Escuela
Politécnica
Superior

LLMSearch: Buscador multimedia basado en lenguaje natural

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Izan Gandía Ruiz
Tutor:
Iván Gadea Saéz

Universitat d'Alacant
Universidad de Alicante

Mayo 2025

LLMSearch: Buscador multimedia basado en lenguaje natural

Encuentra tu contenido multimedia al instante con solo describirlo.

Autor
Izan Gandía Ruiz

Tutor
Iván Gadea Saéz
Departamento de Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Mayo 2025

Preámbulo

Este proyecto surge de una doble motivación. Por un lado, el profundo interés en explorar y adquirir un conocimiento avanzado sobre el uso y configuración de las inteligencias artificiales multimodales. Por otro lado, la identificación de una problemática recurrente y extendida en la gestión de la información digital personal como es la dificultad persistente para localizar archivos específicos (como imágenes o documentos) en volúmenes grandes de datos. Este desafío se presenta también en contextos más actuales, como la búsqueda de elementos específicos (ej. "stickers") dentro de aplicaciones de mensajería como WhatsApp o Telegram.

Agradecimientos

Este rinconcito es para vosotros, para toda esa gente increíble que ha hecho posible que hoy esté aquí, escribiendo estas líneas.

Primero, a ese montón de compañeros y amigos que me he cruzado en la carrera. He aprendido un millón de cosas con vosotros, no solo de manera académica, sino lecciones de vida que se quedan para siempre. Sin vuestras risas, vuestros ánimos en los momentos de bajón y esa forma de tirar para adelante juntos, no sé si habría encontrado las fuerzas para seguir tantas veces. Sois, en gran parte, la razón de que esté celebrando este logro.

A mi familia, mi pilar fundamental. Gracias por creer en mí incluso cuando yo dudaba, por ponerme las cosas fáciles y por todo el apoyo para que pudiera dedicarme a esto. Sois increíbles. Y un gracias enorme y especial para mi hermano, Abel Gandía Ruiz. Tú fuiste quien me abrió los ojos a este mundo tan interesante de la programación cuando yo no tenía ni idea, quien me animó y me echó una mano para empezar.

También quiero acordarme de mis profes. Algunos habéis sido una inspiración, de esos que te contagian la ilusión y te hacen descubrir la magia en sitios donde nunca te lo hubieras imaginado. Gracias por inspirarme y ayudarme a ser un mejor ingeniero.

Y, cómo no, a mi tutor, Iván Gadea Sáez. Gracias por guiarme con este proyecto, por tu paciencia infinita y por ayudarme a calmar todos los nervios y dudas que me han ido surgiendo.

De verdad, a todos y cada uno, ¡muchísimas gracias!

A quienes me inspiraron a soñar y a programar, recordándome que, si puedo imaginarlo, puedo crearlo. [1]

[1] Alejandro Taboada, creador del canal "Programación ATS"

Índice general

Índice de figuras

Índice de tablas

Índice de Códigos

1. Introducción

La gestión y recuperación eficiente de la información digital se ha convertido en un desafío cotidiano en la era de la sobrecarga informativa. Los volúmenes de datos personales y profesionales que almacenamos en nuestros dispositivos crecen exponencialmente, mientras que las herramientas tradicionales de búsqueda a menudo resultan insuficientes para localizar archivos específicos de manera rápida y precisa. Este proyecto se adentra en esta problemática, proponiendo una solución innovadora basada en los avances recientes en Inteligencia Artificial (IA) y sistemas de Generación Aumentada por Recuperación (RAG).

1.1. Panorama Actual: Desafíos en la Recuperación de Información

Los métodos convencionales para la organización y búsqueda de archivos digitales se basan en gran medida en metadatos explícitos, como nombres de archivo, fechas o etiquetas manuales. Sin embargo, estas aproximaciones tienen limitaciones significativas:

- **Insuficiencia de los metadatos tradicionales:** A menudo, los metadatos son inexistentes, incompletos o no capturan la semántica real del contenido del archivo (especialmente en el caso de imágenes, vídeos o audios).
- **Falta de precisión en las búsquedas:** Las búsquedas basadas en palabras clave pueden ser ambiguas y no siempre interpretan correctamente la intención del usuario, llevando a resultados irrelevantes o a la omisión de la información que el usuario desea encontrar.
- **Desafíos técnicos y éticos en IA:** Si bien la IA ofrece nuevas vías, también enfrenta retos. Los modelos pueden carecer de la precisión necesaria para ciertas tareas o, en el caso de modelos generativos, incurrir en "alucinaciones" (generar información incorrecta pero plausible). Además, el propio entrenamiento del modelo puede afectar a la interpretación sobre el archivo que se quiere describir, generando resultados no equitativos o discriminatorios, lo que plantea también importantes consideraciones éticas.
- **Riesgos de seguridad y privacidad de los datos:** La externalización del almacenamiento a la nube y el uso de IA para la catalogación y el análisis de ficheros introducen nuevas formas de ataque y preocupaciones sobre la seguridad y privacidad. Estos sistemas pueden ser vulnerables a accesos no autorizados, fugas de datos o ciberataques, tanto en la infraestructura cloud como en los propios modelos de IA. La información sensible contenida en los archivos, o incluso los metadatos enriquecidos generados por la IA, podrían quedar expuestos, ser alterados o perderse. Esto es especialmente crítico cuando se manejan datos confidenciales o personales, donde una brecha de seguridad no solo implica la pérdida de información valiosa, sino también posibles consecuencias legales, financieras y reputacionales significativas.

Este contexto muestra la necesidad de sistemas más inteligentes y contextuales capaces de comprender el contenido de los archivos de forma más profunda, más allá de sus metadatos superficiales, y que al mismo tiempo garanticen la integridad y confidencialidad de la información.

1.2. Avances Tecnológicos Fundamentales

Para abordar los desafíos mencionados, este proyecto se apoya en los desarrollos más recientes en el campo de la Inteligencia Artificial, particularmente en las siguientes áreas:

1.2.1. Inteligencia Artificial Multimodal: Convergencia de Lenguaje y Visión

La Inteligencia Artificial ha experimentado avances exponenciales, especialmente con el auge del Procesamiento del Lenguaje Natural (PLN) y la Visión Artificial. La multimodalidad representa la capacidad de los sistemas de IA para procesar, comprender y generar información a partir de múltiples tipos de datos (o "modalidades") simultáneamente, como texto, imágenes, audio y vídeo.

- Procesamiento del Lenguaje Natural (PLN): Permite a las máquinas comprender, interpretar y generar lenguaje humano. Los Grandes Modelos de Lenguaje (Large Language Model (LLM)), como Generative Pre-trained Transformer (GPT) (Generative Pre-trained Transformer) y sus variantes, han revolucionado este campo, demostrando una capacidad asombrosa para entender el contexto, generar texto coherente e incluso razonar sobre la información proporcionada.
- Visión Artificial: Es la disciplina que permite a las máquinas "ver" e interpretar el contenido de imágenes y vídeos. Implica tareas como la detección de objetos, el reconocimiento facial, la segmentación de imágenes y la generación de descripciones visuales.
- Modelos Multimodales (ej. Contrastive Language-Image Pre-training (CLIP))[1]: Modelos como CLIP (Contrastive Language-Image Pre-training) de OpenAI son un ejemplo paradigmático de esta convergencia. CLIP aprende representaciones visuales a partir de descripciones en lenguaje natural, permitiendo realizar búsquedas de imágenes mediante consultas textuales con una alta comprensión semántica, o viceversa. Funciona entrenando un codificador de imágenes y un codificador de texto para predecir qué imágenes se emparejan con qué textos en un gran conjunto de datos.

1.2.2. Optimización de Modelos: Cuantización y Modelos Ligeros

Para la aplicación práctica de estos modelos, especialmente en entornos con recursos limitados (como dispositivos personales), es crucial considerar su eficiencia.

- Modelos Cuantizados: La cuantización es un proceso que reduce la precisión numérica de los pesos y activaciones de un modelo de red neuronal (por ejemplo, de punto flotante de 32 bits a enteros de 8 bits). Esto disminuye significativamente el tamaño del modelo y acelera la inferencia, con una pérdida de precisión a menudo mínima.

[1] Más información sobre CLIP de OpenAI disponible en: `https://openai.com/es-ES/index/clip/`

- Modelos Ligeros (Lightweight Models): Son arquitecturas de redes neuronales diseñadas específicamente para ser computacionalmente eficientes y tener un tamaño reducido, facilitando su despliegue en dispositivos móviles o embebidos sin sacrificar excesivamente el rendimiento.

1.2.3. Sistemas de Generación Aumentada por Recuperación (RAG)

La Generación Aumentada por Recuperación (RAG) es una técnica que mejora el rendimiento de los LLM al conectarlos con fuentes de conocimiento externas. En lugar de depender únicamente de la información (potencialmente desactualizada o incompleta) aprendida durante su entrenamiento, un sistema RAG funciona en dos fases:

1. Recuperación (Retrieval): Dada una consulta del usuario, el sistema primero busca y recupera fragmentos de información relevante de una base de datos, un conjunto de documentos o un corpus de conocimiento. Esta base de datos puede estar compuesta por embeddings (representaciones vectoriales densas) del contenido de los archivos.

2. Generación (Generation): La información recuperada se proporciona como contexto adicional al LLM junto con la consulta original. El LLM utiliza este contexto enriquecido para generar una respuesta más precisa, relevante y fundamentada.

Los sistemas RAG ofrecen ventajas significativas, como la reducción de alucinaciones, la capacidad de citar fuentes del contexto dado y la facilidad para actualizar la base de conocimiento sin necesidad de reentrenar el LLM completo. Si bien existen diversas arquitecturas RAG (ej. generando consultas SQL, incorporando texto directamente al prompt, o utilizando embeddings), el enfoque basado en embeddings suele ofrecer un buen equilibrio entre eficiencia y calidad de los resultados, aunque presenta desafíos como la gestión de la ventana de contexto del LLM, punto crucial a tener en cuenta si se utilizan sobre dispositivos personales.

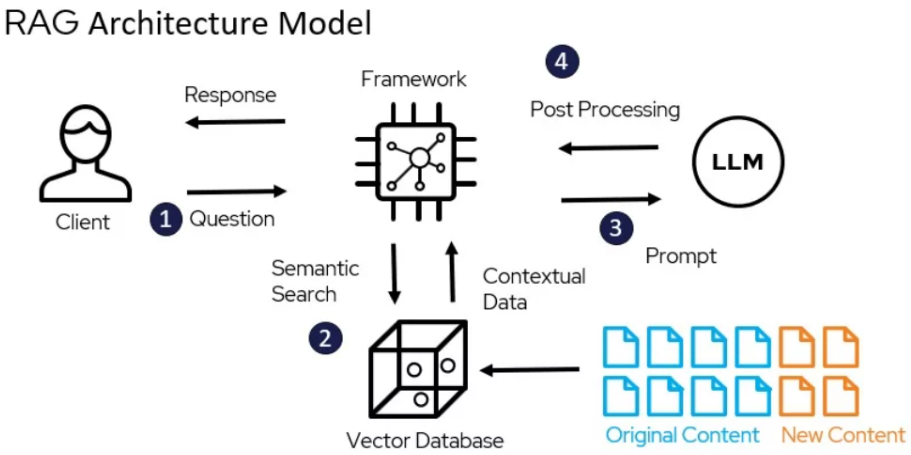


Figura 1.1: Esquema visual del funcionamiento de un sistema RAG, mostrando el flujo desde la consulta del usuario, pasando por la recuperación de información relevante, hasta la generación de la respuesta final por el LLM.

1.2.4. Justificación del Proyecto

Este Trabajo Final de Grado (TFG) busca abordar la necesidad de obtener información de manera rápida, sencilla y muy precisa desarrollando un sistema inteligente de búsqueda de archivos que permita a los usuarios encontrar información utilizando consultas en lenguaje natural, trascendiendo las limitaciones de las búsquedas basadas en metadatos tradicionales. La aplicación de modelos multimodales permitirá indexar el contenido semántico de diversos tipos de ficheros, y la arquitectura RAG proporcionará un marco robusto para recuperar la información más relevante y presentarla de forma útil al usuario.

2. Estado del Arte

Antes de adentrarse en los detalles técnicos, es esencial establecer el contexto actual en el dominio de los buscadores multimedia basados en lenguaje natural. Esta revisión permitirá asentar una fundamentación teórica y metodológica sólida, comprender los desafíos y las limitaciones identificadas en investigaciones previas e identificar las brechas en el conocimiento existente, así como las oportunidades para realizar contribuciones significativas en LLMSearch.

2.1. Modelos de Lenguaje Natural (LLMs) para Búsqueda

Los **modelos de lenguaje de gran tamaño (LLMs)** han revolucionado el procesamiento del lenguaje natural en años recientes. Modelos como *GPT-3* y *GPT-4* (base de **ChatGPT**) demuestran que, con miles de millones de parámetros entrenados en enormes corpus de texto, es posible comprender y generar lenguaje con notable fluidez y contexto. Estos modelos capturan representaciones semánticas ricas, lo que habilita nuevas formas de **búsqueda semántica** y recuperación de información.

Características clave:

- **Búsqueda por significado**: En lugar de limitarse a coincidencias de palabras clave, un LLM puede interpretar la intención de una consulta en lenguaje natural y relacionarla con documentos relevantes aunque no compartan palabras literalmente.
- **Embeddings semánticos**: Técnicas como *embeddings* de oraciones (usando modelos tipo BERT o Sentence Transformers) convierten documentos y consultas a vectores en un espacio vectorial común, donde la similitud de coseno permite recuperar los contenidos más cercanos en significado.
- **Retrieval-Augmented Generation (RAG)**: Los LLMs pueden integrarse en pipelines donde primero se recuperan documentos candidatos y luego el modelo genera una respuesta o resumen usando esos textos.
- **Interfaz conversacional**: Modelos tipo ChatGPT permiten refinar iterativamente las consultas de búsqueda mediante diálogo, mejorando la precisión de resultados en consultas ambiguas.

Los avances más recientes se centran en mejorar la **eficiencia y apertura** de estos modelos. Mientras GPT-4 (de OpenAI) es de uso cerrado y con un tamaño muy grande no divulgado (>100B parámetros), han emergido modelos de código abierto como *LLaMA* (Meta) y sus variantes, que con 7–70B parámetros logran desempeños competitivos.

2.2. Modelos Visión-Lenguaje para Imágenes

En un buscador multimedia, es esencial manejar consultas sobre contenido visual (imágenes) usando lenguaje natural. Aquí destacan los **modelos visiolingüísticos** o **modelos de visión-lenguaje (VLMs)**, que conectan representaciones de imágenes con representaciones textuales en un espacio común.

2.2.1. CLIP y Embeddings Multimodales

Un hito fue el modelo **CLIP** de OpenAI, que entrena conjuntamente un codificador de texto (transformer) y un codificador visual (Red Neuronal Convolucional o Visión Transformer) para proyectar ambos tipos de entrada en **vectores de embedding** de la misma dimensión. Mediante aprendizaje contrastivo en 400 millones de pares imagen–texto, CLIP logró que textos e imágenes con contenido semántico equivalente quedaran cercanos en el espacio vectorial.

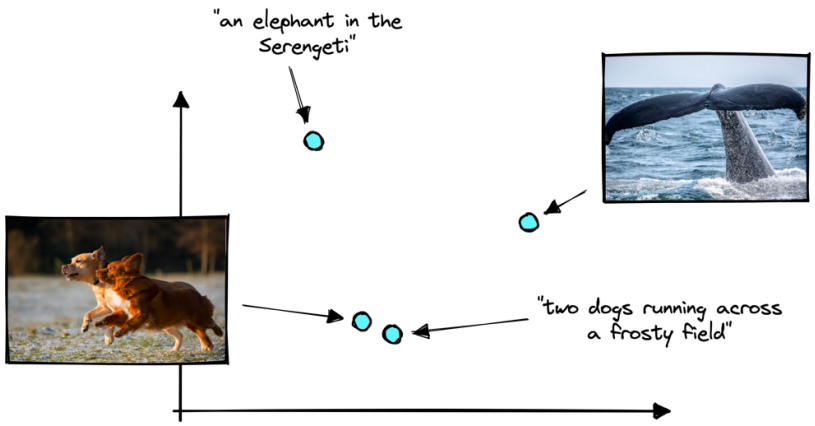


Figura 2.1: Ejemplo conceptual de un espacio vectorial multimodal entrenado por CLIP, donde imágenes y descripciones semánticas correspondientes se representan mediante vectores cercanos.

2.2.2. Modelos Generativos de Descripción de Imágenes

Otra línea de desarrollo son los **modelos generativos de descripción de imágenes**. Estos sistemas realizan *image captioning*, es decir, generan en lenguaje natural una descripción para una imagen dada.

Modelos recientes como **BLIP-2** combinan:

- Un encoder visual pre-entrenado (e.g. CLIP ViT)
- Un modelo de lenguaje grande congelado
- Un transformador ligero intermedio (Q-Former)

Esta arquitectura puentea la brecha visión-lenguaje eficientemente: el encoder de imagen extrae características visuales y el LLM genera la descripción textual.

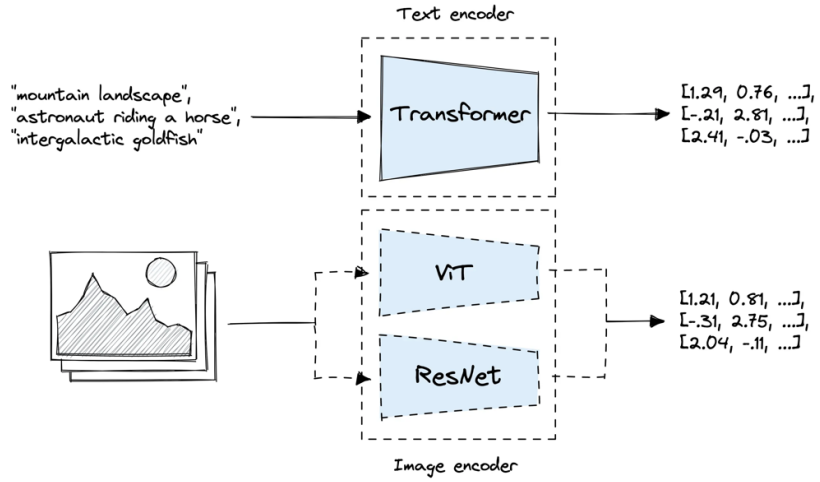


Figura 2.2: Arquitectura del modelo CLIP: encoder de texto y encoder de imagen que proyectan al mismo espacio de embedding.

2.2.3. VQA y Diálogo Multimodal

Junto con BLIP-2, han surgido numerosos modelos abiertos que permiten **Pregunta-Respuesta Visual (VQA)** y diálogo multimodal:

- **LLaVA** (Large Language and Vision Assistant): Usa GPT-4 para generar datos sintéticos de entrenamiento y afina un modelo basado en *Vicuna* (derivado de LLaMA) acoplado a un encoder visual.
- **Moondream**: Un VLM open-source de apenas 2 billones de parámetros (2B) que puede correr en tiempo real incluso en CPU o dispositivos móviles. Demuestra capacidades notables en:
  - Generación de descripciones detalladas
  - Respuesta a preguntas visuales
  - Detección de objetos en modalidad cero-shot
  - OCR básico (leer texto en imágenes)
- **JoyCaption**: Un modelo de captioning de imágenes libre y sin censura, pensado originalmente para generar descripciones ricas que ayuden a entrenar modelos de difusión.
- **GPT-4 con visión** (GPT-4V): Demuestra capacidades impresionantes al responder con acierto a entradas que combinan imagen + texto, aunque su naturaleza propietaria limita su uso en entornos académicos.

En síntesis, el estado del arte en imagen+lenguaje ofrece dos enfoques complementarios para búsqueda multimedia:

1. *Embeddings* multimodales tipo CLIP para realizar **búsqueda directa por similitud** entre consultas textuales y contenido visual

2. *Modelos generativos visiolingüísticos* para **describir o entender imágenes en texto**, permitiendo indexar y razonar sobre imágenes mediante lenguaje natural

2.3. Modelos Multimodales para Vídeo

Extender la búsqueda basada en lenguaje natural al dominio de **vídeo** conlleva retos adicionales, pues los vídeos combinan secuencias de imágenes con audio (y a veces texto incrustado).

2.3.1. Técnicas de Procesamiento de Video

1. **Análisis por frames**:
   - Extraer fotogramas representativos del video
   - Aplicar modelos de visión-lenguaje a cada frame
   - Convertir el problema de vídeo a un conjunto de imágenes con marcas de tiempo
2. **Procesamiento de audio**:
   - Usar modelos de **Reconocimiento Automático de Voz (ASR)** como *Whisper*
   - Transcribir con alta calidad el diálogo o narración presente en videos
   - Indexar cada video por su transcripción textual completa
3. **Modelos video-texto end-to-end**:
   - *VideoCLIP*: Extiende la idea de CLIP al dominio temporal
   - Transformers específicos para vídeo que realizan *video captioning*

2.3.2. Arquitecturas para Búsqueda en Video

Una arquitectura emergente es combinar los enfoques anteriores en un pipeline RAG multimodal:

- Indexar por un lado los *frames* visuales mediante embeddings
- Indexar por otro lado las transcripciones de voz como texto
- Ante una consulta, recuperar fragmentos candidatos por similitud visual o textual
- Usar un modelo de lenguaje para sintetizar ambas fuentes y determinar la respuesta

2.3.3. Modelos Unificados Multimodales

- **MiniGPT-4**: Acepta secuencias de imágenes como entrada (simulando un video corto)
- **MiniCPM-V**: Soporta entradas de video, generando una descripción general
- **Gemini** (Google): Integra visión, video y sonido en un mismo LLM
- **ImageBind** (Meta): Aprende una representación común para imágenes, texto, audio y otros sensores

2.4. Análisis de Audio y Búsqueda mediante Sonido

Para completar un buscador verdaderamente multimedia, se debe considerar contenido **audio** (fuera de videos, es decir, archivos de sonido o música).

2.4.1. Procesamiento de Habla

- Si el audio es habla (podcasts, grabaciones, conferencias), aplicar **ASR** con modelos como Whisper
- Obtener una transcripción textual que se convierte en texto indexable
- Permitir búsquedas por palabras clave o semántica usando LLMs o embeddings

2.4.2. Audio No Verbal

Para audio que no es voz (sonidos ambientales, música, efectos sonoros):

- **CLAP (Contrastive Language-Audio Pretraining)**: Entrena un encoder de audio y uno de texto conjuntamente
- Permite buscar efectos de sonido a partir de descripciones textuales
- Facilita la clasificación cero-shot de audio

2.4.3. Modelos Generadores de Descripciones Auditivas

- **AudioCaption** (Microsoft): Genera frases descriptivas de clips de audio
- Permite describir cada audio en texto y luego indexar esas descripciones
- Facilita un acceso más semántico a archivos de sonido

2.5. Comparativa de Modelos Representativos

	Modalidades	Tamaño	Características principales
ChatGPT (GPT-4)	Texto (y visión en GPT-4V)	>100 B?	LLM propietario de OpenAI, rendimiento puntero en comprensión y generación de lenguaje.
MiniCPM-V 2.5	Texto, Imágenes, Vídeo, Audio	8 B	Open-source, eficiente para despliegue en dispositivos; consultas multimodales.
Moondream 2	Imágenes–Texto	2 B	VLM ultraligero con VQA, captioning, detección y OCR en CPU en tiempo real.
Whisper	Audio–Texto	1.6 B	ASR multilingüe de código abierto, muy robusto ante acentos y ruido.

Tabla 2.1: Comparativa de modelos representativos en lenguaje y multimodalidad.

En la tabla se observa la distinción entre modelos propietarios (e.g. ChatGPT) y una diversidad de iniciativas abiertas. Para **LLMSearch**, los módulos open-source (MiniCPM-V, Moondream, Whisper, CLAP, etc.) pueden combinarse para construir un sistema completo:

- Transcripción de audio con Whisper
- Indexado de sonidos con CLAP

- Descripción de imágenes con Moondream o BLIP-2
- Orquestación conversacional con un LLM general (Vicuna/LLaMA)

2.6. Conclusión

El estado del arte ofrece los bloques fundamentales para un buscador multimedia por lenguaje natural. Este TFG integrará y adaptará estas tecnologías de vanguardia en una única plataforma unificada (*LLMSearch*), evaluando su rendimiento y proponiendo mejoras para lograr búsquedas multimodales más precisas y naturales.

3. Objetivos

3.1. Objetivo general

El objetivo principal de este TFG es diseñar y desarrollar un buscador multimedia inteligente que permita a los usuarios realizar búsquedas avanzadas utilizando lenguaje natural. De esta manera, el usuario podrá localizar documentos de texto, imágenes, vídeos o archivos de audio explorando el contenido semántico intrínseco de los ficheros, trascendiendo las limitaciones de las búsquedas basadas únicamente en metadatos explícitos.

La idea es crear una herramienta que facilite a los usuarios encontrar contenido multimedia de manera eficiente y precisa mediante descripciones detalladas en lenguaje natural. Por ejemplo, se podría buscar una fotografía específica entre miles con una consulta como: "busca una foto en la que salía un elefante levantando la trompa y que la hice en Tailandia hace unos 5 o 6 años"; o encontrar un archivo PDF relevante mediante una búsqueda del tipo: "encuentra los datos para la declaración de la renta de 2016". De esta forma, se pretende obtener un sistema de búsqueda que no solo identifique el archivo específico que se busca, sino que también tenga la capacidad de extraer datos relevantes del contenido del archivo para responder a preguntas específicas formuladas en la consulta, aprovechando las capacidades de los modelos de lenguaje aumentados por recuperación (RAG).

3.2. Objetivos específicos

Adicionalmente, se plantean los siguientes objetivos secundarios que complementan y dan soporte al objetivo principal:

3.2.1. Estudio de modelos multimodales

Estudiar diferentes modelos multimodales con el fin de seleccionar aquellos que ofrezcan los mejores resultados en términos de precisión y eficiencia (tiempo de respuesta razonable).

3.2.2. Selección de solución de base de datos

Investigar y seleccionar una solución de base de datos adecuada para el almacenamiento y consulta eficiente de metadatos enriquecidos y embeddings vectoriales generados por los modelos de IA.

3.2.3. Diseño de arquitectura modular

Diseñar una arquitectura de sistema que sea modular, escalable y eficiente, permitiendo la integración de los diferentes componentes y facilitando futuras expansiones o mejoras.

3.2.4. Desarrollo de interfaz gráfica

Desarrollar una interfaz gráfica de usuario (GUI) intuitiva y amigable que permita a los usuarios interactuar fácilmente con el sistema, realizar búsquedas, visualizar los resultados obtenidos y gestionar sus archivos.

4. Metodología

4.1. Organización

La metodología que se va a seguir es Scrum adaptado simplificado. Scrum es una metodología ágil que facilita la gestión de proyectos complejos a través de equipos autoorganizados que trabajan en ciclos de desarrollo (sprints) de forma iterativa e incrementa y que promueve la entrega continua de valor. Dado que el proyecto se realizará por un único estudiante, este asumirá el rol de Product Owner, Desarrollador y Scrum Master, mientras que el tutor actuará como cliente y pondrá los requisitos y los posibles cambios a lo largo del proyecto. Se harán sprints de aproximadamente 2 semanas desde el inicio del cuatrimestre donde se verán avances, se resolverán dudas y se definirán los objetivos del siguiente sprint.

4.2. Apartado técnico

Para la ejecución y desarrollo del presente TFG, se ha dispuesto del siguiente entorno técnico, tanto a nivel de hardware como de software. Esta configuración ha sido la base sobre la cual se han realizado todas las pruebas, desarrollos y validaciones del sistema propuesto.

4.2.1. Equipamiento Hardware

El equipo informático utilizado para el desarrollo del proyecto cuenta con las siguientes especificaciones:

- **Procesador (CPU):** AMD Ryzen 9 7900X3D 4.4GHz/5.6GHz
- **Memoria RAM:** Corsair Vengeance RGB DDR5 6000MHz 64GB 2x32GB CL30
- **Tarjeta Gráfica (GPU):** RTX 4070 Ti SUPER Trinity 16GB GDDR6X
- **Almacenamiento:** SSD NVMe Samsung 970 EVO Plus de 1TB
- **Sistema Operativo Principal:** Windows 11 Pro / Ubuntu 22.04 LTS

4.2.2. Software y Herramientas de Desarrollo

El desarrollo del software se ha apoyado en las siguientes herramientas, lenguajes y bibliotecas:

- **Entorno de Desarrollo Integrado (IDE):** Visual Studio Code
- **Control de Versiones:** Git con repositorios alojados en GitHub
- **Gestor de Entornos y Paquetes:** pip de Python

- **Navegador Web (para pruebas de interfaz):** Google Chrome y OperaGX en sus versiones más recientes
- **Herramientas de Documentación:** LaTeX (MiKTeX), Excalidraw (para diagramas)

5. Desarrollo

La construcción de un sistema inteligente para la búsqueda y gestión de archivos personales requiere la integración de diversas tecnologías y herramientas consolidadas en el ámbito del desarrollo de software y la inteligencia artificial. Este capítulo tiene como objetivo revisar el estado del arte de los componentes tecnológicos clave que se han considerado o que forman la base para la implementación del presente proyecto. Se analizarán diferentes opciones en áreas fundamentales como la orquestación de tareas, la detección de cambios en el sistema de archivos, las soluciones de bases de datos para el almacenamiento de metadatos y embeddings, la contenerización para el despliegue y, finalmente, los frameworks para el desarrollo de la interfaz de usuario. Esta revisión permitirá contextualizar las decisiones de diseño tomadas y justificar la selección de las herramientas específicas utilizadas en el desarrollo de la solución.

5.1. Orquestador de tareas

La gestión eficiente de flujos de trabajo complejos, especialmente aquellos que involucran procesamiento de datos y tareas de machine learning, es crucial para el sistema propuesto. Un orquestador de tareas permite automatizar, programar y monitorizar estas secuencias de operaciones. A continuación, se presentan las herramientas que se han estudiado en el presente trabajo para seleccionar la más adecuada para la coordinación de las diversas etapas del procesamiento de archivos.

5.1.1. Prefect

Prefect se presenta como una moderna plataforma de orquestación de flujos de trabajo, escrita principalmente en Python. Está diseñada específicamente para permitir a los desarrolladores diseñar, programar, ejecutar y monitorizar pipelines de datos y flujos de machine learning de manera fiable y escalable, con un enfoque en la simplicidad y la experiencia del desarrollador.

5.1.1.1. Ventajas

- **Facilidad de uso:** Prefect ofrece una sintaxis intuitiva y una configuración sencilla, lo que facilita la definición y gestión de flujos de trabajo complejos.
- **Flexibilidad:** Permite la orquestación de tareas en entornos locales, en la nube o híbridos, adaptándose a diversas necesidades.
- **Monitoreo y gestión:** Incluye herramientas integradas para el monitoreo, registro y manejo de errores en tiempo real.

5.1.1.2. Desventajas

- **Madurez:** Aunque ha ganado popularidad, Prefect es relativamente nuevo en comparación con otras herramientas más consolidadas.
- **Comunidad:** Su comunidad es más pequeña, lo que puede limitar la disponibilidad de recursos y soporte.

5.1.2. Kafka

Apache Kafka es un sistema de mensajería distribuido de código abierto, reconocido por su alto rendimiento y capacidad para manejar flujos de datos en tiempo real. Aunque su función principal es la de broker de mensajes, a menudo se utiliza en arquitecturas complejas para desacoplar sistemas y como parte de pipelines de datos más amplios, pudiendo actuar como un componente en la orquestación de eventos.

5.1.2.1. Ventajas

- **Alto rendimiento:** Kafka es conocido por su capacidad para manejar grandes volúmenes de datos con baja latencia.
- **Escalabilidad:** Diseñado para escalar horizontalmente, puede manejar cargas de trabajo crecientes de manera eficiente.
- **Ecosistema robusto:** Cuenta con una amplia gama de herramientas y conectores que facilitan su integración con otros sistemas.

5.1.2.2. Desventajas

- **Complejidad:** La configuración y gestión de Kafka pueden ser complejas, especialmente para usuarios sin experiencia previa.
- **Requisitos de recursos:** Para un rendimiento óptimo, Kafka suele requerir una infraestructura robusta, lo que puede ser excesivo para proyectos más pequeños.

5.1.3. Airflow

Apache Airflow es una plataforma de código abierto ampliamente adoptada para la creación, programación y monitorización programática de flujos de trabajo. Originalmente desarrollada por Airbnb, permite definir flujos de trabajo como Grafos Acíclicos Dirigidos (DAGs) de tareas, utilizando Python para su definición.

5.1.3.1. Ventajas

- **Popularidad y comunidad:** Amplia adopción y una comunidad activa que proporciona numerosos recursos y soporte.
- **Flexibilidad:** Permite la programación y monitoreo de flujos de trabajo complejos.

5.1.3.2. Desventajas

- **Curva de aprendizaje:** Puede ser complejo de configurar y requiere conocimientos avanzados para su implementación efectiva.

5.2. Script de detección de cambios en un path y Servidor

Un componente esencial del sistema es la capacidad de detectar automáticamente la creación, modificación o eliminación de archivos dentro de los directorios personales del usuario. Esta funcionalidad es la que desencadena el proceso de análisis y catalogación. En esta sección se revisan diferentes lenguajes y bibliotecas que ofrecen mecanismos para el monitoreo del sistema de archivos y que podrían servir de base para implementar un servidor que gestione estas detecciones.

5.2.1. Python

Python, debido a su versatilidad y extenso ecosistema de bibliotecas, ofrece múltiples opciones para la detección de cambios en el sistema de archivos. Estas soluciones varían en cuanto a su dependencia de la plataforma y su nivel de abstracción.

- **Watchdogs:** Es una biblioteca multiplataforma escrita en Python, diseñada específicamente para observar eventos del sistema de archivos (como creación, eliminación, modificación de archivos o directorios) de manera eficiente.

- **pyinotify:** Esta biblioteca es un wrapper de Python para la API inotify del kernel de Linux, lo que permite monitorear eventos del sistema de archivos de forma muy eficiente en sistemas Linux, pero no es portable a otras plataformas.

- **inotify-simple:** Se trata de un wrapper más sencillo y ligero alrededor de la API inotify de Linux, que ofrece una interfaz de programación más directa para tareas básicas de monitoreo de archivos en sistemas Linux.

- **inotifyx:** Similar a pyinotify, esta biblioteca proporciona acceso al subsistema inotify de Linux. Está diseñada para ofrecer una API estable, pero su uso también está restringido a plataformas Linux.

- **Polling Methods:** En plataformas donde mecanismos como inotify no están disponibles, o para requisitos más simples, se puede implementar un mecanismo de sondeo (polling). Este método implica verificar periódicamente el estado del sistema de archivos para detectar cambios, aunque puede ser menos eficiente.

5.2.2. Node.js

Node.js, como entorno de ejecución de JavaScript del lado del servidor, también proporciona herramientas robustas para interactuar con el sistema de archivos, incluyendo el monitoreo de cambios.

- **chokidar:** Es una biblioteca popular y eficiente para Node.js que normaliza y mejora el comportamiento de fs.watch y fs.watchFile, ofreciendo una solución multiplataforma robusta para vigilar cambios en el sistema de archivos.

5.2.3. Java

Java, siendo una plataforma madura y ampliamente utilizada, incluye en su biblioteca estándar (NIO.2) mecanismos nativos para el monitoreo de eventos del sistema de archivos.

- **WatchService:** Es una API integrada en Java (desde Java 7) que permite registrar un directorio (o directorios) con el servicio de vigilancia. Este servicio monitorea los cambios en los objetos registrados y los encola para su procesamiento.

5.2.4. C++/C/C#

Los lenguajes de la familia C (C++, C, C#) ofrecen acceso a APIs de bajo nivel del sistema operativo o bibliotecas específicas de cada plataforma para implementar la vigilancia de archivos, permitiendo un control granular y un alto rendimiento.

- **FileSystemWatcher:** En el ecosistema .NET (C#), la clase FileSystemWatcher permite monitorear cambios en el sistema de archivos (archivos y directorios) y reaccionar a ellos. Para C/C++, se suele recurrir a las APIs específicas del sistema operativo como inotify en Linux o ReadDirectoryChangesW en Windows.

5.2.5. Go

Go, conocido por su eficiencia y concurrencia, cuenta con bibliotecas desarrolladas por la comunidad para el monitoreo del sistema de archivos, aprovechando las capacidades nativas de los sistemas operativos.

- **fsnotify:** Es una biblioteca popular en Go para el monitoreo del sistema de archivos que proporciona una interfaz común sobre las APIs específicas de cada plataforma (inotify, kqueue, FSEvents, ReadDirectoryChangesW).

5.2.6. Rust

Rust, un lenguaje de programación moderno enfocado en la seguridad y el rendimiento, también dispone de bibliotecas multiplataforma para la notificación de eventos del sistema de archivos.

- **notify:** Es una biblioteca de Rust multiplataforma para vigilar cambios en el sistema de archivos, ofreciendo una API unificada sobre diferentes backends específicos del sistema operativo.

5.3. Base de datos

El almacenamiento persistente de metadatos, tanto los extraídos directamente de los archivos como los generados por la IA (incluyendo embeddings vectoriales), es fundamental. La elección de la base de datos impacta en el rendimiento, la escalabilidad y la facilidad de consulta. Se exploran opciones relacionales y no relacionales, cada una con sus propias fortalezas y debilidades.

5.3.1. Relacional

Las bases de datos relacionales se caracterizan por su estructura tabular, esquemas predefinidos y el uso de SQL (Structured Query Language) para la manipulación de datos. Son adecuadas para datos con relaciones bien definidas y donde la consistencia transaccional (ACID) es prioritaria.

5.3.1.1. SQLite

SQLite es un sistema de gestión de bases de datos relacional autocontenido, sin servidor, que no requiere configuración y es transaccional (ACID). Toda la base de datos (definiciones, tablas, índices y los propios datos) se almacena como un único archivo en el disco del host, lo que la hace extremadamente portable y fácil de integrar en aplicaciones.

5.3.1.1.1. Ventajas

- **Ligereza y simplicidad:** SQLite es una biblioteca de base de datos integrada que no requiere una configuración de servidor independiente, lo que facilita su implementación y uso.
- **Portabilidad:** Al almacenar toda la base de datos en un único archivo, es fácil de transferir y gestionar, especialmente útil para aplicaciones móviles o integradas.
- **Rendimiento en entornos de bajo recurso:** Funciona eficientemente incluso en sistemas con recursos limitados.

5.3.1.1.2. Desventajas

- **Concurrencia limitada:** SQLite permite múltiples lecturas simultáneas, pero las escrituras se gestionan de una en una, lo que puede ser un cuello de botella en aplicaciones con alta concurrencia de escritura.
- **Escalabilidad:** No está diseñada para manejar grandes volúmenes de datos o aplicaciones que requieren escalabilidad horizontal.

5.3.1.2. MariaDB

MariaDB Server es un popular sistema de gestión de bases de datos relacional de código abierto, creado por los desarrolladores originales de MySQL. Está diseñado para ser un reemplazo directo de MySQL, ofreciendo mayor velocidad, nuevas características y una comunidad más abierta y vibrante.

5.3.1.2.1. Ventajas

- **Rendimiento y escalabilidad:** MariaDB ofrece un alto rendimiento y puede manejar una gran cantidad de transacciones, siendo adecuada para aplicaciones empresariales.
- **Compatibilidad con MySQL:** Como un fork de MySQL, mantiene una alta compatibilidad, facilitando la migración desde MySQL.
- **Soporte para almacenamiento en columnas:** Incluye el motor ColumnStore, optimizado para cargas de trabajo analíticas.

5.3.1.2.2. Desventajas

- **Complejidad en la configuración:** Requiere una configuración y gestión más complejas en comparación con SQLite.
- **Requisitos de recursos:** Necesita más recursos del sistema, lo que puede ser excesivo para aplicaciones pequeñas o integradas.

5.3.2. No relacional

Las bases de datos no relacionales, o NoSQL, ofrecen modelos de datos flexibles, escalabilidad horizontal y están optimizadas para tipos específicos de datos o patrones de acceso. Son una alternativa cuando la rigidez de los esquemas relacionales es una limitación o se requiere un alto rendimiento para grandes volúmenes de datos no estructurados o semiestructurados. Dentro de esta categoría, las bases de datos vectoriales han ganado prominencia para aplicaciones de IA.

5.3.2.1. MongoDB

MongoDB es una base de datos NoSQL orientada a documentos, de código abierto y multiplataforma. Almacena los datos en documentos flexibles similares a JSON (en formato BSON), lo que permite que los campos varíen de un documento a otro y que la estructura de los datos cambie con el tiempo, facilitando la evolución de las aplicaciones.

5.3.2.1.1. Ventajas

- **Flexibilidad del esquema:** Al ser una base de datos NoSQL orientada a documentos, permite almacenar datos en un formato flexible similar a JSON, adaptándose fácilmente a cambios en la estructura de los datos.
- **Escalabilidad horizontal:** Diseñada para escalar horizontalmente mediante sharding, lo que facilita el manejo de grandes volúmenes de datos y altas tasas de tráfico.
- **Alto rendimiento en operaciones de lectura/escritura:** Optimizada para manejar operaciones simultáneas de lectura y escritura de manera eficiente.

5.3.2.1.2. Desventajas

- **Consumo de recursos:** Requiere una cantidad significativa de recursos, especialmente en implementaciones a gran escala.
- **Falta de soporte para transacciones complejas:** Aunque MongoDB ha mejorado en este aspecto con transacciones multi-documento ACID, las transacciones complejas que involucran múltiples colecciones pueden no ser tan robustas o directas como en bases de datos relacionales tradicionales.

5.3.2.2. ChromaDB

ChromaDB es una base de datos vectorial de código abierto diseñada específicamente para facilitar el desarrollo de aplicaciones con IA que requieren el almacenamiento y la búsqueda de embeddings. Permite a los desarrolladores añadir capacidades de memoria a largo plazo y búsqueda semántica a sus modelos de lenguaje y otras aplicaciones de aprendizaje automático de forma sencilla. Puede operar en memoria para prototipado rápido o utilizar almacenamiento persistente en disco.

5.3.2.2.1. Ventajas

- **Especializada en embeddings:** Su diseño está optimizado para almacenar, gestionar y realizar búsquedas de similitud eficientes sobre grandes cantidades de vectores de embeddings.
- **Facilidad de uso y API intuitiva:** Ofrece una API simple, especialmente para desarrolladores de Python, lo que reduce la curva de aprendizaje y acelera la integración.
- **Integraciones con ecosistema de IA:** Se integra de forma nativa con frameworks populares como LangChain y LlamaIndex, simplificando la construcción de flujos de trabajo de IA.
- **Ligera y embebible:** Puede ejecutarse localmente sin necesidad de un servidor complejo, siendo adecuada para desarrollo, prototipado y aplicaciones más pequeñas.
- **Código abierto:** Permite su uso sin restricciones de licencia y fomenta la contribución de la comunidad.

5.3.2.2.2. Desventajas

- **Madurez y escalabilidad para producción masiva:** Aunque evoluciona rápidamente, puede no tener la misma robustez o características avanzadas de escalabilidad horizontal y gestión de clústeres que soluciones de bases de datos vectoriales más maduras o servicios gestionados en la nube para cargas de trabajo extremadamente grandes.
- **Funcionalidades de BD tradicional limitadas:** No está diseñada para ser una base de datos de propósito general. Carece de soporte para consultas relacionales complejas, transacciones ACID en el sentido tradicional o esquemas rígidos que ofrecen las bases de datos SQL.

- **Operaciones y gestión avanzada:** Para despliegues a gran escala, las herramientas de monitorización, backup, y recuperación pueden ser menos sofisticadas en comparación con sistemas de bases de datos más establecidos.

5.4. Docker

La contenerización se ha convertido en un estándar para el desarrollo, despliegue y ejecución de aplicaciones, garantizando la consistencia entre diferentes entornos y simplificando la gestión de dependencias. Docker es la plataforma líder en este ámbito. Docker es una plataforma de software de código abierto que permite automatizar el despliegue de aplicaciones dentro de contenedores de software ligeros y portátiles. Un contenedor empaqueta una aplicación y todas sus dependencias, bibliotecas y archivos de configuración necesarios para que se ejecute de forma aislada.

5.4.1. Ventajas

- **Portabilidad:** Los contenedores Docker aseguran que el software se ejecute de manera consistente en cualquier entorno que soporte Docker, desde el portátil del desarrollador hasta servidores de producción en la nube o locales.
- **Aislamiento:** Cada componente del sistema puede ejecutarse en su propio contenedor, con sus propias dependencias, evitando conflictos entre ellas y con el sistema anfitrión.
- **Facilidad de despliegue:** Simplifica significativamente la distribución y actualización de aplicaciones, permitiendo ciclos de desarrollo y despliegue más rápidos y fiables.

5.4.2. Desventajas

- **Consumo de recursos:** Aunque los contenedores son más ligeros que las máquinas virtuales, el uso de Docker y sus contenedores introduce una capa adicional que consume recursos del sistema (CPU, memoria, disco).
- **Complejidad adicional:** La gestión de contenedores, redes, volúmenes y la orquestación de múltiples contenedores (e.g., con Docker Compose o Kubernetes) requiere conocimientos específicos sobre Docker y sus conceptos asociados.

5.5. Interfaz

La interfaz de usuario (UI) es el punto de interacción principal del usuario con el sistema, permitiéndole realizar búsquedas, visualizar resultados y gestionar sus archivos. La elección de un framework de desarrollo para la UI impacta en la experiencia del usuario, la velocidad de desarrollo y la mantenibilidad de la aplicación. Se consideran varios frameworks web modernos.

5.5.1. Angular

Angular, desarrollado por Google, es un framework de desarrollo de aplicaciones web basado en TypeScript, completo y opinado. Proporciona una plataforma integral para construir aplicaciones web complejas y escalables de una sola página (SPA), ofreciendo herramientas para routing, gestión de estado, y más, directamente desde su núcleo.

5.5.1.1. Ventajas

- **Framework completo:** Angular ofrece una solución integral con herramientas integradas para el desarrollo de aplicaciones web robustas.
- **Arquitectura estructurada:** Su naturaleza opinada y el uso de TypeScript facilitan la escalabilidad y el mantenimiento de aplicaciones complejas a largo plazo.

5.5.1.2. Desventajas

- **Curva de aprendizaje pronunciada:** Requiere tiempo para dominar conceptos como TypeScript, RxJS, y la inyección de dependencias, que son fundamentales en Angular.
- **Complejidad innecesaria para proyectos simples:** Su estructura y conjunto de herramientas pueden resultar excesivos para aplicaciones pequeñas o con funcionalidades limitadas.

5.5.2. React

React, mantenido por Meta (anteriormente Facebook) y una comunidad de desarrolladores individuales y empresas, es una biblioteca de JavaScript de código abierto para construir interfaces de usuario o componentes de UI. Se enfoca principalmente en la capa de vista (el "V" en MVC) y a menudo se utiliza junto con otras bibliotecas para construir aplicaciones completas.

5.5.2.1. Ventajas

- **Biblioteca flexible:** React se centra en la construcción de interfaces de usuario, permitiendo la integración con una amplia variedad de bibliotecas y herramientas según las necesidades específicas del proyecto (e.g., para routing, gestión de estado).
- **Amplia comunidad y recursos:** Su popularidad asegura una gran cantidad de tutoriales, bibliotecas de terceros, y soporte por parte de una comunidad activa y extensa.

5.5.2.2. Desventajas

- **Necesidad de configuraciones adicionales:** Para funcionalidades más allá de la UI básica (como enrutamiento o gestión de estado global), es necesario integrar y configurar bibliotecas adicionales, lo que puede aumentar la complejidad inicial del proyecto.

5.5.3. Vue

Vue.js (comúnmente referido como Vue) es un framework de JavaScript de código abierto, progresivo y accesible, utilizado para construir interfaces de usuario y aplicaciones de una sola página. Se distingue por su facilidad de integración con proyectos existentes y otras bibliotecas, y por su diseño que permite adoptarlo gradualmente.

5.5.3.1. Ventajas

- **Simplicidad y facilidad de uso:** Vue es conocido por su curva de aprendizaje suave y su documentación clara, lo que permite una adopción rápida por parte de los desarrolladores.

- **Flexibilidad:** Adecuado tanto para proyectos pequeños donde se integra en partes de una página existente, como para el desarrollo de aplicaciones de una sola página más complejas.

5.5.3.2. Desventajas

- **Menor adopción en grandes empresas:** Aunque está ganando popularidad rápidamente, Vue aún no tiene la misma penetración en entornos corporativos grandes en comparación con Angular o React, lo que podría traducirse en menos ofertas de empleo o recursos específicos para escenarios empresariales muy complejos.

5.5.4. Astro

Astro es un framework web moderno diseñado para construir sitios web rápidos y centrados en el contenido. Su principal característica es la arquitectura de "islas" (Astro Islands), que permite renderizar componentes de UI en el servidor y enviar menos JavaScript al cliente por defecto, mejorando significativamente el rendimiento de carga y la interactividad inicial.

5.5.4.1. Ventajas

- **Optimización para contenido estático y rendimiento:** Astro está diseñado para generar sitios web estáticos o renderizados en servidor (SSR) muy rápidos, lo que es beneficioso para aplicaciones donde el rendimiento y el SEO son críticos.

- **Integración con otros frameworks:** Permite utilizar componentes de UI escritos en React, Vue, Svelte, y otros frameworks populares dentro de los proyectos Astro, ofreciendo flexibilidad al desarrollador.

5.5.4.2. Desventajas

- **Menor madurez para aplicaciones altamente interactivas:** Aunque soporta componentes interactivos, su enfoque principal en el contenido estático y la minimización de JavaScript del lado del cliente puede hacerlo menos ideal para aplicaciones web muy complejas y altamente dinámicas en comparación con SPAs tradicionales.
- **Ecosistema en crecimiento:** Al ser relativamente nuevo, Astro puede carecer de la amplitud de recursos, herramientas y comunidad que tienen otros frameworks más establecidos, aunque está creciendo rápidamente.

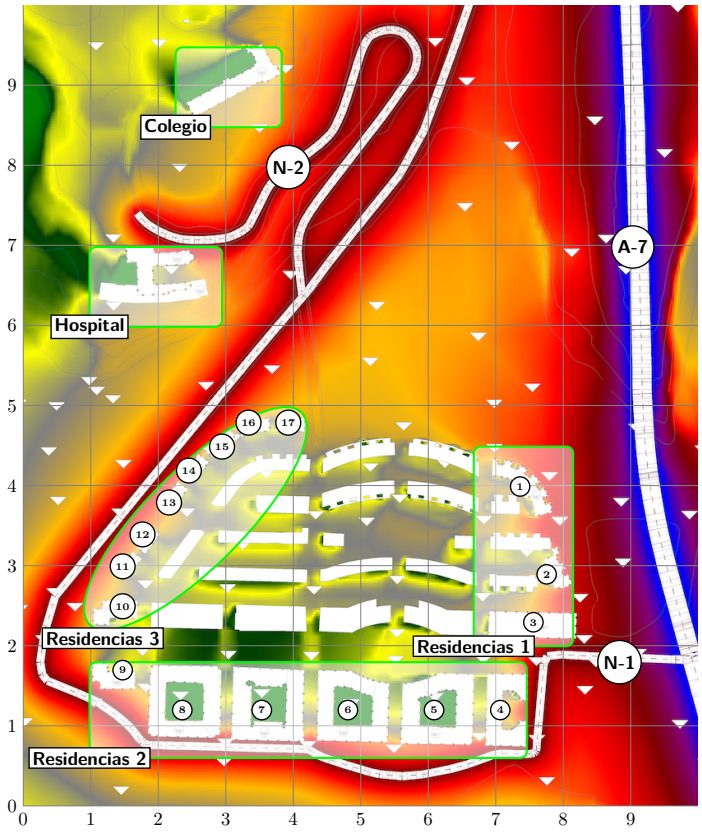
6. Resultados (Con ejemplos de gráficos)

6.1. Diagramas

Gracias al paquete *Tikz* se pueden incluir multitud de medios gráficos, diagramas, capas sobre imágenes, etc. Existen múltiples formas de realizarlo, para ello es recomendable consultar la guía de iniciación disponible aquí: `http://cremeronline.com/LaTeX/minimaltikz.pdf` y también el manual completo disponible aquí: `http://osl.ugr.es/CTAN/graphics/pgf/base/doc/pgfmanual.pdf`.

A continuación se muestran algunos ejemplos. Revisa el archivo .tex para ver cómo se utilizan.

Imagen a la que se le ha añadido cuadros y texto desde latex:



En muchas ocasiones es necesario realizar un diagrama de bloques, más abajo se muestra

un ejemplo de ello. En la red hay multitud de ejemplos que pueden ser fácilmente modificables para un fin concreto, como por ejemplo en esta web: `http://www.texample.net/tikz/examples/tag/block-diagrams/`.

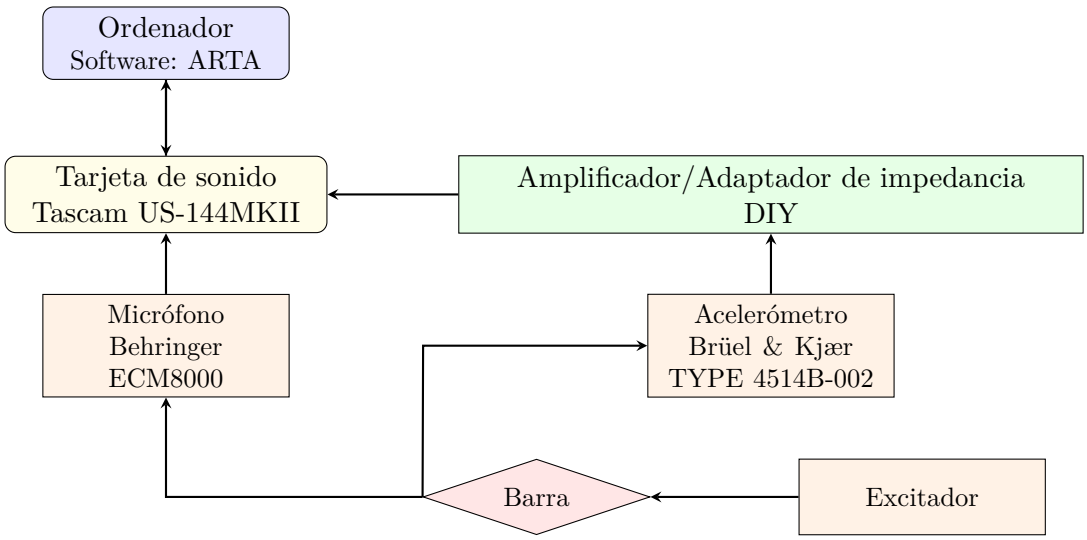


Figura 6.1: Diagrama realizado en latex con Tikz.

6.2. Gráficas

Existen múltiples formas de generar gráficas para latex. Hay disponibles herramientas como GeoGebra que dispone de la utilidad para exportar los gráficos en formato Tkiz. También funciones para Matlab que genera las gráficas que muestra habitualmente pero en código para Tkiz.

6.2.1. Línea

La forma más simple, aunque no sencilla cuando abarca muchos datos es la siguiente:

```
\begin{figure}[ht]
\centering
    \begin{tikzpicture}
        \begin{axis}
            [ymin=0,ymax=5, % Límites del eje y
            xmin=0,xmax=6, % Límites del eje x
            ylabel= eje Y,  % Nombre del eje y
            xlabel= eje X] % Nombre del eje x
            \addplot+[smooth] coordinates % Une los puntos curva suavizada
            {(0,0) (1,2) (2,3 (4,3))}; % Puntos de la gráfica
        \end{axis}
    \end{tikzpicture}
\caption{Gráfica sencilla.}
\end{figure}
```

El resultado es el siguiente:

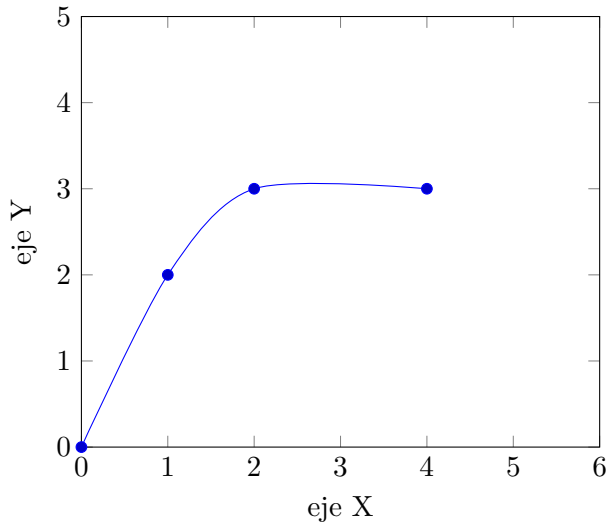


Figura 6.2: Gráfica sencilla.

Otro ejemplo, en este caso las lineas están calculadas directamente en LaTex y después cada una tiene una anotación (el código se encuentra en el archivo archivos/ejemplos/perjudicialesopticacentro.tex):

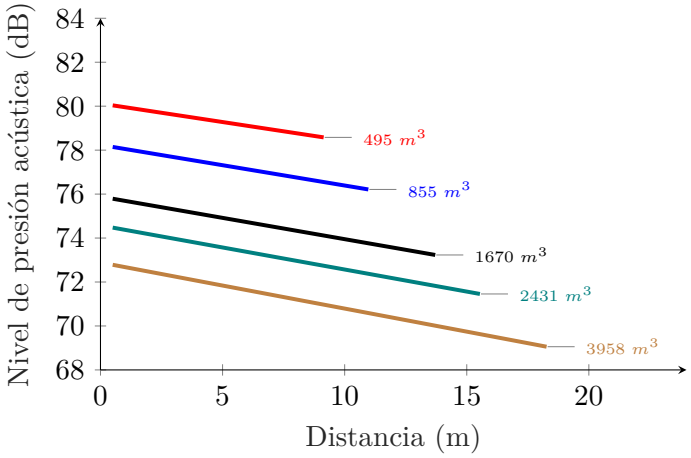


Figura 6.3: OP/S003

6.2.2. Barras

Otro ejemplo es la gráfica de barras:

```
\begin{figure}[ht]
\centering
\begin{tikzpicture}
    \begin{axis}[
        ybar=12pt,
```

```
        ymin=0,ymax=150,
        xtick=data,
        enlarge x limits={abs=2cm},
        symbolic x coords={rubio, moreno},
        bar width = 20pt,
        ylabel= número,
        xlabel= color de pelo,
            ytick align=outside,
            ytick pos=left,
            major x tick style = transparent,
            legend style={at={(0.04,0.96)},anchor=north west, font=\footnotesize, legend cell align=left,},
            ]
        \addplot[ybar,fill=blue, area legend] coordinates {
            (rubio,20)
            (moreno,100)};
        \addplot[ybar,fill=purple, area legend] coordinates {
            (rubio,110)
            (moreno,105)};
    \legend{Chicos, Chicas}
    \end{axis}
\end{tikzpicture}
\caption{Gráfica barras.}
\end{figure}
```

El resultado es el siguiente:

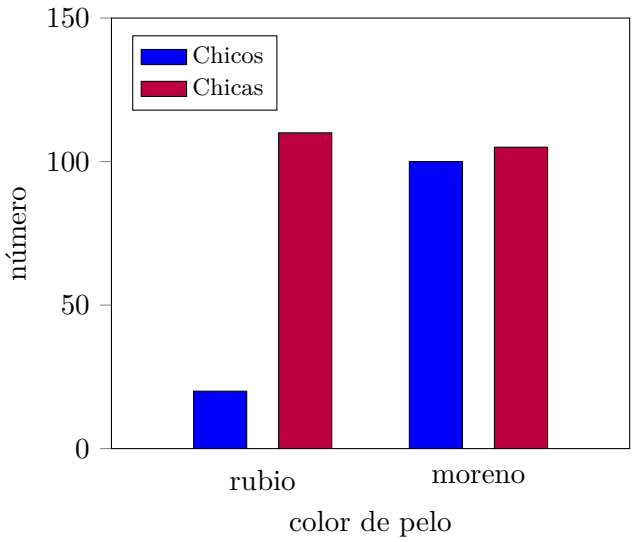


Figura 6.4: Gráfica barras.

6.2.3. Polar

Un ejemplo de gráfica polar semicircular (ver archivo archivos/ejemplos/polarnorm.tex):

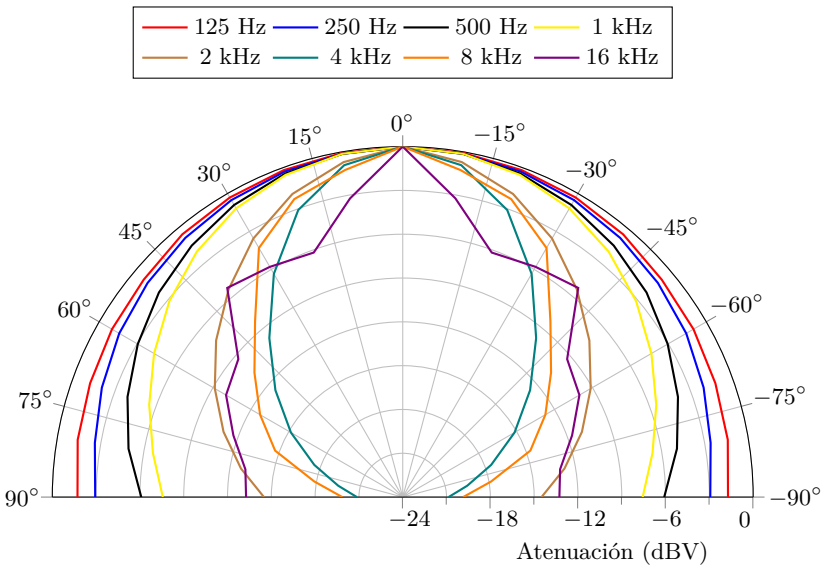


Figura 6.5: Directividad normalizada del altavoz (0 dBV en el eje).

6.3. Importados de MATLAB

Gracias a la herramienta *matlab2tikz* (`https://es.mathworks.com/matlabcentral/fileexchange/22022-matlab2tikz-matlab2tikz`) se pueden exportar las gráficas de cualquier tipo de Matlab a latex. Después de incluir los archivos de *matlab2tikz* se debe escribir una llamada después de crear la figura tal que:

Código 6.1: Ejemplo de llamada a matlab2tikz

```
fig = plot(x,y);
matlab2tikz('figurehandle',fig,'NombreArchivo.tex','height','5cm','width','13.5cm','strict',true,'↩
    ↪ showHiddenStrings',true,'showInfo',false)
```

Y para utilizar el archivo generado por la función en este documento:

```
\begin{figure}[ht]
    \centering
    {\scalefont{0.8}\input{archivos/ejemplos/ParedFina} }
    \caption{Ejemplo de gráfica obtenida con matlab2tikz.}
\end{figure}
```

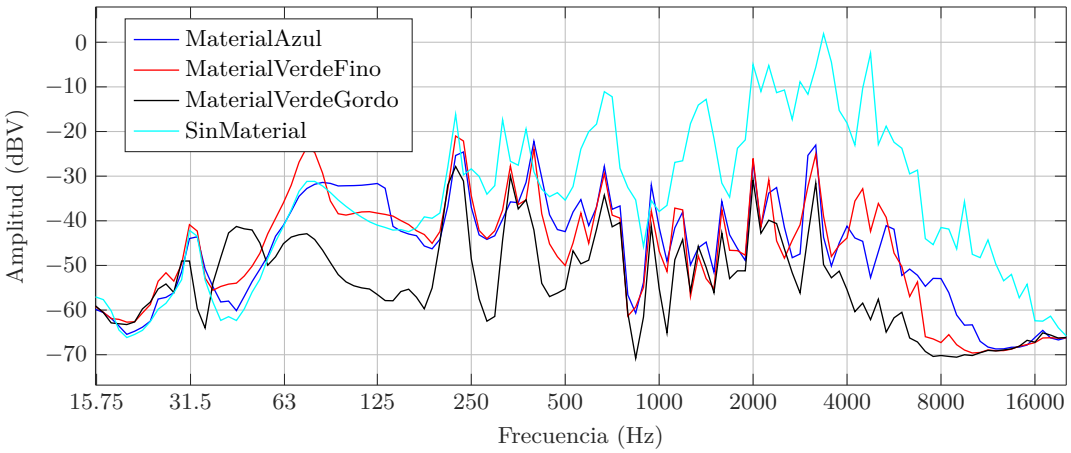


Figura 6.6: Ejemplo de gráfica obtenida con matlab2tikz.

Ejemplo de una gráfica 3D generada en Matlab y exportada por *matlab2tikz*:

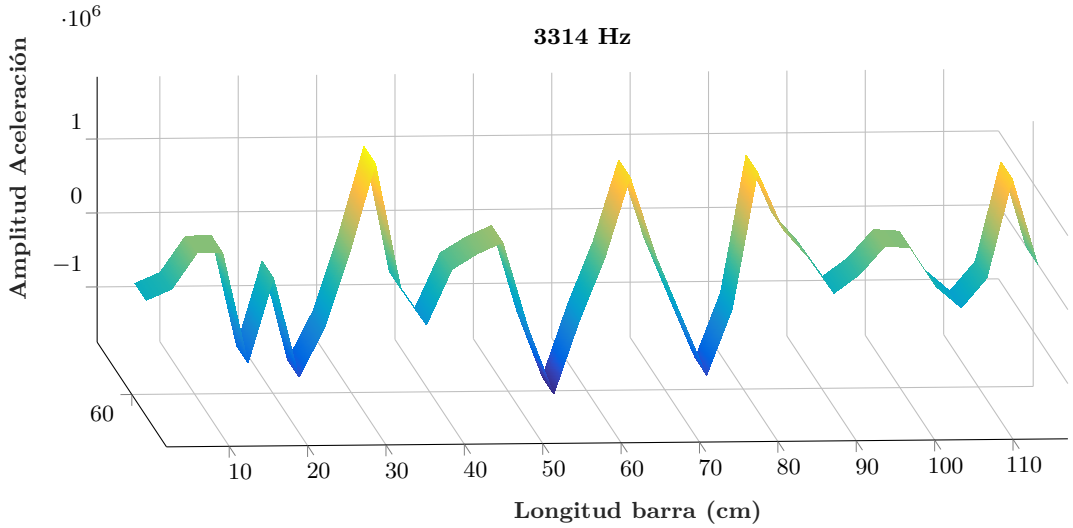


Figura 6.7: Amplitud de la aceleración en el modo número 8.

6.4. Ejemplo avanzado

El potencial del paquete *Tikz* es muy alto, se pueden realizar muchísimas cosas. En la red se facilitan muchos ejemplos para poder ver el funcionamiento y aprender. Existen hilos donde la gente publica sus mejores diseños de *Tikz* como en `https://tex.stackexchange.com/questions/158668/nice-scientific-pictures-show-off` o páginas donde facilitan muchas plantillas como `http://www.texample.net/tikz/examples/all/`.

Un ejemplo de lo que se puede llegar a conseguir es el siguiente:

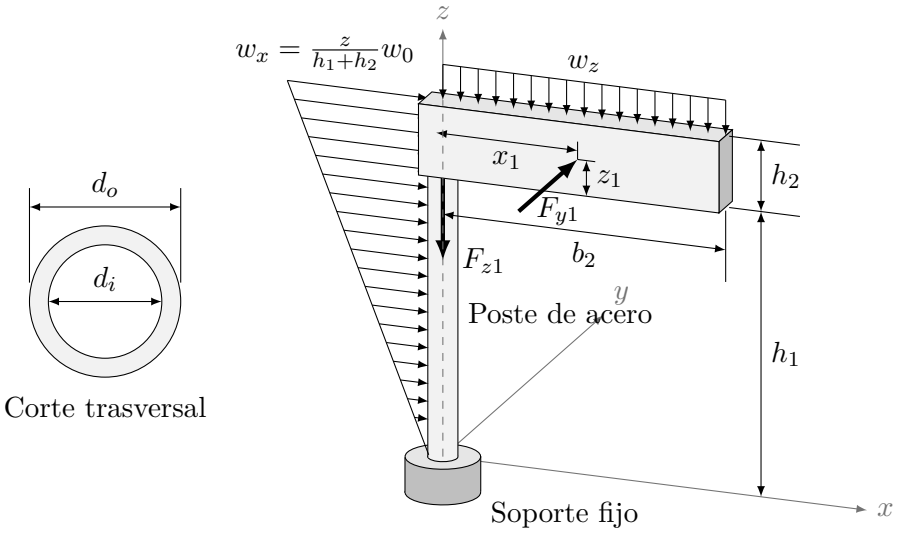


Figura 6.8: Señal realizada con Tikz, sin imágenes.

7. Evaluación

8. Conclusiones (Con ejemplos de matemáticas)

8.1. Matemáticas

En LaTeX se pueden mostrar ecuaciones de varias formas, cada una de ellas para un fin concreto.

Antes de ver algunas de estas formas hay que conocer cómo se escriben fórmulas matemáticas en LaTeX. Una fuente de información completa es la siguiente: `https://en.wikibooks.org/wiki/LaTeX/Mathematics`. También existen herramientas online que permiten realizar ecuaciones mediante interfaz gráfica como `http://www.hostmath.com/`, `https://www.mathcha.io/editor` o `https://www.latex4technics.com/`

Para mostrar una ecuación numerada se debe utilizar:

```
\begin{equation}
    \nabla\times{\mathbf H}=\left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta)-\frac{1}{r}\frac{↩
        ↪ \partial H_r}{\partial\theta}\right]{\hat{\mathbf z}}
    \label{ecuacion}
\end{equation}
```

$$\nabla \times \mathbf{H} = \left[\frac{1}{r}\frac{\partial}{\partial r}(rH_\theta) - \frac{1}{r}\frac{\partial H_r}{\partial \theta}\right]\hat{\mathbf{z}} \tag{8.1}$$

Si es necesario agrupar varias ecuaciones en un mismo índice se puede escribir del siguiente modo:

```
\begin{subequations}
    \begin{eqnarray}
        {\mathbf E}&=&E_z(r,\theta)\hat{\mathbf z}\label{ecu1} \\ % Salto de línea
        {\mathbf H}&=&H_r(r,\theta))\hat{ \mathbf r}+H_\theta(r,\theta)\hat{\bm \theta}\label{ecu2}
    \end{eqnarray}
\end{subequations}
% Se incluye '&' entre la igualdad para centrar las ecuaciones desde el '='.
```

$$\begin{aligned} \mathbf{E} &= E_z(r,\theta)\hat{\mathbf{z}} && \text{(8.2a)} \\ \mathbf{H} &= H_r(r,\theta))\hat{\mathbf{r}} + H_\theta(r,\theta)\hat{\boldsymbol{\theta}} && \text{(8.2b)} \end{aligned}$$

Otras dos formas que son las habituales en muchos lugares para incluir ecuaciones son:

```
Ejemplo de fórmula en línea con el texto $\int_{a}^{b} f(x)dx = F(b) - F(a)$, esta ecuación quedará dentro ↩
    ↪ del texto.

Esta otra, al utilizar dos '\$', se generará en una línea nueva $$\int_{a}^{b} f(x)dx = F(b) - F(a)$$
```

Ejemplo de fórmula en línea con el texto $\int_a^b f(x)dx = F(b) - F(a)$, esta ecuación quedará dentro del texto.

Esta otra, al utilizar dos '$', se generará en una línea nueva

$$\int_a^b f(x)dx = F(b) - F(a)$$

También se puede añadir información adicional a una ecuación con la función *condiciones* creada para esta plantilla:

```
\begin{equation}
    \underset{z=z_0}{\mathrm{Res}}(f(z))=\frac{1}{(m-1)!}\lim_{z \rightarrow z_0}\left[\frac{\text{d↩
        ↪ }^{m-1}}{\text{d}z^{m-1}} \left[\left(z-z_0\right)^m f(z) \right] \right]
\end{equation}

\begin{condiciones}[donde:]
% Excepto 'Descripción y valor' el resto no es necesario el símbolo $para texto matemático.
% Item & Relación & Descripción o valor
    m  & \rightarrow  & Es la multiplicidad del polo $z_0$ \\
    z_0 & \rightarrow  & Es la parte que se iguala a 0 con el polo. \\
    f(z)& \rightarrow  & Es la función contenida en la integral.
\end{condiciones}
```

$$\underset{z=z_0}{\operatorname{Res}}(f(z)) = \frac{1}{(m-1)!}\lim_{z\to z_0}\left[\frac{\mathrm{d}^{m-1}}{\mathrm{d}z^{m-1}}\left[(z-z_0)^m f(z)\right]\right] \tag{8.3}$$

donde: m → Es la multiplicidad del polo z_0
z_0 → Es la parte que se iguala a 0 con el polo.
$f(z)$ → Es la función contenida en la integral.

Si lo que deseas es una ecuación alineada a la izquierda o derecha puedes hacerlo con lo siguiente (el '&' simple es utilizado para alinear las ecuaciones desde ese punto, los iguales):

```
% Alineado a la izquierda al incluir al final el doble '&&'
\begin{flalign}
    y_{h_1}=&\begin{bmatrix}6\cos(\sqrt{6} x) \\ -\sqrt{6}\sin(\sqrt{6}x)\end{bmatrix}e^x &&\\
    y_{h_2}=&\begin{bmatrix}6\sin(\sqrt{6} x) \\ \sqrt{6}cos(\sqrt{6}x)\end{bmatrix}e^x &&
\end{flalign}

% Alineado a la derecha al incluir al inicio el doble '&&'
\begin{flalign}
&& y_{h_1}=&\begin{bmatrix}6\cos(\sqrt{6} x) \\ -\sqrt{6}\sin(\sqrt{6}x)\end{bmatrix}e^x\\
&& y_{h_2}=&\begin{bmatrix}6\sin(\sqrt{6} x) \\ \sqrt{6}cos(\sqrt{6}x)\end{bmatrix}e^x
\end{flalign}
```

$$y_{h_1} = \begin{bmatrix} 6\cos(\sqrt{6}x) \\ -\sqrt{6}\sin(\sqrt{6}x) \end{bmatrix} e^x \tag{8.4}$$

$$y_{h_2} = \begin{bmatrix} 6\sin(\sqrt{6}x) \\ \sqrt{6}cos(\sqrt{6}x) \end{bmatrix} e^x \tag{8.5}$$

$$y_{h_1} = \begin{bmatrix} 6\cos(\sqrt{6}x) \\ -\sqrt{6}\sin(\sqrt{6}x) \end{bmatrix} e^x \tag{8.6}$$

$$y_{h_2} = \begin{bmatrix} 6\sin(\sqrt{6}x) \\ \sqrt{6}cos(\sqrt{6}x) \end{bmatrix} e^x \tag{8.7}$$

Tanto con la función utilizada en (8.1,8.3), como en (8.2a,8.2b) y en las anteriores, si se les incluye un '*' después de 'equation', 'subequation' o 'flalign', se elimina la numeración de las ecuaciones pero manteniendo el resto de características.

A. Anexo I

Aquí vendría el anexo I