

Plataforma para el análisis de datos de Instagram

Trabajo de Fin de Grado

Grado en Ingeniería Informática



**VNiVERSIDAD
D SALAMANCA**

Septiembre de 2022

Autor

Alberto Macías Gutiérrez

Tutores

Lucía Martín Gómez

Gabriel Villarrubia González

Juan Francisco de Paz Santana

Firma de tutores

D^a. Lucía Martín Gómez, profesora de la Universidad Pontificia de Salamanca, D. Gabriel Villarrubia González y D. Juan Francisco de Paz Santana, profesores del Departamento de Informática y Automática de la Universidad de Salamanca.

Hacen constar:

Que el trabajo titulado "Plataforma para el análisis de datos de Instagram" ha sido realizado por D. Alberto Macías Gutiérrez, con DNI 28981048J y constituye la memoria del trabajo realizado para la superación de la asignatura Trabajo de Fin de Grado de la Titulación Grado en Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a miércoles, 7 de septiembre de 2022

Resumen

En la actualidad, gran parte de la población mundial utiliza o utilizará las redes sociales. En concreto la red social Instagram contiene gran cantidad de datos tanto de usuarios como de publicaciones. Este gran volumen de datos puede resultar difícil de asimilar o comprender para las personas ya que se trata de información muy heterogénea. La característica principal del sistema propuesto en este proyecto es el desarrollo de una plataforma, en este caso una aplicación web, que permita recopilar, analizar y mostrar ciertos datos de la red social objetivo, en este caso Instagram. Permite recopilar datos de una cuenta, historias destacadas de una cuenta o una publicación de Instagram mediante el uso de técnicas de *web scraping*. La plataforma ofrece una serie de interfaces para permitir a un usuario registrar una cuenta, iniciar sesión o eliminar una cuenta que tenga registrada. Ofrece una serie de interfaces con un buscador para permitir al usuario introducir la cuenta de Instagram a buscar o el identificador de la publicación a mostrar. Además, la plataforma visualiza los datos recopilados y analizados de la cuenta, las historias destacadas de la cuenta o la publicación de Instagram que ha sido buscada, con ayuda de tablas y gráficos para dirigir la atención del usuario hacia los datos más relevantes. También, ofrece una serie de interfaces para añadir o eliminar cuentas de Instagram y realizar la búsqueda a través de estas para conseguir datos de cuentas o publicaciones que son privadas para la cuenta que utiliza la aplicación por defecto. El proyecto ha sido desarrollado mediante la utilización de la metodología del Proceso Unificado.

Palabras clave: Red social, recopilar datos, Instagram, web scraping, cuentas, historias destacadas, publicaciones

Abstract

Nowadays, a large part of the world's population uses or will use social networks. In particular, the social network Instagram contains a large amount of data on both users and posts. This large volume of data can be difficult for people to assimilate or understand as it is very heterogeneous information. The main feature of the system proposed in this project is the development of a platform, in this case a web application, that allows to collect, analyze and display certain data from the target social network, in this case Instagram. It allows to collect data from an account, featured stories from an account or an Instagram post by using *web scraping* techniques. The platform offers a series of interfaces to allow a user to register an account, log in or delete an account they have registered. It offers a series of interfaces with a search engine to allow the user to enter the Instagram account to search for or the identifier of the post to display. In addition, the platform displays the data collected and analyzed from the account, the featured stories of the account or the Instagram post that has been searched, with the help of charts and graphs to direct the user's attention to the most relevant data. It also offers a series of interfaces to add or remove Instagram accounts and search through them to get data on accounts or posts that are private to the account using the default application. The project has been developed using the Unified Process methodology.

Keywords: Social network, collect data, Instagram, *web scraping*, accounts, featured stories, posts.

Índice

1. Introducción.....	1
2. Objetivos	4
3. Antecedentes	5
4. Conceptos teóricos.....	10
4.1. Protocolo <i>HTTP</i>	10
4.2. Arquitectura Cliente-Servidor	11
4.3. Desarrollo Web	13
4.3.1. Desarrollo <i>Frontend</i>	13
4.3.2. Desarrollo <i>Backend</i>	13
4.3.3. Evolución.....	14
4.4. Red social	15
4.5. <i>Web Scraping</i>	16
4.6. <i>GraphQL</i>	17
4.7. <i>Business Intelligence</i>	19
4.8. Tecnologías y herramientas para el desarrollo web	20
4.8.1. <i>Python</i>	20
4.8.2. <i>Django</i>	21
4.8.3. <i>HTML</i>	24
4.8.4. <i>CSS</i>	24
4.8.5. <i>JavaScript</i>	24

4.8.6. <i>Bootstrap</i>	25
4.8.7. Base de datos.....	26
4.8.8. <i>Visual Studio Code</i>	27
4.9. Tecnologías y herramientas de control de versiones.....	29
4.9.1. <i>Git</i>	29
4.9.2. <i>GitHub</i>	29
4.10. Tecnologías y herramientas de documentación del proyecto.....	30
4.10.1. <i>Visual Paradigm</i>	30
4.10.2. <i>Microsoft Office Project</i>	30
4.10.3. <i>EZEstimate</i>	30
4.10.4. <i>Sphinx</i>	32
5. Desarrollo de la propuesta.....	33
5.1. Definición de la propuesta.....	33
5.1.1. Planificación del proyecto.....	35
5.1.2. Especificación de requisitos software y análisis.....	35
5.1.3. Especificación de diseño.....	36
5.2. Diseño de la base de datos.....	37
5.2.1. Almacenamiento de usuarios.....	37
5.2.2. Almacenamiento de cuentas de Instagram buscadas.....	42
5.2.3. Almacenamiento de publicaciones de Instagram buscadas.....	54
5.3. Proceso de obtención de datos de Instagram.....	64

5.3.1. Cuenta de Instagram para recopilar datos	65
5.3.2. Obtención de datos de cuentas de Instagram.....	67
5.3.3. Obtención de datos de publicaciones de Instagram.....	69
5.3.4. Desarrollo de la aplicación web	71
6. Resultados	88
7. Conclusiones y líneas de trabajo futuras	90
8. Bibliografía	93
9. Glosario.....	97

Índice de figuras

Figura 1. Estudio de las redes sociales 2022.....	1
Figura 2. Historia de las redes sociales	7
Figura 3. Protocolo HTTP.....	10
Figura 4. Modelo Cliente-Servidor	11
Figura 5. Historia del Desarrollo Web	14
Figura 6. Funcionamiento Framework Django	23
Figura 7. Puntos de casos de uso	31
Figura 8. Interfaz EZEstimate	31
Figura 9. Ciclo de vida de un proyecto	34
Figura 10. Diagrama de clases del usuario.....	37
Figura 11. Clase Contacto	38
Figura 12. Clase cuenta scraping Instagram	39
Figura 13. Diagrama de cuenta de Instagram buscada	42
Figura 14. Diagrama de publicación de Instagram buscada	54
Figura 15. Formulario de contacto	73
Figura 16. Formulario del buscador	74
Figura 17. Lista de cuentas buscadas	74
Figura 18. Información de la cuenta buscada	75
Figura 19. Gráficos cuenta buscada	76
Figura 20. Tipos de tablas cuenta buscada	77
Figura 21. Información de las historias destacadas de una cuenta buscada	78
Figura 22. Publicación buscada.....	79
Figura 23. Información de la publicación buscada	81
Figura 24. Información cuentas scraping Instagram	82

Figura 25. Modal añadir cuenta scraping Instagram	83
Figura 26. Modal usar cuenta scraping Instagram	84
Figura 27. Modal eliminar cuenta scraping Instagram.....	85
Figura 28. Iniciar sesión	85
Figura 29. Registrar usuario	86
Figura 30. Modal eliminar usuario.....	87

Índice de tablas

Tabla 1. Información de atributos de la tupla 'Contacto'	39
Tabla 2. Información de atributos de la tupla 'CuentaScrapeoInstagram'	40
Tabla 3. Información de atributos de la tupla 'Usuario'	41
Tabla 4. Información de atributos de la tupla 'DatosBusquedaUsuario'	45
Tabla 5. Información de atributos de la tupla 'IdentificadorUsuario'	46
Tabla 6. Información de atributos de la tupla 'ContadorHighlights'	47
Tabla 7. Información de atributos de la tupla 'DatosHighlight'	48
Tabla 8. Información de atributos de la tupla 'DatosStoryHighlight'	49
Tabla 9. Información de atributos de la tupla 'DatosPostBusquedaUsuario'	51
Tabla 10. Información de atributos de la tupla 'DatosVideosBusquedaUsuario'	52
Tabla 11. Información de atributos de la tupla 'DatosEtiquetadasBusquedaUsuario'	53
Tabla 12. Información de atributos de la tupla 'DatosPost'	57
Tabla 13. Información de atributos de la tupla 'ComentarioMaxLikesPost'	58
Tabla 14. Información de atributos de la tupla 'UsuarioMaxComentariosPost'	59
Tabla 15. Información de atributos de la tupla 'PostSidecar'	60
Tabla 16. Información de atributos de la tupla 'HashtagsSubtituloPost'	60
Tabla 17. Información de atributos de la tupla 'MencionesSubtituloPost'	61
Tabla 18. Información de atributos de la tupla 'PatrocinadoresPost'	62
Tabla 19. Información de atributos de la tupla 'UsuariosEtiquetadosPost'	63
Tabla 20. Funciones de instaloader para iniciar y guardar sesión	65
Tabla 21. Funciones de instaloader para cargar una sesión guardada en el sistema	66
Tabla 22. Funciones de instaloader para obtener lista de cuentas	67
Tabla 23. Funciones de instaloader para obtener datos de la cuenta buscada	68
Tabla 24. Funciones de instaloader para obtener datos de historias destacadas	69

Tabla 25. Funciones de instaloader para obtener datos de la publicación buscada.....70

1. Introducción

En las últimas décadas, el nacimiento de diferentes redes sociales ha supuesto una revolución en la comunicación entre las personas y la difusión de información. Cada vez son más las personas que utilizan las redes sociales, gracias en parte a la popularización de los smartphones con conexión a internet (Portaltic, 2022). Concretamente en España un 93% de la población comprendida entre los 12 y 70 años usan los servicios de internet, de los cuales un 85% son usuarios de redes sociales. Además, los usuarios suelen usar al mes un promedio de 4,5 redes sociales (ELOGIA, 2022).

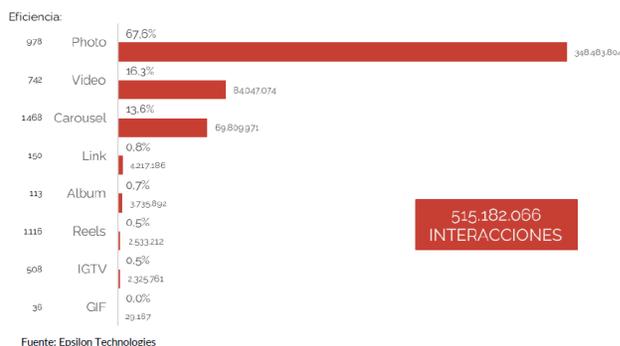
Los usuarios que forman parte de las redes sociales y las usan, postean gran cantidad de datos, pero estos no son fáciles de recuperar y analizar manualmente, por lo que los programas para el análisis de la información de las redes sociales se han convertido en una herramienta esencial hoy en día para saber las tendencias o preferencias que siguen los distintos grupos de usuarios.

Un problema se encuentra en la recopilación de datos, aunque algunas redes sociales proporcionen APIs, muchas otras no ofrecen este tipo de facilidades y medios o en otros casos su coste es demasiado elevado (Lucía Martín Gómez et al., 2021).

Contenido | Contenido consumido (III)



- El contenido más consumido por los usuarios son el formato foto, seguido de vídeo.



Estudio Anual Redes Sociales 2022

515.182.066 INTERACCIONES

PATROCINADO POR: Los datos se extraen de un panel formado por 910 marcas relevantes de España en RRSS. Se miden las reacciones/interacciones realizadas por los usuarios frente a los contenidos publicados por las marcas en sus canales sociales corporativos: (Fb, Tw, Yt, Ig). En total 3000 perfiles sociales analizados. ELABORADO POR:

Figura 1. Estudio de las redes sociales 2022

Introducción

Como podemos ver en la Figura 1 (ELOGIA, 2022), el contenido más consumido por los usuarios es en formato de foto, llevándose un 67,6% de las interacciones, seguido bastante por detrás de vídeo. Este hecho justifica a Instagram como la red social que mayor interacción tiene, gracias a que se trata de una red social de fotos, así como una de las más usadas diariamente, en concreto la segunda solamente detrás de WhatsApp, además de ser una de las favoritas, para el público comprendido entre los 16 y 24 años (Galeano, 2022).

Este documento tratará de analizar los datos de la red social Instagram, para ello, se ha planteado una propuesta, que recopila datos mediante *web scraping* de los usuarios y publicaciones a modo de buscador. Además, también se recopilarán datos de las historias destacadas de los distintos usuarios. Todo esto se va a resolver mediante el desarrollo de una aplicación web multiusuario, que administre al usuario información, gráficos y tablas, contruidos gracias a los datos recogidos anteriormente, de forma interactiva, intuitiva y sencilla gracias al uso de interfaces gráficas visuales, usables y educativas para el usuario (Eniun Diseño Web y Marketing Digital, 2019).

Este documento está organizado en los siguientes apartados o secciones:

- **Introducción:** Sección actual, que busca identificar y justificar el problema a tratar.
- **Objetivos:** Busca redactar los diferentes objetivos o metas que se pretenden lograr con este proyecto.
- **Antecedentes:** Pequeño resumen de las tecnologías que se han ido usando para las redes sociales.
- **Conceptos teóricos:** Se detallan las distintas técnicas y herramientas que se usan para el desarrollo del proyecto.
- **Desarrollo de la propuesta:** Se detallan los aspectos más importantes del desarrollo del proyecto.
- **Resultados:** Se muestra la información obtenida del trabajo.

Introducción

- **Conclusiones y líneas de trabajo futuras:** Se interpretan los resultados que se han extraído de la realización del trabajo y el planteamiento a abordar si se desea continuar con el proyecto.

Este proyecto, también está formado por otro documento compuesto por los siguientes anexos:

- **Plan del proyecto Software:** Se detalla la planificación temporal del proyecto.
- **Especificación de requisitos y análisis:** Se detallan la especificación de requisitos del proyecto y el análisis de este.
- **Especificación de diseño:** Se detalla la documentación del diseño del proyecto.
- **Documentación técnica de programación:** Se detalla la documentación para la comprensión por parte del programador.
- **Manual del usuario:** Se detalla la documentación para dar asistencia a los usuarios que pretendan hacer uso del proyecto.

2. Objetivos

El Trabajo de Fin de Grado tiene como objetivo principal ofrecer a los usuarios una herramienta que permita recopilar, tratar y plasmar información de perfiles o publicaciones de Instagram, con el fin de representar y mostrar estos datos para poder guiar y dar a entender a los usuarios lo más importante y relevante de estos.

De este objetivo principal derivan los siguientes objetivos específicos:

- **Obtener datos de Instagram a través de *web scraping*:** Los datos a utilizar por la herramienta se van a obtener mediante técnicas de *web scraping*.
- **Crear tablas y gráficos con los datos obtenidos:** Los datos a obtener, se van a presentar en forma de tablas y gráficos con el fin de poder transformar esos datos en información más entendible para los usuarios.
- **Crear interfaces para usar la aplicación:** Para que el usuario pueda hacer uso de toda la funcionalidad de la herramienta, se va a crear una aplicación web con varias interfaces escritas en el lenguaje de marcado *HTML*, con el fin de que se pueda usar la aplicación desde cualquier navegador.
- **Almacenar información:** Para almacenar información de las cuentas de usuario, así como la información sobre las búsquedas, se va a hacer dentro de una base de datos relacional *SQLite3*.
- **Crear un modelo de usuario:** Para que cualquier cliente se pueda registrar como usuario, modificar la cuenta de Instagram de *scraping* para realizar las búsquedas y poder borrar la cuenta de usuario.
- **Guardar datos importantes encriptados:** Con el fin de crear una aplicación segura, las contraseñas de los usuarios se van a guardar encriptadas dentro de la base de datos.

3. Antecedentes

Para poder meternos en contexto debemos recordar el nacimiento de Internet (Bahillo, 2022), en 1969 durante la guerra fría, Estados Unidos crea *ARPANET* una red exclusivamente militar, con el objetivo de mantener las comunicaciones en caso de guerra ante la situación de incertidumbre y temor del momento. Un año más tarde, Ray Tomlinson establece las bases para la creación del correo electrónico.

En 1983, el departamento de defensa de Estados Unidos decidió usar el protocolo *TCP/IP* en su red *ARPANET* creando la red *Arpa Internet*, que con el paso de los años acabó nombrándose *Internet*. En 1989, Tim Berners Lee describió por primera vez el protocolo de transferencias de hipertextos que daría lugar a la primera web utilizando los recursos *HTML*, *HTTP* y un programa llamado *Web Browser*. La *World Wide Web* (*www*), creció exponencialmente pasando de tener 100 *World Wide Web Sites* en 1993 a más de 200000 en 1997. Actualmente internet sigue creciendo.

Una vez explicada la historia de internet veremos la historia de las redes sociales (Hera, 2022), en el año 1997 aparece SixDegrees, la que fue considerada como la primera red social del mundo, una red que permitía localizar a otros miembros de la red uniendo conocidos con conocidos de conocidos y crear listas de amigos.

En 2002 se crea Friendster, una red social para los amantes de los videojuegos y en 2003 se crea LinkedIn, una red social mucho más profesional orientada a empresas, que hoy en día cuenta con más de 600 millones de registrados.

En 2004 aparece la red social por excelencia, Facebook, creada por el joven estudiante Mark Zuckerberg. En un principio era un portal llamando Facemash, con la finalidad de que los estudiantes de Harvard pudieran compartir opiniones acerca de quienes eran las personas más

Antecedentes

atractivas de la Universidad. Hoy en día es la mayor red social con más de 2500 millones de usuarios activos al mes.

En 2005 aparece YouTube, una red social creada por Chad Hurley, Steve Chen y Jawn Karim en California. Esta red social se trata de un sitio web dedicado a compartir vídeos. Hoy en día dispone cerca de 2000 millones de usuarios activos al mes.

En 2006 aparece Twitter, una red social creada por Jack Dorsey, Noah Glass, Biz Stone y Evan Williams. Esta red social permite la comunicación con otros usuarios mediante mensajes con un tamaño máximo de 140 caracteres. Hoy en día dispone cerca de 340 millones de usuarios activos al mes.

En 2009 aparece WhatsApp, la app de mensajería instantánea más famosa, creada por el ucraniano Jan Koum y posteriormente comprada por Mark Zuckerberg. Esta se vincula a la agenda de contactos del dispositivo, permitiendo al usuario ver qué estaba haciendo cada persona en cada momento, con el fin de saber si poder iniciar o no una conversación con dicha persona. Hoy en día supera los 2000 millones de usuarios activos al mes.

En 2010 aparece Instagram, la red social de fotografía por excelencia, creada por Kevin Systrom y Mike Krieger. Además, fue pionera junto con Twitter en el uso de *hashtags*, facilitando a los usuarios descubrir fotografías que otros usuarios compartían sobre un mismo tema. Hoy en día supera los 1000 millones de usuarios activos al mes.

También ese mismo año aparece Pinterest, una red social que colecciona imágenes, que permite a los usuarios almacenarlas en tableros y dotarlas de pines. Hoy en día supera los 300 millones de usuarios activos al mes.

En 2011 aparece Twitch, la plataforma de streaming más grande del mundo, creada por Justin Kan, Emmett Shear, Michael Seibel y Kyle Vogt y posteriormente comprada por Amazon.

Antecedentes

En 2016 aparece Tik Tok también conocido como Douyin en China, es una red social que permite compartir videos cortos, desde 15 segundos de duración hasta un máximo de un minuto.

En la Figura 2 podemos observar una línea temporal del nacimiento de las distintas redes sociales.

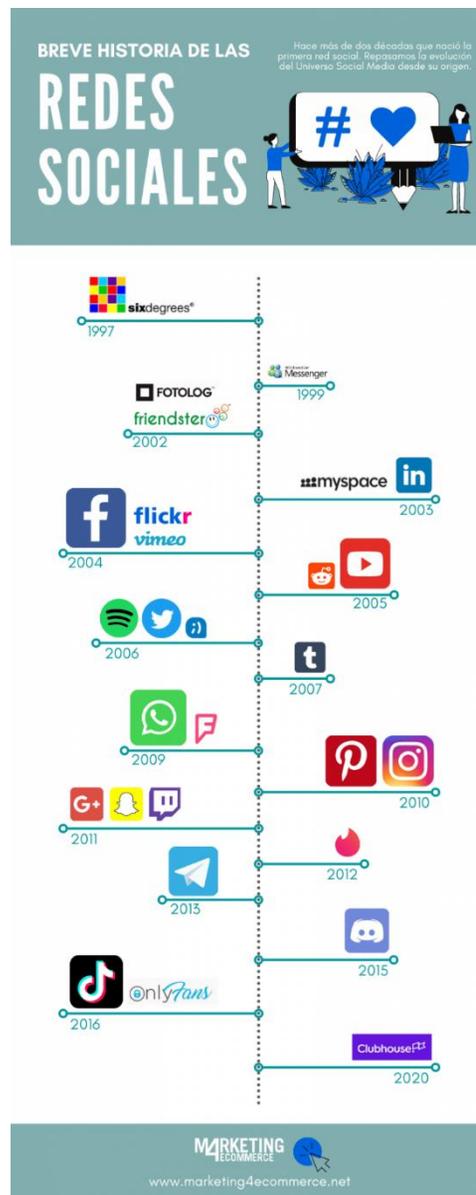


Figura 2. Historia de las redes sociales

Antecedentes

Una vez conocemos la historia y para que esta concebido el uso de ciertas redes sociales, vamos a ver herramientas similares a la que se pretende realizar en este proyecto.

Una de las herramientas más famosas, y que también realiza análisis de datos sobre las redes sociales, pero en este caso sobre Twitter, es la *API* de Twitter (Twitter, 2022). Esta permite recuperar información pública de Twitter, como solicitar información sobre una determinada cuenta o usuario, acceder a *tweets* buscando palabras clave o una cierta cuenta, o brindar acceso a conversaciones por mensajes directos de los usuarios que otorgan permisos de forma explícita (Pío, 2022). Esta *API* permite gran cantidad de cosas, pero sin embargo tiene un problema, que no permite recuperar información de cuentas o *tweets* privados, ya que sería totalmente ilegal.

Otra herramienta para realizar análisis de datos sobre las redes sociales, también para Twitter, es SocioViz (SocioViz, 2014). Esta herramienta además de obtener datos de Twitter permite visualizarlos. La información obtenida puede ser de cualquier palabra clave, *hashtag*, *emoji* o mención de usuario filtrando por fecha e idioma, *tweets* en tiempo real o durante un periodo concreto, además esta información es representada mediante gráficos y tablas para facilitar su comprensión. El funcionamiento de esta herramienta se basa consultar y recopilar resultados de la *API* de Twitter (Zonín, 2019), por lo que solo permite recuperar información pública.

Una vez hemos visto algunas herramientas similares, vamos a centrar nuestra atención en Instagram. Esta red social (Lucía Martín Gómez et al., 2021) ofrece una *API* de visualización básica para obtener información básica de una cuenta para la que se dispone de credenciales, sin embargo, la *API* tiene varias limitaciones, ya que no ofrece información sobre las historias o comentarios del usuario, las publicaciones que contienen medios tampoco están disponibles y la información del usuario solo se puede obtener si las credenciales del usuario están disponibles.

Instagram también ofrece una segunda *API* para cuentas profesionales, con la que se puede obtener y publicar contenido multimedia, administrar y responder comentarios, identificar

Antecedentes

contenido multimedia con menciones, buscar contenido multimedia usando *hashtags* y obtener resultados y metadatos básicos sobre otras empresas y creadores de esta red social. La funcionalidad de esta segunda *API* es mucho más completa, pero solo es posible de aplicar al usuario profesional desde el que se accede, y el usuario también debe de tener una cuenta profesional. Esto limita mucho la recopilación de datos.

Por este motivo, en este proyecto se van a recopilar los datos mediante *web scraping*, ya que además de recopilar información sobre usuarios públicos, se va a poder recopilar información sobre usuarios privados, debido a que se van a utilizar cuentas de Instagram en concreto para realizar esta búsqueda y si la cuenta con la que se realiza la búsqueda sigue a la que se está inspeccionando, se podrá recopilar información de dicha cuenta.

4. Conceptos teóricos

4.1. Protocolo *HTTP*

HTTP (*Hypertext Transfer Protocol*), es el protocolo que permite realizar peticiones de datos y recursos, tales como documentos *HTML* (MDN contributors, 2020). Es la base de cualquier intercambio de datos de la web. Diseñado a principios de la década de 1990, *HTTP* es un protocolo ampliable, que ha ido evolucionando con el tiempo. En la Figura 3 se puede observar que *HTTP* es lo que se conoce como un protocolo de la capa de aplicación, que se transmite sobre el protocolo TCP, o el protocolo encriptado TLS, aunque se podría usar cualquier otro protocolo fiable. Al tratarse de un protocolo ampliable, además de transmitir documentos de hipertexto (*HTML*) permite transmitir imágenes, vídeos o enviar datos o contenido a los servidores a través de formularios.

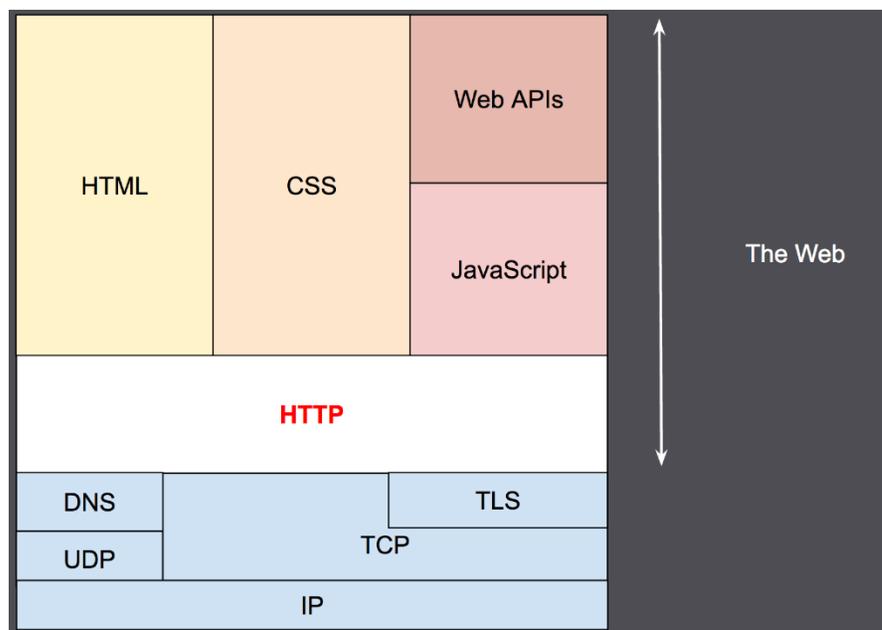


Figura 3. Protocolo *HTTP*

4.2. Arquitectura Cliente-Servidor

Gran parte de las aplicaciones web usan la arquitectura Cliente Servidor (Schiaffarino, 2019), que es una de las principales usadas en muchos servicios y protocolos de internet. Esta arquitectura tiene dos partes bien diferenciadas, por un lado, la parte del servidor, que suele ser una máquina bastante potente que actúa como depósito de datos y ejecuta la aplicación web, y por otro la parte del cliente o clientes, que son dispositivos o máquinas que demandan servicios al servidor.

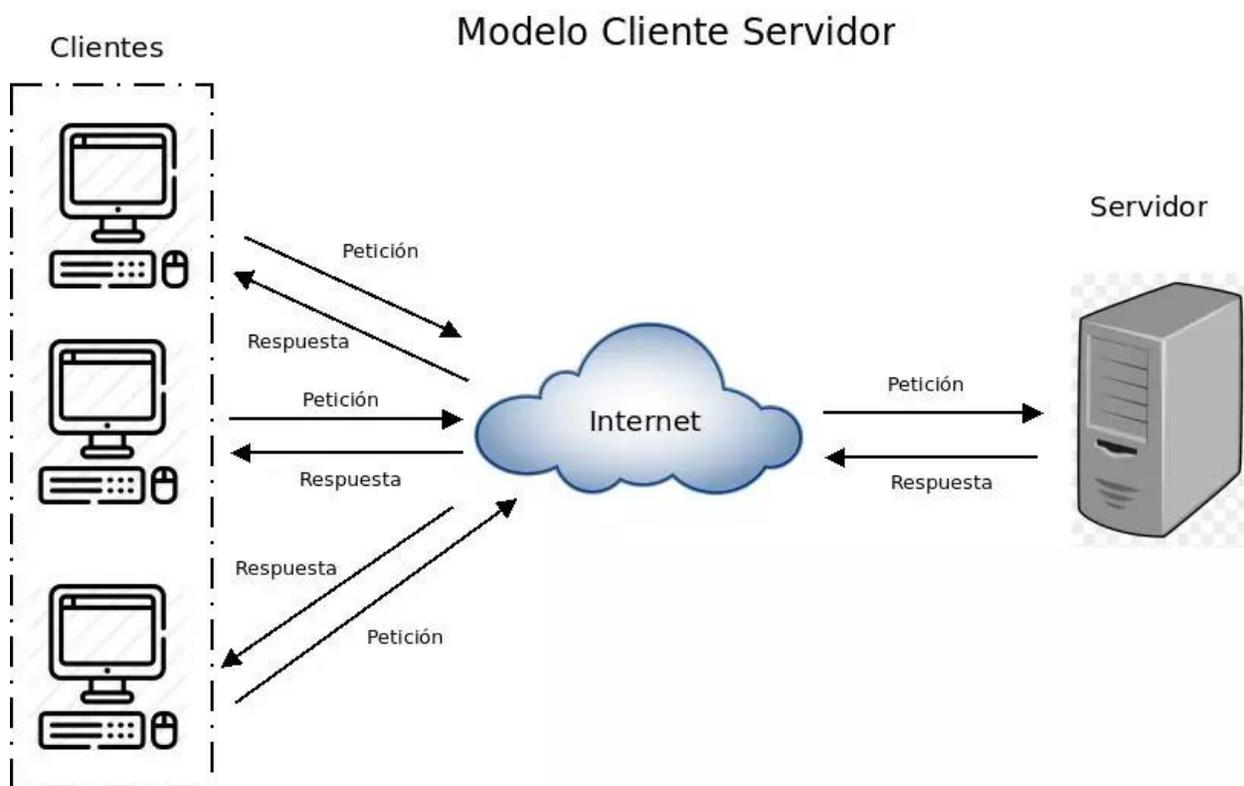


Figura 4. Modelo Cliente-Servidor

En la Figura 4 se observa cómo funciona el Modelo Cliente Servidor dentro de internet, por el que los clientes o usuarios envían peticiones y el servidor responde a esas peticiones.

Un ejemplo de la arquitectura Cliente Servidor es la red de internet donde existen ordenadores de diferentes personas conectados alrededor del mundo, los cuales se conectan a través de los

Conceptos teóricos → Arquitectura Cliente-Servidor

servidores de su proveedor de internet por *ISP* donde son redirigidos a los servidores de las páginas que desean visualizar.

Ventajas:

- Facilita la integración entre diferentes sistemas y comparte información permitiendo que las máquinas ya existentes puedan ser utilizadas mediante una interfaz más amigable para el usuario.
- Al tratarse de un sistema que favorece el uso de las interfaces gráficas interactivas, tiene una mayor interacción con el usuario.
- La estructura modular facilita la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional favoreciendo así la estabilidad de las soluciones.
- Este modelo permite proporcionar a las diferentes áreas de una empresa generar un orden de trabajo en donde cada sector puede trabajar en su área, pero accediendo al mismo servidor e información que los demás sin generar conflictos.

Desventajas:

- Requiere habilidad para que un servidor sea reparado.
- Problema de seguridad, el hecho de compartir canales de información entre servidores y clientes requieren que estos pasen por procesos de validación, es decir, protocolos de seguridad que pueden tener algún tipo de puerta abierta permitiendo que se generen daños físicos, amenazas o ataques de *malware*.
- Este modelo representa una importante limitación en cuanto a los costos económicos debido a que los servidores son computadoras de alto nivel con un *hardware* y *software* específicos, por lo que reemplazar componentes que estén averiados tiene un costo elevado.

4.3. Desarrollo Web

El desarrollo web es un término (Mercedes, 2017) que define la creación o mantenimiento de sitios web para Internet o una intranet. Para conseguirlo se hace uso de tecnologías de software del lado del servidor y del cliente que involucran una combinación de procesos de base de datos con el uso de un navegador web con el fin de realizar unas determinadas tareas o mostrar cierta información.

Un desarrollador web (Seguro, 2021) se encarga de las siguientes tareas:

- Creación de sitios web y aplicaciones.
- Mantenimiento de estos sistemas, para que sean eficientes y ágiles.
- Asegurar que el sistema desarrollado cumpla con las demandas de los usuarios.
- Proporcionar seguridad al entorno de la aplicación web.

4.3.1. Desarrollo *Frontend*

El desarrollo *frontend* es el desarrollo que se encarga de los aspectos funcionales y de la usabilidad. En este ámbito se incluye la *UX* (experiencia de usuario) y la *UI* (relativo a interfaces), por lo que a este desarrollo se lo conoce como desarrollo por parte del cliente. Utiliza *frameworks* de *JavaScript*, además del lenguaje de marcado *HTML*.

4.3.2. Desarrollo *Backend*

El desarrollo *backend* es el desarrollo que se enfoca en las estructuras en las que se apoyan los sitios web y aplicaciones, es decir, de aquello que los usuarios no ven. En este se establecen las conexiones necesarias con la base de datos y el servidor, por lo que se lo conoce como desarrollo del lado del servidor. Utiliza lenguajes de programación como *JavaScript*, *Python*, *SQL*, *PHP* o *Java*, entre otros.

4.3.3. Evolución

El desarrollo web (Gima, 2019) ha evolucionado exponencialmente desde su inicio. En la actualidad es uno de los servicios más solicitados, que nos brinda una gran cantidad de funciones distintas.

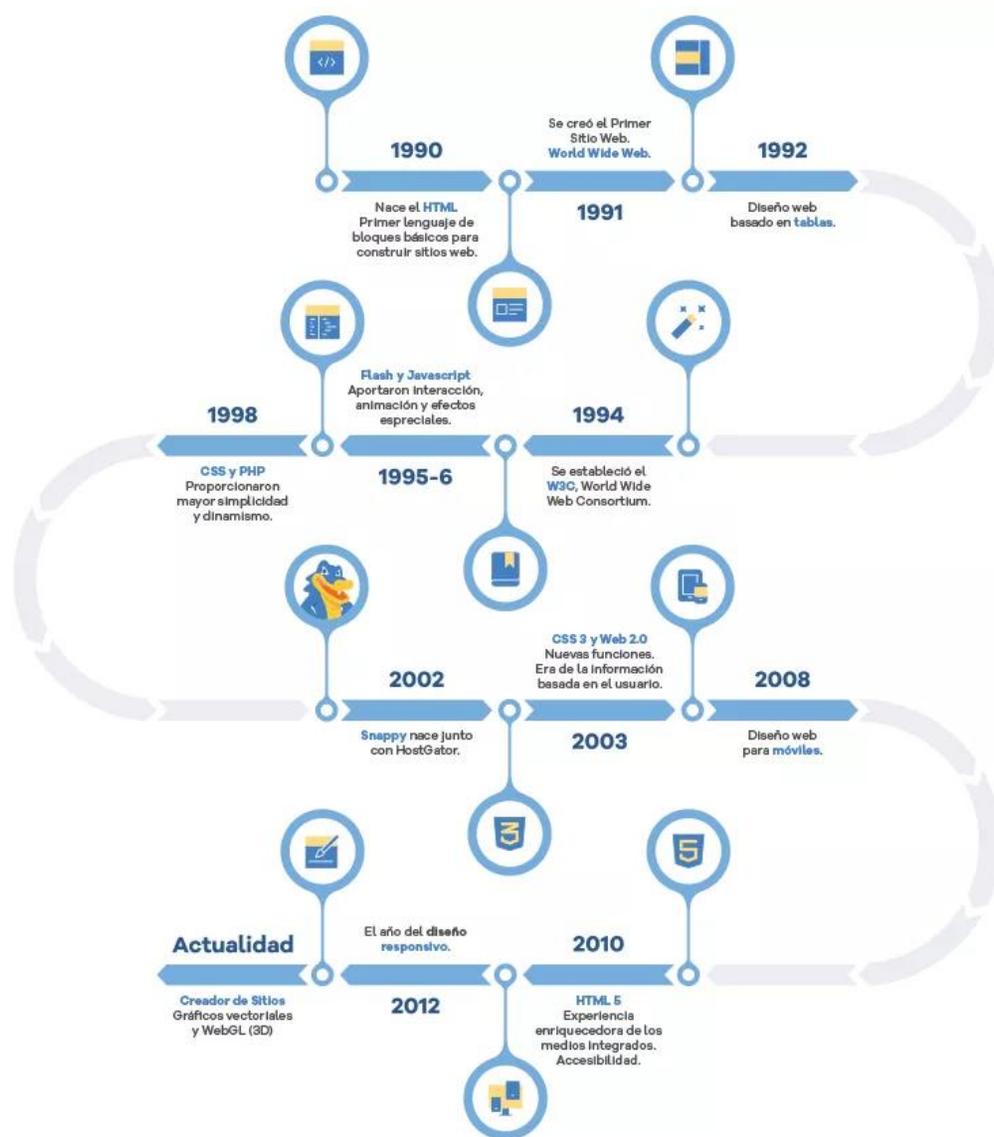


Figura 5. Historia del Desarrollo Web

En la Figura 5 podemos observar, como hemos pasado de páginas *HTML* estáticas en los años 1990 a la creación de páginas web dinámicas, gracias al uso de *JavaScript* para dar dinamismo

Conceptos teóricos → Red social

a las páginas web, las hojas de estilo CSS para dar formato, apariencia y hacer un diseño responsivo o adaptable con el fin de que estas páginas web se puedan representar en pantallas con tamaños y resoluciones muy diferentes.

4.4. Red social

Las redes sociales (Llonch, 2021) son plataformas digitales que conectan entre sí a personas con intereses, actividades o relaciones en común. Estas plataformas permiten el contacto entre los individuos que las componen y funcionan como un medio para intercambiar información. La información publicada por un usuario puede ser pública o privada.

Tipos de redes sociales:

- Redes sociales personales: Las redes sociales personales están pensadas para conectar individuos entre sí basándose en conexiones personales. Un ejemplo es Instagram.
- Redes sociales de entretenimiento: La función principal de estas redes sociales es compartir contenidos. Un ejemplo es YouTube.
- Redes sociales profesionales: Estas están pensadas para hacer contactos, publicar información sobre el currículum y logros, conseguir empleo y otros fines relacionados con el mundo profesional. Un ejemplo es LinkedIn.
- Redes sociales de nicho: Dirigidas a personas con un interés específico común, ya sea profesional o personal. Un ejemplo es DevianArt.

Esta clasificación se basa en la finalidad original de cada red, en realidad muchas redes sociales ofrecen flexibilidad en cuanto a sus usos. Instagram puede considerarse una red personal, de entretenimiento o incluso profesional.

4.5. Web Scraping

Se llama *web scraping* (Zhao, 2017) a la técnica de extraer datos de una página web y guardarlos en un sistema de archivos o base de datos para su posterior recuperación o análisis. Normalmente, los datos de la web se extraen utilizando el protocolo *HTTP* o a través de un navegador web. Esto se lleva a cabo de forma manual por un usuario, automáticamente por un bot o rastreador web. Debido a la gran cantidad de datos heterogéneos que se generan constantemente en internet, el *web scraping* está ampliamente reconocido como una técnica eficaz y poderosa para recopilar big data.

Para adaptarse a gran variedad de escenarios, las técnicas actuales de *web scraping* se han personalizado a partir de procedimientos más pequeños *ad hoc*, asistidos por humanos, hasta la utilización de sistemas totalmente automatizados que son capaces de convertir sitios web enteros en un conjunto de datos. Las herramientas de *web scraping* más modernas son capaces de analizar lenguajes de marcado archivos *JSON* e integrar el análisis visual y el procesamiento del lenguaje natural para simular cómo los usuarios navegan por el contenido de la web.

El proceso de extracción de datos de Internet puede dividirse en dos pasos secuenciales: la adquisición de recursos web y, a continuación, extraer la información deseada de los datos adquiridos. En concreto, un programa comienza componiendo una petición *HTTP* para adquirir recursos de un sitio web, esta solicitud puede tener el formato de una *URL* que contenga una consulta *GET* o un trozo de mensaje *HTTP* que contenga una consulta *POST*. Una vez que la solicitud es recibida y procesada con éxito por el sitio web de destino, el recurso solicitado se recuperará del sitio web y se envía de nuevo al programa de *web scraping*. El recurso puede estar en múltiples formatos, como páginas web construidas a partir de *HTML*, fuentes de datos en formato *XML* o *JSON*, datos multimedia como imágenes, audio o archivos de vídeo.

4.6. GraphQL

GraphQL (Acosta, 2021) es un nuevo estándar de API que está diseñado para ser declarativo, flexible y eficiente. Es un lenguaje de consultas para comunicar clientes y servidores. Es de código abierto y fue desarrollado por Facebook, actualmente lo mantiene una gran comunidad de empresas e individuos de todo el mundo.

Fue inventado por Facebook ya que la empresa necesitaba resolver muchos problemas técnicos con su aplicación móvil nativa, de forma que GraphQL permite mejorar la comunicación entre las diferentes partes de una aplicación de software, por lo que hace que esa aplicación sea más fácil de entender, desarrollar, mantener y escalar. El objetivo principal de este es evitar las múltiples consultas al servidor

GraphQL proporciona los bloques de construcción para que los desarrolladores definan la estructura de sus datos, cómo acceder a ellos y qué datos devolver. Una API basada en GraphQL puede ser consumida por muchos clientes, no solo por los clientes de GraphQL. No está vinculado a una base de datos específica, así que no importa con qué motor de almacenamiento tenga que lidiar.

Es agnóstico de plataforma ya que se puede implementar en más de 20 lenguajes de programación, esto se debe a que GraphQL es una especificación, es decir, especifica cómo debe funcionar, permitiendo a cualquiera implementar GraphQL en cualquier lenguaje de programación. Es un lenguaje tipado, es decir, utiliza tipos para definir recursos, añade tipos a los campos de cada recurso y también utiliza tipos para comprobar estáticamente los errores. Es utilizado tanto por empresas grandes como pequeñas, desde Twitter y Facebook hasta Yelp y Twitch. En 2016, Github cambió su API de REST a GraphQL y en 2017, AWS lanzó una plataforma basada en GraphQL para crear aplicaciones.

Diferencias entre API REST y GraphQL (Ortega, 2019):

API REST

- Es una convención, una manera de comunicarse entre el servidor y el cliente, cada uno tiene sus reglas.
- El servidor expone recursos, los clientes se tienen que adecuar a como están expuestos.
- Hace *overfetching*, envía más información de la que se necesita.
- Múltiples solicitudes por vista, muy costoso en rendimiento.
- Documentación ajena al desarrollo, no hay estándar por lo que depende mucho del desarrollador para mantenerla.

GraphQL

- Lenguaje tipado y validable, se puede dar forma a lo que recibe y lo que devuelve.
- El cliente define que recibe, los clientes hacen una consulta, de la estructura que se define como respuesta.
- Envía lo necesario, se tiene control total de las respuestas que se esperan del servidor.
- Solo una solicitud por vista, prácticamente en una solicitud se puede mandar lo que se necesite.

4.7. *Business Intelligence*

También conocido como inteligencia de negocio o inteligencia empresarial, *business intelligence* (media, 2021) hace referencia a uso de estrategias y herramientas que sirven para transformar datos en conocimiento, con el objetivo de mejorar el proceso de toma de decisiones en una empresa. Este concepto ha ganado popularidad por la relevancia empresarial para cualquier empresa o negocio, y la complejidad en recopilar datos, procesarlos, analizarlos y presentarlos de forma que cualquiera los pueda entender.

4.8. Tecnologías y herramientas para el desarrollo web

4.8.1. Python

Fue creado por Guido van Rossum (Visus, 2020) a principios de los 90, su nombre se debe a la afición de este al grupo Monty Python. *Python* es un lenguaje de programación interpretado cuya filosofía es que sea legible por cualquier persona con conocimientos básicos de programación.

Características:

- Es gratis, ya que es un lenguaje de código abierto y no hay que pagar ninguna licencia para utilizarlo.
- Respaldo por una comunidad enorme, por el carácter gratuito hace que continuamente se estén desarrollando nuevas librerías y aplicaciones. Esto es un factor multiplicativo para los programadores, debido a que casi cualquier duda estará resuelta en los foros.
- Es un lenguaje multiparadigma, es decir, combina propiedades de diferentes paradigmas de programación, lo que permite que sea muy flexible y fácil de aprender.
- Sus aplicaciones no se limitan a un área en concreto, ya que al tratarse de un lenguaje multiparadigma permite utilizarlo en diseño de aplicaciones web o inteligencia artificial, entre muchos otras.
- Es apto para todas las plataformas, es decir, se puede ejecutar en cualquier sistema operativo usando el intérprete correspondiente.

El principal inconveniente del lenguaje de programación *Python* es que se trata de un lenguaje de programación interpretado, es decir, que no se compila, sino que se interpreta en tiempo de ejecución, dando como resultado a que sea más lento que otros lenguajes de programación como *Java*, *C* o *C++*. Pero este inconveniente no es un gran problema actualmente ya que en la actualidad las máquinas son muy potentes.

4.8.2. Django

Fue desarrollado inicialmente entre 2003 y 2005 por un equipo que era responsable de crear y mantener sitios webs de periódicos con el fin de reutilizar códigos y patrones de diseño comunes. *Django* es un *framework* web gratuito (MDN contributors, 2021) de alto nivel que permite el desarrollo rápido de sitios web seguros y mantenibles. *Django* se encarga de una gran parte de las complicaciones del desarrollo web, por lo que permite al programador centrarse únicamente en escribir las funcionalidades de la aplicación. Este es de código abierto, además de poseer una comunidad próspera y activa, mucha documentación y soporte gratuito. *Django* se encuentra escrito en *Python*.

El software escrito en *Django* es:

- Completo: Provee casi todo lo que los desarrolladores quisieran que traiga de serie un *framework* web.
- Versátil: Puede ser usado para construir casi cualquier tipo de sitio web, desde sistemas manejadores de contenidos y *wikis*, hasta redes sociales y sitios de noticias. Puede funcionar con cualquier *framework* en el lado del cliente, devolver contenido en una gran variedad de formatos (*HTML*, *RSS feeds*, *JSON*, *XML*, ...) y también usar distintos motores de bases de datos y plantillas.
- Seguro: Proporciona una manera segura de administrar cuentas de usuario y contraseñas, evitando errores comunes. Además, permite protección contra algunas vulnerabilidades de forma predeterminada, incluida la inyección *SQL*, scripts entre sitios, falsificación de solicitudes y *clickjacking*.
- Escalable: *Django* usa un componente en la arquitectura *shared-nothing*, en la que cada parte de esta arquitectura es independiente de las otras, y por lo tanto puede ser reemplazada o cambiada si es necesario. Habiendo una clara separación entre las diferentes partes significa que puede escalar para aumentar el tráfico al agregar *hardware*

en cualquier nivel, ya sean servidores cache, servidores de bases de datos o servidores de aplicación. Un ejemplo de sitios que han escalado a *Django* son Instagram y Disqus.

- Mantenible: Utiliza el principio No te repitas “*Don’t Repeat Yourself*” (*DRY*) para que no exista una duplicación innecesaria, reduciendo la cantidad de código.
- Portable: Está escrito en *Python*, el cual se ejecuta en muchas plataformas, este puede ejecutar sus aplicaciones en muchas distribuciones de *Linux*, *Windows* y *Mac OS X*. Además de contar con el respaldo de muchos proveedores de alojamiento web, y que a menudo proporcionan una infraestructura específica y documentación para el alojamiento de sitios *Django*.

Django funciona con patrón de diseño organizando la arquitectura en Modelo Vista Plantilla “*Model View Template (MVT)*”, este tiene una estructura muy similar a la arquitectura “*Model View Controller (MVC)*”, con la diferencia que la vista es el controlador intermediario entre el modelo, que se comunica con la base de datos, y las plantillas en formato *HTML* que reproducen la información y los formularios para pedir o cambiar información en la base de datos.

En un sitio web tradicional, una aplicación web espera peticiones *HTTP* del cliente. Cuando se recibe una petición la aplicación elabora la respuesta que se necesita basándose en la *URL* y a veces, en una información incluida en los datos de una petición *POST* o *GET*. La aplicación también devolverá una respuesta al explorador web, normalmente creando dinámicamente una página *HTML* con los datos correspondientes a mostrar.

Sin embargo, las aplicaciones web de *Django* agrupan el código que gestiona cada uno de estos pasos en ficheros separados.

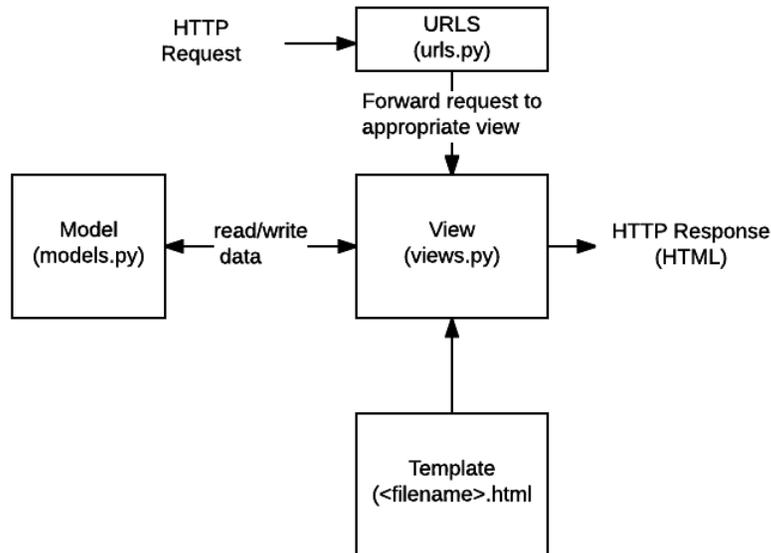


Figura 6. Funcionamiento Framework Django

URLs: Como se observa en la Figura 6, cuando se recibe una petición por parte de un cliente, esta se procesa en el archivo `urls.py`, de forma que funciona como un mapeador URL para redirigir las peticiones *HTTP* a la vista apropiada.

View: El mapeador *URL* redirige la petición a una vista en concreto, la cual accederá a los datos que se necesitan para la petición por medio del modelo y formateará la respuesta al cliente por medio de las plantillas.

Models: Los modelos definen la estructura de los datos de la aplicación y proporcionan mecanismos para gestionar esta (añadir, modificar y borrar) y consultar registros en la base de datos.

Templates: Las plantillas son ficheros de texto que definen la estructura o diagrama de otro fichero, con marcadores de posición para representar contenido real. La vista creará dinámicamente una página usando una plantilla, rellenando esta con los datos del modelo. Normalmente estos ficheros suelen estar codificados en lenguaje *HTML* pero no tiene porque ser así, pueden ser ficheros de cualquier tipo.

4.8.3. HTML

Fue creado por el científico de computación británico Timothy John Berners-Lee en 1991. *HTML* (Reyes, 2016) es un lenguaje de marcado de hipertexto (*HyperText Markup Language*), el cual se escribe en su totalidad con elementos, que a su vez están contruidos por etiquetas, contenido y atributos. *HTML* es el lenguaje que interpreta el navegador web para mostrar los sitios o aplicaciones web, tal y como están descritos.

Los elementos dentro de este lenguaje ayudan a estructurar y dar significado a las distintas partes de un documento *HTML*. Las etiquetas sirven para delimitar el inicio y el fin de un elemento. El contenido de un elemento pueden ser caracteres, comentarios u otro elemento delimitado dentro de las etiquetas de inicio y cierre.

4.8.4. CSS

CSS (Santos, 2022) es un lenguaje que se ocupa del diseño y la presentación de las páginas web, funciona junto con el lenguaje *HTML* que se encarga del contenido básico de las páginas. Sus siglas significan “hojas de estilo en cascada”, ya que puedes tener varias hojas y una de ellas con propiedades heredadas o en cascada de otras. Estas hojas de estilo permiten crear reglas para indicar a la página web como mostrar la información y el estilo de cada elemento de la página.

4.8.5. JavaScript

JavaScript (Ramos, 2021) es el lenguaje de programación encargado de dar más interactividad y dinamismo a las páginas web. Este no necesita ser compilado, ya que el navegador lee directamente el código sin necesidad de terceros. Se trata de uno de los tres lenguajes nativos de las páginas web junto a *HTML* (contenido y estructura) y a *CSS* (diseño del contenido y estructura).

Conceptos teóricos → Tecnologías y herramientas para el desarrollo web

Este lenguaje de programación se ejecuta en la máquina de los clientes, que permite crear efectos y animaciones o responder a eventos causados por el usuario como pulsar botones o modificar el *DOM* (*document object model*). El código *JavaScript* se ejecuta en todos los navegadores, tanto de escritorio como móviles. *JavaScript* es capaz de detectar errores en formularios, crear sliders que se adapten a cualquier pantalla, hacer cálculos matemáticos de forma eficiente, modificar elemento de una página web de forma sencilla y muchas más cosas.

También, existe una tecnología llamada *AJAX* que permite intercambiar información con el servidor sin la necesidad de recargar la página. Esta tecnología ha supuesto uno de los principales avances en el desarrollo web. Como ejemplo, esta tecnología es la encargada de conseguir más mensajes, *tweets*, *emails*... con el único hecho de pulsar un botón, sin tener que recargar la página.

4.8.6. Bootstrap

Bootstrap (Invitado, 2020) es un *framework* CSS desarrollado por Twitter en 2010, para estandarizar las herramientas de la compañía. Este *framework* combina CSS y *JavaScript* para estilizar los elementos de las páginas web *HTML*, además proporciona interactividad en las páginas incluyendo una serie de componentes que facilitan la comunicación con el usuario, como menús de navegación, controles de página y barras de progreso.

El principal objetivo de Bootstrap es permitir la construcción de sitios web responsive (los elementos de la página escalan en función de la resolución de la pantalla) para dispositivos móviles, ordenadores y tablets, de una manera simple y organizada.

La estructura de este *framework* se compone en dos directorios:

- css: Contiene los archivos necesarios para la estilización de los elementos y una alternativa al tema original.

Conceptos teóricos → Tecnologías y herramientas para el desarrollo web

- js: Contiene la parte posterior del archivo bootstrap.js, responsable de la ejecución de aplicaciones de estilo que requieren manipulación interactiva.

Para asignar una característica a un elemento, hay que informar la clase correspondiente en la propiedad “*class*” del elemento que será estilizado.

4.8.7. Base de datos

Una base de datos (Oracle, 2014) es una recopilación organizada de información o datos estructurados, que normalmente se almacenan de forma electrónica en un sistema informático. Normalmente, una base de datos se controla por un sistema de gestión de bases de datos (*DBMS*). El conjunto de los datos, el *DBMS* y las aplicaciones asociadas a ellos, reciben el nombre de sistema de bases de datos.

Los datos de los tipos más comunes de bases de datos en funcionamiento actualmente se suelen utilizar como estructuras de filas y columnas en una serie de tablas para aumentar la eficacia del procesamiento y consulta de datos. La mayoría de las bases de datos utilizan un lenguaje de consulta estructurada denominado *SQL* para escribir y consultar datos.

4.8.7.1. SQLite3

SQLite es un motor (Velasco, 2021) de bases de datos muy ligero, de código abierto y escrito en *C*, donde podremos guardar todo tipo de información relacionada con un programa o una *app*. Su principal ventaja es que, a diferencia de lo que ocurre con otros motores de base de datos, *SQLite* funciona como un servidor propio e independiente, evitando tener que realizar consultas externas en procesos separados, es decir, se incluye la base de datos y el motor dentro del programa y los datos se consultan, modifican o guardan desde este mismo, eliminando la necesidad de tener otros servicios abiertos en segundo plano.

Conceptos teóricos → Tecnologías y herramientas para el desarrollo web

SQLite hace uso de *SQL*, por ello, las consultas y órdenes se pueden enviar a este servidor directamente en este lenguaje. Además, permite combinar estas instrucciones con scripts en *Python* para poder analizar datos complejos.

Algunas de sus características son:

- Permite trabajar sin problemas con bases de datos de hasta 2TB de tamaño.
- Estas bases de datos cuentan con la mayor parte del estándar *SQL-92*, por lo que funcionan sin problemas con otros programas que trabajen con instrucciones *SQL*.
- Cuenta con un sistema de tipos inusual, asignando cada tipo de valores inusuales.
- Permite que varios procesos o hilos consulten la misma base de datos sin problemas, lo que se traduce en una mejora considerable de rendimiento frente a otras alternativas.

4.8.8. Visual Studio Code

Es un editor (Flores, 2022) de código fuente desarrollado por *Microsoft* para *Windows*, *Linux* y *macOS*. Se trata de un software libre y multiplataforma, tiene buena integración con *Git*, cuenta con soporte para la depuración de código y dispone de una gran cantidad de extensiones que dan la posibilidad de escribir y ejecutar código en cualquier lenguaje de programación.

Algunas de sus características son:

- Multiplataforma: Está disponible para cualquier sistema operativo.
- IntelliSense: Proporciona sugerencias de código, terminaciones inteligentes y resaltado de sintaxis, lo que permite ser más ágil a la hora de escribir código.
- Depuración: Incluye la función de depuración que ayuda a detectar errores en el código.
- Uso del control de versiones: Tiene compatibilidad con *Git*, por lo que permite revisar diferencias, organizar archivos, realizar *commits* y hacer *push* y *pull* desde el editor.

Conceptos teóricos → Tecnologías y herramientas para el desarrollo web

- Extensiones: Permite personalizar y agregar funcionalidad adicional de forma modular y aislada. Las extensiones no afectan al rendimiento del editor, ya que se ejecutan en procesos diferentes.

4.9. Tecnologías y herramientas de control de versiones

4.9.1. *Git*

Es un software libre (Rubio, 2019) de control de versiones distribuidas diseñado por Linus Torvalds, pensado en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Su propósito es llevar un registro de los cambios en archivos de computadora incluyendo coordinar el trabajo que varias personas realizan sobre archivos compartidos en un repositorio de código.

4.9.2. *GitHub*

Es una plataforma gratuita (Camacho, 2021) que ofrece a los desarrolladores la posibilidad de crear repositorios de código y guardarlos en la nube de forma segura, usando el sistema de control de versiones *Git*. Facilita la organización de proyectos y permite la colaboración de varios desarrolladores en tiempo real, es decir, permite centralizar el contenido del repositorio para poder colaborar con otros miembros. Es propiedad de *Microsoft*.

Ventajas:

- Permite alojar proyectos de forma gratuita.
- Permite alojar tanto repositorios públicos como privados.
- Facilita compartir proyectos de forma fácil y crear un portafolio.
- Permite colaborar para mejorar proyectos

4.10. Tecnologías y herramientas de documentación del proyecto

4.10.1. Visual Paradigm

Es una herramienta CASE (Ingeniería de Software Asistida por Computación), que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (EcuRed, 2016). Esta ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo *software* a través de la representación de todo tipo de diagramas. Constituye una herramienta privada disponible en varias ediciones, cada una destinada a satisfacer diferentes necesidades.

4.10.2. Microsoft Office Project

Es un *software* (Machuca, 2022) de administración de proyectos y programas de proyectos desarrollado y comercializado por Microsoft para asistir a los administradores de proyectos en el desarrollo de planes, asignar recursos a tareas, dar seguimiento al progreso, administrar presupuesto y analizar cargas de trabajo.

4.10.3. EZEstimate

Herramienta (García, 2021) que permite estimar el esfuerzo de un proyecto a partir de un valor en cada uno de los factores de complejidad técnica y factores de complejidad del entorno, que en su conjunto forman los puntos de los casos de uso de un proyecto. En la Figura 7 se pueden observar cada uno de los diferentes factores y en la Figura 8 se puede observar la interfaz del programa.

Factores de complejidad técnica		
T ₁ Sistemas distribuidos	T ₆ Facilidad de instalación	T ₁₁ Características especiales de seguridad
T ₂ Rendimiento	T ₇ Facilidad de uso	
T ₃ Eficiencia del usuario final	T ₈ Portabilidad	T ₁₂ Acceso directo a terceras partes
T ₄ Procesamiento interno complejo	T ₉ Facilidad de cambio	T ₁₃ Se requiere entrenamiento especial del usuario
T ₅ Reusabilidad	T ₁₀ Concurrencia	

Factores de complejidad del entorno	
E ₁ Familiaridad con UML	E ₅ Experiencia en orientación a objetos
E ₂ Trabajadores a tiempo parcial	E ₆ Motivación
E ₃ Capacidad de los analistas	E ₇ Dificultad del lenguaje de programación
E ₄ Experiencia en la aplicación	E ₈ Estabilidad de los requisitos

Figura 7. Puntos de casos de uso

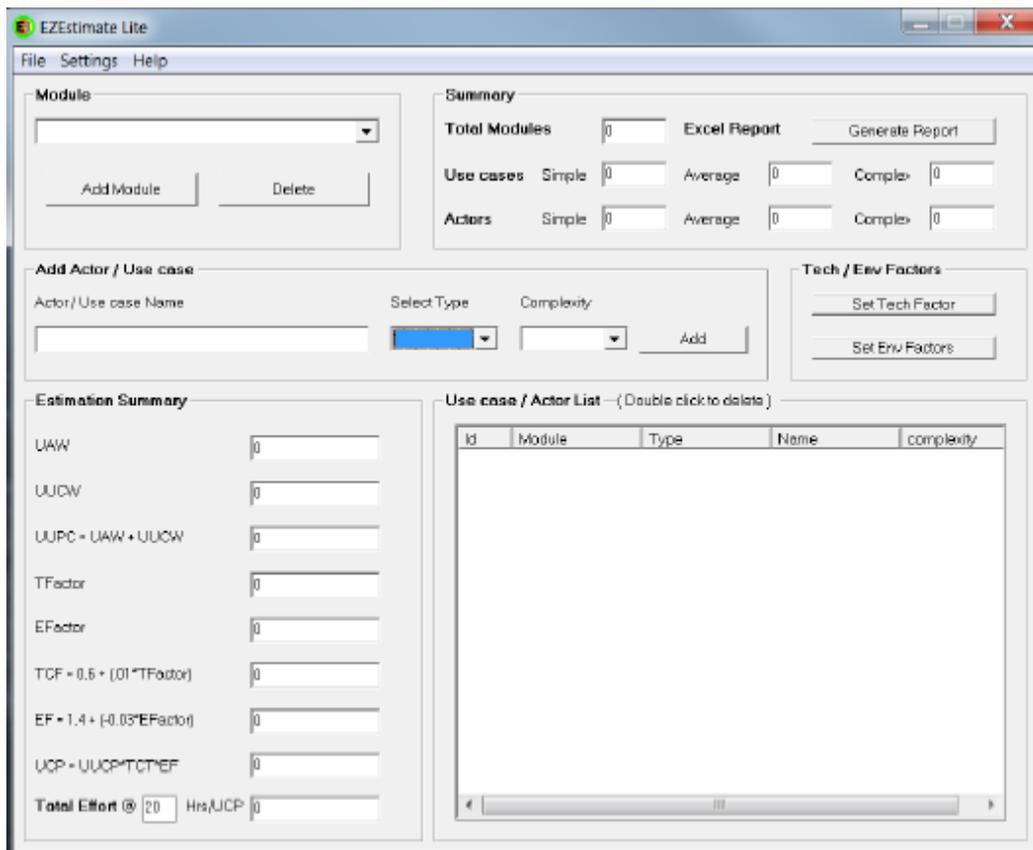


Figura 8. Interfaz EZEstimate

4.10.4. Sphinx

Sphinx (Sphinx, 2022) es un generador de documentación de *Python* o una herramienta que traduce un conjunto de archivos fuente de texto plano a varios formatos de salida, produciendo automáticamente referencias cruzadas, índices, etc. Es decir, si hay un directorio que contiene bastantes documentos *reStructuredText* o *Markdown*, *Sphinx* puede generar una serie de archivos *HTML*, un archivo *PDF*, páginas de manual y mucho más. *Sphinx* se enfoca, en particular, en la documentación escrita a mano.

5. Desarrollo de la propuesta

En este apartado nos centraremos en explicar las fases más importantes del desarrollo de la plataforma, así como, la metodología usada, el diseño de la base de datos, el proceso de obtención de datos de Instagram y el desarrollo web de la aplicación.

5.1. Definición de la propuesta

Para el desarrollo de este proyecto, se ha hecho uso del Proceso Unificado, el cual es más que un simple proceso, es un marco de trabajo genérico que puede especializarse (Francisco José García Peñalvo et al., 2019) para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos.

Algunas de sus características son:

- Está basado en componentes.
- Utiliza *UML*.
- Es un proceso conducido por casos de uso.
- Está centrado en la arquitectura.
- Es iterativo e incremental.

Puntos a tener en cuenta dentro en el marco de trabajo:

- No existe un proceso universal
- Puede extenderse y especializarse para una gran variedad de sistemas software.
- Permite gran variedad de ciclo de vida, pudiéndose definir diferentes conjuntos de productos o definir actividades y encargados de las mismas.

Desarrollo de la propuesta → Definición de la propuesta

El Proceso Unificado se repite a lo largo de una serie de ciclos de desarrollo que constituyen la vida de un sistema. Cada ciclo de desarrollo concluye una versión entregable del producto. Cada ciclo de vida consta de 4 fases:

- Inicio: Se define el alcance del proyecto y se desarrollan los casos de negocio.
- Elaboración: Se planifica el proyecto, se especifican en detalle la mayoría de los casos de uso y se diseña la arquitectura del sistema.
- Construcción: Se construye el producto.
- Transición: El producto se convierte en versión beta, se corrigen problemas y se incorporan mejoras sugeridas en la revisión.

Dentro de cada fase se puede, a su vez, descomponer el trabajo en iteraciones con sus incrementos resultantes. Cada fase termina con un punto de control llamado hito, en el que los participantes del proyecto revisan el progreso de este. En la Figura 9 se puede observar un gráfico del ciclo de vida de un proyecto.

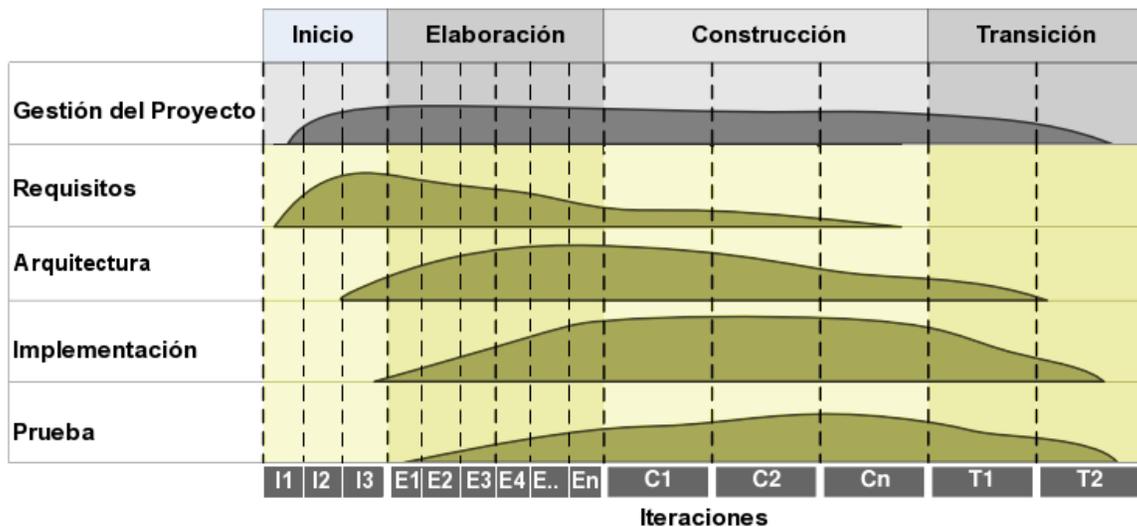


Figura 9. Ciclo de vida de un proyecto

Desarrollo de la propuesta → Definición de la propuesta

5.1.1. Planificación del proyecto

Para la realización de una estimación del esfuerzo se ha hecho uso del programa *EZEstimate* (ver herramienta *EZEstimate*), donde podemos realizar una predicción sobre los costes en relación temporal y personal. Para poder realizar esta predicción, se ha realizado mediante puntos de casos de uso (*UCP*), que considera actores, escenarios y factores técnico y de entorno, para obtener una estimación del esfuerzo basada en meses de persona.

Una vez realizada la estimación temporal, se ha realizado la planificación temporal del proyecto con el fin de crear un calendario de trabajo, identificar las tareas a realizar para resolver el proyecto, asignar los recursos a las diferentes tareas y estimar las duraciones de estas. Esta planificación se ha realizado en el programa *Microsoft Office Project* (ver herramienta *Microsoft Office Project*).

Para ver la información sobre la planificación del proyecto, ver el Anexo A – Plan del proyecto software.

5.1.2. Especificación de requisitos software y análisis

Para la realización de los requisitos software se ha usado la metodología de Durán y Bernárdez. Donde se especifican los participantes de proyecto, los objetivos del sistema, se definen los actores y se realiza la captura de requisitos del sistema, mediante la definición de los requisitos de almacenamiento de información, requisitos funcionales y no funcionales. Para terminar, se realiza el diagrama de los casos de uso.

Posteriormente se ha realizado el modelo del análisis, que consta del modelo del dominio que muestra todas las entidades o clases de almacenamiento de información y como se relacionan estas entre sí. También consta de un diagrama de secuencia por cada caso de uso, que muestran como interactúan los objetos en un sistema *UML*. Por último, la propuesta de la vista de

Desarrollo de la propuesta → Definición de la propuesta

arquitectura, que muestra en que paquete se encuentran las interfaces, objetos de control y entidades, de forma que se puede observar donde están estas y con que paquetes se relacionan.

Para ver la información sobre la especificación de requisitos *software* y análisis, ver el Anexo B – Especificación de requisitos y análisis.

5.1.3. Especificación de diseño

En esta parte se realiza la especificación del diseño del sistema a implementar, por lo que se especifica y explica el patrón arquitectónico de diseño que va a implementar el sistema con respecto a la tecnología utilizada (*Django*), se presentan todos los diagramas de los subsistemas que componen el sistema y las clases que compone cada subsistema con las operaciones a realizar por cada una de estas, todo esto empleando el patrón de diseño que ha sido explicado anteriormente.

Posteriormente se han realizado los diagramas de secuencia de los casos de uso del diseño, los cuales guardan gran relación con la estructura de los subsistemas presentada anteriormente. Finalmente, también se presenta el diagrama de arquitectura de capas del sistema para mostrar en qué nivel se sitúa cada parte del sistema y se presenta el diagrama de despliegue que muestra los componentes que tiene el sistema y como interactúan estos componentes una vez el sistema esté desplegado.

Para ver la información sobre la especificación de diseño, ver el Anexo C – Especificación de diseño.

5.2. Diseño de la base de datos

En esta parte nos centraremos en explicar cómo está formada la base de datos y que tipo de relaciones tienen las tuplas que forman esta. Cabe resaltar que para el desarrollo de este proyecto se ha hecho uso del *framework Django* (ver concepto teórico *Django*), el cual utiliza como lenguaje de programación base *Python* (ver concepto teórico *Python*). Una vez sabido esto podemos empezar a explicar el diseño de la base de datos.

Para el desarrollo de la plataforma se ha hecho uso de una base de datos relacional gestionada por el sistema de gestión de bases de datos *SQLite3* (ver concepto teórico *SQLite3*). En esta almacenaremos información sobre los usuarios, así como, la información relevante obtenida de las cuentas, publicaciones o historias destacadas de Instagram.

5.2.1. Almacenamiento de usuarios

Primero empezaremos hablando de la información que deberá almacenarse acerca de los usuarios que se registren en la plataforma. Para ello vamos a observar la Figura 10 donde se pueden ver las clases, atributos y relaciones, de la información que puede contener un usuario.

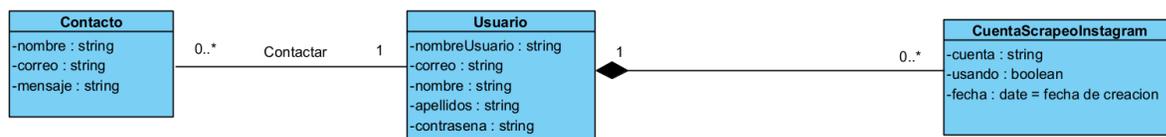


Figura 10. Diagrama de clases del usuario

Como se observa un usuario puede tener varias cuentas para hacer *scraping* en Instagram o ninguna y puede enviar mensajes al creador o no. Si un usuario borra su cuenta tanto la información del usuario como las cuentas de *scraping* que estén asociadas a este usuario se borran de la base de datos, pero la información del contacto no se borra.

Desarrollo de la propuesta → Diseño de la base de datos

Entonces, para poder realizar este diseño dentro de la base de datos, en *Django* hay que modificar el archivo 'models.py' del proyecto (ver concepto teórico *Django*), mediante la declaración de clases que contienen campos que representan los atributos del diagrama de la Figura 10 y a su vez estos campos contienen opciones para poder realizar relaciones o limitar tanto el tipo de dato de entrada como el tamaño.

- Contacto:

En la Figura 11 podemos observar la declaración de la clase 'Contacto', dentro del archivo 'models.py' del proyecto, que crea una tupla en la base de datos y los campos 'nombre', 'correo' y 'mensaje' son los atributos de la tupla. En la Tabla 1 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'Contacto'.

```
class Contacto(models.Model):  
    nombre = models.CharField(max_length=50)  
    correo = models.EmailField()  
    mensaje = models.TextField()
```

Figura 11. Clase Contacto

Campo	Tipo	Descripción
nombre	CharField(max_length = 50)	Campo de tipo cadena con un tamaño máximo de 50 caracteres. Almacena el nombre.
correo	EmailField()	Campo de tipo email. Para que sea válido deberá tener un formato de correo, es decir por lo menos con un símbolo '@'. Almacena el correo del usuario que contacta.

mensaje	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena el mensaje a enviar.
----------------	-------------	--

Tabla 1. Información de atributos de la tupla 'Contacto'

• CuentaScrapeoInstagram:

En la Figura 12 podemos observar la declaración de la clase 'CuentaScrapeoInstagram' dentro del archivo 'models.py' del proyecto, que al igual que la clase anterior formará una tupla en la base de datos, pero en este caso observamos un campo usuario que no vemos en el diagrama de la Figura 10, este campo es una clave externa a la clase 'Usuario', mediante el cual creamos una relación de un 'Usuario' a muchas 'CuentaScrapeoInstagram'. Además con la opción 'on_delete = models.CASCADE' conseguimos hacer que si la cuenta de un usuario es eliminada todos las cuentas para *scraping* Instagram que tenga este usuario serán borradas de la base de datos.

Al igual que la clase anterior, la clase crea una tupla y los campos son los atributos de la tupla en la base de datos. En la Tabla 2 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'CuentaScrapeoInstagram'.

```
class CuentaScrapeoInstagram(models.Model):
    usuario = models.ForeignKey(Usuario, on_delete=models.CASCADE)
    cuenta = models.CharField(max_length=100)
    usando = models.BooleanField()
    fecha = models.DateTimeField(default=timezone.now)
```

Figura 12. Clase cuenta scraping Instagram

Campo	Tipo	Descripción
usuario	ForeignKey(Usuario, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'Usuario'. Almacena el identificador del usuario que crea esta tupla. Con la opción 'on_delete=models.CASCADE' si se elimina un usuario se borran todas la tuplas 'CuentaScrapeoInstagram' que se hayan creado de la base de datos.
cuenta	CharField(max_length = 100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de usuario de la cuenta de Instagram guardada.
usando	BooleanField()	Campo de tipo booleano. Indica si la cuenta de Instagram se está usando.
fecha	DateTimeField(default=timezone .now)	Campo de tipo fecha. Almacena la fecha y hora de creación.

Tabla 2. Información de atributos de la tupla 'CuentaScrapeoInstagram'

- Usuario:

Clase que representa la información de un usuario en la base de datos. Tiene una declaración similar a las clases anteriores en *Django*. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 3 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'Usuario'. También almacena más campos, pero el almacenamiento y modificación de estos es totalmente automático por parte de *Django*, ya que esta clase hereda de la clase genérica 'AbstractUser'.

Desarrollo de la propuesta → Diseño de la base de datos

Campo	Tipo	Descripción
username	CharField(max_length=150)	Campo de tipo cadena con un tamaño máximo de 150 caracteres. Almacena el nombre de usuario.
first_name	CharField(max_length=150)	Campo de tipo cadena con un tamaño máximo de 150 caracteres. Almacena el nombre.
last_name	CharField(max_length=150)	Campo de tipo cadena con un tamaño máximo de 150 caracteres. Almacena los apellidos.
email	EmailField()	Campo de tipo email. Para que sea válido deberá tener un formato de correo, es decir por lo menos con un símbolo '@'. Almacena el correo del usuario.
password	CharField(max_length=128)	Campo de tipo cadena con un tamaño máximo de 128 caracteres, que almacena la contraseña del usuario. Antes de almacenar esta contraseña se encripta y se almacena encriptada.

Tabla 3. Información de atributos de la tupla 'Usuario'

5.2.2. Almacenamiento de cuentas de Instagram buscadas

Para que la plataforma sea más eficiente y no tenga que estar realizando peticiones a Instagram siempre que se quiera buscar una cuenta de Instagram que recientemente ha sido buscada, se almacena la información más importante de la cuenta de Instagram buscada en la base de datos. Aun así, como la plataforma debe tener la información lo más actualizada posible, si pasados tres minutos se vuelve a buscar una cuenta que está almacenada en base de datos, se busca mediante *scraping* en Instagram y se actualizan los datos de la cuenta en la base de datos. Si observamos la Figura 13 podemos ver las clases, atributos y relaciones de la información que puede contener una cuenta de Instagram buscada.

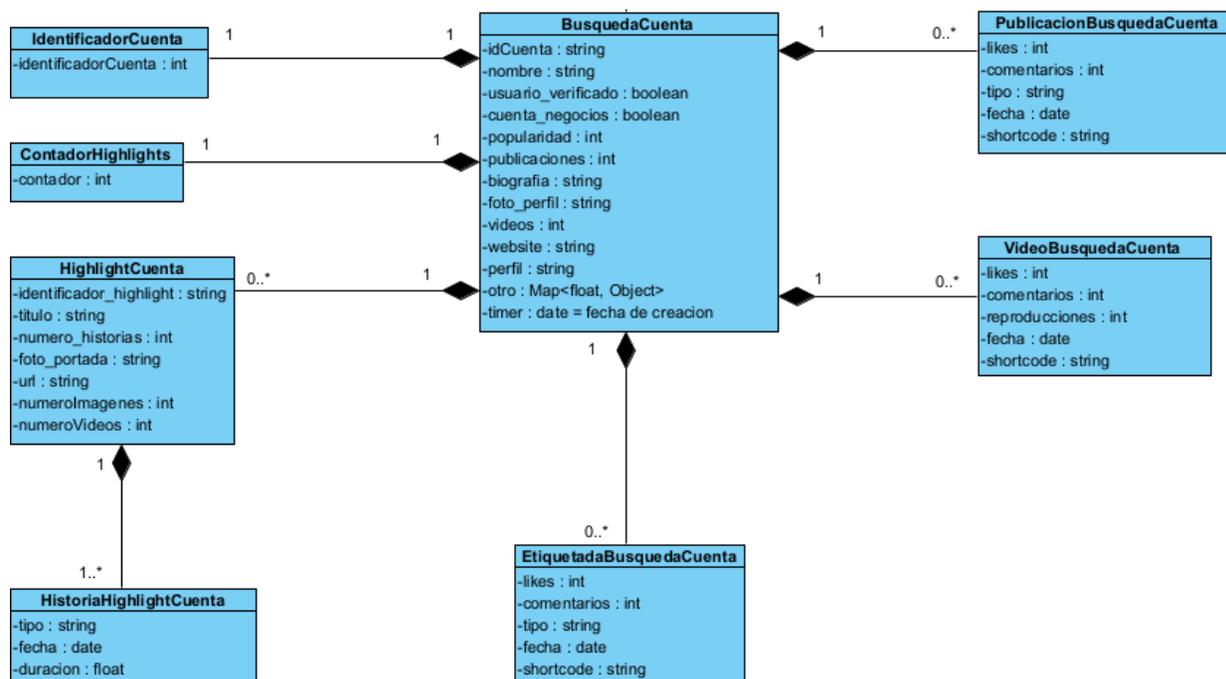


Figura 13. Diagrama de cuenta de Instagram buscada

Desarrollo de la propuesta → Diseño de la base de datos

Como se observa una búsqueda de cuenta solamente puede tener un identificador de cuenta y un contador de *highlights*, también una búsqueda de cuenta puede tener varias publicaciones o ninguna, varios vídeos o ninguno, varias publicaciones en la que salga etiquetada o ninguna y varios *highlights* o ninguno, que a su vez estos tendrán entre una y muchas historias. Si se borra una búsqueda de cuenta de la base de datos, también se borra la información de todo lo demás, ya que dependen de esta.

El proceso para definir esta estructura en la base de datos se realiza igual que en el apartado anterior, mediante la declaración de clases, sus respectivos campos y opciones, en el archivo 'models.py' del proyecto (ver concepto teórico *Django*).

- DatosBusquedaUsuario:

Clase que representa la información de una cuenta de Instagram buscada. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 4 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosBusquedaUsuario'.

Campo	Tipo	Descripción
IDcuenta	CharField(primary_key=True, max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el id del usuario de la cuenta de Instagram buscada, que es la clave primaria de la tupla en la base de datos.
Nombre	TextField(blank=True)	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena el nombre de la cuenta de Instagram buscada, si no tiene se almacena como cadena vacía.

Desarrollo de la propuesta → Diseño de la base de datos

Usuario_verificado	BooleanField()	Campo de tipo booleano. Indica si la cuenta de Instagram buscada está verificada.
Popularidad	IntegerField()	Campo de tipo entero. Almacena la cantidad de seguidores de la cuenta de Instagram buscada.
Perfil	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> del perfil de la cuenta de Instagram buscada.
Usuario_privado	BooleanField()	Campo de tipo booleano. Indica si la cuenta de Instagram buscada es privada.
Cuenta_negocios	BooleanField()	Campo de tipo booleano. Indica si la cuenta de Instagram buscada es de negocios.
Publicaciones	IntegerField()	Campo de tipo entero. Almacena el número de publicaciones de la cuenta de Instagram buscada.
Biografia	TextField(blank=True)	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la biografía de la cuenta de Instagram buscada, si no tiene se almacena como cadena vacía.
Foto_perfil	TextField(blank=True)	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> de la foto de perfil de la cuenta de Instagram buscada, si no tiene se almacena como cadena vacía.
Videos	IntegerField()	Campo de tipo entero. Almacena el número de vídeos de la cuenta de Instagram buscada.
Website	TextField(blank=True)	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la página web de la cuenta de Instagram buscada, si no tiene se almacena como cadena vacía.
mediaLikesPostRecientes	FloatField()	Campo de tipo número real. Almacena la media de likes de las últimas 20 publicaciones de la cuenta de Instagram buscada.
mediaComentariosPostRecientes	FloatField()	Campo de tipo número real. Almacena la media de comentarios de las últimas 20 publicaciones de la cuenta de Instagram buscada.
numeroImágenesPostRecientes	IntegerField()	Campo de tipo entero. Almacena el número de imágenes de las últimas 20 publicaciones de la cuenta de Instagram buscada.

Desarrollo de la propuesta → Diseño de la base de datos

numeroVideosPostRecientes	IntegerField()	Campo de tipo entero. Almacena el número de vídeos de las últimas 20 publicaciones de la cuenta de Instagram buscada.
numeroSidecarsPostRecientes	IntegerField()	Campo de tipo entero. Almacena el número de <i>sidecars</i> de las últimas 20 publicaciones de la cuenta de Instagram buscada.
mediaLikesVideos	FloatField()	Campo de tipo número real. Almacena la media de likes de los últimos 20 vídeos de la cuenta de Instagram buscada.
mediaComentariosVideos	FloatField()	Campo de tipo número real. Almacena la media de comentarios de los últimos 20 vídeos de la cuenta de Instagram buscada.
mediaVisualizacionesVideos	FloatField()	Campo de tipo número real. Almacena la media de visualizaciones de los últimos 20 vídeos de la cuenta de Instagram buscada.
mediaLikesPublicacionesEtiquetadas	FloatField()	Campo de tipo número real. Almacena la media de likes de las últimas 20 publicaciones en las que la cuenta de Instagram buscada sale etiquetada.
mediaComentariosPublicacionesEtiquetadas	FloatField()	Campo de tipo número real. Almacena la media de comentarios de las últimas 20 publicaciones en las que la cuenta de Instagram buscada sale etiquetada.
numeroImágenesPublicacionesEtiquetadas	IntegerField()	Campo de tipo entero. Almacena el número de imágenes de las últimas 20 publicaciones en las que la cuenta de Instagram buscada sale etiquetada.
numeroVideosPublicacionesEtiquetadas	IntegerField()	Campo de tipo entero. Almacena el número de vídeos de las últimas 20 publicaciones en las que la cuenta de Instagram buscada sale etiquetada.
numeroSidecarsPublicacionesEtiquetadas	IntegerField()	Campo de tipo entero. Almacena el número de vídeos de las últimas 20 publicaciones en las que la cuenta de Instagram buscada sale etiquetada.
timer	DateTimeField(auto_now_add=True)	Campo de tipo fecha. Almacena la fecha y hora de creación. Este campo es usado a modo de temporizador.

Tabla 4. Información de atributos de la tupla 'DatosBusquedaUsuario'

Desarrollo de la propuesta → Diseño de la base de datos

- IdentificadorUsuario:

Instagram proporciona un identificador numérico único a cada cuenta que se encuentre registrada dentro de la red social.

'IdentificadorUsuario' es una clase que representa el número de identificación del usuario de una cuenta de Instagram buscada. El campo 'cuentaID' de esta clase crea una relación uno a uno con la clase 'DatosBusquedaUsuario'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 5 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'IdentificadorUsuario'.

Campo	Tipo	Descripción
cuentaID	OneToOneField(DatosBusquedaUsuario, on_delete=models.CASCADE)	Almacena la clave primaria de la búsqueda de cuenta de Instagram a la que se encuentra asociada. Crea una relación uno a uno. Con la opción 'on_delete=models.CASCADE' si se elimina una búsqueda de cuenta de Instagram se borra la tupla asociada 'IdentificadorUsuario' de la base de datos.
usuarioID	IntegerField()	Campo de tipo entero. Almacena el número de identificador de la cuenta de Instagram.

Tabla 5. Información de atributos de la tupla 'IdentificadorUsuario'

- ContadorHighlights:

Clase que representa el número de conjuntos de historias destacadas que posee una cuenta de Instagram buscada. El campo 'cuentaID' de esta clase crea una relación uno a uno con la clase 'DatosBusquedaUsuario'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 6 podemos observar tanto los campos como el tipo de dato *Django* y una

Desarrollo de la propuesta → Diseño de la base de datos

descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'ContadorHighlights'.

Campo	Tipo	Descripción
cuentaID	OneToOneField(DatosBusqueda aUsuario, on_delete=models.CASCADE)	Almacena la clave primaria de la búsqueda de cuenta de Instagram a la que se encuentra asociada. Crea una relación uno a uno. Con la opción 'on_delete=models.CASCADE' si se elimina una búsqueda de cuenta de Instagram se borra la tupla asociada 'ContadorHighlights' de la base de datos.
contadorHighlights	IntegerField()	Campo de tipo entero. Almacena la cantidad de historias destacadas de la cuenta de Instagram.

Tabla 6. Información de atributos de la tupla 'ContadorHighlights'

• DatosHighlight:

Clase que representa la información de un conjunto de historias destacadas que posee una cuenta de Instagram buscada. El campo 'cuentaID' de esta clase crea una relación de un objeto 'DatosBusquedaUsuario' a muchos objetos 'DatosHighlight'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 7 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosHighlight'.

Campo	Tipo	Descripción
cuentaID	ForeignKey(DatosBusquedaUsuario, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosBusqueda'. Almacena el identificador de la cuenta buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la

Desarrollo de la propuesta → Diseño de la base de datos

		cuenta buscada se borran todas la tuplas de 'DatosHighlight' que estén relacionadas con la cuenta buscada.
identificador_highlights_usuario	CharField(primary_key=True, max_length=50)	Campo de tipo cadena con un tamaño máximo de 50 caracteres. Almacena el identificador único del conjunto de historias destacadas de la cuenta de Instagram buscada, que es la clave primaria de la tupla en la base de datos.
titulo	CharField(max_length=30)	Campo de tipo cadena con un tamaño máximo de 50 caracteres. Almacena el identificador único del conjunto de historias destacadas de la cuenta de Instagram buscada.
numero_historias	IntegerField()	Campo de tipo entero. Almacena el número de historias del conjunto de historias destacadas de la cuenta de Instagram buscada.
foto_portada	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> de la foto de portada del conjunto de historias destacadas de la cuenta de Instagram buscada.
url	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> al conjunto de historias destacadas de la cuenta de Instagram buscada.
numeroImágenes	IntegerField()	Campo de tipo entero. Almacena el número de historias que son imágenes del conjunto de historias destacadas de la cuenta de Instagram buscada.
numeroVideos	IntegerField()	Campo de tipo entero. Almacena el número de historias que son videos del conjunto de historias destacadas de la cuenta de Instagram buscada.

Tabla 7. Información de atributos de la tupla 'DatosHighlight'

• DatosStoryHighlight:

Clase que representa la información de una historia que pertenece a un conjunto de historias destacadas que posee una cuenta de Instagram buscada. El campo 'highlight' de esta clase crea una relación de un objeto 'DatosHighlight' a muchos objetos 'DatosStoryHighlight'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 8 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosStoryHighlight'.

Campo	Tipo	Descripción
highlight	ForeignKey(DatosHighlight, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosHighlight'. Almacena el identificador del conjunto de las historias destacadas de la cuenta buscada. Con la opción 'on_delete=models.CASCADE' si se elimina el conjunto de historias destacadas se borran todas las tuplas de 'DatosStoryHighlight' que estén relacionadas con la cuenta buscada.
tipo	CharField(max_length=20)	Campo de tipo cadena con un tamaño máximo de 20 caracteres. Almacena tipo de historia (imagen o vídeo) del conjunto historias destacadas de la cuenta de Instagram buscada.
fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación de la historia del conjunto de historias destacadas de la cuenta buscada.
duracion	FloatField()	Campo de tipo número real. Almacena la duración de la historia del conjunto de historias destacadas de la cuenta buscada.

Tabla 8. Información de atributos de la tupla 'DatosStoryHighlight'

Desarrollo de la propuesta → Diseño de la base de datos

• DatosPostBusquedaUsuario:

Clase que representa la información resumida de una publicación de una cuenta de Instagram buscada. El campo 'cuentaID' de esta clase crea una relación de un objeto 'DatosBusquedaUsuario' a muchos objetos 'DatosPostBusquedaUsuario'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 9 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosPostBusquedaUsuario'.

Campo	Tipo	Descripción
cuentaID	ForeignKey(DatosBusquedaUsuario, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosBusqueda'. Almacena el identificador de la cuenta buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la cuenta buscada se borran todas la tuplas de 'DatosPostBusquedaUsuario' que estén relacionadas con la cuenta buscada.
likes	IntegerField()	Campo de tipo entero. Almacena el número de likes de una publicación de la cuenta de Instagram buscada.
comentarios	IntegerField()	Campo de tipo entero. Almacena el número de comentarios de una publicación de la cuenta de Instagram buscada.
tipo	CharField(max_length=20)	Campo de tipo cadena con un tamaño máximo de 20 caracteres. Almacena el tipo de publicación (imagen o vídeo) de la cuenta de Instagram buscada.
fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación de una publicación de la cuenta de Instagram buscada.

Desarrollo de la propuesta → Diseño de la base de datos

shortcode	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el identificador único de una publicación de la cuenta de Instagram buscada.
------------------	---------------------------	--

Tabla 9. Información de atributos de la tupla 'DatosPostBusquedaUsuario'

• DatosVideosBusquedaUsuario:

Clase que representa la información resumida de un vídeo de una cuenta de Instagram buscada. El campo 'cuentaID' de esta clase crea una relación de un objeto 'DatosBusquedaUsuario' a muchos objetos 'DatosVideosBusquedaUsuario'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 10 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosVideosBusquedaUsuario'.

Campo	Tipo	Descripción
cuentaID	ForeignKey(DatosBusquedaUsuario, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosBusqueda'. Almacena el identificador de la cuenta buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la cuenta buscada se borran todas la tuplas de 'DatosVideosBusquedaUsuario' que estén relacionadas con la cuenta buscada.
reproducciones	IntegerField()	Campo de tipo entero. Almacena el número de reproducciones de un vídeo de la cuenta de Instagram buscada.
likes	IntegerField()	Campo de tipo entero. Almacena el número de likes de un vídeo de la cuenta de Instagram buscada.
comentarios	IntegerField()	Campo de tipo entero. Almacena el número de comentarios de un vídeo de la cuenta de Instagram buscada.

Desarrollo de la propuesta → Diseño de la base de datos

fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación del vídeo de la cuenta de Instagram buscada.
shortcode	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el identificador único del vídeo de la cuenta de Instagram buscada.

Tabla 10. Información de atributos de la tupla 'DatosVideosBusquedaUsuario'

• DatosEtiquetadasBusquedaUsuario:

Clase que representa la información resumida de una publicación en la que la cuenta de Instagram buscada ha sido etiquetada. El campo 'cuentaID' de esta clase crea una relación de un objeto 'DatosBusquedaUsuario' a muchos objetos 'DatosEtiquetadasBusquedaUsuario'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 11 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosEtiquetadasBusquedaUsuario'.

Campo	Tipo	Descripción
cuentaID	ForeignKey(DatosBusquedaUsuario, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosBusqueda'. Almacena el identificador de la cuenta buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la cuenta buscada se borran todas la tuplas de 'DatosEtiquetadasBusquedaUsuario' que estén relacionadas con la cuenta buscada.
likes	IntegerField()	Campo de tipo entero. Almacena el número de likes de una publicación en la que la cuenta de Instagram buscada ha sido etiquetada.

Desarrollo de la propuesta → Diseño de la base de datos

comentarios	IntegerField()	Campo de tipo entero. Almacena el número de comentarios de una publicación en la que la cuenta de Instagram buscada ha sido etiquetada.
tipo	CharField(max_length=20)	Campo de tipo cadena con un tamaño máximo de 20 caracteres. Almacena el tipo de publicación (imagen o vídeo) en la que la cuenta de Instagram buscada ha sido etiquetada.
fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación de una publicación en la que la cuenta de Instagram buscada ha sido etiquetada.
shortcode	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el identificador único de una publicación en la que la cuenta de Instagram buscada ha sido etiquetada.

Tabla 11. Información de atributos de la tupla 'DatosEtiquetadasBusquedaUsuario'

5.2.3. Almacenamiento de publicaciones de Instagram buscadas

Para que la plataforma sea más eficiente y no tenga que estar realizando peticiones a Instagram siempre que se quiera buscar una publicación de Instagram que recientemente ha sido buscada, se almacena la información más importante de la publicación de Instagram buscada en la base de datos. Aun así, como la plataforma debe tener la información lo más actualizada posible, si pasados tres minutos se vuelve a buscar una publicación que está almacenada en base de datos, se busca mediante *scraping* en Instagram y se actualizan los datos de la cuenta en la base de datos. Si observamos la Figura 14 podemos ver las clases, atributos y relaciones de la información que puede contener una cuenta de Instagram buscada.

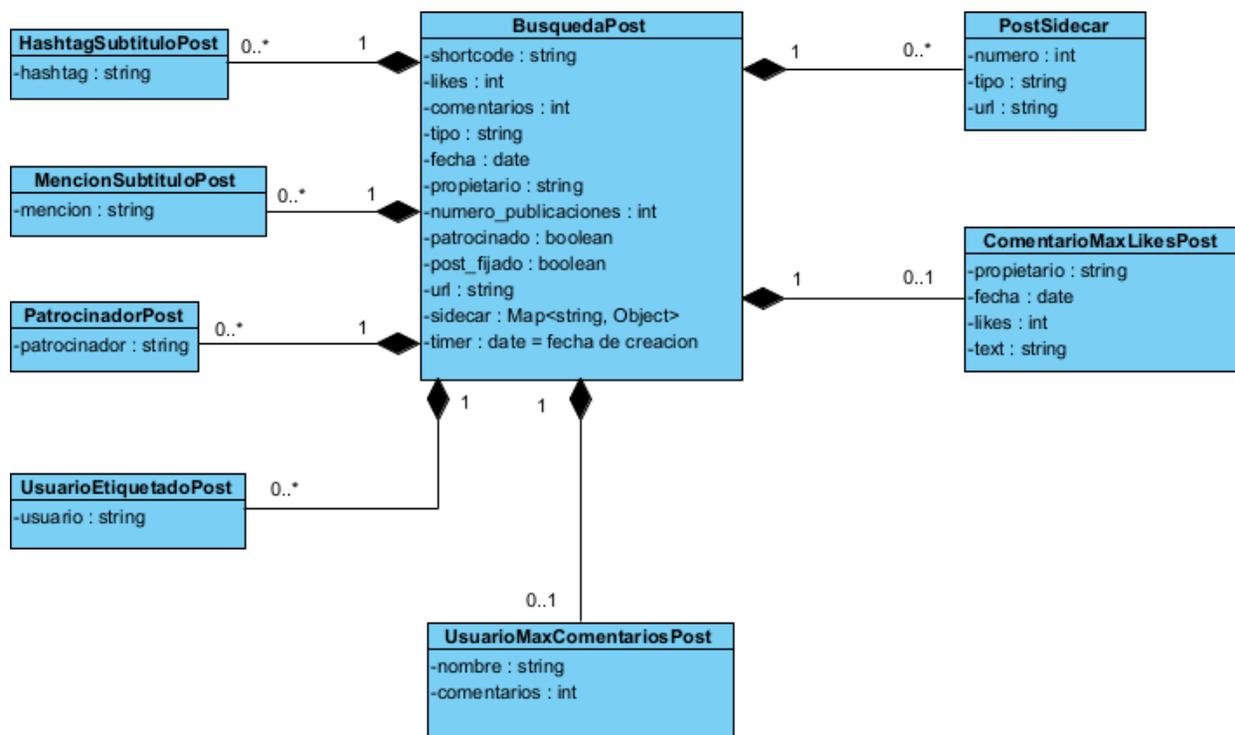


Figura 14. Diagrama de publicación de Instagram buscada

Desarrollo de la propuesta → Diseño de la base de datos

Como se observa una búsqueda de publicación solamente puede tener un usuario que más comenta o ninguno y un comentario con más likes o ninguno, también una búsqueda de publicación puede tener varias imágenes o vídeos si es de tipo 'sidecar', varios *hashtags* en el subtítulo o ninguno, varias menciones en el subtítulo o ninguna, varios patrocinadores o ninguno y varios usuarios etiquetados o ninguno. Si se borra una búsqueda de publicación de la base de datos, también se borra la información de todo lo demás, ya que dependen de esta.

El proceso para definir esta estructura en la base de datos se realiza igual que en el apartado anterior, mediante la declaración de clases, sus respectivos campos y opciones, en el archivo 'models.py' del proyecto (ver concepto teórico *Django*).

• DatosPost:

Clase que representa la información de una publicación de Instagram buscada. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 12 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'DatosPost'.

Campo	Tipo	Descripción
shortcode	CharField(primary_key=True, max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el identificador único de la publicación de Instagram buscada, que es la clave primaria de la tupla en la base de datos.
likes	IntegerField()	Campo de tipo entero. Almacena la cantidad de likes de la publicación de Instagram buscada.
comentarios	IntegerField()	Campo de tipo entero. Almacena la cantidad de likes de la publicación de Instagram buscada.

Desarrollo de la propuesta → Diseño de la base de datos

tipo	CharField(max_length=20)	Campo de tipo cadena con un tamaño máximo de 20 caracteres. Almacena el tipo de publicación (imagen, vídeo o <i>sidecar</i>) de Instagram buscada.
fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación de la publicación de Instagram buscada.
propietario	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de usuario propietario de la publicación de Instagram buscada.
numero_publicaciones	IntegerField()	Campo de tipo entero. Almacena el número de fotos o vídeos de la publicación Instagram buscada. En caso de no ser una publicación de tipo ' <i>sidecar</i> ', el número almacenado siempre será 1.
patrocinado	BooleanField()	Campo de tipo booleano. Indica si la publicación de Instagram buscada es patrocinada por alguna marca.
post_fijado	BooleanField()	Campo de tipo booleano. Indica si la publicación de Instagram buscada esta fijada en alguna cuenta por un usuario.
url	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> de la publicación de Instagram buscada.
numeroVideos	IntegerField()	Campo de tipo entero. Almacena el número de vídeos de la publicación de Instagram buscada.
numeroImágenes	IntegerField()	Campo de tipo entero. Almacena el número de imágenes de la publicación de Instagram buscada.
titulo	CharField(max_length=100, blank=True)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el título de la publicación de Instagram buscada, si no tiene se almacena como cadena vacía.
subtitulo	TextField(blank=True)	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena el subtítulo de la publicación de Instagram buscada, si no tiene se almacena como cadena vacía.
ubicacion	CharField(blank=True, max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de la ubicación

Desarrollo de la propuesta → Diseño de la base de datos

		de la publicación de Instagram buscada, si no tiene se almacena como cadena vacía.
duracion	FloatField()	Campo de tipo número real. Almacena la duración de la publicación de Instagram buscada en caso de que sea un vídeo.
timer	DateTimeField(auto_now_add=True)	Campo de tipo fecha. Almacena la fecha y hora de creación. Este campo es usado a modo de temporizador.

Tabla 12. Información de atributos de la tupla 'DatosPost'

• ComentarioMaxLikesPost:

Clase que representa la información del comentario que tiene más likes de la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'ComentarioMaxLikesPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 13 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'ComentarioMaxLikesPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borra la tupla de 'ComentarioMaxLikesPost' que esté relacionada con la publicación buscada.
propietario	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de usuario

Desarrollo de la propuesta → Diseño de la base de datos

		propietario del comentario con más likes de la publicación de Instagram buscada.
fecha	DateTimeField()	Campo de tipo fecha. Almacena la fecha y hora de creación del comentario con más likes de la publicación de Instagram buscada.
likes	IntegerField()	Campo de tipo entero. Almacena la cantidad de likes del comentario con más likes de la publicación de Instagram buscada.
text	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena el comentario con más likes de la publicación de Instagram buscada.

Tabla 13. Información de atributos de la tupla 'ComentarioMaxLikesPost'

• UsuarioMaxComentariosPost:

Clase que representa la información del usuario que más comentarios tiene de la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'UsuarioMaxComentariosPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 14 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'UsuarioMaxComentariosPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borra la tupla de 'UsuarioMaxComentariosPost' que esté relacionada con la publicación buscada.

Desarrollo de la propuesta → Diseño de la base de datos

nombre	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de usuario propietario que más comentarios tiene en la publicación de Instagram buscada.
comentarios	IntegerField()	Campo de tipo entero. Almacena la cantidad de comentarios del usuario que más comentarios tiene en la publicación de Instagram buscada.

Tabla 14. Información de atributos de la tupla 'UsuarioMaxComentariosPost'

• PostSidecar:

Clase que representa la información de una imagen o un vídeo de la publicación de Instagram buscada en caso de que esta sea de tipo 'sidecar'. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'PostSidecar'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la

Tabla 15 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'PostSidecar'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borran las tuplas de 'PostSidecar' que estén relacionadas con la publicación buscada.
numero	IntegerField()	Campo de tipo entero. Almacena el número en orden de la imagen o vídeo de la publicación tipo 'sidecar'.

Desarrollo de la propuesta → Diseño de la base de datos

tipo	CharField(max_length=20)	Campo de tipo cadena con un tamaño máximo de 20 caracteres. Almacena el tipo de objeto (imagen, vídeo) de la publicación de tipo 'sidecar'.
url	TextField()	Campo de tipo texto, es decir, una cadena sin tamaño límite. Almacena la <i>url</i> de la publicación de tipo 'sidecar'.

Tabla 15. Información de atributos de la tupla 'PostSidecar'

• HashtagsSubtituloPost:

Clase que representa la información de un hashtag que se encuentre en el subtítulo de la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'HashtagsSubtituloPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 16 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'HashtagsSubtituloPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borran las tuplas de 'HashtagsSubtituloPost' que estén relacionadas con la publicación buscada.
hashtag	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena un hashtag que se encuentre en el subtítulo de la publicación de Instagram buscada.

Tabla 16. Información de atributos de la tupla 'HashtagsSubtituloPost'

- MencionesSubtituloPost:

Clase que representa la información de una mención que se encuentre en el subtítulo de la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'MencionesSubtituloPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 17 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'MencionesSubtituloPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borran las tuplas de 'MencionesSubtituloPost' que estén relacionadas con la publicación buscada.
mencion	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena un nombre de usuario que se encuentre en el subtítulo de la publicación de Instagram buscada.

Tabla 17. Información de atributos de la tupla 'MencionesSubtituloPost'

- PatrocinadoresPost:

Clase que representa la información de un patrocinador de la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'PatrocinadoresPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 18 podemos observar tanto los campos como el tipo de dato *Django* y una

Desarrollo de la propuesta → Diseño de la base de datos

descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'PatrocinadoresPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borran las tuplas de 'PatrocinadoresPost' que estén relacionadas con la publicación buscada.
mencion	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena un patrocinador de la publicación de Instagram buscada.

Tabla 18. Información de atributos de la tupla 'PatrocinadoresPost'

• UsuariosEtiquetadosPost:

Clase que representa la información de un usuario etiquetado en la publicación de Instagram buscada. El campo 'shortcode' de esta clase crea una relación de un objeto 'DatosPost' a muchos objetos 'UsuariosEtiquetadosPost'. Crea una tupla en la base de datos y sus campos son los atributos de la tupla. En la Tabla 19 podemos observar tanto los campos como el tipo de dato *Django* y una descripción del dato que finalmente será almacenado en la base de datos dentro de la tupla 'UsuariosEtiquetadosPost'.

Campo	Tipo	Descripción
shortcode	ForeignKey(DatosPost, on_delete=models.CASCADE)	Campo que representa una clave externa hacia la tupla 'DatosPost'. Almacena el identificador único de

Desarrollo de la propuesta → Diseño de la base de datos

		la publicación buscada. Con la opción 'on_delete=models.CASCADE' si se elimina la publicación buscada se borran las tuplas de 'UsuariosEtiquetadosPost' que estén relacionadas con la publicación buscada.
usuario	CharField(max_length=100)	Campo de tipo cadena con un tamaño máximo de 100 caracteres. Almacena el nombre de un usuario etiquetado en la publicación de Instagram buscada.

Tabla 19. Información de atributos de la tupla 'UsuariosEtiquetadosPost'

5.3. Proceso de obtención de datos de Instagram

En esta parte nos centraremos en explicar y resaltar el proceso para obtener los datos de Instagram. Cabe resaltar que para el desarrollo de este proyecto se ha hecho uso del *framework Django* (ver concepto teórico *Django*), el cual utiliza como lenguaje de programación base *Python* (ver concepto teórico *Python*). Una vez sabido esto podemos empezar a explicar el desarrollo para poder recopilar datos de Instagram.

En un principio para obtener los datos de Instagram se usaba un paquete de *Python* dedicado a obtener datos de Instagram llamado *instagrammy* (Yogeshwaran, 2022). Este paquete mediante la realización de *web scraping* en Instagram obtiene información de las cuentas de Instagram, datos de las publicaciones, *hashtags* y datos de ubicaciones, sin dependencias externas.

Este paquete funcionaba bien al comienzo de este proyecto, permitiendo obtener bastantes datos de usuarios de Instagram y *hashtags*, pero pasado un cierto tiempo Instagram rediseño la forma en la que se pasaba la información a las páginas web cuando se hacía una petición para obtener una publicación, cuenta o *hashtags*, entonces este paquete dejó de funcionar correctamente por la falta de actualización.

Finalmente, se optó por usar otro paquete de *Python* dedicado a obtener datos de Instagram llamado *instaloder* (Instaloder, 2022), ya que este paquete es actualizado constantemente, casi cada mes para adaptarse a los cambios que pudiera hacer Instagram. Este paquete permite obtener datos de cuentas de Instagram, *hashtags*, publicaciones y más cosas, además, funciona mediante *web scraping* (ver concepto teórico **¡Error! No se encuentra el origen de la referencia.**) realizando peticiones a Instagram gracias al lenguaje de consulta *GraphQL* (ver concepto teórico *GraphQL*).

Cabe destacar que dentro del archivo del proyecto llamado 'instaloder_funciones.py' se han implementado todas las funciones de *instaloder* utilizadas para obtener datos de Instagram, así

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

como el cálculo de nuevos datos a través de los obtenidos. También, cabe destacar que en la plataforma finalmente no se llegó a introducir la búsqueda de *hashtags* por el hecho de que cuando se realizaba no funcionaba correctamente.

5.3.1. Cuenta de Instagram para recopilar datos

En este apartado vamos a explicar qué proceso y funciones realiza la aplicación para iniciar sesión en una cuenta de Instagram y guardar esta sesión en el sistema para posteriormente usar esta cuenta a la hora de recopilar datos de Instagram.

- Iniciar y guardar sesión en el sistema:

El proceso de iniciar sesión en una cuenta de Instagram desde *instaloader*, no siempre es posible para todas las cuentas, ya que Instagram muchas veces no deja iniciar sesión desde dispositivos poco conocidos. Obviando este caso, para realizar este proceso necesitamos el nombre de usuario y la contraseña de la cuenta de Instagram, asique para poder entender el funcionamiento de este proceso vamos a observar la Tabla 20, donde aparecen las funciones de *instaloader* que se utilizan y el orden en el que se realizan.

Paso	Función	Descripción
1º	L = Instaloader()	Instancia un objeto de tipo ' <i>Instaloader</i> ' en la variable 'L'.
2º	L.login(USER,PASS)	Inicia sesión en la cuenta con el nombre de usuario 'USER' y contraseña 'PASS', y guarda esta sesión en la variable 'L'. Si no se puede iniciar sesión devuelve error.
3º	L.save_session_to_file()	Guarda la sesión iniciada en el lugar predefinido para <i>instaloader</i> dentro del sistema operativo.

Tabla 20. Funciones de *instaloader* para iniciar y guardar sesión

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

Por el hecho de que no es bueno estar continuamente iniciando sesión en Instagram guardamos en el sistema la sesión iniciada, ya que si iniciamos sesión muchas veces Instagram puede llegar a bloquearnos el acceso desde la IP.

• Cargar sesión guardada en el sistema:

Una vez que ya hemos iniciado y guardado la sesión en el sistema (proceso que solo hace falta realizarlo una única vez para una cuenta de Instagram), si se necesita la sesión para recopilar datos de Instagram, se carga la sesión de a cuenta de Instagram en concreto que se encuentra guardada en el sistema. En la Tabla 21 podemos observar este proceso con las funciones de *instaloader* que se utilizan y el orden en el que se realizan.

Paso	Función	Descripción
1º	L = Instaloader()	Instancia un objeto de tipo ' <i>Instaloader</i> ' en la variable 'L'.
2º	L.load_session_from_file(USER)	Carga en la variable 'L' la sesión de Instagram guardada en el sistema operativo con el nombre de usuario 'USER'. Si no se encuentra da error.

Tabla 21. Funciones de *instaloader* para cargar una sesión guardada en el sistema

La mayoría de las veces que la plataforma quiera una sesión de una cuenta de Instagram realizará el proceso de la Tabla 21, realizando una sola vez por cada cuenta nueva que se introduzca en el sistema el proceso de la Tabla 20.

Una vez sabido esto, cabe destacar que el sistema emplea una cuenta de Instagram por defecto para recopilar datos de Instagram. Esta cuenta de Instagram tiene el nombre de usuario 'instaanalysisifg' y ha sido creada para poder realizar este proyecto.

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

Un usuario de la plataforma puede añadir una cuenta de Instagram para que el sistema recopile información a través de esa cuenta en vez desde la cuenta por defecto. La sesión de la cuenta se guardará en el sistema operativo y los datos para buscar esa cuenta, usarla o no usarla se guardan en la base de datos en la tupla de 'CuentaScrapeoInstagram' (en la Tabla 2 se pueden observar los atributos que contiene esta tupla).

5.3.2. Obtención de datos de cuentas de Instagram

En este apartado nos centraremos en explicar cómo se obtienen los datos de una cuenta de Instagram buscada por el usuario. Cabe destacar que dependiendo de la sesión de Instagram que se use, se obtienen más datos o menos, en función si la cuenta es privada o no para la sesión de Instagram que se use.

- Listar cuentas:

Proceso por el cual la plataforma devuelve una lista con las cuentas con un nombre de usuario parecido a la cadena introducida. Si observamos la Tabla 22 podemos observar la función de *instaloader* que realiza este proceso y su respectiva descripción.

Función	Descripción
TopSearchResults(L.context,CADENA).get_profiles()	Se usa para recuperar una lista con el nombre de usuario parecido a la cadena introducida 'CADENA'. La letra 'L' representa a la variable donde se encuentra cargada la sesión de Instagram utilizada para recopilar los datos.

Tabla 22. Funciones de *instaloader* para obtener lista de cuentas

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

- Obtener datos cuenta:

Proceso por el cual se obtienen y calculan datos de una cuenta de Instagram en concreto, así como sus publicaciones más recientes, vídeos más recientes y publicaciones más recientes en la que la cuenta buscada se encuentre etiquetada. En la Tabla 23 podemos observar las funciones de *instaloader* que se utilizan para realizar este proceso.

Función	Descripción
Profile.from_username(L.context, CUENTA)	Se usa para recuperar gran cantidad de datos acerca de la cuenta con el nombre 'CUENTA'. La letra 'L' representa a la variable donde se encuentra cargada la sesión de Instagram utilizada para recopilar los datos.
cuenta.get_posts()	Se usa para recuperar una lista con los datos de las publicaciones de una cuenta almacenada en la variable 'cuenta'.
cuenta.get_igtv_posts()	Se usa para recuperar una lista con los datos de los vídeos de una cuenta almacenada en la variable 'cuenta'.
profile.get_tagged_posts()	Se usa para recuperar una lista con los datos de las publicaciones en las que la cuenta almacenada en la variable 'cuenta' ha sido etiquetada.

Tabla 23. Funciones de *instaloader* para obtener datos de la cuenta buscada

Además de obtener estos datos de la cuenta de Instagram buscada, la plataforma también calcula datos nuevos a través de los obtenidos. La información más relevante de los datos obtenidos y los nuevos datos calculados se almacenan en base de datos (ver Almacenamiento de cuentas de Instagram buscadas).

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

- Obtener datos historias destacadas de una cuenta:

Proceso por el cual se obtienen y calculan datos de las historias destacadas que puede tener una cuenta de Instagram buscada. En la Tabla 24 podemos observar las funciones de *instaloader* que se utilizan para realizar este proceso.

Función	Descripción
L.get_highlights(IDCUENTA)	Se usa par recuperar datos sobre la lista de conjuntos de historias destacadas que posee una cuenta con el identificador único 'IDCUENTA'. La letra 'L' representa a la varibale donde se encuentra cargada la sesión de Instagram utiizada para recopilar los datos.
highlights.get_items()	Se usa para recuperar una lista con los datos de las historias de un conjunto de historias destacadas almacenado en la variable 'cuenta'.

Tabla 24. Funciones de *instaloader* para obtener datos de historias destacadas

Además de obtener estos datos de las historias destacadas de la cuenta de Instagram buscada, la plataforma también calcula datos nuevos a través de los obtenidos. La información más relevante de los datos obtenidos y los nuevos datos calculados se almacenan en base de datos (ver Almacenamiento de cuentas de Instagram buscadas).

5.3.3. Obtención de datos de publicaciones de Instagram

En este apartado nos centraremos en explicar cómo se obtienen los datos de una publicación de Instagram buscada por el usuario. Cabe destacar que dependiendo de la sesión de Instagram que se use, se obtienen más datos o menos, en función si la cuenta es privada o no para la sesión de Instagram que se use.

Desarrollo de la propuesta → Proceso de obtención de datos de Instagram

En la Tabla 25 podemos observar las funciones de *instaloader* que se utilizan para realizar el proceso de obtención de datos de una publicación de Instagram.

Función	Descripción
Post.from_shortcode(L.context, SHORTCODE)	Se usa para recuperar gran cantidad de datos acerca de la publicación con el identificador único 'SHORTCODE'. La letra 'L' representa a la variable donde se encuentra cargada la sesión de Instagram utilizada para recopilar los datos.
publicacion.get_sidecar_nodes()	Se usa para recuperar una lista con los datos de las fotos o imágenes de una publicación de tipo 'sidecar' almacenada en la variable 'publicacion'.
publicacion.get_comments()	Se usa para recuperar una lista con los datos de los comentarios de una publicación almacenada en la variable 'publicacion'.

Tabla 25. Funciones de *instaloader* para obtener datos de la publicación buscada

Además de obtener estos datos de la publicación de Instagram buscada, la plataforma también calcula datos nuevos a través de los obtenidos. La información más relevante de los datos obtenidos y los nuevos datos calculados se almacenan en base de datos (ver Almacenamiento de publicaciones de Instagram buscadas).

Desarrollo de la propuesta → Desarrollo de la aplicación web

5.3.4. Desarrollo de la aplicación web

En esta parte nos centraremos en explicar y resaltar el proceso de desarrollo de la aplicación web para que esta permita realizar las funcionalidades descritas anteriormente. El desarrollo de la aplicación se ha hecho en el editor de código fuente *Visual Studio Code* (ver herramienta *Visual Studio Code*) manteniendo un control de versiones mediante el software de control de versiones *Git* (ver tecnología *Git*) y usando el servicio basado en la nube *GitHub* (ver herramienta *GitHub*) para guardar todo el proyecto y sus respectivas versiones.

Por el hecho de que la aplicación está desarrollada en el *framework Django* (ver concepto teórico *Django*), el patrón usado para el desarrollo de la aplicación es Modelo Vista Plantilla (*MVT*).

El Modelo, es la parte que se encarga de comunicarse con la base de datos, está formado por las clases explicadas anteriormente en el apartado donde se explica el diseño que tiene la base de datos (ver Diseño de la base de datos).

La Vista se encarga de comunicarse con el archivo 'instaloader_funciones.py' para obtener datos de una cuenta o publicación de Instagram para después utilizar el Modelo para introducir estos datos en la base de datos y por último mostrar estos datos al usuario mediante la Plantilla. También se encarga de comunicarse con el modelo para introducir o pedir ciertos datos de la base de datos. Además, es la encargada de mostrar todas las otras plantillas de la aplicación que no se encuentren relacionadas con la obtención de datos de Instagram.

Las Plantillas, en este proyecto, son archivos escritos en el lenguaje de marcado *HTML* (ver concepto teórico *HTML*), estilizados mediante la biblioteca para el diseño de sitios y aplicaciones web *Bootstrap* (ver herramienta *Bootstrap*). Estos archivos también tienen partes escritas en el lenguaje de programación interpretado *JavaScript* (ver lenguaje *JavaScript*) para permitir crear páginas web dinámicas, en nuestro caso lo hemos utilizado para poder ordenar las tablas por

Desarrollo de la propuesta → Desarrollo de la aplicación web

columnas y para crear gráficos según los datos obtenidos. Además, el proyecto también dispone de una hoja de estilo (ver concepto teórico CSS).

Sabiendo todo esto, la parte del *backend* de la aplicación web (ver concepto teórico Desarrollo *Backend*) se corresponde con todos los archivos relacionados con el modelo y todos los archivos relacionados con la vista, y la parte del *frontend* de la aplicación web (ver concepto teórico Desarrollo *Frontend*) se corresponde con todos los archivos relacionados con las plantillas.

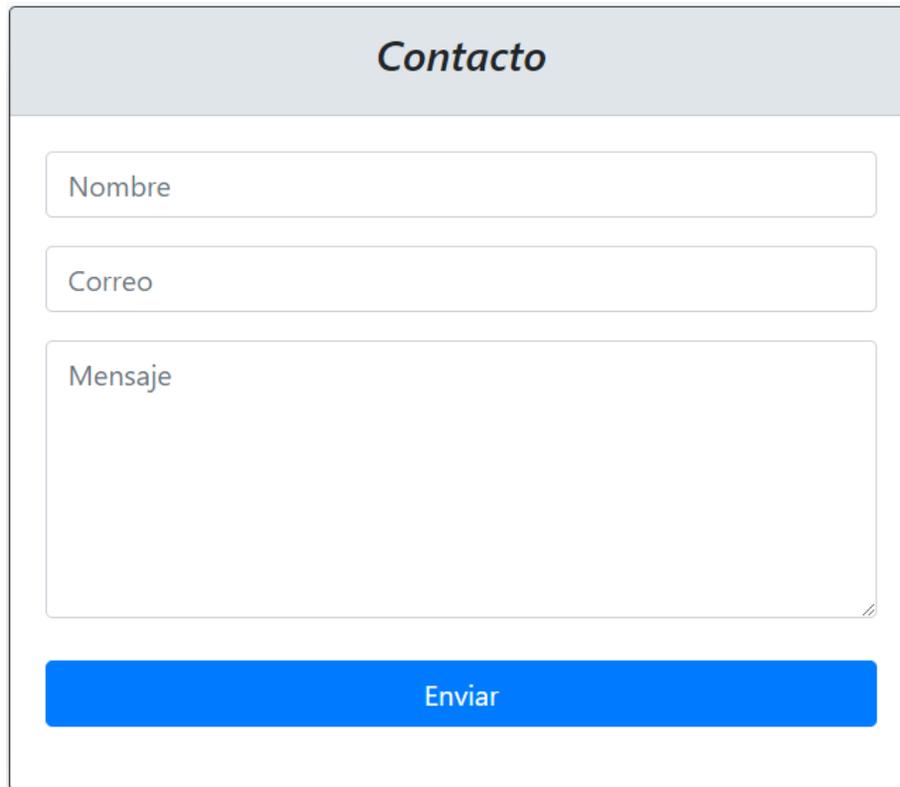
Por último, para entender que direcciones tiene disponible la aplicación web y con respecto a estas que funciones se realizan tanto en el modelo y la vista, y como se representa la información en las plantillas vamos a ir dirección por dirección mostrando que hace y que elementos de la interfaz gráfica son relevantes.

- '*analisisInsta*':

La vista renderiza la plantilla de la pantalla principal.

- '*analisisInsta/contacto*':

La vista renderiza la plantilla de escribir contacto, que tiene un formulario tal y como se ve en Figura 15. Cuando el usuario rellena el formulario y lo envía, si es correcto la vista ordena al modelo que introduzca estos datos en la base de datos.



El formulario de contacto está contenido en un contenedor con un encabezado gris que dice "Contacto". Debajo del encabezado hay tres campos de entrada: un campo de texto para "Nombre", un campo de texto para "Correo" y un campo de texto más grande para "Mensaje". En la parte inferior del contenedor hay un botón azul con el texto "Enviar".

Figura 15. Formulario de contacto

- ' [analisisInsta/buscadorCuenta/](#)':

La vista renderiza la plantilla de buscar cuenta, que tiene un formulario de búsqueda tal y como se ve en Figura 16. Cuando el usuario rellena el formulario y realiza la búsqueda, si es correcto la vista llama a la función correspondiente del archivo de 'instaloader_funciones.py' y renderiza la plantilla de la lista de cuentas encontradas tal y como se ve en la Figura 17, si no encuentra ninguna cuenta muestra un mensaje de cuenta no encontrada.



Figura 16. Formulario del buscador

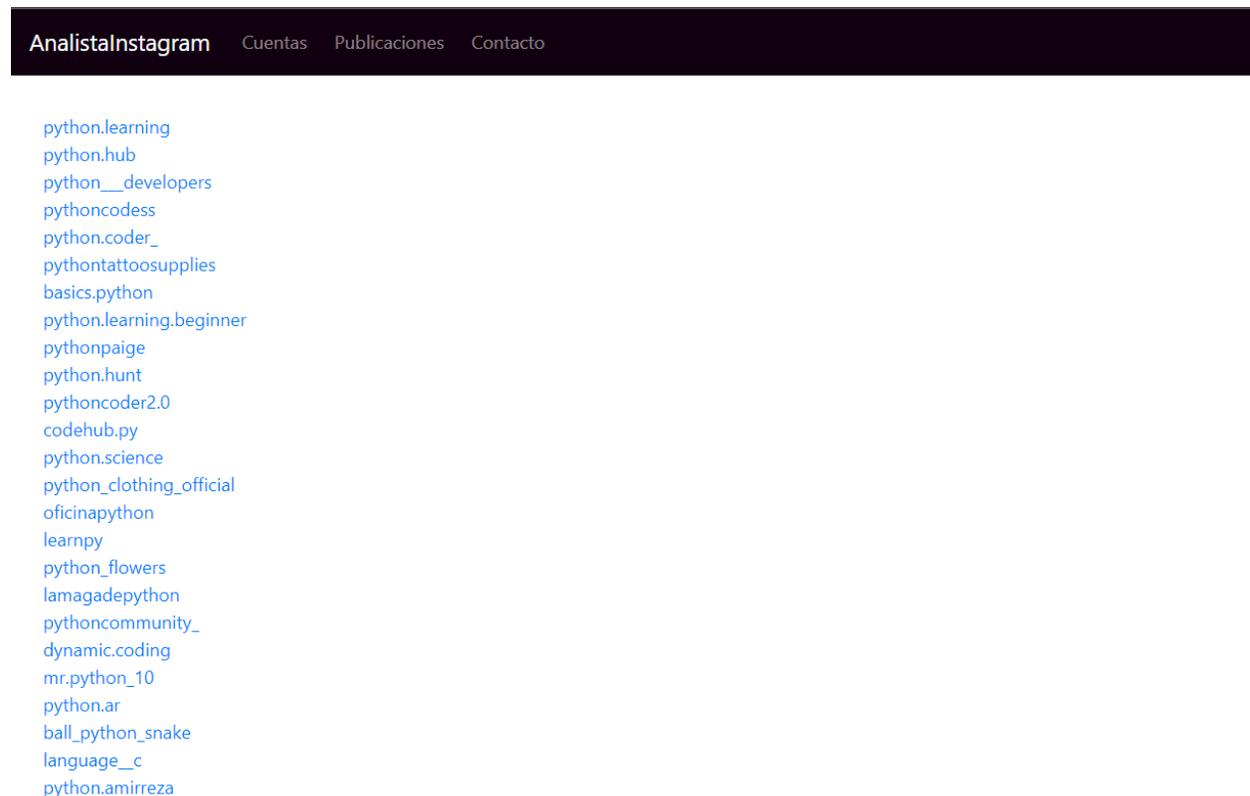


Figura 17. Lista de cuentas buscadas

• '[analisiInsta/buscadorCuenta/busqueda/<IDusuario>/'](#):

La vista comprueba si existe la cuenta a mostrar en la base de datos, si existe comprueba que el temporizador de esta no haya superado los tres minutos y recupera los datos de la base de datos.

Si no existe la cuenta en base de datos o el temporizador ha superado los tres minutos la vista

Desarrollo de la propuesta → Desarrollo de la aplicación web

llama a las funciones correspondientes del archivo 'instaloder_funciones.py' para obtener los datos de la cuenta '<IDusuario>' y almacena estos datos en la base de datos.

Por último, con los datos recibidos renderiza la plantilla de la obtención de datos de la cuenta de Instagram buscada tal y como se muestra en la Figura 18.

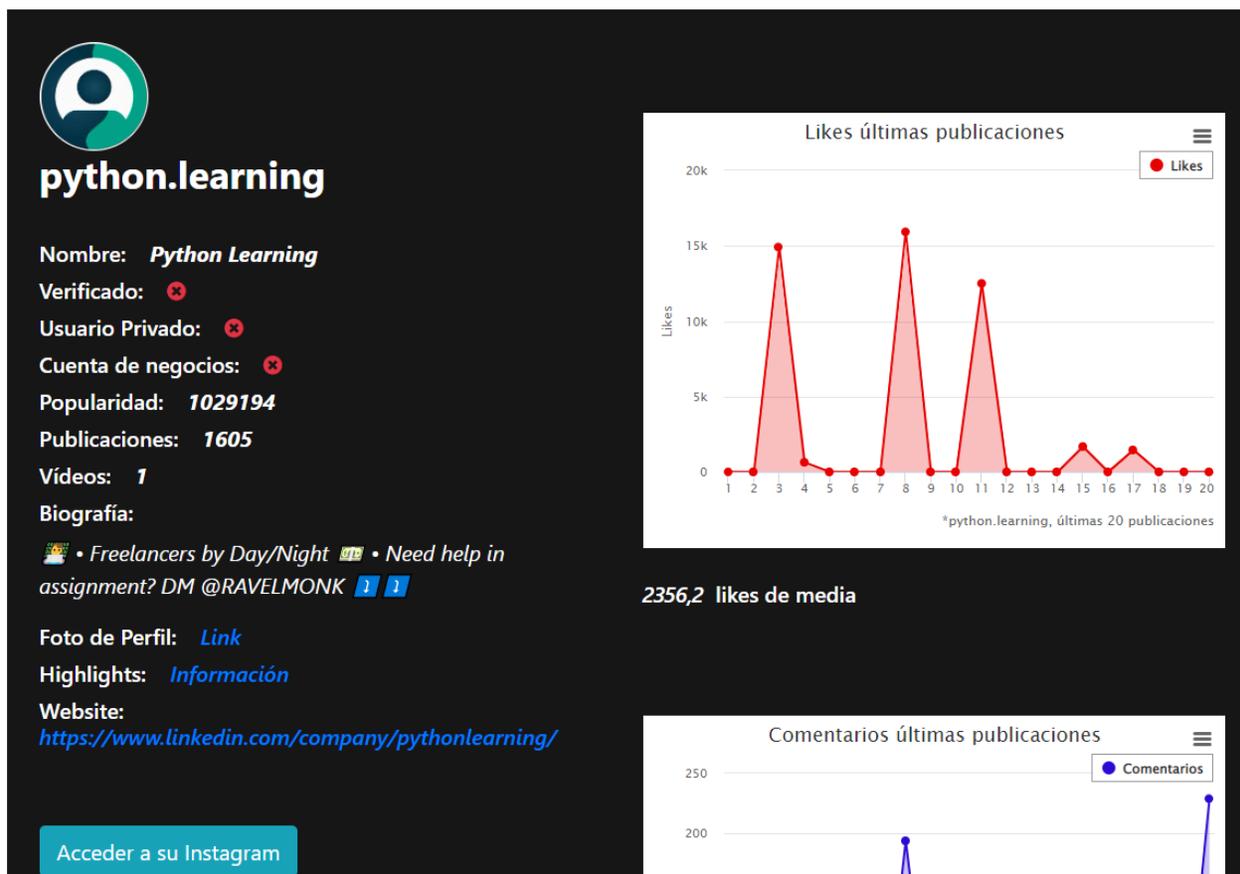


Figura 18. Información de la cuenta buscada

En la Figura 19 podemos observar los tipos de gráficos que presenta la plantilla, de forma que hay un gráfico que compara cuantos vídeos, imágenes y sidecar hay en las últimas publicaciones,

Desarrollo de la propuesta → Desarrollo de la aplicación web

vídeos y publicaciones en las que la cuenta buscada sale etiquetada, además también posee gráficos para comparar likes, comentarios y reproducciones en el caso de los vídeos.

En la Figura 20 podemos observar los tipos de tablas que presenta la plantilla, de forma que una tabla es para publicaciones en general y otra es para vídeos. Además, estas tablas se pueden ordenar numéricamente, por fecha o alfabéticamente pulsando el título de la columna de la tabla.

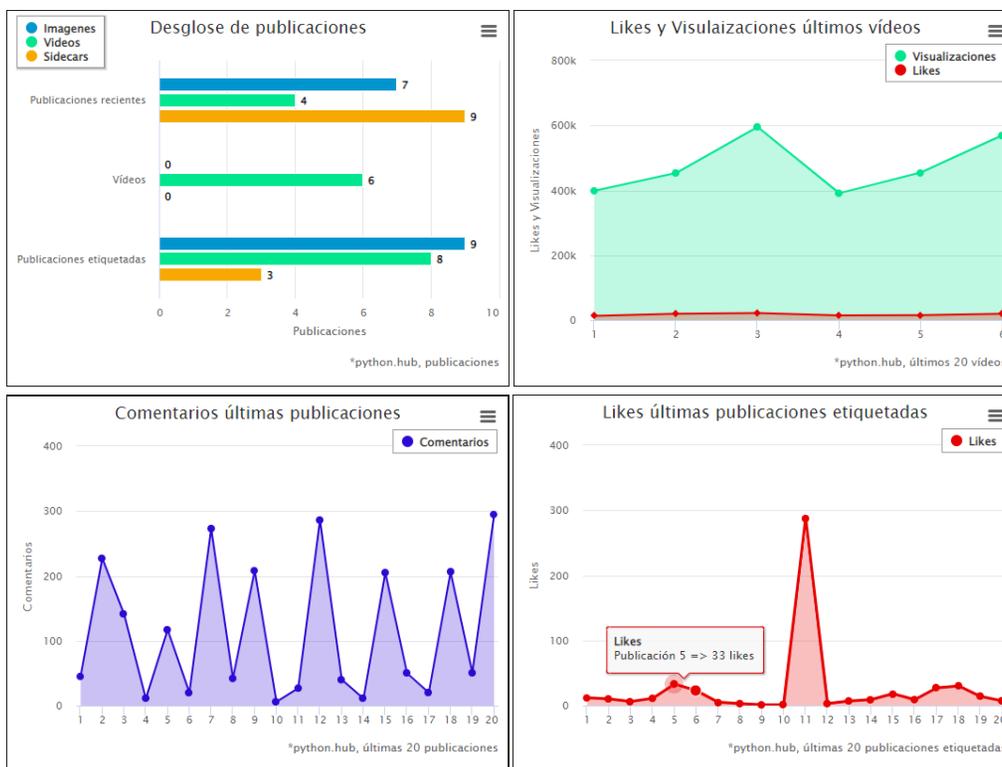


Figura 19. Gráficos cuenta buscada

Desarrollo de la propuesta → Desarrollo de la aplicación web

Likes	Comentarios	Tipo	Fecha	Buscar	Likes	Comentarios	Visualizaciones	Fecha	Buscar
8992	45	Sidecar	19/08/2022	🔍	13220	44	398713	29/07/2022	🔍
19453	227	Sidecar	12/06/2022	🔍	19145	141	453260	23/07/2022	🔍
0	141	Sidecar	04/04/2022	🔍	21748	88	595588	20/07/2022	🔍
3585	11	Sidecar	27/08/2022	🔍	14788	40	391011	14/07/2022	🔍
2005	117	Imagen	27/08/2022	🔍	14915	65	454814	08/07/2022	🔍
6264	20	Sidecar	27/08/2022	🔍	19695	44	569647	08/07/2022	🔍

Figura 20. Tipos de tablas cuenta buscada

- ' [analisisInsta/buscadorCuenta/busqueda/highlights/<IDusuario>/'](#):

La vista comprueba si existen las historias destacadas a mostrar de la cuenta buscada en la base de datos, si existe recupera los datos de las historias destacadas de la cuenta buscada de la base de datos. Si no existen en base de datos la vista llama a las funciones correspondientes del archivo 'instaloader_funciones.py' para obtener los datos de las historias destacadas de la cuenta '<IDusuario>' y almacena estos datos en la base de datos.

Por último, con los datos recibidos la vista renderiza la plantilla de obtención de datos de las historias destacadas de la cuenta de Instagram buscada tal y como se muestra en la Figura 21. La plantilla presenta un gráfico para comparar la repartición de historias totales en las historias destacadas y otro gráfico para comparar la cantidad de imágenes y vídeos que tienen las historias destacadas. También presenta tablas que se pueden ordenar numéricamente, por fecha o alfabéticamente pulsando el título de la columna de la tabla.

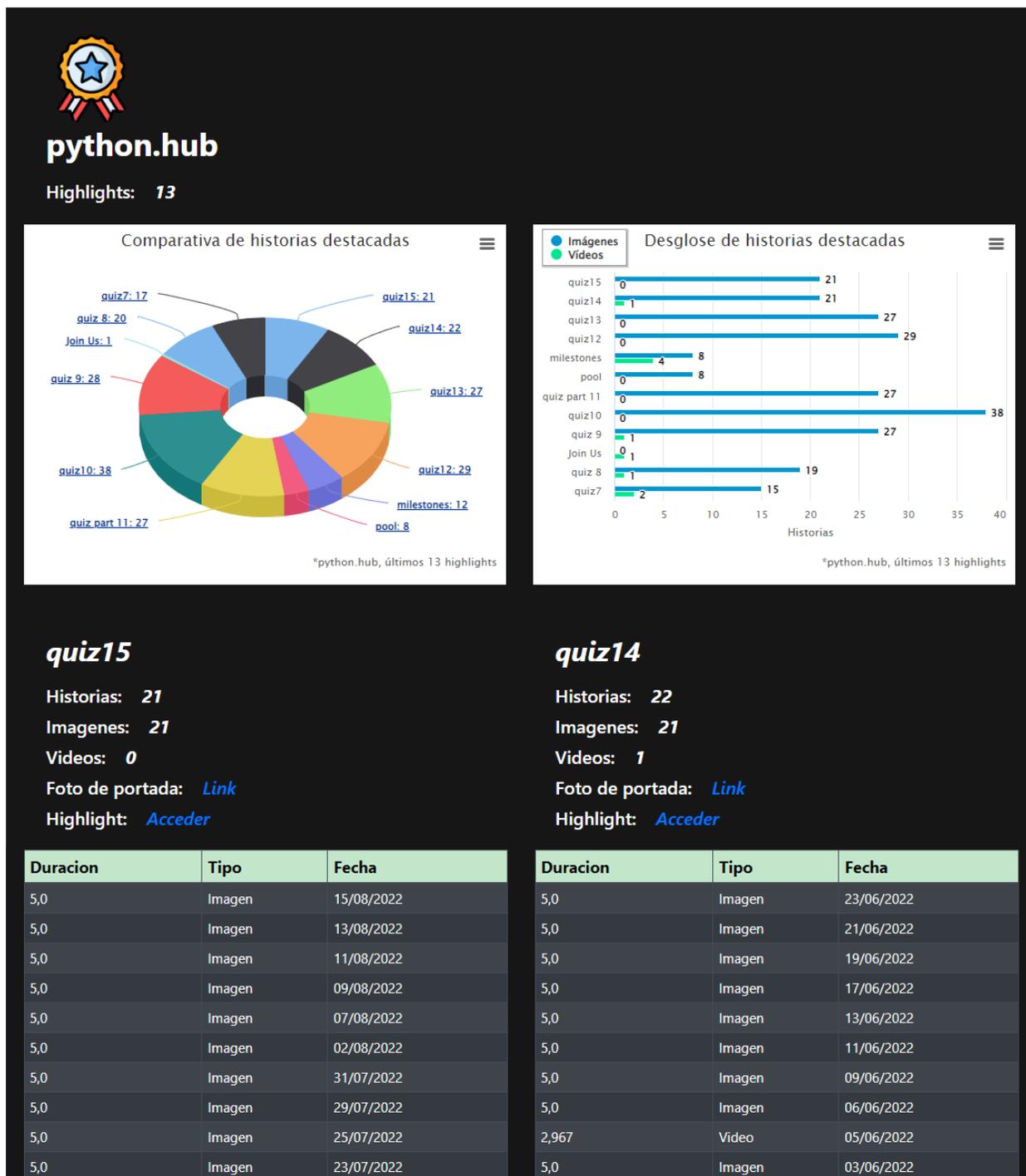


Figura 21. Información de las historias destacadas de una cuenta buscada

Desarrollo de la propuesta → Desarrollo de la aplicación web

- ' [analisisInsta/buscadorPost/](#):

La vista renderiza la plantilla de buscar publicación, que tiene un formulario de búsqueda tal y como se ve en Figura 16. Cuando el usuario rellena el formulario con el identificador único de la publicación y realiza la búsqueda, si el formulario es correcto la vista llama a la función correspondiente del archivo de 'instaloader_funciones.py' y renderiza la plantilla con un enlace que es el título resumido de la publicación buscada como se ve en la Figura 22, si no encuentra ninguna publicación muestra un mensaje de publicación no encontrada.

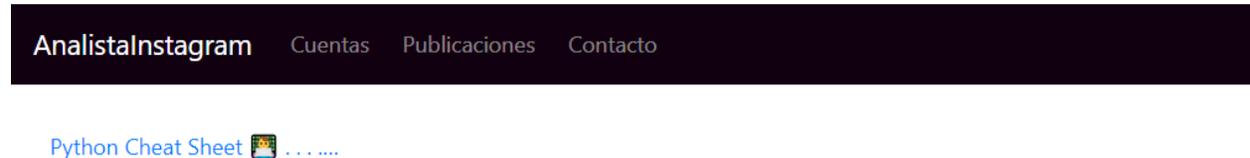


Figura 22. Publicación buscada

Desarrollo de la propuesta → Desarrollo de la aplicación web

- ' [analisisInsta/buscadorPost/busqueda/<IdentificadorPost>/'](#):

La vista comprueba si existe la publicación a mostrar en la base de datos, si existe comprueba que el temporizador de esta no haya superado los tres minutos y recupera los datos de la base de datos. Si no existe la publicación en base de datos o el temporizador ha superado los tres minutos la vista llama a las funciones correspondientes del archivo 'instaloader_funciones.py' para obtener los datos de la publicación '<IdentificadorPost >' y almacena estos datos en la base de datos.

Por último, con los datos recibidos renderiza la plantilla de obtención de datos de la publicación de Instagram buscada tal y como se muestra en la Figura 23. Si la publicación es de tipo '*sidecar*' la plantilla presenta una tabla y un gráfico para comparar las imágenes y vídeos en caso de que se trate de una publicación de tipo *sidecar*. También presenta más tablas sobre los *hashtags*, menciones, patrocinadores y usuarios etiquetados (si la publicación contiene estos elementos) que se pueden ordenar numéricamente o alfabéticamente. Además, si la publicación contiene comentarios presenta el usuario que más comenta y el comentario con más likes (puede darse el caso de que la publicación tenga comentarios, pero esta información no aparezca por el hecho que ha superado un periodo de 30 segundos, como límite para realizar el cálculo de estos datos).



Generate Text from videos using Python

Propietario: [python.hub](#)

Likes: **5210**

Comentarios: **7**

Tipo: **Sidecar**

Publicación patrocinada: ✖

Publicación fijada: ✖

Fecha: **25/08/2022**

Shortcode: **ChrGXACLXsS**

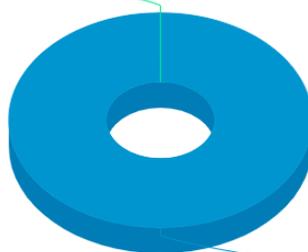
Subtítulo:

Generate Text from videos using Python . Get Python handwritten notes, visit link in bio . Visit our site for free project source codes-- copyassignment.com . . . Turn on post notifications for more such posts like this . . . Follow @python.hub for more content on computer science, programming, technology, and Python language #developer #development #coder #coding #pythonhub

Ver publicación

Comparativa de publicación Sidecar

Videos: 0



Imágenes: 3

*python.hub, 3 imagenes y videos

Fotos y videos

Publicación	Tipo	Ver
1	Imagen	🔗
2	Imagen	🔗
3	Imagen	🔗

Comentario con más likes

Usuario	Likes	Fecha	Comentario
sakshammmmm_..	3	25/08/2022	If you can so many codes why don't you join a multinational like Google, Microsoft etc. ?

Usuario que mas comenta

Usuario	Comentarios
sakshammmmm_..	2

Hashtags en subtítulo

Hashtag
#developer
#development
#coder
#coding
#pythonhub

Menciones en subtítulo

Usuario	Buscar
python.hub	🔍

Figura 23. Información de la publicación buscada

Desarrollo de la propuesta → Desarrollo de la aplicación web

- ['analisiInsta/cuentasScraping/'](#):

La vista renderiza la plantilla con una tabla donde aparecen las cuentas de Instagram que ha añadido el usuario para obtener datos tal y como se muestra en la Figura 24. Si no tiene ninguna cuenta de Instagram añadida el usuario no se muestra la tabla.

Cuentas Scraping Instagram

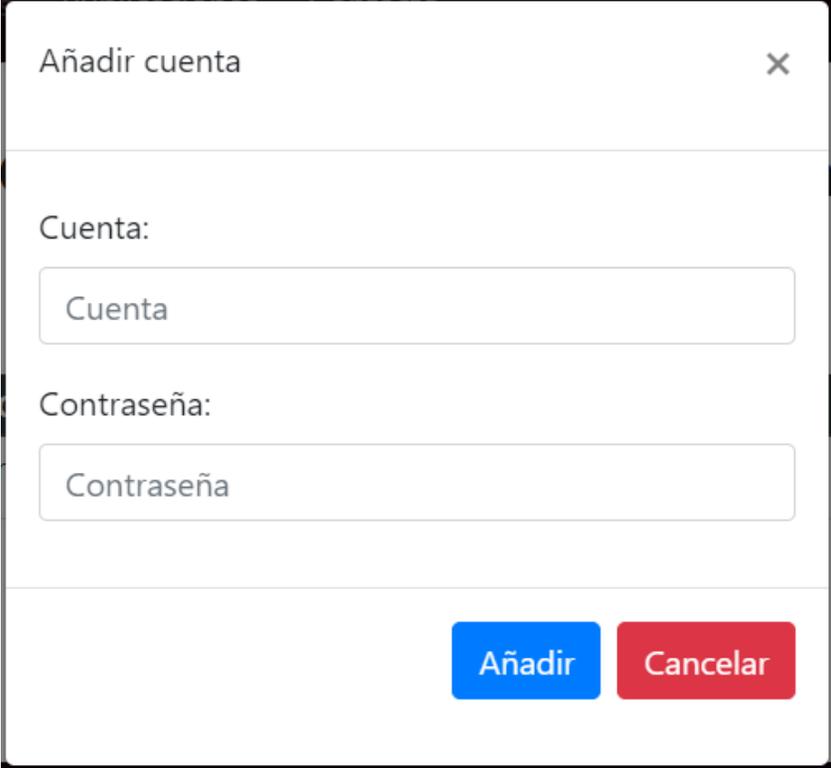
Añadir cuenta +

Cuenta	Usando	Eliminar
macy_guty	<input type="checkbox"/>	

Figura 24. Información cuentas scraping Instagram

- ['analisiInsta/cuentasScraping/crear'](#):

La vista renderiza una plantilla *modal* con un formulario donde se pide el nombre de usuario de la cuenta y la contraseña de Instagram tal y como se muestra en la Figura 25. Cuando el usuario pulsa en añadir si la plataforma consigue iniciar sesión en la cuenta de *scraping* de Instagram se lo indicará al usuario y guarda el nombre de usuario de Instagram en la base de datos.



Añadir cuenta

Cuenta:

Contraseña:

Añadir Cancelar

Figura 25. Modal añadir cuenta scraping Instagram

- ' [analisisInsta/cuentasScraping/usar/<int:pk>](#)':

La vista renderiza una plantilla *modal* con un formulario con un *checkbox* donde para indicar si se desea usar la cuenta de Instagram con el identificador '`<int:pk>`' para obtener datos tal y como se muestra en la Figura 26. Si el usuario selecciona en el *checkbox* y pulsa en confirmar, se actualiza a 'True' el valor de la variable de tipo booleano de la cuenta de *scraping* de Instagram de la base de datos.



Figura 26. Modal usar cuenta scraping Instagram

- ' [analisisInsta/cuentasScraping/eliminar/<int:pk>](#)':

La vista renderiza una plantilla *modal* con un formulario para eliminar la cuenta de Instagram con el identificador '`<int:pk>`' tal y como se muestra en la Figura 27. Si el usuario pulsa en borrar, se borra la cuenta de *scraping* de Instagram de la base de datos.

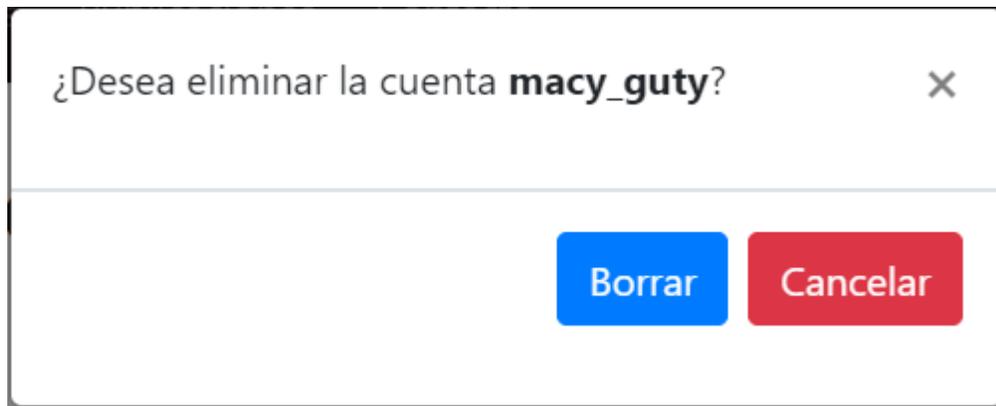


Figura 27. Modal eliminar cuenta scraping Instagram

- 'login/':

La vista renderiza una plantilla donde aparece un formulario donde se pide el nombre de usuario y la contraseña para iniciar sesión tal y como se muestra en la Figura 28. Si el usuario pulsa en iniciar sesión, se inicia la sesión en la cuenta introducida.

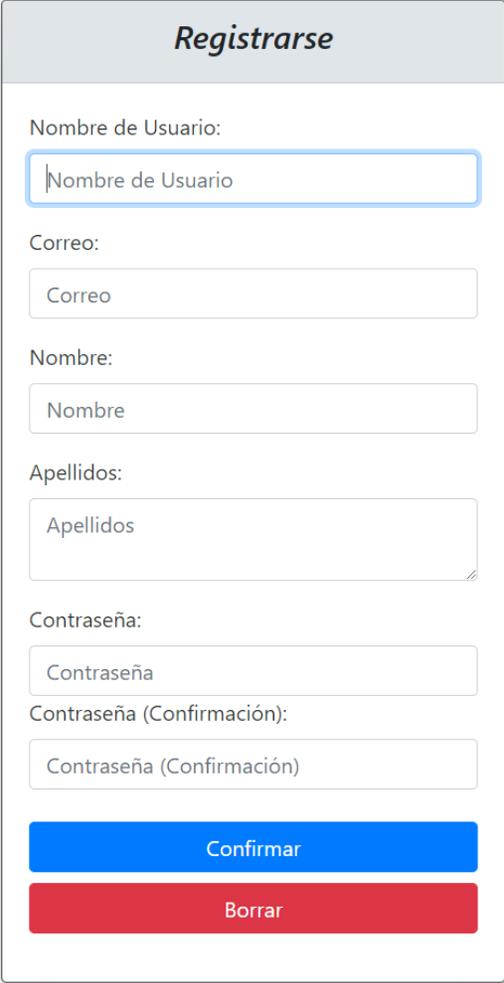
A login form titled "Iniciar Sesión" in a bold, italicized font. The form has a light grey header. Below the header, there are two input fields: "Nombre de Usuario:" followed by a text input field containing "Nombre de Usuario", and "Contraseña:" followed by a text input field containing "Contraseña". Below the input fields is a blue button with the text "Iniciar Sesión". At the bottom of the form, there is a link: "Todavía no tienes cuenta. [Registrar cuenta](#)".

Figura 28. Iniciar sesión

Desarrollo de la propuesta → Desarrollo de la aplicación web

- 'registro/':

La vista renderiza una plantilla donde aparece un formulario para rellenar los datos y registrar una cuenta de usuario tal y como se muestra en la Figura 29. Si el usuario pulsa en confirmar y el formulario es correcto, se añade en la base de datos el nuevo usuario.



El formulario de registro, titulado "Registrarse", contiene los siguientes campos de entrada:

- Nombre de Usuario:
- Correo:
- Nombre:
- Apellidos:
- Contraseña:
- Contraseña (Confirmación):

Debajo de los campos se encuentran dos botones de acción:

- Un botón azul con el texto "Confirmar".
- Un botón rojo con el texto "Borrar".

Figura 29. Registrar usuario

Desarrollo de la propuesta → Desarrollo de la aplicación web

- 'eliminarUsuario/':

La vista renderiza una plantilla *modal* con un formulario tal y como se muestra en la Figura 30.

Si el usuario pulsa en eliminar, se elimina de la base de datos la cuenta de usuario, así como todos los datos que estén relacionados con esta de la base de datos.

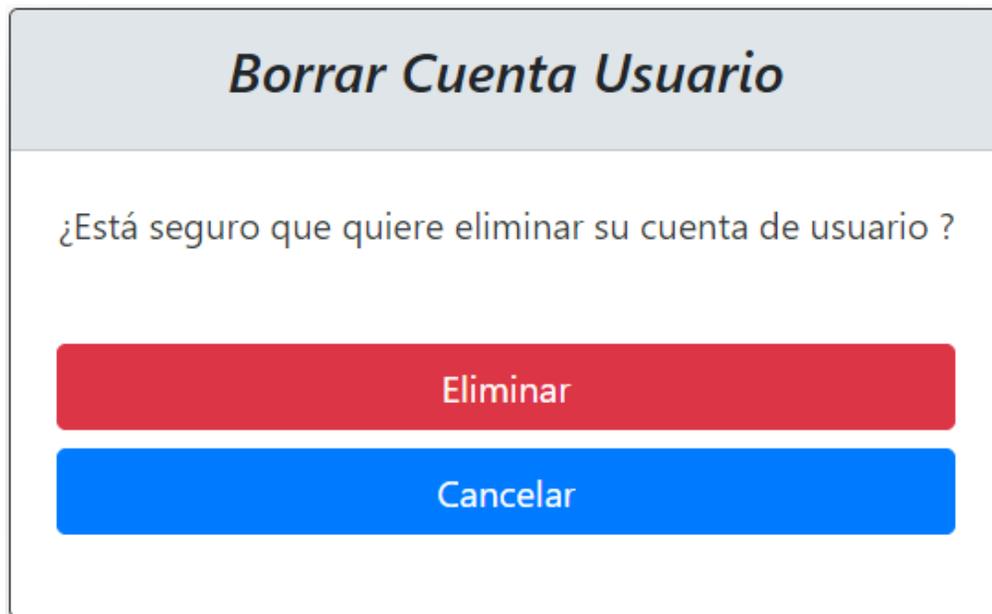


Figura 30. Modal eliminar usuario

6. Resultados

Tras la investigación y realización del proyecto, que consiste en una plataforma para la obtención de datos de Instagram, posteriormente el análisis de estos datos y el cálculo de nuevos datos a través de los extraídos se puede asegurar que datos se obtienen según el tipo de usuario o publicación que se busque. De forma cuando se busque información de un usuario de Instagram en concreto siempre se obtendrán datos como el nombre, popularidad, publicaciones, biografía, si es una cuenta de negocios, si es una cuenta verificada, si es una cuenta privada, y el enlace a la foto de perfil, dando igual si el usuario de Instagram es privado o público para la cuenta de Instagram que se encarga de obtener los datos.

Sin en cambio, para los usuarios de Instagram que sean públicos o que no sean privados para la cuenta de Instagram que se encarga de obtener los datos, también se obtienen datos de las últimas de las últimas 20 publicaciones de Instagram posteadas si es que tiene, datos de los últimos 20 vídeos de Instagram si es que tiene y datos de las últimas 20 publicaciones en la que la cuenta a buscar se encuentre etiquetada. Todos estos datos se representan mediante tablas y gráficos.

Además, para los usuarios de Instagram que no sean privados para la cuenta de Instagram que se encarga de obtener los datos, se buscan las historias destacadas que estos poseen de forma que se obtienen datos como el número de historias que tiene un conjunto de historias destacadas, cuantas de ellas son imágenes y cuantos vídeos, el enlace a la foto de la portada y el enlace a Instagram al conjunto de historias destacadas. Todos estos datos se representan mediante tablas y gráficos.

Por último, la plataforma no obtendrá ningún dato si una publicación buscada pertenece a un usuario de Instagram que es privado para la cuenta de Instagram que se encarga de obtener datos, pero, en caso contrario se obtienen datos como el título de la publicación, su propietario,

Resultados

cantidad de likes, cantidad de comentarios, ubicación si tiene, el comentario con más likes, el usuario que más comenta, menciones, *hashtags* y usuarios etiquetados. Además, si la publicación es de tipo *sidecar* también se muestra un gráfico comparando cuantas imágenes y vídeos tiene este tipo de publicación y una tabla con enlaces donde se pueden ver las imágenes una por una.

7. Conclusiones y líneas de trabajo futuras

Conociendo los resultados obtenidos se puede decir que la aplicación cumple el objetivo principal del proyecto, ya que obtiene, analiza y muestra información sobre perfiles de usuarios y publicaciones de Instagram. De este objetivo principal directamente se cumplen varios específicos ya que la información es obtenida usando técnicas de *web scraping* gracias a el paquete de *Python instaloader* anteriormente mencionado en el apartado del Proceso de obtención de datos de Instagram y también se cumple otro objetivo específico por el hecho de que gran parte de la información obtenida se presenta al usuario en formato de tablas y gráficos.

Además, la aplicación también presenta un sistema con un registro e inicio de sesión para que cada usuario nuevo pueda crearse una cuenta y si lo desea introducir una cuenta de Instagram para poder obtener datos de cuentas o publicaciones de Instagram que son privados para la cuenta predefinida de recopilación de datos de la aplicación. También se almacenan durante un cierto tiempo los datos más importantes de las búsquedas en la base de datos haciendo que la aplicación sea mucho más eficiente ante búsquedas redundantes.

A modo de conclusión personal se puede decir que durante el transcurso del desarrollo del proyecto se han adquirido y perfeccionado muchos conocimientos sobre las tecnologías usadas, sobre todo en *Django*, *Python*, *JavaScript* o *Bootstrap*.

Aunque se hayan cumplido todos los objetivos planteados al principio del proyecto, la aplicación creada puede mejorarse en muchos aspectos, de modo que durante la realización del proyecto se han ido recopilando varias ideas:

- Implementar un temporizador para para eliminar datos guardados en la base de datos sobre Instagram que sean demasiados antiguos.
- Implementar la búsqueda de *hashtags* de Instagram para ampliar la funcionalidad de la aplicación.

Conclusiones y líneas de trabajo futuras

- Crear una gestión de usuarios mucho más completa incluyendo verificación del usuario mediante correo y permitir al usuario modificar la contraseña, el correo o cualquier otro dato que se corresponda con este.
- Integrar en el sistema una base de datos MySQL o del estilo para que pueda almacenar mayor cantidad de información y responda de una manera más eficiente a las distintas peticiones.
- Subir el sistema a la nube, por ejemplo, la de *Amazon Web Services* o cualquier otra, para conseguir que la plataforma escale y que esté disponible a una mayor cantidad de usuarios.
- Desarrollar una *API* de la plataforma para que en un futuro esta se puede implementar en una aplicación móvil.

8. Bibliografía

Acosta, J. (29 de Junio de 2021). *GraphQL: Qué es y qué ventajas ofrece*. OpenWebinars:
<https://openwebinars.net/blog/graphql-que-es-y-que-ventajas-ofrece/>

Bahillo, L. (16 de Mayo de 2022). *Historia de Internet: cómo nació y cuál fue su evolución*.
Marketing4ecommerce: <https://marketing4ecommerce.net/historia-de-internet/>

Camacho, D. (2021). *Qué es GitHub y cómo usarlo para aprovechar sus beneficios*. Platzi:
<https://platzi.com/blog/que-es-github-como-funciona/>

EcuRed. (14 de Noviembre de 2016). *Visual Paradigm*. https://www.ecured.cu/Visual_Paradigm

ELOGIA. (18 de Mayo de 2022). *IAB SPAIN PRESENTA EL 'ESTUDIO DE REDES SOCIALES
2022'*. iabspain: <https://iabspain.es/iab-spain-presenta-el-estudio-de-redes-sociales-2022/>

Eniun Diseño Web y Marketing Digital. (25 de Agosto de 2019). *Características de una interfaz
web: usable, visual, educativa y actualizada*. [https://www.eniun.com/caracteristicas-
interfaz-usable-visual-educativa-actualizada/](https://www.eniun.com/caracteristicas-interfaz-usable-visual-educativa-actualizada/)

Flores, F. (22 de Julio de 2022). *Qué es Visual Studio Code y qué ventajas ofrece*.
OpenWebinars: [https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-
ofrece/](https://openwebinars.net/blog/que-es-visual-studio-code-y-que-ventajas-ofrece/)

Francisco José García Peñalvo, A. G. (2019). *PROCESO UNIFICADO*. GRIAL repository:
<https://repositorio.grial.eu/bitstream/grial/1948/1/7.%20PU-2020.pdf>

Galeano, S. (28 de Enero de 2022). *Cuáles son las redes sociales con más usuarios del mundo
(2022)*. Marketing4ecommerce: [https://marketing4ecommerce.net/cuales-redes-sociales-
con-mas-usuarios-mundo-ranking/](https://marketing4ecommerce.net/cuales-redes-sociales-con-mas-usuarios-mundo-ranking/)

Bibliografía

- García, M. N. (2021). *Práctica 1 Estimación del esfuerzo*. Universidad de Salamanca.
- Gima, N. (30 de Septiembre de 2019). *La evolución del diseño web*. HostGator: <https://www.hostgator.mx/blog/evolucion-del-diseno-web/>
- Hera, C. d. (2 de Junio de 2022). *Historia de las redes sociales: cómo nacieron y cuál fue su evolución*. Marketing4ecommerce: <https://marketing4ecommerce.net/historia-de-las-redes-sociales-evolucion/>
- Instaloader. (11 de Agosto de 2022). *Instaloader*. <https://instaloader.github.io/>
- Invitado, A. (12 de Abril de 2020). *Bootstrap: guía para principiantes de qué es, por qué y cómo usarlo*. Rockcontent: <https://rockcontent.com/es/blog/bootstrap/>
- Llonch, E. (25 de Mayo de 2021). *¿Qué son las redes sociales y cuáles son las más importantes?* Cyberclick: <https://www.cyberclick.es/numerical-blog/que-son-las-redes-sociales-y-cuales-son-las-mas-importantes>
- Lucía Martín Gómez, R. C. (30 de Octubre de 2021). *Business Benefits of Instagram Scraping: Questionable Uses of Data*. Springer: https://link.springer.com/chapter/10.1007/978-3-030-87687-6_22
- Machuca, F. (1 de Junio de 2022). *Descubre cómo funciona Microsoft Project y cómo usarlo para la gestión de proyectos*. crehana: <https://www.crehana.com/blog/desarrollo-web/que-es-microsoft-project/>
- MDN contributors. (8 de Diciembre de 2020). *Generalidades del protocolo HTTP*. MDN web docs: <https://developer.mozilla.org/es/docs/Web/HTTP/Overview>
- MDN contributors. (11 de Febrero de 2021). *Introducción a Django*. MDN Web Docs: <https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introduction>

Bibliografía

media. (14 de Abril de 2021). *¿Qué es Business Intelligence (BI) y qué herramientas existen?*

Signaturit Blog: <https://blog.signaturit.com/es/que-es-business-intelligence-bi-y-que-herramientas-existen>

Mercedes. (11 de Septiembre de 2017). *¿Qué es el desarrollo web?* Openclassrooms:

<https://blog.openclassrooms.com/es/2017/09/11/que-es-el-desarrollo-web/>

Oracle. (31 de Julio de 2014). *¿Qué es una base de datos?*

<https://www.oracle.com/es/database/what-is-database/>

Ortega, J. M. (4 de Abril de 2019). *Que es GraphQL?* Medium: [https://medium.com/@jmz12/que-](https://medium.com/@jmz12/que-es-graphql-bf835e55960)

[es-graphql-bf835e55960](https://medium.com/@jmz12/que-es-graphql-bf835e55960)

Pío. (2022). *Información sobre las API de Twitter.* Centro de Ayuda de Twitter:

[https://help.twitter.com/es/rules-and-policies/twitter-](https://help.twitter.com/es/rules-and-policies/twitter-api#:~:text=Twitter%20permite%20acceder%20a%20partes,opini%C3%B3n%20del%20)

[api#:~:text=Twitter%20permite%20acceder%20a%20partes,opini%C3%B3n%20del%20cliente%20en%20Twitter.](https://help.twitter.com/es/rules-and-policies/twitter-api#:~:text=Twitter%20permite%20acceder%20a%20partes,opini%C3%B3n%20del%20cliente%20en%20Twitter)

Portaltic. (09 de Febrero de 2022). *Nueve de cada diez personas usan redes sociales en España*

y dedican casi 2 horas diarias a ellas.

<https://www.europapress.es/portaltic/socialmedia/noticia-nueve-cada-diez-personas-usan-redes-sociales-espana-dedican-casi-horas-diarias-ellas-20220209163949.html>

Ramos, R. (25 de Septiembre de 2021). *¿Qué es JavaScript y para qué sirve?* Soyrafaramos:

<https://soyrafaramos.com/que-es-javascript-para-que-sirve/>

Reyes, J. J. (6 de Mayo de 2016). *¿Qué es HTML?* DevCode: [https://devcode.la/blog/que-es-](https://devcode.la/blog/que-es-html/)

[html/](https://devcode.la/blog/que-es-html/)

Rubio, J. C. (25 de Febrero de 2019). *Qué es GIT y para qué sirve.* OpenWebinars:

<https://openwebinars.net/blog/que-es-git-y-para-que-sirve/>

Bibliografía

- Santos, D. (8 de Agosto de 2022). *Introducción al CSS: qué es, para qué sirve y otras 10 preguntas frecuentes*. HubSpot: <https://blog.hubspot.es/website/que-es-css>
- Schiaffarino, A. (12 de Marzo de 2019). *Modelo cliente servidor*. Infranetworking: <https://blog.infranetworking.com/modelo-cliente-servidor/>
- Seguro, N. (17 de Septiembre de 2021). *¿Qué es el desarrollo Web y por qué es importante?* Coderhouse: <https://www.coderhouse.es/blog/que-es-el-desarrollo-web>
- SocioViz. (2014). *Twitter analytics made easy*. <https://socioviz.net/>
- Sphinx. (6 de Agosto de 2022). *Getting Started*. <https://www.sphinx-doc.org/en/master/usage/quickstart.html>
- Twitter. (2022). *Twitter API*. Developer Platform: <https://developer.twitter.com/en/products/twitter-api>
- Velasco, R. (28 de noviembre de 2021). *Conoce SQLite, el popular motor de bases de datos*. SoftZone: <https://www.softzone.es/programas/lenguajes/que-es-sqlite/>
- Visus, A. (Octubre de 2020). *¿Para qué sirve Python? Razones para utilizar este lenguaje de programación*. Esic: <https://www.esic.edu/rethink/tecnologia/para-que-sirve-python>
- Yogeshwaran. (3 de abril de 2022). *instagramy 4.5*. PyPI: <https://pypi.org/project/instagramy/>
- Zhao, B. (Mayo de 2017). *Web Scraping*. researchgate: https://www.researchgate.net/profile/Bo-Zhao-3/publication/317177787_Web_Scraping/links/5c293f85a6fdccfc7073192f/Web-Scraping.pdf
- Zonín, A. (6 de Junio de 2019). *SocioViz es completamente nuevo*. Historias de datos digitales: <https://digitaldatastories.it/2019/06/06/socioviz-is-completely-new/>

9. Glosario

API: Cuyo significado es “*Application Programming Interface*”, se trata de un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar software de las aplicaciones, permitiendo la comunicación de dos aplicaciones de software a través de un conjunto de reglas.

Checkbox: También llamado “casilla de verificación”, es un elemento de interacción de la interfaz gráfica con el usuario, para seleccionar o deseleccionar algún elemento.

Clave primaria: Campo que identifica de forma única la tupla de una base de datos.

Hashtags: Conjunto de caracteres precedidos por una almohadilla (#) identificar o etiquetar una publicación o mensaje.

Highlights: También llamados “historias destacadas”, son historias que siempre están presentes en un perfil de Instagram.

Historia: Es una foto o un vídeo de Instagram en formato vertical que desaparece después de 24 horas.

HTML: Cuyo significado es “*HyperText Markup Language*”, es el lenguaje de marcado para la elaboración de páginas web.

HTTP: Cuyo significado es “*Hypertext Transfer Protocol*,”, es el protocolo de comunicación que permite transferencias de información a través de la red.

Modal: Cuadro que aparece en una página web, que bloquea todas las funciones para concentrar el foco en realizar una acción en concreto.

Navegador: Aplicación o programa que permite el acceso a la web, interpretando la información de distintos archivos y sitios web para que estos puedan ser vistos.

Publicación: Elemento visual postado, en Instagram puede ser una imagen, vídeo o *sidecar*.

Renderizar: Término que se refiere a la representación gráfica, en este caso de páginas web elaboradas en el lenguaje de marcado *HTML*.

Sidecar: Publicación de Instagram que contiene un conjunto de imágenes o vídeos.

SQL: Cuyo significado es “*Structured Query Language*”, es un lenguaje diseñado para administrar y recuperar información de sistemas de gestión de bases de datos relacionales.

Tupla: Se trata de un registro (fila) de una base de datos relacional.

UML: Cuyo significado es “*Unified Modeling Language*”, se trata de un estándar para crear esquemas, diagramas y documentación relativa para el desarrollo de software.

URL: Cuyo significado es “*Uniform Resource Locator*”, es la dirección única y específica que se le asigna a un recurso que existe en la red.

Web scraping: Proceso de utilizar una aplicación para extraer información relevante de un sitio web.

