Universidades de Burgos, León y Valladolid

Máster universitario

Inteligencia de Negocio y Big Data en Entornos Seguros

Trabajo Fin de Máster

Automatización del proceso de adquisición de imágenes de redes sociales para sistemas de recomendación

Presentado por Daniel González Alonso
en Universidad de Burgos — 15 de julio de 2022
Tutor: Ángel Manuel Guerrero Higueras

Resumen

Debido a las consecuencias derivadas por la pandemia de COVID 19 el número de turistas que visitaron la ciudad de Valladolid (España) se ha visto reducido en comparación con años previos. Con el fin de recuperar el número de turistas anterior se puede explotar el contenido generado por los usuarios en redes sociales como Instagram.

Una de las posibilidades es la recomendación de publicaciones relacionadas con los destinos turísticos que se asemejen a las características de los potenciales visitantes. Dentro de este contexto en el presente Trabajo de Fin de Máster se presenta una posible implementación de un método de descarga, procesado y almacenamiento en la nube de publicaciones de Instagram mediante el uso de Amazon Web Services. Junto con el desarrollo de un cuadro de mandos para facilitar la visualización y filtrado de los datos generados.

Descriptores

Red social, Instagram, Sistema de Recomendación, Visión Artificial, Análisis de Emoción, Big Data.

Abstract

Due to the consecuences derived from the COVID 19 pandemic the number of tourists that visited the city of Valladolid (Spain) has dropped in comparison to previous years. With the purpose of recovering the old number of tourists the user generated content in social media like Instagram could be exploited.

One posibility is the recommendation of posts related to the tourist destinations to the potential visitors. In this context the current Master's degree final project presents one posible implementation of a cloud based method for downloading, processing and storing Instagram posts using Amazon Web Services. Also, the development of a dashboard with the purpose of helping with the visualization and filtering of the generated data.

Keywords

Social Media, Instagram, Recommendation System, Computer Vision, Emotion Analysis, Big Data.

Índice general

Índice de figuras

Índice de tablas

Memoria

Introducción

El turismo es una de las actividades económicas más importantes de España, en 2019 suponía el 12,4 % del PIB del país, siendo el tercer país más visitado del mundo. Con la pandemia de COVID-19 el sector turístico se vio gravemente afectado por las restricciones de movilidad y el propio temor a la enfermedad. En la actualidad este sector todavía no ha recuperado los niveles previos a la pandemia. Si en 2019 se recibieron 83,5 millones de turistas, en 2021 esa cifra ha sido aproximadamente de 31,2 millones [11].

En el caso de Valladolid en concreto, esta ciudad no partía de una situación ventajosa en comparación con otras ciudades, ubicándose en la zona media-baja del ranking de ciudades más visitadas de España. Y al igual que el resto del país, la pandemia ha reducido el número de visitantes: de acuerdo al INE en la provincia se ha pasado de unos 51000 en 2019 a aproximadamente 33000 en 2021.

Para recuperar los niveles de turismo previos (y seguir aumentándolos) es necesario potenciar el turismo, y uno de los posibles medios que se pueden aprovechar es internet, y en concreto las redes sociales. Saber explotar la información de las redes sociales puede tener una influencia enorme a la hora de mejorar el número de visitantes que pueda tener una ciudad. Pero abordar esta tarea es una asunto complicado.

Con la aparición y popularización de las redes sociales, cada vez más gente comparte los lugares que visita y sus opiniones. Las publicaciones que hace la gente en estas plataformas ayudan a otras personas a descubrir nuevas experiencias y lugares que visitar. También permite a las personas conocer más en detalle aquellos sitios a los que planea viajar. Por otro lado, todos estos contenidos generados por los usuarios son una fuente de

información incalculable para todas aquellas empresas que están buscando mejorar sus productos vacacionales.

Una de las redes sociales más usadas en la actualidad es Instagram. Esta red social permite compartir publicaciones principalmente en forma de imagen o vídeo. La forma en que se muestra el contenido en esta red social la hace ideal para compartir lugares visitados o experiencias vividas, y aunque no sea el único tipo de publicaciones que se comparten en esta red social, si que es bastante común encontrarlo. Es por ello que los efectos que puede tener esta red social a la hora de difundir el potencial turístico que tiene una ubicación son muy importantes.

Cuando se llevan a cabo búsquedas en Instagram, las publicaciones que se muestran se basan en tres criterios, la cadena de búsqueda, los cuentas y hashtags que se siguen o visita y la popularidad de las publicaciones encontradas. Un problema que tiene esta forma de mostrar las publicaciones es que si se pretende hacer turismo a un lugar lejano, es posible que ninguno de tus contactos tenga influencia en los resultados, y es más que probable que Instagram tan solo muestre las publicaciones más populares del destino que estás buscando.

Es por ello que en este trabajo se pretende investigar el posible uso de la visión por computador y el análisis de emoción en las publicaciones de Instagram para facilitar el descubrimiento de actividades y lugares recomendables en Valladolid.

1.1. Organización de la memoria

La memoria se encuentra dividida en los siguientes capítulos:

1. Introducción: se presenta el proyecto y la relación del turismo con las redes sociales.

2. Objetivos del proyecto: se describen los distintos objetivos que se buscan cumplir con el desarrollo del proyecto.

3. Conceptos teóricos: se introducen los conceptos teóricos necesarios para compresión de la memoria.

4. Técnicas y herramientas: se enumeran y describen las múltiples herramientas, servicios, librerías y APIs empleadas para el desarrollo del proyecto.

5. Aspectos relevantes del desarrollo del proyecto: se expone el trabajo desarrollado, los problemas encontrados y sus soluciones.

6. Trabajos relacionados: se comenta el estado del arte del proyecto.

7. Conclusiones y Líneas de trabajo futuras: se indican los resultados, las conclusiones y las posibles mejoras del proyecto desarrollado.

1.2. Organización de los apéndices

La memoria consta de los siguientes apéndices:

1. Plan de Proyecto de Software: se presenta la planificación temporal del proyecto y su viabilidad desde el punto de vista legal y económico.

2. Documentación técnica programación: se describe el material entregado y como instalarlo y ejecutarlo.

Objetivos del proyecto

En este apartado se exponen los distintos objetivos que se pretenden satisfacer con el desarrollo del proyecto. Estos objetivos se han dividido en tres tipos: generales, técnicos y personales, los cuales se describen a continuación.

2.1. Objetivos generales

Los objetivos principales planteados para el proyecto son los siguientes:

- Conocer el estado del arte, investigando artículos que ya hayan sido realizados en relación a la recomendación de publicaciones o lugares donde hacer turismo en base a contenido de redes sociales.
- Desarrollar el software necesario para descargar publicaciones de Instagram y su posterior almacenamiento.
- Usar algún servicio de reconocimiento de imágenes para llevar a cabo análisis de emoción sobre las publicaciones, así como la extracción de información relativa a las personas presentes en la imagen, como su edad o género.
- Implementar o usar algún medio para llevar a cabo la visualización y filtrado de los datos de forma sencilla.

2.2. Objetivos técnicos

En base a los objetivos anteriores se plantean los siguientes requerimientos técnicos:

- Se busca el empleo de herramientas en la nube para el alojamiento y procesamiento de los datos.
- El desarrollo del proyecto no ha de suponer ningún coste, por ello se pretende usar software libre, herramientas gratuitas o herramientas que tengan un periodo de prueba lo suficientemente extenso como para poder desarrollar el proyecto sin complicaciones.
- Para la visualización de los datos se busca desarrollar un cuadro de mandos con alguna herramienta o servicio ya existente.

2.3. Objetivos personales

Entre los objetivos personales que se buscan satisfacer con este proyecto se encuentran los siguientes:

- Familiarizarse con el uso de herramientas y servicios populares de computación en la nube.
- Ampliar mis conocimientos del lenguaje de programación Python así como aprender a usar nuevas librerías e interfaces de programación.
- Poner en práctica mis conocimientos sobre cuadros de mando, profundizando en el uso de herramientas utilizadas a lo largo del máster o aprendiendo a usar otras nuevas.
- Mejorar mis conocimientos sobre almacenamiento y procesado de grandes cantidades de datos.

Conceptos teóricos

En este apartado se presentan los distintos conceptos teóricos necesarios para la correcta comprensión del proyecto.

3.1. Emoción y Sentimiento

A pesar de que en el lenguaje popular en muchas ocasiones se usan ambas palabras de forma equivalente, no lo son. Las emociones y los sentimientos representan dos conceptos distintos aunque relacionados.

Por un lado, las emociones son un conjunto de reacciones neurofisiológicas que sufre un individuo al ser expuesto a algún objeto, lugar, persona, suceso o idea. Las emociones son transitorias, se dan de forma inconsciente y espontánea. En múltiples ocasiones las emociones son el origen de los cambios en la motivación y el comportamiento de los individuos. En ocasiones se dividen las emociones entre primarias o básicas y secundarias, siendo las 4 emociones primarias el miedo, la tristeza, el enfado y la alegría [24].

Por otro lado, los sentimientos son un estado afectivo que se genera a partir de una emoción [15]. En el sentimiento interviene tanto la reacción neurofisiológica como un componente cognitivo. Esto significa que mientras que la emoción se daba de forma instintiva, en el sentimiento interviene un razonamiento que lo hace más duradero.

En resumen, las principales diferencias entre emoción y pensamiento radican en:

1. Mientras que las emociones duran poco tiempo los sentimientos son duraderos.

2. La emoción precede al sentimiento.

3. Los sentimientos se producen instintivamente mientras que en los sentimientos interviene el pensamiento, el cual es afectado por la experiencia previa del individuo, características subjetivas, etc.

3.2. Análisis de Emociones y de Sentimientos

Al igual que sucedía en el apartado anterior, los conceptos de análisis de emoción y de sentimiento a pesar de parecer similares no lo son.

Primero, el análisis de emociones busca descubrir la reacción que puede tener un individuo ante cierto estímulo. Como se comentó en el anteriormente, las emociones en muchas ocasiones son el motivo de las acciones de las personas, con lo que conocer si una persona se encuentra aburrida, contenta o enfadada puede ser de utilidad para dar una respuesta adecuada.

Por otro lado, el análisis de sentimiento no busca descubrir que emoción en particular es expresada por los usuarios, si no más bien conocer si la respuesta de un individuo es positiva o negativa, lo que se suele conocer como polaridad (en ocasiones también se incluye la neutralidad) [31]. Ejemplos de este tipo de análisis pueden ser el conocer el éxito de una campaña publicitaria, la recepción de una película, serie, noticia, etc.

A lo largo de los últimos años se han desarrollado múltiples trabajos relacionados con el análisis de emociones y de sentimiento desde distintos campos como puede ser la psicología, la medicina y las ciencias de la computación. Los trabajos relacionados con este último campo se engloban dentro de lo que se conoce como la *Computación Afectiva*, y entre sus objetivos se encuentran el desarrollo de sistemas capaces de de reconocer, interpretar, procesar y estimular las emociones de las personas [8]. El objetivo detrás de esta interpretación de las emociones suele ser el poder adaptar el comportamiento de los programas para dar una respuesta adecuada a esas emociones. Para ello se puede hacer uso de biometría, procesamiento de lenguaje natural, análisis de voz, imágenes, vídeos, etc. con el fin de extraer, identificar y cuantificar estados afectivos o información subjetiva. Esta información puede ser empleada para múltiples propósitos como puede ser mejorar un servicio, la atención al cliente o el marketing de productos.

3.3. Red social

Existen múltiples definiciones de red social. La RAE define a las redes sociales como un «servicio de la sociedad de la información que ofrece a los usuarios una plataforma de comunicación a través de internet». Este servicio permite la comunicación de sus usuarios mediante mensajes, audio, imágenes o vídeos, lo que facilita la creación de comunidades con gustos similares.

Por otro lado, el INTECO en su documento "Estudio sobre la privacidad de los datos personales y la seguridad de la información en las redes sociales online" define a las redes sociales como unos «servicios prestados a través de Internet que permiten a los usuarios generar un perfil público, en el que plasmar datos personales e información de uno mismo, disponiendo de herramientas que permiten interactuar con el resto de usuarios afines o no al perfil publicado».

Sobre su utilidad, en [29] el ONTSI afirma que las redes sociales sirven como una herramienta de «democratización de la información que transforma a las personas en receptores y en productores de contenidos».

Hay múltiples tipos de redes sociales de acuerdo a distintos criterios de clasificación:

- Según el nivel de integración, podemos tener redes sociales *horizontales* donde no existe una temática definida o *verticales* donde los temas de conversación están dirigidos hacia un tema en específico con el fin de juntar a un grupo de usuarios que tengan interés en dicho tipo de contenido, como podrían ser los foros especializados.
- Según la finalidad, podríamos clasificar las redes sociales entre aquellas que son de *ocio*, donde se busca el entretenimiento, ya sea hablando sobre algun tema como podría ser música, películas o videojuegos, o simplemente conversando de forma libre, y por otro lado podríamos tener redes *profesionales* con el fín de promocionarse o la búsqueda de empleo.
- Según su apertura podemos tener redes sociales *públicas* donde cualquiera con acceso a internet puede comentar ya sea con o sin necesidad de registrarse, o redes *privadas* donde el grupo de usuarios que puede acceder es cerrado.
- Según el modo de funcionamiento, donde se pueden clasificar entre redes sociales de *contenido* donde se busca compartir documentos,

vídeos, imágenes... Las redes basadas en *microblogging* donde el contenido compartido consiste normalmente en mensajes contenidos que facilitan el seguimiento por parte de los usuarios. Y por último aquellas basadas en *perfiles* donde se busca compartir contenido personal o profesional mediante la creación de perfiles o fichas de usuario.

Las redes sociales se han popularizado mucho en los últimos años. De acuerdo con el artículo de wikipedia sobre las redes sociales, las más populares en la actualidad son las siguientes:

Nombre	Número de usuarios (en millones)
Facebook	2910
YouTube	2562
WhatsApp	2000
Instagram	1478
WeChat	1263

Tabla 3.1: Ránking de redes sociales a Enero de 2022

Instagram

Instagram es una red social estadounidense, en la actualidad propiedad del grupo Meta, lanzada en Octubre de 2010. Originalmente se lanzó como aplicación exclusiva de los móviles iPhone, aunque en la actualidad tiene aplicaciones para otras plataformas móviles como Android, así como versión web accesible desde cualquier dispositivo. El objetivo principal de esta red social es compartir imágenes o vídeos entre usuarios, ya sea de forma pública o entre seguidores autorizados, aunque en la actualidad Instagram ofrece más servicios como la mensajería directa entre usuarios, subir contenido de duración limitada con *Instagram Stories* o el streaming de vídeo mediante IGTV. Dentro de la clasificación anteriormente expuesta, Instagram sería una red horizontal, de ocio, pública y basada en contenido.

Instagram permite el etiquetado de su contenido mediante el uso de *hashtags*, con el fin de facilitar la búsqueda de contenido similar y el descubrimiento de cuentas de usuario que comparten publicaciones sobre dicho tema, permitiendo crear comunidades de personas con gustos afines.

3.4. Web scraping

El web scraping, o raspado web, es una técnica empleada para extraer datos de páginas web. Aunque esta acción puede ser llevada a cabo manualmente por un usuario, normalmente esta técnica se refiere a procesos automatizados mediante software como pueden ser *bots* o *crawlers*, los cuales hacen uso de protocolo HTTP para acceder a las páginas web simulando el comportamiento que haría una persona. Normalmente, este tipo de programas extraen, parsean y transforman la información mostrada por la página web (generalmente de forma no estructurada) para después almacenarla en hojas de cálculo o bases de datos. Existen múltiples técnicas para llevar a cabo web scrapping, aunque las más comunes consisten en llevar a cabo búsqueda de patrones o parsear los documentos HTML de las páginas web para buscar ciertas etiquetas.

3.5. Visión por computador

La visión por computador, también conocida como visión artificial, es el campo de las ciencias de la computación encargado analizar, comprender y responder a imágenes o vídeos [1]. Para ello se suele hacer uso de técnicas de aprendizaje profundo para buscar patrones en la entrada, de modo que tras una fase de entrenamiento, el modelo generado sea capaz de reconocer, clasificar y reaccionar a lo que "ve". La visión artificial es un campo en crecimiento, con un potencial enorme puesto que puede ser aplicada en entornos donde se requiera una monitorización constante, ya que permite evitar evitar errores humanos derivados de las distracciones o de la fatiga. Entre las aplicaciones más comunes en la actualidad y con más potencial en el futuro se encuentran el control de calidad en procesos de manufactura, la detección de intrusos, diagnosis médica, conducción autónoma, etc.

3.6. Dashboard

Un dashboard o cuadro de mando es un tipo de interfaz de usuario capaz de mostrar de forma integrada múltiples indicadores clave de rendimiento (KPIs) que son de interés para un proceso de negocio. El uso de cuadros de mando es muy común en la inteligencia de negocios para ayudar a visualizar datos complejos. Esto es debido a que los cuadros de mando permiten:

- Mostrar múltiples visualizaciones simultáneamente.

- Usar diversos tipos de gráficos de distinto tipo.
- Interaccionar con los datos, generalmente mediante filtros que permiten focalizar la representación en una serie de datos que puedan ser de interés del usuario.

A mayores, los cuadros de mandos pueden permitir resaltar o hacer anotaciones, crear alarmas o notificaciones si los datos salen fuera de un umbral o cota de referencia, o refrescar los datos presentados de forma automática según estos se van actualizando.

El objetivo principal de los cuadros de mandos es ayudar a sus usuarios a la gestión facilitando la comprensión de los datos. De acorde al tipo de usuario que va a emplear un cuadro de mando, éstos se podrían clasificar dentro de las siguientes categorías:

- Paneles estratégicos: son usados por el personal directivo y se usan para monitorizar el grado de cumplimiento de la estrategia de una empresa en base a distintos KPIs. En este tipo de panel se trabaja a largo plazo, y generalmente son complejos de crear.
- Paneles operacionales: Son usados por empleados o mandos intermedios para monitorizar y gestionar tareas relacionadas con su trabajo diario. En este tipo de panel se trabaja en intervalos de tiempo más cortos, y son el tipo de panel más habitual.
- Paneles analíticos: son usados por analistas y directivos intermedios, se usan para ayudar a interpretar la información de grandes cantidades de datos, principalmente para comparar el rendimiento de datos históricos para poder predecir comportamientos o tendencias a corto y medio plazo.

Técnicas y herramientas

Este apartado tiene el objetivo de enumerar y describir las múltiples herramientas utilizadas para el desarrollo del presente proyecto.

Primero cabe destacar que a lo largo del proyecto siempre se ha tratado de usar el lenguaje de programación Python. La decisión de usar este lenguaje de programación se basa en que es un lenguaje que conozco, con el que he trabajado con anterioridad, y que considero sencillo y especialmente útil para llevar a cabo prototipos de forma rápida. Además al ser un lenguaje muy popular en la actualidad, es muy fácil encontrar librerías y recursos en la web en caso de duda. La decisión de emplear este lenguaje de programación ha influido mucho a la hora de escoger el resto de herramientas que se exponen a continuación:

4.1. Instalooter

Instalooter es un scraper web empleado para descargar imágenes, vídeos y meta-datos de las publicaciones de Instagram. Esta herramienta es de código abierto, y se distribuye bajo la licencia GNU General Public License v3, pudiendo acceder al código fuente del proyecto a través de GitHub. Instalooter puede ser usado por un lado a través de un cliente de línea de comandos, y por otro mediante una interfaz de programación para Python 3.

La API de Instalooter pone a disposición del usuario de varios *looters* [20], ya sea para descargar las publicaciones llevadas a cabo por un usuario en su perfil o para descargar publicaciones relacionadas con un Hashtag. Estos *looters* permiten descargar tanto las imágenes como la descripción y meta-datos de las mismas en formato JSON. Además, antes de descargar la

información de una publicación Instalooter comprueba que no haya sido ya descargada, con lo que puede ejecutarse repetidamente sobre un perfil o un Hashtag sin peligro de obtener resultados duplicados.

4.2. Servicios en la Nube

Respecto al almacenamiento y procesamiento de los datos descargados de las publicaciones de Instagram, se trató desde un principio de usar computación en la nube. Esto es debido a la facilidad a la hora de escalar tanto la capacidad de almacenamiento como de computación en caso de que fuese necesario.

Debido a los motivos esgrimidos en el apartado 5.1 se decidió que lo mejor era emplear los servicios de **Amazon Web Services**. A continuación se describen los servicios de AWS que se han empleado para el desarrollo del presente proyecto.

Rekognition

Rekognition es un servicio de AWS que permite analizar imágenes y vídeos mediante su tecnología de aprendizaje profundo. En el caso de Rekognition Image, este servicio de visión artificial permite llevar a cabo de forma sencilla tareas como la localización y extracción de texto de imágenes, reconocimiento facial y comparación de caras para saber si en dos imágenes aparece la misma persona, detección de objetos y escenas, análisis facial para reconocimiento de gestos o atributos, etc. Además junto con las posibles etiquetas que devuelve esta herramienta se incluye el *likelihood* o la fiabilidad de la respuesta [3]. Cabe destacar que Rekognition se encuentra integrado con el resto de servicios de AWS como IAM para controlar el acceso y los permisos.

Las posibles aplicaciones o casos de uso de este servicio son múltiples, como por ejemplo reconocer personas famosas, saber si las personas llevan puesto equipos de protección, detección de contenido inapropiado, etc. La principal ventaja de usar este servicio es su capacidad de escalado. Los precios de uso del servicio varían dependiendo del volumen de imágenes a procesar, siendo las primeras 5000 imágenes del mes gratuitas durante el periodo de prueba de 12 meses.

Amazon S3

Amazon S3, *Simple Storage Service* por sus siglas en inglés, es un servicio de almacenamiento de objetos en la nube de Amazon. El almacenamiento en S3 permite almacenar cualquier tipo de objeto de forma escalable, con baja latencia y alta disponibilidad. El acceso a este servicio puede llevarse a cabo mediante la consola web de Amazon AWS, el SDK de AWS o mediante APIs REST.

Cada objeto almacenado en S3 se ubica dentro de un recurso denominado "bucket" o contenedor de objetos, y puede llegar a ocupar hasta 5TBs [5]. Estos objetos se organizan mediante prefijos y se les puede añadir hasta 10 etiquetas clave-valor. Además, se puede habilitar que la modificación de los objetos almacenados en S3 pueda ser supervisada mediante sistemas de control de versiones así como habilitar *Multi-Factor Authentication* para evitar el borrado accidental. Cabe destacar también que los objetos pueden ser replicados en varias regiones para reducir la latencia de acceso si nuestra aplicación así lo requiere.

Amazon S3 tiene varias clases de almacenamiento. Dependiendo del caso de uso y la forma en que se va a acceder a los datos puede ser más conveniente usar una clase u otra con el fin de reducir la latencia y el coste de servicio de nuestra aplicación. Estas clases son las siguientes:

- *S3 Intelligent-Tiering*: preferible para aplicaciones con patrones de acceso desconocidos o cambiantes ya que optimiza de forma automática los costes de almacenamiento.
- *S3 Standard*: preferible para los datos de acceso frecuente.
- *S3 Standard-Infrequent Access* y *S3 One Zone-Infrequent Access*: preferibles para los datos de acceso poco frecuente.
- *S3 Glacier Instant Retrieval*, *S3 Glacier Flexible Retrieval* y *S3 Glacier Deep Archive*: preferibles para el archivado de datos, siendo preferible uno u otro en base a las necesidades de tiempo de recuperación (*instant* en milisegundos, *flexible* en minutos, *deep archive* en horas).
- *S3 Outposts*: útil para aquellos casos donde existe alguna normativa que nos obliga a mantener los datos fuera de alguna región en concreto.

Además Amazon S3 permite regular el acceso a los datos mediante listas de control de acceso (ACL), políticas de bucket, etc. Cabe destacar que el

servicio IAM de AWS simplifica mucho la administración del acceso a los datos.

Amazon EC2

Amazon EC2, Elastic Compute Cloud por sus siglas en inglés, es un servicio de AWS que permite crear instancias o máquinas virtuales en la nube con alta disponibilidad y a un coste asequible. EC2 permite elegir por un lado el hardware que más se adapte a nuestras necesidades, como arquitectura de CPU, modelo de CPU, memoria RAM, capacidad y tipo de almacenamiento, ancho de banda, etc. Y por otro el software, con disponibilidad de distintos sistemas operativos como Microsoft Windows, Amazon Linux, Red Hat, Debian. Ubuntu, SUSE, etc.

Una de las ventajas de usar EC2 es su flexibilidad que permite reducir costes adaptándose a las necesidades de cómputo del usuario, teniendo planes que permiten ejecutar el cómputo solo a ciertas horas, planes que permiten ajustarse a la demanda del usuario, planes para cierta cantidad de uso por hora constante, etc. Además, la capa gratuita de AWS ofrece hasta 750 horas de cómputo al mes durante el primer año.

Cabe destacar que EC2 ofrece distintos tipos de instancias [7]:

- Uso general: es la opción más equilibrada.
- Optimizadas para informática: con procesadores de alto rendimiento.
- Optimizadas para memoria: para aplicaciones que necesitan trabajar con muchos datos en memoria.
- Informática acelerada: con aceleradores hardware como GPUs.
- Optimizadas para almacenamiento: para aplicaciones que dependen mucho de la entrada/salida.

DynamoDB

DynamoDB es una base de datos NoSQL de alta disponibilidad de Amazon. Esta tecnología surgió debido a las necesidades particulares de Amazon, ya que esta empresa tiene cientos de servicios descentralizados con picos de demanda muy elevados [12]. Para dar soporte a estas necesidades de escalabilidad y de alta disponibilidad Amazon ya había desarrollado con anterioridad otros sistemas de almacenamiento como S3. Pero como existen

muchos servicios de Amazon cuyos patrones de acceso son únicamente clave-valor, decidieron crear un sistema de almacenamiento en específico para este tipo de casos que tuviera menor latencia. Otros requisitos que tenía que cumplir este sistema era que tenía que poder ejecutarse sobre hardware básico (barato y extendido), e intentar cumplir con las propiedades ACID: Atomicidad, Consistencia, Aislamiento y Durabilidad. Como cumplir con todas estas propiedades suelen provocar que la disponibilidad no sea muy elevada, DynamoDB sacrifica ligeramente la propiedad de Consistencia.

Para la distribución del conjunto de datos en distintos nodos, DynamoDB emplea una clave de particionamiento, que es implementada como una clave hash. Por otro lado, para garantizar la durabilidad de los datos esta herramienta los replica en varios nodos, y cualquier cambio en los datos se propaga de forma asíncrona ya que DynamoDB garantiza la consistencia de los datos de forma eventual.

Para usar DynamoDB en AWS se dispone de Amazon DynamoDB, que es un servicio que nos permite utilizar esta base de datos NoSQL sin la necesidad de tener que gestionar y mantener nuestro propio servidor, ya que se ejecuta y almacena en la nube de Amazon. El coste de este servicio es bajo demanda, pero la capa gratuita de AWS nos permite utilizar hasta 25GBs de forma permanente.

AWS Lambda

AWS Lambda es un servicio de Amazon que permite ejecutar código "sin servidor", es decir, sin la necesidad de tener que gestionar una instancia o un clúster [6]. Lambda permite ejecutar código de forma nativa de múltiples lenguajes de programación, como puede ser Java, Node.js, Go, Ruby, Python, etc. Este servicio ejecuta nuestro código como respuesta a eventos, como por ejemplo la modificación de un fichero de S3, cambios en nuestra base de datos de DynamoDB, solicitudes HTTP, eventos temporales, etc.

Como se ha comentado, la principal característica y ventaja de Lambda es no tener que gestionar nuestro propio servidor, teniendo que llevar a cabo sus actualizaciones y demás inconvenientes. Pero además ofrece otras ventajas como puede ser la completa integración con otros servicios de AWS, el escalado automático, la alta disponibilidad, un tiempo de despliegue e inicio muy corto y un costo de operación reducido puesto que el modelo de Lambda es de pago por uso. Además, Amazon ofrece 1 millón de llamadas gratuitas al mes.

Amazon API Gateway

Amazon API Gateway es un servicio de Amazon que permite la publicación de APIs de forma sencilla [4]. Este servicio facilita la creación, monitorización y mantenimiento de las APIs, lo que se traslada en una reducción del coste de horas que se tienen que dedicar a este tipo de tareas. Además este servicio ofrece alta disponibilidad y baja latencia. Las APIs que se pueden publicar en API Gateway pueden ser WebSocket o RESTful.

Este servicio se puede integrar con otros servicios de AWS como Lambda para llamar a nuestros métodos fácilmente. Se trata de un servicio de pago por uso, pero Amazon ofrece 1 millón de llamadas a las APIs al mes de forma gratuita durante el primer año.

4.3. Grafana

Grafana es una herramienta multiplataforma de código abierto para la visualización y análisis de datos. Para ello Grafana permite crear, explorar y compartir cuadros de mando [30][18]. Entre las múltiples características que ofrece esta herramienta cabe destacar:

- Es una herramienta flexible que permite crear diversos tipos de gráficos con multitud de opciones, como histogramas, mapas geográficos, gráficos de tarta, de barras, etc.
- Permite la creación de dashboard dinámicos que pueden ser reusados fácilmente. Los cuadros de mando se pueden exportar en formato JSON de forma que se puede copiar y pegar de una forma sencilla.
- Se pueden añadir variables que pueden ser usadas como filtros.
- Permite la creación de consultas de forma dinámica y comparar diferentes rangos de tiempo.
- Puede ser usado para explorar logs en vivo.
- Permite definir alertas en base a distintas métricas.
- Es capaz de mezclar datos de distintos orígenes en un mismo dashboard.

Grafana es extensible mediante plugins que permiten añadir funcionalidad así como conectores con distintos tipos de fuentes de datos. Al ser código abierto existe una gran comunidad detrás de Grafana que ayuda tanto al

desarrollo y soporte de la herramienta en sí, como para la creación de nuevos plugins. En la actualidad hay conectores para múltiples fuentes de datos como pueden ser hojas de cálculo, bases de datos MySQL, Oracle, MongoDB, Splunk...

4.4. Otras herramientas

Además de las anteriores herramientas utilizadas para la descarga, almacenamiento y procesamiento de datos, se han utilizados otras herramientas y utilidades para llevar a cabo tanto la implementación de los programas como la realización de la memoria, entre las que cabe destacar:

- virtualenv: herramienta empleada en Python para crear entornos aislados, evitando que las actualizaciones e instalaciones llevadas a cabo para otros proyectos afecten al actual.
- Visual Studio Code: editor de código fuente usado para la implementación de los scripts.
- Git: Sistema de Control de Versiones empleado tanto con el código como con la memoria del proyecto. El repositorio se decidió almacenar en la forja GitHub para poder compartirlo fácilmente con el tutor, y además para poder acceder a sus herramientas de seguimiento del proyecto.
- LaTeX: Sistema de composición de textos de alta calidad empleado para la creación de la memoria y la presentación. LaTeX se usó junto con la web Overleaf, ya que ésta evita tener que instalar todo el entorno de compilación de LaTeX junto con sus dependencias, y además permite sincronizar los cambios con GitHub fácilmente.

Aspectos relevantes del desarrollo del proyecto

En este capítulo se explican las partes más relevantes del desarrollo del proyecto, así como los problemas surgidos y las decisiones tomadas al respecto. Este apartado se ha decidido dividir en 3 secciones: el estudio previo al desarrollo del proyecto, la implementación y llenado de la base de datos, y la presentación de los datos.

5.1. Estudio preliminar

En esta fase inicial se decidió investigar las posibles herramientas a utilizar para el desarrollo del proyecto, así como desarrollar una serie de scripts para probar su funcionamiento y familiarizarse con sus interfaces de programación.

El primer paso que se llevó a cabo en esta fase del proyecto fue crear un entorno de trabajo sobre el cual poder implementar el resto de scripts del proyecto. Como se indicó en el apartado 4.4, para esta tarea se decidió emplear la herramienta `virtualenv`, que nos permite crear un entorno aislado para Python de modo que se puedan fijar las versiones de las librerías descargadas en este caso mediante `pip3` a cierta versión, algo muy importante en esta fase de pruebas donde se va a probar múltiples herramientas.

Una vez creado el entorno de trabajo, la primera tarea que se decidió abordar fue investigar las posibles herramientas a utilizar para llevar a cabo la descarga de datos de la página web de Instagram. Tras hacer varias pruebas con múltiples scrapers web encontrados en GitHub como `instagram-crawler`, `huaying-instagram-crawler`, etc. se decidió emplear

`Instalooter`, el cual dispone tanto de una interfaz de programación para Python 3 como de un cliente de línea de comandos. El resto de scrapers encontrados no parecían funcionar en el momento en que se hizo la prueba, posiblemente debido a que la web de Instagram hubiese sido actualizada de forma reciente y éstos programas no hubiesen sido actualizados de acorde a los nuevos cambios.

Como se comentó en la sección 4.1, Instalooter pone a disposición del usuario de varios *looters*, `ProfileLooter` y `HashtagLooter`, para descargar las publicaciones relacionadas con un perfil de usuario o con un Hashtag respectivamente, y permite descargar tanto los vídeos, como las imágenes y la descripción y meta-datos de las mismas en formato JSON. Una de las ventajas que tiene esta herramienta en comparación con otras es que permite comprobar si una publicación ha sido ya descargada o no para evitar repetirla, así su puede ejecutar la descarga de forma sucesiva sin riesgo de acabar con datos duplicados.

Durante las pruebas con Instalooter se pudo descargar con éxito tanto las imágenes como los meta-datos de varias publicaciones de Instagram. El JSON generado por cada publicación mediante esta herramienta cuenta con la siguiente información:

Nombre del Campo	Ejemplo	Descripción
___typename	`"GraphImage"`	Indica el tipo de publicación que se ha descargado, exactamente si es un vídeo o una imagen.
comments _disabled	`false`	Si los comentarios de la publicación se encuentran habilitados o no.
dimensions	`{"height": 640,"width": 640}`	Anchura y altura de la imagen.
display_url	`"https://scontent-ams2-1.cdninstagram.com/v/t51.2885-15/..."`	URL de descarga de la imagen en Instagram, este link expira a los pocos días.
edge_liked_by	`{"count": 73}`	Número de "likes" de la publicación.
edge_media_to_caption	`{"edges": [{"node": { "text": "El peso de una nube de color gris" }}]}`	comentario o descripción de la publicación generada por su autor.

edge_media_to_comment	`{"count": 1}`	Número de comentarios de la publicación.
id	`"1001662920005405392"`	Número identificador de la publicación.
is_video	`false`	Si la publicación contiene un video o no.
owner	`{"id": "216171681"}`	Identificador de la cuenta de instagram que hizo la publicación.
shortcode	`"3mnx5jmkrQ"`	Shortcode de la publicación, se puede usar para obtener un link permanente a la publicación.
taken_at_timestamp	`1433627546`	Timestamp de la publicación.
thumbnail_src	`"https://scontent-ams2-1.cdninstagram.com/v/t51.2885-15/..."`	URL del "thumbnail" de la imagen, este link expira a los pocos días.

Tabla 5.2: Contenido del JSON de Instalooter

Después de comprobar que se puede descargar con éxito publicaciones de instagram, se procedió a comprobar si es posible extraer información de las imágenes descargadas. Para ello se trató desde un principio de recurrir a herramientas ya existentes de reconocimiento de imágenes, preferiblemente de servicios en la nube.

Inicialmente por recomendación del tutor se trató de emplear los servicios de Google Cloud. Esta plataforma de Google Cloud provee diversos servicios junto con sus respectivas interfaces de programación compatibles con múltiples lenguajes de programación. Entre los servicios relevantes para el presente proyecto cabe destacar el almacenamiento en la nube con Cloud Storage, servicios de procesamiento de datos y creación de máquinas virtuales con Compute Engine, servicios de visión artificial mediante Cloud Vision, almacenamiento de bases de datos en Cloud SQL, etc.

Todos los servicios de Google Cloud se pueden probar de forma gratuita con tan solo crearse una cuenta en la plataforma. Esto es debido a que Google ofrece un crédito de 300 USD para invertir libremente en su plataforma a toda cuenta recién creada durante un periodo de 3 meses, 400 USD si se demuestra ser alumno [17]. Este crédito inicial se va gastando según se vaya haciendo uso de los servicios, y una vez consumido o pasado el periodo de

3 meses, Google empieza a cobrar por el uso de sus servicios. Debido al alcance de este proyecto y los precios que tienen los servicios de Google, el anterior límite de dinero no es preocupante, aunque el de tiempo si que podría llegar a serlo.

Como se ha comentado, para emplear los servicios de Google Cloud, Google pone a disposición de los usuarios APIs de forma gratuita para distintos lenguajes de programación, entre los cuales se encuentra Python. Mediante este lenguaje de scripting y la API de Cloud Vision [16], se llevaron a cabo unas pruebas iniciales donde se pudo comprobar que la información que se puede obtener mediante el uso de la API de Google Cloud Vision si que incluye cierta capacidad de análisis de emociones, permitiendo obtener etiquetas de emociones como alegría, tristeza, enfado, sorpresa, etc. junto con su *likelihood*, pero no incluye información como edad y género de las personas, capacidad que fue eliminada de este servicio de forma bastante reciente [14]. Esto es un problema ya que se pretendía hacer uso de este tipo de información a la hora de recomendar publicaciones o lugares al usuario.

Debido a las anteriores limitaciones de Cloud Vision, y el problema del límite de tiempo del periodo de prueba, se decidió explorar otras alternativas. A continuación se hace un pequeño resumen de los servicios ofrecidos por las distintas alternativas valoradas y que posiblemente se puedan llegar a necesitar en el proyecto.

Servicios	Límite de uso	API de Visión Artificial			Almacenamiento en bucket/blobs gratuito	Almacenamiento en Base de Datos SQL	Almacenamiento en Base de Datos NoSQL	Máquinas Virtuales
		Límite de uso	Análisis de Sentimiento	Reconocimiento de edad y género				
Google Cloud	300 USD durante 3 meses (100 USD más si se es alumno)	1000 unidades al mes	Si, con tags alegría, tristeza, enojo, sorpresa	No	Si, con 5 Gbs al mes en Google Cloud Storage	Si, con motor de base de datos MySQL, PostgreSQL y SQL Server	Si, bases de datos documentales como Cloud Datastore, Cloud Firestore... y clave/valor con MongoDB, Bigtable...	Si, Máquina virtual en Compute Engine empleando el saldo inicial
Microsoft Azure	12 meses para servicios gratuitos + 200 USD para servicios no gratuitos durante 1 mes	Hasta 30.000 transacciones al mes	Si, con tags felicidad, tristeza, neutralidad, ira, desprecio, asco, sorpresa y temor	Si	Si, con 5 Gbs en Azure Blob Storage y 5 GBs en Azure File Storage 5	Si, con Azure Database for MySQL (750 horas al mes), Azure Database for PostgreSQL (750 horas al mes) y SQL Database (250 Gbs)	Si, Azure Cosmos DB (25 Gbs)	Si, Máquina virtual B1S (1 núcleo, 1 GB RAM y 4GB de almacenamiento) con Linux 750 horas al mes
Amazon Web Services (AWS)	12 meses para la capa gratuita	5000 imágenes al mes	Si, con tags feliz, triste, enfado, confuso, disgustado, sorprendido, calma, miedo, desconocido	Si	Si, con 5 Gbs al mes en Amazon S3	Si, se puede usar RDS (750 horas al mes) con motor de base de datos MySQL, MariaDB, Oracle, PostgreSQL, SQL Server y Amazon Aurora	Si, bases de datos documentales con DocumentDB (compatible con MongoDB, solo 1 mes de prueba) y clave/valor con DynamoDB (25 GB gratuitos)	Si, Máquina virtual en EC2 (1 CPU, 1GB de RAM, 1 CPU y 1GB de RAM, 30 GBs SSD) con Linux 750 horas al mes

Tabla 5.3: Servicios en la Nube: Alternativas

Cabe destacar que se seleccionaron estas alternativas en base a su popularidad, los servicios potencialmente necesarios, las restricciones de tiempo para probarlos y al requerimiento de que la plataforma usada fuera gratuita u ofreciese un periodo de prueba, ya sea para estudiantes o no, lo suficientemente extenso como para poder desarrollar el proyecto sin complicaciones. También hay que destacar que se intentó llevar a cabo pruebas con las distintas plataformas. Durante estas pruebas preliminares, con Microsoft Azure no se consiguió llegar a crear una máquina virtual de tipo `B1S` gratuita, puesto que siempre salían como no disponibles. Tras revisar la página de preguntas y respuestas de Microsoft parece que a mucha gente le ha sucedido lo mismo recientemente, el problema parece ser la alta demanda de este tipo de máquinas. Debido a este problema y la imposibilidad de obtener la edad y el género en los servicios de Google Vision, finalmente se decidió que lo mejor era emplear los servicios de **Amazon Web Services**.

Un vez tomada la decisión de la plataforma a utilizar, se procedió a hacer más pruebas con AWS. Para interactuar con cualquier servicio de Amazon Web Services, sobre todo si se hace a través de su SDK, es necesario obtener un identificador y su clave de acceso. Por lo tanto, una vez creada la cuenta lo primero que se hizo fue generar estas credenciales a través del servicio IAM (Identification and Access Management), en su apartado *Credenciales de Seguridad*. Una vez generado, nos devolverá un CSV con dos valores, `AWSAccessKeyId` y `AWSSecretKey`, los cuales deberemos conservar para acceder posteriormente a los servicios.

Después de generar las credenciales se procedió a probar el servicio de Amazon Rekognition, cuyos detalles ya se introdujeron en el apartado 4.2. Para poder analizar una imagen con Rekognition, primero ésta debe de estar subida a un bucket o contenedor de objetos de Amazon S3, por ello se procedió a generar un bucket de prueba y a subir una serie de imágenes a través de la consola web. Por otro lado, para llevar a cabo pruebas con Rekognition hubo que instalar el SDK de AWS para Python, el cual es conocido como `Boto3`. Su uso a través de Python es muy sencillo, puesto que se puede acceder a cualquier servicio de AWS con tan solo usar la función `boto3.client` empleando como parámetros: el nombre del servicio a usar, la región donde se ejecuta en el servicio, que en el caso de este proyecto siempre se ha escogido "eu-west-1" (Irlanda), y el id y clave de acceso que anteriormente se obtuvieron mediante IAM. Durante las pruebas con Rekognition se pudo comprobar que mediante su funcionalidad de reconocimiento facial se puede obtener un JSON que contiene un array "FaceDetails" con un objeto por cara presente en la imagen. Cada uno de estos objetos cuenta con los siguientes detalles:

Nombre del Campo	Ejemplo	Descripción
BoundingBox	`{"Width": 0.0358, "Height": 0.0476, "Left": 0.3780, "Top": 0.6708}`	Coordenadas de la caja donde se ubica la cara.
AgeRange	`{"Low": 49, "High": 57}`	Rango de edad de la persona en años.
Smile	`{"Value": false, "Confidence": 94.6054}`	Si la persona se encuentra sonriendo o no.
Eyeglasses	`{"Value": false, "Confidence": 97.4112}`	Si la persona lleva o no gafas o no.
Sunglasses	`{"Value": false, "Confidence": 99.9965}`	Si la persona lleva o no gafas de sol o no.
Gender	`{"Value": "Female", "Confidence": 68.2432}`	Género de la persona.
Beard	`{"Value": false, "Confidence": 62.0366}`	Si la persona tiene barba o no.
Mustache	`{"Value": false, "Confidence": 94.0625}`	Si la persona tiene bigote o no.
EyesOpen	`{"Value": false, "Confidence": 99.8537}`	Si la persona tiene los ojos abiertos o no.
MouthOpen	`{"Value": false, "Confidence": 93.5364}`	Si la persona tiene la boca abierta o no.
Emotions	`[ {"Type": "SAD", "Confidence": 99.9044}, {"Type": "CALM", "Confidence": 23.9581}... } ]`	Un array con la confianza de las emociones expresadas por la persona basada en su expresión facial. Los posibles valores son triste, calmado, sorprendido, aterrado, confundido, enfadado, feliz, disgustado.
Landmarks	`[{"Type": "eyeLeft", "X": 0.3885, "Y": 0.6877},{"Type": "eyeRight", "X": 0.4037, "Y": 0.6898}...]`	Coordenadas de las características de la cara (nariz, boca, etc.).
Pose	`{"Roll": 8.8927, "Yaw": 1.8882, "Pitch": 0.5692}`	Pose (rotación) de la cara.
Quality	`{"Brightness": 63.9795, "Sharpness": 4.3748}`	Indican el brillo y nitided de la imagen.
Confidence	`99.8897`	Nivel de confianza en el que se encuentra una cara dentro de la caja.

Tabla 5.4: Contenido del JSON de Rekognition para detección facial

Una vez elegidas las herramientas a utilizar, comprobado su funcionamiento y los datos que se pueden obtener, se puede concluir que es posible llevar a cabo la descarga de datos de Instagram y el procesamiento de las imágenes.

5.2. Diseño e implementación de la base de datos

En este apartado se explica la base de datos diseñada, y los scripts implementados para la descarga, procesado y almacenamiento de los datos generados mediante Instalooter y Rekognition de forma automática.

Como se ha comentado en el apartado anterior, se va a emplear Instalooter para la descarga de publicaciones de Instagram. Desde un principio se decidió emplear una lista de hashtags de los cuales descargar publicaciones relacionadas con Valladolid, ya que estas suelen estar etiquetadas con hashtags relacionados a eventos o lugares de la ciudad. De esta lista se hablará posteriormente. Por lo tanto, en nuestros scripts de descarga se va a emplear el looter `HashtagLooter` para este cometido, limitando la descarga de publicaciones solo a imágenes.

Una vez que se obtengan las publicaciones, antes de almacenar cualquier información en la base de datos se va a llevar a cabo el procesamiento de las imágenes mediante Rekognition para así tener la información completa. Pero como se comentó anteriormente, para ello primero las imágenes han de estar subidas a Amazon S3. Por lo tanto, primero se ha de conseguir subir las imágenes descargadas por Instalooter a Amazon S3 de forma automática. Esto inicialmente podría parece una tarea sencilla, solo tendríamos que descargar la imagen y meta-datos de una publicación de forma local, subirlo mediante boto3 a Amazon S3, y borrarlo del almacenamiento local. Pero como se comentó en la sección 4.1, Instalooter antes de descargar una publicación primero comprueba si ya existe en el directorio donde se pretende guardar, para así acelerar el proceso de *scraping* y poder ejecutar Instalooter sobre un hashtag o perfil sucesivamente evitando tener publicaciones duplicadas. Esto es un problema ya que nos obliga a que el equipo que ejecute Instalooter tenga que mantener almacenadas todas las publicaciones ya descargadas, teniendo los datos duplicados en S3 y en el equipo, y dependiendo del volumen de publicaciones esto puede dar problemas de escalabilidad.

Tras revisar el código fuente de Instalooter se pudo comprobar se puede pasar una objeto que herede de la interfaz `fs.base.FS` de *PyFilesystem* en

vez de una ruta al sistema de archivos local, aunque esto no se encuentra documentado en su página web. Este objeto permite abstraerse del sistema de archivos local, y por ello, se pensó que tal vez se podría usar un objeto implementase esta interfaz junto con las comunicaciones con Amazon S3, para así no tener que usar el almacenamiento local del equipo. Mediante el uso de este objeto, Instalooter descargaría directamente las publicaciones en un bucket, y podría comprobar en ese contenedor si la publicación ya existe.

Gracias a GitHub se consiguió encontrar un proyecto que hace exactamente lo anterior llamado S3FS. Tras una serie de pruebas, se pudo comprobar que esta librería funciona, pudiendo acceder a un bucket de prueba y subir ficheros sin mayor problema. Después de estas pruebas también se intentó probar si la idea inicial de usar Instalooter junto con una instancia que implementase la interfaz de *PyFilesystem* era correcta, y se pudo comprobar que se estaba en lo cierto, descargando las publicaciones al bucket sin duplicados.

Con las imágenes ya guardadas en el bucket, el proceso de llevar a cabo el reconocimiento facial, fue relativamente simple. Solo hubo que iterar sobre las publicaciones almacenadas en el contenedor y guardar en el propio bucket el JSON generado por Rekognition, siempre comprobando antes que esa publicación no hubiese sido ya procesada mediante este servicio. De este modo, por cada publicación se tiene almacenado en Amazon S3 tres ficheros: la imagen de la publicación con el nombre `<id>.jpg`, el JSON con los meta-datos de la publicación llamado `<id>.json` y el JSON con la información generada por Rekognition llamado `<id>_rek.json`, siendo `<id>` el id de la publicación devuelto por Instalooter.

Una vez se ha obtenido toda la información necesaria y se ha almacenado en S3, para poder usarla de una forma más cómoda se decidió almacenarla en una base de datos. En este caso se decidió emplear Amazon DynamoDB, una base de datos documental (NoSQL). La decisión de emplear esta base de datos se debe a múltiples motivos. Primero, respecto al resto de bases de datos ofrecidas por AWS, ésta es la que ofrece más capacidad de almacenamiento de forma gratuita (25 GBs) y además para siempre, a diferencia de las bases de datos con DocumentDB que tan solo se puede usar por un mes, y de las bases de datos SQL que solo pueden ser usadas por 12 meses. Por otro lado, esta base de datos se caracteriza por su alta disponibilidad, y por su gran capacidad de escalar horizontalmente, algo que no suele ser posible en las bases de datos SQL.

A la hora de diseñar la base de datos inicialmente se planteó tener dos tablas:

1. La primera tabla contendría la información obtenida mediante Instalooter. Exactamente se plantearon los siguientes campos del JSON:

   - id
   - timestamp
   - shortCode
   - displayUrl
   - description
   - likesCount
   - commentsCount

2. La segunda tabla contendría la información obtenida mediante Rekognition. Exactamente se plantearon almacenar por cada cara existente en el array "FaceDetails" del JSON los siguientes campos:

   - confidence
   - ageLow
   - ageHigh
   - gender
   - eyeglasses
   - sunglasses
   - beard
   - moustache
   - happyConfidence
   - surprisedConfidence
   - fearConfidence
   - sadConfidence
   - angryConfidence
   - disgustedConfidence
   - confusedConfidence
   - calmConfidence

   Además de los campos anteriores, a esta tabla se tendría que añadir primero un campo que actuara como identificador único por cada entrada en la tabla, y segundo el valor del id que se usó como nombre de fichero del JSON y que coincide con el id de Instalooter, para poder tener una clave foránea que apunte al campo id de la primera tabla.

Este diseño sería correcto si trabajásemos con una base de datos SQL. El problema es que al trabajar con DynamoDB, a pesar de que el uso de esta base de datos es muy sencillo y ofrece muchas posibilidades, a su vez es algo limitado. A diferencia de las bases de datos SQL, en esta base de datos no es posible llevar a cabo ciertas operaciones complejas como *join* de tablas, ni tampoco agregaciones. Está pensado únicamente para accesos clave-valor, y optimizar cierto tipos de consultas o patrones de acceso que pueda tener una aplicación. Otro detalle a comprender respecto a DynamoDB es la forma en que se identifican los elementos en las tablas. En DynamoDB existen dos tipos de claves primarias:

1. Partition Key: Se emplea una clave de particionamiento para dividir el conjunto de datos. Sobre esta clave se aplica una función hash que determina el lugar donde se almacenará el elemento. El atributo que se emplee como Partition Key no puede repetirse, ha de ser único por cada elemento.

2. Partition Key + Sort Key: Es una clave compuesta por dos atributos, el atributo que forma parte de la Partition Key determina el lugar donde se almacenará el elemento, y todos los elementos que formen parte de una misma partición se guardarán ordenados en base al atributo que forma parte de la Sort Key. Con este tipo de clave compuesta, el atributo que forme parte de la Partition Key puede repetirse para varios elementos, pero la combinación de Partition Key y Sort Key ha de ser única.

Teniendo en cuenta estos detalles sobre DynamoDB, para implementar la relación de *one-to-many* de las tablas planteadas anteriormente, en DynamoDB existen varias opciones:

1. Una única tabla con una clave primaria compuesta por el id de Instalooter como Partition Key y el índice de la cara en el array "FaceDetails" como Sort Key para garantizar que cada entrada de la tabla sea única. Los datos obtenidos con Instalooter irían duplicados por cada cara obtenida mediante Rekognition.

2. Una única tabla con una clave primaria compuesta por el id de Instalooter como Partition Key y el índice de la cara en el array "FaceDetails" como Sort Key para garantizar que cada entrada de la tabla sea única. Los datos obtenidos con Instalooter irían duplicados por cada cara obtenida mediante Rekognition, pero se metería el JSON de Instalooter entero en un atributo "complejo" en vez de generar atributos a partir de él.

3. Una única tabla con una clave primaria compuesta por el id de Instalooter como Partition Key, y una Sort Key generada a partir del índice de la cara en el array "FaceDetails" y un prefijo que indique si la entrada proviene de Instalooter o Rekognition.

Cada opción tiene sus ventajas e inconvenientes. Las dos primeras opciones tienen el inconveniente de la duplicidad de datos, esto por un lado puede dar problemas de consistencia si se tienen que modificar datos aunque esto

no aplicaría en nuestro caso puesto que no se planea alterar los contenidos de la base de datos, y por otro que tiene un mayor coste de almacenamiento. La segunda opción además tiene el inconveniente de no poder acceder a los atributos de Instalooter directamente en la consulta y que cada atributo de DynamoDB está limitado a 400KBs. La tercera opción tiene el inconveniente de que como no existen operaciones de agregación, no es posible usar esta tabla si queremos recuperar los datos de Instalooter junto con los de Rekognition en una misma consulta. Finalmente se decidió que la opción que más ventajas tenía en este caso era la primera opción, ya que se pueden acceder a todos los campos en una misma consulta. Por lo tanto, la tabla final, a la que se llamó `valltourisminsta`, tiene la siguiente forma:

Nombre del campo	Tipo	Orígen
id	Cadena	Instalooter.id
faceIndex	Número	Índice de la cara en Rekognition.FaceDetails
dTime	Cadena	Instalooter.taken_at_timestamp, formateado como `YYYY-MM-DDTHH:MM:SS`
shortCode	Cadena	Instalooter.shortcode
displayUrl	Cadena	Instalooter.display_url
description	Cadena	Instalooter.edge_media_to_caption.edges[0].node.text
likesCount	Número	Instalooter.edge_liked_by.count
commentsCount	Número	Instalooter.edge_media_to_comment.count
confidence	Número	Rekognition.FaceDetails[*].Confidence
ageLow	Número	Rekognition.FaceDetails[*].AgeRange.Low
ageHigh	Número	Rekognition.FaceDetails[*].AgeRange.High
gender	Cadena	Rekognition.FaceDetails[*].Gender.Value
eyeglasses	Boolean	Rekognition.FaceDetails[*].Eyeglasses.Value
sunglasses	Boolean	Rekognition.FaceDetails[*].Sunglasses.Value
beard	Boolean	Rekognition.FaceDetails[*].Beard.Value
moustache	Boolean	Rekognition.FaceDetails[*].Mustache.Value
happyConfidence	Número	Rekognition.FaceDetails[*].Emotions. HAPPY.Confidence
surprisedConfidence	Número	Rekognition.FaceDetails[*].Emotions. SURPRISED.Confidence
fearConfidence	Número	Rekognition.FaceDetails[*].Emotions. FEAR.Confidence
sadConfidence	Número	Rekognition.FaceDetails[*].Emotions. SAD.Confidence

angryConfidence	Número	Rekognition.FaceDetails[*].Emotions. ANGRY.Confidence
disgustedConfidence	Número	Rekognition.FaceDetails[*].Emotions. DISGUSTED.Confidence
confusedConfidence	Número	Rekognition.FaceDetails[*].Emotions. CONFUSED.Confidence
calmConfidence	Número	Rekognition.FaceDetails[*].Emotions. CALM.Confidence

Tabla 5.5: Diseño de la tabla `valltourisminsta`

Una vez diseñada la base de datos que se empleará para almacenar las publicaciones, se procedió a implementarla. Para ello, al igual que con Rekognition, se empleó el SDK de Amazon Web Services, en este caso usando el servicio "dynamodb" de `boto3` y su función `create_table` con los parámetros necesarios. Una vez creada, poblar la base de datos es muy sencillo, tan solo hay que obtener todas las publicaciones que estén almacenadas en S3 que tengan el JSON de Instalooter y de Rekognition, y que no se encuentren ya en la base de datos. Para ello, solo hay que listar los elementos que hay en el bucket mediante la librería S3FS que se usó junto con Instalooter para subir las publicaciones, parsear y extraer los campos que queramos de los JSONs, y mediante el servicio "dynamodb" de `boto3` llamar a la función `put_item` para insertar cada elemento en la tabla `valltourisminsta`.

Finalmente hay que decir que todos los pasos que se han explicado a lo largo de este apartado han sido implementados en un único script Python llamado `mainScriptAWS.py`. Este script se encuentra completamente automatizado, de forma que se puede ejecutar sin la intervención del usuario. Así, si fuese necesario se podría crear un *cron* que ejecutase periódicamente el script. En resumen, el script ejecuta la siguiente secuencia:

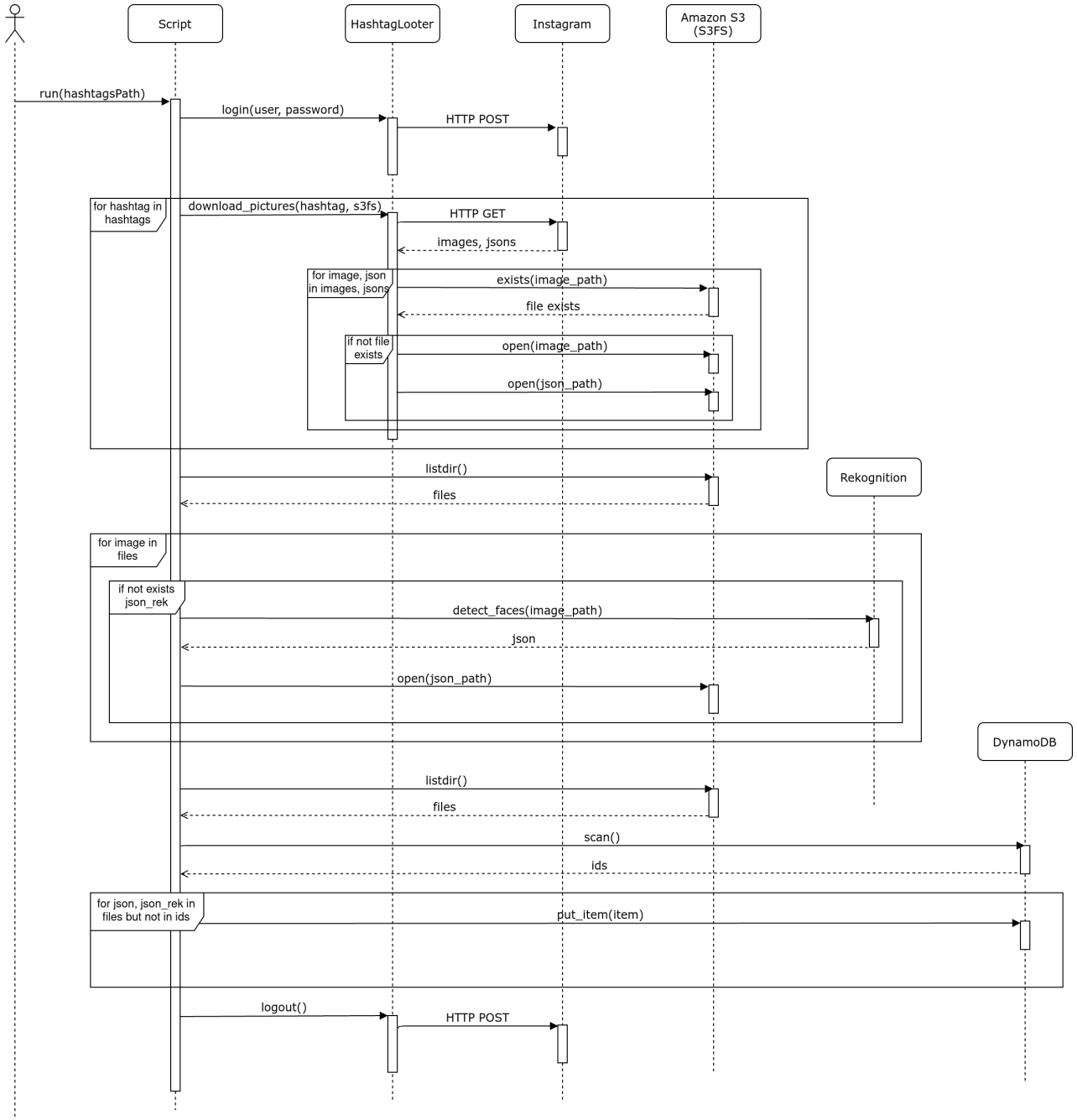

Figura 5.1: Diagrama de secuencia de `mainScriptAWS.py`

Como se puede ver en el diagrama anterior, el script tan solo toma como argumento la ruta del fichero que contiene la lista de hashtags a consultar en Instagram (un hashtag por línea). El resto de identificadores y credenciales

para acceder a Instagram o a los servicios de AWS han de estar definidos como variables de entorno.

5.3. Proceso de descarga de las publicaciones

Para llevar a cabo el proceso de descarga de publicaciones y el poblado de la base de datos se tuvo que ejecutar el script de la sección anterior en varias ocasiones. El entorno de ejecución desde el que puso en funcionamiento siempre fue una máquina virtual, con Python 3, `virtualenv` y todas las librerías y SDKs instalados. Esta maquina virtual es una de las instancias que Amazon Web Services permite crear en su capa gratuita mediante su servicio EC2. Exactamente, la configuración escogida para esta instancia es la siguiente: 1 CPU `t2.micro`, 1 GB de RAM, 30 GBs de almacenamiento SSD, 100 GBs de ancho de banda con IP estática y el sistema operativo Debian 11 de 64 bits.

Sobre la lista de hashtags que se pasa al script como argumento, inicialmente se decidió obtener un número cercano a 50 hashtags relacionados con Valladolid y el turismo en Valladolid. Esta lista habría que obtenerla manualmente buscando en Instagram hashtags populares dentro de este contexto así como las etiquetas de las publicaciones encontradas. La lista finalmente creada consta de los siguientes 44 hashtags:

- Naturaleza_valladolid
- estaes_valladolid
- valladolidturismooficial
- turismovalladolid
- valladolidturismo
- turismovalladolid
- valladolidspain
- valladolidfotos
- valladolidenfotos
- valladolid
- todo_valladolid
- total_valladolid
- igersvalladolid
- igervalladolid
- catedraldevalladolid
- paseandoporvalladolid
- valladolidgram
- instavalladolid
- valladolidlove
- valladolidlife
- valladolidmola
- megustavalladolid
- fotosdevalladolid
- valladolidhoy
- vallafotos
- valladolidinquieta
- love_valladolid
- visitvalladolid
- look_valladolid
- ok_valladolid

- megustavalladolid
- asi_es_valladolid
- pucela
- megustapucela
- arquitecturavalladolid
- valladolidespaña
- culturavalladolid
- valladolidcapital
- valladolidcampogrande
- valladolidpaseozorrilla
- iglesiadelaantiguavalladolid
- igerspucela
- rinconesdevalladolid
- megustapucela

A la hora de ejecutar el script un detalle a tener en cuenta es el límite de publicaciones que podemos descargar y analizar en la capa gratuita. Al mes, en Rekognition se pueden analizar hasta 5000 imágenes, en DynamoDB se tiene un límite de 200 millones de solicitudes y en S3 existe un límite de 2000 solicitudes *put*, siendo este último punto el mayor factor limitante. Debido a esto y al límite de la fecha de entrega del proyecto, en la base de datos finalmente generada se descargaron un total de 3916 publicaciones de Instagram. Tras el análisis con Rekognition, a partir de estas publicaciones se extrajeron unas 5408 caras.

Un detalle a destacar fue que, durante una de las ejecuciones del script, se detectó un problema a la hora de iniciar sesión en Instagram mediante Instalooter, impidiendo llevar a cabo el proceso de descarga de publicaciones. Tras revisar el error generado por el scraper y comprobar que este no era puntual, se llegó a la conclusión de que debió de haber alguna actualización en la web de Instagram que provocó que la librería dejara de funcionar correctamente. Revisando la pestaña de problemas en el repositorio de GitHub de la librería, se observó que es un error que le sucedía a más usuarios.

Como las últimas actualizaciones del repositorio de Instalooter fueron hace varios meses en el momento en que se hizo la comprobación, se decidió probar de nuevo otras librerías como *Instaloader*, pero se llegó a la conclusión de que su uso era muy distinto a Instalooter y no ofrecía las ventajas de poder comprobar publicaciones ya descargadas y poder conectarlo de forma sencilla a los servicios en la nube. Es por ello que se decidió revisar el código fuente de la librería, y tras unas pruebas se consiguió dar con la fuente del error: un cambio en como se gestionaban las *cookies* donde se guardan los *tokens* empleados para logearse en la web. Como el cambio no era muy complicado se decidió corregirlo en un fork `https://github.com/Zalez95/InstaLooter.git` de la librería, y tras comprobar que funcionaba correctamente, se instaló mediante `pip3` en el entorno de la máquina virtual. Tras está corrección no volvió a dar problemas, y de hecho se hizo un *pull request* a la librería original que en la actualidad está en proceso de revisión.

5.4. Visualización

Una vez se han obtenido los datos necesarios para el proyecto, para poder usarlos de forma más cómoda se decidió que era necesario implementar algún tipo de visualización. Esta visualización debería permitir filtrar los datos de forma sencilla de acuerda a criterios como edad o género, y permitir seleccionar aquellas publicaciones que tuvieran un sentimiento positivo. También debería poder llevarnos fácilmente hasta las publicaciones mostradas en Instagram.

De acuerdo a las necesidades anteriores se llegó a la conclusión de que la forma más óptima de visualizar los datos era llevar a cabo algún tipo de cuadro de mandos. Para implementar un cuadro de mandos sobre una base de datos de DynamoDB se encontraron varias posibilidades:

1. QLik: herramienta para crear dashboards con la que se ha trabajado anteriormente en este mismo máster, exactamente en la asignatura de Inteligencia de Negocio. No es una herramienta gratuita ni software libre, pero ofrece un periodo de prueba. El problema es que no se puede usar directamente con DynamoDB, sino que hay que usar un conector ODBC de la empresa Simba que no es gratuito.

2. Microsoft PowerBI: Al igual que en la herramienta anterior, no es gratuito pero ofrece un periodo de prueba. Tampoco tiene conectores con DynamoDB, habría que usar el conector ODBC de Simba anteriormente comentado.

3. Tableau: Ofrece un periodo de *trial* de 3 meses, además existe una versión *public* gratuita. Para usarlo junto con DynamoDB habría que usar el conector JDBC de Rockset que tampoco es gratuito (hay periodo de prueba pero no se indica durante cuanto tiempo).

4. Grafana: Es open source pero no tiene conector gratuito con DynamoDB. Parece que tiene plugins que permiten conectarlo con servicios REST de forma sencilla. Se podría crear un conector creando un servidor que implementara estas APIs REST para acceder a DynamoDB como se hace en [9].

5. Web estática junto con alguna librería para hacer gráficos: En esta opción habría que crear una página web estática manualmente y programar los gráficos en JavaScript mediante alguna librería como ChartJS o D3.js, conectándose con la base de datos a través del

SDK de AWS para este lenguaje de programación. Un ejemplo de implementación se puede encontrar en [13].

Finalmente se decidió emplear Grafana aunque haya que implementar el conector, puesto que al ser open source no se tiene restricciones de tiempo de uso. Crear una página web estática desde cero o usando librerías consume mucho tiempo, y el resto de herramientas para hacer cuadros de mando también presentan el mismo problema a la hora de conectar con DynamoDB.

Para crear el cuadro de mandos con Grafana hubo que instalar Grafana Enterprise. En este caso se decidió hacerlo en la misma máquina virtual que se usa también para la ejecución del script de la sección anterior. Grafana es una aplicación web que por defecto se ejecuta en el puerto 3000, es por ello que para acceder desde fuera de este equipo hubo que abrir el puerto añadiendo una regla TCP en los grupos de seguridad de Amazon EC2. Una vez instalado Grafana y con el servicio corriendo, el siguiente paso fue instalar el plugin `JSON` para conectarse con fuentes de datos (*data sources*) a través de HTTP. Para que este plugin funcione, el servidor al que se conecte ha de implementar una API REST con las siguientes operaciones:

- `GET /` → Ha de devolver un código 200 OK si el servicio funciona correctamente.
- `POST /query` → Ha de devolver los valores pedidos. Para ello esta función tiene como parámetros un rango de fechas en el que se tienen que encontrar los datos, y da la posibilidad de filtrar por más valores mediante unos *adhoc filters*. Los valores devueltos pueden estar en formato de tabla o en formato clave-valor, siendo la clave una fecha y el valor el contenido de una columna.
- `POST /search` → Ha de devolver las distintas métricas para llevar a cabo el filtrado, en nuestro caso se corresponde con el nombre de las columnas, o los posibles valores si como parámetro llega el nombre de una columna.
- `POST /tag-keys` → Ha de devolver las posibles claves para llevar a cabo el filtrado, en nuestro caso se corresponde con el nombre de las columnas.
- `POST /tag-values` → Ha de devolver los posibles valores de una clave para llevar a cabo el filtrado, en nuestro caso se corresponde con los valores distintos que hay para una columna.

Para poder crear el conector entre Grafana y DynamoDB hubo que hacer un servidor web que implementara las anteriores APIs, en este caso mediante Python 3 y el SDK de AWS `boto3`. Las pruebas iniciales se hicieron mediante un framework llamado *bottle* para poder crear las APIs de forma local, y una vez que se comprobó que éstas funcionaban correctamente, se decidió implementar las APIs en Amazon Web Services para tener mejor rendimiento. Para implementar las APIs en AWS se usó sus servicios AWS Lambda, Amazon API Gateway y Amazon DynamoDB de la siguiente forma:

Figura 5.2: Acceso a DynamoDB mediante APIs REST con API Gateway

Siendo el diagrama de componentes del sistema completo el siguiente:

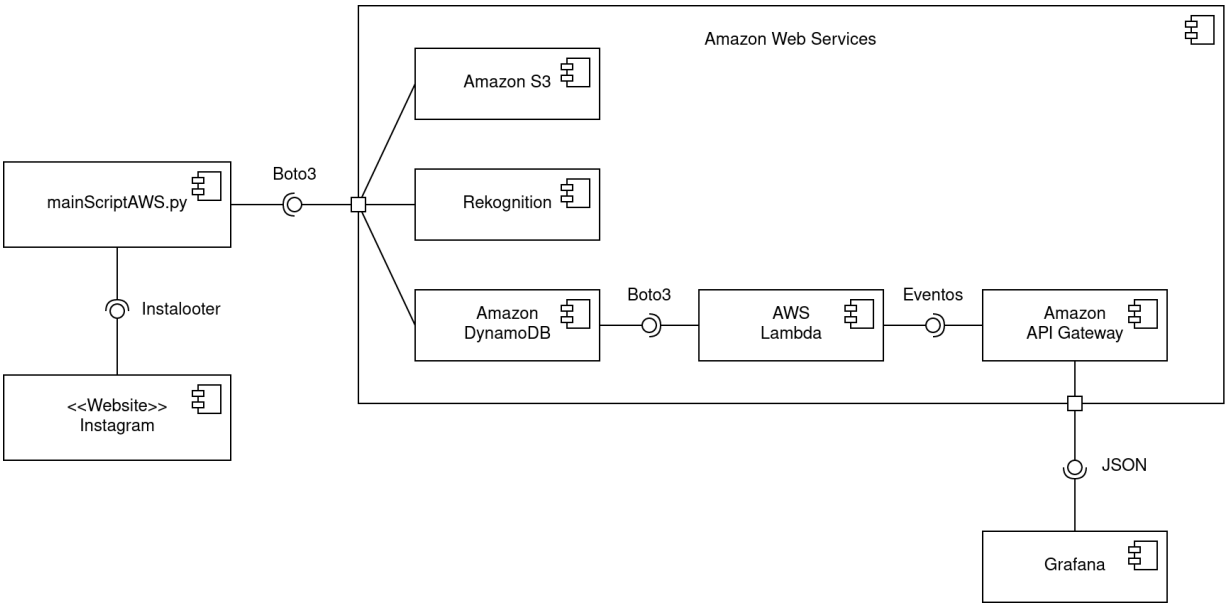

Figura 5.3: Diagrama de Componentes del sistema

AWS Lambda permite ejecutar código en la nube de Amazon sin tener que crear instancias de máquinas virtuales. En total se generaron 5 funciones para acceder a DynamoDB: `grafana_query`, `grafana_tagkeys`, `grafana_helloworld`, `grafana_tagvalues`, `grafana_search`, implementando el acceso a DynamoDB y la lógica correspondiente. Todas estas funciones se ejecutan en un entorno Python 3.8 en equipos x86_64, con un tiempo de ejecución de máximo 1 minuto. Por otro lado, API Gateway es el servicio que permite implementar de forma sencilla y escalable APIs RESTful o Websocket. En este caso las APIs REST se generaron como recursos que tienen asociado un método HTTP y un *endpoint*, y que al ser accedidas ejecutan su correspondiente función de AWS Lambda:

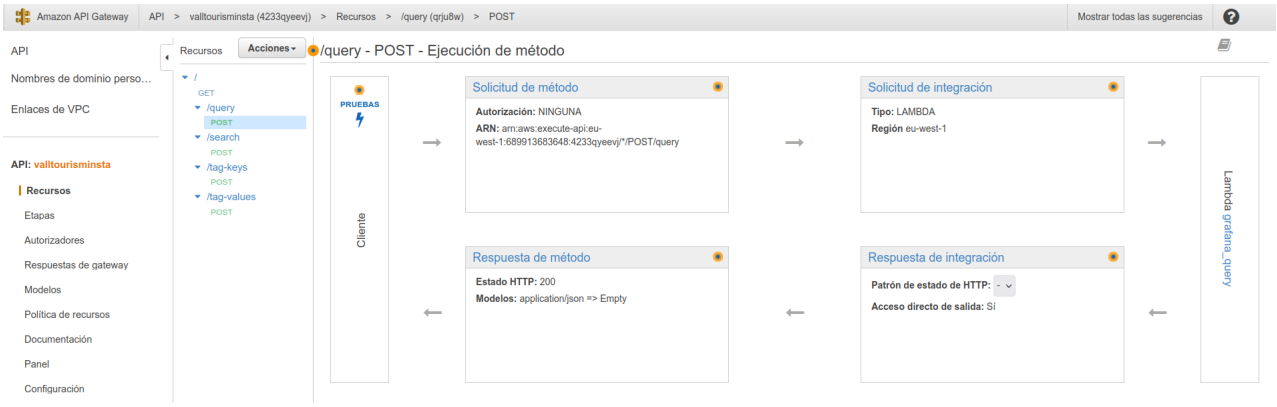

Figura 5.4: Configuración de AWS API Gateway y AWS Lambda

Una vez que se crearon las APIs, el siguiente paso fue emplearlas con Grafana. Para ello se generó un *data source* de tipo JSON que apuntaba a la URL del servicio de API Gateway. Tras conectar con éxito con la fuente de datos se procedió a la implementación del cuadro de mandos.

Para el diseño del dashboard se decidió emplear 3 columnas y 2 filas, ubicándose la información más relevante en la esquina superior izquierda puesto que está probado que las personas siguen un patrón en forma de Z a la hora visionar documentos o cuadros de mandos. El resultado final se puede ver en la figura 5.5.

En la esquina superior izquierda se muestra una combinación de estadísticas generales de las publicaciones mostradas, entre las que se encuentra el número de publicaciones mostradas, el número de caras y la proporción de genero de las caras mediante un gráfico de tarta. A su derecha se encuentra una tabla con información condensada de cada publicación, con un enlace

hacia la publicación en Instagram, su fecha, la confianza, el género y la edad. Después, en la esquina superior derecha se muestra el número de publicaciones por fecha en un histograma para así poder ver que días hubo más publicaciones.

En la siguiente fila se muestra, a la izquierda la emoción predominante en todas las publicaciones, para ello se calcula el promedio de la confianza de cada emoción de todas las publicaciones y se muestra en un gráfico de tarta. A su derecha, el "éxito" de las publicaciones, en un histograma que muestra el promedio del número de comentarios y de likes que hubo cada día. Finalmente a su derecha se muestra la emoción promedio de las publicaciones por día.

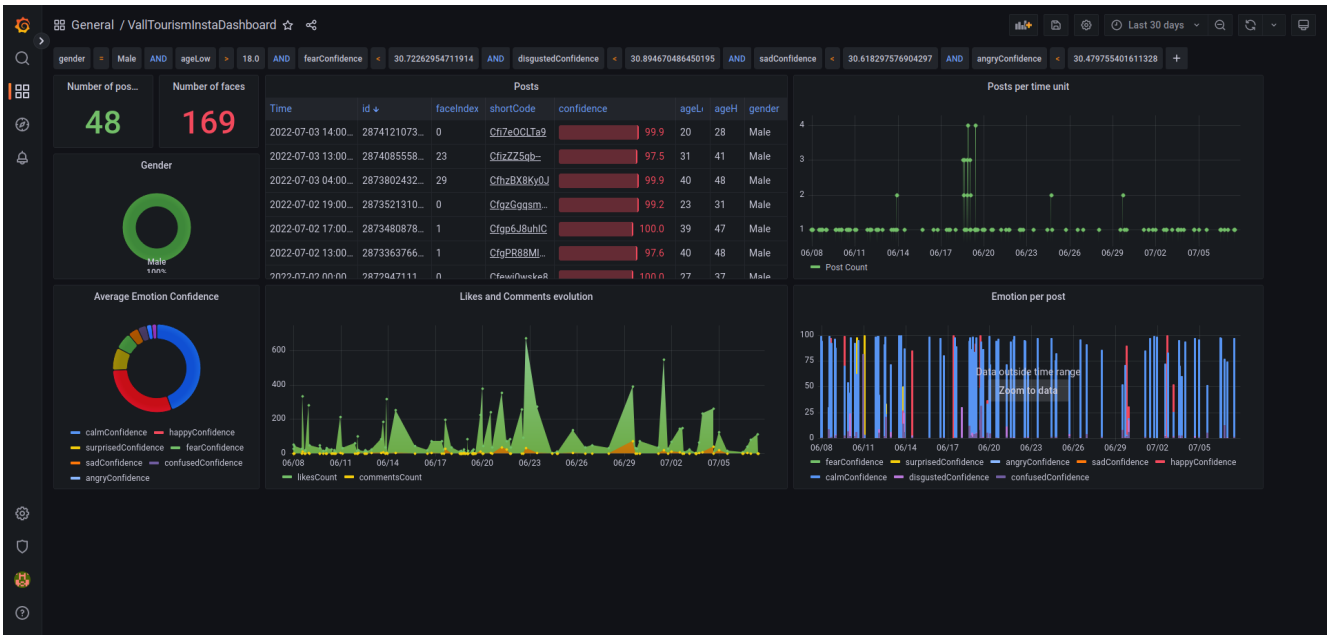

Figura 5.5: Cuadro de Mandos creado con Grafana

Cabe destacar que Grafana muestra en la parte superior los distintos filtros a aplicar, por un lado las fechas, y por otro los filtros *adhoc*, que pueden ser añadidos o modificados dinámicamente. En la captura anterior se puede ver que se están mostrando los últimos 30 días, y que se tienen habilitados varios filtros, exactamente género masculino, edad superior a los 18 años y se ha intentado que no haya emociones negativas predominantemente, descartando aquellas publicaciones que tengan una confianza superior al 30 % aproximadamente en las emociones triste, enfadado y disgustado.

5.5. Acceso al proyecto

Para concluir este apartado, se incluyen los enlaces a los distintos recursos desarrollados a lo largo del proyecto:

- Enlace de acceso público al repositorio del proyecto en GitHub.
- Enlace de acceso público al cuadro de mandos generado en la sección anterior, con los filtros indicados en la imagen 5.5.

Trabajos relacionados

En este apartado se va a exponer el estado del arte, o los trabajos encontrados relacionados con el tema del presente proyecto con el fin de conocer si ya existe algún trabajo anterior que intente resolver el mismo problema que el aquí presentado.

6.1. Metodología de trabajo

Durante este proceso de documentación, lo primero que se llevó a cabo fue una búsqueda preliminar con el fin de conocer si existía algún artículo previo donde se tratara en específico el análisis de emociones en imágenes de Instagram para recomendar posts o lugares a los que visitar. Para ello se hizo uso de bases de datos y exploradores gratuitos como Google Scholar, IEEE Xplore y ResearchGate. El resultado de esta búsqueda fue negativo, pero sí que se encontraron multitud de trabajos en los que se intentaba llevar a cabo análisis de sentimiento de comentarios de diversas redes sociales para saber la reacción de los usuarios que hacían turismo en cierto lugares en específico como [25], o trabajos generales sobre análisis de sentimiento para ayudar a recomendadores como [21] o [2].

Como este trabajo preliminar no fue satisfactorio, se decidió ampliar los criterios de búsqueda, llevando a cabo un proceso de *map-review* con el objetivo de recopilar los trabajos previos más relevantes para el presente proyecto. Debido a que la cantidad de artículos relacionados con análisis de sentimiento es abrumadora, para llevar a cabo el map-review se decidió emplear la metodología PRISMA (*Preferred Reporting Items for Systematic Reviews and Meta-Analyses*) inspirado por el artículo [28]. Esta metodología es empleada para llevar a cabo revisiones sistemáticas de conjuntos de

artículos, permitiendo descartar aquellos que no son relevantes sin tener que leerlos todos por completo, agilizando todo el proceso. En resumen, esta metodología cuenta de varias etapas:

1. Planificación de búsqueda: Hay que comprobar que no haya ya un trabajo previo que resuelva la misma pregunta. En nuestro caso como se comentó al inicio del apartado, aparentemente no parece haber un trabajo anterior similar.

2. Proceso de búsqueda: En esta fase hay que generar las cadenas de búsqueda que se emplearan en los buscadores de artículos. En nuestro caso se decidió emplear las siguientes cadenas:

   - `(Sentiment Analysis OR Emotion Recognition) AND Social Media AND Tourism`
   - `(Sentimen Analysis OR Emotion Recognition) AND Images AND Recommendation System`

   Además de las anteriores cadenas de búsqueda, se ha decidido también aplicar otros filtros a mayores como:

   - El acceso al artículo se ha de poder hacer de forma gratuita.
   - El artículo ha de ser "actual", de entre 2014 y 2022.

   De este modo se obtuvieron un total de 22 artículos a analizar en las siguientes fases.

3. Selección de muestras. En esta fase se parte de los artículos anteriormente obtenidos y, tras descartar aquellos que estén duplicados, se decide si descartar alguno tras leer el *abstract* basándose en una serie de criterios de inclusión y de exclusión. En este caso se ha decidido que:

   - Se incluye si:
     - El abstract está relacionado con análisis de sentimientos en redes sociales
     - El abstract está relacionado con análisis de sentimientos en imágenes
     - El abstract está relacionado con sistemas de recomendación de turismo
   - Se excluye si :

- El idioma del artículo no es inglés ni español.
- El artículo no se puede acceder gratuitamente
- El artículo es una recopilación de trabajos previos
- El artículo no está relacionado con análisis de sentimientos
- La fuente de datos del artículo no son redes sociales

Si tras leer el abstract es detectado algún criterio de exclusión o no cumple con ningún criterio de inclusión, el artículo es directamente es descartado. Después de descartar por el abstract, los artículos restantes son leídos por completo y se decide si incluirlos o no basándose en un test con las siguientes preguntas binarias (Sí o No):

a) ¿Emplea herramientas de análisis de sentimientos?

b) ¿Emplea análisis de sentimiento basado en imágenes?

c) ¿Emplea análisis de sentimiento en redes sociales?

d) ¿Hace referencia a sistemas de recomendación de turismo?

e) ¿Contiene tablas y gráficos de resultados?

Solo son aceptados aquellos artículos que superen el anterior test con 3 sí-es. Tras este proceso de cribado se llegó a la conclusión de que solo 6 de los 22 trabajos considerados anteriormente eran relevantes para el nuestro.

De los trabajos finalmente considerados, estos podrían ser divididos en tres categorías: por un lado tenemos los trabajos donde se busca recomendar contenido, posts o usuarios que seguir, por otro lado, tenemos trabajos donde se recomiendan lugares que visitar, y por último tenemos trabajos donde se busca analizar las opiniones sobre un destino turístico.

6.2. Recomendación de contenido

Dentro del primer grupo se encuentran trabajos como [23] donde se investiga el uso de análisis de sentimiento en la red social Facebook para mejorar los resultados de las recomendaciones de usuarios/amigos. Para llevar a cabo este objetivo, en este trabajo se analiza el texto de las publicaciones de los usuarios para extraer conceptos de las mismas, la subjetividad u objetividad de las frases, y los sentimientos encontrados en los documentos. A la hora de recomendar o no a un usuario, en este artículo se propone una

novedosa función de peso que tiene en cuenta el sentimiento de un usuario hacia un concepto así como el número de publicaciones de tal usuario acerca del mismo. La evaluación del método propuesto se hizo empleando un conjunto de datos de más de 6 millones de publicaciones extraídas de Facebook, dando un resultado superior a otros métodos ya establecidos.

Por otro lado, en el artículo [19] se propone un sistema de recomendación de *tweets* personalizado dentro del dominio de la salud. Para ello hace uso del análisis de sentimiento para la creación de un perfil de usuario basándose en el historial en la red social del usuario. El sistema propuesto consta de dos módulos, el primero se encarga de generar un perfil de usuario mediante la extracción información de su cuenta, sus intereses en temas de salud y los patrones de emoción a lo largo del tiempo, mientras que el segundo recoge datos públicos de Twitter, lleva a cabo un procesamiento de lenguaje natural y los clasifica para poder recomendarlos a los usuarios basándose en su correspondiente perfil. Este sistema se probó contra casi 1 millón de *tweets* de distintas categorías, alcanzando una precisión del 96 %.

6.3. Recomendación de lugares

Dentro del segundo grupo existen trabajos como [27] donde se presenta un sistema de recomendación de turismo personalizado teniendo en cuenta las preferencias del usuario. Su método denominado SMTM divide el problema en dos dominios, por un lado la extracción de los tópicos relacionados con una atracción turística, y por otro las preferencias del turista respecto a las atracciones a partir de textos e imágenes. Por último trata de proyectar los resultados del dominio del turista en el dominio de la atracción. Para probar el modelo generado con otros modelos se emplearon dos datasets, el primero generado a partir de de información del portal web TripAdvisor, y el segundo a partir de datos de la web Trip, en ambos casos los resultados obtenidos fueron mejores que el resto de métodos comparados.

Además, en el artículo [26] se propone un sistema de recomendación de turismo con dos objetivos, primero satisfacer las expectativas del turista, y segundo ayudar a las agencias de marketing a orientar sus promociones. Este sistema de recomendación se planteó con el objetivo de poder recomendar tanto lugares altamente valorados como aquellos que son pasados por alto por muchos turistas pero que merecen la pena, a los que llama lugares poco enfatizados, y cabe destacar que este sistema de recomendación es personalizado, teniendo en cuenta el historial del usuario. El sistema consta de varios módulos, el primero se usa para llevar a cabo una extracción de

tópicos de las reviews publicadas en Google y en TripAdvisor. El segundo lleva a cabo un análisis de sentimiento mediante máquinas de vectores soporte. Por último emplea una red neuronal artificial para recomendar los lugares a los usuarios, empleando una función de optimización para conseguir el objetivo de recomendar lugares poco enfatizados. Para probar el sistema se empleó un dataset con casi 150.000 publicaciones, llegando a obtener tener una precisión del 94 %.

6.4. Análisis sobre destinos turísticos

A diferencia de los trabajos anteriores donde el objetivo era el estudio o creación de sistemas de recomendación, existen trabajos como [22] donde se compara el rendimiento de múltiples métodos actuales de aprendizaje profundo con el fin de llevar a cabo un análisis de sentimiento de los tweets relacionados con Cilento, un popular destino turístico del sur de Italia. En este artículo se llegan a comparar 4 tipos de redes neuronales profundas empleando un dataset de casi 20000 tweets en inglés e italiano, cuyo sentimiento tuvo que ser clasificado manualmente de forma previa. Además de la comparación de estos métodos, en dicho artículo consiguen generar un mapa mediante la geolocalización de los tweets para poder así visualizar el sentimiento promedio en ciertas regiones del municipio.

También en [10] se lleva a cabo un análisis sobre el turismo en España y la percepción que tienen los turistas chinos empleando publicaciones en portales y redes sociales de China con el objetivo de medir la calidad de los distintos destinos turísticos. Para ello obtuvieron una dataset de casi 40 mil publicaciones, y llevaron a cabo una extracción de tópicos relacionados con las publicaciones así como análisis de sentimientos mediante procesamiento de lenguaje natural.

Conclusiones y Líneas de trabajo futuras

En este apartado se presentan las conclusiones finalmente obtenidas del desarrollo del proyecto, así como posibles mejoras o alternativas a desarrollar en el futuro.

7.1. Conclusiones

Del desarrollo del proyecto se han extraído múltiples conclusiones:

- Primero y lo más importante, se considera que se ha conseguido cumplir los objetivos generales inicialmente planteados. Se ha conseguido descargar, almacenar y llevar a cabo el análisis de emoción sobre las publicaciones de Instagram mediante el uso de plataformas en la nube. También se ha conseguido desarrollar un cuadro de mandos para visualizar la información almacenada en la base de datos. Todo ello sin incurrir en ningún coste puesto que se ha desarrollado todo el proyecto mediante software libre o durante el periodo de prueba sin mayores complicaciones.
- Dentro de los objetivos personales planteados al inicio del proyecto cabe destacar que finalmente si que se ha conseguido explorar múltiples herramientas, tanto de almacenamiento y procesamiento de datos como de visualización.
- Sobre el desarrollo de la base de datos final, ésta ha estado más limitada de lo esperado debido al máximo número de operaciones de inserción de elementos de Amazon S3 en la capa gratuita de AWS.

- Los *crawlers* son muy sensibles a las actualizaciones de las páginas web, y teniendo en cuenta la popularidad de Instagram y la frecuencia de sus actualizaciones, los fallos son comunes incluso en las herramientas más utilizadas para la descarga de publicaciones.

- A pesar de su gran facilidad de escalado, el uso de DynamoDB limita bastante las consultas que se pueden hacer incluso si lo comparamos con otras bases de datos NoSQL, faltando operaciones bastante básicas como la agregación. Además en la actualidad no existen muchas herramientas de visualización que permitan conectarse con esta base de datos, y menos aún que lo permitan gratuitamente.

7.2. Líneas de trabajo futuras

Sobre los posibles cambios y mejoras que se podrían llevar a cabo en un futuro sobre este proyecto cabría destacar los siguientes:

- Lo primero que se debería hacer es agrandar la base de datos, ya sea repitiendo los procedimientos que se han estado llevando a cabo para la adquisición de datos teniendo en cuenta el límite de operaciones de inserción en Amazon S3, o empezando a pagar por los servicios de Amazon para poder introducir más elementos mensualmente.

- Debido al anterior límite de Amazon S3, a pesar de que el script de descarga y procesamiento de publicaciones que se ha desarrollado está completamente automatizado, éste no se está ejecutando periódicamente de forma automática. Como se indicó en la sección 5.2, una posibilidad muy fácil de implementar sería simplemente añadirlo a un *cron* de Linux para que lo ejecute con la periodicidad deseada y que no requeriría de hacer ningún cambio en el script desarrollado. Otra opción posible sería portarlo a AWS Lambda, para poder así deshacernos de la instancia creada en EC2 para llevar a cabo este cometido, aunque esto posiblemente requeriría más trabajo e investigación sobre como generar eventos periódicamente que ejecuten el script.

- Con el fin de mejorar el análisis de emoción de las publicaciones, se podría explotar el texto de la descripción de las mismas, que actualmente está siendo descargado pero no se está empleando para nada.

- Se podría desarrollar una aplicación o una página web que fuese más *user friendly* en vez de usar un cuadro de mandos, para así

facilitar el acceso a las publicaciones. Esto además permitiría llevar a cabo consultas en DynamoDB optimizadas mediante el uso de índices secundarios, ya que actualmente con el dashboard de Grafana, como se permite filtrar por cualquier criterio, esto no tiene sentido. Por otro lado, en esta posible aplicación o web, se podría automatizar el proceso de extraer meta-datos del usuario en base a su perfil de Instagram en vez de tener que introducirlos manualmente. También se podría valorar emplear sus publicaciones para mejorar la precisión de las publicaciones recomendadas como se ha propuesto en varios de los trabajos relacionados.

Apéndices

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En este apartado se presenta la planificación temporal del proyecto empleando la metodología SCRUM, así como la viabilidad del proyecto desde el punto de vista legal y económico.

A.2. Planificación temporal

Como se ha indicado en el apartado anterior, para el desarrollo del proyecto se siguió la metodología *SCRUM*, adaptándola a las necesidades y al tiempo disponible. Las características a destacar de esta metodología son las siguientes:

- Desarrollo iterativo e incremental que permite adaptarse a cambios inesperados en los requerimientos o herramientas empleadas.
- El proyecto se subdivide en múltiples tareas, que se abordan a lo largo de varias iteraciones denominadas *sprints*.
- Todos los *sprints* tienen una duración prefijada, en este caso de 2 semanas.
- Todo *sprint* se inicia o finaliza con una reunión con el tutor del proyecto.

Sprint 0

Durante el *sprint 0* se abordaron las tareas requeridas para la inicialización del proyecto, así como la evaluación y la toma de decisión de las distintas herramientas que se pretendían utilizar durante la posterior implementación.

Tareas
Creación del repositorio del proyecto
Creación del entorno de trabajo
Evaluación de la forma de extraer hashtags de Instagram
Evaluación de los distintos *crawlers* y herramientas a utilizar
Registro y experimentación con las herramientas de Google Cloud

Tabla A.1: Tareas del *sprint 0*

Sprint 1

Durante el *sprint 1* se abordaron las tareas relacionadas con la automatización de la descarga de publicaciones de Instagram y se inició el desarrollo de la memoria.

Tareas
Creación de la máquina virtual
Implementación de la descarga de publicaciones mediante Instalooter
Subida de publicaciones a un *blob* de Google Cloud
Pruebas de reconocimiento facial con Cloud Vision
Documentación sobre la metodología PRISMA para map-review
Inicio de la memoria en Overleaf

Tabla A.2: Tareas del *sprint 1*

Sprint 2

Durante el *sprint 2* se evaluaron las alternativas a Cloud Vision tras comprobar que no se podía extraer la edad ni el género. También se redactó la parte de trabajos relacionados o el estado del arte de la memoria.

Tareas
Evaluación de las alternativas a Cloud Vision y Google Cloud
Pruebas con Microsoft Azure
Pruebas con Amazon Web Services
Elección de los artículos relacionados
Redacción del apartado de Trabajo relacionados

Tabla A.3: Tareas del *sprint 2*

Sprint 3

Durante el *sprint 3* se decidió emplear Amazon Web Services tras comprobar que era la mejor alternativa. Se llevaron a cabo los cambios en los scripts y se implementó la descarga de datos de reconocimiento facial.

Tareas
Adaptación de los scripts para usar los servicios de Amazon
Implementación de la descarga de datos de Rekognition
Evaluación de las posibles bases de datos a utilizar
Evaluación de las posibles visualizaciones a implementar

Tabla A.4: Tareas del *sprint 3*

Sprint 4

Durante el *sprint 4* se diseñó e implementó la base de datos en DynamoDB. También se tuvo que llevar a cabo una corrección en Instalooter después de que dejara de funcionar.

Tareas
Diseño e implementación de la base de datos
Implementación de los scripts de subida de datos a DynamoDB
Hotfix de Instalooter
Evaluación de los posibles dashboards a utilizar

Tabla A.5: Tareas del *sprint 4*

Sprint 5

Durante el *sprint 5* se tomó la decisión de implementar un cuadro de mandos con Grafana, para ello se iniciaron las pruebas para crear un conector con DynamoDB, también se finalizó el script para poblar la base de datos.

Tareas
Creación del script final de descarga y subida de datos a DynamoDB
Llenado de la base datos
Pruebas con Grafana
Desarrollo de un servidor local para conectar Grafana con DynamoDB

Tabla A.6: Tareas del *sprint 5*

Sprint 6

Durante el *sprint 6* se finalizó el conector de Grafana con DynamoDB, se implementó el cuadro de mandos final y se continuó con la redacción de la memoria.

Tareas
Portado del conector de Grafana a AWS Lambda y Amazon API Gateway
Diseño e implementación final del Dashboard
Redacción del apartado de Objetivos del proyecto
Redacción del apartado de Conceptos teóricos
Redacción del apartado de Técnicas y herramientas

Tabla A.7: Tareas del *sprint 6*

Sprint 7

Durante el *sprint 7* se finalizó la memoria y se reorganizó el repositorio de cara a la entrega final.

Tareas
Redacción del apartado de Introducción
Redacción del apartado de Aspectos relevantes del desarrollo del proyecto
Redacción del apartado de Conclusiones y líneas de trabajo futuras
Redacción de los apéndices

Tabla A.8: Tareas del *sprint 7*

A.3. Estudio de viabilidad

Viabilidad económica

En este apartado se describe el coste estimado del proyecto.

Costes de personal

Mediante el "XVII Convenio colectivo estatal de empresas de consultoría y estudios" publicado en el BOE Núm. 57 el 6 de marzo de 2018, podemos obtener los costes del personal. En este proyecto se ha empleado a un Analista Programador durante 4 meses. De acuerdo al anterior documento se puede concluir que el salario mensual a tiempo completo es de 1.916,15€. Como la dedicación ha sido parcial, el sueldo es de 958,07€.

Tarea	Perfil	Salario mensual	Meses	Total
Desarrollo del proyecto	Analista Programador	958,07€	4	3.832,29€

Tabla A.9: Costes de personal

Costes de Hardware

Para el desarrollo del proyecto el único hardware necesario ha sido un **ordenador personal**, cuyo coste es de 1.139,86€.

Costes de uso de Amazon Web Services

Como se ha comentado con anterioridad, en el desarrollo del proyecto no se ha incurrido en ningún coste respecto al uso de Amazon Web Services puesto que se ha empleado su capa gratuita. Pero algunos de los servicios de esta capa gratuita están limitados a tan solo los 12 primeros meses. Es por ello que se ha decidido calcular cual sería el coste mensual del uso de esta plataforma bajo las mismas condiciones que las que se han empleado a lo largo del proyecto. Para ello se ha empleado la calculadora de coste de AWS, empleando como región `eu-west-1` (Irlanda):

- **Amazon EC2**: Empleando una máquina virtual `t2.micro` con Linux con 30GB de almacenamiento SSD, el coste por hora está a 0,0126 USD. En la capa gratuita, se permite usar hasta 750 horas al mes que son más o menos las que se llegaron a utilizar. Por ello en total el coste calculado al mes es de 9,80 USD.

- **Amazon S3**: Los primeros 50 TB de almacenamiento al mes tienen un coste de 0,023 USD por GB, y en la capa gratuita se permite usar hasta 5GB. Por otro lado, 1000 solicitudes de operaciones PUT cuestan 0,005 USD, y como se ha comentado anteriormente, se usaron las 2000 operaciones de este tipo que se permiten en la capa gratuita. Además, 1000 solicitudes GET son 0,0004 USD, y en la capa gratuita se permite hasta 20000. Por lo tanto, el coste final es de 0,13 USD al mes.

- **Amazon Rekognition**: El coste del primer millón de imágenes está actualmente a 0,001 USD por imagen. Como se han analizado 2000 imágenes al mes el coste estimado es de 2.00 USD.

- **Amazon DynamoDB**: Se tiene 25 GBs gratuitos para siempre, y esto por ahora ha sido más que suficiente, por lo que no habría costes por el almacenamiento y las consultas en la base de datos.

- **AWS Lambda**: Se tiene un millón de operaciones al mes (3,2 millones de segundos) gratuitas para siempre. Esto ha sido más que suficiente durante las pruebas llevadas cabo, con lo que no habría costes por las consultas llevadas a cabo por el cuadro de mandos.

- **Amazon API Gateway**: Las primeras 333 millones de peticiones API de REST tienen un coste 3,50 USD por millón. Como en la capa gratuita se permite hasta un millón, el coste sería de 3,50 USD al mes.

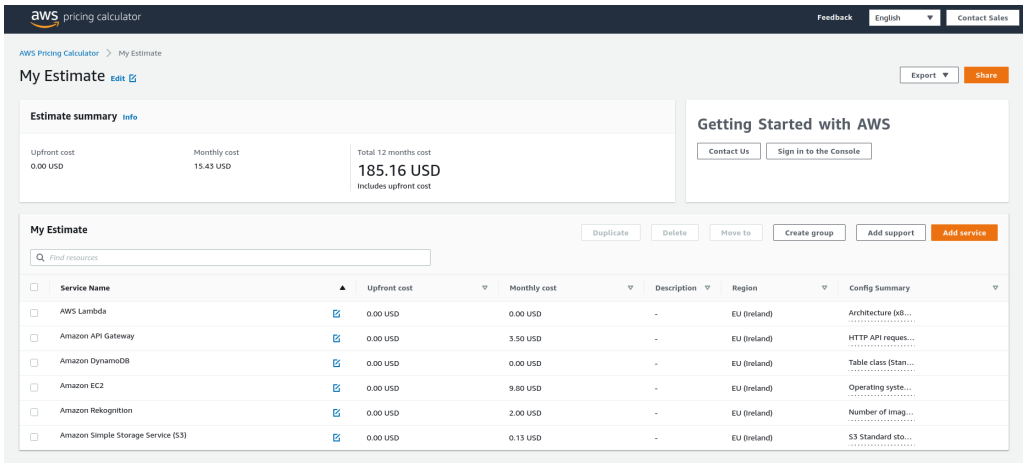

Figura A.1: Cálculo de coste con la calculadora de AWS

Por lo tanto, mensualmente el coste estimado sería de 15,43 USD por usar los servicios de AWS. Como en la actualidad el tipo de cambio es de 1,00 USD = 1,00 EUR, el coste final sería el siguiente:

Coste mensual	**Meses**	**Total**
15,43€	4	61,72€

Tabla A.10: Costes de uso de AWS

Beneficio industrial

Según el artículo 101.2 de la Ley 9/2017 de Contratos del Sector Público (LCSP) el beneficio industrial se calcula como el 6 % del coste total del proyecto sin aplicar el IVA. Por lo tanto el beneficio industrial generado es el siguiente:

Concepto	**Valor (€)**
Coste de personal	3.832,29€
Costes de hardware	1.139,86€
Costes de AWS	61,72€
Subtotal	5.033,87€
Beneficio industrial (6 %)	302,03€

Tabla A.11: Beneficio industrial

Coste total

El coste total tras contar todos los subapartados anteriores se puede ver en la siguiente tabla:

Concepto	**Valor (€)**
Coste de personal	3.832,29€
Costes de hardware	1.139,86€
Costes de AWS	61,72€
Beneficio industrial (6 %)	302,03€
Subtotal	5.335,90€
IVA (21 %)	1.120,54€
Total	6.456,44€

Tabla A.12: Coste total

Viabilidad legal

En esta sección se muestran las licencias empleadas por las distintas librerías y herramientas utilizadas a lo largo del proyecto, y se describe la licencia finalmente empleada para la publicación del mismo.

Los herramientas empleadas durante el desarrollo del proyecto emplean las siguientes licencias:

Herramienta	**Licencia**
boto3	Apache License 2.0
fs-s3fs	MIT
Instalooter	GPL v3.0
Grafana	AGPL v3.0
JSON API Grafana Datasource	MIT

Tabla A.13: Licencias de las herramientas utilizadas

Por lo tanto, se ha decidido que el proyecto emplee la licencia GPL v3.0, ya que es compatible con todas las anteriores licencias. Entre otras características, esta licencia permite el uso comercial y privado, la modificación y redistribución del código, y a la vez protege ante problemas de garantía, pero al mismo tiempo obliga a que cualquier herramienta que haga uso del proyecto tenga que ser liberada con la misma licencia.

Apéndice B

Documentación técnica de programación

B.1. Introducción

En este apartado se presenta la forma en que se estructura el proyecto final, el contenidos de sus directorios y ficheros, y como poder llevar a cabo la instalación y ejecución de los distintos módulos.

B.2. Estructura de directorios

En este apartado se presenta la estructura de directorios y el contenido final que tiene el repositorio generado durante el desarrollo del proyecto. Este repositorio se encuentra alojado en GitHub y es público, se puede acceder desde el enlace incluido en el apartado 5.5.

La estructura base del repositorio se puede ver en el siguiente árbol:

```
/
├── memoria/
├── presentacion/
├── scripts/
├── .gitignore
├── LICENSE
├── Notas.md
└── README.md
```

En resumidas cuentas, estos ficheros y directorios cumplen la siguiente función:

- `/memoria/`: directorio donde se ubican todos los ficheros necesarios para compilar la memoria del proyecto, es decir, el PDF actual. La memoria, como se comentó en la sección 4.4, ha sido generada mediante LaTeX.
- `/presentacion/`: directorio donde se ubican todos los ficheros necesarios para compilar la presentación del proyecto. Para ello también ha sido empleado LaTeX.
- `/scripts/`: directorio donde se ubican todos los scripts creados durante el desarrollo del proyecto. También se incluye algún otro fichero necesario para el funcionamiento de los mismos. El contenido de este directorio es el siguiente:

```
/scripts/
├── lambda/
├── tests/
├── grafana_json_model.json
├── hashtags.txt
└── mainScriptAWS.py
```

  - `lambda`: scripts Python empleados en AWS Lambda para generar las APIs para acceder a la base de datos de DynamoDB desde Grafana.
  - `tests`: scripts de test generados durante el desarrollo del proyecto: los tests iniciales de los scrapers de Instagram, las pruebas con Google Cloud, las posteriores pruebas con AWS, la página web estática para probar la visualización de datos con JavaScript y el servidor REST de prueba para Grafana.
  - `grafana_json_model.json`: JSON que contiene el modelo del cuadro de mandos generado con Grafana.
  - `hashtags.txt`: fichero de hashtags que hay que pasar al script de descarga de publicaciones.
  - `mainScriptAWS.py`: fichero de descarga, procesado y almacenamiento de publicaciones en la base de datos de DynamoDB.
- `/.gitignore`: fichero empleado para evitar subir ficheros no deseados al repositorio.

- `/LICENSE`: licencia del proyecto, como se comentó anteriormente, la licencia final es GPL v3.0.
- `/Notas.md`: fichero de notas tomadas durante el desarrollo del proyecto, podría decirse que es el *cuaderno de bitácora* del proyecto.
- `/README.md`: fichero de presentación del repositorio que se muestra al entrar en GitHub.

B.3. Compilación, instalación y ejecución del proyecto

Script de descarga

Para poder ejecutar el script `mainScriptAWS.py` primero es necesario instalar las dependencias necesarias y generar el entorno de trabajo. A continuación se describen los pasos que hay que llevar a cabo para ello, mostrando los comandos que habría que usar en una máquina Debian puesto que, como ya se ha comentado, es el sistema operativo que se ha utilizado para el desarrollo del proyecto. Aún así, se debería de poder conseguir el mismo resultado en otros sistemas operativos empleando su correspondiente gestor de paquetes.

1. Primero hay que instalar Python 3, `git` y `pip3` si no viene con la distribución.

```
sudo apt-get update
sudo apt-get install python3 python3-dev python3-venv
sudo apt-get install git
sudo apt-get install wget
wget https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
```

2. El siguiente paso es crear el entorno de ejecución mediante `virtualenv`, en este caso se ha llamado *testvisualenv*.

```
sudo apt-get install virtualenv
virtualenv --python=python3 testvisualenv
source testvisualenv/bin/activate
```

3. Después hay que instalar en nuestro entorno de trabajo las librerías y el SDK de AWS mediante `pip3`.

   ```
   pip3 install boto3
   pip3 install fs-s3fs
   pip3 install git+https://github.com/Zalez95/InstaLooter.git
   ```

   Como se puede ver, se instala el fork creado en la sección 5.3 puesto que la versión oficial de Instalooter todavía no incluye las correcciones.

4. Para que Instalooter funcione hay que iniciar sesión mediante un navegador por primera vez para que se generen las cookies. Si en nuestro caso estamos empleando un sistema operativo sin navegador, entonces hay que ejecutar los siguientes pasos:

   ```
   sudo apt-get install firefox-esr
   ```

   Tras instalar el navegador, hay que entrar en la web de Instagram puesto que por primera vez siempre sale una ventana con un mensaje para aceptar las cookies, y tras iniciar sesión puede que salgan más ventanas para habilitar notificaciones que no son soportadas por Instalooter. Si nuestro sistema operativo no tiene interfaz gráfica como sucedió con las instancias de AWS, este proceso se tiene que hacer a través de SSH habilitando la opción de *X11 Forwarding*. Para ello hay que instalar:

   ```
   sudo apt-get install xauth x11-apps
   ```

   Después hay que modificar el fichero `/etc/ssh/sshd_config` y habilitar la opción `X11Forwarding yes`. Tras reiniciar, se puede conectar con la máquina virtual a través de ssh con la opción -Y para poder ejecutar aplicaciones con interfaz gráfica, en este caso con el comando `firefox` podemos lanzar el navegador y que se muestre en nuestro equipo aunque se esté ejecutando en la máquina virtual.

Tras instalar las dependencias, el siguiente paso para poder ejecutar el script anterior es definir las siguientes variables de entorno:

- `INSTA_USER`: usuario de Instagram con el que iniciar sesión.
- `INSTA_PASS`: contraseña de Instagram con el que iniciar sesión.

- `AWS_AKEY_ID`: identificador de AWS para poder usar los servicios. Este valor junto con `AWS_AKEY_SECRET` se puede obtener desde la web de AWS en el menú de la cuenta en *Credenciales de Seguridad* > *Claves de Acceso*.

- `AWS_AKEY_SECRET`: contraseña de AWS para poder usar los servicios.

Una opción para no tener que introducirlo en cada sesión es añadir las opciones anteriores al fichero `.bashrc` de la siguiente forma:

```
# ENV VARS
export INSTA_USER="..."
export INSTA_PASS="..."
export AWS_AKEY_ID="..."
export AWS_AKEY_SECRET="..."
```

Por último hay que destacar que este script espera que los servicios de Amazon Web Services se estén ejecutando en `eu-west-1`, contra un bucket de Amazon S3 llamado `valltourisminstabucket` y una tabla de DynamoDB llamada `valltourisminsta`, en caso de querer emplear otras regiones, buckets o tablas habrá que modificar el script para poner los nombres correspondientes.

Grafana y conector con DynamoDB

Para poder crear el conector de Grafana con DynamoDB a través de las APIs REST, primero hay que generar las funciones de AWS Lambda. Para ello desde el menú principal de AWS Lambda hay que ir a *Funciones* > *Crear función* y crear las funciones indicadas en la sección 5.4. El código que las implementa se encuentra en la carpeta `scripts/lambda`, tan solo hay que pegarlo en la pestaña de *Código*. Todas las funciones han de tener configurado las variables de entorno `AWS_AKEY_ID` y `AWS_AKEY_SECRET` para poder acceder a DynamoDB, para ello hay que añadirlas en la pestaña *Configuración* > *Variables de entorno*. Una vez configuradas solo hay que darle al botón de *Deploy*:

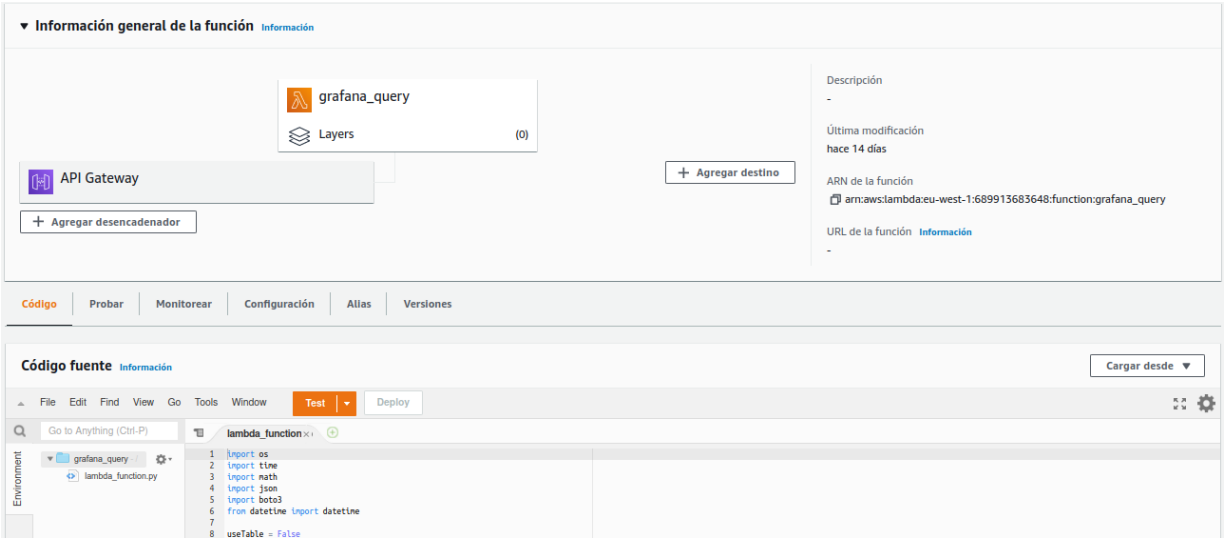

Figura B.1: Configuración de una función de AWS Lambda

Después hay que generar las APIs REST en AWS API Gateway, para ello desde su menú en AWS hay que crear una API, en este caso llamada *valltourisminsta*. Una vez creada hay que generar los recursos indicados en la sección 5.4 apuntando a las correspondientes funciones de AWS Lambda. Una vez configuradas, la API se despliega desde el menú de *Acciones* > *Implementar API*, en una etapa que en este caso se llamó "demo".

Por otro lado, una vez que se tiene el conector hay que instalar Grafana. Para ello, en un equipo Debian hay que ejecutar los siguientes comandos:

```
sudo apt-get install -y adduser libfontconfig1
wget https://dl.grafana.com/enterprise/release/grafana-
    enterprise_9.0.0_amd64.deb
sudo dpkg -i grafana-enterprise_9.0.0_amd64.deb
sudo service grafana-server start
```

Tras iniciar el servicio de Grafana, éste se encontrará ejecutando en el puerto 3000, y para poder acceder solo hay que entrar mediante un navegador apuntando al equipo y puerto correspondiente. En el caso de este proyecto, como la máquina virtual que lo ejecutaba era una instancia de EC2, hubo que añadir una regla para permitir las conexiones a este puerto desde el menú de *EC2* > *Grupos de Seguridad*, de tipo TCP, origen `0.0.0.0/0` y puerto 3000. Tras iniciar sesión hay que instalar el plugin JSON para poder conectar a las APIs REST.

Para crear el cuadro de mandos con Grafana primero hay que crear un *data source* mediante el plugin JSON que apunte a la URL de nuestro servidor en AWS API Gateway. Después desde el menu de Dashboards se crear uno nuevo. En el apartado de configuración del dashboard, existe una opción llamada llamada *JSON Model*, donde si se pega el contenido del fichero `grafana_json_model.json` se podría obtener un cuadro de mandos idéntico al creado para este proyecto.

Bibliografía

[1] Ajgaonkar and Seier. The value of computer vision: More than meets the eye. *Insight*, 3 2021.

[2] Hyeon-woo An and Jinah Kim. Design of recommendation system for tourist spot using sentiment analysis based on cnn-lstm. *Journal of Ambient Intelligence and Humanized Computing*, 13, 03 2022.

[3] AWS. Amazon rekognition image. https://aws.amazon.com/es/rekognition/image-features/, 07 2022. Accedido: 2022-07-02.

[4] AWS. Características de amazon api gateway. https://aws.amazon.com/es/api-gateway/features/, 06 2022. Accedido: 2022-07-02.

[5] AWS. Características de amazon s3. https://aws.amazon.com/es/s3/features/, 06 2022. Accedido: 2022-07-02.

[6] AWS. Características de aws lambda. https://aws.amazon.com/es/lambda/features/, 06 2022. Accedido: 2022-07-02.

[7] AWS. Tipos de instancias de amazon ec2. https://aws.amazon.com/es/ec2/instance-types/, 06 2022. Accedido: 2022-07-02.

[8] Banafa. ¿qué es la computación afectiva? https://www.bbvaopenmind.com/tecnologia/mundo-digital/que-es-la-computacion-afectiva/, 6 2016. Accedido: 2022-07-01.

[9] Jonas Birmé. Using aws lambda and api gateway for server-less grafana adapters. https://www.linkedin.com/pulse/using-aws-lambda-api-gateway-server-less-grafana-adapters-jonas-birmé, 07 2017. Accedido: 2022-07-06.

[10] Fernando Borrajo-Millán, María-del-Mar Alonso-Almeida, María Escat-Cortes, and Liu Yi. Sentiment analysis to measure quality and build sustainability in tourism destinations. *Sustainability*, 13(11), 2021.

[11] Datosmacro. España - turismo internacional. https://datosmacro.expansion.com/comercio/turismo-internacional/espana. Accedido: 2022-07-01.

[12] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: Amazon's highly available key-value store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, 10 2007.

[13] Richard Freeman. Serverless dynamic real-time dashboard with aws dynamodb, s3 and cognito. https://medium.com/@rfreeman/serverless-dynamic-real-time-dashboard-with-aws-dynamodb-a1a7f8d3bc01, 02 2016. Accedido: 2022-07-06.

[14] Tracy Frey. Ethics in action: removing gender labels from cloud's vision api. *Diversity Google*, 9 2020.

[15] Glover. Diferencia entre emoción y sentimiento en psicología. https://www.psicologia-online.com/diferencia-entre-emocion-y-sentimiento-en-psicologia-3942.html, 6 2021. Accedido: 2022-07-01.

[16] googleapis. Ethics in action: removing gender labels from cloud's vision api. https://googleapis.dev/python/vision/latest/index.html, 6 2022. Accedido: 2022-07-02.

[17] googlecloud. Prueba gratuita y nivel gratuito. https://cloud.google.com/free?hl=es, 6 2022. Accedido: 2022-07-02.

[18] Grafana. Grafana. https://github.com/grafana/grafana, 06 2022. Accedido: 2022-07-06.

[19] Khan, Khattak, Batool, Satti, Hussain, Khan, Khan, and Hayat. Tweets classification and sentiment analysis for personalized tweets recommendation. *Complexity*, 2020, 12 2020.

[20] Martin Larralde. Instalooter. https://instalooter.readthedocs.io/en/latest/. Accedido: 2022-07-02.

[21] Selvi Munuswamy, Saranya M S, Satish Ganapathy, Muthurajkumar Sannasy, and Kannan Arputharaj. Sentiment analysis techniques for social media-based recommendation systems. *National Academy Science Letters*, 44, 07 2020.

[22] Paolanti, Mancini, Frontoni, Felicetti, Marinelli, Marcheggiani, and Pierdicca. Tourism destination management using sentiment analysis and geo-location information: a deep learning approach. *Information Technology and Tourism*, 06 2021.

[23] S. Rastogi, Rohit Singhal, and Rajeev Kumar. A sentiment analysis based approach to facebook user recommendation. *International Journal of Computer Applications*, 90, 02 2014.

[24] Reyes. 6 diferencias entre emociones y sentimientos. https://www.psicoemocionat.com/6-diferencias-entre-emociones-y-sentimientos/, 3 2017. Accedido: 2022-07-01.

[25] Melati Rosanensi, Miftahul Madani, Rizki Tri Puji Wanggono, Arief Setyanto, Andhika Agus Selameto, and Sri Ngudi Wahyuni. Analysis sentiment and tourist response to rinjani mountain tour based on comments from photo upload in instagram. In *2018 3rd International Conference on Information Technology, Information System and Electrical Engineering (ICITISEE)*, pages 184–188, 2018.

[26] Shafqat and Byun. A recommendation mechanism for under-emphasized tourist spots using topic modeling and sentiment analysis. *Sustainability*, 12:320, 12 2019.

[27] Xi Shao, Guijin Tang, and Bing-Kun Bao. Personalized travel recommendation based on sentiment-aware multimodal topic model. *IEEE Access*, 7:113043–113052, 2019.

[28] Sobrín, Campazas, Guerrero, Rodríguez, and Fernández. Systematic mapping of detection techniques for advanced persistent threats. In *13th International Conference on Computational Intelligence in Security for Information Systems*, volume 1267, pages 426–435, 2020.

[29] Urueña, Ferrari, Blanco, and Valdecasa. Las redes sociales en internet. *ONTSI*, pages 12–13, 12 2011.

[30] Wikipedia. Grafana. https://en.wikipedia.org/wiki/Grafana, 06 2022. Accedido: 2022-07-06.

[31] Yamini. Emotion and sentiment analysis: What are the differences? https://www.analyticssteps.com/blogs/emotion-and-sentiment-analysis-what-are-differences, 1 2022. Accedido: 2022-07-01.