



# MEMORIA DE FINALIZACIÓN DE PRÁCTICAS

UNIVERSIDAD NEBRIJA GRADO EN  
INGENIERÍA INFORMÁTICA MEMORIA  
PRÁCTICAS EN EMPRESA

Guillermo Martínez de Hurtado Aricha

Junio de 2025



# MEMORIA DE FINALIZACIÓN DE PRÁCTICAS

UNIVERSIDAD NEBRIJA GRADO EN  
INGENIERÍA INFORMÁTICA MEMORIA  
PRÁCTICAS EN EMPRESA

Guillermo Martínez de Hurtado Aricha

Tutor: Adrián Pradilla Portoles

Junio de 2025

## Derechos

D. Guillermo Martínez de Hurtado Aricha autoriza a que el presente trabajo se guarde y custodie en los repositorios de la Universidad Nebrija y además NO autoriza a su disposición en abierto.

## Contenido

Derechos	3
Índice de ilustraciones	5
Glosario de términos	6
1. RESUMEN	9
2. PRÁCTICAS	9
2.1. Antecedentes	9
2.2. Objetivos de las prácticas	15
2.3. Onboarding	16
2.4. Funciones y tareas en las prácticas. (evidencias)	18
2.5. Relaciones de problemas planteados y procedimientos para su resolución	22
2.6. Aprendizajes y desarrollo profesional (habilidades adquiridas)	25
2.7. Metodologías utilizadas	25
2.8. Herramientas utilizadas en las prácticas (evidencias)	26
2.9. Logros, resultados y discusión.	28
3. DESARROLLO	29
3.1. Introducción	29
3.2. Marco técnico	30
3.3. Equipo de trabajo y metodología	31
3.4. Proyecto	31
3.5. Resultados y discusión	34
3.6. Conclusiones	36
3.7. Líneas futuras	36
4. BIBLIOGRAFÍA.	36
5. ANEXOS	37

# Índice de ilustraciones

Ilustración 1: Logotipo de Serveo. (Serveo, n.d.)	10
Ilustración 2: Horario de onboarding en Serveo. (Elaboración propia)	17
Ilustración 3: Imagen del google summit 2025 (IA Google)	18
Ilustración 4: Ejemplo expresión REGEX.(Elaboración propia)	19
Ilustración 5: Ejemplo de librerías Python.(Elaboración propia)	20
Ilustración 6: Interfaz aplicación corporativa.(Elaboración propia)	21
Ilustración 7: Organización SharePoint jerarquía .(Elaboración propia)	22
Ilustración 8: Organización del planner del equipo.(Elaboración propia)	25
Ilustración 9: Ejemplo de uso PBI.(Elaboración propia)	26
Ilustración 10: Ejemplo de uso GIT.(Elaboración propia)	27
Ilustración 11: Interfaz de la aplicación análisis de sentimientos.(Elaboración propia)	32
Ilustración 12: Librerías incorporadas en Python .(Elaboración propia)	33
Ilustración 13: Procesamiento de textos de Youtube.(Elaboración propia)	35
Ilustración 14: Gráficas comparativas de resultados.(Elaboración propia)	36

# Glosario de términos

## **API (Interfaz de Programación de Aplicaciones):**

Conjunto de protocolos y herramientas que permiten la comunicación entre distintos sistemas. En este trabajo se utiliza para acceder a datos públicos, como los precios de carburantes ofrecidos por el gobierno.

## **CSV (Comma-Separated Values):**

Formato de archivo que almacena datos en forma de texto plano, donde cada línea representa un registro y los campos están separados por comas (u otros delimitadores). Se emplea para almacenar y manipular datos extraídos de archivos XML.

## **Data Lake (Lago de Datos):**

Repositorio centralizado que permite almacenar datos en su formato original, sean estructurados o no estructurados. En el proyecto con Microsoft Fabric se crean lagos de datos para integrar y transformar la información proveniente de diversas fuentes.

## **Delta (Formato Delta):**

Formato de almacenamiento de datos optimizado para trabajar con lagos de datos, permitiendo actualizaciones y transacciones de manera eficiente.

## **Eficiencia Operativa:**

Conjunto de prácticas y procesos enfocados en optimizar el rendimiento y la productividad dentro de una organización, a través de la automatización, mejora de procesos y uso de tecnología.

## **Excel:**

Herramienta de Microsoft Office ampliamente utilizada para el manejo, análisis y visualización de datos en formato de hojas de cálculo.

## **IA (Inteligencia Artificial):**

Campo de estudio de la informática que busca desarrollar sistemas capaces de aprender, razonar y tomar decisiones de manera autónoma. En el contexto del proyecto se emplea para entrenar modelos que permitan extraer información de documentos repetitivos.

## **Log (Registro de Eventos):**

Archivo que almacena información y eventos ocurridos durante la ejecución de un programa. Se utiliza para el seguimiento y depuración de procesos en los scripts desarrollados.

**OCR (Reconocimiento Óptico de Caracteres):** Tecnología que permite convertir diferentes tipos de documentos, como archivos PDF o imágenes, en datos editables y procesables. Se ha utilizado para extraer IBANs de documentos PDF.

**Planner:**

Herramienta de Microsoft 365 que facilita la gestión de tareas y proyectos, permitiendo asignar actividades, establecer fechas límite y coordinar el trabajo en equipo.

**Power Automate:**

Plataforma de Microsoft que permite la automatización de flujos de trabajo sin necesidad de escribir código (no-code). Se utiliza para tareas repetitivas, como la automatización del procesamiento de correos electrónicos.

**Power BI:**

Herramienta de visualización y análisis de datos de Microsoft que permite transformar datos en informes y dashboards interactivos, facilitando la interpretación de la información.

**Power Query:**

Componente de Microsoft que facilita la transformación, limpieza y preparación de datos, utilizando tanto una interfaz visual como el lenguaje de programación M para personalizaciones avanzadas.

**PowerApps:**

Plataforma de Microsoft que permite crear aplicaciones empresariales de forma rápida, con muy poco o sin necesidad de código. Se ha empleado para desarrollar aplicaciones internas, como el registro de EPIs.

**Pipeline (Flujo de Datos):**

Secuencia de procesos automatizados que permiten la ingesta, transformación y almacenamiento de datos. En Microsoft Fabric se utilizan pipelines para gestionar la actualización de datos.

**Python:**

Lenguaje de programación de alto nivel, muy utilizado en análisis y automatización de datos por su versatilidad y amplia disponibilidad de librerías.

**PySpark:**

API de Python para trabajar con Apache Spark, utilizada para el procesamiento y análisis de grandes volúmenes de datos de forma distribuida.

**SQL (Structured Query Language):**

Lenguaje de programación especializado en la gestión y manipulación de bases de datos relacionales. Se utiliza para realizar consultas y transformaciones en las bases de datos del proyecto.

**SharePoint:**

Plataforma de Microsoft para la gestión y almacenamiento de documentos y datos, que permite la colaboración y el acceso compartido a la información dentro de la empresa.

**XML (Extensible Markup Language):**

Lenguaje de marcado utilizado para almacenar y transportar datos de forma estructurada. En el proyecto se extraen datos en formato XML para luego transformarlos a CSV.

**FTE (Full time-equivalent):**

Tiempo de trabajo equivalente a una jornada laboral (8 horas).



# 1. RESUMEN

Este trabajo documenta y analiza el recorrido y las experiencias acumuladas durante mis prácticas en el Departamento de Eficiencia Operativa y Análisis de Datos. En él se recoge el conjunto de proyectos que he llevado a cabo hasta la fecha, abarcando tanto la parte de eficiencia operativa como la de análisis de datos. Durante este periodo, he tenido la oportunidad de trabajar en la automatización de tareas, la transformación de datos y el desarrollo de aplicaciones que optimizan procesos internos. Entre los proyectos más destacados se encuentran

La segunda parte de este trabajo se dedicará en profundidad a explicar el código desarrollado para un aplicación con interfaz de usuario capaz de detectar los sentimientos a través de transcripciones de videos de plataformas como Youtube o Twitch.

## 2. PRÁCTICAS

### 2.1. Antecedentes

#### 2.1.1. ¿Qué entrevistas he hecho?

Para la asignatura de Evaluación del desarrollo de capacidades en la empresa II he continuado desarrollando mis prácticas como becario en Serveo, misma empresa que para la asignatura anterior.

A la hora de firmar las prácticas para el primer cuatrimestre, me ofrecieron hacer también tanto las horas del segundo cuatrimestre como 300 horas más extracurriculares que acabarán en junio.

#### 2.1.2. Datos de la empresa



*Ilustración 1: Logotipo de Serveo. (Serveo, n.d.)*

## Nombre y ubicación

Continuo mis prácticas en Serveo, compañía cuyas oficinas centrales están en Las Tablas, Madrid, aunque al tratarse de una compañía que se dedica principalmente al Facility Management industrial es común visitar algún contrato y trabajar desde las oficinas del contrato al que se va a dar apoyo.

## Fechas de realización

Para la segunda asignatura de evaluación de capacidades en la empresa mi periodo de realización de prácticas es desde enero hasta abril, y como he mencionado antes haré horas extracurriculares hasta junio.

## Descripción de la actividad de la empresa

Serveo es una empresa del sector de servicios especializada en el mantenimiento y la gestión integral de infraestructuras, facility management y servicios relacionados. Su propuesta de valor se basa en ser un **integrador transversal de servicios de facility management**, lo que le permite centralizar la interlocución con el cliente y optimizar recursos para impulsar la eficiencia operativa y los objetivos de sostenibilidad (Net Zero). A través de un equipo multidisciplinar de más de 47 000 profesionales, Serveo cubre desde el mantenimiento técnico de instalaciones (Hard Services) hasta servicios de limpieza, jardinería, seguridad, teleasistencia y atención a personas vulnerables (Soft Services), además de servicios auxiliares como gestión de residuos y prevención de riesgos laborales.

La compañía opera en cinco grandes sectores:

- **Industria:** Incluye mantenimiento eléctrico, mecánico, climatización, automatización de líneas de producción y otros servicios técnicos especializados.
- **Salud:** Comprende servicios en hospitales, centros de salud y residencias, tales como mantenimiento de equipamiento médico, limpieza especializada, servicios sociales y atención a pacientes.
- **Corporate:** Engloba a empresas y sedes corporativas donde ofrece servicios integrales que incluyen mantenimiento de edificios, gestión de espacios, limpieza de oficinas, jardinería y conserjería.
- **Government (Administraciones Públicas):** Presta servicios a entidades gubernamentales en edificios administrativos, organismos públicos y centros educativos, con énfasis en la eficiencia energética y el cumplimiento normativo.
- **Infraestructuras:** Atiende aeropuertos, carreteras, estaciones y otras infraestructuras críticas, ofreciendo servicios a bordo (en aeropuertos), mantenimiento de carreteras

(Skyway) y conservación de infraestructuras viarias.

Serveo se autodefine como **impulsor del cambio y del progreso sostenible**, comprometiéndose con la innovación, el desarrollo de las personas y la mejora continua mediante procesos basados en indicadores de gestión y tecnología avanzada para garantizar la fiabilidad de sus servicios.

### **Descripción de la estructura organizativa y puesto ocupado en ella**

Internamente, Serveo organiza su modelo de negocio en cinco áreas principales para facilitar la comprensión por parte de clientes y socios externos. Sin embargo, a nivel interno, la empresa se estructura en distintas delegaciones según la región o el sector: Delegación Norte, Delegación Sur, Delegación Centro, Delegación Industria y Delegación Servicios.

Durante mis prácticas como Serveo sigue creciendo y escalando puestos a nivel empresa en España y con una estrategia de internacionalización en sus planes, mi puesto se ha visto afectado pasando de ser soporte a todos los departamentos de estructura (Recursos Humanos, Dpto. Financiero, ...) a dar soporte a la parte de negocio, es decir, a todos los contratos como Airbus, Iberia, hospitales públicos,....

### **Tecnologías y metodologías utilizadas**

Serveo apuesta por la innovación y la optimización de procesos mediante herramientas y enfoques que facilitan el trabajo diario y mejoran la colaboración. Entre las principales tecnologías y metodologías que emplea se encuentran:

- **Power BI, Excel y Power Query** para la visualización, análisis y transformación de datos, permitiendo generar informes dinámicos y dashboards que ayudan a la toma de decisiones.
- **Power Automate y PowerApps** para automatizar flujos de trabajo rutinarios y crear aplicaciones de bajo código que simplifican tareas operativas sin necesidad de programación compleja.
- **Python** para desarrollar aplicaciones personalizadas que facilitan el día a día de los usuarios, desde pequeños scripts de automatización hasta soluciones más completas que integran bases de datos y servicios web.
- **Planner y SharePoint** como herramientas colaborativas, que permiten organizar tareas, gestionar proyectos en equipo y compartir documentación de forma centralizada y ordenada.
- **SQL** para facilitar el manejo de bases de datos y ser capaz de transformar grandes cantidades de datos.

### **Descripción del puesto**

Para este segundo cuatrimestre me he centrado sobretodo en 3 de las herramientas que he mencionado en el apartado anterior aunque la finalidad de mi puesto sigue siendo la misma, capacidad de transformar los datos para que todas las personas puedan entenderlos y crear herramientas que agilicen el día a día de todo el personal de la empresa, las herramientas con las que más he trabajado son:

- **Python**, durante este segundo periodo me he visto en la constante necesidad de crear scripts y ejecutables con python usando librerías distintas y variadas de las que hablaré más adelante en otro de los apartados.
- **Power BI** con el fin de procesar los datos de la compañía y tener una forma interactiva y dinámica de visualizarlos, power BI es la mejor herramienta sobretodo para la parte de visualizarlos, aunque es cierto que para la parte de transformar datos he usado otras herramientas más potentes como Python, mencionada antes o **Big Query**.
- Desarrollo de aplicaciones con **Power apps**, he continuado con el desarrollo de una aplicación que mencionaba el cuatrimestre pasado, para facilitar el registro de la entrega de EPIs, a la vez que lo he usado para crear nuevas aplicaciones y con la ayuda de **Sharepoint** y **Power Automate** he conseguido desarrollar flujo de trabajo que facilitan el día a día al personal de la empresa.

### **Descripción del Sector. Competidores, evolución, tendencias y oportunidades**

El entorno en el que se mueve Serveo es uno de los sectores de servicios más dinámicos y exigentes, marcado por la rápida incorporación de nuevas tecnologías. La adopción de la digitalización y la automatización se ha convertido en un factor decisivo, obligando a las empresas a reinventar sus procesos para no quedarse rezagadas. En este escenario, compiten grandes firmas de facility management y proveedores de servicios integrales, todos ellos en la carrera por desarrollar propuestas cada vez más eficaces e innovadoras. Esta transformación constante abre la puerta a la adopción de herramientas avanzadas de análisis y gestión de datos, lo que mejora notablemente la eficiencia operativa y la capacidad de respuesta ante las demandas del mercado. Consciente de este contexto, Serveo ha puesto en marcha un ambicioso plan de expansión hacia Europa, respaldado en su know-how y en su modelo integral, para llevar su oferta de servicios a nuevos territorios.

### **Posicionamiento de la empresa en el sector. Puntos fuertes y débiles. Aspectos mejorables**

Serveo ocupa un lugar destacado entre las principales firmas de facility management en España, sustentando su liderazgo en una estrategia de crecimiento constante y diversificación en sectores como Industria, Salud y Administración Pública. Su capacidad para gestionar infraestructuras de alto nivel se refleja en la reciente obtención de la certificación ISO 41001, que avala la excelencia en la gestión de inmuebles y servicios de soporte, así como en su rápida integración de equipos y proyectos tras la adquisición de Sacyr Facilities. Además, la incorporación del área de mantenimiento industrial de Dominion ha reforzado su posicionamiento técnico y ha incrementado

su presencia en el sector manufacturero, aportando recursos especializados y mayores oportunidades de negocio. Entre sus puntos fuertes se encuentran la solidez de sus procesos internos, la implementación de tecnologías avanzadas para el análisis y seguimiento de KPI, y una oferta integral que abarca tanto servicios “hard” como “soft”. Sin embargo, para seguir elevando su competitividad, Serveo debería profundizar en la integración total de sus sistemas de automatización y optimizar la comunicación interna entre delegaciones, asegurando así una coordinación aún más ágil y homogénea en todas sus áreas de actuación.

### 2.1.3. Información previa

Durante el primer cuatrimestre ya estuve colaborando en este equipo en proyectos de análisis de datos y automatización, lo que me permitió continuar desarrollando proyectos con ellos y asignando proyectos nuevos. Debido a esa experiencia previa, al continuar con las prácticas ya tenía varias iniciativas asignadas: el equipo, consciente de mis habilidades, me confió tareas alineadas con mis fortalezas. De este modo, me encargaron proyectos de generación de informes avanzados en Power BI, desarrollo de scripts en Python para optimizar procesos operativos y creación de flujos de trabajo mediante las herramientas low-code de Microsoft. Gracias a esa continuidad, no tuve que empezar desde cero; pude centrarme en profundizar en las soluciones concretas que se estaban desarrollando en el departamento.

En mi día a día, me dedico principalmente al análisis de datos con Power BI, construyendo dashboards interactivos que facilitan la toma de decisiones, y a la mejora de la eficiencia operativa mediante Python, desarrollando aplicaciones y automatizaciones que reducen tiempos y errores en tareas rutinarias. Además, gestiono flujos de trabajo utilizando Power Automate y Power Apps, integrando diversas fuentes de información para crear procesos robustos y escalables. Esta combinación de herramientas me permite ofrecer soluciones integrales: desde la extracción y transformación de datos hasta la generación de alertas automatizadas y la publicación de informes dinámicos.

En cuanto a mis estudios universitarios, varias asignaturas me han proporcionado una base técnica sólida que me ha sido de gran utilidad durante mis prácticas. Una de las más relevantes ha sido la asignatura de Inteligencia Artificial, donde se puso un gran énfasis en la limpieza y el preprocesamiento de datos. Durante el curso, trabajamos intensamente en proyectos que requerían la preparación de grandes volúmenes de datos, algo que resulta fundamental en mi rol actual. Además, fue una de las pocas asignaturas en las que trabajamos con Python, lo que me

ha permitido aplicar algunas de las técnicas de programación aprendidas en proyectos más complejos en el entorno laboral. La limpieza de datos es un paso crucial en el análisis de datos y haber practicado tanto esta habilidad en la universidad me ha facilitado muchísimo adaptarme a los procesos de análisis de datos en Serveo.

Otra asignatura que ha sido muy valiosa en mi formación es Programación Avanzada, donde aprendí a estructurar de manera correcta el código y a utilizar buenas prácticas en el desarrollo de software. Esta asignatura me enseñó la importancia de escribir código claro y bien documentado, algo esencial no solo para crear programas funcionales, sino también para trabajar de manera colaborativa en entornos empresariales. Durante esta asignatura, también aprendimos a diseñar diagramas de flujo de código y arquitectura de software, herramientas que, aunque en ese momento parecían algo teóricas, he descubierto que tienen una aplicación muy práctica en mi trabajo diario. Gracias a esos conocimientos, he podido familiarizarme rápidamente con herramientas como Power Automate y Power Apps, dos aplicaciones de Microsoft que se utilizan en Serveo para automatizar procesos y desarrollar aplicaciones internas de una manera eficiente.

Por último, la experiencia práctica adquirida en la universidad, tanto en las asignaturas como en los trabajos de clase, ha sido clave para afrontar los retos técnicos que se presentan en el día a día de mis prácticas. Haber trabajado en proyectos que simulan entornos reales me ha permitido desarrollar una capacidad de adaptación rápida a las herramientas y procesos dentro de la empresa. Además, el conocimiento adquirido en los cursos adicionales que he realizado de manera autónoma me ha dado una ventaja a la hora de integrar diferentes tecnologías en mi flujo de trabajo.

#### 2.1.4. Motivos para elegir estas prácticas

Desde que entré en informática siempre he tenido mucho interés por todo lo que es la ciencia del dato y el manejo de grandes cantidades de datos, entonces cuando estaba buscando prácticas pensé que mejor forma de empezar en el mundo de la informática que en unas prácticas de análisis de datos y se presentó esta oportunidad, así que no dude ni un solo momento en decir que si.

Otro motivo muy importante es que tal y como avanza la inteligencia artificial todo el mundo de los datos va a estar más cotizado a la vez que por estos mismos avances de la IA trabajos como los analistas de datos y otros van a ir desapareciendo o siendo cada vez más difícil entrar en este

mundo, por eso aprovechando ahora las prácticas tengo un trasfondo en mi curriculum y con suerte puedo haber entrado en este mundillo de los datos para quedarme mucho tiempo, ya que de momento me está gustando bastante todo lo que hago y me veo en un futuro siguiendo en este mundillo.

## 2.2. Objetivos de las prácticas

PROYECTO FORMATIVO, actividades a desarrollar por el alumno:

1. Ser partícipe en los proyectos de optimización y mejora de procesos en los que trabaja el departamento de Eficiencia Operativa, desarrollando e implantando mejoras operativas con los diferentes interlocutores (interno y externo).
2. Realizar cuadros de mando y análisis de datos, con la finalidad de obtener conclusiones y establecer recomendaciones que optimicen las operaciones.
3. Elaborar informes y reportes sobre el resultado de los análisis realizados con el objetivo de facilitar la toma de las decisiones a la dirección.
4. Dar soporte en la identificación e implantación de acciones de mejora en los diferentes servicios de la compañía.
5. Colaborar en la implantación y desarrollo de proyectos de implantación de metodología Lean /Six Sigma.
6. Trabajar realizando consultas y análisis en bases de datos en entornos SQL y Google Cloud Platform.

Ahora que ya he finalizado las prácticas puedo describir cuánto he trabajado en cada uno de los puntos que se exigían de mí en el contrato (Los citados arriba). La gran mayoría de mi trabajo se corresponde con los 3 primeros puntos, diría que alrededor de un 90% entre esos 3 puntos con el uso de Power BI(30%), Python(30%), Power Apps(15%) y Power Automate(15%), los puntos 4 y 5 diría que corresponde alrededor de un 8% de mi trabajo por la parte de implantación de los proyectos que iba desarrollando para los distintos departamentos o contratos, y por último las consultas con SQL corresponden con el 2% restante ya que solo se utiliza para la obtención y guardado de datos y no tanto para el manejo y transformación de estos.

## 2.3. Onboarding

Mi departamento se divide en dos áreas principales: eficiencia operativa y datos. Yo formo parte del equipo de datos, que está compuesto por tres personas: Daniel, nuestro líder (Head) y responsable de la línea estratégica; Roberto, con quien colaboro directamente la mayor parte del tiempo; y yo, encargándome de gran parte del tratamiento y la explotación de la información. Entre los tres gestionamos una parte sustancial de los datos de la empresa. A continuación, explico con más detalle las reuniones y la organización interna.

Todos los días, o prácticamente todos, nos reunimos a las 10:00 de la mañana en las llamadas Daily. Estas reuniones pueden ser presenciales o a través de videollamada si alguien está teletrabajando tengo 5 días al mes de teletrabajo que puedo gestionar como quiera aunque lo más común es hacer 1 por semana. Su duración es de unos 10 o 15 minutos, y sirven para comentar los proyectos en los que vamos a trabajar durante la jornada, repasar los problemas surgidos el día anterior y compartir las soluciones implementadas, ya que esas experiencias pueden resultar útiles para el resto del equipo. En estas sesiones también se discuten los nuevos proyectos que llegan desde otros departamentos: la mayoría son canalizados primero a Daniel, quien luego los distribuye entre Roberto y yo, asignándolos mediante Planner según la naturaleza y la complejidad de cada tarea. Por ejemplo, si se trata de un proyecto de datos de gran envergadura—como aquellos que se reportan directamente al CEO o dependen de información financiera—suele quedar en manos de Daniel. Los trabajos más rutinarios, como informes en Power BI o generación de tablas en Excel, suelen asignárselos a Roberto. A mí, por mi parte, me confían proyectos variados que implican código (especialmente Python), desarrollo en Power Apps o la creación de flujos complejos en Power Automate. Aunque al principio tuve ciertas dificultades con Power Apps y Power Automate, las fui comentando en las reuniones diarias y recibí apoyo de mis compañeros; progresivamente, he alcanzado un nivel de manejo en estas herramientas incluso superior al suyo.

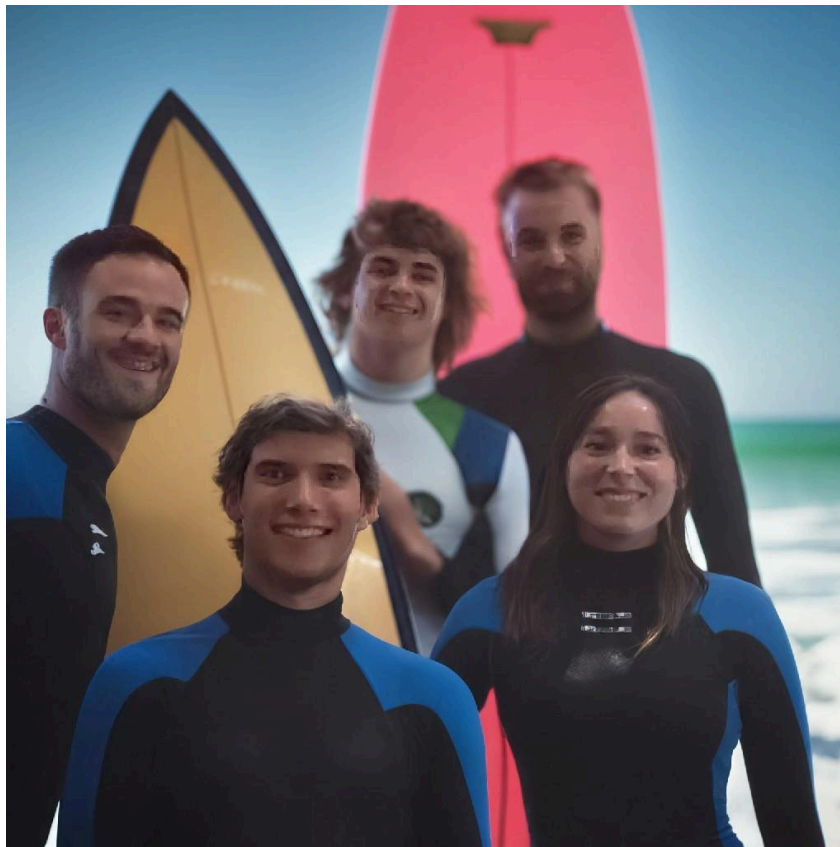


Lunes	Martes	Miércoles	Jueves	Viernes	Sábado	Domingo
31 Mar	1 Abr	02	03	04	05	06
10:00 Diaria - E	10:00 Cancelado	10:00 Diaria - E	10:00 Diaria - E 12:00 Cancelado	10:00 Diaria - E		
07	08	09	10	11	12	13
10:00 Diaria - E 13:00 Organizar P	10:00 Diaria - E 14:00 V Jornada Tr 16:00 revisión Pov	10:00 Diaria - E 15:00 Foro Univer	10:00 Diaria - E	10:00 Diaria - E 14:30 Cancelada		
14	15	16	17	18	19	20
10:00 Diaria - E 12:30 Aplicación E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E		
21	22	23	24	25	26	27
10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E		
28	29	30	1 May	02	03	04
10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E	10:00 Diaria - E		

*Ilustración 2: Horario de onboarding en Serveo. (Elaboración propia)*

Cuando tengo que abordar un proyecto para otro departamento, organizo una reunión con la persona responsable para definir requisitos y pautas. Así, coordinamos desde el alcance del alcance de la entrega hasta los criterios de validación. Esto suele estar reflejado en el calendario interno de la empresa mediante invitaciones y documentos compartidos.

Además de las reuniones diarias, he asistido a varias jornadas y eventos relevantes. Por ejemplo, participé en una jornada de becarios en el Hospital Zendal para colaborar con pacientes de Esclerosis Lateral Amiotrófica (ELA) que se encontraban en el centro de día, aportando apoyo logístico y humano a los usuarios del servicio. También tuve la oportunidad de asistir al Google Cloud Summit Madrid 2025 en IFEMA, donde pude conocer de primera mano los proyectos más recientes de inteligencia artificial presentados por Google, como demos de soluciones de IA generativa para el sector público y privado, modelos de análisis de datos en la nube y casos de éxito de migración de Big Data. Estos eventos me han permitido ampliar mi visión sobre las últimas tendencias tecnológicas y las mejores prácticas de automatización y análisis de datos, conocimientos que luego traslado a mis actividades diarias en Serveo.



Google Arts & Culture

[goo.gle/ArtSelfie2](https://goo.gle/ArtSelfie2)

AI generated with Google

*Ilustración 3: Imagen del google summit 2025 (IA Google)*

## 2.4. Funciones y tareas en las prácticas. (evidencias)

Mi puesto en las prácticas es el de becario en eficiencia operativa y análisis de datos. Al comenzar, entramos dos becarios para el mismo puesto, o al menos con el mismo nombre. Ambos nos incorporamos al Departamento de Eficiencia Operativa como becarios en eficiencia operativa y analistas de datos. Dado que el otro becario tenía un perfil más enfocado a la ingeniería y yo venía de ingeniería informática, me asignaron principalmente a la parte de análisis de datos, mientras que a él lo orientaron más hacia la eficiencia operativa. Sin embargo, dado que vengo de ingeniería informática y mejorar el rendimiento en eficiencia operativa a menudo implica

desarrollar aplicaciones, código o flujos, también me encargaron proyectos relacionados tanto con eficiencia operativa como con análisis de datos.

Para aclarar mejor esta parte, explicaré las tareas que he realizado durante las prácticas, primero enfocándome en la eficiencia operativa y luego en la parte de datos:

## Eficiencia Operativa: Proyectos

- El primer proyecto del que quiero hablar y en uno de los que más he trabajado durante este periodo consiste en un script de Python para la automatización de detección de facturas en los archivos de cobros diarios. El proyecto nos llegó en enero más o menos y resulta que el departamento financiero dedicaba la mayoría de su tiempo y recursos a la detección, dedicaban entre 3 personas 16 horas al día a la extracción de las facturas lo equivalente a 2 FTEs, al acabar el proyecto revisamos las mediciones de tiempos con la nueva herramienta implementada y tardaban alrededor de 1 hora y media de media. El script consiste en la mezcla de una serie de lógicas para la extracción de valores clave de una columna de texto libre de un excel. La primera búsqueda que se hace es con expresiones REGEX,

```
buscar_regex = {'56': r'56\d{8}', '56_extendido': r'56\d{8}(?:/\d{4})*', '57': r'57\d{8}'}
```

*Ilustración 4: Ejemplo expresión REGEX.(Elaboración propia)*

esto funciona ya que el formato de facturas que Serveo maneja es siempre igual, consiste en un 56 o 57 seguido de 8 dígitos, el siguiente paso es en caso de no encontrar una factura buscará usando las librerías de Spacy o FuzzyWuzzy un cliente que esté en el archivo de deudas o en el de clientes de la empresa, una vez se encuentra un cliente o una factura se maneja como una dataframe y se muestran los resultados.

```

1  """
2  IMPORTS Y VARIABLES GLOBALES
3  """
4  import sys
5  import tkinter as tk
6  from tkinter import ttk, filedialog, messagebox
7  from tkinter.font import Font
8  import pandas as pd
9  import re, os, threading, time, signal, string
10 from fuzzywuzzy import fuzz, process
11 import spacy
12 from spacy.matcher import PhraseMatcher
13 import multiprocessing as mp
14 from openpyxl import load_workbook
15 from openpyxl.styles import Font, PatternFill, numbers
16
17 import locale
18 import time
19
20 import es_core_news_sm
21 nlp = es_core_news_sm.load()
22 matcher = PhraseMatcher(nlp.vocab, attr="LEMMA")
23
24 locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')
25

```

*Ilustración 5: Ejemplo de librerías Python.(Elaboración propia)*














Todo esto junto con una interfaz de usuario fácil de usar, intuitiva y con instrucciones claras gracias a la librería Tkinter de python que permitió que en este proceso se redujeran tantas horas.



*Ilustración 6: Interfaz aplicación corporativa.(Elaboración propia)*

## Análisis de Datos: Proyectos

- **Power BI:** Diseñé dos dashboards principales:
  - **Recursos Humanos:** seguimiento del proceso de selección de personal en toda la compañía, con métricas de plazos, estados y ratios de éxito.
  - **Órdenes de Trabajo:** control centralizado de órdenes por contrato, implementando seguridad a nivel de fila para restringir datos según el usuario.
- **Jerarquía de Sociedades:** desarrollé un notebook en Python que unifica dos extractos de SAP en un DataFrame y genera un archivo consolidado de jerarquías. Este recurso alimenta Power BI con dimensiones seguras y actualizadas.

Documentos 				
  Nombre ▾	Modificado ▾	Modificado por ▾	+ Agregar columna	
 AP-JERARQUIA	24 de abril	Herguedas Alonso, Roberto		
  DescargaSAP ...  	5 de febrero	Martinez de Hurtado Aricha, Guillermo		
 JerarquiaMigracion	25 de abril	Herguedas Alonso, Roberto		
 Manual	5 de febrero	Martinez de Hurtado Aricha, Guillermo		
 Municipios	29 de abril	Herguedas Alonso, Roberto		
 Transformados	5 de febrero	Martinez de Hurtado Aricha, Guillermo		
 UTE	20 de mayo	Herguedas Alonso, Roberto		

*Ilustración 7: Organización SharePoint jerarquía .(Elaboración propia)*

- **SQL y BigQuery:** aunque el uso ha sido residual, preparé vistas y consultas para alimentar modelos de Power BI y pipelines en Microsoft Fabric.

## 2.5. Relaciones de problemas planteados y procedimientos para su resolución

### Eficiencia Operativa: Script de detección de facturas

#### 1. Inconsistencias en el texto de las facturas

- **Problema:** Algunos registros incluían espacios extra, guiones “-” o barras “/” que rompían el patrón de búsqueda.
- **Solución:** Añadí una fase de normalización previa que limpia el texto (elimina espacios múltiples, convierte guiones y barras a un formato estándar y pasa todo a minúsculas) para garantizar que el regex encuentre siempre las secuencias “56” o “57” seguidas de ocho dígitos.

## 2. Falsos negativos en la expresión regular

- *Problema:* Los códigos con formatos ligeramente distintos (por ejemplo, “5612345678” o “-5712345678”) no casaban con la regex original.
- *Solución:* Adapté el patrón a `\\D?56\\d{8}` y `\\D?57\\d{8}` para permitir un carácter no numérico previo opcional, y probé exhaustivamente hasta cubrir más del 98 % de los casos reales.

## 3. Rendimiento insuficiente en ficheros voluminosos

- *Problema:* Procesar Excel de 50 000 filas en un único hilo bloqueaba la interfaz Tkinter durante minutos.
- *Solución:* Implementé multiprocessing dividiendo el DataFrame en “chunks” y ejecutándolos en paralelo, lo que recortó el tiempo de ejecución en un 70 %. Además, pasé el trabajo pesado a un hilo secundario para mantener la UI responsiva.

## 4. Asignaciones de cliente incorrectas

- *Problema:* SpaCy y FuzzyWuzzy a veces devolvían coincidencias con puntuaciones bajas, generando etiquetas erróneas.
- *Solución:* Elevé el umbral de similitud al 85 % y, en caso de puntuaciones entre 85 % y 90 %, marqué la fila para revisión manual en lugar de asignarla automáticamente.

## 5. Errores por archivos dañados o mal formateados

- *Problema:* Excel corruptos detenían todo el proceso con excepciones no controladas.
- *Solución:* Rodeé cada paso crítico (lectura de archivo, regex, análisis NL) con bloques try/except, registrando el error en un log y continuando con el resto de los datos; al finalizar, el usuario recibe un informe de incidencias.

## Análisis de Datos: Proyectos en Power BI y Python

## 1. Modelado monolítico en Power BI

- *Problema:* Un único modelo con tablas muy densas causaba “timeouts” y lentitud en la actualización.
- *Solución:* Separé hechos y dimensiones en tablas independientes, eliminé columnas no usadas y habilité la actualización incremental.

## 2. Configuración de seguridad a nivel de fila (RLS)

- *Problema:* La complejidad de la jerarquía de contratos dificultaba asignar permisos correctamente.
- *Solución:* Desarrollé un script en Python que extrae la jerarquía desde SAP y genera una tabla de roles lista para importar en Power BI, simplificando la configuración de RLS.

## 3. Desalineación de jerarquías

- *Problema:* Cambios en los nombres de sociedades provocaban discrepancias entre el archivo regenerado y el extracto original de SAP.
- *Solución:* Incorporé una rutina en Python que compara ambos ficheros y envía alertas automáticas (vía Power Automate) cuando detecta diferencias, para corregir antes de cargar.

## 4. Consultas SQL lentas en BigQuery

- *Problema:* Vistas con múltiples joins provocaban tiempos de respuesta excesivos.
- *Solución:* Optimicé las consultas creando vistas intermedias con filtros previos, particioné tablas por fecha y añadí índices adecuados, reduciendo el tiempo de ejecución un 60 %.

## 2.6. Aprendizajes y desarrollo profesional (habilidades)



## adquiridas)

- **Gestión de proyectos:** planifiqué y prioricé tareas con Planner, mejorando mi organización.
- **Transformación de datos:** dominen Power Query, pandas y SQL para preparar grandes volúmenes.
- **Automatización y scripting:** creé aplicaciones y flujos que liberan al equipo de tareas manuales.
- **Visualización:** interioricé buenas prácticas de diseño de dashboards en Power BI.
- **Trabajo en equipo:** colaboración diaria y uso de Git contribuyeron a un desarrollo fluido.

## 2.7. Metodologías utilizadas

- Tareas asignadas y priorizadas en Planner.
- Daily de coordinación.
- Iteraciones semanales con entrega de prototipos.
- Control de versiones con Git para cada módulo.

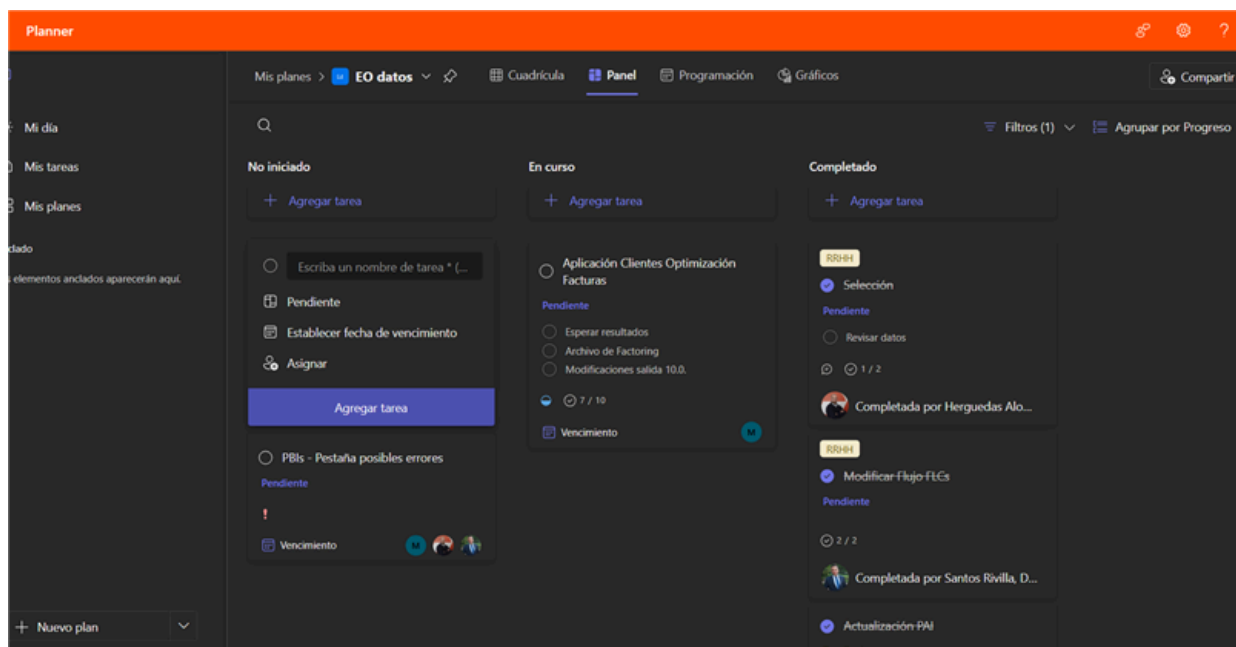


Ilustración 8: Organización del planner del equipo.(Elaboración propia)

## 2.8. Herramientas utilizadas en las prácticas (evidencias)

He empleado tanto herramientas tradicionales como entornos de programación avanzados para cubrir todas las necesidades del departamento:

- **Python (pandas, numpy, NLTK):** Base de mi flujo de trabajo. Con **pandas** y **numpy** manipulo DataFrames para limpiar, transformar y consolidar grandes volúmenes de información; utilizo **NLTK** para tokenizar texto, eliminar stopwords y extraer entidades clave de informes y transcripciones. Esta combinación me ha permitido depurar datos con precisión y preparar los inputs para análisis posteriores.
- **Power Query:** Integrado en Excel y Power BI, lo empleo para ingestar diversas fuentes (CSV, bases de datos, APIs) y aplicar transformaciones en M (filtros, pivotes y uniones) antes de pasar los DataFrames a Python o a los dashboards. Su uso evita escribir código manualmente en Python para tareas repetitivas de limpieza.
- **Power BI:** Principal herramienta de visualización. A menudo he tenido que enfrentarme a limitaciones de filas o problemas de rendimiento al renderizar gráficos complejos con millones de registros. Para solventarlo, diseño consultas DAX optimizadas y desacoplo el modelo de datos en varias tablas intermedias, reduciendo el tamaño del modelo y mejorando los tiempos de carga.

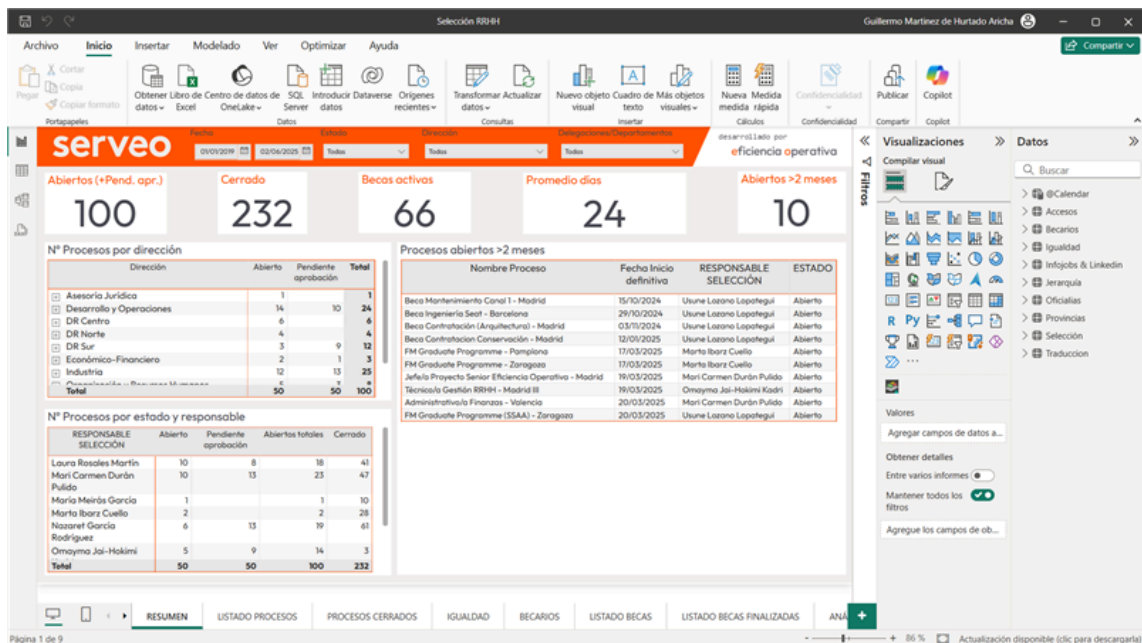
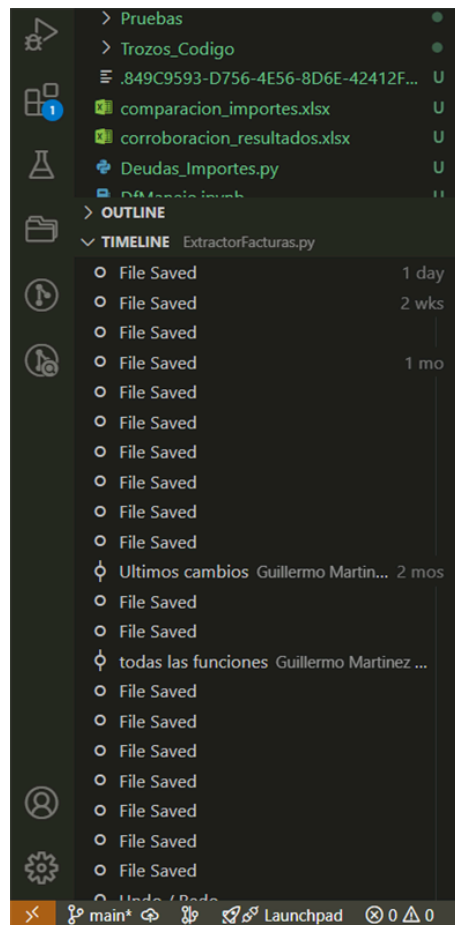


Ilustración 9: Ejemplo de uso PBI.(Elaboración propia)

- **Jupyter Notebooks:** Entorno mixto de documentación y código donde elaboro prototipos rápidos de análisis, combino ejemplos de **NLTK** y **spaCy** y genero visualizaciones preliminares con matplotlib. Esto facilita validar ideas antes de integrarlas en scripts más estructurados.
- **Tkinter** y **Streamlit** (proyectos piloto): Mientras que Tkinter me ha servido para crear aplicaciones de escritorio con formularios de carga y botones de ejecución, he explorado **Streamlit** para desplegar prototipos web de análisis de texto, aprovechando su simplicidad para compartir resultados con usuarios no técnicos.
- **Git:** Control de versiones continuo en cada script y notebook. Hago commits frecuentes al repositorio para documentar cambios en transformaciones de datos y en pipelines de Python, lo que facilita rastrear ajustes cuando surgen discrepancias en los informes.



*Ilustración 10: Ejemplo de uso GIT.(Elaboración propia)*

- **Microsoft Fabric:** Plataforma de ingesta y modelado. Aunque su uso principal no está en esta sección, cabe mencionar que lo empleo para desarrollar pipelines Gen 2 con Power Query y Spark SQL, y para orquestar actu

## 2.9. Logros, resultados y discusión.

Uno de los mayores logros para mí ha sido confirmar mi pasión por el mundo del dato, algo que siempre me ha fascinado desde que empecé en informática. Poder estar en esta beca como analista de datos ya es, en sí mismo, un gran logro personal, y cada día me siento más convencido de que estoy donde quiero estar.

Dentro del departamento, he notado que mi trabajo se valora y eso se refleja en la confianza que han depositado en mí, por ejemplo, dejándome a mí solo montar un BBDD que se pretende que use toda la compañía. Cumpló con las fechas de entrega e incluso a veces se sorprenden con la velocidad que termino alguna tarea. Me encanta ver cómo, desde la automatización de procesos con Power Automate hasta la creación de informes complejos, puedo aportar mi granito de arena para mejorar la eficiencia de la empresa.

Algo que me ha sorprendido mucho es observar la diversidad en la forma de trabajar con los datos. Aunque todos estamos en el mismo equipo, se nota que cada persona tiene su propio estilo y sus propios métodos para trabajar y transformar los datos o los flujos de trabajo. Algunas personas son súper meticulosas y otras más espontáneas, lo que me ha enseñado a valorar la importancia de convertir esos datos en algo fiable y legible para todos. Este contraste me inspira a seguir perfeccionando mis técnicas, porque sé que de ahí nace el verdadero valor de nuestro trabajo.

Mirando hacia el futuro, me gustaría la idea de profundizar aún más en áreas innovadoras como el análisis en tiempo real. Buscando implementar nuevas tecnologías para mejorar el rendimiento del equipo, como por ejemplo he notado mucho la falta de IA en la empresa y ahorraría tiempo de tareas redundantes que se hacen. Personalmente, estoy ansioso por aprender y experimentar, y creo firmemente que, a medida que sigamos explorando estas herramientas, podremos transformar por completo la manera en la que trabajamos con la información.

Creo que con la gente que me rodea en este equipo voy a mejorar mucho, ya que son gente que apoya todas las ideas, mientras no sean ideas locas, y dejan crecer a la gente mucho estando con ellos, aparte de grandes profesionales son buena gente y eso hace que haya algo más de motivación por querer hacer cosas, porque nadie quiere ver a gente buena haciendo de más simplemente porque haya gente que no quiere trabajar.

## 3. DESARROLLO

### 3.1. Introducción

#### 3.1.1. Motivación

Trabajar a diario con Python me ha permitido profundizar en librerías de procesamiento de lenguaje como NLTK y en herramientas de tokenización y análisis de texto. En enero inicié un proyecto que empleaba estas tecnologías y, al principio, pensé en aplicarlas al análisis de emociones en libros u otros textos. Sin embargo, buscando un desafío más dinámico y motivador, me inspiré al ver contenidos en YouTube y se me ocurrió trasladar ese mismo enfoque a vídeos: aprovechar el potencial de la plataforma para extraer y analizar emociones directamente del audio y la transcripción.

#### 3.1.2. Análisis de mercado y necesidades

El mercado de análisis de contenido multimedia está en pleno auge, impulsado por la demanda de insights en tiempo real para áreas como marketing, comunicaciones y atención al cliente. Existen soluciones comerciales que ofrecen pipelines de transcripción y análisis, pero suelen ser servicios en la nube estancos, con costes por minuto transcrito y poca flexibilidad para integración local. Además, no cubren en un único paquete la comparación de distintos algoritmos de sentimiento (modelos estadísticos como VADER, redes neuronales como RoBERTa, y librerías de detección de emociones como Text2Emotion), ni permiten un control completo del flujo de datos y la personalización de umbrales o preprocesamiento.

#### 3.1.3. Objetivos

- Diseñar una interfaz gráfica intuitiva mediante Tkinter que oriente al usuario paso a paso.
- Integrar técnicas de extracción de audio de YouTube y Twitch (pytube, streamlink) y pipeline de transcripción (Whisper, YouTubeTranscriptApi).
- Implementar módulos de preprocesamiento de texto (NLTK) que automaticen la limpieza y tokenización.

- Comparar y visualizar los resultados de diferentes motores de análisis de sentimiento (Text2Emotion, VADER, RoBERTa) en un único dashboard.
- Paralelizar tareas intensivas mediante threading y multiprocessing para optimizar tiempos de respuesta.
- Generar reportes gráficos interactivos embebidos en la aplicación.

### 3.1.4. Requisitos técnicos

- Lenguaje: Python 3.9+ con soporte para typing.
- Dependencias: tkinter, pytube, streamlink, whisper, youtube\_transcript\_api, nltk, text2emotion, transformers, fuzzywuzzy, spacy, matplotlib.
- Recursos: conexión a Internet para descarga de modelos y actualizaciones de NLTK; CPU con al menos 4 núcleos para multiprocessing.
- Entorno: Windows y Linux compatibles; librerías instaladas vía pip y modelos descargados con Hugging Face y NLTK.

### 3.1.5. Análisis de mercado

El análisis de mercado confirma que, aunque hay plataformas SaaS de transcripción y análisis, no existen soluciones integrales de código abierto que combinen transcripción automática de alta calidad (Whisper), extracción de emociones (Text2Emotion) y clasificación de sentimiento robusta (RoBERTa) en un único ejecutable, con un interfaz de bajo código. Este nicho justifica el desarrollo de una herramienta propia, adaptable a necesidades específicas dentro de la empresa.

## 3.2. Marco técnico

- Plataforma de desarrollo: Python como lenguaje principal, aprovechando su ecosistema de librerías de NLP y multimedia.
- Arquitectura: aplicación monolítica con GUI en Tkinter, pero de diseño modularizable; cada funcionalidad (descarga, transcripción, análisis, visualización) implementada en un módulo independiente.
- Modelos de NLP: Whisper (OpenAI) para transcripción, NLTK para tokenización y

- lematización implícita en algunas funciones, VADER para sentimiento, transformers de Hugging Face para RoBERTa, y text2emotion para análisis de emociones.
- Multihilo y multiproceso: threading para mantener la interfaz responsive durante tareas I/O, multiprocessing para distribuir cargas de CPU en análisis de grandes fragmentos de texto.
- Gráficos: matplotlib integrado con TkAgg para mostrar resultados en ventanas emergentes.

### 3.3. Equipo de trabajo y metodología

Este proyecto ha sido responsabilidad mía en su mayor parte: durante cuatro meses he estado diseñando y programando la aplicación de forma autónoma. Mi profesor ha actuado como apoyo puntual, revisando partes del código cuando surgían dudas y sugiriendo mejoras del proyecto.

Para avanzar de forma ordenada, me marqué objetivos semanales: implementar cada funcionalidad (descarga de audio, transcripción, limpieza de texto, análisis de sentimiento) y probarla antes de pasar al siguiente bloque. Esta forma de trabajo me permitió iterar rápido, corregir errores sobre la marcha y asegurar que cada módulo funcionara correctamente antes de integrarlo en el flujo principal.

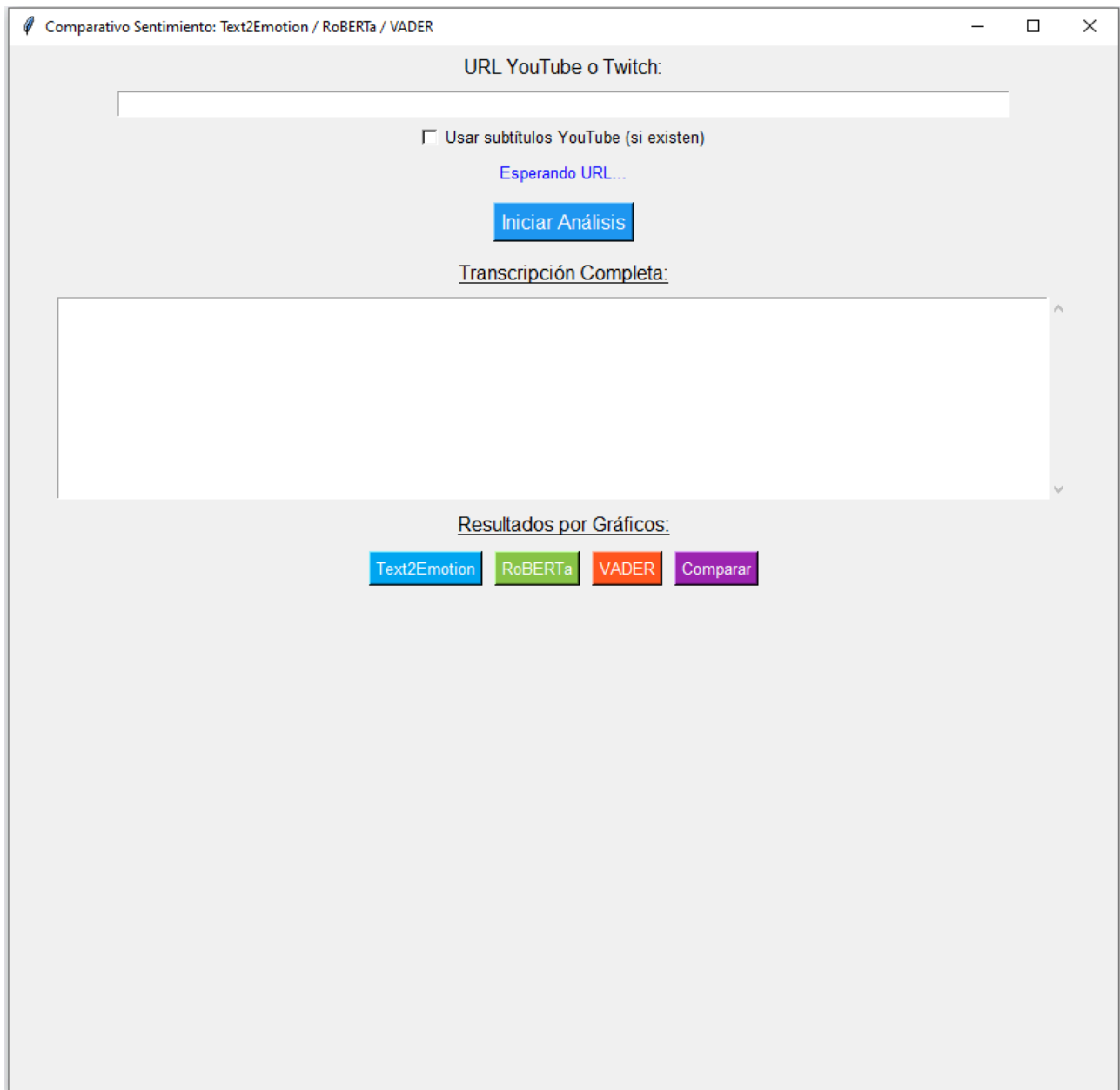
En cuanto a la metodología, utilicé control de versiones con Git para hacer commits frecuentes y documentar los cambios más relevantes. Así podía revertir rápidamente si algo fallaba y mantener un historial claro de las etapas de desarrollo.

### 3.4. Proyecto

#### 3.4.1. Resumen de contribuciones y productos desarrollados

- Módulo de descarga de audio: implementación de funciones para YouTube (pytube).
- Pipeline de transcripción: capa de abstracción que decide entre YouTubeTranscriptApi o Whisper según disponibilidad de subtítulos.
- Preprocesamiento: funciones de NLTK para tokenizar, eliminar stopwords y normalizar texto.
- Análisis de emociones: integración de text2emotion con filtrado previo de stopwords.
- Análisis de sentimiento: configuración de VADER y RoBERTa (fragmentación de texto en

- chunks).
- UI: diseño de interfaz Tkinter con botones, cuadros de texto y ventanas emergentes para gráficos.



*Ilustración 11: Interfaz de la aplicación análisis de sentimientos.(Elaboración propia)*

### 3.4.2. Planificación temporal

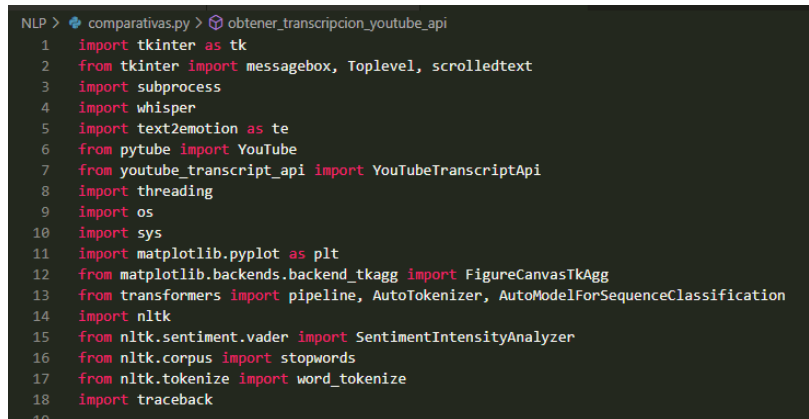
- Paso 1: Pensar en el proyecto de algo que me interese.
- Paso 2: Marcar unas líneas generales de alcance y herramientas posibles.



- Paso 3: Investigación de herramientas compatibles para el proyecto.
- Paso 4: Primeros pasos de código.
  - Paso 4.1: Obtención de transcripciones de Youtube.
  - Paso 4.2: Incorporar Text2Emotion
  - Paso 4.3: Incorporar VADER y RoBERTa
- Paso 5: diseño de la interfaz gráfica.
- Paso 6: Optimizar el código y depurarlo.

### 3.4.3. Recursos empleados

- Hardware: servidor local con 8 núcleos, 16 GB RAM.
- Software: Python 3.9.



```

NLP > comparativas.py > obtener_transcripcion_youtube_api
1 import tkinter as tk
2 from tkinter import messagebox, Toplevel, scrolledtext
3 import subprocess
4 import whisper
5 import text2emotion as te
6 from pytube import YouTube
7 from youtube_transcript_api import YouTubeTranscriptApi
8 import threading
9 import os
10 import sys
11 import matplotlib.pyplot as plt
12 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
13 from transformers import pipeline, AutoTokenizer, AutoModelForSequenceClassification
14 import nltk
15 from nltk.sentiment.vader import SentimentIntensityAnalyzer
16 from nltk.corpus import stopwords
17 from nltk.tokenize import word_tokenize
18 import traceback
19

```

*Ilustración 12: Librerías incorporadas en Python .(Elaboración propia)*

- Modelos descargados: Whisper base, transformers RoBERTa, VADER.

### 3.4.4. Trabajo desarrollado

Es un trabajo que ha llevado una gran parte de investigación por mi parte ya que era algo nuevo, estuve investigando sobre las mejores herramientas para hacer un buen análisis de emociones en python, al final sabía que hacerlo con python gracias a sus librería NLTK era lo más eficaz para desglosar un texto y poder analizarlo.

El primer resultado con el que dí fue Text2Emotion, una librería un poco antigua pero que cumple muy bien las funciones de analizar emociones en textos extensos, las otras dos opciones las descubrí a la vez al ver unos cuantos videos de comparación de modelos, VADER y RoBERTa están entrenados a base de redes sociales, al final al pasarle videos de Youtube o de Twitch

consiste un poco en pasarle frases de redes sociales, pero más en conjunto entonces requiere de una mayor limpieza del texto que con NLTK se consigue.

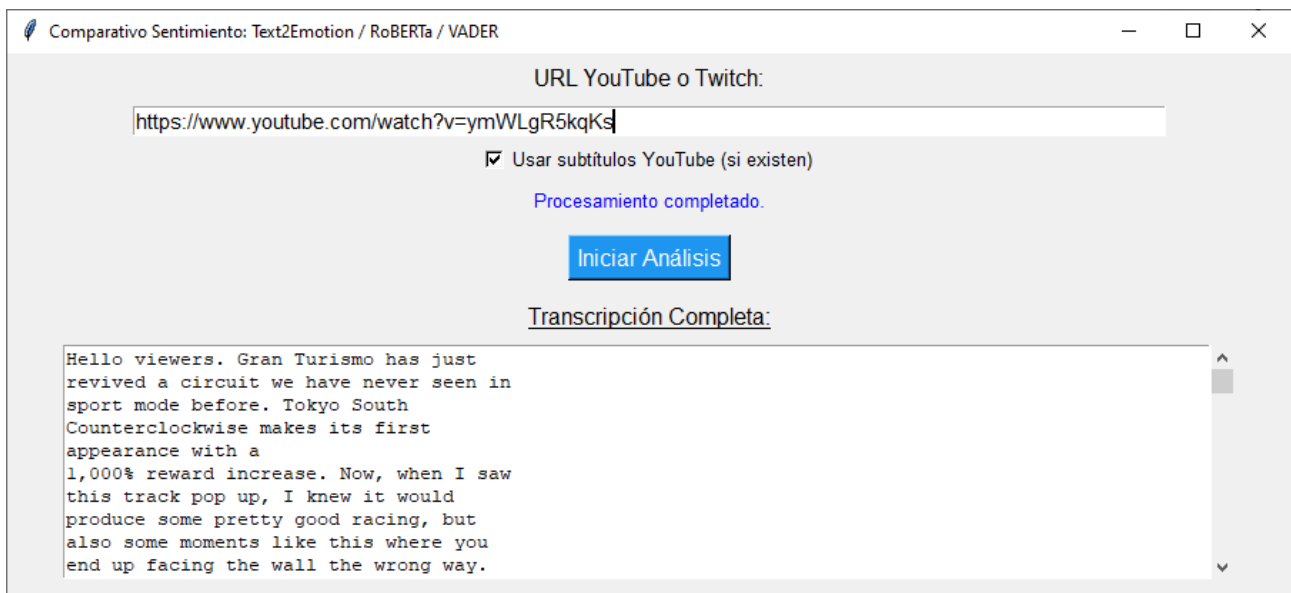
Los siguientes pasos eran conseguir una buena transcripción de los videos de Youtube para lo que también tuve que hacer una pequeña investigación y al final descubrí que hay una librería de python que te conecta con la API de Youtube y puedes obtener las transcripciones automáticas de los videos pasandole el URL a través de una variable.

Una vez hechas las investigaciones fue hora de juntar en código todos estos conocimientos y ponerlo a prueba, lo que me dió buenos resultados.

### 3.5. Resultados y discusión

Como menciono en el apartado anterior este trabajo ha llevado una gran cantidad de investigación por mi parte y estoy muy contento con los resultados que me ha dado.

Para empezar con lo que menos fé tenía era la arte de obtener las transcripciones de los videos, pero cuando descubrí la herramienta correcta, Pytube, cambio todo ya tenía una muy buena forma de empezar el código y empezar a aplicar los algoritmos de análisis de emociones, hasta ese momento solo estaba practicando con textos que sacaba de internet.

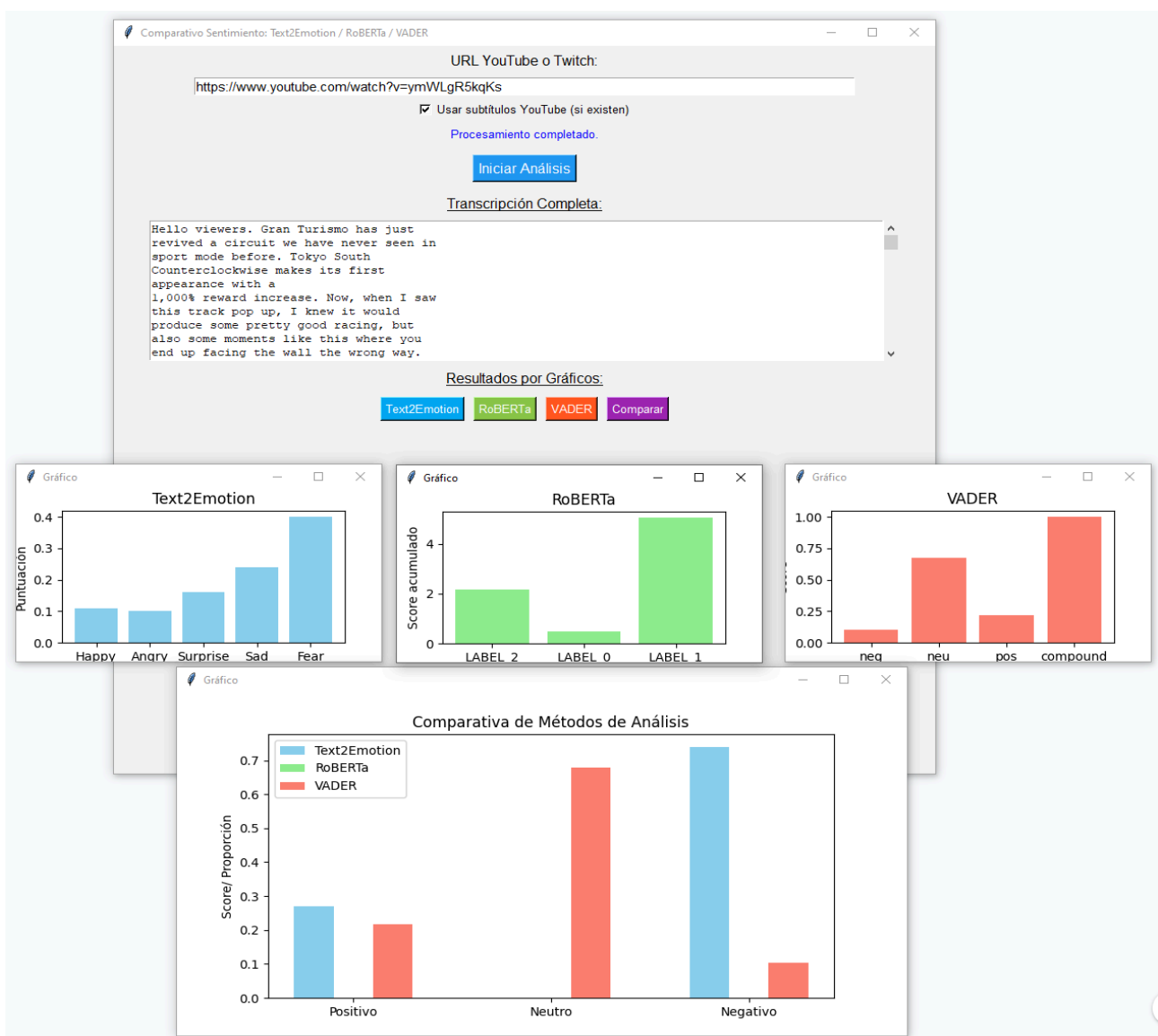


*Ilustración 13: Procesamiento de textos de Youtube.(Elaboración propia)*

Una vez ya podía practicar con los videos y con la librería Text2Emotion decidí que era hora de

probar modelos más modernos ya que esta librería es bastante antigua y todas las palabras neutrales te las mete en la categoría de miedo en lugar de tener una categoría propia como son los dos modelos que incorporé después, tanto VADER como RoBERTa funcionan de forma similar, calculan positividad y negatividad en el texto en intervalos de -1 a 1 en RoBERTa funciona como clusters y hay que etiquetarlos, pero en VADER ya vienen etiquetados.

Cuando incorporé estos nuevos modelos me gustaba como había quedado y quería ya poner unos gráficos para hacer comparaciones de un modelo con otro, lo más sencillo en python eran gráficos de barras de matplotlib y así ya concluiría con un proyecto bastante satisfactorio.



*Ilustración 14: Gráficas comparativas de resultados.(Elaboración propia)*

## 3.6. Conclusiones

Me ha gustado mucho desarrollar este proyecto ya que se complementa muy bien con mi trabajo de análisis de datos, en mi trabajo uso python casi día a día y con este proyecto he sido capaz de ampliar mis conocimientos en algunas librerías muy útiles como son NLTK para toda la parte de tokenización y luego la parte de visualización con matplotlib, aunque sea sencilla de usar siempre que la uso descubro aplicaciones nuevas donde aplicar esta librería, por parte de los modelos va a ser más difícil que los use en mi día a día pero siempre cabe esa posibilidad de que los use para algo.

A parte de lo personal, en cuanto al proyecto yo creo que los resultados son bastante satisfactorios aunque se demuestra que la parte de análisis de emociones queda mucho por aplicar, sobretodo me he dado cuenta en videos que tenían mucho sarcasmo por ejemplo cuando dicen “funciona igual de bien que una tostadora”, nosotros sabemos que es una referencia mala, pero el modelo lo clasifica como algo positivo, entonces se nota que faltan cosas por etiquetar correctamente en los modelos, pero en general todo funciona muy bien.

## 3.7. Líneas futuras

- Incorporar modelos multilingües (XLM-RoBERTa).
- Mejorar las conexiones con Whisper para poder acceder a una mayor variedad de videos que no tengan transcripciones automaticas.
- Migrar la UI a un framework web para accesibilidad remota, siendo Streamlit la mejor opción un framework de python sencillo de usar y fácil de interactuar con el.

## 4. BIBLIOGRAFÍA.

- VADER: <https://pypi.org/project/vaderSentiment/>
- Text2Emotion: <https://pypi.org/project/text2emotion/>
- VADER: <https://www.geeksforgeeks.org/python-sentiment-analysis-using-vader/>
- RoBERTa: [https://huggingface.co/docs/transformers/en/model\\_doc/roberta](https://huggingface.co/docs/transformers/en/model_doc/roberta)
- VADER vs RoBERTa: <https://www.youtube.com/watch?v=QpzMWQvxXWk&t=6s>

## 5. ANEXOS

### Anexo 1: Código de python

```
import tkinter as tk
from tkinter import messagebox, Toplevel, scrolledtext
import subprocess
import whisper
import text2emotion as te
from pytube import YouTube
from youtube_transcript_api import YouTubeTranscriptApi
import threading
import os
import sys
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from transformers import pipeline, AutoTokenizer,
AutoModelForSequenceClassification
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import traceback

# Descargar recursos NLTK
try:
    nltk.download('vader_lexicon', quiet=True)
    nltk.download('punkt', quiet=True)
    nltk.download('stopwords', quiet=True)
    analyzer_vader = SentimentIntensityAnalyzer()
    STOPWORDS = set(stopwords.words('english'))
except Exception:
    print("Error al inicializar NLTK:")
    traceback.print_exc()
    STOPWORDS = set()

# Cargar modelo RoBERTa para sentimiento y extraer etiquetas dinámicas
```

```

try:
    model_name = "cardiffnlp/twitter-roberta-base-sentiment"
    tokenizer_roberta = AutoTokenizer.from_pretrained(model_name)
    model_roberta = AutoModelForSequenceClassification.from_pretrained(model_name)
    classifier_roberta = pipeline('sentiment-analysis', model=model_roberta,
tokenizer=tokenizer_roberta)
    labels_roberta = model_roberta.config.id2label
except Exception:
    print("Error al cargar modelo RoBERTa:")
    traceback.print_exc()
    labels_roberta = {0: 'NEG', 1: 'NEU', 2: 'POS'}

# === TRANSCRIPCIÓN AUTOMÁTICA YOUTUBE via API ===
def obtener_transcripcion_youtube_api(url):
    try:
        if 'youtu.be' in url:
            video_id = url.split('/')[1]
        else:
            video_id = url.split('v=')[1].split('&')[0]
        transcript_list = YouTubeTranscriptApi.get_transcript(video_id)
        texto = '\n'.join(seg['text'] for seg in transcript_list)
        return texto
    except Exception:
        print("Error al obtener transcripción automática de YouTube:")
        traceback.print_exc()
        raise RuntimeError('No existe transcripción automática para este video.')

# === DESCARGA Y TRANSCRIPCIÓN CON WHISPER ===
def descargar_audio_youtube(url, nombre_archivo):
    try:
        yt = YouTube(url)
        audio_stream = yt.streams.filter(only_audio=True).first()
        audio_stream.download(filename=nombre_archivo)
        return nombre_archivo
    except Exception:
        print("Error al descargar audio de YouTube:")
        traceback.print_exc()
        raise

```

```

def descargar_audio_twitch(url, nombre_archivo):
    try:
        comando = f"streamlink {url} best -o {nombre_archivo}"
        subprocess.call(comando, shell=True)
        return nombre_archivo
    except Exception:
        print("Error al descargar audio de Twitch:")
        traceback.print_exc()
        raise

def transcribir_audio_whisper(nombre_archivo):
    try:
        modelo = whisper.load_model('base')
        resultado = modelo.transcribe(nombre_archivo)
        return resultado['text']
    except Exception:
        print("Error al transcribir audio con Whisper:")
        traceback.print_exc()
        raise

# === PREPROCESAMIENTO: REMOVER STOPWORDS ===
def remover_stopwords(texto):
    try:
        tokens = word_tokenize(texto)
        filtered = [t for t in tokens if t.lower() not in STOPWORDS]
        return " ".join(filtered)
    except Exception:
        return texto

# === ANÁLISIS DE EMOCIONES con Text2Emotion ===
def analizar_emociones_text2emotion(texto):
    try:
        texto_filtro = remover_stopwords(texto)
        return te.get_emotion(texto_filtro)
    except Exception:
        print("Error al analizar emociones con Text2Emotion:")
        traceback.print_exc()
        raise

```

```

# === ANÁLISIS DE SENTIMIENTO con RoBERTa (fragmentando texto largo) ===
def analizar_sentimiento_roberta(texto, max_words_per_chunk=200):
    try:
        texto_filtro = remover_stopwords(texto)
        words = texto_filtro.split()
        fragmentos = [" ".join(words[i:i+max_words_per_chunk]) for i in range(0,
len(words), max_words_per_chunk)]
        resultados = []
        for frag in fragmentos:
            res = classifier_roberta(frag)
            for r in res:
                label_id = int(r['label'].split('_')[-1])
                r['label'] = labels_roberta.get(label_id, r['label'])
                resultados.append(r)
        return resultados
    except Exception:
        print("Error al analizar sentimiento con RoBERTa:")
        traceback.print_exc()
        raise

# === ANÁLISIS DE SENTIMIENTO con VADER ===
def analizar_sentimiento_vader(texto):
    try:
        texto_filtro = remover_stopwords(texto)
        return analyzer_vader.polarity_scores(texto_filtro)
    except Exception:
        print("Error al analizar sentimiento con VADER:")
        traceback.print_exc()
        raise

# === PROCESAMIENTO GENERAL ===
def procesar_url(url, usar_youtube, data_container):
    nombre_archivo = 'audio_temp.mp4'
    try:
        # 1. Obtener transcripción (API YouTube si está seleccionado y hay
subtítulos disponibles)
        if usar_youtube and ('youtube.com' in url or 'youtu.be' in url):
            texto = obtener_transcripcion_youtube_api(url)
        else:

```



```

        # Descargar audio y transcribir con Whisper
        if 'youtube.com' in url or 'youtu.be' in url:
            descargar_audio_youtube(url, nombre_archivo)
        elif 'twitch.tv' in url:
            descargar_audio_twitch(url, nombre_archivo)
        else:
            raise ValueError("URL no válida. Solo YouTube y Twitch.")
        texto = transcribir_audio_whisper(nombre_archivo)
        os.remove(nombre_archivo)

    # Guardar transcripción en el contenedor
    data_container['texto_transcripcion'] = texto

    # 2. Análisis Text2Emotion
    data_container['emociones_text2emotion'] = analizar_emociones_text2emotion(texto)

    # 3. Análisis RoBERTa (fragmentando texto largo)
    data_container['sentimiento_roberta'] = analizar_sentimiento_roberta(texto)

    # 4. Análisis VADER
    data_container['sentimiento_vader'] = analizar_sentimiento_vader(texto)

except Exception as e:
    print("Error en procesar_url:")
    traceback.print_exc()
    raise e

# === FUNCIONES AUXILIARES para mostrar gráficos en popups ===
def crear_popup_grafico(figura):
    popup = Toplevel()
    popup.title("Gráfico")
    canvas = FigureCanvasTkAgg(figura, master=popup)
    canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)
    canvas.draw()

# === FUNCIONES DE INTERFAZ para mostrar resultados ===
def mostrar_text2emotion(data_container):
    try:

```

```

        if 'emociones_text2emotion' not in data_container:
            messagebox.showinfo("Info", "Ejecuta primero el análisis.")
            return

        emociones = data_container['emociones_text2emotion']
        fig, ax = plt.subplots(figsize=(4,2))
        ax.bar(list(emociones.keys()), list(emociones.values()), color='skyblue')
        ax.set_title("Text2Emotion")
        ax.set_ylabel("Puntuación")
        crear_popup_grafico(fig)
    except Exception:
        print("Error en mostrar_text2emotion:")
        traceback.print_exc()
        messagebox.showerror("Error", "Falló al mostrar resultados Text2Emotion.")

def mostrar_roberta(data_container):
    try:
        if 'sentimiento_roberta' not in data_container:
            messagebox.showinfo("Info", "Ejecuta primero el análisis.")
            return

        resultados = data_container['sentimiento_roberta']
        agr = {}
        for r in resultados:
            label = r['label']
            agr[label] = agr.get(label, 0) + r['score']
        fig, ax = plt.subplots(figsize=(4,2))
        ax.bar(list(agr.keys()), list(agr.values()), color='lightgreen')
        ax.set_title("RoBERTa")
        ax.set_ylabel("Score acumulado")
        crear_popup_grafico(fig)
    except Exception:
        print("Error en mostrar_roberta:")
        traceback.print_exc()
        messagebox.showerror("Error", "Falló al mostrar resultados RoBERTa.")

def mostrar_vader(data_container):
    try:
        if 'sentimiento_vader' not in data_container:
            messagebox.showinfo("Info", "Ejecuta primero el análisis.")
            return

```

```

vader_score = data_container['sentimiento_vader']
fig, ax = plt.subplots(figsize=(4,2))
ax.bar(list(vader_score.keys()), list(vader_score.values()),
color='salmon')
ax.set_title("VADER")
ax.set_ylabel("Score")
crear_popup_grafico(fig)
except Exception:
    print("Error en mostrar_vader:")
    traceback.print_exc()
    messagebox.showerror("Error", "Falló al mostrar resultados VADER.")

def mostrar_comparativa(data_container):
    try:
        if not all(k in data_container for k in
['emociones_text2emotion', 'sentimiento_roberta', 'sentimiento_vader']):
            messagebox.showinfo("Info", "Ejecuta primero el análisis completo.")
            return
        emo = data_container['emociones_text2emotion']
        rob = data_container['sentimiento_roberta']
        vader = data_container['sentimiento_vader']

        # Agrupar RoBERTa
        agr_rob = {}
        for r in rob:
            label = r['label']
            agr_rob[label] = agr_rob.get(label, 0) + r['score']

        # Convertir Text2Emotion a Pos/Neu/Neg
        pos_t2e = emo.get('Happy',0) + emo.get('Surprise',0)
        neg_t2e = emo.get('Sad',0) + emo.get('Angry',0) + emo.get('Fear',0)
        neu_t2e = max(0, 1 - pos_t2e - neg_t2e)

        fig, ax = plt.subplots(figsize=(8,4))
        labels = ['Positivo', 'Neutro', 'Negativo']
        t2e_vals = [pos_t2e, neu_t2e, neg_t2e]
        rob_vals = [agr_rob.get('POS',0), agr_rob.get('NEU',0),
agr_rob.get('NEG',0)]
        vader_vals = [vader.get('pos',0), vader.get('neu',0), vader.get('neg',0)]

```

```

        x = range(len(labels))
        ax.bar([p - 0.2 for p in x], t2e_vals, width=0.2, label='Text2Emotion',
color='skyblue')
        ax.bar(x, rob_vals, width=0.2, label='RoBERTa', color='lightgreen')
        ax.bar([p + 0.2 for p in x], vader_vals, width=0.2, label='VADER',
color='salmon')

        ax.set_xticks(x)
        ax.set_xticklabels(labels)
        ax.set_ylabel('Score/ Proporción')
        ax.set_title('Comparativa de Métodos de Análisis')
        ax.legend()
        crear_popup_grafico(fig)
    except Exception:
        print("Error en mostrar_comparativa:")
        traceback.print_exc()
        messagebox.showerror("Error", "Falló al mostrar gráfica comparativa.")

# === FUNCIONES ASÍNCRONAS E INTERFAZ ===
def iniciar_proceso(entry_url, var_yt, lbl_status, data_container,
cuadro_transcripcion):
    url = entry_url.get().strip()
    if not url:
        messagebox.showwarning("Atención", "Introduce una URL.")
        return
    lbl_status.config(text="Procesando, por favor espere...")
    threading.Thread(
        target=lambda: run_background(url, var_yt.get(), lbl_status,
data_container, cuadro_transcripcion),
        daemon=True
    ).start()

def run_background(url, usar_youtube, lbl_status, data_container,
cuadro_transcripcion):
    try:
        procesar_url(url, usar_youtube, data_container)
        # Mostrar la transcripción en el cuadro principal
        texto = data_container.get('texto_transcripcion', '')

```

```

        cuadro_transcripcion.delete('1.0', tk.END)
        cuadro_transcripcion.insert(tk.END, texto)
        lbl_status.config(text="Procesamiento completado.")
    except Exception as e:
        lbl_status.config(text="Error en procesamiento.")
        print("Error en run_background:")
        traceback.print_exc()
        messagebox.showerror("Error", str(e))

def interfaz():
    try:
        ventana = tk.Tk()
        ventana.title('Comparativo Sentimiento: Text2Emotion / RoBERTa / VADER')
        ventana.geometry('900x850')
        ventana.configure(bg='#f0f0f0')

        # Asegurar que cerrar la ventana termine el script
        ventana.protocol("WM_DELETE_WINDOW", lambda: (ventana.destroy(),
sys.exit()))

        tk.Label(ventana, text='URL YouTube o Twitch:', bg='#f0f0f0',
font=('Arial',12)).pack(pady=5)
        entry_url = tk.Entry(ventana, width=90, font=('Arial',11))
        entry_url.pack(pady=3)

        var_yt_check = tk.IntVar()
        tk.Checkbutton(
            ventana,
            text='Usar subtítulos YouTube (si existen)',
            variable=var_yt_check,
            bg='#f0f0f0',
            font=('Arial',10)
        ).pack()

        lbl_status = tk.Label(ventana, text='Esperando URL...', bg='#f0f0f0',
fg='blue', font=('Arial',10))
        lbl_status.pack(pady=5)

        tk.Button(

```

```

        ventana,
        text='Iniciar Análisis',
        bg='#2196F3',
        fg='white',
        font=('Arial',12),
        command=lambda: iniciar_proceso(entry_url, var_yt_check, lbl_status,
data_container, cuadro_transcripcion)
    ).pack(pady=8)

    tk.Label(ventana, text='Transcripción Completa:', bg='#f0f0f0',
font=('Arial',12,'underline')).pack(pady=5)

    global cuadro_transcripcion
    cuadro_transcripcion = scrolledtext.ScrolledText(ventana, height=10,
width=100, font=('Courier',10))
    cuadro_transcripcion.pack(pady=3)

    tk.Label(ventana, text='Resultados por Gráficos:', bg='#f0f0f0',
font=('Arial',12,'underline')).pack(pady=5)

    frame_botones = tk.Frame(ventana, bg='#f0f0f0')
    frame_botones.pack(pady=5)

    tk.Button(
        frame_botones,
        text='Text2Emotion',
        bg='#03A9F4',
        fg='white',
        font=('Arial',10),
        command=lambda: mostrar_text2emotion(data_container)
    ).grid(row=0, column=0, padx=5)

    tk.Button(
        frame_botones,
        text='RoBERTa',
        bg='#8BC34A',
        fg='white',
        font=('Arial',10),
        command=lambda: mostrar_roberta(data_container)

```

```

).grid(row=0, column=1, padx=5)

tk.Button(
    frame_botones,
    text='VADER',
    bg='#FF5722',
    fg='white',
    font=('Arial',10),
    command=lambda: mostrar_vader(data_container)
).grid(row=0, column=2, padx=5)

tk.Button(
    frame_botones,
    text='Comparar',
    bg='#9C27B0',
    fg='white',
    font=('Arial',10),
    command=lambda: mostrar_comparativa(data_container)
).grid(row=0, column=3, padx=5)

ventana.mainloop()
except Exception:
    print("Error en interfaz principal:")
    traceback.print_exc()

# Diccionario global para almacenar resultados
data_container = {}

if __name__ == '__main__':
    interfaz()

```