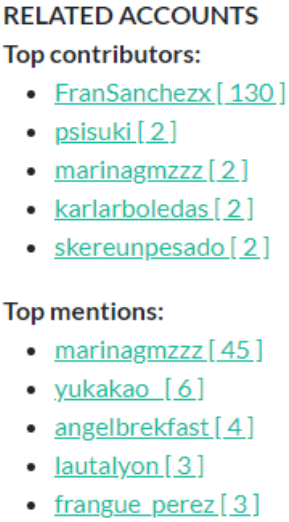

Figura B.8 Cuentas relacionadas Twitter

Acceso a ficheros de datos descargados

Si se desea, se pueden acceder a los directorios donde se descargan los datos obtenidos pulsando los botones *folder* localizados juntos a los botones de información básica. Esto mostrará por pantalla el directorio de la forma que se ve en la Figura B.9.

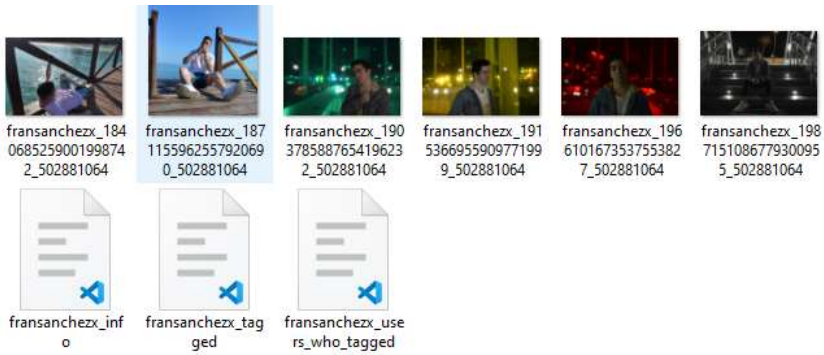

Figura B.9 Fichero con los archivos descargados del objetivo

Es necesario recordar que los archivos descargados se eliminan cuando se reinicia la aplicación. Sin embargo, al realizar búsquedas consecutivas se mantienen todos los archivos obtenidos.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga