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

Qué pongo aquí???

Autor
Izan Gandía Ruiz

Tutor
Iván Gadea Saéz
Departamento del tutor???

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.

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.

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

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.

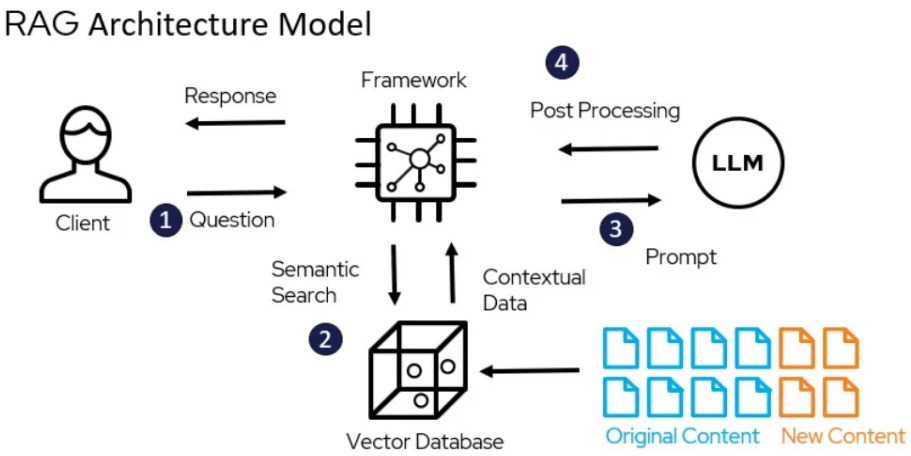


Figura 1.1: Esquema visual del funcionamiento de un sistema RAG, mostrando el flujo desde la consulta del usuario, pasando por la recuperación de información relevante, hasta la generación de la respuesta final por el LLM.

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

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.

2.1. Orquestador de tareas

Herramientas para la automatización y gestión de flujos de trabajo.

2.1.1. Prefect

Plataforma de orquestación de flujos de trabajo en Python que permite diseñar, ejecutar y monitorizar pipelines de datos y machine learning de forma fiable y escalable.

2.1.1.1. 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.

2.1.1.2. 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.

2.1.2. Kafka

Sistema de mensajería distribuido de alto rendimiento.

2.1.2.1. 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.

2.1.2.2. 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.

2.1.3. Airflow

Plataforma para crear, programar y monitorear flujos de trabajo.

2.1.3.1. 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.

2.1.3.2. Desventajas

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

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

Herramientas para monitorear cambios en el sistema de archivos.

2.2.1. Python

Lenguaje de programación versátil con varias bibliotecas para detección de cambios.

- **Watchdogs:** Biblioteca multiplataforma diseñada específicamente para detectar eventos en el sistema de archivos.
- **pyinotify:** Biblioteca que proporciona monitoreo de eventos del sistema de archivos en sistemas Linux aprovechando el subsistema inotify. Es eficiente para entornos Linux pero no es multiplataforma.
- **inotify-simple:** Un wrapper sencillo alrededor de la API inotify de Linux, que ofrece simplicidad y facilidad de uso para tareas básicas de monitoreo de archivos en sistemas Linux.
- **inotifyx:** Similar a pyinotify, esta biblioteca proporciona acceso al sistema inotify de Linux, permitiendo monitorear eventos del sistema de archivos. Está diseñada para tener una API estable pero también está limitada a plataformas Linux.
- **Polling Methods:** Para plataformas donde inotify no está disponible, o para requisitos más simples, implementar un mecanismo de sondeo puede ser una alternativa viable. Esto implica verificar periódicamente el sistema de archivos para detectar cambios.

2.2.2. Node

Entorno de ejecución para JavaScript con opciones para monitoreo de archivos.

- **chokidar:** Biblioteca eficiente para vigilar cambios en el sistema de archivos.

2.2.3. Java

Lenguaje de programación con APIs nativas para monitoreo.

- **WatchService:** API integrada en Java para monitorear cambios en directorios.

2.2.4. C++/C/C#

Familia de lenguajes con herramientas para vigilancia de archivos.

- **FileSystemWatcher:** Clase para monitorear cambios en el sistema de archivos.

2.2.5. Go

Lenguaje de programación con bibliotecas específicas.

- **fsnotify:** Biblioteca Go para monitoreo del sistema de archivos.

2.2.6. Rust

Lenguaje de programación moderno con enfoque en seguridad.

- **notify:** Biblioteca Rust para vigilar cambios en archivos.

2.3. Base de datos

Opciones para almacenamiento de datos.

2.3.1. Relacional

Bases de datos con estructura definida y relaciones entre tablas.

2.3.1.1. SQLite

Base de datos relacional ligera contenida en un único archivo.

2.3.1.1.1. 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.

2.3.1.1.2. 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.

2.3.1.2. MariaDB

Sistema de gestión de bases de datos fork de MySQL.

2.3.1.2.1. 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.

2.3.1.2.2. 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.

2.3.2. No relacional

Bases de datos con esquemas flexibles no basados en tablas relacionales.

2.3.2.1. MongoDB

Base de datos NoSQL orientada a documentos.

2.3.2.1.1. 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.

2.3.2.1.2. 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, las transacciones en múltiples documentos pueden no ser tan robustas como en bases de datos relacionales.

2.4. Docker

Plataforma de contenedores para desarrollo, envío y ejecución de aplicaciones.

2.4.1. Ventajas

- **Portabilidad:** Los contenedores Docker aseguran que el software se ejecute de manera consistente en cualquier entorno.
- **Aislamiento:** Cada componente del sistema puede ejecutarse en su propio contenedor, evitando conflictos de dependencias.
- **Facilidad de despliegue:** Simplifica la distribución y actualización de aplicaciones.

2.4.2. Desventajas

- **Consumo de recursos:** Aunque ligero, el uso de contenedores añade una capa adicional que consume recursos del sistema.
- **Complejidad adicional:** Requiere conocimientos sobre Docker y la gestión de contenedores.

2.5. Interfaz

Frameworks para el desarrollo de interfaces web.

2.5.1. Angular

Framework completo para desarrollo de aplicaciones web.

2.5.1.1. Ventajas

- **Framework completo:** Angular ofrece una solución integral con herramientas integradas para el desarrollo de aplicaciones web robustas.
- **Arquitectura estructurada:** Facilita la escalabilidad y el mantenimiento de aplicaciones complejas.

2.5.1.2. Desventajas

- **Curva de aprendizaje pronunciada:** Requiere tiempo para dominar conceptos como TypeScript y la inyección de dependencias.
- **Complejidad innecesaria para proyectos simples:** Puede ser excesivo para aplicaciones con funcionalidades limitadas.

2.5.2. React

Biblioteca de JavaScript para construcción de interfaces de usuario.

2.5.2.1. Ventajas

- **Biblioteca flexible:** React se centra en la construcción de interfaces de usuario, permitiendo la integración con diversas bibliotecas según las necesidades del proyecto.
- **Amplia comunidad y recursos:** Su popularidad asegura una gran cantidad de recursos y soporte.

2.5.2.2. Desventajas

- **Necesidad de configuraciones adicionales:** Para funcionalidades más allá de la UI, es necesario integrar bibliotecas adicionales, lo que puede aumentar la complejidad.

2.5.3. Vue

Framework progresivo para construir interfaces de usuario.

2.5.3.1. Ventajas

- **Simplicidad y facilidad de uso:** Vue es conocido por su curva de aprendizaje suave, lo que permite una adopción rápida.
- **Flexibilidad:** Adecuado tanto para proyectos pequeños como para aplicaciones más complejas.

2.5.3.2. Desventajas

- **Menor adopción en grandes empresas:** Aunque está ganando popularidad, Vue aún no es tan común en entornos corporativos grandes.

2.5.4. Astro

Framework moderno para sitios web con enfoque en rendimiento.

2.5.4.1. Ventajas

- **Optimización para contenido estático:** Astro está diseñado para generar sitios web estáticos rápidos, lo que puede ser beneficioso para aplicaciones con contenido predominantemente estático.
- **Integración con otros frameworks:** Permite utilizar componentes de React, Vue y otros dentro de sus proyectos.

2.5.4.2. Desventajas

- **Menor madurez:** Al ser relativamente nuevo, Astro puede carecer de la amplitud de recursos y comunidad que tienen otros frameworks más establecidos.

3. Objetivos

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).

Además se plantean objetivos secundarios que complementan el objetivo principal, como lo son el estudio de diferentes modelos multimodales para obtener los mejores resultados en un tiempo razonable, la búsqueda de una base de datos adecuada para este tipo de problemas y la estructura óptima del sistema para que sea escalable y eficiente. También se busca la creación de una interfaz gráfica intuitiva que permita a los usuarios interactuar fácilmente con el sistema, facilitando la búsqueda y la visualización de los resultados obtenidos.

4. Metodología

4.1. Organización

La metodología que se va a seguir es Scrum Adaptado (con un toque de Kanban). 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. La adaptación en este caso consiste en que yo asumiré el rol de Product Owner, Desarrollador y Scrum Master, mientras mi tutor actúa como cliente (requisitos). 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

Explicar que cosas he elegido y por qué + my hardware

5. Análisis, Especificación y diseño

5.1. Requisitos del sistema

En esta sección se detallan los requisitos del sistema, divididos en requisitos funcionales, no funcionales y de configuración. Los requisitos se han estructurado en formato tabular para facilitar su comprensión y seguimiento durante el desarrollo del proyecto.

5.1.1. Requisitos funcionales

Los requisitos funcionales describen el comportamiento que debe tener el sistema, las funcionalidades que debe ofrecer y las operaciones que debe realizar.

ID	Nombre	Descripción
RF-01	Detección de ficheros	El sistema debe detectar nuevos ficheros en el directorio observado.
RF-02	Diferenciación de tipos	El sistema debe diferenciar el tipo de archivo a analizar (texto, imagen, vídeo, audio, otros).
RF-03	Ejecución de modelos	El sistema debe ejecutar el modelo correspondiente que extraerá la información del fichero a la base de datos.
RF-04	Almacenamiento	El sistema debe almacenar todos los datos posibles sobre el fichero analizado en una base de datos.
RF-05	Interfaz web	El sistema debe tener una interfaz web super-simple donde el usuario podrá escribir su consulta en lenguaje natural y darle a un botón para realizar la búsqueda.
RF-06	Resultados de búsqueda	El sistema responderá con un conjunto de resultados potencialmente interesantes a partir de la consulta de búsqueda, ordenados de más a menos ”interesante”.
RF-07	Entrada por línea de comandos	El sistema debe tener una entrada por línea de comandos (ej: `LLMSearch --query "mapa del mundo en el que hay marcados los mejores parques naturales"`).
RF-08	Presentación de resultados	El resultado será la ruta del fichero junto a una pequeña descripción del mismo (enlaces clicables al fichero y a la carpeta que lo contiene).
RF-09	Inspección de archivos comprimidos	Los ficheros comprimidos deberían poder inspeccionarse por dentro.
RF-10	Tipos de ficheros a procesar	El sistema debe procesar los siguientes tipos de ficheros: - Documentos de texto - Imágenes - Vídeos - Ficheros de sonido - Otros (bases de datos, ejecutables, etc.)

Tabla 5.1: Requisitos funcionales del sistema

5.1.2. Requisitos no funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

ID	Nombre	Descripción
RNF-01	Configuración web	La web debe tener una pequeña parte de configuración discreta pero accesible en todo momento.
RNF-02	Arquitectura modular	La arquitectura se debe dividir en un "buscador" y un "explorador" y deben ser completamente separadas para poder ser reutilizadas.
RNF-03	Ejecución sin GPU	El sistema debe poder ejecutarse en un ordenador sin GPU (opcional).
RNF-04	Parámetro de consulta	La entrada por línea de comandos aceptará un parámetro `--query` junto al término de búsqueda.
RNF-05	Resultados en CLI	La entrada por línea de comandos devolverá los resultados de la misma manera que el buscador web con la diferencia de que solo devolverá información adicional si se le añade el parámetro `--verbose`.
RNF-06	Estado del sistema	La entrada por línea de comandos tendrá un parámetro `--status` que devolverá el estado del sistema: número de archivos procesados sobre el número total de archivos en observación, cantidad de ficheros de cada tipo, errores encontrados...
RNF-07	Configuración por CLI	Se añadirán los parámetros necesarios para poder configurar el sistema desde línea de comandos.

Tabla 5.2: Requisitos no funcionales del sistema

5.1.3. Requisitos de configuración

Los requisitos de configuración especifican las opciones que el usuario debe poder ajustar en el sistema.

ID	Nombre	Descripción
RC-01	Directorio de observación	Directorio donde se están observando nuevos ficheros.
RC-02	Regulación de carga	Regular la carga (limitar la CPU al X%).
RC-03	Tipo de modelo LLM	Tipo de modelo LLM a utilizar (Local *(LLM Studio)* ó en la nube).
RC-04	Búsqueda por imagen	Posibilidad de poner una foto de una persona y que la busque en los ficheros.

Tabla 5.3: Requisitos de configuración del sistema

5.2. Diagrama de arquitectura del sistema

A continuación se presenta un diagrama de la arquitectura del sistema que muestra la división entre el "buscador" y el "explorador" según el requisito no funcional RNF-02.

5.3. Casos de uso

Los casos de uso describen las interacciones típicas entre los usuarios y el sistema, mostrando cómo se utilizarían las funcionalidades principales.

6. Análisis, Especificación y diseño

6.1. Requisitos del sistema

En esta sección se detallan los requisitos del sistema, divididos en requisitos funcionales, no funcionales y de configuración. Los requisitos se han estructurado en formato tabular para facilitar su comprensión y seguimiento durante el desarrollo del proyecto.

6.1.1. Requisitos funcionales

Los requisitos funcionales describen el comportamiento que debe tener el sistema, las funcionalidades que debe ofrecer y las operaciones que debe realizar.

ID	Nombre	Descripción
RF-01	Detección de ficheros	El sistema debe detectar nuevos ficheros en el directorio observado.
RF-02	Diferenciación de tipos	El sistema debe diferenciar el tipo de archivo a analizar (texto, imagen, vídeo, audio, otros).
RF-03	Ejecución de modelos	El sistema debe ejecutar el modelo correspondiente que extraerá la información del fichero a la base de datos.
RF-04	Almacenamiento	El sistema debe almacenar todos los datos posibles sobre el fichero analizado en una base de datos.
RF-05	Interfaz web	El sistema debe tener una interfaz web super-simple donde el usuario podrá escribir su consulta en lenguaje natural y darle a un botón para realizar la búsqueda.
RF-06	Resultados de búsqueda	El sistema responderá con un conjunto de resultados potencialmente interesantes a partir de la consulta de búsqueda, ordenados de más a menos ”interesante”.
RF-07	Entrada por línea de comandos	El sistema debe tener una entrada por línea de comandos (ej: `LLMSearch --query "mapa del mundo en el que hay marcados los mejores parques naturales"`).
RF-08	Presentación de resultados	El resultado será la ruta del fichero junto a una pequeña descripción del mismo (enlaces clicables al fichero y a la carpeta que lo contiene).
RF-09	Inspección de archivos comprimidos	Los ficheros comprimidos deberían poder inspeccionarse por dentro.
RF-10	Tipos de ficheros a procesar	El sistema debe procesar los siguientes tipos de ficheros: - Documentos de texto - Imágenes - Vídeos - Ficheros de sonido - Otros (bases de datos, ejecutables, etc.)

Tabla 6.1: Requisitos funcionales del sistema

6.1.2. Requisitos no funcionales

Los requisitos no funcionales especifican criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos.

ID	Nombre	Descripción
RNF-01	Configuración web	La web debe tener una pequeña parte de configuración discreta pero accesible en todo momento.
RNF-02	Arquitectura modular	La arquitectura se debe dividir en un "buscador" y un "explorador" y deben ser completamente separadas para poder ser reutilizadas.
RNF-03	Ejecución sin GPU	El sistema debe poder ejecutarse en un ordenador sin GPU (opcional).
RNF-04	Parámetro de consulta	La entrada por línea de comandos aceptará un parámetro `--query` junto al término de búsqueda.
RNF-05	Resultados en CLI	La entrada por línea de comandos devolverá los resultados de la misma manera que el buscador web con la diferencia de que solo devolverá información adicional si se le añade el parámetro `--verbose`.
RNF-06	Estado del sistema	La entrada por línea de comandos tendrá un parámetro `--status` que devolverá el estado del sistema: número de archivos procesados sobre el número total de archivos en observación, cantidad de ficheros de cada tipo, errores encontrados...
RNF-07	Configuración por CLI	Se añadirán los parámetros necesarios para poder configurar el sistema desde línea de comandos.

Tabla 6.2: Requisitos no funcionales del sistema

6.1.3. Requisitos de configuración

Los requisitos de configuración especifican las opciones que el usuario debe poder ajustar en el sistema.

ID	Nombre	Descripción
RC-01	Directorio de observación	Directorio donde se están observando nuevos ficheros.
RC-02	Regulación de carga	Regular la carga (limitar la CPU al X%).
RC-03	Tipo de modelo LLM	Tipo de modelo LLM a utilizar (Local *(LLM Studio)* ó en la nube).
RC-04	Búsqueda por imagen	Posibilidad de poner una foto de una persona y que la busque en los ficheros.

Tabla 6.3: Requisitos de configuración del sistema

6.2. Diagrama de arquitectura del sistema

A continuación se presenta un diagrama de la arquitectura del sistema que muestra la división entre el "buscador" y el "explorador" según el requisito no funcional RNF-02.

6.3. Casos de uso

Los casos de uso describen las interacciones típicas entre los usuarios y el sistema, mostrando cómo se utilizarían las funcionalidades principales.

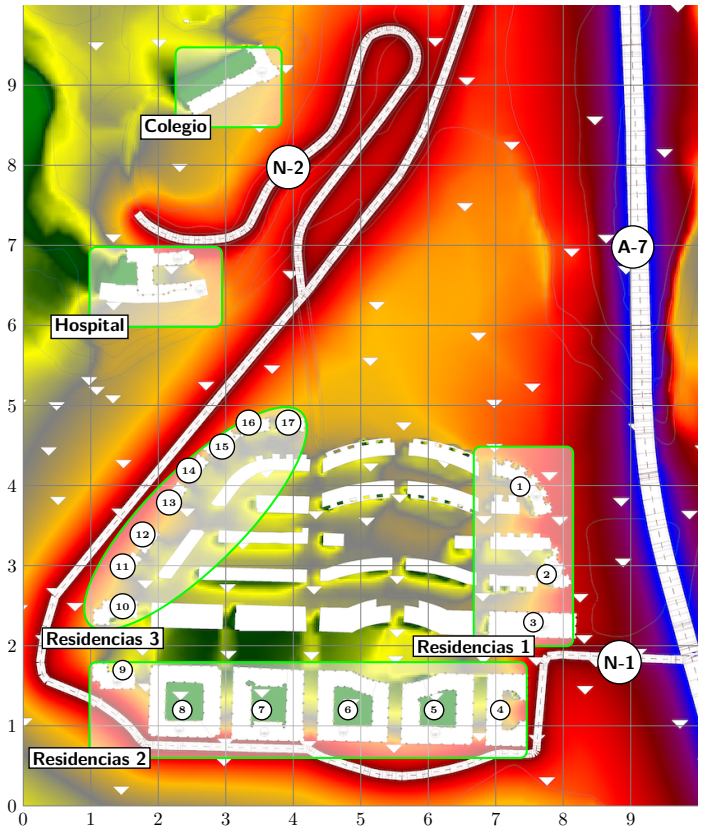
7. Resultados (Con ejemplos de gráficos)

7.1. 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/`.

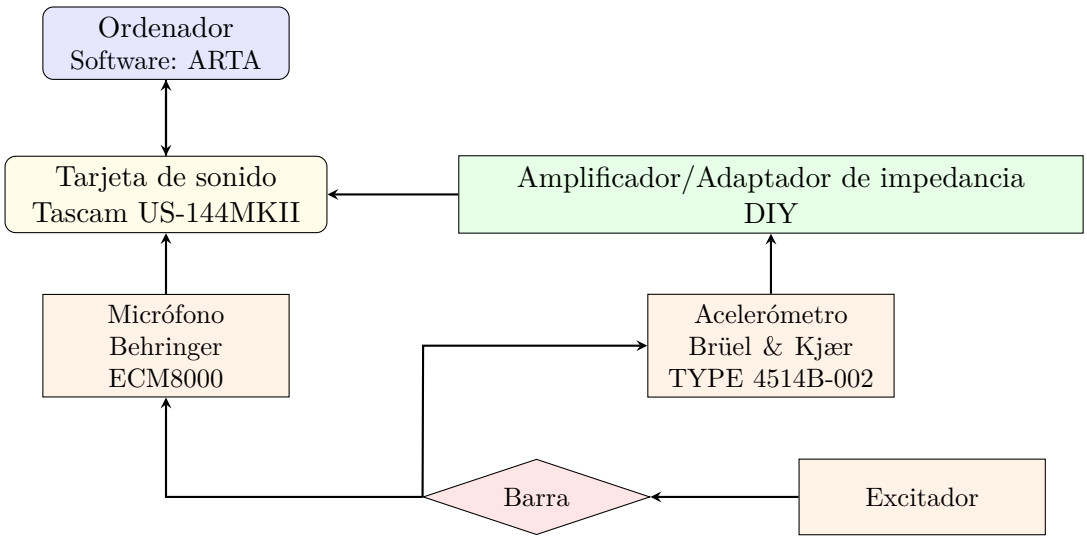


Figura 7.1: Diagrama realizado en latex con Tikz.

7.2. 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.

7.2.1. 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:

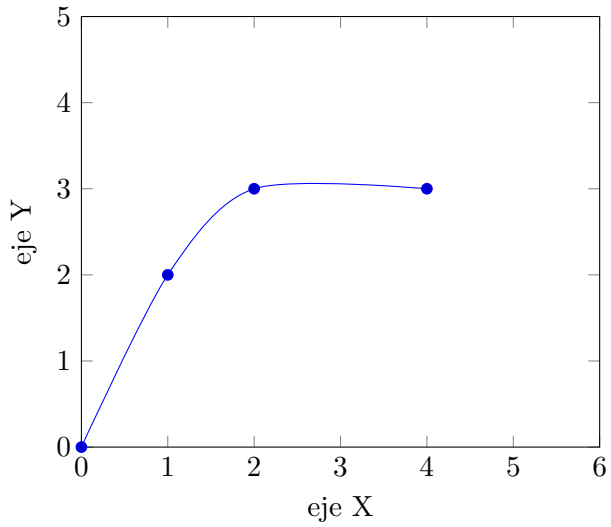


Figura 7.2: Gráfica sencilla.

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):

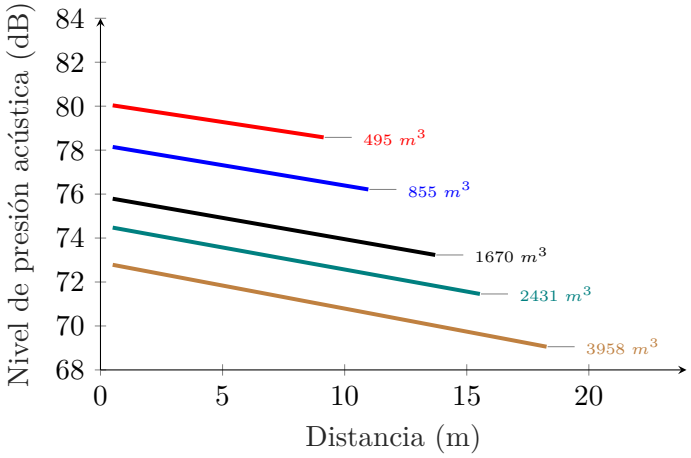


Figura 7.3: OP/S003

7.2.2. 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:

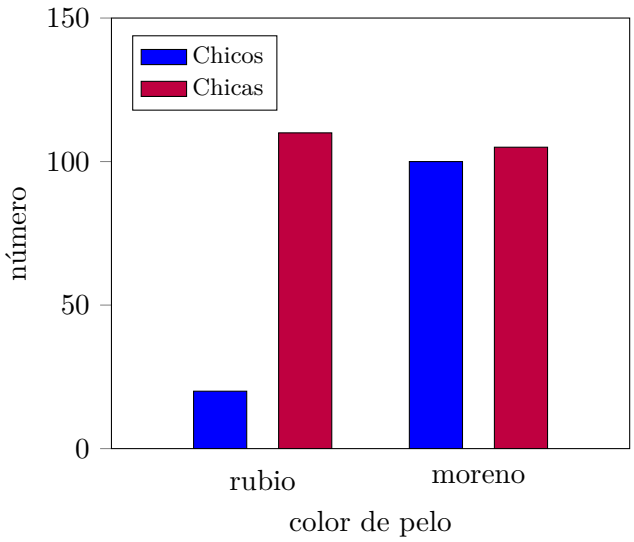


Figura 7.4: Gráfica barras.

7.2.3. Polar

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

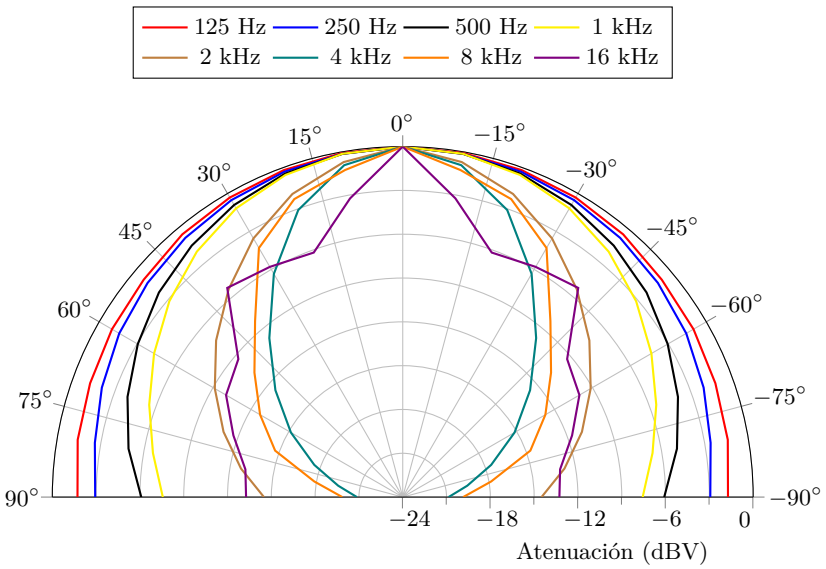


Figura 7.5: Directividad normalizada del altavoz (0 dBV en el eje).

7.3. 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 7.1: 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}
```

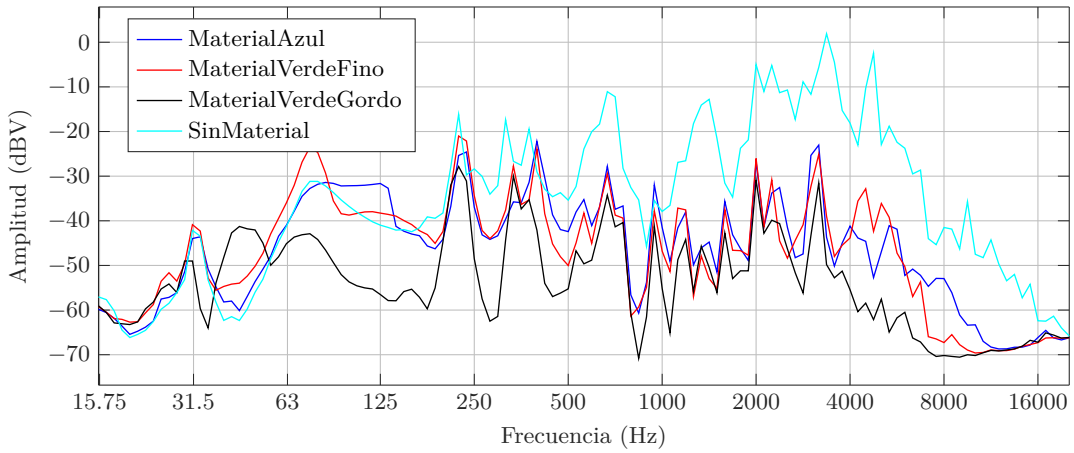


Figura 7.6: Ejemplo de gráfica obtenida con matlab2tikz.

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

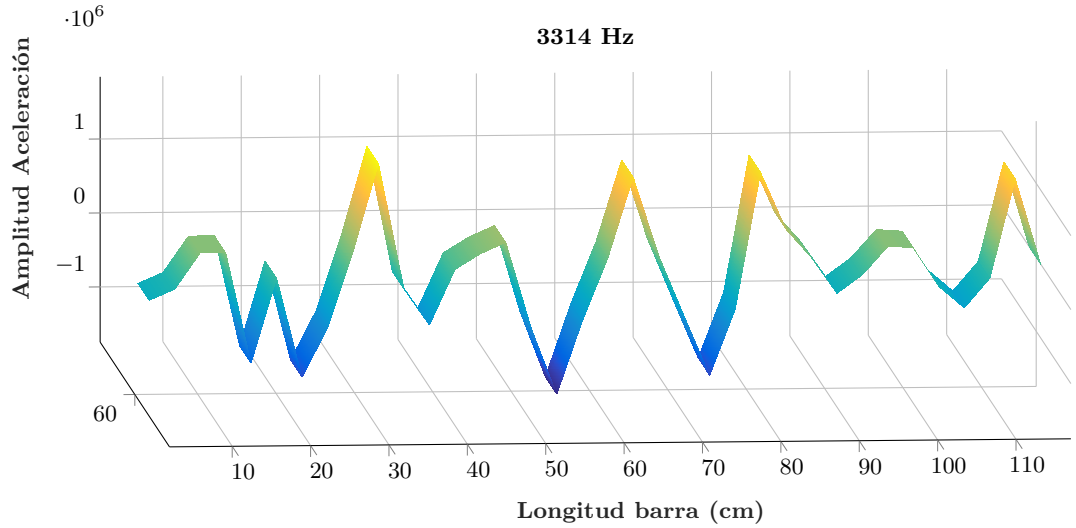


Figura 7.7: Amplitud de la aceleración en el modo número 8.

7.4. 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:

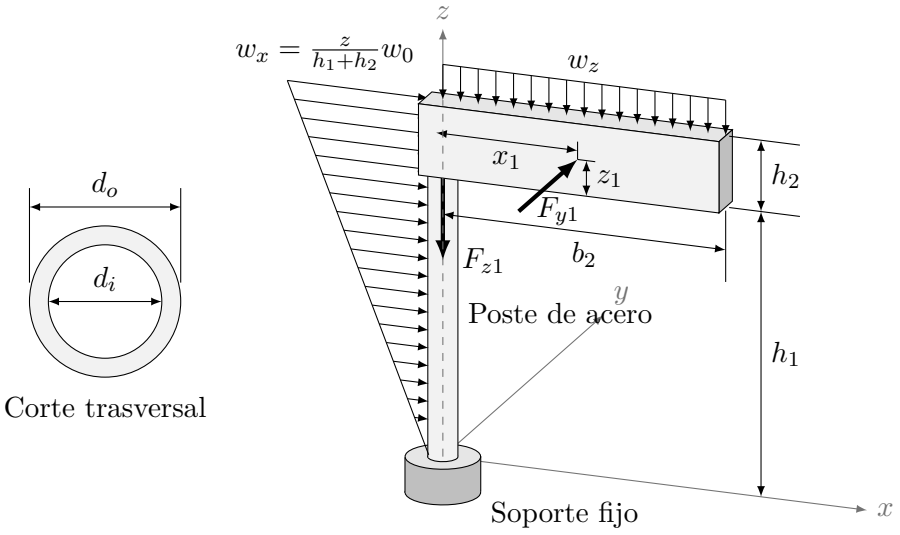


Figura 7.8: Señal realizada con Tikz, sin imágenes.

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

B. Páginas horizontales

Aquí se muestra cómo incluir páginas en horizontal.
Esta página está en vertical

Esta página está en horizontal

Esta página también está en horizontal

Esta página está de nuevo en vertical

C. Importar PDF

A continuación se muestra una página importada de un PDF externo. Observar los comentarios en el código de este anexo para más información. También puedes leer el manual con todas las opciones en `http://osl.ugr.es/CTAN/macros/latex/contrib/pdfpages/pdfpages.pdf`.

Alicante 15 DE MARZO DE 2007

Expediente número

Referencia del peticionario **AYUNTAMIENTO DE ALICANTE**

Departamento de Medio Ambiente
C/San Nicolás, nº 2, 4º
03001 ALICANTE
Contacto: Juan Luís Beresaluze

DOCUMENTO DE SÍNTESIS

ELABORACIÓN DEL MAPA ACÚSTICO MUNICIPAL DE LA CIUDAD DE ALICANTE

Fecha de realización del estudio: MAYO 2005 – MARZO 2007